NDB - PARAMETER STYLE

The PARAMETER STYLE identifies the linkage convention used to pass parameters to a DB2 stored procedure or a DB2 UDF.

This section describes the PARAMETER STYLEs and the STCB NDB uses for processing Natural DB2 stored procedures or Natural UDFs. Note that PARAMETER STYLE GENERAL (or GENERAL WITH NULL) and STCB Layout only apply to Natural stored procedures.

- GENERAL and GENERAL WITH NULL
- STCB Layout
- DB2SQL

GENERAL and GENERAL WITH NULL

Only applies to Natural stored procedures.

A Natural stored procedure defined with PARAMETER STYLE GENERAL only receives the user parameters specified.

A Natural stored procedure defined with PARAMETER STYLE GENERAL WITH NULL receives the user parameters specified and, additionally, a NULL indicator array that contains one NULL indicator for each user parameter.

Natural stored procedures defined with PARAMETER STYLE GENERAL/WITH NULL, require that the definition of the stored procedure within the DB2 catalog includes one additional parameter of the type VARCHAR in front of the user parameters of the stored procedure.

This parameter in front of the parameters is the STCB (Stored Procedure Control Block); see also STCB Layout below.

Below is information on:

- STCB - Stored Procedure Control Block
- Example of PARAMETER STYLE GENERAL
- Example of GENERAL WITH NULL

STCB - Stored Procedure Control Block

The STCB contains information the NDB server stub uses to execute Natural stored procedures, such as the library and the subprogram to be invoked. It also contains the format descriptions of the parameters passed to the stored procedure.

The STCB is invisible to the Natural stored procedure called. The STCB is evaluated by the NDB server stub and stripped off the parameter list that is passed to the Natural stored procedure.

If the caller of a Natural stored procedure defined with PARAMETER STYLE GENERAL/WITH NULL is a Natural program, the program must use a CALLDBPROC statement with the keyword CALLMODE=NATURAL.

If the caller of the Natural stored procedure is *not* a Natural program, the caller has to set up the STCB for the DB2 CALL statement and pass the STCB as the first parameter.

If an error occurs during the execution of a Natural stored procedure defined with PARAMETER STYLE GENERAL/WITH NULL, the error message text is returned to the STCB.

If the caller is a Natural program that uses CALLDBPROC and CALLMODE=NATURAL, the NDB runtime will wrap up the error text in the NAT3286 error message.

Example of PARAMETER STYLE GENERAL

In the Natural stored procedure, define the parameters as shown in the example program below:

```
DEFINE DATA PARAMETER
01 P1 ...
01 P2 ...
...
...
01 Pn ...
LOCAL
...
...
END-DEFINE
```

Example of GENERAL WITH NULL

In the Natural stored procedure, define the parameters as shown in the example program below:

```
DEFINE DATA PARAMETER
01 P1 ...
01 P2 ...
...
...
01 Pn ...
01 NULL-INDICATOR-ARRAY  (I2/1:n)
LOCAL
...
...
END-DEFINE
```

STCB Layout

Only applies to Natural stored procedures.

The following table describes the first parameter passed between the caller and the Natural stored procedure if CALLMODE = NATURAL (see also the relevant section in CALLDBPROC) is specified.

NAME	**FORMAT**	**PROCESSING MODE SERVER**
STCBL	I2	Input (size of following information)
Procedure Information		
STCBLENG	A4	Input (printable STCBL)
STCBID	A4	Input (’STCB’)
STCBVERS	A4	Input (version of STCB ’310 ’)
STCBUSER	A8	Input (user ID)
STCBLIB	A8	Input (library)
STCBPROG	A8	Input (calling program)
STCBPSW	A8	Unused (password)
STCBSTNR	A4	Input (CALLDBPROC statement number)
STCBSTPC	A8	Input (procedure called)
STCBPANR	A4	Input (number of parameters)
Error Information		
STCBERNR	A5	Output (Natural error number)
STCBSTAT	A1	Unused (Natural error status)
STCBLIB	A8	Unused (Natural error library)
STCBPRG	A8	Unused (Natural error program)
STCBLVL	A1	Unused (Natural error level)
STCBOTP	A1	Unused (error object type)
STCBEDYL	A2	Output (error text length)
STCBEDYT	A88	Output (error text)
	A100	Reserved for future use
Parameter Information		
STCBPADE	A variable	Input. See also PARAMETER DESCRIPTION (STCBPADE) below.

Below is information on:

- PARAMETER DESCRIPTION (STCBPADE)

PARAMETER DESCRIPTION (STCBPADE)

PARAMETER DESCRIPTION contains a description for each parameter passed to the Natural stored procedure consisting of parameter type, format specification and length. Parameter type is the AD attribute of the CALLNAT statement as described in the Natural Statements documentation.

