software AG

Natural

Tools and Utilities

Version 9.1.1

October 2018

ADABAS & NATURAL

Document ID: NATUX-NNATUTILITIES-911-20200402

Table of Contents

Preface

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

Utility Activation	Describes how Natural invokes a tool or utility.
FTOUCH	Makes a downloaded object executable by Natural.
INPL	Loads or scans Natural objects and shared resources supplied by Software AG.
Natural Profiler	Monitors the internal process flow of a Natural application and analyzes the performance and code coverage of the application.
Natural Termcap (NATTERMCAP) Utility	Adapts your terminal to terminal-dependent parts of Natural.
Object Handler	Processes Natural and non-Natural objects for distribution in Natural environments. This is done by unloading the objects in the source environment into work files and loading them from work files into the target environment.
SYSERR	Creates application-specific messages. In addition, it can be used to modify the texts of the existing Natural system messages (not recommended).
SYSEXT	Locates Natural Application Programming Interfaces (APIs).
SYSEXV	Provides examples of the new features of the current Natural versions.
SYSMAIN	Performs object operations in Natural such as copy, move, delete and import.
SYSNCP	Defines command-driven navigation systems for Natural applications.
SYSPCI	Configures and initializes a product after a first-time installation.
SYSRPC	Establishes and maintains Natural RPC (Remote Procedure Call) environments.

1 About this Documentation

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at **http://documentation.softwareag.com**. The site requires credentials for Software AG's Product Support site Empower. If you do not have Empower credentials, you must use the TECHcommunity website.

Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at **https://empower.softwareag.com/**.

You can find product information on the Software AG Empower Product Support website at **https://empower.softwareag.com**.

To submit feature/enhancement requests, get information about product availability, and download products, go to **Products**.

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the **Knowledge Center**.

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at **https://empower.softwareag.com/public_directory.asp** and give us a call.

Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at **http://techcommunity.softwareag.com**. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

I Utility Activation

2 Utility Activation

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

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

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

Exception:

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

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

II FTOUCH Utility

3 FTOUCH Utility

The FTOUCH utility is used to make a downloaded object executable by Natural. This is done by importing the object into the Natural system file FNAT or FUSER and updating the *FILEDIR.SAG* file.

Note: Using the FTOUCH utility is limited to 60000 objects because this is the maximum number of entries in the file *FILEDIR.SAG*.

Related Topics:

- *The File FILEDIR.SAG* - *Operations* documentation
- *Using NFS to Store Natural Libraries* - *Operations* documentation
- *Transferring Natural Generated Programs* - *Programming Guide*

Using the Utility FTOUCH

This section provides instructions for executing the FTOUCH utility.

Note: Terms enclosed in brackets (`[ ]`) are optional; bold letters are actual values that must be entered as shown.

To execute the FTOUCH utility

1. Go to an operating system command prompt.
2. Ensure that the transferred file is in the desired FNAT or FUSER directory (as specified in your global configuration file) and has the correct extension.
3. Enter the command `ftouch` using the following syntax:

```
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
```

Or:

For migration, use the following syntax:

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

Or:

For endian conversion of the *FILEDIR.SAG* file, use the following syntax:

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

Or:

For encoding of single or multiple objects contained in the *FILEDIR.SAG* file, use the following syntax:

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

Or:

For setting the line number suppression state of a library in *FILEDIR.SAG*, use the following syntax:

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

Syntax of ftouch

The following options are provided with the `ftouch` command:

Option	Explanation
`fnat=`*`dbid`*`,`*`fnr`*	Specifies the database ID and file number of the FNAT system file to be used; default is the value specified in the NATPARM parameter file. See also *Example 2*.
`fuser=`*`dbid`*`,`*`fnr`*	Specifies the database ID and file number of the FUSER system file to be used; default is the value specified in the NATPARM parameter file. See also *Example 2*.

Option	Explanation
bp=*bp-name*	Specifies the buffer pool to be used. You can omit the *bp-name* if you want to use the Natural default buffer pool NATBP; otherwise, you have to specify the appropriate *bp-name*. **Note:** 1. If the Natural default buffer pool is not active or if the specified buffer pool does not exist, an appropriate error message is displayed. 2. Do not delete the default buffer pool NATBP, as it is possible that Natural may no longer function properly.
parm=*parm-name*	Specifies the name of the parameter file to be used if you want to use a parameter file other than the default NATPARM parameter file.
lib=*library-name*	Specifies the library to be used. You can omit the *library-name* if you are already in the appropriate subdirectory; otherwise you have to specify the appropriate *library-name*.
userep=*rep-use*	Specifies whether to use the repository or not. *rep-use* must be one of the following: ON — The repository is used. OFF — The repository is not used.
-v	Displays statistics on disk I/O operations during processing.
-q	Indicates that quiet mode is to be used: only error messages but no status messages are displayed.
-ignoreext	Specifies that files with unkown extensions contained in a library are ignored. The -ignoreext option can be combined with one of the following options: -a -d
mode	Specifies the programming mode; sm specifies that a program is in structured mode; the default is reporting mode. See also *Example 1*.
kind	Specifies the subdirectories SRC and/or GP for input; it can be one of the following: -s for source objects (default), -g for cataloged objects/generated programs, -b for both source objects and cataloged objects/generated programs. See also *Example 2*.

Option	Explanation
`files`	Specifies the files to be processed; you can specify *`filename.ext`* for individual files or: `-a` to add new files; all files in the directory which are not currently found in *FILEDIR.SAG* are added (already existing files are not touched). `-d` to build a new *FILEDIR.SAG* directory. **Caution:** Be careful when using this option, since the old *FILEDIR.SAG* is deleted and rebuilt from scratch. See also ***Example 4***.
`-f`	Forces an update of the specified object's timestamp in *FILEDIR.SAG*. This option can only be specified if an individual file has been specified with the *`files`* option (see above).
`convert`	Indicates that an old *FILEDIR.SAG* file is to be migrated. The *FILEDIR.SAG* file from a Natural version earlier than Version 6.2 is converted into a new portable *FILEDIR.SAG* file. A copy of the original (old) *FILEDIR.SAG* file is saved as *FILEDIR.BCK* file in the directory of the specified library. If a *FILEDIR.BCK* file already exists in the specified library, the old *FILEDIR.SAG* will *not* be converted. For further information, see *Portable Natural System Files* in the *Operations* documentation. See also ***Example 3*** and ***Example 5***.
`sync`	Indicates that the specified library and system files are to be synchronized between Natural and the repository (Windows only); this function must be executed each time *FILEDIR.SAG* is modified by FTOUCH. **Caution:** When specifying `sync`, ensure that either `userep=ON` is set or the Natural profile parameter `USEREP` is set to `ON`.

`encoding=`*`encoding-name`*	Specifies the code page to be used for the files contained in *FILEDIR.SAG*.
	
	The `encoding` option generates or changes the internal code page information maintained in *FILEDIR.SAG* for each object affected by the `ftouch` command. This option does *not* convert the contents of a source object or a cataloged object/generated program.
	
	The `encoding` option can be combined with the following options:
	
	`-a`
	`-d`
	`convert`
	`objname`

Option	Explanation
	`encoding-name` can be any code page name valid with the `CP` session parameter specified in the *NATPARM* parameter file. See also *CP - Default Code Page Name* in the *Parameter Reference*. See also *Example 4*, *Example 5*, *Example 7* and *Example 8*.
`endian=endian-mode`	Specifies the endian format to be used for the *FILEDIR.SAG* directory. The `endian` option applies to the entire *FILEDIR.SAG* directory. The option does not apply when adding files to *FILEDIR.SAG* or when generating a new *FILEDIR.SAG*. `endian-mode` can be one of the following formats: `BIG` Converts to big endian. `LITTLE` Converts to little endian. `DEFAULT` Converts to the endian format used on your current platform. See also *Example 6*.
`objname=object-name`	Selects the object(s) for which to maintain internal format information in *FILEDIR.SAG*. The `objname` option only applies if the `encoding` option is specified. `object-name` selects all objects with names equal to the specified value. You can use asterisk (*) notation for a name range. See also *Example 7* and *Example 8*.
`suprln=library-state`	Specifies whether the line number suppression state is set for the specified library. `library-state` must be one of the following: `ON` Source line numbers are not written to the files contained in *FILEDIR.SAG*, when saving the sources of the objects contained in this library. `OFF` Source line numbers are written to the files contained in *FILEDIR.SAG*.

Examples of ftouch

The following section provides examples of the `ftouch` command.

Example 1:

Change to the following directory:	*`fuser-directory`*`/TESTLIB/SRC`
Enter the following command:	`ftouch sm TESTFILE.NSP`

As a result, the program `TESTFILE` in library `TESTLIB` is available in structured mode to Natural.

Example 2:

Change to the following directory:	*`fuser-directory`*`/MYLIB`
Enter the following command:	`ftouch fnat=21,21 fuser=22,22 -b`

As a result, all files in the directories `MYLIB/SRC` and `MYLIB/GP` are available in reporting mode (default) to Natural.

Example 3:

Change to the following directory:	*`fuser-directory`*
Enter the following command:	`ftouch lib=MYLIB convert`

As a result, a new portable *FILEDIR.SAG* file is saved for the `MYLIB` library and the old *FILEDIR.SAG* is saved as *FILEDIR.BCK* file in this library.

Example 4:

Change to the following directory:	*`fuser-directory`*
Enter the following command:	`ftouch lib=MYLIB encoding=UTF-8 -a -s`

As a result, the internal format information is generated as `UTF-8` for all objects which are added to the *FILEDIR.SAG* directory from the `MYLIB/SRC` subdirectory.

Example 5:

Change to the following directory:	`fuser-directory`
Enter the following command:	`ftouch lib=OLDLIB encoding=windows-1251 convert`

As a result, a new portable *FILEDIR.SAG* file is saved for the `OLDLIB` library and the internal format information changes to `windows-1251` for all objects contained in the *FILEDIR.SAG* file.

Example 6:

Change to the following directory:	`fuser-directory`
Enter the following command:	`ftouch lib=MYLIB endian=BIG`

As a result, the *FILEDIR.SAG* file of the `MYLIB` library is converted to big endian. The internal format information changes to `BIG` for all objects contained in the `MYLIB` library.

Example 7:

Change to the following directory:	`fuser-directory`
Enter the following command:	`ftouch lib=MYLIB objname=MYPROG1 encoding=UTF-8`

As a result, the internal format information of object `MYPROG1` changes to `UTF-8` if `MYPROG1` is contained in library `MYLIB` in the *FILEDIR.SAG* file.

Example 8:

Change to the following directory:	`fuser-directory`
Enter the following command:	`ftouch lib=MYLIB objname=MY* encoding=UTF-8`

As a result, the internal information of all objects with names that start with `MY` changes to `UTF-8` if they are contained in library `MYLIB` in the *FILEDIR.SAG* file.

Example 9:

Change to the following directory:	`fuser-directory`
Enter the following command:	`ftouch lib=MYLIB suprln=ON`

As a result, the line number suppression state is set to `ON` for library `MYLIB` in the *FILEDIR.SAG* file.

III INPL Utility

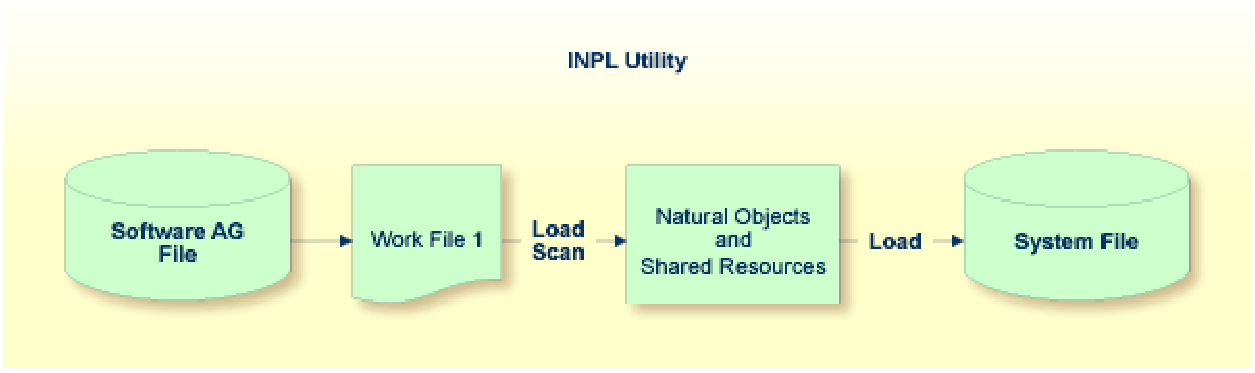
4 INPL Utility

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

Introducing the INPL Utility

The INPL utility processes Natural objects and shared resources provided by Software AG.

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



The Natural objects and shared resources are delivered as installation or update files which are assigned to Work File 1. The INPL utility loads the Natural objects and shared resources from Work File 1 into Natural system files.

The Natural objects and shared resources include cataloged objects and source objects that are contained in libraries in the Natural system files FNAT and FUSER.

In addition to loading Natural objects and shared resources, the INPL utility provides a scan function to check the contents of the file assigned to Work File 1 and a **Natural Security Recover** function which forces initialization of the Natural Security environment.

When loading cataloged objects into Natural system files, the INPL utility deletes any buffer pool entries of cataloged objects with identical names if contained in the same buffer pool used by the INPL utility.

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

This section covers the following topics:

- Restrictions
- Special Case
- Invoking INPL
- Options Available

- INPL Report

Restrictions

You can process only files which are marked as "SAG system INPL file".

Special Case

When an INPL is to be performed in a Natural Security environment, the INPL command can be specified using the dynamic Natural profile parameter `STACK`.

Invoking INPL

To invoke the INPL utility

1 Enter the following Natural system command:

```
INPL
```

An INPL menu similar to the example below is displayed:

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

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

For detailed information on these functions, refer to the corresponding sections.Modify or complete the remaining input fields as described in ***Options Available***.

3 Choose ENTER to confirm your entries.

Options Available

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

Field	Description
Replace	Specifies whether the Natural objects and shared resources to be processed are to replace any that already exist on the system files. Possible settings are: `Y` All existing Natural objects and shared resources are replaced. This is the default setting. `N` Existing Natural objects and shared resources are *not* replaced. `O` Resets the owner information of specified objects. Only applies to the function **Natural Security Recover**. See also **Check Date** to replace only Natural objects and shared resources that are older than the Natural objects and shared resources to be processed. If you use the function **Natural Security Recover**, you can enter Option `O` in this field to reset the owner information of specified objects.
DDM Name	The name of a DDM or a range of names. If you enter a value that ends with an asterisk (*), each DDM with a name that starts with the specified value is processed. If only an asterisk (*) is entered or if this field is empty, all DDMs are processed.

Field	Description
Library	The name of a library or a range of names. If you enter a value that ends with an asterisk (*), each library with a name that starts with the specified value is processed. The library name is mandatory if **Object Name** is specified.
Object Name	The name of a Natural object (except DDMs) or a range of names. If the value ends with an asterisk (*), each object with a name that starts with the specified value is processed. If this field is empty, all objects contained in the library specified in the **Library** field are processed.
Check Date	Specifies whether existing Natural objects and shared resources are to be replaced depending on their time stamp. This parameter has no effect if **Replace** is set to `N`. Possible settings are: `Y` Only objects which are older than the Natural objects or shared resources of the same name are replaced. An object is older if it was saved or cataloged before the object to be loaded. `N` All objects are replaced. This is the default setting.
Load Except	Specifies whether to exclude Natural objects and shared resources from processing. This parameter does not apply to error messages. Possible settings are: `Y` All Natural objects and shared resources are processed except for the objects specified in the fields **DDM Name**, **Library** and/or **Object Name**. `N` No exceptions; all Natural objects and shared resources are processed. This is the default setting. **Examples of load exceptions:** All libraries except the library ABC are loaded: **Code** = `L` **Library** = `ABC` All DDMs with a prefix other than XY are loaded: **Code** = `D` **DDM Name** = `XY*` All objects contained in libraries with a prefix other than AB and all DDMs with a prefix other than CD are loaded: **Code** = `B` **Library** = `AB*`

Field	Description
	DDM Name = `CD*`
Date	Restricts processing to Natural objects and shared resources which were saved or cataloged on or after the date entered in this field. The date must be entered in the format *YYYY-MM-DD* (*YYYY* = year, *MM* = month, *DD* = day).
Number	Limits processing of Natural objects and shared resources to a specified number. All objects are counted which are loaded or scanned according to the selection criteria specified in the INPL menu. If the number of Natural objects processed has reached the value entered in the **Number** field, processing is terminated with a corresponding message.
File Type (batch or direct commands only)	INPL automatically recognizes the type of the load file such as binary or portable. However, due to compatibility reasons, the `File type` parameter must still be specified when executing INPL in batch or direct command mode, but it will not be evaluated.
Load File	The name of the file to be loaded.
Report File	The name of the file into which the **INPL report** (see below) is to be written.

INPL Report

When the selected INPL function is complete, a corresponding INPL report is written to the file you specified in the **Report File** field. If no work file was specified, the report is displayed on a screen.

Load Libraries Only

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

To load libraries

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

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

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

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

2 Confirm your entries.

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

Load DDMs Only

This function of the INPL utility is used to load DDMs into the libraries indicated in the work file.

To load DDMs

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

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

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

2 Confirm your entries.

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

Load Error Messages Only

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

To load error messages

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

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

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

2 Confirm your entries.

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

Load All Objects

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

To load all objects and shared resources

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

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

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

2 Confirm your entries.

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

Scan INPL File

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

To scan an INPL File

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

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

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

2 Confirm your entries.

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

Natural Security Recover

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

The following options are provided:

- Reset Environment

Reset Environment

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

To reset the environment

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

Remove Owners

To remove owners

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

User Exit Routines

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

To activate a user exit routine

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

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

The following user exit routines are available:

Name	Function
`INPLUX01`	Prevent error message texts to be replaced.

INPLUX01

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

IV Natural Termcap (NATTERMCAP) Utility

5 Natural Termcap (NATTERMCAP) Utility

The Natural Termcap (NATTERMCAP) utility is used to create, modify and test terminal capabilities used by Natural. These terminal capabilities are stored in the terminal database SAGtermcap.

Since there are no standard terminal type definitions, Software AG does not assume any responsibility for the completeness and the correctness of the terminal types contained in SAGtermcap. A terminal type standard used for Digital Equipment Corporation's VT terminals is ANSI X3.64 (corresponds to ISO 6429).

General Settings

This section provides information on environment variables and parameters that can be used to specify general settings of the Natural Termcap utility. For the parameters that can be specified when invoking the Natural Termcap utility, see ***Dynamic Parameters***.

- Environment Variables
- Terminal Mode
- Special Control Codes

Environment Variables

The Natural Termcap utility uses the following environment variables:

Variable	Explanation
`TERM`	Used for the currently active terminal type.
`NATTERM`	Used for the Natural terminal type. If this variable is not set or defined, the value assigned to `TERM` (see above) is used.
`NATTCAP`	Used to specify a different terminal database than SAGtermcap.
`COLUMNS`	Used for terminal-screen width. If this variable is not set or defined, the current screen width is used. Otherwise, the value assigned to the capability `co` is used.
`LINES`	Used for terminal screen page size. If this variable is not set or defined, the current screen page size is used. Otherwise, the value assigned to the capability `li` is used.

Terminal Mode

The Natural Termcap utility is a screen I/O application. Therefore, it needs a terminal database and a terminal type to display menus and windows on the screen. By default, the Natural Termcap utility uses the same terminal database and type that Natural uses.

Natural retrieves the terminal type as follows:

1. It takes the contents of the environment variable `NATTERM`.
2. If `NATTERM` is not set or defined, it takes the contents of the system environment variable `TERM`.

Natural retrieves the terminal database as follows:

1. It takes the contents of the environment variable `NATTCAP`.
2. If `NATTCAP` is not set, it retrieves the terminal database name from the `NATTCAP` entry in the local configuration file *NATURAL.INI*.
3. If this terminal database could not be found, the Natural Termcap utility tries to locate a terminal database named SAGtermcap in the current directory.

To avoid this automatism, the Natural Termcap utility offers a predefined terminal database in the dynamic parameter `TERMCAP` with a limited number of terminal types. These types can be accessed by specifying the dynamic parameter `DISPLAY`.

Special Control Codes

The following table gives an overview of unprintable characters, as well as characters that have a special meaning in terminal capability syntax:

Control Code	Explanation
`\E`	Escape character
`\b`	Backspace character
`\n`	New line
`\r`	Carriage return character
`\t`	Tab character
`\xxx`	Octal value of xxx; must be three characters
`\072`	The character colon (:); the Natural Termcap utility uses the colon (:) as an internal separator
`^x`	Control-x, where x is any letter

Example:

If the function key PF10 is to be defined as F10 on a DEC VT220 terminal, the code of F10 is:

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

Specify the following for capability PF10:

```
\E[21~
```

If the character tilde (~) is not available on the keyboard, use the octal value of tilde instead. Tilde is defined as octal 176. The alternative specification is then:

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

Invoking the Natural Termcap Utility

To invoke the Natural Termcap utility

- At the operating system prompt, enter the following:

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

  where *dynamic-parameters* denotes one or more parameters that can be specified with the command (see ***Dynamic Parameters***).

  The **NATURAL Termcap Utility** screen appears with the name of the currently active terminal in the top right-hand corner of the screen.

  The screen provides the following menu options:

Menu	Explanation
File	Creates, reads, saves and deletes a terminal entry.
Edit	Views and sets terminal capabilities.
Search	Searches for a specific capability by name.
Test	Tests capabilities.
Options	Modifies the default key definition and shows or hides terminal copy capabilities. Terminal copy capabilities are capabilities included from another terminal entry.
Help	Provides help on each capability and on the usage of the dynamic parameters.

The menus provided on the **NATURAL Termcap Utility** screen are explained in the following sections.

Dynamic Parameters

The dynamic parameters that can be supplied with the `NATTERCAMP` command when invoking the Natural Termcap utility are described in the following section. These parameters provide quick access to the capabilities of a terminal type. As an alternative, you can use the menu options provided on the **NATURAL Termcap Utility** screen.

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

Dynamic Parameter	Explanation	
`DISPLAY`	Used to define the terminal type for the Natural Termcap utility itself. If no `DISPLAY` parameter is specified, the Natural terminal database SAGtermcap is used and the same terminal type tracking mechanism as for Natural is in effect.	
	Syntax: `DISPLAY = {#vt100|#vt100ng|#vt220|#vt220ng|wyse60|#tty|`*`other`*`}`	
	Possible Values:	
	`#vt100`	Use the terminal entry `DEC vt100` from the internal terminal database.
	`#vt100ng`	Similar to `#vt100`, but graphic line characters will be replaced by single characters such as `-` (minus signs), | (vertical lines) and + (plus signs).
	`#vt220`	Use the terminal entry `DEC vt220` from the internal terminal database.
	`#vt220ng`	Similar to `#vt220`, but graphic line characters will be replaced by single characters such `-` (minus signs), | (vertical lines) and + (plus signs).
	`#wyse60`	Use the terminal entry `wyse60` from the internal terminal database.
	`#tty`	Use the terminal entry `tty` from the internal terminal database. The tty terminal works in a line-oriented way without using escape control sequences. Only a few functions are available if this terminal entry is selected.
	`other`	Use any other terminal type in the terminal database, for example, `xterm`.
`EDIT`	Used to view and/or modify a specific terminal capability. If the capability is found, the associated window is displayed and the cursor is positioned in the specified field. If the capability cannot be found, an error message will be displayed and the Natural Termcap utility terminates. **Syntax:**	

Dynamic Parameter	Explanation
	`EDIT = capability` **Capability:** Any terminal capability known in Natural can be specified. A list of capabilities can be found in the sections *Terminal Capabilities - Overview* and *Terminal Capabilities - Sorted by Name*.
`EXIT`	Used to terminate the utility after all parameters have been processed. **Example:** `NATTERM EDIT = PF10 EXIT` After modifying the function key PF10, the utility terminates immediately.
`HELP`	Used to get help about a specific capability or about using the Natural Termcap utility. **Syntax:** `HELP = {CAP|USAGE|capability}` **Possible Values:** `CAP`: Displays help for all capabilities sorted by capability name. `USAGE`: Displays all dynamic parameters in the Natural Termcap utility. `capability`: Displays help for a specific capability.
`REPORT`	Used to create by default a text file with a detailed description of the current terminal in the Natural TMP directory. **Syntax:** `REPORT[=file-name]` where `file-name` is the name of the text file, which is to contain the description. If no name is specified, `terminal-name.txt` is used.
`SAVE`	Used to save all modifications of the current terminal entry. **Syntax:** `SAVE[=terminal-name]`

Dynamic Parameter	Explanation
	where *terminal-name* is a new terminal entry in the terminal database where all capabilities of the current terminal are stored. Corresponding menu option: **File** > **Save As**. If no name is specified, the current *terminal-name* is used. Corresponding menu option: **File** > **Save**.
TERM	Used to read in a different terminal entry. If this parameter is not specified, the current Natural terminal type is used (NATTERM or TERM). **Syntax:** `TERM = terminal-name` where *terminal-name* is any type of a given terminal contained in the terminal database.
TERMCAP	Used to work with a different terminal database. If this parameter is not specified, the current Natural terminal database is used (NATTCAP). **Syntax:** `TERMCAP = database-name` where *database-name* is the database path and file name.
TEST	Opens the specified test window. **Syntax:** `TEST = {CONSISTENCY|COLORS|GRAPHICS|KEYS|VIDEO}` **Possible Values:** CONSISTENCY: Checks whether the function keys are uniquely defined. COLORS: All available colors are displayed with sample text. GRAPHICS: A single-line and a double-line box are displayed. KEYS: A text on any pressed key will be displayed. The window can be closed by pressing one of the following character keys: E, Q, X, or . (period). VIDEO: Displays video attributes such as blinking, underlined and reversed video.

Terminating the Natural Termcap Utility

To terminate the Natural Termcap utility

- From the **File** menu, choose **Exit**.

  Or:

  Set the `EXIT` parameter as described in *Dynamic Parameters*.

  The Natural Termcap utility is terminated and the operating system prompt appears.

Terminal Copy Capabilities

Terminal copy capabilities (TCs) are capabilities transferred from another terminal entry, like the `#include` directive of a C program. However, if capabilities are already defined in the current entry, the transferred capabilities are ignored. This makes the entries more efficient, not only by reducing redundancies, but also by ensuring that related entries are kept consistent. Capabilities read from a terminal copy entry are marked with `[TC]` to the right of the input field. Additionally, the name of the entry from where this capability is transferred is shown in the top right corner of the menu, above the terminal name.

Once a terminal capability has been modified, it loses the link to the transferred terminal entry and the modification is made to the current terminal entry.

To display the current terminal entry without any terminal copy capabilities

1. On the **NATURAL Termcap Utility** screen, choose **Terminal Copy Capabilities** from the **Options** menu.

   The **Terminal Copy Capabilities** window appears.
2. Select **HIDE**.

Example:

Assume `TERM` is set to `vt100` and the vt100 (vt220) terminal entry in the terminal database looks as follows:

vt100 entry:

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

vt220 entry:

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

The combined terminal entry for the terminal type vt100 would be:

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
```

The capability `ti( = \E[0m)` from the vt220 terminal is ignored, because `ti` is already defined in the vt100 entry.

Key Definitions

The function keys provided in the definition windows of the Natural Termcap utility are described in the following table:

Key	Explanation
CTRL+A	Inserts ANSI definitions.
CTRL+E	Evaluates keys automatically.
CTRL+N	Inserts non-graphic characters for frames.
CTRL+P	Gets help.
CTRL+V	Tests capabilities.

To modify the predefined function keys

1 On the **NATURAL Termcap Utility** screen, choose **Key Assignments** from the **Options** menu.

The **Key Assignments** window appears.

2 Modify the required function key(s) listed under the **Key Name** column.

You can only specify control keys (CTRL+A to CTRL+Z).

File Menu

When you select **File** from the **NATURAL Termcap Utility** screen, a selection list containing the following functions is displayed:

Function	Explanation
New	Creates a new terminal entry in the current terminal database.
Read	Reads a terminal entry from the terminal database.
Save	Saves terminal capabilities to the current terminal entry.
Save As	Saves terminal capabilities to a different or new terminal entry.
Delete	Removes the current terminal entry from the terminal database.
Generate Report	Generates a text file including information about the description, aliases and capabilities of the current terminal entry. The text file will be stored by default in the Natural TMP directory as `terminal-name.txt`, for example, `xterm.txt`.
Move	Moves the terminal entry physically to the top of the terminal database. If a terminal is on top of the database, the access time during the terminal initialization will be improved.
Import Database	Allows working with a terminal database other than Natural's SAGtermcap.
Export Database	Saves the whole terminal database and all terminal entries with a different path and/or name than Natural's SAGtermcap.
Properties	Displays detailed information about the terminal database, terminal entry, environment variables and display type.
Exit	Exits the Natural Termcap utility.

Edit Menu

When you select **Edit** from the **NATURAL Termcap Utility** screen, a selection list containing all capabilities grouped by topic is displayed:

- 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

After you have selected a topic, the corresponding window is displayed in which you can edit individual Natural terminal capabilities.

Depending on the window displayed, the following different types of input fields are provided:

- boolean, where only `ON` or `OFF` can be specified.
- numeric, where only digits (0 to 9) can be specified.
- string, where 32 alphanumeric characters can be specified, with the exception of terminal capabilities `te` and `ti`, for which 132 characters can be specified;
- description, where 132 alphanumeric characters can be specified.

For further information on the individual Natural terminal capabilities to be edited, see the sections *Terminal Capabilities - Overview* and *Terminal Capabilities - Sorted by Name*.

Search Menu

When you select **Search** from the **NATURAL Termcap Utility** screen, a selection list containing all capabilities sorted by name is displayed: see *Terminal Capabilities - Sorted by Name*. After you have selected a capability, a window is displayed which corresponds to the window invoked with the appropriate **Edit** menu option.

Test Menu

When you select **Test** from the **NATURAL Termcap Utility** screen, a selection list containing the following functions is displayed:

Function	Explanation
Colors	Tests all foreground and background colors.
Consistency	Searches for inconsistent key definitions. A list of affected keys is displayed if they are not unique.
Keys	Displays the name of a pressed key. Leave this functions by pressing one of the following character keys: `E`, `Q`, `X`, or `.` (period).
Line Graphics	Tests the graphic line capabilities used for drawing window frames.
Video Attributes	Tests all video attributes.

Options Menu

When you select **Options** from the **NATURAL Termcap Utility** screen, a selection list containing the following functions is displayed:

Function	Explanation
Terminal Copy Capabilities	Shows or hides the terminal capabilities included from a different terminal entry specified by the capability `tc`.
Key Assignments	Modifies the default key assignments of the utility.

Help Menu

When you select **Help** from the **NATURAL Termcap Utility** screen, a selection list containing the following functions is displayed:

Function	Explanation
Topics	Invokes a detailed help section for a given topic, such as `NAME`, `REPORT`, `TC` or `TEST`.
Capabilities	Invokes a detailed help section for each capability.
Usage	Displays information on how the dynamic parameters are used.
About	Displays product information.

Terminal Capabilities - Overview

This section provides an overview of all terminal capabilities sorted by topic. The topics correspond to the items in the **Edit** menu.

Colors

Name	Description
ct	Terminal type: color (`ON`) or monochrome (`OFF`)
bgbla	Screen background color
fgblu	Foreground color blue; Natural color definition `CD=BL` (*)
fggre	Foreground color green; Natural color definition `CD=GR` (*)
fgmag	Foreground color pink; Natural color definition `CD=PI` (*)
fgred	Foreground color red; Natural color definition `CD=RE` (*)
fgcya	Foreground color turquoise; Natural color definition `CD=TU` (*)
fgwhi	Foreground color white; Natural color definition `CD=NE` (*)
fgyel	Foreground color yellow; Natural color definition `CD=YE` (*)
ctres	Foreground color for reverse video; Natural attribute definition `AD=V` (*)
bgblu	Background color blue
bggre	Background color green
bgmag	Background color pink
bgred	Background color red
bgcya	Background color turquoise
bgwhi	Background color white
bgyel	Background color yellow

* For detailed information on the Natural definitions `AD` and `CD`, see the appropriate session parameters `AD` and `CD` described in the *Natural Reference* documentation.

Cursor Keys and Modes

Cursor keys can be set in two modes: application mode or normal (numeric) mode. In application mode, the numeric keypad keys are assigned different tasks than when in normal mode.

Name	Description
kd	Cursor key down (in normal mode)
kl	Cursor key left (in normal mode)
kr	Cursor key right (in normal mode)
ku	Cursor key up (in normal mode)
@7	Cursor key end
kh	Cursor key home
CKNO	Normal cursor key mode
CKAP	Application cursor key mode
cm	Cursor motion
CNL	Cursor next line
ve	Cursor visible
vi	Cursor invisible
DK	Cursor key down (in application mode)
LK	Cursor key left (in application mode)
RK	Cursor key right (in application mode)
UK	Cursor key up (in application mode)

Description and Comments

These fields can be used to describe the terminal entry or to add some comments.

Editing Keys

Name	Description
bc	Backspace key
bcvt	Alternative backspace key
bt	Backtab key
cr	Carriage return key
dc	Delete character key
KDEL	Delete to end of field key

Name	Description
ESC	Escape key
ETO	Escape timeout value in milliseconds
%1	Help key
kI	Insert or overstrike mode key
NLFF	Next line first field key
kN	Page down (next) key
PD	Alternative page down (next) key
kP	Page up (previous) key
PU	Alternative page up (previous) key
&2	Refresh key
ta	Tab key

Initialization and Reset

Name	Description
TICL	Clear screen after initialization
TIRA	Reset attributes after initialization
TIGR	Enable line graphics after initialization
TICI	Cursor invisible after initialization
TICV	Cursor visible after initialization
TIAK	Application keypad after initialization
TINK	Numeric keypad after initialization
TIAC	Application cursor key mode after initialization
TINC	Normal cursor key mode after initialization
TIDB	Dark background after initialization
TILB	Light background after initialization
TIIM	Insert mode after initialization
TIOM	Overstrike mode after initialization
ti	Additional initialization sequence
TECL	Clear screen after termination
TERA	Reset video attributes after termination
TENL	Cursor next to line after termination
TECV	Cursor visible after termination
TECI	Cursor invisible after termination
TEAK	Application keypad after termination
TENK	Numeric keypad after termination

Name	Description
TEAC	Application cursor key mode after termination
TENC	Normal cursor key mode after termination
TEDB	Dark background after termination
TELB	Light background after termination
te	Additional sequence after termination

Keypad Keys for Mathematical Operations

Keypad	Description
KP01	Single null
KP1	One
KP2	Two
KP3	Three
KP4	Four
KP5	Five
KP6	Six
KP7	Seven
KP8	Eight
KP9	Nine
KPADD	Add
KPSUB	Subtract
KPDIV	Divide
KPMUL	Multiply
KPTS	Thousand separator
KPDP	Decimal point
KPRES	Result

Line Graphics

Name	Description
eA	Enable line graphics
as	Graphics mode on
ae	Graphics mode off
G1	Single upper-right corner character
GD1	Double upper-right corner character
G2	Single upper-left corner character

Name	Description
GD2	Double upper-left corner character
G3	Single lower-left corner character
GD3	Double lower-left corner character
G4	Single lower-right corner character
GD4	Double lower-right corner character
GH	Single horizontal bar character
GDH	Double horizontal bar character
GV	Single vertical bar character
GDV	Double vertical bar character

Miscellaneous

Name	Description
bl	Audio bell
vb	Visual bell
cl	Clear screen
ce	Clear to end of line
ks	Keypad mode application
ke	Keypad mode numeric
xi	Scroll glitch
TCS	External terminal/printer character set, for more information, see *Support of Different Character Sets with NATCONV.INI* in the *Operations* documentation.
tc	Terminal copy

Name and Aliases

A name and up to 30 aliases can be defined for each entry.

PA and PF Keys

Key	Description
PA1	Attention key PA1
PA2	Attention key PA2
PA3	Attention key PA3
PF1	Function key PF1
PF2	Function key PF2
...	

Key	Description
PF47	Function key PF47
PF48	Function key PF48

Right-To-Left Support

Name	Description
RTLF	Right-to-left language toggle key for fields
RTLS	Right-to-left screen toggle key
RTLM	Set RTL input mode
LTRM	Set LTR input mode

Screen Dimension and Appearance

Name	Description
li	Number of screen rows (if not specified, take the current screen size)
co	Number of screen columns (if not specified, take the current screen size)
DAR	Dark background, light text
LIG	Light background, dark text

Video Attributes

Name	Description
mb	Blinking on; Natural attribute definition `AD=B` (*)
BR	Blinking off
adc	Cursive/italics on; Natural attribute definition `AD=C` (*)
adc0	Cursive/italics off
md	Intensified (bold) on; Natural attribute definition `AD=I` (*)
HR	Intensified off
mr	Reversed on; Natural attribute definition `AD=V` (*)
mr0	Reversed off
us	Underlined on; Natural attribute definition `AD=U` (*)
ue	Underlined off
me	Reset attributes; Natural attribute definition `AD=D` (*)
so	Standout mode on
se	Standout mode off
xs	Standout glitch

* For detailed information on the Natural attribute definition `AD`, see the appropriate session parameter `AD` described in the *Natural Reference* documentation.

Terminal Capabilities - Sorted by Name

This section lists all terminal capabilities sorted by name. These capabilities correspond to the capabilities in the **Search** menu.

Name	Description
%1	Help key
&2	Refresh key
@7	Cursor key end
adc	Cursive/italic on; Natural attribute definition `AD=C` (*)
adc0	Cursive/italic off
ae	Graphics mode off
as	Graphics mode on
bc	Backspace key
bcvt	Alternative backspace key
bgbla	Screen background
bgblu	Background color blue
bgcya	Background color turquoise
bggre	Background color green
bgmag	Background color pink
bgred	Background color red
bgwhi	Background color white
bgyel	Background color yellow
bl	Audio bell
BR	Blinking off
bt	Backtab key
ce	Clear to end of line
CKAP	Application cursor key mode
CKNO	Normal cursor key mode
cl	Clear screen
cm	Cursor motion
CNL	Cursor next line
co	Number of screen columns; if not specified, take the current screen size
cr	Carriage return key

Name	Description
ct	Color terminal
ctres	Foreground color for reverse video; Natural attribute definition `AD=V` (*)
DAR	Dark background, light text
dc	Delete character key
DK	Cursor key down (in application mode)
eA	Enable line graphics
ESC	Escape key
ETO	Escape timeout value
fgblu	Foreground color blue; Natural color definition `CD=BL` (*)
fgcya	Foreground color turquoise; Natural color definition `CD=TU` (*)
fggre	Foreground color green; Natural color definition `CD=GR` (*)
fgmag	Foreground color pink; Natural color definition `CD=PI` (*)
fgred	Foreground color red; Natural color definition `CD=RE` (*)
fgwhi	Foreground color white; Natural color definition `CD=NE` (*)
fgyel	Foreground color yellow; Natural color definition `CD=YE` (*)
G1	Single upper-right corner character
G2	Single upper-left corner character
G3	Single lower-left corner character
G4	Single lower-right corner character
GD1	Double upper-right corner character
GD2	Double upper-left corner character
GD3	Double lower-left corner character
GD4	Double lower-right corner character
GDH	Double horizontal bar character
GDV	Double vertical bar character
GH	Single horizontal bar character
GV	Single vertical bar character
HR	Intensified off
kd	Cursor key down (in normal mode)
KDEL	Delete to end of field key
ke	Keypad mode numeric
kh	Cursor key home
kI	Insert or overstrike mode key
kl	Cursor key left (in normal mode)
kN	Page down (next) key
kP	Page up (previous) key

Name	Description
KP01	Single zero (0) keypad key
KP1	One
KP2	Two
KP3	Three
KP4	Four
KP5	Five
KP6	Six
KP7	Seven
KP8	Eight
KP9	Nine
KPADD	Add
KPDIV	Divide
KPDP	Decimal point
KPMUL	Multiply
KPRES	Result
KPSUB	Subtract
KPTS	Thousand separator
kr	Cursor key right (in normal mode)
ks	Keypad mode application
ku	Cursor key up (in normal mode)
li	Number of screen rows; if not specified, take the current screen size
LIG	Light background, dark text
LK	Cursor key left (in application mode)
LTRM	Set LTR input mode
mb	Blinking on; Natural attribute definition `AD=B` (*)
md	Intensified (bold) on; Natural attribute definition `AD=I` (*)
me	Reset attributes; Natural attribute definition `AD=D` (*)
mr	Reversed on; Natural attribute definition `AD=V` (*)
mr0	Reversed off
NLFF	Next line first field key
PA1	Attention key PA1
PA2	Attention key PA2
PA3	Attention key PA3
PD	Alternative page down (next) key
PF1	Function key PF1
PF2	Function key PF2

Name	Description
...	
PF47	Function key PF47
PF48	Function key PF48
PU	Alternative page up (previous) key
RK	Cursor key right (in application mode)
RTLF	Right-to-left language toggle key for fields
RTLM	Set RTL input mode
RTLS	Right-to-left screen toggle key
se	Standout mode off
so	Standout mode on
ta	Tab key
tc	Terminal copy
TCS	External terminal/printer character set.
te	Additional sequence after termination
TEAC	Application cursor key mode after termination
TEAK	Application keypad after termination
TECI	Cursor invisible after termination
TECL	Clear screen after termination
TECV	Cursor visible after termination
TEDB	Dark background after termination
TELB	Light background after termination
TENC	Normal cursor key mode after termination
TENK	Numeric keypad after termination
TENL	Cursor next to line after termination
TERA	Reset video attributes after termination
ti	Additional initialization sequence
TIAC	Application cursor key mode after initialization
TIAK	Application keypad after initialization
TICI	Cursor invisible after initialization
TICL	Clear screen after initialization
TICV	Cursor visible after initialization
TIDB	Dark background after initialization
TIGR	Enable line graphics after initialization
TIIM	Insert mode after initialization
TILB	Light background after initialization
TINC	Normal cursor key mode after initialization

Name	Description
TINK	Numeric keypad after initialization
TIOM	Overstrike mode after initialization
TIRA	Reset attributes after initialization
ue	Underlined off
UK	Cursor key up (in application mode)
us	Underline on; Natural attribute definition `AD=U` (*)
vb	Visual bell
ve	Cursor visible
vi	Cursor invisible
xi	Scroll glitch
xs	Standout glitch

* For detailed information on the Natural definitions `AD` and `CD`, see the appropriate session parameters `AD` and `CD` described in the *Natural Reference* documentation.

V Object Handler

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

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

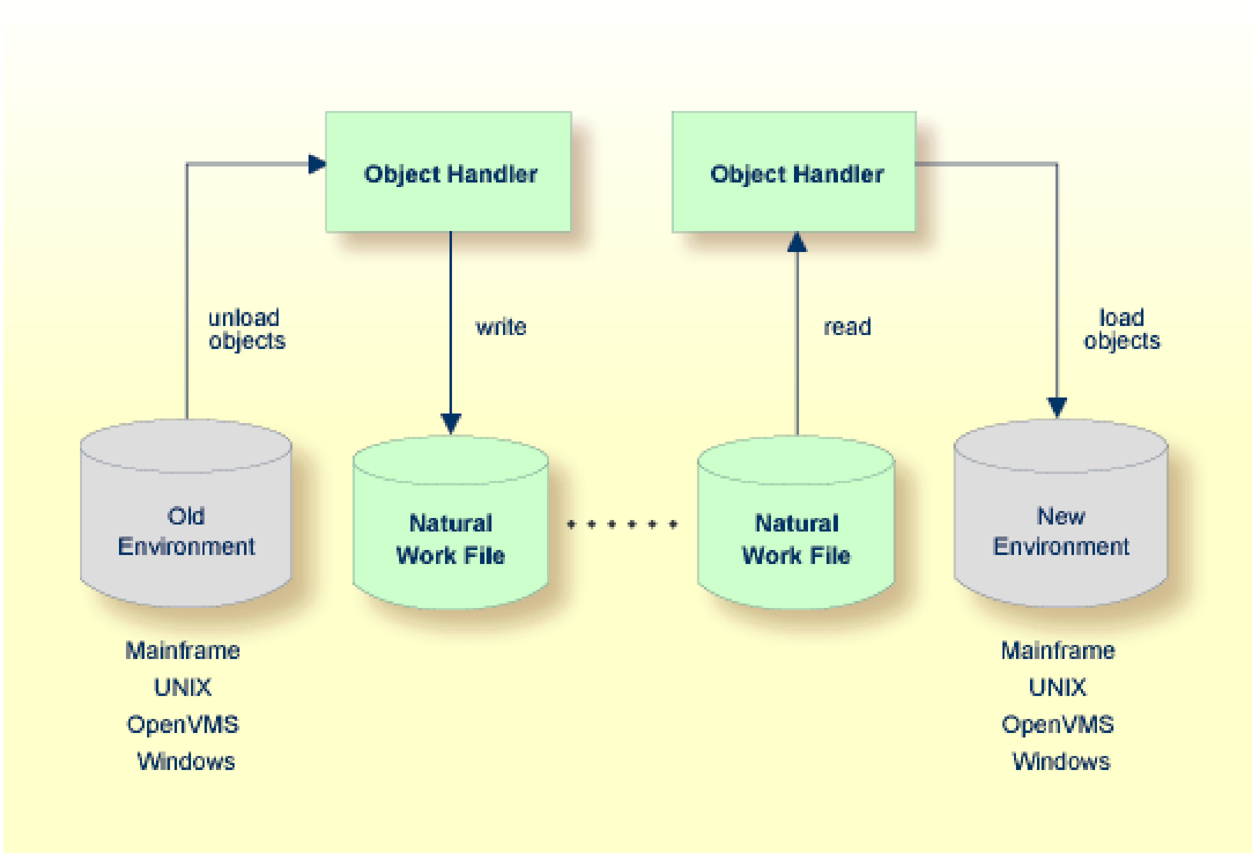
General Information on the Object Handler	Invoking the Object Handler in batch or online mode; applying Natural Security.
Functions	Using the Object Handler menu functions: unload, load, restart load, scan, view, find and administration.
Object Specification	Specifying the objects to be processed with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

6 General Information on the Object Handler

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

Principles of Object Transfer

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



This section covers the following topics:

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

Transfer Environment and File Security

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

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

Objects Processed by the Object Handler

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

Formatting Options

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

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

Text format applies to source objects, Natural command processor sources, error messages and Adabas FDTs. You can only transfer text data between mainframe and UNIX/OpenVMS/Windows platforms. You can transfer binary data between identical platforms. Between UNIX or OpenVMS and Windows platforms, you can transfer binary data by using portable work files of internal format.

Invoking the Object Handler

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

To invoke the Object Handler online from any Natural library

1. Enter the following system command:

   ```
   SYSOBJH
   ```

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

   - Unload
   - Load
   - Scan
   - View
   - Administration

   See the section ***Functions*** for descriptions of these functions, and how to process the functions in advanced-user mode or by using wizards.
2. Select a function by choosing one of the following methods:

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

   Or:

   Choose the PF key that corresponds to the function required.

   Or:

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

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

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

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

Batch or Direct Command Calls

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

The section covers the following topics:

- Batch Mode
- Online Mode

Batch Mode

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

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

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

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

Related Topics:

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

Online Mode

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

Example:

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

Issuing Object Handler Commands from a Natural Program

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

Natural Security

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

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

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

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

Standard PF Keys

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

PF Key	Explanation
PF1	Invokes the help function for the field at which the cursor is positioned. If positioned at the fields **Work file**, **External path**, **Ext. Path**, **Object name**, **Report file** or **Restart file**: invokes an extra window where you can enter a long work file name of up to 253 characters.
PF3	Exits the current screen and returns to the previous screen.
PF6	Goes to the top of a list.
PF7	Scrolls up one page in a list. On wizard screens: goes back one screen/step.
PF8	Scrolls down one page in a list. On wizard screens: goes to the next screen/step.
PF9	Goes to the bottom of a list.
PF10	Invokes the **Commands** menu to select commands for navigation purpose and to assign special settings. See also ***Commands for Navigation and Special Functions*** in *Direct Commands*.
PF12	Cancels the current function.
PF20	Lists all active programs of the Object Handler. This can be helpful information for reporting technical problems to Software AG.

Using FDDM System Files

Natural DDMs (data definition modules) can be stored in libraries or the system file FDDM. See also: *FDDM - Natural System File for DDMs* in the *Parameter Reference* documentation).

To use the system file FDDM for processing DDMs with the load, unload or find function, the Object Handler provides the option **Use FDDM file for processing DDMs**. This option is set by using **Set additional options** (see the section *Settings*).

Consider the following when selecting **Use FDDM file for processing DDMs**:

- This option is selected by default if FDDM has been activated in the NATPARM parameter file.
- You cannot process DDMs that are stored in libraries.
- You need to specify the library SYSTEM and the Natural object type V (see ***Natural Library Object Details*** in the section *Object Specification*.

- If used with the load function, all DDMs are loaded into the system file FDDM. In this case, the parameter `NEWLIBRARY` is ignored.

7 Functions

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

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

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

This section covers the following topics:

Notes:

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

8 Wizards

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

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

To activate the wizards

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

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

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

Step 1 - Start the Procedure

To start the unload, load or scan procedure

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

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

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

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

  Or:

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

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

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

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

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

3 Choose ENTER to continue.

4 Proceed with *Step 5 - Execute Processing*.

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

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

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

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

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

Field	PF Key	Explanation
Unicode work file		Only applies to the unload function and if **Transfer format** has been selected. If this option is selected, all object sources are converted to Unicode/UTF-8 (Universal Transformation Format, 8-bit form) before they are written to the work file.
Use PC File		Only applies if Entire Connection is installed. If selected, the data to be processed is read from or written to an Entire Connection work file.
Portable work file		Not required for the load and scan functions, which automatically choose the appropriate work file type and ignore this option if set. **Portable work file** is only valid if the following applies: ■ **Use default options** (this is the default) has been selected. ■ **Transfer format** has *not* been selected. If **Portable work file** has been selected, the work file is written or read in portable format. See also ***Work File Format*** in *Work Files*.
Work file		Only valid if **Use default options** (this is the default) is selected. ■ The name of the work file to be used for the function. If the name exceeds the space available, choose PF11 (WorkF) and enter a longer name of up to 253 characters. Alternatively, position the cursor at this field and choose PF1 (Help). See *Work Files*. ■ Or, if **Use PC File** has been selected, the name of the path and the Entire Connection work file to be used. See *Work Files*.
Use default options		Default options are used (this is the default). For the options available, see *Profile Settings* and *Set Additional Options* in *Settings*.
Set additional options	PF4	Only valid if **Use default options** has been selected. Invokes the **Options** screen of the wizard where you can modify the default settings and enter additional options for the processing sequence. See *Set Additional Options* in *Settings*.
Use Option Workplan		If selected, a Workplan of the type OPTION is used (see *Workplans*).
Name		Only valid if **Use Option Workplan** has been selected. The name of a Workplan of the type OPTION to be used.

Field	PF Key	Explanation
List Option Workplan	PF6	Only valid if **Use Option Workplan** has been selected. Displays the contents of the Workplan specified in the field Name.
Select Option Workplan	PF5	Only valid if **Use Option Workplan** has been selected. Displays a selection list of available Workplans of the type OPTION (see also *List the Available Workplans in the Workplan Library* in *Administration*).

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

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

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

Step 3 - Set Parameters

To set parameters for the processing procedure

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

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

Field	PF Key	Explanation
Select Parameter Workplan	PF5	Only valid if the field **Use Parameter Workplan** has been selected. If selected, a selection list of available Workplans of the type PARAMETER is displayed (see *List the Available Workplans in the Workplan Library* in *Administration*).

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

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

Step 4 - Select Objects

To select the type of object you want to process

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

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

   Choose ENTER or choose PF8 (Next) to continue.

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

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

   Choose ENTER or choose PF8 (Next) to continue.

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

- Type in the name of a Workplan.

  Or:

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

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

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

2 Choose ENTER or PF8 (Next) to continue.

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

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

Step 5 - Execute Processing

To execute the processing procedure

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

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

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

2 Choose ENTER to continue.

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

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

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

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

Or:

Choose PF12 to terminate the function.

The **Main Menu** appears.

Step 6 - Continue Processing

To continue processing

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

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

2 Choose **Yes**.

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

9 Advanced User

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

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

Activating Advanced User

To activate advanced-user mode

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

  Or:

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

Processing Objects

Note: To load FDTs, see also *FDTs* in the section *Object Specification*.

To process objects in advanced-user mode

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

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

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

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

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

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

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

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

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

10 Compact Mode

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

How to Select Compact Mode

There are two ways to select compact mode:

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

How Instructions are Processed in Compact Mode

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

- Compact Mode Main Menu
- Compact Mode Subsequent Menu

Compact Mode Main Menu

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

Field	PF Key	Explanation
Function		Choose a function: U unload (default) L load S scan

Field	PF Key	Explanation
		Caution: Once the unload function has been started, further modifications for **Function** are not permitted unless you terminate the unload function by exiting the menu and restart compact mode.
Object		Type of object to be processed: L Natural library object (default) E Natural system error message C Natural command processor source R Natural-related object X External Object F FDT A Any Natural object. Only available if `L` or `S` has been selected for **Function.**
Work file format		File format for loading and unloading operations to be applied to data to be processed: I Internal format (default) T Transfer format See *Work File Format*.
Portable work file		Not required for the load and scan functions, which automatically choose the appropriate work file type and ignore this option if set. **Portable work file** is only valid if the following applies: ■ **Use default options** (this is the default) has been selected. ■ **Transfer format** has *not* been selected. If **Portable work file** has been selected, the work file is written or read in portable format. See also *Work File Format* in *Work Files*.
Use PC file		Only applies if Entire Connection is installed. If selected, the data to be processed is read from or written to an Entire Connection work file.
PC file		Only applies if Entire Connection is installed. The complete path name to the Entire Connection work file. If your system environment does not accept a backslash (\) separator, use a slash (/) instead.
Write report		If set to `Y` writes a report of the objects processed to the report text object specified in the**Report text member** field. The **Write report** option is set to `Y` by default. To display the report, enter the internal command `SHOW REPORT FILE` (see *Commands for Navigation and Special Functions* in *Direct Commands*).

Field	PF Key	Explanation
Report text member		Only valid if **Write report** has been selected. The name of the text object stored in the Workplan library to which the report is written.
Set additional options	PF4	If set to "Y" invokes the **Options** screen where you can modify the default settings and enter additional options for the processing sequence. For the options available, see*Set Additional Options*. Cannot be used / Option specifications are ignored when **Use Option Workplan** is set to 'Y'. Note: If any Options have been defined, the text '(Options are defined) ' is displayed.
Use Option Workplan		If set to `Y`, a Workplan of type `OPTION` is used. See also *Workplans*
Option Workplan name		The name of a Workplan of type `OPTION` to be used.
	PF5	Display a selection list of available Workplans of type `OPTION`. See also *List the Available Workplans in the Workplan Library* in *Administration*.
	PF6	Displays the contents of the Workplan specified in the **Option Workplan name** field.
Set global parameters	PF7	Invokes the **Parameters** screen. See *Set Global Parameters* and *parameter-setting* (*Direct Commands*) for descriptions of keywords and valid input values. Cannot be used / Parameter specifications are ignored when **Use Parameter Workplan** is set to `Y`. Note: If any Parameters have been defined, the text '(`Parameters are defined`) ' is displayed.
Use Parameter Workplan		If set to `Y`, a Workplan of the type PARAMETER is used. See also *Workplans*.
Parameter Workplan name		The name of the `PARAMETER` Workplan to be used.
	PF8	Displays a selection list of available Workplans of the type `PARAMETER`. See also *List the Available Workplans in the Workplan Library* in *Administration*.
	PF9	Display the contents of the Workplan specified in **Parameter Workplan name**.

Compact Mode Subsequent Menu

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

11 Restart Load

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

The restart load requires that restart information is written to Work File 6 or a specified restart file in accordance with the selection criteria, options and parameter settings specified for the load.

To set up the environment during the load

1. On the **Load Options** screen:

   - Mark the **Write restart information** option.
   - In the **Restart file** field, enter the name of the file to which the restart information data is written.

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

2. Execute the load function.

To execute the restart load after an interrupted load

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

  ```
  GO RESTART
  ```

  The **Restart Options** screen appears, where you can specify a file by entering a name in the **Restart file** field.

  Or:

  Use the following direct command:

  ```
  RESTART
  ```

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

Related Topics:

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

12 View

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

To invoke the view function

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

  Or:

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

  ```
  GO VIEW
  ```

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

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

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

Natural Library Objects

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

To view Natural library objects

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

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

   For explanations of the screen columns, see the **Select System File** screen with identical columns, which are described in *Select System File*.
2. In the **Cmd** column, enter any single character next to the system file you want to select. The current FUSER or FNAT system file is selected by default.

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

   You can start the list of libraries from a particular library, or filter objects by entering a library name or a range of names in the **Library** field. For valid name ranges, see *Name* in the section *Name, Date and Time Specification*.
3. In the **Cmd** column, next to the library you want to select, enter one of the following line commands:

   ```
   L
   ```

```
LI
```

```
S
```

```
SE
```

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

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

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

```
L
```

```
LI
```

Or:

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

```
DE
```

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

Natural System Error Messages

To view Natural system error messages

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

The **View System Error Messages** screen appears with a list of all system error messages available in Natural.

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

2 In the **Cmd** column, next to the error message you want to view, enter either of the following line commands:

```
L
```

```
LI
```

Or:

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

```
DE
```

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

Natural Command Processor Sources

To view Natural command processor sources stored in an Adabas file

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

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

2 If the Natural command processor sources required are not stored in the current FUSER system file (LFILE 190 is set as the default; see also the ***SYSNCP Utility*** documentation), enter the required database ID in the **DBID** field and the file number in the **FNR** field.

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

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

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

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

```
L
```

```
LI
```

```
S
```

```
SE
```

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

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

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

```
DE
```

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

FDTs

To view the FDTs available in an Adabas database

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

The **View FDTs** screen appears.

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

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

13 Find

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

To invoke the find function

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

  ```
  GO FIND
  ```

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

14 Administration

This function is used to maintain Object Handler Workplans.

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

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

List the Available Workplans in the Workplan Library

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

To list Workplans

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

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

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

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

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

- Columns and Commands on the List Workplans screen

Columns and Commands on the List Workplans screen

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

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

Column	PF Key	Explanation
Cmd		The following line commands can be entered in the input field next to the Workplan required: `C` or `CH` — Checks the syntax. Only applies to Workplans of the types PROCEDURE, SELECTION, PARAMETER and OPTION. `DE` — Deletes the Workplan. `ED` — Edits the Workplan. You can modify the name of a Workplan or its description in the **Save Workplan** window described in ***Saving a Workplan***. `EX` — Executes the Workplan. Only applies to Workplans of the type PROCEDURE. `L` or `LI` — Lists the Workplan. `S` or `SE` — Selects the Workplan to be used for the current function. Only applies if the **List Workplans** screen is invoked with the select function, for example, from the **Unload/Load/Scan Settings** screen.
Name		The name of the Workplan. You can enter a name or a range of names as described in ***Name*** in *Name, Date and Time Specification*.

Type		The type of Workplan such as PROCEDURE.
		
		Valid input values are:
		
		`PROCEDURE` or `P`
		`SELECTION` or `S`
		`LIST` or `L`
		`PARAMETER` or `A`
		`OPTION` or `O`
		`TEXT` or `T`
		
		You can also enter an asterisk (*) for all types, or any combination of the short types, for example `SL`.

Description		The description of the Workplan. You can enter a description or a range of descriptions as described in ***Name*** in *Name, Date and Time Specification*.
User ID		Only displayed with PF5. The ID of the user who created the Workplan.

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

Create a New Workplan

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

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

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

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

This section covers the following topics:

- Creating a PROCEDURE Workplan
- Creating a LIST Workplan

- Saving a Workplan

Creating a PROCEDURE Workplan

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

To create a PROCEDURE Workplan from a generated command

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

Or:

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

- Enter the following Object Handler command:

```
SET ADVANCEDCMD ON
```

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

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

2 Choose PF5 (Save).

The **Save Workplan** window appears.

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

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

Creating a LIST Workplan

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

Saving a Workplan

To save a Workplan

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

