Natural for Mainframes

Using Natural

Version 4.2.6 for Mainframes

October 2009

Natural

This document applies to Natural Version 4.2.6 for Mainframes and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © Software AG 1979-2009. All rights reserved.

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

Table of Contents

1 Using Natural

This documentation introduces you to the usage of Natural in a mainframe environment.

The instructions and methods described in this documentation relate to the default standards as delivered with the original Natural software. They are not a comprehensive description of all features provided by Natural. For a full description of all options and functions, refer to the related topics mentioned in this documentation.

The layout of the example screens provided in the *Using Natural* documentation and the behavior of Natural described here can differ from your results. For example, a command or message line may appear in a different screen position, or the execution of a Natural command may be protected by security control. The default settings in your environment depend on the system parameters set by your Natural administrator.

2 Invoking and Terminating a Natural Session

A Natural session can be invoked for online or batch mode execution.

Related Topics:

- *Using Commands and Menu Function*
- *Natural Main Menu*

Starting a Natural Online Session

The way you invoke a Natural session depends on your local system environment. Ask your Natural administrator for details. If Natural Security is installed, the access to particular libraries as well as the use of Natural functions can be restricted.

Proceeding after Session Start

After starting Natural according to the procedures at your site, depending on the default settings in your environment, the Natural **Main Menu**, the NEXT command prompt or a user-defined startup menu appears.

When using the Natural **Main Menu**, you are guided through Natural development functions, environment settings, maintenance utilities and libraries that contain example programs. For information on the functions provided by the Natural **Main Menu** and its subordinate menus and instructions on invoking or closing the Natural **Main Menu**, refer to the section *Natural Main Menu*.

When using the NEXT command prompt, you continue with a Natural command or execute a Natural object of the type program. For detailed instructions, see *Using Natural Commands*.

Terminating a Natural Online Session

You can terminate a Natural session from the Natural **Main Menu**, a Natural command prompt or from a Natural object.

To terminate a Natural online session

- Use one of the following methods:
  - In the Natural **Main Menu**, choose PF3 (Exit), or select **Exit Natural Session** and choose ENTER.

- Or:
  In the command line (see *Command Line*), enter one of the following commands:

  ```
  .
  ```

  (a period)

  or

  ```
  FIN
  ```

  or

  ```
  %%
  ```

  and choose ENTER.
- Or:
  At the NEXT prompt (see *NEXT and MORE*), enter one of the following commands:

  ```
  FIN
  ```

  or

  ```
  %%
  ```

  or

  ```
  CLEAR
  ```

  and choose ENTER.
- Or:
  At the MORE prompt (see *NEXT and MORE*), enter one of the following commands:

  ```
  FIN
  ```

  or

  ```
  %%
  ```

  and choose ENTER.
- Or:
  From within a Natural object, execute a `TERMINATE` statement.

Related Topics:

- *Using Natural Commands*
- *Introducing Natural Objects*

Starting and Terminating a Natural Batch Session

The Natural batch interface provides the option to execute a Natural command or a Natural object of the type program in batch mode.

To invoke a Natural session in batch

- Proceed as described in *Starting a Natural Session* in the *Operations* documentation.

A Natural batch-mode session is terminated when one of the following is encountered during the session:

- A `FIN` command in the input dataset CMSYNIN.
- An end-of-input condition in the command input dataset CMSYNIN.
- A `TERMINATE` statement in a Natural object that is being executed.

Related Topic:

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

3 Using Commands and Menu Functions

You can perform an operation during a Natural session by using a Natural command or a menu function. When using a Natural command, you can directly perform an operation without navigating through different menus.

A Natural object of the type program can also be executed without using a Natural command as mentioned in ***Executing Programs***.

This section describes the different categories of commands provided with Natural, how to execute a command and how to use a menu function.

Related Topic:

- ***Natural Main Menu***

Categories of Natural Commands

This section describes the different categories of Natural commands:

- System Commands
- Terminal Commands
- Editor and Utility Commands

System Commands

Natural system commands perform functions you need in order to create, maintain or execute a Natural object. In addition, Natural system commands are used to monitor and administer your Natural environment.

Related Topics in the System Commands Documentation:

- *System Commands* (Overview)
- *System Commands Grouped by Functions*
- *System Command Syntax*

Terminal Commands

Natural terminal commands, for example, can be used for the following:

- Arrange the screen display and layout such as the positioning of the PF-key and message line and the assignment of colors.
- Obtain debug information on the current environment.
- Interrupt a current Natural operation.

You can invoke a terminal command while an application is executing. In addition to the Natural command prompts, terminal commands can be entered in any alphanumeric input field. A terminal command starts with a control character that can be specified by setting a Natural session parameter. The default control character is the percent (%) sign.

Related Topics:

- *Screen Design - Programming Guide*
- *Copying Data from a Screen - Programming Guide*
- *Terminal Commands Grouped by Function - Terminal Commands* documentation
- *Terminal Commands* (Overview) - *Terminal Commands* documentation
- ***Using Session Parameters***

Editor and Utility Commands

In addition to Natural system and Natural terminal commands, each Natural editor and Natural utility provides its own commands that only apply to this very environment. These commands are documented in the relevant section of the editor or utility documentation.

Using Natural Commands

You enter a Natural system command at any command prompt. Natural command prompts are:

- The command line in the Natural **Main Menu** or on the screen of a Natural utility or system command (see also the following **example**).
- An editor command prompt such as the greater than (>) sign of the program editor (see also the following **example**).
- The NEXT prompt and the MORE prompt.

Some utilities require that a system command is preceded by a special sign such as double forward slashes (//). For details, refer to the relevant section in the *Utilities* documentation.

Most of the system commands can be entered in conjunction with one ore more individual parameters and operands that further specify the operation to be performed. See also ***Example of a System Command***.

You enter a Natural terminal command at any command prompt or in any alphanumeric input field.

You enter a Natural editor or utility command at the command prompt or in the command line of the editor or utility.

The input of a Natural command is *not* case-sensitive. After you have entered a Natural command, you choose ENTER. ENTER confirms the action and executes the command or invokes an extra confirmation window where you explicitly acknowledge command execution.

This section covers the following topics:

- Command Line
- NEXT and MORE
- Example of a System Command

Command Line

The command line is located above the PF-key lines and looks as follows:

```
Command ===>


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

NEXT and MORE

The NEXT prompt appears if no Natural program (for example, the program that invokes the Natural **Main Menu**) has been started yet. The NEXT prompt indicates that Natural is awaiting your next command input.

The MORE prompt appears during the execution of a program and indicates that additional output is available. Choose ENTER to display the additional output. If you enter a command in response to the MORE prompt, the program that is being executed will be terminated and the command will be executed.

The NEXT or MORE prompt is usually located in the left upper or lower corner of the screen as shown in the example below:

```
NEXT                                                                LIB=TEST
```

Example of a System Command

The following instructions are an examples of how to execute a Natural system command in order to edit an object.

To invoke a Natural editor for a Natural object

1. At any command prompt, enter the system command `EDIT` and one or more operands, if required.

   For example:

   ```
   EDIT P PROGX
   ```

   where `P` is the type of object (program) and `PROGX` the name of the object to be edited.

2. Choose ENTER.

   The Natural program editor is invoked and the source code of PROGX is displayed in the editing area as shown in the example below:

```
>                                        > +  Program     PROGX    Lib TEST
All    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  0010 READ (3) EMPLOYEES BY NAME
  0020 DISPLAY NAME
  0030 END
  0040
  0050
  0060
  0070
  0080
  0090
  0100
  0110
  0120
  0130
  0140
  0150
  0160
  0170
  0180
  0190
  0200
  0210
  0220
  0230
  0240
  0250
  0260
  0270
```

```
0280
     ....+....1....+....2....+....3....+....4....+....5....+... S 3    L 1
```

See also the equivalent input when using the corresponding menu functions as described in *Example of a Menu Function*.

Related Topic:

- *Example of Command Syntax - System Commands* documentation

Using Natural Menus

Every Natural menu screen provides a list of functions. The way you invoke a function from a menu depends on the menu structure and the options provided.

Natural menus provide the following alternative methods that can be used to select and invoke a menu function:

- An individual input field for each function listed.
- The **Code** field, where you can enter the one- or two-character code that is assigned to each function.
- Function keys (PF keys) that correspond to a particular function in the menu.

  The PF-key lines (usually located at the bottom of the screen) indicate which function is assigned to which key. See also *Standard PF Keys*.

In addition to the functions, most of the Natural menus provide fields where you can specify further options and/or selection criteria. For example, the **Development Functions** menu contains the **Name** field in which you can enter the name of a Natural object. For explanations of these fields and their valid input values, refer to the relevant documentation or use the help function as described in the section *Natural Online Help*.

This section covers the following topics:

- Example of a Menu Function

Performing a Menu Function

This section provides instructions for performing and terminating menu functions.

Performing a Menu Function

- Place the cursor in the input field next to the menu function required and choose ENTER.

  Or:

  Place the cursor in the input field next to the menu function required and enter any character.

  Or:

  In the **Code** field, enter the one- or two-character code displayed before the function required.

  Or:

  If available, from the PF-key lines, choose the PF key that corresponds to the function required (see also ***Standard PF Keys***).

  Or:

  Double-click on the input field next to the function required.

If additional input fields are displayed in the menu, enter the information required. If you fail to do so, you will receive either a window from which you can select a valid input value or a corresponding Natural error message.

For explanations of possible field entries, invoke the help function by entering a question mark (?) in the required field.

Terminating a Menu Function

The following instructions describe alternative methods you can usually use to terminate a menu function with or without saving modifications made previously on another Natural screen.

To terminate a function without saving changes

- Enter a period (.) and choose ENTER.

  Or:

  Choose PF12.

To terminate a function and save changes

- Choose PF3.

Standard PF Keys

The following function keys (PF keys) are assigned to the following functions throughout most Natural menus:

PF Key	PF-Key Name	Explanation
PF1	Help	Invokes the online help function.
PF2	Menu	Invokes the Natural **Main Menu**.
PF3	Exit	Terminates a function.
PF12	Canc	Terminates a function and cancels the changes made previously.

Example of a Menu Function

The following instructions are an example of how to use a Natural menu in order to edit an object.

To invoke a Natural editor for a Natural object

1 On the **Development Functions** screen:

In the **Code** field, enter the one-letter code that corresponds to the function **Edit Object**.

In the **Type** field, enter the one-letter code that corresponds to the type of Natural object (in the example below: `P` for program).

In the **Name** field, enter the name of the Natural object (in the example below: `PROGX`).

```
13:33:16                    *****  NATURAL  *****                    2009-05-20
User SAG                  - Development Functions -          Library TEST
                                                               Mode Structured
                                                           Work area empty
                          Code  Function

                           C    Create Object
                           E    Edit Object
                           R    Rename Object
                           D    Delete Object
                           X    Execute Program
                           L    List Object(s)
                           S    List Subroutines Used
                           ?    Help
                           .    Exit
```

```
                   Code .. E    Type .. P
                                Name .. PROGX___________________________


Command ===>








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

2 Choose ENTER.

The Natural program editor is invoked and the source code of PROGX is displayed in the editing area as shown in ***Example of a System Command***.

4 Natural Online Help

Natural provides several types of help:

- General information on Natural components such as Natural statements, commands, variables, editors and utilities.
- Detailed information on Natural system error messages and user-defined messages.
- Specific information on Natural menus and input fields that appear in Natural utilities and editors.

General information on Natural components is contained in the Natural Help utility which is accessed with the system command `HELP`.

Detailed information on Natural messages is provided by the system commands `HELP` and `LASTMSG`.

Specific information on menus and input fields is provided by individual help functions.

Related Topics:

- *HELP - System Commands* documentation
- *LASTMSG - System Commands* documentation

This section describes the help topics provided and how to access them.

General Information - Natural Help Utility

The Natural Help utility provides general information on Natural statements, commands, variables, editors, utilities and Natural messages. You can invoke the online help for a particular help topic either by navigating through the Natural Help utility and its subordinate menus, or by directly accessing specific help topics.

This section covers the following topics:

Invoking the Menu of the Natural Help Utility

To invoke the menu of the Help utility

1 Enter the following system command:

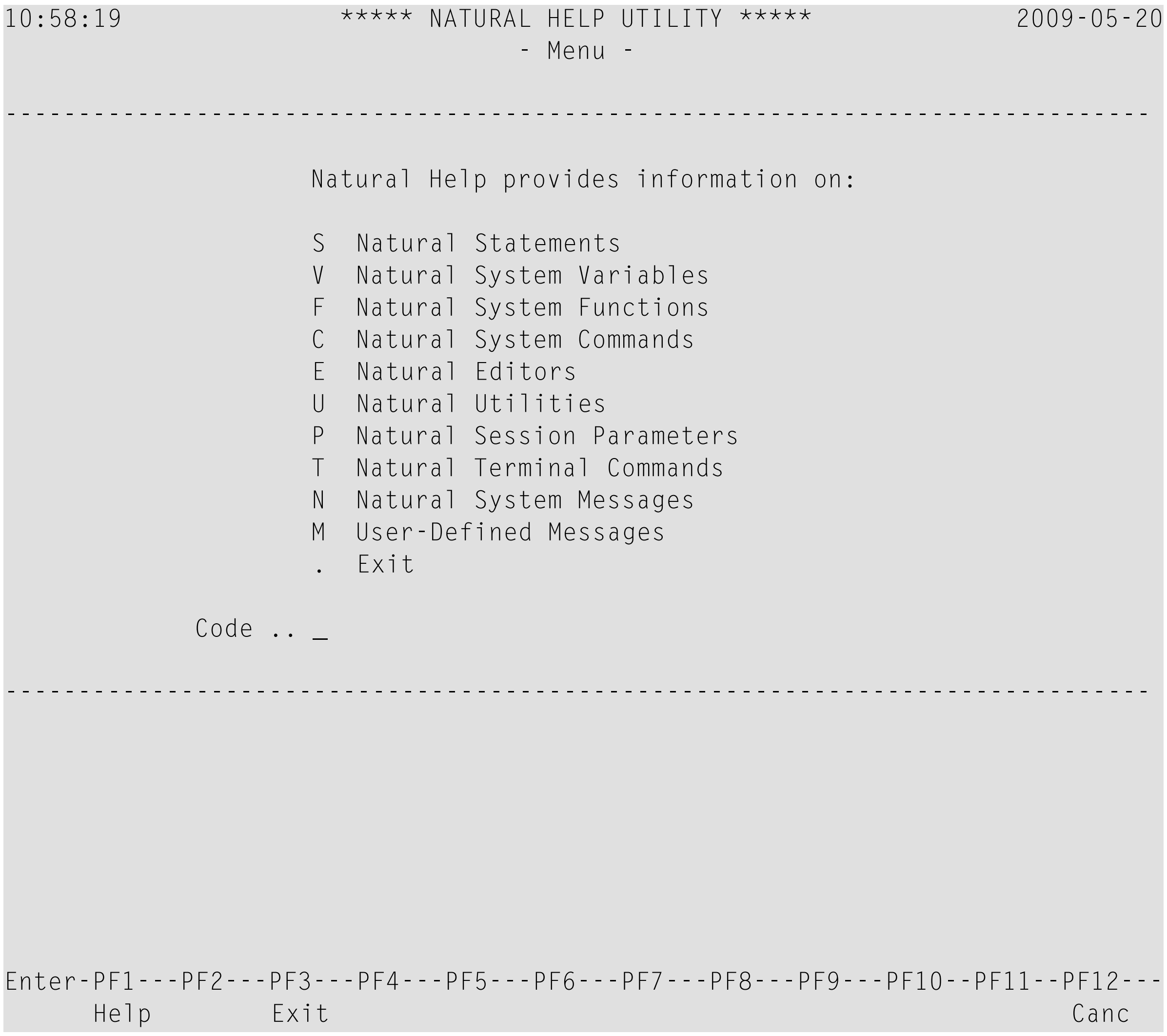
```
HELP
```

or

```
?
```

2 Choose ENTER.

The **Menu** of the Natural Help utility similar to the example below appears with the list of help topics provided:



```
10:58:19              ***** NATURAL HELP UTILITY *****             2009-05-20
                                  - Menu -

------------------------------------------------------------------------------

                    Natural Help provides information on:

                    S  Natural Statements
                    V  Natural System Variables
                    F  Natural System Functions
                    C  Natural System Commands
                    E  Natural Editors
                    U  Natural Utilities
                    P  Natural Session Parameters
                    T  Natural Terminal Commands
                    N  Natural System Messages
                    M  User-Defined Messages
                    .  Exit

             Code .. _

------------------------------------------------------------------------------








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

3 In the **Code** field, enter the one-letter code that corresponds to the help topic desired.

4 Choose ENTER.

A result screen appears with information on the help topic selected, or another menu is invoked with further help topics that help narrowing down your search.

Directly Accessing Help Topics

As an alternative to navigating through the subordinate menus of the Natural Help utility, you can also directly access help information on a Natural command or the Natural programming language.

To directly access a help topic

1 Enter the following system command:

```
HELP command
```

where `command`, for example, is the system command `EDIT` about which you request information:

```
HELP EDIT
```

2 Choose ENTER.

The **EDIT** screen of the Natural Help utility appears with information on the command requested as shown on the example screen below:

```
15:20:54               ***** NATURAL HELP UTILITY *****              2009-05-20
                                  - EDIT -                           Page  1

-------------------------------------------------------------------------------

Function: Invoke a Natural editor for the purpose of editing a Natural object.

