



# NATURAL

---

**Natural for DL/I**  
Version 2.3.7



This document applies to Natural for DL/I Version 2.3.7 and to all subsequent releases. Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

© June 2002, Software AG  
All rights reserved

Software AG and/or all Software AG products are either trademarks or registered trademarks of Software AG. Other products and company names mentioned herein may be the trademarks of their respective owners.

# Table of Contents

|   |    |
|---|----|
| <b>Natural for DL/I - Overview</b>                                  | 1  |
| Natural for DL/I - Overview   | 1  |
| <b>General Information</b>  | 2  |
| General Information   | 2  |
| Basic Principles  | 2  |
| Accessing DL/I Data   | 3  |
| <b>Natural Parameter Modifications for DL/I</b>                     | 4  |
| Natural Parameter Modifications for DL/I                            | 4  |
| Parameters in NDLPARM   | 4  |
| DFBNUM - Maximum Entries in Translated Format Buffer                | 5  |
| DFFNUM - Maximum Fields in Single Entry of Translated Format Buffer | 5  |
| FLBNUM - Number of Entries in Fast Locate Buffer                    | 5  |
| INGSIZE - Initial Size of Buffer to Copy Parameter List             | 6  |
| INGOSIZ - Initial Size of I/O Area for DL/I Calls                   | 6  |
| INITCAL - Issues INIT Call at Transaction Start                     | 6  |
| PCBLEV - Maximum Number of PCB Levels                               | 6  |
| PCBNUM - Maximum Number of PCBs in a PSB                            | 7  |
| RELEVNT - Requests Relocation Event                                 | 7  |
| RESINDB - NDB Resident in Buffer Pool                               | 7  |
| RESINSB - NSB Resident in Buffer Pool                               | 7  |
| RESIUDF - UDF Resident in Buffer Pool                               | 8  |
| SASIZE - Size of Natural Save Area for DL/I                         | 8  |
| SEQNUM - Maximum Number of Nested Sequential Accesses               | 8  |
| SEQSSA - Maximum Size of an SSA                                     | 8  |
| THCSIZE - Table Size to Save Natural Field Values                   | 9  |
| TRACE - Trace Options   | 9  |
| TYPCHCK - Numeric/Packed Data Check                                 | 9  |
| TYPWARN - Issues Data Check Warning                                 | 10 |
| WORKLGH - Size of Work Areas  | 10 |
| Storage Estimates   | 11 |
| Natural for DL/I in OS/390 Environments                             | 12 |
| <b>Installing Natural for DL/I</b>                                  | 13 |
| Installing Natural for DL/I   | 13 |
| Prerequisites   | 13 |
| Installation Tape - OS/390 Systems                                  | 13 |
| Copying the Tape Contents to Disk                                   | 13 |
| Installation Tape - VSE/ESA Systems                                 | 15 |
| Copying the Tape Contents to Disk                                   | 15 |
| Installation Procedure  | 16 |
| Installation Verification   | 19 |
| <b>Operation</b>  | 20 |
| Operation   | 20 |
| Procedure NATPSB  | 20 |
| Procedure NATDBD  | 25 |
| Using Logical Databases with Natural                                | 26 |
| Using Index Databases with Natural                                  | 26 |
| Procedure NATUDF  | 27 |
| Segment Identification Statement                                    | 28 |
| Segment Field Description   | 29 |
| Generation of DDMs from DL/I Segment Types                          | 31 |
| <b>System File Structure</b>  | 32 |
| System File Structure   | 32 |
| The NDB Subfile   | 32 |

|   |           |
|---|-----------|
| The NSB Subfile . . . . .   | 33        |
| The UDF Subfile . . . . .   | 33        |
| Natural for DL/I Objects . . . . .                                      | 34        |
| Displaying Keys of UDF Blocks . . . . .                                 | 34        |
| Displaying the Size of NDL Objects . . . . .                            | 34        |
| Displaying NDL Objects . . . . .  | 34        |
| Control Blocks in Separate Buffer Pool . . . . .                        | 35        |
| Control Blocks in Buffer Pool Blacklist . . . . .                       | 36        |
| Natural for DL/I Objects and Natural DDMs . . . . .                     | 36        |
| <b>Natural Batch Utilities . . . . .</b>                                | <b>37</b> |
| Natural Batch Utilities . . . . .                                       | 37        |
| Transfer of NDBs/NSBs/UDFs from one System File to Another . . . . .    | 37        |
| Unloading the NDBs, NSBs and UDFs . . . . .                             | 38        |
| Loading NDBs, NSBs and UDFs . . . . .                                   | 39        |
| Selecting NDBs, NSBs and UDFs from a Dataset . . . . .                  | 40        |
| Utility NDUDFGEN for Natural Data Areas . . . . .                       | 42        |
| Input for NDUDFGEN . . . . .  | 42        |
| <b>Execution . . . . .</b>  | <b>45</b> |
| Execution . . . . .   | 45        |
| PSB Scheduling . . . . .  | 45        |
| The NATPSB Command . . . . .  | 45        |
| PSB Scheduling in a Batch Environment . . . . .                         | 46        |
| PSB Scheduling in a CICS Environment . . . . .                          | 49        |
| PSB Scheduling in an IMS/TM Environment . . . . .                       | 49        |
| CALLNAT Interface . . . . .   | 50        |
| The NDLPBAD Subprogram . . . . .  | 50        |
| The NDLPBSC Subprogram . . . . .  | 51        |
| Support of IMS-Specific Features . . . . .                              | 52        |
| Symbolic Checkpoint/Restart Functions - CHKP, XRST . . . . .            | 52        |
| The INIT Call to Enable Data Availability Status Codes . . . . .        | 54        |
| Fast Path Support . . . . .   | 54        |
| Support of GSAM . . . . .   | 55        |
| Processing in CICS Pseudo-Conversational Mode or under IMS/TM . . . . . | 57        |
| <b>Programming Language Considerations . . . . .</b>                    | <b>58</b> |
| Programming Language Considerations . . . . .                           | 58        |
| Natural versus Third Generation Languages . . . . .                     | 58        |
| Natural Statements with DL/I . . . . .                                  | 59        |
| BACKOUT TRANSACTION . . . . .   | 59        |
| DELETE . . . . .  | 59        |
| DISPLAY . . . . .   | 59        |
| END TRANSACTION . . . . .   | 60        |
| FIND . . . . .  | 60        |
| GET TRANSACTION DATA . . . . .  | 62        |
| READ . . . . .  | 62        |
| RELEASE . . . . .   | 63        |
| STORE . . . . .   | 63        |
| UPDATE . . . . .  | 64        |
| WRITE . . . . .   | 64        |
| Statements not Available for DL/I . . . . .                             | 64        |
| Natural System Variables with DL/I . . . . .                            | 65        |
| *ISN . . . . .  | 65        |
| *NUMBER . . . . .   | 65        |
| <b>Problem Determination Guide . . . . .</b>                            | <b>66</b> |
| Problem Determination Guide . . . . .                                   | 66        |
| <b>Performance Considerations . . . . .</b>                             | <b>68</b> |
| Performance Considerations . . . . .                                    | 68        |

|   |           |
|---|-----------|
| Parameters . . . . .                            | 68        |
| DBID . . . . .                                  | 68        |
| Global and Local Data Areas . . . . .           | 68        |
| FIND Statements . . . . .                       | 68        |
| Direct Access to Lower Levels . . . . .         | 68        |
| DBLOG Utility . . . . .                         | 68        |
| <b>DL/I Services . . . . .</b>                  | <b>69</b> |
| DL/I Services . . . . .                         | 69        |
| NDB Maintenance . . . . .                       | 69        |
| Menu and Functions . . . . .                    | 69        |
| Select an NDB from a List . . . . .             | 70        |
| Select an NDB Segment from a List . . . . .     | 71        |
| Edit an NDB Segment Description . . . . .       | 73        |
| Generate DDM from Segment Description . . . . . | 79        |
| NSB Maintenance . . . . .                       | 82        |
| Select an NSB from a list: . . . . .            | 82        |
| List PCBs and SENSECs of an NSB: . . . . .      | 83        |



# Natural for DL/I - Overview

This documentation provides information on Natural in a DL/I environment. It describes the installation and operation of Natural for DL/I, as well as special considerations on Natural statements when used with DL/I.

This documentation covers:

- General Information      Brief information on features.
- Natural Parameter Modifications for DL/I      Explains parameters contained in NDLPARM, storage estimates, and Natural for DL/I in OS/390 environments.
- Installing Natural for DL/I      How to install Natural for DL/I.
- Operation      Describes procedures NATPSB, NATDBD, NATUDF, and the generation of DDMs from DL/I segment types.
- System File Structure      Describes the database structure, the segment data and the processing intent of an application.
- Natural Batch Utilities      Describes the system file transfer of NDBs, NSBs and UDFs from one FDIC and the use of the batch utility NDUDFGEN to generate Natural data areas.
- Execution      Describes PSB scheduling, the CALLNAT interface, support of IMS-specific features, fast path and GSAM, and CICS mode processing under IMS/TM.
- Programming Language Consideratons      Natural versus Third Generation Languages, Natural Statements with DL/I , Natural System Variables with DL/I .
- Problem Determination Guide      Actions required to correct a given problem.
- Performance Considerations      How to increase the performance of Natural in a DL/I environment.
- DL/I Services      Terminology and maintenance of NDBs and NSBs.

For a list of DL/I status codes and abend codes (under CICS only), refer to Status Codes and Abend Codes (in the Natural Messages and Codes documentation).

# General Information

This section covers the following topics:

- Basic Principles
  - Accessing DL/I Data
- 

## Basic Principles

With Natural for DL/I, a Natural user can access and update data stored in a DL/I database. The Natural user can be executing in batch mode or under the control of the TP monitor CICS or IMS/TM .

A DL/I database is represented to Natural as a set of files, each file representing one database segment type. Each file or segment type must have an associated DDM generated and stored on the Natural FDIC system file.

Since Natural for DL/I is an extension to Natural, nearly all of the information contained in the Natural documentation applies to its use in the DL/I environment as well as in the Adabas environment.

The Natural statements used to access DL/I databases are a subset of those provided with the Natural language. No new statements are needed to access a DL/I database.

Applications developed using Natural for DL/I operate as standard DL/I applications. This means that all access to DL/I databases performed by Natural follows the DL/I product conventions. For an online Natural session or batch Natural program to issue a DL/I database call, a PSB must first be scheduled. The PCB in use must have segment sensitivity and the appropriate PROCOPT parameter must be specified for Natural, to be able to perform a segment update. Only standard DL/I database calls are issued by Natural.

## Accessing DL/I Data

Natural for DL/I allows Natural programs to access DL/I databases using Natural statements.

To access DL/I data, Natural requires certain information on these data. This information mainly consists of four types of control blocks:

- the original database descriptions (DBDs) and program specification blocks (PSBs) which are required by DL/I itself;
- suitable copies of DL/I DBDs and PSBs for Natural, called NDBs and NSBs;
- user-defined fields (UDFs);
- Natural DDMs generated from NDBs and UDFs.

All information required by Natural to access DL/I databases is stored and maintained in the Natural FDIC system file. The Natural FDIC system file can be an Adabas file (if Adabas is installed), or a VSAM file (only in CICS environments).

As is the case with any DL/I application, a DL/I DBDGEN and PSBGEN must be performed to define the data structure the Natural application is to have access to, and the processing intent this application has on these data. This same information, which is contained in the DBD and PSB source statements, must also be defined to Natural.

The Natural batch procedures NATDBD and NATPSB are used to add this information to the Natural FDIC system file. They generate NDBs and NSBs from the respective DBDs and PSBs, using the DBDGEN and PSBGEN source respectively, as input.

It is the administrator's responsibility to ensure that the contents of the DL/I DBDLIB and PSBLIB and the Natural FDIC system file are compatible. It is therefore recommended that the DL/I procedures DBDGEN and PSBGEN and the Natural procedures NATDBD and NATPSB always be executed as a pair.

The DBDGEN source usually does not define all fields within a segment. Additional segment fields, called user-defined fields (UDFs), can be entered as part of creating the DDMs. UDFs in Natural are added by using either the batch utility NATUDF, the EDIT Segment Description facility of SYSDDM, or Predict.

Once all the necessary information has been stored on the Natural FDIC system file, Natural DDMs defining the DL/I database segment types can be created.

# Natural Parameter Modifications for DL/I

Natural parameter default values for DL/I can be changed to meet your particular requirements. The object module NDLPARM, which is used for Natural static parameter assignment in a DL/I environment, must then be appropriately modified and reassembled.

This section covers the following topics:

- Parameters in NDLPARM
  - Storage Estimates
  - Natural for DL/I in OS/390 Environments
- 

## Parameters in NDLPARM

The following parameters are contained in NDLPARM:

- DFBNUM - Maximum Entries in Translated Format Buffer
- DFFNUM - Maximum Fields in Single Entry of Translated Format Buffer
- FLBNUM - Number of Entries in Fast Locate Buffer
- INGSIZE - Initial Size of Buffer to Copy Parameter List
- INGOSIZ - Initial Size of I/O Area for DL/I Calls
- INITCAL - Issues INIT Call at Transaction Start
- PCBLEV - Maximum Number of PCB Levels
- PCBNUM - Maximum Number of PCBs in a PSB
- RELEVNT - Requests Relocation Event
- RESINDB - NDB Resident in Buffer Pool
- RESINSB - NSB Resident in Buffer Pool
- RESIUFD - UDF Resident in Buffer Pool
- SASIZE - Size of Natural Save Area for DL/I
- SEQNUM - Maximum Number of Nested Sequential Accesses
- SEQSSA - Maximum Size of an SSA
- THCSIZE - Table Size to Save Natural Field Values
- TRACE - Trace Options
- TYPCHCK - Numeric/Packed Data Check
- TYPWARN - Issues Data Check Warning
- WORKLGH - Size of Work Areas

## DFBNUM - Maximum Entries in Translated Format Buffer

| Possible Values | Default Value |
|-----------------|---------------|
| 5 - 200         | 25            |

This parameter is used to indicate the maximum number of entries in the table of translated format buffers.

An entry in this table is created for each active Natural input/output statement (FIND, READ, UPDATE, STORE).

When increasing DFBNUM or DFFNUM, take into consideration that the allocated storage area size is obtained by multiplying these values and **not** by adding them.

## DFFNUM - Maximum Fields in Single Entry of Translated Format Buffer

| Possible Values | Default Value |
|-----------------|---------------|
| 5 - 1000        | 10            |

This parameter is used to indicate the average number of fields contained in each single entry of the table of translated format buffers.

A field entry in this table is created for each field referenced in a Natural input/output statement (FIND, READ, UPDATE, STORE).

When increasing DFFNUM or DFBNUM, take into consideration that the allocated storage area size is obtained by multiplying these values and **not** by adding them.

## FLBNUM - Number of Entries in Fast Locate Buffer

| Possible Values | Default Value |
|-----------------|---------------|
| 0 - 32767       | 50            |

This parameter is used to indicate the number of entries in the Fast Locate Buffer. This buffer holds absolute addresses of NDL objects (that is, NDBs, NSBs, UDFs) in the buffer pool.

The addresses are stored in wrap-around technique.

This buffer is especially useful if NDL objects have been marked as "resident" in the buffer pool (see the related parameters RESINDB, RESINSB, RESIUDF).

It allows Natural for DL/I to use the Fast Locate algorithm of the Natural buffer pool manager when locating objects.

## INGSIZE - Initial Size of Buffer to Copy Parameter List

| Possible Values      | Default Value |
|----------------------|---------------|
| 1000 - 32767 (bytes) | 1000          |

This parameter is used to indicate the initial size of the buffer which is used to copy the DL/I call parameter list and the call parameters below 16 MB if Natural operates in an OS/390 environment. If the initial size is not sufficient, Natural automatically increases the size of this buffer accordingly.

## INGOSIZ - Initial Size of I/O Area for DL/I Calls

| Possible Values      | Default Value |
|----------------------|---------------|
| 1000 - 32767 (bytes) | 1000          |

This parameter is used to indicate the initial size of the I/O area for DL/I calls. This area is re-used for subsequent DL/I calls if no GET HOLD call has been issued.

If the initial size is not sufficient, Natural automatically increases the size of this buffer accordingly.

## INITCAL - Issues INIT Call at Transaction Start

| Possible Values | Default Value |
|-----------------|---------------|
| NO/YES          | NO            |

This parameter is used to inform IMS that Natural is prepared to accept status codes BA or BB regarding data unavailability.

The setting of this parameter only applies if Natural runs in a BMP or MPP region.

## PCBLEV - Maximum Number of PCB Levels

| Possible Values | Default Value |
|-----------------|---------------|
| 1 - 15          | 10            |

This parameter is used to indicate the maximum number of PCB levels which can be processed by Natural.

When increasing PCBLEV, take into consideration that the allocated storage area size is obtained by multiplying these values and **not** by adding them.

## PCBNUM - Maximum Number of PCBs in a PSB

| Possible Values | Default Value |
|-----------------|---------------|
| 1 - 255         | 25            |

This parameter is used to indicate the maximum number of PCBs which can be contained within a single PSB.

When increasing PCBNUM, take into consideration that the allocated storage area size is obtained by multiplying these values and **not** by adding them.