```
SAVE
```

Or:

Choose PF5 (Save).

The **Save Workplan** window appears.

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

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

Change the Workplan Library

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

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

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

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

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

15 Select System File

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

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

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

To select a system file from a list

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

   The **Select System File** window appears where the system files available in the current Natural environment are listed with their directory path names (**Path**), database IDs (**DBID**) and file numbers (**FNR**).
2. Select the system file you want to use for function processing by entering any single character in the **Sel** column next to the system file required.

   If you select `FNAT/FUSER` (selected by default), all libraries contained in the current FNAT and FUSER system files are used.

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

16 Select Library

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

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

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

To select a single library from a list

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

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

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

3 Choose ENTER.

  The **Library** field and the **DBID/FNR** fields of the **Unload Natural Library Objects** screen are filled with the specified name and numbers respectively. If no values (or `0`) are entered in the **DBID/FNR** fields, the current FUSER and FNAT system files are selected. For Natural command processor sources, the current setting of LFILE 190 is used.

To list and select multiple libraries

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

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

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

Or:

In the **DBID** and **FNR** fields, enter the database ID and file number of the system file that contains the libraries you want to select. If no values (or `0`) are entered, the current FUSER and FNAT system files are used. For Natural command processor sources, the current setting of LFILE 190 is used.

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

3 Choose ENTER.

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

4 Choose PF4 (Se Rng).

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

Note: If Natural command processor sources (which are stored in Adabas files) are to be unloaded, the libraries are searched on the Adabas file specified by DBID and FNR. If no values (or `0`) are specified for DBID and FNR, the current setting of LFILE 190 is used.

17 Select System Error Messages

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

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

To select Natural System error messages

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

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

  This screen is described in the following section.

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

Columns and Commands

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

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

Column	PF Key	Explanation	
Cmd		One of following line commands can be entered in the input field next to the system error message required:	
		`L` or `LI`	Lists the short and long texts of the message.

		`S` or `SE`
		`UL` or `U`
			Selects the message for subsequent unloading.
			Attention:
			Any of these commands only marks the message selected for subsequent processing. To execute the unload function, you need to choose **PF2** (Unloa) described below.

		`DE`	Deletes the message.

Column	PF Key	Explanation
		`DL` — Only deletes the long text of the message.
Number		The number of the system error message. You can enter a number or a range of numbers. Valid ranges are: *value** — All messages with numbers that begin with *value*. *value*> — All messages with numbers greater than or equal to *value*. Example: `10>` *value*< — All messages with numbers less than or equal to *value*. Example: `100<`
S/L		The kind of system error message text: `S` — Short text. `L` — Long text. `A` — Short and/or long text.
Language		The language code of the system error message. You can enter up to 8 valid language codes (for example, `1` for English) for the error messages to be selected. An asterisk (*) selects all language codes.
Error Message Text		The short text of the system error message.

	PF2	Starts unloading the system error messages selected for processing.
		
		As an alternative, in the Command line, you can enter either of the following direct commands:
		
		`UNLOAD`
		or
		`UNLD`

	PF11	Marks all system error messages listed for subsequent unloading with **PF2**.
		
		As an alternative, in the Command line, you can enter either of the following direct commands:
		
		`SELECT ALL`
		or
		`SEL ALL`

18 Select Objects

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

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

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

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

Columns and Commands on List Screens

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

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

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

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

Column	PF Key	Explanation
		For a user-defined error message, the message number and the language code is displayed. For example: `10 (Lang =1)` denotes message number 10 in language 1 (English).
Type		The type of Natural library object such as `Program`. Valid input values are one or more object-type codes such as `P` for program. For a list of codes, see `NATTYPE` in the section *`select-clause`*.
S/C		The kind of Natural library object: by default, all source (`S`) objects and/or cataloged (`C`) objects available are displayed on the screen. Valid input values are one or more of the following codes: `S` – Source objects only. `C` – Cataloged objects only. `S/C` – Both source and cataloged objects if both exist. `W` – All STOWed objects: source and cataloged objects with identical date and time. `*` – All source objects and/or cataloged objects. For a user-defined error message, this column contains the short text of the error message.
M		The programming mode of the Natural library object. By default, any mode is displayed. Valid input values are one or more of the following codes: `S` Structured mode only. `R` Reporting mode only. `*` Any mode, structured and/or reporting. For a user-defined error message, this column contains the short text of the error message.
Version		The Natural version under which the Natural library object was saved and/or cataloged (range specification not possible). For a user-defined error message, this column contains the short text of the error message.
User ID		The ID of the user who saved or cataloged the Natural library object. You can enter a single user ID or a range of user IDs: see ***Name***. For a user-defined error message, this column contains the short text of the error message.
Date		The date when the Natural library object was saved or cataloged. You can enter a date or a range of dates: see ***Date*** in *Name, Date and Time Specification*. For a user-defined error message, this column contains the short text of the error message.

Column	PF Key	Explanation
Time		The time when the Natural library object was saved or cataloged. You can enter a time or a range of times: see ***Time*** in *Name, Date and Time Specification*. For a user-defined error message, this column contains the short text of the error message.

	PF2	Starts unloading the objects selected for processing.
		
		As an alternative, in the Command line, you can enter either of the following direct commands:
		
		`UNLOAD`
		or
		`UNLD`

	PF11	Marks all objects listed for subsequent unloading with **PF2**.
		
		As an alternative, in the Command line, enter either of the following direct commands:
		
		`SELECT ALL`
		or
		`SEL ALL`

19 Object Specification

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

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

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

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

20 Object Specification - All Objects on the Work File

Only applies to the load or scan function.

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

21 Object Specification - Natural Library Objects

This section describes the options provided on the object-specification screens for processing Natural library objects. Natural library objects are programming objects (including Natural DDMs), user-defined error messages and shared resources.

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

Natural Library Objects

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

Field	PF Key	Explanation
DBID/FNR		Only applies to the unload function. The database ID (DBID) and file number (FNR) of the system file where the Natural libraries are stored. If no values (or `0`) are specified, the current FUSER or FNAT system file is used.
Select DBID/FNR	PF5 (advanced-user mode only)	Only applies to the unload function. Displays a selection list of system files available.
Library		The name of a library or a range of names: see *Name* in *Name, Date and Time Specification*.
Select library	PF4	Displays a selection list of all libraries available. See also *Select Library*.
Object name		The name of a Natural programming object or shared resource or a range of names: see *Name*. Only evaluated if the fields **Natural programming objects** (default setting) and/or **Shared resources** are selected on the screen **Natural Library Objects, Details**. See also *Natural Library Object Details*.
Select objects		Only applies to the unload function in advanced-user mode. If no library range is specified, a selection list of all Natural objects available is displayed (see also *Select Objects*).
Error number from/to		A valid range (1 - 9999) of user-defined error messages delimited by the first and the last message number. Only evaluated if the field **Error messages** (default setting) is selected on the screen **Natural Library Objects, Details** (see also *Natural Library Object Details*).
Details	PF6	Invokes the screen **Natural Library Objects, Details** where you can enter more detailed object specifications. See *Natural Library Object Details*.

Field	PF Key	Explanation
Settings	PF7	Only applies to functions executed in advanced-user mode. Invokes the **Unload/Load/Scan Settings** screen where you can specify option and parameter settings: see *Settings*.
Work file	PF11 PF1	The name of the work file to be used for the function. If the name exceeds the space available, choose PF11 (WorkF) and enter a longer name of up to 253 characters. Alternatively, position the cursor at this field and choose PF1 (Help). See also *Work Files*.

Natural Library Object Details

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

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

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

Field	PF Key	Explanation
Library		The name of a library or a range of names: see *Name* in *Name, Date and Time Specification*. Ranges are not allowed if the **Use Predict set** option is selected.
Select (Library)	PF4	Displays a selection list of all libraries available. See also *Select Library*.
DBID/FNR		See **DBID/FNR** in *Natural Library Objects* above.
Object Types: **Natural programming objects**		Natural programming objects including Natural DDMs (data definition modules). If the FDDM system file has been activated, see also **Use FDDM file for processing DDMs** in the section *Settings*.
Object Types: **Error messages**		User-defined error messages.
Object Types: **Shared resources**		Any non-Natural file that is used in a Natural environment and is maintained in the Natural library system.

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

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

Natural Library Object Properties

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

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

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

Field	Explanation
User ID	The ID of the user who saved or cataloged a Natural programming object. Specify a single user ID or a range of user IDs: see ***Name*** in *Name, Date and Time Specification*.
Programming mode	The programming mode of the Natural programming objects: `R` – Reporting mode only. `S` – Structured mode only. `A` – No mode check performed. This is the default setting.
Natural version	The Natural version of the Natural programming objects. You can also specify a range of versions: see ***Name***.
DDM DBID	The database ID (DBID) of the data definition modules (DDMs). Valid entries are: 1 to 65535 or 0 (all DBIDs)
DDM FNR	The file number (FNR) of the DDMs: Valid entries are: 1 to 65535 or 0 (all FNRs).
Object Date: **Select all objects (no date check)**	Selects all objects, regardless of their date.
Object Date: **Select objects modified between/and**	Selects all objects with a save or catalog date and/or time within the range specified in these fields by entering a precise start date and/or time and/or an end date and/or time. For valid input values, see ***Date*** and ***Time*** in *Name, Date and Time Specification*. Special dates allowed are: `TODAY`, `YESTERDAY`, `MONTH` and `YEAR`.
Object Date: **Select objects modified on**	Selects all objects with a save or catalog date and/or time that fits the date/time specified in these fields by entering a precise date and/or time. For valid input values, see ***Date*** and ***Time***. Special dates allowed are: `TODAY` and `YESTERDAY`.
Object Size: **Select all objects (no size check)**	Selects all objects, regardless of their size.
Object Size: **Select objects with size between/and**	Selects all objects with a size within the range specified in these fields by entering a start size and/or an end size.
Object Size: **Select objects with size**	Selects all objects with a size that fits the size specified in this field.

Natural Library Object Exceptions

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

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

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

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

Field	PF Key
Library	PF4
Select (Library)	
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 Library Object Exception Properties

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

The screen provides the following fields:

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

22 Object Specification - Natural System Error Messages

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

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

Natural System Error Messages

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

Field	PF Key	Explanation
Error number from/to		A range of Natural system error messages delimited by the first and the last message number. Select **Select system error messages** for a list of all system error messages available.
Details	PF6	Invokes the screen **Unload/Load/Scan Natural Library Objects, Details** where you can enter more detailed object specifications: see ***Natural System Error Message Details***.
Settings	PF7	Invokes the screen **Unload/Load/Scan Settings** where you can specify option and parameter settings. See ***Settings***.
Work file	PF11	The name of the work file to be used for the function: see also **Work file** in the section *Natural Library Objects* .
	PF1	

Natural System Error Message Details

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

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

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

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

Natural System Error Message Exceptions

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

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

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

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

23 Object Specification - Natural Command Processors

This section describes the options provided on the object-specification screens for processing Natural command processor sources (which are stored in Adabas files).

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

Natural Command Processors

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

Field	PF Key	Explanation
Library		The name of a Natural command processor library or a range of names: see *Name* in *Name, Date and Time Specification*.
Select library	PF4	Invokes a selection list of Natural command processor libraries available. See also *Select Library*.
DBID/FNR		Only applies to the unload function. The database ID (DBID) and the file number of the Adabas (FNR) file where the Natural command processor sources are stored. If no values are specified, the current setting of LFILE 190 is used. For details, see the *SYSNCP Utility* in the *Utilities* documentation.
Password/Cipher		Only applies to the unload function. The password and cipher code for the Adabas file where the Natural command processor sources are stored.
Object name		The name of a Natural command processor source or a range of names: see *Name*.
Select objects		Only applies to the unload function. If no library range has been specified and this field is selected, a selection list of Natural command processor sources available is displayed (see also *Select Objects*).
Exceptions	PF8	Invokes an extra screen where you can specify exceptions to the selection of Natural command processor sources: see *Natural Command Processor Source Exceptions*.
Settings	PF7	Invokes the **Unload/Load/Scan Settings** screen where you can specify option and parameter settings. See *Settings*.
Work file	PF11 PF1	The name of the work file to be used for this function: see also **Work file** in the section *Natural Library Objects*.

Natural Command Processor Source Exceptions

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

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

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

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

24 Object Specification - Natural-Related Objects

Only applies when processing data in internal format, that is, if **Transfer format** has *not* been selected. See also ***Work File Format*** in the section *Work Files*.

This section describes the options provided on the object-specification screens for processing Natural-related objects. Natural-related objects are objects that exist in a Natural environment but are not located in Natural libraries and Adabas files, such as the NATPARM parameter file, which is located in Natural path PARM_PATH.

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

Natural-Related Objects

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

Field	PF Key	Explanation
Natural path		The name of the path where the Natural-related object is located. Load and scan: The name of a path or asterisk (*) to select all paths. Valid input values are: `NATROOT`, `NATBIN`, `NATERR`, `NATSAG`, `PARM_PATH`, `PROFILE_PATH`, `TEXT_PATH`, `TMP_PATH`.
Select Natural path	PF4	Invokes a selection list of Natural paths available.
Object name	PF5 PF1	The name of a Natural-related object. Load and scan: A single name or a range of names: see ***Name*** in *Name, Date and Time Specification*. If the name exceeds the space available, choose PF5 (Objct) and enter a longer name of up to 253 characters. Alternatively, position the cursor at this field and choose PF1 (Help).
Details	PF6	Invokes the screen **Natural-related Objects, Details** where you can enter further object specifications: see ***Natural-Related Object Details***.
Settings	PF7	Invokes the screen **Unload/Load/Scan Settings** where you can specify option and parameter settings. See ***Settings***.
Work file	PF11 PF1	The name of the work file to be used for this function: see also **Work file** in *Natural Library Objects* in *Object Specification*.

Natural-Related Object Details

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

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

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

Field	PF Key	Explanation
Natural path		See **Natural path** in *Natural-Related Objects* above.
Select Natural path	PF4	Invokes a selection list of Natural paths available.
Object name	PF5	See **Object name** in *Natural-Related Objects* above.
Object Date: **Select all objects (no date check)**		Selects all objects, regardless of their date.
Object Date: **Select objects modified between/and**		See **Object Date** in *Natural Library Object Properties*.
Object Date: **Select objects modified on**		See **Object Date** in *Natural Library Object Properties*.
Object Size: **Select all objects (no size check)**		Selects all objects, regardless of their size.
Object Size: **Select objects with size between/and**		Selects all objects with a size within the range specified in these fields by entering a start size and/or an end size.
Object Size: **Select objects with size**		Selects all objects with a size that fits the size specified in this field.
Exceptions	PF8	Invokes an extra screen where you can specify exceptions to the selection of Natural-related objects: see *Natural-Related Object Exceptions*.

Natural-Related Object Exceptions

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

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

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

The screen **Unload/Load/Scan Natural-related Objects, Exceptions** provides the following fields and PF keys:

Field	PF Key	Explanation
Natural path	PF4	See **Natural path** in *Natural-Related Objects* above.
Select Natural path		Invokes a selection list of Natural paths available.
Object name	PF5	See **Object name** in *Natural-Related Objects* above.
Object Date: **Ignore object date**		Performs no date check. Objects are processed, regardless of their date.
Object Date: **Exclude objects modified between/and**		See **Object Date** in *Natural Library Object Exception Properties*.
Object Date: **Exclude objects modified on**		See **Object Date** in *Natural Library Object Exception Properties*.
Object Size: **Ignore object size**		Performs no size check. Objects are processed, regardless of their size.
Object Size: **Exclude objects with size between/and**		Exempts from processing all objects with a size within the range specified in these fields by entering a start size and/or an end size.
Object Size: **Exclude objects with size**		Exempts from processing all objects with a size that fits the size specified in this field.

25 Object Specification - External Objects

Only applies when processing data in internal format, that is, if **Transfer format** has *not* been selected. See also ***Work File Format*** in the section *Work Files*.

This section describes the options provided on the object-specification screens for processing external objects. External objects are objects that are located outside Natural and Adabas environments, such as bitmaps.

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

External Objects

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

Field	PF Key	Explanation
External path	PF4 PF1	The name of the path where the external object is located. Load and scan: The name of a path or asterisk (*) to select all paths. If the name exceeds the space available, choose PF4 (Path) and enter a longer name of up to 253 characters. Alternatively, position the cursor at this field and choose PF1 (Help).
Object name	PF5 PF1	The name of an external object. Load and scan: A single name or a range of names: see ***Name*** in *Name, Date and Time Specification*. If the name exceeds the space available, choose PF5 (Objct) and enter a longer name of up to 253 characters. Alternatively, position the cursor at this field and choose PF1 (Help).
Details	PF6	Invokes the screen **Unload/Load/Scan External Objects, Details** where you can enter further object specifications: see ***External Object Details***.
Settings	PF7	Invokes the screen **Unload/Load/Scan Settings** where you can specify option and parameters setting. See the section ***Settings***.
Work file	PF11 PF1	The name of the work file to be used for this function: see also **Work file** in *Natural Library Objects*.

External Object Details

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

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

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

Field	PF Key	Explanation
External path	PF4	See **External path** in *External Objects* above.
Object name	PF5	See **Object name** in *External Objects* above.
Object Date: **Select all objects (no date check)**		Selects all objects, regardless of their date.
Object Date: **Select objects modified between/and**		See **Object Date** in *Natural Library Object Properties*.
Object Date: **Select objects modified on**		See **Object Date** in *Natural Library Object Properties*.
Object Size: **Select all objects (no size check)**		Selects all objects, regardless of their size.
Object Size: **Select objects with size between/and**		See **Object Size** in *Natural Library Object Properties*.
Object Size: **Select objects with size**		Selects all objects with a size that fits the size specified in this field.
Exceptions	PF8	Invokes an extra screen where you can specify exceptions to the selection of external objects: see ***External Object Exceptions***.

External Object Exceptions

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

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

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

The screen **Unload/Load/Scan External Objects, Exceptions** provides the following fields and PF keys:

Field	PF Key	Explanation
External path	PF4	See **External path** in *External Objects* above.
Object name	PF5	See **Object name** in *External Objects* above.
Object Date: **Ignore object date**		Performs no date check. Objects are processed, regardless of their date.
Object Date: **Exclude objects modified between/and**		See **Object Date** in *Natural Library Object Exception Properties*.
Object Date: **Exclude objects modified on**		See **Object Date** in *Natural Library Object Exception Properties*.
Object Size: **Ignore object size**		Performs no size check. Objects are processed, regardless of their size.
Object Size: **Exclude objects with size between/and**		Exempts from processing all objects with a size within the range specified in these fields by entering a start size and/or an end size.
Object Size: **Exclude objects with size**		Exempts from processing all objects with a size that fits the size specified in this field.

26 Object Specification - FDTs

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

Note: When loading FDTs, all FDT data is written to Work File 5. You can use the contents of this work file as input for the Adabas utility ADAFDU.

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

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

Field	PF Key	Explanation
DBID		The database ID where the FDT is located. Load and scan: A valid DBID or 0 for all DBIDs.
FNR		The file number where the FDT is located. Load and scan: A valid FNR or 0 for all FDTs.
Password/Cipher		Only applies to the unload and load functions. The Adabas password and the cipher code of the Adabas file where the FDT is located.
Settings	PF7	Invokes the **Unload/Load/Scan Settings** screen where you can specify option and parameter settings. See *Settings*.
Work file	PF11 PF1	The name of the work file to be used for this function: see also **Work file** in *Natural Library Objects*.

27 Use Selection or List Workplan

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

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

Field	PF Key	Explanation
Name	PF4	The name of the Workplan to be processed.
List Workplan		Displays the contents of the Workplan specified in the **Name** field.
Select Workplan	PF5	Displays a list of all Workplans available. See also ***List the Available Workplans in the Workplan Library*** in *Administration*.
Settings	PF7	Invokes the **Unload/Load/Scan Settings** screen where you can specify option and parameter settings. See ***Settings***.
Work file	PF11 PF1	The name of the work file to be used for this function: see also **Work file** in *Natural Library Objects*.

28 Settings

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

To invoke the Unload/Load/Scan Settings screen

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

  ```
  SETTINGS
  ```

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

  Or:

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

  Or:

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

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

Settings Screen Fields

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

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

Field	PF Key	Explanation
Unicode work file		Only applies to the unload function and if **Transfer format** has been selected. If this option is selected, all object sources are converted to Unicode/UTF-8 (Universal Transformation Format, 8-bit form) before they are written to the work file. If a Unicode work file is specified, you cannot use the transfer options **Use conversion table**, **Substitute line references** and **Incorporate free rules**.
Use PC File		Only applies if Entire Connection is installed. If selected, the data to be processed is read from or written to an Entire Connection work file.
Portable work file		Not required for the load and scan functions, which automatically choose the appropriate work file type and ignore this option if set. **Portable work file** is only valid if the following applies: ■ **Use default options** (this is the default) has been selected and ■ **Transfer format** has *not* been selected. If **Portable work file** has been selected, the work file is written or read in portable format. See also *Work File Format* in *Work Files*.
Work file	PF11 PF1	Only valid if **Use default options** (this is the default) is selected. ■ The name of the work file to be used for the function. If the name exceeds the space available, choose PF11 (WorkF) and enter a longer name of up to 253 characters. Alternatively, position the cursor at this field and choose PF1 (Help). See *Work Files*. ■ Or, if **Use PC File** has been selected, the name of the path and the Entire Connection work file to be used. See *Work Files*.
Use default options		Default options are used (this is the default). See also *Profile Settings* and *Set Additional Options*.
Set additional options	PF4	Only valid if **Use default options** has been selected. Invokes the **Options** screen where you can modify the default settings and enter additional options for the processing sequence. For the options available, see *Set Additional Options*.
Use Option Workplan		A Workplan of the type OPTION is used. See also *Workplans*.
Name		Only valid if **Use Option Workplan** has been selected. The name of a Workplan of the type OPTION to be used.

Field	PF Key	Explanation
(next to Use Option Workplan)		
List Option Workplan	PF6	Only valid if **Use Option Workplan** has been selected. Displays the contents of the Workplan specified in the **Name** field next to **Use Option Workplan**.
Select Option Workplan	PF5	Only valid if **Use Option Workplan** has been selected. Displays a selection list of available Workplans of the type OPTION (see also *List the Available Workplans in the Workplan Library* in *Administration*).
Do not use parameters		If selected (default setting), no parameters are set.
Use global parameters		Global parameters are used. See *Set Global Parameters*.
Set global parameters	PF7	Only valid if **Use global parameters** has been selected. Invokes the **Parameters** screen. See *Set Global Parameters* and `parameter-setting` (*Direct Commands*) for descriptions of keywords and valid input values.
Use Parameter Workplan		A Workplan of the type PARAMETER is used. See also *Workplans*.
Name (next to Use Parameter Workplan)		Only valid if **Use Parameter Workplan** has been selected. The name of a Workplan of the type PARAMETER to be used.
List Parameter Workplan	PF9	Only valid if **Use Parameter Workplan** has been selected. Displays the contents of the Workplan specified in the **Name** field next to **Use Parameter Workplan**.
Select Parameter Workplan		Only valid if **Use Parameter Workplan** has been selected. Displays a selection list of available Workplans of the type PARAMETER. See also *List the Available Workplans in the Workplan Library* in *Administration*.

Set Additional Options

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

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

This section covers the following topics:

Work File and Report Options

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

Field	Explanation
Use PC File	Only applies if Entire Connection is installed. If selected, the data to be processed is read from or written to an Entire Connection work file.
Work file	Only valid if **Use default options** (this is the default) is selected. ■ The name of the work file to be used for the function. If the name exceeds the space available, choose PF11 (WorkF) and enter a longer name of up to 253 characters. Alternatively, position the cursor at this field and choose PF1 (Help). See *Work Files*. ■ Or, if **Use PC File** has been selected, the name of the path and the Entire Connection work file to be used. See *Work Files*.
Unicode work file	Only applies to the unload function and if **Transfer format** has been selected. If this option is selected, all object sources are converted to Unicode/UTF-8 (Universal Transformation Format, 8-bit form) before they are written to the work file. If a Unicode work file is specified, you cannot use the transfer options **Use conversion table**, **Substitute line references** and **Incorporate free rules**.
Write report	Writes a report of the objects processed to Work File 4. The **Write report** option is selected by default. To display the report, enter the internal command `SHOW REPORT FILE` (see *Commands for Navigation and Special Functions* in *Direct Commands*).
Start new report	Only valid if **Write report** has been selected. Deletes the contents of Work File 4 before a new report is written. Otherwise, a new report is appended to the existing one.

Field	Explanation
Error report only	Only valid if **Write report** has been selected. Write only error messages to the report. This includes messages from Natural Security and messages that have incurred during the execution of a LOAD command, for instance "not replaced". See also `REPORT-OPTION-1` in *Direct Commands*, *option-setting*.
Report file	Only valid if **Write report** has been selected. The name of the report file: enter the complete path name assigned to Work File 4. If the name exceeds the space available, choose PF5 (RepoF) and enter a longer name of up to 253 characters. Alternatively, position the cursor at this field and choose PF1 (Help).
Write restart information	Only applies to the load function. When this option is set, restart information is provided for the **restart load** function. For details, see ***Restart Load*** in *Functions*.
Restart file	Only applies to the load function and if **Write restart information** has been selected. The name of the work file to be used for the restart data: Work File 6 (default setting) or the *`restart-file`* specified. If the name exceeds the space available, choose PF6 (RestF) and enter a longer name of up to 253 characters. Alternatively, position the cursor at this field and choose PF1 (Help).
Use FDDM file for processing DDMs	Only applies in environments where the FDDM system file has been activated in the NATPARM parameter file. If this option has been selected (this is the default), the FDDM system file is used for processing DDMs with the load, unload or find function. Specify the library `SYSTEM` and the Natural object type `V` (see ***Natural Library Object Details*** in *Object Specification*) for processing DDMs. If used with the load function, all DDMs are loaded into the FDDM system file. In this case, the parameter `NEWLIBRARY` is ignored. See also the syntax diagram of the *`option-clause`* in *Direct Commands*.
Delete allowed	Only applies to the load function. Processes delete instructions from work files when loading objects in internal format.

XREF Options

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

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

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

XRef Considerations

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

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

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

Consider the following settings for XREF:

- XREF set to OFF or No
- XREF set to ON or Yes or Force
- XREF set to Force
- XREF set to Doc
- XREF set to Special

- XREF option with Natural Security

XREF set to OFF or No

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

XREF set to ON or Yes or Force

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

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

XREF set to Force

Only applies to LOAD.

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

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

XREF set to Doc

Only applies to LOAD.

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

XREF set to Special

Only applies to LOAD.

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

XREF option with Natural Security

If Natural Security is active, the following applies:

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

Transfer Options

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

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

Option	Explanation	Function
Substitute line references	Only applies if source-code line numbers are used for statement references. If line numbers are used as references in the source code, the line numbers of referenced lines and the line number references are replaced with labels. The sources are not modified in the database.	Unload
Include line numbers	If you choose this option, the line numbers will be transferred. (By default, line numbers in Natural objects are not transferred.)	Unload

Option	Explanation	Function
Incorporate free rules	If Predict is installed, Predict rules associated with a map are incorporated into the map source.	Unload
Use conversion table	**Caution:** Use this option only in special cases, such as when performing a non-FTP transfer between environments with different character sets, where no conversion is performed by the transfer tool. Unload: Converts data to EBCDIC format by using the internal Natural conversion table (**System table**) or a conversion table defined by the user (**User table**). Load: Converts data to ASCII format by using the internal Natural conversion table (**System table**) or a conversion table defined by the user (**User table**). Note that this only applies if the data in the work file is in EBCDIC format or if a conversion program is specified (see **User table**).	Unload Load
System table	Only valid if **Use conversion table** has been selected. Unload: Converts data to EBCDIC format by using the internal Natural conversion table. Load: Converts data to ASCII format by using the internal Natural conversion table.	Unload Load
User table	Only valid if **Use conversion table** has been selected. If the name of a conversion program has been entered in the field, data is converted to EBCDIC or ASCII format by using the conversion program defined. To specify an individual conversion program, the program must be located in the library SYSOBJH or one of its steplibs. See the example subprograms OTNCONAE and OTNCONEA in the library SYSOBJH. If no conversion program is specified, by default, the corresponding conversion table in the Natural file *NATCONV.INI* is used for the unload ([ISO8859_1->EBCDIC]) and the load ([EBCDIC->ISO8859_1]) functions.	Unload Load
Use load code page	If you choose this option, a window appears where you can enter the name of the code page to be used for the load function. If this option is selected, all object sources unloaded into a work file in UTF-8 will be converted with the specified code page when they are loaded into a work file. See also **Unicode work file**. If you enter `*CODEPAGE` as the code page name, the value assigned to the system variable `*CODEPAGE` is used (see the *System Variables* documentation). If no code page name is specified, the source objects are converted with the code page used when unloading them. If **Use load code page** is specified, you cannot use the options **Use conversion table** and **Translate to upper case**.	Load

Option	Explanation		Function
Translate to upper case	Translates any source code to be loaded into upper case.		Load
Data area format	Only applies to data areas.		Unload Load
	Specifies the format in which to unload or load data area sources. Possible input values are:		
	`N`	Converts data areas to the new internal data area format.	
	`O`	Converts data areas to the old internal data area format. If one or more data area sources cannot be converted to the old internal data area format, the Object Handler issues a corresponding message when unloading is complete. In addition, in the **Status** column of the unload report generated by the unload function, a corresponding remark appears next to the names of the data area sources affected.	
	`*`	Does not convert data areas. This is the default.	
	For details, see *Data Area Editor* in the *Editors* documentation.		

Replace Options

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

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

Number to Process

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

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

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

FDIC Settings

FDIC settings only apply if Predict is installed.

They specify the Predict file (FDIC) to be used for processing XRef data:

DBID	The database ID where the FDIC file is located.
FNR	The file number where the FDIC file is located.
Password	Optional. The Adabas password of the Adabas file where the FDIC file is located.
Cipher	Optional. The cipher code of the Adabas file where the FDIC file is located.

FSEC Settings

FSEC settings only apply if Natural Security is installed.

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

DBID	The database ID where the FSEC file is located.
FNR	The file number where the FSEC file is located.
Password	Optional. The Adabas password of the Adabas file where the FSEC file is located.
Cipher	Optional. The cipher code of the Adabas file where the FSEC file is located.

Set Global Parameters

Only applies to the load or unload function.

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

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

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

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

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

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

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

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

Field/Section	Explanation
Error number difference	Only applies when processing Natural system error messages or user-defined error messages. A 4-digit positive or negative value (+/-*nnnn*) to be used as a new number range for error messages. Start and end values must be provided in the **Error number from/to** fields (see *Natural Library Objects*) to validate whether the new range can be applied to the selected error messages. Example: If **Error number from/to** selects message numbers 1 to 10 and **Error number difference** is set to `2000`, the messages will be renumbered from 2001 to 2010. A value of `-1000` in **Error number difference** would cause a validation error.
FDT DBID/FNR	**Check Value/New Value:** A valid database ID (DBID) and/or file number (FNR) for Adabas FDTs.
Ext. Path	**Check Value/New Value:** The name of the path for external objects. If the name exceeds the space available, choose PF6 (CPath) for **Check Value** or PF7 (NPath) for **New Value** and enter a longer name of up to 253 characters. Alternatively, position the cursor at either field and choose PF1 (Help).
System files for load: Load FNAT **DBID** **FNR**	Only applies to the load function. The database ID (**DBID**) and file number (**FNR**) of the target FNAT system file. This system file is used for all library objects whose library name starts with `SYS`, but not SYSTEM.
System files for load: Load FUSER **DBID** **FNR**	Only applies to the load function. The database ID (**DBID**) and file number (**FNR**) of the target FUSER system file. This system file is used for all library objects whose library name does not start with `SYS`, and for the library SYSTEM.
System files for load: Load FNAT/FUSER **Select**	Only applies to the load function. Invokes the **Select System File** window with a list of all system files available in your Natural environment: see ***Select System File***.
System files for load: Load NCP **DBID** **FNR**	Only applies to the load function. The database ID (**DBID**) and file number (**FNR**) of the target Adabas file into which the Natural command processor sources are to be loaded.

This section covers the following topic:

- Rules for New Values

Rules for New Values

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

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

Examples:

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

29 Workplans

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

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

Creating, Selecting and Modifying Workplans

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

Contents of Workplans

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

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

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

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

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

Examples of Workplans

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

Workplan Type	Instruction	Explanation
PROCEDURE	`FINDLIB * LIB TEST`	Check whether the library TEST exists.
PROCEDURE	`UNLOAD A* LIB TEST`	Unload from the library TEST into Work File 1 all Natural programming objects and shared resources starting with `A`, and all user-defined error messages; write the report into Work File 4.
SELECTION	`* LIB TEST`	Process all objects from the library TEST.
TEXT	`This is a Workplan comment.`	Any text.

This section covers the following topic:

Example of Workplan Contents

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

```
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
```

Referencing Workplans

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

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

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

The syntactical options are explained in the following section:

- Keyword Explanation

Keyword Explanation

The table below describes the keywords and values that apply to the syntax for referencing Workplans.

Keyword	Values	Default Value
workplan-name	The name of the Natural text object in the Workplan library to be used as the Workplan.	No default
LIBRARY	The name of the library where the Workplan is located.	WORKPLAN
DBID	The ID of the Adabas database where the Workplan library is located.	0 (current FNAT/FUSER)
FNR	The number of the Adabas file where the Workplan library is located.	0 (current FNAT/FUSER)
NAME	Only applies to objects on mainframes. The name of a valid VSAM file where the Workplan library is located.	blank (current FNAT/FUSER)
CIPHER	Only applies to objects on mainframes. An 8-digit cipher code.	blank (current FNAT/FUSER)
PASSWORD	Only applies to objects on mainframes. An 8-character Adabas password.	blank (current FNAT/FUSER)

30 Name, Date and Time Specification

You can use a name, a date, a time or a range of names, dates and times to select Natural library objects, Natural command processor sources, Natural-related objectsexternal files (external objects).

Name

You can specify a name or a range of names.

In the list of options below, `value` is any combination of one or more characters:

	Input	Items Selected
	`value`	All items with names equal to `value`.
	* >	All items.
	?	All items with any single character for each question mark (?) entered.
Leading Characters	`value`*	All items with names that start with `value`. Example: `AB*` Selected: AB, AB1, ABC, ABEZ Not selected: AA1, ACB
Wildcard	`value`?	All items with names that start with `value` and end with any single character for each question mark (?) entered. Example: `ABC?` Selected: ABCA, ABCZ Not selected: AXC, ABCAA
	`value`?`value`? `value`*`value`? *`value`?`value`*	All items that match `value` combined with asterisk (*) and question mark (?) in any order. Example: `A?C*Z` Selected: ABCZ, AXCBBBZ, ANCZ Not selected: ACBZ, ABDEZ, AXCBBBZA
Start Value	`value`>	All items with names greater than or equal to `value`. Example: `AB>` Selected: AB, AB1, BBB, ZZZZZZZ Not selected: AA1, AAB
End Value	`value`<	All items with names less than or equal to `value`. Example: `AX<` Selected: AB, AWW, AX Not selected: AXA, AY

Note: The parameter specification option **New Value** only allows leading characters (asterisk (*) notation). See ***Rules for New Values*** in *Set Global Parameters* in the section *Settings*.

Date

All date values within the Object Handler are specified in international date format.

You can specify a date, a range of dates, a special date or a range of special dates. A date must be specified in the format *YYYY-MM-DD* (*YYYY* = year, *MM* = month, *DD* = day).

In the list of options below, the underlined portion of a keyword represents its valid abbreviation, and *value* is any combination of one or more digits:

	Input Value	Items Selected
Date	*YYYY-MM-DD*	All items with a date equal to *YYYY-MM-DD*. Example: `2003-02-15`
Leading characters	*value**	All items with a date that starts with *value*. Example: `2002*` Selected: 2002-01-01, 2002-12-31 Not selected: 2001-12-31, 2003-01-01
Start value	*value*>	All items with a date greater than *value*. Example: `2002-05>` Selected: 2002-05-01, 2002-12-31, 2003-01-01, 2003-12-31 Not selected: 2002-04-31, 2001-12-31 **Special dates** can be used as *value* (see below).
End value	*value*<	All items with a date less than *value*. Example: `2003-02<` Selected: 2002-05-01, 2002-12-31, 2003-01-01, 2003-01-31 Not selected: 2003-02-01, 2003-05-18 **Special dates** can be used as *value* (see below).
Special Dates		
`TODAY` (+/- *nnnn*)		All items with the date of the current day. The day can be followed by +*nnnn* or -*nnnn* where *nnnn* has a maximum of 4 digits. The resulting date is computed as the date of the current day plus or minus *nnnn* days. Example: If the current date is `2003-03-01`, `TODAY +5` results in 2003-03-06.
`YESTERDAY`		All items with the date of the day before the current day.
`MONTH`		All items with the date range of the current month. Example: The current month is `2003-02`.

	Input Value	Items Selected
		Selected: 2003-02-01, 2003-02-30 Not selected: 2003-03-01
		`FMDATE`: Starts with the first day of the current month. `TODATE`: Ends with the last day of the current month. If the values of `FMDATE` and `TODATE` are identical, the selection is restricted to one day.
`YEAR`		All items with the date range of the current year. Example: The current year is `2003`. Selected: 2003-01-01, 2002-12-31 Not selected: 2002-31-12
		`FMDATE`: Starts with the first day of the current year. `TODATE`: Ends with the last day of the current year. If the values of `FMDATE` and `TODATE` are identical, the selection is restricted to one year.

Note: The parameter specification option **New Value** only allows leading characters (asterisk (*) notation). See ***Rules for New Values*** in *Set Global Parameters* in the section *Settings*.

Time

You can specify a time or a range of times. The time must be specified in the format *HH*:*II*:*SS* (*HH* = hours, *II* = minutes, *SS* = seconds).

In the list of options below, *value* is any combination of one or more digits:

	Input Value	Items Selected
Time	*HH*:*II*:*SS*	All items with a time equal to *HH*:*II*:*SS*. Example: `14:15:16`
Leading characters	*value**	All items with a time that starts with *value*. Example: `13:*` Selected: 13:00:00, 13:10:53, 13:59:59 Not selected: 12:59:59, 14:00:00

31 Work Files

This section describes work files and valid formats that apply to the unload, load and scan functions of the Object Handler.

See also ***Work File Options*** in the section *Settings*.

Note: Whenever the name of the work file exceeds the space available, choose PF11 and enter a longer name of up to 253 characters. Alternatively, position the cursor at the field **Work file** and choose PF1 (Help).

Work File Assignment

The following table lists the work files used by the Object Handler.

File	Explanation
Work File 1	Used for the unload, load and scan functions. Contains the data unloaded.
Work File 3	An internal report file.
Work File 4	Used when the option **Write report** (see ***Work File and Report Options*** in *Settings*) is set. **Write report** is the default setting for object processing. Contains report data.
Work File 5	The target file for the Adabas FDTs (Field Definition Tables) loaded.
Work File 6	Used for the load function if the option **Write restart information** (see *Set Additional Options* in *Settings*) is set. Contains restart information.
Work File 7	Only used if Entire Connection is installed and if **Use PC File** is selected on the **Options** screen (see also ***Set Additional Options***). Work File 7 must be defined as Entire Connection work file to be used for the unload, load and scan functions. Contains the data unloaded.
Work File 8	Only used if Entire Connection is installed and if **Use PC File** is selected on the **Options** screen (see also ***Set Additional Options***). Work File 8 must be defined as Entire Connection work file to be used for the unload, load and scan functions. Used as internal file for processing Entire Connection commands. **Note:**

File	Explanation
	The number of the work file can be changed with user exit routine **OBJHEX03** (see ***Batch Condition Codes and User Exit Routines***) or with the options `PCCOMMANDFILENUMBER`, `PCCOM` and `PCCFN` (see *`option-setting`*) when using a Workplan of the type PROCEDURE.
Work File 9	An internal work file.
Work File 10	The trace work file. Used when the trace mode is set. See `SET TRACE WORKFILE` in *Commands for Navigation and Special Functions* in *Direct Commands*.

Work File Format

There are two file formats for unloading objects in the source environment into work files and for loading them from work files into the target environment: an internal format and the Transfer format. Work files must be of internal format to transfer binary data. Work files must be of Transfer format to transfer text data.

This section covers the following topics:

- Internal Format
- Transfer Format

Internal Format

The internal format is an internal record layout for work files that are used to transfer Natural sources and cataloged objects, error messages, command processors, Adabas FDTs (Field Definition Tables) and non-Natural objects from one environment to another.

Use work files of internal format to transfer objects between identical platforms. Use portable work files of internal format if you want to transport objects between different UNIX, OpenVMS or Windows platforms, for example, from a little-endian machine to a big-endian machine. See also **Portable work file** in the sections *Settings*, *Portable Natural Generated Programs* (*Programming Guide*) and `DEFINE WORK FILE` (*Statements* documentation).

The Object Handler uses internal format by default. When using the internal format (**Transfer format** option not selected), Work File 1 must be of binary format. To achieve this, omit the file extension or use the file extension `.sag`.

With the internal format activated, Natural objects are read from the source environment and written to a Natural work file by using the unload function of the Object Handler. This work file can be transported to another environment with standard file transfer services. In the target environment, the objects can then be read from the work file and loaded into the local file or database system with the load function of the Object Handler.

Note: Work files created by the utility NATUNLD on the server, must be processed in internal format. The work files must be created on a server of the same platform where NATUNLD was applied.

Transfer Format

See also **Transfer format** in the section *Settings*.

The Transfer format is a general record layout for work files that contain load or unload data. This format is platform-independent and can be used to transfer the sources of Natural objects, Natural command processor sources, error messages and Adabas FDTs from one hardware platform to another and between UNIX, OpenVMS, mainframe and Windows platforms.

With the option **Transfer format** set, the unload function of the Object Handler reads Natural objects from a hardware platform and then restructures them.

Formatted records are written to a Natural work file that can be transported to another platform with standard file transfer services. On the target platform, the load function of the Object Handler then reads the objects from the work file and loads them into the local file or database system. The objects read from the work file are restructured according to the structure of the new hardware platform.

Specifying Work Files

If Transfer format is specified (option **Transfer format** set), Work File 1 must be of text (ASCII) format. To achieve this, a file extension must be used, but not the file extension `.sag`.

Handling Sources in Unicode/UTF-8

Transfer format is also used to unload or load sources of Natural objects in Unicode/UTF-8 (Universal Transformation Format, 8-bit form). If you specify the corresponding unload option (`WORKFILETYPE` set to `UTF-8` in command mode or **Unicode work file** in menu mode), all object sources will be unloaded into a work file in UTF-8. If you specify the corresponding load option (`LOAD-CODE-PAGE` in command mode or **Use load code page** in menu mode), all object sources in UTF-8 will be converted with the specified code page when they are loaded into a Natural system file.

Work Files from SYSTRANS

Use Transfer format to process work files created by the utility SYSTRANS. Work files that contain object sources encoded in UTF-8 cannot be processed with SYSTRANS.

32 Direct Commands

The Object Handler provides direct commands for the following purposes:

- To execute an Object Handler function such as unloading or loading objects in batch mode or in direct command online mode without using Object Handler menus (see also ***Batch or Direct Command Calls***).
- To execute or reference a Workplan (see also the section ***Workplans***).
- To be used as an instruction in a Workplan.
- To navigate through screens.
- To perform special functions.

This section describes the basic command syntax and the individual clauses, parameter and option settings available to perform these tasks. In addition, you can view examples that illustrate the use of direct commands.

The symbols used in the syntax diagrams shown in this section are explained in *System Command Syntax* in the *System Commands* documentation.

This section covers the following topics:

33 Basic Command Syntax

This section describes the Object Handler direct commands provided for executing Object Handler functions and Workplans of the type PROCEDURE. It also describes the commands used for migrating from the old utility SYSTRANS to the Object Handler.

For explanations of the variable values contained in the syntax diagrams shown in this section, refer to the relevant sections in the *Object Handler* documentation. For explanations of the symbols used in the syntax diagrams, see *System Command Syntax* in the *System Commands* documentation.

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

Executes a Workplan of the type PROCEDURE. See also the section ***Workplans***.

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

Unloads the objects defined in the `select-clause` with the parameters defined in `parameter-setting` with the options defined in `option-setting`.

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

Loads the objects defined in the `select-clause` with the parameters defined in `parameter-setting` with the options defined in `option-setting`.

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

Loads all objects from a work file with the parameters defined in `parameter-setting` with the options defined in `option-setting`.

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

Scans a work file for the objects defined in the `select-clause` with the options defined in `option-setting`.

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

Scans a work file for all objects with the options defined in `option-setting`.

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

Finds the objects defined in the `select-clause` with the options defined in `option-setting` and writes a report of the objects found into Work File 3. In addition, a report of the objects found can be written to Work File 4 or a specified report file.

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

Finds the libraries for Natural objects or Natural command processor sources defined in the `select-clause` with the options defined in `option-setting` and writes a report of the objects found into Work File 3. In addition, a report of the objects found can be written to Work File 4 or a specified report file.

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

Deletes the objects defined in the `select-clause` with the options defined in `option-setting`.
Restriction: It is not possible to delete an FDT.

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

Unloads delete instructions for the objects defined in the `select-clause` with the options defined in `option-setting`.

```
RESTART  [restart-file]
```

Continues an interrupted load function. This is only possible if information was written to a restart file during the aborted load. Restart load information can be written to Work File 6 or a specified restart file. See also `RESTART` in the section `option-setting` (*Direct Commands*) and ***Restart Load***.

```
DISPLAY STATISTICS
```

Displays statistics information about the objects processed.

```
NATUNLD natunld-direct-command
```

Executes an Object Handler command in the syntax of the old utility NATUNLD. See also ***Migration from NATUNLD/NATLOAD and SYSTRANS to the Object Handler***.

```
NATLOAD natload-direct-command
```

Executes an Object Handler direct command issued in the syntax of the old utility NATLOAD. See also ***Migration from NATUNLD/NATLOAD and SYSTRANS to the Object Handler***.

```
SYSTRANS systrans-direct-command
```

Executes an Object Handler direct command issued in the syntax of the old utility SYSTRANS. See also ***Migration from NATUNLD/NATLOAD and SYSTRANS to the Object Handler***.

34 select-clause

The `select-clause` comprises either a Workplan of the type SELECTION or LIST, or selection specifications for the objects, FDTs or applications to be processed.

This section describes the syntax that applies to the `select-clause`. The keywords and variable values contained in the syntax diagrams represent the parameters that can be used to specify object selection criteria. If indicated, a variable value must be supplied with a keyword.

Syntax of select-clause

The `select-clause` consists of one of the following options:

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

The `selection-workplan` and `list-workplan` options are explained in *SELECTION or LIST Workplan* below.

The use of `object-selection` depends on the object type, DDM, FDT or application you want to process, for each of which the appropriate syntax and keywords are explained in the remainder of this section.

The `delete-instruction-selection` options are explained in ***Delete Instructions for Selected Objects***.

SELECTION or LIST Workplan

A Workplan of the type SELECTION contains a header (`TYPE SELECTION`) and a selection from one of the following types of object or file: Natural library objects, Natural-related objects, Natural system error messages, Natural command processor sources, external files (external objects) or Adabas FDTs (Field Definition Tables).

A Workplan of the type LIST contains a header (`TYPE LIST`) and a selection list of objects as described in the section ***Object List - LIST Workplan***. Such an object list can be used for the `UNLOAD`, `LOAD` or `FIND` command only.

For further information on using Workplans, see the section ***Workplans***.

Natural Library Object and DDM Selection

This selection is used to select Natural objects for processing including Natural DDMs, user-defined error messages and shared resources.

The appropriate syntax is shown and explained in the following section.

- Syntax of Natural Library Object and DDM Selection

Syntax of Natural Library Object and DDM Selection

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

Note: For the command `FINDLIB`, only the following keywords are processed: `LIBRARY`, `DBID` and `FNR`.

Keyword Explanation of Natural Library Object and DDM Selection

The keywords and valid values for the objects to be processed are described in the following section.

Keyword	Valid Values	Default Value
object-name	A valid object name or a range of names. If *object-name* contains blank characters, it must be enclosed in double quotation marks (" "). See also ***Name*** in *Name, Date and Time Specification*.	none
`LIBRARY`	A valid library name or a range of names. If `SETNO` is specified, a range of names is not allowed. See also ***Name***.	none
`DBID`	A valid database ID.	`0`

Keyword	Valid Values	Default Value
		(current FNAT/FUSER)
`FNR`	A valid file number.	`0` (current FNAT/FUSER)
`OBJTYPE`	Types of object are: `E` User-defined error messages `N` Natural programming objects `R` Shared resources `*` Asterisk (all) or a valid combination.	`*`
`SETNO`	Only applies to the unload and find functions and if Predict is installed. Not applicable to application objects (see also ***Selecting Application Objects***). A one- or two-digit number that identifies the retained set to be used for the names of the objects to be processed. A retained set is created with the save set option of the `LIST XREF` command. If `SETNO` is specified, the value specified for *`object-name`* is ignored. For detailed information on Predict sets, refer to the *Predict* documentation.	none
`SETUSER`	Only applies to the unload and find functions and if Predict is installed. Not applicable to application objects (see also ***Selecting Application Objects***). The ID of the user who created the Predict set. If no ID is specified, the value of the system variable `*USER` (see also the *System Variables* documentation) is used.	`*USER`
`SETLIBRARY`	Only applies to the unload and find functions and if Predict is installed. Not applicable to application objects (see also ***Selecting Application Objects***). The name of the library to be searched for a Predict set. If you do not specify `SETLIBRARY`, the library specified with `LIBRARY` is used instead.	
`NATTYPE`	One or more single-character codes for Natural object types: `P` Program `N` Subprogram `S` Subroutine `C` Copycode	`*`

Keyword	Valid Values	Default Value
	H Helproutine T Text 7 Function 8 Adapter G Global data area L Local data area A Parameter data area M Map 4 Class 3 Dialog 5 Natural command processor V DDM * All object types	
SCKIND	The kind of Natural programming objects. Valid input values are: S Source objects: objects that are only stored in source form. C Cataloged objects: objects that are only stored in cataloged form. A All source and cataloged objects. W All STOWed objects: source and cataloged objects with identical date and time. B Source and cataloged objects if both exist. **Note:** W and B are valid for the UNLOAD and FIND commands only. For LOAD and SCAN, W and B are valid entries, but they are treated like A (all objects). If data is processed in Transfer format, only S (source objects) or A applies.	A
MODE	The programming mode of the Natural programming objects. Valid input values are: A Any. R All objects in reporting mode. S All objects in structured mode.	
FMNUM	A start number of Natural error messages. Valid range: 1 to 9999.	1
TONUM	An end number of Natural error messages. Valid range: 1 to 9999.	9999 or value of FMNUM

Keyword	Valid Values	Default Value
	The value must be greater than or equal to the value of FMNUM, if specified.	(if specified)
SLKIND	The kind of Natural error message text. Valid input values are: S Short text. Cannot be applied to the DELETE command (see ***Basic Command Syntax***). L Long text. A Short and/or long text. B Short and long text, if both exist.	A
LANGUAGE	Up to 8 valid language codes (for example, code 1 for English) of user-defined error messages. An asterisk (*) selects all language codes.	*
DDMDBID	The valid database ID (1 to 65535) of a DDM. UNLOAD, LOAD and SCAN: 0 denotes that no check is performed. DDMs are processed, regardless of their database ID (DBID).	0
DDMFNR	The valid file number (1 to 65535) of a DDM. UNLOAD, LOAD and SCAN: 0 denotes that no check is performed. DDMs are processed, regardless of their file number (FNR).	0
NATVERS	The Natural version of Natural programming objects. You can also specify a range of versions: see ***Name***.	blank (no check)
DATE	The save or catalog date of Natural programming objects, and the date of shared resources. You can add a time by inserting a blank between date and time. For the format and ranges allowed, see ***Date*** and ***Time*** in *Name, Date and Time Specification*. Special terms allowed are YESTERDAY and TODAY. See ***Special Dates*** in *Date*.	blank (no check)
FMDATE	A start value: The date on or after which Natural programming objects were cataloged or saved, and the date of shared resources. The format is identical to DATE. See ***Date***. Special terms allowed are YEAR, MONTH, YESTERDAY and TODAY. See ***Special Dates*** in *Date*.	blank (no check)
TODATE	An end value: The date on or before which Natural programming objects were cataloged or saved, and the date of shared resources. The format is identical to DATE. See ***Date***.	blank (no check) or high value (if FMDATE specified)

Keyword	Valid Values	Default Value
	Special terms allowed are YEAR, MONTH, YESTERDAY and TODAY. See ***Special Dates*** in *Date*.	
SIZE	The size of Natural programming objects and shared resources (up to 7 digits).	0 (no check)
FMSIZE	A start value: The minimum size of Natural programming objects and shared resources (up to 7 digits).	0 (no check)
TOSIZE	An end value: The maximum size of Natural programming objects and shared resources (up to 7 digits).	0 (no check) or high value (if FMSIZE specified)
USERID	The ID of the user who saved or cataloged the Natural programming objects. You can also specify a range of user IDs: see ***Name***.	blank (no check)
TID	The ID of the terminal where the Natural programming objects were saved or cataloged (provided by the Natural system variable *INIT-ID). You can also specify a range of terminal IDs: see also ***Name***.	blank (no check)
EXCEPT	All items that match the selection criteria entered before EXCEPT are checked against *all* parameters contained within the parentheses following the keyword EXCEPT. If they match all these parameters too, they are not processed.	not applicable

Notes:

1. Parameters that are irrelevant for OBJTYPE are ignored. For example: DATE, SIZE and USERID have no meaning for Natural error messages.
2. DBID and FNR are ignored by the LOAD or SCAN command. These parameters must instead be specified in the *parameter-setting* clause as described for LOADFNAT... and LOADFUSER... in ***Keyword Explanation of parameter-clause***.
3. If an object for shared resources contains blank characters, it must be enclosed in double quotation marks (" ").

Natural-Related Object Selection

This selection is used to select Natural-related objects for processing.

The appropriate syntax is shown and explained in the following section.

- Syntax of Natural-Related Object Selection

Syntax of Natural-Related Object Selection

```
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]
         )]
```

Keyword Explanation of Natural-Related Object Selection

The keywords and valid input values for the objects to be processed are described in the following section.

Keyword	Valid Values	Default Value
object-name	The name of a Natural-related object. If *object-name* contains blank characters, it must be enclosed in double quotation marks (" "). See also ***Name*** in *Name, Date and Time Specification*.	none
NATPATH	NATROOT NATGUI_BMP TMP_PATH NATBIN PROFILE_PATH PARM_PATH NATERR	none

Keyword	Valid Values	Default Value
`DATE`	The modification date of Natural-related objects. You can add a time by inserting a blank between date and time. For the format and ranges allowed, see ***Date*** and ***Time*** in *Name, Date and Time Specification*. Special terms allowed are: `YESTERDAY` and `TODAY`. See ***Special Dates*** in *Date*.	blank (no check)
`FMDATE`	A start value: The date on or after which Natural-related objects were modified. The format is identical to `DATE`. See ***Date***. Special terms allowed are: `YEAR`, `MONTH`, `YESTERDAY` and `TODAY`. See ***Special Dates*** in *Date*.	blank (no check)
`TODATE`	An end value: The date on or before which Natural-related objects were modified. The format is identical to `DATE`. See ***Date***. Special terms allowed are: `YEAR`, `MONTH`, `YESTERDAY` and `TODAY`. See ***Special Dates*** in *Date*.	blank (no check) or high value (if `FMDATE` specified)
`SIZE`	The size of Natural-related objects (up to 10 digits).	`0` (no check)
`FMSIZE`	A start value: The minimum size of Natural-related objects (up to 10 digits).	`0` (no check)
`TOSIZE`	An end value: The maximum size of Natural-related objects (up to 10 digits).	`0` (no check) or high value (if `FMSIZE` specified)
`EXCEPT`	See `EXCEPT` in *Natural Library Object and DDM Selection*.	

Note: The `NATPATH` clause in the `EXCEPT` part is evaluated by the `LOAD` or `SCAN` command only.

Natural System Error Message Selection

This selection is used to select Natural system error messages for processing.

The appropriate syntax is shown and explained in the following section.

Syntax of Natural System Error Message Selection

```
ERROR NATERROR
[DBID dbid FNR fnr]
[FMNUM error-number-from] [TONUM error-number-to]
[SLKIND message-type]
[LANGUAGE languages]
[ EXCEPT
                                        (
                                        [FMNUM error-number-from] [TONUM
                                        error-number-to]
                                        [SLKIND message-type]
                                        [LANGUAGE languages]
                                        ) ]
```

Keyword Explanation of Natural System Error Message Selection

The keywords and valid input values for the Natural system error messages to be processed are described in the following section.

Keyword	Valid Values	Default Value
DBID	Only applies to system error messages on mainframes. A valid database ID.	0 (current FNAT)
FNR	Only applies to system error messages on mainframes. A valid file number.	0 (current FNAT)
FMNUM	A start number of system error messages. Valid range: 1 to 9999.	1
TONUM	An end number of system error messages. Valid range: 1 to 9999. The value must be greater than or equal to the value of FMNUM if specified.	9999 or value of FMNUM (if specified)
SLKIND	See SLKIND in *Natural Library Object and DDM Selection*.	A
LANGUAGE	Up to 8 valid language codes (for example, code 1 for English) of system error messages. An asterisk (*) selects all language codes.	*
EXCEPT	See EXCEPT in *Natural Library Object and DDM Selection*.	

Note: DBID and FNR are ignored by the LOAD or SCAN command. These parameters must instead be specified in the *parameter-setting* clause as described for LOADFNAT... in ***Keyword Explanation of** parameter-clause*.

Natural Command Processor Selection

This selection is used to select Natural command processor sources for processing.

The appropriate syntax is shown and explained in the following section.

- Syntax of Natural Command Processor Source Selection

Syntax of Natural Command Processor Source Selection

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

file-options

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

Note: For the command FINDLIB, only the following keywords are processed: PROCESSOR, DBID, FNR, CIPHER and PASSWORD or PSW.

Keyword Explanation of Natural Command Processor Source Selection

The keywords and valid input values for the Natural command processor sources to be processed are described in the following section.

Keyword	Valid Values	Default Value
object-name	The name of a valid Natural command processor source or a range of names. See also ***Name*** in *Name, Date and Time Specification*.	none
PROCESSOR	A valid library name or a range of names. See also ***Name***.	none
DBID	The valid database ID of the Adabas file where the Natural command processor sources are stored.	Value of LFILE 190
FNR	The valid file number of the Adabas file where the Natural command processor sources are stored.	Value of LFILE 190
CIPHER	The 8-digit cipher code of the Adabas file where the Natural command processor sources are stored.	blank
PASSWORD or PSW	The 8-character Adabas password of the Adabas file where the Natural command processor sources are stored.	blank
EXCEPT	See EXCEPT in *Natural Library Object and DDM Selection*.	

Note: DBID, FNR, CIPHER and PASSWORD or PSW are ignored by the LOAD or SCAN command. These parameters must instead be specified in the *parameter-setting* clause as described for LOADNCP... in ***Keyword Explanation of*** *parameter-clause*.

External File Selection

This selection is used to select external files (external objects) for processing.

The appropriate syntax is shown and explained in the following section.

- Syntax of External File Selection

Syntax of External File Selection

```
external-file-name PATH external-path-name
[        DATE date                                                    ]
         [FMDATE date-from] [TODATE date-to]
[        SIZE size                                                    ]
         [FMSIZE size-from] [TOSIZE size-to]
[ EXCEPT
         (external-file-name [PATH external-path-name]
 [      DATE date                                      ]
        [FMDATE date-from] [TODATE date-to]
 [      SIZE size                                      ]
        [FMSIZE size-from] [TOSIZE size-to]
 ) ]
```

Keyword Explanation of External File Selection

The keywords and valid input values for the external files to be processed are described in the following section.

Keyword	Valid Values	Default Value
`external-file-name`	The name of an external file. If `external-file-name` contains blank characters, it must be enclosed in double quotation marks (" "). See also ***Name*** in *Name, Date and Time Specification*.	none
PATH	The name of the path where the external file is located.	none
DATE	The modification date of external files. You can add a time by inserting a blank between date and time. For the format and ranges allowed, see ***Date*** and ***Time*** in *Name, Date and Time Specification*. Special terms allowed are YESTERDAY and TODAY. See ***Special Dates*** in *Date*.	blank (no check)
FMDATE	A start value: The date on or after which external files were modified. The format is identical to DATE. See ***Date***. Special terms allowed are YEAR, MONTH, YESTERDAY and TODAY. See ***Special Dates***.	blank (no check)
TODATE	An end value: The date on or before which external files were modified. The format is identical to DATE. See ***Date***. Special terms allowed are YEAR, MONTH, YESTERDAY and TODAY. See ***Special Dates***.	blank (no check) or high value (if FMDATE specified)
SIZE	The size of external files (up to 10 digits).	0 (no check)
FMSIZE	A start value: the minimum size of external files (up to 10 digits).	0 (no check)
TOSIZE	An end value: The maximum size of external files (up to 10 digits).	0 (no check)

Keyword	Valid Values	Default Value
		or high value (if FMSIZE specified)
EXCEPT	See EXCEPT in *Natural Library Object and DDM Selection*.	

Note: The NATPATH clause in the EXCEPT part is only evaluated by the LOAD and SCAN commands.

FDT Selection

This selection is used to select Adabas FDTs (Field Definition Tables) for processing.

For loading FDTs, see also ***FDTs*** in the section *Object Specification*.

The appropriate syntax is shown and explained in the following section.

- Syntax of FDT Selection

Syntax of FDT Selection

```
FDT
DBID dbid
⎧ FNR fnr [CIPHER cipher] [ ⎧ PASSWORD ⎫  password ] ⎫
⎨                           ⎩ PSW      ⎭             ⎬
⎩ FMFNR fnr-start TOFNR fnr-end                        ⎭
```

Keyword Explanation of FDT Selection

The keywords and valid input values for the FDTs to be processed are described in the following section.

Keyword	Valid Values	Default Value
DBID	The database ID of the FDT.	none
FNR	The file number of the FDT.	none
CIPHER	The 8-digit Adabas cipher code of the FDT.	none
PASSWORD or PSW	The 8-character Adabas password of the FDT.	none
FMFNR	Only applies to the FIND or UNLOAD command. A start value:	none

Keyword	Valid Values	Default Value
	The file number (FNR) of an FDT.	
TOFNR	Only applies to the FIND or UNLOAD command. An end value: The file number (FNR) of an FDT.	none

Application Selection

This selection applies to applications created and maintained in Natural Studio's application workspace and the libraries or objects that belong to these applications.

The appropriate syntax is shown and explained in the following section.

Selecting Base and Compound Applications

This selection only applies to the find function.

Syntax

```
APPLICATION APNAME application-name
[APTYPE application-type]
[COMPAPPLICATION compound-application-name]
[                    EXCEPT
                     (APNAME application-name
                     [APTYPE application-type]
                     ) ]
```

Selecting Application Libraries

This selection only applies to the find function.

Syntax

```
APPLICATION  APLIBRARY application-library-name
[BASEAPPLICATION base-application-name]
[ COMPAPPLICATION compound-application-name]
[            DBID dbid[FNR fnr]  ]
[            EXCEPT
             (APLIBRARY application-library-name
             [BASEAPPLICATION base-application-name]
             ) ]
```

Selecting Application Objects

This selection only applies to the find and unload functions.

Syntax

```
APPLICATION  APOBJECTS application-object-name
[BASEAPPLICATION base-application-name]
[COMPAPPLICATION compound-application-name]
[LIBRARY library-name]
[object-specification]
[                        EXCEPT
                         (APOBJECT application-object-name
                         [LIBRARY library-name]
                         [BASEAPPLICATION base-application-name]
                         [object-specification]
                         ) ]
```

Keyword Explanation of Application Selection

The keywords and valid input values for the applications, application libraries or application objects to be processed are described in the following section.

Keyword	Valid Values	Default Value
`APNAME`	A valid name of a Natural application or a range of names. See also ***Name*** in *Name, Date and Time Specification*.	*
`APTYPE`	A valid application type: | | | |---|---| | `B` | Base application | | `O` | Compound application | | `*` | All: base and/or compound applications |	*
`COMPAPPLICATION`	Only applies if `APTYPE` is set to `*` or `B`. The name of a compound application to which the specified base application belongs or a range of names. Only base applications that belong to the specified compound application(s) are selected; base applications that do not belong to a compound application are not selected.	none
`EXCEPT`	See `EXCEPT` in *Natural Library Object and DDM Selection*.	not applicable
`APLIBRARY`	The valid name of a library that belongs to a Natural base or compound application or a range of names. See also ***Name*** in *Name, Date and Time Specification*.	*
`BASEAPPLICATION`	The valid name of a Natural base application to which an application library or application object belongs. See also ***Name*** in *Name, Date and Time Specification*.	*
`DBID`	The valid database ID of an application library.	`0` (no check)
`FNR`	The valid file number of an application library.	`0` (no check)
`APOBJECT`	The valid name of an application object that belongs to a base or compound application, or a range of names. See also ***Name*** in *Name, Date and Time Specification*.	*
`LIBRARY`	A valid library name or a range of names. If `OBJTYPE` is set to `D` (see *Natural Library Object and DDM Selection*), the library name is ignored. See also ***Name*** in *Name, Date and Time Specification*.	*
`object-specification`	Indicates that additional selection criteria can be specified for application objects as shown in the **syntax** diagram for Natural library objects and DDMs: all items listed below `LIBRARY` *`library-name`* can also be applied to application objects whereas	not applicable

Keyword	Valid Values	Default Value
	`object-name` in the `EXCEPT` clause is irrelevant for application objects.	

Object Selection for Delete Instructions

This selection is used to specify delete instructions for Natural library objects, DDMs, user-defined error messages and Natural system error messages. The delete instructions are executed when a work file of internal format is loaded in the target environment with the `DELETEALLOWED` option specified.

The appropriate syntax is shown and explained in the following section.

- Syntax of Delete Instructions for Natural Library Objects and DDMs
- Syntax of Delete Instructions for User-Defined Error Messages
- Syntax of Delete Instructions for Natural System Error Messages

Syntax of Delete Instructions for Natural Library Objects and DDMs

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

Keyword Explanation of Delete Instructions for Natural Library Objects and DDMs

The keywords and valid values for the objects to be processed are described in the following section.

Keyword	Valid Values	Default Value
`object-name`	A valid object name or a start value (`value*`) for a range of names such as `ABC*`.	none
`LIBRARY`	A valid library name. A range specification is *not* allowed.	none
`OBJTYPE`	A valid object-type code: `N` Natural programming objects	*
`NATTYPE`	A Natural object type. Valid input values are:	*

Keyword	Valid Values	Default Value
	`*` All object types `V` DDMs	
SCKIND	The kind of Natural programming objects. Valid input values are: `S` Source objects. If used in the `except-clause` (see *Syntax of Natural Library Object and DDM Selection*): objects that are stored only in source form. `C` Cataloged objects. If used in the `except-clause`: objects that are stored only in cataloged form. `A` All source and cataloged objects.	A

Syntax of Delete Instructions for User-Defined Error Messages

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

library-name denotes the name of a single library; a range specification is not allowed.

For explanations of the other elements used in this syntax, see ***Keyword Explanation of Natural Library Object and DDM Selection***.

Syntax of Delete Instructions for Natural System Error Messages

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

For explanations of the elements used in this syntax, see ***Keyword Explanation of Natural System Error Message Selection***.

35 Object List - LIST Workplan

An object list is a Workplan of the type LIST, which specifies object selection criteria for the objects to be processed in the `UNLOAD`, `LOAD`, `FIND` or `DELETE` command. An object list can be used as an alternative to the *`select-clause`* and the SELECTION Workplan.

The following syntax applies to an object list:

```
TYPE LIST

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

The syntactical options are explained in the following section. The keywords and variable values contained in the syntax diagrams shown in this section represent parameters that are used to specify object selection criteria. If indicated, a variable value must be supplied with a keyword. Each syntax element (except for the ones enclosed in parentheses) must start on a new line and end on the same line.

