

# Interface Subprograms

Several Natural and non-Natural subprograms are available to provide you with either internal information from the Natural interface to DB2 or specific functions that are not available within the interface itself.

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
- DB2SERV Interface

## Natural Subprograms

The following Natural subprograms are provided:

- NDBDBRM Subprogram
- NDBDBR2 Subprogram
- NDBERR Subprogram
- NDBISQL Subprogram
- NDBNOERR Subprogram
- NDBNROW Subprogram
- NDBSTMP Subprogram

Subprogram	Function
NDBDBRM	Checks whether a Natural program contains SQL access and whether it has been modified for static execution.
NDBDBR2	Checks whether a Natural program contains SQL access and whether it has been modified for static execution.
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 library SYSD2 and should be copied to the SYSTEM or steplib library, or to any library where they are needed. In addition, the subprogram DBTLIB2N and the subroutine DBDL219S have to be copied from SYSD2. They are used by NDBDBRM and NDBDBR2. The corresponding parameters must be defined in a DEFINE DATA statement.

### NDBDBRM Subprogram

The Natural subprogram NDBDBRM is used to check whether a Natural program contains SQL access and whether it has been modified for static execution. It is also used to obtain the corresponding DBRM name from the header of a Natural program generated as static (see also Preparing Natural Programs for Static Execution).

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

The calling Natural program must use the following syntax:

```
CALLNAT 'NDBDBRM' #LIB #MEM #DBRM #RESP
```

The various parameters are described in the following table:

Parameter	Format/Length	Explanation
#LIB	A8	Contains the name of the library of the program to be checked.
#MEM	A8	Contains the name of the program (member) to be checked
#DBRM	A8	Returns the DBRM name.
#RESP	I2	Returns a response code. The possible codes are listed below.

The #RESP parameter can contain the following response codes:

Code	Explanation
0	The member #MEM in library #LIB has SQL access; it is static if #DBRM contains a value.
-1	The member #MEM in library #LIB has no SQL access.
-2	The member #MEM in library #LIB does not exist.
-3	No library name has been specified.
-4	No member name has been specified.
-5	The library name must start with a letter.
<-5	Further negative response codes correspond to error numbers of Natural error messages.
> 0	Positive response codes correspond to error numbers of Natural Security messages.

## NDBDBR2 Subprogram

The Natural subprogram NDBDBR2 is used to check whether a Natural program contains SQL access and whether it has been modified for static execution. It is also used to obtain the corresponding DBRM name from the header of a Natural program generated as static (see also Preparing Natural Programs for Static Execution) and the time stamp generated by the precompiler.

A sample program called CALLDBR2 is provided on the installation tape; it demonstrates how to invoke NDBDBR2. A description of the call format and of the parameters is provided in the text member NDBDBR2T.

The calling Natural program must use the following syntax:

```
CALLNAT 'NDBDBR2' #LIB #MEM #DBR2 #TIMESTAMP #PCUSER #PCRELLEV #ISOLLEVL #DATEFORM #TIMEFORM #RESP
```

The various parameters are described in the following table:

Parameter	Format/Length	Explanation
#LIB	A8	Contains the name of the library of the program to be checked.
#MEM	A8	Contains the name of the program (member) to be checked
#DBR2	A8	Returns the DBR2 name.
#TIMESTAMP	B8	Consistency token generated by precompiler
#PCUSER	A1	User ID used at precompilile (only SQL/DS)
#PCRELLEV	A1	Release level of precompiler (only SQL/DS)
#ISOLLEVL	A1	Precompiler isolation level (only SQL/DS)
#DATEFORM	A1	Date format (only SQL/DS)
#TIMEFORM	A1	Time format (only SQL/DS)
#RESP	I2	Returns a response code. The possible codes are listed below.

The #RESP parameter can contain the following response codes:

Code	Explanation
0	The member #MEM in library #LIB has SQL access; it is static if #DBRM contains a value.
-1	The member #MEM in library #LIB has no SQL access.
-2	The member #MEM in library #LIB does not exist.
-3	No library name has been specified.
-4	No member name has been specified.
-5	The library name must start with a letter.
<-5	Further negative response codes correspond to error numbers of Natural error messages.
> 0	Positive response codes correspond to error numbers of Natural Security messages.

