

# Using Objects and Shared Resources

This document describes the use of Natural objects and also the use of Shared Resources (Non-Natural objects). The following topics are covered below:

- What is a Natural Object?
- Object Types
- Object Visualisation
- Object Naming Conventions
- Object Editors
- Object Commands
- Object Operations
- Object Retrieval
- Shared Resources

See also:

Library Limit (for max. number of objects in a library)

---

## What is a Natural Object?

An object in Natural terms is a programming module which is used in creating a Natural application. It is always associated with a specific library. Several object types exist depending on the Natural application task (e.g. a component-based application or an application providing a graphical user interface).

## Object Types

Natural provides the following object types:

- **Data area**  
A module containing the descriptions of data to be used by a Natural program (which is a Natural object for itself). It usually contains a declaration of user-defined variables and constants, as well as referenced database fields in the form of data definition modules. Data areas can be global (capable of being shared by two or more Natural programs), local (contained within programs), parameter data areas, or object data areas.
- **Data definition module (DDM)**  
A logical grouping of database fields and field descriptions which makes it possible to access database information from within a program. The data definition module must be referenced in a data area serving the program.
- **Program**  
A set of instructions or actions that are executed procedurally.
- **Subprogram**  
A Natural object that is called by another Natural object. A subprogram can receive parameter values from the Natural object which calls it. Passed parameters must be included in a parameter data area.
- **Subroutine**  
A Natural object that is called by another Natural object. A subroutine can receive parameter values from the calling program. A subroutine has automatic access to the same global data area as the Natural object which calls it.
- **Map**  
A layout for the information provided on screens referenced by another Natural object (for example, in a program) for data input and output. In a map, the end user typically enters the information that is necessary for processing a Natural object.
- **Helproutine**

A text element that can be assigned to a map or field in order to provide users with context-sensitive help for an application.

- **Copycode**

A portion of Natural source code which is automatically included in an external program when this program is compiled. Copycode promotes modularization in applications by making it possible for many different programs to use the same module of source code.

- **Text**

Text can be added and modified with the program editor, and stored in a Natural library using the SAVE command to store it. The naming conventions for Natural object types apply.

- **Dialog**

Dialogs are used in conjunction with event-driven programming when creating Natural applications for graphical user interfaces (GUIs). For further information, see Event-Driven Programming Techniques.

- **Class**

(for further information, see NaturalX documentation and the Class Builder).

- **Resource**

Resources are only available with Natural under Windows.

## Changing an Object's Type

You can change the type of any object except a DDM, map, or dialog. For the remaining objects, the following restrictions apply:

- A data area can be converted only to another type of data area: local, global, or parameter.
- A programming object can be converted only to another type of programming object: program, subprogram, subroutine, help routine, copycode, text.



### To change an object's type

1. Open the object and then, from the **Object** menu, choose **Object Type**.  
A cascading menu is displayed listing all possible object types for selection.
2. Choose the type of object to convert to.  
The new type is displayed in the title bar. The object must be saved to make the conversion complete.

## Object Visualisation

The Natural objects displayed can consist of the Natural source, the Natural generated program or both. The difference is reflected by a bitmap. The green ball always indicates that a generated program is available, without the green ball it is indicated that only the source of the displayed object is available and finally a green ball as a non-greyed bitmap indicates that both a source and a generated program is available.

## Object Naming Conventions

### Length

The name of a Natural object can be 1 to 8 characters long.

### Structure

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

Character	Explanation
A - Z	upper case alphabetical characters
0 - 9	numeric characters
-	hyphen
@	commercial "at" sign
_	underline
/	slash
\$	dollar sign
&	ampersand (only as language code character)
#	hash/number sign
+	plus sign (only allowed as first character)

### Additional Conditions

- The **first** character of the name must be one of the following:
  - an upper-case alphabetical character (A - Z)
  - a hash/number sign (#)
  - a plus sign (+)
- If the first character is a "#" or "+", the name must contain at least one additional character.

