



NATURAL

Natural

Operations

Version 5.1.1 for UNIX and OpenVMS



This document applies to Natural Version 5.1.1 for UNIX/OpenVMS 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.

© March 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

Operations for OpenVMS and UNIX - Overview	1
Operations for OpenVMS and UNIX - Overview	1
Operations Environment	2
Operations Environment	2
System-File Simulation	3
System Files FNAT and FUSER	3
Natural Root Directory	5
Configuration Files	6
Local Configuration File (NATURAL.INI)	6
Global Configuration File (NATCONF.CFG)	7
DBMS Assignments	7
System-File Assignment	9
Dictionary-Server Assignment	10
Printer Profiles	11
Natural Version-Independent Operating System Files (NATOSDEP)	12
Work-Files	13
Defining Work-Files	13
Accessing Work-Files	14
Work-File Types	15
Record File Format for OpenVMS	16
Special Considerations for Work-Files with Extension NCD	16
Natural in Batch Mode	18
General Information	18
Batch-Mode Simulation	18
Real Batch Mode	20
Sample Session for Batch Mode	22
Natural Exit Codes	25
Special Considerations for Natural on OpenVMS	25
Natural Startup Errors	27
Invoking Natural Subprograms from 3GL Programs	29
Passing Parameters from the 3GL Program to the Subprogram	29
Natural TX Interface (for UNIX only)	30
Overview	30
Operational Requirements	30
Natural with TOP END	31
Restrictions	31
Using Versioning Software	31
Issuing OpenVMS System Commands from a Natural Program	32
Issuing UNIX System Commands from a Natural Program	34
Tuning SQL Database Access	36
SQLRELCMD	36
SQLMAXSTMT	36
Example	36
User Exit for Computation of Sort Keys - NATUSK nn	36
Profile Parameters	38
Profile Parameters	38
General Information	39
Creating Parameter Files	40
Modifying NATPARM Parameters	40
NATPARM Utility	40
File	41
Search	41
Changing Parameter Settings	41

Performing NATPARAM Functions from the Operating System Prompt	42
Profile Parameters - Overview	42
Adabas Parameters	42
Batch Mode	43
Buffer Sizes	43
Character Assignments	44
Compiler Options	44
Database Management System	44
DCOM Support	44
Devices	45
Entire Transaction Propagator	45
Environment Assignments	45
Error Handling	46
Function Keys	46
Limits	46
Miscellaneous	46
Program Loading and Deletion	47
Remote Debugging	48
Remote Dictionary Access	48
Remote Procedure Call	48
Report Parameters	49
System Files	50
System Variables	50
Work Files	50
Year 2000 Support	50
Dynamic Assignment of Parameter Values	51
Runtime Assignment of Parameter Values	52
Natural Buffer Pool	53
Natural Buffer Pool	53
General Information	54
Introduction	54
Setting up a Buffer Pool	56
NATBPSRV Utility to Start a Buffer Pool	59
NATBPMON Utility to Monitor or Shut Down a Buffer Pool	60
Invoking NATBPMON	60
NATBPMON Commands	60
Buffer-Pool Trouble Shooting	68
Problem 1 (under OpenVMS)	68
Problem 1 (under UNIX)	68
Problem 2 (under UNIX only)	69
Shutting Down and Restarting the Buffer Pool	71
Dispatch Vector Addresses (OpenVMS/VAX only)	73
Support of Different Character Sets	74
Support of Different Character Sets	74
Character Sets that are Supported	75
Internal Character Set	75
External Character Sets	75
Character-Set Conversion	76
Configuration File NATCONV.INI	77
NATTERMCAPI Utility	84
NATTERMCAPI Utility	84
General Settings	85
Environment Variables	85
Terminal Mode	85
Key Definitions	85
Dynamic Parameters	86

Special Control Codes	87
Terminal Copy Capabilities	88
Invoking NATTERMCAP	90
File	91
Edit	92
Search	92
Test	93
Options	94
Help	95
Terminal Capabilities - Overview	96
Colors	96
Cursor Keys and Modes	96
Description and Comments	97
Editing Keys	97
Initialization and Reset	98
Keypad Keys for Mathematical Operations	99
Line Graphics	100
Miscellaneous	101
Name and Aliases	101
PA and PF Keys	101
Right-To-Left Support	101
Screen Dimension and Appearance	101
Video Attributes	102
Terminal Capabilities - Sorted by Name	103

Operations for OpenVMS and UNIX - Overview

This documentation provides information on the operation of Natural in a UNIX or OpenVMS environment.

The following topics are covered:

- Operations Environment provides information on Natural execution under UNIX and OpenVMS.
- Profile Parameters describes the Natural profile parameters you can specify in your parameter files.
- Natural Buffer Pool describes the usage of the Natural buffer pool.
- Support of Different Character Sets describes the use of different character sets with Natural.
- NATTERMCAPI Utility describes how to adapt your terminal to terminal-dependent parts of Natural.

See also:

NaturalX for UNIX

(The NaturalX documentation contains information required to administer and use NaturalX on all available platforms.)

Operations Environment

- System-File Simulation
 - System Files FNAT and FUSER
 - Natural Root Directory
- Configuration Files
 - Local Configuration File (NATURAL.INI)
 - Global Configuration File (NATCONF.CFG)
- Work-Files
 - Defining Work-Files
 - Accessing Work-Files
 - Work-File Types
 - Record-File Format for OpenVMS
 - Special Considerations for Work-Files with Extension NCD
- Natural in Batch Mode
 - General Information
 - Batch-Mode Simulation
 - Real Batch Mode
 - Input and Output Channels
 - Sample Session for Batch Mode
 - Hints for Migration from Natural for OpenVMS Version 2.1 to Version 5.1
- Natural Exit Codes (for OpenVMS only)
- Startup Errors
- Invoking Natural Subprograms from 3GL Programs
 - Passing Parameters from the 3GL Program to the Subprogram
- Natural TX Interface (for UNIX only)
 - Overview
 - Operational Requirements
 - Natural with TOP END
 - Restrictions
- Using Versioning Software
- Issuing OpenVMS System Commands from a Natural Program
- Issuing UNIX System Commands from a Natural Program
- Tuning SQL Database Access
 - SQLRELCMD
 - SQLMAXSTMT
 - Example
- User Exit for Computation of Sort Keys - NATUSKnn

System-File Simulation

To ensure database independence, Natural does not store objects in a database system file. The system file is simulated by a structure of directories on the disk where Natural is installed.

Since the operating system is used instead of a database system, you can support database systems other than Adabas. Therefore, any text editor can be used to write source code files which are stored as ASCII text files on the disk.

Note for UNIX:

Semaphores are used to synchronize access to the Natural system files. Since this requires additional operating-system resources, you should consider incrementing the kernel parameters SEMMNI and SEMMNS by the number of system files to be emulated (refer to Change Kernel Parameters appropriate to your environment) in Activating the Natural Buffer Pool on UNIX in the Natural Installation Guide.

The Natural libraries are created as subdirectories below your current system file root directory. The command LOGON is therefore similar to the Change Directory command.

Under OpenVMS and UNIX, Natural requires ".NS n " as source filename extension, where n stands for the type of source. ".NSP", for example, stands for Natural source programs, and ".NSM" for Natural source maps.

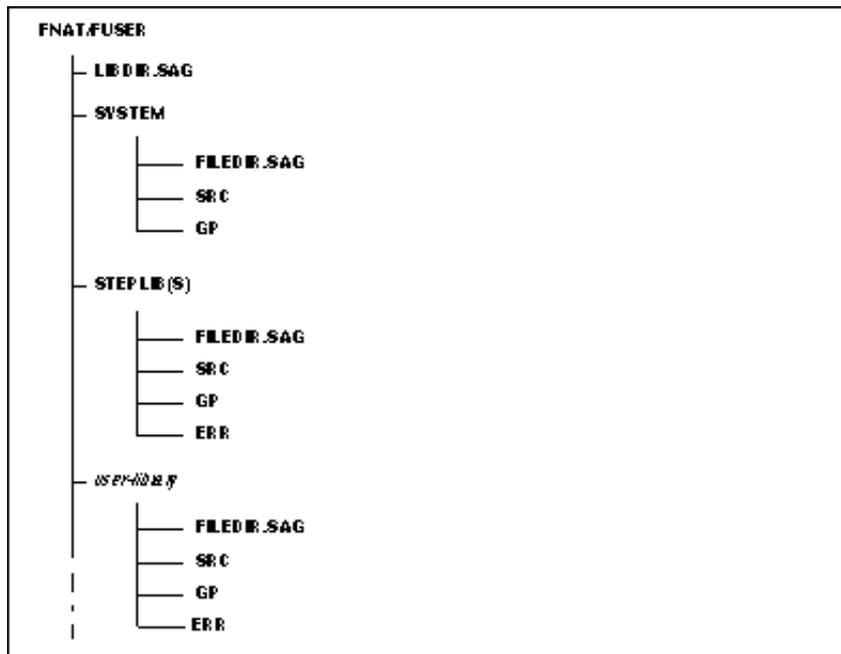
The available types of sources and the corresponding letters to be used for the extension are as follows:

P	Program
N	Subprogram
S	Subroutine
C	Copycode
H	Helproutine
T	Text
M	Map
L	Local data area
G	Global data area
A	Parameter data area
4	Class

Under OpenVMS and UNIX, a Natural library is modelled as a subdirectory of a system file root directory (for example, "/NATURAL") with the same name as the library. This subdirectory contains an additional subdirectory named "SRC" in which all sources and data areas are stored.

System Files FNAT and FUSER

The Natural system files FNAT (for system programs) and FUSER (for user-written programs) can be located in different subdirectories. They assume the following directory structure:



This directory structure is generated during the installation of Natural.

The file LIBDIR.SAG contains information on all further installed Software AG products using Natural. This information can be displayed by using the SYSPROD system command.

The directory representing the system libraries SYSTEM, STEPLIB, and each *user-library* contain the following:

- A file FILEDIR.SAG containing library information used by Natural including in particular the programming mode of an object (structured or reporting) and internally converted object names. These internal object names are automatically created when storing Natural objects to disk with names longer than 8 characters (which can be the case with DDMs), or names containing any special character supported by Natural but not by the operating system.
Internal object names are unique and consist of an abbreviation of the actual object name and an arbitrary number.
Both the actual object name and the corresponding internal object name are documented in FILEDIR.SAG. Even if an object is located in the correct directory, it can only be used by Natural after this library information is included in FILEDIR.SAG. For objects created within Natural, the library information is included automatically. For all other objects, the Import function of the SYSMAN utility should be used. Also externally Natural generates a unique file name when a name is to be used that already exists in the system file, but not in the FILEDIR.SAG. That could be the case if a file was copied using operating system utilities instead of using Natural or its utilities. Natural will not overwrite the already existing file. The utility FTOUCH can be used to update FILEDIR.SAG without entering Natural.
- A subdirectory SRC containing the source objects stored in the library.
- A subdirectory GP containing the generated programs stored in the library.
- A subdirectory ERR containing the error messages stored in the library (this subdirectory is optional).

Note: Do not access Natural files with operating system utilities! These utilities might modify and destroy the Natural directory information!

Natural Root Directory

During the installation, the logical names NATDIR and NATVERS (for OpenVMS) or the environment variables \$NATDIR and \$NATVERS (for UNIX) are created automatically. They point to the Natural version-dependant root directory.

For UNIX:

```
$NATDIR/$NATVERS
```

or

For Open VMS:

```
NATDIR: [ ' F$TRNLNM( "NATVERS" ) ' ]
```

Configuration Files

The base directory for FNAT/FUSER is found by looking in the following configuration files:

- NATURAL.INI
- NATCONF.CFG

Every user should have read-access to these files; however, only system administrators should be allowed to modify them (write-access is granted by using NATURAL.INI).

Notes:

If you use multiple Natural versions, the NATURAL.INI file is located in the appropriate "etc" directory for each installed Natural version. It is strongly recommended that only one NATCONF.CFG exists on a single computer or cluster. Maintaining only one file ensures a consistently defined database and/or system file for all users.

The configuration files contain plain ASCII text and should only be changed with the NATPARM utility.

To invoke the NATPARM utility, enter "NAT51PARM" at the OpenVMS DCL prompt or at the UNIX system prompt. The Natural Parameter Setting menu is displayed. On this menu, select "Configuration". A window appears with the two options:

- Local Configuration File
With this option, you can modify the configuration file NATURAL.INI.
- Global Configuration File
With this option, you can modify the configuration file NATCONF.CFG.

Note: If the "Configuration" option is not displayed on your Natural Parameter Setting menu, this means that you have no permission to modify the configuration files.

The contents of the two files are described on the following pages.

Local Configuration File (NATURAL.INI)

The local configuration file NATURAL.INI contains the following assignments:

Buffer-Pool Assignments

Various specifications related to the Natural Buffer Pool (BPID, Maxusers, MEMSIZE, Shared Memory Key, Semaphore Key and others).

Installation Assignments

- **Name of Global Configuration File (CONF_NAME)**
The name and path of the global configuration file (default name is NATCONF.CFG).
- **Path to Error File (ERROR_FILES)**
The location of the Natural error files.
- **Name of Natural I/O Conversion Table (NATCONV)**
The name of the file which contains the character translation tables used with the internal character set "ISO-8859-1". By default, this file is called NATCONV.INI; see Support of Different Character Sets for details.
- **Path to Binary Libraries of Software AG Products (NATEXTLIB)**
The location of binary libraries of further Software AG products using Natural.
- **Name of Natural Terminal Database (NATTCAP)**
The name of the database which contains the descriptions of the terminal capabilities for each terminal type supported by Natural. Refer to the NATTERM CAP utility for details.
- **Path to Profile Parameters (PARM_PATH)**

The location of the Natural parameter files.

- **Path to Natural Profiles (PROFILE_PATH)**

The location of DDM editor (*.D00) profiles.