Parameters:
  Object-type: Copycode(C), Description(D), Global(G), Helproutine(H),
               Local(L), Map(M), Parameter(A), Program(P), Subprogram(N),
               Subroutine(S), Text(T) or Class(4).
               -> Must be specified if object-name is not specified.
  Object-name: The name of the object to be edited, which will then be
               loaded in the edit work area.
  Library-ID : May only be specified if the object is contained in a library
               other than the one to which you are currently logged on.
               -> Must not start with 'SYS' except 'SYSTEM'
               -> Must not be specified if Natural Security is installed.

-------------------------------------------------------------------------------

                                                                      More ...
```

```



Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Menu  Exit                        -     +                     Canc
```

If the help text extends beyond the screen, `Page 1` appears in the upper-right corner of the screen and `More ...` appears in the bottom-right corner of the screen.

3 Choose ENTER or choose PF8 to scroll down the text.

The end of the text is indicated by `Last page` in the bottom-right corner of the screen.

Choose PF7 if you want to scroll up the text and return to the first page (`Page 1`).

Detailed Information on Error Messages

This section contains information on Natural system error messages and user-defined messages.

The system messages issued by Natural begin with `NAT` followed by a four-digit number *nnnn*.

For each Natural error message, there is a short text and a long text:

- The short text is the one-line message which is displayed when the error occurs.
- The long text is an extended explanation of the error and the action to be taken.

When Natural issues a system message, only the error number and the short text are displayed on the screen. When using the system command `HELP`, you can also display the long text of the message.

This section covers the following topics:

Displaying the Long Message Text

This section provides instructions for displaying the long message text of a Natural system error message or user-defined message. See also the **instructions** for displaying the long text of the Natural error that occurred last.

To display the long text of a Natural system message

1 Enter one of the following system commands:

```
HELP NATnnnn
```

or

```
? nnnn
```

where *nnnn* is the four-digit error number.

2 Choose ENTER.

The **Natural System Message** screen of the Natural Help utility appears with the long text of the error requested. This screen is similar to the **example screen** of Natural system error NAT0082 shown in the following section.

To display the long text of a user-defined message

1 Log on to the library where the required user-defined message is stored.

2 Enter the following system command:

```
HELP USER nnnn
```

or

```
? U nnnn
```

where *nnnn* is the four-digit error number.

3 Choose ENTER.

The **User Message** screen of the Natural Help utility appears with the long text of the user-defined message requested. This screen is similar to the **example screen** of Natural system error NAT0082 shown in the following section.

Information on the Last Error

You can display the short and the long text of the error message that occurred last in the current Natural session either by using the command `HELP ERROR` or by positioning the cursor as described below.

You can list the short text of the error message(s) that occurred last and additional information on the error situation by using the system command `LASTMSG`. The information displayed includes associated error messages that possibly preceded the last message.

To display the long text of the error that occurred last

- Use one of the following methods:

  1. Enter the following system command:

     ```
     HELP ERROR
     ```

Choose ENTER.

The **Natural System Message** screen of the Natural Help utility appears with the long text of the Natural error that occurred last as shown in the example screen below:

```
15:18:27             ***** NATURAL HELP UTILITY *****            2009-05-20
Library SAG          - Natural System Message NAT0082 -            Page 1


   Invalid command, or Program ANTON does not exist in library.


Tx *** Short Text ***


   Invalid command, or ... ... does not exist in library.


Ex *** Explanation ***


   One of the following has occurred:

   - You entered a value in the command line which is neither a

     Natural command nor the name of a Natural program contained

     in the active library or in a library defined as a steplib.

   - An object which is required during execution of a program,

     subprogram, subroutine or helproutine is not contained in






                                                                  More ...
```

```





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

2. Or:
   Place the cursor in the message line and choose PF1 (Help). This alternative only applies when the message line is displayed on a menu screen.

   A **Current Natural Message** window similar to the example below appears:

```
11:22:48                   *****  NATURAL  *****                    2009-05-20
User SAG                        - Main Menu -                Library TEST



                   Function



                _  Development Functions

                _  Development Environment Settings

                _  Maintenance and Transfer Utilities

                _  Debugging and Monitoring Utilities

                _  Example Libraries

 +-------------------- Current Natural Message NAT0082 --------------------+

 | Sh Invalid command, or Program ANTON does not exist in library.
|
 |
|
 | Tx Invalid command, or ... ... does not exist in library.
|
 | Ex One of the following has occurred:
|
 |    - You entered a value in the command line which is neither a
|
 |      Natural command nor the name of a Natural program contained
```

```
|
 |      in the active library or in a library defined as a steplib.
|
 |    - An object which is required during execution of a program,
|
 |      subprogram, subroutine or helproutine is not contained in
|
 |      the active library or in a library defined as a steplib.
|
 |    - Your Natural session is currently applying system files other
|
 |      than those containing the object you specified.
|
 | Ac Check to ensure that you entered a valid Natural command or the name
|
 |    of an existing Natural object. Use the command SYSPROF to check
|
 |    whether you are using the correct system file.
|
 +-----------------------------------------------------------------------+

NAT0082 Invalid command, or Program ANTON does not exist in library.



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

Note: With the user exit routine USR2002P (see the *Operations* documentation), you can customize the **Current Natural Message** window.

To display the last error message(s) and further details

1 Enter the following system command:

```
LASTMSG
```

2 Choose ENTER.

The **LASTMSG** window appears with the short text of the error message(s) that occurred last.

3 To open the **Detailed Information** window with further information on the error message(s):

In the **LASTMSG** window, place the cursor in the line that contains the message for which you require additional information and choose ENTER.

A **Detailed Information** window similar to the example below appears:

```
+---------------------------------- LASTMSG ------------------------More:  ++
! ERRTST1  0080 NAT0917 Error 920 in COPYCODE ERRTST2.                       !
! ERRTST2  0090 NAT0917 Error 920 in COPYCODE ERRTST3.                       !
! ERRTST3  0070 NAT0920 Program HUGO cannot be loaded (00000004).            !
!  +--- Detailed Information for NAT0920 ---+                                !
!  ! Error Number .. 920                     !                               !
!  ! Error Line .... 70                      !                               !
!  ! Object ........ ERRTST3                 !                               !
!  ! Object Type ... Copycode                !                               !
!  ! Level ......... 5                       !                               !
!  ! Library ....... SYSEXV                  !                               !
!  ! DBID/FNR ...... 10 / 410                !                               !
!  ! Error Class ... System                  !                               !
!  ! Error Type .... Runtime                 !                               !
!  ! Error Time .... 2003-02-27 15:58:01     !                               !
!  +-----------------------------------------+                               !
!                                                                            !
! Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF1 !
!                  Exit                    --    ++          <<    >     Can !
+----------------------------------------------------------------------------+
Command ===> LASTMSG









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

For further details, see *LASTMSG* in the *System Commands* documentation.

Specific Information on Menus and Fields

In addition to the help topics of the Natural Help utility, Natural provides individual help information on all Natural menus and the input fields available.

To invoke help for a menu

- In the command line or in the **Code** field, enter a question mark (?) and choose ENTER.

  Or:

Choose PF1 (Help).

To invoke help for a field

1 Place the cursor at the beginning of the field required and enter a question mark (?).

2 Choose ENTER.

5 Using Natural Libraries

Natural objects stored in the Natural system files FNAT and FUSER are grouped into logical constructs called libraries. A Natural library is used to organize objects according to functional criteria. For example, a library can consist of a set of objects that perform a particular task. If Natural Security is installed, a library can also be used to restrict object access to a particular group of users such as administrators.

A Natural application can access objects in multiple libraries depending on how the environment is set up.

All operations on a library are performed with Natural menu functions or corresponding commands. See also the section ***Using Commands and Menu Functions***.

Related Topics:

- *Natural System Files* - *Natural System Architecture* documentation
- *Libraries in System Files* - *Natural System Architecture* documentation
- *Library Maintenance* - *Natural Security* documentation

Objects Contained in a Library

A library contains object modules (source object and cataloged object, if applicable) of the following types of Natural object: program, subprogram, subroutine, copycode, helproutine, class, text, recording, map, local data area, global data area, parameter data area, command processor source, error message, dialog, debug environment, adapter and resource.

Related Topics:

- ***Cataloged Object***
- ***Source Object***

Library Types

This section describes the types of Natural library provided:

- User Library

- System Library

User Library

A user library contains objects of a user application which are specific to this application and which are required to run the application. A user library is contained in the FUSER system file.

System Library

A system library is maintained by Software AG. It is contained in the FNAT system file.

A system library contains all objects required to run a Natural system application.

In addition, there are system libraries that contain objects of the type program, which are provided for demonstration purposes: see ***Example Libraries*** in the section *Natural Main Menu*.

Note that you cannot log on to the system libraries SYSLIB and SYSLIBS. They are reserved for internal use by Software AG. If you try to log on to either library, you will receive the following message: `Libraries SYSLIB and SYSLIBS reserved for system usage`.

⚠ **Important:** Do not store any user-specific objects in a Natural system library. User-specific objects can be overwritten when Software AG installation datasets (for example, system maintenance upgrades, new versions or fix tapes) are loaded into the system files.

Steplib Libraries

A steplib is a Natural user library or system library that is concatenated with the current user or system library. A steplib can be used as a single storage location for multiple objects that are shared by different applications. This avoids redundant storage of identical objects and helps organize applications.

A steplib is a library in which Natural searches when an object is not found in the current library (see also the following section). The standard steplibs are the libraries SYSTEM in the FUSER and the FNAT system files.

Related Topics in the Parameter Reference Documentation:

- *STEPLIB - Additional Steplib Library*
- *BPSFI - Object Search First in Buffer Pool*

This section covers the following topics:

- Additional Steplibs with Natural Security
- Additional Steplibs without Natural Security

- Checking the Steplib Setting

Additional Steplibs with Natural Security

If Natural Security is active, you can define additional steplibs in the security profile of each library. The entries in a library security profile override any definitions made outside Natural Security. These steplibs are searched for an object before the standard steplibs SYSTEM (FUSER and FNAT).

Related Topic:

- *Steplibs - Natural Security* documentation

Additional Steplibs without Natural Security

If Natural Security is not active, you can specify one additional steplib with the Natural profile parameter `STEPLIB` at session start. In addition, you can define further steplibs by using one of the Application Programming Interfaces (for example, USR1025N or USR3025N) that are supplied for this purpose in the Natural system library SYSEXT.

These steplibs are searched for an object before the standard steplibs SYSTEM (FUSER and FNAT).

Checking the Steplib Setting

To check the steplib setting of your current library

1. Enter the following system command:

   ```
   TECH
   ```

   The **TECH** window appears.
2. Scroll down the window by choosing ENTER repeatedly until the **Steplib** column appears.

Related Topic:

- *TECH - System Commands* documentation

Search Sequence for Object Execution

This section describes the sequence in which Natural libraries and system files are searched for a requested object that is to be executed from either a user library or a system library.

Note: If the profile parameter `BPSFI` is set to `ON` (the default setting is `OFF`), objects are searched for in the buffer pool first.

The search sequence for a user-written object to be executed from a user library is as follows:

1. The current library in the FUSER system file as defined by the system variable `*LIBRARY-ID`.
2. The steplibs (in sequence) as specified in the Natural Security profile for the current library or in the steplib table.
3. The default steplib as defined by the system variable `*STEPLIB`.
4. The library SYSTEM in the FUSER system file.
5. The library SYSTEM in the FNAT system file.

The search sequence for a Natural object to be executed from a system library is as follows:

1. The current "SYS" library in the FNAT system file as defined by the system variable `*LIBRARY-ID`.
2. The steplibs (in sequence) as specified in the Natural Security profile for the current library or in the steplib table.
3. The library SYSLIBS in the FNAT system file, which contains objects shared by system commands and utilities.
4. The library SYSTEM in the FNAT system file.
5. The library SYSTEM in the FUSER system file.

   Since the FUSER system file is searched last, you must provide an object that is used in both the FUSER and the FNAT system files (for example, a user-exit routine for a Natural utility) only in one location, namely in FUSER.

Related Topics in the System Variables Documentation:

- **LIBRARY-ID*
- **STEPLIB*

Default Library Assignment

When you start a Natural session, you are logged on to a library assigned by Natural. A Natural screens usually provides a field (for example, **Library** or **LIB=**) that indicates the name (ID) of your library, that is, the current library (for example, `LIB=SYSTEM`) where Natural objects are stored and from which they are retrieved .

To find out your current library

- Enter the following system command:

```
TECH
```

The **TECH** window appears. The library where you are currently logged on is listed in the **Library** field.

The default library ID assigned by Natural is `SYSTEM` if the Natural profile parameter `AUTO` is set to `OFF`. You can change the default library ID at session startup by setting `AUTO` to `ON`. `AUTO=ON` initiates an automatic logon to the library that corresponds to your user ID. However, different rules apply if you log on to Natural under Natural Security as described in the *Natural Security* documentation.

You may have to log on to another library to perform a maintenance function or work on a different application as described in ***Logging on to a Library***.

Related Topics:

- *TECH - System Commands* documentation
- *AUTO - Parameter Reference* documentation
- *Logging On - Natural Security* documentation

Logging on to a Library

When you create or maintain Natural objects or execute a Natural program inside a specific library, you may have to switch libraries and first log on to the library or steplib (see ***Steplib Libraries***) that contains (or is to contain) the object.