## RELEVNT - Requests Relocation Event

| Possible Values | Default Value |
|-----------------|---------------|
| NO/YES          | NO            |

This parameter is used to inform the Natural nucleus whether or not Natural for DL/I requests relocation events.

With RELEVNT=YES, Natural for DL/I is called for relocation on every relocation event, that is, even if no DL/I call has been issued since the last relocation event.

With RELEVNT=NO, Natural for DL/I is not called for relocation. Instead, it checks itself whether relocation is required before a DL/I call is issued.

## RESINDB - NDB Resident in Buffer Pool

| Possible Values | Default Value |
|-----------------|---------------|
| NO/YES          | YES           |

This parameter is used to indicate whether NDBs are to be kept resident in the buffer pool.

## RESINSB - NSB Resident in Buffer Pool

| Possible Values | Default Value |
|-----------------|---------------|
| NO/YES          | YES           |

This parameter is used to indicate whether NSBs are to be kept resident in the buffer pool.

**RESIUDF - UDF Resident in Buffer Pool**

| Possible Values | Default Value |
|-----------------|---------------|
| NO/YES          | YES           |

This parameter is used to indicate whether UDFs are to be kept resident in the buffer pool.

**SASIZE - Size of Natural Save Area for DL/I**

| Possible Values     | Default Value |
|---------------------|---------------|
| 1000 - 3000 (bytes) | 1000          |

This parameter is used to indicate the size of the save area.

Do not increase the default value, unless you receive an error message which indicates that a save area overflow has occurred.

**SEQNUM - Maximum Number of Nested Sequential Accesses**

| Possible Values | Default Value |
|-----------------|---------------|
| 5 - 100         | 20            |

This parameter is used to indicate the maximum number of nested sequential accesses which can be processed by Natural.

When increasing the values for the SEQNUM and SEQSSA parameters, remember that the storage area allocated is dependent on the product of these areas, **not** their sum.

**SEQSSA - Maximum Size of an SSA**

| Possible Values  | Default Value |
|------------------|---------------|
| 10 - 500 (bytes) | 50            |

This parameter is used to indicate the maximum size of an SSA related to sequential access.

When increasing the values for the SEQNUM and SEQSSA parameters, remember that the storage area allocated is dependent on the product of these areas, **not** their sum.

## THCSIZE - Table Size to Save Natural Field Values

| Possible Values      | Default Value |
|----------------------|---------------|
| 2000 - 32000 (bytes) | 3000          |

This parameter only applies under IMS/TM or under CICS in pseudo-conversational mode.

This parameter is used to indicate the size of the table which is used to save field values in hold status when running under IMS/TM or under CICS in pseudo-conversational mode.

## TRACE - Trace Options

| Possible Values | Explanation                         |
|-----------------|-------------------------------------|
| ALL             | Trace all modules                   |
| CMD             | Trace command execution             |
| REQ             | Trace request modules               |
| ROU             | Trace routines                      |
| SER             | Trace service modules               |
| OFF             | Trace is not active. Default value. |

This parameter is used to indicate whether Natural trace information is to be created and printed or not.

The options CMD, REQ, SER and ROU can be combined.

## TYPCHCK - Numeric/Packed Data Check

| Possible Values | Default Value |
|-----------------|---------------|
| NO/YES          | NO            |

This parameter is used to indicate whether numeric or packed segment fields from DL/I are to be checked for valid data and repaired, if necessary.

With TYPCHCK=NO, no data check is performed. Natural for DL/I would abend with data exception if, for example, a packed field contained blanks.

With TYPCHCK=YES, a data check is performed. If the field does not contain format compatible data, it is filled with zeroes. In addition, a message is issued, depending on the setting of the parameter TYPWARN (see below).

## TYPWARN - Issues Data Check Warning

| Possible Values | Default Value |
|-----------------|---------------|
| NO/YES          | NO            |

This parameter only applies if TYPCHCK has been specified (see above).

This parameter is used to indicate whether a message is to be issued if a data check and repair has been performed.

With TYPWARN=NO, no message is issued if a data repair has been performed.

With TYPWARN=YES, a message is issued if a data repair has been performed. This message displays the short name of the field in error. The message is issued as a warning (only), which means that:

- The message is not issued via the Natural error exit but is directly inserted into the page buffer.
- The message(s) is (are) only issued when the page buffer is full.
- There is no backout transaction.
- The program flow is not interrupted.

## WORKLGH - Size of Work Areas

| Possible Values | Default Value |
|-----------------|---------------|
| 1000 - 3000     | 1000          |

This parameter is used to indicate the size of the work areas. Natural allocates six work areas of this size.

Do not increase the default value, unless you receive an error message which indicates that a work area overflow has occurred.

## Storage Estimates

The memory size required by Natural for DL/I is determined by the following items:

1. Object code: 90 KB.
2. Save areas: 3 KB.
3. Work areas: 6 KB.
4. Fast Locate Buffer: 12 bytes for each entry.
5. XRST buffer: 2 KB.
6. Internal tables: the amount of storage allocated depends on parameters specified in the module NDLPARM.

The following formula can be used to compute the amount of storage required for initial table allocation:

Amount of Storage =

$$\begin{aligned} & \text{SEQNUM} * (\text{SEQSSA} + 64) + 32 + \\ & \text{DFBNUM} * (28 + (\text{DFFNUM} * 12)) + 20 + \\ & \text{PCBNUM} * (24 + 12 + (\text{PCBLEVL} * 5)) + 20 + \\ & \text{TCHSIZE} \end{aligned}$$

The above formula can be described as follows:

| Term                    | Computational Expression                                |
|-------------------------|---|
| Sequential Access Table | $\text{SEQNUM} * (\text{SEQSSA} + 64) + 32$             |
| Field Table             | $\text{DFBNUM} * (28 + (\text{DFFNUM} * 12)) + 20$      |
| PCB Map                 | $\text{PCBNUM} * (24 + 12 + (\text{PCBLEVL} * 5)) + 20$ |
| Table of Fields in Hold | TCHSIZE   |

If the standard values of these NDLPARM parameters are used in the above formula, 14 KB of storage is allocated.

7. Segment I/O areas are to be added on additionally.

### Note:

The object code is shared among all Natural sessions. There is a copy of all other areas for each active Natural session.

The storage required for save areas, work areas, Fast Locate Buffer, XRST buffer and internal tables is allocated from the thread at the initialization of the Natural session. Six GETMAINS are performed, the sizes of which are determined by the values of the parameters in the NDLPARM module. If the default values of the NDLPARM parameters are used, the total size required is 27 KB.

The total size available is determined by the profile parameter DLISIZE in the Natural parameter module (NATPARM); see the Natural Parameter Reference documentation.

The BUS (Buffer Usage Statistics) command can be used to obtain information on the sizes of the buffers allocated by Natural for DL/I. The following information is provided:

| Buffer   | Content   |
|----------|---|
| DLISIZE0 | contains the Fast Locate Buffer, the XRST buffer, and the save areas. |
| DLISIZE1 | contains the work areas.  |
| DLISIZE2 | contains the sequential access table.                                 |
| DLISIZE3 | contains the field table .  |
| DLISIZE4 | contains the PCB map .  |
| DLISIZE5 | contains the table of fields in hold status.                          |

## Natural for DL/I in OS/390 Environments

Before Natural issues a DL/I call in an OS/390 environment, it checks whether the call parameter list or any of the call parameters reside above the 16 MB line. This is the case if the Natural threads have been placed above this line. If so, the parameter list and all parameters are copied into a buffer which has been allocated below the line via GETMAIN. The pointers in the parameter list are modified accordingly to point to the new parameters.

The initial size of this buffer is set by the INGSIZE parameter of NDLPARM. If the initial size is not sufficient, Natural automatically increases the size of this buffer accordingly.

This overhead is required because DL/I terminates programs abnormally if parameter addresses passed in DL/I calls do not refer to code or storage areas below the 16 MB line.

# Installing Natural for DL/I

This section describes step by step how to install Natural for DL/I, also referred to as NDL.

- Prerequisites
  - Installation Tape - OS/390 Systems
  - Installation Tape - VSE/ESA Systems
  - Installation Procedure
  - Installation Verification
- 

## Prerequisites

Products and versions are specified under Natural and Other Software AG Products in and Operating/Teleprocessing Systems Required in the current Natural Release Notes for Mainframes.

## Installation Tape - OS/390 Systems

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  |
|-----------------|---|
| NDL $nnn$ .LOAD | Natural executable modules necessary for the linkage editor.                                    |
| NDL $nnn$ .SRCE | Macros and sources for the parameter module NDLPARM and for the batch procedures NATDBD/NATPSB. |
| NDL $nnn$ .JOBS | Example installation jobs.  |

The notation  $nnn$  in dataset names represents the version number of the product.

## Copying the Tape Contents to Disk

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

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

This section explains how to:

- Copy data set COPY.JOB from tape to disk.
- Modify this data set to conform with your local naming conventions.

The JCL in this data set is then used to copy all data sets from tape to disk.

If the datasets for more than one product are delivered on the tape, the dataset COPY.JOB contains the JCL to unload the datasets for all delivered products from the tape to your disk.

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

### Step 1 - Copy data set COPY.JOB from tape to disk

The data set COPY.JOB (label 2) contains the JCL to unload all other existing data sets from tape to disk. To unload COPY.JOB, use the following sample JCL:

```
//SAGTAPE JOB SAG,CLASS=1,MSGCLASS=X
//* -----
//COPY EXEC PGM=IEBGENER
//SYSUT1 DD DSN=COPY.JOB,
// DISP=(OLD,PASS),
// UNIT=(CASS,,DEFER),
// VOL=(,RETAIN,SER=<Tnnnnn>),
// LABEL=(2,SL)
//SYSUT2 DD DSN=<hilev>.COPY.JOB,
// DISP=(NEW,CATLG,DELETE),
// UNIT=3390,VOL=SER=<vvvvvvv>,
// SPACE=(TRK,(1,1),RLSE),
// DCB=*.SYSUT1
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//
```

Where:

<hilev> is a valid high level qualifier

<Tnnnnn> is the tape number

<vvvvvvv> is the desired volser

### Step 2 - Modify COPY.JOB to conform with your local naming conventions

There are three parameters you have to set before you can submit this job:

- Set HILEV to a valid high level qualifier.
- Set LOCATION to a storage location.
- Set EXPDT to a valid expiration date.

### Step 3 - Submit COPY.JOB

Submit COPY.JOB to unload all other data sets from the tape to your disk.

## Installation Tape - VSE/ESA Systems

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         |
|----------------------|------------------|
| NDL <i>nnn</i> .LIBR | LIBR backup file |

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

### Copying the Tape Contents to Disk

Copy the sublibrary containing the sample installation jobs from tape using the following JCS:

```
* $$ JOB JNM=NATJOBS,CLASS=0,DISP=D,LDEST=*,SYSID=1
* $$ LST CLASS=A,DISP=D
// JOB NATJOBS
// ASSGN SYS005,IGN
// ASSGN SYS006,cuu,VOL=Tnnnnn
// MTC REW,cuu
// MTC FSF,SYS006,nn
* Tape positioned at tape mark nn
* *** NOW PROCESSING NDLnnn.LIBR - SUBLIBRARY NDLnnnJ ***
// EXEC LIBR,PARM='MSHP'
RESTORE SUBLIB=SAGLIB.NDLnnnJ:SAGLIB.NDLnnnJ -
TAPE=SYS006 -
LIST=YES -
REPLACE=NO
/*
// MTC REW,SYS006
/*
/&
* $$ EOJ
```

#### Notation:

|            |  |
|------------|--|
| <i>cuu</i> | represents the physical unit address of the tape drive.                      |
| <i>nn</i>  | represents the file sequence number as shown in the Report of Tape Creation. |
| <i>nnn</i> | represents the version number of the product.                                |

If you are not using System Maintenance Aid, adapt and run job NDLTAPE to copy the dataset from tape to disk. NDLTAPE is contained in sublibrary NDL*nnn*J on the Natural installation tape.

The dataset type and the space it requires on disk are shown in the Report of Tape Creation.

## Installation Procedure

The NDL installation procedure consists of the following steps:

### Step 1: Create the NDL Parameter Module - Job I055, Step 1500

Modify the NDL parameter module NDLPARM as described in the section Natural Parameter Modifications for DL/I.

Assemble and link/catalog NDLPARM.

### Step 2: Modify the Natural Parameter Modules - Jobs I060 and I080

Modify the appropriate I060 and I080 jobs according to the TP monitor or batch modules you are relinking; for example, NATI060 for batch and NCII080 for CICS. This applies also to Step 3 below.

Add the parameter DLISIZE and specify DLISIZE=27.

This value applies if the default values of the NDLPARM parameters are used.

Add an NTDB macro specifying the database identification list (DBID list) that relates to DL/I segment types. The numbers specified in this DBID list must be in the range from 1 to 254. They indicate which DBIDs are reserved for DL/I segment types. Up to 254 entries can be specified. All Natural DDMs that refer to a DL/I segment type are cataloged with a DBID from this list. The number with the lowest value in this list is the default DBID for DL/I segment types.

#### Examples:

```
NTDB DLI,(250,253,252)
NTDB DLI,250
```

#### Note:

Values for DL/I database IDs above 255 are not possible.

### Step 3: Link the Natural Nucleus - Job I060 and I080

#### Under OS/390:

Add the following INCLUDE instructions and the corresponding DD statements to the link step for Natural and link-edit the executable module:

| CICS                       | IMS/TM                   | Batch Mode               |
|----------------------------|--------------------------|--------------------------|
| INCLUDE NDLLIB(NDLNUC)     | INCLUDE NDLLIB(NDLNUC)   | INCLUDE NDLLIB(NDLNUC)   |
| INCLUDE NDLLIB(NDLSIOCX)   | INCLUDE NDLLIB(NDLSIOBA) | INCLUDE NDLLIB(NDLSIOBA) |
| INCLUDE<br>SMALIB(NDLPARM) | INCLUDE SMALIB(NDLPARM)  | INCLUDE SMALIB(NDLPARM)  |
| INCLUDE TPSLIB(ASMTDLI)    | INCLUDE DLILIB(ASMTDLI)  | INCLUDE DLILIB(ASMTDLI)  |

**Under VSE/ESA:**

Add the following INCLUDE instructions and the corresponding sublibraries for NDL to the search chain for the linkage editor and link-edit the executable module:

| CICS             | Batch Mode       |
|------------------|------------------|
| INCLUDE NDLNUC   | INCLUDE NDLNUC   |
| INCLUDE NDLSIOCX | INCLUDE NDLSIOBA |
| INCLUDE NDLPARM  | INCLUDE NDLPARM  |
| INCLUDE ASMTDLI  | INCLUDE ASMTDLI  |

Under CICS, the link-edit of the load module that contains NDL can be done in any of the following ways:

- Include all NDL modules (that is, NDLNUC, NDLPARM and NDLSIOCX) and the DL/I module ASMTDLI in the link-edit of Natural.
- Include all NDL modules (that is, NDLNUC, NDLPARM and NDLSIOCX) and the DL/I module ASMTDLI in the link-edit of the Natural TP driver.  
This way of link-editing only applies if the Natural TP driver runs separately from the Natural nucleus.
- Link-edit all NDL modules (that is, NDLNUC, NDLPARM and NDLSIOCX), the DL/I module ASMTDLI and an alternate Natural parameter module as a separate module with the mandatory *entry* name CMPRMTB. The *name* of the resulting module is optional.  
This way of link-editing only applies if an alternate parameter module ("PARM=") is used. If so, under CICS, an additional CICS PPT entry with PROGRAM=name is required.
- Link-edit all NDL modules (that is, NDLNUC, NDLPARM and NDLSIOCX) and the DL/I module ASMTDLI as a separate module with the mandatory *entry* name NATGWDLI. The *name* of the resulting module is optional. If it is different from NATGWDLI, however, it 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. If so, under CICS, an additional CICS PPT entry with PROGRAM=name is required.
- Include all environment-independent NDL Modules (i.e. NDLNUC and NDLPARM) in the link-edit of Natural  
Include the environment-dependent NDL I/O module (NDLSIOCX) in the link-edit of the Natural TP driver.  
This way of link-editing only applies if a shared nucleus is created.

**Step 4: Establish a Natural Environment for DL/I**

To verify the installation of NDL with a sample database rather than with existing databases, you perform the following steps:

1. Allocate VSAM spaces for the sample database  
(Job I008, Steps 1500 to 1502).
2. Create the DBDs, PSBs and ACB, and perform the initial load  
(Job I053, Steps 1500 to 1560).  
Creation of an ACB only applies to VSE/ESA.
3. Execute procedures NATPSB and NATDBD for the sample database  
(Job I075, Steps 1500 and 1510).  
To enable Natural to access DL/I databases, additional data must be added to the FDIC system file. To do so, the procedures NATPSB and NATDBD must be executed for each PSB/DBD to be used.