- **Path to TMP Directory (TMP_PATH)**

The location of Natural temporary output. In order to support separate user-specific TMP directories, the directory name may include embedded environment variables. These will be evaluated and substituted at run time. For example, C:\temp\%usertmp%\ %usertmp% will be replaced at Natural start-up with the contents of this environment variable.

- **Path to Text Files (TXT_PATH)**

The location of messages, choices and help texts used by the editors and by the NATTERMCA utility.

Administrator Assignments

With the Administrator Assignments function you can specify whether a user is to be a Natural Administrator or a Configuration Administrator.

A user who is defined as Configuration Administrator can modify the configuration files NATURAL.INI and NATCONF.CFG. Only Natural Administrators can be defined as Configuration Administrators.

Global Configuration File (NATCONF.CFG)

The global configuration file NATCONF.CFG contains the following assignments:

- DBMS Assignments
- System-File Assignment
- Dictionary-Server Assignment
- Printer Profiles
- Natural Version-Independent Operating System Files (NATOSDEP)

DBMS Assignments

Parameter	Function
DBID and DBMS Type	Specifies a value for each database ID.
DBMS Parameter	Establishes a connection to the database system. (Applies to SQL-type databases only).
SQL Date/Time Conversion	Specifies the date/time format for SQL databases.
Multifetch Disabling	Disables multifetch on an Adabas file and command-level basis.

If you select "DBMS Assignments", a dialog is displayed in which you can specify the DBID, DBMS Type and DBMS Parameter.

DBID and DBMS Type

Since the types of all databases which are to be accessed by Natural must be defined in the global configuration file, specify one of the following values for each database ID:

Value	Database Type
ADA	Adabas database server (this is the default).
ADX	Adabas single-user database (for PCs only).
OSQ	Any SQL database that can be accessed using Entire Access, Software AG's common interface to various SQL database systems. Note that in the user interface of the Natural Configuration utility the term OSQ is replaced with the term SQL .

Note:

You should define a database type for a DBID which has already been assigned to a system file; if you try to, an error message will be issued at Natural startup, indicating an inconsistency in the system file setting and an error when reading the database assignments.

A list of existing DBMS assignments is displayed in a list box.

DBMS Parameter

This field applies to SQL-type databases only.

In this field, you establish the connection to the database system that you want to work with. In addition, you can specify logon parameters specific to the database type.

See your Natural and Entire Access documentation for further information on how to access SQL-type database systems.

SQL Date/Time Conversion

As Natural has only one specific time format, the user has to decide how this format should be interpreted in the context of SQL database access.

If you select "SQL Date/Time Conversion", a dialog is displayed in which you can specify the conversion masks.

Mask

The value specifies the configuration for Entire Access. It also specifies the format used to retrieve SQL DATE/TIME/DATETIME information into Natural format A fields. The mask should match the RDBMS-specific configuration for the DATE, TIME, or DATETIME character string representation.

Date

This mask (usually a sub-string of the Mask value) specifies the character-string representation into which the Natural format D fields are converted during update or retrieval of SQL DATE columns.

Time

This mask (usually a sub-string of the Mask value) specifies the character-string representation into which the Natural format T fields are converted during update or retrieval of SQL TIME or DATETIME columns.

Remark

You can enter your remarks here, for example, to document how the SQL DATE/TIME character-string representation is configured on the database site.

See also the Entire Access documentation, Using Natural with Entire Access, Date/Time Conversion.

Multifetch Disabling

Natural uses Adabas command-level multifetch for Adabas L1, L2, L3 and L9 calls to minimize the number of database calls. For performance reasons it may be useful to disable multifetch on an Adabas file and command-level basis. This can be done with the Multifetch option. Before a multifetch call is issued, Natural checks if multifetch is disabled for the actual dbid/file/command code combination.

System-File Assignment

For each Natural system file, a database ID (DBID) and a file number (FNR) are specified in the Natural parameter file NATPARM. To ascertain the location of each system file in the structure of directories, you use the System File Assignment function to assign a path name to the DBID/FNR combination. If you assign a path name that has already been assigned to another system file, all relevant internal information of the first assignment are copied automatically to the new assignment.

The path name must be a valid OpenVMS or UNIX directory path, indicating the physical path to the location of the system files on the disk.

Example:

System-file specifications in NATPARM:

FNAT:	DBID=098 , FNR=099
FUSER:	DBID=042 , FNR=043

Path assignments in NATCONF.CFG (example below is valid for UNIX; for OpenVMS users, the path names are "Path: SAG\$ROOT:[path1]" and "[path2]" respectively):

DBID:	098
FNR:	099
Path:	/usr/SAG/path1

DBID:	042
FNR:	043
Path:	/usr/SAG/path2

The above assignments represent the following directory structure:

/usr/SAG/path1	(for FNAT)
- SYSLIB	
.....	

/usr/SAG/path2	(for FUSER)
- SYSTEM	
.....	

Dictionary-Server Assignment

With the Dictionary-Server Assignment function, you can assign three so-called dictionary servers to one common logical server name, or you can assign them to separate logical server names.

With dictionary servers, you can access remote DDMs, free rules and automatic rules maintained in Predict once you have access to Predict on a mainframe or UNIX host. The three dictionary servers are the following:

The DDM Server	Server for remote DDM access.
The Free Rule Server	Server for remote access to Predict free rules.
The Automatic Rule Server	Server for remote access to Predict automatic rules.

Since the servers to be assigned can be located on different nodes, both server and node name must be specified.

All dictionary servers must first be defined by using the Natural RPC facility. The RPC parameter MAX-BUFF must be set to at least 8192 bytes.

Note: Currently, a server's node name must start with "FBKR" if you want it to be addressed by using a central mainframe broker.

Multiple logical server names can be defined; server assignments can be modified or deleted. A list of existing assignments can be displayed by pressing F4.

Specify the dictionary servers you want to use by setting the remote access parameter USEDIC to the corresponding logical server name; if USEDIC is set to blank, remote DDM access will not be possible.

Note: If you want to use this feature, Predict Version 3.3 or above must be installed; with Predict versions prior to 3.3, the Predict update tape PD2302 is required.

Printer Profiles

This function is used to define, modify or delete printer profiles.

Printer profiles are used for printing additional reports, for hardcopies and for batch output generation. They recognize particular Natural field attributes and insert the appropriate control sequences (see below) as defined in the profile.

With the ability to translate Natural field attributes into escape sequences, you can control your printer in various ways by using a specific profile name, and you can use the print features of a given device by using simple attributes in Natural programs.

To define a printer profile, you have to specify a profile name and leading and/or trailing commands (that is, printer control sequences) to be triggered at job, page or field level.

Each profile can be assigned to a Natural report number either statically by using the NATPARM Device/Report Assignments or dynamically by using the DEFINE PRINTER statement within a Natural program.

Triggering Events

A triggering event controls the level on which specified printer control sequences are to be applied. Available triggering events are: JOB, PAGE, AD and CD (a list of options can be obtained by pressing PF2).

- Specify JOB if you want your control sequences to apply to an entire print job; the specified control sequences will represent the job header and/or job trailer respectively.
- Specify PAGE if you want the control sequences to apply to each physical output page; the specified control sequences will then represent the page headers and/or page trailers respectively.
- Specify one of the Natural session parameters AD or CD along with appropriate field attributes if you want the control sequences to be applied at field level only; any field in a Natural program with corresponding attributes will then cause these control sequences to take effect. See the Natural Reference documentation for details on these session parameters.

Control Sequences

The leading control sequence is inserted immediately before the triggering event (for example, to define a job header or to set attributes for field representation). The trailing control sequence is inserted immediately after the triggering event (for example, to define a job trailer or to reset attributes previously set).

For each control sequence, a window appears, in which you can specify the control characters in either alphanumeric or hexadecimal format.

Example:

Alphanumeric format : **^1b(s1P**

Hexadecimal format : **^1b^28^73^31^50**

Note: The escape character must always be specified in hexadecimal format.

Natural Version-Independent Operating System Files (NATOSDEP)

This function is used to specify a directory common to all installed Natural versions, which is to contain operating system-specific Work-Files and temporary files used by all these Natural versions. These files are required for synchronization purposes and must not be deleted or modified by a Natural Administrator.

Note: If you use the system directory for temporary files, which is "/tmp" for UNIX and "SYS\$SCRATCH" for OpenVMS, ensure that no automatic procedures regularly delete the contents.

Work-Files

Work-Files are files where data can be written to and read from by Natural programs. They are used for intermediate storage of data and for data exchange between programs. Data can be transferred from or to a Work-File by using the Natural statements READ Work-File and WRITE Work-File, or UPLOAD and DOWNLOAD.

Defining Work-Files

To define a work-file, select Edit > Work-Files in NATPARM. In the Work-File Specification column, you can enter the name and extension for each of the up to 32 work-files.

If the specified work-file was previously defined, the current work-file definition is displayed, which can be overwritten. Enter the complete definition; that is, name *and* extension of your work-file. Press ENTER to save your settings, or press ESC to exit.

If the work-file was never defined, the Workfile Specification field is blank. Enter the work-file name including its path name.

For any numbered work-file, a complete work-file specification can be entered. This specification must contain the path and the work-file name.

If no work-file definition is specified, Natural automatically creates the file name and writes the work-file into the "tmp" directory specified in the local configuration file. The number consists of the specified work-file number and an arbitrary number assigned by the operating system.

Note for OpenVMS, AXP and VAX only:

The work-file name consists of the prefix "W", a number of up to seven digits and the extension "SAG".

Work-Files with Environment Variables

Work files can also be defined by using OpenVMS logical names or UNIX environment variables in the same way they are used with OpenVMS DCL commands or UNIX shell commands respectively.

Example:

Specify the following name for Work-File Number 4 in your Natural parameter module:

```
NATWORK04
```

Under OpenVMS, you use a logical name to define this work-file:

```
$ DEFINE NATWORK04 mydevice:[mydir]network04.dat
```

Under UNIX, you use an environment variable to define this work-file:

```
setenv NATWORK04 mydevice/mydir/network04.dat
```

Work-Files with Environment Variables (for UNIX Users)

Work files can also be defined by using them in the same way as work-file names that can be set without changing the NATPARM parameter file.

Example:

Specify the following name for a work-file:

```
$NATURAL/$myfile
```

and assume the following settings:

```
set NATURAL=/usr/natural
set myfile=sub/test
```

which will expand in the file name during Natural startup:

```
usr/natural/sub/test
```

Work-Files with User Exit

Work files can also be defined by using the user exit USR1050N in the library SYSEXT.

Accessing Work-Files

Work-files can be accessed in two different ways: either locally or by using a data transfer with Entire Connection. Depending on the access method, a work-file must have a certain format.

Work-File Formats for Local Access

Three different work-file formats are available for local access. Natural recognizes the format by checking the work-file specification. This consists of a file name and an extension separated by a period:

file-name.extension

The three work-file formats are:

- a binary format specific to Software AG (to be preferred),
- ASCII format,
- a format which corresponds to the one used by Entire Connection.

For the work-file number to be used, the profile parameter "ENTIRE CONNECTION" in the NATPARM parameter file must be set to "N". A work-file specification is required for the appropriate format as described in the following sections.

Binary Format

The binary format specific to Software AG is defined by using any file name and either a period and the extension "SAG" or no period and extension at all.

Note: A two (2) byte length precedes each record written.

Examples:

xxxxxxx.SAG (any file name with a period and an "SAG" extension)

xxxxxxx (any file name without a period or extension)

The binary format can be used with all data types.

ASCII Format

To define an ASCII work-file format, enter a file name, a period and either any extension except "SAG" and "NCD" or no extension at all.

Examples:

xxxxxxxx.xxx (any file name with a period and any extension except "SAG" or "NCD")

xxxxxxxx . (any file name and a period)

Since each record written is terminated with the LF character sequence, this format is recommended for alphanumeric data only.

Entire Connection Format

Entire Connection uses two files: a data file, which contains the actual data, and a format file, which contains format information about the data in the data file.

The data file is defined by using any file name, a period and the extension "NCD"; the corresponding format file is automatically generated; it has the same name as the data file, but the extension ".NCF". Both extensions must be in upper case characters.

Examples:

xxxxxxxx.NCD (any file name with a period and an "NCD" extension for the data file)

xxxxxxxx.NCF (any file name with a period and an "NCF" extension for the format file)

With local access (that is, without any data transfer being involved), you can read/write work-files in Entire Connection format directly from/to your local disk. However, work-files in Entire Connection format can also be accessed by using a data transfer (see below). Both access methods can be used simultaneously, but with different work-file numbers only.

Note: With Entire Connection, the RECORD option of the READ WORK FILE statement makes no sense and can therefore not be used.

Format for Work-Files to be Accessed by using a Data Transfer

Work files to be accessed by using a data transfer must be in Entire Connection format (see above).

With data transfer, the Natural statements READ WORK FILE and WRITE WORK FILE do not read from and/or write to your local disk, but transfer the data to a PC that runs Entire Connection. The read/write operations are then done by Entire Connection from/to the disk of the PC.

For the work-file number to be used, you have to set the profile parameter ECPMOD to "Y" in NATPARM. No work-file specification is required in this case, because Entire Connection prompts you to enter a file name.

Work-File Types

Work-File Type PORTABLE

The work-file type PORTABLE performs an automatic endian conversion of a work-file when it is transferred to a different machine. For example, a work-file written on a PC (little endian) can be read correctly on an RS6000 or HP machine (big endian). The endian conversion applies only to field formats I2, I4, F4 and F8. The floating point format is assumed to be IEEE. There are, however, slight differences in IEEE floating point representation

by different hardware systems. As far as we know, these differences apply only to infinity and NaN representations, which are normally not written into work-files. Check the hardware descriptions if you are uncertain.

