Installing Natural for SQL/DS

This section describes step by step how to install the Natural interface to SQL/DS (in the remainder of this section also referred to as NSQ).

This section covers the following topics:

- Installation Jobs
- Using System Maintenance Aid
- Prerequisites
- Installation under CMS
- Installation under z/VSE
- Installation Verification
- Natural Parameter Modification for SQL/DS
- Parameter Module NDBPARM

Notation vrs or vr: If used in the following document, the notation *vrs* or *vr* stands for the relevant version, release, system maintenance level numbers. For further information on product versions, see Version in the *Glossary*.

Installation Jobs

The installation of Software AG products is performed by installation "jobs". These jobs are either created "manually" or generated by System Maintenance Aid (SMA).

For each step of the installation procedures described later in this section, the job number of a job performing the respective task is indicated. This job number refers to an installation job generated by SMA. If you are not using SMA, an example job of the same number is provided in the job library on the NSQ installation tape; you must adapt this example job to your requirements. That the job numbers on the tape are preceded by a product code (for example, `NSQI070`).

Using System Maintenance Aid

For information on using Software AG’s System Maintenance Aid for the installation process, refer to the *System Maintenance Aid* documentation.

Prerequisites

- Base Natural must be installed first; you cannot install Natural and Natural for SQL/DS at the same time.

- The Software AG Editor must be installed (as described in the Natural *Installation* documentation).

Further product/version dependencies are specified under *Natural and Other Software AG Products* and *Operating/Teleprocessing Systems Required* in the current Natural *Release Notes*.

Installation under CMS

This section only applies to the installation of NSQ under CMS.

Installation Tape

The installation tape was created under z/OS; it has standard z/OS labels and headers. It contains the datasets listed in the table below. The sequence of the datasets is shown in the *Report of Tape Creation* which accompanies the installation tape.

Dataset Name	Contents
NSQ*vrs*.TAPE	NSQ source modules, load modules and installation EXECs. This dataset is in TAPE DUMP format and must be loaded onto the installation minidisk.
NSQ*vrs*.INPL	NSQ utility programs in INPL format.
NSQ*vrs*.ERRN	NSQ error messages.

The notation *vrs* in dataset names represents the version number of the product.

Copying the Tape Contents to Disk

The tape file NSQ*nnn*.TAPE was created with the CMS TAPE DUMP facility. Load the contents of the tape to your A-disk. The free space should be at least 450 4-KB blocks; for example, 3 cylinders on 3350 or 3380 disks.

Ask the system operator to attach a tape drive to your virtual machine at address X'181' and mount the NSQ installation tape.

To position the tape for the TAPE LOAD command, calculate the number of tape marks as follows: If the sequence number of NSQ*nnn*.TAPE - as shown by the *Report of Tape Creation* - is *n*, you must position over 3*n*-2 tape marks; that is, FSF 1 for the first dataset, FSF 4 for the second, etc.

Position the tape by issuing the CMS command:

```
TAPE FSF fsfs
```

where *fsfs* is calculated as described above.

Load the NSQ installation material by issuing the CMS command:

```
TAPE LOAD * * A
```

You may wish to keep the tape drive attached to your virtual machine, because the tape is still needed in Step 7 of the installation procedure.

Preparing the Installation

Perform the following steps to prepare the installation of NSQ:

1. Ensure that the required SQL/DS database machine is activated in multiple- user mode and that the user machine for this installation is properly configured and initialized to access the SQL/DS database machine.
2. All precompilations as well as NSQ itself take advantage of the implicit CONNECT mechanism provided by VM. Therefore, ensure that the VM user ID is authorized for SQL/DS.
3. Ensure that your user machine has access to the following minidisks: the SQL/DS production minidisk, the Natural installation minidisk.
4. Ensure that the Adabas environment for your user machine is set up.

Concerning the following installation steps, also refer to the section *Installing Natural under VM/CMS* in the Natural *Installation* documentation.

Installation Procedure

Perform the following steps to install NSQ:

Step 1: Generate the NSQ I/O module NDBIOMO

Generate `NDBIOMO` by using the command:

```
GENIOMO SQL/DS n
```

GENIOMO generates the assembly source for `NDBIOMO` from the existing source `NDBIOTM`. It prompts you for the Natural/CMS batch module and invokes the Natural program `NDBGENI`, which is loaded with `INPL` during the base Natural installation.

