Natural 3GL CALLNAT Interface - Usage, Examples

Natural 3GL CALLNAT Interface - Usage, Examples

This section describes the usage of the 3GL CALLNAT interface and describes a number of sample 3GL CALLNAT environments.

The following topics are covered:

- Usage
- Sample Environments

Usage

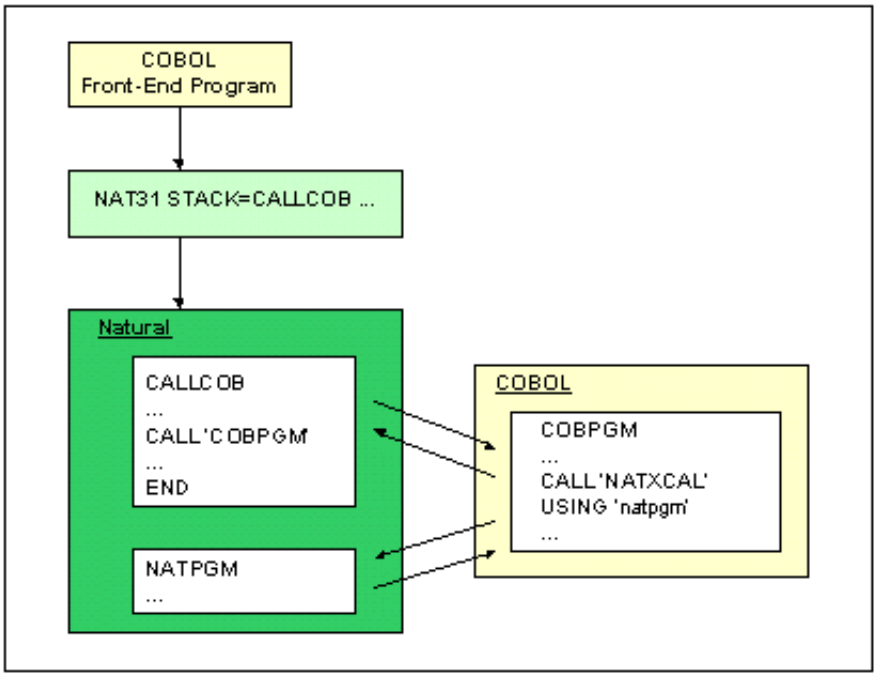
The following topics are covered:

- Overview
- Call Structure
- Parameter Handling

Overview

When you invoke a Natural subprogram from a 3GL program, a Natural session must be active, i.e. the 3GL program itself must be called by Natural.

Therefore you must take special precautions if you do not want the Natural layer to show up. The following figure is intended to give you an overview of how an application using the Natural 3GL CALLNAT interface may be designed in such a case:



The necessary environment is established by first invoking a Natural start-up program. By using the Natural `CALL` statement, this start-up program can then invoke a 3GL program from where you can invoke the `CALLNAT` interface.

Call Structure

The Natural main program is very simple; it only calls, for example, a COBOL program:

```
.....
CALL 'cobpgm'
END
.....
```

The `CALL` statement of the 3GL programming language (for example, COBOL) must have access to the Natural 3GL CALLNAT interface, which then invokes the Natural subprogram:

```
.....
CALL 'interface' USING natpgm p1 ... pn
.....
```

The parameter *`interface`* is environment-dependent (for example, `NATXCAL`) and linked to the calling program. The parameter *`natpgm`* must be an alphanumeric variable of 8 bytes that contains the name of the Natural subprogram to be invoked. The parameters *`p1 ... pn`* are passed to the Natural subprogram.

Example (for all environments except CICS):

The COBOL program *`cobpgm`* could contain coding similar to the following one:

```
.....
MOVE 'FINDNPGM' TO natpgm
CALL 'interface' USING natpgm number name
IF natpgm NE 'FINDNPGM'
THEN GOTO error_handling_1
.....
```

The invoked Natural subprogram *natpgm* calculates the number of persons in the file EMPLOYEES with *name* equal to a value passed from the COBOL program:

```
DEFINE DATA
PARAMETER
1 pnumber (P10)
1 pname (A20)
LOCAL
1 emp VIEW OF employees
END-DEFINE
*
RESET presp
FIND NUMBER emp WITH name=pname
MOVE *NUMBER TO pnumber
ESCAPE ROUTINE
```

If an error occurs while the subprogram is executed, information about this error will be returned in the variable *natpgm* in the form *NAT*nnnn*, where *nnnn* is the corresponding Natural error number.

Example (for CICS only):

Under CICS, the call of a Natural subroutine from, for example, COBOL should be as follows:

```
...
WORKING STORAGE SECTION
...
01 PARM-LIST PIC X(132).
01 NATPGM PIC X(8).
01 NUMBER PIC 9(10) comp-3.
01 NAME PIC X(20).
...
PROCEDURE DIVISION
...
MOVE 'FINDNPGM' TO NATPGM
CALL 'NCIXCPRM' USING PARM-LIST NATPGM NUMBER NAME ...
EXEC CICS LINK PROGRAM('NCIXCALL')
COMMAREA(PARM-LIST) LENGTH(132) END-EXEC.
...
```

The called subroutine NCIXCPRM builds the parameter address list used as COMMAREA in the subsequent EXEC CICS LINK command.