The files are always written in the computer-specific representation, so that a conversion is performed only if the file is read by a computer with different representation.

This keeps performance as fast as possible. There are no other conversions for the work-file type PORTABLE, but the file format makes it possible to add them in future releases.

When a READ WORK is done for a dynamic variable, it is resized to the length of the current record.

Work-File Type UNFORMATTED

The work-file type UNFORMATTED reads or writes a complete file with just one dynamic variable and just one record, e.g., to store a video which was read from a database. No formatting information is inserted, everything is written/read just as it is.

Record File Format for OpenVMS

Work files (1 to 32) with binary format are created with RMS file format variable length, whether all other files are created with RMS file format stream line feed and record attribute carriage return carriage control.

Special Considerations for Work-Files with Extension NCD

The following considerations apply for work-files in Entire Connection format:

- If an ".NCD" file is read with a READ WORK FILE statement and the corresponding ".NCF" format file is not available or contains invalid information, the ".NCD" file is assumed to be an ASCII work-file.
- The maximum work-file record size and format file size that can be handled by Entire Connection is 1900 bytes.
- When an array is written to or read from an ".NCD" work-file, a maximum of 255 array elements can be written/read at a time.
- The VARIABLE option of the WRITE WORK FILE statement cannot be used for ".NCD" work-files.
- If you have "old" work-files whose names have ".NCD" extensions, and are not Entire Connection files, the extensions must be changed.
- Each of the following profile parameters must be set to the same value for both read and write operations:
 - DC (decimal character),
 - IA (input assign character),
 - ID (input delimiter character).
- Since a data transfer involves screen I/Os to send and receive the data, you should use a CLOSE WORK FILE statement after you finish writing the work-file and make sure that no Natural screen I/Os occur during the transfer.

After a read operation, the transfer is automatically closed when the end of the file has been detected; this enables screen I/Os, however, the work-file is still open in Natural and has to be closed by either using a CLOSE WORK FILE statement or returning to the command level.

Note: A work-file must also be explicitly closed in ON ERROR routines if these contain any screen I/Os.

- When transferring data by using Natural work-files to other platforms, remember that the range of possible values for floating point variables on a mainframe computer is different from that on other platforms.
- The possible value range for F4 and F8 variables on a mainframe is:

$\pm 5.4 * 10^{-79}$ to $\pm 7.2 * 10^{75}$

- The possible value range on most other platforms is, for F4 variables:

$\pm 1.17 * 10^{-38}$ to $\pm 3.40 * 10^{38}$

- for F8 variables:

$\pm 2.22 * 10^{-308}$ to $\pm 1.79 * 10^{308}$

- Data transfer is possible only for one work-file number at a time.
- A Natural error message is returned if DBMS calls are issued during an Entire Connection data transfer and their number exceeds the limit for DBMS calls permitted between screen I/Os (specified with the MADIO profile parameter).
- To circumvent this error, the user exit USR1068N in library SYSEXT is provided. USR1068 resets the database call counter to "0". It must be invoked each time a DBMS call is issued during data transfer.
- If files with the extension ".NCD" created by Entire Connection are read into Natural with the READ WORK FILE statement, you must ensure that Entire Connection does not remove trailing blanks from the records.
- To provide for that, Entire Connection must be set up as follows:
From the Host menu, select Session setup. Select the session, then click the Modify button. Select the tab File Transfer 1. Ensure that the Keep trailing blanks checkbox is checked.

Natural in Batch Mode

This section contains special considerations that apply when running Natural in batch mode.

General Information

Natural can be used in two processing modes:

- Interactive Mode
- Batch Mode

In interactive mode, the input of commands and data comes from a terminal keyboard and the output is displayed on a terminal screen. In batch mode, input is read from a file and output is written to a file.

When Natural is run as a background batch job, no interaction between Natural and the person who submitted the batch job is necessary. The batch job consists of a set of programs such that each is completed before the next program is started. The programs are executed serially and receive sequential input data.

Batch mode is of particular interest for mass data processing and re-usable execution.

Detecting Batch Mode

The system variable *DEVICE shows whether Natural is running in batch or interactive mode.

in batch mode	*DEVICE is equal to "BATCH"
in interactive mode	*DEVICE is not equal to "BATCH" (in most of the cases *DEVICE is equal to "VIDEO")

Example:

```

IF *DEVICE = "BATCH" THEN
  /* Batch Mode Process */
  INPUT #NAME #AGE
ELSE
  /* Interactive Process */
  INPUT USING MAP ...
END-IF

```

Batch-Mode Simulation

This section describes how to simulate batch mode processing in an OpenVMS environment. With batch-mode simulation, the input and/or output is directed to a file instead to a terminal.

If the input channel is redirected to a file, Natural does not read the input commands and data from the keyboard but from this file. You have to specify the data in exactly the same way as you would do on the terminal. For example, for two input fields you have to fill up the first field with trailing blanks to position to the second field. No keyword delimiter mode is supported. To use keyword delimiter mode, use real batch mode instead.

If the output channel is redirected to a file, Natural writes any output that would appear on the screen to this file. Control sequences are also written to the file, which makes the file unreadable. To get a formatted output, use real batch mode instead.

Use the dynamic parameter BATCH when starting Natural, to set the system variable *DEVICE to the value "BATCH". This value can be checked within a Natural program.

Example for OpenVMS to redirect the input channel:

```
$ DEFINE NATINPUT input-file-name
$ NATURAL BATCH
```

Example for UNIX to redirect the input channel:

```
Natural Batch < input-file-name
```

Natural then receives all input operations from this input file (see below for an example of this input file).

Example for OpenVMS to redirect the input and output channel:

```
$ DEFINE NATINPUT input-file-name
$ DEFINE NATOUTPUT output-file-name
$ NATURAL BATCH
```

Example for UNIX to redirect the input and output channel:

```
natural Batch < input-file-name > output-file-name
```

If you want to keep Natural reports only and hide all other output (write output to the null device), set the MAINPR profile parameter to a valid printer number and assign an executable file to the corresponding logical printer (device) in NATPARM, then specify:

For OpenVMS:

```
$ DEFINE NATINPUT input-file-name
$ DEFINE NATOUTPUT nla0:
$ NATURAL BATCH
```

For UNIX:

```
natural Batch < input-file-name > /dev/null
```

Any Natural reports are written to the executable file, whereas any screen output is suppressed. An input file must be specified even if Natural does not expect any input at all. In this case, also the null device may be used.

Sample Input File for OpenVMS:

```
dlist program * /* select the direct command line and issue LIST command
. /* terminate the LIST command
fin /* terminate NATURAL
```

Sample Input File for UNIX:

```
dlist program **^M /*select the direct command line and issue LIST command
.^M /* terminate the LIST command
fin^M /* terminate NATURAL
```

The character "^M" is required for simulating the ENTER key.

The content of this input file has the same effect as the corresponding terminal input during an online session: after Natural has been started, the direct command line is selected, the LIST command is invoked and terminated, and Natural is terminated, too.

Real Batch Mode

To run Natural in real batch mode you have to specify the dynamic parameter BATCHMODE. In addition, some input and output channels have to be defined (as described below).

Advantages over Batch-Mode Simulation

It is recommended to use real batch mode instead of batch-mode simulation, because real batch mode has the following advantages:

- Easy data input with support of keyword delimiter mode
- Configurable and formatted output processing
- Extended error handling
- Faster startup and shutdown
- Faster program execution

Restrictions

When running Natural in batch mode, some features are not available or are disabled.

- The terminal database SAGtermcap will be not supported. Therefore, the terminal capability TCS which is used for a different character set is not supported. To use a different character set, use environment variable NATTCHARSET instead.
- No colors and video attributes (such as blinking, underlined and reverse video) are written to the batch output file CMPRINT
- Filler characters are not displayed within an INPUT statement
- No interactive input or output is possible

Input and Output Channels

The following input and output channels are of interest in batch mode:

- CMSYNIN
- CMOBJIN
- CMPRINT
- CMPRTnn
- CMWRKnn
- NATLOG

CMSYNIN (Batch Input File for Natural Commands and INPUT Data)

CMSYNIN is used for the input file that contains Natural commands, and (optionally) data to be read by INPUT statements during execution of Natural programs.

CMOBJIN (Batch Input File for Natural INPUT Data)

CMOBJIN is used for data intended to be read by Natural INPUT statements. This type of data can alternatively be placed in the CMSYNIN file immediately following the relevant RUN or EXECUTE command.

CMPRINT (Batch Output File)

CMPRINT is used for the output resulting from DISPLAY, PRINT and WRITE statements in a Natural program.

CMPRTnn (Output File for Additional Reports)

CMPRTnn is used for additional reports referenced by any Natural program executed during the session. "nn" is two digit decimal number in the range 01 to 31 corresponding to the report number used in a DISPLAY, PRINT and WRITE statement.

In order to allow the user to specify variable print-file names, alpha-format system variables and numeric counter markers may be embedded in the filename specification for CMPRTnn. The supported alpha-format system variables are:

In order to allow the user to specify variable print-file names, alpha-format system variables and numeric counter markers may be embedded in the filename specification for CMPRTnn. The supported alpha-format system variables are:

*APPLIC-ID

*APPLIC-NAME

*DEVICE

*ETID

*INIT-USER

*LIBRARY-ID

*NET-USER

*PID

*PROGRAM

*USER

*USER-NAME

If any of these strings (in upper-case only) are encountered within the print file specification, they will be replaced at run time with the contents of the appropriate system variable. Additionally, a counter marker (#) may be used. This will be replaced by a 2-digit counter which will automatically be incremented for each print file.

Example:

The specification CMPRT01=abc_*PID_*ETID_*PROGRAM_#.dat in a Natural session with process-id 1234, ETID XYZ running a program with the name PRINT which produces print file output to file 01 would produce print-files with the following names (assuming the program runs 3 times):

```
abc_1234_XYZ_PRINT_01.dat
abc_1234_XYZ_PRINT_02.dat
abc_1234_XYZ_PRINT_03.dat
```

CMWRKnn (Batch Output File for Natural Work-Files)

CMWRKnn is used for Natural work-files referenced by any Natural program executed during the session. "nn" is a two digit decimal number in the range 01 to 32 corresponding to the number used in a READ WORK FILE or WRITE WORK FILE statement.

NATLOG (Natural Log File)

NATLOG is used to log messages that could not be written to the batch output file CMPRINT. It is recommended to enable NATLOG in batch mode.

Sample Session for Batch Mode

The following example demonstrates how to start Natural in batch mode. A simple Natural program is executed and three data items are taken from the batch input file. After the items are processed from the INPUT statement, a subsequent DISPLAY statement writes the data to the batch output file. Then, Natural terminates.

Natural Commands (CMSYNIN=batch.cmd):

```
LOGON SYSEXBAT
EXECUTE RECCONT
FIN
```

Natural Input Data (CMOBJIN=batch.inp):

```
Ben,%
Smith
```

Natural Program RECCONT in Library SYSEXBAT:

```
DEFINE DATA LOCAL
1 #firstname (A10)
1 #lastname (A10)
END-DEFINE
INPUT (IP=OFF AD=M) #firstname #lastname
DISPLAY #firstname #lastname
END
```

Natural Command Line:

```
NATURAL BATCHMODE CMSYNIN=batch.cmd CMOBJIN=batch.inp CMPRINT=batch.out
BMSIM=MF NATLOG=ALL
```

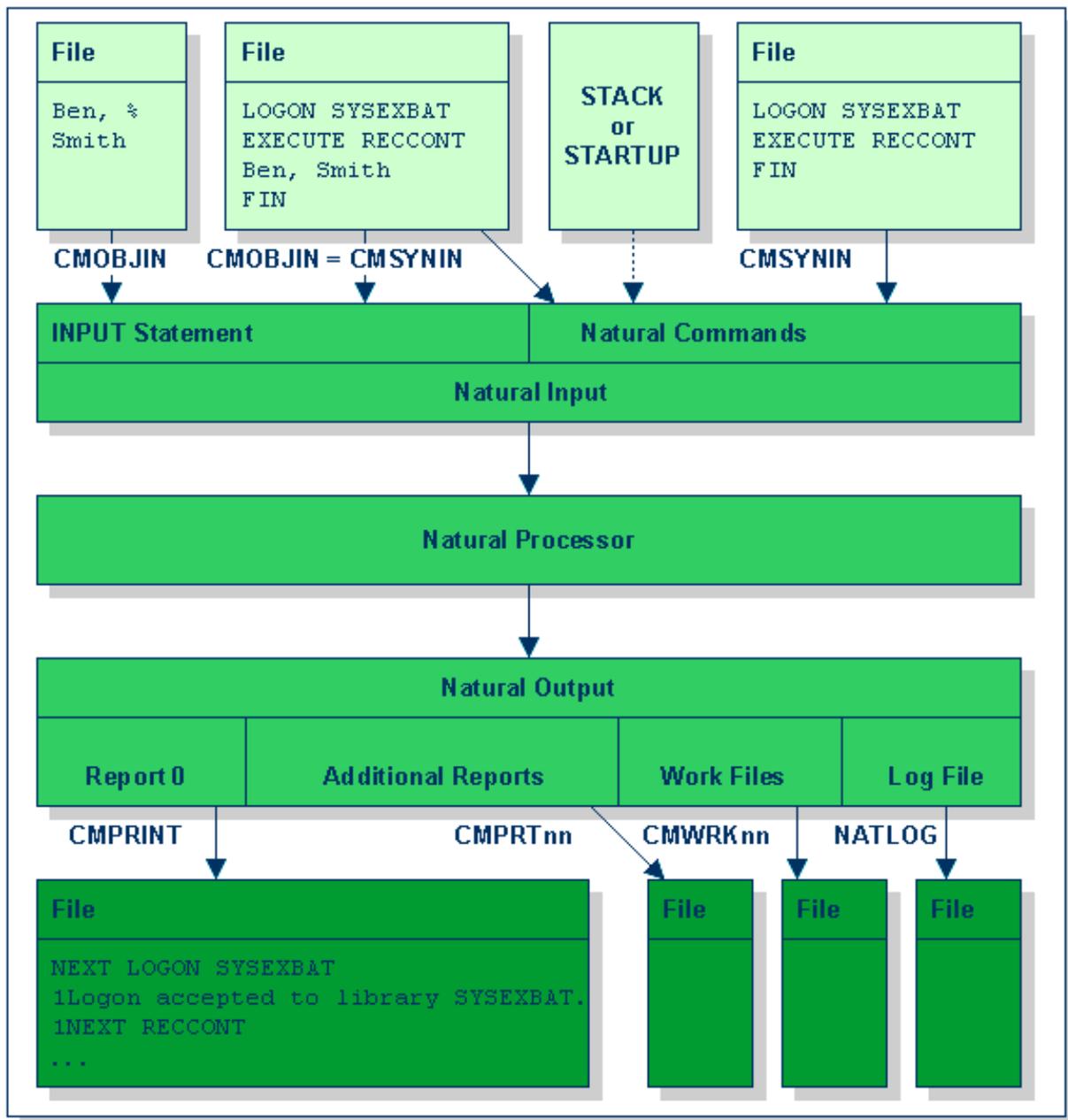
After Natural is invoked by using the command line described above, Natural produces the following output.