`GENIOMO` is invoked with the following two parameters:

- the DB-environment parameter, which must be set to SQL/DS,
- the parameter *n* to specify the number of statements for dynamic access; the default value is `10`.

`NDBIOMO` performs the dynamic access to SQL/DS and contains all necessary `EXEC` SQL statements. In addition, it contains some special SQL statements which cannot be executed in dynamic mode.

An output report is created by this job and should be checked for successful completion. In addition, a condition code of 0 indicates normal completion.

Step 2: Precompile and assemble NDBIOMO

Precompile and assemble `NDBIOMO` using the command:

```
NDBIOMO
```

Note:
Since no precompiler options are specified, the default SQL/DS isolation level Repeatable Read may lead to locking problems, because all SQL/DS locks are held until the end of the transaction. Thus, depending

on your application, it may be necessary to specify a different isolation level.

Step 3: Modify and assemble the NSQ parameter module NDBPARM

Assemble NDBPARM using the command:

```
NDBPARM
```

The NSQ parameter module contains the macro NDBPRM, which contains parameters specific to the Natural interface to SQL/DS.

You can generally use the default values for all parameters. Modify only the values of the parameters whose default values do not suit your requirements.

The individual parameters are described in the section *Parameter Module NDBPARM*.

Step 4: Modify NATPARM

Adapt your Natural parameter module NATPARM by adding parameters specific to Natural for SQL/DS as described in the section *Natural Parameter Modification for SQL/DS*.

Step 5: Modify NAT$LOAD LOADLIST

Edit the member NAT$LOAD EXEC provided on the Natural/CMS installation tape and add the following line to the existing LOADLIST statements:

```
LOADLIST = LOADLIST 'NDBNUC NDBNSQ NDBPARM NDBIOMO ARIRVSTC'
```

Step 6: Generate a Natural module

Generate the Natural/CMS load module using the command:

```
NATBLDM
```

NATBLDM is provided on the Natural CMS/installation tape and prompts you for the name of the Natural nucleus and generates the executable Natural module.

Step 7: Load Natural objects and error messages into system file

In this step, the NSQ system programs, maps and DDMs (dataset NSQ*vrs*.INPL) and the NSQ error messages file (dataset NSQ*vrs*.ERRN) are loaded into the Natural system file.

If the tape drive used when copying the contents of the installation tape to disk was detached from your virtual machine, ask the system operator to attach a tape drive to your virtual machine at address X'181' and mount the Natural installation tape.

Issue the following command:

```
NSQINPL
```

You are prompted for the name of the command to invoke Natural. Enter the name of the Natural module generated in the preceding step.

NSQINPL then positions the tape and loads the Natural objects and error messages.

The INPL job loads objects into the libraries SYSDDM, SYSTEM and SYSSQL.

The ERRLODUS job loads error messages into the library SYSERR.

The NSQ system programs and error messages *must* be loaded into the FNAT system file

Warning:
Ensure that your newly created SYSSQL library contains all necessary Predict interface programs, which are loaded into SYSSQL when installing Predict (see the relevant Predict documentation).

Installation under z/VSE

Under z/VSE, Natural for SQL/DS basically consists of two parts:

1. An environment-independent nucleus, which can be linked to a shared Natural nucleus and loaded in the shared virtual area (SVA) of the operating system.
2. Environment-dependent components, which must be linked to the appropriate Natural environment-dependent interface.

This section covers the following topics:

- Installation Tape
- Copying the Tape Contents to a z/VSE Disk
- Installation Procedure

Installation Tape

The installation tape contains the datasets listed in the table below. The sequence of the datasets is shown in the Report of Tape Creation which accompanies the installation tape.

Dataset Name	Contents
NSQ*vrs*.LIBR	LIBR backup file.
NSQ*vrs*.INPL	NSQ utility programs in INPL format.
NSQ*vrs*.ERRN	NSQ error messages.

The notation *vrs* in dataset names represents the version number of the product.

Copying the Tape Contents to a z/VSE Disk

If you are using SMA, refer to the *System Maintenance Aid* documentation (included in the current edition of the Natural documentation CD).

If you are *not* using SMA, follow the instructions below.

This section explains how to:

- Copy dataset COPYTAPE.JOB from tape to disk.
- Modify this dataset to confom with your local naming conventions.

The JCL in this member is then used to copy all datasets from tape to disk.

If the datasets for more than one product are delivered on the tape, the member COPYTAPE.JOB contains the JCL to unload the datasets for all delivered products from the tape to your disk, except the datasets that you can directly install from tape, for example, Natural INPL objects.

