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 or 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 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 *psbname*	Issues a PSB schedule of the PSB *psbname*.
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 (DFSRRC00 for z/OS, DLZRRC00 for z/VSE).

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 z/OS

Under z/OS, the DL/I region controller program (DFSRRC00) 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 - z/OS 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 - z/OS 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 z/VSE

Under z/VSE, the DL/I region controller program (DLZRRC00) 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 DLZRRC00,
  - `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 - z/VSE with Adabas System File:

```
 // EXEC DLZRRC00
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 - z/VSE with VSAM System File:

```
// EXEC DLZRRC00
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('NAT42n')
END-EXEC.
```

This example taken from a COBOL/CICS program assumes that NAT42*n* 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 NDLPCBAD and NDLPSBSC are provided, which can be invoked with a CALLNAT statement from within a Natural program.

See the following sections:

- The NDLPCBAD Subprogram
- The NDLPSBSC Subprogram

The NDLPCBAD Subprogram

The Natural subprogram NDLPCBAD 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 'NDLPCBAD' 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 NDLPSBSC Subprogram

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

Using CALLNAT 'NDLPSBSC' (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 'NDLPSBSC' COMMAND PSBNAME RETCODE
DISPLAY PSBNAME RETCODE
END
```

Under IMS TM, NDLPSBSC 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: CKPTID=LAST — to restore data areas written to the log by the job at the last successful checkpoint; or CKPTID=*cccccccc* — to restore data areas written with checkpoint ID *cccccccc*.

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 GROUPA". 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/1.

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.