Each parameter has the following format description element in the STCBPADE string

atl, *p*[, *d1*]....

where

- *a* is an attribute mark which specifies the parameter type:

Mark	Type	Equivalent AD Attribute	Equivalent DB2 Clause
M	modifiable	AD=M	INOUT
O	non-modifiable	AD=O	IN
A	input only	AD=A	OUT

- *t* is one of the following Natural format tokens:

t	Description	*l*	*p*	*dl*	Example
A	Alphanumeric	1-253	0	1-32767 or -	A30,0 or A30,0,10
N	Numeric unpacked	1-29	0-7	-	N10,3
P	Packed numeric	1-29	0-7	-	P13,4
I	Integer	2 or 4	0	-	I2,0
F	Floating point		0	-	I4,0
B	Binary		0	-	B23,0
D	Date	6	0	-	D6
T	Time	12	0	-	T12
L	Logical (unsupported)				

- *l* is an integer denoting the length/scale of the field. For numeric and packed numeric fields, *l* denotes the total number of digits of the field that is, the sum of the digits left and right of the decimal point. The Natural format N7.3 is, for example, represented by N10.3. See also the table above.

- *p* is an integer denoting the precision of the field. It is usually 0, except for numeric and packed fields where it denotes the number of digits right of the decimal point. See also the table above.

- *d1* is also an integer denoting the occurrences of the alphanumeric array (alphanumeric only). See also the table above.

This descriptive/control parameter is invisible to the calling Natural program and to the called Natural stored procedure, but it has to be defined in the parameter definition of the stored procedure row with the CREATE PROCEDURE statement and the DB2 PARAMETER STYLE GENERAL/WITH NULL.

The following table shows the number of parameters which have to be defined with the CREATE PROCEDURE statement for a Natural stored procedure defined with PARAMETER STYLE GENERAL depending on the number of user parameters and whether the client (i.e. the caller of a stored procedure for DB2) and the server (i.e. the stored procedure for DB2) is written in Natural or in another standard programming host language. *n* denotes the number of user parameters.

Client\Server	**Natural**	**not Natural**
Natural	*n* + 1	*n* (CALLMODE=NONE)
non-Natural	*n* + 1	*n*

DB2SQL

PARAMETER DB2SQL applies to Natural stored procedures and Natural UDFs.

A Natural stored procedure or Natural UDF with PARAMETER STYLE DB2SQL first receives the user parameters specified and then the parameters listed below, under Additional Parameters passed. For a Natural UDF, the input parameters are passed before the output parameters.

Additional Parameters passed:

- A NULL indicator for each user parameter of the CALL statement,
- The SQLSTATE to be returned to DB2,
- The qualified name of the Natural stored procedure or UDF,
- The specific name of the Natural stored procedure or UDF,
- The SQL DIAGNOSE field with a diagnostic string to be returned to DB2.

The SQLSTATE, the qualified name, the specific name and the DIAGNOSE field are defined in the Natural parameter data area (PDA) DB2SQL_P which is supplied in the Natural system library SYSDB2.

If the optional feature SCRATCHPAD *nnn* is specified additionally in the CREATE FUNCTION statement for the Natural UDF, the SCRATCHPAD storage parameter is passed to the Natural UDF.

Use the following definitions:

```
01 SCRATCHPAD A(4+nnn)
01 REDEFINE SCRATCHPAD
02 SCRATCHPAD_LENGTH(I4)
02 ...
```

Redefine the SCRATCHPAD in the Natural UDF according to your requirements.

The first four bytes of the SCRATCHPAD area contain an integer length field. Before initially invoking the Natural UDF with an SQL statement, DB2 resets the SCRATCHPAD area to x’00’ and sets the size *nnn* specified for the SCRATCHPAD into the first four bytes as an integer value.

Thereafter, DB2 does not reinitialize the SCRATCHPAD between the invocations of the Natural UDF for the invoking SQL statement. Instead, after returning from the Natural UDF, the contents of the SCRATCHPAD is preserved and restored at the next invocation of the Natural UDF.

Below is information on:

- Parameter CALL TYPE
- Parameter DBINFO
- Determining Library, Subprogram and Parameter Formats
- Invoking a Natural Stored Procedure
- Error Handling
- Lifetime of Natural Session
- Example of DB2SQL - Natural Stored Procedure
- Example of DB2SQL - Natural UDF

Parameter CALL TYPE

This parameter is optional and only applies to Natural UDFs.