After that, you will have to perform the individual install procedure for each component.

- Step 1 - Copy Dataset COPYTAPE.JOB from Tape to Disk
- Step 2 - Modify COPYTAPE.JOB
- Step 3 - Submit COPYTAPE.JOB

Step 1 - Copy Dataset COPYTAPE.JOB from Tape to Disk

The dataset COPYTAPE.JOB contains the JCL to unload all other existing datasets from tape to disk. To unload COPYTAPE.JOB, use the following sample JCL:

```
* $$ JOB JNM=LIBRCAT,CLASS=0,                                          +
* $$ DISP=D,LDEST=(*,UID),SYSID=1
* $$ LST CLASS=A,DISP=D
// JOB LIBRCAT
* ***************************************
*     CATALOG COPYTAPE.JOB TO LIBRARY
* ***************************************
// ASSGN SYS004,nnn                              <------  tape address
// MTC REW,SYS004
// MTC FSF,SYS004,4
ASSGN SYSIPT,SYS004
// TLBL IJSYSIN,'COPYTAPE.JOB'
// EXEC LIBR,PARM='MSHP; ACC S=lib.sublib'       <------- for catalog
/*
// MTC REW,SYS004
ASSGN SYSIPT,FEC
/*
/&
* $$ EOJ
```

where:

nnn is the tape address
lib.sublib is the library and sublibrary of the catalog

Step 2 - Modify COPYTAPE.JOB

Modify COPYTAPE.JOB to conform to your local naming conventions and set the disk space parameters before submitting this job.

Step 3 - Submit COPYTAPE.JOB

Submit `COPYTAPE.JOB` to unload all other datasets from the tape to your disk.

Installation Procedure

The following steps describe the procedure for installing the components of NSQ.

Step 1: Generate the NSQ I/O Module NDBIOMO

Job I055, Step 1600

By executing a standard Natural batch job, this step generates the assembly source for `NDBIOMO` from the member `NDBIOTM`.

This batch job invokes the Natural program `NDBGENI`, which is loaded `INPL` during the base Natural installation. `NDBGENI` contains the following two parameters, which can be modified to meet your specific requirements:

- the DB-environment parameter, which must be set to SQL/DS,
- the parameter to specify the number of statements for dynamic access.

`NDBIOMO` performs the dynamic access to SQL/DS and contains all necessary `EXEC` SQL statements (see further information on `NDBIOMO` in the section *Internal Handling of Dynamic Statements*). In addition, it contains some special SQL statements which cannot be executed in dynamic mode.

An output report is created by this job and should be checked for successful completion. In addition, a condition code of 0 indicates normal completion.

Step 2: Precompile and Assemble NDBIOMO

Job I055, Steps 1610 and 1620

Precompile (using the SQL precompiler) and assemble `NDBIOMO`. Ensure that an appropriate SQL/DS user ID and password is specified for precompiling.

Note:
Since no precompiler options are specified, the default SQL/DS isolation level Repeatable Read may lead to locking problems, because all SQL/DS locks are held until the end of the transaction. Thus, depending on your application, it may be necessary to specify a different isolation level.

Step 3: Modify and Assemble the NSQ Parameter Module NDBPARM - Job I055, Step 1640

The NSQ parameter module contains the macro `NDBPRM` with parameters specific to the Natural interface to SQL/DS.

You can generally use the default values for all parameters. Modify only the values of the parameters whose default values do not suit your requirements.

The individual parameters are described in the section *Parameter Module NDBPARM*.

Step 4: Modify and Reassemble NATPARM

Adapt your Natural parameter module NATPARM by adding parameters specific to Natural for SQL/DS and reassemble NATPARM.

Step 5: Relink your Natural Nucleus

Modify the JCL used to link your Natural nucleus by adding the following INCLUDE cards and the corresponding DLBL statements:

INCLUDE NDBNUC	Environment-independent NSQ nucleus
INCLUDE NDBNSQ	Environment-independent SQL/DS interface
INCLUDE NDBPARM	NSQ parameter module created in Step 3
INCLUDE NDBIOMO	NSQ I/O module created in Step 1
INCLUDE *xxxxxxx*	Environment-dependent SQL/DS interface (see below)

Depending on your environment(s), INCLUDE the appropriate environment-specific language interface *xxxxxxxx* as shown in the following table:

Interface	Environment
ARIPRDID	In batch mode
ARIRRTED	Under CICS

Note:
If you want to use NSQ in both environments, repeat this step for each of these environments.