You usually use the system command `LOGON` to log on to a library. However, different rules apply if you log on to Natural under Natural Security as described in the *Natural Security* documentation.

After a successful logon to a library that does not contain any objects, the confirmation message reads: `This library is empty.`

Note: The system command `LOGON` does not consider objects of the types error message and debug environment. Therefore, the confirmation message only refers to all object types other than error messages and debug environments. You can find out whether a library contains error messages or debug environments, for example, by using the SYSMAIN utility, or the utilities SYSERR and the debugger respectively.

You can also find out whether a library is empty (except for error messages and/or debug environments) by using the system command `LIST` described in ***Listing Objects in a Library***.

The library to which you are logged on remains active until you log on to another library or terminate your Natural session.

To log on to a library

1. Enter the following system command:

   ```
   LOGON library-ID
   ```

   where *library-ID* is the name (ID) of the library you want to access.

   Or:
   From the Natural **Main Menu** or its subordinate menus:

   In the top right-hand corner of the screen, in the **Library** field, replace the library ID displayed with another library ID (see also *Example of a Menu Function* in the section *Using Commands and Menu Functions*).
2. Choose ENTER.

   If the library ID you entered complies with the **library naming conventions**, the following confirmation message appears: `Logon accepted to library library-ID`.

Related Topics:

- *LOGON* and *LOGOFF* - *System Commands* documentation
- *Logging On* - *Natural Security* documentation

Creating a Library

You create a library by using either the system command `LOGON` for an empty library where you create an object, or the move or copy function of a Natural utility when transferring objects.

To create a library

- If you use the system command `LOGON`, proceed as described in ***Logging on to a Library***. Specify a library ID (see also ***Library Naming Conventions***) that logs you on to a library that does not contain any objects. In this library, create at least one cataloged object or one source object.

  Or:

  If you want to move or copy objects from an existing library to a new one, proceed as described in ***Moving, Copying and Renaming Libraries***.

Listing Libraries

You can obtain a list of all libraries available in your current Natural system environment, for example, by using the SYSMAIN utility.

The instructions in this section are examples of listing libraries by using either SYSMAIN menu functions or a corresponding command.

To list all libraries using menu functions

1 From the Natural **Main Menu**, choose **Maintenance and Transfer Utilities**.

2 Choose ENTER.

The **Maintenance and Transfer Utilities** screen appears.

3 Select **Transfer Objects to Other Libraries**.

Or:

Enter the following system command:

```
SYSMAIN
```

4 Choose ENTER.

The **Main Menu** of the SYSMAIN utility similar to the example below appears:

```
17:33:01             ***** NATURAL SYSMAIN UTILITY *****            2009-05-20
 User SAG                         - Main Menu -


             Code  Object                           Code  Function

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

Object Code .. A                       Function Code .. _
```

```
Command ===>









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

5 In the **Object Code** field, enter an `A` (default setting) to list all libraries that contain Natural objects except error messages and debug environments (for error messages, enter an `E`, for debug environments a `D`).

In the **Function Code** field, enter an `L` (for **List**).

6 Choose ENTER.

The **List Programming Objects** screen appears.

7 In the **Code** field, enter an `A` to search for libraries that contain all types of object module: cataloged objects and source objects.

In the **Source Library** field, enter an asterisk (*) to search for all libraries.

(For valid name ranges, see *Specifying a Range of Names* in the *SYSMAIN Utility* documentation.)

Leave all other input fields unchanged.

8 Choose ENTER.

The **Library Selection** screen appears with a list of all libraries available in the current system file.

Choose ENTER to scroll down the list, PF11 to scroll right and PF12 to scroll left.

From the **Library Selection** screen, you can select a particular library and list the objects contained in that library. For further information on the **Library Selection** screen, see *Related Topic* below.

To list all libraries using a command

1 Enter the following system command:

```
SYSMAIN LIST ALL * IN LIBRARY *
```

2 Choose ENTER.

The **Library Selection** screen appears with a list of all libraries available in the current system file.

Related Topic:

- *Listing and Selecting Libraries - SYSMAIN Utility* documentation

Listing Objects in a Library

You can obtain a list of objects contained in a library by using either the system command `LIST` or the corresponding function of the Natural utility SYSMAIN or the Object Handler.

Note: The system command `LIST` does not process objects of the types error message and debug environment. You can list error messages or debug environments, for example, by using the SYSMAIN utility.

This section describes how to obtain a list of objects by using `LIST`.

To list objects using `LIST`

1 Enter the following system command:

```
LIST *
```

where asterisk (*) specifies that all source objects and cataloged objects are to be listed (see also: *`object-name-range`* in *LIST* in the *System Commands* documentation).

2 Choose ENTER.

A **LIST Objects in a Library** screen similar to the example below appears with a list of source objects and cataloged objects available in the current library (in the example below: `TEST`) in the current system file:

```
12:22:23              ***** NATURAL LIST COMMAND *****              2009-05-20
User SAG                 - LIST Objects in a Library -         Library TEST

Cmd  Name        Type          S/C  SM Version  User ID    Date         Time
---  *________   *__________   *__  *  *______  *________  *__________  *________
__   DBTEST      Program       S/C  S  4.2.01   SAG        2002-02-27   16:46:36
__   DEMOPROG    Program       S/C  S  4.1.02   SAG        2000-12-20   11:40:46
__   DEMOSPRG    Subprogram    S/C  S  4.1.02   SAG        2000-12-22   11:32:34
__   DEMOTEXT    Text          S       4.1.02   SAG        2000-12-20   13:59:26
__   EMP-L       Local         S/C     4.1.02   SAG        2002-07-29   12:46:43
__   EMPL        Program       S/C  S  4.1.02   SAG        2002-07-29   12:46:30
__   GDATEST     Global        S/C     4.1.03   SAG        2004-11-22   13:24:32
__   LDATEST     Local         S       4.2.01   SAG        2004-05-21   17:18:13
__   MAPTEST    Map          S/C  S  4.2.03   SAG         2000-12-20  11:40:46
__   PDATEST    Parameter    S/C     4.1.03   SAG         2004-05-05  11:46:24
__   PDATEST2   Parameter    S/C     4.1.03   SAG         2004-11-23  10:12:07
__   PROCTEST   Processor    S/C     4.2.01   SAG         2004-05-05  11:25:22
__   SUBRTEST   Subroutine   S/C  S  4.1.03   SAG         2002-07-23  14:37:16
__   TESTPGM    Subprogram   S    S  4.1.03   SAG         2000-12-20  11:40:45
                                                               14 Objects found
Top of List.
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Print Exit  Sort        --    -     +     ++          >     Canc
```

The list is sorted by the name of the object (**Name** column) and contains further information on the object such as the type of object (for example, `Program`), the type of object module (**S/C** column: `S` = source object, `C` = cataloged object) available and the ID of the user who created or modified the object.

From the **LIST Objects in a Library** screen, you can select an object for further processing such as listing the source code of an object or deleting objects from a library as described in ***To delete objects using LIST*** in the section *Deleting Objects in a Library*.

For further information on the **LIST Objects in a Library** screen and other options provided with `LIST` such as listing a particular range of object names or sorting the list of objects, see the related topics below.

Related LIST Topics in the System Commands Documentation:

- *`object-name-range`*
- *Explanation of the Column Headers*
- *Sorting the List of Objects*
- *Performing a Function on an Object*

Printing a List of Objects

You can print a list of objects contained in the current library in the current system file.

To print a list of objects

1. Invoke the **LIST Objects in a Library** as described in **Steps 1 and 2** in *Listing Objects in a Library*.
2. Choose PF2 (Print).

   The **PRINT** window appears.

3 In the **Destination** field, enter a valid printer name (if required, ask your Natural administrator for a printer available in your current environment). If required, change the page size (the default setting is 60 lines).

4 Choose ENTER.

The **Printout Specification** screen appears where you can specify printer settings such as the amount of copies to be printed.

5 Choose ENTER.

The list of objects contained in the library is printed on the specified printer device.

Related Topics:

- *Printing Objects*
- *LIST - System Commands* documentation

Finding Objects in a Library

You can use the SYSMAIN utility or the system command `SCAN` to search for objects contained in a library.

- Searching for Objects using SYSMAIN
- Searching for Objects using SCAN

Searching for Objects using SYSMAIN

The following instructions are examples of finding objects by specifying search criteria by using either SYSMAIN menu functions or a corresponding command.

To find objects using menu functions

1 Invoke the **Main Menu** of the SYSMAIN utility as described in **Steps 1 through 4** of *To list all libraries using menu functions*.

2 In the **Object Code** field, enter an `A` (default setting) to search for all types of object.

(Exceptions: for error messages, enter an `E`, for debug environments a `D`.)

In the **Function Code** field, enter an `F` (Find).

3 Choose ENTER.

The **Find Programming Objects** screen appears.

4 In the **Code** field, enter an `A` to search for all types of object module: cataloged objects and source objects.

In the **Object Name** field, enter an asterisk (*) to search for all object names. Asterisk (*) is the default setting.

(For valid name ranges, see *Specifying a Range of Names* in the *SYSMAIN Utility* documentation.)

In the **Source Library** field, enter the ID of the library, in which to perform the search.

In the **Criteria** field, replace `N` (No) by `Y` (Yes). `N` is the default setting.

Leave all other input fields unchanged.

5 Choose ENTER.

The **Additional Criteria** window appears where you can enter additional search criteria as shown in the example below:

```
18:23:16              ***** NATURAL SYSMAIN UTILITY *****            2009-05-20
 User SAG                  - Find Programming Objects -

                    +------------------------------------------+
                    !       --- Additional Criteria ---        !
                    !                                          !
                    !  Object Type ..... PM__________________  !
                    !  Date/Time From .. 2006-05-09 _____      !
                    !  Date/Time To .... 2009-05-01 _____      !
                    !  User ID ......... SAG______             !
                    !  Terminal ID ..... ________              !
                    !                                          !
                    !                                          !
                    !  Command ===>                            !
            Code .. !                                          !
    Object  Name .. +------------------------------------------+

    Source  Library ... TEST____    Database .... 10___  File .. 32___
```

```



   Options                            Criteria .... Y





Command ===>









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

6 Enter one or more search criteria and choose ENTER.

The window is closed. A plus (+) sign in front of the **Criteria** field indicates when additional criteria other than the object type are specified in the window.

7 Choose ENTER again.

The **Find Selection** screen appears with a list of all source objects and cataloged objects that are available in the specified library in the current system file and that meet the specified search criteria. In the example above, all objects of the types `P` (program) and `M` (map) that were created or modified between `2006-05-09` and `2009-05-01` by the user `SAG` are listed.

8 Choose ENTER to scroll down the list until the following message appears: `Library has been successfully listed.`

From the **Find Selection** screen, you can select an object for further processing such as listing the source code of an object as described in *Selection Lists for Programming Objects* in the *SYSMAIN Utility* documentation.

Related Topic:

- *Using Menu Functions and Commands* - *SYSMAIN Utility* documentation

To find objects using a command

1 Enter the following system command:

```
SYSMAIN FIND ALL * TYPE PN WITH USER user-ID IN library-ID
```

where:

ALL specifies that all source objects and cataloged objects are to be selected for the search.

Asterisk (*) specifies that all object names are to be selected for the search (see also: *Specifying a Range of Names* in the *SYSMAIN Utility* documentation).

PN specifies the types of object for which to search: P denotes program, N denotes subprogram.

library-ID is the ID of the library in which to search.

user-ID is the ID of the user for which to search.

2 Choose ENTER.

The **Find Selection** screen appears with a list of all source objects and cataloged objects of the types program and subprogram that were created or modified by the specified user.

Related Topic:

- *Keywords and Variables in Direct Commands - SYSMAIN Utility* documentation

Searching for Objects using SCAN

The following instructions are examples of finding objects by scanning their sources for a particular string of characters by using the system command SCAN.

Note: The SCAN does not process objects of the types error message and debug environment. You can scan error messages by using the **Scan in messages** function of the Natural utility SYSERR.

To scan sources using menu functions

1 Enter the following system command:

```
SCAN
```

2 Choose ENTER.

A **Scan Objects in Libraries** screen similar to the example below appears where you can specify an object range and a scan value:

```
18:24:53               ***** NATURAL SCAN COMMAND *****            2009-05-20
User SAG                - Scan Objects in Libraries -         Library TEST

                    Code  Function

                     T    Statistics
                     L    List of Objects Containing Scan Value
                     S    Object Lines with Scan Value
                     ?    Help
                     .    Exit


   Code ............ L
   Scan value ...... LOCAL___________________________
   Replace value ... GLOBAL__________________________
   Library ......... TESTLIB_
   Object name ..... *________                Selection list .. N
   Object type(s) .. *_____________
   Absolute scan ... N                        Trace ........... N

Command ===>










Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Menu  Exit
```

In the **Code** field, enter an `L` (for **List of Objects Containing Scan Value**) and specify a scan value, a library and an object range; asterisk (*) selects all object names and object types.

In the **Replace value** field, you can enter a character string (for example, `GLOBAL`) that is to replace the specified scan value (for example, `LOCAL`).

3 Choose ENTER.

The **Select Objects for Library** screen appears for the specified library in the current system file with a list of all source objects that meet the specified search criteria. In the example above, all objects in the library `TESTLIB` that contain the character string `LOCAL` are selected.

From the **Select Objects for Library** screen, you can select an object and display or modify the source line that contains the scan value.

To scan sources using a command

1. Enter the following system command:

   ```
   SCAN FUNC=L,SVAL=scan-value,LIB=library-ID,*
   ```

   where:

   `scan-value` is the character string (for example, `LOCAL`) for which to scan sources.

   `library-ID` is the ID of the library in which to search (for example, `TESTLIB`).

   Asterisk (*) specifies that all source objects are selected for the search.
2. Choose ENTER.

Related Topic:

- *SCAN* - *System Commands* documentation

Copying, Moving and Renaming Libraries

You copy or move a library by copying or moving all Natural objects from a source library to an existing or a new target library.

Renaming a library requires the same steps as moving a library. Once all objects of a library have been moved to a new target library, the old library is deleted automatically.

You can copy or move single or multiple objects from one library to another by using either the SYSMAIN utility or the Object Handler. The instructions below are examples of copying or moving all objects with SYSMAIN by using either menu functions or corresponding commands.

To copy or move all objects using menu functions

1. Invoke the **Main Menu** of the SYSMAIN utility as described in **Steps 1 through 4** of *To list all libraries using menu functions*.
2. In the **Object Code** field, enter an `A` (default setting) to select all types of object.

   (Exceptions: for error messages, enter an `E`, for debug environments a `D`.)

   In the **Function Code** field, enter a `C` (for **Copy**) or an `M` (for **Move**).
3. Choose ENTER.

   Depending on the function code entered, either the **Copy Programming Objects** or the **Move Programming Objects** screen appears.

4 In the **Code** field, enter an `A` to select all types of object module: cataloged objects and source objects.

In the **Sel. List** (Selection List) field, replace `Y` (Yes) by `N` (No). `Y` is the default setting.

In the **Object Name** field, enter an asterisk (*) to select all object names. Asterisk (*) is the default setting.
(For valid name ranges, see *Specifying a Range of Names* in the *SYSMAIN Utility* documentation.)

In the **Source Library** field, enter the ID of the library that contains the objects to be copied or moved.