Contents of Batch Output File (CMPRINT=batch.out) after Execution:

```

_NEXT LOGON SYSEXBAT
!Logon accepted to library SYSEXBAT.
!NEXT EXECUTE RECCONT
!#firstname          #lastname
_DATA Ben,%
_DATA Smith
!Page      1                      01-10-13  10:55:30
0#FIRSTNAME #LASTNAME
-----
0Ben        Smith
!NEXT FIN
!NAT9995 NATURAL session terminated normally.
    
```

The following graphic illustrates how Natural reads input and writes output in batch mode.



Hints for Migration from Natural for OpenVMS Version 2.1 to Version 5.1

The following table shows the main batch environment differences between Natural for OpenVMS Version 2.1 and Version 5.1.

Action	Natural Version 2.1	Natural Version 5.1
Activate Batch Mode	-	Specify dynamic parameter BATCHMODE
Define channel for Commands	Define logical symbol NATURAL\$COMMAND	Set parameter CMSYNIN
Define channel for Input Data	Define logical symbol NATURAL\$INPUT	Set parameter CMOBJIN
Define channel for Output	Define logical symbol NATURAL\$OUTPUT	Set parameter CMPRINT

Optionally, you may set the parameter BMSIM to VM to generate as for Natural Version 2.1 output.

Natural Exit Codes

There are two types of Natural exit codes:

- Startup Errors, where Exit Code 1 is assumed to indicate success and all other exit codes are assumed to indicate errors.
- Errors generated by the TERMINATE statement, where Exit Code 0 to 255 is possible.

Special Considerations for Natural on OpenVMS

If Natural fails during startup or is terminated by the TERMINATE statement, you can query the exit code from within a DCL file. As opposed to Natural for UNIX, Natural for OpenVMS does not return low return code values such as "12" (startup error "Parameter module not found"). Instead, Natural for OpenVMS adds the value "268,435,456" to the Natural error code one would get using Natural for UNIX. If this value were not added, the image run-down handler would interpret the value "12" as a Natural internal access violation and a system access violation message would be printed.

Exception:

If Natural is terminated by using the TERMINATE statement with Value 0, Natural exits with Exit Code 1. This is because under OpenVMS the C-runtime makes no difference between an exit (0) and an exit (1) statement. In both cases, a 1 is returned.

You can retrieve this error code in a DCL command file by inquiring the status value \$STATUS. The following example shows how to retrieve the Natural exit code.

Recommendation:

The value of \$STATUS does not explain if this is a startup error or an error generated by the TERMINATE statement. For Natural version 5.1.1 there are about 60 startup errors defined. For that reason, use Exit Code 100 to 255 for TERMINATE, otherwise you could not clearly indicate the reason for the exit.

```

$ SET NOON
$ NAT := $NATBIN:NATURAL.EXE
$ DEFINE/USER NATINPUT SYS$INPUT
$ DEFINE/USER NATOUTPUT SYS$OUTPUT
$
$ nat parm=unknown
$
$! Note:
$! %X10000000 is hexadecimal and stands for the decimal number 268,435,456
$!
$ status_value = $STATUS .AND. .NOT. %X10000000
$
$ IF status_value .GE. 100
$ THEN
$ WRITE SYS$OUTPUT "NATURAL terminates by using TERMINATE statement"
$ ELSE
$ IF status_value .EQ. 1
$ THEN
$ WRITE SYS$OUTPUT "NATURAL terminates successfully"
$ ELSE
$ WRITE SYS$OUTPUT "NATURAL terminates with startup error " + "'status_value'"
$ ENDIF
$ ENDIF
$
$ EXIT

```


Natural Startup Errors

0	On UNIX: No error occurred.
1	On Open VMS: No error occurred.
2	Terminal Control String (TCS) capability specified in SAGtermcap or Environment Variable NATTCHARSET.
3	Failed to initialize character conversion table.
4	Failed to load character conversion table.
5	Error reading the DATABASE assignments in the global configuration file.
6	Unable to find FNAT or FUSER. Check your configuration files.
7	Cannot initialize LFILE table.
12	Unable to read specified parameter module. Please verify the parameter file.
13	Unable to read parameter module.
14	Storage manager initialization failed.
15	End of file (EOF) encountered while reading from STDIN.
16	Unable to open buffer pool; contact the Natural system administrator.
17	Error reading the buffer pool assignments in the local configuration file.
18	Invalid FDIC assignment.
19	Invalid FNAT assignment.
20	Invalid FSEC assignment.
21	Invalid FUSER assignment.
22	Unable to load Natural login module.
23	Unable to allocate memory for local data. Reduce USIZE and/or SSIZE parameter.
24	Unable to load Natural display module.
25,26	Error loading shareable image or DLL.
27	Login cancelled. Natural terminates.
28	Security violation during start of Natural. Natural terminates.
29	Security violation during start of Natural. Login aborted due to too many login failures.
30	Natural system error message raised.
31	This is not a security nucleus.
32	Password check failed.
33	Lock manager cannot create/initialize semaphores.
34	Unable to build Natural Library structure. Check your System File assignment. User/group settings might not be correct. S-bit setting might be required. Path specifications in global configuration file might be corrupted. LIBDIR.SAG might be missing.
35	Internal wfc i/o terminal driver error.
36	Internal XVT error.

37	DCOM Startup error.
38	Creation of runtime context failed.
39	NATDIR and/or NATVERS not set.
40	Natural zmodem error.
41	Creation of TF table failed because there are entries with different database types from older parameter module. Check parameter module with NATPARM utility.
42	Batch mode driver error.
43	Screen window size is too small.
44	Exit from SQL signal handler.
45	Unable to load add-on product.
46	Unable to access FNAT library SYSLIB. Insufficient privilege or file protection violation.
47	Unable to read PARM_PATH.
48	Unable to read NATURAL.INI for CONFIG_NAME.
49	Unable to read NATURAL.INI for NATTCAP.
50	Unable to read NATURAL.INI for NATCONV.
51	Unable to read TMP_PATH.
52	Unable to read PROFILE_PATH.
53	Unable to open NATURAL.INI.
54	Unable to read NATCONF.CFG for NATOSDEP.
55	Unable to read NATURAL.INI for NATEXTLIB.
56	Unable to find NATDIR in SAG.INI.
57	Unable to find DEFAULT-VERSION in SAG.INI.
58	Unable to find NATINI in SAG.INI.
59	Invalid option specified.
60	Not enough memory to initialize internal tables.
61	Batch error occurred, but processing continued due to CC=ON parameter.
62	More than one Natural session with active repository not allowed
63	Natural session with active repository already running

Invoking Natural Subprograms from 3GL Programs

Natural subprograms can be invoked from a programming object written in a 3rd generation programming language (3GL). The invoking program can be written in any programming language that supports a standard CALL interface.

For this purpose, Natural provides the interface "ncxr_callnat". The 3GL program invokes this interface with a specification of the name of the desired subprogram.

Note: Natural must have been activated beforehand; that is, the invoking 3GL program must in turn have been invoked by a Natural object with a CALL statement.

The subprogram is executed as if it had been invoked from another Natural object with a CALLNAT statement.

When the processing of the subprogram stops (either with the END statement or with an ESCAPE ROUTINE statement), control will be returned to the 3GL program.

Passing Parameters from the 3GL Program to the Subprogram

Parameter fields can be passed from the invoking 3GL program to the Natural subprogram. For passing the parameter fields, the same rules apply as for passing parameter fields with a CALL statement.

The 3GL program invokes the Natural interface "ncxr_callnat" with four parameters:

Parameter	Contents
1	The name of the Natural subprogram to be invoked.
2	The number of parameter fields to be passed to the subprogram.
3	The address of the table that contains the addresses of the parameter fields to be passed to the subprogram.
4	The address of the table that contains the format/length specifications of the parameter fields to be passed to the subprogram.

The sequence, format and length of the parameter fields in the invoking program must match exactly the sequence, format and length of the fields in the DEFINE DATA PARAMETER statement of the subprogram. The names of the fields in the invoking program and the invoked subprogram can be different.

For an example of how to invoke a Natural subprogram from a 3GL program, refer to the samples "MY3GL.NSP" (for the main program), "MY3GLSUB.NSN" (for the subprogram) and "MYC3GL.C" (for the "C" function).

For UNIX:

```
$NATDIR/$NATVERS/samples/sysexuex
```

For OpenVMS:

```
NATDIR:[v5110,samples]
```

Natural TX Interface (for UNIX only)

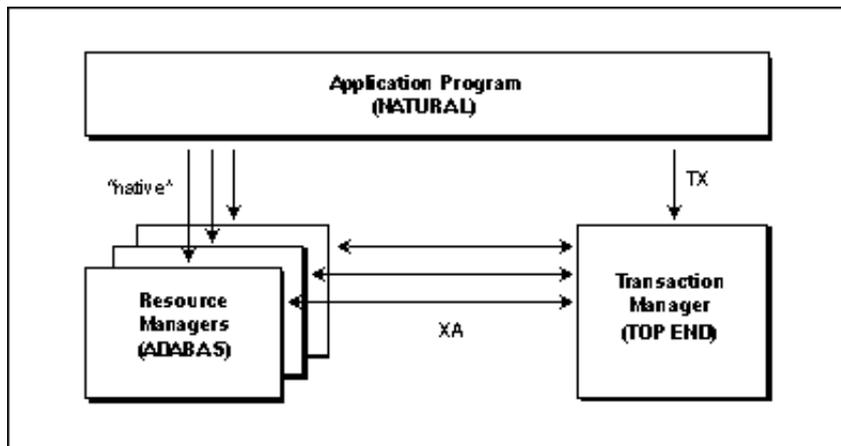
The Natural TX interface is an optional feature to enable Natural users to perform distributed transaction processing.

Overview

By supporting distributed transaction management systems, the Natural TX interface offers distributed transaction processing to the Natural user, based on the X/Open model for distributed transaction processing. This model comprises three basic elements:

- The application program; that is, Natural.
- The transaction manager; that is, for example, TOP END.
- The resource managers; that is, for example, several Adabas database management systems.

Note: Currently, the Natural TX interface only supports the database management system Adabas for UNIX Version 2.2.1 (or above) and the distributed transaction management system TOP END.



The Natural application program is the component that determines when a distributed transaction begins and ends. It uses resource (that is, database) managers to perform work that is to be part of the distributed transaction. Natural ends a transaction with an END TRANSACTION (commit) or BACKOUT TRANSACTION (rollback) statement and is not involved in the two-phase commit procedure.

The transaction manager processes the transaction begin and end statements from Natural and conveys this information to the database managers. It also coordinates the two-phase commit and transaction recovery processing.

A resource manager manages a certain part of the system's shared resources, for example, the databases.

The application programming interface (API) between the application program and the transaction manager is called TX interface; the system programming interface between the transaction manager and the resource managers is called XA interface; for information on the Adabas XA interface, refer to the Adabas for UNIX DBA Reference Documentation.

Operational Requirements

To be able to use the Natural TX interface, you have to relink the Natural nucleus. To do so, perform the following steps:

1. To make the Natural "build" directory the current directory, issue the command:

```
cd $NATDIR/$NATVERS/bin/build
```

2. To build a new Natural environment issue the command:

```
make Natural tp=yes ada=dyn
```

This "make" command generates a new "Natural" file which is linked to the TX interface.

3. To copy this new "Natural" file into the "bin" directory, issue the following command:

```
make install
```

To make the Natural TX interface operational, you have to modify your Natural parameter file as follows:

1. Invoke the NATPARM utility by entering the command "natparm" at the system prompt: The NATPARM menu is displayed.
2. Select the Edit option: A window appears with a list of parameter groups.
3. Select "DBMS": A second window is displayed with the individual parameters that belong to this group.
4. Select "XA Databases (XADB)"; see also the description of the XADB profile parameter: A further window is displayed, in which the DBIDs of all databases which will be used in distributed transaction processing must be marked.

Note: The selected databases must be equipped with an XA interface to the corresponding transaction manager.

Natural with TOP END

To make Natural known to TOP END, you have to define a new TOP END System and a new Application Component (for example "natural").

The name of the TOP END System should correspond to the user and group definitions specified for Natural (for example, "sag").

In addition, all relevant environment variables (TP_SYSTEM, TP_NAME, etc.) must be set at runtime.

Restrictions

When using the Natural TX interface, no transaction data can be stored with the END TRANSACTION statement or retrieved with the GET TRANSACTION DATA statement.