## Object Editors

- Types of Object Editors
- Invoking an Object Editor

### Types of Object Editors

Natural provides the following object editors:

- **Data Area Editor**  
Used to create and maintain global data areas, local data areas, parameter data areas, and object data areas. This editor has a column format that is designed for defining the user-defined variables and database fields used in Natural programs, subprograms, subroutines and dialogs.
- **DDM Editor**  
Used to create and maintain data definition modules (DDMs).
- **Program Editor**  
Used to create and maintain Natural programs, subprograms, subroutines, help routines, copycode, text and classes (see also Class Builder).
- **Map Editor**  
Used to create and maintain maps. Extended field editing features make it possible to assign special properties to fields. Processing rules can be attached to fields in the map.
- **Dialog Editor**  
Used to create event-driven applications composed of dialogs.
- **Class Builder**  
Used to create and maintain classes.

### Invoking an Object Editor

#### **To invoke an object editor from the tree view**

- Double-click the object in the tree view.  
Or click on the corresponding icon in the status bar.

#### **To invoke an object editor from the command line**

- Issue the system command EDIT.  
If you select an existing object for editing, the appropriate editor is invoked automatically.

## Object Commands

This section describes all of the System Commands that you can perform on Natural objects.

<b>Command</b>	<b>Purpose</b>
<b>EDIT</b>	Edit the source form of an object.
<b>CLEAR</b>	Close the currently active object and open a new editor window that has no content and no name. If this object has been modified since the last save, you are prompted to save any changes.
<b>CHECK</b>	Check the source code of an object for syntax errors. Syntax checking is also performed as part of the RUN and STOW commands.
<b>CATALOG</b>	Cataloging an object compiles the source program in the active editor window and stores the resulting object module. For a full description, see the CATALOG command.
<b>UNCATALOG</b>	Delete one or more generated programs (GPs). An object can only be uncataloged if it has already been cataloged.
<b>SAVE</b>	Save the <b>source form</b> of the Natural object currently in the work area of the editor. Syntax is not checked. A saved program can be RUN, but not executed (see EXECUTE command below).
<b>STOW</b>	Save the <b>source form</b> of an object, compile the object and store the resulting generated program (GP) as well as the source. The object is syntax-checked during the compilation process.
<b>SCRATCH</b>	Delete the <b>source and object form</b> of an object. A list of all objects stored in the current library will be displayed; on the list you can then mark the object(s) to be deleted.
<b>RUN</b>	Compile and execute a source program.
<b>EXECUTE</b>	Execute a program that has been compiled and stored in object form.
<b>DEBUG</b>	Invoke the Natural debugging facility for a cataloged program or dialog. For further information, refer to the Natural Debugging documentation.

## Object Operations

The Object Operations such as copy, move, rename, delete, import or export can be applied inside the Library Workspace and inside any open Listview. In contrast to the library workspace where only one object can be selected at a time, multiple objects of a listview can be selected.

It is possible to use these operations in both the Local and Remote environment. The operations can also be used across environments.

The following topics are covered below:

- Creating an Object
- Copying or Moving Objects - Rules
  - Valid Source Nodes
  - Valid Target Nodes
- Copying an Object
- Deleting Objects
- Exporting Objects
- Importing Objects
- Moving Objects
- Listing Objects
- Printing Objects
- Renaming an Object
- Object Retrieval
- Saving an Object
- Stowing an Object

### Creating an Object

#### To create an object using a context menu

- Select a library node, open the context menu, and select **New** with the corresponding object type. Or select a group node in the logical view (e.g. "Programs") and choose **New** from the context menu. The editor object of the currently selected object type is opened.

**Note:**

New classes can only be created in the Logical or Flat View. For more information, refer to the Class Builder.

#### To create an object using the menu bar

- From the **Object** menu, choose **New**.  
Except for classes, a specific node needs not to be selected.  
For classes, a library node or the "Classes" group node has to be selected.

#### To create a program editor object using accelerator keys

- Use **CTRL-N**.

### Copying or Moving Objects - Rules

#### Using Drag and Drop / Cut, Copy and Paste

With these operations, Natural Studio provides a powerful technique for the operations Copy and Move. It is possible to use these operations for nearly all of the available nodes in the library workspace and the various list views.