## Installation Verification

▶ To verify the installation of Natural for DL/I

1. Invoke online Natural.
2. Invoke the Natural utility SYSDDM by entering the following command:  
SYSDDM
3. On the SYSDDM menu, enter function code "D" to invoke the DL/I Services function.
4. On the resulting screen, enter function code "D" to invoke the NDB Maintenance function.
5. On the resulting screen, enter function code "S" to select the NDB which was created in substep 3 of Step 4.
6. On the resulting screen, enter function code "L" to list the NDB segments.
7. On the resulting screen, enter function code "A" to assign DBID and FNR to the segments.
8. On the same screen, enter function code "G" to generate a DDM from the segment description.
9. Catalog the generated DDM.
10. Only if running under CICS:  
Enter "NATPSB ON *psbname*" in the command line.
11. Edit and run the following program:

```
        DEFINE DATA LOCAL
01 COURSE VIEW OF DPQA03-COURSE
02 COURSEN
02 TITLE
02 DESCRIPN

                                                /* End of DPQA03-COURSE View

END-DEFINE
READ (100) COURSE BY COURSEN
    DISPLAY COURSEN TITLE DESCRIPN
END
```

# Operation

Natural for DL/I operates as a standard DL/I application.

Prior to running a Natural application, a PSB must be scheduled. The method for scheduling PSBs varies depending on the actual environment (see the relevant sections under PSB Scheduling), but as for any other DL/I application, PSB scheduling is a requirement.

This section covers the following topics:

- Procedure NATPSB
  - Procedure NATDBD
  - Procedure NATUDF
  - Generation of DDMs from DL/I Segment Types
- 

## Procedure NATPSB

Every PSB required by DL/I to accommodate Natural requests must be processed by the Natural batch utility NDPBNSB0. This utility stores DL/I PSB information, in a form suitable for Natural, on the FDIC system file. This information is referred to as NSB control block. A batch procedure called NATPSB has been established for this purpose.

A sample NATPSB job has been included in the source library from the installation tape. The information used to create NSB control blocks comes from the actual PSBGEN source. It is essential that the same input is used for the NATPSB procedure as was used for the DL/I PSBGEN. Otherwise, unpredictable results are likely.

The NATPSB job is a three step procedure:

- The first step executes the normal DL/I PSBGEN procedure. This step is included to guarantee compatibility between DL/I and Natural.
- The second step performs another assembly and link of the PSBGEN source, this time using macros supplied by Natural.
- The final step executes the Natural batch utility NDPBNSB0, which uses the linked PSB module from the previous step to create NSB control blocks which are stored on the FDIC system file. NDPBNSB0 dynamically loads the Natural module NDLB0002, which therefore must be present in an allocated load library.

Natural requires one or more PSBs for batch and/or online processing. Depending on application requirements, the PSB can be switched during a Natural session. Each PSB describes all user views that can be used to access DL/I databases from Natural programs if this PSB is active. A PSB must contain one or more program communication blocks (PCBs) for each DBD to be accessed. Since Natural only uses the single positioning option on PCBs, Natural programs that maintain two or more independent positions in a database require a PCB (of the appropriate type) for each separate position.

If this requirement is not fulfilled, Natural for DL/I issues the runtime error message:

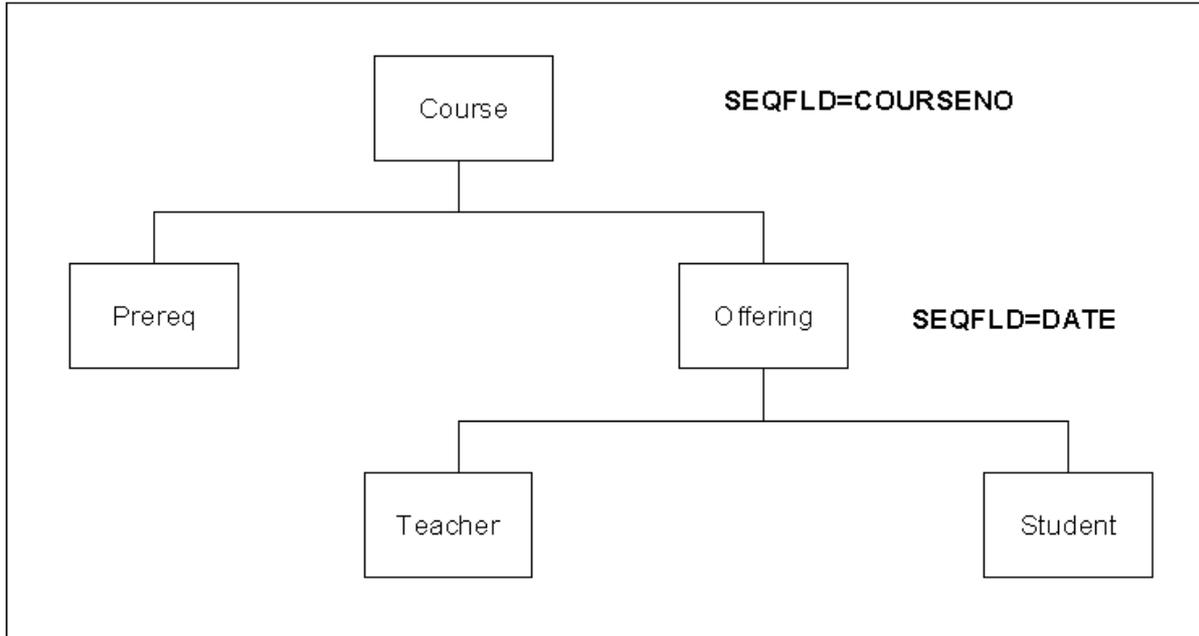
NAT3789 Active PSB contains too few PCBs for program execution.

The PCB in use must have segment sensitivity and the appropriate PROCOPT parameter specified for Natural, to be able to perform a segment update.

Nested I/O loops (FIND or READ) in Natural programs frequently require separate positions in the same database to be maintained. To reduce the number of PCBs needed, as many I/O loops as possible should be closed before opening subsequent I/O loops.

Consider the following sample DL/I database:

**Sample Education Database ED00DBD:**



The following Natural program based on the above database requires two PCBs:

```

READ ED00DBD-COURSE BY COURSENO
  FIND ED00DBD-PREREQ WITH COURSENO-COURSE = COURSENO
  FIND ED00DBD-OFFERING WITH COURSENO-COURSE = COURSENO
  LOOP
  LOOP
LOOP
END

```

The first PCB is used to maintain position on the COURSE and PREREQ segments. A second PCB is required for the OFFERING segment since the FIND loop has not been terminated for the PREREQ segment prior to invoking a FIND on the OFFERING segment. By closing the first FIND loop prior to opening the second one, this program would only require one PCB.

Natural selects the PCB to be used for a database request in the following manner:

1. Natural selects the first PCB in the PSB with the correct DBD name and the appropriate PROCSEQ parameter (if applicable).
2. Natural then determines if the PCB can be used for the request or if there is a conflict due to current database positioning.
3. If there was a positioning conflict or the PCB did not contain the correct DBD name or PROCSEQ parameter, Natural would continue scanning the PSB.
4. If the database search request refers to a secondary index, Natural attempts to use a PCB with the corresponding PROCSEQ parameter. If there is no PCB of this type in the PSB, Natural tries to use a PCB without the PROCSEQ parameter. In this case, it is assumed that the INDICES parameter has been coded in the appropriate SENSEQ statement.
5. If no eligible PCB could be found, an error message would be generated.

In general, PCBs for use by Natural can have different PROCOPT parameters. However, if there are two or more PCBs in the PSB referring to the same DBD, these PCBs must appear consecutively in the PSB source and they must specify the same SENSEG statements and same PROCOPT parameters. They can, however, have different PROCSEQ parameters.

When locating an eligible PCB, Natural disregards the PROCOPT parameter of the PCB. The first free PCB is selected independently of the PROCOPT parameter, so that if the chosen PCB has a PROCOPT that does not support the request, an error message that corresponds to a DL/I status code is returned.

Natural assumes that all PCBs with the same DBD name and the same PROCSEQ parameter contain the same SENSEG statements as the first PCB. If this is not true and a PCB is selected that does not contain a SENSEG statement for the segment being referenced, an error message that corresponds to a DL/I status code is returned.

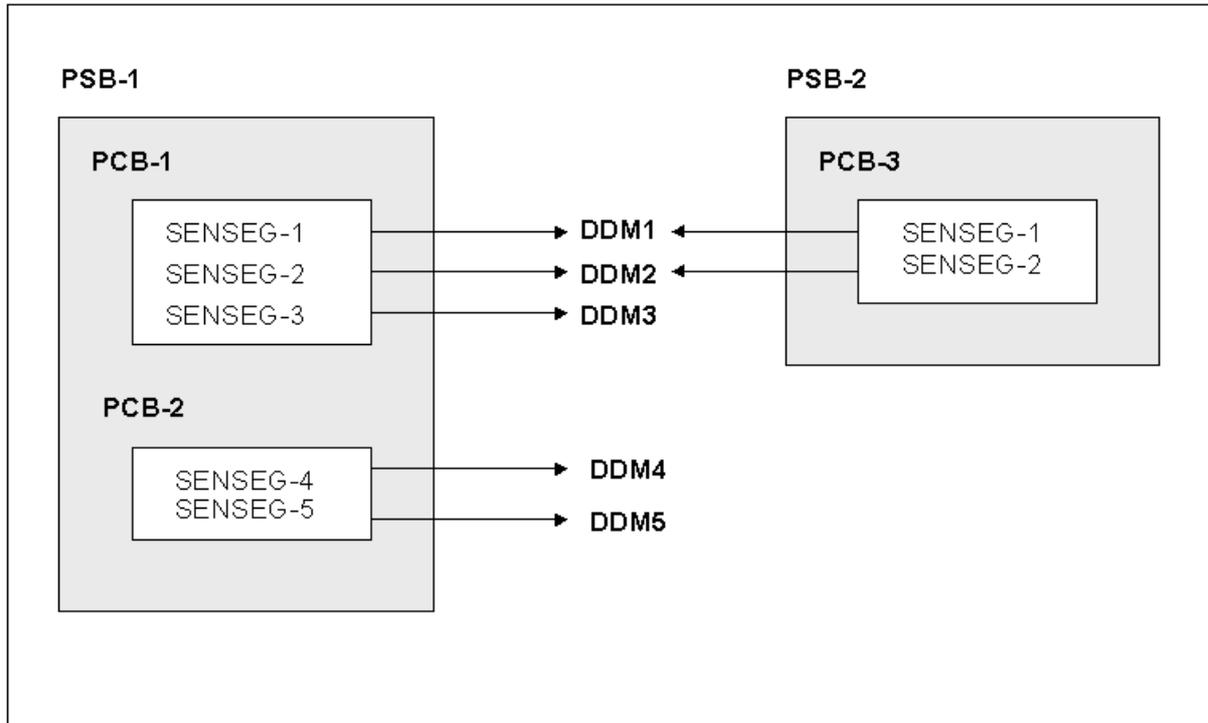
The following example PSB and Natural program demonstrate that the sequence of the PCBs, referring to the same DBD, may affect Natural programs if the PROCOPT= parameters are different:

```
PCB      TYPE=DB, DBDNAME=ED00DBD, PROCOPT=GO, . . .
SENSEG  NAME=COURSE
SENSEG  NAME=OFFERING, PARENT=COURSE
PCB      TYPE=DB, DBDNAME=ED00DBD, PROCOPT=A, . . .
SENSEG  NAME=COURSE
```

The following program requires two PCBs: the first PCB is used for the READ loop (which reads all COURSE segments) and the second nested FIND loop (which finds one offering to a given course); the second PCB is used for the first FIND loop (which updates a specific COURSE segment). The program does not work if the order of the two PCBs is reversed.

```
READ COURSE BY COURSENO
  FIND (1) COURSE WITH COURSENO = '120'
    UPDATE WITH TITLE = 'Natural'
  LOOP
  FIND (1) OFFERING WITH COURSENO-COURSE = COURSENO (0010)
    DISPLAY COURSENO-COURSE
  LOOP
LOOP
END
```

The following figure shows the logical connections between DL/I PSBs, PCBs, sensitive segment types and Natural DDMs:



Natural DDMs which are derived from segment descriptions in the DBD correspond to DL/I segment types.

Since each DL/I application program requires the specification of its sensitive segment types, an appropriate PSB must be scheduled before Natural program execution. A PSB can be scheduled at the start of a Natural session or at any time during the session.

If, in the configuration shown in the diagram above, PSB-2 has been scheduled, only the DDMs DDM1 and DDM2 are accessible to Natural application programs. If an attempt is made to use DDM5, for example, Natural for DL/I returns the error message:

NAT3768 PCB with requested DBD not found in NSB.

## Procedure NATDBD

Every DL/I database structure, both physical and logical, which is supposed to be used by Natural, must be processed by the Natural batch utility NDPBNDB0.

This utility stores DL/I database information on the FDIC system file, in a form suitable for Natural. This information is referred to as *NDB control block*. A batch procedure called NATDBD has been established for this purpose.

A sample NATDBD job has been included in the source library from the installation tape. The information used to create NDB control blocks comes from the actual DBDGEN source. It is essential that the same input is used for the NATDBD procedure as was used for the DL/I DBDGEN. Otherwise, unpredictable results are likely.

The NATDBD job is a three step procedure:

- The first step executes the normal DL/I DBDGEN procedure. This step is included to guarantee compatibility between DL/I and Natural.
- The second step performs another assembly and link of the DBDGEN source, this time using macros supplied by Natural.
- The final step executes the Natural batch utility NDPBNDB0, which uses the linked DBD module from the previous step to create NDB control blocks which are stored on the FDIC system file. NDPBNDB0 dynamically loads the Natural module NDLB0001, which therefore must be present in an allocated load library.

The NATDBD procedure assigns a short name of two bytes to each DL/I field; that is, to each field defined in the DBD. All field short names are generated in the range from NA to Z9, which means that up to  $13 * (26 + 10) = 468$  DL/I fields can be managed per DBD. DL/I short names are generated uniquely within an NDB.

When replacing an NDB, NATDBD reassigns short names in a consistent way; that is, the same short name to the same field name. In addition, the UDFs are maintained, where the new NDB contains the new DL/I layout followed by the old UDF layout, which means that UDFs are not deleted by NATDBD. It is the administrator's responsibility to edit the segment description after NATDBD has been executed, in order to modify the UDFs accordingly.

## Using Logical Databases with Natural

The following information must be considered when using logical databases with Natural:

- Execute the NATDBD procedure for a logical database only after successful execution of the procedure for the physical databases referred to. In other words, if the input DBD is a "logical" DBD, the NDBs generated from the "physical" DBDs must already be stored in the Natural FDIC system file to correctly generate the NDB control blocks related to this segment.
- When a segment specifying the SOURCE=keyword is processed by the NATDBD procedure, the related "physical" DBD must already be stored in the Natural FDIC system file.  
If the SOURCE=keyword is specified (in one or more segments) in a "physical" DBD, which means that one or more logical virtual child segments are involved (recursively or not), the NATDBD procedure run against this DBD stores the NDB structure on the Natural FDIC system file even if one or more physical DBDs referred to by the SOURCE=keywords have not already been stored.  
In this case, the logical virtual child segments whose source DBD is not yet in the Natural FDIC system file as well as their descendants are not accessible to the user since Natural has marked these segments as inhibited. An appropriate Natural error message is issued indicating the name(s) of the related physical DBD(s) that need to be stored into the Natural system file.  
If the logical relationship is the result of a recursive database structure, the NATDBD procedure for the physical DBD must be run at least twice: the first time, the NDB is stored on the Natural system file with the undefined segment marked as inhibited; the second time, the reference to the SOURCE segment is resolved.  
If multiple physical databases are logically related, the NATDBD procedure must be run for each of these physical databases and then rerun for any database that contained logical child segments marked as inhibited.
- If the SOURCE=keyword is specified in a "logical" DBD and one or more source DBDs are not found in the Natural FDIC system file while running the NATDBD procedure, the NDB structure is not stored and an appropriate error message is returned.
- If an attempt is made to generate a DDM for a segment whose NDB control blocks are not in the Natural FDIC system file, a Natural error message is returned.

## Using Index Databases with Natural

The following information must be considered when using index databases with Natural:

- To access a secondary index database as data, the secondary index database must be defined as an independent physical database to both DL/I and Natural.
- The NATDBD procedure need not be executed for primary or secondary index DBDs.

## Procedure NATUDF

The DBDGEN source usually does not define all fields within a segment. Additional segment fields called User-Defined Fields (UDFs) can be entered as part of creating the DDMs. UDFs define the additional data in the segment that can be referenced by a Natural program. UDFs can be generated online using Predict or the Natural utility SYSDDM, or they can be generated in batch mode using the NATUDF procedure.

The NATUDF procedure invokes the batch utility NDPBCUDF, which stores segment description layout information on an FDIC system file.

**Important:**

Before NDPBCUDF can be executed, the DL/I DBD must have been stored as an NDB on the FDIC system file, and a DBID and FNR must have been assigned (with Predict or SYSDDM) to each segment concerned. Otherwise, NDPBCUDF cannot read the segments concerned.

The input for this utility is provided by the segment description read from a work file. This work file contains segment identification statements and segment field descriptions.

You can format data by using either delimiter mode (IM=D) or forms mode (IM=F); see also the IM profile parameter in the Natural Parameter Reference documentation. In delimiter mode, the delimiter character can be used. In forms mode (for example, if input is passed from other programs), input data fields are assumed to be in contiguous storage and must be filled up to the internally defined full length.

One line is required for the segment identification statement, and two lines are required for each segment field description.