In the **Target Library** field, enter the ID of an existing or a new library to which you want to copy or move the objects.

Leave all other input fields unchanged.

5 Choose ENTER.

All source and cataloged objects are copied or moved between the specified source and target libraries in the current system file and the following message appears: `Function completed successfully.`

To copy or move all objects using commands

1 Enter one of the following system commands:

```
SYSMAIN COPY ALL * FM old-library TO new-library
```

(copies objects)

or

```
SYSMAIN MOVE ALL * FM old-library TO new-library
```

(moves objects)

where:

`ALL` specifies that all source objects and cataloged objects are to be selected.

Asterisk (*) specifies that all object names are to be selected
(see also: *Specifying a Range of Names* in the *SYSMAIN Utility* documentation).

`old-library` is the ID of the library that contains the objects to be copied or moved.

`new-library` is the ID of an existing or a new library (see also ***Library Naming Conventions***).

2 Choose ENTER.

If the copy or move operation was successful, the command processed is displayed on the screen followed by the message: `Function completed successfully.`

For further information on the functions provided with the SYSMAIN utility such as replacing or renaming objects, refer to the *Utilities* documentation.

Deleting Objects in a Library

A library is only maintained in a Natural environment as long as it contains at least one source object or one cataloged object. If you delete all objects from a library, this library is no longer available.

You can delete single or multiple objects from a library by using the system command `LIST` or a Natural utility.

The instructions below are examples of deleting objects from the current library by using the system command `LIST`, and deleting objects from a different library by using the menu functions or commands of the SYSMAIN utility.

To delete objects using `LIST`

1. Enter the following system command:

   ```
   LIST *
   ```

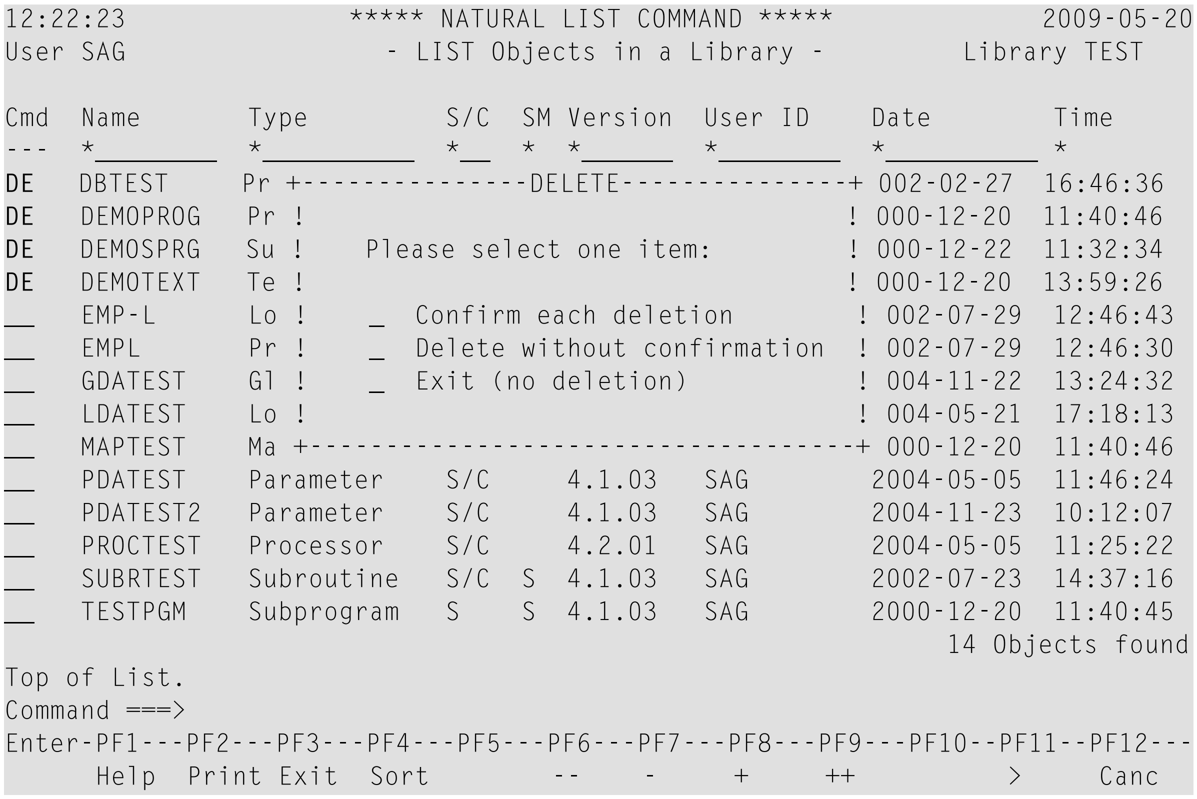
   where asterisk (*) specifies that all source objects and cataloged objects are to be listed (see also: *`object-name-range`* in *LIST* in the *System Commands* documentation).
2. Choose ENTER.

   The **LIST Objects in a Library** screen appears with a list of source objects and cataloged objects available in the current library in the current system file.
3. In the **Cmd** column, next to the object(s) required, enter the following:

   ```
   DE
   ```

   as shown on the example screen below.
4. Choose ENTER.

A **DELETE** window similar to the example below appears:



```
12:22:23              ***** NATURAL LIST COMMAND *****              2009-05-20
User SAG                - LIST Objects in a Library -         Library TEST

Cmd  Name        Type         S/C  SM Version  User ID    Date         Time
---  *________   *__________  *__  *  *______  *________  *__________  *
DE   DBTEST     Pr +---------------DELETE---------------+ 002-02-27  16:46:36
DE   DEMOPROG   Pr !                                    ! 000-12-20  11:40:46
DE   DEMOSPRG   Su !    Please select one item:         ! 000-12-22  11:32:34
DE   DEMOTEXT   Te !                                    ! 000-12-20  13:59:26
__   EMP-L      Lo !    _  Confirm each deletion         ! 002-07-29  12:46:43
__   EMPL       Pr !    _  Delete without confirmation   ! 002-07-29  12:46:30
__   GDATEST    Gl !    _  Exit (no deletion)            ! 004-11-22  13:24:32
__   LDATEST    Lo !                                     ! 004-05-21  17:18:13
__   MAPTEST    Ma +-------------------------------------+ 000-12-20  11:40:46
__   PDATEST     Parameter    S/C     4.1.03   SAG        2004-05-05  11:46:24
__   PDATEST2    Parameter    S/C     4.1.03   SAG        2004-11-23  10:12:07
__   PROCTEST    Processor    S/C     4.2.01   SAG        2004-05-05  11:25:22
__   SUBRTEST    Subroutine   S/C  S  4.1.03   SAG        2002-07-23  14:37:16
__   TESTPGM     Subprogram   S    S  4.1.03   SAG        2000-12-20  11:40:45
                                                              14 Objects found
Top of List.
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Print Exit  Sort        --    -     +     ++          >     Canc
```

5 Mark an item by typing in any character next to the option required and choose ENTER:

Confirm each deletion invokes a window for each object to be deleted in which you type in the name of the object to confirm the delete operation.

Delete without confirmation immediately executes the delete operation(s).

Exit cancels the delete operation(s).

To delete objects using SYSMAIN menu functions

1 Invoke the **Main Menu** of the SYSMAIN utility as described in **Steps 1 through 4** of *To list all libraries using menu functions*.

2 In the **Object Code** field, enter an `A` (default setting) to select all types of object.

(Exceptions: for error messages, enter an `E`, for debug environments a `D`.)

In the **Function Code** field, enter a `D` (for **Delete**).

3 Choose ENTER.

The **Delete Programming Objects** screen appears.

4 In the **Code** field, enter an `A` to select all types of object module: cataloged objects and source objects.

In the **Sel. List** (Selection List) field, replace `Y` (Yes) by `N` (No). `Y` is the default setting.

In the **Object Name** field, enter an asterisk (*) to select all object names. Asterisk (*) is the default setting.
(For valid name ranges, see *Specifying a Range of Names* in the *SYSMAIN Utility* documentation.)

In the **Source Library** field, enter the ID of the library you want to delete.

Leave all other input fields unchanged.

5 Choose ENTER.

All source objects and cataloged objects are deleted from the specified library in the current system file and the following message appears: `Function completed successfully.`

To delete all objects using a SYSMAIN command

1 Enter the following system command:

```
SYSMAIN DELETE ALL * FM library-ID
```

where:

`ALL` specifies that all source objects and cataloged objects are to be selected.

Asterisk (*) specifies that all object names are to be selected
(see also: *Specifying a Range of Names* in the *SYSMAIN Utility* documentation).

`library-ID` is the ID of the library that contains the objects to be deleted.

2 Choose ENTER.

If the delete operation was successful, the command processed is displayed on the screen followed by the message: `Function completed successfully.`

6 Maintaining and Executing Natural Objects

An object is a component of an application. A Natural application consists of a set of objects that interact with one another to perform a particular task.

Objects available for setting up and maintaining a Natural application comprise Natural objects and foreign objects.

Foreign objects are objects that have not been created with a Natural development function and that are stored outside a Natural and an Adabas environment. Examples of foreign objects are bitmaps, XML sources, HTML files, DL/I subfiles and Predict rules.

This section provides general information on Natural objects and describes the steps required to create, maintain, delete or execute an object.

All operations on a Natural object are performed with Natural commands and/or menu functions. For instructions on using commands and menu functions, refer to the section ***Using Commands and Menu Functions***.

Related Topics:

- ***Listing Objects in a Library***
- ***Finding Objects in a Library***
- ***Deleting Objects in a Library***

Introducing Natural Objects

The following characteristics identify a Natural object:

- It is stored in a Natural system file.
- It comprises a cataloged object and/or a source object.
- It is created with any of the Natural editors or utilities.

This section covers the following topics:

- Cataloged Object
- Source Object

- Object Types

Cataloged Object

A cataloged object is the executable (compiled) form of a Natural object. It is created by the Natural compiler and is stored as an object module in a Natural system file. Compiling source code and creating a cataloged object is referred to as cataloging an object. A cataloged object is created by using the Natural system command `CATALOG` or `STOW`.

At execution time, the cataloged object is loaded into the Natural buffer pool and executed by the Natural runtime system. Natural objects can only be executed or reference one another if they have been stored as cataloged objects in a Natural system file.

A cataloged object cannot be modified or decompiled.

Source Object

A source object (or a saved object) contains the human-readable form of Natural source code. Source code is saved as a source object in a Natural system file by using the Natural system command `SAVE` or `STOW`.

To execute source code contained in a source object, you need to compile the source code in order to create generated object code that can be interpreted and executed by the Natural runtime system.

Related Topics:

- *Natural System Files - Natural System Architecture* documentation
- ***Saving and Cataloging Objects***

Object Types

Within a Natural application, several types of Natural object can be used to establish an efficient application structure and to meet particular programming and application requirements. Natural object types include programs, subprograms, routines and data areas. For a description of all types of object available, refer to the section *Object Types* in the *Programming Guide*.

For information on data definition modules (DDMs), see *Natural Data Definition Modules* in the *Programming Guide*.

Using Natural Editors or Utilities

When you create, maintain or delete a Natural object, you use either a Natural editor or a Natural utility.

There are maintenance functions that do not apply to all types of object. For example, you cannot edit an object of the type adapter.

A Natural editor is invoked for all object types that can be specified with the system command `EDIT` or on the **Development Functions** screen. Depending on the object type specified, Natural invokes the appropriate editor: the program editor, the data area editor or the map editor. For example, for an object of the type program, Natural invokes the program editor.

A Natural utility is used for object types that either require additional administration services and/or are not maintained in a library such as DDMs. A utility provides its own editor.

The table below is an overview of Natural object types and their appropriate editor or utility:

Object Type	Editor or Utility
Local data area Global data area Parameter data area	Data area editor
Map (screen layout) Map editor profile Device profile	Map editor
Program Subprogram Subroutine Copycode Helproutine Class Text Program editor profile	Program editor
Error message	SYSERR utility
Data definition module (DDM)	SYSDDM utility
Command processor source	SYSNCP utility
Parameter profile	SYSPARM utility
Debug environment	Debugger

Related Topics:

- *Editors* documentation

- *Utilities* documentation

This section covers the following topics:

- Invoking a Natural Editor
- Invoking a Natural Utility
- Setting Editor Preferences

Invoking a Natural Editor

To invoke a Natural editor

- Use the system command `EDIT`.

  For an example of using `EDIT`, see ***Example of a System Command***.

  Or:

  From the Natural **Main Menu**, invoke the **Development Functions** menu (see *Natural Main Menu*) and choose either the function **Create Object** or **Edit Object**.

  For an example of invoking an editor, see ***Example of a Menu Function*** in the section *Using Commands and Menu Functions*.

Related Topic:

- *EDIT* - *System Commands* documentation

Invoking a Natural Utility

To invoke a Natural utility

- Enter one of the following system commands:

  ```
  SYSERR
  ```

  (for error messages)

  ```
  SYSDDM
  ```

  (for DDMs)

  ```
  SYSNCP
  ```

  (for command processor sources)

  ```
  SYSPARM
  ```

(for parameter profiles)

```
TEST
```

(for debug environments)

Or:

From the Natural **Main Menu**, invoke the appropriate menu and select the appropriate utility:

- **Maintenance and Transfer Utilities** for SYSERR, SYSDDM and SYSNCP.
- **Development Environment Settings** for SYSPARM.
- **Debugging and Monitoring Utilities** for TEST.

Related Topic:

- ***Natural Main Menu***

Setting Editor Preferences

When working with the Natural program editor or data area editor, you can use the editor profile function to display the current settings of the editor and set preferences to be in effect when editing source code.

To display or modify editor profile settings

1 At the command prompt of the program editor or data area editor, enter the following:

```
PROFILE
```

2 Choose ENTER.

The **Editor Profile** screen appears.

For information on the fields and options provided on the screen, see *Editor Profile* in the *Editors* documentation.

Selecting and Displaying Objects

You can display a source object to view or copy source code without modifying the source object. The source code of the specified object is then displayed in read-only mode in the editing area of the appropriate editor.

You can either select an object from a list or specify the name of the object you want to display.

This section describes how to list source code by using the system command `LIST`. As an alternative to `LIST`, you can use the **List Object(s)** function provided in the **Development Functions** menu described in *Natural Main Menu*.

To select an object from a list of objects

1 Invoke the **LIST Objects in a Library** screen as described in **Steps 1 and 2** of *To list objects using LIST*.

2 In the **Cmd** column, next to the object required, enter the following:

```
LI
```

3 Choose ENTER.

The source code of the selected object is displayed.

To display source code of a specified object

1 Enter the following system command:

```
LIST object-name
```

where *`object-name`* is the name of the object to be displayed.

If you do not specify *`object-name`*, the source code currently contained in the source work area is displayed.

2 Choose ENTER.

The source code of the specified object is displayed in read-only mode.

Related Topics:

- *Listing Objects in a Library*
- *LIST - System Commands* documentation

Creating and Editing Objects

This section describes the steps required to create and edit a Natural object by using a Natural editor. For information on using the Natural utilities mentioned earlier, refer to the relevant sections in the *Utilities* documentation.

Checking the Current Environment

A Natural object is created in the current library in the current system file. Before you start creating or editing an object, make sure that you are logged on to the library where you want to store or retrieve the object.

For instructions on library assignments and switching libraries, see ***Default Library Assignment*** and ***Logging on to a Library***.