For explanations of the keywords contained in the syntax diagrams, refer to the section *`select-clause`*.

Syntax of object-type-and-location

The syntax diagrams that apply to *`object-type-and-location`* are shown in the following section.

- Natural Objects and DDMs
- Natural System Error Messages
- Natural Command Processor Sources
- Natural-Related Objects
- External Files (External Objects)
- FDTs

Natural Objects and DDMs

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

Note: No ranges are allowed for *`library-name`*.

Natural System Error Messages

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

Natural Command Processor Sources

```
PROCESSOR ncp-library-name
  [ DBID dbid FNR fnr [CIPHER cipher] [ { PASSWORD } password ] ]
                                         { PSW      }
```

Note: No ranges are allowed for *ncp-library-name*.

Natural-Related Objects

```
NATPATH natural-path-name
```

External Files (External Objects)

```
PATH external-path-name
```

FDTs

```
FDT
```

Syntax of object-name-description

The syntax diagrams that apply to *object-name-description* are shown in the following section:

- Natural Objects
- Natural System Error Messages
- Natural Command Processor Sources
- Natural-Related Objects
- External Files (External Objects)

- FDTs

Natural Objects

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

Natural System Error Messages

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

Natural Command Processor Sources

```
object-name
```

Natural-Related Objects

```
related-object-name
```

External Files (External Objects)

```
external-file-name
```

FDTs

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

Example of an Object List

The follwing is an example of a Workplan of the type LIST:

```
TYPE LIST
  LIBRARY LIB-1 OBJTYPE N    /* process Natural objects from library 'LIB-1'
   ( A* SCKIND S             /* all sources objects whose names start with 'A'
  B1                         /* source and/or cataloged object of  'B1'
  CDE> SCKIND C )          /* all cataloged objects with names greater than/equal ↩
to 'CDE'
  /*                         /* comment line
  LIBRARY LIB-2              /* process Natural objects from library 'LIB-2'
                             /* including error messages and shared resources
  ( *                        /* all source and/or cataloged objects
                             /* including shared resources
  FMNUM 1 TONUM 100          /* error messages from 1 to 100
  )
```

36 parameter-setting

The *parameter-setting* clause is used to change attributes for the LOAD or UNLOAD command for the objects to be processed and to define target destinations for the LOAD command (for example, FNAT).

The following syntax applies to the *parameter-setting* clause:

```
WITH
⎧  (parameter-workplan)  ⎫
⎨                        ⎬
⎩  parameter-clause      ⎭
```

For an explanation of the syntax that applies to *parameter-workplan*, refer to ***Referencing Workplans*** in the section *Workplans*.

This section covers the following topics:

Syntax of parameter-clause

The syntax of the *parameter-clause* is shown in the following diagram. If indicated, a variable value must be supplied with a keyword.

```
[        [NAME old-name] NEWNAME                 ]
         new-name
[        [LIBRARY old-library-name]              ]
         NEWLIBRARY new-library-name
[        LOADFNATDBID fnat-dbid LOADFNATFNR fnat-fnr                                  ]
[        LOADFUSERDBID fuser-dbid LOADFUSERFNR fuser-fnr                              ]
[        LOADNCPDBID ncp-file-dbid LOADNCPFNR ncp-file-fnr                            ]
         [LOADNCPCIPHER ncp-file-cipher]
   [ {              LOADNCPPASSWORD }    ncp-file-password                       ]
                    LOADNCPPSW      }
[        [FDTDBID old-fdt-dbid FDTFNR old-fdt-fnr] NEWFDTDBID new-fdt-dbid NEWFDTFNR  ]
         new-fdt-fnr
[ERRNUMDIFF modification-of-error-message-range]
[        [LANGUAGE old-language]                 ]
         NEWLANGUAGE new-language
[        [DATE old-date] NEWDATE                 ]
         new-date
```

```
[  [USERID old-userid]  NEWUSERID  ]
   new-userid
[  [TID old-terminal-id] NEWTID    ]
   new-terminal-id
   [PATH old-external-path-name]
[  NEWPATH                         ]
   new-external-path-name
```

Keyword Explanation of parameter-clause

The keywords and variable values (if relevant) of the `parameter-clause` are explained in the following section.

Keyword	Values	Restricted to Command
NAME	The object name to be checked if NEWNAME is specified.	
NEWNAME	A new object name. **Note:** Not applicable to DDMs on mainframe platforms.	
LIBRARY	The library name to be checked if NEWLIBRARY is specified.	
NEWLIBRARY	A new library name. **Note for the LOAD function:** NEWLIBRARY does *not* affect the library name used in the delete instruction of a work file that is processed with the DELETEALLOWED option.	
LOADFNATDBID	The database ID (DBID) of FNAT libraries.	LOAD
LOADFNATFNR	The file number (FNR) of FNAT libraries.	LOAD
LOADFUSERDBID	The DBID of FUSER libraries.	LOAD
LOADFUSERFNR	The FNR of FUSER libraries.	LOAD
LOADNCPDBID	The DBID of the Adabas file for Natural command processor sources.	LOAD
LOADNCPFNR	The FNR of the Adabas file for Natural command processor sources.	LOAD
LOADNCPCIPHER	The cipher code of the Adabas file for Natural command processor sources.	LOAD
LOADNCPPASSWORD or LOADNCPPSW	Only applies to objects on mainframes. The Adabas password of the Adabas file for Natural command processor sources.	LOAD
FDTDBID	The DBID of the Adabas FDT (Field Definition Table) to be checked if NEWFDTDBID is specified.	
NEWFDTDBID	A new DBID of the FDT.	
FDTFNR	The DBID of the FDT to be checked if NEWFDTFNR is specified.	

Keyword	Values	Restricted to Command
NEWFDTFNR	A new FNR of the FDT.	
ERRNUMDIFF	A number (positive or negative) that is to be added to the Natural error messages during the UNLOAD or LOAD command. ERRNUMDIFF can only be specified if FMNUM and TONUM (see *select-clause*) have been specified as selection criteria. Otherwise, it is not possible to check for valid results.	
LANGUAGE	Up to 8 valid language codes (for example, code 1 for English) of Natural error messages to be checked if NEWLANGUAGE (see below) is specified. If *language* contains more than one language code, *new-language* must contain the same numbers of language codes. Each *language* language code is replaced by the language code in the corresponding position of *new-language*. If *language* is not specified, *new-language* must not contain more than one language code.	
NEWLANGUAGE	Up to 8 valid language codes (for example, code 4 for Spanish) for new user-defined error messages. This option does not apply to the long texts of Natural system error messages for which English (language code 1) is the only valid language. See also LANGUAGE above.	
DATE	An object date. You can add a time by inserting a blank between date and time. For the format and ranges allowed, see ***Date*** and ***Time*** in *Name, Date and Time Specification*.	
NEWDATE	A new object date. NEWDATE can be a date followed by a time value. You can add a time by inserting a blank between date and time. See also ***Date*** and ***Time*** in *Name, Date and Time Specification*.	
USERID	The user ID to be checked if NEWUSERID is specified.	
NEWUSERID	A new user ID.	
TID	Only applies to objects on mainframes. The terminal ID to be checked if NEWTID is specified.	
NEWTID	Only applies to objects on mainframes. A new terminal ID.	
PATH	The path name to be checked if NEWPATH is specified.	
NEWPATH	A new path name.	

Notes:

1. Parameters not applicable to the selection criterion processed are ignored.
2. `LOADFNAT...`, `LOADFUSER...` and `LOADNCP...` are used for the `LOAD` command only, and ignored otherwise.
3. `LOADFNAT...` is used for libraries starting with `SYS` (except SYSTEM).
4. `LOADFUSER...` is used for libraries not starting with `SYS` (but including SYSTEM).
5. `LOADNCP...` is used for Natural command processor sources.

37 option-setting

The `option-setting` clause is used to change the default values of Object Handler command options.

The syntax that applies to the `option-setting` clause is shown and explained in the following section. The keywords and variable values contained in the syntax diagrams shown represent the parameters that are used to specify the default values. If indicated, a variable value must be supplied with a keyword.

Syntax of option-setting

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

The syntax diagram that applies to `option-workplan` is shown and described in ***Referencing Workplans*** in the section *Workplans*.

The syntax of the `option-clause` is shown in the following section.

- Syntax of option-clause

Syntax of option-clause

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

[ transfer-options                   ]
[ internal-format-options            ]

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

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

[                  {         S       }         ]
[ REPORT-FORMAT    {         T       }         ]
[ REPORT-MODE          { S }       ]
                       { L }

[ NORESTART                    ]
[ RESTART [restart-file]       ]

[NUMBERPROCESS number]

[FIXEDLENGTH]

[FDIC (dbid,fnr,password,cipher)]

[FSEC (dbid,fnr,password,cipher)]

[ { WORKFILETYPE }   { DEFAULT } ]
[ { WFTYPE       }   { UTF-8   } ]

[ { PC    }   [file-name]        ]
[ { NEWPC }                      ]

[ { PCCOMMANDFILENUMBER }                       ]
[ { PCCFN               }   command-file-number ]

[ USE-FDDM { YES }       ]
           { NO  }

[ { NEWWORKFILE }   file-name [ { WORKFILETYPE } { DEFAULT  } ] ]
[ { WORKFILE    }             [ { WFTYPE       } { PORTABLE } ] ]
                                                 { UTF-8    }

[ADAFDUWORKFILE file-name]
```

Separators

Commas must be used as separators between the values following the `FDIC` and `FSEC` keywords, or if a value is missing. For example: `FDIC (10,21,,2a)`.

If the session parameter `ID` (see *ID - Input Delimiter Character* in the *Parameter Reference* documentation) has been set to a comma, use a slash (/) as the separator between values.

transfer-options

```
TRANSFER

[   CONVERSION-TABLE { SYSTEM-TABLE          }  ]
                     { USER-TABLE            }
                     { [conversion-program]  }

[SUBSTITUTE]

[INCLUDE-LINE-NUMBERS]

[UPPERCASE-TRANSLATION]

[INCORPORATE-FREE-RULES]
```

```
[LOAD-CODE-PAGE code-page-name]
[DA-FORMAT data-area-format]
```

internal-format-options

```
[       ⎧ ON      ⎫ ]
[       ⎪ OFF     ⎪ ]
[ XREF  ⎨ DOC     ⎬ ]
[       ⎪ FORCE   ⎪ ]
[       ⎩ SPECIAL ⎭ ]
[DELETEALLOWED]
```

Keyword Explanation of option-setting

The keywords and the variable values (if relevant) of *option-setting* are explained in the following section:

Option	Explanation	Restricted to Command
REPLACE	Replaces existing objects according to the option specified: ALL — All objects (default setting). OBSOLETE — All objects with a date older than the date of the object in the load file. EXCEPT — All objects except those with a date newer than the date of the object in the load file.	LOAD LOADALL
TRANSFER	Set Transfer mode. The data is read and written in Transfer format. For valid options, see ***Keyword Explanation of*** *transfer-options*.	UNLOAD LOAD SCAN
NOREPORT	Specifies the report file setting: No data is recorded to a report file. This is the default setting for the FIND and FINDLIB commands.	
NEWREPORT	Specifies the report file setting: Report data is recorded and written to Work File 4 or *file-name*. An existing file will be overwritten.	

Option	Explanation	Restricted to Command
REPORT	Specifies the report file setting: Report data is recorded and written to Work File 4 or *file-name*. This is the default setting for the commands UNLOAD, LOAD, LOADALL, SCAN, SCANALL and DELETE.	
BATCHREPORT	Specifies the report setting for batch processing or when using the OBJHAPI Application Programming Interface: Report data is either written to SYSOUT or output on the screen respectively (report data is *not* written to a file).	
REPORT-OPTION-1 or REPOPT1 or REP-OPT-1	Specifies the report option to be used when a direct command is executed and a report is to be written: A Display all report items (default). E Display only error messages. This includes messages from Natural Security and messages that have incurred during the execution of a LOAD command, for instance "not replaced". S For batch mode only. The error report is split into two parts: Report items except error messages are written to the default report device (REPORT(0)/CMPRINT), error messages (including messages from Natural Security and messages that have incurred during the execution of a LOAD command) are written to the second report device (REPORT(1)/CMPRT01). Note that in online mode S has the same effect as A.	UNLOAD LOAD SCAN DELETE
REPORT-FORMAT or REPFRMT	Specifies the report format to be used when a direct command is executed and a report is to be written: S Display the source data in the unload report, i.e. the data of the unloaded object before parameters (e.g. the library name) are changed (default setting). T Display the target data in the unload report, i.e. the data after parameters (e.g. the library name) have been changed.	UNLOAD
REPORT-MODE or REPM	Specifies the report mode to be used when a direct command is executed and a report is to be written: S Write a short report, i.e. the most relevant data is displayed on the first 80 columns of the report line with short delimiters. L Write a large report, i.e. the data is displayed in the original order with large delimiters (default setting).	
NORESTART	No restart information is written to a file.	LOAD
RESTART	Restart information is written to Work File 6 or *restart-file*.	LOAD

Option	Explanation	Restricted to Command
NUMBERPROCESS	Specifies the number of objects to be processed. The LOAD or SCAN command stops execution after the number specified.	LOAD SCAN
FIXEDLENGTH	Sets the format of the unload work file to a maximum record length of fixed size. Every data record contains 256 bytes if written in internal format, or 100 bytes in Transfer format.	UNLOAD
FDIC	Specifies the system file FDIC to be used for processing: the database ID (*dbid*), file number (*fnr*), password (*password*) and cipher code (*cipher*) of the Adabas file. If no values (or 0) are specified, the current FDIC system file is used.	UNLOAD LOAD DELETE
FSEC	Specifies the system file FSEC to be used for processing: the database ID (*dbid*), file number (*fnr*), password (*password*) and cipher code (*cipher*) of the Adabas file. If no values (or 0) are specified, the current FSEC system file is used.	UNLOAD LOAD DELETE
USE-FDDM	Specifies that the FDDM system file is used for processing: see ***Keyword Explanation of USE-FDDM*** below.	UNLOAD LOAD FIND DELETE
NEWWORKFILE or WORKFILE	Specifies the work file to be used. The UNLOAD or LOAD data is transferred into/from Natural Work File 1. If NEWWORKFILE is specified, the data overwrites the contents of the existing work file or fills a new work file from the top. Otherwise, the data is appended.	UNLOAD LOAD SCAN
WORKFILETYPE or WFTYPE	Not required by the LOAD and SCAN commands, which automatically choose the appropriate work file type and ignore this keyword if specified. The work file type of Natural Work File 1 when data is read and written in internal format: DEFAULT — Default binary work file. PORTABLE — Portable work file. UTF-8 — Unicode/UTF-8 encoded binary work file. UTF-8 only applies to the unload function and if TRANSFER is specified.	UNLOAD LOAD SCAN

Option	Explanation	Restricted to Command
	If `UTF-8` is specified, you cannot use the options `CONVERSION-TABLE`, `SUBSTITUTE` and `INCORPORATE-FREE-RULES`. (See also ***Work File Format*** in *Work Files*.) If `WORKFILETYPE` has not been specified, the current type is used.	
`ADAFDUWORKFILE`	The complete path name assigned to the work file (Natural Work File 5) into which Adabas FDT data is loaded.	`LOAD`

`PC`
`NEWPC`
	Only applies if Entire Connection is installed.
	
	Writes data to or reads data from an Entire Connection work file. *`file-name`* denotes the complete path name assigned to the Entire Connection work file. If your system environment does not accept a backslash (\) separator, use a slash (/) instead. If you do not specify *`file-name`*, Entire Connection prompts you for the name of a work file.
	
	If `NEWPC` is specified, the data unloaded overwrites the contents of the existing work file or fills a new work file from the top. Otherwise, the data is appended.
	
	See also ***Work File Assignment*** in *Work Files*.
		`UNLOAD`
		`LOAD`
		`SCAN`

`PCCOMMANDFILENUMBER`
or
`PCCOM`
or
`PCCFN`
	Only applies if Entire Connection is installed.
	
	Specifies the number of the work file that is used for processing Entire Connection commands.
	
	The default value is `8` for Work File 8, which must be defined as Entire Connection work file.
	
	See also ***Work File Assignment*** in *Work Files*.
		`UNLOAD`
		`LOAD`
		`SCAN`

The keywords and the variable values (if relevant) of *`transfer-options`* and *`internal-format-options`* are explained in the following section:

- Keyword Explanation of transfer-options
- Keyword Explanation of internal-format-options

- Keyword Explanation of USE-FDDM

Keyword Explanation of transfer-options

When using the `TRANSFER` keyword, you can specify the following options:

Option	Explanation	Restricted to Command

`CONVERSION-TABLE`	Converts data processed in Transfer format by using either of the following conversion tables:
	
	`SYSTEM-TABLE`:
	
	The internal Natural conversion table.
	
	`USER-TABLE`:
	
	A user-defined conversion table if *`conversion-program`* has been specified. This program must be stored in the library SYSOBJH or one of its steplibs; see the example programs OTNCONAE and OTNCONEA in the library SYSOBJH.
	
	If no *`conversion-program`* is specified, the corresponding conversion table is used in NATCONV.INI ([ISO8859_1->EBCDIC] or [EBCDIC->ISO8859_1]).
		`UNLOAD`
		`LOAD`
		`SCAN`

`SUBSTITUTE`	Replaces line references by labels during the unload in Transfer format. This option only applies if your source-code line numbers are used for statement references. If so, the line numbers of referenced lines and the line number references are replaced by labels. The sources are not modified in the database.	`UNLOAD`
`INCLUDE-LINE-NUMBERS`	Transfers line numbers during the unload in Transfer format. By default, line numbers in Natural objects are *not* unloaded.	`UNLOAD`

`USE-LINE-NUMBER-INCREMENT`
or
`USE-LNI`
	`UNLOAD` If the option `INCLUDE-LINE-NUMBERS` is not specified, the line number increment of Natural source objects will be unloaded. By default, the line number increment in Natural source objects is *not* unloaded.
	
	`LOAD` If the line number increment was transferred, it is used to rebuild the line numbers of the Natural source objects.
		`UNLOAD`
		`LOAD`

`UPPERCASE-TRANSLATION`	Translates any source code into upper case during the load in Transfer format. By default, source code in Natural objects is *not* translated.	`LOAD`

Option	Explanation	Restricted to Command
INCORPORATE-FREE-RULES	Incorporates source text of Predict free rules associated with a map into a map source during the unload in Transfer format if Predict is installed.	UNLOAD
LOAD-CODE-PAGE	Specifies the code page to be used for converting object sources encoded in Unicode/UTF-8 (Universal Transformation Format, 8-bit form). If you use this option, all object sources unloaded into a work file in UTF-8, will be converted with the specified code page when they are loaded into a work file. If you specify *CODEPAGE as *code-page-name*, the value assigned to the system variable *CODEPAGE is used (see the *System Variables* documentation). If *code-page-name* is not specified, the source objects are converted with the code page used when unloading them. If LOAD-CODE-PAGE is specified, you cannot use the options CONVERSION-TABLE and UPPERCASE-TRANSLATION.	LOAD LOADALL
DA-FORMAT	Specifies format conversion of data area sources: see **Data area format** in *Transfer Options* in *Settings*.	UNLOAD LOAD

Keyword Explanation of internal-format-options

When using *internal-format-options*, you can specify the following:

Option	Explanation	Restricted to Command
XREF	Only applies if Predict is installed. Loads or unloads XRef data of cataloged Natural objects. You can specify one of the following values: ON: UNLOAD: Unloads cataloged objects and their cross-reference data (if any). LOAD: Loads cataloged objects and their cross-reference data if cross-references exist in the work file. OFF: No XRef data is processed. This is the default.	LOAD UNLOAD

Option	Explanation		Restricted to Command
	DOC	Only applies to LOAD. Loads cataloged objects and their cross-reference data (if any) only if Predict entries exist for the objects in the FDIC system file.	
	FORCE	Only applies to LOAD. Loads cataloged objects and their cross-reference data only if cross-references exist in the work file and if Predict entries exist for the objects in the FDIC system file.	
	SPECIAL	Only applies to LOAD. Loads cataloged objects and their cross-reference data (if any).	
DELETEALLOWED	Processes delete instructions from work files when loading objects in internal format.		LOAD

Keyword Explanation of USE-FDDM

Only applies when processing Natural library objects on UNIX, OpenVMS or Windows platforms.

Specifies that the FDDM system file is used for processing.

If the FDDM file has been activated in the NATPARM parameter file, the default setting is YES.

The following applies when specifying the values YES or NO:

Value	Explanation
YES	UNLOAD, FIND and DELETE: If the parameter NATTYPE is set to V, DDMs are only processed from the library SYSTEM located in the FDDM file or the file specified by the database ID (DBID) and the file number (FNR). No DDMs are processed if the parameter NATTYPE is set to *, or if NATTYPE is a combination of any Natural object types that does not include the type V. LOAD: DDMs are loaded into the library SYSTEM located in the FDDM file. See also NATTYPE in ***Natural Library Object and DDM Selection*** in *select-clause*.

Value	Explanation
`NO`	`UNLOAD`, `FIND` and `DELETE`: DDMs are processed from the libraries specified. `LOAD`: DDMs are loaded into the libraries specified.

38 Examples of Using Direct Commands

This section provides examples for using Object Handler direct commands.

Tip: For additional examples, you can view the command generated for an Object Handler function. This command is automatically displayed when you use a wizard. In advanced-user mode, you can activate the display of the command by either entering the Object Handler command `SET ADVANCEDCMD ON` or setting the parameter `Display-Cmd-in-Advanced-Mode` to `Y` (Yes) in the Object Handler profile (see also ***Profile Settings***).

Unloading Objects for the Same Platform

This section contains examples of how to unload objects in internal format to a work file in order to load them on the same platform, within either a local mainframe, UNIX, OpenVMS or Windows environment:

- Unload all Natural programming objects (source objects only) from library `ABC`:

  ```
  UNLOAD * LIB ABC OBJTYPE N SCKIND S
  ```

- Unload all Natural programming objects (cataloged objects only) from library `ABC`:

  ```
  UNLOAD * LIB ABC OBJTYPE N SCKIND C
  ```

- Unload all Natural programming objects (cataloged objects and source objects) from library `ABC`:

  ```
  UNLOAD * LIB ABC OBJTYPE N SCKIND A
  ```

- Unload all Natural programming objects (source objects only) from library `ABC` to load in library `ABCNEW`:

  ```
  UNLOAD * LIB ABC OBJTYPE N SCKIND S WITH NEWLIBRARY ABCNEW
  ```

- On a mainframe: Unload all DDMs whose names start with `EMP` and which point to database `88`:

  ```
  UNLOAD EMP* LIB * OBJTYPE D DDMDBID 88
  ```

- On UNIX, OpenVMS or Windows: Unload all DDMs whose names start with `EMP` and which point to database `88`:

  ```
  UNLOAD EMP* LIB * OBJTYPE N NATTYPE V DDMDBID 88
  ```

- On UNIX, OpenVMS or Windows: Unload all DDMs whose names start with `EMP` from library `VLIB` to load in library `VLIBNEW`:

```
UNLOAD EMP* LIB VLIB OBJTYPE N NATTYPE V WITH NEWLIBRARY VLIBNEW
```

- Unload all user-defined error messages from library `ERRLIB` to load in library `NEWERR`:

  ```
  UNLOAD * LIB ERRLIB OBJTYPE E SLKIND A WITH NEWLIBRARY NEWERR
  ```

- On Windows: Unload all Natural programming objects (cataloged objects and source objects) from library `ABC` to a portable work file on a PC:

  ```
  UNLOAD * LIB ABC OBJTYPE N WHERE WORKFILE C:\WF1.SAG WORKFILETYPE PORTABLE
  ```

  or

  ```
  UNLOAD * LIB ABC OBJTYPE N WHERE WORK C:\WF1.SAG WFT P
  ```

Unloading Objects for Different Platforms

This section contains command examples of how to unload objects in Transfer format to a work file in order to load them on a different platform such as unloading in a mainframe and loading in a UNIX, an OpenVMS or a Windows environment.

- Unload all Natural programming objects (source objects only) from library `ABC`:

  ```
  UNLOAD * LIB ABC OBJTYPE N WHERE TRANSFER
  ```

- Unload all Natural programming objects (source objects only) and user-defined error messages from library `ABC`:

  ```
  UNLOAD * LIB ABC WHERE TRANSFER
  ```

- Unload all Natural programming objects (source objects only) from library `ABC` with fixed record length:

  ```
  UNLOAD * LIB ABC OBJTYPE N WHERE TRANSFER FIXEDLENGTH
  ```

Loading Objects in Internal Format

This section contains command examples of how to load objects from a work file in internal format.

- Load all objects to library `LIBNEW` and replace any that already exist:

  ```
  LOADALL WITH NEWL LIBNEW WHERE REPLACE ALL
  ```

- Load all object with target library `TGTLIB` to the new target library `NEWTGT`:

```
LOAD * LIB TGTLIB WITH NEWLIBRARY NEWTGT
```

- Load the user-defined error messages `1000` to `1500` from library `ERRLIB` only:

```
LOAD * LIB ERRLIB OBJTYPE E FMNUM 1000 TONUM 1500
```

Loading Objects in Transfer Format

This section contains command examples of how to load objects from a work file in Transfer format.

- Load all objects to library `LIBNEW` and replace any that already exist:

```
LOADALL WITH NEWL LIBNEW WHERE TRANSFER REPLACE ALL
```

- Load all object with target library `TGTLIB` to new target library `NEWTGT`:

```
LOAD * LIB TGTLIB WITH NEWLIBRARY NEWTGT WHERE TRANSFER
```

39 Commands for Navigation and Special Functions

The Object Handler commands in CUI (character user interface) environments are mainly provided for navigation purpose and special function settings such as specifying trace files.

An Object Handler command is entered in the Command line of any Object Handler screen. If you want to execute a Natural system command from an Object Handler screen, enter two slashes (//) before the command. Note that any Natural system command terminates the Object Handler.

To invoke the Commands menu of the Object Handler

- Choose PF10 (Cmds).

  Or:

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

  ```
  CMDS
  ```

The Object Handler commands are listed below. An underlined portion of a keyword represents an acceptable abbreviation, `Sub` denotes subcommand.

Command	Sub 1	Sub 2	Explanation
`CANCEL`			Cancels the current function and displays the Object Handler **Main Menu**.
`CHANGE`	`WORKPLAN`	`LIBRARY`	Invokes the administration function and displays a screen where you can change the Workplan library.
`CLEAR`			Resets the current contents of the input fields in the map to the default values.
`CMDS` or			Invokes the **Commands** screen.

Command	Sub 1	Sub 2	Explanation
COMMANDS			
BYE			Terminates the Object Handler.
EXIT			
QUIT			
.			
FIN			Terminates the Object Handler and ends the Natural session.
GO	HOME		Displays the Object Handler **Main Menu**.
GO	UNLOAD		Invokes the unload function.
GO	UNLOAD	END	Ends the current unload function.
		ERROR	Invokes the unload function for Natural system error messages.
		EXTERNAL	Invokes the unload function for external objects.
		FDT	Invokes the unload function for FDTs.
		LIBRARY	Invokes the unload function for Natural library objects.
		NCP	Invokes the unload function for Natural command processor sources.
		RELATED	Invokes the unload function for Natural-related objects.
		SELECTION or LIST	Displays a screen where you can enter or select the SELECTION or LIST Workplan to be used for the unload function.
GO	LOAD		Invokes the load function.
GO	LOAD	ALL	Invokes the load function for all objects contained in the work file.
		END	Ends the current load function.
		ERROR	Invokes the load function for Natural system error messages.
		EXTERNAL	Invokes the load function for external objects.
		FDT	Invokes the load function for FDTs.
		LIBRARY	Invokes the load function for Natural library objects.
		NCP	Invokes the load function for Natural command processor sources.
		RELATED	Invokes the load function for Natural-related objects.
		SELECTION or	Displays a screen where you can enter or select the SELECTION or LIST Workplan to be used for the load function.

Command	Sub 1	Sub 2	Explanation	
		LIST		
GO	RESTART		Displays a screen where you can specify the file to be used for the restart load function.	
GO	SCAN		Invokes the scan function.	
GO	SCAN	ALL	Invokes the scan function for all objects contained in the work file.	
		END	Ends the current scan function.	
		ERROR	Invokes the scan function for Natural system error messages.	
		EXTERNAL	Invokes the scan function for external objects.	
		FDT	Invokes the scan function for FDTs.	
		LIBRARY	Invokes the scan function for Natural library objects.	
		NCP	Invokes the scan function for Natural command processor sources.	
		RELATED	Invokes the scan function for Natural-related objects.	
		SELECTION or LIST	Displays a screen where you can enter or select the SELECTION or LIST Workplan to be used for the scan function.	
GO	ADMIN		Invokes the administration function.	
GO	ADMIN	CHANGE	Displays a screen where you can change the Workplan library.	
		CREATE	Opens a menu with which you can create a Workplan.	
		LIST	Generates a list of Workplans available in the Workplan library.	
		GO	VIEW	Invokes the view function.
GO	VIEW	ERROR	Invokes the view function for Natural system error messages.	
		FDT	Invokes the view function for FDTs.	
		LIBRARY	Invokes the view function for Natural library objects.	
		NCP	Invokes the view function for Natural command processor sources.	
GO	FIND		Invokes the find function.	
GO	FIND	ERROR	Invokes the find function for Natural system error messages.	
		FDT	Invokes the find function for FDTs.	
		LIBRARY	Invokes the find function for Natural library objects.	

Command	Sub 1	Sub 2	Explanation
		NCP	Invokes the find function for Natural command processor sources.
		SELECTION or LIST	Displays a screen where you can enter or select the SELECTION or LIST Workplan to be used for the find function.
HELP			Invokes the Object Handler help function.
INIT			Reinitializes the Object Handler utility.
READ	PROFILE		Updates Object Handler settings as defined in the text object PROFILE (see also *Profile Settings*).
SET	ADVANCEDCMD	ON	Activates the display of commands generated by the Object Handler in advanced-user and compact input mode.
		OFF	Deactivates the display of commands generated by the Object Handler in advanced-user and compact input mode.
	EXECUTIONMSG	ON	Activates a window that displays the processing status.
		OFF	Deactivates a window that displays the processing status.
	FREE	ON	Activates free format editing.
		OFF	Deactivates free format editing.
	Input-Mode, IM	W	Select **wizard mode**.
		A	Select **advanced user mode**.
		C	Select **compact mode**.
	TRACE	ON	Activates trace mode: a trace of each Object Handler action is output to the screen.
		OFF	Deactivates trace mode.
		WORKFILE	Activates trace mode: a trace of each Object Handler action is output to Work File 10.
	TRACEFILE		Displays a screen where you can specify the name of the trace file (Work File 10).
SETTINGS			Displays a screen where you can specify the unload, load or scan settings.
SHOW or DISPLAY	LAST	MESSAGE	Displays the last interface return code and message issued by the processing interface of the Object Handler.
		RESULT	Displays the last result issued by the processing interface of the Object Handler.
	PROFILE		Display or modify the Object Handler profile.

Command	Sub 1	Sub 2	Explanation
	REPORT		Displays the report created last.
	STATISTICS		Displays statistics information about the objects processed.
	STATUS		Displays the current Object Handler status (contents of global variables).
	TRACE	FILE	Displays the contents of the trace file (Work File 10).

40 Batch Condition Codes and User Exit Routines

This section describes the condition codes returned for Object Handler functions in batch mode and the user exit routines available for function processing.

Condition Codes Returned in Batch

Object Handler processing in batch mode terminates with one of the following condition codes:

Condition Code	Explanation
0	Object Handler process terminated successfully.
30	An internal Object Handler error occurred.
40	An error was detected in the Object Handler command.
50	An error occurred during Object Handler processing.
60	A Natural Security error occurred during Object Handler processing.
99	A Natural error occurred during Object Handler processing.

Applying User Exit Routines

The Object Handler user exit routines are supplied as source objects in the Natural system library SYSOBJH. These source objects are named SRC-EX*nn*, where *nn* denotes the number of the user exit routine.

To activate a user exit routine

- `CATALOG` or `STOW` source object SRC-EX*nn* under the name `OBJHEX`*nn* in the Natural system library SYSOBJH.

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

  For detailed descriptions of the user exit routines, see the source objects of SRC-EX*nn* in the library SYSOBJH.

User Exit Routines Available

The following user exit routines are available:

OBJHEX01 for Processing Failures

Whenever a condition code is set to a value greater than 0 (zero) in batch mode, the user exit routine OBJHEX01 (if available) will be invoked before the Object Handler stops processing. With this user exit routine, you can specify whether to continue or terminate Object Handler processing. In the case of termination, you can change the condition code. For further details, see the source of the user exit routine SRC-EX01 in the Natural system library SYSOBJH.

OBJHEX02 for Object Rejection

If the Object Handler load function was executed successfully in batch mode (with Condition Code 0) or in online command mode, but one or more objects were rejected during loading (for example, not replaced), before the Object Handler stops processing, the user exit routine OBJHEX02 (if available) is invoked. With OBJHEX02, you can specify whether to continue or terminate Object Handler processing. In the case of termination, you can set a condition code. For further details, see the source of the user exit routine SRC-EX02 in the Natural system library SYSOBJH.

OBJHEX03 for Default Option Values

You can apply user exit routine OBJHEX03 to set default options for processing Object Handler commands. This user exit is invoked before an Object Handler command is processed. For further details, see the source object of the user exit routine SRC-EX03 in the Natural system library SYSOBJH.

OBJHEX04 for Natural Object Type Statistics

If the Object Handler unload, load or scan function was executed successfully in command mode, the user exit routine OBJHEX04 (if available) is invoked. It provides statistics for each Natural object type on the sources and catalogued objects processed, replaced, not replaced or rejected. With OBJHEX04, you can specify whether to continue or terminate the Object Handler processing in batch mode. In the case of termination, you can set a condition code. For further details, see the source of the user exit routine SRC-EX04 in the Natural system library SYSOBJH.

41 Tools

The Object Handler provides special features to display status information and reports and to check or modify trace settings.

Status

Displays the Object Handler functions currently used, the user environment, the Workplan library and the setting of the trace option described below.

To display the status

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

  ```
  SHOW STATUS
  ```

  See also the `SHOW` command described in *Commands for Navigation and Special Functions* in the section *Direct Commands*.

Last Result

Displays the last internal command issued by the processing interface of the Object Handler and possible return codes and messages.

To display the last result

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

  ```
  SHOW LAST RESULT
  ```

  See also the `SHOW` command described in *Commands for Navigation and Special Functions* in the section *Direct Commands*.

Traces

Activates or deactivates the trace function. Traces record internal Object Handler program flows to provide control information for error diagnoses. The trace option is set off by default.

To change the setting

- Use the command `SET TRACE` as described in *Commands for Navigation and Special Functions* in the section *Direct Commands*.

Reports

Lists the objects loaded, unloaded or scanned, and records errors that may interrupt processing. See also ***Work File Options*** in the section *Settings*. The report option is set on by default and is displayed after the unload, load or scan function has been executed.

To display the contents of the latest report file

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

  ```
  SHOW REPORT
  ```

  See also the `SHOW` command described in *Commands for Navigation and Special Functions* in the section *Direct Commands*.

42 Profile Settings

Natural provides the option to customize the default settings of your current Object Handler utility environment. For this purpose, Natural provides the text object OBJHPROF in the Natural system library SYSOBJH. OBJHPROF is used to specify environment-specific default values for flags and options that appear when entering the corresponding Object Handler screens.

To activate individual profile settings

- In online mode, enter the Object Handler internal command `PROFILE` (or `SHOW PROFILE`).

  This command invokes the Profile Maintenance tool that

  - displays a map with the general or user-specific profile parameters and their current values;
  - creates a new Object Handler profile with default values used internally in case an Object Handler profile does not exist;
  - allows you to modify general settings for the profile parameters in the Object Handler profile (controlled by Natural Security);
  - allows you to modify user-specific settings for the profile parameters in the Object Handler profile (controlled by Natural Security);
  - provides a description and help information for each profile parameter.

Enter PF4 to toggle user profile settings. You can use a ***Line Command*** to add, modify, or delete parameters from the user-specific profile.

Notes:

1. If new parameters are added, the Profile Maintenance tool internal command `UPDATE` updates the changes into the Object Handler profile. For more information, invoke **Help**.
2. The Object Handler profile itself is named OBJHPROF. It is located in library SYSOBJH. The default profile that is used for updates of the Object Handler profile is named OBJHDEFP. It is also located in library SYSOBJH.

To deactivate individual profile settings

- Delete the text object OBJHPROF from the library SYSOBJH.

PF Keys

The following PF keys are available:

PF key	Description
PF1 (**Help**)	Invoke general or context-sensitive help.
PF2 (**Print**)	Print the current profile settings.
PF3 (**Exit**)	Save the profile data and terminate the Profile Maintenance tool.
PF4 (**Gener**/**User**)	Toggle between general (**Gener**) and user-specific (**User**) profile settings.
PF5 (**Modif**)	Modify the parameter (marked by cursor selection) in a separate menu.
PF6 (**--**)	Scroll to the beginning of the parameter list.
PF7 (**-**)	Scroll one page backwards.
PF8 (**+**)	Scroll one page forward.
PF9 (**++**)	Scroll to the end of the parameter list.
PF10 (**ALLDU**/**ALLAU**)	For every parameter: insert line command `DU` or `AU`.
PF12 (**Canc**)	Terminate the Profile Maintenance tool.

Line Commands

The following line commands are available :

Line Command	Description
MO	Modify parameter value in extended mode.
AU	Add entry to user profile
DU	Delete entry from user profile
DI	Display description of the parameter
HE	Display help information on the parameter

Profile Parameters

The table below lists the parameters contained in the Object Handler profile OBJHPROF, the possible values that can be entered and the Object Handler functions to which the parameters apply. In addition, the table provides a brief description of the parameters or a reference to the corresponding Object Handler documentation section. Default parameter values are underlined.

Parameter	Possible Values	Function	Description/ Documentation Section
`Input-Mode`	`W`	Unload Load Scan	*Wizards*
	`A`		*Advanced User*
	`C`		*Compact Mode*
`Display-Cmd-in-Advanced-Mode`	`N` or `Y`	Unload Load Scan	Displays the Object Handler command generated for a function executed in advanced-user mode.
`Display-ExecutionMsg`	`N` or `Y`	Unload Load Scan	Activates a window that displays the processing status.
`Display-Statistics`	`N` or `Y`	Unload Load Scan	Displays statistics on objects processed after the function has been executed. (equivalent to direct command `SHOW STATISTICS`).
`Workplan-Library`	`WORKPLAN` or any other Workplan library	Unload Load Scan Administration	*Workplans* and *Change the Workplan Library* in *Administration*
`Workplan-Library-DBID`	`0` (current FNAT/FUSER) or any other Adabas database ID (DBID)	Unload Load Scan Administration	*Change the Workplan Library*
`Workplan-Library-FNR`	`0` (current FNAT/FUSER) or any other Adabas file number (FNR)	Unload Load Scan Administration	*Change the Workplan Library*
`TRACE`	`N` or `Y`	Unload Load Scan	*Traces* in *Tools*

Parameter	Possible Values	Function	Description/ Documentation Section
`TRACE-TARGET`	`S` (Screen) or `W` (Work file)	Unload Load Scan	*Traces*

`Option-Replace`	`N`
	or
	`Y`
	or
	`O` (Obsolete)
	`E` (Except)
		Load	*Replace Options* in *Settings*

`Option-TRANSFER-FORMAT`	`N`
	or
	`Y`
		Unload
		Load
		Scan
			Work File Format in *Work Files*

`Option-Use-PC-Work-File`	`N`
	or
	`Y`
		Unload
		Load
		Scan
			See
			
			Use PC File in *Set Additional Options*

`Option-TR-INCLUDE-LINE-NUMBERS`	`N`
	or
	`Y`
		Unload	*Include line numbers* in
			
			Transfer Options (*Settings*)
			or
			Transfer Options (Direct Commands)

`Option-TR-LINE-NUMBER-INCREMENT`	`N`
	or
	`Y`
		Unload
		Load
			USE-LINE-NUMBER-INCREMENT in *Transfer Options* (Direct Commands)

`Option-TR-SUBSTITUTE`	`N`
	or
	`Y`
		Unload	**Substitute line references** in *Transfer Options* (*Settings*)
			or
			SUBSTITUTE in *Transfer Options* (*Direct Commands*)

`Option-TR-TRANSLATE-TO-UPPER`	`N`
	or
	`Y`
		Load	**Translate to upper case** in *Transfer Options*(*Settings*)
			or
			UPPERCASE-TRANSLATION in *Transfer Options* (*Direct Commands*)

`Option-TR-USE-CONVERSION-TABLE`	`N` or `S` (System table) or `U` (User table)	Unload Load	**Use conversion table** in *Transfer Options* (*Settings*) or **CONVERSION-TABLE** in *Transfer Options* (*Direct Commands*)
`Option-TR-CONV-TABLE-NAME-LOAD`	`OTNCONEA` or a user-written subprogram	Load	**Use conversion table** in *Transfer Options* (*Settings*) or **CONVERSION-TABLE** in *Transfer Options* (*Direct Commands*)

Parameter	Possible Values	Function	Description/ Documentation Section
`Option-TR-CONV-TABLE-NAME-UNLD`	`OTNCONAE` or a user-written subprogram	Unload	**Use conversion table** in *Transfer Options* (*Settings*) or **CONVERSION-TABLE** in *Transfer Options* (*Direct Commands*)

`Option-TR-DA-FORMAT`	`N`
	or
	`*`
		Unload
		Load
			Data area format in *Transfer OptionsOptions*
			or
			DA-FORMAT in *Transfer Options* (*Direct Commands*)

`Option-TR-UNICODE-WORK-FILE`	`N`
	or
	`Y`
		UNLOAD	**Unicode work file** in *Settings Screen Fields*

`Option-TR-LOAD-CODE-PAGE`	`Code page name`
	or
	`*CODEPAGEY`
		LOAD	**Use load code page** in *Tansfer Options* (*Settings*)

`Option-Write-Report`	`N`
	or
	`Y`
		Unload
		Load
		Scan
			Reports in *Tools*
			
			Write report in *Work File and Report Options*

`Default-Report-Direct-Command`	`B`
	or
	`N`
	or
	`Y`
		Unload
		Load
		Scan
			Defines the kind of report to be written when a direct command is executed and no report option is specified in the Object Handler command.
			
			`B` Write a batch report. Report data is written directly to the output device. Corresponds to option `BATCHREPORT`.
			
			`Y` Write a report using Work file 4, or when working on mainframe a text object. Corresponds to option `REPORT`.
			
			`N` Do not write a report. Corresponds to option `NOREPORT`.

`Default-Report-Option-1`	`A`
	or
	`E`
	or
	`S`
		Unload
		Load
		Scan
		Delete
			Defines the option for the report to be written when a direct command is executed and option `REPORT-OPTION-1` is not specified in the Object Handler command. For possible values, see `REPORT-OPTION-1`.

Parameter	Possible Values	Function	Description/ Documentation Section

`Default-Report-Format`	`S`
	or
	`T`
		Unload	Defines the format of the report to be written when a direct command is executed and option `REPORT-FORMAT` is not specified in the Object Handler command. For possible values, see `REPORT-FORMAT`.

`Default-Report-Mode`	`S`
	or
	`L`
		Unload
		Load
		Scan
			Defines the report mode to be used when a direct command is executed and option `REPORT-MODE` is not specified in the Object Handler command. For possible values, see `REPORT-MODE`.

`Option-Write-Restart-Info`	`Y`
	or
	`N`
		Load	**Write restart information** in *Work File and Report Options*

`USE-OPTION-WORKPLAN`	`N`
	or
	`Y`
		Unload
		Load
		Scan
			Workplans

`OPTION-WORKPLAN-Name`	`OPTIONWP` or any other Workplan of the type OPTION	Unload Load Scan	***Workplans***

`USE-PARAMETER-WORKPLAN`	`N`
	or
	`Y`
		Unload
		Load
			Workplans

`PARAMETER-WORKPLAN-Name`	`PARAWPLN` or any other Workplan of the type PARAMETER	Unload Load	***Workplans***
`WORK-FILE-1-Name`	The complete path name assigned to Work File 1.	Unload Load Scan	***Work Files***
`Report-File-Name`	The complete path name assigned to Work File 4.	Unload Load Scan	**Write report** in *Work File and Report Options*
`Restart-File-Name`	The complete path name	Load	**Write restart information** in *Work File and Report Options*

Parameter	Possible Values	Function	Description/ Documentation Section
	assigned to Work File 6.		
`Trace-File-Name`	The complete path name assigned to Work File 10.	All functions	*Traces* in *Tools*
`DELETE-TEMPORARY-REPORT-TEXT`	`N` or `Y`	Unload Load Scan	Valid on mainframe only: `Y` = temporary text objects for the report are deleted when the Object Handler is terminated (default value) `N` = temporary text objects for the report are not deleted when the Object Handler is terminated

43 Migration from NATUNLD/NATLOAD and SYSTRANS to the Object Handler

You can migrate from the old utilities NATUNLD/NATLOAD and SYSTRANS to the Object Handler by using the two methods described in this section.

Converting Individual Commands

You can convert NATUNLD/NATLOAD or SYSTRANS direct commands to the corresponding Object Handler commands by using the Object Handler commands provided for migration. These migration commands automatically convert the command syntax used by the old utilities to the command syntax used by the Object Handler.

To convert a single command

1 Use one of the following Object Handler direct commands:

```
NATUNLD
```

followed by a NATUNLD direct command.

Or:

```
NATLOAD
```

followed by a NATLOAD direct command.

Or:

```
SYSTRANS
```

followed by a SYSTRANS direct command.

The specified utility command is converted to the corresponding Object Handler command.

2 Specify any subsequent command for the Object Handler in the syntax that applies to the utility NATUNLD, NATLOAD or SYSTRANS respectively.

The syntax of this utility remains valid for the duration of the Object Handler session.

Example of a NATUNLD Command:

The following is an example of two consecutive NATUNLD utility commands and their corresponding Object Handler commands.

Old NATUNLD commands:	`NATUNLD ALL * FM LIB1 TO LIB2`
	`ALL PG* FM LIB2`
New Object Handler command:	`SYSOBJH NATUNLD ALL * FM LIB1 TO LIB2`
Subsequent Object Handler command in NATUNLD syntax:	`ALL PG* FM LIB2`

Example of a SYSTRANS Command:

The following is an example of two consecutive SYSTRANS utility commands and their corres-ponding Object Handler commands.

Old SYSTRANS commands:	`TRANSCMD EXECUTE UNLOAD N FROM LIB1 NAME ETID`
	`END`
New Object Handler command:	`SYSOBJH SYSTRANS EXECUTE UNLOAD N FROM LIB1 NAME ETID END`
Subsequent Object Handler command in SYSTRANS syntax:	`END`

Example of SYSTRANS Batch Processing:

The following is an example of processing a SYSTRANS utility command in batch by using map input data, and the corresponding Object Handler command and input data.

Old SYSTRANS batch sequence:

```
SYSTRANS
U
N,N,N,Y,N,N,N,N
N
SRCLIB1,PGM1,*,TGTLIB1
```

New Object Handler batch sequence:

```
SYSOBJH SYSTRANS
U
N,N,N,Y,N,N,N,N
N
SRCLIB1,PGM1,*,TGTLIB1
```

Processing SYSTRANS Commands with OBJHAPI

You can use the OBJHAPI Application Programming Interface (supplied in the Natural system library SYSOBJH) to execute an Object Handler command in the syntax of the SYSTRANS utility.

If you use OBJHAPI for this purpose, you have to specify the parameter `P-EXTENSIONS-EXEC-SYSTRANS-CMD` in the program that invokes OBJHAPI. For details, see the example program DOC-API supplied in the library SYSOBJH.

Unsupported SYSTRANS Options

The Object Handler does not support the following SYSTRANS direct command options: `WORK-FILE-INPUT`, `SPECIAL-CONVERSION`, `RULE-LOAD` and `UNLOAD-RULES`.

VI SYSERR Utility

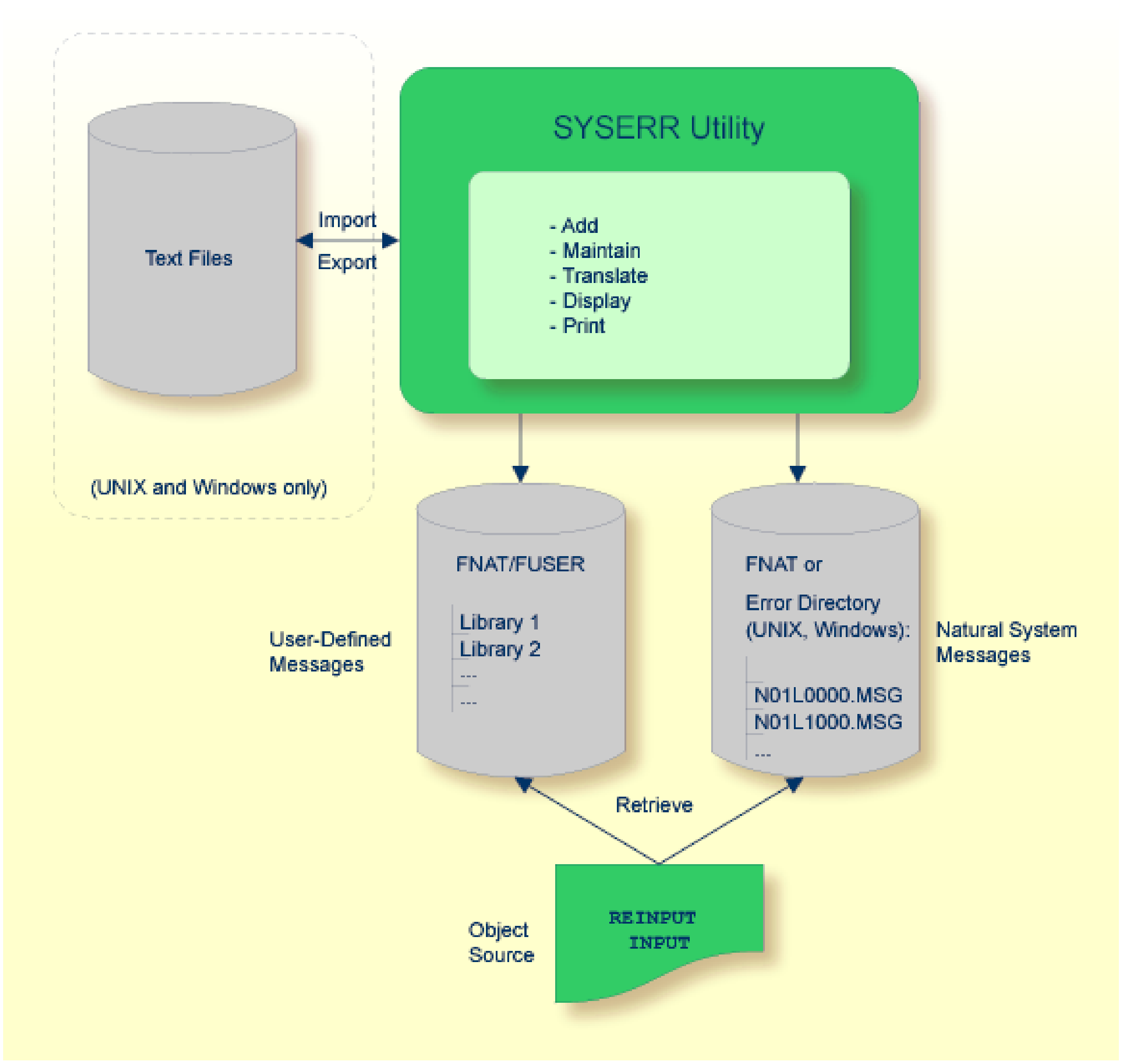
When you develop a Natural application, you may want to separate error or information messages from your Natural code and manage them separately. This makes it easy for you, for example, to standardize messages, to have predefined message ranges for different types of message, to translate messages into other languages or to attach to a message a long text that explains it in more detail.

The SYSERR utility provides the option to write application-specific messages. In addition, you can use the SYSERR utility to customize the texts of the existing Natural system messages.

44 General Information on Messages

This section contains information on the types of message and message languages that can be managed with the SYSERR utility and how messages are issued and retrieved in your Natural system environment.

The following graphic illustrates the features of the SYSERR utility and how messages are processed within Natural:



Message Types

There are two types of message: Natural (system) messages and user-defined messages:

Natural system messages are issued by the Natural nucleus and Natural utilities. Natural system messages are delivered by Software AG and stored as message files in the Natural Err directory. Natural system messages begin with `NAT`, followed by a four-digit number, for example, NAT0230.

User-defined messages are issued by applications written by a user. User-defined messages are stored as message files in libraries (including SYS-libraries) in the system file FUSER or FNAT.

A message can be translated into different languages. Each language is stored in a separate message file. A maximum of 9999 messages can be stored per library and message file.

There are four types of message text:

- Natural system short message
- Natural system long message
- User-defined short message
- User-defined long message

A short message is the one-line message which is displayed in the message line when the corresponding error situation occurs.

A long message is a detailed explanation of the corresponding short message and includes instructions for solving problems.

Caution: Keep in mind that any modifications of Natural system messages can result in wrong messages or a loss of modifications when a new Natural version is released.

Message Languages

Messages can be created in up to 60 languages as described for the system variable `*LANGUAGE` in the *System Variables* documentation.

The following rules and restrictions apply:

- Natural system short messages must be entered in English first, and can then be translated into any other language.
- Natural system long messages can be entered in English, but cannot be translated into other languages.

- User-defined short messages can be entered in any language, and then be translated into any other language.
- User-defined long messages can be entered in any language, but only if the corresponding short message in the same language already exists.

Issuing Messages

This section contains information on the Natural statements `INPUT` and `REINPUT` that are used to issue a Natural system short message or a user-defined short message in a Natural program.

To issue a Natural system short message in a program

- Specify one of the following Natural statements:

  ```
  INPUT WITH TEXT *-nnnn ' '
  ```

  or

  ```
  REINPUT WITH TEXT *-nnnn
  ```

  where *nnnn* is the number of the requested message (you can omit leading zeros).

To issue a user-defined short message in a program

- Specify one of the following Natural statements:

  ```
  INPUT WITH TEXT *nnnn ' '
  ```

  or

  ```
  REINPUT WITH TEXT *nnnn
  ```

  where *nnnn* is the number of the requested message (you can omit leading zeros).

Dynamic Replacement of Message Text

A message text can contain variable parts that are identified by the notation `:n:`, where *n* represents occurrences 1 to 7. These variable parts are replaced by a value at runtime.

For details, see *operand3* in the section *INPUT Syntax 1 - Dynamic Screen Layout Specification* and *operand3* in the section *REINPUT* in the *Statements* documentation.

Retrieving Natural System Short Messages

When a program references a Natural system short message, Natural looks for the requested message number in the Natural Err directory in the following order:

1. Under the current language code as determined by the system variable `*LANGUAGE`,
2. Under language code 1 (English).

If neither of the above is found, a program references a message that does not exist and you only receive the message number prefixed with `NAT`, for example, NAT0230.

Retrieving User-Defined Short Messages

When a program references a user-defined short message, Natural first looks for the requested message number *nnnn* under the current language code as determined by the system variable `*LANGUAGE` (see the *System Variables* documentation). If that message does not exist, Natural looks for the requested message number *nnnn* under language code 1 (English). If that message does not exist either, Natural looks for message number *n*`000` (where *n* is the first digit of the requested message number) under language code 1.

These three search steps are first performed in the current library. If nothing is found there, further libraries are searched in the same way until a corresponding message is found.

The sequence of libraries for the search is as follows:

1. The current library as determined by the system variable `*LIBRARY-ID`,
2. The steplibs; if Natural Security is installed, the sequence in which the steplibs are specified in the Natural Security profile of the current library,
3. The default steplib as determined by the system variable `*STEPLIB`,
4. The library SYSTEM in the system file FUSER (*),
5. The library SYSTEM in the system file FNAT (*).

(*) If the name of the current library begins with SYS, SYSTEM FNAT is searched before SYSTEM FUSER.

Obtaining Message Information

When you receive a short message, you may be looking for additional information on the problem situation.

- With the system command `HELP`, you can display Natural system long messages or user-defined long messages.
- With the system command `LASTMSG`, you can list the short test of the message(s) that occurred last and additional information on the error situation. The information displayed includes associated error messages that possibly preceded the last message.

Both commands are described in the *System Commands* documentation.

45 Invoking SYSERR

To invoke the SYSERR utility

- Enter the following system command:

  ```
  SYSERR
  ```

  The main menu of the SYSERR utility is displayed:

```
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
```

From the main menu of the SYSERR utility, you can execute all SYSERR functions available for creating and maintaining messages. The individual functions are explained in the section ***Functions***. The parameters that apply with the functions are explained in general in the section ***Parameters***, any restrictions that apply to the use of these parameters are described for each function concerned in the section *Functions*.

The SYSERR utility provides an extensive online help system. To obtain field-specific help information, either enter a question mark in the relevant field and press ENTER or place the cursor in the field and press PF1.

46 Functions

You invoke a SYSERR utility function by entering the code that corresponds to the required function and one or more parameters in the input fields of the SYSERR main menu. This section describes the functions provided in the menu and the parameters that can be specified for each function. For general instructions on the use of parameters, see the section ***Parameters***.

Adding Messages

To add new messages

1 The SYSERR utility is case-sensitive by default. If you want lower to upper case translation for the messages to be created, enter the following terminal command:

```
%U
```

Any lower case characters you type when adding message text are then converted to upper case characters for the duration of the current Natural session.

For detailed information on `%U`, see the *Terminal Commands* documentation.

2 Invoke the SYSERR main menu and enter the following values:

Field	Input Value
Code	`AD`
Message type	`NS` Natural system short messages `NL` Natural system long messages `US` User-defined short messages `UL` User-defined long messages A long message can only be added if the corresponding short message already exists, as the long message is intended to be an explanation of the short message.
Library	Any existing Natural library.
Message number	Two numbers of up to four digits corresponding to the first and last numbers of the range of messages to be added. If you only want to add one message, either enter the number of the new message in the left **Message number** field and clear the right field, or enter the number in both fields.
Language codes	The code of the language for which the message is to be added. If the message type is `NS` or `NL`, the language code must be `1` for English. For other message types, the first language code entered in the field is used; all others are ignored.

3 Press ENTER.

An **Add Short Message** screen similar to the example below is displayed:

```
15:53:03              ***** 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
```

The **Number** field indicates the message number (in the example above, `1004`), which is prefixed with the library ID (in the example above, `SYSERR`).

4 In the input line next to the message number, type in a short message text and press ENTER.

Or:

If the line labeled **Sample** contains a sample message text as shown in the example above, copy this text into the input line by entering `.C` and then pressing ENTER. If the sample message text contains the string `0000`, this string is replaced by the new message number as illustrated in the following example:

```
15:57:14             ***** 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  ↩
 ↩
```

For instructions on creating a sample message, see the SAMPLE command described in *Direct Commands*.

5 Press PF9 to add a corresponding long message text.

An **Add Long Message** screen similar to the example below appears:

```
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 Enter text in the three input areas: **Tx.** (text), **Ex.** (explanation) and **Ac.** (action).

7 Press ENTER to save the long message.

8 Press PF9 to return to the short message or to add the next short message in ascending order if you selected a range of message numbers.

9 Press PF3 or PF12 to return to the SYSERR main menu.

Or:

Press PF8 or PF7 to add the next short message in ascending or descending order if you selected a range of message numbers.