The section below covers the following topics:

- Segment Identification Statement
- Segment Field Description

## Segment Identification Statement

One line has to be supplied for each segment being defined. The following syntax is used (the parameters must be specified in the sequence shown below):

**FUNC**=( *function* ), **DBD**=*dbd-name* , **SEGM**=*segment-name*

|                     |  |
|---------------------|--|
| <i>function</i>     | Function to be applied to the segment:<br><br>ADD to create a new segment layout;<br>REP to replace an existing segment layout;<br>MOD to add or modify fields without deleting existing fields not present in the input file;<br>END to indicate termination of the UDF redefinition. |
| <i>dbd-name</i>     | A 1 to 8 character alphanumeric DBD name; that is, the name of the DL/I DBD which owns the segment to be defined.  |
| <i>segment-name</i> | A 1 to 8 character alphanumeric name of the DL/I segment to be defined.  |

## Segment Field Description

The segment identification statement has to be followed by at least one segment field description. The following syntax is used for each field to be defined (the parameters must be specified in the sequence shown below):

**FUNC=FLD, NAME=fnam, TYPE=type, LEVEL=lev, LENGTH=lgh, MAXOCC=moc, VAR=var**  
**FUNC=STR, BEGIN=begin**

After each FLD card, a STR card must be coded, except for the last FLD card, which is specified with four dollar signs (\$\$\$\$) in the field name. After this last FLD card, an END card must be coded.

|              |   |
|--------------|---|
| <i>fnam</i>  | The name of the field being defined. This must be an alphanumeric value of 1 to 19 bytes. The value "\$\$\$\$" closes the definition of the current segment.  |
| <i>type</i>  | The UDF field format (1 character). The following formats can be specified: A, B, F, P, U, N, S.  |
| <i>lev</i>   | The field level (1 digit).  |
| <i>lgh</i>   | The field length (4 digits).  |
| <i>moc</i>   | The maximum occurrence (3 digits) of the field (only applicable for a multiple-value field or a periodic group).  |
| <i>var</i>   | Possible values:<br><br>V   variable field length<br>N   fixed field length   |
| <i>begin</i> | The starting position of the field being redefined. This can be specified either in terms of bytes relative to the beginning of the segment or as a field name of the DL/I field being redefined. The value must be alphanumeric and 1 to 19 characters long (32 bytes in forms mode, as the field is 32 characters long in this mode). |

The short name is automatically assigned by the utility in the range from AA to G9, excluding EA to E9. The range from HA to M9 is reserved for UDFs of logical child segments. Thus, up to 216 fields can be provided as input, which is the maximum number of UDF fields.

For further information on UDF field parameters, please refer to DL/I Services.

**Delimiter Mode (IM=D) Example:**

---

**Forms Mode (IM=F) Example:**

```
ADDDBD1      SEGM1
FLD          1FIELD-1          000A0012N000
STR
FLD          1FIELD-ANY       000A    N000
STRFIELD-1
FLD          2FIELD-ANY2     000A0024N000
STR
FLD          $$$$
STR
REPDBD2      SEGM2
FLD          1NEW-FIELD-NAME  000A0012N000
STR
FLD          $$$$
END
```

**Sample JCL:**

```
//NATUDF   JOB   .....
//NATUDF   EXEC  PGM=NATBATCH,PARM='...'
//STEPLIB DD   DSN=...
//        DD   DSN=...
//SYSUDUMP DD   DUMMY
//CMPRINT DD   SYSOUT=Y
//DDCARD  DD   DSN=NAT23n.SRCE(ADAPARM),DISP=SHR
//CMSYNIN DD   *
LOGON  SYSDDM
NDPBCUDF
FUNC=REP,DBD=ED02DBD,SEGM=COURSE
FUNC=FLD,NAME=DUM1,TYPE=A,LEVEL=1,LENGTH=6
FUNC=STR,BEGIN=TITLE
FUNC=FLD,NAME=DUM2,TYPE=A,LEVEL=1,LENGTH=6
FUNC=STR,BEGIN=DESCRIPN
FUNC=FLD,NAME=$$$$
FUNC=END
FIN
```

## Generation of DDMs from DL/I Segment Types

DDMs that represent DL/I segment types are generated from information contained in the NDB and UDF control blocks. These DDMs contain all fields that have been defined for the segment, both in the NDB and in the UDF.

In addition, the DDMs contain the fields from the ancestor segments that have been defined in the DBDGEN for these segments. Ancestor segments are defined as segments that form the hierarchical path from the root segment down to the current segment. Ancestor segment fields that might have been defined in the DBDGEN for a segment include sequence fields, secondary index fields and search fields.

The DDM for a DL/I segment contains all fields that could be specified in the segment search argument (SSA), all fields that are available as part of the key feedback area and any segment I/O fields as well. Each DDM, therefore, contains all the fields that Natural requires to automatically build the concatenated key for the segment.

Once all fields have been defined for a specific segment DDM, the corresponding Natural DDM can be generated and cataloged (stored) on the Natural FDIC system file. This is done either with Predict or with the Natural utility SYSDDM.

If you do not have Predict installed, use the SYSDDM function DL/I Services to generate Natural DDMs from DL/I segment types. This function is invoked from the main menu of SYSDDM.

# System File Structure

As described in section Accessing DL/I Data, certain information must be stored and maintained on the Natural FDIC system file in order to access DL/I data. This information describes the database structure, the segment data and the processing intent of an application. Four elements on the Natural FDIC system file contain this information. One of these elements, the Natural DDM, is common to all DBMS environments. The remaining three elements, however, are used only by Natural for DL/I; they are NDB control blocks, NSB control blocks and UDF control blocks. Therefore, the Natural FDIC system file used by Natural for DL/I contains three subfiles:

- The NDB Subfile
- The NSB Subfile
- The UDF Subfile

This section also provides information on:

- Natural for DL/I Objects
- Displaying Keys of UDF Blocks
- Displaying the Size of NDL Objects
- Displaying NDL Objects
- Control Blocks in Separate Buffer Pool
- Control Blocks in Buffer Pool Blacklist
- Natural for DL/I Objects and Natural DDMs

## The NDB Subfile

The NDB subfile contains the NDBs. The NDB, or Natural DBD, control blocks contain most of the information present in the DL/I DBD, combined with additional data used by Natural, such as the file number (FNR) and database identification (DBID) of the segment, and short names for fields defined in the DBD. The NDB control blocks are created and stored on the Natural FDIC system file by the NATDBD procedure.

An NDB consists of the following fields:

| Field | Description   |
|-------|---|
| ND    | DBD name (8 characters) combined with sequence number (1 byte, "binary").   |
| NC    | The first two bytes contain the number of NZ fields in the record times 20. The second two bytes contain the total number of NZ fields in the NDB multiplied by 20. |
| NZ    | NDB data.   |

## The NSB Subfile

The NSB subfile contains the NSBs. The NSB, or Natural PSB, control blocks contain most of the information present in the DL/I PSB. These control blocks are created and stored on the Natural FDIC system file by the NATPSB procedure.

An NSB consists of the following fields:

| Field | Description   |
|-------|---|
| NP    | PSB name (8 characters) combined with sequence number (1 byte, "binary").   |
| NC    | The first two bytes contain the number of NZ fields in the record times 20. The second two bytes contain the total number of NZ fields in the NSB multiplied by 20. |
| NZ    | NSB data.   |

## The UDF Subfile

The UDF subfile contains the UDFs. The UDF, or User-Defined Field, control blocks contain information on segment fields which have been specified by the user, either through the online DL/I Services function of the SYSDDM utility, the NATUDF procedure, or by using Predict.

The fields are as follows:

| Field | Description  |
|-------|--|
| NS    | Database identification (1 byte, "binary"), file number (1 byte, "binary") and sequence number (1 byte, "binary"). The DBID and FNR are those of the segment being described by this record. |
| NC    | The first two bytes contain the number of NZ fields in the record times 20. The second two bytes contain the total number of NZ fields in the UDF multiplied by 20.                          |
| NZ    | Field description as specified by the user using Predict, the "EDIT segment layout" facility of SYSDDM or the procedure NATUDF.  |
| NW    | The long field name.   |

## Natural for DL/I Objects

Natural for DL/I objects are created during execution of the NATPSB procedure (NSB), during execution of the NATDBD procedure (NDB), or when assigning DBID/FNR to a segment type (UDF). Consequently, at least one UDF block for each segment type with an assigned DBID/FNR is always present on FDIC - whether User-Defined Fields (UDF fields) have been defined by the user or not.

When displaying type definitions in SYSDDM, the NDB and its related UDF are combined automatically. The only way to display an UDF separately (for debugging purposes) is by using NDLBLOCK.

## Displaying Keys of UDF Blocks

The utility program NDULIST, cataloged in the library SYSDDM, is provided for listing the keys of all UDF blocks and for checking for duplicates.

For each duplicate found the following warning is issued:

**More than one record with same DBID/FNR.**

## Displaying the Size of NDL Objects

The following utility programs, cataloged in library SYSDDM, are provided for displaying the sizes of the various NDL objects:

- NDLSIZED displays the sizes of all NDBs stored on FDIC.
- NDLSIZEP displays the sizes of all NSBs stored on FDIC.
- NDLSIZEU displays the sizes of all UDFs stored on FDIC.

## Displaying NDL Objects

The utility program NDLBLOCK, cataloged in library SYSDDM, is provided for displaying the NDBs, NSBs and UDFs stored on FDIC. The utility displays the objects in hexadecimal format.

## Control Blocks in Separate Buffer Pool

The Natural for DL/I control blocks NDB, NSB and UDF are read from FDIC and loaded into a buffer pool - resident or not, depending on the NDLPARM parameters RESINDB, RESINSB, and RESIUDF. This allows a given object to be shared by several users.

By means of the NTBPI macro (as described in Parameter Modules in the Natural Parameter Reference documentation) it is possible to have a buffer pool for NDB, NSB and UDF control blocks which is different from the buffer pool for Natural programs, thus allowing for better isolation between the different Natural objects.

If a separate buffer pool is allocated, Natural for DL/I locates its control blocks in this buffer pool. Otherwise, they are located in the Natural buffer pool.

The "Individual Object Statistics" function of the SYSBPM utility displays the NDB, NSB and UDF control blocks kept in the buffer pool as follows:

|     | Library         | DBID | FNR |
|-----|-----------------|------|-----|
| NDB | SYSDLIND        | 255  | 253 |
| NSB | SYSDLINS        | 255  | 253 |
| UDF | <i>Ummmmnnn</i> | 255  | 253 |

**Note:**

The library names of NDB and NSB are fixed internal names and are not related to any Natural library.

The DBID/FNR values are fixed internal values and are not related to any Natural system file.

"mmm" is the DBID of the corresponding segment, "nnn" is the FNR of the corresponding segment.

The "Display Object Hexadecimally" function of the SYSBPM utility also allows you to display Natural for DL/I objects. This function might be useful when in doubt if the expected object has been read from FDIC, or if the object has been read from the expected FDIC (test/production).

## Control Blocks in Buffer Pool Blacklist

The Natural for DL/I control blocks NDB, NSB and UDF can be added to the buffer pool blacklist.

This is done by the "Blacklist Maintenance" function of the SYSBPM utility.

As "Library" you enter "SYSDLIND" for NDBs, "SYSDLINP" for NSBs, and "SYSDLINS" for UDFs.

As "Object" you enter the NDB name for NDBs, the NSB name for NSBs, and *Ummmmnnn* for UDFs where *mmm/nnn* are the DBID/FNR of the corresponding segment.

This feature allows you to modify NDBs, NSBs or UDFs without causing unpredictable results for active users.

If an attempt is made to load a locked object into the buffer pool, Natural for DL/I will issue error message NAT3935.

## Natural for DL/I Objects and Natural DDMs

When referencing a DDM in a Natural program, Natural translates the DDM name into the corresponding DBID/FNR pair. If this DBID identifies the DDM as a DL/I DDM (by means of the NTDB macro), the Adabas control block is passed to Natural for DL/I for further processing.

Natural for DL/I takes DBID from the control block and tries to locate an UDF with this DBID/FNR in the buffer pool. If it is not found there, it is read from FDIC and loaded into the buffer pool.

The UDF contains the name of the related NDB in its header. Using this name, Natural for DL/I tries to locate the NDB in the buffer pool. If it is not found there, it is read from FDIC and loaded into the buffer pool.

The segment description including all DL/I fields is part of the NDB.

From this it is clear that:

- the NDB/UDF is required during runtime,
- the relation between the Natural program and the related NDB is established by means of DBID/FNR only.

This implies that the DBA has to ensure that DDMs and NDBs are always kept in synchronization. For example, it is not sufficient to transfer only the Natural programs from test to production.

# Natural Batch Utilities

This section covers the following topics:

- Transfer of NDBs/NSBs/UDFs from one System File to Another
  - Utility NDUDFGEN for Natural Data Areas
- 

## Transfer of NDBs/NSBs/UDFs from one System File to Another

- Unloading the NDBs, NSBs and UDFs
- Loading NDBs, NSBs and UDFs
- Selecting NDBs, NSBs and UDFs from a Dataset

The transfer of NDBs, NSBs and UDFs from one FDIC system file to another is performed either online using the utility SYSMAIN (as described in the Natural Utilities for Mainframes documentation) or in two batch steps, using two Natural batch utilities provided for this purpose:

- With the ULDDLI unload utility, the NDBs, NSBs and UDFs are transferred from one FDIC system file to a sequential work file.
- With the INPLDLI load utility, the NDBs, NSBs and UDFs are transferred from the sequential work file to another FDIC system file.

Both programs, ULDDLI and INPLDLI, are contained in the library SYSDDM.

## Unloading the NDBs, NSBs and UDFs

The utility ULDDLI is used to unload NDBs, NSBs and UDFs from an FDIC system file to a sequential work file.

ULDDLI requires the following input:

- the specification of the FDIC system file to be unloaded (either in the NATPARM module or dynamically) and
- one or more parameter lines containing the following:
  - **Function code** (A1); the following function codes can be specified:

|   |   |
|---|---|
| A | All NSBs, NDBs and UDFs are unloaded.   |
| D | All NDBs with valid object names and their UDFs are unloaded. If no object names are specified, all NDBs and their UDFs are unloaded. |
| P | All NSBs with valid object names are unloaded. If no object names are specified, all NSBs are unloaded.                               |
| U | All UDFs with valid object names are unloaded. If no object names are specified, all UDFs are unloaded.                               |
| . | Terminate ULDDLI; at least one parameter card with function code "." is required.   |

- **Object name** (A8); 0 - 6 occurrences.

**Note:**

With UDFs, the object name must be in the form "*nmn\*\*nmn*"; that is, a 3-digit database ID, followed by 2 asterisks, followed by a 3-digit file number.

Work files: CMWKF01 DD card must be provided with:

```
DCB=(RECFM=VB , LRECL=4624 , BLKSIZE=4628 )
```

When ULDDLI is executed, the specified NDBs, NSBs and UDFs are written from the FDIC system file to the CMWKF01 dataset.

**Note:**

DL/I fields of a segment are part of the NDB block and not of the UDF block, which means that you must still transfer the entire NDB block if you have modified a DL/I field in a segment.

**Example 1 - Unload the NDBs TESTDB1 and TESTDB2:**

```
LOGON SYSDDM
ULDDLI D TESTDB1 TESTDB2
.
```

**Example 2 - Unload all UDF Blocks:**

```
LOGON SYSDDM
ULDDLI U
.
```

**Example 3 - Unload UDF Blocks with DBID 10/FNR 150 and DBID 246/FNR 3:**

```
LOGON SYSDDM
ULDDLI U 010**150 246**003
.
```

## Loading NDBs, NSBs and UDFs

The utility INPLDLI is used to load NDBs, NSBs and UDFs - previously unloaded with ULDDLI - from the work file to an FDIC system file.

INPLDLI requires the following input:

- the specification of the FDIC system file into which the NDBs, NSBs and UDFs are to be loaded (either in the NATPARM module or dynamically);
- (optionally) the parameter "DEL=Y":  
If you specify "DEL=Y", all existing NDBs and UDFs found on the FDIC system file are first deleted. The ones contained on the input work file are added to the file. NSB definitions contained on the work file replace any identically named NSBs on the FDIC system file.  
If you do not specify "DEL=Y", existing identically named NDBs and NSBs are not replaced. Existing UDFs which have been allocated identical DBID/FNR combinations are not replaced either. Non-existent definitions are added. If you do not specify "DEL=Y", it may occur that an NDB is loaded but all or some of its segments (UDFs) are not, or that segments (UDFs) are loaded without the corresponding NDB being loaded.
- (optionally) the parameter "REP=Y": If you specify "REP=Y", NDBs, NSBs and UDFs contained on the work file replace any identically named NDBs, NSBs and UDFs on the FDIC system file.

"DEL=Y" and "REP=Y" are mutually exclusive. If neither "DEL=Y" nor "REP=Y" is specified, existing NDBs, NSBs and UDFs are neither deleted nor replaced.

Work files: CMWKF01 DD card must be assigned to the work file which was created by the utility program ULDDLI.

When INPLDLI is executed, the NDBs, NSBs and UDFs are loaded from the work file into the specified FDIC system file, depending on whether they already exist and on whether "DEL=Y" was specified.

### Example:

```
LOGON SYSDDM
INPLDLI REP=Y
```

## Selecting NDBs, NSBs and UDFs from a Dataset

The utility SELDLI allows you to select NDL objects (NDBs, NSBs, UDFs) from a dataset created by the ULDDLI utility. The output of SELDLI can be used as input for INPLDLI. Since INPLDLI does not allow to select objects from a dataset created by ULDDLI, you can use SELDLI to perform this function on desired objects prior to running INPLDLI.