Setting the Programming Mode

Natural offers two programming modes: reporting mode and structured mode.

For explanations of the two modes and instructions on how to change the mode from reporting to structured (or vice versa), see ***Programming Modes*** in the section *Natural Main Menu*.

Using the Natural Programming Language

The Natural programming language consists of statements, system functions and system variables.

Natural statements are programming instructions used to create a Natural program source.

Natural system functions, for example, are used to perform mathematical functions.

Natural system variables are standard variables that are provided and generated by Natural. System variables, for example, are used to obtain the date and time.

Related Topics:

- *Statements* documentation (overview)
- *System Functions* documentation

- *System Variables* documentation

Creating Source Code

This section describes how to create source code by using the system command `EDIT` and the program editor as an example. In addition, this section provides examples of editor commands and instructions for navigating in a source.

As an alternative to `EDIT`, you can use the **Create Object** function provided in the **Development Functions** menu described in *Natural Main Menu*.

To enter source code

1 Enter the following system command:

```
EDIT object-type
```

where *`object-type`* is the type of object you want to create.

For example, to create an object of the type program, enter the following:

```
EDIT PROGRAM
```

If you do not specify *`object-type`*, the program editor is invoked by default.

(See also ***Setting the Object Type***.)

2 Choose ENTER.

The editing area of the program editor appears where the type of object (here: `Program`) is displayed at the top of the screen as shown in the example below:

```
>                                        > +  Program                Lib SYSTEM
  All    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
    0010

    0020

    0030

    0040

    0050

    0060

    0070

    0080
```

```
0090

0100

    ....+....1....+....2....+....3....+....4....+....5....+... S 0    L 1
```

3 If the editing area is not empty, at the editor command prompt (>), enter the following editor command:

```
CLEAR
```

and choose ENTER.

`CLEAR` deletes the contents of the source work area.

4 Starting in the first line (numbered with `0010`) of the empty editing area, insert the source code by using the copy and paste functions provided by your terminal emulation (for example, Entire Connection), or by typing in the source code.

If you want to stop automatic conversion from lower to upper case, change the default setting in the editor profile as described in *General Defaults* in the *Editors* documentation.

5 As you fill up a screen, for more empty lines, enter the following editor command:

```
ADD
```

and choose ENTER.

The editor command `ADD` adds nine empty lines. From these lines, only the lines you fill in will be added to the program source. With the next ENTER, lines that are left empty are eliminated. You can change this default setting in the editor profile as described in *Editor Defaults* in the *Editors* documentation. For all program editor commands available, see the *Program Editor* documentation.

To scroll through a source

1 To return to the beginning of the source code, enter the following editor command:

```
TOP
```

2 To go to the end of the source code, enter the following editor command:

```
BOT
```

3 To scroll down one page in the source code, choose PF8 or ENTER.

4 To scroll up one page in the source code, choose PF7.

For all program editor commands available, see *Editor Commands for Positioning* in the *Program Editor* documentation.

Editing a Source Object

Once source code has been saved as a source object (as described in ***Saving and Cataloging Objects***), you open a Natural editor for a source object by specifying the name of the source object.

To edit source code of a source object

1. Enter the following system command:

   ```
   EDIT object-name
   ```

   where `object-name` is the name of an existing source object that is contained in the current library in the current system file.

2. Choose ENTER.

   The source code of the specified source object is displayed in modify mode in the editing area of the appropriate editor.

As an alternative to `EDIT`, you can use the **Edit Object** function provided in the **Development Functions** menu described in *Natural Main Menu*.

As an alternative to `EDIT`, you can also use the system command `READ` as described in ***Copying Objects***.

Related Topic:

- *EDIT - System Commands* documentation

Setting the Object Type

The object type is specified when creating an object (the default setting is program) or set automatically when an existing source object is read into the source work area. When working with the program editor or data area editor, you can change the object type any time by using the editor command `SET TYPE`.

To change the object type

1. Enter the following editor command:

   ```
   SET TYPE object-type
   ```

   where `object-type` denotes the type of object to be created.

   For example:

```
SET TYPE SUBPROGRAM
```

2 Choose ENTER.

The new object type specified with the command is indicated on the screen (here: `Subprogram`).

Related Topics:

- *SET TYPE - Program Editor* documentation

  SET TYPE - Data Area Editor documentation

Checking and Testing Objects

Source code compilation (cataloging) performs a syntax check and generates executable object code.

The source code contained in the source work area can be compiled without saving the source code first (as described in ***Saving and Cataloging Objects***). Additionally, compilation of source code for objects of the type program can be combined with program execution. See also ***Executing Programs***.

To compile source code for syntax checks

1 Enter the following system command:

```
CHECK
```

2 Choose ENTER.

If no syntax error is found, the source code contained in the source work area is compiled.

To compile source code for program execution

1 Enter the following system command:

```
RUN
```

2 Choose ENTER.

If no syntax error is found, the source code contained in the source work area is compiled and the generated code is executed.

Related Topics:

- *CHECK - System Commands* documentation
- *RUN - System Commands* documentation

Online Help for Syntax Errors

Source code compilation has been successful if no error message appears.

If Natural encounters a syntax error during compilation, an error message is displayed on the screen and the statement line that contains the error is highlighted and marked with an `E` as shown in the example below:

```
>                                       > +  Program     PGM01    Lib SYSTEM
       ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  0250   RD1. READ EMPLOYEES-VIEW BY NAME
  0260       STARTING FROM #NAME-START
  0270       THRU #NAME-END
  0280 *
  0290     IF LEAVE-DUE >= 20
  0300       PERFORM MARK-SPECIAL-EMPLOYEES
  0310     ELSE
  0320       RESET #MARK
  0330     END-IF
  0340 *
E 0350     DISPLAY NAME 3X DEPT 3X LEAVE-DUE 3X '>=20 #MARK
  0360 *
  0370   END-READ
  0380 *
  0390   IF *COUNTER (RD1.) = 0
  0400     REINPUT 'PLEASE TRY ANOTHER NAME'
  0410   END-IF
  0420 *
  0430 END-REPEAT
  0440 *
       ....+....1....+....2....+....3....+....4....+....5....+... S 49   L 25
NAT0305 Text string must begin and end on the same line.
```

You cannot compile an object before you have corrected the error. If there is a syntax error, you can only save the source code as a source object (see the following section). You can use the online help function for information on an error and advice on solving the problem.

To obtain help on error messages

1 Enter one of the following system commands:

```
HELP nnnn
```

or

```
? nnnn
```

where *nnnn* is the four-digit error number.

For example:

```
HELP NAT0305
```

2 Choose ENTER.

The **Natural System Message** screen appears with an explanation of the specified error.

For further information on online help, refer to ***Detailed Information on Error Messages***.

Saving and Cataloging Objects

You can save the source code currently contained in the source work area as a source object by using the system command `SAVE`. `SAVE` does *not* catalog (compile) source code and hence no syntax check is performed.

You can save the source code currently contained in the source work area as a source object *and* as a cataloged (compiled) object by using the system command `STOW`.

You can catalog the source code currently contained in the source work area and save it as a cataloged object only by using the system command `CATALOG`. `CATALOG` does *not* save the source code as a source object, which can be edited. See also ***Cataloging Multiple Objects***.

To save source code as a source object

1 At the editor command prompt, enter the following:

```
SAVE object-name
```

where *object-name* is the name of the source object you want to create. The name of the object must be unique and comply with the **object naming conventions** (see the relevant section).

For all syntax rules that apply to `SAVE`, see the *System Commands* documentation.

2 Choose ENTER.

The source code is stored as a source object under the specified name in the current library in the current system file.

To save source code as a source object and/or a cataloged object

1 At the editor command prompt, enter one of the following:

```
STOW object-name
```

or

```
CATALOG object-name
```

where `object-name` is the name of the source object and/or the cataloged object you want to create. The name of the object must be unique and comply with the **object naming conventions**.

For all syntax rules that apply to `STOW` and `CATALOG`, see the *System Commands* documentation.

2 Choose ENTER.

When using `STOW`, the source code is stored as a source object under the specified name in the current library in the current system file. Additionally, the generated object code is stored as a cataloged object in the same library and system file.

When using `CATALOG`, the source code is only stored as a cataloged object under the specified name in the current library in the current system file. The source code is *not* stored (or updated if the command is executed on an existing source object) as a source object in the system file. Source code is only stored or updated with `SAVE` or `STOW`.

If you want to find out whether an object has been saved as a source object and/or a cataloged object, see ***To display object directory information***.

Cataloging Multiple Objects

You can catalog and recatalog multiple source objects contained in the current library by using the system command `CATALL`.

To catalog multiple objects

1 Enter the following system command:

```
CATALL
```

2 Choose ENTER.

A **Catalog Objects in Library** screen similar to the example below appears where you can specify the objects to be processed, the commands to be executed and additional options such as the creation of an error report.

```
10:10:55              ***** NATURAL CATALL COMMAND *****             2009-05-20
User SAG                   - Catalog Objects in Library -        Library SAGTEST

Catalog Objects from .. *_______  (start value, range, input list)
                to .... ________  (end value)

                                       X Recatalog only existing modules
Select object types:                   _ Catalog all sources
X   Global data areas                Select function:
X   Parameter data areas                 Save
X   Local  data areas                  X Catalog
X   Copycodes                            Stow
X   Texts                                Check
X   External subroutines             Select options:
X   Subprograms                          Condition code in batch
X   Helproutines                       X Renumber source-code lines
X   Maps                                 Keep result list
X   Adapter                            X Processing information
X   Programs                           X Error report
X   Classes                               Extended error report
Command ===>








Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Menu  Exit  AddOp Sel.                                      Canc
```

For detailed information on the options provided on the screen, refer to *CATALL* in the *System Commands* documentation.

Related Topic:

- *Example of Compilation - Natural System Architecture* documentation

Displaying Object Directory Information

The directory of a Natural object contains general information on the object such as the object name, the name of the library where it resides, and the date when the source object was created or modified.

To display object directory information

1 Enter the following system command:

```
LIST DIR object-name
```

where object-name is the name of an existing object that is contained in the current library in the current system file.

For example:

```
LIST DIR PGMTEST
```

2 Choose ENTER.

A **List Directory** screen similar to the example of program PGMTEST below appears:

```
09:34:31               ***** NATURAL LIST COMMAND *****              2009-05-20
User SAG                      - List Directory -               Library SAGTEST

Directory of Program PGMTEST                  Saved on ... 2008-05-14 13:30:33
-------------------------------------------------------------------------------
Library .... SAGTEST    User-ID ...... SAG       Mode ....... Structured
TP-System .. COMPLETE   Terminal-ID .. 1     24
Op-System .. MVS/ESA    Transaction .. NAT42
NAT-Ver .... 4.2.4      Code page .... IBM01140
Source size .........................    1046 Bytes

Directory of Program PGMTEST                  Cataloged on 2006-05-23 16:36:12
-------------------------------------------------------------------------------
Library .... SAGTEST    User-ID ...... SAG       Mode ....... Structured
TP-System .. COMPLETE   Terminal-ID .. 1     16
Op-System .. MVS/ESA    Transaction .. NAT42
NAT-Ver .... 4.2.1      Code page .... IBM01140
Used GDA ...            Options ...... PCHECK DBSHORT PSIGNF GFID TQMARK
Size of global data ...       0 Bytes  Size in DATSIZE ......       720 Bytes
Size in buffer pool ...    3416 Bytes
Size of OPT-Code ......       0 Bytes
Initial OPT string ....
                                                              ENTER to continue
```

For detailed information on the **List Directory** screen, refer to *Displaying Object Directory Information* in *LIST* in the *System Commands* documentation.

Copying Objects

You can create new objects by either copying the source code contained in the source work area or using the copy function of a Natural utility such as SYSMAIN.

To copy source code from the source work area

1 Read in the source code you want to copy by entering the following system command:

```
READ object-name
```

where `object-name` is the name of the object that contains the source code you want to copy.

2 Choose ENTER.

The source code of the specified source object is read into the source work area.

3 Enter one of the following system commands:

```
SAVE object-name
```

or

```
STOW object-name
```

where `object-name` is the name of the object you want to create.

4 Choose ENTER.

The new object is saved as a source object (using `SAVE`) and as a cataloged object (using `STOW`) in the current library in the current system file.

To copy one or more objects using SYSMAIN

1 Invoke the **Main Menu** of the SYSMAIN utility as described in **Steps 1 through 4** of *To list all libraries using menu functions*.

2 In the **Object Code** field, enter an `A` (default setting) to select all types of object. For object types that are listed separately on the menu screen, enter another code such as `E` for error messages.

In the **Function Code** field, enter a `C` (for **Copy**).

3 Choose ENTER.

The **Copy Programming Objects** screen appears.

4 In the **Code** field, enter an `A` to select all types of object module: cataloged objects and source objects.

In the **Sel. List** (Selection List) field, replace `Y` (Yes) by `N` (No). `Y` is the default setting.

In the **Object Name** field, enter the name of the object you want to copy or specify a range of names. An asterisk (*) select all object names. Asterisk (*) is the default setting.

(For valid name ranges, see *Specifying a Range of Names* in the *SYSMAIN Utility* documentation.)

In the **Source Library** field, enter the ID of the library that contains the objects to be copied.

In the **Target Library** field, enter the ID of an existing or a new library to which you want to copy the objects.

Leave all other input fields unchanged.

5 Choose ENTER.

All source and cataloged objects are copied from the specified source library to the specified target library in the current system file and the following message appears: `Function completed successfully.`

Related Topic:

- *READ* - *System Commands* documentation

Printing Objects

You can print the source code of a source object by using the system command `LIST`.

You can also print a list of objects contained in a library as described in ***Printing a List of Objects***.

To print a source object

1 Choose one of the following methods:

- Select an object from a list by invoking the **LIST Objects in a Library** screen as described in **Steps 1 and 2** of *To list objects using LIST*.

  In the **Cmd** column, next to the object required, enter the following:

  ```
  PR
  ```

  Choose ENTER.

- Or:

Enter the following system command:

```
LIST object-name
```

where *object-name* is the name of the object to be printed.

Choose ENTER.

The source code of the specified object is displayed in read-only mode.

Choose PF2.

The **PRINT** window appears.

2 In the **Destination** field, enter a valid printer name (if required, ask your Natural administrator for a printer available in your current environment). If required, change the page size (the default setting is 60 lines).

3 Choose ENTER.

The **Printout Specification** screen appears where you can specify printer settings such as the amount of copies to be printed.

4 Choose ENTER.

The specified source objects is printed on the specified printer device.

Related Topics:

- *Printing Objects in a Library*
- *LIST - System Commands* documentation

Renaming Objects

You can rename either single objects by using the system command `RENAME` or multiple objects by using the Natural utility SYSMAIN.

As an alternative to `RENAME`, you can use the **Rename Object** function provided in the **Development Functions** menu described in *Natural Main Menu*.

To rename an object by using `RENAME`

1 Enter the following system command:

```
RENAME object-name
```

where *object-name* is the name of the object you want to rename.

2 Choose ENTER.

3 The **Rename Object** window appears where the name of the specified object is entered in the **Name** field.

4 In the **New Name** field, enter a new object name.

If required, in the **New Type** field, enter a new object type.

5 Choose ENTER.

The following message appears: `Object renamed successfully.`

To rename one or more objects using SYSMAIN

1 Invoke the **Main Menu** of the SYSMAIN utility as described in **Steps 1 through 4** of *To list all libraries using menu functions*.