Instead of link-editing your Natural nucleus in the way described above, you have the following alternatives:

1. If you use a shared Natural nucleus, only include NDBNUC and NDBNSQ in the link-edit of this nucleus. All other modules must be included in the link-edit of your Natural environment-dependent nucleus.

2. Remove NDBNUC and NDBNSQ from the link-edit of the Natural nucleus and link-edit them as a separate module with the mandatory *entry* name NATGWDB2. The *name* of the resulting phase is arbitrary. However, if you use a name different from NATGWDB2, this name must be specified as an alias name in an NTALIAS macro entry of the Natural parameter module. This way of link-editing only applies if the Natural Resolve CSTATIC Addresses feature (RCA) is used.

3. Include all modules in the link-edit job of a separate Natural parameter module with the mandatory *entry* name CMPRMTB. The *name* of the resulting phase is arbitrary. This way of link-editing only applies if an alternative parameter module (PARM profile parameter) is used.

4. If link-editing is done in this way, you can install NSQ without having to modify your Natural nucleus or driver.

If link-editing is done according to number 2. or 3., the following applies:

Under CICS: _ the resulting module must be defined via a PPT entry or RDO:

```
DFHPPT TYPE=ENTRY,PROGRAM=module-name,PGMLANG=ASSEMBLER
```

Step 6: Load Natural Objects Into System File

Job I061, Step 1600

In this step, the NSQ system programs, maps and DDMs are loaded into the Natural system files. The `INPL` job loads objects into the libraries `SYSDDM`, `SYSTEM` and `SYSSQL`.

The NSQ system programs *must* be loaded into the `FNAT` system file.

Warning:
Ensure that your newly created `SYSSQL` library contains all necessary Predict interface programs, which are loaded into `SYSSQL` when installing Predict (see the relevant Predict documentation).

Step 7: Load Natural Error Messages into System File

Job I061, Step 1620

This step executes a batch Natural job that runs an error load program using the `NSQ` *nnn*`.ERRN` dataset as input. The `ERRLODUS` job loads error messages into the library `SYSERR` in the `FNAT` system file.

The NSQ error messages *must* be loaded into the Natural `FNAT` system file.

Installation Verification

This section covers the following topics:

- Prepare your SQL/DS Environment
- Online Verification Methods
- Sample Batch Verification Job - z/VSE only

Prepare your SQL/DS Environment

As all dynamic access to SQL/DS is performed by `NDBIOMO`, all NSQ users must have `RUN` privilege on `NDBIOMO`.

Online Verification Methods

To verify the installation of the Natural interface to SQL/DS online, you can use either of the following methods:

- SQL Services
- DEM2* Sample Programs

SQL Services

Perform the following steps to verify and check the installation of NSQ using the SQL Services of the Natural utility SYSDDM.

1. Invoke Natural.
2. Invoke SYSDDM.

   On the SYSDDM main menu enter function code B to invoke the SQL Services function.

   Enter function code S to select all SQL/DS tables.

   The communication between Natural and SQL/DS works if all existing SQL/DS tables are displayed.
3. For one of the tables, generate a Natural DDM as described in the section *Generate DDM from an SQL Table*.

   To enable SYSDDM to generate a DDM, the Natural administrator requires access to the following SQL/DS tables:

SYSTEM.SYSCATALOG
SYSTEM.SYSCOLUMNS
SYSTEM.SYSINDEXES
SYSTEM.SYSVIEWS
SYSTEM.SYSSYNONYMS
SYSTEM.SYSUSAGE

4. After you have generated a DDM, access the corresponding SQL/DS table with a simple Natural program:

   Example:

```
FIND view-name WITH field = value
  DISPLAY field
LOOP
END
```

5. If you receive the message SYSFUL 3700, enter the command SQLERR to display the corresponding SQL return code. See the description of the SQLERR command.

DEM2* Sample Programs

To verify and test your installation you can also use the sample programs DEM2* in the library SYSSQL provided on the installation tape.

Using these sample programs, you can create an SQL/DS table using DEM2CREA and create the corresponding DDM via SYSDDM. You can then store data in the created table using DEM2STOR and retrieve data from the table using DEM2FIND or DEM2SEL. You can also drop the table using program DEM2DROP.

