

Linking Natural Objects to the Natural Nucleus

The Natural nucleus is a collection of service programs such as memory administration, string handling, operating system interfaces, the compiler and the runtime environment which comprise the kernel of Natural. It is independent of the operating-system and the TP system.

This document describes the advantages of linking Natural objects to the Natural nucleus and provides information on how to proceed.

The following topics are covered:

- Benefits
 - ULDOBJ Utility
 - Using ULDOBJ to Generate an Object Module
 - Additional Considerations for Linking Subroutines
 - Operating System Dependency of Object Module Generation
 - Example of Linking a Natural Object to the Natural Nucleus
-

Benefits

Linking Natural objects to the Natural nucleus provides the following benefits:

- **Better Performance**

The objects are executed from the nucleus and not from the Natural buffer pool. This saves space in the buffer pool and also results in fewer database calls. (If Natural cataloged objects are not linked to the Natural nucleus, they are stored in a database file, for example Adabas, and the actual code must be loaded from this file into the buffer pool before it can be executed.)
- **Consistency**

As an object which is linked to the Natural nucleus is always executed from the nucleus, there is no effect if the cataloged object from which it was derived is deleted or changed in the Natural system file. Thus, during each TP-monitor session, the status of the object remains unchanged. A new version of an object which is linked to the nucleus can be obtained by unloading it with ULDOBJ (see below), relinking the new version to the Natural nucleus and refreshing the Natural module. (Refreshing implies that a new copy of a module is loaded into the TP monitor region.)
- **Global Error Handling**

If a cataloged object fetches another program to handle errors (for example, by using the Natural system variable *ERROR-TA), and the error-handling program cannot be loaded into the buffer pool, the original error might be missed and any subsequent error may mask the first error and lead to confusion. To prevent this situation, you can link a user-written global error-handling program to the nucleus.

ULDOBJ Utility

You can use the ULDOBJ utility to link Natural cataloged objects to the Natural nucleus. With the ULDOBJ utility, you generate an object module from a Natural cataloged object and write it to a Natural work file. The generated object module is then processed by the linkage editor and linked to the Natural nucleus.

Under z/OS and z/VSE: When a Natural shared nucleus is used, the generated object module has to be linked to the environment-*independent* part of the nucleus.

Using ULDOBJ to Generate an Object Module

▶ To invoke the ULDOBJ utility

1. Log on to the library SYSMISC and issue the command ULDOBJ.

```

10:12:19          ***** NATURAL OBJECT MAINTENANCE *****          2005-01-05
User: XYZ          - NATURAL ULDOBJ UTILITY -                          Library: SYSMISC
                                                                Opsys .. z/OS

Specify parameters below ....

Object ..... _____ (Enter '.' to exit)
Library ..... SYSMISC_
OP System ... _____

```

2. Specify and confirm the following parameters:

Object	The name of the cataloged object to be processed. The object can be a program, subprogram, subroutine, help routine or map.
Library	The name of the library containing the cataloged object.
OP System	The name of the operating system for which the object module is to be generated. (Different operating systems have different rules to which the object module must conform.) The name of the operating system must be one of the following: <p>z/OS z/OS systems</p> <p>z/VSE z/VSE systems</p> <p>BS2000 BS2000/OSD systems</p> <p>CMS VM/CMS systems</p>

For each object processed, the ULDOBJ utility displays a report containing the following information:

- the object type (Program, Subprogram, Subroutine, Helproutine, Map, Adapter);
- the name of the cataloged object processed;
- the programming mode (S = structured mode, R = reporting mode)
- the name of the library containing the cataloged object;
- the name of the operating system for which the object deck was generated;
- the size of the cataloged object and optimized code (if applicable);
- the Natural version and system maintenance (SM) level of the cataloged object;
- statistics about the last cataloging of the object, including user and terminal IDs.

ULDOBJ prompts for another object and library after the data from the initial input have been processed. The operating system is not requested, because it does not make sense to generate object modules for more than one operating system for the same Natural work file.

To terminate the ULDOBJ utility

- After the last cataloged object has been processed, enter a "." in the first input field (Object) and press ENTER.

The generated object module conforms to the format of the specified operating system. It is in relocatable format with non-executable code and consists of:

- an external symbol directory (ESD),
- a relocation dictionary (RLD),
- text with the instructions and data corresponding to the program,
- an END statement (end-of-module indicator for the load module).

The generated object module is written to a Natural work file, which is used as input to a linkage editor. (Depending on the operating system, it may be better to use ULDOBJ in batch mode.)

The generated object module must be processed by the linkage editor of the corresponding operating system before the code is executable as a load module (see the example given below). Each load module is valid once it is linked to the Natural nucleus and defined by an NTSTAT entry definition in the Natural configuration module NATCONFIG (see *Natural Configuration Tables*).

Additional Considerations for Linking Subroutines

Once a cataloged object has been unloaded by the ULDOBJ utility and linked to the Natural nucleus, the cataloged object can be deleted from the Natural system file.

However, this is *not* true for an object of type "subroutine". A subroutine has two names:

- the name specified in the statements `PERFORM` and `DEFINE SUBROUTINE` and
- the name of the object that contains the `DEFINE SUBROUTINE` statement.

Natural internally associates these two names, but this is possible only if the cataloged object still exists on the Natural system file. If the cataloged object were deleted, this association would be lost and the subroutine linked to the nucleus would not be executable.

Operating System Dependency of Object Module Generation

The object module is generated in different ways, according to the operating system. These differences are listed below.