The CALL TYPE parameter is passed if the FINAL CALL option is specified for a Natural scalar UDF, or if the Natural UDF is a table UDF. The CALL TYPE parameter is an integer indicating the type of call DB2 performs on the Natural UDF. See the DB2 SQL GUIDE for details on the parameter values provided in the CALL_TYPE parameter.

Parameter DBINFO

This parameter is optional.

If the option DBINFO is used, the DBINFO structure is passed to the Natural stored procedure or UDF. The DBINFO structure is described in the Natural PDA DBINFO_P supplied in the Natural system library SYSDB2.

Determining Library, Subprogram and Parameter Formats

The NDB server stub determines the subprogram and the library from the qualified and specific name of the Natural stored procedure or UDF. The SCHEMA name is used as library name, and the procedure or function name is used as subprogram name.

The ROUTINEN subprogram is supplied in the Natural system library SYSDB2. This subprogram is used to access the DB2 catalog to determine the formats of the user parameters defined for the Natural stored procedure or UDF. After the formats have been determined, they are stored in the Natural buffer pool. During subsequent invocations of the Natural stored procedure, the formats are then retrieved from the Natural buffer pool. This requires that at least READS SQL DATA is specified for Natural stored procedures or UDFs with PARAMETER STYLE DB2SQL.

The ROUTINEN subprogram is generated statically. The DBRM of ROUTINEN is bound as package in the COLLECTION SAGNDBROUTINENPACK. Before starting to access the DB2 catalog, the subprogram will save the CURRENT PACKAGESET and set SAGNDBROUTINENPACK to CURRENT PACKAGESET. After processing, the ROUTINEN subprogram will restore the CURRENT PACKAGESET saved.

Invoking a Natural Stored Procedure

If the caller of the Natural stored procedure with PARAMETER STYLE DB2SQL is a Natural program, the caller must use the CALLDBPROC statement with the specification CALLMODE=NONE, which is the default.

Error Handling

If a Natural runtime error occurs during the execution of a Natural stored procedure or UDF with PARAMETER STYLE DB2SQL, SQLSTATE is set to 38N99 and the diagnostic string contains the text of the Natural error message.

If an error occurs in the NDB server stub during the execution of the Natural stored procedure or UDF with PARAMETER STYLE DB2SQL, the SQLSTATE is set to 38S99 and the diagnostic string contains the text of the error message.

If the application wants to raise an error condition during the execution of a Natural stored procedure or UDF, the SQLSTATE parameter must be set to a value other than ’00000’. See the DB2 SQL Guide for specifications of user errors in the SQLSTATE parameter.

Additionally, a text describing the errors can be placed in the DIAGNOSE parameter.

If a Natural table UDF wants to signal to DB2 that it has found no row to return, ’02000’ must be returned in the SQLSTATE parameter.

Lifetime of Natural Session

For a Natural UDF that contains the attributes DISALLOW PARALLEL and FINAL CALL, the NDB server stub retains the Natural session allocated earlier. This Natural session will then be reused by all subsequent UDF invocations until Natural encounters the final call.

Example of DB2SQL - Natural Stored Procedure

In a Natural stored procedure, define the parameters as shown in the example program below:

```
DEFINE DATA PARAMETER
01 P1 ...
01 P2 ...
...
...
01 PN ...
01 N1 (I2)
01 N2 (I2)
...
...
01 N
n (I2)
PARAMETER USING DB2SQL_P
[ PARAMETER USING DBINFO_P ]   /*  only if DBINFO is defined
```

```
LOCAL
...
...
END-DEFINE
```

Example of DB2SQL - Natural UDF

In a Natural UDF, define the parameters as shown in the example program below:

```
DEFINE DATA PARAMETER
01 PI1 ...  /* first input parameter
01 PI2 ...
...
...
01 PIn ...  /*  last input parameter
01 RS1...  /* first result parameter
...
...
01 RSn ...  /* last result parameter
01 N_PI1 (I2)  /* first NULL indicator
01 N_PI2 (I2)
...
...
01 N_Pin (I2)
01 N_RS1 (I2)
...
...
01 N_RSn (I2) /* last NULL indicator
PARAMETER USING DB2SQL_P /* function, specific, sqlstate, diagnose
PARAMETER
01 SCRATCHPAD A(4+nnn)  /* only if SCRATCHPAD nnn is specified
  01 REDEFINES SCRATCHPAD
02 SCRATCHPAD_LENGTH (I4)
02 ...
01 CALL_TYPE (I4) /* --- only if FINAL CALL is specified or table UDF


PARAMETER USING DBINFO_P   /* ---- only if DBINFO is specified
LOCAL
...
...
END-DEFINE
```