Deleting Messages

To delete messages

- In the fields of the SYSERR main menu, enter the following values:

Field	Input Value
Code	`DE`
Message type	`NS` Natural system short messages `NL` Natural system long messages `US` User-defined short messages `UL` User-defined long messages It is possible to delete a long message without deleting the corresponding short message, but not vice versa. If you try to delete a short message for which a long message exists, you are asked to confirm the deletion of both.
Library	Any existing Natural library.
Message number	Two numbers of up to four digits corresponding to the first and last numbers of the range of messages to be deleted.
Language codes	The code(s) of the language(s) in which the messages are to be deleted. To indicate that the messages specified are to be deleted in all languages available, enter an asterisk (*).

Displaying Messages

To display messages

1 In the fields of the SYSERR main menu, enter the following values:

Field	Input Value
Code	`DI`
Message type	`NS` Natural system short messages `NL` Natural system long messages `US` User-defined short messages `UL` User-defined long messages
Library	Any existing Natural library.
Message number	Two numbers of up to four digits corresponding to the first and last numbers of the range of messages to be displayed.
Language codes	The code of the language in which the messages are to be displayed. Only one language code is accepted. If more than one code is specified, only the first one is used; all others are ignored.

2 Press ENTER.

For short messages, a **Display Short Messages** screen similar to the example below appears:

```
15:41:11              ***** NATURAL SYSERR UTILITY *****             2008-11-28
                          - Display Short Messages -

Number         Short Message (English)
------------   ----------------------------------------------------------------
NAT0001        Missing/invalid syntax; undefined variable name/keyword.
NAT0002        No file is available with specified name or number.
NAT0003        Invalid character string for file name or file number.
NAT0004        DEFINE DATA must be the first statement if present.
NAT0005        Closing parenthesis missing in arithm/logical expression.
NAT0006        ESCAPE statement used when no processing loop active.
NAT0007        Invalid THRU or TO clause in READ LOGICAL or HISTOGRAM.


















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

Press PF8 to page forwards.

For long messages, the **Display Long Message** screen is displayed where the messages are displayed one after another by pressing PF8 to page forwards or PF7 to page backwards. The **Display Long Message** screen is similar to the **Modify Long Message** screen shown in *Modifying Messages*.

Modifying Messages

To modify messages

1 The SYSERR utility is case-sensitive by default. If you want lower to upper case translation for the messages to be modified, enter the following terminal command:

```
%U
```

Any lower case characters you type when editing message text are then converted to upper case characters for the duration of the current Natural session.

For detailed information on `%U`, see the *Terminal Commands* documentation.

2 In the fields of the SYSERR main menu, enter the following values:

Field	Input Value
Code	`MO`
Message type	`NS` Natural system short messages `NL` Natural system long messages `US` User-defined short messages `UL` User-defined long messages
Library	Any existing Natural library.
Message number	Two numbers of up to four digits corresponding to the first and last numbers of the range of messages to be modified.
Language codes	The code of the language in which the messages are to be modified. Only one language code is accepted. If more than one code is specified, only the first one is used; all others are ignored.

3 Press ENTER.

A **Modify Short Message** screen similar to the example below is displayed:

```
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
```

For reference purposes, the long message is displayed in the bottom half of the screen.

When you modify long messages, the **Modify Long Message** screen is displayed:

```
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 Press ENTER to save any modifications.

5 Press PF8 or PF7 to modify the next message in ascending or descending order if you selected a range of numbers.

Printing Messages

To print messages

1 In the fields of the SYSERR main menu, enter the following values:

Field	Input Value
Code	`PR`
Message type	`NS` Natural system short messages `NL` Natural system long messages `US` User-defined short messages `UL` User-defined long messages
Library	Any existing Natural library.

Field	Input Value
Message number	Two numbers of up to four digits corresponding to the first and last numbers of the range of messages to be printed.
Language codes	The code of the language in which the messages are to be printed. Only one language code is accepted. If more than one code is specified, only the first one is used; all others are ignored.

2 Press ENTER.

A print window similar to the example below opens:

```
+------Print Natural System Messages-------+
!                                          !
!   Language code .... 1                   !
!                                          !
!   Long texts, too .. N                   !
!   Message number ... 1___  -   25        !
!   Lines per page ... 60_                 !
!   Left margin ...... 10                  !
!   Top margin ....... 0_                  !
!   Bottom margin .... 0_                  !
!   Printer ID ....... PRT1____            !
!                                          !
!                                          !
+------------------------------------------+
```

3 Specify the options provided in the print window and the logical printer name.

See the `DEFINE PRINTER` statement in the *Natural Statements* documentation for details on logical printer names.

4 Press ENTER to output the selected messages on a printer.

To print all Natural system messages

- In the fields of the SYSERR main menu, enter the following values:
  - Code `PR`,
  - Message type `NS` or `NL`,
  - Message number range `1 - 9999`,
  - Language code `1` (English) or `2` (German).

  A library ID is not required and possible entries are ignored.

Scanning Messages

This function is used to scan messages for a specific string of characters. Only short messages can be scanned.

To scan messages

1 In the fields of the SYSERR main menu, enter the following values:

Field	Input Value
Code	`SC`
Message type	`NS` Natural system short messages `US` User-defined short messages
Library	Any existing Natural library.
Message number	Two numbers of up to four digits corresponding to the first and last numbers of the range of messages to be scanned.
Language codes	Specify a maximum of nine language codes from the ranges 1 - 9, A - Z and a - y, or enter an asterisk (*) for all languages.

2 Press ENTER.

A scan window similar to the example below opens:

```
+---------------------------------------+
|   Scan value(s)       Or/And/Not      |
|   ----------------    ----------      |
|   ________________       OR_          |
|   ________________                    |
|   ________________                    |
|   ________________                    |
|                                       |
|   Absolute ..... X                    |
|   Immediate .... _                    |
|                                       |
+---------------------------------------+
```

In the fields provided, you can specify the search criteria to be used for scanning:

Scan value(s)	In the four empty fields, enter up to four character strings to be searched for. The scan finds the specified terms in both upper and lower case.	
Or/And/Not	You can perform a Boolean search query by entering one of the following operators:	
	`OR`	Searches for one or more of the character strings entered in **Scan value(s)**. This is the default setting.
	`AND`	Searches for all of the character strings entered in **Scan value(s)**.
	`NOT`	Searches for none of the character strings entered in **Scan value(s)**.
	The operator is ignored if you only fill one of the **Scan value(s)** fields.	
Absolute	If you mark this field, the string of characters is found even if it is part of a word. For example, if you scan for the value `meter`, the search would also find words such as parameter and millimeter. If you remove the mark, the search is restricted to match entire words only.	
Immediate	If you mark this field, messages are displayed individually, one after another. Otherwise, a list of messages is displayed after the search is completed. If you specify more than one language or an asterisk (*) in the **Language codes** field, **Immediate** must be marked.	

3 Specify search criteria as shown in the following example:

```
+-----------------------------------+
!   Scan value(s)      Or/And/Not    !
!   ----------------   ----------    !
!   BUFFER__________       AND       !
!   POOL____________                 !
!   ________________                 !
!   ________________                 !
!                                    !
!   Absolute ..... X                 !
!   Immediate .... _                 !
!                                    !
+-----------------------------------+
```

In the example above, the scan finds all short messages that contain both the words `buffer` and `pool`.

4 Press ENTER.

All messages to which the specified search criteria apply are listed on the screen as shown in the following example:

```
11:32:27              ***** NATURAL SYSERR UTILITY *****              2008-11-28
                           - Scan in Short Messages -

Number         Short Message (English)
------------- -----------------------------------------------------------------
NAT0777        Buffer pool full.













End of scan reached.


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

The word in which the search string is found is highlighted.

From this screen, you can display the search criteria used for the current scan by pressing PF10.

Selecting Messages from a List

This function is used to display a range of messages and select single ones for further processing. Only short messages can be displayed.

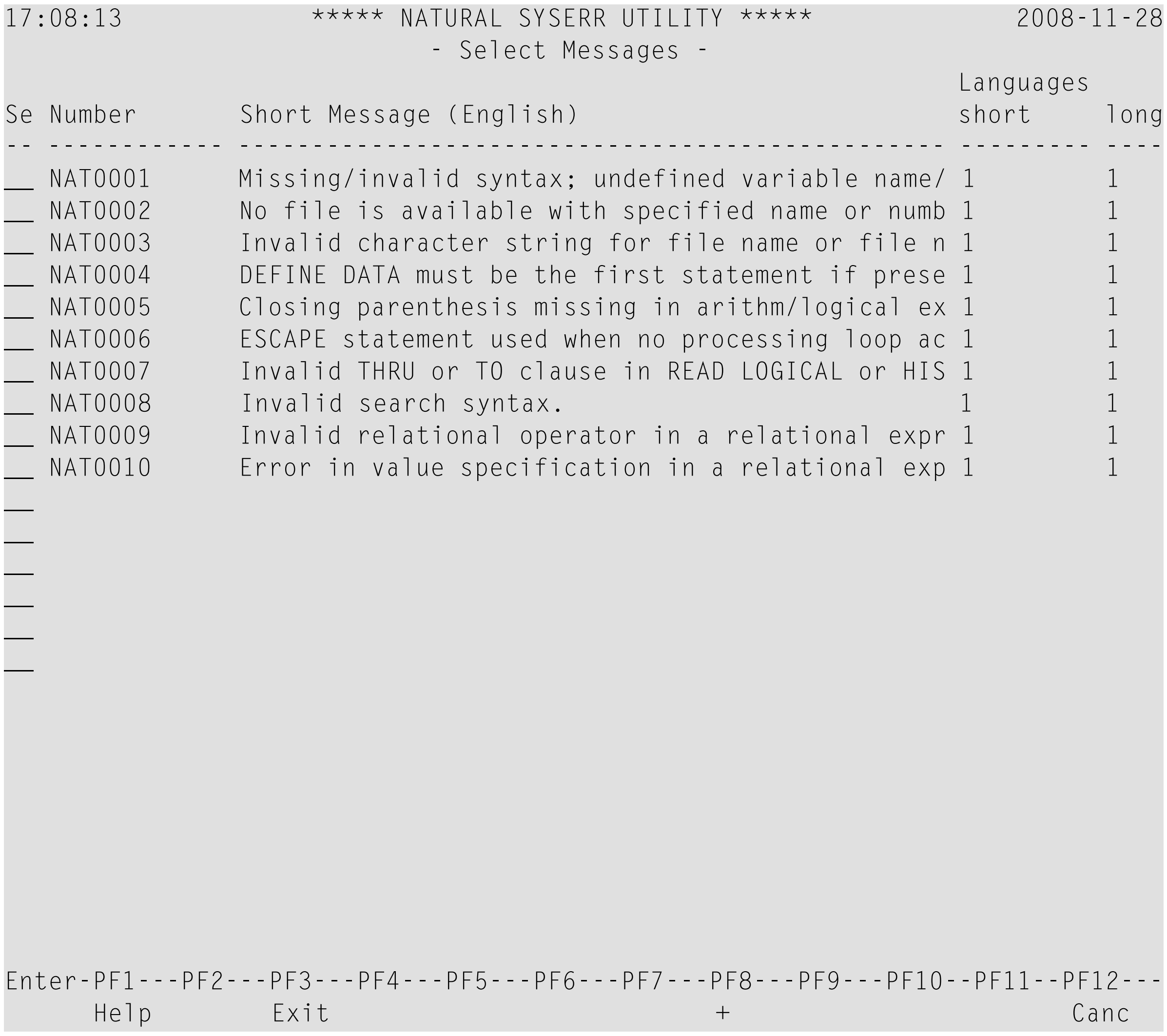
To select messages

1 In the fields of the SYSERR main menu, enter the following values:

Field	Input Value
Code	`SE`
Message type	`NS` Natural system short messages `US` User-defined short messages
Library	Any existing Natural library. If an asterisk (*) is appended to the library ID, a list of all libraries available is displayed for selection.
Message number	Two numbers of up to four digits corresponding to the first and last numbers of the range of messages to be displayed for selection.
Language codes	The code of the language in which the messages are to be displayed. If more than one code is specified, only the short message text of the first one is displayed. Enter an asterisk (*) to display the languages available for each message.

2 Press ENTER.

A **Select Messages** screen similar to the example below is displayed:



```
17:08:13            ***** NATURAL SYSERR UTILITY *****          2008-11-28
                             - Select Messages -
                                                                Languages
Se Number       Short Message (English)                         short     long
-- ------------ ----------------------------------------------- --------- ----
__ NAT0001      Missing/invalid syntax; undefined variable name/ 1         1
__ NAT0002      No file is available with specified name or numb 1         1
__ NAT0003      Invalid character string for file name or file n 1         1
__ NAT0004      DEFINE DATA must be the first statement if prese 1         1
__ NAT0005      Closing parenthesis missing in arithm/logical ex 1         1
__ NAT0006      ESCAPE statement used when no processing loop ac 1         1
__ NAT0007      Invalid THRU or TO clause in READ LOGICAL or HIS 1         1
__ NAT0008      Invalid search syntax.                           1         1
__ NAT0009      Invalid relational operator in a relational expr 1         1
__ NAT0010      Error in value specification in a relational exp 1         1
__
__
__
__
__
__









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

3 In the **Se** column, enter one of the following line commands:

Command	Explanation
`DE`	Deletes the message.
`DI`	Converts the **Select Messages** screen to the **Display Short Messages** screen shown in *Displaying Messages*. Additionally, places the message selected with this command at the top of the list and reduces the number of messages displayed as described for `.X`.
`LA`	Shows the codes of the languages for which translations exist.
`MO`	Modifies the message.
`PR`	Outputs the message on a printer.
`SH`	Displays the short message. This command is only available if an asterisk (*) has been entered in the **Language codes** field of the SYSERR main menu.
`TR`	Translates the message into another language.

Command	Explanation
`.X`	Defines a shorter message range by placing a selected message at the top of the list and thus reducing the number of messages displayed: The message selected with this command is placed at the top of the list and any messages that were listed above this message are removed from the display. The message range in the SYSERR main menu is reset accordingly and starts with the message selected here on the **Select Messages** screen.
`.Y`	Defines a shorter message range by listing messages only up to a selected message: All messages that were listed below the message selected with this command are removed from the display. The message range in the SYSERR main menu is reset accordingly and ends with the message selected here on the **Select Messages** screen.

4 Press ENTER to continue.

Translating Messages into other Languages

This function is used to translate short messages from one language to one or more other languages. To translate long messages into other languages, proceed as described in ***Adding Messages***.

To translate short messages

1 In the fields of the SYSERR main menu, enter the following values:

Field	Input Value
Code	`TR`
Message type	`NS` Natural system short messages `US` User-defined short messages
Library	Any existing Natural library.
Message number	Two numbers of up to four digits corresponding to the first and last numbers of the range of messages to be displayed for selection.
Language codes	Specify a maximum of nine language codes. The language codes are single alphanumeric characters in the ranges 1 - 9, A - Z and a - y.

2 Press ENTER.

A **Translate Short Message** screen similar to the example below appears:

```
13:42:31            ***** 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
```

The **Languages** field displays the language codes in which the message already exists (in the example above: 1, 4 and 5).

The section below **Number** and **Languages** lists all languages for which a language code was entered earlier in the **Language codes** field of the SYSERR main menu (in the example above: 1, 2, 3, 4, 5). English (1), Spanish (4) and Italian (5) translations already exist whereas new translations can be entered for German (2) and French (3).

For reference purposes, the bottom section of the screen displays three lines of the long message that corresponds to the language that is listed first in the languages/short messages section (in the example above, English). Lines 1, 4 and 18 are displayed by default. You can display any other line of the long message by overwriting any of the three line numbers (**1**, **4** or **18**) with another line number and pressing ENTER.

3 Enter the translation in the input line next to the new language specified.

4 Press ENTER.

To modify translations of short messages

1 On the **Translate Short Message** screen, press PF10.

An **Options** window similar to the example below opens:

```
+----------------------- Options -----------------------+
!                                                       !
!  Modification of all fields allowed ....... N         !
!                                                       !
!  Currently recognized language codes ...... 123456789   !
!                                                       !
+-------------------------------------------------------+
```

2 In the upper field, replace `N` (default) by `Y` and enter the required language code(s) in the lower field. You can specify up to nine new language codes for translation.

To copy a translation into an empty input line

1 On the **Translate Short Message** screen, enter `.C` in the first two positions of an empty line.

2 Place the cursor anywhere in the line of a short message that already exists for another language. (You can only copy text that appears in display mode.)

3 Press ENTER.

47 Parameters

This section describes the parameters that can be specified for a function in the SYSERR main menu. Any restrictions that apply to the use of parameters with a particular function are described in the section ***Functions***.

Message Type

Specifies the type of message to be processed. The table below lists the message types available:

Type	Explanation
NS	Natural system short messages
NL	Natural system long messages
US	User-defined short messages
UL	User-defined long messages

Library

Specifies the library for which messages are to be created or maintained. The specification of a library is not required when accessing Natural system messages (Message types NS and NL); any input values in the **Library** field are ignored.

Message Number

Specifies the first and last number of a message range. The maximum message number for a library and language is 9999. The message number 0000 is not allowed. To specify only one message number, either enter the number of the message in the left **Message number** field and clear the right field, or enter the number in both fields.

Language Codes

Specifies a maximum of 9 from 60 language codes available. The language codes are single alphanumeric characters in the ranges 1 - 9, A - Z and a - y. To view or select language codes, enter a question mark (?) in the first position of the **Language codes** field and press ENTER. For more information, see the system variable *LANGUAGE in the *System Variables* documentation.

48 Direct Commands

From the SYSERR main menu, you can execute the following commands by entering them in the command line:

Command	Explanation	
`EXPORT`	Exports a message file and converts it into a text file. Note that you always need to specify the full path of a file. For further information on file formats and **generating message and text files**, see the relevant section.	
	From (application)	The message file/application (library) from which the text file will be generated. User-defined messages: The default values entered in the export window are taken from the values entered in the fields **Message type**, **Library** and **Language codes** in the SYSERR main menu. The message file generated contains all messages, regardless of any range specifications. Natural system messages: Modify the default directory path (if required) and the file name. The file name must be `N`*nn*`L`*mmmm*`.MSG`.
	To (text file)	The name of the text file that will be generated automatically.

Command	Explanation
`IMPORT`	Imports a text file and converts it into a message file. Note that you always need to specify the full path of a file. For further information on file formats and generating message and text files, see the relevant section. **From (text file)**: The name of the text file from which the message file will be generated. **To (application)**: The message file/application (library) into which the text file will be generated. User-defined messages: The default values entered in the import window are taken from the values entered in the fields **Message type**, **Library** and **Language codes** in the SYSERR main menu. The message file generated contains all messages, regardless of any range specifications. Natural system messages: The file name must be `N`*`nn`*`L`*`mmmm`*`.MSG`.
`LAYOUT`	Specifies valid message ranges to categorize messages. Overlapping of ranges is possible. A new message can only be added if its number is within the range specified in the layout.
`NEXT`	Searches for the next free message number within the message number range specified. Free means that this message number is available and has not yet been assigned to a message file in any language.
`NEXTTAB`	Same as `NEXT`, but returns a list of message numbers from which you can select a number.
`RESTART`	Re-initializes SYSERR (and its default values) without leaving the utility.
`SAMPLE`	Invokes the **Edit SAMPLE message** window where you create or modify a sample message to be used as a master for creating new short messages. To create or modify a sample message, proceed as follows: ■ In the editor area of the **Edit SAMPLE message** window, type in the message text required or modify the existing text. If you enter the string `0000` (combined with text or not), the string `0000` is replaced by the number of the new message when copying the message. See also Step 4 of *Adding Messages* in the section *Functions*. ■ In the **Read or Write sample** field, enter a `W` to save your entries. ■ In the **Library** field, enter the name of the library for which the sample message is to be used. If you leave the **Library** field blank, the sample applies to Natural system messages. ■ Press PF3 to exit the **Edit SAMPLE message** window. You can define one sample message for each language and library.

Command	Explanation
`SHIFT`	If activated, automatically shifts the text of a short message to the left margin when confirming a modification or adding a new message.
`TRACE`	Counts the number of database accesses. When the message number specified has been reached, a window is displayed. The default number is `900`. If set to `0`, the trace facility is shut off. The commands `TRACE ON` and `TRACE OFF` can be entered directly in the command line. `TRACE ON` sets the access counter to `900`; `TRACE OFF` sets the access counter to `0`.
`USEREXIT`	Invokes the USEREXIT program in the Natural system library SYSERR.

49 Upper Case Conversion - ERRUPPER

Natural system messages are provided in lower case. If your terminals cannot display lower case characters correctly, convert the messages from lower to upper case by executing the program ERRUPPER in the Natural system library SYSERR.

However, once the messages have been converted to upper case, you cannot convert them back to lower case. To recover lower case messages, you have two options:

- Reload the messages by using the Object Handler.
- Unload the lower case messages to a free language code by using the Object Handler before conversion so that a backup always exists.

For detailed information, see the *Object Handler* documentation.

50 Replacing Characters - ERRCHAR

If your terminal does not display certain characters correctly, it is possible to search for these characters and replace them by new characters of your choice. This is done by executing the program ERRCHAR in the Natural system library SYSERR. However, it is only possible to replace characters in Natural system short messages. When using ERRCHAR, you scan for a specific character and replace the hexadecimal code that represents this character with another hexadecimal code.

After executing the program ERRCHAR, the **ERRCHAR** menu is displayed with the following functions:

- Scan for a given character
- Scan and Replace characters
- Display one message in hexadecimal format
- ASCII character table for your terminal
- Translate using character set ERRCSET

The following input fields are provided in the **ERRCHAR** menu:

Field	Explanation
Message Number	The range of messages to be included in the search or search/replace operation.
Language Code	The language code of Natural system short messages to be included in the search or search/replace operation.
Scan Value	The hexadecimal value to be scanned for.
Replace Value	The hexadecimal value to replace all scan values found. Use the function **ASCII character table for your terminal** to determine which characters your terminal can represent.

51 Generating Message and Text Files

You can create messages as text files in any environment outside Natural and convert them into message files to be maintained with the SYSERR utility. Message files are created and maintained with the import and export functions of the SYSERR utility.

Message files are created in a platform-independent format, which is portable across any Natural-supported UNIX, OpenVMS and Windows platforms. For example, a message file created in a Natural for Windows environment, can be copied onto a UNIX or an OpenVMS platform without manual conversion; the necessary endian conversion is performed by Natural. For further information, see *Portable Natural System Files* in the *Operations* documentation and *Transferring Natural Generated Programs* in the *Programming Guide*.

Storing a Message File

The message files must be stored with the file extension .MSG in the Natural Err directories.

User-defined message files are stored in the Err subdirectory of the library in the FNAT or FUSER system file from which the application is executed, the steplib, or the SYSTEM library.

For Natural system messages, the message files must be stored in the Err subdirectory in the Natural root directory. Natural system messages are stored in eight message files.

Creating a Text File

For Natural system or user-defined messages, the import function of the SYSERR utility generates a message file from one text file.

To create such a text file, you must use a specific layout, as shown in the following example:

Example:

```
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.
```

Explanation:

`NAT` or *`library-ID`*	The prefix of the message number to be displayed with the message. The default prefix is `NAT` for Natural system messages and the library ID for user-defined messages.
`0010`	The four-digit starting number of a range of messages.
`0100`	The four-digit ending number of a range of messages. All message numbers that are defined in this text file must be within this range.

`0010E`	`NO MESSAGE TEXT DEFINED!`
	
	This is the short message for message number 0010. The `E` is mandatory and means error. This message will be issued with the following Natural statement:
	
	`REINPUT *0010`
	
	Explanatory long messages must be placed immediately below this short message; each of these additional lines must start with a hash/number (#) sign. Up to 20 additional lines of long message text are allowed for each short message.

Generating a Message File

The SYSERR utility provides the option to generate a message file from a text file.

For user-defined messages, one output message file can be created in one language for each library. Each message file must be stored in the Err subdirectory of that library.

Naming Conventions

For user-defined messages, the name of the message file must be:

```
NnnAPMSL.MSG
```

where *`nn`* is the language code (`01` - `60`), for example `01` for English.

For Natural system messages, the name of the message file must be:

```
NnnLmmmm.MSG
```

where *nn* is the language code to be used and *mmmm* the starting number of the message range. The ranges of message numbers are fixed, as defined during Natural system installation, for example:

`N01L0000`	Messages 1 - 1999
`N01L2000`	Messages 2000 - 2999

To generate a message file

1. Enter the `IMPORT` command of the SYSERR utility.

   The **Import Text File to Message File** window is displayed.
2. In the **From** input field, specify the name of the input text file from which all information is to be read. The full path name of the file must be specified. In the **To** input fields, specify the language and the library of the message file to be generated.

Recreating a Text File

The SYSERR utility provides the option to recreate a text file for message text maintenance. This is done by reconverting a messages file into a text file.

To recreate a message text file

1. Enter the `EXPORT` command of the SYSERR utility.

   The **Export Text File from Message File** window is displayed.
2. In the **From** input fields, specify the language and the library of the message(s) to be used as input. In the **To** input field, specify the name of the text file to be created. The text file created will have the same format as an input text file.

52 Managing Messages in Different Libraries

You can transfer messages between different libraries by using either the `EXPORT` and `IMPORT` commands as described in *Direct Commands*, or the *Object Handler* as described in the relevant documentation.

You can also use the Object Handler to move, rename, find, list or delete messages in different libraries.

53 Application Programming Interface USR0020P

The application programming interface USR0020P in the Natural system library SYSEXT is provided to read messages from the FNAT or FUSER system file. Thus, it is possible, for example, to have long messages displayed in an application (as part of your own user-defined help system) without having to use the Natural system library SYSERR.

Log on to the Natural system library SYSEXT and, in the command line, enter the command `MENU`. In the list provided, mark the program USR0020P with a question mark (?). A window is then displayed, in which you can select the function to be executed for the program. If you enter an `I`, further information on the use of USR0020P is displayed.

VII SYSEXT Utility - Natural Application Programming Interfaces

54 SYSEXT Utility - Natural Application Programming Interfaces

The utility SYSEXT is used to locate and test Natural Application Programming Interfaces (APIs) contained in the current system library SYSEXT.

A Natural API is a Natural subprogram (cataloged object) that is used for accessing and possibly modifying data or performing services that are not accessible by Natural statements. Natural APIs refer to Natural, a subcomponent or a subproduct.

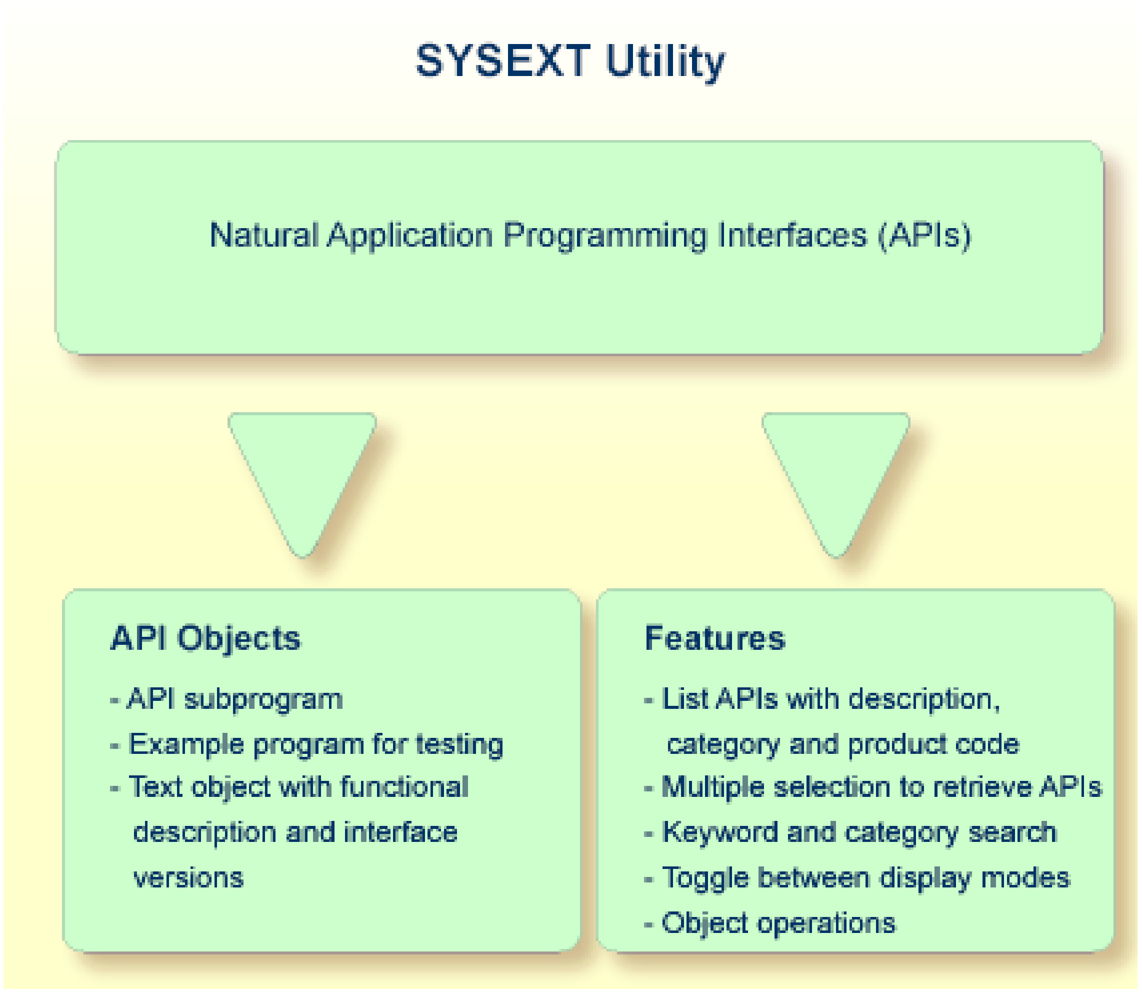
Related Topics:

- *Application Programming Interfaces - Natural Security* documentation

Introduction to SYSEXT

For each Natural API, the utility SYSEXT provides a functional description, one example program, one category and API-specific keywords.

The following diagram is an overview of the Natural objects and major features SYSEXT provides:


SYSEXT Utility
Natural Application Programming Interfaces (APIs)
API Objects
- API subprogram
- Example program for testing
- Text object with functional description and interface versions
Features
- List APIs with description, category and product code
- Multiple selection to retrieve APIs
- Keyword and category search
- Toggle between display modes
- Object operations

Objects Provided for Natural APIs

The types of Natural object typically provided for each Natural API are listed in the following section. Additional objects that might exist for a particular API are not covered.

All API-related objects are contained in the library SYSEXT on the system file FNAT.

In the following table, *nnnn* denotes the 4-digit number to identify the API as well as the corresponding example program and text object.

Object Name	Explanation
USR*nnnn*N	The API subprogram (cataloged object) that performs the designated function.
USR*nnnn*P	An example program (source object) that can be used to test the effect of the API. The example program invokes the corresponding subprogram USR*nnnn*N.
USR*nnnn*T	A text object listing a short and a long description, and information on usage, keywords, category and **interface versions**.

Object Name	Explanation
	You can display a text object by using the line command `T` as described in ***Line Commands***.

For some APIs, copycodes are available which provide functions related to the API. The copycodes are named `USRnnnnX`, where `X` is an identification character (such as "Z", "Y" etc.).

Invoking and Terminating SYSEXT

The SYSEXT utility is invoked by the system command `SYSEXT` which is described by the following syntax diagram:

```
SYSEXT [ ALL     ] [ FIRST  ] [ DESCENDING ]
       [ CURRENT ] [ SECOND ] [ ASCENDING  ]
```

The table below gives a description of the parameters:

Parameter	Description
ALL	List all APIs (default).
CURRENT	List only current APIs, that is all unique APIs and the current version of APIs with interface versions (with keyword +CURRENT-VERSION). See ***Interface Versions***.
FIRST	Display first menu with product code items (default), see **below** for an example.
SECOND	Display second menu with category items, see **below** for an example.
ASCENDING	Display APIs in ascending order (default).
DESCENDING	Display APIs in descending order.

To invoke SYSEXT

- Enter the following system command:

  ```
  SYSEXT
  ```

  A menu similar to the example below appears with a list of all available Natural APIs displayed according to the first menu:

```
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
```

You can invoke SYSEXT with any parameters, for example to display the second menu at startup:

```
SYSEXT SECOND
```

This results in a menu similar to the one below:

```
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
```

As an alternative, you can invoke SYSEXT without parameters and then adjust the resulting menu, see ***PF Keys***.

To terminate SYSEXT

- On the SYSEXT utility menu, press PF3 or PF12.

  Or:

  In the command line, enter a period (`.`) or enter `EXIT`.

Using the SYSEXT Utility

This section covers the following topics:

- Utility Commands

Elements of the SYSEXT Utility Menu

The SYSEXT utility menu provides a list of APIs. For each API, there is an API name (**Interface**), a description (**Description**), and, depending on the menu chosen, a classification according to product code (**Prod**) or category (**Category**). Except for the leftmost column (**Cmd**), each column is headed by a selection field that allows you to enter selection criteria.

You can also use the search field **Category** at the bottom of the menu to search for available categories.

The input field **Keywords** can either be used as search field to retrieve available keywords, or as selection field to retrieve APIs.

You can also specify selection criteria on multiple selection fields. For example, you can select in one step APIs beginning with `USR4`, and with keyword `PF-KEY` and category `NATURAL ENVIRONMENT`.

A detailed description of all menu elements and their usage is given below:

Element	Explanation	Usage
Cmd	The input field for a line command to be executed on a text object or an example program: see ***Line Commands***.	Enter a line command, example: `K` List all keywords relevant to the specified API.
Interface	The name of the API subprogram. APIs with interface versions are displayed intensified.	Enter an asterisk (*) , or a prefix to be delimited by an asterisk, example: `USR4*` List all APIs with prefix `USR4`.
Description	A brief description of the purpose of the API.	Enter a string, example: `default` List all APIs with a description containing the string `default` or `Default`. `*` List all APIs with a description that contains the string *.
Prod	The product code of Natural (`NAT`) or a Natural add-on product affected by the API. Available product codes: `NAT` = Natural, , `NDV` = Natural Development Server, `PRD` = Predict, `RPC` = Natural RPC (Remote Procedure Call). Product codes other than `NAT` are displayed intensified.	Enter a name, or a prefix to be delimited by an asterisk (*), example: `N*` List all APIs with a product code beginning with `N`.

Element	Explanation	Usage
	Only visible on the first menu.	
Category (selection field)	The category by which the API is classified according to its functional area or purpose. An API can only have one category. Only visible on the second menu and positioned above the list of APIs.	Enter a name, or a prefix to be delimited by an asterisk (*), example: `NATURAL OBJECTS` List all APIs with category `NATURAL OBJECTS`.
Category (search field)	List all categories. Positioned below the list of APIs.	See *Category Search*.
Keyword	List all keywords or enter a keyword as selection criterion.	See *Keyword Search* or *Keyword Selection*.
Command	Command line to enter utility commands.	Enter a utility command. See *Utility Commands*.

Category Search

- Enter an asterisk (*), or a prefix in the category search field optionally delimited by an asterisk, for example:

  ```
  N*
  ```

  As a result, a menu similar to the example below appears:

  ```
           Search for Categories

    Mark Category
    ---- N*____________________________
      _  NATURAL ENVIRONMENT
      _  NATURAL OBJECTS
  ```

  To select a specific category as selection criterion, enter any character in the **Mark** column. As a result, a list of all APIs with this selection criterion is displayed.

Keyword Search

- Enter an asterisk (*), or a prefix delimited by an asterisk, for example:

  ```
  PF*
  ```

  A menu similar to the example below appears with a separate window displaying keywords starting with `PF`.

```
          Search for Keywords

Mark Keyword
---- PF*__________________
  _  PF-KEY
  _  PF-KEY LINE
```

To select a specific keyword as selection criterion, enter any character in the **Mark** column. As a result, a list of all APIs with this selection criterion is displayed.

Keyword Selection

- Enter a keyword in full, for example:

```
PF-KEY
```

As a result, a list of all APIs with keyword `PF-KEY` is displayed.

Note: A non-trailing asterisk is interpreted literally, for example entering `*LANGUAGE` results in a list of APIs with keyword `*LANGUAGE`.

PF Keys

You can use the following PF keys:

PF Key	Name	Function
PF1	**Help**	Display context-sensitive help. There is a specific help text for each input field. In other contexts, for example the command line, a general help text is displayed.
PF2	**Reset**	Clear all selection fields and readjust the list of APIs.
PF3	**Exit**	Exit the SYSEXT utility, or the current menu or window.
PF4	**Asc/Desc**	Toggle between ascending (**Asc**) and descending (**Desc**) order of APIs.
PF5	**All/Curr**	Toggle between menus displaying all APIs (**All**) and menus displaying only current APIs (**Curr**). See *Interface Versions*.
PF6	**--**	Scroll to the beginning of the list.
PF7	**-**	Scroll one page up.
PF8	**+**	Scroll one page down.
PF9	**++**	Scroll to the end of the list.
PF10	**<**	Shift to first menu.
PF11	**>**	Shift to second menu.
PF12	**Canc**	Exit the SYSEXT utility or the current menu or window.

Line Commands

Line commands are used to perform object operations. You can enter a line command in the **Cmd** column next to the API required. For a list of valid line commands, enter a question mark (`?`) or press PF1.

The following line commands are available:

Line Command	Function
`K`	List keywords relevant to the specified API.
`T`	List text object USR*nnnn*T for a description of the corresponding API. The description comprises purpose, function and calling conventions of the API, relevant keywords and its category.
`L`	List example program USR*nnnn*P.
`E`	Edit example program USR*nnnn*P. **Note:** This command is not available if the Natural program, data area and map editor are disabled in your environment. For more information, see *Disabled Natural Editors* in the *Editors* documentation.
`R`	Run example program USR*nnnn*P.
`X`	Execute example program USR*nnnn*P.
`.`	Exit the SYSEXT utility.

Utility Commands

This section covers the following utility commands to be entered in the command line:

- **EXIT**
  Exit the SYSEXT utility.
- **REFRESH**
  Update API information using data from the objects contained in the current library SYSEXT. The `REFRESH` command is only required if an API description or a keyword has been modified or if a text object has been added or removed.

  Upon successful completion, there is a confirmation that the text modules EXT-XML1 and EXT-XML2 have been generated in library SYSEXT.

  Note: Do *not* modify the source objects EXT-XML1 and EXT-XML2. They are required for configuring the SYSEXT utility and intended for Software AG internal use only.

Interface Versions

Interface versions can be seen as a collection of APIs with (almost) the same functionality but with differently extended parameter specifications. Thus, they cover a development cycle to be kept explicit for sake of compatibility (of later versions with earlier versions).

If an API has interface versions, they are displayed in the corresponding text object USR*nnnn*T. Interface versions are ordered within a list according to the version they belong to. The rightmost element belongs to the current version. This status is expressed by the reserved keyword `+CURRENT-VERSION`. All other elements belong to a previous version and are marked with the reserved keyword `+PREVIOUS-VERSION`. APIs without interface versions are called unique.

APIs with interface versions are displayed intensified on the menu.

The list of all current APIs (see **PF5**) consists of all unique APIs and the current version of APIs with interface versions (with keyword `+CURRENT-VERSION`).

Reserved Keywords

Reserved keywords refer to meta information on APIs, for example the Natural version in which an API has been added. Reserved keywords always start with a plus sign (+). See the table below for a description:

Reserved Keyword	Description
+CURRENT-VERSION	The current version of an API with interface versions (see ***Interface Versions***).
+PREVIOUS-VERSION	A previous version of an API with interface versions (see ***Interface Versions***).
+NEW-*PROD*-*version*	An API that has been added to a specific product in a specific version. For example, `+NEW-NAT-6.3.11` refers to an API that has been added to the product Natural in version `6.3.11`.
+MOD-*PROD*-*version*	An API that belongs to a specific product and has been modified in a specific version. For example, `+MOD-NAT-6.3.12` refers to an API that belongs to product Natural and has been modified in version `6.3.12`.

Using a Natural API

If you want to use a Natural API contained in the system library SYSEXT, perform one of the following steps:

- Define the system library SYSEXT in the system file FNAT as a steplib library for the user library that contains the Natural objects that use this API. Thus, no API-specific actions are required when upgrading your Natural version.
- Copy the required API to the system library SYSTEM in the system file FNAT. Thus, you only need to check a single library for APIs when upgrading your Natural version.
- Copy the required API to the system library SYSTEM in the system file FUSER (not recommended).
- Copy the required API to the user library (or one of its steplibs) in the system file FUSER which contains the Natural objects that use this API (not recommended).

An API can only be used in the Natural version with which it is delivered. It is strongly recommended to store the APIs only in the FNAT system file. This will ensure that the right version is always executed.

To make use of an interface

1. In the calling program, use the `DEFINE DATA` statement to specify the parameters listed in the text object `USR`*nnnn*`T` of that API. In the example program `USR`*nnnn*`P`, the parameters are defined within the `DEFINE DATA LOCAL` statement. Alternatively, you can specify the parameters outside the calling program in a separate LDA (Local Data Area) or PDA (Parameter Data Area), with a `DEFINE DATA LOCAL USING` statement referencing that data area.
2. Enter the following statement:

   ```
   CALLNAT 'USRnnnnN' parameters
   ```

   For further information, see the *CALLNAT* statement in the *Statements* documentation.

Note: Non-standard usage is always documented in the respective text object `USR`*nnnn*`T`.

If you want to use a Natural copycode contained in the system library SYSEXT, perform the following step:

- Copy the required copycode to the user library in the system file FUSER which contains the Natural objects that use this copycode.

To make use of a copycode

1. In the calling program, use the `INCLUDE` statement to specify the parameters listed in the text object `USR`*nnnn*`T` of that API or in the copycode itself. In the example program `USR`*nnnn*`P`, the parameters are defined within the `DEFINE DATA LOCAL` statement. Alternatively, you can specify the parameters outside the calling program in a separate LDA (Local Data Area) or PDA (Parameter Data Area), with a `DEFINE DATA LOCAL USING` statement referencing that data area.
2. Some copycodes require additional data definitions. These are described in the text object `USR`*nnnn*`T` of the API and in the copycode itself.
3. Enter the following statement:

   ```
   INCLUDE USRnnnnX 'parameter'...
   ```

   For further information, see the *INCLUDE* statement in the *Statements* documentation.

Note: Non-standard usage is always documented in the respective text object `USR`*nnnn*`T`.

VIII SYSEXV Utility

55 SYSEXV Utility

The utility SYSEXV utility provides example programs that demonstrate the use of Natural features introduced in the current or a previous version of Natural.

All example programs are available as source objects. You obtain detailed functionality descriptions when you execute the programs.

Executing Example Programs

To execute an example program

1. Enter the following system command:

   ```
   SYSEXV
   ```

   A **SYSEXV** menu is displayed which lists Natural version(s) and categories.
2. Select a version or a category.

   A commented list of Natural features and corresponding programs is displayed.
3. Choose the required feature/program.

   The program is executed.

PF Keys

You can use the following PF keys:

PF Key	Name	Function
PF1	Help	Display the help text.
PF3	Exit	Exit the SYSEXV utility, or the current menu or window.
PF5	All	Select all items of the current page.
PF7	Prev	Scroll one page up.
PF8	Next	Scroll one page down.
PF12	Canc	Exit the SYSEXV utility.

You can also enter the name of a PF key in the command line, to perform the corresponding function.

Terminating an Example Program or the SYSEXV Utility

To terminate an example program

- Press PF3 (Exit).

To terminate the SYSEXV utility

- In the SYSEXV utility menu, press PF3 or PF12, or

  enter a period (.) in any line, or

  enter `EXIT` or a period (.) in the command line.

  Or:

  In an example program, press PF12 (Canc).

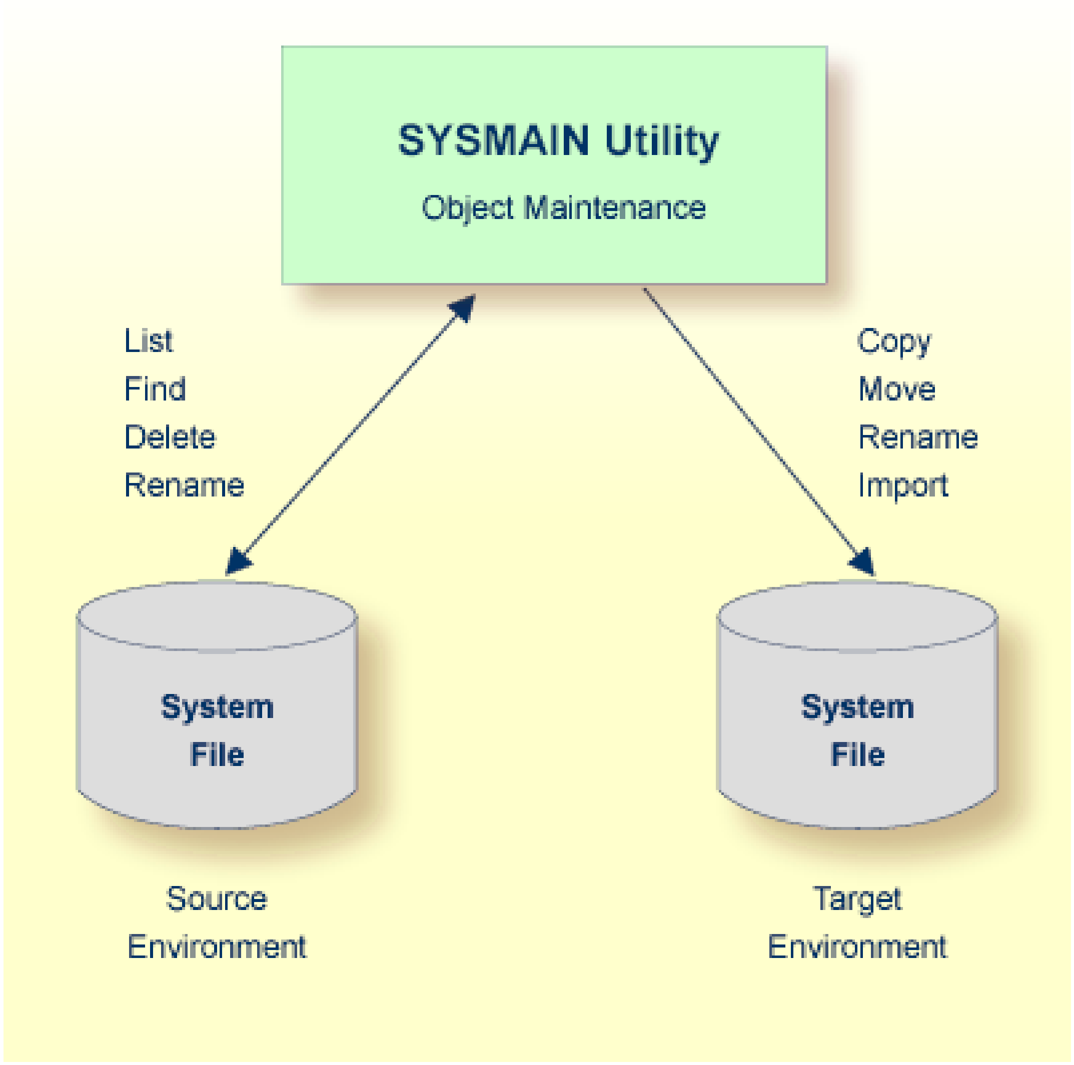
IX SYSMAIN Utility - Object Maintenance

The SYSMAIN utility is used to perform object maintenance functions such as copy, move, replace and delete.

56 General Information

The SYSMAIN utility is used to maintain Natural objects in a Natural environment.

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



The SYSMAIN utility copies, moves or imports objects from a source environment to a target environment and performs object operations (for example, delete) in a source environment. A source or a target environment is a library in an FNAT or FUSER system file.

The SYSMAIN utility functions are available online and in batch mode.

The Natural objects that can be maintained with the SYSMAIN utility are programming objects and data definition modules (DDMs). Programming objects comprise the following object types: program, subprogram, subroutine, map, text, data area (local, parameter and global), copycode, helproutine, dialog, class, function and adapter.

57 Invoking and Terminating SYSMAIN

This section describes how to invoke and terminate the SYSMAIN utility either in menu mode or by using a subprogram.

Invoking SYSMAIN

The following instructions describe the methods of invoking the SYSMAIN utility by using menu functions or a subprogram.

To invoke SYSMAIN from the Natural main menu

1 Select **Services** and then select **SYSMAIN** or enter S.

Or:

Select **Direct** and, in the **Direct Command** window, enter the following system command:

```
SYSMAIN
```

A SYSMAIN menu similar to the example below appears:

```

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

```

The current setting of the system variable `*LIBRARY-ID` is passed to SYSMAIN and used as the default source library for processing objects (in the example above: `SYSTEM`).

2 Select a function or enter the one-letter code that corresponds to the function required (for example, `L` for **List**).

Except for **Import** and **Terminate**, a selection window similar to the example below appears:

```
+------- LIST --------+
¦X Programming Objects¦
¦  Views (DDM)        ¦
+---------------------+ ↩
```

3 If you want to process any object types except DDMs, mark **Programming Objects** by choosing UP ARROW if required (marked by default).

Or:

If you want to process DDMs only, mark **Views (DDM)** by choosing DOWN ARROW.

To invoke SYSMAIN with a subprogram in online or in batch mode

- Use a Natural program with a `CALLNAT` statement that invokes and executes the MAINUSER subprogram, which issues commands to SYSMAIN. See also ***Using SYSMAIN with Subprogram***.

Terminating SYSMAIN

To terminate SYSMAIN

- From the SYSMAIN menu, choose **Terminate**.

  Or:

  Choose ESC repeatedly.

  Or:

  When using the MAINUSER subprogram, in the command string, specify a period (.). See also ***Using SYSMAIN with Subprogram***.

58 Description of Functions

The functions SYSMAIN provides in menu mode or as a command in the MAINUSER subprogram are described in the following table. For each function provided in menu mode, there is a corresponding command with the same name. Exception: **Terminate**.

Function/Command	Explanation
List	Lists a Natural object or a range of objects for a specified library and provides the option to view the sources of these objects. In addition, this function can be used to produce a list of all libraries contained in the current Natural system environment.
Find	Locates and lists a Natural object or a range of objects in a single library or a range of libraries and provides the option to view the sources of these objects.
Copy	Copies a Natural object from a source library to a target library. The object remains unchanged in the source library. If the target library already contains an object with the same name as the object to be copied, the specified object is not copied. You can use the replace option (see *Using the Replace Option*) if you want to overwrite an object in the target library.
Move	Transfers a Natural object from a source library to a target library. The object is deleted from the source library and added to the target library. If the target library already contains an object with the same name as the object to be moved, the specified object is not moved. You can use the replace option (see *Using the Replace Option*) if you want to overwrite the object in the target library.
Delete	Deletes a Natural object from a source library. In menu mode, you can use the confirm option to confirm the deletion or cancel it.
Rename	Gives an object a new name using either of the following methods: 1. Rename the object or a range of objects in the source library. 2. Copy the object or a range of objects from the source to the target library and rename them in the target library.

Function/Command	Explanation
	If the library already contains an object with the same name as the object to be renamed, the original object is not renamed. You can use the replace option (see also ***Using the Replace Option***) if you want to overwrite the original object.
Import	The import function is used to copy objects (files) from an external source to a Natural library. Alternatively, you can use the unload and load functions of the **Object Handler** (see the relevant documentation). The following must be considered before attempting to import objects: **FILEDIR.SAG:** *FILEDIR.SAG* contains internal library information required by Natural such as the object name, programming mode (structured or reporting), object kind (source object and/or cataloged object) and user ID. When you import objects, the file directory *FILEDIR.SAG* of the target library is automatically updated to contain information on the newly imported objects. Be aware that Natural will *not* update the file directory *FILEDIR.SAG* if you use a non-Natural function or facility to copy objects to a Natural library. As a result, you cannot access the objects contained in this library. The objects to be imported with SYSMAIN must have been created with Natural.
Terminate	Terminates the SYSMAIN utility.

Instructions for executing a SYSMAIN function with either menu functions or commands issued to SYSMAIN with the MAINUSER subprogram are provided in the relevant sections of this documentation. Because of the similarities between the processing of programming objects and DDMs, the instructions for the menu functions only refer to the selection of programming objects.

59 Listing and Finding Objects

This section provides instructions for using the list and find functions in menu mode.

To find or list single or multiple objects

1. In the SYSMAIN menu, select **List** (or enter `L`), or **Find** (or enter `F`) and then select **Programming Objects**.

   An object-specification window similar to the example below appears:

```
+--------------------- LIST ---------------------+
¦                                                ¦
¦ OBJECT:  *                                     ¦
¦ LIBRARY: SYSTEM          CODE: X (S)ource      ¦
¦ DBID:     99   FNR:  51        X (C)ataloged   ¦
¦ USER ID:                                       ¦
¦ DATE:        -  -     :                        ¦
+------------------------------------------------+
```

2. Enter the selection criteria required to select the objects you want to process. For explanations of the fields contained in this window and valid input values, see *Using the Fields in an Object-Specification Window*.
3. The following step applies to the list function only:

   - If you entered a name range in the **LIBRARY** field, all libraries within the range specified and available in the system file specified are listed for selection. In the example below, `TEST*` was entered to list all libraries whose names begin with TEST:

```
+-------- LIST --------+
¦  Library   dbid/fnr  ¦
¦ -------------------- ¦
¦  TEST       99/51    ¦
¦  TESTLIB1   99/51    ¦
¦  TESTLIB2   99/51    ¦
¦  TESTLIB3   99/51    ¦
¦ ____________________ ¦
¦ *** ENTER==>list *** ¦
¦ ***   ESC==>exit *** ¦
+----------------------+
```

- Choose UP ARROW or DOWN ARROW to scroll up or down the list and select the library required with ENTER.

  Or:
  Choose ESC if you want to exit the window without any action.

When you selected a library, the object-specification screen appears with this library entered in the **LIBRARY** field.

4 When you have finished entering selection criteria in the object-specification window and confirm your entries with ENTER, an **OBJECT TYPE** window similar to the example below appears:

```
+---OBJECT TYPE---+
¦X ==> select ALL ¦
¦   Program       ¦
¦   Subroutine    ¦
¦   Copycode      ¦
¦   Map           ¦
¦   Text          ¦
¦   Helproutine   ¦
¦   Subprogram    ¦
¦   Global Data   ¦
¦   Local Data    ¦
¦   Parameter Data¦
¦   Dialog        ¦
¦   Class         ¦
¦   Function      ¦
¦   Adapter       ¦
+-----------------+
```

5 If you want to process objects of all types listed in this window, mark **select ALL** with an `X` (marked by default).

Or:

If you want to process only objects of one or more specific types, enter X in the input fields next to the types required and confirm with ENTER.

A result window appears, which lists a single object or all objects within the name range specified in the **OBJECT** field similar to the example below:

```
+---- 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 *** ¦
+-------------------------------------+
```

The window title indicates the number of objects found (in the example above: 10) and the library in which the search was made (in the example above: TESTLIB). For each object listed, the window displays the object name, the object type, the object kind available (S denotes source object, C denotes cataloged object) and the ID of the user who saved and/or cataloged the object. The object list is sorted in alphabetical order of object names.

The following applies to the find function only:

If you entered a name range in the **LIBRARY** field, a result window appears for each library where the objects requested are found. Choose ESC to open one result window after the other, in alphabetical order of the library names.

6 If you want to view the source code of an object and if a source object exists for this object, select the object required from the list and choose ENTER.

A **List** window appears, which displays the source code of the object selected similar to the example of program PGM3 shown below:

```
+-------------------------------- 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)                                                           ¦
+-----------------------------------------------------------------------------+
```

Choose ESC to exit this window.

60 Copying, Moving and Renaming Objects

This section provides instructions for using the copy, move and rename functions in menu mode.

To copy, move or rename single or multiple objects

1 In the SYSMAIN menu, select **Copy** (or enter `C`), **Move** (or enter `M`) or **Rename** (or enter `R`) and then select **Programming Objects**.

An object-specification window similar to the example below appears:

```
+--------------------- 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 In the **Source** section, enter all object selection criteria required for specifying the source environment. For explanations of the fields contained in this window and valid input values, see ***Using the Fields in an Object-Specification Window***.

3 Confirm your entries in the **Source** section with ENTER:

- If you entered single names in the **OBJECT** and **LIBRARY** fields and a matching object is found, skip to **Step 10**.

- If you entered a name range in the **LIBRARY** field, all libraries within the range specified are listed for selection (see the **example window** shown earlier).

4 Select the library required.

The object-specification screen appears with the selected library entered in the **LIBRARY** field of the **Source** section.

5 Choose ENTER to continue:

- If you entered a single name in the **OBJECT** field and a matching object is found, skip to **Step 10**.
- If you entered a name range in the **OBJECT** field, the **OBJECT TYPE** window appears (shown and described earlier).

6 Select one or more object types required.

7 If you use the copy or move function, a window similar to the example below appears:

```
+-------------------------------------------+
¦X Select the specified Object(s) for copy ¦
¦  Copy  ALL  specified Object(s)          ¦
+-------------------------------------------+
```

If you do not want to select all matching objects individually from a list and/or rename them in the target library, mark **Copy ALL specified Object(s)** by choosing DOWN ARROW, then skip to **Step 10**.

Or:
If you want to select all matching objects individually from a list and/or rename them in the target library, mark **Select the specified Object(s) for copy** by choosing UP ARROW if required (marked by default):

- If a single object is found, skip to **Step 10**.
- If multiple objects are found, a selection window appears that looks similar to the **result window** shown and explained earlier.

8 Select the objects you want to process by entering `X` in the input fields of the **X** column next to the objects required. If both the source object and the cataloged object exist for a selected object, you can process only the source object or the cataloged object by replacing the `S` or the `C` respectively in the **S/C** column with a blank character and choosing ENTER.

9 Once you have selected the objects required, a window similar to the one below appears.

Copy or move function:

```
+-------------------------------------------------+
¦X Copy selected Objects using   same    Object name ¦
¦  Copy selected Objects using different Object name ¦
+-------------------------------------------------+
```

- If you want to use the same object names for both the source and the target library, mark the first option by choosing UP ARROW if required (marked by default).
- Or:
  If you want to rename each object in the target library, mark the second option by choosing DOWN ARROW.

Rename function:

```
+----------------------------------------------+
¦X Rename every single Object with a new name    ¦
¦  Rename all selected Objects with one new name* ¦
+----------------------------------------------+
```

- If you want to rename each object individually, mark the first option by choosing UP ARROW if required (marked by default).
- Or:
  If you want to rename a range of objects with a new name range, mark the second option by choosing DOWN ARROW.

10 If the **XREF** option has been set to `Y` (Yes), a window similar to the example below appears:

```
+-------------------- XREF --------------------+
¦                                              ¦
¦          --- PREDICT (FDIC) Files ---        ¦
¦                                              ¦
¦    - Source -              - Target -        ¦
¦ DBID:         0         DBID:         0      ¦
¦ FNR:          0         FNR:          0      ¦
¦ PASSWORD:               PASSWORD:            ¦
¦ CIPHER:                 CIPHER:              ¦
+----------------------------------------------+
```

11 If required, replace the current database ID (DBID) and/or file number (FNR) and enter a password and a cipher code. For further information, see *XREF Considerations*.

12 The object-specification screen appears, which now looks similar to the example shown below:

```
+--------------------- 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                        ¦
+------------------------------------------------+
```

- If multiple objects were found, a message indicates the number of objects selected for processing (in the example above: `3`).
- If an object name range was specified in the **Source** section and the rename option (see **Step 9**) was *not* selected, the **OBJECT** field of the **Target** section has turned into a read-only field, which contains the same name range (in the example above: `PRO*`).

  If a single object was selected for processing, the name of this object is entered in the **OBJECT** field of the **Target** section. You can replace the name in this field.

  If an object name range was specified in the **Source** section and the rename option (see **Step 9**) was selected, the name of each object selected appears in the **OBJECT** field of the **Target** section field. You can then replace the name of each current object in this field.

  When using the rename function (see **Step 9**), you can also replace multiple objects by specifying name ranges as shown in the example below:

```
+-------------------- 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                        ¦
+------------------------------------------------+
```

  In the example above, all objects whose names begin with PRO are replaced with objects whose names begin with PGM.

13 Confirm your entries in the **Target** section with ENTER.

If the target library already contains objects with the same names specified and if the **REPLACE** option is set to `N`, for each object found a window similar to the one below appears:

```
+-----------------------------------------------------------------------------+
¦ Object(s) already exist, do you want to overwrite source ?   Y/N            ¦
+-----------------------------------------------------------------------------+
```

- Enter `Y` (Yes) to confirm the object replacement.
- Or:
  Enter `N` (No) to reject the object replacement.

See also ***Using the Replace Option***.

14 When the copy, move or rename function completed successfully, an appropriate confirmation message appears in the object-specification window.

61 Deleting Objects

This section provides instructions for using the delete function in menu mode.

To delete single or multiple objects

1 In the SYSMAIN menu, select **Delete** or enter `D` and then select **Programming Objects**.

An object-specification window similar to the example below appears:

```
+-------------------- DELETE --------------------+
¦                                                ¦
¦                                                ¦
¦ OBJECT:                                        ¦
¦ LIBRARY: SYSTEM            CODE: X (S)ource    ¦
¦ DBID:      99   FNR:  51         X (C)ataloged ¦
¦ USER ID:                   XREF:    n          ¦
¦ DATE:         -  -     :   CONFIRM: y          ¦
+------------------------------------------------+
```

2 Enter the selection criteria required for specifying the objects to be deleted. For explanations of the fields contained in this window and valid input values, see *Using the Fields in an Object-Specification Window*.

Note: Since XRef data is always deleted when you delete an object, you can ignore the **XREF** setting.

3 Confirm your object specifications with ENTER:

- If you entered single names in the **OBJECT** and **LIBRARY** fields and a matching object is found, skip to **Step 9**.

- If you entered a name range in the **LIBRARY** field, all libraries within the range specified are listed for selection (see the **example window** shown earlier).

4 Select the library required.

The object-specification screen appears with the selected library entered in the **LIBRARY** field.

5 Choose ENTER to continue:

- If you entered a single name in the **OBJECT** field and a matching object is found, skip to **Step 9**.
- If you entered a name range in the **OBJECT** field, the **OBJECT TYPE** window appears (shown and described earlier).

6 Select one or more object types required.

A window similar to the example below appears:

```
+---------------------------------------------+
¦X Select the specified Object(s) for delete ¦
¦  Delete ALL specified Object(s)            ¦
+---------------------------------------------+
```

7 If you do not want to select all matching objects individually from a list, mark **Delete ALL specified Object(s)** by choosing DOWN ARROW, then skip to **Step 9**.

Or:

If you want to select all matching objects individually from a list, mark **Select the specified Object(s) for delete** by choosing UP ARROW if required (marked by default):

- If a single object is found, skip to **Step 9**.
- If multiple objects are found, a selection window appears that looks similar to the **result window** shown and explained earlier.

8 Select the objects you want to delete.

9 If the **CONFIRM** option is set to `Y` (Yes; this is the default setting), a window similar to the one below appears :

```
+---------------------------------------------------------------------------+
¦ Are you sure you want to delete this Object ?   Y/N                       ¦
+---------------------------------------------------------------------------+
```

- Enter `Y` (Yes) to confirm each object replacement.
- Or:
  Enter `N` (No) to reject each object replacement.

10 When the delete function completed successfully, an appropriate confirmation message appears in the object-specification window.

62 Importing Objects

This section provides instructions for importing external objects (files) in menu mode.

For the points that must be considered before importing objects, see the description of the **import** function.

To import single or multiple objects

1 In the SYSMAIN menu, select **Import** or enter `I`.

An object-specification window similar to the example below appears:

```
+------------------------ IMPORT -------------------------+
¦                                                         ¦
¦                        - Source -                       ¦
¦                                                         ¦
¦ PATH:                                                   ¦
¦ OBJECT:              CODE: X (S)ource                   ¦
¦                            X (C)ataloged                ¦
¦                                                         ¦
¦                        - Target -                       ¦
¦                                                         ¦
¦ LIBRARY:             USER ID:          REPLACE:         ¦
¦ DBID:                MODE:                              ¦
¦ FNR:                                                    ¦
+---------------------------------------------------------+
```

2 In the **Source** section, enter all object selection criteria required for specifying the source environment. For explanations of the fields contained in this window section and valid input values, see ***Using the Fields in an Object-Specification Window***.

3 Confirm your entries with ENTER:

- If you entered a single name in the **OBJECT** field and a matching object is found, skip to **Step 5**.
- If you entered a name range in the **OBJECT** field, all objects within the range specified are listed similar to the example shown below:

```
+------- 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    ¦
+-----------------------------------+
```

  For explanations of the columns contained in this window, see the **result window** shown and explained earlier.

4 Select the objects you want to import.

The object-specification window appears, which indicates the number of objects selected.

5 In the **Target** section, enter all specifications required. For explanations of the fields contained in this window section and valid input values, see ***Using the Fields in an Object-Specification Window***.

If the target library already contains objects with the same names as the objects to be imported and if the **REPLACE** option is set to `N`, a confirmation window appears. Proceed as described earlier in **Step 13** of *Copying, Moving or Renaming Objects*.

6 When the import function completed successfully, an appropriate confirmation message similar to the example below appears:

```
+------------------------- 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 Using the Fields in an Object-Specification Window

This section describes the fields and input options provided in an object-specification window, in which you can specify selection criteria for the objects to be processed with a SYSMAIN function.