## NDBERR Subprogram

The Natural subprogram NDBERR replaces the "E" function 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); see also Error Handling.

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 documentation) 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
```

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.

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.
-10		Interface not available.
-11	EXECUTE	First two bytes of statement not blank.

## Call Sequence

The first call must be an EXECUTE 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; see also Error Handling.

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 "-1" indicates that all rows of a table in a segmented tablespace have been deleted (see also \*NUMBRT).

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 <b>TIMESTAMP</b> ) and labeled durations.
C2TN	Converts a DB2 time stamp (column type <b>TIMESTAMP</b> ) 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.

## DB2SERV Interface

DB2SERV is an Assembler program entry point which can be called from within a Natural program.

DB2SERV performs either of the following functions:

- Function "D", which performs the SQL statement EXECUTE IMMEDIATE;
- Function "P", which is used to establish the DB2 connection (TSO and batch mode only).

The parameter or variable values returned by each of these functions are checked for their format, length, and number.

### Function "D"

Function "D" performs the SQL statement EXECUTE IMMEDIATE. This allows SQL statements to be issued from within a Natural program.

The SQL statement string that follows the EXECUTE IMMEDIATE statement must be assigned to the Natural program variable STMT. It must contain valid SQL statements allowed with the EXECUTE IMMEDIATE statement as described in the relevant IBM documentation. Examples can be found below and in the demonstration programs DEM2\* in library SYSDB2.

**Note:**

The conditions that apply to issuing the Natural END TRANSACTION or BACKOUT TRANSACTION statements also apply when issuing the SQL COMMIT or ROLLBACK statements.

## Command Syntax

```
CALL 'DB2SERV' 'D' STMT STMTL SQLCA RETCODE
```

The variables used in this command are described in the following table:

Variable	Format/Length	Explanation
STMT	<i>Annn</i>	Contains a command string which consists of SQL syntax as described above.
STMTL	I2	Contains the length of the string defined in STMT.
SQLCA	A136	Returns the current contents of the SQL communication area.
RETCODE	I2	Returns an interface return code. The following codes are possible: <ul style="list-style-type: none"> <li>0 No warning or error occurred.</li> <li>4 SQL statement produced an SQL warning.</li> <li>8 SQL statement produced an SQL error.</li> <li>12 Internal error occurred; the corresponding Natural error message number can be displayed with SQLERR.</li> </ul>

The current contents of the SQLCA and an interface return code (RETCODE) are returned. The SQLCA is a collection of variables that are used by DB2 to provide an application program with information on the execution of its SQL statements.

The following two examples show you how to use DB2SERV with function "D":

### Example of Function "D" - DEM2CREA:

```
*****
* DEM2CREA - CREATE TABLE NAT.DEMO *
*****
*
DEFINE DATA
LOCAL USING DEMSQLCA
LOCAL
*
*                               Parameters for DB2SERV
1 STMT          (A250)
1 STMTL         (I2)      CONST <250>
1 RETCODE       (I2)
*
END-DEFINE
*
COMPRESS 'CREATE TABLE NAT.DEMO'
' (NAME          CHAR(20)      NOT NULL, '
' ADDRESS       VARCHAR(100) NOT NULL, '
' DATEOFBIRTH  DATE          NOT NULL, '
' SALARY        DECIMAL(6,2), '
' REMARKS      VARCHAR(500)) '
INTO STMT
CALL 'DB2SERV' 'D' STMT STMTL SQLCA RETCODE
*
END TRANSACTION
*
IF RETCODE = 0
WRITE 'Table NAT.DEMO created'
ELSE
FETCH 'SQLERR'
END-IF
END
*****
```

### Note:

The functionality of the DB2SERV function "D" is also provided with the PROCESS SQL statement (see also SQL Statements in the Natural Statements documentation).

**Example of Function "D" - DEM2SET:**

