NSB - Interface Subprograms

Several Natural subprograms are available to provide you with either internal information from the Natural SQL Gateway or specific functions that are not available within the interface itself.

This section covers the following topics:

- Natural Subprograms

Natural Subprograms

The following Natural subprograms are available:

Subprogram	Function
NDBERR	Provides diagnostic information on the most recently executed SQL call.
NDBISQL	Executes SQL statements in dynamic mode.
NDBNOERR	Suppresses normal Natural error handling.
NDBNROW	Obtains the number of rows affected by a Natural SQL statement.

All these subprograms are provided in the Natural system library SYSTEM in the FNAT system file. In addition, the Natural library SYSTEM in the FNAT system file contains the subprogram DBTLIB2N and the subroutine DBDL219S. They are used by NDBDBRM and NDBDBR2. The corresponding parameters must be defined in a DEFINE DATA statement.

The Natural subprograms NDBBRM, NDBDBR2, NDBDBR3 allow the optional specification of the database ID, file number, password and cipher code of the library file containing the program to be examined.

If these parameters are not specified, either the actual FNAT file or the FUSER file is used to locate the program to be examined depending on whether the library name begins with SYS or the library name does not begin with SYS.

Programs invoking NDBBRM, NDBDBR2, NDBDBR3 without these parameters will also work as before this change as the added parameters are declared as optional.

NDBERR Subprogram

The Natural subprogram NDBERR replaces function E of the DB2SERV interface, which is still provided but no longer documented. It provides diagnostic information on the most recent SQL call. It also returns the database type which returned the error. NDBERR is typically called if a database call returns a non-zero SQL code.

A sample program called CALLERR is provided on the installation tape; it demonstrates how to invoke NDBERR. A description of the call format and of the parameters is provided in the text member NDBERRT.

The calling Natural program must use the following syntax:

```
CALLNAT 'NDBERR' #SQLCODE #SQLSTATE #SQLCA #DBTYPE
```

The various parameters are described in the following table:

Parameter	Format/Length	Explanation
SQLCODE	I4	Returns the SQL return code.
SQLSTATE	A5	Returns a return code for the output of the most recently executed SQL statement.
SQLCA	A136	Returns the SQL communication area of the most recent SQL access.
DBTYPE	B1	Returns the identifier (in hexadecimal format) for the currently used database. X'04' identifies access via Natural SQL Gateway (NSB) X'02' identifies access via Natural for DB2 (NDB).

NDBISQL Subprogram

The Natural subprogram NDBISQL is used to execute SQL statements in dynamic mode. The SELECT statement and all SQL statements which can be prepared dynamically can be passed to NDBISQL.

A sample program called CALLISQL is provided on the installation tape; it demonstrates how to invoke NDBISQL. A description of the call format and of the parameters is provided in the text member NDBISQLT.

The calling Natural program must use the following syntax:

```
CALLNAT 'NDBISQL'#FUNCTION #TEXT-LEN #TEXT (*) #SQLCA #RESPONSE #WORK-LEN #WORK (*)
```

The various parameters are described in the following table:

Parameter	Format/Length	Explanation	
#FUNCTION	A8	For valid functions, see below.	
#TEXT-LEN	I2	Length of the SQL statement or of the buffer for the return area.	
#TEXT	A1(1:V)	Contains the SQL statement or receives the return code.	
#SQLCA	A136	Contains the SQLCA.	
#RESPONSE	I4	Returns a response code.	
#WORK-LEN	I2	Length of the workarea specified by #WORK (optional).	
#WORK	A1(1:V)	Work area used to hold SQLDA/SQLVAR and auxiliary fields across calls (optional).	
#DBTYPE	I2	Database type (optional).	
		0	Default
		2	DB2
		4	CNX

Valid functions for the #FUNCTION parameter are:

Function	Parameter	Explanation
CLOSE		Closes the cursor for the SELECT statement.
EXECUTE	#TEXT-LEN #TEXT (*)	Executes the SQL statement. Contains the length of the statement. Contains the SQL statement. The first two characters must be blank.
FETCH	#TEXT-LEN #TEXT (*)	Returns a record from the SELECT statement. Size of #TEXT (in bytes). Buffer for the record.
TITLE	#TEXT-LEN #TEXT (*)	Returns the header for the SELECT statement. Size of #TEXT (in bytes); receives the length of the header (= length of the record). Buffer for the header line.