Using Versioning Software

If you want to use any third-party versioning software together with Natural, you have to write your own interface. This interface is called by Natural each time an object is modified. For more information on how to build this interface, refer to the example in directory "\$NATDIR/\$NATVERS/samples/natncvc".

Issuing OpenVMS System Commands from a Natural Program

The Natural user exit "SHCMD" can be used to issue a system command, call a DCL command procedure or execute an executable program on OpenVMS from within a Natural program.

Format:

```
CALL 'SHCMD' command [option]
```

Parameters:

Command	Command which is executed under control of the operating system command interpreter. Natural will wait until the command is completed and then Natural returns the control back to the Natural program.
Option This parameter is optional	Option describes how the command should be executed.

The following options are available:

SCREENIO	Used to refresh the Natural screen output after the command is completed
NOSCREENIO	Used to hide the output generated by the command. The hidden output is redirected to the null device.

Note: Option SCREENIO and NOSCREENIO may be not used at the same time.

Return Codes:

Natural will manipulate return codes from the control interpreter in the following way:

- Even numbers are interpreted as error codes and are retained unchanged.
- Odd numbers are interpreted as success codes and are converted to 0.

The following return code values are available:

0	Command successfully executed.
4	Illegal SHCMD parameter specified.
all others codes	Command has returned an error.

Examples:

Execute a DCL command procedure from within Natural:

```
CALL 'SHCMD' '@MYDCL.COM'
```

Execute an executable program from within Natural:

```
CALL 'SHCMD' 'MYPROGRAM.EXE'
```

Executing system command DIRECTORY on OpenVMS to view a directory:

```
CALL 'SHCMD' 'DIRECTORY'
```

After executing the DIRECTORY command, you will recognize that the output generated by the system command has changed the last Natural screen output. You have to press the refresh-screen key to clear the screen. To do this automatically, you can specify the 'SCREENIO' option.

```
CALL 'SHCMD' 'DIRECTORY' 'SCREENIO'
```

Retrieve the return code by using the RET function:

```
DEFINE DATA LOCAL
  1 rc (I4)
END-DEFINE
CALL 'SHCMD' 'DIRECTORY' 'SCREENIO'
ASSIGN rc = RET( 'SHCMD' )           /* retrieve return code
IF rc <> 0 THEN
  IF rc = 4 THEN
    WRITE NOTITLE 'Illegal option specified'
  ELSE
    WRITE NOTITLE 'Command not executed successfully (rc=' rc ') '
  END-IF
ELSE
  WRITE NOTITLE 'Command executed successfully'
END-IF
END
```

Execute the DCL command procedure MYSUCCESS.COM. Natural will recognize that the command procedure returns an odd return number and assumes success by mapping the exit code 15 to 0.

DCL command procedure MYSUCCESS.COM:

```
$ WRITE SYS$OUTPUT "DCL returns SUCCESS"
$ EXIT 15
```

Executing the DCL command procedure MYERROR.COM. Natural would recognize that the command procedure returns an even return number, assumes error and leaves the number unchanged.

DCL command procedure MYERROR.COM:

```
$ WRITE SYS$OUTPUT "DCL returns ERROR"
$ EXIT 14
```

Issuing UNIX System Commands from a Natural Program

The Natural user exit "SHCMD" can be used to issue a system command, call a shell script or execute an executable program on UNIX from within a Natural program.

Format:

```
CALL 'SHCMD' command [option]
```

Parameters:

command	Command which is executed under control of the operating system command shell. Natural will wait until the command is completed and then Natural returns control back to the Natural program.
option This parameter is optional	Option which describes how the command should be executed.

The following options are available:

SCREENIO	Used to refresh the Natural screen output after the command is completed.
NOSCREENIO This parameter is optional	Used to hide the output generated by the command. The hidden output is redirected to the null device.

Note:

Option SCREENIO and NOSCREENIO may be not used at the same time.

Return Codes:

The following return code values are available:

0	Command successfully executed.
4	Illegal SHCMD parameter specified.
All others codes This parameter is optional	Command shell return code.

Examples:

Execute a command shell from within Natural:

```
CALL 'SHCMD' 'myshell.sh'
```

Execute an executable program from within Natural:

```
CALL 'SHCMD' 'myprogram'
```

Execute system command ls on UNIX to list the contents of a directory:

```
CALL 'SHCMD' 'ls'
```

After executing the `ls` command, you will recognize that the output generated by the system command has changed the last Natural screen output. You have to press the refresh-screen key to clear the screen. To do this automatically, you can specify the `'SCREENIO'` option.

```
CALL 'SHCMD' 'ls' 'SCREENIO'
```

Retrieve the return code by using the `RET` function:

```
DEFINE DATA LOCAL
  1 rc (I4)
END-DEFINE
CALL 'SHCMD' 'ls' 'SCREENIO'
ASSIGN rc = RET( 'SHCMD' )           /* retrieve return code
IF rc <> 0 THEN
  IF rc = 4 THEN
    WRITE NOTITLE 'Illegal option specified'
  ELSE
    WRITE NOTITLE 'Command not executed successfully (rc=' rc ') '
  END-IF
ELSE
  WRITE NOTITLE 'Command executed successfully'
END-IF
END
```

Tuning SQL Database Access

By default, the Natural SQL driver manages a table with the 16 most recently used Natural statements. All statements in this table are marked as prepared, which indicates that the statement can be executed immediately without being compiled by the database system.

To ensure maximum performance, the dynamic parameters `SQLRELCMD` and `SQLMAXSTMT` are provided. These parameters configure the handling of the SQL driver's statement table. Note that these parameters are not profile parameters.

SQLRELCMD

This parameter determines when commands are to be released from the SQL statement table.
Possible values:

- `ENDGP` (default): if a generated program terminates, all statements from this program that are in the statement table are removed from the table;
- `NEVER`: No statement will be deleted from the table.

SQLMAXSTMT

This parameter determines the size of the statement table.
Possible values:

- 1 to 64 (default: 16)

If you set the `SQLMAXSTMT` parameter, please keep the following things in mind:

- Resource consumption may be higher if you are keeping more prepared statements in the table;
- if the size of the statement table exceeds the limit of dynamic SQL statements in the target database, the application will receive SQL errors;
- it depends on the database whether there is a real benefit from the `SQLMAXSTMT` optimization;
- in general, performance in batch-type applications will be improved if the number of `PREPARE` statements is minimized, while performance in online applications will probably be worse because of the increased resource consumption of the target database.

Example

To set the above parameters dynamically, enter the following at the Natural command prompt:

```
natural sqlrelcmd=never sqlmaxstmt=40
```

Natural will then start with a statement table size of 40 and the statement table will only be cleared when Natural is terminated.

User Exit for Computation of Sort Keys - `NATUSKnn`

Some national languages contain characters which are not sorted in the correct alphabetical order by a sort program or database system. With the system function `SORTKEY` you can convert such "incorrectly sorted" characters into other characters that are "correctly sorted" alphabetically.

When you use the SORTKEY function in a Natural program, the user exit NATUSKnn will be invoked - nn being the current language code (that is, the current value of the system variable *LANGUAGE).

You can write a NATUSKnn user exit in the C programming language using the CALL interface. The *character-string* specified with SORTKEY will be passed to the user exit. The user exit has to be programmed so that it converts "incorrectly sorted" characters in this string into corresponding "correctly sorted" characters. The converted character string is then used in the Natural program for further processing.

Note: A conversion table is currently not supplied.

NATUSKnn is called using the CALL interface. The parameters of the C function have the following values:

Parameter	Contents
1	The number of arguments.
2	The array of pointers to the operands.
3	the array of field information for each operand.

If you use the Natural system function #OP1= SORTKEY(#OP2), the source operand is in the arrays at index 0 and the target operand (#OP1) is in the arrays at index 1.

A sample user exit, natusk01.c, is provided in source form: it applies to English and converts all English lower case letters in the character string to upper case. The sample is to be found in \$NATDIR/\$NATVERS/samples/sysexuex, where you can also find the other user exits.

The source code of the example contains all comments which are needed to write a specific user exit for SORTKEY.

For linkage and loading conventions, refer to the CALL statement.

Profile Parameters

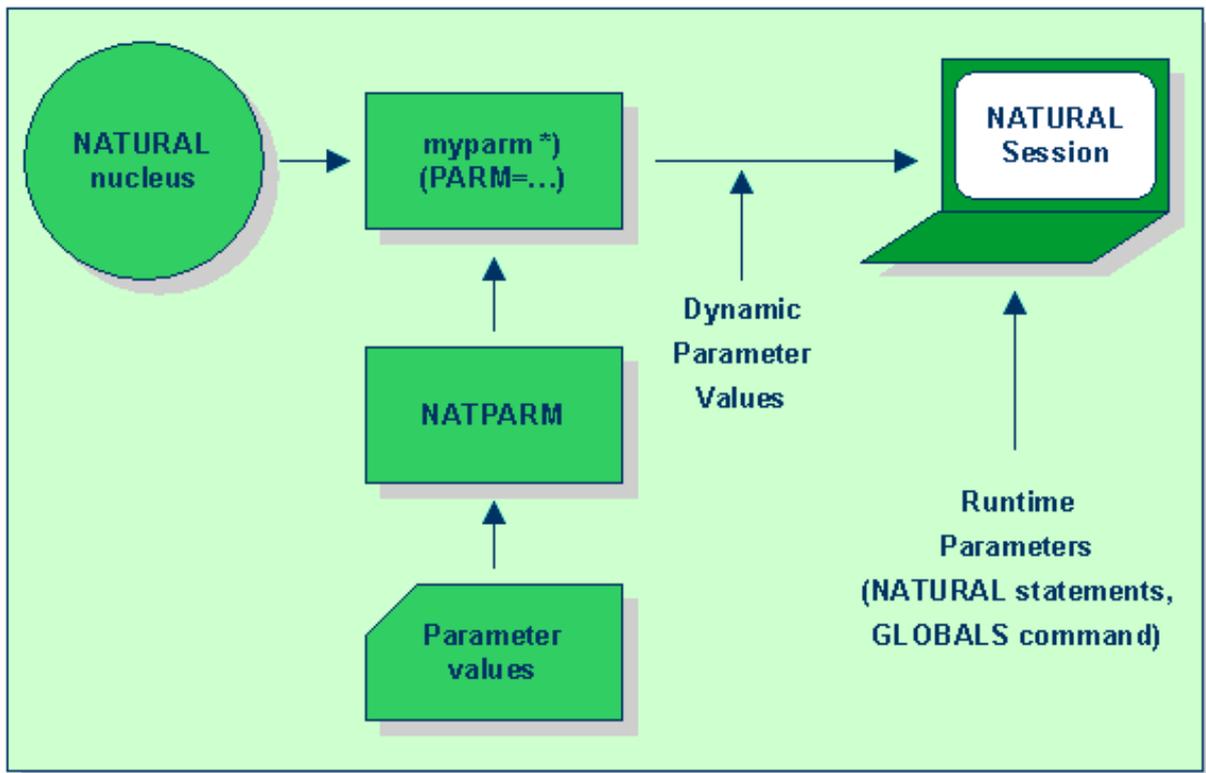
- General Information
- Creating Parameter Files
- NATPARM Utility
- Profile Parameters - Overview
- Dynamic Assignment of Parameter Values
- Runtime Assignment of Parameter Values

See also alphabetic list of Profile Parameters

General Information

The Natural profile parameters define various characteristics of the Natural environment. The values for these parameters are taken from the following three sources:

1. Runtime assignment of session parameters using the Natural SET GLOBALS statement or GLOBALS system command (highest priority).
2. Dynamic assignments which are valid for the current Natural session. These are made by specifying individual parameters and/or an alternative parameter file when starting Natural.
3. Static assignments, which are made by parameters specified in the Natural parameter file NATPARM (lowest priority).



* The file "myparm" represents an alternative user-defined parameter file, that overrides the settings in NATPARM.

Creating Parameter Files

By default, the parameter specifications in the parameter file NATPARM are used to determine the characteristics of your Natural environment. Initially, the NATPARM parameter file contains the default values as supplied by Software AG.

If you wish to use Natural with parameter values other than the system defaults, you use the NATPARM utility. With this utility, you can modify the default parameter file NATPARM and/or create your own parameter files.

Modifying NATPARM Parameters

Most parameters can be set by all users. Some parameters, however, can only be set by users who are authorized to do so. This authorization is granted by entering the user as "Natural Administrator" under Administrator Assignments in the Configuration section of the Natural Parameter Setting menu:

- Users who are Natural Administrators can view and set *all* parameters.
- Users who are not Natural Administrators can only view and set some parameters.

The following parameters can only be set by Natural Administrators (this is also indicated in the description of the individual parameters later in this section):

BPSFI	DD	ETID	NC	SD	TD
CDYNAM	DYNPARM	LC	OPRB	SHELL	USER
CM	ESCAPE	LSTEP	RECAT	SSIZE	USIZE
DBUPD	ET	MAINPR	ROSY	SYNERR	XREF

The following screen and printer configuration specifications for the Devices parameter group can also only be set by Natural Administrators:

- logical device name
- physical device name
- line size
- page size
- maxpage

NATPARM Utility

The NATPARM utility is used to create and modify parameter files.

To invoke the NATPARM utility, enter "NAT51PARM" at the system prompt. A menu is displayed with the name of the parameter file currently active or to be edited in the top right-hand corner of the screen.

Note:

Parameter files are upward-compatible, but not downward-compatible. Once you have created a parameter file in a higher version of the NATPARM utility and you try to read it with a lower version, an error will be displayed. To be able to read the file, export the file to the higher version and then import this file to the lower version of the NATPARM utility.

The menu provides the following options:

Option	Use
File	To maintain parameter files; see below.
Edit	To set the parameters of the current parameter file; see Profile Parameters - Overview.
Configuration	To change the Natural local and global configuration files, which are described in the section Configuration Files.
Search	To access a specific profile parameter directly, without having to use the Edit option; see Search.