```

*****
* DEM2SET - Set Current SQLID *
*****
*
DEFINE DATA
LOCAL USING DEMSQLCA
LOCAL
*
*                               Parameter for DB2SERV
1 STMT          (A250)
1 STMTL         (I2)      CONST <250>
1 RETCODE       (I2)
1 OLDSQLID      (A8)
1 NEWSQLID      (A8)
*
END-DEFINE
*
SELECT DISTINCT CURRENT SQLID
      INTO OLDSQLID
      FROM SYSIBM.SYSTABLES
ESCAPE BOTTOM
END-SELECT
*
MOVE 'SET CURRENT SQLID="PROD"' ;
TO STMT
CALL 'DB2SERV' 'D' STMT STMTL SQLCA RETCODE
*
IF RETCODE > 0
      FETCH 'SQLERR'
ELSE
      SELECT DISTINCT CURRENT SQLID
            INTO NEWSQLID
            FROM SYSIBM.SYSTABLES
            ESCAPE BOTTOM
            END-SELECT
*
      WRITE ' Old SQLID was :' OLDSQLID
      WRITE ' New SQLID is   :' NEWSQLID
END-IF
*
END
*****

```

When using SET CURRENT SQLID, the creator name of a table can be substituted by the current SQLID. This enables you to access identical tables with the same table name but with different creator names. Thus, table names should not be qualified by a creator name if this is to be substituted by the SQLID.

In all supported TP-monitor environments, the SQLID can then be kept across terminal I/Os until either the end of the session or its resetting via DB2SERV.

## Function "P"

Function "P" invokes an Assembler module named NDBPLAN, which is used to establish and/or terminate the DB2 connection under TSO and in batch mode. This allows a Natural application to perform plan switching under TSO and in batch mode.

The program DEM2PLAN is an example for the use of DB2SERV with function "P".

The name of the current DB2 subsystem (#SSM) and the name of the new application plan (#PLAN) must be specified. In addition, an interface return code (#RETCODE) and the DB2 reason code (#REASON) are returned.

### Command Syntax

```
CALL 'DB2SERV' 'P' #SSM #PLAN #RETCODE #REASON
```

Variable	Format/Length	Explanation
#SSM	A4	Contains the name of the current DB2 subsystem.
#PLAN	A8	Contains the new plan name.
#RETCODE		Returns an interface return code. The following codes are possible: <ul style="list-style-type: none"> <li>0 No warning or error occurred.</li> <li>12 The specified new application plan is not scheduled.</li> <li>99 The current environment is not a CAF environment.</li> <li><i>nnn</i> Return code from the CAF interface (see also the relevant DB2 literature).</li> </ul>
#REASON	I4	Returns the reason code of the CAF interface (see also the relevant DB2 literature).

**Example of Function "P" - DEM2PLAN:**

```

*****
* DEM2PLAN - Switch application plan under TSO/Batch with CAF interface *
*****
*
DEFINE DATA
LOCAL
*
*                               Parameter for DB2SERV
01 #SSM          (A4)    CONST <'DB2'>
01 #PLAN         (A8)
01 #RETCODE      (I2)
01 #REASON       (I4)
*
END-DEFINE
*
INPUT 'PLEASE ENTER NEW PLAN NAME' #PLAN (AD='_' I)
*
END TRANSACTION
*
CALL 'DB2SERV' 'P' #SSM #PLAN #RETCODE #REASON
*
DECIDE FOR FIRST VALUE OF #RETCODE
*
  VALUE 0
    IGNORE
  VALUE 99
    INPUT 12/23 'This is not a CAF environment !!'
  VALUE 8,12
    INPUT 12/18 'New plan not scheduled, reason code'
                #REASON (AD=OI EM=H(4))
  NONE
    INPUT 12/15 'CAF interface error'
                #RETCODE (AD=OI EM=Z(3))
                'with reason code'
                #REASON (AD=OI EM=H(4))
*
END-DECIDE
*
END
*****

```

**Important:**

Plan switching under TSO and in batch mode is possible with the CAF interface only; see also the section Plan Switching under TSO and in Batch Mode.