Sample Batch Verification Job - z/VSE only

To verify the installation of the Natural interface to SQL/DS, a sample batch verification job (Job I065) is provided. This step contains sample JCL and sample programs to test Natural with NSQ in batch mode.

The sample program DEM2CONN performs the connection to the database, which is required before you can run a Natural program that accesses SQL/DS. DEM2CONN calls the DB2SERV module with function code U which in turn calls the database connect services.

Sample program DEM2JOIN performs a JOIN combining information from SQL/DS SYSTEM.SYSDBSPACE and SYSTEM.SYSCATALOG.

Natural Parameter Modification for SQL/DS

This section covers the following topics:

- DB2SIZE Parameter
- NTDB Macro
- Performance Considerations for the DB2SIZE Parameter

DB2SIZE Parameter

Add the following Natural profile parameter to your NATPARM module:

```
DB2SIZE=nn
```

The DB2SIZE parameter can also be specified dynamically. It indicates the size of the SQL/DS buffer area, which should be set to at least 6 KB.

The setting of DB2SIZE can be calculated according to the following formula:

```
((808 + n1     * 40 + n2       * 100) + 1023) / 1024 KB
```

The variables *n1* and *n2* correspond to:

n1	the number of statements for dynamic access as specified as the second parameter in job I055, step 1600 (under z/VSE).
n2	the maximum number of nested database loops as specified with the MAXLOOP parameter in NDBPARM.

Note:
Ensure that you have also added the Natural parameters required for the Software AG Editor (see *Installing the Software AG Editor* in the Natural *Installation* documentation).

Since DB2SIZE applies to Natural for SQL/DS and Natural for DB2, it should be set to the maximum value if you run more than one of these environments.

NTDB Macro

Add an NTDB macro for database type SQL specifying the list of logical database numbers that relate to SQL/DS tables. All Natural DDMs that refer to an SQL/DS table must be cataloged with a DBID from this list.

DBIDs can be any number from 1 to 254; a maximum of 254 entries can be specified. For most user environments, one entry is sufficient.

Note:
Ensure that all NSQ DDMs used when cataloging a given program have a valid SQL/DS DBID. Also ensure that the DBIDs selected in the NTDB macro for SQL/DS do not conflict with DBIDs selected for other database systems.

The DBID for SQL/DS used when cataloging a Natural program does not have to be in the NTDB list of DBIDs used when executing this program. Therefore, when executing existing Natural programs, DBID 250 is not mandatory.

Two sample NTDB macros follow:

```
NTDB SQL,250
```

```
NTDB SQL,(200,250,251)
```

Performance Considerations for the DB2SIZE Parameter

During execution of an SQL statement, storage is allocated dynamically to build the SQLDA for passing the host variables to SQL/DS.

In previous Natural for SQL/DS versions, this storage was always obtained from the TP monitor or operating system. For performance reasons, it is now first attempted to meet the storage requirements by free space in the Natural for SQL/DS buffer (DB2SIZE). Only if there is not enough space available in this buffer, the TP monitor or operating system is invoked.

To take advantage of this performance enhancement, you must specify your DB2SIZE larger than calculated according to the formula. The additional storage requirements (in bytes) can be calculated as follows:

- With sending fields:

  `64 + n * 56`

  where *n* is the number of sending fields in an SQL statement.

  The storage is freed immediately after the execution of the SQL statement.

- With receiving fields (that is, with variables of the INTO list of a SELECT statement):

  `64 + n * 56 + 24 + n * 2`

  where *n* is the number of receiving fields in an SQL statement.

The storage remains allocated until the loop is terminated.

Example:

If you use the default value `10` for both variables (*n1* and *n2*), the calculated `DB2SIZE` will be 2200 bytes. However, if you specify a `DB2SIZE` of 20 KB, the available space for dynamically allocated storage will be 18272 bytes, which means enough space for up to either 325 sending fields or 313 receiving fields.

As space for receiving fields remains allocated until a database loop is terminated, the number of fields that can be used inside such a loop is reduced accordingly: for example, if you retrieve 200 fields, you can update about 110 fields inside the loop.

Note:
When using `VARCHAR` fields (that is, fields with either an accompanying `L@` field in the Natural view or an explicit `LINDICATOR` clause), additional storage is allocated dynamically if the `L@` or `LINDICATOR` field is not specified directly in front of the corresponding base field. Therefore, always specify these fields in front of their base fields.

Parameter Module NDBPARM

The source module `NDBPARM` is used in several Natural add-on products. It contains parameter macros specific to an SQL environment:

- `NDBPRM`
- `NDBID`

These macros are described below.