SELDLI can, therefore, be used for backup/recovery or transfer of selected objects from test to production.

SELDLI also supports a SCAN (command SCN) feature that will list all of the objects on the input dataset without selecting any for output.

SELDLI can be used in batch mode only.

SELDLI requires the following input:

- the specification of the output dataset CMWKF01 from ULDDLI
- up to 30 parameter lines containing the following:
  - Object type (A3); the following types can be specified:
    - NSB - Select specified NSB
    - NDB - Select specified NDB
    - NDU - Select specified NDB and related UDF
    - UDF - Select specified UDF
    - SCN - List input dataset CMWKF01
  - terminate SELDLI
- Object name (A8); 1 occurrence

**Note:**

With NDB/NSB, a wildcard (\*) can be specified at the end of the name to select a range of names.

With UDFs, the object name must be in the form "*nnn\*\*nnn*"; that is, a 3-digit database ID, followed by 2 asterisks, followed by a 3-digit file number.

SELDLI provides the following output:

- Dataset containing selected objects to be used as input to INPLDLI. It is specified with DDNAME "CMWKF02".

When SELDLI is executed, the specified NDBs, NSBs and UDFs are copied from CMWKF01 to CMWKF02.

**Example 1 - Select all NDBs:**

```
LOGON SYSDDM
SELDLI
NDB,*
.
FIN
```

**Example 2 - Select NSB "ORDPSB" and UDF for DBID 151, FNR 3:**

```
LOGON SYSDDM
SELDLI
NSB,ORDPSB
UDF,151**003
.
FIN
```

**Example 3 - Select NDB "CUSTDBD" and its related UDFs:**

```
LOGON SYSDDM
SELDLI
NSB,ORDPSB
NDU,CUSTDBD
.
FIN
```

**Example 4 - List all objects on the input dataset:**

```
LOGON SYSDDM
SELDLI
SCN
FIN
```

## Utility NDUDFGEN for Natural Data Areas

The batch utility NDUDFGEN can be used to generate Natural data areas.

Input is provided by a UDF definition read from a work file.

Two kinds of data areas can be generated:

- a Natural view,
- a data structure (local data area).

A view in a local data area is generated from all fields contained in the input work file. The utility normalizes the data to the requirements of a view according to the Natural syntax. The field lengths are adapted to Natural field lengths, multiple-value fields and periodic groups are generated from record data structures. Arrays are generated by NDUDFGEN with the maximum length allowed by Natural. Field definitions are collected into a redefinition and the redefined field is generated according to the length of the individual fields collected. The generated field can then be used in the segment description as UDF; this means that not all UDFs need to be defined in the segment description, but only the generated fields.

A data structure as local data area is generated of all input fields. A level increment value can be specified for the fields. No other modifications to the input file data are permitted, so that the data are generated as specified in the input file.

### Input for NDUDFGEN

The input layout is similar to the one for the NDPBCUDF utility.

The first card is the definition card; it contains the definition which is valid for all of the UDF definitions.

The "FLD" cards contain the actual field definitions and are separated from each other by "STR" cards.

The "END" card indicates the end of the field definitions. The input is required in forms mode (IM=F) as follows:

| Definition Card | Explanation   |
|-----------------|---|
| Bytes 1 - 3     | The first 3 bytes are not used.   |
| Bytes 4 - 11    | These 8 bytes contain the DBD name.   |
| Bytes 12 - 19   | These 8 bytes contain the segment name.   |
| Bytes 20 - 27   | These 8 bytes contain a prefix (generated for fields).  |
| Bytes 28 - 30   | These 3 bytes contain the maximum occurrence (default is 191).  |
| Byte 31         | This byte contains either "S" if a data structure is to be generated or "V" if a view is to be generated. |
| Byte 32         | This byte contains the level increment.   |

| <b>Field Card</b> | <b>Explanation</b>  |
|-------------------|---|
| Bytes 1 - 3       | The first 3 bytes contain "FLD".  |
| Bytes 4 - 19      | These 16 bytes are not used.  |
| Byte 20           | This byte contains the field level.   |
| Bytes 21 - 39     | These 19 bytes contain the name of the field being defined. This must be an alphanumeric value.                 |
| Bytes 40 - 42     | These 3 bytes are not used.   |
| Byte 43           | This byte contains the format of the field.   |
| Bytes 44 - 47     | These 4 bytes contain the byte length of the field.   |
| Byte 48           | This byte is not used.  |
| Byte 49 - 52      | This byte contains the length as required by Natural (if this length is specified, the byte length is ignored). |
| Byte 53 - 57      | These 4 bytes contain the maximum size of the 1st dimension of an array.  |
| Byte 58 - 62      | These 4 bytes contain the maximum size of the 2nd dimension of an array.  |
| Byte 63 - 66      | These 4 bytes contain the maximum size of the 3rd dimension of an array.  |

**Example 1 - View Generation:**

```

DBDNAME SEGMENT PREFIX 191V
FLD          1VAR1          000A0745
STR
FLD          1GROUP        000A0000N0000000200020000
STR
FLD          2VAR2          000A0006N00060005
STR
FLD          2VAR3          000A0030
STR
END
    
```

The above input generates the following view:

|               |       |         |                            |       |                 |
|---------------|-------|---------|----------------------------|-------|-----------------|
| 13:38:41      | ***** | E D I T | DATA                       | ***** | 89-05-03        |
| Library: XYZ1 | Name: | LOCAL   |                            |       | DBID: 10 FNR: 5 |
| Command:      |       |         |                            |       | > +             |
| I T L Name    |       | F Leng  | Index/Init/EM/Name/Comment |       |                 |
| -----         |       |         |                            |       |                 |
| 1 VAR1        |       | A 149   | (5)                        |       |                 |
| 1 GROUP       |       |         | (4)                        |       |                 |
| 2 VAR2        |       | A 6     | (5)                        |       |                 |
| 2 VAR3        |       | A 30    |                            |       |                 |

**Example 2 - Structure Generation:**

```

DBDNAME SEGMENT PREFIX 191S
FLD          1VAR1          000A0745
STR
FLD          1GROUP        000A0000N0000000200020000
STR
FLD          2VAR2          000A0006N00060005
STR
FLD          2VAR3          000A0030
STR
END
    
```

The above input generates the following data structure:

|               |                      |         |                            |       |                 |
|---------------|----------------------|---------|----------------------------|-------|-----------------|
| 13:41:20      | *****                | E D I T | DATA                       | ***** | 89-05-03        |
| Library: XYZ1 | Name:                | LOCAL   |                            |       | DBID: 10 FNR: 5 |
| Command:      |                      |         |                            |       | > +             |
| I T L Name    |                      | F Leng  | Index/Init/EM/Name/Comment |       |                 |
| -----         |                      |         |                            |       |                 |
| V 1           | DBDNAME-SEGMENT-VIEW |         | DBDNAME-SEGMENT            |       |                 |
| M 2           | VAR1                 | A 149   | (5)                        |       |                 |
| P 2           | GROUP                |         | (4)                        |       |                 |
| 3             | PREFIX-1             | A 60    | /*PREFIX-1                 |       |                 |
| R 3           | PREFIX-1             |         |                            |       |                 |
| 4             | VAR2                 | A 6     | (5)                        |       |                 |
| 4             | VAR3                 | A 30    |                            |       |                 |

# Execution

This section covers the following topics:

- PSB Scheduling
  - CALLNAT Interface
  - Support of IMS-Specific Features
  - Fast Path Support
  - Support of GSAM
  - Processing in CICS Pseudo-Conversational Mode under IMS/TM
- 

## PSB Scheduling

In all environments, Natural must know the name of the scheduled PSB, not only the address of the PCB list. In the online environments, the application developer must have the ability to change the scheduled PSB during a Natural session. This is accomplished by the Natural command NATPSB (in batch or CICS environments) or by calling CMDEFSWX/CMDIRSWX (in IMS/TM environments).

- The NATPSB Command

PSB scheduling in Natural depends upon the actual operating environment. Therefore, this section is further subdivided into:

- PSB Scheduling in a Batch Environment
- PSB Scheduling in a CICS Environment
- PSB Scheduling in an IMS/TM Environment

### The NATPSB Command

The NATPSB command handles PSB scheduling status and can be invoked with one of the following three options:

| Option            | Description   |
|-------------------|---|
| INQ               | Performs an inquiry on PSB scheduling status.                   |
| ON <i>psbname</i> | Issues a PSB schedule of the PSB <i>psbname</i> .               |
| OFF               | Issues a syncpoint to commit all updates and terminate the PSB. |

**Note:**

The NATPSB INQ command is valid in an IMS/TM environment, too.

The following command, for example, issues a PSB schedule of ED00PSB:

**NATPSB ON ED00PSB**

A PSB scheduling operation is allowed only if there is no active PSB. If a PSB is active and another PSB is to be scheduled, the "ON" request for this new PSB must be preceded by an "OFF" request. Otherwise, the following message is issued:

**NAT3900 PSB ... scheduled, but PSB ... already active**

Since NATPSB is actually a Natural program, it can also be invoked with a FETCH or FETCH RETURN statement. The options described above should then be passed in the FETCH statement as two parameters. The first parameter would be an alphanumeric field of three bytes for "INQ", "ON" or "OFF". If the first parameter is "ON", the second parameter must also be passed. It is an alphanumeric field of eight bytes and contains the name of the PSB to be scheduled.

Execution time errors of NATPSB can be intercepted by an ON ERROR statement. The error messages from NAT3900 to NAT3903 and from NAT3817 to NAT3820 are generated by NATPSB.

**Example:**

```

FETCH RETURN 'NATPSB' 'ON' 'PBNDL01'
ON ERROR
  IF *ERROR = 3900                               /* PSB already scheduled
    STACK TOP COMMAND 'NATPSB' 'ON' 'PBNDL01'
    STACK TOP COMMAND 'NATPSB' 'OFF'
  STOP
END-IF
END-ERROR
END

```

## PSB Scheduling in a Batch Environment

To execute a batch program that accesses a DL/I database, it is necessary to use the DL/I batch procedure which executes an application program under DL/I control. Therefore in the JCL/JCS used to execute Natural batch accessing DL/I databases, the first program in the step is a DL/I system program (DFSRRRC00 for OS/390, DLZRRC00 for VSE/ESA).

PSB scheduling is performed by DL/I before control is passed to Natural. Since Natural requires the name of the scheduled PSB, it is necessary to invoke the Natural PSB scheduling program NATPSB before executing a Natural application program. This can be achieved by specifying the command NATPSB ON *psbname* as the first command in the batch input stream to Natural.

### Batch Execution under OS/390

Under OS/390, the DL/I region controller program (DFSRRRC00) invokes the NDLSINIB bootstrap module for Natural for DL/I by specifying MBR=NDLSINIB in the PARM field of the EXEC card. NDLSINIB reads two statements from the NDINPUT DD card:

- Statement 1 contains the name of the Natural module to be executed.
- Statement 2 contains the dynamic Natural parameters.

Before executing the user program, the command "NATPSB ON *psbname*" must be specified in the input stream to pass the name of the current PSB to Natural.

**Example 1 - OS/390 with Adabas System File:**

```
//          EXEC DLIBATCH,PSB=psbname,MBR=NDLSINIB
//G.STEPLIB DD ...          Steplibs
//G.NDINPUT DD *           Input for NDLSINIB
natbatch                    Natural load module name
STACK=(LOGON user),DU=ON    Any Natural parameters
//DDCARD DD *             Primary input file
ADARUN MODE=MULTI,PR=USER   ADARUN cards
//G.CMSYNIN DD *          Primary input file
NATPSB ON psbname          Mandatory Natural PSB scheduling
pgmname                     Natural user program name
/*                          End of Natural commands
```

**Example 2 - OS/390 with VSAM System File:**

```
//          EXEC DLIBATCH,PSB=psbname,MBR=NDLSINIB
//G.STEPLIB DD ...          Steplibs
//G.NDINPUT DD *           Input for NDLSINIB
natbatch                    Natural load module name
STACK=(LOGON user),DU=ON    Any Natural parameters
//G.CMSYNIN DD *          Primary input file
NATPSB ON psbname          Mandatory Natural PSB scheduling
pgmname                     Natural user program name
/*                          End of Natural commands
```

In both examples, *natbatch* is assumed to be the load module produced by the respective link-edit procedure.

**Batch Execution under VSE/ESA**

Under VSE/ESA, the DL/I region controller program (DLZRR00) invokes the NDLSINID bootstrap module for Natural for DL/I.

The SYSIPT cards are as follows:

- DL/I control statements:
  - DLI,NDLSINID, psbname**
  - natbatch*
  - where:
    - **DLI** is a parameter for DLZRR00,
    - **NDLSINID** is the name of the bootstrap module,
    - *psbname* is the name of the PSB,
    - *natbatch* is the name of the Batch Natural nucleus;
- dynamic parameters to be passed to Natural;
- ADARUN statements (only if Adabas system file is being used);
- Natural input cards.

A "/\*" delimiter card is required before the ADARUN statements (if present) and before the Natural dynamic parameters and input cards.

Before executing the user program, the "NATPSB ON *psbname*" command must be specified in the input stream to pass the name of the current PSB to Natural.

**Example 1 - VSE/ESA with Adabas System File:**

```

// EXEC DLZRRRC00
DLI,NDLSINID,psbname          DL/I control statements
natbatch                      Batch Natural nucleus name
/*
STACK=(LOGON user),DU=ON      Any Natural parameters
/*                              End of Natural parameters
ADARUN MODE=MULTI,PR=USER     ADARUN cards
/*                              End of ADARUN cards
NATPSB ON psbname             Mandatory Natural PSB scheduling
pgmname                       Natural user program name
/*                              End of Natural commands

```

**Example 2 - VSE/ESA with VSAM System File:**

```

// EXEC DLZRRRC00
DLI,NDLSINID,psbname          DL/I control statements
natbatch                      Batch Natural nucleus name
/*
STACK=(LOGON user),DU=ON      Any Natural parameters
/*                              End of Natural parameters
NATPSB ON psbname             Mandatory Natural PSB scheduling
pgmname                       Natural user program name
/*                              End of Natural commands

```

In both examples, *natbatch* is assumed to be the load module produced by the respective link-edit procedure.

## PSB Scheduling in a CICS Environment

Under CICS, the PSB must be scheduled using the NATPSB command, which actually invokes the appropriate scheduling or termination calls.

The active PSB can be changed dynamically during the Natural session using the NATPSB command. Therefore, more than one PSB can be used during a Natural session. Only one PSB, however, can be active for a CICS task at a time.

The NATPSB command can be entered in the Natural command line or passed to Natural dynamically with the Natural STACK statement when starting a Natural session.

### Examples:

```
MOVE 'STACK=(NATPSB ON ED00PSB)'
    TO DYNAMIC-PARM-KEYWORD-LIST.
EXEC CICS
    XCTL PROGRAM('NAT31n')
END-EXEC.
```

This example taken from a COBOL/CICS program assumes that NAT31n is the value supplied for the PROGRAM keyword in the CICS PPT.

Another possibility is to assign NATPSB commands to one or more PF keys when starting a Natural session as illustrated in the following example:

```
NATD STACK=(KEY PF1 = ED00PSB)
```

This example assumes that "NATD" is the value supplied for the TRANSID keyword in the CICS PCT. ED00PSB is the following Natural program (cataloged in the library SYSTEM):

```
STACK TOP COMMAND 'NATPSB ON ED00PSB'
STACK TOP COMMAND 'NATPSB OFF'
END
```

Whenever PF1 is pressed, the commands "NATPSB OFF" and "NATPSB ON ED00PSB" are executed.

## PSB Scheduling in an IMS/TM Environment

Under IMS/TM, Natural for DL/I runs as a conversational transaction. It has the ability to perform direct or deferred message switching. This means that several different Natural transactions and PSBs can be invoked during a single Natural session. It is also possible to invoke multiple PSBs and provide the user with access to databases defined in different PSBs. This is accomplished by calling CMDEFSWX or CMDIRSWX.

Under IMS/TM, PSB scheduling is performed by the IMS Control Region before control is passed to the Natural transaction running as an MPP (Message Processing Program) or BMP (Batch Message Processing). As in the batch environment, Natural needs to know the name of the scheduled PSB. This is accomplished internally at Natural session start by the driver which stores the pointer to the PCB address list and the name of the PSB into IOCB fields. The NATPSB INQ command can be issued in this environment but the NATPSB ON/OFF commands cannot.

## CALLNAT Interface

The Natural subprograms NDLPBAD and NDLPBSC are provided, which can be invoked with a CALLNAT statement from within a Natural program.

See the following sections:

- The NDLPBAD Subprogram
- The NDLPBSC Subprogram

### The NDLPBAD Subprogram

The Natural subprogram NDLPBAD provides the calling Natural program with the name of the currently scheduled PSB and the pointer to the PCB address list.

**Example:**

```
DEFINE DATA LOCAL
01 PSBNAME (A8)
01 PCBADDR (B4)
END-DEFINE
CALLNAT 'NDLPBAD' PSBNAME PCBADDR
DISPLAY PSBNAME PCBADDR
END
```

This pointer can then be used by non-Natural programs to obtain the individual PCB addresses and to establish addressability to the PCBs. For example, move these addresses to the BLL cells (COBOL/VS) or use the SET ADDRESS instruction (COBOL II).