For example, you can copy all programs of a library by applying the operation on the "Programs" node in the "library workspace"

logical view or you can move all generated programs of a library by applying the operation on the "Gp" node in the "library workspace" file view.

Two important rules have to be taken into account when these operations are used.

- **The target node of a copy/move operation will only accept the objects of the selected source node when actually ALL objects can be copied/moved to the target node.**

Imagine a library is opened in a flat list view and the "Copy" operation is applied on a couple of selected objects. The "Paste" operation on the "Src" Node in the file view is not allowed (indicated by the greyed "Paste") when the object list contains any Natural generated program since the target only accepts Natural source files.

- **A group node can only be copied to a library, it cannot be copied into another group node.**

For example, it is not possible to copy the "Programs" node of a library into the "Programs" node of another library, but it is possible to copy the "Programs" node into the "Library" node itself.

## Valid Source Nodes

Regarding the previously described rules, the following nodes can be selected to act as a source for Copy or Move operations:

- All group nodes in the logical view ("Programs", "Subprograms", ...).
- All subdirectory nodes in the file view ("Gp", "Src").
- All objects in any view.
- Libraries.

## Valid Target Nodes

Regarding the previously described rules the following nodes can be selected to act as a target for Copy or Move operations:

- All group nodes in the logical view ("Programs", "Subprograms", ...).
- All subdirectory nodes in the file view ("Gp", "Src").
- All objects in any view.
- The library nodes in any view.

## Copying an Object

### ▶ To copy an object using the context menu

1. Select a source node.
2. Open the context menu and choose **Copy**.
3. Select a target node, open the context menu and choose **Paste**.

### ▶ To copy an object using accelerator keys

1. Select a source node and press **CTRL-C**.
2. Then select a target node and press **CTRL-V**.

### ▶ To copy an object using left mouse button drag & drop

1. Select the source node with the left mouse button and drag it to the target node.
2. Before releasing the mouse button, press **CTRL**.
3. Apply a drop of the source node by releasing the left mouse button.

### ▶ To copy an object using right mouse button drag & drop

1. Select the source node with the right mouse button and drag it to the target node.  
After the right mouse button has been released, a context menu pops up.
2. Choose **Copy**.

## Deleting Objects

It is, for example, possible to delete all Natural objects of type Copycode of a library by deleting the "Copycodes" node of a library in the "library workspace" Logical View or delete all generated programs by deleting the "Gp" node of a library in the "library workspace" File View.

### ▶ To delete objects

- Select the object and press **DEL**.  
Or choose **Delete** from the context menu.  
Or choose **Delete** from the **Object** menu.

## Exporting Objects

Applying the copy and move operations described in the section Copying or Moving Objects, it is also possible to export Natural objects from the Natural environment to the Windows Explorer using Drag & Drop or Cut, Copy and Paste.

This operation can be done for any node except the system file nodes in general ("user libraries", "system libraries").

- When the "Src" or "Gp" node of a library in the "library workspace" File View is exported, the whole directory is copied/moved.
- When a logical group node (e.g. the "Programs" node) of a library in the "library workspace" Logical View is exported, all the individual files of the selected type ("Program"), that means, all files with the extension ".NGP" (Natural Generated Program) and ".NSP" (Natural Source Program) will be copied/moved.
- When a Natural object node (e.g. an object of type Dialog named "MYDLG" ) of a library in the "library workspace" Flat View is exported, the corresponding files will be copied or moved (MYDLG.NS3 if the source is available, MYDLG.NG3 if the corresponding generated program is available).

## Importing Objects

Applying the described copy or move operations in the section Copying or Moving Objects, it is also possible to import files from the Windows Explorer to the Natural environment using Drag & Drop or Cut, Copy and Paste.

- If a complete directory is being imported, only objects with a file extension valid to Natural (e.g ".NSP", "NG3...") will be accepted, subdirectories will be ignored.
- If multiple directories are being imported, the same rule for importing a single directory applies.
- If multiple files are being imported, all files with an invalid file extension are ignored.