File

When you select File, a selection list containing the following functions is displayed:

Function	Explanation
New	Opens a new, unnamed parameter module and sets the parameters to default values.
Open	Selects a parameter file from a list for subsequent editing.
Save	Saves all parameters in the current parameter file.
Save As	Saves all parameters under a different parameter-file name.
Export	Generates a text file showing the current values of the active parameter file. In text file form, you can transfer a parameter file to another hardware platform, where you have to generate a binary file from the text file to make the parameter file usable by Natural. This generation is done with the Import function (see below). The text file must be stored as " <i>parameter-file.LST</i> " (in any directory).
Import	Generates a binary parameter file from a parameter file generated as text file by the Export function (see above). The generated binary file must then be saved with the Save function (see above).
Delete	Deletes an existing parameter file. When you invoke this function, a list of parameter files is displayed, from which you select the file to be deleted.
Exit	This function is used to leave the NATPARM utility.

Search

When you select Search, a window appears that gives you the choice to directly find a Natural profile parameter by entering its name in the Search Parameter input field.

If you enter a qualified parameter name, you are taken directly to the corresponding window within the Edit option. If you use asterisk notation (*), a list with all existing profile parameters is displayed for selection.

For details on the Edit option and for further information on the individual profile parameters, see Profile Parameters - Overview.

Changing Parameter Settings

To change parameter settings, proceed as follows:

1. Use the Open function to select the desired parameter file.
2. Use the Edit or Search option to modify the parameters of the selected parameter file as necessary.
3. Use the Save or the "Save As" function to save your modifications.

The alternative parameter file can be activated by invoking the command:

natural parm=*parameter-file*

Performing NATPARM Functions from the Operating System Prompt

The following NATPARM functions can also be performed directly from the operating system prompt:

- exit
- import=
- export=
- parm=
- save
- save=

Multiple functions can be specified one after the other (see the examples below).

Examples:

nat51parm parm=*parameter-file*

With this command, you invoke the NATPARM utility and at the same time perform the Open function of the NATPARM utility: the specified *parameter-file* is selected and then is available for editing in the NATPARM utility.

nat51parm parm=*parameter-file* save=*new-parameter-file-name*

With this command, you invoke the NATPARM utility and at the same time perform the Open and Save functions of the NATPARM utility: the specified *parameter-file* is selected and saved under *new-parameter-file-name*.

nat51parm import=*parameter-file* save

With this command, you invoke the NATPARM utility and at the same time perform the Import and Save functions of the NATPARM utility: the specified *parameter-file* is imported and saved.

Profile Parameters - Overview

The individual parameters are divided into groups according to their functions.

If you select the "Edit" option of the "Natural Parameter Setting" menu, a list of the following parameter groups is displayed:

The following sections provide an overview of the parameters contained in the individual groups.

Adabas Parameters

If Natural is used with Adabas, the following parameters should be reviewed and, if necessary, the default values should be adjusted to meet your specific requirements:

Parameter	Function
ET	Execution of END / BACKOUT TRANSACTION statements.
ETID	Adabas user identification.
OPRB	Adabas open/close processing.
WH	Record hold processing.

Batch Mode

Parameter	Function
BATCH	Enable Batch Mode Simulation.
BATCHMODE	Enable Real Batch Mode.
BMBLANK	Display Trailing Blanks.
BMCONTROL	Display Control Characters.
BMFRAME	Frame Characters.
BMSIM	Similar Output.
BMTIME	Display Process Time.
BMTITLE	Display Window Title.
BMVERSION	Display Natural Version.
CC	Enable Error Processing.
CMOBJIN	Input Data Channel.
CMPRINT	Output Channel.
CMPRTnn	Output Channel.
CMSYNIN	Input Commands Channel.
ECHO	Display Input Data.
ENDMSG	Session End Message.
NATLOG	Natural Log File.

Buffer Sizes

Natural uses several buffer areas for the storage of programs and data. You may need to adjust their sizes in order to achieve maximum buffer efficiency.

Parameter	Function
DSLMM	Data size limitation.
EDTBPSIZE	SAG Editor Bufferpool Size.
EDTLFILES	SAG Editor Logical Files.
SORTSZE	Size of sort buffer area.
SSIZE	Source area size.
USIZE	Size of user buffer area.

Character Assignments

The following parameters are used to change default character assignments:

Parameter	Function
CF	Control character for terminal commands.
CLEAR	Processing of CLEAR key at runtime.
DC	Character to be used as decimal point.
FC	Filler character for maps generated with an INPUT statement.
HI	Character to invoke field- or map-related help.
IA	Input assign character.
ID	Input delimiter character.

Once a character has been defined to replace a default character, this character cannot be used as data.

Compiler Options

The following parameters are used to set options for the Natural compiler:

Parameter	Function
DBSHORT	Interpretation of database short names.
GFID	Generation of global format IDs.
V22COMP	Natural Version 2.2 compatibility option.

Database Management System

The following parameters are used to assign database management system settings:

Parameter	Function
ETDB	Database for transaction data.
LFILE	Dynamic specification of a logical file.
LFILMAX	Maximum number of logical files.
TF	Translation of file number.
UDB	User database ID.
XADB	XA databases.

DCOM Support

The following parameters are used to provide DCOM support:

Parameter	Function
ACTPOLICY	Activation policy.
AUTOREGISTER	Automatic update of registry.
COMSERVERID	Server name.

Devices

The "Devices" parameter group is used to modify your screen and printer configurations as well as report assignments.

Device Parameter Assignments

- With the "Dev. Para. Ass" function (logical device VIDEO, LPT1 to LPT31), you can specify line size, page size and maximum number of pages for your video output device and for your printers.
- In addition, you can specify a physical device. "Physical Dev." denotes the name of any program which reads from standard input (STDIN/SYSS\$INPUT), for example the standard print command of your printer spooler with all options:
 UNIX System V: **lp <options>**
 BSD-UNIX: **lpr <options>**
 Line size, page size and maximum page number should be compatible with your hardware printer assignments.

Report Assignments

With the "Report Assignments" function, you can assign a Natural report number (Report 1 to Report 31) to a logical device name. Possible values for the output medium are: VIDEO, LPT1 to LPT31, SOURCE (source area), DUMMY and INFOLINE.

Report Number 0 must be set to VIDEO and is not reassignable; no report number other than 0 can be assigned to VIDEO.

In addition to the logical device name, you can assign a printer profile as defined in the global configuration file. All defined printer profiles are listed for selection in the "Profile" combo box. Select "blank" if you do not want to use any of these profiles.

Entire Transaction Propagator

The following parameters are used in conjunction with Software AG's Entire Transaction Propagator (ETP).

Parameter	Function
ETPDB	Database list for Entire Transaction Propagator.
ETPSIZE	Size of buffer for Entire Transaction Propagator.

Environment Assignments

The following parameters are used to adjust Natural environment variables:

Parameter	Function
EDITOR	Other program editor.
EDTRB	Program editor ring buffer.
LSTEP	Natural steplibs.
SHELL	Shell exit from Natural main menu.
USER	User ID.

Error Handling

The following parameters are used to control error handling within Natural.

Parameter	Function
IKEY	Error processing for PA/PF keys.
MSGSF	Display system error messages in full.
SA	Sound terminal alarm.
SNAT	Sound bell in the case of a syntax error.

Function Keys

The following parameter is used to assign values to PA, PF and CLEAR keys:

Parameter	Function
KEY	Value assignments to PA, PF and CLEAR keys.

Limits

The following parameters are used to prevent a single program from consuming an excessive amount of internal resources:

Parameter	Function
LE	Limit for error processing.
LT	Processing loop limit.
MADIO	Maximum number of DBMS calls.
MAXCL	Maximum number of program calls.
SD	System time delay.

Miscellaneous

These parameters control various functions within the Natural environment.

Parameter	Function
BPSFI	Search first in buffer pool.
CM	Command mode allowed.
CO	Compiler output.
DBUPD	Database updating.
DD	Day differential.
DU	Dump generation.
DYNPARM	Dynamic parameters.
ESCAPE	Enable % %.
FCDP	Filler character for dynamically protected input fields.
FS	Default format/length setting for user-defined variables.
IM	Default input mode.
KC	Keyword checking.
NC	Control use of Natural system commands.
NENTRY	Entry of numeric fields.
OPF	Overwriting of protected fields by helproutines.
PC	Personal computer support.
PD	Size of page dataset.
RECAT	Dynamic recataloging.
REINP	Issue internal REINPUT statement for invalid data.
SM	Programming mode (structured or reporting mode).
STACK	Place data on Natural stack.
SYMGEN	Generation of symbol tables.
SYNERR	Control of syntax errors.
TD	Time differential.
TQ	Translation of quotations marks.
ULANG	User language.
XREF	Handling of cross-reference data (only applicable with Predict).
ZD	Zero division.

Program Loading and Deletion

The following parameters are used to control the dynamic loading and deletion of programs:

Parameter	Function
CDYNAM	Dynamic loading of non-Natural programs.
ETA	Program to receive control after error in transaction.
PROGRAM	Program to receive control after Natural termination.
PRGPAR	Data to be passed to the program defined by the parameter PROGRAM.
ROSY	Disable storage of Natural programs.

Remote Debugging

The following parameters are used to allow for remote debugging:

Parameter	Function
RDACTIVE	Activate remote debugger.
RDNODE	Remote debugger node name.
RDPORT	Remote debugger port.

Remote Dictionary Access

The following parameter is used for remote dictionary access:

Parameter	Function
USEDIC	Remote dictionary access.

Remote Procedure Call

The following parameters are used for options in the Natural Remote Procedure Call (RPC):

Parameter	Function
ACIPATT	Node pattern for ACI.
ACIVERS	ACI Version.
AUTORPC	Automated remote execution.
COMPR	Send buffer compression.
CSCPATT	Node pattern for CSCI.
DFS	Default server.
LOGONRQ	Logon required for server request.
MAXBUFF	Request buffer size.
RDS	Remote directory servers.
RPCSIZE	RPC buffer size.
SERVER	Start session as RPC server session.
SRVNAME	Server name.
SRVNODE	Server node.
SRVUSER	Server user ID.
TIMEOUT	Request timeout.
TRACE	RPC trace level.
TRANSP	Transport protocol.
TRYALT	Retry Service on alternate server.

Report Parameters

The following parameters control various attributes of Natural reports:

Parameter	Function
CVMIM	Control variable modified at input.
DTFORM	Date format.
EJ	Page eject control.
LC	Enable lower case.
LS	Line size.
MAINPR	Override default report number.
PM	Print mode.
PS	Page size.
SF	Spacing factor between fields.
ZP	Zero printing.

System Files

The following parameters are used to specify the Natural system files:

Parameter	Function
FDIC	Predict system file.
FNAT	Natural system file for system programs.
FSEC	Natural Security system file.
FUSER	Natural system file for user programs.

System Variables

The following parameters are used to adjust Natural system variables for the start of a Natural session:

Parameter	Function
AUTO	Automatic logon.
INIT-LIB	Startup library.
STARTUP	Startup program.

Work Files

The following parameters can be used to specify work-file settings:

Parameter	Function
WFOPFA	Work file to be opened on first access.
WORK	maximum number of work files.
CMWRKnn	Natural work files.

Year 2000 Support

The following parameters can be used for Year 2000 support:

Parameter	Function
DFOUT	Date format on output.
DFSTACK	Date format on stack.
DFTITLE	Date format in standard titles.
YSLW	Year sliding window.

Dynamic Assignment of Parameter Values

Dynamically assigned parameter values are used to override Natural profile parameter settings, as specified in the default parameter file NATPARM, for the duration of a single Natural session, and/or to select a specific system profile that is to be in effect for the Natural session.

A value can be assigned dynamically to a profile parameter if "Dynamic Specification" is specified as "YES" in the description table of the parameter:

Possible values	
Default value	
Dynamic specification	YES
Specification within session	

The setting of dynamic parameters enables a certain environment to be set up when starting Natural. The values for the dynamic parameters are passed by the operating system to Natural when the session is started. To specify such parameter values, do the following:

- Select the Natural program-item icon, click the right mouse button and choose the "Properties" function. The "Natural Properties" dialog box appears. Choose the "Shortcut" item. You can enter the desired dynamic parameters and their values in the "Target" field after the Natural path.

Examples:

UNIX:

Natural PARM=MYPARM SM=ON DTFORM=I

OpenVMS:

\$ NAT51 PARM=MYPARM SM=ON DTFORM=I

OpenVMS example only: "NAT51" is defined in SAG\$Root:[Natural]login.com as foreign command for passing parameters to Natural.

The parameter file MYPARM is to be used for the Natural session.

SM and DTFORM are dynamic parameters.

Runtime Assignment of Parameter Values

A value can be assigned to a profile parameter at runtime if "Specification within Session" is specified as "YES" in the description table of the parameter:

Possible values	
Default value	
Dynamic specification	
Specification within session	YES

The parameter keyword and the required value are entered as session parameter following the Natural SET GLOBALS statement or GLOBALS system command.

Examples:

SET GLOBALS SA=ON,IM=D

GLOBALS SA=ON,IM=D

Note:

The SET GLOBALS statement can be used in reporting mode only.

Natural Buffer Pool

- General Information
 - Setting up a Buffer Pool
 - NATBPSRV Utility to start a Buffer Pool
 - NATBPMON Utility to Monitor or Shut Down a Buffer Pool
 - Invoking NATBPMON
 - NATBPMON Commands
 - Buffer-Pool Trouble Shooting
 - Problem 1 (under OpenVMS)
 - Problem 1 (under UNIX)
 - Problem 2 (under UNIX only)
 - Shutting Down and Restarting the Buffer Pool
 - Dispatch Vector Addresses (OpenVMS/VAX only)
-

General Information

- Introduction
- Object Types
- Coordination under UNIX
- Coordination under OpenVMS
- Multiple Buffer Pools
- Storing Objects in the Buffer Pool
- Restrictions

Introduction

The purpose of the Natural buffer pool is to share Natural objects between several Natural users working on the same computer. It is a pool of storage into which programs compiled with Natural are loaded in preparation for execution. They are moved into and out of the buffer pool as Natural users request Natural objects.