Parameter Macro NDBPRM

The default values of the parameters contained in this macro can be modified to meet site-specific requirements (see the corresponding step of the *Installation Procedure*). The values of the parameters cannot be dynamically overwritten.

Complete List of Parameters Contained in NDBPRM

Below is a description of all parameters contained in the `NDBPRM` macro:

`BTIGN` | `CONVERS` | `CONVRS2` | `DDFSERV` | `DELIMID` | `EBPFSRV` | `EBPPRAL` | `EBPSEC` | `EBPMAX` | `ETIGN` | `FSERV` | `MAXLOOP` | `NNPSF` | `PSCIGN` | `REFRESH` | `RETRYPO` | `RWRDONL` | `STATDYN`

List of Parameters Applicable to Natural for SQL/DS

The following parameters in the `NDBPRM` parameter macro are relevant to Natural for SQL/DS. All other parameters contained in the module are ignored.

`BTIGN` | `CONVERS` | `CONVRS2` | `DELIMID` | `MAXLOOP` | `PSCIGN` | `REFRESH` | `RWRDONL` | `STATDYN`

BTIGN - Ignore BACKOUT TRANSACTION Error

BTIGN is used to ignore the error which results from a BACKOUT TRANSACTION statement that was issued too late for backing out the current transaction, because an implicit Syncpoint has previously been issued by the TP monitor.

Possible Values:

Value	Explanation
ON	The error after a late BACKOUT TRANSACTION is ignored. This is the default value.
OFF	The error after a late BACKOUT TRANSACTION is *not* ignored.

CONVERS - Conversational Mode under CICS

This parameter is used to allow conversational mode in CICS environments.

Possible Values:

Value	Explanation
ON	Conversational mode is allowed. This is the default value.
OFF	Conversational mode is *not* allowed.

If this parameter is set to OFF and no Natural file server is used, you cannot continue database loops across terminal I/Os; if so, the DB2 SQL codes -501, 504, 507, 514, or 518 may occur.

If you use SYSDDM SQL Services in a CICS environment, specify CONVERS=ON, otherwise the aforementioned errors could occur. See also the section *SQL Services*.

CONVRS2 - Allow Conversational Mode 2 under CICS

This parameter is used to allow conversational mode 2 in CICS environments.

Possible Values:

Value	Explanation
ON	Conversational mode 2 is allowed.
OFF	Conversational mode 2 is *not* allowed. This is the default value.

This parameter is used to control conversational mode 2 in CICS environments. Conversational mode 2 means that update transactions are spawned across terminal I/Os until either an explicit COMMIT or explicit ROLLBACK has been issued (Caution: DB2 and CICS resources are kept across terminal I/Os!). This means CONVRS2=ON 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.

See also CALLNAT subprogram NDBCONV, which allows setting or resetting conversational mode 2 dynamically.

DDFSERV - Alternate DD Name for Natural File Server

This parameter specifies a DD name for the Natural file server module other than CMFSERV.

Possible Values:

Value	Explanation
DD-name	Any valid DD name. There is no default value.

DELIMID - Escape Character for Delimited Identifiers

This parameter determines the escape character to be used for generating delimited SQL identifiers for the column names and table names in SQL statements. A delimited identifier is a sequence of one or more characters enclosed in escape characters. You must specify a delimited identifier if you use SQL-reserved words for column names and table names, as demonstrated in the *Example of DELIMID* below.

Possible Values:

Value	Explanation
"	Double quotation mark
'	Single quotation mark
None	No value: Delimited identifiers are not enabled. This is the default value.

To enable generation of delimited identifiers, DELIMID must be set to double quotation mark (") or single quotation mark (’).

The escape character specified for DELIMID and the SQL STRING DELIMITER are mutually exclusive. This implies that the mark (double or single quotation) used to enclose alphanumeric strings in SQL statements must be different from the value specified for DELIMID. If you enable delimited identifiers, ensure that the value specified for DELIMID also complies with the SQL STRING DELIMITER value of your DB2 installation.

See also the RWRDONL parameter to determine which delimited identifiers are generated in the SQL string.

Example of DELIMID:

In the following example, a double quotation mark (") has been specified as the escape character for the delimited identifier:

Natural statement:

```
SELECT FUNCTION INTO #FUNCTION FROM XYZ-T1000
```

Generated SQL string:

```
SELECT "FUNCTION" FROM XYZ.T1000
```

EBPFSRV - Editor Buffer Pool for Natural File Server

Note:
This parameter does not apply to Natural for SQL/DS and is ignored.

