Interface Subprograms

Several Natural subprograms and a non-Natural program (DB2SERV Interface, written in Assembler) are available to provide you with internal information from the Natural interface to SQL/DS or specific functions that are not available within the interface itself.

- Natural Interface Subprograms
- NDBDBRM Subprogram
- NDBDBR2 Subprogram
- NDBDBR3 Subprogram
- NDBERR Subprogram
- NDBISQL Subprogram
- NDBNOERR Subprogram
- NDBNROW Subprogram
- NDBSTMP Subprogram
- DB2SERV Interface

Natural Interface Subprograms

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

All Natural subprograms are provided in the library SYSSQL and should be copied to the SYSTEM or steplib library, or to any library where they are needed. The corresponding parameters must be defined by using either the DEFINE DATA statement in structured mode or the RESET statement in reporting mode.

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 current system file FNAT or the system file FUSER 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 like before this change as the added parameters are declared as optional.

Overview of Interface Subprograms

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.
NDBDBR3	Checks whether a Natural program contains SQL access, whether it has been modified for static execution, and whether it can be generated as static.
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 an SQL/DS TIMESTAMP column as an alphanumeric field and vice versa.

For detailed information on these subprograms, follow the links shown in the table above and read the description of the call format and of the parameters in the text member provided with the subprogram (*subprogram-name*`T`).

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 package 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 #DBID #FILENR #PASSWORD #CIPHER
```

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.
#DBID	N5	Optional, Database ID of library file.
#FILENR	N5	Optional, File number of library file.
#PASSWORD	A8	Optional, Password of library file.
#CIPHER	N8	Optional, Cipher code of library file.

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 #DBRM #TIMESTAMP #PCUSER #PCRELLEV #ISOLLEVL #DATEFORM #TIMEFORM #RESP #DBID #FILENR #PASSWORD #CIPHER
```

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.
#TIMESTAMP	B8	Consistency token generated by precompiler
#PCUSER	A1	User ID used at precomplile (only SQL/DS)
#PCRELLEV	A1	Release level of precompiler (only SQL/DS)
#ISOLLEVL	A1	Precomplier 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.
#DBID	N5	Optional, Database ID of library file.
#FILENR	N5	Optional, File number of library file.
#PASSWORD	A8	Optional, Password of library file.
#CIPHER	N8	Optional, Cipher code of library file.

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 #DBR2 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.

NDBDBR3 Subprogram

The Natural subprogram NDBDBR3 is used to check whether a Natural program contains SQL access (#RESP 0), whether the Natural program contains solely SQL statements, which are dynamically executable (#RESP 0, #DBRM '*DYNAMIC') and whether it has been modified for static execution (#RESP 0, #DBRM *dbrmname*). It is also used to obtain the corresponding DBRM name from the header of a Natural program generated as static (see also *Preparing Programs for Static Execution*) and the time stamp generated by the precompiler.

A sample program called CALLDBR3 is provided on the installation tape; it demonstrates how to invoke NDBDBR3. A description of the call format and of the parameters is provided in the text member NDBDBR3T.

The calling Natural program must use the following syntax:

```
CALLNAT 'NDBDBR3' #LIB #MEM #DBRM #TIMESTAMP #PCUSER #PCRELLEV #ISOLLEVL #DATEFORM #TIMEFORM #RESP #DBID #FILENR #PASSWORD #CIPHER
```

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. ● Space, if program has SQL access ● *DYNAMIC, if program contains only dynamically executable SQL ● DBRM name, if program has been generated static.
#TIMESTAMP	B8	Consistency token generated by precompiler
#PCUSER	A1	User ID used at precomplile (only SQL/DS)
#PCRELLEV	A1	Release level of precompiler (only SQL/DS)
#ISOLLEVL	A1	Precomplier 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.
#DBID	N5	Optional, Database ID of library file.
#FILENR	N5	Optional, File number of library file.
#PASSWORD	A8	Optional, Password of library file.
#CIPHER	N8	Optional, Cipher code of library file.