### Note:

Before the import is started, it is important to set the proper mode (structured mode or reporting mode) in which the Natural objects to be imported were developed, otherwise some of them might not be compiled. This can be done using the command `GlobalS SM=ON/OFF`.

If an object is imported and the object name is unknown to Natural and exists in the library, a container name will be generated with the object name identical plus a running index.

## Moving Object

### ▶ To move an object using the context menu

1. Select a source node and, from the context menu, choose **Cut**.
2. Then select a target node and, from the context menu, choose **Paste**.

### ▶ To move an object using accelerator keys

1. Select a source node and press **CTRL-X**.
2. Then select a target node and press **CTRL-V**.

### ▶ To move an object using left mouse button drag & drop

1. Select the source node with the left mouse button and drag it to the target node.
2. Apply a drop by releasing the left mouse button.

### ▶ To move an object using right mouse button drag & drop

1. Select the source node with the right mouse button, drag it to the target node and release the right mouse button.
2. From the resulting context menu, choose **Move Here** to accomplish the drop.  
After the operation is complete, the source node is deleted from the environment.

## Listing Objects

The LIST command is used to display the source code of objects. When an object is displayed using the LIST command, its content can be copied but not modified.

### To list an object

- Select the object and choose the List command from the context menu.  
Or click the List button in the object tool bar.

## Printing Objects

You can print the source listing of an object. A dialog box is displayed in which you can select the number of copies you want and modify printer defaults. To modify your printer defaults, choose **Print Setup**.

### To print an object

- Select the object and, from the context menu, choose **Print**.  
Or click the **Print** button in the **Object** tool bar.  
Or choose **Print** from the **Object** menu.

### Printing an Open Object

#### To print an object that is open on your desktop and active

1. Click the **Print** tool bar button.
2. In the "Copies" text box, enter the number of copies you want to print.

In the "Properties" dialog box, you can specify various page setup and advanced printer settings such as those applicable on the Windows Explorer. If you mark the Print to File option, the document is "printed" to a file whose location will be the current directory, unless you specify a specific directory.

## Renaming an Object

Renaming of objects can be accomplished by in-place-editing. Only one object can be renamed at a time. If several nodes in a list view are selected and a **Rename** is applied from the context menu, the operation is performed for the node currently having the focus.

### To start in-place-editing

1. Select the node.
2. Press the right mouse button and, from the resulting context menu, choose **Rename**.  
Or press **F2**.  
Or click on the selected node with the left mouse button.  
Or press **ESC** to abort the rename process.

### To rename an object from the Object menu

1. Select the object and, from the menu bars's **Object** submenu, choose **Rename**.
2. In the object's name field, enter the new name.
3. Press **ENTER** to finish the in-place-editing process.  
Or click with any mouse button on a different position.  
Or press **ESC** to abort the rename process.

## Object Retrieval

With the "Find Objects" dialog, it is possible to find Natural objects and the specified containing text. It can be applied on any node in Natural Studio.

### ▶ To start the "Find Objects" dialog

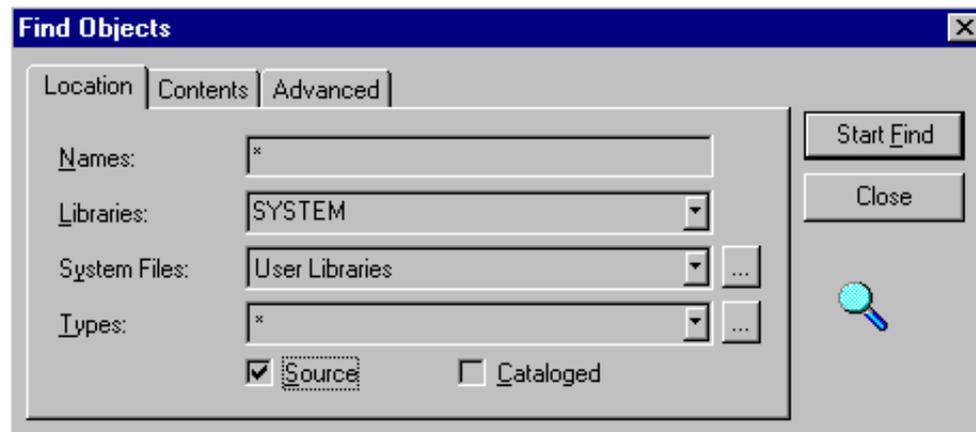
- Open menu **Library - Find Objects** in the Menu bar.  
Or open a context menu for any node in the library workspace.  
Or open a context menu for any node in an active list view