This parameter is used to determine whether the Natural file server uses the Software AG Editor buffer pool as the storage medium.

Possible Values:

Value	Explanation
ON	The Software AG buffer pool is to be used as the storage medium for the Natural file server. ON *must* be set if the file server is to be used in a Parallel Sysplex environment. In this case, your Natural session must use the auxiliary editor buffer pool (see also *Support of a z/OS Parallel Sysplex Environment* in the *Installation* documentation).
OFF	A VSAM file is to be used as the storage medium for the Natural file server. This is the default value.

EBPPRAL - Editor Buffer Pool Primary Allocation

Note:
This parameter does not apply to Natural for SQL/DS and is ignored.

This parameter specifies the number of blocks to be allocated primarily to each user of the Natural file server, if the Software AG Editor buffer pool is used as the storage medium.

Possible Values:

Value	Explanation
0 - 32676	Number of blocks to be allocated primarily.
20	This is the default value.

If the EBPFSRV parameter is set to OFF, EBPPRAL is not used at runtime.

EBPSEC - Editor Buffer Pool Secondary Allocation

Note:
This parameter does not apply to Natural for SQL/DS and is ignored.

This parameter specifies the number of blocks to be allocated secondarily to each user of the Natural file server if the Software AG Editor buffer pool is used as the storage medium. The secondary allocation is used to allocate buffer pool blocks to the user if the primary allocation amount is already exhausted.

Possible Values:

Value	Explanation
`0 - 32676`	Number of blocks to be allocated secondarily.
`10`	This is the default value.

If the `EBPFSRV` parameter is set to `OFF`, `EBPSEC` is not used at runtime.

EBPMAX - Editor Buffer Pool Maximum Allocation

Note:
This parameter does not apply to Natural for SQL/DS and is ignored.

This parameter specifies the maximum number of blocks to be allocated to each user of the Natural file server if the Software AG Editor buffer pool is used as the storage medium. This parameter serves as upper limit for the allocation of buffer pool blocks to a single user.

Possible Values:

Value	Explanation
`0 - 32676`	Maximum number of blocks to be allocated.
`100`	This is the default value.

If the `EBPFSRV` parameter is set to `OFF`, `EBPMAX` is not used at runtime.

ETIGN - Ignore END TRANSACTION Error

Note:
This parameter does not apply to Natural for SQL/DS and is ignored.

This parameter is relevant in IMS TM MPP and message-oriented BMP environments only.

It is used to handle `END TRANSACTION` statements in a message-driven IMS region (MPP or message-oriented BMP).

In such a region, an `END TRANSACTION` cannot be executed by the Natural/IMS interface and is therefore ignored without any notification. In such situations, the `ETIGN` parameter can be used to issue an error message instead.

Possible Values:

Value	Explanation
`ON`	The `END TRANSACTION` error is ignored and processing is continued. This is the default value.
`OFF`	The `END TRANSACTION` error is *not* ignored.

FSERV - Activate Natural File Server

Note:
This parameter does not apply to Natural for SQL/DS and is ignored.

This parameter determines whether the Natural file server is to be used and whether it can be disabled in the case of an initialization error.

Possible Values:

Value	Explanation
`ON`	Natural file server is to be used.
`OFF`	Natural file server is not to be used. This is the default value.
`DIS`	Natural file server is to be used but is to be disabled if it cannot be initialized.

If `FSERV` is set to `ON` and the file server is not operational, the initialization of the Natural SQL Gateway is terminated with a corresponding Natural error message. The Natural SQL Gatewayis disabled and any SQL call is rejected with a corresponding error message.

MAXLOOP - Maximum Number of Nested Program Loops

This parameter specifies the maximum possible number of nested database loops accessing SQL databases.

Possible Values:

Value	Explanation
`1 - 99`	Maximum possible number of nested database loops.
`10`	This is the default value.

NNPSF - Set Natural Numerics' Positive Sign to F

Note:
This parameter does not apply to Natural for SQL/DS and is ignored.

This parameter changes the sign character of positive Natural variables which have format N, if they are filled from the SQL database system. Usually these variables have the `C` as positive sign character. If the parameter `NNPSF` is set to `ON`, `F` is used as positive sign character.