If a field only applies to a particular function, this is indicated by an appropriate remark.

Field	Explanation
OBJECT	The name of the object to be processed or a range of names. The default setting is asterisk (*) which means that all objects are selected for processing. For valid name ranges, see *Specifying a Range of Names*.
LIBRARY	The name of a source or a target library or a range of names. The source library contains the object to be processed. The target library is an existing or a new library to which the object is to be copied or moved, or where the object is renamed or imported. For valid name ranges, see *Specifying a Range of Names*.
DBID	The database ID of a source or a target library. The source database contains the library and system file where the object to be processed is stored. The target database contains the library and system file to which the object is to be copied or moved, or where the object is renamed. Valid database IDs are `1` to `65535`. If no value (or `0`) is specified, the current FUSER or FNAT system file is used.
FNR	The file number of a source or a target system file (FNAT or FUSER). The source file contains the library where the object to be processed is stored. The target file contains the library to which the object is to be copied or moved, or where the object is renamed or imported. Valid file numbers are `1` to `65535`. If no value (or `0`) is specified, the current FUSER or FNAT system file is used.
USER ID	Not applicable to the import function. The ID of the user who last saved and/or cataloged the object to be processed.
DATE	Not applicable to the import function. Selects all objects that were saved and/or cataloged on or after the date and/or time entered in these fields. By default, no date or time is entered. A start date must be specified in the following format: *YYYY-MM-DD* (*YYYY* = year, *MM* = month, *DD* = day). Example: `2007-01-31`. A start time must be specified in the following format: *HH*:*II* (*HH* = hours, *II* = minutes). Example: `09:15`.
CODE	Not applicable to the import function. Selects the object kind: **(S)ource** — The source (saved) object only.

Field	Explanation	
	(C)ataloged	The cataloged object only.
	By default, both the source object and the cataloged object are selected. Find or list function only: The source code of an object can only be displayed if a source object exists. If you select **(C)ataloged** or if the result window only contains cataloged objects, you cannot select any objects from this list to display their source codes.	
XREF	Not applicable to the find, list or import function. Indicates whether cross-reference (XRef) data stored on Predict system files is to be processed for programming objects (not applicable to DDMs). Possible input values are:	
	`N`	No. XRef data is not processed. This is the default setting.
	`Y`	Yes. All XRef data is processed.
	See also *XRef Considerations*.	
TYPE	Not applicable to the find or list function. A read-only field that indicates the object types as selected from the **OBJECT TYPE** window: The field contains either `all` indicating all object types, an object type such as `Program` (import function only) or one or more object-type codes such as `P` for program. For possible codes, see *TYPE Specification*.	
REPLACE	Not applicable to the find, list or delete function. Specifies whether the replace option is activated:	
	`Y`	An object is automatically replaced.
	`N`	An object is only replaced after prior confirmation. This is the default setting.
	See also *Using the Replace Option*.	
CONFIRM	Only applies to the delete function.	
	Indicates whether a confirmation window appears before the selected objects are deleted. Possible input values are:	

Field	Explanation	
	Y	A confirmation window appears, where you can enter Y to confirm the deletion or enter N to cancel it. This is the default setting.
	N	All objects are deleted immediately, without prior confirmation window. This is the default setting.
PATH	Only applies to the import function. The complete UNIX path name of the directory from which the import function is to be executed. The path name can start with a UNIX environment variable such as `$HOME`. When you choose ENTER, the environment variable is replaced by the full path name. If you want to import objects from the default path assigned to you at Natural session start, enter the following: `./`	
MODE	Only applies to the import function.	
	Specifies the Natural programming mode to be set for the programming object to be imported:	
	Structured	Structured mode is used.
	Report	Reporting mode is used.
	For further information, see *Natural Programming Modes* in the *Programming Guide*.	

Specifying a Range of Names

All SYSMAIN functions provide the option to specify either a name or a range of names for the libraries or the objects to be selected.

The valid asterisk (*) notations for name ranges are listed below where `value` denotes any combination of one or more characters:

Input	Objects or Libraries Selected
*	All objects or libraries.
`value`*	All objects or libraries with names that start with `value`. Example: `AB*` Selected: AB, AB1, ABC, ABEZ Not selected: AA1, ACB
`value`*`value`*	All objects or libraries that match `value` combined with one or two asterisks (*) in any order. Example: `A*C*` Selected: ABCZ, AXXCBBBZ, ANCZ

Input	Objects or Libraries Selected
	Not selected: ABDEZ, ACBBBZA

Renaming Multiple Objects

If you want to rename multiple objects, `value`* must be specified in both the source environment and the target environment; you cannot specify a single name for the source environment and a range of names for the target environment, or vice versa.

If `value`* is used, the number of characters before the asterisk (*) in the source environment determines the number of characters to be replaced. For example, if you specify `ABC` for the source environment and `WXYZ` for the target environment, each object in the target environment that starts with ABC will be replaced by an object name that starts with WXYZ. The remainder of each name (after the first four characters, in this example) is retained.

Using the Replace Option

If the target library already contains an object with the same name as the object to be copied, moved, renamed or imported, the specified object is not processed and processing continues with the next object. You can use the replace option to override this default feature and overwrite the object in the target library. If an object is replaced, it is also deleted from the Natural buffer pool; any existing cross-reference records are also deleted.

To activate or deactivate the replace option in menu mode

- In the **REPLACE** field of a SYSMAIN object-specification window:

  Enter `N` to activate the replace option.

  You are prompted to confirm each object replacement.

  Or:

  Enter `Y` to deactivate the replace option (this is the default setting).

  All objects are replaced without prior confirmation message.

To activate the replace option using a command in the MAINUSER subprogram

- In the command string, specify the keyword `REPLACE` as described in *Using SYSMAIN with Subprogram*.

64 Using SYSMAIN with Subprogram

The MAINUSER subprogram is an Application Programming Interface, which allows you to perform SYSMAIN utility functions from any user-written object (subroutine, program or subprogram) as an alternative to using SYSMAIN utility menus. Upon completion of the SYSMAIN function, the utility is terminated and control is returned to the object from which the request was issued. MAINUSER can be used in either online or batch mode. An example of a callable routine is the MAINCALL program, which is supplied in the SYSMAIN system library.

This section provides instructions for using MAINUSER and the syntax that applies when specifying commands for executing SYSMAIN utility functions.

Invoking and Executing MAINUSER

To invoke and execute MAINUSER

- Issue a `CALLNAT` statement that contains the following syntax elements:

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

  where the variable values denote the following parameters:

Parameter	Natural Data Format/Length	Explanation
command	A250	The command string to be executed by SYSMAIN: see ***Using Commands***.
error	N4	The return code issued by SYSMAIN at the end of processing to indicate a normal end of processing or an error.
message	A72	The message corresponding to the error given online.
library	A8	The name of the library containing the utility SYSMAIN; by default, this is the library SYSMAIN. (This parameter is provided for compatibility reasons only.)

Using Commands

SYSMAIN functions can be executed by using commands issued as a parameter of the MAINUSER subprogram.

A *command* consists of keywords and variable values. For each SYSMAIN function to be performed, the keywords and variable values are shown in the corresponding syntax diagrams below and explained in the section ***Keywords and Variables in Commands***. The symbols in the syntax diagrams correspond to the syntax symbols used for system commands. These symbols are explained in *System Command Syntax* in the *System Commands* documentation.

The sequence of the command syntax is not completely fixed. The following rules apply:

- SYSMAIN function, object type and object name must be the first three parameters of the command string.
- A period (.) indicates the end of a command. If this character is detected anywhere within a command string, all subsequent data is ignored.
- In the syntax diagrams, `FM` or `IN` is shown instead of the `FROM` keyword to make the diagrams easier to read; however, `FROM` can always be used as a synonym for `FM` or `IN` and vice versa.
- The syntax of the *`where-clause`* and the *`with-clause`* is identical for each command.

LIST and FIND Command Syntax

The following command syntax applies to the list and find functions:

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

Examples of LIST and 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 and MOVE Command Syntax

The following command syntax applies to the copy and move functions:

```
           ALL
         [ CATALOGED ]
{ COPY } [ SAVED     ]  name [ FM [LIBRARY] lib-name ]       [where-clause]
{ MOVE } [ STOWED    ]
         [ VIEW      ]
           RESOURCE             TO [LIBRARY] lib-name         [where-clause]
                                                              [with-clause]
```

Examples of COPY and 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 Command Syntax

The following command syntax applies to the delete function:

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

Examples of DELETE

```
DELETE SA * IN LIBTEST TYPE GLA
```

```
D * IN TESTORD TYPE PM
```

```
D VIEW FINANCE IN TESTLIB DBID 12 FNR 27
```

RENAME Command Syntax

The following command syntax applies to the rename function:

```
         ALL
       [ CATALOGED ]
         SAVED
RENAME   STOWED       name  AS new-name [with-clause]
         VIEW               FM [LIBRARY] lib-name [where-clause]
         RESOURCE           TO [LIBRARY] lib-name [where-clause]
```

Examples of RENAME

```
RENAME PGM1 AS PROG1
```

```
R PGM1 AS PROG1 FM TESTLIB DBID 1 FNR 5 TO PRODLIB DBID 2 FNR 6
```

IMPORT Command Syntax

The command syntax that applies to the import function is shown in the following section.

For the points that must be considered before importing objects, see the description of the **import** function.

```
         ALL
       [ CATALOGED ]
         SAVED
IMPORT   STOWED       name  FM [PATH] path-name
         VIEW
         RESOURCE           TO [LIBRARY] lib-name [where-clause] [with-clause]
```

Examples of IMPORT

```
IMPORT ALL PGM* FM D:\NAT-PROGRAMS TO IMP-LIB
```

```
I RES res1.bmp FM D:\RESOURCES TO IMP-LIB
```

where-clause

```
[WHERE] [DBID dbid] [FNR fnr]
              [DIC (dbid,fnr,password,cipher)]
              [SEC (dbid,fnr,password,cipher)]
```

Separators

Commas must be used as separators between the values following the `DIC` and `SEC` keywords, or if a value is missing. For example: `DIC (10,,secret,2a)`. If the `ID` session parameter (see also *ID - Input Delimiter Character* in the *Parameter Reference*) has been set to a comma, use a slash (/) as the separator between values.

with-clause

```
[WITH]  [TYPE type] [FMDATE date] [FMTIME time]
                                       Y
        [USER user-id]   [    XREF {       }   ]
                                       N
        [REPLACE] [RCOP] [NOPROMPT] [HELP]
        [    STRUCT  ]
        [    SM      ]
        [    REPORT  ]
```

Keywords and Variables in Commands

This section explains the keywords and corresponding variable values (if required) used in a command.

Keywords are listed alphabetically. Letters in italics represent variable values that must be supplied with a keyword. For each variable value, the Natural data format and length is indicated.

Keyword	Value	Natural Data Format/ Length	Explanation
ALL	*name*	A9	Only applies to programming objects. The name of the object to be processed or a range of names; see ***Specifying a Range of Names***. Any saved (source) objects and/or cataloged objects are processed.
CATALOGED	*name*	A9	Only applies to programming objects. The name of the cataloged object to be processed or a range of names; see ***Specifying a Range of Names***.
SAVED	*name*	A9	Only applies to programming objects. The name of the saved (source) object to be processed or a range of names; see ***Specifying a Range of Names***.
STOWED	*name*	A9	Only applies to programming objects. The name of an object (or a range of names) for which the saved (source) *and* the cataloged object are to be processed (see also ***Specifying a Range of Names***). Only an object that exists as both a saved (source) object *and* a cataloged object is processed. The exceptions to this are copycode and text, neither of which can be cataloged. However, they are included in processing when this option is specified.
VIEW	*name*	A32	Only applies to DDMs. The name of the DDM to be processed or a range of names; see ***Specifying a Range of Names***.
RESOURCE	*name*	A255	Only applies to shared resources. The name of the shared resource to be processed or a range of names; see ***Specifying a Range of Names***.
FROM or FM or IN	*lib-name* or *path-name*	A8 or A253	Specifies a source library or a source path. The source library or path contains the object to be processed.
TO	*lib-name*	A8	Specifies a target library.
AS	*new-name*	A8 or A32 or A255	The new name to be given to an object when it is renamed with the RENAME command. Format/length A8 applies to programming objects, A32 to DDMs and A255 to shared resources.

Keyword	Value	Natural Data Format/ Length	Explanation
LIBRARY	*lib-name*	A8	An optional keyword that indicates the name (*lib-name*) of a source or a target library. If you omit the keyword and respective value, the library where you logged on before you invoked SYSMAIN is used for processing. The source library contains the object to be processed. The target library is the library to which the object is to be copied or moved, or where the object is renamed. *lib-name* must be specified immediately after the FROM/FM/IN or TO keyword. If LIBRARY is used, it must be entered between FROM/FM/IN or TO and *lib-name*.
PATH	*path-name*	A253	Only applies to the IMPORT command. An optional keyword that indicates the name (*path-name*) of a source path. For a valid path name, see **PATH** in *Using the Fields in an Object-Specification Window*. *path-name* must be specified immediately after the FROM/FM/IN or TO keyword. If PATH is used, it must be entered between FROM/FM/IN or TO and *path-name*.
WHERE	*where-clause*	-	An optional keyword that indicates the start of a *where-clause*. The *where-clause* must always follow the FROM/FM/IN or TO keyword and the library name (*lib-name*) or path name (*path-name*) if relevant; the sequence of the keywords and values within the clause can be specified in any order.
DBID	*dbid*	N5	The database ID (DBID) of a source or a target system file. The source system file contains the object to be processed. The target system file is the system file to which the object is to be copied or moved, or where the object is renamed if relevant. Valid DBIDs are 1 to 65535. If no DBID or **FNR** (file number) is specified, the following applies: The DBID and FNR of the system file where the current library resides are always used. For example: if you specify a library contained in the FUSER system file, the DBID and FNR of this file are used.
FNR	*fnr*	N5	The file number (FNR) of a source or a target system file. The source system file contains the object to be processed. The target system file is the system file to which the object is to be copied or moved, or where the object is renamed if relevant.

Keyword	Value	Natural Data Format/ Length	Explanation
			Valid FNRs are 1 to 65535. If no DBID (database ID) or FNR is specified, the following applies: The DBID and FNR of the system file where the current library resides are always used. For example: if you specify a library contained in the FUSER system file, the DBID and FNR of this file are used.
DIC	*dbid* *fnr* *password* *cipher*	A80	Specifies the environment of the FDIC source and/or target system file: database ID (*dbid*), file number (*fnr*), Adabas password (*password*) and Adabas cipher code (*cipher*).
SEC	*dbid* *fnr* *password* *cipher*	A80	Specifies the environment of the FSEC source and/or target system file: database ID (*dbid*), file number (*fnr*), Adabas password (*password*) and Adabas cipher code (*cipher*).
WITH	*with-clause*	-	An optional keyword that indicates the start of a *with-clause*. The keywords and values of the *with-clause* can be specified in any order, and the *with-clause* can be placed in any location within the command string, except in the first three positions.
TYPE	*type*	A20	The type(s) of object to be processed as listed in ***TYPE Specification*** below.
FMDATE	*date*	A10	The start date of a time period: All objects which were saved or cataloged on or after the specified date are processed. A date must be specified in a valid Natural date format. The default format is the international format *YYYY-MM-DD* (*YYYY* = year, *MM* = month, *DD* = day), for example, 2007-05-20.
FMTIME	*time*	A5	Only applies if FMDATE is specified. Specifies a start time: All objects which were saved or cataloged at or after the specified time (and date) are processed. A time must be specified in the format *HH*:*II* (*HH* = hours, *II* = minutes), for example, 11:33.
USER	*user-id*	A8	A user ID: All objects that were saved or cataloged by the specified user are processed.
XREF	N or Y	A1	Only applies to programming objects and if Predict is installed.

Keyword	Value	Natural Data Format/ Length	Explanation
			Indicates whether cross-reference (XRef) data stored on Predict system files is to be processed. You can specify one of the following values: `N` XRef data is not processed, except when using the `DELETE` command. If a cataloged object is deleted, SYSMAIN always deletes any existing XRef data for this object. `Y` All XRef data is processed. See also *XRef Considerations*.
`REPLACE`	-	-	Activates the replace option used in a *`with-clause`*. An object is automatically replaced. See also *Using the Replace Option*.
`RCOP`	-	-	Specifies that a copy of the object being renamed is to be made.
`NOPROMPT`	-	-	Not applicable in batch mode. Disables (`NOPROMPT`) the SYSMAIN prompts. With `NOPROMPT`, no confirmation screen is displayed. For example, before any deletion, SYSMAIN prompts you for confirmation.
`HELP`	-	-	This keyword is provided for compatibility reasons only.

`STRUCT`
or
`SM`
			Only applies to the `IMPORT` command.
			
			Indicates structured mode described in *Natural Programming Modes* in the *Programming Guide*.

`REPORT`			Only applies to the `IMPORT` command. Indicates reporting mode described in *Natural Programming Modes* in the *Programming Guide*.
.	-	-	A period (.) indicates the end of a command. If this character is detected anywhere within a command string, all subsequent data is ignored.

TYPE Specification

The following table lists all valid object-type codes for programming objects that can be used with the `TYPE` keyword:

Code	Object Type
`P`	Program
`N`	Subprogram
`S`	Subroutine
`M`	Map
`H`	Helproutine
`3`	Dialog
`5`	Processor
`A`	Parameter data area
`G`	Global data area
`L`	Local data area
`C`	Copycode
`T`	Text
`4`	Class
`7`	Function
`V`	View (DDM)
`8`	Adapter
*	All programming object types

65 XRef Considerations

All cross-reference (XRef) data stored in the Predict system file for a cataloged programming object (not applicable to DDMs) can be processed with SYSMAIN.

The Predict system file is determined by the value assigned to the profile parameter `FDIC` (see *FDIC - Predict System File* in the *Parameter Reference* documentation) in the parameter file or at the start of the Natural session.

You can override the current FDIC settings for the duration of the current SYSMAIN function by replacing the values in the **XREF** window (database ID and file number `0` denote the current FDIC) or by specifying the `DIC` keyword in the *`where-clause`* of a command.

The **XREF** field in an object-specification window or the `XREF` keyword in a command indicates whether SYSMAIN should process XRef data.

If Predict has not been installed, enter `N` in the **XREF** field (or specify `XREF=N`) to not process XRef data in Predict files. This is the default setting.

XREF set to N

No XRef data is processed if the **XREF** field contains an `N` (or `XREF=N`).

However, regardless of what setting you choose, XRef data is always deleted when a programming object is deleted.

XREF set to Y

XRef data is processed if `Y` is entered in the **XREF** field (or `XREF=Y`).

If XRef data is to be processed, the following actions are applied during SYSMAIN processing:

- SYSMAIN checks whether XRef data exists in the Predict source system file for the specified programming object.
- If a programming object is to be deleted from the target environment, XRef data is deleted from the Predict target system file.
- If a programming object is copied to a new environment, the XRef data of the programming object is copied from the Predict source system file to the Predict target system file. The library name is changed accordingly and, in the case of the rename function, the object name is also changed.
- If the move function is executed, the XRef data of the programming object is deleted from the Predict source system file.

FDIC File Security

If file security has been defined for the FDIC system file, you need to specify a password and a cipher code for the required source and/or target system file before you perform a SYSMAIN function. Otherwise, an appropriate error message appears. You do not have to provide security information for the default system files assigned to you at the start of the SYSMAIN utility.

XRef Processing Errors

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

- The value of the XREF option in Natural Security is set to `Y` and the **XREF** field contains an `N` (or `XREF=N`).
- The **XREF** field contains a `Y` (or `XREF=Y)` and the FDIC file(s) being used are not valid Predict files.

X SYSNCP Utility

66 SYSNCP Utility

The utility SYSNCP is used to define command-driven navigation systems for Natural applications.

The Natural Command Processor (NCP) consists of two components: maintenance and runtime. The utility SYSNCP is the maintenance part which comprises all facilities used to define and control navigation within an application. The PROCESS COMMAND statement (see the *Statements* documentation) is the runtime part used to invoke Natural programs.

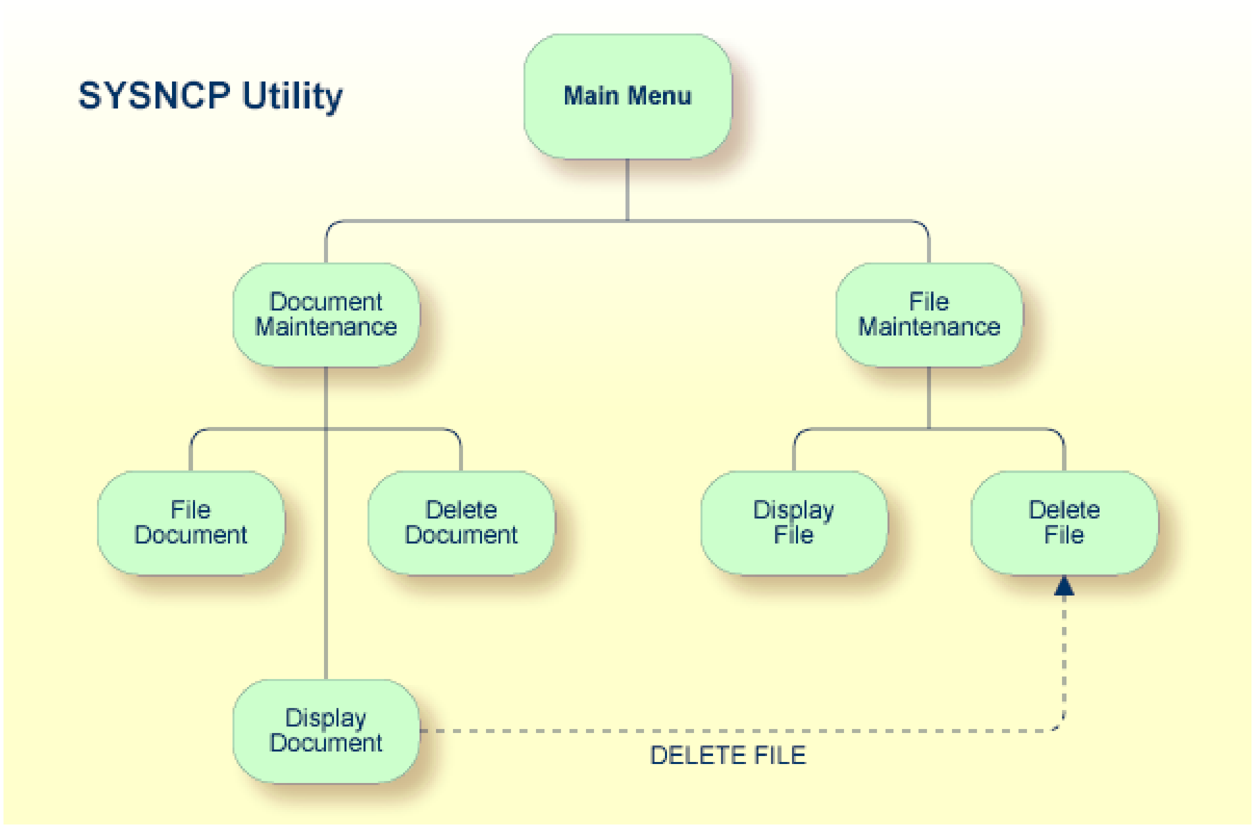
Prerequisites for UNIX

This section lists the prerequisites required for installing the command processor under UNIX:

- Logical file (LFILE) 190 (NCP Command Proc).
- FDT "SYSTEM-NCP" must be loaded during installation, see the section *SYSPCI Utility - Product Configuration and Initialization* in the *Installation* documentation.

Introducing the SYSNCP Utility

Applications which enable users to move from one activity to another activity by using direct commands far exceed in usability the ones which force the user to navigate through menu hierarchies to a desired activity.



The figure above illustrates the advantage of using direct commands. In an application in which menu hierarchies form the basis for navigation, a user wishing to advance from the Display Document facility to the Delete File facility would have to return to the Main Menu via the document branch and then enter the file branch. This is clearly less efficient than accessing the Delete File facility directly from the Display Document facility.

Below is information on:

- Object-Oriented Data Processing
- Features of the Command Processor
- Components of the Command Processor
- What is a Command?

- Creating a Command Processor

Object-Oriented Data Processing

The Natural command processor is used to define and control navigation within an application. It could be used, for example, to define a command DISPLAY DOCUMENT to provide direct access to the Display Document facility. When a user enters this command string in the Command line of a screen (for which this command is allowed), the Natural command processor processes the input and executes the action(s) assigned to the command.

In contrast to menu-driven applications, the command-driven applications implemented with the Natural command processor take a major step toward object-oriented data processing. This approach has the following advantages:

- The design of an application need not depend on the way in which a certain result can be reached, but only on the desired result itself. Thus, the design of an application is no longer influenced by the process flow within its components.
- The processing units of an application become independent of one another, making application maintenance easier, faster and much more efficient.
- Applications can be easily expanded by adding independent processing units. The resulting applications are, therefore, not only easy to use from an end-user's view, but also easier to create from a programmer's view.

The Natural command processor has the following additional benefits:

- **Less Coding**
  Instead of having to repeatedly program lengthy and identically structured statement blocks to handle the processing of commands, you only have to specify a PROCESS COMMAND statement that invokes the command processor; the actual command handling need no longer be specified in the source code. This considerably reduces the amount of coding required.
- **More Efficient Command Handling**
  As the command handling is defined in a standardized way and in one central place, the work involved in creating and maintaining the command-processing part of an application can be done much faster and much more efficiently.
- **Improved Performance**
  The Natural command processor has been designed with particular regard to performance aspects: it enables Natural to process commands as fast as possible and thus contributes to improving the performance of your Natural applications.

Features of the Command Processor

The Natural command processor provides numerous features for efficient and user-friendly command handling:

- **Flexible Handling of Commands**
  You can define aliases (that is, synonyms for keywords), and abbreviations for frequently used commands.
- **Automatic Check for Uniqueness of Abbreviated Keywords**
  The command processor automatically compares every keyword you specify in SYSNCP with all other keywords and determines the minimum number of characters in each keyword required to uniquely identify the keyword. This means that, when entering commands in an application, users can shorten each keyword to the minimum length required by the command processor to distinguish it from other keywords.
- **Local and Global Validity of Commands**
  You can specify in SYSNCP whether the action to be performed in response to a specific command is to be the same under all conditions or situation-dependent. For example, you can make the action dependent on which program was previously issued. In addition, you can define a command to be valid under one condition but invalid under another.
- **Error Handling for Invalid Commands**
  You can attach your own error-handling routines to commands or have error input handled by Natural.
- **Functional Security**
  With Natural Security, library-specific and user-specific conditions of use can be defined for the tables generated with SYSNCP. Thus, for your Natural applications you can allow or disallow specific functions or keywords for a specific user. This is known as functional security. See also the section *Functional Security* in the *Natural Security* documentation.
- **Help Text**
  In SYSNCP, you can attach help text to a keyword or a command. Then, by specifying a PROCESS COMMAND ACTION TEXT statement, you can return command-specific help text to the program.
- **Online Testing of Command Processing**
  If the execution of a command does not produce the intended result, you can find out why the command was not processed correctly by using the PROCESS COMMAND statement (see the *Statements* documentation) and the EXAM* sample test programs (source form) provided in the Library SYSNCP. The endings of the EXAM-* program names appear as abbreviations at the top border line of the relevant action windows (for example, EXAM-C appears as **C**).

To test a command processor at runtime

1 Enter the direct command EXAM to list all test programs. The **Demonstrate PROCESS COMMAND Statement** window is displayed.

2. Enter Function Code **O** to open a processor.
3. Enter the name of the processor.
4. Choose any of the Functions Codes listed (for example, C for CHECK) to apply command actions.
5. Enter Function Code **Q** to close the processor.

Components of the Command Processor

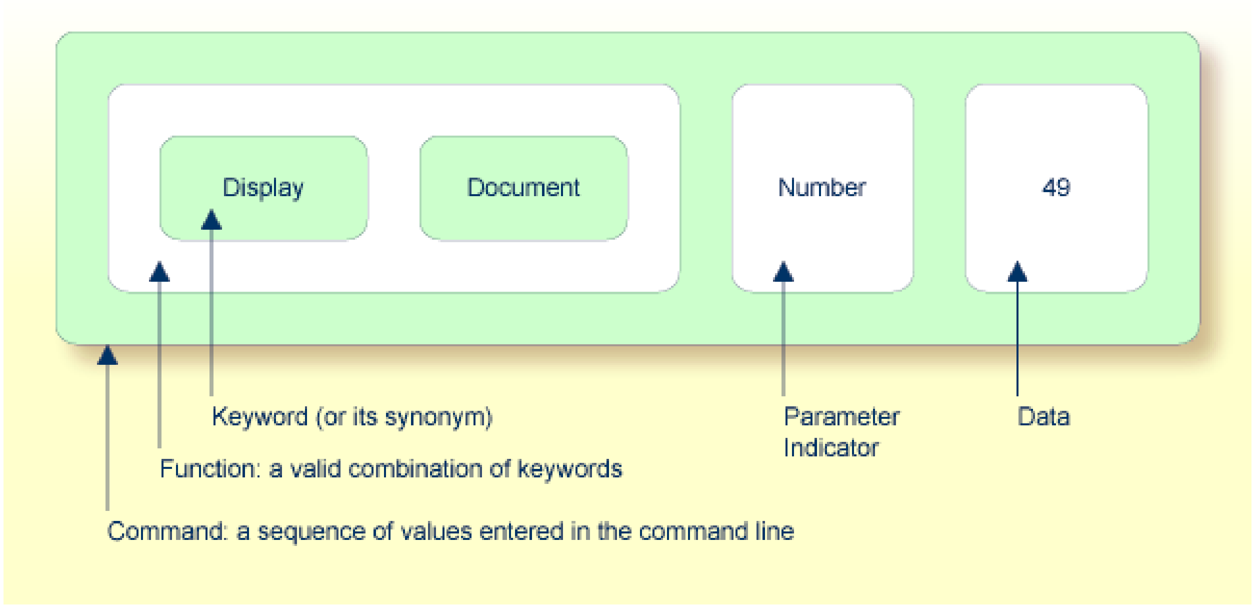
The Natural command processor consists of two parts: a development part and a runtime part:

- The development part is the utility SYSNCP, which is described in this section. With the utility SYSNCP you define commands (as described below) and the actions to be performed in response to the execution of these commands. From your definitions, SYSNCP generates decision tables which determine what happens when a user enters a command. These tables are contained in a Natural member of type Processor.
- The runtime part is the statement PROCESS COMMAND, which is described in the *Statements* documentation. This statement is used to invoke the command processor within a Natural program. In the statement, you specify the name of the processor to be used to handle the command input by a user at this point.

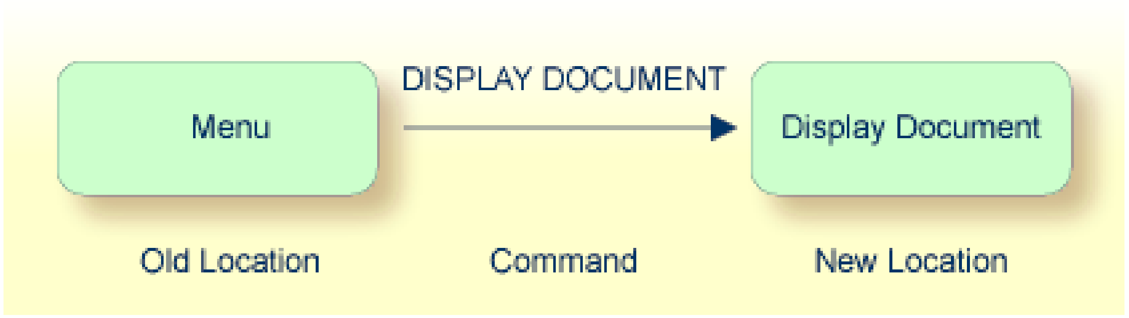
What is a Command?

A command is any sequence of values entered in the Command line which is recognized and processed by an application. Commands can contain up to three elements:

- **Function:**
  One or more valid keywords. For example, MENU or DISPLAY DOCUMENT.
- **Parameter Indicator:**
  Optional. A keyword which introduces command data.
- **Command Data:**
  Information to be sent to a function. Command data can be alphanumeric or numeric, for example, the name or the number of the file to be displayed.


Display
Document
Number
49
Keyword (or its synonym)
Function: a valid combination of keywords
Parameter Indicator
Data
Command: a sequence of values entered in the command line

Commands are always executed from a situation within an application; the position where this situation is reached is referred to as a location. Commands take the user from one location to another location; thus, each command can be viewed as a vector:

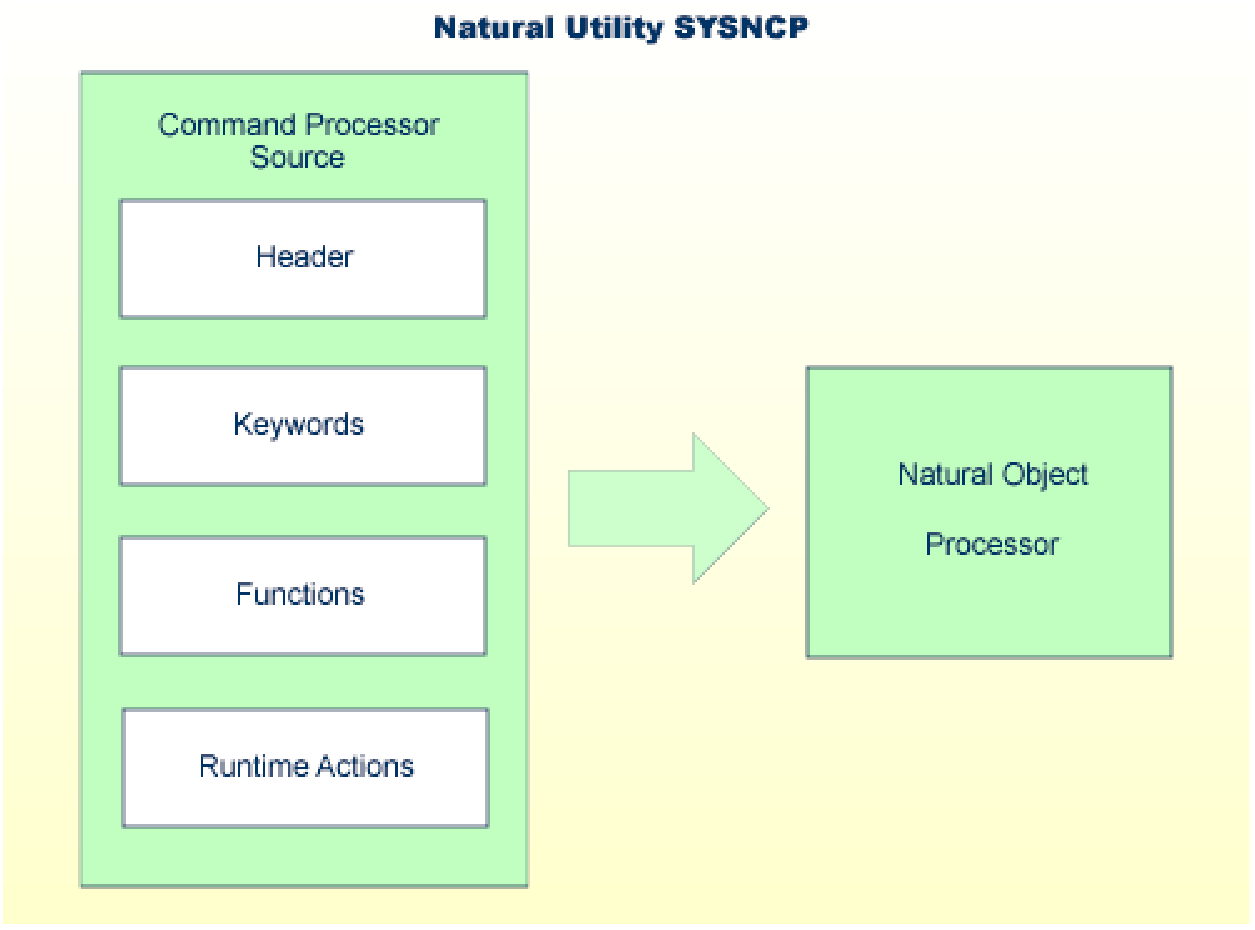


The location from which a certain command can be issued can be restricted on a system-wide or user-specific basis. On a system-wide basis, for example, the functions specified within commands can be local or global. A global function can be issued from *any* location while a local function can only be issued from specified locations. Restrictions can be placed on keywords and functions, however, if Natural Security is active in your environment.

Creating a Command Processor

The utility SYSNCP is used to create and maintain command processors. A command processor contains decision tables which determine what happens when a user enters a valid command.

The creation of a command processor is a cumulative operation involving several steps, from header definition, which establishes general defaults for the processor, to keyword definition, function definition and the linking of actions to functions. Special editors are provided by SYSNCP for the purpose of specifying keywords, functions and actions.



The end product of command processor development is a complex command processor source, which, when cataloged, generates a Natural object of type Processor. Whenever this object is referenced by the Natural statement PROCESS COMMAND, the runtime system of the Natural command processor is triggered.

The following is a summary of the steps necessary to create a command processor.

To create a command processor

1 **Verify/Modify the Session Profile.**

SYSNCP itself uses a Session Profile which contains various parameters which control how SYSNCP is to perform certain actions and how information is to be displayed. Desired modifications can be made and the resulting profile can be saved with a given user ID. See the section *Session Profile*.

2 **Initialize the Command Processor.**
The name of the command processor and the library into which it is to be stored are specified.

3 **Define Global Settings (Header).**
Various global settings for the command processor are defined. For example, descriptive text for keywords during editing, minimum and maximum length for keywords, in which sequence keywords are to be processed at runtime, runtime error-handling, and whether PF keys can be used at runtime to invoke functions. See the section *Header Records*.

4 **Define Keywords.**
Each keyword which is to be processed by the command processor is defined together with an indication as to whether the keyword is to be entered as the first, second or third entry of a command. Keyword synonyms can also be defined as well as parameter indicators. User text can be defined for each keyword. This text can subsequently be read at runtime using the PROCESS COMMAND ACTION TEXT statement. See the section *Keyword Maintenance*.

5 **Define Functions.**
Functions are defined by validating keyword combinations. A function can be defined as local (can only be invoked from a specific location within an application) and/or global (can be invoked from anywhere within an application). See the section *Function Maintenance*.

6 **Define Runtime Actions.**
The actions to be taken by the command processor when a command is issued at runtime are specified. Example actions are: fetch a Natural program, place a command at the top of the Natural stack, place data at the top of the Natural stack, change contents of the Command line. See the section *Runtime Actions*.

7 **Catalog Command Processor.**
The resulting source is cataloged as a Natural object (type Processor) in the designated Natural library. The command processor can now be invoked by a Natural program using the `PROCESS COMMAND` statement. See the section *Processor Cataloging*.

Invoking SYSNCP

To invoke the SYSNCP utility

- Enter the system command SYSNCP.

  The Processor Source Maintenance menu is displayed:

```
18:22:53              ***** NATURAL SYSNCP UTILITY *****           2000-05-22
User SAG                 - Processor Source Maintenance -

                     Code  Function

                      S    Select Processor
                      N    Create New Processor
                      H    Modify Header
                      K    Define Keywords
                      F    Define Functions
                      R    Define Runtime Actions
                      C    Catalog Processor
                      A    Administrator Services
                      ?    Help
                      .    Exit

              Code .. _    Name .. SAGTEST_  Library .. SYSNCP__



Logon to SYSNCP accepted.
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Cmd   Exit  Last  List  Flip                                    Canc
```

From this menu, you can invoke all functions necessary to create and maintain a command processor. To invoke a function, enter the code letter in the Code field.

Note: When you invoke the SYSNCP utility or restart SYSNCP, the user exit NCP-USR1 is invoked for dynamic customization purposes: see the program NCP-USR1 delivered in the Natural system library SYSNCP.

Help

For help on individual input fields (and also on some output fields) in SYSNCP, place the cursor on the field and press PF1.

Processor Selection

The Select Processor function results in a list of all existing command processor sources with related information. If Natural Security is installed, only those sources are listed which can be cataloged to a library to which you are allowed to log on. These restrictions do not apply to those users who have administrator status.

To invoke the Select Processor function

1. In the Processor Source Maintenance menu, enter Function Code **S**.
2. Press ENTER.

   The following information is provided for each processor:

Name	The name of the command processor.
Library	The name of the Natural library for which a processor is created. When the processor is cataloged, it is stored in this library.
User ID	The ID of the user who created the processor.
Date	The date the processor was created.
Status	The stage of development of the processor. For possible status values, see *Current Status* in the section *Header Records*.
Cat	Indicates if the processor has been cataloged.

   Note: With the user exit NCP-SELX (delivered in the Natural system library SYSNCP), you can limit the display to certain processors.

3. In the **Ac** field, enter any character to select a processor.

   The Processor Source Maintenance menu is displayed, where the name of the selected processor is automatically placed in the Name field.

   If you enter a question mark (?) in the Ac field, a window is be displayed, listing other possible options.

The name and library name of a command processor can be one to eight characters long. It can consist of upper-case alphabetical characters (A - Z), numeric characters (0 - 9) and the special characters: "-", "/ ", "$", "&", "#", "+" and "_".

Header Records

The header maintenance facility defines various global settings for a command processor. These definitions are collectively referred to as a header. Seven header maintenance screens are provided for creating and modifying headers. Header settings for a command processor can be updated at any stage of development (see the following section). After the settings have been modified, the status of a command processor is always set to Header (see also *Current Status*).

Below is information on:

- Create New Processor

Create New Processor

To create a new command processor

1 In the Processor Source Maintenance menu, enter Function Code **N** (Create New Processor),
the name of the command processor to be created, and
the name of the Natural library in which the command processor is to be later cataloged.

2 Press ENTER.

The first header maintenance screen is displayed.

The first header maintenance screen and the following ones are filled with default values that can be edited.

Modify Header - General Explanations

The Modify Header function is used to maintain an existing header; that is, to modify the various header settings for a given command processor.

To modify an existing header

1 In the Processor Source Maintenance menu, enter Function Code **H** (Modify Header),
the name of the corresponding command processor, and
the name of the library into which this command processor has been cataloged.

2 Press ENTER.

The first header maintenance screen is displayed.

3 Modify any input field in the header maintenance screens described below.

4 Press ENTER to confirm modifications.

Seven different screens are available for the definition and maintenance of a processor header (for the definition of a header, see the previous section).

To navigate between the header maintenance screens

- Use PF8 (forward) or PF7 (backward).

Each of the screens contains the following information:

Name	The name of the command processor.
Library	The name of the library into which the resulting command processor object is to be placed after being cataloged.
DBID, FNR	The database ID and file in which the specified library is located.
Created by	The user ID of the Natural user who initialized this command processor.
Date	The date the command processor was initially created.
Current Status	The command processor status: Init — The command processor has been initialized. Header — The header for the command processor has been created/modified. Keysave — Keywords have been defined and saved. Keystow — Keywords have been checked and stowed. Function — Keyword combinations have been defined. Action — Runtime actions have been defined. Object — An object form of the command processor has been created. Frozen — The command processor has been frozen. Copied — The command processor has been copied. Error — An error has been detected.

Keyword Runtime Options - Header 1

When you select the Modify Header function (as described above), the **Processor Header Maintenance 1** screen is displayed:

```
16:40:19               ***** NATURAL SYSNCP UTILITY *****         2000-05-04
User SAG               - Processor Header Maintenance 1 -

Modify Processor            Name SAGTEST  Library SYSNCP   DBID 10    FNR 32
Created by SAG      Date 2000-04-29                      Current Status Init

Keyword Runtime Options:
------------------------
   First  Entry used as ....... Action__________
   Second Entry used as ....... Object__________
   Third  Entry used as ....... Addition________

   Minimum Length ............. _1
   Maximum Length ............. 16
   Dynamic Length Adjustment .. -

   Keyword Sequence ........... 123_____
   Alternative Sequence ....... ________
   Local/Global Sequence ...... LG______

Processor Header with name SAGTEST for library SYSNCP has been added.
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Cmd   Exit  Last  List  Flip  -     +                         Canc
```

Various attributes which are to apply for the keywords defined for the command processor are entered on this screen.

Field	Explanation
First Entry used as	A descriptive text which is to be associated with all keywords which are entered as the first entry (entry type 1) when defining a keyword sequence. For example, if the first keyword of a keyword sequence is to represent the action to be performed (DISPLAY, DELETE, etc.), the descriptive text "Action" could be entered in this field. The first four characters of the text entered in this field appear under the column heading **Use** in the Keyword Editor as described in the section *Keyword Maintenance*.
Second Entry used as	A descriptive text which is to be associated with all keywords which are entered as the second entry (entry type 2) when defining a keyword sequence. If, for example, the second keyword of a keyword sequence is to represent the object to be used (DOCUMENT, FILE, etc.), the descriptive text "Object" could be entered in this field.

Field	Explanation
	The first four characters of the text entered in this field appear under the column heading **Use** in the Keyword Editor as described in the section *Keyword Maintenance*.
Third Entry used as	A descriptive text (TITLE, PARAGRAPH, etc.) which is to be associated with all keywords which are entered as the third entry (entry type 3) when defining a keyword sequence. The first four characters of the text entered in this field appear under the column heading **Use** in the Keyword Editor as described in the section *Keyword Maintenance*.
Minimum Length	The minimum length permitted when defining a keyword. Valid values are 1 - 16 characters. The default is one character.
Maximum Length	The maximum length permitted when defining a keyword. Valid values are 1 - 16 characters. The default is 16 characters.
Dynamic Length Adjustment	The following values are permitted: + At runtime, each keyword must be entered in its entirety. - At runtime, each keyword can be abbreviated provided that it retains uniqueness with respect to other keywords. S The number of characters which must be entered for a given keyword is to be specified during keyword definition in the **ML field** of the Keyword Editor as described in the section *Keyword Maintenance*.
Keyword Sequence	The sequence in which keyword entries are to be processed at runtime. Possible values are 1, 2, 3 and P (for parameter indicator); the default sequence is 12, which means first the first keyword entry and then the second keyword entry. See also the **field E** as described in the section *Keyword Maintenance*.
Alternative Sequence	An alternative sequence in which keywords are to be processed at runtime in the event that the default sequence (specified above) results in an error during runtime.
Local/Global Sequence	This option specifies the order of command validation to be performed at runtime. Possible values are: L Command is to be validated as a local command. G Command is to be validated as a global command. The default validation sequence is LG, which means that the command is to be validated first as a local command and then (if necessary) as a global one.

Keyword Editor Options - Header 2

Further keyword attributes can be entered on the **Processor Header Maintenance 2** screen:

Field	Explanation
Header 1 for User Text Header 2 for User Text	These two fields are used to enter a descriptive text which appears in the Keyword Editor above the column reserved for user text. This text is also output during runtime when the TEXT option is specified with the PROCESS COMMAND statement as described in the *Statements* documentation.
Prefix Character 1	This field and the next three are used to attach a hexadecimal prefix to keywords. This enables the processing of internal keywords which cannot be represented by a normal keyboard. When the command processor is cataloged, all prefix characters in keywords are replaced by the hexadecimal values specified. If a non-blank character is entered in one of the Prefix Character fields, the specified character is replaced by the hexadecimal value specified in the Hexadecimal Replacement field.
Hex. Replacement 1	The value specified in this field replaces the character specified in the field Prefix Character and is used as a prefix for a keyword at runtime.
Prefix Character 2	See above Prefix Character 1.
Hex. Replacement 2	See above Hex. Replacement 1.
Keywords in Upper Case	This option specifies whether keywords are to be translated to upper case in the Keyword Editor and the application: Y Keywords entered in the Keyword Editor are automatically converted to upper case. In the application, end-users can enter the keywords in upper or lower case. N Keywords entered in the Keyword Editor are not converted to upper case. In the application, end-users must enter the keywords *exactly* as they appear in the Keyword Editor.
Unique Keywords	This option specifies whether keywords within the processor must be unique. Y Each keyword defined must be unique within this processor, regardless of its type. N Each keyword defined for a given keyword type (1, 2, 3 or P) must be unique.

Miscellaneous Options - Header 3

Miscellaneous options can be entered on the **Processor Header Maintenance 3** screen:

Field	Explanation
Invoke Action Editor	This option specifies whether the Runtime Action Editor is to be activated from the Function Editor (see the sections *Runtime Action Editor* and *Define Functions*). Y The Runtime Action Editor is invoked whenever a valid keyword combination is defined in the Function Editor. N The Runtime Action Editor is suppressed in the Function Editor. **Note:** If you use the user exit NCP-REDM (delivered in the Natural system library SYSNCP), you should set this option to Y; otherwise, invalid runtime action values cannot be detected in time and can lead to runtime errors.
Catalog User Texts	This option specifies whether user texts are to be cataloged with the command processor. Y Text portions of the edit line (Keyword Editor; see the section *Define Keywords*) and the user text portion of the action line (Runtime Action Editor) are bound to the associated keyword or function when the command processor is cataloged. This text can then be read at runtime using the TEXT option of the PROCESS COMMAND statement. N Texts are not cataloged with the command processor and cannot be read at runtime.
Security Prefetch	This option specifies whether security checking is to be performed when the command processor is initially invoked during runtime or at each command evaluation. Y If Natural Security is installed, security checking is performed for all keywords when the processor is invoked. N If Natural Security is installed, security checking is performed with the evaluation of each keyword. If option Y is selected, security checking is performed only once for all keywords when the command processor is invoked. Since the checking procedure takes time, evaluation of the first command is comparatively slow at runtime, while the evaluation of all remaining commands is comparatively fast. Conversely, if option N is selected, the evaluation time for each command is always the same because security is checked for each keyword individually before it is evaluated.
Command Log Size	Commands processed at runtime can be stored in a command log area by the command processor. Specify in the input field the number of KBs storage space allocated to command logging: 0 No storage space is allocated to command logging. Command logging is inactive. 1 1 KB of storage space is allocated to command logging. Command logging is active.
Implicit Keyword Entry	This option specifies whether a keyword of type 1 is to be retained as an implicit keyword for all subsequent commands.

Field	Explanation
	1 If a command is entered which only contains a keyword of type 2, the command processor assumes the most recently entered keyword of type 1 as implicit keyword. N Option is disabled.
Command Delimiter	This option specifies the character used to separate commands if more than one command is specified in the Command line. At runtime, only the first command will be executed. For example: DISPLAY CUSTOMER; MODIFY CUSTOMER; PRINT.
PF-Key may be Command	This option specifies whether commands can be allocated to PF keys: if the command processor receives at runtime a command line which contains all blanks, it checks if a PF key has been pressed by the user. Possible values are: A The identifier for this PF key (system variable *PF-NAME) is used as the command. K The content of the *PF-KEY system variable is used as the command. Y If *PF-NAME is empty, the content of the *PF-KEY system variable is used instead. N PF keys cannot be used as command, Natural error NAT6913 is issued with message "Command line not accepted". For more information on the system variables *PF-NAME and *PF-KEY see the *System Variables* documentation.

Command Data Handling - Header 4

The attributes to be entered on the **Processor Header Maintenance 4** screen specify how command data are handled for a function; command data are optional.

Options are:

Field	Explanation
Data Delimiter	Specifies the character to be used to precede data. Default data delimiter is "#". Example: ADD CUSTOMER #123
Data Allowed	Specifies if data input is allowed at runtime. N A runtime error occurs if data is found. D Data is dropped if present. S Data is placed at the top of the Natural stack. No verification is performed.

Field	Explanation
	Y Data is checked and keyword entries of type P (parameter indicator) are evaluated. Example of Y: DISPLAY CUSTOMER NAME=SMITH
More than one Item Allowed	Only applies if the option Data Allowed is set to Y. Specifies whether more than one data string is permitted. N A runtime error occurs if more than one data string is found. D All data after the first data string are dropped. Y More than one data string is permitted. Example: ADD ARTICLE #111 #222 As long as uniqueness is guaranteed, the data delimiter can be omitted. Example: ADD ARTICLE 123
Maximum Length of one Item	Only applies if the option Data Allowed is set to Y. Specifies the maximum number of characters allowed for a data string. If the specified maximum is exceeded, a runtime error occurs. Valid range: 1 - 99.
Item Must be Numeric	Only applies if the option Data Allowed is set to Y. Specifies whether each data value must be an integer value. Y Data input must be a positive integer value. If not, a runtime error occurs. N Data can be of any type.
Put to Top of Stack	Only applies if the option Data Allowed is set to Y. Specifies where data is to be placed. Y Data is placed at the top of the Natural stack. 1-9 Data is placed in the *n*th occurrence of the DDM field RESULT-FIELD. If the occurrence has already been filled as a result of a runtime action, it is overwritten.
If Error, Drop all Data	Only applies if the option Data Allowed is set to Y or N. Specifies the reaction to a data evaluation error: Y If an error occurs during evaluation of the data, data is discarded and processing continues. N If an error occurs during data evaluation, control is given to the error handler as described below.

Runtime Error Handling - Header 5

The attributes to be entered on the **Processor Header Maintenance 5** screen specify how to handle runtime errors:

Field	Explanation
General Error Program	The name of the program which is to receive control when an error is detected during runtime processing by the command processor. The Natural stack contains the following information when this program is invoked: Error Number (N4) Line Number (N4) Status (A1) Program Name (A8) Level (N2) If no error program and no specific error handling is specified (see below), the program with the name as contained in the Natural system variable *ERROR-TA is invoked; otherwise, a Natural system error message is issued.
Keyword not found	Indicates whether an action has been specified that is to be performed if a keyword could not be found.
Keyword missing	Indicates whether an action has been specified that is to be performed if the keyword type is missing.
Keyword Sequence Error	Indicates whether an action has been specified that is to be performed in the case of a keyword sequence error.
Command not defined	Indicates whether an action has been specified that is to be performed in the case of an undefined command.
Data disallowed	Indicates whether an action has been specified that is to be performed in the case of disallowed data.
Data Format/Length Error	Indicates whether an action has been specified that is to be performed in the case of a format/length error.
General Security Error	Indicates whether an action has been specified that is to be performed if an error is detected during a general security check.
Keyword Security Error	Indicates whether an action has been specified that is to be performed if an error is detected during a keyword security check.
Command Security Error	Indicates whether an action has been specified that is to be performed if an error is detected during a command security check.

Statistics - Header 6

The **Processor Header Maintenance 6** screen contains only output fields which report statistical data about the keywords specified for a command processor.

The following statistical information is provided:

Field	Explanation
Entry *n* Keywords	The number of keywords of type *n* defined in the command processor (not including synonyms).
Entry *n* Keywords + Synonyms	The sum of keywords of type *n* and their assigned synonyms.
Highest IKN for Entry *n*	The largest Internal Keyword Number for the keyword of type *n*.
Possible Combinations	The number of possible combinations for keywords defined.
Cataloged Functions	The number of keyword combinations currently cataloged.

Status - Header 7

The **Processor Header Maintenance 7** screen contains only output fields which report the time and the date when parts of the command processor were executed or modified.

Keyword Maintenance

Keywords are the basic components for defining functions. Before it is possible to define keywords, the header maintenance records must be created (see the section ***Header Records***).

- Define Keywords
- Editor Commands
- Positioning Commands
- Line Commands

Define Keywords

Keywords used in commands are created with the Define Keywords function and the Keyword Editor. The Keyword Editor is similar to existing Natural editors except that lines of the editor are broken up into separate fields. Most of the **editor commands** (see the relevant section) and the **line commands** (see the relevant section) which are used in the Natural program editor can also be used in the Keyword Editor.

To invoke the Keyword Editor

1 In the Processor Source Maintenance menu, enter Function Code **K** (Define Keywords).

2 Press ENTER.

The Keyword Editor screen is displayed.

The Keyword Editor screen is shown below. Several keywords have already been defined to serve as examples for this section.

```
09:42:39                    - SYSNCP Keyword Editor -                2000-05-04
Modify Keywords              Name SAGTEST   Library SYSNCP   DBID 10   FNR 32

I Line E Use  Keyword          IKN   ML Comment
- ---- - ---- ---------------- ----  -- ----------------------------------------
     1 1 Acti MENU             1004   1
     2 1 Acti DISPLAY          1002   2
     3 S Syno SHOW             1002   1
     4 1 Acti DELETE           1001   2
     5 S Syno PURGE            1001   1
     6 S Syno ERASE            1001   1
     7 1 Acti FILE             1003   4
     8 P Parm NAME             4002   2
     9 2 Obje FILE             2001   4
    10 P Parm NUMBER           4001   2
    11 2 Obje DOCUMENT         2003   2
    12 1 Acti INFORMATION      1005   1
    13
    14
- ---- - ---- ----- All ------ ----  -- ----------------------------------------

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Cmd   Exit  Last  List  Flip  -1    +1    Top   Bot   Info  Canc
```

Enter in the Keyword Editor all the keywords which you want to have in your command language. These can be entered in any order desired, except synonyms, which must immediately follow the keywords they are related to. To each keyword you assign a type which specifies to which part of command syntax the keyword belongs. Rules of command syntax for a command processor are specified in the processor header; see *Keyword Runtime Options - Header 1* in the section *Header Records*. For example, you can specify whether a keyword is to be of type 1 (entered in first position in a command), type 2, type 3, a synonym for another keyword or a parameter indicator.

Note: A command language requires a strict syntax because, to date, no computer is capable of understanding semantics. Word type is, therefore, the only practical way to communicate meaning in a command language.

In the example above, the keywords DELETE and DISPLAY are defined as keywords of type 1. As specified in the processor header, these keywords denote actions. The keyword DOCUMENT is defined as a keyword of type 2 and it denotes an object. The keyword FILE, however, is defined as both type 1 and 2, and it can, therefore, denote an action or an object, depending on where it is positioned in the command. It is possible to compose the two keyword types to make commands, such as DELETE FILE and FILE DOCUMENT.

You can save the keywords you have entered by issuing the SAVE or STOW command from the Command line. In addition to saving the keyword definitions in source form, the STOW command performs a consistency check on them. Once a keyword is stowed successfully, it is given an internal keyword number (IKN) which is used at runtime to evaluate a command. Synonyms are always linked to a master keyword and always take the IKN of their master.