2 In the **Object Code** field, enter an `A` (default setting) to select all types of object. For object types that are listed separately on the menu screen, enter another code such as `E` for error messages.

In the **Function Code** field, enter an `R` (for **Rename**).

3 Choose ENTER.

The **Rename Programming Objects** screen appears.

4 In the **Code** field, enter an `A` to select all types of object module: cataloged objects and source objects.

In the **Name** field, enter the name of the object you want to rename or specify a range of names (for example, `TEST*`on the following example screen:). An asterisk (*) select all object names. Asterisk (*) is the default setting.

(For valid name ranges, see *Specifying a Range of Names* in the *SYSMAIN Utility* documentation.)

If you only rename a single object: in the **New Name** field, enter a new name, and, in the **Sel. List** field, replace `Y` (Yes) by `N` (No).

In the **Source Library** field, enter the ID of the library that contains the objects to be renamed.

If required, in the **Target Library** field, enter the ID of an existing or a new library where you want to store the renamed object(s).

Leave all other input fields unchanged.

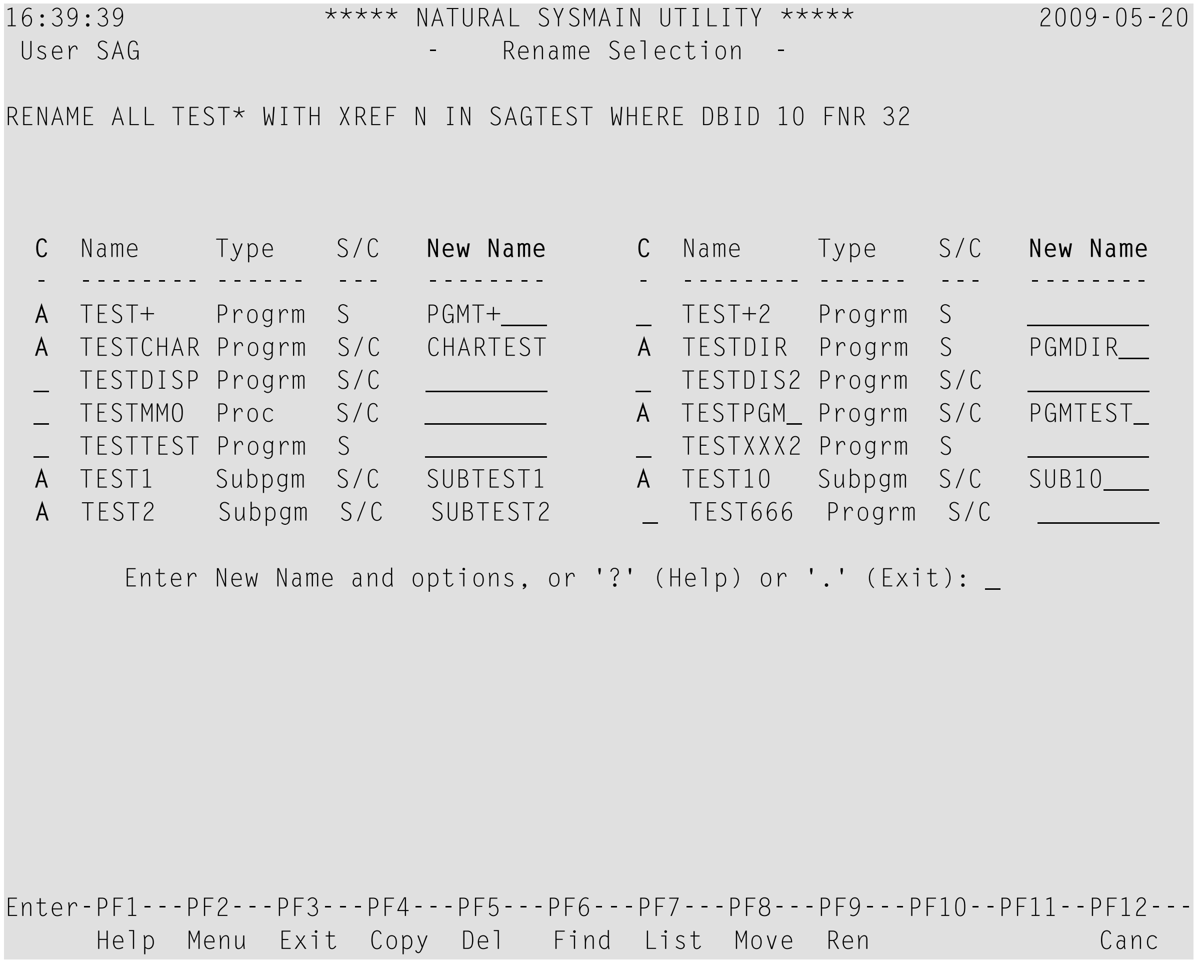
5 Choose ENTER.

A window appears, where you can enter a `Y` (Yes) to keep a copy of the object(s) to be renamed.

6 Choose ENTER.

In you specified a range of objects, a **Rename Selection** screen similar to the example below appears with a list of all objects that meet the specified selection criteria (on the example screen below: TEST*).

In the **C** column, next to the object(s) required, enter an A to rename both source object(s) and cataloged object(s). In the **New Name** column, enter a new name as shown below:



```
16:39:39            ***** NATURAL SYSMAIN UTILITY *****          2009-05-20
 User SAG                     -    Rename Selection  -

RENAME ALL TEST* WITH XREF N IN SAGTEST WHERE DBID 10 FNR 32



   C  Name     Type    S/C   New Name      C  Name     Type    S/C   New Name
   -  -------- ------  ---   --------      -  -------- ------  ---   --------
   A  TEST+    Progrm  S     PGMT+___      _  TEST+2   Progrm  S     ________
   A  TESTCHAR Progrm  S/C   CHARTEST      A  TESTDIR  Progrm  S     PGMDIR__
   _  TESTDISP Progrm  S/C   ________      _  TESTDIS2 Progrm  S/C   ________
   _  TESTMMO  Proc    S/C   ________      A  TESTPGM_ Progrm  S/C   PGMTEST_
   _  TESTTEST Progrm  S     ________      _  TESTXXX2 Progrm  S     ________
   A  TEST1    Subpgm  S/C   SUBTEST1      A  TEST10   Subpgm  S/C   SUB10___
   A  TEST2    Subpgm  S/C   SUBTEST2      _  TEST666  Progrm  S/C   ________

        Enter New Name and options, or '?' (Help) or '.' (Exit): _











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

7 Choose ENTER.

A **Message Text** column appears where a confirmation message is displayed next to each renamed object. Depending on whether you marked the option to keep a copy of the original object, the message reads either Renamed as or Copied as.

Moving Objects

You can move objects from one library into another by using a Natural utility such as SYSMAIN.

To move objects using SYSMAIN menu functions

1 Invoke the **Main Menu** of the SYSMAIN utility as described in **Steps 1 through 4** of *To list all libraries using menu functions*.

2 In the **Object Code** field, enter an `A` (default setting) to select all types of object. For object types that are listed separately on the menu screen, enter another code such as `E` for error messages.

   In the **Function Code** field, enter an `M` (for **Move**).

3 Choose ENTER.

   The **Move Programming Objects** screen appears.

4 In the **Code** field, enter an `A` to select all types of object module: source objects and cataloged objects.

   In the **Sel. List** (Selection List) field, replace `Y` (Yes) by `N` (No). `Y` is the default setting.

   In the **Object Name** field, enter the name of the object you want to move or specify a range of names. An asterisk (*) select all object names. Asterisk (*) is the default setting.

   (For valid name ranges, see *Specifying a Range of Names* in the *SYSMAIN Utility* documentation.)

   In the **Source Library** field, enter the ID of the library that contains the objects to be moved.

   In the **Target Library** field, enter the ID of an existing or a new library to which you want to move the objects.

   Leave all other input fields unchanged.

5 Choose ENTER.

   A confirmation window appears.

6 Choose ENTER to execute the move operation or enter a period (.) to cancel the operation.

   If the move operation has performed successfully, all source and cataloged objects were moved from the specified source library into the specified target library in the current system file and the following message appears: `Function completed successfully.`

Deleting Objects

You can delete objects by using either the system command `DELETE`, the system command `LIST` or a Natural utility such as SYSMAIN. For instruction on deleting objects by using `LIST` or SYSMAIN, see ***Deleting Objects in a Library***.

As an alternative to `DELETE`, you can use the **Delete Object** function provided in the **Development Functions** menu described in *Natural Main Menu*.

To delete single or multiple objects using `DELETE`

1 Enter one of the following system commands:

```
DELETE object-name
```

or

```
DELETE object-name*
```

or

```
DELETE *
```

where:

`object-name` is the name of the object to be deleted.

`object-name*` is a particular range of objects to be selected (for example, `TEST*` selects all objects that start with TEST).

Asterisk (*) selects all objects available in the current library in the current system file.

2 Choose ENTER.

- If you specified an individual object, the **DELETE** window appears.

  Type in the name of the object to confirm the delete operation.
- If you specified a range of objects, the **Delete Sources and Objects** screen appears.

  In the **M** column, next to the object(s) required, enter a `B` to delete both source object(s) and cataloged object(s).

  Choose ENTER.

  The **DELETE** window appears.

  Mark an item by typing in any character next to the option required:

Confirm each deletion invokes the **DELETE** window for the first object to be deleted. After you typed in the name of the object, choose ENTER to confirm the deletion and open the **DELETE** window for the next object to be deleted.

Delete without confirmation immediately executes the delete operation(s).

Exit (no deletion) cancels the delete operation(s).

3 Choose ENTER.

The **Delete Sources and Objects** screen appears where a message is displayed next to the object selected for deletion. The message indicates either that the object was `deleted` or that the delete operation was canceled (`not deleted`).

Related Topic:

- *DELETE - System Commands* documentation

Executing Programs

An object of the type program can be executed by using a system command. All other types of object are only executed or invoked when they are referenced in this program or in a subordinate object. See also *Multiple Levels of Invoked Objects* described in the *Programming Guide*.

You execute a program by using either the system command `RUN` or `EXECUTE`.

As an alternative to `EXECUTE`, you can use the **Execute Program** function provided in the **Development Functions** menu described in *Natural Main Menu*.

`RUN` executes the source code currently contained in the source work area or a cataloged object stored in a system file.

`EXECUTE` only executes cataloged objects. Unlike `RUN`, `EXECUTE` does not consider latest changes that may have been made to the corresponding source code in the source work area. These modifications are only considered after the source object has been updated and recompiled accordingly.

The execution of a cataloged object does not affect the source code currently contained in the source work area.

To execute a program using `RUN`

1 Enter one of the following system commands:

```
RUN
```

or

```
RUN program-name
```

where *program-name* is the name of a source object of the type program that is read into the source work area.

2 Choose ENTER.

If no syntax error is found, the source code contained in the source work area is compiled and executed.

To execute a program using `EXECUTE`

1 Enter the following system command:

```
EXECUTE program-name
```

where *program-name* is the name of a cataloged object of the type program.

The keyword `EXECUTE` is optional; it is sufficient to specify *program-name*.

2 Choose ENTER.

The program is executed.

Related Topics:

- *RUN - System Commands* documentation
- *EXECUTE - System Commands* documentation
- *Object Execution - Natural System Architecture* documentation
- *Search Sequence for Object Execution*
- *Example of Object Loading - Natural System Architecture* documentation

7 Natural Main Menu

The Natural **Main Menu** provides access to Natural development functions, environment settings, utilities and example libraries.

This section contains information on the functions and input options provided by the Natural **Main Menu** and its subordinate menus.

Invoking or Closing the Natural Main Menu

There are two methods of invoking or closing the Natural **Main Menu**:

- You can define a default setting by switching menu mode on or off. Menu mode causes the Natural **Main Menu** to be invoked automatically for the next session started.
- You can invoke or close the Natural **Main Menu** within a Natural session whenever required.

To switch menu mode on or off before session start

- At Natural startup, specify either of the following profile parameters:

  ```
  MENU=ON
  ```

  (activates menu mode)

  or

  ```
  MENU=OFF
  ```

  (deactivates menu mode)

  See also *MENU* in the *Parameter Reference* documentation.

To invoke or close the Natural Main Menu within a session

1. Enter either of the following system commands:

   ```
   MAINMENU
   ```

   (invokes the menu)

   or

   ```
   MAINMENU OFF
   ```

   (closes the menu)
2. Choose ENTER.

The Natural **Main Menu** looks similar to the example shown below:

```
10:20:23                    *****  NATURAL  *****                2009-05-20
User SAG                        - Main Menu -                Library TEST


                   Function

                _  Development Functions
                _  Development Environment Settings
                _  Maintenance and Transfer Utilities
                _  Debugging and Monitoring Utilities
                _  Example Libraries
                _  Other Products
                _  Help
                _  Exit Natural Session






Command ===>









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

At session start, Natural assigns you to a default library that is indicated on the screen. In the example screen above, the ID (name) of the library is shown in the **Library** field in the top right-hand corner of the screen. See also ***Default Library Assignment***.

For instructions on **performing a menu function**, refer to the relevant section in *Using Commands and Menu Functions*.

Each function listed in the Natural **Main Menu** invokes a menu of the corresponding name where you can select further functions:

Function in Natural Main Menu	Explanation of Functions in Corresponding Menu
Development Functions	Creates and maintains programs, maps, data areas and other components that make up a Natural application.
Development Environment Settings	Displays and modifies various settings that affect your Natural session.
Maintenance and Transfer Utilities	Invokes a Natural utility to create and maintain certain objects or transfer them to another environment.
Debugging and Monitoring Utilities	Invokes a Natural utility to monitor your Natural applications and locate errors in their processing flow.
Example Libraries	Selects libraries containing example programs and Application Programming Interfaces (APIs).
Other Products	Invokes other Software AG products.

Note: The position and color of the message line and PF-key lines in the Natural **Main Menu** and its subordinate menus can be changed with the user exit routine USR2003P described in the *Operations* documentation.

Development Functions

The functions listed in the **Development Functions** menu are those you will need most frequently when you develop an application with Natural. The functions apply to all Natural objects that are available in the library where you are currently logged on.

The table below contains information on the fields provided in the **Development Functions** menu:

Field	Explanation
User	The ID of the Natural user who logged in the current session.
Library	The library currently active. See also the section ***Using Natural Libraries***.
Mode	The programming mode: reporting or structured mode. See ***Programming Modes***.
Work area empty	Indicates that no source has been loaded into the source work area. If a source has already been loaded into the source work area, the **type** and the **name** of the object will be displayed instead, for example, `Program PROGX`.
Code	The code that corresponds to the function required, for example, **C** for **Create Object**. See also ***Performing a Menu Function***.

Field	Explanation
Type	The type of object such as `P` for program. For details, see *Object Types* in the *Programming Guide*. You can leave the **Type** field blank if you specify the name of a Natural object that already exists. If you want to change the object type, see also ***Setting the Object Type***.
Name	The name of the object. For an explanation of valid object names, see ***Object Naming Conventions***.
Command ===>	The command line. It is an input field in which you can enter a Natural command. For example: To edit an existing program named PROGX, you would enter the following system command: `EDIT PROGX` See also ***Using Commands and Menu Functions***.
PF (function keys)	PF keys (function keys) can be used as an alternative to using commands or menu functions. The PF-key lines at the bottom of the screen indicate which function is assigned to which key. See also ***Standard PF Keys***.

