NDB - Interface Subprograms

Several Natural and non-Natural subprograms are available to provide you with either internal information from Natural for DB2 or specific functions for which no equivalent Natural statements exist.

From within a Natural program, Natural subprograms are invoked with the CALLNAT statement and non-Natural subprograms are invoked with the CALL statement.

This section covers the following topics:

- Natural Subprograms

Natural Subprograms

Subprogram	Function
NDBCONV	Sets or resets conversational mode 2.
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.
NDBSTMP	Provides a DB2 TIMESTAMP column as an alphanumeric field and vice versa.

All these subprograms are provided in the Natural system library SYSDB2 and the Natural library SYSTEM in the FNAT system file.

NDBCONV Subprogram

The Natural subprogram NDBCONV is used to either set or reset the conversational mode 2 in CICS environments. Conversational mode 2 means that update transactions are spawned across terminal I/Os until either a COMMIT or ROLLBACK has been issued (Caution DB2 and CICS resources are kept across terminal I/Os!). This means conversational mode 2 has the same effect as the Natural parameter PSEUDO=OFF, except that the conversational mode is entered after an DB2 update statement (UPDATE, DELETE, INSERT) and left again after a COMMIT or ROLLBACK, while PSEUDO=OFF causes conversational mode for the total Natural session.

A sample program called CALLCONV is provided in library SYSDB2; it demonstrates how to invoke NDBCONV. A description of the call format and of the parameters is provided in the text member NDBCONVT.

The calling Natural program must use the following syntax:

```
CALLNAT 'NDBCONV' #CONVERS #RESPONSE
```

The various parameters are described in the following table:

Parameter	Format/Length	Explanation
#CONVERS	I1	Contains the desired conversational mode(input)
#RESPONSE	I4	Contains the response of NDBCONV(output)

The #CONVERS parameter can contain the following values:

Code	Explanation
0	The conversational mode 2 has to be reset.
1	The conversational mode 2 has to be set.

The #RESPONSE parameter can contain the following response codes:

Code	Explanation
0	The conversational mode 2 has been successfully set or reset.
-1	The specified value of #CONVERS is invalid, the conversational mode has not been changed.
-2	NDBCONV is called in a environment, which is not a CICS environment, where the conversational mode 2 is not supported.

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 (which means a NAT3700 error).

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 DB2 access.
DBTYPE	B1	Returns the identifier (in hexadecimal format) for the currently used database (where X'02' identifies DB2).

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 (according to the DB2 literature by IBM) 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)	Workarea 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 (see also the Natural system variable *NUMBER as described in Natural System Variables).

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.

NDBSTMP Subprogram

For DB2, Natural provides a TIMESTAMP column as an alphanumeric field (A26) of the format *YYYY-MM-DD-HH.MM.SS.MMMMMM*.

Since Natural does not yet support computation with such fields, the Natural subprogram NDBSTMP is provided to enable this kind of functionality. It converts Natural time variables to DB2 time stamps and vice versa and performs DB2 time stamp arithmetics.

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

The functions available are:

Code	Explanation
ADD	Adds time units (labeled durations) to a given DB2 time stamp and returns a Natural time variable and a new DB2 time stamp.
CNT2	Converts a Natural time variable (format T) into a DB2 time stamp (column type TIMESTAMP) and labeled durations.
C2TN	Converts a DB2 time stamp (column type TIMESTAMP) into a Natural time variable (format T) and labeled durations.
DIFF	Builds the difference between two given DB2 time stamps and returns labeled durations.
GEN	Generates a DB2 time stamp from the current date and time values of the Natural system variable *TIMX and returns a new DB2 time stamp.
SUB	Subtracts labeled durations from a given DB2 time stamp and returns a Natural time variable and a new DB2 time stamp.
TEST	Tests a given DB2 time stamp for valid format and returns TRUE or FALSE.

Note:
Labeled durations are units of year, month, day, hour, minute, second and microsecond.