Since Natural generates reentrant Natural object code, it is possible that a single copy of a Natural object can be executed by more than one user at the same time. For this purpose, each object is loaded only once from the system file into the Natural buffer pool, instead of being loaded by every caller of the object.

Object Types

Objects in the buffer pool can be programs, subprograms, maps, global data areas, local data areas, parameter data areas and copycodes. Local data areas, parameter data areas and copycodes are only placed in the buffer pool for compilation purposes.

When a Natural object is loaded into the buffer pool, a control block is allocated for that object. This control block contains information such as the name of the object, what library or application it belongs to, what database ID and Natural system file number the object was retrieved from, and certain statistical information (for example, the number of users who are concurrently executing a program).

Coordination under UNIX

Resource sharing requires that access to the buffer pool is coordinated among all users. Several system resources are necessary to accomplish this. For example, shared memory on the UNIX operating system is used to store the objects and their administrative information. To synchronize access to these objects, a set of semaphores is used. The amount of available shared memory and the number of semaphores is configured statically in the operating system, and as a result, it may be necessary to change system parameters and to recreate the operating system kernel for your installation. Further information about these topics is system-dependent and described in the installation documentation for your UNIX computer.

Coordination under OpenVMS

For OpenVMS, the buffer pool uses a standard locking mechanism which need not be configured by the administrator.

Multiple Buffer Pools

Several instances of the Natural buffer pool can be started on one computer, depending on the individual requirements. It is also possible to run different versions of the buffer pool on one computer without any problems. These buffer pools have nothing in common, except that they run on the same computer.

Storing Objects in the Buffer Pool

When a user executes a program, a call is made to the buffer pool manager. The directory entries are searched to see if the program exists. If it does not exist in the buffer pool, a copy is retrieved from the appropriate library and loaded into the buffer pool.

When a Natural object is being loaded into the buffer pool, a new directory entry is defined to identify this object, and one or more other Natural objects which are currently not being executed may be deleted from the buffer pool in order to make room for the newly loaded object.

For this purpose, the buffer pool maintains a record of which user is currently using which object, and it detects situations in which a user exits Natural without releasing all its objects. It dynamically deletes unused or out-of-date objects to accommodate new objects belonging to other applications.

Restrictions

When using the Natural buffer pool, only minimum restrictions must be considered:

- When a Natural session hangs, do not terminate it by using the OpenVMS command "STOP" or the UNIX command "kill", the terminal command "break" or the "interrupt" key.
If this process is currently performing changes in the buffer pool internal data structures, an interruption may occur at a stage where the update is not fully completed. If the buffer pool internal data structures are inconsistent, this could have negative effects.

Note: This can only happen when the Natural nucleus is executing buffer pool routines.

- All resources must be shared among all users of one Natural buffer pool.
Group membership of a process is used to give access rights for the buffer pool. This means that the shared memory can be changed by all group members, but not by anyone else. The same applies to the semaphores.

Note: All users of the same Natural buffer pool must belong to the same user group on the UNIX operating system. OpenVMS group or system membership depends on how the SYSTEMWIDE parameter is set (0 or 1).

The following applies for OpenVMS only:

- Since on OpenVMS the common resource 'bufferpool' is administrated by the OpenVMS lock manager, each bufferpool on each node has to be addressed by its own cluster-wide unique LCKRESNAME and GSDNAME. To ensure this, the installation procedure inserts for each node where it is executed a new node-specific bufferpool section into the Natural.ini file. Consequently the bufferpool service has to be started on a given node using its specific BPID, therefore the natetc:natbpenv_<node>.com was changed. Also each Natural session has to be started using the BPID for the node where the user is currently logged on. To ensure that, the symbol to start Natural contains the string BPID = BP <node> (defined in the LOGIN.COM for Natural). If applications with different priorities are in use, then every application should have its own bufferpool. (System Dead Lock).

Setting up a Buffer Pool

A buffer pool is set up by making various specifications with the NATPARM utility.

To invoke the NATPARM utility, enter "NAT51PARM" at the system prompt. The Natural Parameter Setting menu is displayed on which you first select "Config" and then "Local Configuration File".

Note: If the "Config" option is not displayed on your Natural Parameter Setting menu, this means that you have no permission to modify the configuration files (see the section Administrator Assignments).

In the window that appears, select "Buffer Pool Assignment". A further window is displayed in which you can make the following buffer pool specifications:

General Parameters

- BPNLE - Number of Directory Entries
- Buffer Pool ID (BP or BPID) - Name of the Buffer Pool.
- MAXUSERS - Maximum number of concurrent users.
- MEMSIZE - Size (in MB) of the buffer pool.

Parameters for UNIX Only

- Shared Memory Key
- Semaphore Key

Parameters for OpenVMS Only

- GSDNAME
- LCKRESNAME
- SECTIONFILE
- SFDELETE
- SYSTEMWIDE

Buffer Pool ID (BP or BPID) - Name of the Buffer Pool

Possible values:	<i>name</i> (1 - 8 characters)
Default value:	NATBP

With this parameter, you specify the name (ID) of the buffer pool. BP or BPID can also be specified dynamically.

MAXUSERS - Maximum Number of Concurrent Users

Possible values:	1 - 5000
Default value:	20

This parameter determines the maximum number of users that can have simultaneous access to the buffer pool. This number determines the sizes of some internal tables stored inside the shared memory during startup.

MEMSIZE - Size of the Buffer Pool

Possible values:	1 - 512
Default value:	3

This parameter determines the size (in MB) of the buffer pool. This value is used during startup by the NATBPSRV utility to create a shared memory segment of the specified size.

Shared Memory Key (for UNIX only)

Possible values:	00000000 - FFFFFFFF
Default value:	<i>none</i>

With this parameter, you specify the shared memory key for the IPC facilities that the buffer pool is using. Ask your UNIX administrator to supply the value for this parameter.

Note: The shared memory key must be unique across your entire UNIX system.

Semaphore Key (for UNIX only)

Possible values:	00000000 - FFFFFFFF
Default value:	<i>none</i>

With this parameter, you specify the semaphore key for the IPC facilities that the buffer pool is using. Ask your UNIX administrator to supply the value for this parameter.

Note: The semaphore key must be unique across your entire UNIX system.

SYSTEMWIDE (for OpenVMS only)

Possible values:	0 or 1
Default value:	1

With this parameter, you specify whether the system resources of a buffer pool, such as locks and the global section, are created system-wide or group-wide. Value "1" specifies system-wide, Value "0" specifies group-wide.

Note: It is recommended to set the value of SYSTEMWIDE before starting the buffer pool and to keep the chosen value until the buffer pool has been terminated.

SFDELETE (for OpenVMS only)

Possible values:	0 or 1
Default value:	0

With this parameter, you specify whether the section file is to be deleted when the buffer pool is shut down. Value "0" specifies it is not deleted, value "1" specifies it is deleted.

LCKRESNAME (for OpenVMS only)

Possible values:	any valid filename
Default value:	<i>filename</i>

With this parameter, you specify the name of the lock resource needed by the buffer pool.

GSDNAME (for OpenVMS only)

Possible values:	any valid section name
Default value:	<i>sectionname</i>

With this parameter, you specify the name of the global section to be used by the buffer pool.

SECTIONFILE (for OpenVMS only)

Possible values:	any valid filename
Default value:	NATBPDEV

With this parameter, you specify the file that contains the global section. The default value NATBPDEV is a logical name which is defined when the buffer pool is started by the command procedure "NATETC:NATBPENV_<node>.COM". This command procedure is created during installation.

BPNLE - Number of Directory Entries

Possible values:	10 - 9999
Default value:	10

This parameter determines the number of directory entries. This value is used by NATBPSRV during the start of the buffer pool. If BPNLE has reached the maximum value, this value is automatically increased.

NATBPSRV Utility to Start a Buffer Pool

The NATBPSRV utility is used to start a Natural buffer pool.

Note.

This utility should not be accessible to all Natural users, because it can cause damage to the work of other buffer pool users.

NATBPSRV allocates the resources required by the buffer pool and creates the permanent communication facilities (that is, shared memory and semaphores) used for the buffer pool. The necessary specifications for the resources and facilities are made with the NATPARM utility (as described in the section **Setting up the Buffer Pool** earlier in the section).

This NATBPSRV utility should only be used during system startup, from within the startup file "startup_nat4.com" (for OpenVMS) or "natstart.bsh" (for UNIX).

By default, the buffer pool "NATBP" is started. If another buffer pool is to be started, you specify its name with the following NATBPSRV command line option:

NATBPSRVBP = *buffer-pool-name*

If in the process of starting a buffer pool NATBPSRV discovers that a buffer pool of the same name is already active, it deletes the already active buffer pool. If the deletion fails, NATBPSRV terminates with an appropriate error message.

NATBPMON Utility to Monitor or Shut Down a Buffer Pool

The Natural utility NATBPMON is used to monitor the buffer pool's activity during its operation. NATBPMON can also be used to shut down the buffer pool, when Natural must be stopped on a computer.

Note.

This utility should not be accessible to all Natural users, because it can cause damage to the work of other buffer pool users.

NATBPMON collects information on the current state of your Natural buffer pool.

The buffer pool contains Natural objects (such as maps, programs and subprograms). When an object is invoked, Natural tries to find the object in the buffer pool. If the object is found in the buffer pool, it will be executed without accessing the system file where it is stored. If the object is not found in the buffer pool, the system file containing the object will be accessed, the object copied, and the copy placed in the buffer pool.

All objects in the buffer pool are node-specific and if an object is updated or changed in one node, the object will automatically be deleted from the buffer pool running on this node. But if the same object exists in the buffer pool of another node, the object in the buffer pool of the other node remains unchanged.

Invoking NATBPMON

You invoke the NATBPMON utility by entering a command at the DCL command line.

If the buffer pool "NATBP" is to be used, you invoke NATBPMON as follows:

```
$ NAT51BPMON
```

If another buffer pool is to be used, you invoke NATBPMON as follows:

```
$ NAT51BPMON BP = buffer-pool-name
```

Once you have invoked NATBPMON, the following prompt is displayed:

```
NATBPMON>
```

NATBPMON Commands

NATBPMON provides several commands, that you enter at the NATBPMON> prompt. The individual commands are described below.

CLEAR

```
CLEAR
```

This is a synonym of the ZERO command.

CORPSES

```
CORPSES
```

The CORPSES command is used to display the list of "corpses".

A "corpse" is an object that has been deleted, but was still being used in the buffer pool when its deletion took place.

Once this object is no longer being used, it will automatically disappear from the list of "corpses".

Note: The column "cusr" (described with the DIR command) indicates if an object is being used.

DELETE

```
DELETE {pattern
        *}
```

The DELETE command is used to delete an object from the buffer pool.

All objects can be deleted from a buffer pool by using an asterisk (*).

A *pattern* is used to specify a collection of objects, similar to current operating systems which allow the specification of a class of files with wildcards. For details on how to specify a *pattern*, see the DIR command below.

DIR

```
DIR {pattern
     *}
```

The DIR command is used to display a directory listing objects in a buffer pool.

This list of objects contains the following information:

Column	Explanation
indx	A number that the NATBPMON utility automatically assigns to an object when it is loaded into the buffer pool.
cusr	The current number of users that are using an object in the buffer pool.
pusr	The peak number of concurrent activations of an object in the buffer pool.
nusg	The number of times an object has been activated in the buffer pool.
g	Specifies if an object is being loaded into the buffer pool from the system file and will have either of the following values: 0 - The object is not being loaded. 1 - The object is being loaded.
size	Specifies the size (in bytes) of an object in the buffer pool.
key	Specifies the following information about an object: N - The name of the object. L - The library in which the object is located. K - The kind of object (G = generated object module; S = source; D = part of FILEDIR.SAG). T - The object type (which is "blank" in the case of "D" in the K field).

When the DIR command is issued, all objects in the pool will be displayed in a notation similar to the following:

```
NATBPMON>dir
indx:      index of the element
cusr:      current number of concurrent users
pusr:      peak number of concurrent users
nusg:      number of usages
g :        set if object is generating
indx  cusr  pusr  nusg  g   size  key
  1    0    1    4    0   920   (N = "SEL-MAP" L = "DEMO" K = 'G' T = 'M')
```

indx	cusr	pusr	nusg	g	size	key
1	0	1	4	0	920	(N = "SEL-MAP" L = "DEMO" K = 'G' T = 'M')
2	0	1	2	0	3096	(N = "EMWND" L = "DEMO" K = 'G' T = 'P')
3	0	1	4	0	604	(N = "HDR" L = "DEMO" K = 'G' T = 'P')
4	0	1	7	0	412	(N = "MMUPROG1" L = "RPA" K = 'G' T = 'P')
5	0	1	5	0	372	(N = "MMUPROG2" L = "RPA" K = 'G' T = 'P')
6	0	1	4	0	372	(N = "MMUPROG3" L = "RPA" K = 'G' T = 'P')

To select some objects, it is possible to restrict the values of certain key fields by specifying a matching pattern expression.

To restrict the allowed field values of a given field, the following pattern notation must be used:

name = expression

You can specify multiple patterns by separating them with a comma.

The specified patterns must all match their corresponding fields in order to accept the entire key field.

The expression accepts the specification of the wildcard characters "*" and "?".

The character "*" matches any sequence of characters (also none), and the wildcard character "?" matches exactly one character.