Parameter Handling

There is no format and length checking. It is the caller's responsibility to pass a correct parameter list. The number, format and length of the parameters are defined by the invoked Natural subprogram.

When you are passing parameters, group arrays should not be passed, since they are resolved as individual arrays:

Example of Invalid Syntax:

```
.....
01 GROUP (1:2)
02 F1
02 F2
.....
.....
CALL ..... F1 F2
.....
```

Example of Valid Syntax:

```
.....
01 F1 (1:2)
01 F2 (1:2)
....
.....
CALL ..... F1 F2
.....
```

Arrays with dynamic ranges are not possible.

Sample Environments

The objective for the sample 3GL CALLNAT environments below is to demonstrate how a COBOL routine can call a Natural subprogram under specific TP-monitor systems or in batch mode, and to give system-specific instructions to create such environments.

The following topics are covered:

- Sample Environment for CICS
- More Samples
- Sample for Any Other Supported Environment

Sample Environment for CICS

Perform the following steps to create a sample Natural 3GL CALLNAT environment under CICS:

Step 1: Create the Environment Initialization

- Set up the front-end program that initializes the 3GL CALLNAT environment.
- Use the COBOL front-end `XNCIFRCX` in the Natural/CICS source library. It starts Natural, stacks `LOGON YOURLIB` and executes the program `TSTCOB`, which initializes the Natural 3GL CALLNAT environment.
- Locate the string `NC`*`vr`* in the source code and replace it with the valid transaction ID for Natural.
- Compile and link-edit the COBOL program and define program to CICS via `CEDA DEFINE PROGRAM`.

Step 2: Install the Sample COBOL Call

Provided in the Natural/CICS source library NCI.SRCE is the sample member XNCI3GC1, which contains a default call to the Natural subprogram MYPROG.

- For test purposes, create the following program in the library SYSTEM and stow it as:

```
WRITE 'BEFORE PGM EXECUTION'
CALL 'COBNAT'
WRITE 'AFTER PGM EXECUTION'
END
```

- Look at the XNCI3GC1 source and review the CALL and LINK. Compile and link as COBNAT with the following CICS INCLUDE directives or use Step 2 of the Sample Job NCTI070:

```
INCLUDE CICSLIB(DFHECI)
INCLUDE XNCI3GC1                    <= output from translator and compiler
INCLUDE NCILIB(NCIXCPRM)
ENTRY XNCI3GC1
NAME COBNAT(R)
```

Step 3: Create a Sample Natural Subprogram

By default, the source member XNCI3GC1 is set up to call the Natural subprogram MYPROG in the library YOURLIB. The program TSTCOB, as mentioned above, starts up the process by calling COBNAT that contains the actual call to the Natural subprogram MYPROG.

- Create the subprogram MYPROG to demonstrate the executing Natural subprogram.

```
DEFINE DATA PARAMETER
  01 PARM1 (A18)
  01 PARM2 (A18)
  01 PARM3 (A18)
END-DEFINE
*
  MOVE 'PARAM01' TO PARM1
  MOVE 'PARAM02' TO PARM2
  MOVE 'PARAM03' TO PARM3
END
```

Step 4: Verify the CICS Resources

- Use the job NCII005 for a guide to defining the CICS resources (PPT and PCT).
- Define the required CICS resources (PPT and PCT).

Step 5: Test the Environment

Test the environment by using the NCYC default transaction. Use CEDF to monitor the program control and observe the data areas in use.

Important:
Since Natural is at the top of the CICS program hierarchy, any COBOL subprogram issuing terminal I/Os must run in conversational mode. Pseudo-conversational programs would need to be modified, and any new development using the Natural 3GL CALLNAT interface should be done in conversational mode.

More Samples

XNCI3GC2	COBOL sample with same functionality as XNCI3GC1, but accepting parameters from the calling Natural program.
XNCI3GP1	PL/I sample with same functionality as COBOL sample XNCI3GC1.
XNCI3GP2	PL/I sample with same functionality as XNCI3GC1, but accepting parameters from the calling Natural program.

More Non-CICS Samples

XNAT3GC2	COBOL sample with same functionality as CICS sample XNCI3GC2.
XNAT3GP2	PL/I sample with same functionality as CICS sample XNCI3GP2.

Sample for Any Other Supported Environment

Perform the following steps to create a sample Natural 3GL CALLNAT:

Step 1: Assemble and Link ASMNAT

The sample Assembler routine XNAT3GA1 contains a basic example to access the CALLNAT interface. The register calling conventions are in the source of this program.

Link NATXCAL with XNAT3GA1 with entry point ASMNAT.

Step 2: Start the Natural Session

Start a Natural session stacking a program that calls the ASMNAT program which in turn calls the Natural subroutine ASMNAT.