The "Find Objects" dialog comprises the "Location" sheet.

### "Location" Sheet

With the "Location" sheet, the following settings can be applied to find Natural objects:

<b>Names</b>	Names of the objects to be found. It is possible to specify multiple names separated with a semicolon. Additionally wildcards can be used.
<b>Libraries</b>	The libraries in which to search for objects; it is possible to specify multiple names separated with a semicolon.
<b>System File</b>	The systemfile to be used for the search.
<b>Types</b>	The types of Natural objects to be included in the search; open the "Types" dialog to select the types.
<b>Source</b>	If this box is checked, a search only for Natural sources is to be performed.
<b>Cataloged</b>	If this box is checked, a search only for Natural generated programs is to be performed.



In the above "Location" sheet, a search for any object is started in the "system" library of the "user libraries" system file. Only Source objects are to be included in the search.

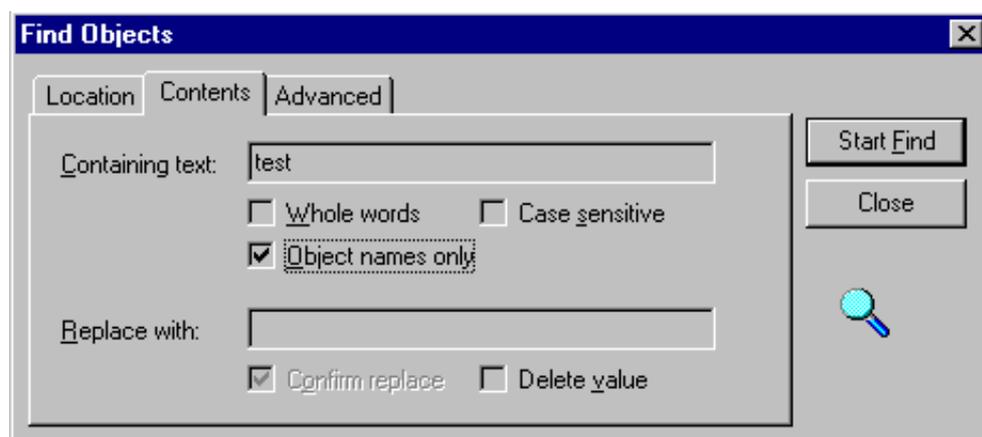
#### Note:

If both the "Source" **and** the "Cataloged" check boxes are checked, both the source and the generated program of the object must exist in order for the object to be found.

### "Contents" Sheet

With the "Contents" sheet, it is possible to scan the objects requested in the "Location" sheet for text including an optional replacement of text.

<b>Containing text</b>	The text to be searched for.
<b>Whole words</b>	If this box is checked, only whole words are taken into account.
<b>Case sensitive</b>	If this box is checked, case sensitivity of the text is taken into account.
<b>Object names only</b>	If this box is checked the object where the text is found will only be displayed once in the result list.
<b>Replace with</b>	The text to replace the found text with.
<b>Delete value</b>	If this box is checked, the found text will be deleted. This button is disabled when a "replace" text is specified.
<b>Confirm replace</b>	If this box is checked, a confirmation dialog is displayed for replacing the text.