## The NDLPBSC Subprogram

The Natural subprogram NDLPBSC allows for scheduling a PSB in CICS or batch environments. It performs the same functions as the NATPSB command.

Using CALLNAT 'NDLPBSC' (instead of FETCH RETURN 'NATPSB') avoids the NAT1108 error message, which is issued if a PSB is scheduled in an INPUT loop as follows:

```
INPUT ...
FETCH RETURN 'NATPSB' 'ON' 'psbname'
REINPUT ... /* returns NAT1108
```

### Example:

```
DEFINE DATA LOCAL
01 COMMAND (A3)
* 'ON'
* 'OFF'
* 'INQ'
01 PSBNAME (A8)
01 RETCODE (B1)
* 01: Command invalid
* 02: PSB name missing
* 03: PSB psbname active
* 04: PSB psbname not active
* 05: Not used
* 06: No PSB active
END-DEFINE
MOVE 'ON' TO COMMAND
MOVE 'psbname' TO PSBNAME
CALLNAT 'NDLPBSC' COMMAND PSBNAME RETCODE
DISPLAY PSBNAME RETCODE
END
```

Under IMS/TM, NDLPBSC can only be used with parameter 'INQ', because PSB scheduling is performed by the IMS control region before control is passed to Natural.

## Support of IMS-Specific Features

This section covers the following topics:

- Symbolic Checkpoint/Restart Functions - CHKP, XRST
- The INIT Call to Enable Data Availability Status Codes

### Symbolic Checkpoint/Restart Functions - CHKP, XRST

A Natural program can make use of the IMS/TM symbolic checkpoint and restart facilities by using the statements GET TRANSACTION DATA and END TRANSACTION.

The executing program can checkpoint user data on the IMS system log datasets by supplying an 8-byte checkpoint ID as the first operand in the END TRANSACTION statement and by specifying the areas to be checkpointed as additional operands.

To ensure that the checkpoints are written to the IMS log dataset, the Natural profile parameter ETDB (see the Natural Parameter Reference documentation) must be specified, and the database specified with the ETDB parameter must be a DL/I database.

If no operands are specified with the END TRANSACTION statement, Natural uses "NATDLICK" as the default checkpoint ID.

This checkpoint data are retrieved by executing the GET TRANSACTION DATA statement. The first operand of this statement must also be an 8-byte checkpoint ID. The remaining operands must be listed in the same sequence, length and format as in the corresponding END TRANSACTION statement.

#### Example:

```

RESET CKPID(A8) KEY(A10) AREA1(A20) AREA2(N6) AREA3(A120)
GET TRANSACTION DATA CKPID KEY AREA1 AREA2 AREA3
IF CKPID NE ' ' /* checkpoint restart
    MOVE KEY TO START-KEY(A10)
ELSE
    RESET START-KEY /* normal restart
MOVE *PROGRAM-ID TO CKPID
:
READ DLI-DB BY XKEY> START-KEY
:
UPDATE
:
END TRANSACTION CKPID XKEY AREA1 AREA2 AREA3
:
END

```

|                     |  |
|---------------------|--|
| Normal Restart:     | Simply run the job. The checkpoint ID parameter in the program's GET TRANSACTION DATA statement is set to blanks by the DL/I call handler NDLSIOBA.  |
| Checkpoint Restart: | To restart after an abnormal termination, specify one of the following checkpoint IDs in the PARM field of the EXEC statement in your program's JCL:<br><br>CKPTID=LAST    to restore data areas written to the log by the job at the last successful checkpoint; or<br><br>CKPTID=ccccccc to restore data areas written with checkpoint ID ccccccc. |

These are the usual IMS/TM restart procedures. Each checkpoint ID used in an END TRANSACTION statement is displayed in the job output once the extended checkpoint has been successfully executed by IMS.

The checkpoint ID parameter of the program's GET TRANSACTION DATA statement is set to the actual checkpoint ID used by IMS.

The data areas are restored into the areas you specify in your GET TRANSACTION DATA statement.

Ensure that the //IMSLOGR DD statement specifies the correct IMS log dataset.

When Natural is started in a BMP region, the initialization routine issues an XRST call, to ensure that symbolic checkpointing is available. This is done whether the Natural user programs to be executed make use of IMS symbolic checkpoint logic or not. If the XRST was unsuccessful, Natural returns the following error message:

**NAT3959 XRST call failed with DL/I status code xx**

When a GET TRANSACTION DATA statement is directed to the Natural call handler and the initial XRST call has been flagged as successfully executed, the restart checkpoint ID and contents of this buffer are copied into the program's user fields.

When an END TRANSACTION statement is directed to the Natural call handler, the user fields to be checkpointed are copied into the buffer before a symbolic checkpoint call (CKPT) is issued.

If the database specified with the profile parameter ETDB (see the Natural Parameter Reference documentation) is not the same as the database affected by the transaction, the first operand of the END TRANSACTION statement will be used as checkpoint ID for the ETDB database, while "NATDLICK" will be used as checkpoint ID for the other database **not** specified with the ETDB parameter.

The total area to be checkpointed must not exceed 1992 bytes.

## The INIT Call to Enable Data Availability Status Codes

If the INITCAL parameter of NDLPARM is set to "YES", Natural issues an INIT call during session initialization and during each MPP transaction start. The character string in the I/O area is "STATUS GROUPE". This informs IMS that Natural is prepared to accept status codes regarding data unavailability. IMS returns status codes BA or BB when the DL/I call requires access to unavailable data (for example, if the accessed database has been stopped).

The corresponding error messages of Natural for DL/I are:

**NAT3897 DL/I status code 'BA'**

**NAT3898 DL/I status code 'BB'**

For compatibility reasons, the default setting of INITCAL is "NO".

The INIT call is issued only if Natural runs in a BMP or MPP region.

## Fast Path Support

Natural supports Fast Path databases.

Fast Path database types include Main Storage Databases (MSDB) and Data Entry Databases (DEDB).

- **MSDB:**  
MSDBs have root only segments that are fixed-length. There are two types of MSDBs: terminal-related and non-terminal-related.  
To read segments in an MSDB GU and GN are used.  
To update segments in an MSDB REPL, DLET, ISRT, and FLD are used.
- **DEDB:**  
DEDBs use the design concept that database content can be physically partitioned by ranges of root keys or by groupings produced by a randomizing algorithm.

As a basic requirement, the non-conversational NATIMS driver must be used. This is because Fast Path programs cannot be conversational programs, i.e., they cannot use an SPA.

For DEDB databases, no special processing is required by Natural for DL/I.

For MSDB databases, the (one and only) SSA is built without command codes because DL/I does not allow for it (not even the null command code must be used in case of MSDB databases).

When updating segments in an MSDB database, Natural for DL/I uses the REPL call (rather than the FLD call) because the UPDATE statement of the Natural language does not provide a search condition that indicates which segments must be updated (searched update).

## Support of GSAM

Natural for DL/I supports the Generalized Sequential Access Method (GSAM), with which a sequential dataset can be handled as a sequential non-hierarchic database by IMS.

Although GSAM databases have no segments, keys or parentage, they are handled internally by Natural as root-only databases with fixed or variable-length segment types. Thus, it is possible to use DDMs instead of work files for GSAM record types.

For variable-length GSAM records, Natural maintains the record length; you need not reserve a field for the record length in the DDM.

A FIND or READ statement generates a GN (get next) call sequence for GSAM. Due to GSAM restrictions, UPDATE and DELETE statements are not allowed. Due to GSAM restrictions, a STORE statement must insert records at the **end** of the database.

IMS repositions GSAM databases for sequential processing, which means that the position need not be re-established by the application program after checkpoint calls. Therefore, Natural performs no repositioning after checkpoint calls in the case of PCBs for GSAM.

In order to use the extended restart feature of IMS, the Natural job has to terminate abnormally. This can be accomplished by calling the Natural IMS/TM service module CMSVC13D. If the job terminates either normally or with a condition code, IMS does a clean-up and no restart is possible.

Every GSAM database structure which is to be used by Natural must be processed by the NATDBD procedure. The assembly step of this procedure extracts the relevant information from the DBD source and simulates an appropriate SEGM statement as shown in the following examples.

**Example 1 - Segment Description of Fixed-Length GSAM Records:**

```

DBD      NAME=TESTDB , ACCESS= ( GSAM , BSAM )
DATASET DD1=INPUT , DD2=OUTPUT , RECFM=F , RECORD=80
DBDGEN
END

```

From the above source statements, NATDBD would simulate a segment with the name of the DBD and the length as specified with the RECORD keyword:

```

SEGM NAME=TESTDB , BYTES=80

```

**Example 2 - Segment Description of Variable-Length GSAM Records:**

```

DBD      NAME=TESTDB , ACCESS= ( GSAM , BSAM )
DATASET DD1=INPUT , DD2=OUTPUT , RECFM=VB
DBDGEN
END

```

From the above source statements, NATDBD would simulate a segment with the name of the DBD, a maximum length of 32760 and a minimum length of 8:

```

SEGM NAME=TESTDB , BYTES= ( 32760 , 8 )

```

In both examples, the NDB name and the segment name are "TESTDB", and the generated DDM name would be "TESTDB-TESTDB".

The Natural program to read this GSAM database would be as simple as:

```

READ TESTDB-TESTDB
  DISPLAY FIELDS-OF-TESTDB
LOOP
END

```

## Processing in CICS Pseudo-Conversational Mode or under IMS/TM

When Natural is running under CICS in pseudo-conversational mode (that is, with NATPARM parameter PSEUDO=ON) or under IMS/TM, the Natural task/transaction is terminated following each write to a terminal, and a new task/transaction is started when new input is entered through the terminal. Because a syncpoint is forced at the end of the task/transaction, all resources are released when the message is sent to the terminal. Therefore, the DL/I PSB is no longer active, nor are any DL/I GET HOLD calls in effect.

To avoid consistency problems on the DL/I databases, Natural performs additional processing when it is running in CICS pseudo-conversational mode or under IMS/TM:

1. If a DL/I GET HOLD call is still active at the end of the task/transaction, the values of the fields read by the program that issued the corresponding READ or FIND (only the fields used, not the whole segment) are saved in an internal table of Natural for DL/I.
2. When a new task/transaction resumes the Natural session and the program issues an UPDATE or DELETE statement, Natural checks whether the field contents have been changed. If the check shows that the field contents have not been changed, the UPDATE/DELETE is executed. If they have been changed, an error message is returned by Natural notifying the user that the field values just read were changed by another user in the system and that, therefore, the UPDATE/DELETE operation is not carried out.

Natural also performs automatic PSB repositioning following resumption of the task/transaction. A Natural application is, therefore, not affected by pseudo-conversational mode, unless it uses conventional programming techniques, for example COBOL or PL/I.

If the task/transaction is terminated due to a screen I/O while a READ or FIND loop is being executed on a segment without a unique sequence field, Natural is not able to reposition the PSB in the database when the task/transaction is resumed. The same may occur when using secondary indices with non-unique key fields in pointer segments. Natural is not able to reposition the PSB in these instances because DL/I does not provide a method of re-establishing position in the middle of non-unique keys or non-keyed segments.

# Programming Language Considerations

This section covers the following topics:

- Natural versus Third Generation Languages
  - Natural Statements with DL/I
  - Natural System Variables with DL/I
- 

## Natural versus Third Generation Languages

With a few exceptions Natural provides all of the functionality of third generation language programming in the DL/I environment.

However, accessing DL/I data using Natural is significantly different from programming techniques used in a third generation language. Natural application programmers do not have to code specific DL/I calls or build the segment search arguments (SSAs). They do not need to concern themselves with PCB mask information or keep track of PCB positioning between syncpoints.

Natural for DL/I operates as a standard DL/I application and although most of the DL/I call processing is done internally, it is important to realize that all of the required DL/I processing is still performed:

PSBs are scheduled and terminated, PCBs are selected for use, database positioning is maintained, SSAs are created, the most efficient DL/I calls are issued, PCB mask information is evaluated, GET HOLD calls are issued before update or delete operations.

These tasks are all being performed for the application by Natural.

It is important to note that Natural is performing these tasks based on the information available in the application program. If, for example, a READ or FIND statement in a program is lacking essential segment search information, Natural selects a PCB, builds an SSA and issues a certain DL/I call based on this lacking information.

The Natural programmers use the same Natural statements to manipulate data in DL/I as they would for VSAM, Adabas or DB2.

Natural accesses DL/I segments based on the Natural DDM which is being referenced. Since the data access is always for one specific segment type (the one defined by the DDM), Natural does not issue path calls nor unqualified calls; that is, calls where the segment name is not specified.

**Note:**

Due to the structure of the Natural programming language, application control over DL/I call command codes is not available.

The LOG, STAT and GSCD call functions are not supported for the IMS/TM environment.

## Natural Statements with DL/I

- BACKOUT TRANSACTION
- DELETE
- DISPLAY
- END TRANSACTION
- FIND
- GET TRANSACTION DATA
- READ
- RELEASE
- STORE
- UPDATE
- WRITE
- Statements not Available for DL/I

This section mainly consists of information also contained in the Natural Statements documentation, where each Natural statement is described in detail, including notes on DL/I usage where applicable. Summarized below are the particular points a programmer has to bear in mind when using Natural statements with DL/I.

Any Natural statement not mentioned in this section can be used without restrictions with DL/I.

### BACKOUT TRANSACTION

This statement is used to back out all database updates performed during the current logical transaction.

How the statement is translated and which command is actually issued depends on the TP-monitor environment:

- Under CICS, the BACKOUT TRANSACTION statement is translated into an EXEC CICS ROLLBACK command.  
However, in pseudo-conversational mode (PSEUDO=ON), only changes made to the database since the last terminal I/O are undone. This is due to CICS-specific transaction processing.
- In batch mode and under IMS/TM, Natural for DL/I issues ROLB calls without checking the CMPAT setting in the corresponding NSB.  
However, under IMS/TM, only changes made to the database since the last terminal I/O are undone. This is due to IMS/TM-specific transaction processing.

Because PSB scheduling is terminated by a syncpoint/checkpoint request, Natural saves the PCB position before executing the BACKOUT TRANSACTION statement. Before the next command execution, Natural reschedules the PSB and tries to set the PCB position as it was before the backout.

#### Note:

The PCB position might be shifted forward if any pointed segment had been deleted in the time period between the BACKOUT TRANSACTION and the following statement.

### DELETE

The DELETE statement is used to delete a segment from a DL/I database, which also deletes all descendants of the segment.

### DISPLAY

The DL/I AIX fields can be displayed only if a PCB is used with the AIX specified in the parameter PROCSEQ. If not, an error message is returned by Natural for DL/I at runtime.

## END TRANSACTION

This statement indicates the end of a logical transaction and releases all DL/I data locked during the transaction. All data modifications are committed and made permanent.

How the statement is translated and which command is actually issued depends on the TP-monitor environment:

- Under CICS, the END TRANSACTION statement is translated into an EXEC CICS SYNCPOINT command.
- In batch mode and non message-driven BMP environments, Natural for DL/I issues CHKP calls without checking the CMPAT setting in the corresponding NSB.
- In MPP and message-driven BMP environments, the END TRANSACTION statement is not translated into a CHKP call, but is ignored, because CHKP calls imply GU calls. As Natural is a conversational transaction, you must reply to the terminal before requesting the next message (that is, before issuing the next GU call). An implicit end-of-transaction is issued after each terminal I/O.

Because PSB scheduling is terminated by a syncpoint/checkpoint request, Natural saves the PCB position before executing the END TRANSACTION statement. Before the next command execution, Natural reschedules the PSB and tries to set the PCB position as it was before the END TRANSACTION statement.

### Note:

The PCB position might be shifted forward if any pointed segment had been deleted in the time period between the END TRANSACTION and the following command.

With batch-oriented BMP regions, user data can be checkpointed on the IMS system log data sets. This is done by supplying an 8-byte checkpoint ID as the first operand in the END TRANSACTION statement, and by specifying the areas to be checkpointed as additional operands.

If the database specified with the profile parameter ETDB (as described in the Natural Parameter Reference documentation) is not the same as the database affected by the transaction, the first operand of the END TRANSACTION statement will be used as checkpoint ID for the ETDB database, while "NATDLICK" will be used as checkpoint ID for the other database **not** specified with the ETDB parameter.

The total area to be checkpointed must not exceed 1992 bytes; see also Symbolic Checkpoint/Restart Functions.

## FIND

With DL/I, the Natural FIND statement is typically used when a specific search criterion is known and specific segments are to be retrieved. This issues a DL/I GET UNIQUE call. However, if the FIND statement specifies a lower level segment and is within an active READ or FIND loop for an ancestor segment, it generally results in a DL/I GET NEXT WITHIN PARENT call.

The FIND statement initiates loop processing, which is active until all segment occurrences which match the search criterion have been read.

When accessing a field starting after the last byte of the given segment occurrence, the storage copy of this field is filled according to its format (numeric, blank, etc.).

FIND FIRST, FIND NUMBER and FIND UNIQUE are not permitted. The PASSWORD, CIPHER, COUPLED and RETAIN clauses are not permitted either.

In the WITH clause, you can only use descriptors that are defined as key fields in DL/I and marked with "D" in the DDM.