Each line in the Keyword Editor contains the following fields:

Field	Explanation
I	Output field. An information field which can contain the following values: E Indicates that a definition error has been detected. X Line is marked with X. Y Line is marked with Y. Z Line is marked with both X and Y. S Scan value found in this line.
Line	Output field. The line number of the editor.
E	Specifies the entry type for a keyword; that is, the position the keyword is to be entered in a command: first, second or third position, synonym or parameter indicator. For instance, in the **Keyword Editor screen** example above the keyword DELETE is of entry type 1 and DOCUMENT of type 2. Using these keywords, the command DELETE DOCUMENT can be defined. The field takes any of the following characters as input: 1 The keyword defined in this line is to be used as the first titem in a command sequence. 2 The keyword defined in this line is to be used as the second titem in a command sequence. 3 The keyword defined in this line is to be used as the third titem in a command sequence. S The keyword defined in this line is to be used as a synonym for the preceding keyword with titem type 1, 2, 3 or P. P The keyword defined in this line is to be used as a parameter indicator in a command sequence. * No keyword is to be defined in this line. Instead, the line is to be used solely as a comment line. ? This symbol is an output value which indicates an invalid keyword specification.
Use	Output field. The value displayed is determined by the value entered in the preceding field E:

Field	Explanation
	1-3 The first four characters of the user text specified in the processor header for the first, second and third keyword entries respectively are displayed. See also ***Keyword Editor Options - Header 2*** in the section *Header Records*. S SYNO, the abbreviation for synonym, is displayed. P PARM, the abbreviation for parameter indicator, is displayed.
Keyword	Enter the keyword to be defined. Embedded blanks are not permitted. If you have specified in the processor header that keywords can only be upper case, then keywords are always translated to upper case, regardless of how they are entered. Otherwise, the case remains as entered. The maximum and minimum length of keywords depends on the settings specified in the header (default: 1 - 16 characters). Keywords must be unique unless specified otherwise in the header. Keyword prefixes can be used as described in ***Keyword Editor Options - Header 2*** in the section *Header Records*.
IKN	Output field. The Internal Keyword Number (IKN) is an identifier assigned to each valid keyword. IKNs are useful for testing and debugging. They are allocated only when a keyword is successfully stowed (see also the **STOW** command under *Editor Commands*). Each keyword is assigned a unique IKN, except synonyms, which take the IKN of their master term (see the **Keyword Editor screen** example above: DISPLAY and SHOW).
ML	Input and output field indicating the minimum length of a keyword. The field is an input field if **S** is specified in the **Dynamic Length Adjustment** field of the processor header as described in *Keyword Runtime Options - Header 1*, *Header Records*. In this case, you must specify the number of characters which must be entered for the keyword. For all other input, this field contains the minimum number of characters of a keyword a user must specify to avoid ambiguity with other keywords. For instance, in the **Keyword Editor screen** example above, keyword MENU requires only input of **M** while keyword DISPLAY requires input of **DI** to avoid ambiguity with keyword DELETE.
Comment	Enter free text for a keyword. There are no input restrictions. The user text is included in the cataloged command processor if the field **Catalog User Texts** is set to Y in the header definition as described in "Miscellaneous Options - Header 3", Header Records. It can be read at runtime using the TEXT option of the PROCESS COMMAND statement. The header text appearing at the top of this column is controlled by the header definition fields "Header for User Text 1" and "Header for User Text 2".

Editor Commands

In the Command line of the Keyword Editor, you can enter the following commands:

Command	Function
ADD	Adds ten empty lines to the end of the editor.
CANCEL	Returns to Processor Maintenance Menu.
CHECK	Tests the keyword source for consistency.
EXIT	Returns to Processor Maintenance Menu.
HELP	Displays valid escape characters and other useful processor settings.
INFO	Displays information on the keyword on which your cursor is positioned.
LET	Undoes all modifications made to the current screen since the last time ENTER was pressed.
POINT	Positions the line in which a line command **.N** is entered to the top of the current screen.
RECOVER	Returns keyword source that existed before last SAVE/STOW.
RESET	Deletes the current X and Y line markers.
SAVE	Keyword source is saved.
SCAN	Scans for the next occurrence of the scan value.
STOW	Keyword source is stowed and Internal Keyword Numbers (IKNs) are generated for valid keywords.

Positioning Commands

Editor positioning commands are the same as the ones provided for the Natural program editor. For more information, see the description of the program editorprogram editor in the *Editors* documentation.

The last line of the editor contains an output field which informs you of where your display is located in the editor. The following output values are displayed:

Top	Editor is currently positioned at the top of the keyword source.
Mid	Editor is currently positioned at the center of the keyword source.
Bot	Editor is currently positioned at the bottom of the keyword source.
Emp	Editor is currently empty.
All	The entire source is contained on the current screen.

Line Commands

Line commands in the Keyword Editor are the same as in the Natural program editor with the exception of the commands .J and .S, which cannot be used.

Each command is entered beginning in the **E** field; the remaining part of the command is entered in the **Keyword** field, as illustrated in the screen below:

```
09:42:39                     - SYSNCP Keyword Editor -                2000-05-04
Modify Keywords               Name SAGTEST   Library SYSNCP   DBID 10   FNR 32

I Line E Use  Keyword           IKN   ML Comment
- ---- - ---- ---------------- ----  -- ----------------------------------------
     1 1 Acti MENU              1004   1
     2 1 Acti DISPLAY           1002   2
     3 S Syno SHOW              1002   1
     4 . Acti i(3)TE            1001   2
     5 S Syno PURGE             1001   1
```

Caution: When you move (.M) or copy (.C) lines, ensure that individual keywords are always moved or copied together with their synonyms.

When you delete (.D) lines, the corresponding keywords and any functions containing these keywords will not be deleted from the database until you issue the STOW editor command. As long as you do not issue the STOW command, these functions will still be displayed within the Function Editor.

Function Maintenance

Functions are composed of the keywords entered in the Keyword Editor. Before it is possible to define functions, the keywords must be successfully stowed (see the section *Keyword Maintenance*).

- Define Functions
- Editor Commands
- Direct Command QUICK-EDIT
- Local and Global Functions
- Procedure for Validating Functions

Define Functions

Use the Define Functions function and the Function Editor to specify functions and compose valid commands which can be accessed from a specific location.

To invoke the Function Editor

1. In the Processor Source Maintenance menu, enter Function Code **F** (Define Functions).
2. Press ENTER.

   The Function Editor screen is displayed.

The Function Editor displays all possible combinations of the keywords stowed in the Keyword Editor.

The screen below, shows the Function Editor with keywords used as examples in the **Keyword Editor screen** in the section *Keyword Maintenance*:

```
 09:45:53              ***** NATURAL SYSNCP UTILITY *****            2000-05-04
 User SAG                            - Function Editor -
 Edit Global Combinations    Name SAGTEST  Library SYSNCP    DBID 10   FNR 32

 Global
 I Ac   Action           Object           Addition        Global  Local  Any Loc
 - --   ---------------- ---------------- --------------- ------ ----- -------
        DELETE
        DELETE           DOCUMENT                                        Yes
        DELETE           FILE                                            Yes
        DISPLAY
        DISPLAY          DOCUMENT                                        Yes
        DISPLAY          FILE                                            Yes
        FILE
        FILE             DOCUMENT                                        Yes
        FILE             FILE                                            Yes
        INFORMATION                                          Yes
        INFORMATION      DOCUMENT
        INFORMATION      FILE
 Repos: ________________ ________________ ________________ ------ ----- -------

 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Cmd   Exit  Last  List  Flip        +     Top    Loc   Loc+  Canc
```

You have to validate each keyword combination that you want to designate as a valid function in your application. A keyword combination can be validated as a global function, local function or both. A global function can be invoked from anywhere in an application, whereas a local function can only be invoked from a specific location within an application.

Two fields in the upper left corner of this screen indicate the current validation mode (local or global) and the location for which keyword combinations can currently be validated. In the screen above, the text "Edit Global Combinations" indicates that global mode is active. If the local mode were active, the text "Edit Local Combinations" would appear here. In the screen above, the text "Global" appears below this text. This indicates that global validation can be performed for all of the combinations listed. In local mode, in this field the name of the location appears for which local validation can be performed (for example, "Local DISPLAY FILE").

The Function Editor contains the following columns:

Column	Explanation
I	Output field. The following values are output as a result of function editing. E Runtime action edited. D Referenced locations displayed. V Validation issued. R Validation removed.
Ac	Action to be taken. The following values can be entered: VG Validate as global function. VL Validate as local function. RG Remove validation as global function. RL Remove validation as local function. DL Display all functions which reference the specified function as a local function. EG Invoke the Runtime Action Editor for a global function (see ***Runtime Action Editor*** in the section *Runtime Actions*). EL Invoke the Runtime Action Editor for a local function (see ***Runtime Action Editor*** in the section *Runtime Actions*). +G Invoke global mode, so that you can maintain any global functions. +L Invoke local mode for the current line, so that you can maintain local functions for this line. IN Information about keywords in this line.
Action Object Addition	These three columns are used to display all possible combinations of currently defined keywords. The text which appears at the top of each keyword column is controlled by the fields **First Entry used as**, **Second Entry used as** and **Third Entry used as** as specified in the processor header (see ***Keyword Runtime Options - Header 1*** in the section *Header Records*).
Global	If the function has been defined as a global command, Yes appears in this field.
Local	If the function has been defined as a local command, Yes appears in this field for the current location (only displayed in local mode).
Any Loc	Any Location. If the function has been defined as a local command anywhere else within the processor, Yes appears in this field for any other location.

Editor Commands

In the Command line of the Function Editor, you can enter the following commands:

Command	Function
ANY ON	Enable the column Any Loc.
ANY OFF	Disable the column Any Loc (the column will be filled with question marks). This allows for faster scrolling in the Function Editor. Moreover, the third repositioning field is available. Also, processing-in-progress information windows will not be displayed.
FIELD	Display keyword-specific combinations.
GLOBAL	Activate global mode.
LOC	Position to next location group.
LOC+	Position forward by one location.
SINGLE ON	Display only single-word functions.
SINGLE OFF	Display all possible combinations.
TOP	Position to top of list.

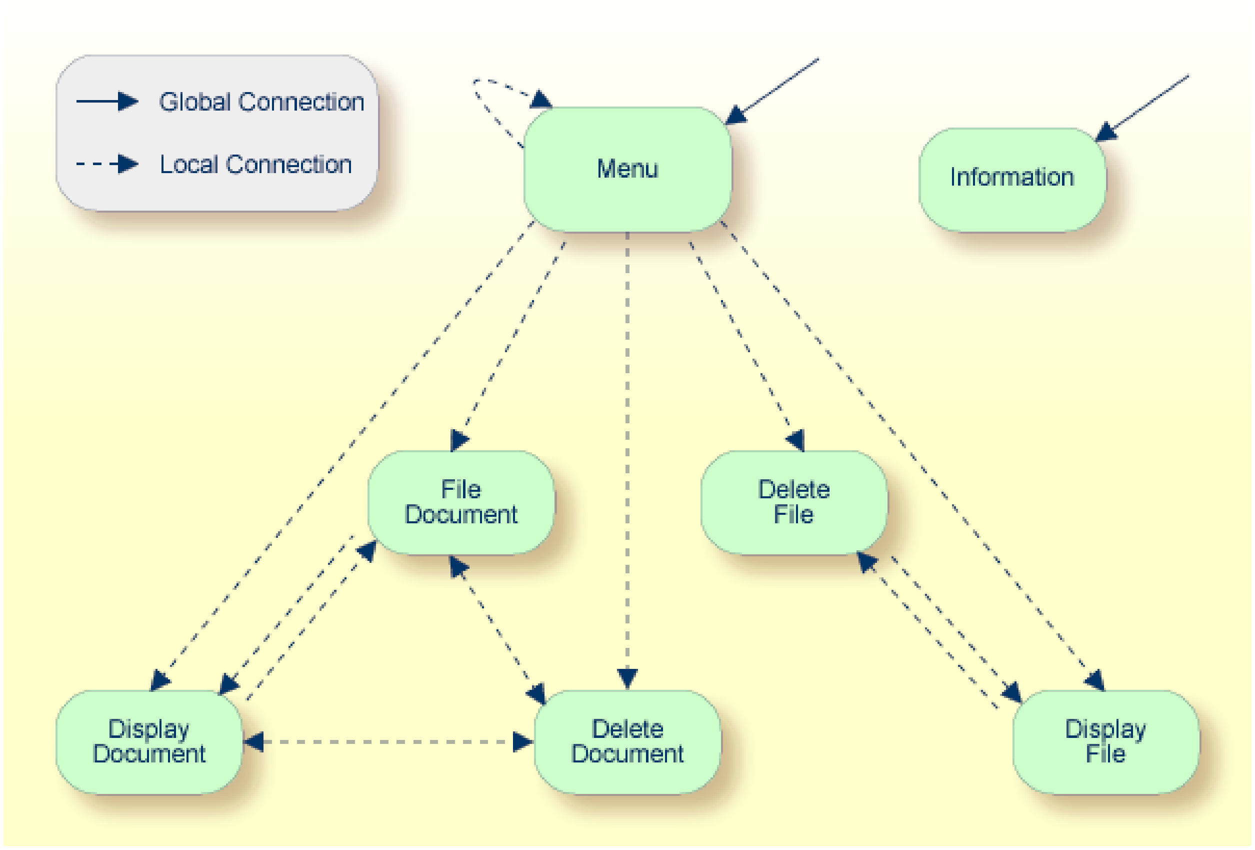
Direct Command QUICK-EDIT

The direct command QUICK-EDIT enables you to quickly define local/global functions, as well as the corresponding runtime actions, by entering keywords or IKNs directly. This may be helpful for extremely large command processors. Note, however, that the location from which the command can be issued is not verified and navigation may not function correctly at runtime.

Local and Global Functions

To understand the concept of local and global functions, you have to picture each valid keyword combination as a location in your application (for example, a location called Display File). In the Function Editor, you specify the commands which can be issued from this location, as well as from which locations this location can be reached using the command DISPLAY FILE.

Local and Global Connections within a Sample Application:



In the sample application above, the Menu and Information locations are the only locations which have been designated as global. Thus, they can be accessed directly from all of the remaining locations in the application. All locations have been designated as local to the location Menu, except Information. The only way to get from the location Display File to Display Document is via Menu.

Procedure for Validating Functions

The Function Editor operates in two modes: global and local. From global mode you can validate global functions and from local mode you can validate global and local functions. Global mode is the default mode. You can determine whether the editor is in global or local mode by the output field above the **I** field in the editor. If the editor is in global mode, then Global is displayed. If the editor is in local mode, then the location for which local functions are to be validated is displayed. Below is a general procedure for validating global and local functions for an application.

To validate global and local functions

1 With the Function Editor in global mode, enter **VG** (validate global) in the Ac field next to the corresponding action to validate all global functions.

Press ENTER.

The **Runtime Action Definition** screen appears.

2 Press PF3 to return to the Function Editor.

`Yes` appears under the column heading Global beside the validated functions.

3 Enter **+L** in the Ac field for each global function validated in the previous step, to switch to local mode.

Press ENTER.

4 Enter **VL** (validate local) in the **Ac** field for each function that is to serve as a location for this global function.

Press ENTER.

The Runtime Action Definition screen appears.

5 Press PF3 to return to the Function Editor.

`Yes` appears under the column heading Local beside the validated functions.

6 To validate local functions for a *local* location: Enter **+L** (invoke local mode) in the **Ac** field for each location validated in the previous step, to validate all local functions which are to be used from this location.

Press ENTER.

7 Enter **VL** (validate local) in the **Ac** field for each function that is to serve as a local function for the current location.

8 Press PF3 to return to the Function Editor.

`Yes` appears under the column heading Local beside the validated functions.

Note: If in the command processor header (Processor Header Maintenance 3) the field Invoke Action Editor is set to `Y`, in addition, the window Runtime Action Definition (see ***Runtime Action Editor*** in the section *Runtime Actions*) is displayed for each action.

Runtime Actions

Once valid keyword combinations have been identified as either local or global functions in the Function Editor, it is possible to link each function with one or more runtime actions. Runtime actions consist of one or more steps which are to be carried out whenever a function is issued.

Below is information on:

- Runtime Action Editor

Define Runtime Actions

There are two different locations in SYSNCP from which you can define runtime actions: the Function Editor (see the section *Function Maintenance*) and the Result Editor. The Result Editor is explained in this section, including how to specify runtime actions for a function.

To invoke the Result Editor

1 In the **Processor Source Maintenance** menu, enter Function Code **R** (Define Runtime Actions).

2 Press ENTER.

The Result Editor screen is displayed:

```
09:47:03              ***** NATURAL SYSNCP UTILITY *****         2000-05-04
 User SAG                           - Result Editor -
 List defined combinations   Name SAGTEST  Library SYSNCP   DBID 10    FNR 32


 I Ac Location                      Command                         Result
 - -- ------------------------------ ------------------------------ --------
      < Global >                     MENU                             KR
      < Global >                     INFORMATION                      SF
      DELETE FILE                    DISPLAY FILE                     SF
      DELETE DOCUMENT                DISPLAY DOCUMENT                 SF
      DISPLAY FILE                   DELETE FILE                      SF
      DISPLAY DOCUMENT               DELETE DOCUMENT                  SF
      DISPLAY DOCUMENT               FILE DOCUMENT                    SF
      FILE DOCUMENT                  DELETE DOCUMENT                  SF
      FILE DOCUMENT                  DISPLAY DOCUMENT                 SF
      MENU                           DELETE FILE                      KCS
      MENU                           DELETE DOCUMENT                  KCCS
      MENU                           DISPLAY FILE                     KRCS
Repo ______________________________ ______________________________ --------

 Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Cmd   Exit  Last  List  Flip        +     Top   Loc-- Loc+  Canc
```

The Result Editor contains all of the local and global functions specified in the Function Editor. Each line in the editor represents the location from which a command can be issued (Location field), the command itself (Command field) and an abbreviated summary of the action to be carried out when the command is issued (Result field).

The fields of the screen are explained in detail in the table below:

Field	Explanation
I	Output field. Information on the last action carried out on this line.
Ac	Action to be taken. The following values can be entered: DI Display the runtime action definitions for this function. ED Edit the runtime action definitions for this function. PU Purge this function.
Location	Output field. The location within the application from which the command (see Command field below) can be issued. If the function is global, then < Global > appears in this field (the command can be issued from any location).
Command	Output field. The command. The contents of the Location and Command fields may be truncated if very long keywords are used.
Result	Output field. Contains an abbreviated summary of the action to be performed when the command is issued. The first character represents the **Keep Location** information (see the following section); for all other characters, see the **Runtime Action Definition** table below.

Runtime Action Editor

The Runtime Action Editor is used to define the actions to be taken when a command is issued from a specific location. The editor can only be invoked for functions which have been defined as global or local functions. The editor can be invoked either from the Function Editor or the Result Editor.

To invoke the Runtime Action Editor from the Function Editor

1 In the **Ac** field, enter **EG** (edit global) for global functions.

Or:

In the **Ac** field, enter **EL** (edit local) for local functions.

2 Press ENTER.

To invoke the Runtime Action Editor from the Result Editor

1 In the **Ac** field, enter **ED**.

2 Press ENTER.

The **Runtime Action Definition** window is displayed:

```
                           Runtime Action Definition

   Location .... DISPLAY DOCUMENT
   Command ..... DELETE DOCUMENT

   Keep Location .... S
   Data allowed ..... Y   More than one .... N   Max. Length ...... 99
   Numeric .......... N   TOP of STACK ..... Y   Error: Drop ...... Y

   A  Runtime Action Definition
   -  ----------------------------------------------------------------
   F  DE-PGM__________________________________________________________
   _  ________________________________________________________________
   _  ________________________________________________________________
   _  ________________________________________________________________
   _  ________________________________________________________________
   _  ________________________________________________________________
   _  ________________________________________________________________
   _  ________________________________________________________________
```

Actions are always associated with an origin and a destination. The origin is the location from which the command is issued, and the destination is the command itself. Thus, it is possible to link different actions to a command based on the context in which it is used.

In the Runtime Action Editor, you also specify whether the location is to remain the same after the actions have been carried out, or whether the command itself is to become the new current location.

Actions are specified by entering a single-letter code in the left column of the editor. Enter any parameters accompanying an action in the field next to the code. If the characters "/*" are entered in this field, all subsequent input is considered a comment. If you omit a required parameter, you will be prompted for input.

The sequence in which actions are performed at runtime is determined by the order of entry in the editor (from top to bottom). Thus, if a FETCH is specified, all of the actions specified below it are not to be performed.

The Runtime Action Editor contains the following fields:

Field	Explanation
Location	Output field. The location from which the command is issued. If the function is defined as global, the field shows < Global >.
Command	Output field. Command for which actions are to be specified.
Keep Location	Specifies whether the current or a new location is to be active once the actions have been performed. A value in this field only affects commands with a specified EXEC option. Possible values are: K Keep current location. The actions to be performed affect the current location only. S Set new location (global/local). Once the actions are performed, the command processor makes the command the new current location. Every command entered subsequently has to be either a local command of this new location or a global command. **Note:** The defined actions themselves have no influence on the location; that is, any action performed does *not* cause the current location to be changed.
Other Options	All other options are related to the handling of parameters provided with this command sequence. For further information, see ***Command Data Handling - Header 4*** in the section *Header Records*. To activate the header defaults of these options, enter an asterisk (*).

To define runtime actions

1 Invoke the **Runtime Action Definition** window as described earlier.

2 In the field **A**, enter an action code and the corresponding action in the field opposite to it:

Code	Runtime Action Definition
V	Default value. No runtime action is specified.
T	Text which can be read at runtime using the TEXT or GET option of the PROCESS COMMAND statement.
M	Modify command line. The data are placed in the command line.
C	Command. This command is placed at the top of the Natural stack. If an asterisk (*) is specified here, the name of the program which issued this PROCESS COMMAND statement is put on top of the stack (STACK TOP COMMAND '*PROGRAM'). (*)
D	Data. These data are placed on top of the Natural stack. (*)
F	Natural program name. The program is invoked with a FETCH statement. (*)
S	Natural STOP statement. The statement is executed at runtime. (*)
E	The value specified in this line is to be moved immediately into the system variable *ERROR-NR.
R	A return code is entered in the DDM field RETURN-CODE as described in PROCESS COMMAND in the *Statements* documentation.
1 to 9	A text string. This value is entered into the multiple DDM field RESULT-FIELD as described in PROCESS COMMAND in the *Statements* documentation.

Code	Runtime Action Definition
*	Comment line.

* These actions are only performed with the EXEC option of the PROCESS COMMAND statement.

3 Press PF3 to leave the **Runtime Action Definition** window.

Note: The user exit NCP-REAM allows you to use some or all of the above codes. The user exit NCP-REEM allows you to modify the line that follows the heading of the Runtime Action Definition table. The user exit NCP-REDM allows you to define default values for runtime action definitions (if you use this user exit, see also ***Invoke Action Editor*** in the section *Header Records*). All user exits mentioned above are delivered in the Natural system library SYSNCP.

Processor Cataloging

Once you have specified runtime actions for all of the functions you want to use in your command processor, you should catalog the command processor. Cataloging a command processor generates a Natural object of type Processor.

To catalog a command processor

1 In the Processor Maintenance menu, enter Function Code **C** (Catalog Processor),
the name of the command processor to be cataloged,
and the name of the Natural library in which the command processor is to be cataloged.

2 Press ENTER.

Note: If you have Natural Security installed, you have to allow the use of your command processor as described in the *Natural Security* documentation in the section *Functional Security*.

Note for Windows, UNIX and OpenVMS:
Unlike on mainframes, SYSNCP does not create a report when cataloging a command processor.

Administrator Services

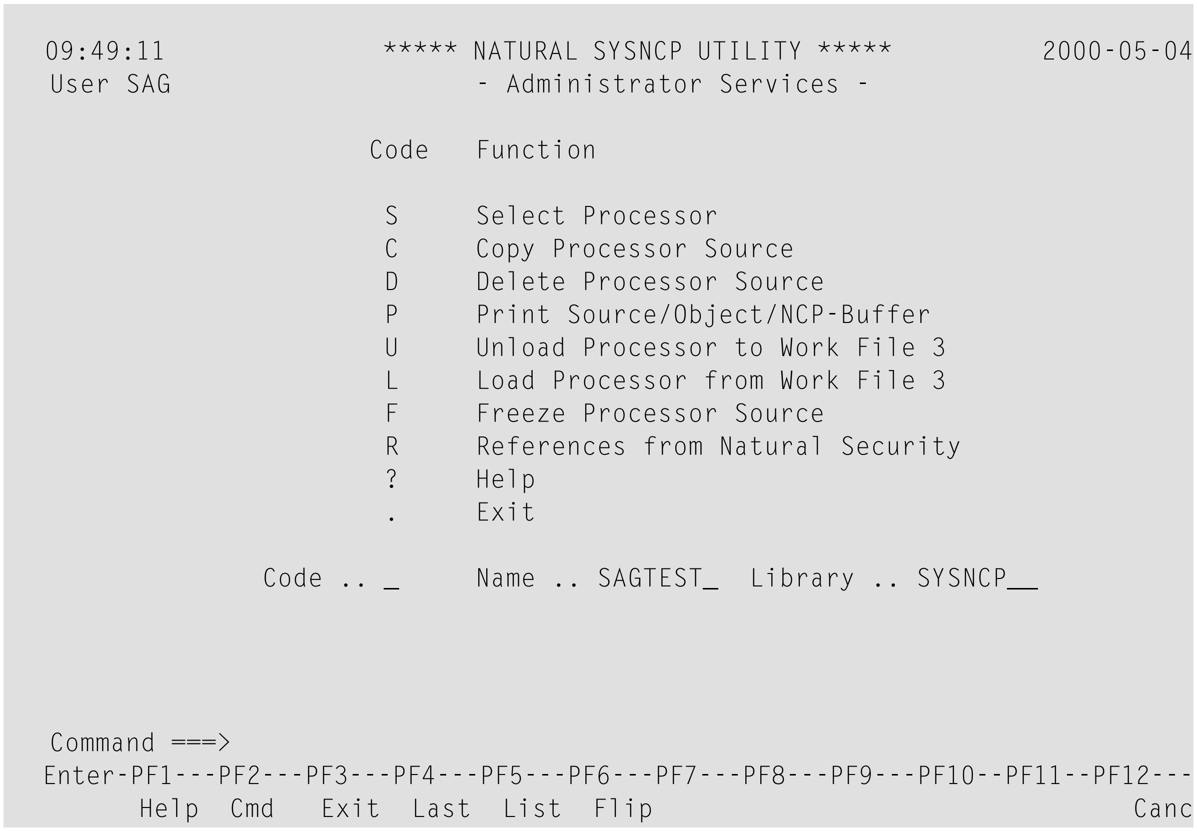
SYSNCP provides facilities for the administration of command processors. Only system administrators, as defined in *Natural Security*, are authorized to access these services.

To access the administrative services

1 In the **Processor Source Maintenance** menu, enter Function Code **A** (Administrator Services).

2 Press ENTER.

The Administrator Services screen is displayed:



```
 09:49:11              ***** NATURAL SYSNCP UTILITY *****          2000-05-04
 User SAG                     - Administrator Services -

                        Code   Function

                         S     Select Processor
                         C     Copy Processor Source
                         D     Delete Processor Source
                         P     Print Source/Object/NCP-Buffer
                         U     Unload Processor to Work File 3
                         L     Load Processor from Work File 3
                         F     Freeze Processor Source
                         R     References from Natural Security
                         ?     Help
                         .     Exit

                 Code .. _     Name .. SAGTEST_  Library .. SYSNCP__




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

Note: If you do not have Natural Security installed, be aware that all other users have administrator status.

Below is information on:

- Select Processor
- Copy Processor Source
- Delete Processor Source

- Print Source/Object/NCP Buffer
- Unload Processor
- Load Processor
- Freeze Processor Source
- References from Natural Security

Select Processor

See the section ***Processor Selection***.

Copy Processor Source

In copying processor sources, you have the choice of copying the entire processor or only selected sources (header, keywords, functions, runtime action definitions).

To copy a command processor

1 In the Administrator Services menu, enter Function Code **C**.

2 Press ENTER.

The Copy Processor Source window is displayed to provide source and target information:

```
                         Copy Processor Source

                  Source           Target

     Name ........ SAGTEST_        ________
     Library ..... SYSNCP__        SYSNCP__
     DBID ........ 10___           10___
     FNR ......... 32___           32___
     Password ....
     Cipher Key ..

     Replace ..... NO_
```

3 In the Source fields, enter the name of the processor to be copied, and the library, database ID (DBID) and file number (FNR) in which the processor is stored. The default values correspond to the processor specified on the **Administrator Services** menu.

In the **Target** fields, enter the name of the processor to be copied to, and the library, database ID (DBID) and file number (FNR) into which the processor is to be copied.

In the **Cipher Key** field, enter the appropriate password and/or cipher key if the source and/or target file is protected by a password and/or cipher key.

In the **Replace** field, enter `YES` if you want to overwrite a processor in the target environment. The default for this field is `NO`.

4 Press ENTER.

The following window is displayed to select sources:

```
                          Copy Processor Source

      Mark  Copy                          Source  Target
      ----  ----------------------------  ------  ------
       _    Header .....................    yes      no
       _    Keywords ...................    yes      no

       _    Functions ..................    yes      no
            Runtime Action Definitions ..   no       no


     Source Name SAGTEST     Library SYSNCP    DBID 10    FNR 32
     Target Name TEST2       Library SYSNCP    DBID 10    FNR 32

     Replace ... NO
```

5 In the appropriate **Mark** fields, enter any character to select the sources you want to copy.

6 Press ENTER.

Delete Processor Source

This function is used to delete processor sources.

To delete a command processor

1 In the **Administrator Services** menu, enter Function Code **D**.

2 Press ENTER.

The **Delete Processor Source** window is displayed.

3 Specify the name of the processor to be deleted, and the library, database ID and file number in which the processor is stored. If the file is protected by a password and/or cipher key, you also have to enter the appropriate password and/or cipher key.

4 Press ENTER.

The following window is displayed to select the sources to be deleted:

```
                    Delete Processor Source

        Mark  Delete                         Available
        ----  -----------------------------  ---------
         _    Header ......................    yes
         _    Keywords ...................     yes
         _    Functions ..................     yes
         _    Runtime Action Definitions ..    yes


        Name SAGTEST   Library SYSNCP    DBID 10      FNR 32
```

To the right of each processor source (header, keywords, functions, runtime action definitions) is a field which indicates whether the source exists. As command processor creation is a cumulative activity, you cannot delete a source without deleting all sources which are based on it. Thus, for example, in the screen above, you cannot delete the source of the functions without also deleting the source of the runtime action definitions.

5 In the appropriate **Mark** fields, enter any character to select each source indicated as **Available**.

6 Press ENTER.

Print Source/Object/NCP Buffer

In addition to processor sources, you can also print the processor object and the NCP.

To print a command processor item

1 In the **Administrator Services** menu, enter Function Code **P**.

2 Press ENTER.

The **Print Source/Object/NCP-Buffer** window is displayed.

3 Specify the name of the processor to be printed, and the library, database ID and file number in which the processor is stored. If the file is protected by a password and/or cipher key, you also have to enter the appropriate password and/or cipher key.

4 Press ENTER.

5 The following window is displayed to select items for printing:

```
                     Print Source/Object/NCP-Buffer

        Mark  Print                          Available
        ----  ----------------------------   ---------
         _    Header ......................  yes
         _    Keywords ....................  yes

         _    Functions ...................  yes
         _    Runtime Action Definitions ..  yes

         _    Processor Object ............  yes
              NCP-Buffer ..................  no

              Printer .....................  ________


        Name SAGTEST   Library SYSNCP    DBID 10      FNR 32
```

To the right of each processor source (header, keywords, functions, runtime action definitions) is a field which indicates whether the item exists.

Possible input values for the **Printer** field are the logical printer ID, VIDEO or SOURCE; see also DEFINE PRINTER in the *Statements* documentation.

6 In the appropriate **Mark** fields, enter any character to select the items you want to have printed and enter the logical printer name or the value VIDEO or SOURCE in the Printer field.

7 Press ENTER.

Unload Processor

To unload a command processor

1 In the **Administrator Services** menu, enter Function Code **U**.

2 Press ENTER. The **Unload Processor to Work File 3** window is displayed:

```
                      Unload Processor to Work File 3

                   Source           Target

     Name ........ SAGTEST_
     Library ..... SYSNCP__         SYSNCP__
     DBID ........ 10___
     FNR ......... 32___
     Password ....
     Cipher Key ..

     Report ...... NO_
```

3 In the **Source** fields, enter the name of the processor to be unloaded, the library, database ID, and file number in which the processor can be found; the default value is the processor specified in the **Administrator Services** menu. Enter the appropriate password and/or cipher key if the file is protected by a password and/or cipher key.

4 In the **Report** field, enter `YES` if you want a report to be produced. Default is `NO`. You do not have to use a file extension. If you wish to use an extension, you must use the file extension ".sag".

5 Press ENTER.

When the processor is unloaded, all processor sources (header, keywords, functions, runtime action definitions) are written to Work File 3.

Note: Use the Object Handler to transfer command processors from one hardware platform to another.

Load Processor

To load a command processor

1 In the **Administrator Services** menu, enter Function Code **L**.

2 Press ENTER.

The **Load Processor from Work File 3** window is displayed for loading processors from Work File 3 to a Natural library:

```
          Load Processor from Work File 3

     Replace existing processors .. N
     Produce load report .......... NO_
```

3 In the **Replace existing processors** field, enter **Y** or **N** (default is N) to specify whether existing processors with the same name are to be replaced by the processor to be loaded.

4 In the **Produce load report** field, enter `YES` (default is `NO`) if you want a report to be produced.

5 Press ENTER.

Note: Input for the processor name and the library into which the processor is to be loaded is taken from the work file.

Freeze Processor Source

You can freeze a processor in its current state to prevent users from modifying it further.

To freeze a command processor

1 In the **Administrator Services** menu, enter Function Code **F**.

2 Press ENTER. The **Freeze Processor Source** window is displayed.

3 Specify the name of the processor to be frozen, and the library, database ID and file number in which the processor is stored. If the file is protected by a password and/or cipher key, you also have to enter the appropriate password and/or cipher key.

4 Press ENTER.

5 In the following window, specify with **Y** or **N** whether modification of the processor sources is to be allowed or not. Default is **Y**.

6 Press ENTER.

References from Natural Security

This function is only available if Natural Security is active in your environment. It is used to delete functional security references from Natural Security.

If functional security is defined for a processor in Natural Security, references are created automatically. These references are stored in the FNAT/FUSER system files along with the processor sources, not in FSEC.

To invoke References from Natural Security function

1 In the **Administrator Services** menu, enter Function Code **R**.

2 Press ENTER.

   The **Delete References** window appears.

3 Specify the name of the processor, and the library, database ID and file number in which the processor is stored. If the file is protected by a password and/or cipher key, you also have to enter the appropriate password and/or cipher key.

4 Press ENTER.

5 In the following window, you can delete main references, function references and auxiliary references.

For further information on functional security for command processors, refer to the section *Functional Security* in the *Natural Security* documentation.

Session Profile

A session profile is a collection of user-definable defaults which determine how the SYSNCP screens appear or how SYSNCP reacts to input. In a session profile, for example, you can determine which command processor you want as default for a session or which colors you want assigned to screen attributes. In SYSNCP, there is a standard session profile called STANDARD which is issued to all new users. You can create several different session profiles and activate them as required.

Administrators for SYSNCP can access and modify any session profile in SYSNCP. Other users can access all session profiles, but can modify only those session profiles which are created under their user ID or which have the same name as their user ID.

To define or modify a session profile

- Issue the `PROFILE` command from the Command line of the **Processor Source Maintenance** menu.

  The first of three session profile maintenance screens is displayed.

Below is information on:

- Session Profile Name
- Session Parameters - Profile 1
- Color Attributes - Profile 2

- Miscellaneous Attributes - Profile 3

Session Profile Name

The standard profile STANDARD or the value of the system variable *USER is taken as default for the profile name.

If you are defining a new session profile, the parameters/attributes are defaults. You can modify these defaults as required and save them by entering the new name and pressing PF5.

The field Session Profile Name on each profile screen is both an input and output field. Thus, it is possible to define, read or save another profile from any of these screens by entering its name in the Profile Name field and pressing PF5 or PF4, respectively.

Session Parameters - Profile 1

On the first profile maintenance screen, you can modify the following fields:

Field	Explanation
Apply Terminal Control 1 Apply Terminal Control 2	These fields can be used to enter the parameters of a SET CONTROL statement to be issued by SYSNCP at startup. For example, when you enter **Z** in any of the fields, SYSNCP issues the statement `SET CONTROL 'Z'`.
Default Processor Name	The default command processor name to be used for this session.
Default Processor Library	The Natural library to be used to store a command processor.
Cancel Reaction	Specifies whether a warning is to be issued whenever the requested modification is not completed and the CANCEL command is issued. W Issue warning. B Back out and cancel without issuing warning.
Clear Key Allowed	Specifies whether clear key is allowed. N Clear key disallowed. Y Clear key active and has same effect as CANCEL.
Default Cursor Position	Specifies placement of the cursor. 1 Cursor to be positioned in first field of the screen. C Cursor to be positioned in command line.
Exec/Display Last Command	Specifies action to be taken as a result of the LAST command: E Execute last command issued in command line. D Display last command issued in command line.

Color Attributes - Profile 2

On the second profile maintenance screen, you can assign colors to various screen attributes, or overwrite existing color assignments.

By specifying the following color codes, you can assign the following colors:

Code	Color
BL	Blue
GR	Green
NE	Neutral
PI	Pink
RE	Red
TU	Turquoise
YE	Yellow

For color assignments to screen attributes, see also the terminal command %= in the *Terminal Commands* documentation.

Miscellaneous Attributes - Profile 3

The following attributes can be specified on the third profile maintenance screen:

Field	Explanation
Message Line Position	The line on which messages are to be displayed. The value 21 is recommended. See also the terminal command %M in the *Terminal Commands* documentation for more information.
Text for PF5 Key	The PF5 function key is reserved for global (session-wide) use. The text to be displayed on the PF-key line for PF5 can be entered in this field.
Command for PF5 Key	The PF5 function key is reserved for global (session-wide) use. The command to be executed when PF5 is pressed can be entered in this field.

In addition, the screen displays when and by which user this profile was last modified.

XI SYSPCI Utility - Product Configuration and Initialization

67 SYSPCI Utility - Product Configuration and Initialization

The SYSPCI utility is used after a first-time installation of Natural or one of its add-on products which uses the Software AG Installer. It sets up a number of files, parameters and individual settings depending on your environment.

Important: If you want to use the SYSPCI utility, you must have been defined as a Natural administrator in the local configuration file (see also *Administrator Assignments* in the *Configuration Utility* documentation). You will be able to invoke the SYSPCI utility even if you are not a Natural administrator, however, when you start the configuration of the selected product, the SYSPCI utility will not be able to load the initialization files and the Natural startup error 12 will occur.

Using the SYSPCI utility, you can do the following:

- Enter the necessary information for the required Adabas files for your products, and add these files if they do not exist.
- Enter the database IDs of the required Adabas files into Natural's global configuration file.
- Enter the database IDs and file numbers of the new or existing Adabas files into the default parameter files for your products.
- Initialize your product.
- Optional, depending on the selected product: execute additional functions (such as loading product data).

Note: After an update installation, you need not invoke the SYSPCI utility if the required Adabas files and the parameters in the required default parameter files have already been set up previously (for example, after a first-time installation). Previously set up parameters will be kept with an update installation.

You can call the SYSPCI utility in different ways, as described in the following topics:

Configuring the Installed Products Using a Screen

You can configure the installed Software AG products that the SYSPCI utility can detect in your environment.

The description below provides general information on how to use the SYSPCI utility, and it explains the options that are normally available for all products. For detailed information on the files that need to be set up for a specific product, see the installation documentation for that product.

To configure an installed product

1 Enter the following command:

```
SYSPCI
```

Note: If you invoke the SYSPCI utility in an environment which is protected by Natural Security, Natural Security will validate the utility profile for SYSPCI.

A screen appears which lists the installed Software AG products that have been detected in your environment. For example:

```
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 Mark the product that you want to configure and press ENTER.

Important: It is recommended that you configure your products in the same sequence as listed on the screen.

The content of the resulting screen depends on the selected product. For example:

```
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 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 If you mark **Use existing Adabas file** and press PF2, a new screen will be shown.

The content of the resulting screen depends on the selected product. For example:

```
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  ↩
  ↩
```

You can select the following

Option	Description
List FDU file for `ppp-nnnn`	This option will list the FDU file for the single product file. Where `ppp` in this case represents the product code (NDV in the above screen) and `nnnn` the file name (FDIC in the above screen).
List `ppp`PARM (parameter file)	This option will list the default parameter file for the product. `ppp` in this case represents the product code (NDV in the above screen).
Select a parameter file	This option will list any existing parameter file and allow you to select it.

For any file that is listed, you can change the file contents by switching to EDIT mode using either the command `EDIT` or pressing PF11. Use command `COMMANDS` or press PF10 to show the available commands.

4 Specify the following information for the selected product:

Option	Description
Create new Adabas file or Use existing Adabas file	If **Create new Adabas file** is marked, the first file must not exist and a new Adabas file will be created. Depending on the selected product, there may be more than one file. If one or more of the other files already exist they can be used for the product. If **Use existing Adabas file** is marked, the first file must exist. Depending on the selected product, there may be more than one file. If one or more of the other files do not exist, they can be created. If the Adabas file exists already, however, the SYSPCI utility will only check whether the file has the correct structure (FDT). In both cases (new and existing file), the following actions will be performed: ■ The database ID of each required Adabas file will be entered into Natural's global configuration file. ■ The database ID and file number of each new or existing Adabas file will be entered into the default parameter file for your product.
Initialize product	This option is only available for products which have an initialization program. If marked (default), the initialization program for the selected product will be loaded and executed. If you want to activate the product, you have to mark this option. **Note:** If you want to use this option, the database for which you specify the database ID must be online.
Database ID	The database ID of the Adabas file. When you mark the **Select** field next to **Database ID** or when you press PF5, a list of all databases is shown which can be found on the machine. The list also shows whether a database is currently online or offline. You can then mark a database in the list to select it.

Option	Description
File number	The number of a file in the selected database. This can be the number of an existing file or for a new file. When you mark the **Select** field next to **File number** or when you press PF6, a list of all files is shown which can be found for the specified database ID. You can then mark a file in the list to select it. The Adabas system files are also shown in the list, so that you can see which file numbers have already been assigned. However, you must not mark an Adabas system file. Otherwise, an error will occur. When you specify the number for a new file, make sure that the **Create new Adabas file** option is marked.
File name	The name of the Adabas file. When you have selected an existing file, the corresponding file name is automatically shown. This name cannot be changed. When you have specified a new file number which does not yet exist, you can enter a file name (optional). If you do not enter a file name, a product-specific default name will be used.

For some products (such as Predict), you have to specify additional options. See the installation documentation for that product for further information.

After you have entered information for an option, you can press ENTER or PF4 to validate your input.

5 In the **Start selected action(s)** field, enter "Y" and press ENTER to start the configuration of the selected product.

After the selected actions for the selected product have been performed, a message such as the following is shown:

```
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 Press ENTER to proceed.

The initial screen of the SYSPCI utility is shown again and you can configure further products. The configuration of Natural Security, however, is an exception. In this case, Natural is terminated after the initialization program has been executed.

Calling the SYSPCI Utility with Direct Command Data

You can call the SYSPCI utility using a direct command that consists of keywords and their corresponding values. Thus, you can also use the SYSPCI utility in batch mode.

Exception: When the initialization of the Adabas file for Natural Security has been completed (by loading the initialization program with the INPL utility and executing it), the Natural session is terminated by the INPL utility. Therefore, it is not possible to execute any additional commands after this step.

You can use the following keywords with the `SYSPCI` command (see also the examples below):

Keyword	Meaning
`PRODUCT` *	Product to be processed. Valid values: `NDV` for Natural Development Server. `NCP` for Natural Command Processor. `PRD` for Predict. `CST` for Construct (alternative to `NBS`). `NBS` for Natural Business Services. `NEE` for Natural Engineer `SAT` for System Automation Tools `NOM` for Entire Output Management `NOP` for Entire Operations. `NSL` for Natural Security Log. `NSC` for Natural Security. **Important:** It is recommended that you configure your products in the same sequence as listed above.
`DBID` *	Database ID of the Adabas file. **Note:** When using `FUNCTION ADU` or `ALU` you can leave this blank. In this case, the SYSPCI utility will use the DBID value found in the text member INST-<*productcode*> in library SYSPCI.
`DBID2` **	Database ID of the second Adabas file if `PRODUCT` is `PRD`, `NBS`, `NEE`, `SAT`, `NOM` or `NOP`.
`DBID3` **	Database ID of the third Adabas file if `PRODUCT` is `NBS` or `NOM`.
`FNR` *	File number of the Adabas file. **Note:** 1. When using `FUNCTION ADU` or `ALU` you can leave this blank. In this case, the SYSPCI utility will use the FNR value found in the text member INST-<*productcode*> in library SYSPCI. 2. When using `FUNCTION ADA` or `ALL` you can enter -1. In this case, the SYSPCI utility will use the next free file number.

Keyword	Meaning
FNR2 **	File number of the second Adabas file if PRODUCT is PRD, NBS, NEE, SAT, NOM or NOP.
FNR3 **	File number of the third Adabas file if PRODUCT is NBS or NOM.
<u>FUNCT</u>ION or FCT	Function to be executed. Valid values: ADA: Create new Adabas file. ADU: Use existing Adabas file. This requires entering a valid DBID and FNR (see above) or entering the value ADR: Use existing Adabas file on a remote system. This requires entering a valid DBID and FNR (see above) or entering the value INT: Load and execute initialization program. INR: Load and execute initialization program using Adabas file on a remote system ALL: Both (ADA and INT). Default. ALU: Both (ADU and INT). ALR: Both (ADR and INR).
<u>FILE-N</u>AME	Name of the Adabas file if FUNCTION is ADA or ALL. Valid values: 16 characters without blanks.
FILE-NAME-2 or FILE-N2	Name of the second Adabas file if PRODUCT is PRD, NBS, NEE, SAT, NOM or NOP. Valid values: 16 characters without blanks.
FILE-NAME-3 or FILE-N3	Name of the third Adabas file if PRODUCT is NBS or NOM. Valid values: 16 characters without blanks.
<u>SUBF</u>UNCTION	Additional function to be executed. The valid values depend on the product. If PRODUCT is PRD: ■ PRC: Convert FDIC data. ■ PRP: Load FDIC description. ■ PRD: Load example data. ■ PRA: Both (PRP and PRD). ■ PR1: Convert FDIC data and load FDIC description. ■ PR2: Convert FDIC data and load example data. ■ PR3: Convert FDIC data and load FDIC description and example data. If PRODUCT is NBS or CST: ■ LDC: Load Construct data. ■ LDP: Load Predict data. ■ LDA: Both (LDC and LDP). If PRODUCT is NOM: ■ LDM: Load NOM data. If PRODUCT is NEE: ■ LHD: Load NEE help data.

Keyword	Meaning
`END`, `STOP`, `EXIT`, `QUIT` or `.`	Exit the SYSPCI utility. The keyword must be entered as a single command.
`FIN`	Exit the SYSPCI utility and terminate the Natural session. The keyword must be entered as a single command.

Notes:

1. The keywords marked with an asterisk (*) are mandatory.
2. The keywords marked with two asterisks (**) are mandatory for the corresponding products.
3. All other keywords are optional.

Examples

- **Batch Mode**
  Commands in the batch input file which is defined by the `CMSYNIN` profile parameter:

  ```
  SYSPCI
  FIN
  ```

  Data in the batch input file which is defined by the `CMOBJIN` profile parameter:

  ```
  FUNCTION ALL PRODUCT PRD DBID 77 FNR 2002 DBID2 12 FNR2 2003
  FUNCTION ALL PRODUCT NSC DBID 77 FNR 1600
  END
  ```

  See also *Natural in Batch Mode* in the *Operations* documentation.

- **Interactive Mode - Natural Command Line**

  ```
  SYSPCI FUNCTION ALL PRODUCT NSL DBID 77 FNR 1601
  ```

- **Interactive Mode - Natural Stack**

  ```
  natural stack='(SYSPCI FUNC ALL PROD NCP DBID 77 FNR 1501: PROD NSL DBID 77 FNR ↩
  1601; FIN)'
  ```

XII 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.

Note: 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.

68 Profiling Natural Applications

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

- Profiles interactive Natural applications from UNIX, Windows or mainframe platforms in an Eclipse-based development environment.
- Reads and analyzes Profiler resource files containing event data collected by the mainframe Profiler utility in batch mode or by the Natural Profiler for UNIX and Windows.
- Provides features for big data handling:
  - Event filter,
  - Sampling technique,
  - Data consolidation.
- Performance analyses of programs, statements and program lines:
  - CPU time,
  - Elapsed time,
  - Hit count.
- Displays an event trace.
- Provides direct navigation from a profiled program line to the corresponding source code.
- Saves and reloads the Profiler data as an XML-formatted file.

Features of the Natural Profiler for UNIX and Windows

- Profiles interactive or Natural batch applications from UNIX or Windows platforms.
- Provides features for big data handling:
  - Event filter,
  - Sampling technique,
  - Data consolidation.
- Saves the Profiler data as a Profiler resource file.

Features of the Profiler Utility

Online Mode (Mainframes)

- Profiles interactive Natural applications from mainframe platforms.
- Provides an event filter.
- Displays an event trace.
- Saves the Profiler data in a table format.
- Saves the Profiler data as a Profiler resource file.

Note: The amount of data collected by the Profiler utility in online mode is restricted by the relatively small size of the Natural Data Collector buffer which works in a wrap-around mode. Moreover, when running under CICS or Com-plete, the CPU time is not provided. In general, we recommend that you use the NaturalONE Profiler for profiling interactive Natural mainframe applications because the NaturalONE Profiler has no size restrictions and supports CPU performance analyses.

Batch Mode (Mainframes)

- Profiles Natural batch and Natural RPC applications from mainframe platforms.
- Profiles mainframe interactive applications executed remotely from Natural Studio.
- Provides features for big data handling:
  - Event, program, count and time filters,
  - Sampling technique,
  - Data consolidation.
- Saves Profiler data as a Profiler resource file.
- Reads and analyzes Profiler resource files.
- Prints program and event traces.
- Analyzes program performance.
- Collects and displays Profiler properties and statistics.
- Exports Profiler data for MashZone visualization.

Batch Mode (UNIX and Windows)

- Reads and analyzes Profiler resource files.
- Provides features for big data handling:
  - Data consolidation.
- Saves consolidated Profiler data as a Profiler resource file.
- Prints program and event traces.
- Analyses program performance.

- Displays Profiler properties and statistics.
- Exports Profiler data for MashZone visualization.

Features of the Natural Profiler MashApp

- Visualizes Profiler data on a graphical, interactive MashZone dashboard.
- Analyzes application performance with selection criteria such as library, program, program line and user:
  - CPU time,
  - Elapsed time,
  - Adabas command time,
  - Hit count.
- Displays Profiler properties and statistics.

Natural Profiler Evaluations

The evaluation criteria provided by the Natural profiling tools are summarized in the following table:

Evaluation	Profiling Tool	Description
Program Summary	Profiler utility (batch)	Shows the CPU time spent for each Natural object that executed and the Natural events that occurred in an object. See also *Example of a Program Summary*.
Line Summary	Profiler utility (batch)	Shows the CPU and elapsed time spent during Natural program execution for each individual source line and the number of Natural events that occurred in the line. See also *Example of a Line Summary*.
Hot Spots	NaturalONE Profiler	Shows the CPU and elapsed time used by Natural objects, statements and program lines and how often an object or statement executed. From a profiled program line you can directly navigate to the corresponding source code line. See also the appropriate description of hot spots in *Using the Natural Profiler* in the *NaturalONE* documentation.
MashZone Evaluation	Natural Profiler MashApp	Visualizes the Profiler data on an interactive MashZone dashboard. You can evaluate the distribution of the CPU and

Evaluation	Profiling Tool	Description
		elapsed time, the Adabas command time or the hit count and select criteria for the distribution. See also the example of a **MashZone evaluation**.
Program Trace	Profiler utility (batch)	Shows the program flow of the profiled application in the call hierarchy and the number of events that occurred. See also *Example of a Program Trace*.
Event Trace	NaturalONE Profiler, Profiler utility (batch)	Lists the recorded event data in chronological order. See also *Example of an Event Trace* and the appropriate description in *Using the Natural Profiler* in the *NaturalONE* documentation.
MashZone Profiler Properties, **Profiler Statistics**	Natural Profiler MashApp, Profiler utility (batch)	Lists Profiler properties such as the Profiler revision, and statistics of the monitored application that show, for example, the total CPU and elapsed time. See also the example of **MashZone Profiler properties** and **Profiler Statistics**.

69 Code Coverage of Natural Applications

This document provides general information on code coverage of Natural applications.

Introducing Code Coverage

In general, code coverage measures the degree to which the source code of a program is executed. It is often used for systematic software testing. The higher the code coverage percentage is, the lower is the chance that the code contains undetected software bugs in code that is not executed.

The Natural code coverage is used to monitor the executed statements of a Natural application. It collects the coverage data while the application is executed and provides tools to analyze the collected data afterwards.

The **Code Coverage** view of NaturalONE (Software AG's Eclipse-based development environment) and the **Program Coverage** table of the Profiler utility show - expressed as a percentage - how many of the statements of the Natural objects have been executed. The Natural Coverage Plugin for Jenkins visualizes the outcome of a Natural coverage cycle directly in the Jenkins job result pages.

In the NaturalONE editor and in the **Statement Coverage** table of the Profiler utility you can see, which individual statement lines of a Natural object have been executed. Here you can also see the statement lines which have been missed or have only been partly covered.

If a statement source uses multiple lines, only the line in which the statement begins is mentioned in the coverage reports.

You can export the coverage data with the Profiler utility output in text or table (CSV) format. The CSV table can be analyzed with a spreadsheet software such as Microsoft Excel.

GP and Source Coverage

If a Natural source contains `INCLUDE` statements, the corresponding copycode is included in the generated object (the **GP**). For the coverage, we can monitor two statement counts:

1. The number of the statements in the GP which includes all copycodes recursively (a copycode can include further copycodes).
2. The number of the statements in the source which does not include the copycodes.

The GP coverage reflects the percentage of the covered statements in the GP including copycodes; whereas the source coverage reflects the percentage of the covered statements in the source not including copycodes.

Platform-Specific Code Coverage

You can perform the Natural code coverage on UNIX, Windows and mainframe platforms. How to proceed depends on the platform and the application processing mode used:

Mainframes

- Code coverage of mainframe interactive applications remotely executed from Natural Studio or RPC is performed using the Profiler utility in batch mode.
- Code coverage of mainframe batch applications is performed using the Profiler utility in batch mode.

Note: Code coverage is not available for mainframe interactive applications running locally on a mainframe or remote from NaturalONE.

UNIX and Windows

- Code coverage of UNIX and Windows interactive applications is performed with the NaturalONE code coverage or the Natural code coverage for UNIX and Windows, respectively.
- Code coverage of UNIX and Windows batch applications is performed with Natural code coverage for UNIX and Windows, respectively.

Code Coverage Tools

This section summarizes the key features of the Natural profiling tools:

- Features of the NaturalONE Code Coverage
- Features of the Natural Code Coverage for UNIX and Windows
- Features of the Profiler Utility

- Features of the Natural Code Coverage Spreadsheet

Features of the NaturalONE Code Coverage

- Reads and analyzes Natural code coverage resource files containing coverage data collected by the mainframe Profiler utility in batch mode or by the Natural code coverage for UNIX or Windows.
- Interactive Natural applications from UNIX or Windows can be covered by activating the Natural code coverage for UNIX or Windows and reading the corresponding Natural code coverage resource file.
- The Natural code coverage view shows which percentage of the statements of the Natural objects have been executed.
- From the Natural code coverage view the involved Natural objects can be edited. The NaturalONE editor displays all covered lines with a green background.

Note: Interactive code coverage of Natural applications from mainframe platforms is currently not supported.

Features of the Natural Code Coverage for UNIX and Windows

- Code coverage of interactive or Natural batch applications from UNIX or Windows platforms.
- Provides features for big data handling:
  - Automatic event filter,
  - automatic data consolidation.
- Saves the code coverage data as a Natural code coverage resource file.

Features of the Profiler Utility

Batch Mode (Mainframes)

- Code coverage of Natural batch applications from mainframe platforms.
- Code coverage of mainframe interactive applications remotely executed from Natural Studio or against a Natural RPC server.
- Provides features for big data handling:
  - Program, count and time filters,
  - automatic event filter,
  - automatic data consolidation.
- Saves code coverage data as a Natural code coverage resource file.
- Reads and analyzes Natural code coverage resource files.

- Lists the **Program Coverage** table for all accessed Natural objects with
  - percentage of the covered statements,
  - number of covered statements,
  - number of missed (not covered) statements and
  - total number of statements of the object.
- The **Statement Coverage** lists the source of each accessed Natural objects and shows for each line the percentage of the covered statements.
- Exports Natural code coverage data in CSV (comma-separated values) format which can be further analyzed with a spreadsheet software (e.g. Microsoft Excel).
- Collects and displays Profiler and code coverage properties and statistics.

Note: On the mainframe, there is no one-to-one relationship between a Natural source code statement and the corresponding object code in the cataloged object. The Natural code coverage on the mainframe monitors the object code rather than the Natural source code. Therefore, multiple Natural statements can be merged into one coverage entry and conversely, one Natural statement can cover multiple coverage entries.

Batch Mode (UNIX and Windows)

- Reads and analyzes Natural code coverage resource files.
- Lists the **Program Coverage** table for all accessed Natural objects with
  - percentage of the covered statements,
  - number of covered statements,
  - number of missed (not covered) statements and
  - total number of statements of the object.
- The **Statement Coverage** lists the source of each accessed Natural objects and shows for each line the percentage of the covered statements.
- Exports Natural code coverage data in CSV (comma-separated values) format which can be further analyzed with a spreadsheet software (e.g. Microsoft Excel).

Note: On Windows and UNIX, missed statements are not collected. Therefore the Statement Coverage can only mark lines containing covered statements and the coverage of these lines is always 100%.

Features of the Natural Code Coverage Spreadsheet

- Template for coloring the Natural code coverage data exported in CSV (comma-separated values) format by the Natural Profiler utility.
- Program and copycode coverage with source and GP counters for
  - percentage of the covered statements,
  - number of covered statements,
  - number of missed (not covered) statements and
  - total number of statements of the object.
- Statement Coverage of the object source whereby the lines are colored in
  - green – if all statements of the line are covered,
  - yellow – if the statements of the line are partly covered,
  - red – if all statements of the line are missed,
  - gray – if the line is empty or contains only comments.
- Profiler and code coverage properties and statistics (for mainframe data).

Note: A Microsoft Excel spreadsheet template for Natural code coverage is available as a resource in the Natural Profiler library `SYSPRFLR` on UNIX and Windows.