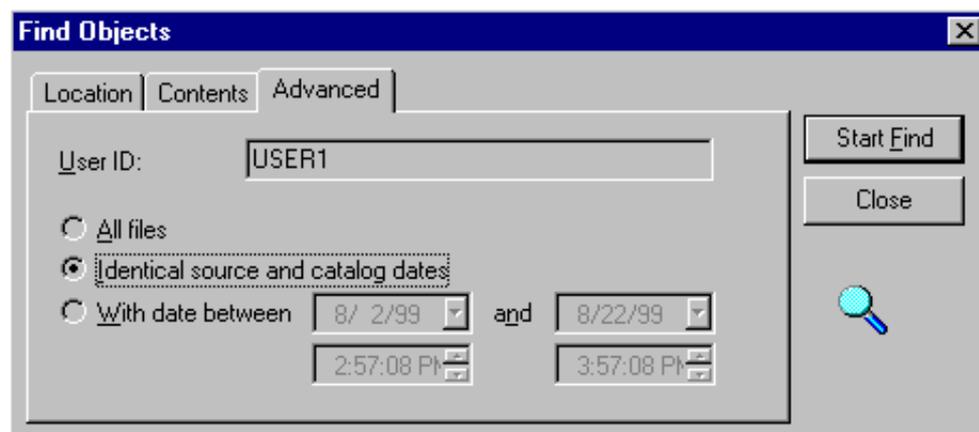


In the above "Contents" sheet the containing text "test" is being searched. If multiple occurrences are found in an object, the result list displays the object only once with the first occurrence displayed.

### "Advanced" Sheet

With the "Advanced" sheet it is possible to specify additional criteria for finding objects.

<b>User ID</b>	If specified, only objects with a matching user ID will be found.
<b>All files</b>	If this radio button is set, a search in all specified files takes place.
<b>Identical source and catalog dates</b>	If this radio button is set, only objects where the catalog and the source date match are found.
<b>With date between</b>	If this radio button is set, only objects where the last modification date is located between the specified range are found.



In the above "Advanced" sheet, only objects which were created with the user ID "USER1" are found. Additionally, the search is only successful when the last modification date of the object's source and generated program match.

## Saving an Object

The SAVE command is used to store the source object in the active editor window. The window is not closed afterwards. If no library name is specified in the SAVE command, the object is saved to the library from which it was opened or created, which is not necessarily the active library (the library displayed in the status line).

### ▶ To save an object:

- From the **Object** menu, choose **Save**.  
Or click the **Save** toolbar button.  
Or enter the SAVE command at the command line.

The source of the object in the active window is saved to the library displayed in the status line.

## Saving an Object with Another Name

This function creates a new object by copying the current contents of the editor to a new object and closing the original object. If no modifications have been made to the object since it was last saved, the function operates like a simple copy. If, however, modifications have been made since the last save, the new object contains the modifications and the old object is closed without saving the changes.

### ▶ To save an object to another name:

1. From the **Object** menu, choose **Save As**.  
Or click the **Save As** toolbar button.  
A dialog is displayed where you can specify a new object name, library, and type.
2. Choose **OK**.

## Stowing an Object

The STOW command stores an object in both source and object module form. First the source object is saved, then a syntax-check is performed to determine whether the object can be compiled. If no syntax errors are found, the object is compiled and the resulting object module is stored. The window is not closed afterwards. If no library name is specified in the STOW command, the object is stowed to the library from which it was opened or created, which is not necessarily the active library (the library displayed in the status line).

 **To stow an object:**

1. In the "Objects" window, select the object.  
Or open the object.
2. From the **Object** menu, choose **Stow**.  
Or click the **Stow** toolbar button.  
Or enter the STOW command at the command line.

The object code of the object in the active window is stowed to the library displayed in the status line.

## Shared Resources

A shared resource is any non Natural object such as a bitmap or a help file and is always associated with a specific library.

Natural Studio provides the same operations on shared resources as those applicable on the Windows Explorer. As with Natural objects, new shared resources can be created inside the Studio, they can be exported or imported, deleted, renamed, copied or moved. Since they are part of a library, they are also included in the library search order defined by the STEPLIB assignments.

For more information on resources, see the Programming Guide - Object Types - Using Non-Natural Files - Resource and User's Guide - Dialog Editor - ActiveX Control Property Pages.