The table below contains information on the functions provided in the **Development Functions** menu. For most of the menu functions, there are equivalent Natural system commands. These alternative system commands are listed in the table and further explained in the relevant sections in the *System Commands* documentation.

Function	Code	Explanation
Create Object	`C`	Invokes a **Natural editor** where you can create a new object such as a program, map or data area. Specify the type and the **name** of the object to be created. You can enter a question mark (?) in the **Type** field, to select an object type from a list of all types available for this function. See also ***Creating and Editing Objects***.
Edit Object	`E`	Invokes a **Natural editor** and displays the source of the specified object in modify mode. Specify the **name** of an existing object to be edited. You can also invoke a selection list of objects: see ***Specifying Object Ranges***. You can enter a question mark (?) in the **Type** field, to select an object type from a list of all types available for this function.

Function	Code	Explanation
		See also *Creating and Editing Objects*. Equivalent system command: `EDIT`
Rename Object	`R`	Invokes the **Rename Objects** window where you change the name of the specified object and/or the object **type**. See also *Renaming Objects*. Equivalent system command: `RENAME`
Delete Object	`D`	Invokes the **Delete** window for the specified object. In the **Delete** window, confirm the deletion by entering the name of the object again in the relevant input field. You can also invoke a selection list of objects as described in *Specifying Object Ranges*. In this list, you can mark one or more objects for deletion. See also *Deleting Objects*. Equivalent system command: `DELETE`
Execute Program	`X`	Executes an object of the **type** program. Specify the name of the object to be executed. Other object types cannot be executed by themselves, but must be invoked from another object. See also *Executing Programs*. Equivalent system command: `EXECUTE`
List Object(s)	`L`	Displays the source code of the specified object. Specify the name of the object to be displayed. You can also invoke a selection list of objects: see *Specifying Object Ranges*. You can enter a question mark (?) in the **Type** field, to select an object type from a list of all types available for this function. See also *Listing Objects in a Library*. Equivalent system command: `LIST`
List Subroutines Used	`S`	Ascertains which objects use which external subroutines and classes. Equivalent system command: `ROUTINES`

This section covers the following topics:

- Programming Modes
- Natural Editors

- Specifying Object Ranges

Programming Modes

Natural offers two programming modes: reporting mode and structured mode. We recommend that you exclusively use structured mode, because it provides for more clearly structured applications. Therefore, all explanations and examples in the Natural tutorial *First Steps* and the *Editors* documentation refer to structured mode. Any peculiarities of reporting mode will not be taken into consideration.

For further information on programming modes, see the section *Natural Programming Modes* in the *Programming Guide*.

The **Mode** field in the top right-hand corner of the **Development Functions** menu indicates the programming mode currently in effect: structured or reporting.

To switch programming modes

1. In the upper right-hand corner of the **Development Functions** screen, in the **Mode** field, overwrite the first position with an `S` to switch on structured mode, or an `R` to switch on reporting mode.

   Or:

   Enter either of the following system commands:

   ```
   GLOBALS SM=ON
   ```

   (switches on structured mode)

   or

   ```
   GLOBALS SM=OFF
   ```

   (switches on reporting mode)

2. Choose ENTER.

   The contents of the **Mode** field has changed from `Reporting` to `Structured` or vice versa.

Related Topics:

- *Natural Programming Modes - Programming Guide*
- *GLOBALS - System Commands* documentation

Natural Editors

Depending on the type of object specified in the **Development Functions** menu, Natural invokes the appropriate editor: the program editor, the map editor or the data area editor. For further information on these editors, see the relevant sections in the *Editors* documentation.

Specifying Object Ranges

The functions **Edit Object**, **List Object(s)** and **Delete Object** provide the option to either specify the name of an individual object or a range of names. When you specify a range of names, a list of objects is displayed from which you can select one or more objects you wish to edit or list, or mark for deletion.

To list all objects

1. In the **Name** field, enter an asterisk:

   ```
   *
   ```

2. Choose ENTER.

   A list of all objects available in the current library is displayed.

To list objects using a start value

1. In the **Name** field, enter a start value followed by an asterisk (*).

   This option to enter a value followed by an asterisk is referred to as asterisk notation.

   For example:

   ```
   AB*
   ```

2. Choose ENTER.

   A list of all objects with names that start with AB (for example, AB, AB1, ABC, ABEZ) is displayed for the current library. The list does not include object names that start with AA1 or ACB, for example.

Note: The **List Object(s)** function provides further options to specify object name ranges as described for the equivalent system command `LIST`.

Development Environment Settings

The table below contains brief descriptions of the functions provided in the **Development Environment Settings** menu, and lists the Natural system commands that correspond to these functions. For further information on a system command, refer to the relevant section in the *System Commands* documentation.

Function	Explanation	Correspond. Command
Function-Key Settings	Assigns functions to PF keys to be used in your Natural session.	`KEY`
Compilation Settings	Sets options that affect the way in which Natural objects are compiled.	`COMPOPT`
Session Parameter Settings	Changes the settings of Natural session parameters. See also ***Configuring your Natural Environment*** and *Session Parameters* in the *Parameter Reference* documentation.	`GLOBALS`
Profile Parameter Settings	Changes the settings of Natural profile parameters. Profile parameters are described in the *Parameter Reference* documentation and in *Profile Parameter Usage* in the *Operations* documentation. The system command `SYSPARM` invokes a utility of the same name that is described in the *Utilities* documentation.	`SYSPARM`
Technical Session Information	Displays technical information on your Natural session such as the current user ID, library and operating system.	`TECH`
System File Information	Displays the current definitions of Natural system files. See also *Natural System Files* in the *Natural System Architecture* documentation.	`SYSPROF`
Product Installation Information	Displays a list of the products installed at your site and information on these products.	`SYSPROD`
Security Profile Information	Only available if Natural Security is installed. Displays the security profile currently in effect.	`PROFILE`

Maintenance and Transfer Utilities

The table below contains brief descriptions of the functions provided in the **Maintenance and Transfer Utilities** menu, and lists the Natural system commands that correspond to these functions. Each of these commands invokes a Natural utility that is described in the *Utilities* documentation.

Function	Explanation of Utility	Correspond. Command
Maintain Error Messages	Creates and maintains messages you wish to issue in your Natural applications.	`SYSERR`
Maintain DDMs	Creates and maintains data definition modules (DDMs).	`SYSDDM`
Maintain Command Processors	Creates and maintains the command processors you wish to use in your Natural applications.	`SYSNCP`
Maintain Remote Procedure Calls	Establishes and maintains remote procedure calls and provides the settings required to execute a Natural subprogram located on a remote server.	`SYSRPC`
Transfer Objects to Other Libraries	Transfers Natural objects between different libraries.	`SYSMAIN`
Transfer Objects to Other System Files	Unloads or loads Natural objects. You can use either the system command `SYSUNLD` to invoke the initial utility menu for unloading or loading objects, or the system command `NATUNLD` or `NATLOAD` to directly invoke the subordinate load or unload utility. NATUNLD utility: unloads Natural objects from a Natural system file to a work file. NATLOAD utility: loads Natural objects from a work file into a Natural system file. Note that the functionality of NATUNLD and NATLOAD is covered by the Natural Object Handler. We recommend that you use the Object Handler instead.	`SYSUNLD`
Transfer Objects to Other Platforms	Transfers Natural objects and Adabas FDTs from one hardware platform to another. Note that the functionality of SYSTRANS is covered by the Object Handler. We recommend that you use the Object Handler instead.	`SYSTRANS`
Transfer Objects to Other Systems	Invokes the Object Handler to process Natural objects and foreign objects for distribution in Natural environments.	`SYSOBJH`

Debugging and Monitoring Utilities

The table below contains brief descriptions of the functions provided in the **Debugging and Monitoring Utilities** menu, and lists the Natural system commands that correspond to these functions. Each of these commands invokes a Natural utility that is described in the *Utilities* documentation.

Function	Explanation of Utility	Correspond. Command
Debugging	Searches for errors in the processing flow of programs.	`TEST`
Logging of Database Calls	Logs database commands.	`TEST DBLOG`
Issuing Adabas Calls	Passes Adabas commands directly to the database.	`SYSADA`
Buffer Pool Maintenance	Monitors the Natural buffer pool and adjusts it to meet your requirements.	`SYSBPM`
Editor Buffer Pool Maintenance	Monitors the buffer pool of the Software AG Editor and adjusts it to meet your requirements.	`SYSEDT`
TP-Specific Monitoring	Monitors and controls TP-monitor-specific characteristics of Natural.	`SYSTP`
Data Collection and Tracing	Collects monitoring and accounting data about the processing flow of a Natural application.	`SYSRDC`
Error Information on Abnormal Termination	Provides information Software AG technical support requires for error diagnosis.	`DUMP`

Example Libraries

When you select **Example Libraries** from the Natural **Main Menu**, a list of libraries is displayed. These libraries contain example programs for demonstration purposes and Application Programming Interfaces (APIs) provided by Software AG:

Library	Contents
SYSEXPG	Example programs shown and referred to in the *Programming Guide*.
SYSEXRM	Example programs shown and referred to in the *Statements* documentation and the *System Variables* documentation.
SYSEXV	Example programs that illustrate new Natural features.
SYSEXT	APIs and example programs for using the APIs. See also the system command *SYSEXT* described in the *System Commands* documentation.
SYSEXTP	Example programs and APIs for specific functions that apply only under certain TP monitors.

Other Products

When you select **Other Products** from the Natural **Main Menu**, a list of Software AG add-on products appears. These products are installed at your site and can be accessed from this menu.

8 Print and Work Files

Print files and work files are logically defined for a Natural online or batch environment and can be physically assigned to a file or a printer by using a Natural parameter and/or control statements of the underlying operating system or TP monitor. Assignments can be changed for the current session during runtime.

Data is written to and read from a work file, or written to a print file by using the appropriate Natural statements.

Physical print file or work file assignments are independent of the objects maintained in the corresponding Natural environment. Therefore, when an assignment changes, you do not have to change statements in objects that reference print files or work file.

For further general information on print files and work files, refer to *Print Files - Work Files* in the *Natural System Architecture* documentation.

For detailed information on using print files and work files, refer to the documentation listed below.

Operations:

- *Datasets Used by Natural in z/OS Batch Mode*
- *Natural Datasets Used under a BS2000/OSD Batch Mode Session*
- *NATVSE Print and Work File Support for z/VSE Library Members*
- *Print File and Work File Support* (VM/CMS Environments)
- *Print and Work File Handling with External Datasets in a Server Environment - Natural as a Server under z/OS*
- *Output Reports and Work Files*

Statements:

- *Control of Work Files / PC Files*

- *CLOSE WORK FILE*
- *DEFINE WORK FILE*
- *READ WORK FILE*
- *WRITE WORK FILE*

Parameter Reference:

- *PRINT - Print File Assignments*
- *WORK - Work File Assignments*

9 Configuring your Natural Environment

Natural parameters manage the configuration of a Natural environment.

Natural parameters are used to standardize and automate development and production processes or adjust standard settings to the needs of individual users. A Natural parameter, for example, is used to set defaults for report creation, define the size of a report or define the size of storage area required such as the source area of an editor.

Most of the characteristics of a Natural environment are predefined by Software AG. The Natural administrator can configure different default environment settings valid for all Natural users. A user can adapt the settings to his needs by overriding default environment settings with a dynamic profile parameter or session parameter.

Using Profile Parameters

Profile parameters are specified statically or dynamically.

Static parameters are specified in the Natural parameter module NATPARM, during the installation of Natural. They are used as the default for each Natural session.

Dynamic parameters are specified at the startup of a Natural session. You can predefine a set of dynamic parameters with the Natural SYSPARM utility.

A predefined set of dynamic parameters is a Natural object of the type parameter profile.

At session start, you can specify one or more parameter profiles and one or more single profile parameters as indicated in ***To specify profile parameters at session start***.

A parameter specified at session start (with a parameter profile or without), overrides any value specified for the corresponding parameter in a standard parameter profile assigned to a session and/or in the Natural parameter module NATPARM; see also ***Parameterization Levels***.

To set profile parameters in NATPARM

- Follow the instructions provided in *Using a Natural Parameter Module* in the *Operations* documentation.

To create a parameter profile using SYSPARM

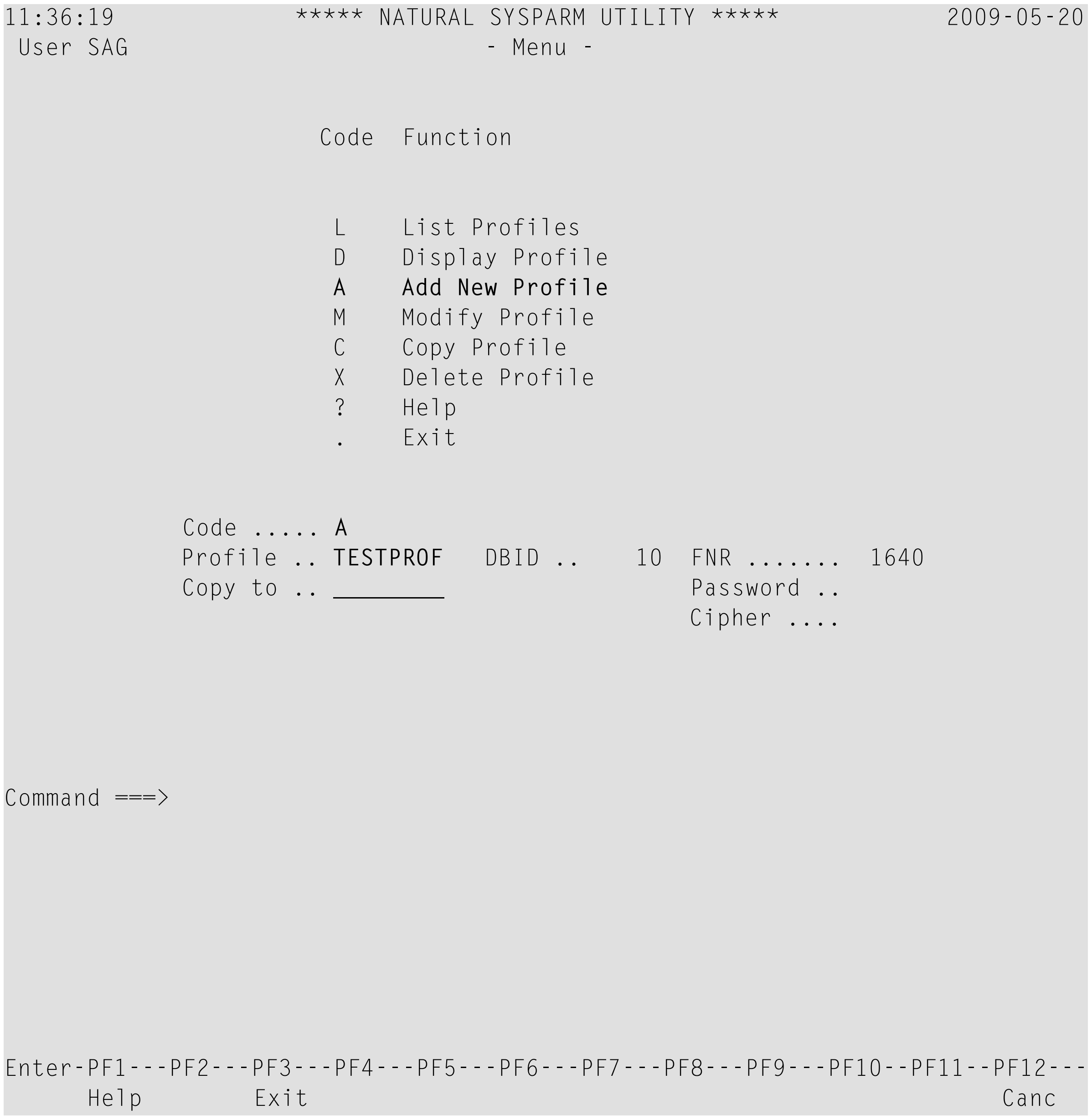
1 Invoke the SYSPARM utility by entering the following system command:

```
SYSPARM
```

2 Choose ENTER.

The **Menu** of the Natural SYSPARM utility appears.