The #RESPONSE parameter can contain the following response codes:

Code	Function	Explanation
5	EXECUTE	The statement is a SELECT statement.
6	TITLE, FETCH	Data are truncated; only set on first TITLE or FETCH call.
100	FETCH	No record / end of data.
-2		Unsupported data type (for example, GRAPHIC).
-3	TITLE, FETCH	No cursor open; probably invalid call sequence or statement other than SELECT.
-4		Too many columns in result table.
-5		SQL code from call.
-6		Version mismatch.
-7		Invalid function.
-8		Error from SQL call.
-9		Workarea invalid (possibly relocation).
-10		Interface not available.
-11	EXECUTE	First two bytes of statement not blank.

Call Sequence

The first call must be an EXECUTE call. NDBISQL has a fixed SQLDA AREA holding space for 50 columns. If this area is too small for a particular SELECT, it is possible to supply an optional work area on the calls to NDBISQL by specifying #WORK-LEN (I2) and #WORK(A1/1:V).

This workarea is used to hold the SQLDA and temporary work fields like null indicators and auxiliary fields for numeric columns. Calculate 16 bytes for SQLDA header and 44 bytes for each result column and 2 bytes null indicator for each column and place for each numeric column, when supplying #WORK-LEN and #WORK(*) during NDBISQL calls. If these optional parameters are specified on an EXECUTE call they have also to be specified on any following call.

If the statement is a SELECT statement (that is, response code 5 is returned), any sequence of TITLE and FETCH calls can be used to retrieve the data. A response code of 100 indicates the end of the data.

The cursor must be closed with a CLOSE call.

Function code EXECUTE implicitly closes a cursor which has been opened by a previous EXECUTE call for a SELECT statement.

In TP environments, no terminal I/O can be performed between an EXECUTE call and any TITLE, FETCH or CLOSE call that refers to the same statement.

NDBNOERR Subprogram

The Natural subprogram NDBNOERR is used to suppress Natural NAT3700 errors caused by the next SQL call. This allows a program controlled continuation if an SQL statement produces a non-zero SQL code. After the SQL call has been performed, NDBERR is used to investigate the SQL code.

A sample program called CALLNOER is provided on the installation tape; it demonstrates how to invoke NDBNOERR. A description of the call format and of the parameters is provided in the text member NDBNOERT.

The calling Natural program must use the following syntax:

```
CALLNAT 'NDBNOERR'
```

There are no parameters provided with this subprogram.

Note:
Only NAT3700 errors (that is, non-zero SQL response codes) are suppressed, and also only errors caused by the next following SQL call.

Restrictions with Database Loops

- If NDBNOERR is called before a statement that initiates a database loop and an initialization error occurs, no processing loop will be initiated, unless the IF NO RECORDS FOUND clause has been specified.
- If NDBNOERR is called within a database loop, it does not apply to the processing loop itself, but only to the SQL statement subsequently executed inside this loop.

NDBNROW Subprogram

The Natural subprogram NDBNROW is used to obtain the number of rows affected by the Natural SQL statements Searched UPDATE, Searched DELETE, and INSERT. The number of rows affected is read from the SQL communication area (SQLCA). A positive value represents the number of affected rows, whereas a value of minus one (-1) indicates that all rows of a table in a segmented tablespace have been deleted.

A sample program called CALLNROW is provided on the installation tape; it demonstrates how to invoke NDBNROW. A description of the call format and of the parameters is provided in the text member NDBNROWT.

The calling Natural program must use the following syntax:

```
CALLNAT 'NDBNROW' #NUMBER
```

The parameter #NUMBER (I4) contains the number of affected rows.