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*

This section covers the following topics:

- Introducing Natural Objects
- Using Natural Editors or Utilities
- Selecting and Displaying Objects
- Creating and Editing Objects
- Checking and Testing Objects
- Saving and Cataloging Objects
- Displaying Object Directory Information
- Copying Objects
- Printing Objects
- Renaming Objects
- Moving Objects

- Deleting Objects
- Executing Programs

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
- Setting the Programming Mode
- Using the Natural Programming Language
- Creating Source Code
- Editing a Source Object
- Setting the Object Type

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