3 In the **Code** field, enter an `A` (**Add New Profile**), and in the **Profile** field, enter the name of the parameter profile you want to create as shown in the example below:



```
11:36:19             ***** NATURAL SYSPARM UTILITY *****            2009-05-20
 User SAG                         - Menu -


                       Code  Function


                        L    List Profiles
                        D    Display Profile
                        A    Add New Profile
                        M    Modify Profile
                        C    Copy Profile
                        X    Delete Profile
                        ?    Help
                        .    Exit


             Code ..... A
             Profile .. TESTPROF   DBID ..    10  FNR .......  1640
             Copy to .. ________                  Password ..
                                                  Cipher ....







Command ===>








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

4 Choose ENTER.

The editing area of the SYSPARM utility appears.

5 Enter the profile parameter(s) and parameter value(s) required as shown in the example of a parameter profile below. For possible parameter settings, refer to the relevant descriptions of profile parameters in the section *Profile Parameters* in the *Parameter Reference* documentation.

```
16:57:37             ***** NATURAL SYSPARM UTILITY *****            2009-05-20
 >  FUSER=(10,32)                                                            <
 >  LS=250,PS=60                                                             <
 >  WORK=((6-8),AM=PC)                                                       <
 >  DB=(ADAV7,*)                                                             <
 >  PC=ON                                                                    <
 >  AUTO=ON                                                                  <
 >  MENU=OFF                                                                 <
 >                                                                           <
 >                                                                           <
 >                                                                           <
 >                                                                           <
Help with parameters .. _______                       (Profile name: TESTPROF)

Command ===>









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  Check Save                        Insrt Del   Copy  Canc
```

6 Choose PF4 (Check) to check and, if required, correct the syntax.

For the syntax that applies, refer to *Dynamic Assignment of Parameter Values* in the *Operations* documentation.

7 Choose PF5 (Save) to save the parameter profile as a source object under the specified name in the current system file (the default setting is FNAT).

To set profile parameters at session start

- Enter one or more parameter profiles (if available) and/or one or more profile parameters. Parameter profiles and profile parameters can be entered in any order. However, if a parameter is specified more than once (in the example below: `FUSER`), the value assigned to this parameter in the last specification is used (in the example below: `10,245`).

```
PROFILE=TESTPROF FUSER=(10,123) DSIZE=40 LS=250,PS=50 FUSER=(10,245)
```

Related Topics:

- *Profile Parameter Usage* - *Operations* documentation
- *Overview of Profile Parameters* - *Parameter Reference* documentation
- *Profile Parameters Grouped by Function* - *Operations* documentation

- *Assignment of Parameter Values - Operations* documentation
- *SYSPARM Utility* documentation

Using Session Parameters

Session parameters are specified within an active Natural session and/or within a Natural object. The main purpose of session parameters is to control the execution of Natural programs.

To set a session parameter

- Follow the instructions provided in *How to Set Session Parameters* in the *Parameter Reference* documentation.

To check or modify parameter settings for the current session

- Choose either of the following methods:

  1. Enter the following system command:

     ```
     GLOBALS
     ```

     Choose ENTER.

     A **Session Parameters** screen similar to the example below appears which displays the current settings of session parameters:

```
10:28:03             ***** NATURAL GLOBALS COMMAND *****         2009-05-20
                              - Session Parameters -



 (CC) Cond.Prog.Execution ....... OFF  (LT) Limit ............ __99999999

 (CF) Term.Control Character ....   %  (MT) Max.CPU Time ....... _______0

 (CPCVERR) CP Conversion Error ..  ON  (NC) Nat.Sys.Commands ........ OFF

 (DC) Dec. Character .............. .  (OPF) Overw.Prot.Fields ......  ON

 (DFOUT) Date Format Output ....... S  (PD) Page Dataset ............ _50

 (DFSTACK) Date Format Stack Cmd .. S  (PM) Print Mode ............... RP

 (DFTITLE) Date Format Title ...... S  (PS) Page Size ............... _31
```

```
 (DO) Data to Display Order ....... L  (REINP) Reinput on Error .....  ON

 (DU) Dump Generation .........   OFF  (SA) Sound Alarm ............. OFF

 (EJ) Page Eject ................  ON  (SF) Spacing Factor ........... _1

 (FS) Default Format ............ OFF  (SL) Source Line Length ...... _72

 (FCDP) FC on Dyn.Prot.Fields ...  ON  (SM) Structured Mode ......... OFF

 (IA) Input Assign ................ =  (THSEPCH) Thousands Separator .. ,

 (ID) Input Delimiter ............. ,  (TS) Translate Sys.Prog ...... OFF

 (IM) Input Mode .................. F  (WH) Wait on Hold ............ OFF

 (LE) Limit Error ............... OFF  (ZD) Zero Division ...........  ON

 (LS) Line Size ................. _80  (ZP) Zero Printing ...........  ON


Command ===>
















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

The letters in parentheses indicate the session parameter; see also *List of Parameters* in *GLOBALS* in the *System Commands* documentation.

You can change the setting of a parameter by replacing the default value entered next to the parameter required. For valid input values, refer to the relevant description of the session parameter in the section *Session Parameters* in the *Parameter Reference* documentation.

For example:

To change the setting of the date format parameter `DFOUT`, overwrite the value next to the input field of **(DFOUT) Date Format Output** (in the example above `S`) by an `I`.

Choose PF5 (Updat) to save the modification.

The format of date variables changes from *`yy-mm-dd`* (for example: `2009-06-16`) to *`yyymmdd`* (for example: `20090616`).

2. Enter the following system command:

```
GLOBALS parameter=value
```

where:

`parameter` is the session parameter (in the example below: `DFOUT`).

`value` is a valid value for this session parameter (in the example below: `I`).

You can specify multiple parameters and values.

For example:

```
GLOBALS DFOUT=I PS=60
```

Choose ENTER.

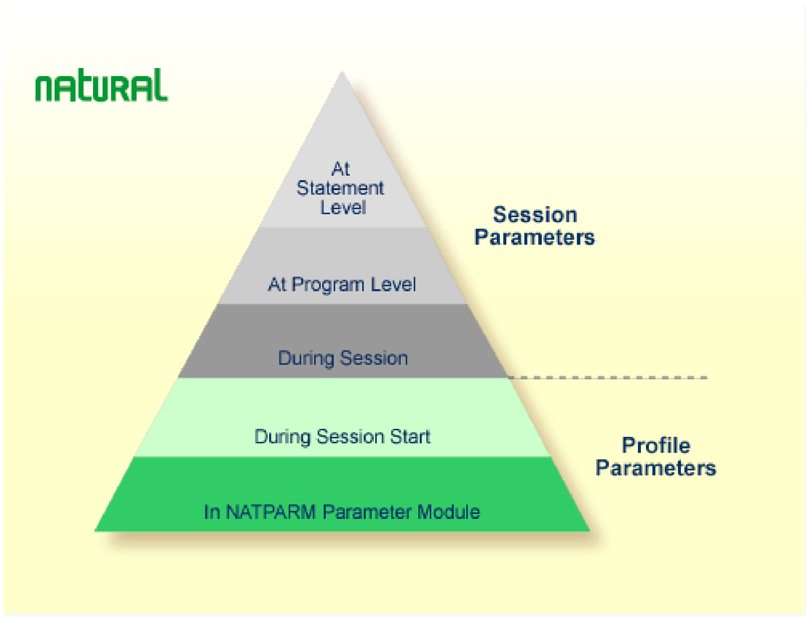
Related Topics:

- *Introduction to Session Parameters - Parameter Reference* documentation
- *Overview of Session Parameters - Parameter Reference* documentation
- *GLOBALS - System Commands* documentation

Parameterization Levels

There is a hierarchical structure of the levels at which Natural parameters can be set. A parameter value set on a higher level overrides the value defined on a lower level. For example, when you specify a parameter dynamically, the new parameter value overrides the static specification as set for the corresponding parameter in the Natural parameter module NATPARM.

The diagram below illustrates when a parameter can be set and the Natural parameter hierarchy from the lowest level at the base of the pyramid to the highest level at the apex:



Related Topic:

- *Natural Parameter Hierarchy* - *Operations* documentation

10 Rules and Naming Conventions

This section describes Natural-specific rules and naming conventions.

Standard Character Settings

The tables in this section list the standard character settings that apply to Natural. These settings comply with the IBM code page 1140 (US English), which determines the assignment of code points to graphic characters. Code points are represented by hexadecimal values. For further details on IBM code pages, refer to the corresponding IBM documentation.

The display control unit or emulator determines what character to display for a hexadecimal value. The use of a code page other than 1140 can result in the following:

- A character provides a hexadecimal value that is not accepted by Natural.
- A character that differs from the standard character setting provides the same hexadecimal value defined for the standard character.

Examples:

The character @ (commercial at) used with the German code page 1141 does not provide the hexadecimal value 7C.

The character £ (pound sign) used with the UK English code page 1146 provides the same hexadecimal value as the $ (dollar sign) character used with the US English code page 1140.

The character § (section sign) used with the German code page 1141 provides the same hexadecimal value as the character @ used with the US English code page 1140.

Object Naming Conventions

This section describes the naming conventions that apply when saving and/or cataloging a Natural object in a Natural system file.

The name of a Natural object can be 1 to 8 characters (listed in the following table) where the *first* character must be one of the following:

- an upper-case alphabetical character
- a number sign (#)
- a plus sign (+)

If the first character is a number sign (#) or a plus sign (+), the name must consist of at least one additional character.

Exception:

The name of a Natural DDM can be 1 to 32 characters (listed in the following table) where the *first* character must be an upper-case alphabetical character.

The name of a Natural object can consist of the following characters:

Character	EBCDIC Hex Value	ISO Character Name	Remark
A - Z	C1-C9 D1-D9 E2-E9	Latin capital letter A - Z	Upper-case alphabetical character
0 - 9	F0-F9	Digit zero - digit nine	Numeric character
-	60	Hyphen-minus	Hyphen
_	6D	Low line	Underscore
/	61	Solidus	Slash
@	7C	Commercial at	
$	5B	Dollar sign	
&	50	Ampersand	Only allowed in language codes See also *Defining the Language of a Natural Object* in the *Programming Guide*.
#	7B	Number sign	Hash sign
+	4E	Plus sign	Only allowed as the first character

Library Naming Conventions

This section describes the naming conventions that apply to a Natural library.

The name (ID) of a library can be 1 to 8 characters and must *not* start with "SYS". The prefix "SYS" is reserved for Natural system libraries.

A library name must start with an upper-case alphabetical character. Any other alphabetical character must also be upper case. A library name must *not* contain any blank characters.

A library name can consist of the following characters:

Character	EBCDIC Hex Value	ISO Character Name	Remark
A - Z	C1-C9 D1-D9 E2-E9	Latin capital letter A - Z	Upper-case alphabetical character
0 - 9	F0-F9	Digit zero - digit nine	Numeric character
-	60	Hyphen-minus	Hyphen
_	6D	Low line	Underscore Not allowed as the last character

Naming Conventions for User-Defined Variables

This section describes the naming conventions that apply to a user-defined variable:

- Length of Variable Names
- Limitations of Variable Names
- Characters Allowed in Variable Names
- First Character of Variable Names
- Case of Characters in Variable Names

For further information on user-defined variables, refer to the section *User-Defined Variables* in the *Programming Guide*.

Length of Variable Names

The name of a user-defined variable can be 1 to 32 characters long.

You can use variable names of over 32 characters (for example, in complex applications where longer meaningful variable names enhance the readability of programs); however, only the first 32 characters are significant and must therefore be unique, the remaining characters will be ignored by Natural.

Limitations of Variable Names

The name of a user-defined variable must *not* be a Natural reserved keyword.

Within one Natural program, you must *not* use the same name for a user-defined variable and a database field, because this might lead to referencing errors (see *Qualifying Data Structures* in the *Programming Guide*).

Characters Allowed in Variable Names

The name of a user-defined variable can consist of the following characters:

Character	EBCDIC Hex Value	ISO Character Name	Remark
A - Z	C1-C9 D1-D9 E2-E9 81-89 91-99 A2-A9	Latin capital and/or small letter A - Z	Upper-case and/or lower-case alphabetical character Lower-case *not* allowed as the first character
0 - 9	F0-F9	Digit zero - digit nine	Numeric character
-	60	Hyphen-minus	Hyphen
_	6D	Low line	Underscore
/	61	Solidus	Slash
@	7C	Commercial at	
$	5B	Dollar sign	
&	50	Ampersand	
#	7B	Number sign	Hash sign
+	4E	Plus sign	Only allowed as the first character

First Character of Variable Names

The first character of the name must be one of the following:

Character	EBCDIC Hex Value	ISO Character Name	Remark
A - Z	C1-C9 D1-D9 E2-E9	Latin capital letter A - Z	Upper-case alphabetical character
&	50	Ampersand	
#	7B	Number sign	Hash sign
+	4E	Plus sign	

If the first character is a number sign (#), a plus sign (+) or an ampersand (&), the name must consist of at least one additional character.

Variables in a global data area (GDA) with a plus sign (+) as the first character must be defined at Level 1; see also *Global Data Area* in the *Programming Guide*. Other levels are only used in a redefinition.

A plus sign (+) as the first character of a name is only allowed for application-independent variables (AIVs) and variables in a global data area (GDA).

Names of AIVs must begin with a plus sign (+); see also *Defining Application-Independent Variables* in the description of the `DEFINE DATA` statement in the *Statements* documentation.

An ampersand (&) as the first character of a name is used in conjunction with dynamic source program modification (see the `RUN` statement in the *Statements* documentation), and as a dynamically replaceable character when defining processing rules; see the relevant description in the *Map Editor* documentation.

Case of Characters in Variable Names

Lower-case characters can only be entered as the second and subsequent characters of a variable name.

Lower-case characters entered as part of a variable name are internally converted to upper case if the `LOWSRCE` option of the `COMPOPT` system command (see also the *System Commands* documentation) is set to `ON`.

Lower-case characters are not translated to upper case and are therefore interpreted as being different from the respective upper-case characters, if

- the `LOWSRCE` option of the `COMPOPT` system command is set to `OFF` (the default value) and
- input in the editor is not translated to upper case (translation to upper case in the editor is controlled by editor profile options and by options depending on the operating system).

For example, this will cause the names `#FIELD` and `#field` to be interpreted as two different field names.

Note: For compatibility reasons, you should not use this feature if you plan to port Natural applications developed on mainframe computers to Windows, UNIX or OpenVMS. If you use lower-case characters as part of the variable name, it is highly recommended that variable names are unique regardless of their case.

Index