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 other than space and *DYNAMIC.
-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 *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 SQL/DS access.
#DBTYPE	B1	Returns the identifier (in hexadecimal format) for the currently used database (where X'03' identifies SQL/DS).

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 Adabas SQL Server 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 #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).

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.

Notes:

1. Function code EXECUTE implicitly closes a cursor which has been opened by a previous EXECUTE call for a SELECT statement.
2. 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 *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 a 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 the Natural *System Variables* documentation.

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 SQL/DS, Natural provides a TIMESTAMP column as an alphanumeric field (A26) of the format *YYYY-MM-DD-HH.MM.SS.MMMMMM*; see also *List Columns of an SQL Table*.

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 SQL/DS time stamps and vice versa and performs SQL/DS 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 SQL/DS time stamp and returns a Natural time variable and a new SQL/DS time stamp.
CNT2	Converts a Natural time variable (format T) into a SQL/DS time stamp (column type TIMESTAMP) and labeled durations.
C2TN	Converts a SQL/DS time stamp (column type TIMESTAMP) into a Natural time variable (format T) and labeled durations.
DIFF	Builds the difference between two given SQL/DS time stamps and returns labeled durations.
GEN	Generates a SQL/DS time stamp from the current date and time values of the Natural system variable *TIMX and returns a new SQL/DS time stamp.
SUB	Subtracts labeled durations from a given SQL/DS time stamp and returns a Natural time variable and a new SQL/DS time stamp.
TEST	Tests a given SQL/DS 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 U, which calls the database connection services (z/VSE 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 SYSSQL.

Note:
The conditions that apply to issuing Natural END TRANSACTION or BACKOUT TRANSACTION statements also apply when issuing Natural 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	A*nnn*	Contains a command string which consists of SQL syntax as described above.	
STMTL	I2	Contains the length of the string defined in the Natural program variable STMT.	
SQLCA	A136	Returns the current contents of the SQL communication area.	
RETCODE	I2	Returns an interface return code. The following codes are possible:	
		0	No warning or error occurred.
		4	SQL statement produced an SQL warning.
		8	SQL statement produced an SQL error.
		12	Internal error occurred;the corresponding Natural error message number can be displayed with SYSERR.

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 SQL/DS to provide an application program with information on the execution of its SQL statements.

The following example shows 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 the section *SQL Statements* in the Natural *Statements* documentation).

Function U

Function U calls the database connection services when running in batch mode under z/VSE; see also *Sample Batch Verification Job (z/VSE only)*.

The user ID and password for the connection to SQL/DS must be assigned to the Natural program variables USER-ID and PASSWORD, respectively. An interface return code (RETCODE) is returned.

Command Syntax

```
CALL 'DB2SERV' 'U' USER-ID PASSWORD RETCODE
```

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

Variable	Format/Length	Explanation
USER-ID	A8	A Natural variable that contains the user ID for the connection to SQL/DS.
PASSWORD	A8	A Natural variable that contains the user password for the connection to SQL/DS.
RETCODE	I2	A Natural variable that returns an interface return code. The following codes are possible: 0 No warning or error occurred. 4 SQL statement produced an SQL warning. 8 SQL statement produced an SQL error. 12 Internal error occurred; information on this error can be displayed with the Natural Utility SYSERR.

Variable	Format/Length	Explanation	
USER-ID	A8	A Natural variable that contains the user ID for the connection to SQL/DS.	
PASSWORD	A8	A Natural variable that contains the user password for the connection to SQL/DS.	
RETCODE	I2	A Natural variable that returns an interface return code. The following codes are possible:	
		0	No warning or error occurred.
		4	SQL statement produced an SQL warning.
		8	SQL statement produced an SQL error.
		12	Internal error occurred;the corresponding Natural error message number can be displayed with the Natural Utility SYSERR.