Natural Code Coverage Evaluations

This section describes the evaluations provided by the Natural code coverage tools:

Program Coverage

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
```

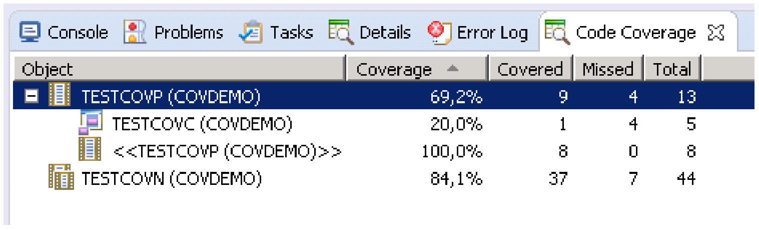
In the **Program Coverage** example above, 69.2% of 13 statements in the `TESTCOVP` program were covered, corresponding to 9 covered and 4 missed statements. 80.7% of the statements of the accessed objects in the library `COVDEMO` were covered, which is also the total value for the whole application 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:



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

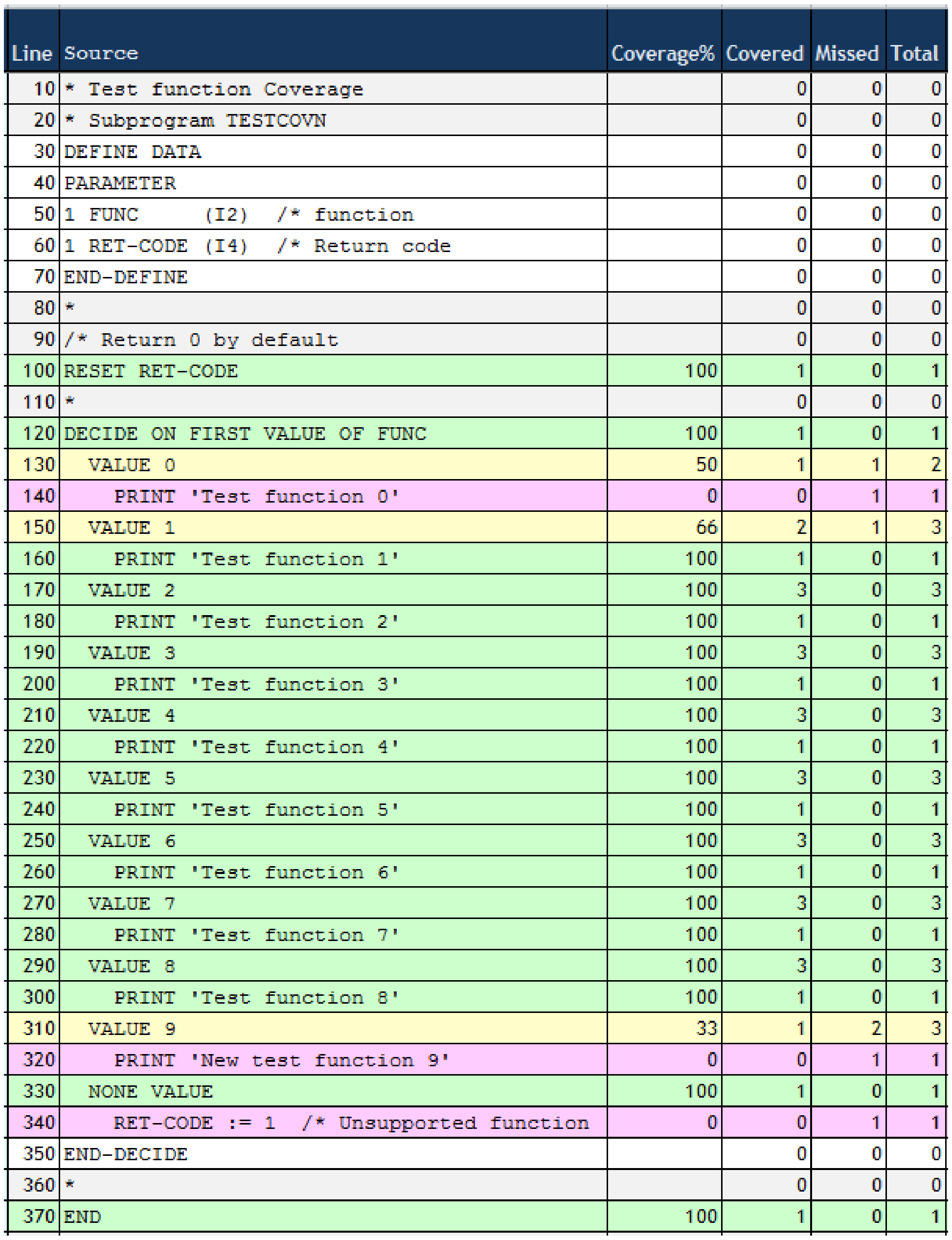
The statement coverage shows which lines of the program have been executed. For mainframe data, the Profiler utility also indicates which lines containing statements have not been executed or are only executed partly.

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

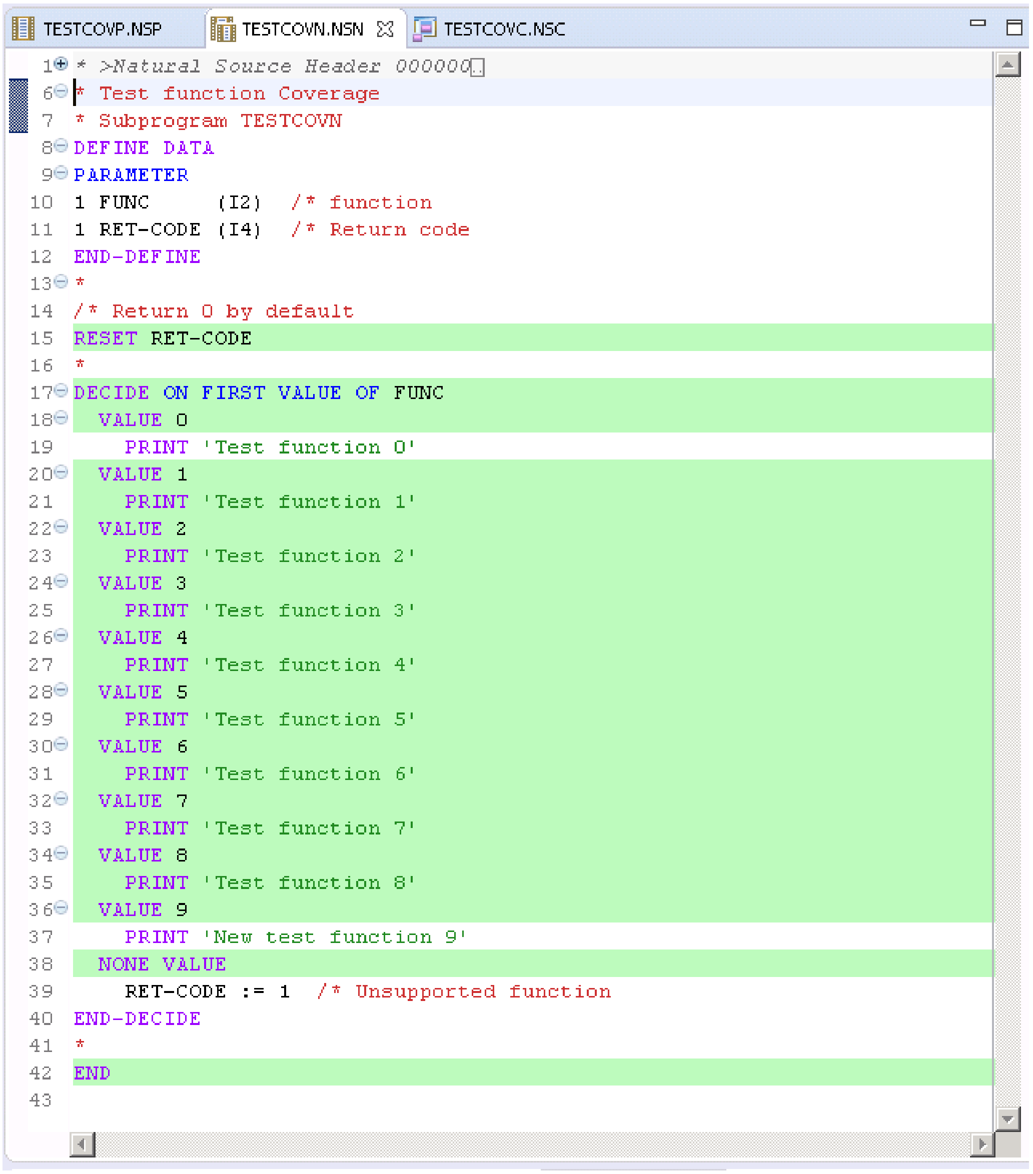
The three red lines of the subprogram `TESTCOVN` have not been executed. Thus the test run does not cover the new test function `9`. It also neither covers the (old) function `0` nor the case when the subprogram is called with an unsupported function.

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:



```
 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
```

70 Basic Concepts of the Profiler Utility

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.

Notes:

1. An NPRC resource file that has been consolidated on the server side contains the same hot spot values as the corresponding unconsolidated NPRF resource but opens much faster with NaturalONE.
2. The consolidated data does not contain the event history (timestamps). Therefore, it is not possible to view the event trace when you open an NPRC resource in NaturalONE.
3. Data consolidation is a prerequisite if you want to analyze the event data in MashZone by using the **Natural Profiler MashApp**.

Natural Code Coverage

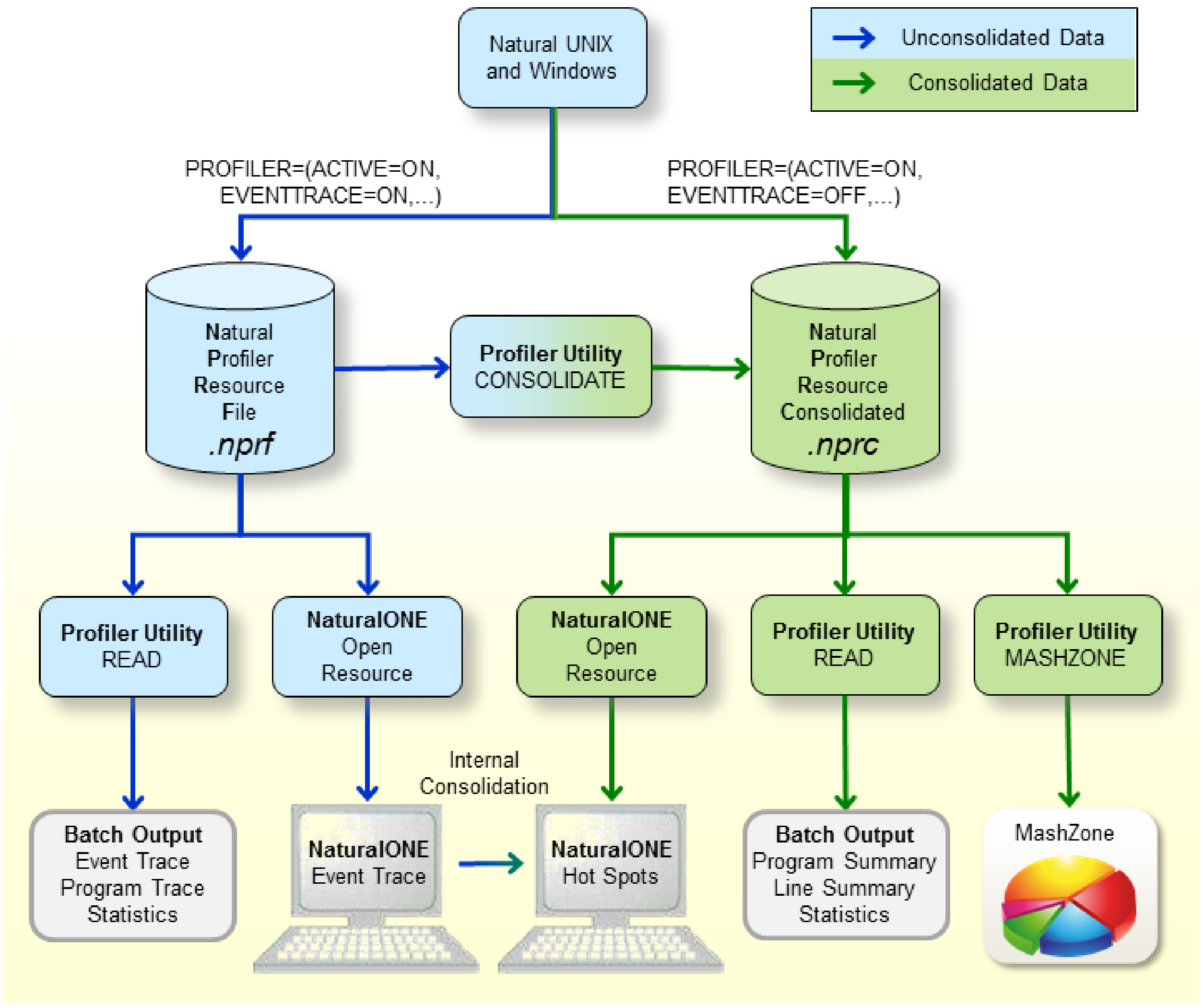
Natural code coverage is used to monitor executed and not-executed statements of a Natural application. It is started by switching on the `ACTIVE` subparameter of the `COVERAGE` profile parameter described in the *Parameter Reference* documentation.

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:


Natural UNIX and Windows
Unconsolidated Data
Consolidated Data
PROFILER=(ACTIVE=ON, EVENTTRACE=ON,...)
PROFILER=(ACTIVE=ON, EVENTTRACE=OFF,...)
Natural Profiler Resource File .nprf
Profiler Utility CONSOLIDATE
Natural Profiler Resource Consolidated .nprc
Profiler Utility READ
NaturalONE Open Resource
NaturalONE Open Resource
Profiler Utility READ
Profiler Utility MASHZONE
Batch Output Event Trace Program Trace Statistics
NaturalONE Event Trace
Internal Consolidation
NaturalONE Hot Spots
Batch Output Program Summary Line Summary Statistics
MashZone

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.

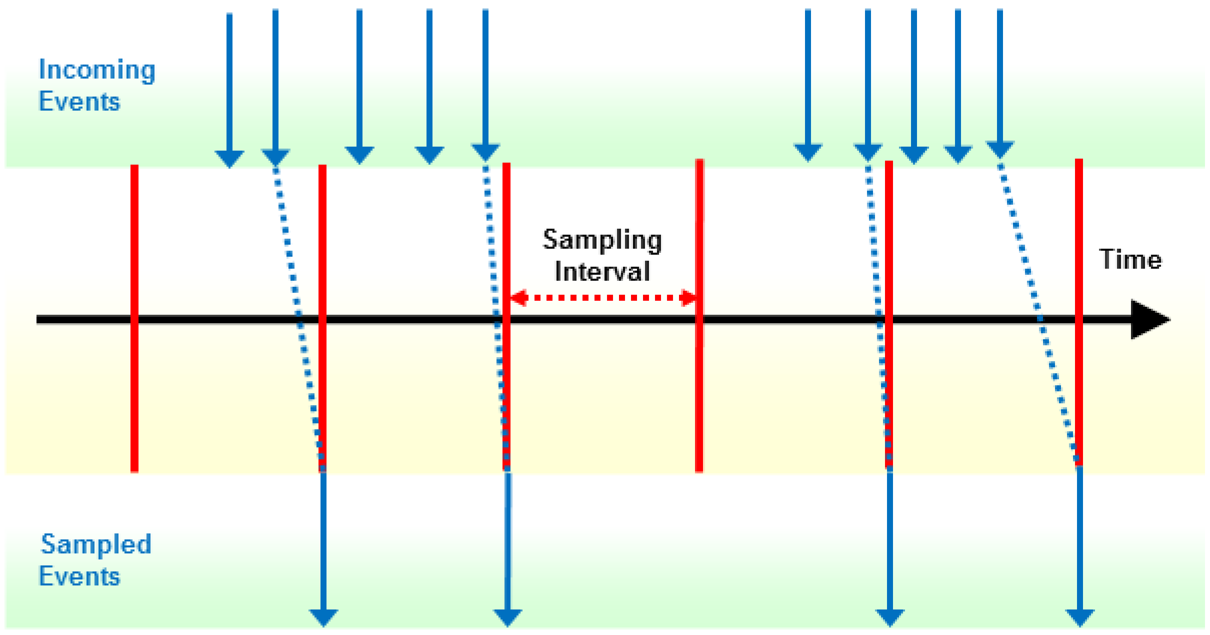
- The Profiler utility `READ` function reads the event data from the NPRF resource file. It provides an event trace, a program trace and the Profiler statistics. The resulting data can be exported to a file in text or CSV (comma-separated values) format.
- If the NPRF resource file is opened from NaturalONE, 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, a line 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:

`Number of events ≈`	$\frac{\texttt{Total CPU time in microseconds}}{\texttt{Sampling interval}}$

Example

In the following example application, the program `XPROF` calls three subprograms. The application is profiled twice:

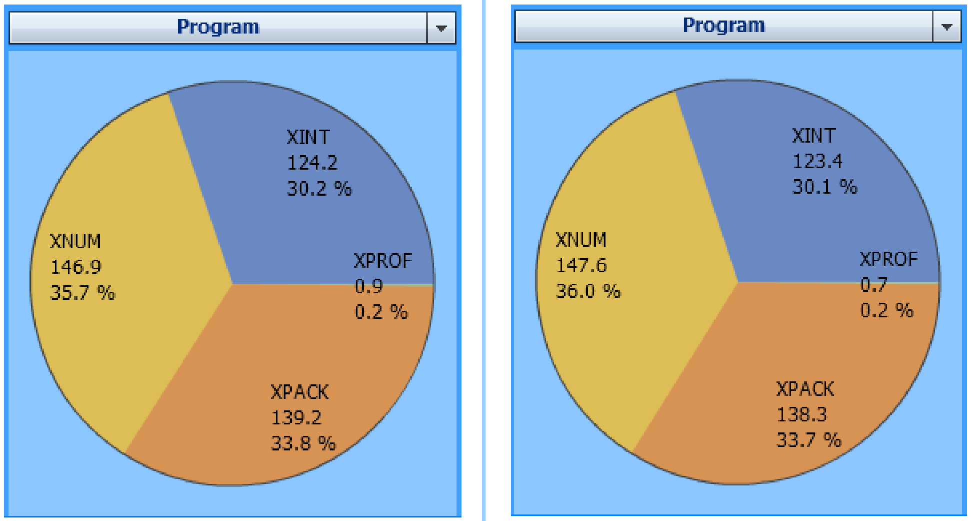
1. Without sampling.

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	Sampling 100 µs
411.4 ms total CPU time	410.0 ms total CPU time
240,086 events	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

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` sub-parameter 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.

Note: 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.

71 Using the Profiler Utility

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.

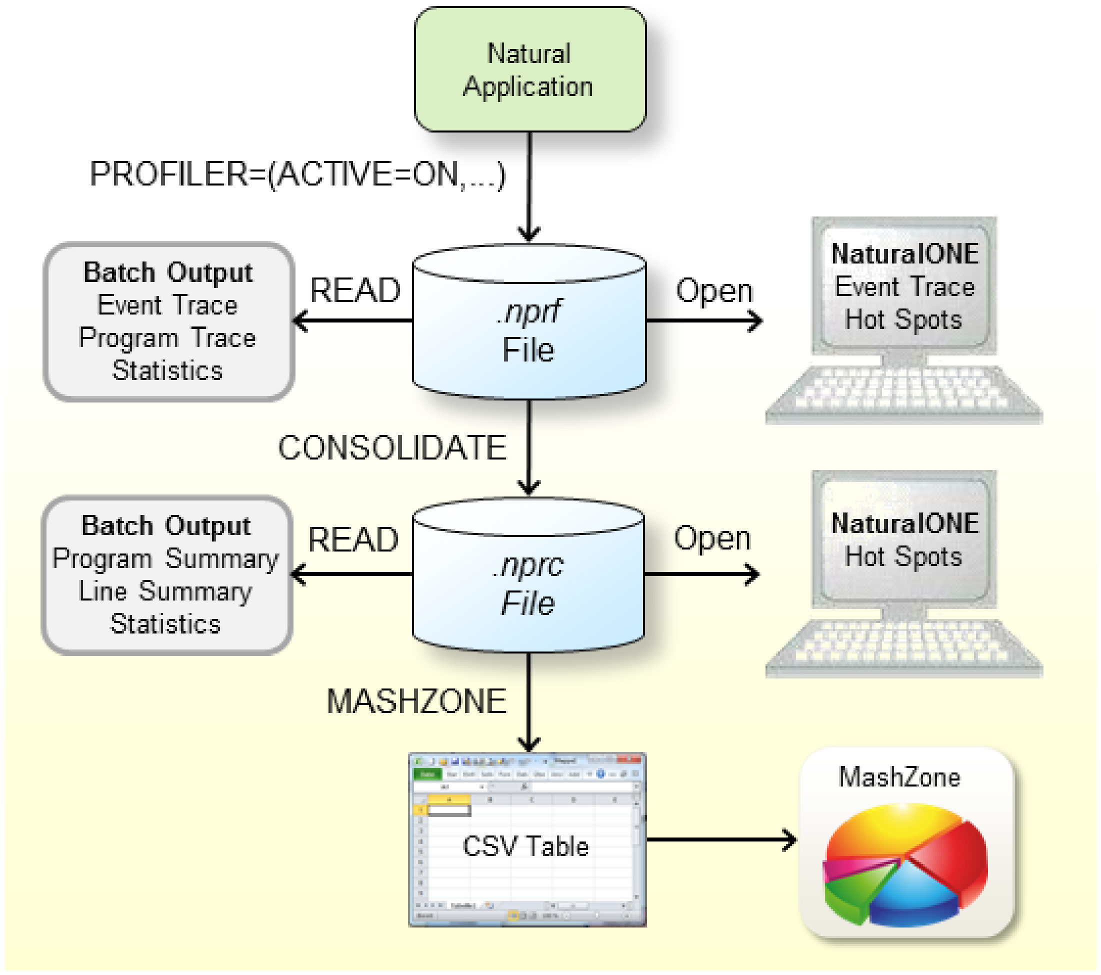
   You can output statistics, a program summary, a line summary, a program trace, an event trace with the most important data, and reports on program and statement coverage. You can export the resulting data in text or CSV (comma-separated values) format.

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, a line 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
  LINE=ON                 /* Print line 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.

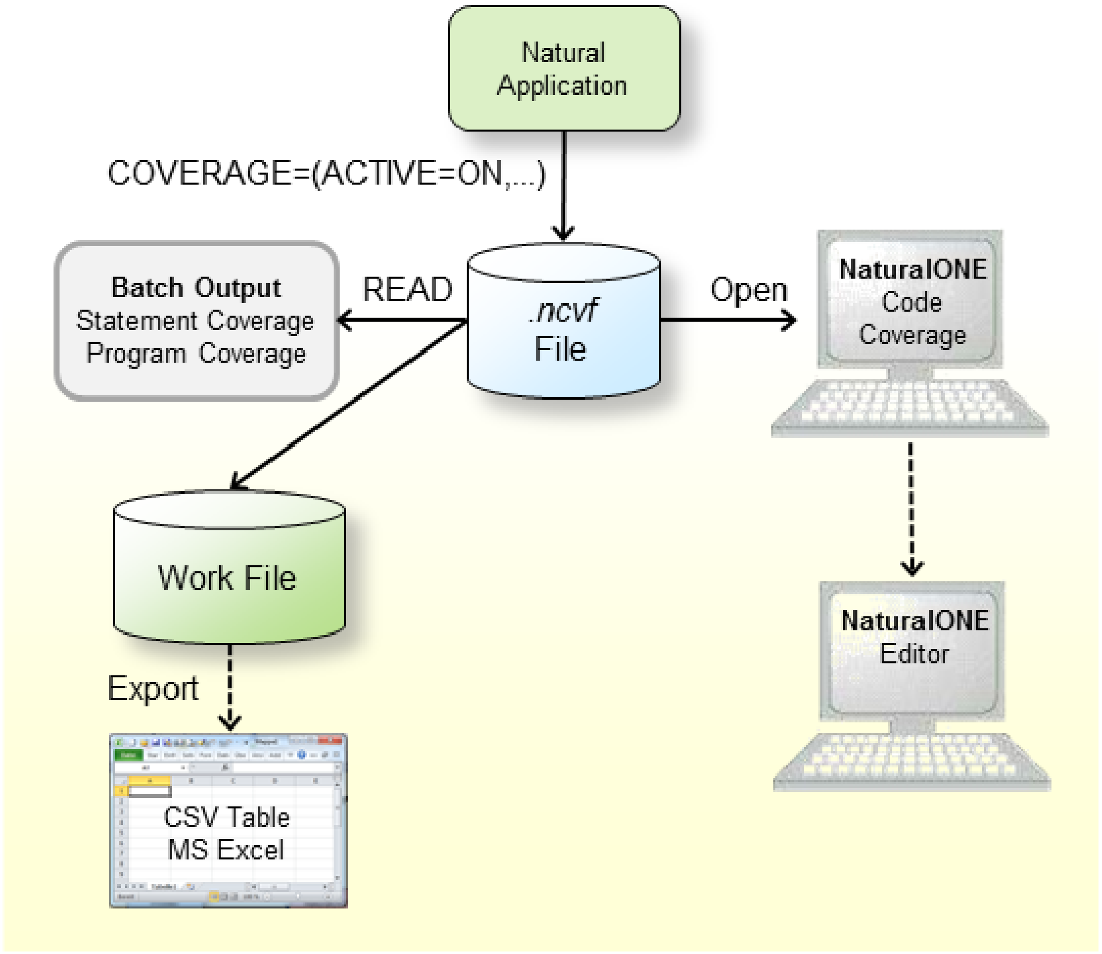
Notes:

1. If the resource name is not explicitly specified in the `READ`, `CONSOLIDATE` or `MASHZONE` function of the Profiler utility, the last created NPRF or NPRC resource in the library is used.
2. If you plan to profile a long-running batch application, refer to the section ***Profiling Long-Running Applications***. It covers strategies of how to minimize the number of events to be monitored.
3. The NaturalONE Profiler is described in the *NaturalONE* documentation.

Quick Start for Code Coverage

This section briefly describes the steps required for performing the code coverage of a Natural batch applications and viewing the results. The instructions provided here may serve as a guideline when starting to use **Natural code coverage**. Detailed information regarding the steps is provided in the remainder of this chapter.

The steps to take depend on the evaluation you want to perform for your application as illustrated in the following graphic:



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.

Notes:

1. If the resource name is not explicitly specified in the `READ` function of the Profiler utility, the NCVF resource created last in the library is used.
2. The NaturalONE **Code Coverage** view and editor are described in the *NaturalONE* documentation.

Prerequisites

The following 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.

Notes:

1. Natural statement events (`NS`) are only generated for profiling if the corresponding Natural object was compiled with the profile parameter `GPGEN` set to `(PROFILER=ON)`.`GPGEN` is described in the *Parameter Reference* documentation.
2. Code coverage of a Natural application can only be performed if the corresponding Natural objects were compiled with the profile parameter `GPGEN` set to `(COVERAGE=ON)`.

Invoking and Terminating the Profiler Utility

This section provides instructions for invoking and terminating the Profiler utility in batch mode.

To invoke the Profiler utility

- Enter the following system command into the primary command input data set `CMSYNIN`:

```
PROFILER
```

Note: After the `PROFILER` system command, the Profiler expects one or more lines with Profiler keyword entries.

To terminate the Profiler utility

- Enter the following Profiler keyword into the primary command input data set `CMSYNIN`:

  ```
  END-PROFILER
  ```

  Or:

  ```
  END
  ```

  Or:

  ```
  .
  ```

Syntax and Keywords

The Profiler utility in batch mode reads the Profiler keywords that control the profiling from the primary command input data set `CMSYNIN`. The Profiler reads the input lines until it reaches the `END-PROFILER` keyword (or `END` or `.`).

This section covers the following topics:

- Profiler Utility Syntax
- Profiler Utility Keywords

Profiler Utility Syntax

The symbols used in the syntax diagrams shown in this section are explained in *System Command Syntax* in the *System Commands* documentation.

You enter a Profiler utility command using either of the following syntax formats:

```
keyword[=value][,keyword[=value]]...
```

Or:

```
keyword
[value]
...
```

Notes:

1. If a value is associated with a keyword but no equal sign is found, the Profiler expects the value in a separate input line without any other keyword (second syntax format).
2. The first syntax format expects input in delimiter mode (`IM=D`).
3. The second syntax format can be used if the Profiler is to be executed with the Natural `STACK` profile parameter or if the data is entered in forms mode (`IM=F`).

The following rules apply:

- Empty lines and lines starting with an asterisk (`*`) are ignored.
- All characters in a line from `/*` to `*/` or to the end of the line are ignored.
- Some keywords have no associated value.
- Blanks can be added before or after the keyword or value.
- Multiple keywords in a line are separated by commas (applies to the first syntax format only).
- A value can be enclosed in apostrophes (`'value'`).
- A value must not contain a comma.
- Keywords and values can be specified in upper or lower case.
- The maximum input line length is 78 characters.

The Profiler utility can be executed multiple times in one Natural session. For example, it is first executed with the `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)

Notes:

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`.

To use Profiler utility programs

- Logon to the library `SYSPRFLR` or copy the programs to the library `SYSTEM`, to the appropriate steplib library, or to the required library.

  If `PRFFCT` is used, the application programming interface `USR8210N` has to be copied as well (see the following section).

  If `PRFFCT` is used in a client/server environment, copy `PRFFCT` to the client library and `USR8210N` to the server library.

  Note: `PRFFCT` expects as input the value `START`, `PAUSE` or `STATE` to perform the corresponding function.

To start the data collection

- Execute the following program:

  ```
  PRFSTART
  ```

  Or:

  ```
  PRFFCT
  START
  ```

To pause the data collection

- Execute the following program:

  ```
  PRFPAUSE
  ```

  Or:

  ```
  PRFFCT
  PAUSE
  ```

To retrieve the current state of the data collection

- Execute the following program:

  ```
  PRFSTATE
  ```

  Or:

  ```
  PRFFCT
  STATE
  ```

Using the Application Programming Interface

The data collection can be started and paused from the profiled Natural application by calling the application programming interface (API) `USR8210N`. The API can also be used to get the current state of the monitoring process. The API is delivered in the `SYSEXT` library. For more information, see *SYSEXT Utility - Natural Application Programming Interfaces*.

To use the API

- Copy the subprogram `USR8210N` to the library `SYSTEM`, to the appropriate steplib library, or to the required library.

  Note: `USR8210N` expects as the first parameter the value `START`, `PAUSE` or `STATE` to perform the corresponding function. The parameter values can be specified in uppercase or lowercase. On return, `P-RETURN` contains the return code and `P-MESSAGE` the success or error message.

To start the data collection

- Use the interface with the `CALLNAT` statement:

```
CALLNAT 'USR8210N' 'START' P-RETURN P-MESSAGE /* Start Profiler
```

To pause the data collection

- Use the interface with the `CALLNAT` statement:

```
CALLNAT 'USR8210N' 'PAUSE' P-RETURN P-MESSAGE /* Pause Profiler
```

To retrieve the current state of the data collection

- Use the interface with the `CALLNAT` statement:

```
CALLNAT 'USR8210N' 'STATE' P-RETURN P-MESSAGE /* Get Profiler state
```

  The state is coded in the field `P-RETURN`:

P-RETURN	Description
0	Natural Profiler data collection is started.
1	Natural Profiler data collection is paused.

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.

Resource File Type	Evaluation	Description
	Line summary	Table of executed Natural source lines The table shows how many events occurred during line execution and the CPU and elapsed time spent executing the line.
	Statistics	Statistics of profiling, the consolidation and the NPRC resource file
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

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}]
 [LINE={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.

Keyword for READ	Value	Description
		NPRC: Write the Natural Profiler program summary. NCVF: Write the program coverage table.
	`OFF`	Do not evaluate programs.
`LINE`		This function is only available for NPRC resources. Specifies whether the Natural Profiler evaluates executed source lines. See also ***Line Summary***.
	`ON`	Write the Natural Profiler line summary.
	`OFF`	Do not evaluate executed source lines.
`STATISTICS`		Specifies whether the Natural Profiler writes statistics. See also ***Profiler Statistics***. **Note:** Statistics data for Natural code coverage is not collected on UNIX and Windows.
	`ON`	Write statistics.
	`OFF`	Do not write statistics.
`PRINT`		Specifies whether the result is written to standard output.
	`ON`	Write to standard output.
	`OFF`	Do not write to standard output.
`EXPORT`		Specifies whether the evaluated data is written to the Work File 7.
	`ON`	Write to Work File 7.
	`OFF`	Do not write to Work File 7.
`FORMAT`		Specifies the format in which the exported data is written to Work File 7.
	`TEXT`	Write the data in free text format.
	`COMMA`	Write the data in CSV format with a comma (`,`) separator.
	`SEMICOLON`	Write the data in CSV format with a semicolon (`;`) separator.

Example of READ

The following example reads the Natural Profiler resource `Test.nprf` in the library `PRFDATA` and writes the event trace, program trace and the Profiler statistics to standard output and to Work File 7 in text format.

```
FUNCTION=READ               /* Read Profiler Data
  RESOURCE-NAME='Test'    /* Resource name
  RESOURCE-LIB=PRFDATA    /* Resource library
  RESOURCE-TYPE=NPRF      /* Use resource type NPRF
  EVENT=ON                /* Print event trace
  PROGRAM=ON              /* Print program trace
  STATISTICS=ON           /* Print statistics
  PRINT=ON                /* Write to standard output
  EXPORT=ON               /* Write to Work File 7
  FORMAT=TEXT             /* Export in text format
```

Event Trace

If `EVENT=ON` is specified for an NPRF resource file, the Profiler event trace is generated.

The event trace shows the data of each Natural event which occurred while the application executed. The trace can be referenced if detailed information of an event is required. For example, if a Natural error occurred during application execution, the event trace shows the corresponding error number and message.

If the event trace is written to standard output (`PRINT=ON`) or exported in text format (`EXPORT=ON`, `FORMAT=TEXT`), it is similar to the event trace written by the Profiler monitor session while the application was profiled (see ***Event Trace***. If the data is exported in CSV (comma-separated values) format, it contains all data fields provided by the Profiler (see ***Data Collected***).

Example of an Event Trace

The following example shows an extract of an event trace:

```
Natural Profiler Event Trace
----------------------------

     Count Time              CPU-Time (ms) Ev Lev Library  Program  Line CC-Lib   CC-Name  Statement  Local-Data
         0 10:20:58.219911        63.318   MP 003 SYSPRFD  PRBINIT  8350                   Call       Monitor pause requested
       102 10:20:58.277586        76.106   PL 000                   0000                              Execute SYSEDMD/MENU
       103 10:20:58.277591        76.139   PS 001 SYSEDMD  MENU     0000                   PgmStart   00010/02430 Type: P
       103 10:20:58.277594        76.151   NS 001 SYSEDMD  MENU     0250                   Compute    Assign/Compute/Move
       103 10:20:58.277596        76.155   NS 001 SYSEDMD  MENU     0270                   Fetch      Fetch
       104 10:20:58.277598        76.169   DB 001 SYSEDMD  MENU     0270                   Fetch      00010/02430 S1
...
```

Explanations:

- The **Count** column shows the number of the event.
- 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 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 microseconds.

Example of a Program Summary

The following example shows the 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.

Line Summary

If `LINE=ON` is specified for an NPRC resource file, the Profiler line summary is generated.

The line summary shows for each source line in a Natural object, the number of Natural events that occurred (hit count), the CPU and elapsed time (in milliseconds and percent) spent by the line. The percentage of times is calculated in relation to the total times of the application.

The line summary does not count Monitor Pause events and events at Level 0.

If the data is exported in CSV (comma-separated values) format, the count of each event type is listed. Additionally, the Adabas times (absolute and percentage values) are displayed. The exported time values are indicated in microseconds.

Example of a Line Summary

The following example shows the extract of a line summary:

```
Natural Profiler Line Summary
--------------------------------
Library  Program  Line CC-Lib   CC-Name  Hit-Count       CPU (ms)  CPU %  Elapsed (ms)  Ela %
PRFTEST  XINT     0000                           1          0.016   0.46         0.003   0.01
PRFTEST  XINT     0140                           1          0.005   0.14         0.001   0.00
PRFTEST  XINT     0150                           1          0.006   0.17         0.002   0.01
PRFTEST  XINT     0160                           1          0.004   0.11         0.001   0.00
PRFTEST  XINT     0170                          23          0.128   3.75         0.029   0.18
PRFTEST  XINT     0180                          10          0.049   1.43         0.012   0.07
PRFTEST  XINT     0190                          10          0.054   1.58         0.010   0.06
...
Total                                          371          3.408 100.00        15.992 100.00
```

Explanations:

- Line `0170` in the Natural object `XINT` consumed `0.128` ms of the CPU time and `0.029` ms of the elapsed time. This corresponds to `3.75` percent of the total CPU time and `0.18` percent of the total elapsed time. `23` events (hit count) were executed in the line.
- At the end of the line 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.	
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.	

Column	Description
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`.	
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	

Column	Description
	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.

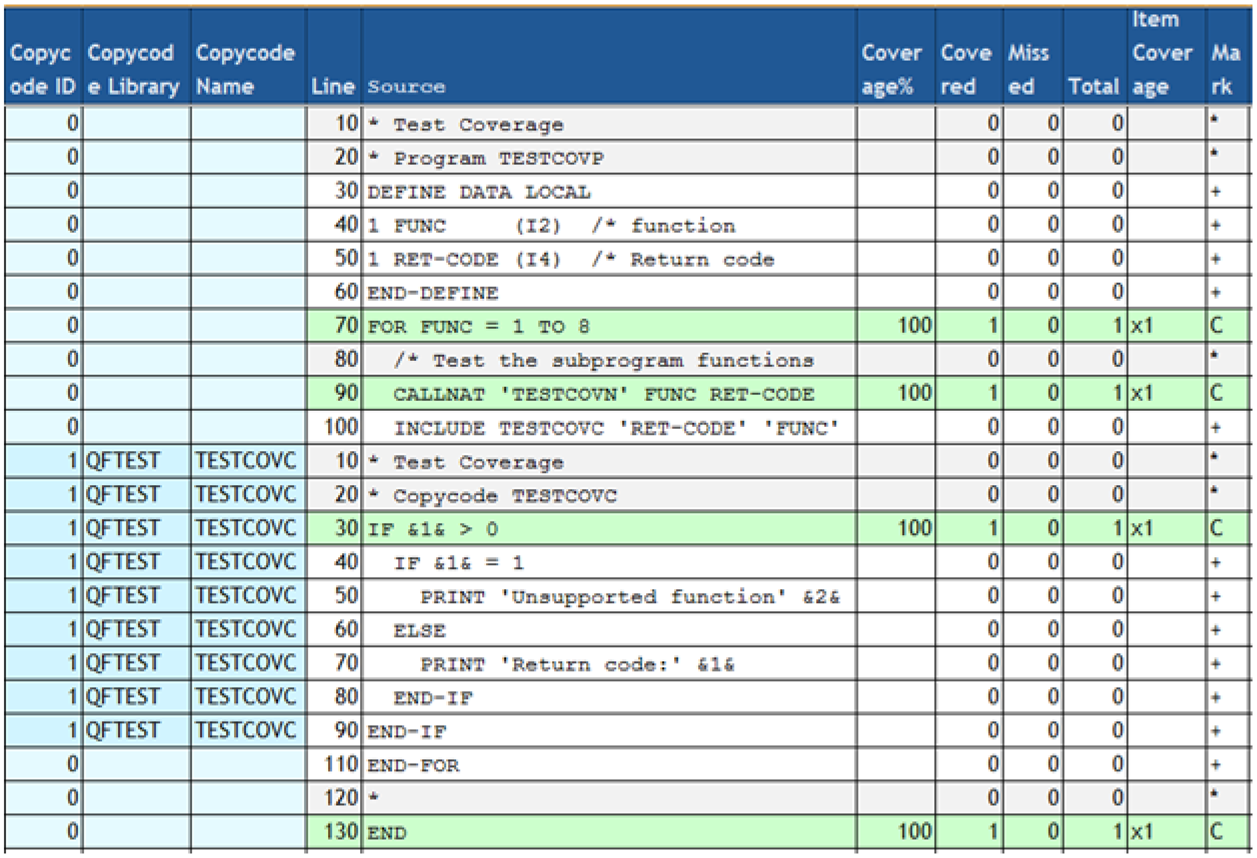
6. Open the delivered template `TESTCOV.XLSX` with Microsoft Excel. The template is contained in the **RES** (Resources) subdirectory of the Natural `SYSPRFLR` system library.
7. For each worksheet, copy the format from the template to your Microsoft Excel:
   - Click on the upper left corner of the 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.

Note: 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***.

Note: 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
  LINE=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***).

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.

Note: The event trace can also be listed in NaturalONE.

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.

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.

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***).

Notes:

1. By default (if `TRACE` is not specified), Trace Level 2 (warnings) is used.
2. The trace is activated as soon as the `TRACE` keyword is specified. It is therefore recommended to specify the `TRACE` keyword as soon as possible.
3. If you execute the Profiler utility multiple times in the job, you need to specify the `TRACE` keyword with each execution.

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

Note: The properties listed in the following section are the properties provided by the Profiler in all environments. The Profiler Statistics contains only the properties that are relevant for the current run. Therefore, not all of the properties listed in the section are displayed in every case.

General Info

Display environment and Natural Profiler related information.

Property	Unit	Description
Machine class		The name of the machine class on which the Natural application is running.
Environment		The environment in which the Natural application is running, such as `NaturalONE`, `Batch` or `RPC`.
Codepage		The code page used while the Natural application was monitored.
User		The ID of the user running the application (value of `*USER`).

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:

```
                                 BytesTrimmedLeft + BytesTrimmedRight -
CompressionRate := 100 ×                CompressionHeaderLength
                                        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.

Note: Natural code coverage statistics are collected on the mainframe only.

Property	Description
Coverage	Indicates whether Natural code coverage is performed (`ON`/`OFF`).
Missed statements recorded	Indicates whether missed statements are recorded (`ON`/`OFF`).
Coverage records	The total number of coverage records. These are program information and Natural statement records.
Program information records	The number of program information records written to the resource file. Each program information record contains program and copycode related information.
Coverage records/block	The average number of coverage records contained in one data block.
Bytes/coverage record	The average length in bytes of a compressed coverage record.
Programs covered	The number of covered programs.
Programs NOC-ed	The number of covered programs compiled with the Natural Optimizer Compiler.
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.

72 Natural Profiler MashApp

MashZone is a browser-based application from Software AG which is used to visualize data on a graphical, interactive dashboard, a so-called MashApp. The Natural Profiler MashApps evaluate the Profiler event data and depict it in MashZone.

Preparing to Use the MashApps

This section provides instructions for implementing the MashApps:

- Downloading the MashApps
- Unpacking the Zip File
- Editing the Overview.csv Resource File
- Activating the MashApps

Downloading the MashApps

The Natural Profiler MashApps and related data are supplied as a Natural component in a zip file.

To download the MashApps zip file

1. Log in to Software AG's Empower web site at *https://empower.softwareag.com/* (password required).
2. Go to **Products & Documentation** > **Download Components**.

   The **Download Components** section is displayed.
3. From the **Download Components** section, select **Natural Profiler MashApp**.
4. Download the `NaturalProfiler_MashApp.zip` file.

   In addition to the zip file, Empower also provides the Readme file `Readme_NaturalProfiler_MashApp.txt` which contains the latest update information.

Unpacking the Zip File

You have to unpack the MashApp zip file in the appropriate MashZone directory which depends on the MashZone version installed at your site.

To unpack the MashApp zip file

- Unpack the `NaturalProfiler_MashApp.zip` file in the appropriate user data directory of MashZone:

  - For MashZone Version 9.0 and above:

```
installation-directory\server\bin\work\work_mashzone_server-type\mashzone_data
```

where *server-type* indicates the type of the MashZone server: `s`, `m` or `i`. For example, `work_mashzone_m` for a medium type.

- For MashZone versions below Version 9.0:

```
installation-directory
```

where *installation-directory* is the MashZone installation directory.

After unpacking the zip file, the following subdirectories are available in the user data directory of MashZone:

Directory	Content
`importexport\Profiler_date`	MashApps for the Natural Profiler. *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.

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.