Possible Values:

Value	Explanation
`ON`	Positive numbers put into Natural numeric variables by the SQL database system get the sign `F`.
`OFF`	Positive numbers put into Natural numeric variables by the SQL database system remain unchanged. This is the default value.

PSCIGN - Treat Positive Sqlcodes as Slqcode 0

This parameter influences the treatment of positive sqlcodes returned from the SQL database system. If the parameter `PSCIGN` is set to `OFF`, a NAT3700 error message is issued. If the parameter `PSCIGN` is set to `ON`, positive sqlcodes are treated as if they were zero, that is, no NAT3700 error message is issued.

Possible Values:

Value	Explanation
`ON`	Positive sqlcodes are treated as zero.
`OFF`	Positive sqlcodes cause a NAT3700 error message. This is the default value.

REFRESH - Refresh Setting of DB2 Server and Package Set

This parameter is used to automatically set the DB2 server and package set to the values that applied when the last transaction was executed.

To refresh the server connection, use the following SQL statement of DB2:

```
CONNECT ? IDENTIFIED BY ? TO ?
```

Possible Values:

Value	Explanation
`ON`	An automatic refresh is performed every time before a database transaction starts.
`OFF`	No automatic refresh is performed. This is the default value.

RETRYPO - Number of Positioning Retries

Note:
This parameter does not apply to Natural for SQL/DS and is ignored.

This parameter delimits the number of retries in order to reposition a dynamic scrollable cursor in a pseudo-conversational environment (IMS MPP or CICS).

Possible Values:

Value	Explanation
`0 - 2147483648`	Number of retries.
`10`	This is the default value.

This parameter applies only for dynamic scrollable cursors.

In pseudo-conversational environments, cursors are closed at terminal I/O. For dynamic scrollable cursors the current absolute position number and the current key column values are saved. After terminal I/O the dynamic scrollable cursor is opened again and positioned absolutely to the position of the saved absolute position. The contents of the key columns are compared with the saved values. If they match, processing continues with the next requested database operation.

If the contents of the key columns do not match the saved values, the next rows are fetched and compared with the saved values until either the values match or no row is found or the RETRYPO count is exhausted. In the latter cases the cursor is repositioned to the saved position and the prior rows are fetched and compared until either the values match or no row is found or the RETRYPO count is exhausted. In the latter cases a NAT3703 error message is issued. If a row is fetched whose key columns matches the saved values, processing continues with the next database instruction.

RETRYPO delimits the retries in each direction (*next* or *prior*).

If RETRYPO is zero no repositioning takes place.

RWRDONL - Generate Delimited Identifiers for Reserved Words Only

This parameter determines which identifiers are generated as delimited identifier in an SQL string. RWRDONL only takes effect if the setting of the DELIMID parameter allows delimited identifiers.

Possible Values:

Value	Explanation
ON	Only identifiers that are reserved words are generated as delimited identifiers. The list of reserved words is contained in the NDBPARM macro. This list has been merged from the lists of reserved words for DB2 for z/OS, DB2 for VSE/VM, DB2 for LINUX, OS/2, Windows and UNIX, and ISO/ANSI SQL99. This is the default value.
OFF	All identifiers are generated as delimited identifiers.

STATDYN - Allow Static to Dynamic Switch

This parameter is used to allow dynamic execution of statically generated SQL statements if the static execution returns an error.

Possible Values:

Value	Explanation
NEVER	Dynamic execution is never allowed. This is the default value.
ALWAYS	Dynamic execution is always allowed after an error.
SPECIAL	Dynamic execution is allowed after special errors only. These special errors are: • NAT3706: Load module not found • SQL -805: DBRM (database request module) does not exist in plan • SQL -818: Mismatch of timestamps

Parameter Macro NDBID

The parameter macro NDBID determines the database type of an SQL DBID.

The `NDBID` macro is specified as follows:

1. Default Database Definition

   The default database type is specified as follows. It applies to all database IDs not explicitly specified by `NDBID`.

   `NDBID=`*`database-type`*

2. Single Database Definition

   A single database ID and its type is specified as follows:

   `NDBID=`*`database-type`*`,`*`database-id`*

3. Multiple Database Definition

   Multiple database IDs of the same database type can be specified together, enclosed in parentheses:

   `NDBID=(`*`database-type`*`,`*`database-id1`*`,`*`database-id2`*`,...)`

database-type

Possible Values	Explanation
`DB2`	Databases are accessed via NDB. This is the default value.

database-id

Possible Values
`1-254`