Platform:	Requirement:
z/OS	<p>A <code>NAME</code> control statement is generated as the last card of the object module. It specifies the replace function. For example:</p> <pre>NAME TEST (R)</pre> <p><code>TEST</code> is the name of the cataloged object.</p>
z/VSE	<p>The object module(s) will be in <code>LIBR</code> format. A <code>CATALOG</code> control statement is generated as the first card and a <code>"/*</code> as the last card of the object module. For example:</p> <pre>CATALOG TEST.OBJ REPLACE=YES object module ... /+</pre> <p><code>TEST.OBJ</code> is the name of the cataloged object.</p> <p>When the <code>LIBR</code> utility is executed, assign <code>SYSIPT</code> to the work file written by the <code>ULDOBJ</code> utility (<code>ASSIGN SYSIPT=work-file-1</code>).</p>
BS2000/OSD	<p>The object module(s) will be in <code>LMS</code> format. An <code>ADD</code> control statement is generated as the first card and an <code>END</code> statement as the last card of the object module. For example:</p> <pre>ADDR >TEST object module ... END</pre> <p>When the <code>LMS</code> utility is executed, assign <code>SYSDTA</code> to the work file written by the <code>ULDOBJ</code> utility (<code>SYSFILE SYSDTA=work-file-1</code>). The file name generated is <code>Nvr.MOD</code>, where <code>vr</code> stands for the current Natural version and release number.</p> <p>If multiple cataloged objects are unloaded during execution of the utility, the object decks are appended to each other.</p>

Example of Linking a Natural Object to the Natural Nucleus

If, for example, the objects LOGPROG and EDITPROG in the library SYSLIB are to be linked to the Natural nucleus, the following steps could be taken:

1. Identify the cataloged objects to be linked.

Object	Library
LOGPROG	SYSLIB
EDITPROG	SYSLIB

2. Set up the batch Natural job stream. Assuming a z/OS environment, include the following cards:

```
//CMWKF01 DD DSN=ULD.NAT.PGMS,UNIT=SYSDA,DISP=(,KEEP),
//          SPACE=(CYL,(3,1),,RLSE),VOL=SER=VVVVVV,
//          DCB=(RECFM=FB,BLKSIZE=800,LRECL=80)
//CMSYNIN DD *
LOGON SYSMISC
ULDOBJ LOGPROG,SYSLIB,OS
EDITPROG,SYSLIB
.
FIN
/*
```

3. Set up the linkage editor job stream.

```
//JOB CARD JOB (ACCTING),CLASS=A,MSGCLASS=X
/**
/** GENERATE OS LOAD MODULE FROM ULDOBJ UTILITY
/**
//LINK1 EXEC PGM=IEWL,PARM='LIST,LET,XREF,NCAL,RENT,REUS'
//SYSLMOD DD DSN=NATURAL.USER.LOAD,DISP=SHR
//SYSUT1 DD UNIT=SYSDA,SPACE=(1024,(200,20))
//SYSPRINT DD SYSOUT=X
//SYSLIN DD DSN=NAT.ULD.PGMS,DISP=OLD,UNIT=SYSDA,VOL=SER=VVVVVV
/*
```

This step places the load modules LOGPROG and EDITPROG in the NATURAL.USER.LOAD dataset.

With an additional link-edit job, these modules can be linked together as a single load module before being linked to the nucleus in Step 5.

```
//JOB CARD JOB (ACCTING),CLASS=A,MSGCLASS=X
/**
/** OPTIONAL JOB TO LINK CATALOGED OBJECTS TOGETHER
/**
//LINK2 EXEC PGM=IEWL,PARM='LIST,LET,XREF,NCAL,RENT,REUS'
//SYSLMOD DD DSN=NATURAL.USER.LOAD,DISP=SHR
//SYSUT1 DD UNIT=SYSDA,SPACE=(1024,(200,20))
//SYSPRINT DD SYSOUT=X
//SYSLIN DD *
INCLUDE SYSLMOD(LOGPROG) LOGON NATURAL PGM
INCLUDE SYSLMOD(EDITPROG) EDITOR NATURAL PGM
NAME XXXXXX(R)
/*
```

4. Define the statically linked Natural programs in source module NATCONFIG in the NSTATIC table for linked Natural programs:

```
NTSTAT INPL,TYPE=W
NTSTAT INPLLIB,TYPE=W
NTSTAT AERROR,TYPE=W
NTSTAT LOGPROG          <==== your entries
NTSTAT EDITPROG        <====
```

TYPE=W means that a "weak" external reference to the specified program is generated rather than a normal one.

5. Review the linkage editor job stream for the Natural nucleus and include the following:

```
/*
/* INCLUDE DDNAME AND DSN OF DATASET WHERE OBJECTS RESIDE
/*
//SYSLMOD DD DSN=NATURAL.USER.LOAD,DISP=SHR
//NATLIB DD DSN=NATURAL.V2.USER.LOAD,DISP=SHR/*
//SYSLIN DD*
...
...          INCLUDE MODULES FOR NUCLEUS
...
INCLUDE NATLIB(NATPARM)    NATPARM MODULE
INCLUDE SYSLMOD(LOGPROG)  LOGON NATURAL PGM
INCLUDE SYSLMOD(EDITPROG) EDITOR NATURAL PGM
...
...          INCLUDE ENTRY AND NAME CARDS
...
/*
```

If the cataloged objects were linked together (as done optionally in Step 3), include this load module instead of the individual load modules in the link of the nucleus.