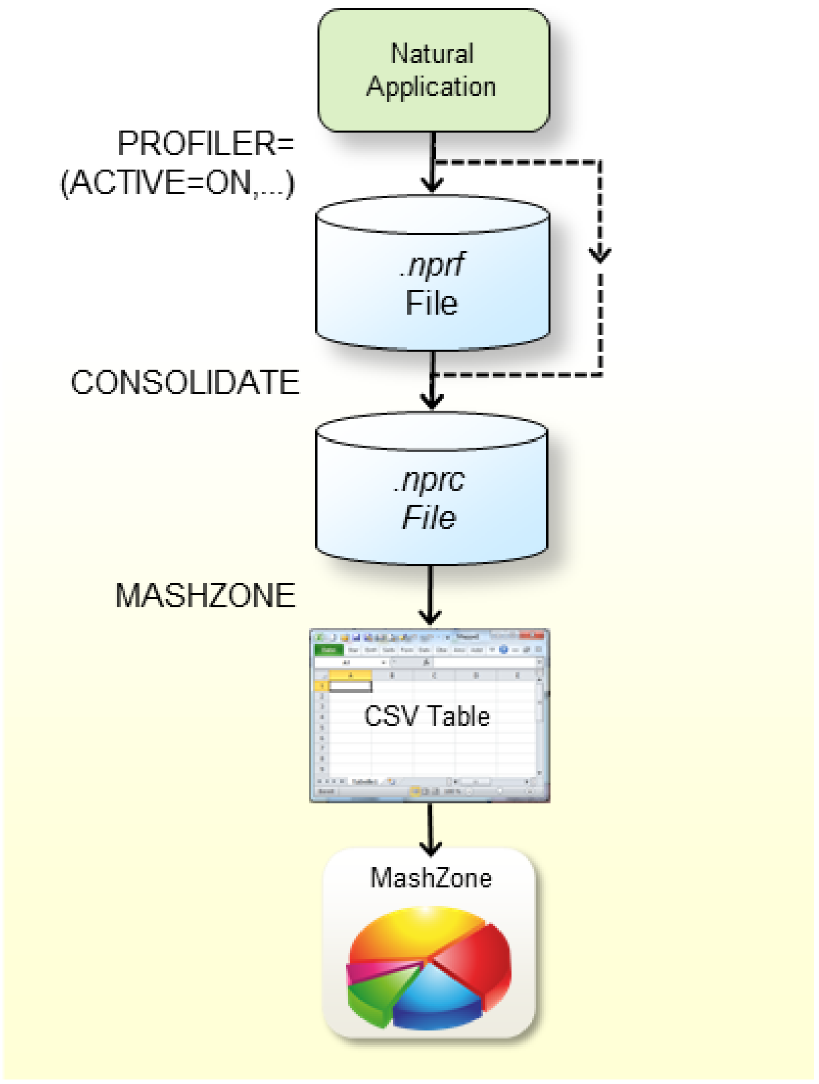
The MashApps in the `importexport\Profiler_`*`date`* directory are named as follows:

```
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:

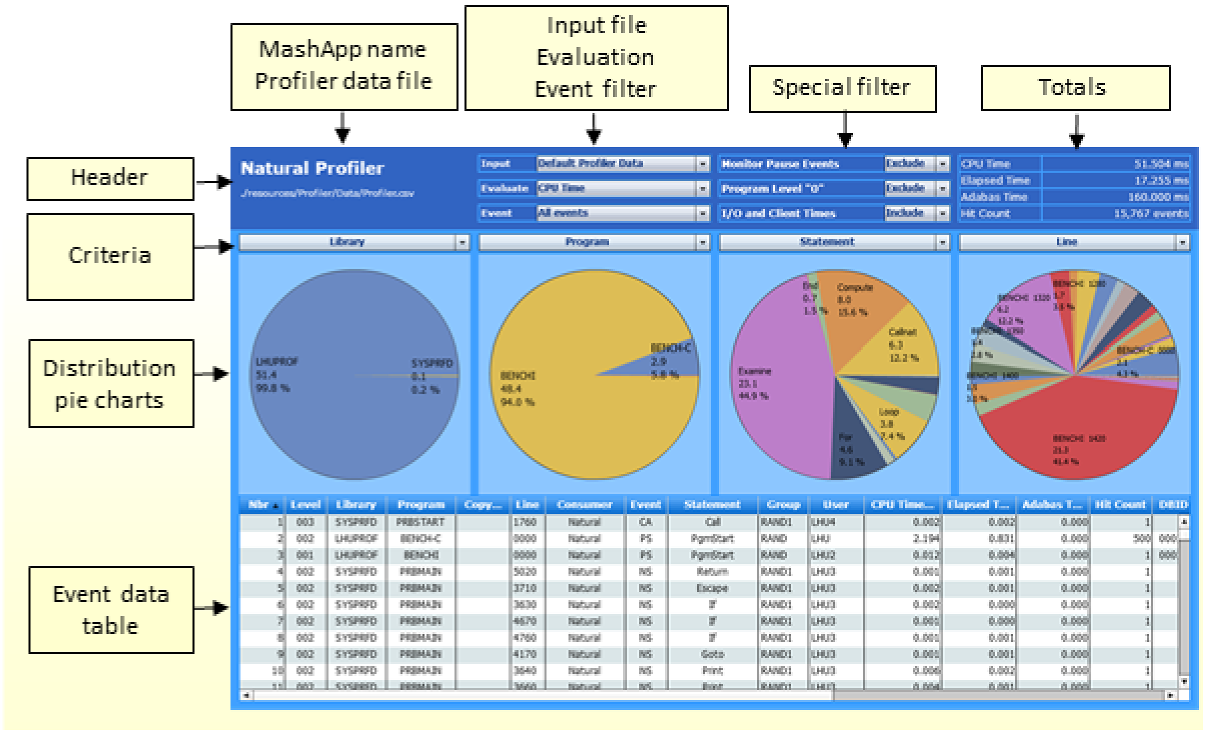
The **Compare page** compares the Profiler event data of two monitored applications.

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:



Natural Profiler
./resources/Profiler/Data/Profiler.csv

Input	Default Profiler Data	Monitor Pause Events	Exclude
Evaluate	CPU Time	Program Level "0"	Exclude
Event	All events	I/O and Client Times	Include

CPU Time	51.504 ms
Elapsed Time	17.255 ms
Adabas Time	160.000 ms
Hit Count	15,767 events

Nbr	Level	Library	Program	Copy...	Line	Consumer	Event	Statement	Group	User	CPU Time...	Elapsed T...	Adabas T...	Hit Count	DBID
1	003	SYSPRFD	PRBSTART		1760	Natural	CA	Call	RAND1	LHU4	0.002	0.002	0.000	1	
2	002	LHUPROF	BENCH-C		0000	Natural	PS	PgmStart	RAND	LHU	2.194	0.831	0.000	500	000
3	001	LHUPROF	BENCHI		0000	Natural	PS	PgmStart	RAND	LHU2	0.012	0.004	0.000	1	000
4	002	SYSPRFD	PRBMAIN		5020	Natural	NS	Return	RAND1	LHU3	0.001	0.001	0.000	1	
5	002	SYSPRFD	PRBMAIN		3710	Natural	NS	Escape	RAND1	LHU3	0.002	0.001	0.000	1	
6	002	SYSPRFD	PRBMAIN		3630	Natural	NS	IF	RAND1	LHU3	0.002	0.000	0.000	1	
7	002	SYSPRFD	PRBMAIN		4670	Natural	NS	IF	RAND1	LHU3	0.001	0.000	0.000	1	
8	002	SYSPRFD	PRBMAIN		4760	Natural	NS	IF	RAND1	LHU3	0.001	0.001	0.000	1	
9	002	SYSPRFD	PRBMAIN		4170	Natural	NS	Goto	RAND1	LHU3	0.001	0.001	0.000	1	
10	002	SYSPRFD	PRBMAIN		3640	Natural	NS	Print	RAND1	LHU3	0.006	0.002	0.000	1	
11	002	SYSPRFD	PRBMAIN		3660	Natural	NS	Print	RAND1	LHU3	0.004	0.001	0.000	1	

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

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

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.

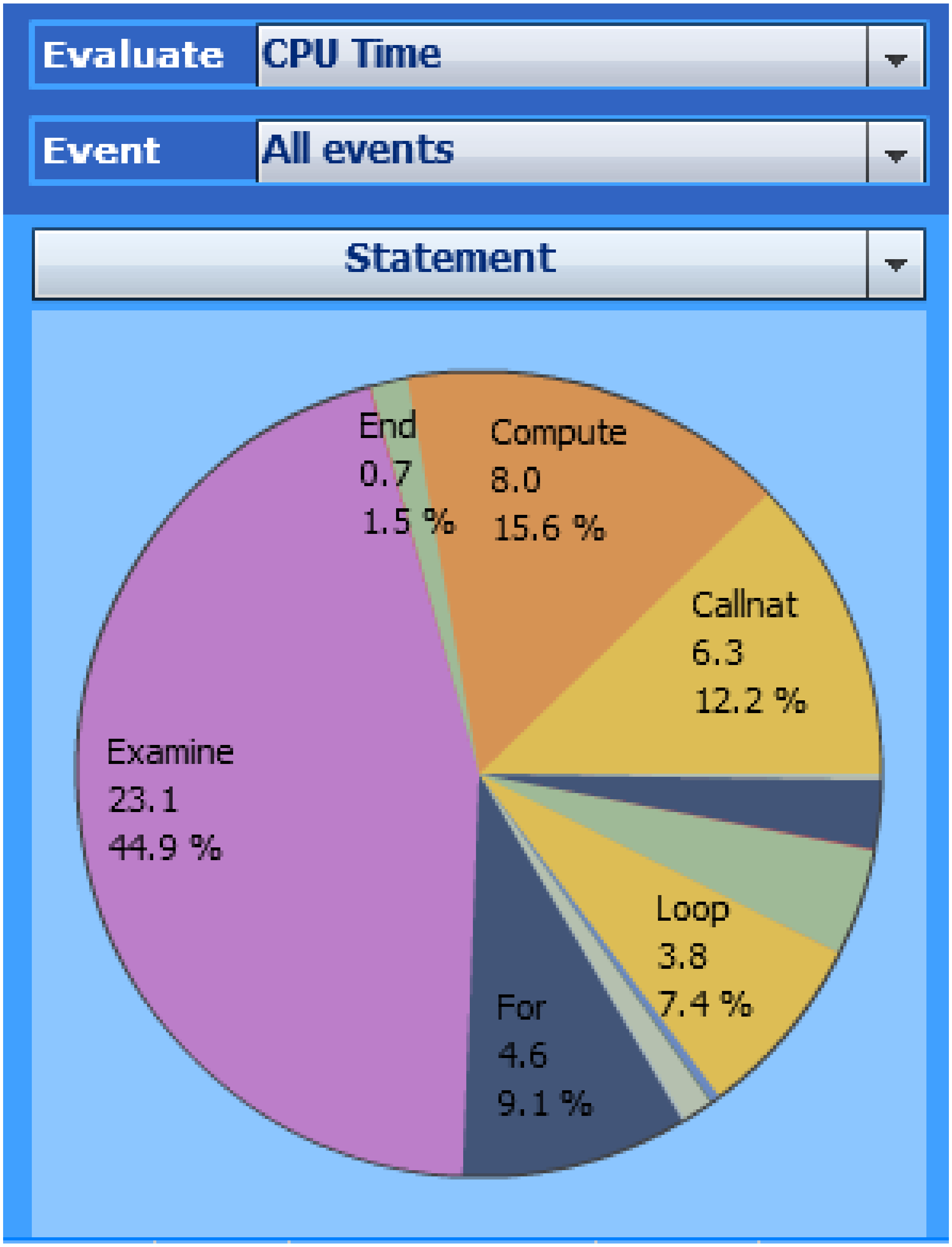
Type

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.

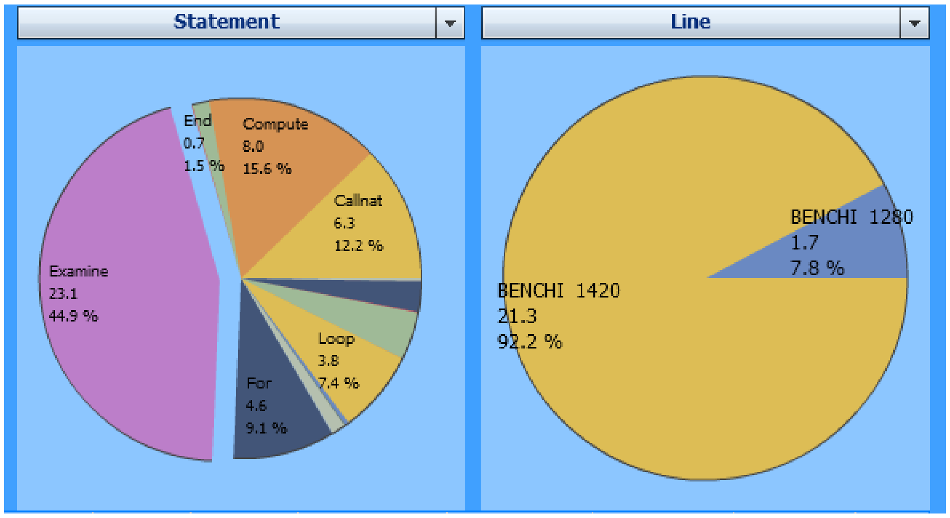
- A distribution pie chart shows the distribution for the criterion currently selected in the selection box directly above the pie chart. In the following example, **Statement** has been selected as the criterion for evaluating the CPU time:


Evaluate CPU Time

Event All events

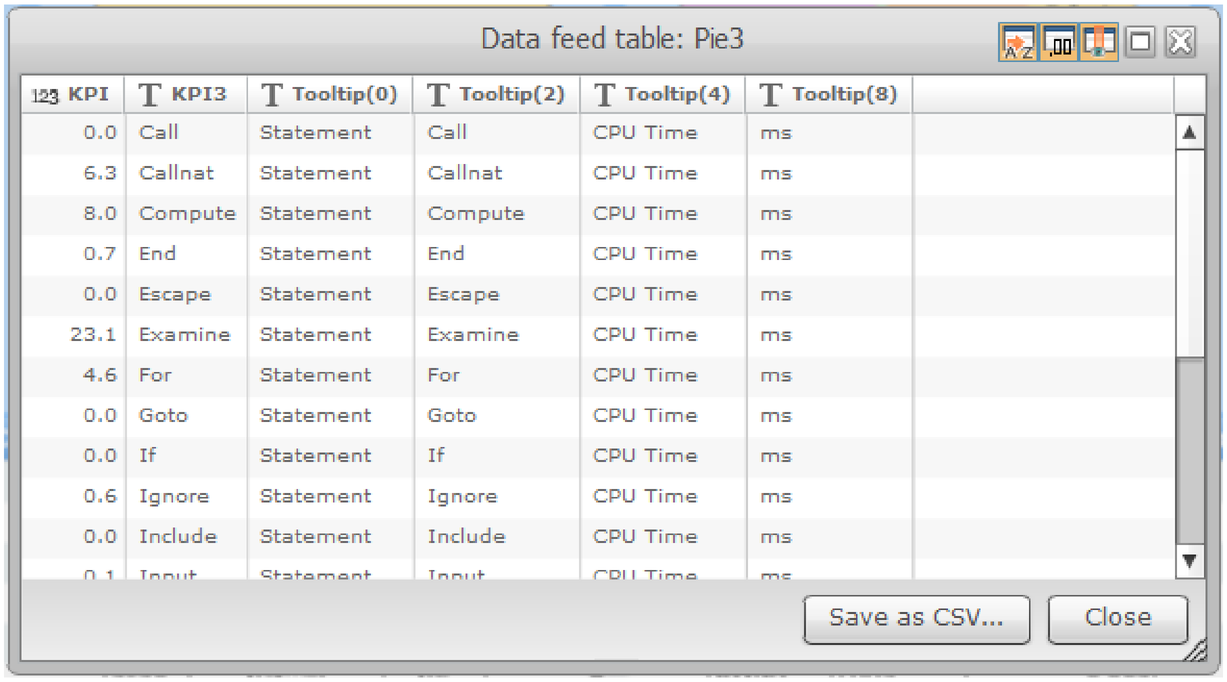
The pie chart shows the distribution of the CPU time for the used Natural statements. It indicates that the **Examine** statement consumed the most CPU time (`23.1` ms / `44.9` percent).

- 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

KPI	KPI3	Tooltip(0)	Tooltip(2)	Tooltip(4)	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

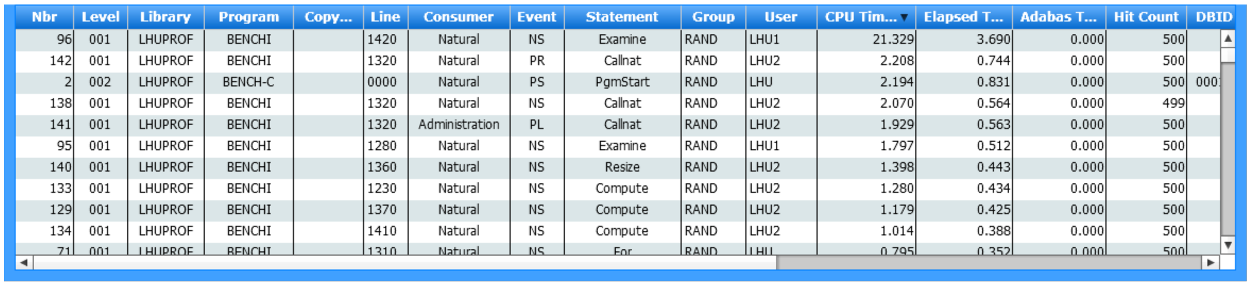
In the example above, the table lists the values of the left pie chart in the previous graphic. The **KPI** column lists the CPU time and the **KPI3** column the corresponding Natural statement.

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.

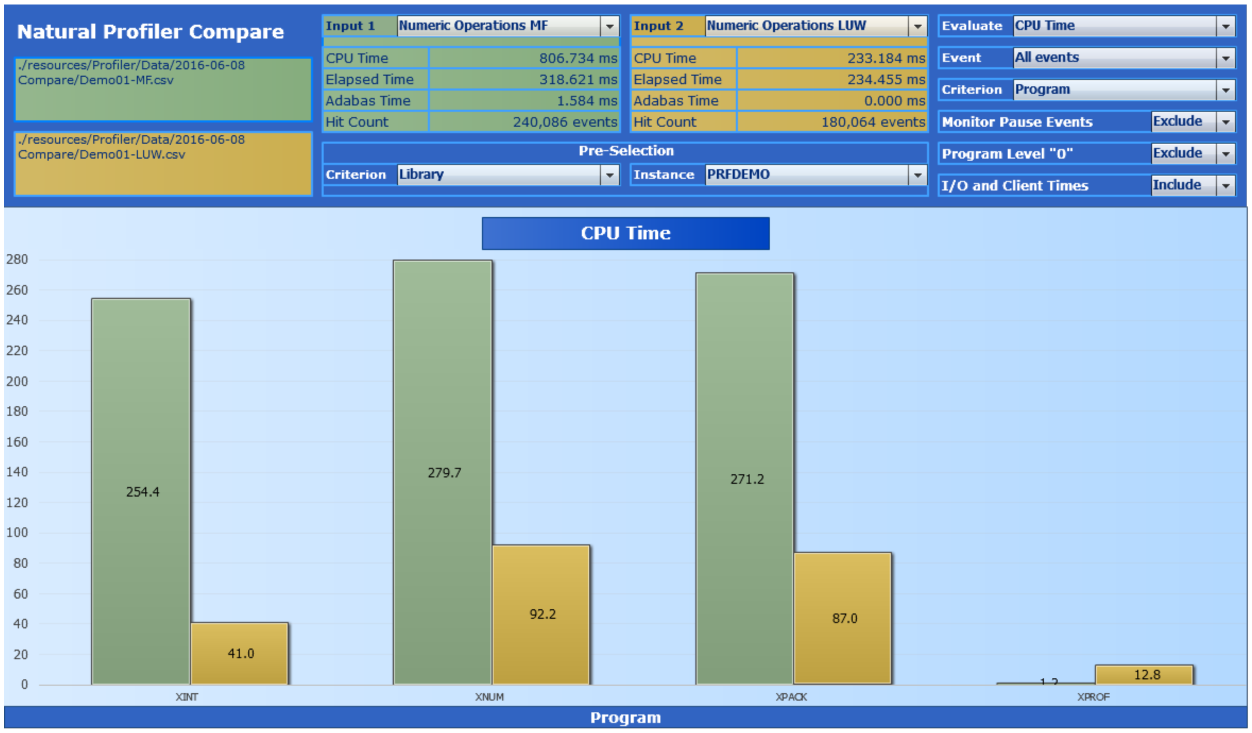
In the following example, the event data table is sorted by the CPU time (descending):



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:



Natural Profiler Compare

./resources/Profiler/Data/2016-06-08 Compare/Demo01-MF.csv

./resources/Profiler/Data/2016-06-08 Compare/Demo01-LUW.csv

Input 1	Numeric Operations MF
CPU Time	806.734 ms
Elapsed Time	318.621 ms
Adabas Time	1.584 ms
Hit Count	240,086 events

Input 2	Numeric Operations LUW
CPU Time	233.184 ms
Elapsed Time	234.455 ms
Adabas Time	0.000 ms
Hit Count	180,064 events

Pre-Selection

Criterion: Library
Instance: PRFDEMO

Evaluate	CPU Time
Event	All events
Criterion	Program
Monitor Pause Events	Exclude
Program Level "0"	Exclude
I/O and Client Times	Include

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 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.

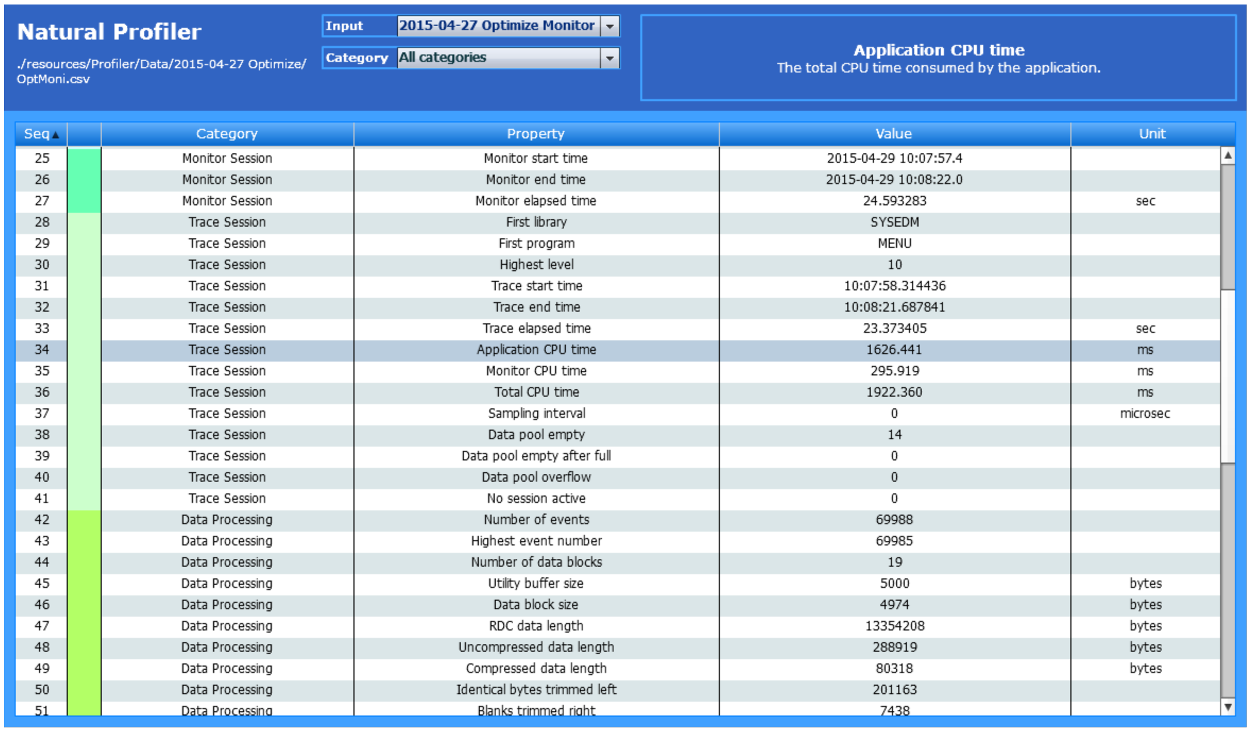
Compare Column Chart

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:



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 **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***.

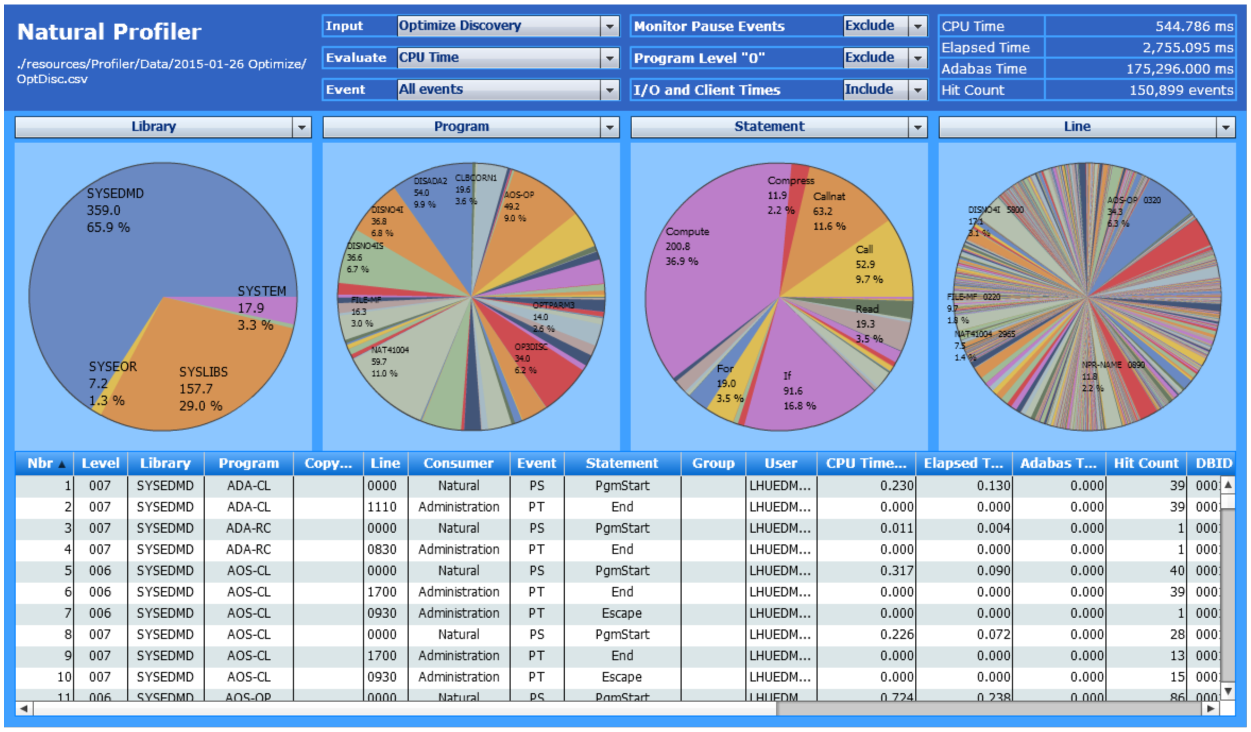
Use Cases

This section describes the following use cases:

Application Performance Analysis

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:



Natural Profiler

./resources/Profiler/Data/2015-01-26 Optimize/OptDisc.csv

Input	Optimize Discovery	Monitor Pause Events	Exclude
Evaluate	CPU Time	Program Level "0"	Exclude
Event	All events	I/O and Client Times	Include

CPU Time	544.786 ms
Elapsed Time	2,755.095 ms
Adabas Time	175,296.000 ms
Hit Count	150,899 events

Nbr ▲	Level	Library	Program	Copy...	Line	Consumer	Event	Statement	Group	User	CPU Time...	Elapsed T...	Adabas T...	Hit Count	DBID
1	007	SYSEDMD	ADA-CL		0000	Natural	PS	PgmStart		LHUEDM...	0.230	0.130	0.000	39	000
2	007	SYSEDMD	ADA-CL		1110	Administration	PT	End		LHUEDM...	0.000	0.000	0.000	39	000
3	007	SYSEDMD	ADA-RC		0000	Natural	PS	PgmStart		LHUEDM...	0.011	0.004	0.000	1	000
4	007	SYSEDMD	ADA-RC		0830	Administration	PT	End		LHUEDM...	0.000	0.000	0.000	1	000
5	006	SYSEDMD	AOS-CL		0000	Natural	PS	PgmStart		LHUEDM...	0.317	0.090	0.000	40	000
6	006	SYSEDMD	AOS-CL		1700	Administration	PT	End		LHUEDM...	0.000	0.000	0.000	39	000
7	006	SYSEDMD	AOS-CL		0930	Administration	PT	Escape		LHUEDM...	0.000	0.000	0.000	1	000
8	007	SYSEDMD	AOS-CL		0000	Natural	PS	PgmStart		LHUEDM...	0.226	0.072	0.000	28	000
9	007	SYSEDMD	AOS-CL		1700	Administration	PT	End		LHUEDM...	0.000	0.000	0.000	13	000
10	007	SYSEDMD	AOS-CL		0930	Administration	PT	Escape		LHUEDM...	0.000	0.000	0.000	15	000
11	006	SYSEDMD	AOS-OP		0000	Natural	PS	PgmStart		LHUEDM	0.724	0.238	0.000	86	000

You can immediately see which library, program, statement or line has consumed how much of the CPU time.

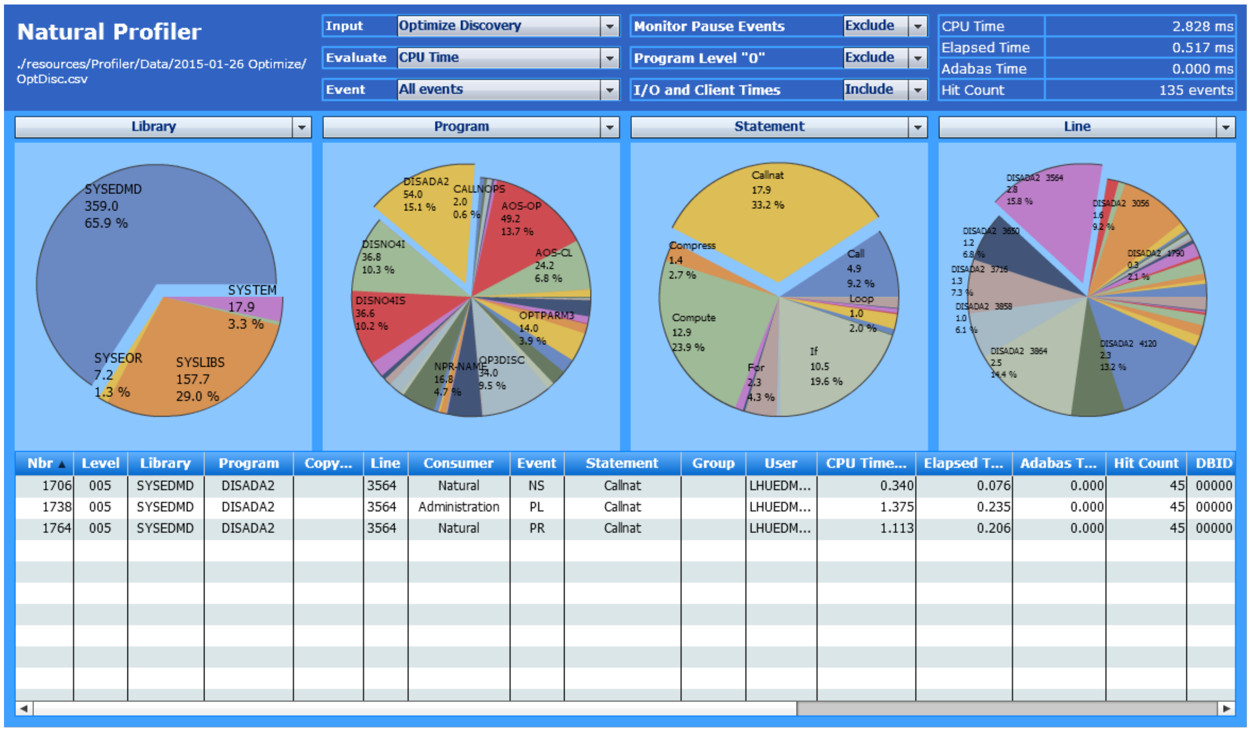
The example above uses the following selections:

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:



Natural Profiler

./resources/Profiler/Data/2015-01-26 Optimize/OptDisc.csv

Input	Optimize Discovery	Monitor Pause Events	Exclude	CPU Time	2.828 ms
Evaluate	CPU Time	Program Level "0"	Exclude	Elapsed Time	0.517 ms
Event	All events	I/O and Client Times	Include	Adabas Time	0.000 ms
				Hit Count	135 events

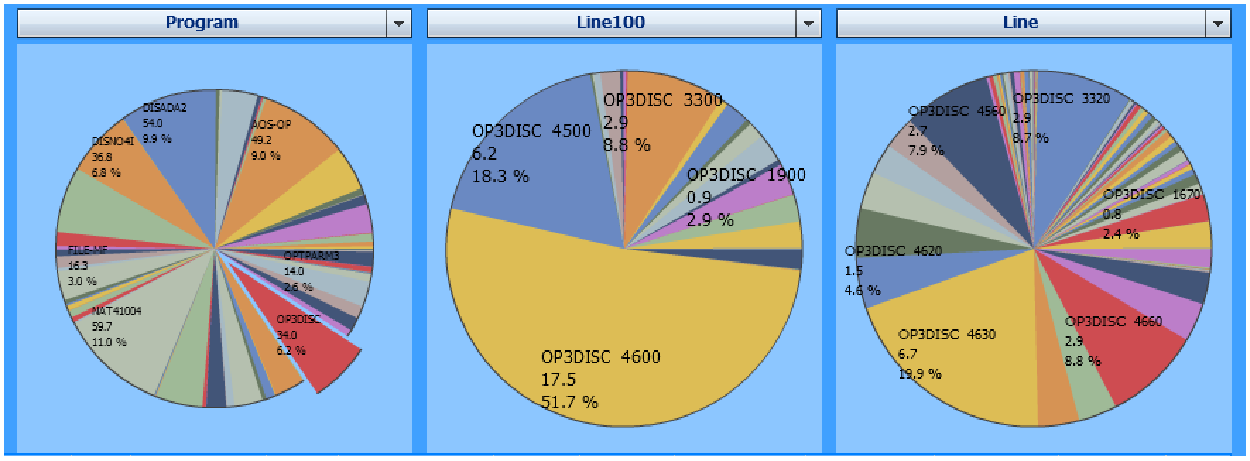
Nbr ▲	Level	Library	Program	Copy...	Line	Consumer	Event	Statement	Group	User	CPU Time...	Elapsed T...	Adabas T...	Hit Count	DBID
1706	005	SYSEDMD	DISADA2		3564	Natural	NS	Callnat		LHUEDM...	0.340	0.076	0.000	45	00000
1738	005	SYSEDMD	DISADA2		3564	Administration	PL	Callnat		LHUEDM...	1.375	0.235	0.000	45	00000
1764	005	SYSEDMD	DISADA2		3564	Natural	PR	Callnat		LHUEDM...	1.113	0.206	0.000	45	00000

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.

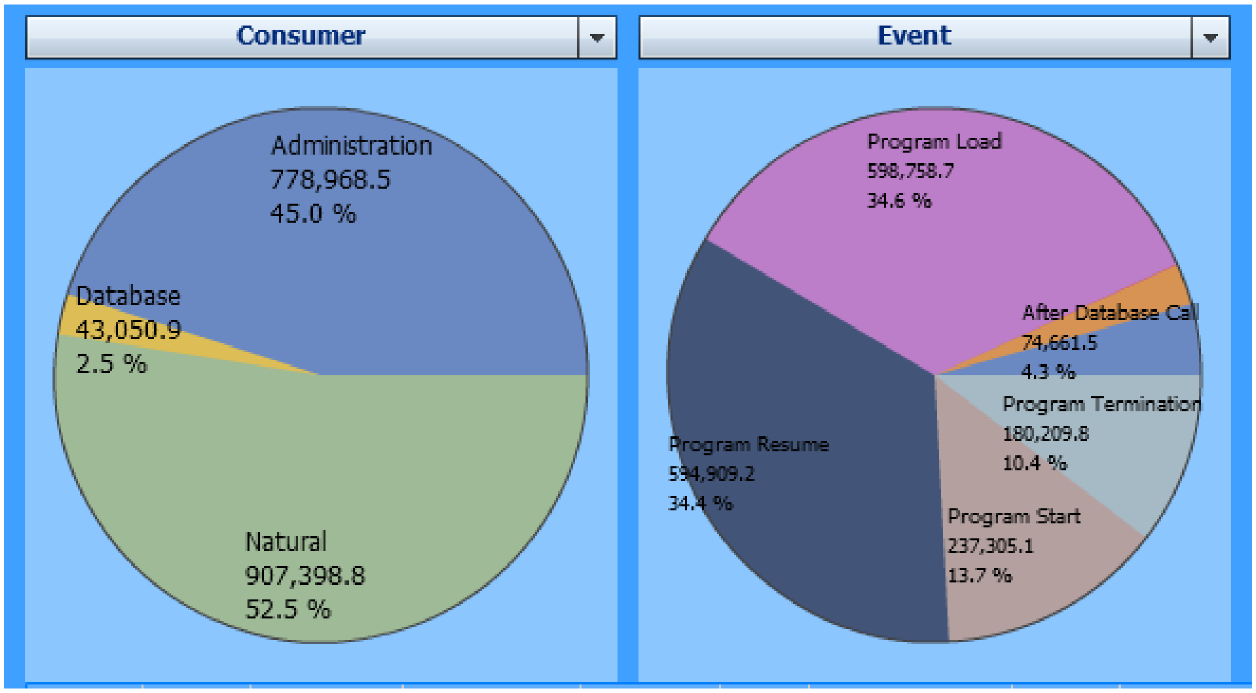
The **Line100** chart shows that more than half of the time was consumed by the statements in the lines ranging from 4600 through 4690. Additionally, considering the statements in the lines ranging from 4500 through 4590, this part of the program even consumes 70 percent of the entire execution time. 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.

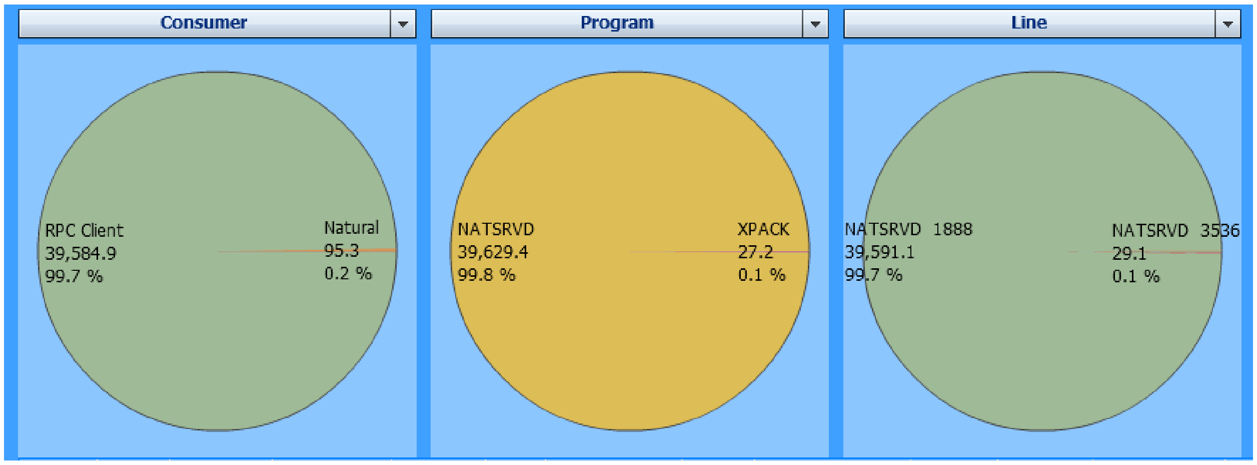
The example above uses the following selections:

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.

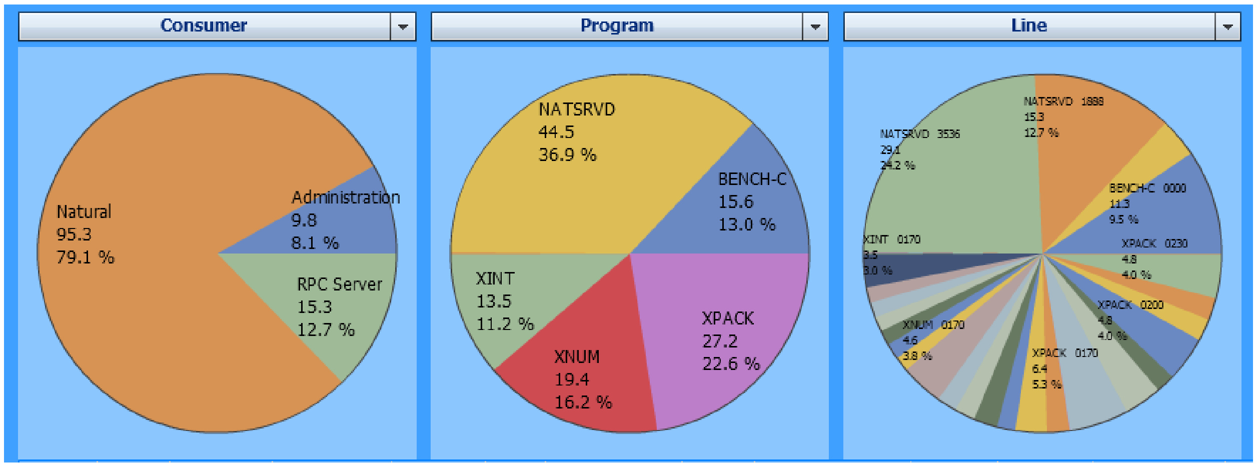
In the example below, the Natural RPC client consumes nearly all of the elapsed time:



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

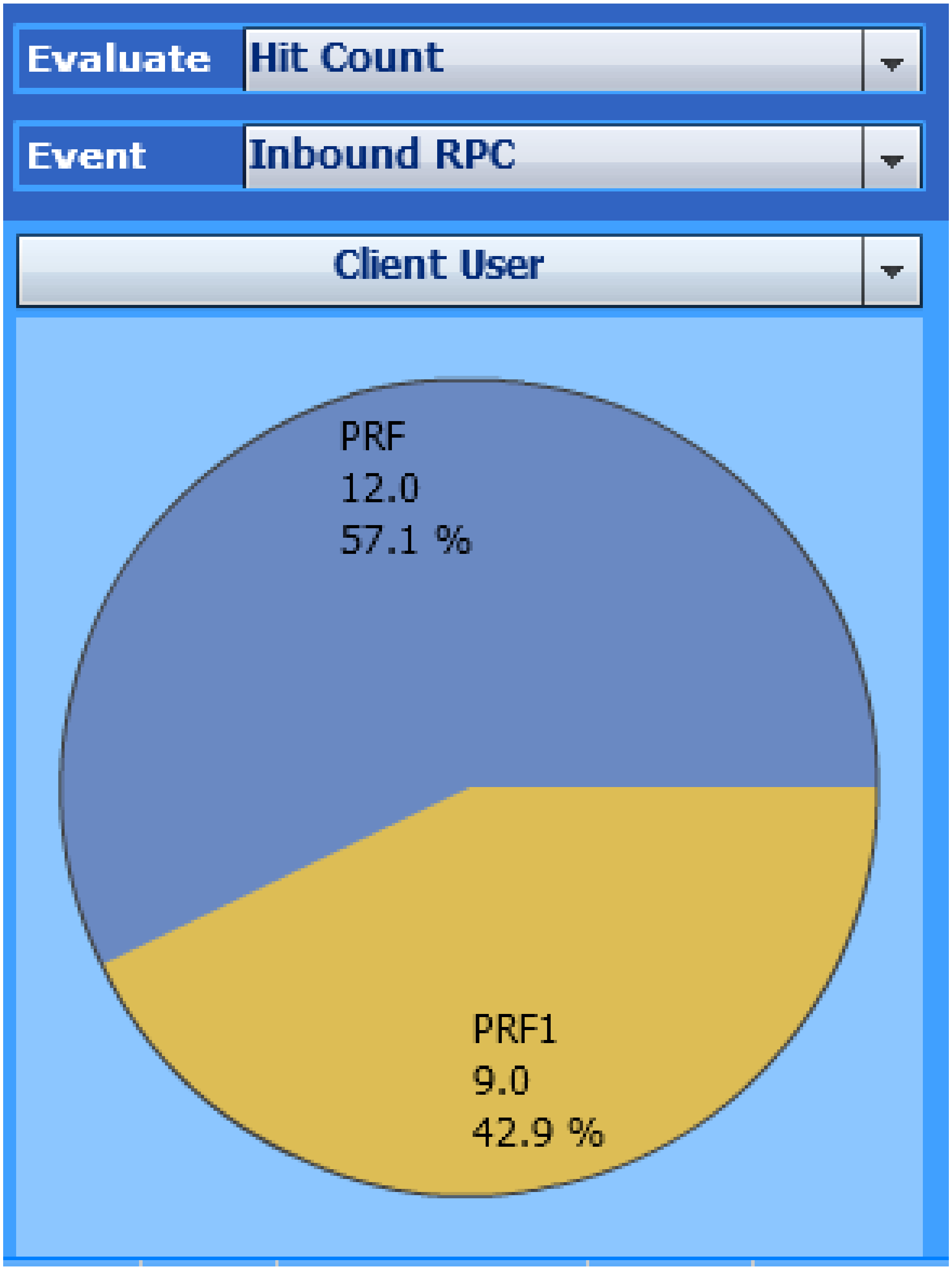
Criteria:	Consumer, Program, Line

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:



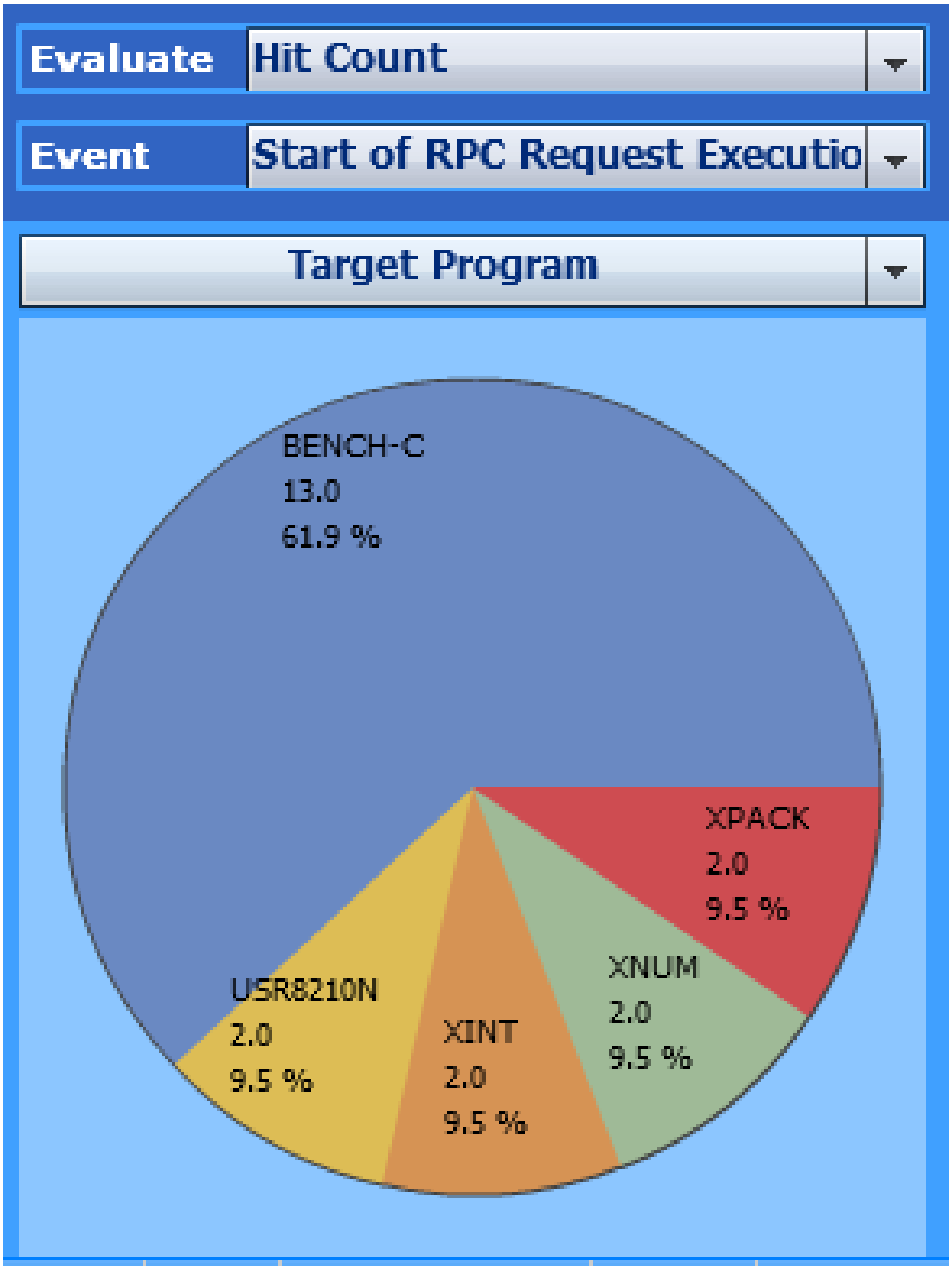
In the example above, the user `PRF` issued `12` Natural RPC requests.

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:


Evaluate Hit Count

Event Start of RPC Request Executio

Target Program

In the example above, `13` Natural RPC requests were issued for the server program `BENCH-C`.

The example uses the following selections:

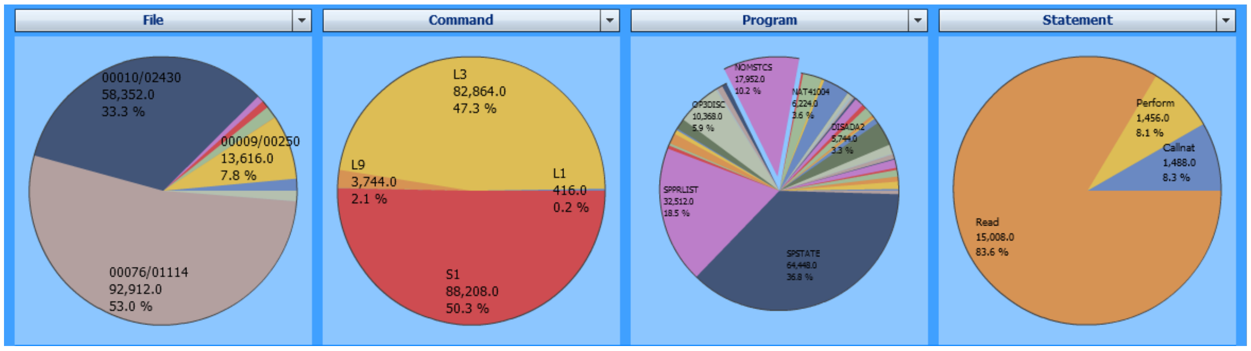
Evaluate:	Hit Count
Event:	Start of RPC Request Execution
Criterion:	Target Program

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`.

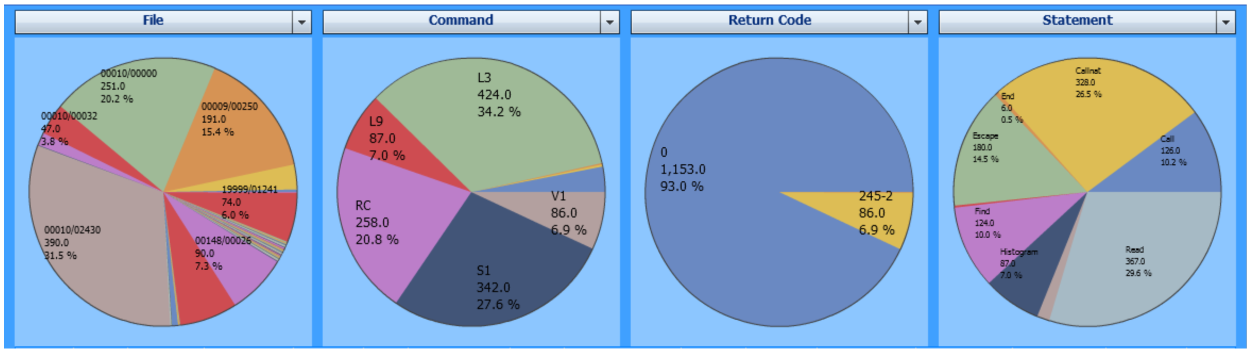
The example uses the following selections:

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.

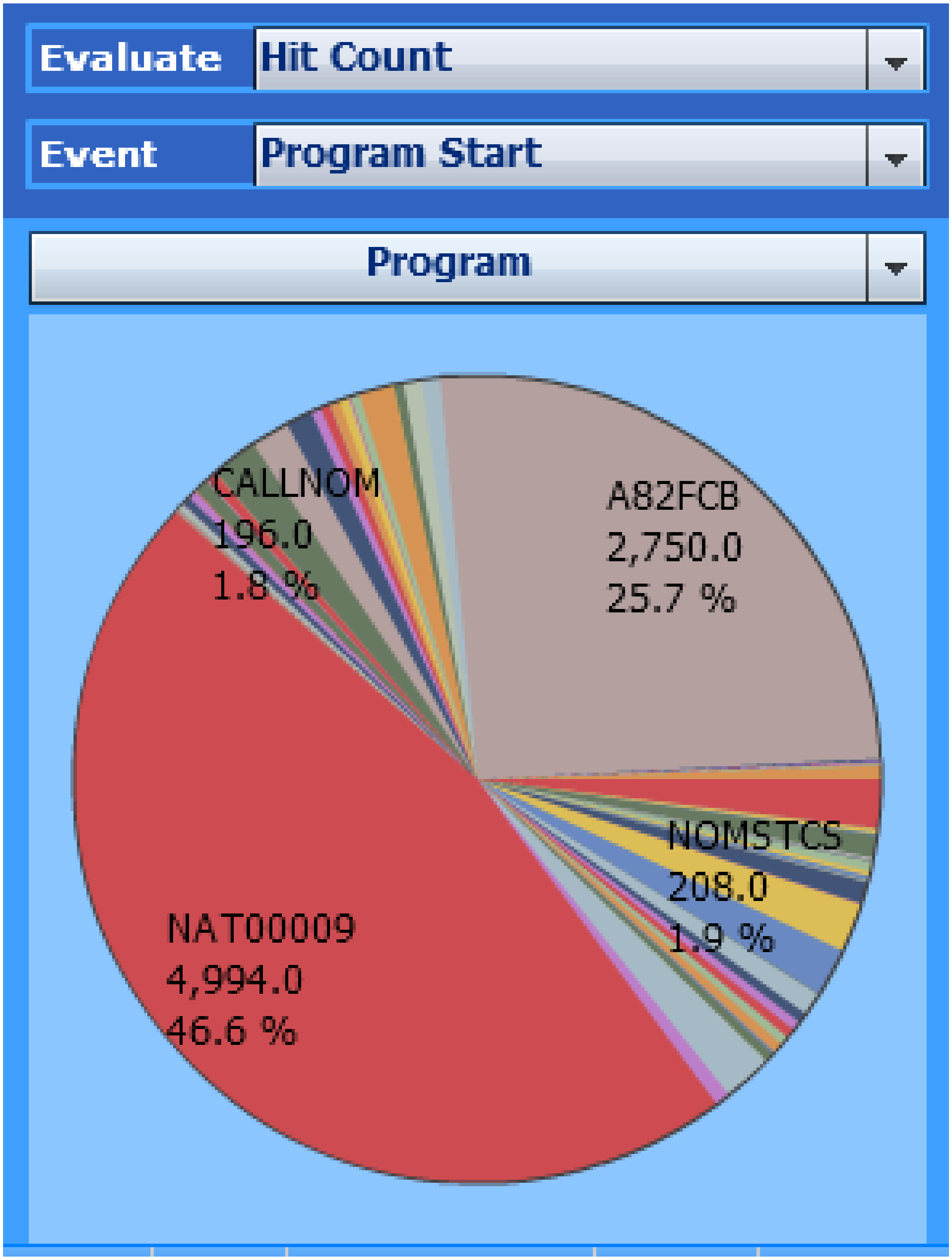
- How often were Natural objects started?
- How many statements were executed in the monitored programs?
- Which objects were called by a selected program and how often?
- How often was a selected object called?

- 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

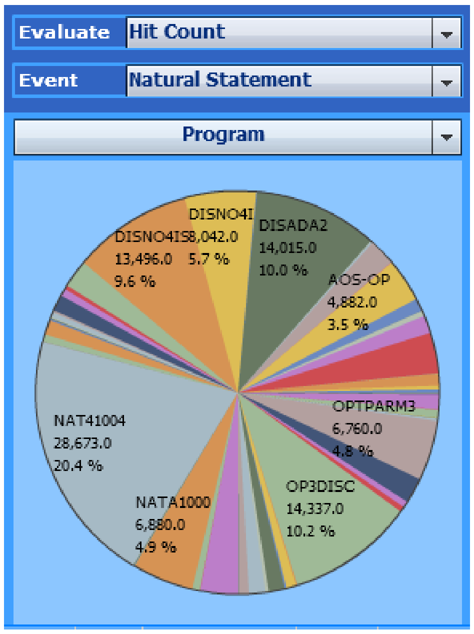

Evaluate
Hit Count
Event
Program Start
Program
A82FCB
2,750.0
25.7 %
CALLNOM
196.0
1.8 %
NOMSTCS
208.0
1.9 %
NAT00009
4,994.0
46.6 %

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

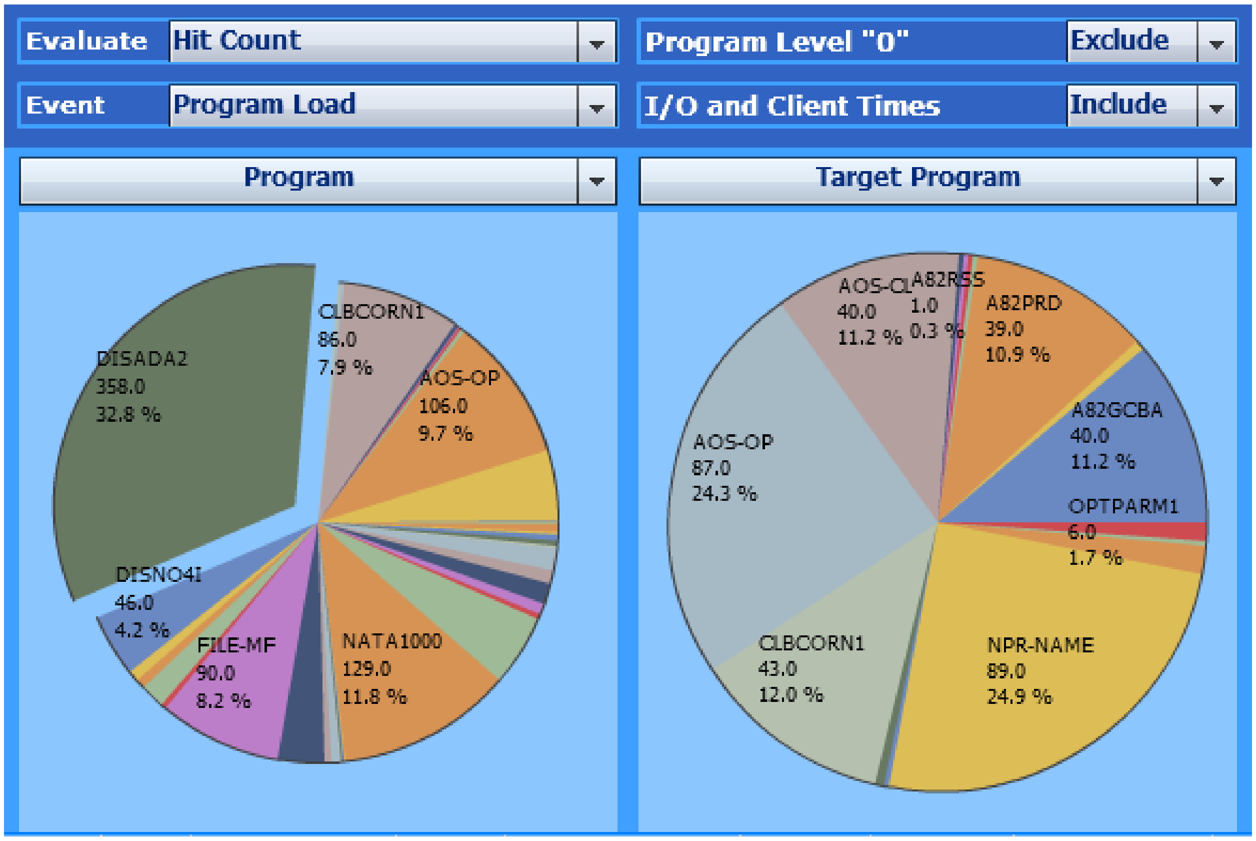

Evaluate: Hit Count
Event: Natural Statement
Program

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



Evaluate: Hit Count

Program Level "0": Exclude

Event: Program Load

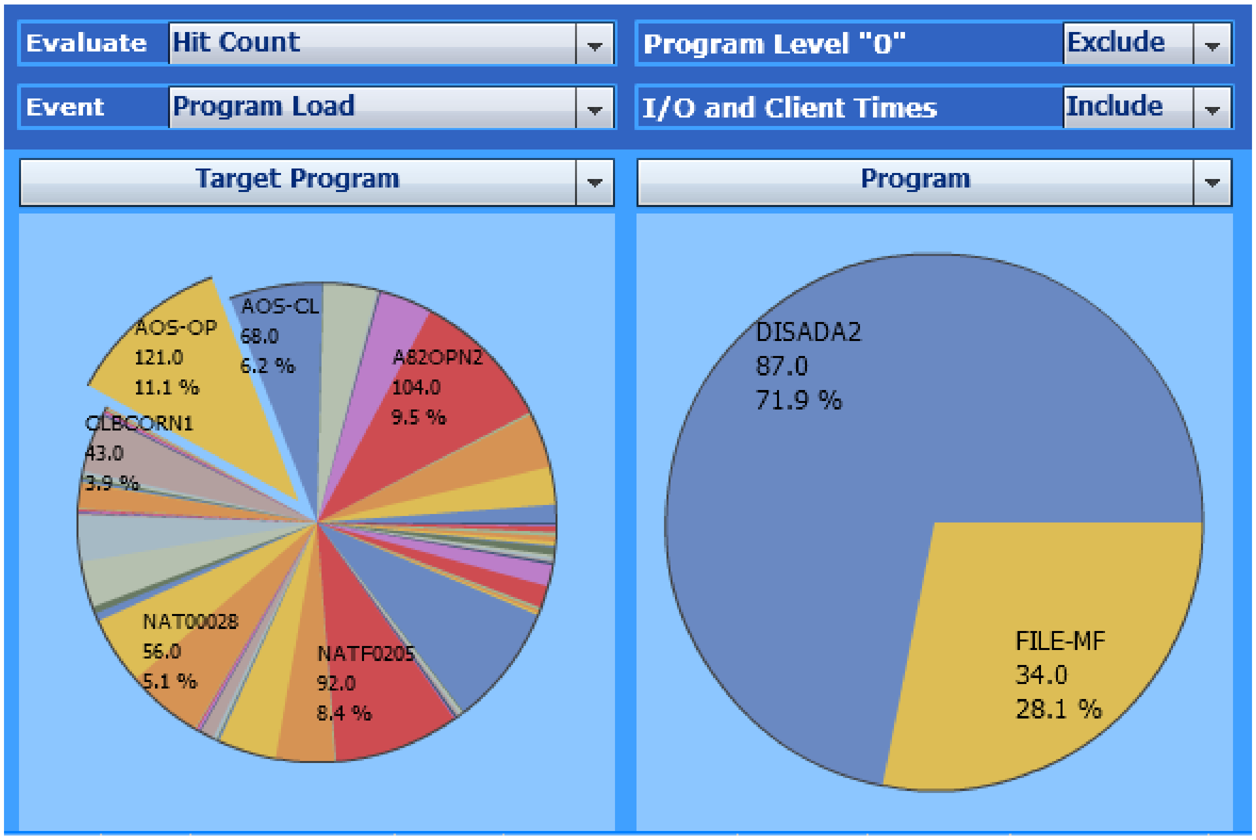
I/O and Client Times: Include

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

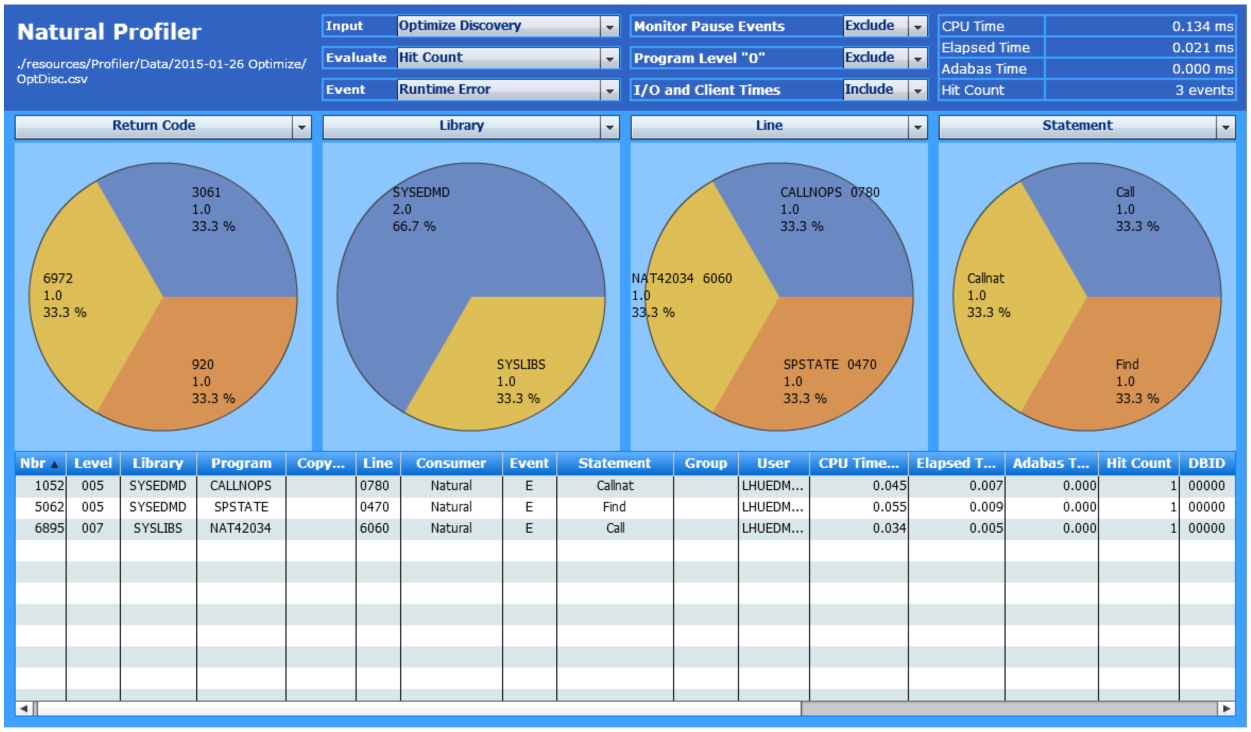


In the example above, the right pie chart shows the objects which called the program `AOS-OP`. `AOS-OP` was called `87` times by the program `DISADA2` and `34` times by the program `FILE-MF`.

How many runtime errors occurred?

Use the following selections to find out:

Evaluate:	Hit Count
Event:	Runtime Error
Criteria:	Return Code, Library, Line, Statement



Input	Optimize Discovery	Monitor Pause Events	Exclude
Evaluate	Hit Count	Program Level "0"	Exclude
Event	Runtime Error	I/O and Client Times	Include

CPU Time	0.134 ms
Elapsed Time	0.021 ms
Adabas Time	0.000 ms
Hit Count	3 events

Nbr ▲	Level	Library	Program	Copy...	Line	Consumer	Event	Statement	Group	User	CPU Time...	Elapsed T...	Adabas T...	Hit Count	DBID
1052	005	SYSEDMD	CALLNOPS		0780	Natural	E	Callnat		LHUEDM...	0.045	0.007	0.000	1	00000
5062	005	SYSEDMD	SPSTATE		0470	Natural	E	Find		LHUEDM...	0.055	0.009	0.000	1	00000
6895	007	SYSLIBS	NAT42034		6060	Natural	E	Call		LHUEDM...	0.034	0.005	0.000	1	00000

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.

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.

73 Invoking and Terminating SYSRPC

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.

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

To terminate the SYSRPC utility

- In the **Code** field of the **Client Maintenance** menu, enter a period (.).

  Or:

  Choose PF3 (Exit).

Invoking Online Help

To invoke the online help function

- Choose PF1 (Help).

74 Service Directory Maintenance

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

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.

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:

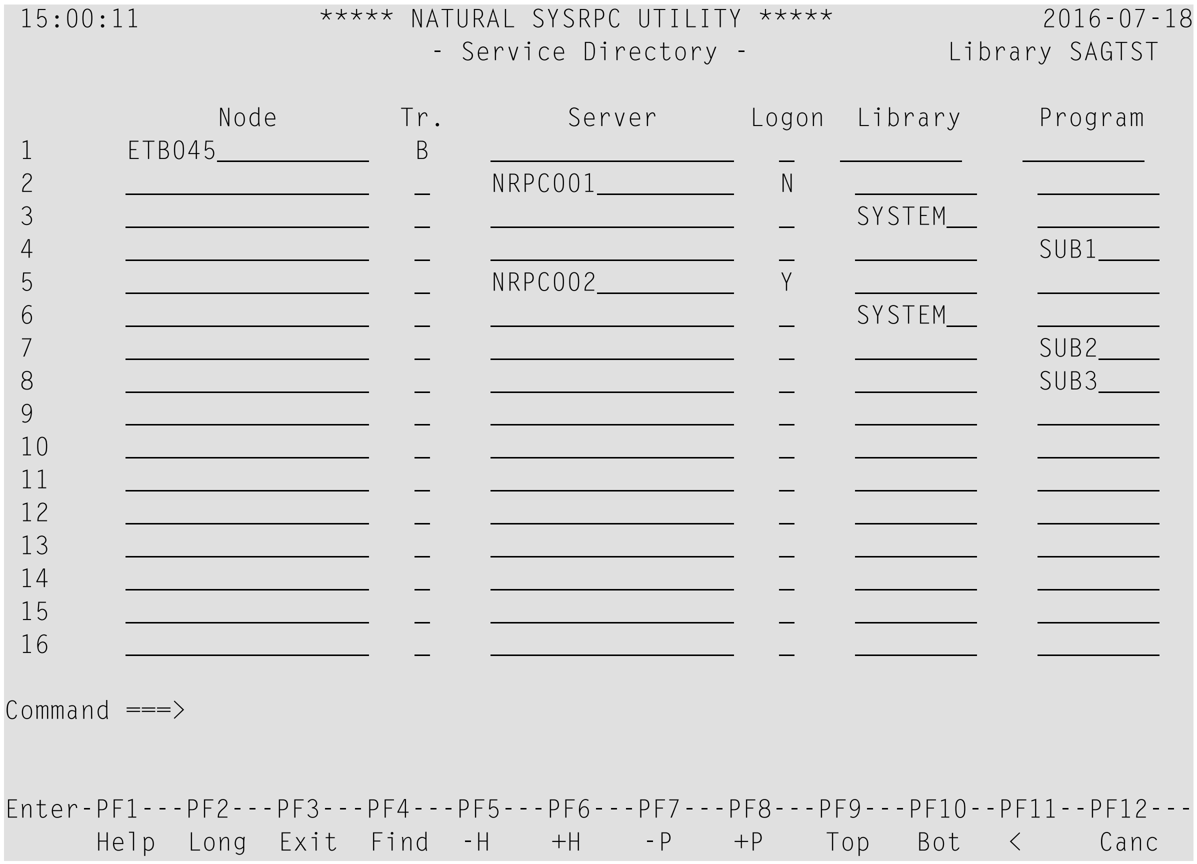
  ```
  Existing service definitions found
  ```

  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

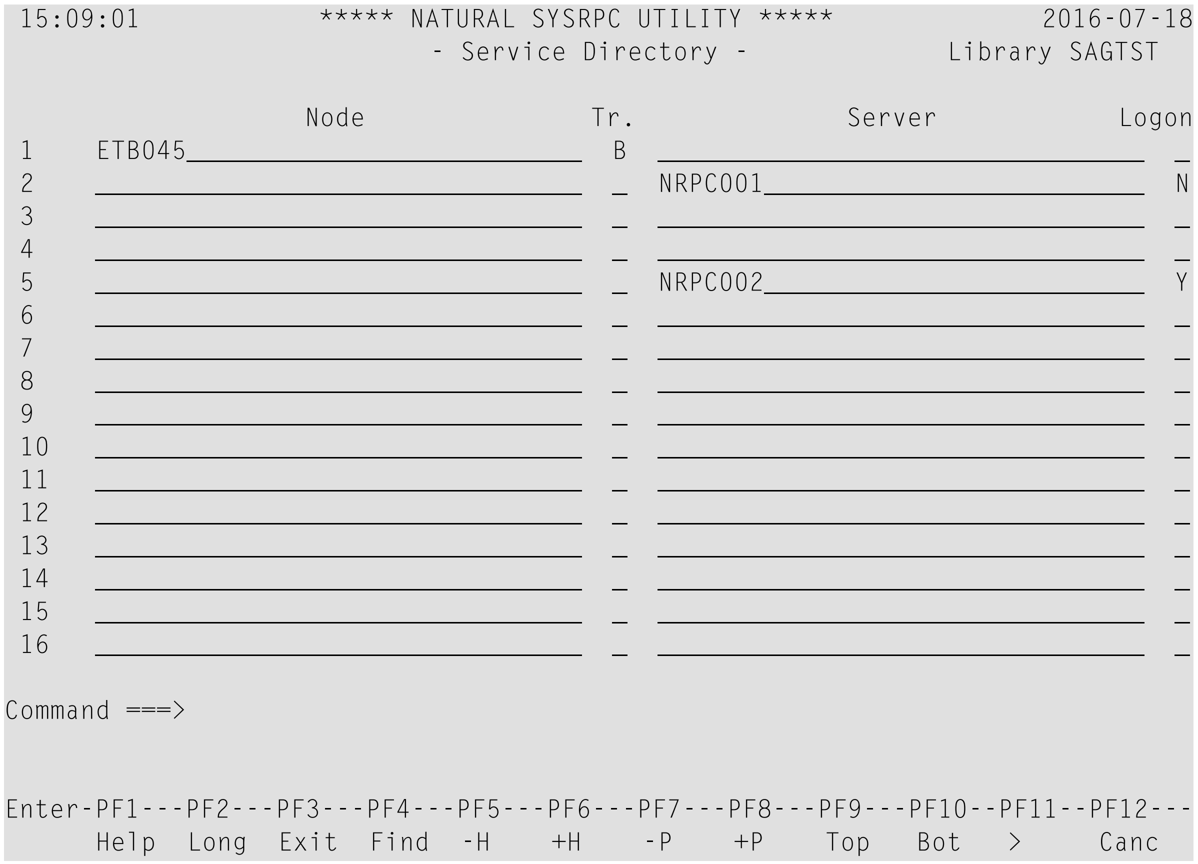


```
 15:00:11          ***** NATURAL SYSRPC UTILITY *****           2016-07-18
                      - Service Directory -           Library SAGTST

             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

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. The cursor is placed in the first new line below the column **Server** or **Library** depending on the item (node, server, library, program) contained in the line where you enter the command. Examples: If the line contains a node or server name, the cursor is placed below the **Server** column. If the line contains a library or program name, the cursor is placed below the **Library** column.
`MM`	Marks the block of lines to be moved.
`P`	Copies or moves the block of lines marked with `CC` or `MM` above the line in which the command was entered.

To copy or move a block of lines

1 At the beginning of the line where the block starts, overwrite the sequence number with either of the following line commands:

```
CC
```

to copy the block or

```
MM
```

to move the block.

2 At the beginning of the line where the block ends, overwrite the sequence number with either of the following line commands:

```
CC
```

to copy the block or

```
MM
```

to move the block.

3 Choose ENTER.

The line commands disappear, the sequence numbers are displayed again and the block of lines has been marked.

4 At the beginning of the line below or above which you want to place the marked block of lines, enter either of the following line commands:

```
A
```

to copy or move the block *below* the specified line or

```
P
```

to copy or move the block *above* the specified line.

Note that you can only execute `A` or `P` on lines where at least one field is filled.

5 Choose ENTER.

The block of lines is copied or moved below or above the specified line.

Direct Commands and PF Keys

The following direct commands can be entered in the Command line of the **Service Directory** screen and/or are provided as PF keys:

Direct Command	PF Key	Function
`EXPIRATION`		The remote directory data is loaded at runtime. The expiration time in seconds determines the period of validity of this data. If directory data is requested after the expiration time set, it will automatically be reloaded. If the expiration time is set to `0`, the remote directory data will not be reloaded. 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*. 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.

Direct Command	PF Key	Function
		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 `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.

75 Replacing Items in the Service Directory

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.

Note: The search operation is not case-sensitive.

76 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.

Caution: 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

77 Generating Single Interface Objects with Parameter Specification

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`.

Important: The function **Interface Object Generation** overwrites any data contained in the source work area. When you invoke the function, a corresponding message will warn you not to delete any existing data unintentionally: choose PF12 to cancel or choose ENTER to confirm the action and overwrite the contents of the source work area.

To generate a single interface object

1 Before you invoke the SYSRPC utility, 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. **Note:** 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

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.

Caution: 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:*)`.

Caution: 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				

78 Generating Multiple Interface Objects

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                                              2018-01-17 11:58:33

    SYSRPC - Interface Object Generation in Library SAGTEST


Generation Criteria:

    Object name or range: RPC*
             Compression: 1


Generation Results (sizes in bytes):

 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 (sizes in bytes):**
  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.

Caution: 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.

79 Calculating Size Requirements

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                                                2018-01-17 15:54:12

    SYSRPC - Calculation of Buffer Sizes for RPC Without Interface Objects


Calculation Criteria:

    Object name or range: RPC*
             Compression: 1


Calculation Results (sizes in bytes):

 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 (sizes in bytes):**
  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.

80 Parameter Maintenance

Applies to client sessions only.

The **Parameter Maintenance** function is used to dynamically (within a session) modify RPC-specific Natural profile parameters.

Caution: The parameter modifications are only retained as long as the user session is active; they are lost when the session is terminated.

Invoking Parameter Maintenance

To invoke the Parameter Maintenance function

1 In the **Code** field of the **Client Maintenance** menu, enter the following command:

```
PM
```

The **Client Parameter Maintenance** screen appears.

2 Modify the values of the input fields: see *Specifying Profile Parameters*.

3 Choose PF3 (Exit) to save any modifications and exit the **Client Parameter Maintenance** screen.

Or:

Choose PF12 (Canc) to exit without saving any parameter modifications.

The **Client Maintenance** menu appears.

Specifying 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.

81 Server Command Execution

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

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. 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**: Pings the selected RPC server. See also the `SYSRPC PING` direct command.	X	X
`TE`	Terminates the RPC server. See also ***Terminating an RPC Server.***.		X
`LN`	Lists sequence number(s) for a selected RPC server. You can select and terminate a server from this list. See also ***Terminating an 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**.

To ping an RPC server from the Server Command Execution screen

1 In the **Cmd** column next to the server(s) to be pinged, enter the following line command:

```
PI
```

as shown in the example of a **Server Command Execution** screen below:

```
 16:41:32           ***** NATURAL SYSRPC UTILITY *****              2016-07-18
                         - Server Command Execution -         Library SAGTRPC2

    Cmd Node                            Server                           Message
 1   __ ETB045
 2   PI                                   NRPC001
 3   PI                                   NRPC002





Command ===>


Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  ERR   Exit        -H    +H    -P    +P    TOP   BOT   >     Canc
```

2 Choose ENTER. The server(s) return the message:

```
Server version on operating system
```

where

`Server` denotes the type of server; `version` denotes the version `operating system` denotes on which operating system the server runs.

Example message:

```
Natural RPC Server 8.3.7.0 on WNT-x86
```

If pinging the server fails and an error occurs instead, you can choose PF2 (ERR) to display RPC-related Natural and EntireX Broker messages as described in *Using the RPCERR Program* (*Monitoring the Status of an RPC Session*, *Natural RPC (Remote Procedure Call)* documentation).

3 To display more of the message text which appears truncated in the standard view of the **Server Command Execution** screen (see also *Example of a Standard View*) proceed as follows:

Choose PF11.

Or:

In the Command line, enter the less than (<) sign.

This section covers the following topic:

- Using the SYSRPC PING Direct Command

Using the SYSRPC PING Direct Command

You can ping an RPC server by using the `SYSRPC PING` direct command in online or batch mode.

The following command syntax applies:

```
SYSRPC PING { server-name ON broker-name [[PORT] port-number][TRANSPORT {TCP|SSL|NET}] }
            { ALL                                                                    }
```

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)
`ALL`	n/a	Pings all RPC servers defined in the service directory of the current library.	

Terminating an RPC Server

The following commands are available to terminate an RPC server or EntireX Broker service from the standard or extended message view of the **Server Command Execution** screen:

`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 replica 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.
`LN`:	Lists the sequence number(s) for an RPC server registered on an EntireX Broker. Each server task is identified by start date/time, host name, application name and IP address, and can be selected and terminated from this list with the `TE` 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.

The following section provides instructions for terminating a single RPC server task or an EntireX Broker service from the standard view.

For alternative methods of terminating servers, see *Terminating a Natural RPC Server* described in the *Natural RPC (Remote Procedure Call)* documentation.

To terminate a single RPC server task

1 In the **Code** field of the **Client Maintenance** menu, enter the following command:

```
XC
```

Choose ENTER.

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
```

Choose ENTER.

(This is similar to entering the command `PI` as shown in the **example** of pinging a server.)

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.

3 If the **Logon** option is set in the service directory, logon data (user ID, password and library name) is sent to the server with the `TE` command, as is usual for remote `CALLNAT` execution. The **Security Token Data** window pops up and requests input of user ID and password if no Natural Security is installed on the client side and no logon data is set with the Application Programming Interface USR1071N for the current Natural session. See also USR1071N described in *Using Security*, *Using Natural RPC with Natural Security*, in the *Natural RPC (Remote Procedure Call)* documentation.

If `LOGONRQ=ON` (see also *Using Security* in the *Natural RPC (Remote Procedure Call)* documentation) has been set on the server side, logon data must be sent from the client with the `TE` command.

If Natural Security is installed on the server, the logon data transferred must enable a logon to the Natural system library SYSRPC.

To terminate a single server task with sequence number

1 In the **Code** field of the **Client Maintenance** menu, enter the following command:

```
XC
```

Choose ENTER.

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:

```
LN
```

Choose ENTER.

A list of server tasks appears with the following information:

Column	Description
SequenceNo	The number assigned to the server task in the sequence in which the task was registered on EntireX Broker.
Date	The date and time (UTC) when the server task was started.
Time	
Host Name	The node where the server is hosted.
Status	The status of the server. Possible status values are: `idle`, `busy` or `term` (for terminated).

The application name and IP address (if available) are indicated in the line below each numbered row.

The application name depends on the environment where the server runs. It can be the name of a Natural image (on UNIX), an LPAR (on mainframes) or a CICS started task, for example.

The IP address shows the IPv4 or IPv6 address of the node.

You can use PF keys **to navigate** in the screen and return to the list of all servers (PF4) or the **Client Maintenance** menu (PF3).

3 In the **Cmd** column, next to the sequence number of the server task to be terminated, enter the following line command:

```
TE
```

Choose ENTER.

The server returns the message `Successful response` and the status changes to `term` (terminated).

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.

82 Listing Servers Registered on EntireX Broker

You can obtain information on RPC servers registered on EntireX Broker by using the `SYSRPC SRVLIST` direct command.

`SYSRPC SRVLIST` sends a call to EntireX Broker requesting information on RPC servers registered on EntireX Broker with the attributes `SERVER-CLASS=RPC` and `SERVICE=CALLNAT`.

You can execute `SYSRPC SRVLIST` online (issued from a Natural command prompt) or in batch mode.

Note: When you execute this command online, a window prompts you to logon to the EntireX Broker specified in the command.

The following command syntax applies to `SYSRPC SRVLIST`:

```
SYSRPC SRVLIST server-name ON broker-name [[PORT] port-number][TRANSPORT
{TCP|SSL|NET}][USING{HEAD1MAP|object-name}]
```

The symbols used in the syntax diagram are explained in the section *Syntax Symbols* in the *Statements* documentation.

The syntax elements are explained in the following table:

Syntax Element	Format/Length	Description	
server-name	A32	Name of an RPC server or a range of names 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.

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.

83 Remote Directory Maintenance

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

Caution: If you create a new remote directory by entering code `C` (see the instructions below), the entries of an existing directory will be overwritten.

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.

Note: 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

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.

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.

84 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.

```
       ⎡ CSMASS /* Calculating Size Requirements                 ⎤
       ⎢ PING /* Pinging an RPC Server                           ⎥
       ⎢ SGMASS /* Generating Multiple Interface Objects         ⎥
SYSRPC ⎢ SM REPLACE /* Replacing Items in the Service Directory  ⎥
       ⎢                                                         ⎥
       ⎣ SRVLIST /* Listing Servers Registered on EntireX Broker ⎦
```