When connecting search criteria, the following has to be observed:

$$[\text{NOT}] \left\{ \begin{array}{l} \textit{basic-search-criterion} \\ \textit{(search-expression)} \end{array} \right\} \left[ \left\{ \begin{array}{l} \text{OR} \\ \text{AND} \end{array} \right\} \textit{search-expression} \right] \dots$$

Connecting search criteria for segment type A results in multiple qualification statements within one DL/I segment search argument (SSA). Connecting search criteria for segment types A and B results in multiple SSAs. Therefore, the Boolean operator OR cannot be used to combine search criteria for different segment types.

## GET TRANSACTION DATA

This statement retrieves checkpoint data saved by an END TRANSACTION statement. The first parameter of this statement must be an 8-byte checkpoint ID. The remaining operands must be listed in the same sequence, length and format as in the corresponding END TRANSACTION statement; see also Symbolic Checkpoint/Restart Functions.

## READ

The READ statement should be used to process a set of segment occurrences in sequential order and usually results in a DL/I GET NEXT call.

When the READ statement is used, segments are retrieved based on the sequence field of the root segment or based on a secondary index field. Since the READ statement initiates sequential access of the database, it is important to understand that the EQUAL TO clause means the same thing as the STARTING FROM clause; it initiates a sequential read loop beginning with the key value specified.

The READ statement initiates loop processing. A loop is active until all segment occurrences which match the search criterion have been read.

The PASSWORD and CIPHER clauses are not permitted.

READ IN PHYSICAL SEQUENCE is used to read records in the order in which they are physically stored in a database. The physical sequence is the default sequence.

**Note:**

This is only valid when using Natural with HDAM databases.

READ BY ISN is not valid when using Natural with DL/I.

For Natural, the descriptor used must be either the sequence field of the root segment or a secondary index field. If a secondary index field is specified, it must also be specified in the PROCSEQ parameter of a PCB. Natural uses this PCB and the corresponding hierarchical structure to process the database.

## RELEASE

The RELEASE statement is not applicable for DL/I usage, since it releases sets of records retained by a FIND statement that contained a RETAIN clause, which is not valid when using Natural with DL/I.

## STORE

This statement can be used to add a segment occurrence.

If the segment occurrence is defined with a primary key, a value for the primary key field must be provided.

In the case of a GSAM database, records must be added at the end of the database (due to GSAM restrictions).

The USING/GIVING NUMBER clause is not valid when using Natural with DL/I.

If the SET/WITH clause is used, the following applies with Natural for DL/I:

- Values must be provided for the segment sequence field and for all sequence fields of the ancestors.
- Only I/O (sensitive) fields can be provided.
- A segment of variable length is stored with the minimum length necessary to contain all fields as specified with the STORE statement. The segment length will never be less than the minimum size specified in the SEGM macro of the DBD.
- If a multiple-value field or a periodic group is defined as variable in length, at the end of the segment only the occurrences as specified in the STORE statement are written to the segment and define the segment length.

## UPDATE

This statement can be used to update a segment in a DL/I database. The segment length is increased (if necessary) to accommodate all fields specified with the UPDATE statement. If a multiple-value field or a periodic group is defined as variable in length, only the occurrences as specified in the UPDATE statement are written to the segment.

The DL/I AIX field name cannot be used in an UPDATE statement. AIX fields, however, can be updated by referring to the source field which comprises the AIX field.

DL/I sequence fields cannot be updated because of DL/I restrictions.

If the SET/WITH clause is used, only I/O (sensitive) fields can be provided. A segment sequence field cannot be updated (DELETE and STORE must be used instead).

Due to GSAM restrictions, the UPDATE statement cannot be used for GSAM databases.

## WRITE

With the WRITE statement, the DL/I AIX fields can be displayed only if a PCB is used with the AIX specified in the parameter PROCSEQ. If not, an error message is returned by Natural for DL/I at runtime.

## Statements not Available for DL/I

The following Natural statements are not available for DL/I users:

- GET
- GET SAME
- HISTOGRAM
- PASSW
- RELEASE

## Natural System Variables with DL/I

With DL/I, the following restrictions apply to the following Natural system variables:

### **\*ISN**

As there is no DL/I equivalent to Adabas ISNs, the system variable \*ISN is not available with Natural for DL/I.

### **\*NUMBER**

With Natural for DL/I, the Natural system variable \*NUMBER does not contain the number of segment occurrences found. It contains 0 if no segment occurrence satisfies the search criterion and a value of 8,388,607=X'7FFFFFF' if at least one segment occurrence satisfies the search criterion.

# Problem Determination Guide

The items listed below are cross-referenced by Natural for DL/I error messages. They are supplied to advise Natural programmers, DL/I database administrators and system support personnel of actions required to correct a given problem.

## Item 1: Activate Natural Trace Facility for DL/I

How to activate the Natural trace facility for DL/I:

- **Dynamic trace activation:**  
Execute the command NDLTRACE in library SYSDDM as follows:  
**NDLTRACE ON** *parm1 parm2 parm3*  
Permitted values for trace parameters are either CMD, SER, ROU (according to the specifications in the given error message) or ALL to trace all events of Natural for DL/I.
- **Initial trace activation:**
  1. Either code the TRACE parameter in the NDLPARM module according to the specifications in the given error message or specify TRACE=ALL to trace all events of Natural for DL/I.
  2. Assemble the NDLPARM module.
  3. Link-edit the load module that contains Natural for DL/I.

How to create and display the Natural trace for DL/I:

1. Start the Natural session with DSIZE=64 (or smaller). This is required because the trace data is written into in the DSIZE buffer.
2. Activate the trace facility (see above) and specify the following commands:

```
TEST DBLOG D   Start DBLOG for DL/I.  
...           Reproduce your problem here.  
TEST DBLOG D   Display the data logged.
```

### Note:

The Natural trace facility for DL/I is available in all Natural for DL/I environments.

**Item 2: Obtain the Program Listing****Item 3: Obtain the View Listing****Item 4: Obtain the DBD Macros****Item 5: Obtain the PSB Macros****Item 6: Obtain the NDB Description Printout**

To obtain the printout, execute the Natural module NDLBLOCK in the library SYSDDM with the following parameters:

- block type (3 bytes alphanumeric) = NDB
- block name (8 bytes alphanumeric) = *dbd-name*

**Item 7: Obtain the NSB Description Printout**

To obtain the printout, execute the Natural module NDLBLOCK in the library SYSDDM with the following parameters:

- block type (3 bytes alphanumeric) = NSB
- block name (8 bytes alphanumeric) = *psb-name*

**Item 8: Obtain the UDF Description Printout**

To obtain the printout, execute the Natural module NDLBLOCK in the library SYSDDM with the following parameters:

- block type (3 bytes alphanumeric) = UDF
- block name (8 bytes alphanumeric) = *db-id\*\*file-number*  
(that is, 3 digits for the database ID, a literal separator "\*\*" and 3 digits for the *file-number*)

**Item 9: Obtain a DUMP****Item 10: Obtain the NDLPARM Listing****Item 11: Obtain the NATDBD Procedure Output****Item 12: Obtain the NATPSB Procedure Output**

# Performance Considerations

This section lists some special considerations which may help you increase the performance of your Natural for DL/I environment.

- Parameters
  - Global and Local Data Areas
  - FIND Statements
  - Direct Access to Lower Levels
  - DBLOG Utility
- 

## Parameters

Set the DLISIZE parameter to 0 if no DL/I database is to be accessed.

Do not modify NDLPARM parameters, unless requested by a corresponding Natural for DL/I error message. Unused buffers are compressed by the Natural compression algorithm.

## DBID

Use the same DBID for all segment types (DDMs) of a given NDB, because an OPEN command is generated for each DBID.

## Global and Local Data Areas

Keep global and local data areas as small as possible, because the format buffer contains **all** fields of the global and local data areas, not only those which are referenced by a Natural I/O statement.

## FIND Statements

If the sequence field is unique, use a FIND(1) statement instead of a FIND statement to prevent an unnecessary second DL/I call.

## Direct Access to Lower Levels

Access segments on lower levels directly (by using the field sequence of the parent); that is, access ancestor segments only if their contents are required by the application program.

In such cases, UDFs of ancestor segments as well as DL/I fields of ancestor segments which are not sequence fields are not available to the application program.

## DBLOG Utility

Use the Natural utility DBLOG utility ("TEST DBLOG D") to tune your application; see Logging Database Calls (DBLOG) in the section Debugging and Monitoring.

# DL/I Services

When you invoke "DL/I Services" from the SYSDDM main menu, the DL/I Services Main Menu is displayed which offers you the following functions:

- NDB Maintenance  
An NDB is a DL/I DBD (database description) which is defined to Natural.
- NSB Maintenance  
An NSB is a DL/I PSB (program specification block) which is defined to Natural.

## NDB Maintenance

This section covers the following topics:

- Menu and Functions
- Select an NDB from a List
- Select an NDB Segment from a List
- Edit an NDB Segment Description
- Generate DDM from Segment Description

### Menu and Functions

When you select NDB Maintenance on the DL/I Services Main Menu, the NDB Maintenance menu is displayed:

```

14:37:12                **** DL/I Services ****                97-02-15
                        - NDB Maintenance -

Code Functions
-----
S  Select an NDB from a List
P  Purge an NDB
L  Select an NDB Segment from a List
E  Edit an NDB Segment Description
G  Generate DDM from Segment Description
?  Help
.  Back
M  End
-----

Enter Code: ?
NDB Name:
Segment Name:

ENTER  PF1  PF2  PF3  PF4  PF5  PF6  PF7  PF8  PF9  PF10  PF11  PF12
      Help      Back                                End

```

The individual NDB maintenance functions are listed below:

| Function                              | Explanation   |
|---------------------------------------|---|
| Select an NDB from a List             | List the NDBs which are defined on the Natural system file.<br>You can then select NDBs from this list by entering the following function codes:<br><br>P to purge an NDB,<br><br>L to list the segments of an NDB.   |
| Purge an NDB                          | Purge an NDB and its related segment descriptions from the Natural system file. The name of the NDB to be purged must be specified.<br>Before this function is executed you are prompted to confirm the purge request.  |
| Select an NDB Segment from a List     | List the segments of the specified NDB. You can then select segments from this list for further processing.   |
| Edit an NDB Segment Description       | Edit a segment description. The segment name and its corresponding NDB name are required when invoking this function.<br>A database ID (DBID) and file number (FNR) must have been assigned to the segment description (function code "A" on the Segment List display) before it can be edited. |
| Generate DDM from Segment Description | Generate a DDM from a segment description. The DDM definition is a Natural DDM of the segment. Prior to execution of this function, a DBID and FNR must have been assigned to the segment (function code "A" on the Segment List display).  |

### Select an NDB from a List

When you select an NDB from a list, a list containing all NDBs defined on the Natural system file is displayed. In addition to the NDB name the following is displayed:

|        |  |
|--------|--|
| L/P    | Indicates if ACCESS=LOGICAL or not.          |
| length | Length of the NDB.                           |
| NoSGMS | Number of the segment types in the NDB.      |
| ACCESS | The access specification taken from the DBD. |

From the list, you can select NDBs for further processing by entering the following function codes in the "Func" column next to the NDB Names:

| Code | Function  |
|------|---|
| P    | <b>Purge NDB</b><br>This function is identical to the "Purge NDB" function available on the NDB Maintenance menu. It deletes an NDB and its related segment descriptions from the Natural system file. Before the function is executed you are prompted to confirm the purge request. |
| L    | <b>List NDB Segments</b><br>This function is identical to the "Select NDB Segment from a List" function available on the NDB Maintenance menu. It lists the segments of the selected NDB. For details, see Select an NDB Segment from a List.   |

## Select an NDB Segment from a List

When you select an NDB segment from a list, a list containing all segments of the specified NDB is displayed. If you do not know the NDB name, use the "Select an NDB from a List" function.

```

10:50:48                **** DL/I SERVICES ****                97-08-20
                        - Segment List -

DBD Name = ED00DBD
  Func   Level  Segment      DBID   FNR    Seg-Lgh      UDF-Lgh  Response
----- Top of Data -----
  _       1    COURSE        246    _10    75-80         100
  _       2    PREREQ         246    _11    36-36          40
  _       2    OFFERING       246    _12    41-41          40
  _       3    TEACHER        246    _13    24-24          60
  _       3    STUDENT        246    _14    40-40          40
  _
  _
  _
  _
  _
  _
  _
----- Bottom -----
Code .. _ ( ? Help   . Back   M End )
Func = E Edit Segment Description   A Assign DBID and FNR
      F Free DBID and FNR           ' ' Change DBID and FNR
      G Generate DDM                 N Take New Copy of UDF

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
Exec Help           Exit                               Canc

```

Next to each segment you can enter one of the function codes listed below. You can mark several segments at the same time with a function code. If you do not enter any code, the list is scrolled forward until the bottom of the list is reached.

You can enter one of the following codes next to a segment on the segment list to perform one of the following functions:

| Code         | Function   |
|--------------|--|
| A            | <p><b>Assign DBID/FNR</b><br/>                     Assign a DBID and a FNR to the selected segment.<br/>                     The DBID is a number in the range from 1 to 254. It must be contained in the database ID list (NTDB macro) of the Natural parameter module NATPARM. All Natural DDMs which refer to a DL/I segment must have a DBID belonging to this range. If a DBID has not been entered, a default value is assigned, which is the entry with the lowest value in the DBID list specified in the Natural parameter module. For a given NDB, all segments should be assigned the same DBID. Otherwise, Natural assumes different databases and generates an OPEN command (which, however, is ignored by the DL/I call handler).<br/>                     The FNR is a number in the range from 1 to 254. The FNR must be specified; no default value is assumed by Natural. The segments of a logical NDB must have file numbers different from those assigned to the segments of the physical NDB.<br/>                     The DBID/FNR combination must be unique on the Natural system file. It is used by Natural to uniquely determine the NDB and the segment within the NDB.</p> |
| E            | <p><b>Edit Segment Description</b><br/>                     Edit the description of a segment within a given NDB. The function is the same as the "Edit an NDB Segment Description" function which you can invoke from the NDB Maintenance menu. Before you can edit a segment description, a DBID and FNR must have been assigned to the segment (see above).</p>   |
| F            | <p><b>Free DBID/FNR</b><br/>                     Release the DBID and FNR which have previously been assigned to the segment. Once a DBID and FNR have been released, they are available for assignment to another segment.</p>  |
| G            | <p><b>Generate DDM</b><br/>                     Generate a DDM definition from a segment description. The function is the same as the "Generate DDM from Segment Description" function which you can invoke from the NDB Maintenance menu.<br/>                     Before you can generate a DDM from a segment description, a DBID and FNR must have been assigned to the segment (see above).<br/>                     The generated DDM is a Natural view of the segment. Once it has been generated, the DDM can be modified and cataloged.</p>   |
| N            | <p><b>Take New Copy of UDF</b><br/>                     Refresh the user-defined fields (UDFs) of a segment when the UDFs of the source segment have been changed. This applies to UDFs of segments belonging to a logical NDB and to UDFs of logical virtual children. Before you can execute this function, a DBID and FNR must have been assigned to the segment (see above).</p>   |
| <i>blank</i> | <p><b>Change DBID/FNR</b><br/>                     Change a previously assigned DBID and/or FNR.<br/>                     For changing a DBID or FNR, the same rules concerning DBID and FNR specification apply as for assigning a DBID/FNR (see above).</p>  |

## Edit an NDB Segment Description

Additional segment fields, so-called *user-defined fields (UDFs)*, can be defined.

This function is invoked either by entering function code "E", an NDB name and a segment name on the NDB Maintenance menu, or by selecting the segment from the "Segment List" (by marking it with function code "E"). A DBID and a FNR must have been assigned to a segment description (function code "A" on the Segment List display) before it can be edited.

| EDIT command: |     | DBD | ED00DBD      | SEGMENT | STUDENT | SEGLGH | 40-40     |
|---------------|-----|-----|--------------|---------|---------|--------|-----------|
| ALL           | LEV | SN  | FIELD NAME   | START   | DLI     | MAXOCC | FOR LGH V |
|               | 1   | PM  | EMPNO        | 00001   | SQU     | A      | 6         |
|               | 1   | PN  | NAME         | 00007   | SRC     | A      | 33        |
|               | 1   | PO  | GRADE        | 00040   | SRC     | A      | 1         |
|               | 1   | AA  | BIRTHDATE    | 00025   |         | A      |           |
|               | 2   | AB  | DATE-DD      |         |         | N      | 2         |
|               | 2   | AC  | DATE-MM      |         |         | N      | 2         |
|               | 2   | AD  | DATE-YY      |         |         | N      | 2         |
|               | 1   | AE  | BIRTHPLACE   |         |         | A      | 10        |
|               | 1   | AF  | STUDENT-NAME | PN      |         | A      | 18        |

The following information is displayed on the status line at the top of the screen:

|         |  |
|---------|--|
| DBD     | Name of the DBD which contains the edited segment.                       |
| SEGMENT | Name of the edited segment.  |
| SEGLGH  | Minimum and maximum length of the edited segment, separated by a hyphen. |

DL/I fields and user-defined fields are displayed as shown above. You can add, delete or modify UDFs. DL/I fields, however, can neither be added nor deleted. If the specification "TYPE=P" is included in the FIELD statement of the DL/I DBD, the format of the field can be changed from "P" (decimal packed unsigned) to "S" (decimal packed signed) on the edit segment description screen. FOR (format) is the only attribute of a DL/I field you can modify. In particular, it is not possible to change the name of a DL/I field, because it is used by Natural to build the segment search arguments (SSA). If the name of a DL/I field is to be changed, the field can be redefined as an UDF.

Edit commands are available to copy or delete single lines or to insert a group of empty lines. In addition, commands for scrolling forward or backward are provided. For details you can enter a question mark in the "command" field to display the corresponding help information.

After modification of segment field attributes you can save the description by entering "SAVE" in the "command" field.

The following field definition attributes are displayed and can be modified for user-defined fields:

| Attribute  | Description  |
|------------|--|
| LEV        | Level number used to define a group of fields.   |
| SN         | Short name of the field as used internally by Natural.   |
| FIELD NAME | Name of the field as used in the application programs.   |
| START      | Start position of the field in the segment.  |
| DLI        | Type of the DL/I field, as follows:<br><br>SIX secondary index field<br>SQU sequence field (unique)<br>SQM sequence field (multiple)<br>SRC search field |
| MAXOCC     | Maximum number of occurrences of a multiple field or periodic group.   |
| FOR        | Format of the field.   |
| LGH        | Length of the field.   |
| V          | Variable field length indicator.   |

Each user-defined field can be defined as follows:

| Field Type       | Description   |       |         |       |       |    |      |       |       |    |        |   |   |    |        |   |   |
|------------------|---|-------|---------|-------|-------|----|------|-------|-------|----|--------|---|---|----|--------|---|---|
| Elementary Field | A field that contains only one value in a single segment.<br>Example: Personnel number  |       |         |       |       |    |      |       |       |    |        |   |   |    |        |   |   |
| Multiple Field   | A field that can contain more than one value in a single segment. Reference to a particular value of a multiple field can be made by appending a one to three-digit subscript (value 1 - 191) to the field name.<br>Example: Languages - English, German, Italian   |       |         |       |       |    |      |       |       |    |        |   |   |    |        |   |   |
| Group            | A series of one or more adjacent fields that can be referenced with a single name (the group name). You can also refer to a single field of a group by specifying its name.<br>Example:<br><table style="margin-left: 40px; border: none;"> <tr> <td>01</td> <td>Address</td> <td>Group</td> <td>field</td> </tr> <tr> <td>02</td> <td>City</td> <td>Elem.</td> <td>field</td> </tr> <tr> <td>02</td> <td>Street</td> <td>"</td> <td>"</td> </tr> <tr> <td>02</td> <td>Number</td> <td>"</td> <td>"</td> </tr> </table> | 01    | Address | Group | field | 02 | City | Elem. | field | 02 | Street | " | " | 02 | Number | " | " |
| 01               | Address   | Group | field   |       |       |    |      |       |       |    |        |   |   |    |        |   |   |
| 02               | City  | Elem. | field   |       |       |    |      |       |       |    |        |   |   |    |        |   |   |
| 02               | Street  | "     | "       |       |       |    |      |       |       |    |        |   |   |    |        |   |   |
| 02               | Number  | "     | "       |       |       |    |      |       |       |    |        |   |   |    |        |   |   |
| Periodic Group   | A group which is repeated in multiple adjacent occurrences in a single segment. For a periodic group it is possible to refer to a range of occurrences (or a field within a periodic group) by specifying the first and the last occurrence number to be referenced (connected by a hyphen (-)) after the name and in ascending order. Multiple-value fields or periodic groups are not allowed within a periodic group.<br>Example: Several addresses  |       |         |       |       |    |      |       |       |    |        |   |   |    |        |   |   |

Since DL/I fields cannot be modified as described above (with the exception of FORMAT), they cannot be directly defined as a group. To define a DL/I field as a group, it is necessary to redefine it as a user-defined field which then can be redefined as a group. In a DDM, these user-defined fields must not be specified as descriptor fields. When a DDM is generated, the UDFs are marked as non-descriptor fields.

**Example - Redefinition of a DL/I Sequence Field as a Group:**

The description of the segment STUDENT within the DBD named ED00DBD is used as shown in the Segment List screen above:

| LEV | SN | FIELD NAME | START | DLI | NOCC | FOR | LGH | V |
|-----|----|------------|-------|-----|------|-----|-----|---|
| 1   | PM | EMPNO      | 00001 | SQU |      | A   | 6   |   |
| .   |    |            |       |     |      |     |     |   |
| .   |    |            |       |     |      |     |     |   |

If the DL/I sequence field PM is to be "structured", it must be redefined as a user-defined field ("AAAAA" in the figure below). This UDF can then be structured as required.

| LEV | SN | FIELD NAME | START | DLI | NOCC | FOR | LGH | V |
|-----|----|------------|-------|-----|------|-----|-----|---|
| 1   | PM | EMPNO      | 00001 | SQU |      | A   | 6   |   |
| .   |    |            |       |     |      |     |     |   |
| .   |    |            |       |     |      |     |     |   |
| 1   | AA | AAAAA      |       | PM  |      |     |     |   |
| 2   | AB | BBBBB      |       |     |      | A   | 3   |   |
| 2   | AC | CCCCC      |       |     |      | A   | 3   |   |
| .   |    |            |       |     |      |     |     |   |
| .   |    |            |       |     |      |     |     |   |
| .   |    |            |       |     |      |     |     |   |

The group field "AAAAA" has no FORMAT/LENGTH specified. The length of a group is set equal to the sum of all fields belonging to the group.

## UDF Parameters

For each user-defined field on the above screen, parameters can be specified as listed and described in the following table. The total length of all DL/I fields and user-defined fields must not exceed the segment length.

When attributes of a UDF are modified and an old copy of this UDF is contained in the shared UDF buffer pool, the old copy is marked "invalid". If the UDF is referred to again by a Natural program, the modified UDF is read from the Natural system file. Therefore, it is not necessary to restart the Natural session if a UDF has been modified. However, this applies only to physical UDFs; that is, to UDFs of a physical NDB. If a physical UDF is modified and a logical NDB refers to the appropriate segment type, the logical UDF is not marked "invalid" in the buffer pool. To invalidate a logical UDF it is necessary to restart the TP monitor or to execute function "N" (Take New Copy of UDF) of the Segment List screen on the appropriate segments in the logical NDB.

| Field              | Description   |
|--------------------|---|
| LEV (level number) | A one-byte value used to define a group. A field is a group only if the subsequent field has a higher level number. The field immediately after the last group element must have a lower level number. A group can be defined within another group. The level number of the first user-defined field must be 1.   |
| SN (short name)    | The name used internally by Natural to identify the field. It must be two bytes in length, the first character must be alphabetic in the range from A to G (E is not permitted). The second character can be alphanumeric (that is, up to 216 UDF fields can be defined). If the segment is a logical child, the first character must be alphabetic in the range from H to M. Short names must be unique among a segment type.  |
| FIELD NAME         | External field name, up to 19 bytes long.   |
| START              | The start position of the field in the segment. The position can be specified as absolute by giving a three-digit number or it can be specified as relative, by giving the short name of a previously defined field which is being redefined.<br>It is important to specify the start position for the first user-defined field; otherwise, a default of 1 is used, which may cause overlapping with previous DL/I fields. The default for all other user-defined fields is the position immediately after the previous field.<br>The redefinition of fields is possible only for fields which have the same level number. When the level is higher than 1 (that is, for a field inside a group), only the last field can be redefined with the same level number. An absolute position must not be specified for a field within a group. |
| MAXOCC             | The maximum number of occurrences of a multiple -value field or periodic group in a segment.  |
| FOR (format)       | Standard field formats are:<br><br>A Alphanumeric<br>B Binary<br>F Fixed Point<br>P Packed decimal unsigned; that is, the zone halfbyte of the last byte is X'F'.<br>S Packed decimal signed; that is, the zone halfbyte of the last byte is X'C' (positive) or X'B' (negative).<br>N Unpacked<br><br>A group has no format. When the group is referenced, the fields within the group are always returned (by Natural) according to the standard format of each individual field.  |

| Field               | Description   |
|---------------------|---|
| <p>LGH (length)</p> | <p>Field length is a three-digit number; it must not exceed the maximum length permitted. These are as follows:</p> <p style="padding-left: 40px;">253 bytes for alphanumeric fields (A),<br/>                     126 bytes for binary fields (B),<br/>                     4 bytes for fixed point (F),<br/>                     14 bytes for packed decimal unsigned (P),<br/>                     14 bytes for packed decimal signed (S),<br/>                     27 bytes for unpacked decimal (N)</p> <p>In addition, the length specified must not exceed the segment length. Length must not be specified for a group. The length of packed fields is the field length in bytes.</p>   |
| <p>V (variable)</p> | <p>Depending on its value, "V" or blank, this parameter indicates whether a field has a variable length. Fields can be specified as variable only if the segment is a segment of variable length.</p> <p>Only one field can be defined as variable within a given segment description.</p> <p>An elementary field can be specified as variable in length only if it is the last field in the segment. A multiple field or a periodic group can be specified as variable in length regardless of its position in the segment.</p> <p>When applied to a multiple field or a periodic group, a setting of "VARIABLE" means that the number of occurrences is not known at definition time; therefore, MAXOCC should be specified using the maximum expected value.</p> |

## Generate DDM from Segment Description

This function is invoked either by using the "G" function code of the NDB Maintenance menu - then an NDB name and a segment name must be specified -, or by selecting the segment from the "Segment List", by marking it with the "G" function code.

A DBID and a FNR must have been assigned to a segment description (function code "A" on the Segment List display) before a DDM can be generated.

The DDM is generated from a segment description and represents a Natural view of the segment. It must be generated and cataloged before the corresponding segment can be referenced by a Natural program. After generation, default options for field headers or edit masks (decimal positions) can be modified in the DDM. See Catalog DDM and Edit DDM in the Natural Utilities for Mainframes documentation for corresponding information.

It should be noted, however, that default options for field headers or edit masks (decimal positions) are stored with the DDM and not with the NDB or UDF. The data in the NDB or UDF reflects what is allowed by the DL/I FIELD macro in which the length can be specified only in bytes (decimals are not allowed). Consequently, when regenerating the DDM, prior modifications in the DDM must be applied again by the user.

In DL/I a program must be able to reference search fields, sequence fields and secondary index fields of ancestor segments in order to build a certain search criterion; therefore, DDMs for DL/I segments can also include fields which are not part of the actual physical segment.

To satisfy the requirements for DL/I processing, a DDM must contain all the fields which can be referenced. Therefore, the generated DDM can contain the following fields:

- DL/I sequence fields, search fields and secondary index fields of the current (physical) segment. These fields have been defined in the DBDGEN source for this segment. When the DDM is generated, information on these fields is obtained from the NDB control block for this segment. DL/I sequence fields and secondary index fields are marked as descriptor ("D"), search fields are marked as non-descriptor ("N"). All of these fields can be used to qualify search requests.
- DL/I sequence fields and secondary index fields of all the ancestor segments. These fields have been defined in the DBDGEN source for the ancestor segments. When the DDM is generated, information on these fields is obtained from the NDB control blocks for the ancestor segments. These fields are marked as descriptor ("D"). They can be used to qualify search requests.
- DL/I search fields of all the ancestor segments. These fields have also been defined in the DBDGEN source for the ancestor segments. When the DDM is generated, information on these fields is also obtained from the NDB control blocks for the ancestor segments. However, these fields are marked as superdescriptor ("S"). They can be used to qualify search requests.
- Fields of the current segment defined by the user (UDFs). When the DDM is generated, information on these fields is obtained from the UDF control blocks. These fields cannot be used to qualify search requests.

Fields of format "S" in the segment description (see UDF Parameters) generate format "P" in the DDM.

The following tables summarize how the various types of fields can be processed using Natural I/O statements. They illustrate which fields can be used to qualify search requests, and which fields can be used with the statements DISPLAY, UPDATE or STORE. In addition, the tables indicate whether the field in the generated DDM is marked as descriptor, superdescriptor or non-descriptor.

**Current Segment:**

| Type of field | FIND/READ | DISPLAY | UPDATE | STORE | Marked |
|---------------|-----------|---------|--------|-------|--------|
| DL/I sequence | yes       | yes     | no     | yes   | D      |
| DL/I search   | yes       | yes     | yes    | yes   | D      |
| DL/I SIX      | yes       | yes     | no     | no    | D      |
| UDF           | no        | yes     | yes    | yes   | blank  |

**Ancestor Segment:**

| Type of field | FIND/READ | DISPLAY | UPDATE | STORE | Marked |
|---------------|-----------|---------|--------|-------|--------|
| DL/I sequence | yes       | yes     | no     | yes   | D      |
| DL/I search   | yes       | no      | no     | no    | S      |
| DL/I SIX      | yes       | yes     | no     | no    | D      |
| UDF           | no        | no      | no     | no    | blank  |

**Note:**

The DL/I SIX fields can be DISPLAYed only if a PCB is used with this SIX specified in the PROCSEQ parameter. If not, an error message is returned by Natural at runtime.

The DL/I SIX field name cannot be used in an UPDATE or STORE statement. SIX fields, however, can be updated/stored by referring to the source fields which comprise the SIX.

The READ statement returns records in ascending sequence. The possible sequences for DL/I segments are root sequence or the sequence of any secondary index.

As mentioned above, the generated DDM contains all fields of the current segment and all DL/I fields of the ancestor segment(s), marked either as "D" or "S". The UDFs of the ancestor segments are not included in the generated DDM because a DDM refers only to one segment.

The generated external name of the DDM is equal to the segment name prefixed by the DBD name.

**Example:**

|                        |                 |
|------------------------|-----------------|
| Name of DBD:           | ED00DBD         |
| Name of segment:       | STUDENT         |
| Name of generated DDM: | ED00DBD-STUDENT |

The generated external name of DL/I fields is equal to the name specified in the DL/I FIELD macro during the DL/I DBDGEN procedure.

The generated external name of DL/I fields of ancestor segments is equal to the field name suffixed by the segment name.

**Example:**

|                           |                   |
|---------------------------|-------------------|
| Name of DL/I field:       | LOCATION          |
| Name of ancestor segment: | OFFERING          |
| Name of generated field:  | LOCATION-OFFERING |

The generated external name of the UDFs is equal to the name specified by the user at definition time.

# NSB Maintenance

When you select NSB Maintenance on the DL/I Services Main Menu, the NSB Maintenance menu is displayed.

From this menu, you can select the following NSB maintenance functions:

| Function                        | Explanation   |
|---------------------------------|---|
| Select an NSB from a list       | List the DL/I PSBs defined on the Natural system file. You can select NSBs from this list by entering the function code<br><br>P to purge an NSB, or<br><br>L to list all PCBs and SENSEGs of an NSB.         |
| Purge an NSB                    | Delete an NSB and its related PCB descriptions from the Natural system file. The name of the NSB to be deleted must be specified. Before this function is executed, you are prompted to confirm the deletion. |
| List PCBs and SENSEGs of an NSB | For any NSB specified, this function lists the PCBs and their sensitive segments. If an indexed database exists, its name is displayed under the header "PROCSEQ".  |

## Select an NSB from a list:

```

10:44:50                **** DL/I SERVICES ****                97-08-20
                        - NSB List -

      Func   NSB Name  CMPAT  Length  NoPCBs  Response
-----
      _      DFSIVP6   YES    140     3
      _      PBNDL01   NO     160     3
      _      PBNDL02   YES    160     1
      _      PBNDL03   YES    160     3
      _      PBNDL04   YES    160     1
      _      PBNDL05   NO     80      1
      _      PBNDL97   YES    160     3
      _      PBNDL98   YES    200     5
      _      PBNDL99   NO     200     5
      _      PBPQA01   YES     60     5
      _      PBSUP06   NO    440     5
-----
                        - More -
Code .. _ ( ? Help, . Back, M End )

Func .. P (Purge NSB) L (List PCBs and SENSEGs)

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
Exec  Help           Exit                               Canc
    
```

**List PCBs and SENSECs of an NSB:**

```

10:46:57                **** DL/I SERVICES ****                97-08-20
                        - PCB List -
NSB Name: PBNDL01 (CMPAT=NO ,Length=00160)

Number of PCB's  NDB Name      Level  SENSEG      PROCSEQ
-----
          3      ED00DBD
                        1  COURSE
                        2  PREREQ
                        2  OFFERING
                        3  TEACHER
                        3  STUDENT

----- Bottom -----

Code .. _ ( ? Help . Back M End )

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
Exec  Help           Exit                               Canc
    
```

```

10:49:10                **** DL/I SERVICES ****                97-08-20
                        - NDB List -

Func      NDB Name  L/P  Length  NoSGMs  Access  Response
-----
          CCCBTD00  P    460     6
          DNDL01   P    540     5
          DNDL02   P    620    10
          DNDL03   L    820    10
          DNDL04   P     60     1  GSAM
          DPQA04   P    480     5
          DSUP02   L   1720    15
          DSUP05   P    380     5
          DSUP09   P    340     2
          DSUP10   L    880    10
          DUSA01   P    320     5  HDAM

----- More -----

Code .. _ ( ? Help, . Back, M End )

Func: P (Purge NDB) L (List NDB Segments)

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
Exec  Help           Exit                               Canc
    
```