Examples

To select all objects of type "P" in the sample above, the following command would be used:

DIR T = P

To select all programs in the demo library, the following command would be used:

DIR T = P, L = DEMO

To select all objects containing an "M" in their name, the following command would be used:

DIR N = *M*

The special pattern "*" exists, which matches all objects stored in the buffer pool. To select all objects, the following command can be used:

DIR *

DUMP

DUMP

Do not use this command unless you are requested to do so by Software AG Support.

The DUMP command is used for error analysis.

EXIT

```
EXIT
```

The EXIT command is used to exit the NATBPMON utility.

HELP

```
HELP
```

The HELP command is used to display a list of all available NATBPMON commands.

IPCRM

```
IPCRM
```

The IPCRM command is used to free the resources allocated to the buffer pool.

This command should only be issued following a SHUTDOWN command (see below) when there are no active users.

KILL

```
KILL n
```

The KILL command is used to start the OpenVMS command "stop" from within the NATBPMON utility in a safe way; that is, without the danger of destroying the internal data structure of the buffer pool.

As "*n*" you specify the number of the user to be "killed". This number corresponds to the *index* number as displayed by the WHO command.

PARAM

The PARAM command is used to display the parameter settings for the buffer pool .

Example under Open VMS

```
NATBPMON>param
Active since .....: 25-OCT-2001 11:10:46.14, Version 1.3(435)
Last time cleared .....: 25-OCT-2001 11:10:46.14
Bpid .....: BPALF2
Systemwide .....: 1
Sfdelete .....: 0
Lckresname .....: NATBPLCKV5121_ALF2
Gsdname .....: NATBPV5121_ALF2
Sectionfile .....: NATBPDEV
Memsize .....: 1048576
Maxusers .....: 10
```

QUIT

QUIT

The QUIT command is used to exit the NATBPMON utility; it is a synonym for EXIT.

SHUTDOWN

SHUTDOWN

The SHUTDOWN command is used to shut down the buffer pool.

No new users will be able to use the buffer pool once this command has been issued.

As soon as all users have stopped using the buffer pool, the buffer pool's resources can be deleted with the IPCRM command (see above).

STATUS

STATUS

The STATUS command is used to display statistical information about the buffer pool.

The following statistics are displayed:

```

NATBPMON> status
Active since .....: 25-OCT-2001 11:10:48.35, Version 1.3(435)
Last time cleared .....: 25-OCT-2001 11:10:48.35
Bpid .....: NATBP
Allocated memory (bytes) .....: 1256436 Current users .....: 2
Smallest allocation .....: 20 Peak users .....: 2
Largest allocation .....: 145404 Dead users purged .....: 4
Free memory (bytes) .....: 9229336
Smallest free .....: 15996
Largest free .....: 6355900

Dormant objects .....: 204 Smallest object (bytes) .....: 316
Active objects .....: 0 Largest object (bytes) .....: 43048
Generating objects .....: 0 Total object sizes .....: 1072570
Obsolete objects .....: 0

Attempted locates .....: 126396 Stored objects .....: 0
Attempted fast locates .....: 60116 Loaded objects .....: 3126
Successful fast locates .....: 55528 Activated objects .....: 114323
Percent .....: 92.37 Aborted loads .....: 0

Dormant objects purged .....: 0 Peak parallel activations ...: 2
Object reuse factor .....: 36.57
NATBPMON>
    
```

The individual statistics are explained in the following tables.

General Information

Active since	Date and time when the buffer pool was started and the version number of the buffer pool.
Last time cleared	Date and time when the buffer pool was most recently cleared.
BPID	Buffer pool ID.

Memory Allocation

Allocated memory (bytes)	Sum total of all allocated memory.
Smallest allocation	Smallest amount of allocated memory.
Largest allocation	Largest amount of allocated memory.
Free memory (bytes)	Sum total of all free memory.
Smallest free	Smallest amount of contiguous free memory.
Largest free	Largest amount of contiguous free memory.

User Statistics

Current users	Number of users currently using the buffer pool.
Peak users	Peak number of users that have been using the buffer pool.
Dead users purged	Number of inactive users, that have been deleted from the buffer pool.

Object Use Statistics

Dormant objects	Number of available, but inactive objects. These objects are in the buffer pool, but are not being used. They are available for later use.
Active objects	Number of active objects. These objects are currently in use by one of the buffer pool users.
Generating objects	Number of objects that are currently being loaded into the buffer pool. These objects will become available as soon as the load operation completes.
Obsolete objects	Number of objects that are to be deleted from the buffer pool, but are still being used. These object can be displayed by using the CORPSES command.

Object Size Statistics

Smallest object (bytes)	Size of smallest object in the buffer pool.
Largest object (bytes)	Size of largest object in the buffer pool.
Total object sizes	Sum total of all objects in the buffer pool.

Locate Statistics

Attempted locates	Number of object locates. This is the number of object activations when the former location of an object was known.
Attempted fast locates	Number of attempted activations with known slot. This is the number of object activations when the former location of an object was known. It is highly probable that an object can be found in the same place in the buffer pool when it is reactivated.
Successful fast locates	Number of successful fast locates.
Percent	Percentage of successful fast locates.

Object Loading Statistics

Stored objects	The number of objects stored in the buffer pool. This is the number of objects that were stored into the buffer pool and which were not loaded from the system file.
Loaded objects	The number of objects loaded from the system file.
Activated objects	The number of objects activated from the bufferpool.
Aborted loads	The number of load operations that were aborted due to memory shortages within the buffer pool, or due to other events.

General Loading Statistics

Dormant Objects purged	The number of unused objects deleted from the buffer pool to make room for newly loaded ones.
Peak parallel activations	The maximum number of parallel activations of one of the objects in the buffer pool.
Object reusage factor	Average number of times an object was reactivated. This number is the ratio of the number of object activations, and the number of objects loaded into the buffer pool.

WHO

WHO

The WHO command is used to display a list of all users who are using the buffer pool.

The following statistics are displayed:

Statistic	Explanation
index	A number that the NATBPMON utility automatically assigns to each buffer pool user.
tid	The user ID, terminal ID and process ID of a process using the buffer pool.

ZERO

ZERO

The ZERO command is used to reset to "0" all counters that are displayed by the STATUS command.

Buffer-Pool Trouble Shooting

This section describes problems that may occur when using the Natural buffer pool and how to solve them.

It is assumed that UNIX users are familiar with the UNIX commands "ipcs" and "adb".

The following are some typical command output examples, with an explanation of what went wrong during their execution. Problem 1 (Natural or the NATBPMON utility cannot be started) and its solution is described for both an OpenVMS and a UNIX environment. Problem 2 and its solution are described only for a UNIX environment.

Problem 1 (under OpenVMS)

Either Natural or the NATBPMON utility cannot be started.

Examples

\$ NATURAL

```
Unable to open Buffer Pool,
Buffer Pool error: "unexpected system call error occurred" (20)
Global shared memory could not be attached.: NATBP5110X_ALFCOM

%SYSTEM-W-NOSUCHSEC, no such (global) section
```

\$ NAT51BPMON

```
Buffer Pool version 1.3(435) of 25-OCT-2001 11:10:52.19
NATBPMON>status
%SYSTEM-W-NOSUCHSEC, no such (global) section
Buffer Pool error: "unexpected system call error occurred" (20)
Global shared memory could not be attached.: NATBP5110X_ALFCOM
NATBPMON>
```

These two examples describe one of the most typical problems you are likely to encounter as a Natural administrator or user. These problems occur when you start Natural or the NATBPMON utility, and the buffer pool is not active.

Solution

Start the buffer pool as described in the example below.

Example:

```
$ @STARTUP_NAT5.COM STARTUP BP
```

Problem 1 (under UNIX)

Either Natural or the NATBPMON utility cannot be started.

Examples

```
hpn::sag 49 => natural bp = sag
```

```
NATURAL V 5.1.1  DEVELOP SAG 2001

NATURAL Startup Error:    16
Unable to open Buffer Pool,
Buffer Pool error:    "unexpected system call error occurred " (20)
Global shared memory could not be attached.:  shmkey = 11111111

Operating System Error 2 - No such file or directory
```

hpn::sag 50 => natbpmn bp = sag

```
Buffer Pool version 1.1(198) of 25-OCT-2001 11:10:57.44
NATBPMON>who
Buffer Pool error:    unexpected system call error occurred (20)
Global shared memory could not be attached.:  shmkey = 11111111
Operating System Error 2 - No such file or directory
NATBPMON>
```

These two examples describe one of the most typical problems you are likely to encounter as a Natural administrator or user. These problems occur when you start Natural or the NATBPMON utility, and the buffer pool is not active.

Solution

Start the buffer pool as described in the section Activating the Natural Buffer Pool.

Use the "ipcs" command to verify the existence of the necessary semaphores and the shared memory:

hpn::sag 51 => ipcs -m -s

```
IPC status from /dev/kmem as of Thu 25-OCT-2001 12:03:24.30
T      ID      KEY          MODE          OWNER      GROUP
Shared Memory:
m      807 0x4e425031 --rw-rw----      sag  natural
Semaphores:
s      85 0x4e425031 --ra-ra----      sag  natural
```

Note: The above output was edited to exclude memory segments and semaphores that do not belong to the Natural buffer pool.

If you cannot find a shared memory segment, or a set of semaphores with the key you assigned them, the buffer pool was not started.

Problem 2 (under UNIX only)

The Natural buffer pool and a Natural utility are not of the same Natural version. If a utility tries to use the buffer pool, the utility and buffer pool versions are checked for equality. If they differ, the access is denied and an error message is output.

Examples

hpn::sag 54 => natural

```
Unable to open buffer pool,
contact your system administrator
bp_error: 25, version mismatch of buffer pool
```

hpn::sag 55 => natbpmon

```
Buffer Pool version 1.1(198) of 25-OCT-2001 12:05:12.06
```

NATBPMON>dir

```
bp_init: res = -1, bp_errno = 25, errno = 0
buffer pool error message: "version mismatch of buffer pool"
```

Solution

Verify that all utility programs used with the buffer pool are of the same Natural version. To ascertain the versions, use the following commands:

```
hpn::sag 75 = > adb $NATDIR/$NATVERS/bin/natural
bp_majrel?D
bp_majrel:
bp_majrel: 1
bp_minrel?D
bp_minrel:
bp_minrel: 1
bp_version?D
bp_version:
bp_version: 237
@bp_date?s
$global$+1718: Thu Oct 25 12:07:58.01 MET 2001
Ctrl D
```

The sample output above identifies a buffer pool for Version 1.1, which has the sequence version 237. This buffer pool was generated on Thursday, October 25, 2001 at 12:07:58 MET.

The above commands also work with the images NATBPMON and NATBPSRV.

If all programs are of the same Natural version, but contain different buffer pool versions, contact Software AG Support.

Shutting Down and Restarting the Buffer Pool

Usually it should not be necessary to shut down and restart the buffer pool. This may only be necessary if the buffer pool should become unusable due to serious internal errors in the buffer pool, which is extremely unlikely to occur.

If the NATBPMON utility is still able to access the buffer pool, shut down the buffer pool with the SHUTDOWN command.

Once the SHUTDOWN command is executed, new users are denied access to the buffer pool.

After the last user has stopped accessing the buffer pool, buffer pool resources can be deleted by issuing the IPCRM command.

Active buffer pool users can be monitored by issuing the WHO and STATUS commands.

To restart the buffer pool, you call "startup_nat4.com" (under OpenVMS) or "natstart.bsh" (under UNIX) from a sufficiently privileged account.

If the NATBPMON utility is not able to perform a clean shutdown of the buffer pool, the buffer pool must be deleted by using operating system commands. On OpenVMS, it is only possible to rename the sectionfile, because a sectionfile cannot be deleted as long as it is locked by another user. Then restart the buffer pool using "Startup_Nat5.com" and delete the renamed sectionfile after the next reboot.

Note: The rest of this section applies to UNIX environments only.

Use the "ipcs" command to find out the status of the buffer pool's shared memory and semaphores. In the column NATTCH of the output of an "ipcs -m -a" command, you can see the number of processes currently attached to a shared memory segment:

hpn::sag 42 => ipcs -a -m

```
IPC status from /dev/kmem as of Thu Oct 25 12:15:38.39 2001
T   ID          KEY      ... OWNER      GROUP    ... NATTCH  SEGSZ
Shared Memory:
m   707  0x4e425031  ...    sag    natural  ...     7    153600
```

It is highly probable that the number of processes attached to shared memory incorporates a Natural nucleus or the NATBPMON utility currently running. Inform the users who run these processes and ask them to terminate them. Or you can terminate them yourself by using the UNIX command "kill", once you have found out their process IDs using the "ps" command.

Once you are sure that no one is using the buffer pool for important work, its resources can be deleted by using the "ipcrm" command:

hpn::sag 43 => ipcrm -M 0x4e425031 -S 0x4e425031

The values specified for the "-M" and "-S" options must be those that were specified inside the parameter module used to start the buffer pool.

Be careful when you delete shared memory and semaphores using the "ipcrm" command. If you accidentally delete the wrong resource, this might have a serious impact on other software products running on your computer.

The result of deletion can be verified by using the "ipcs" command again.

If there are still some memory segments or message queues displayed, they could belong to other software, or they are marked for deletion because some other process is still attached to them.

If the buffer pool cannot be started after removing the shared memory and semaphores, you should consider either rebooting your computer or contacting Software AG Support.

Dispatch Vector Addresses (OpenVMS/VAX only)

Adabas and Natural define dispatch vectors for private system services. These vectors use the change mode codes from -900 to -1024. You should therefore not use these change mode codes for your own purposes.

Support of Different Character Sets

- Character Sets that are Supported
 - Internal Character Set
 - External Character Sets
 - Character-Set Conversion
- Configuration File NATCONV.INI

This section describes how Natural supports different character sets.

The support of multiple languages with different character sets represents Natural's first step towards internationalization. It can help you when using:

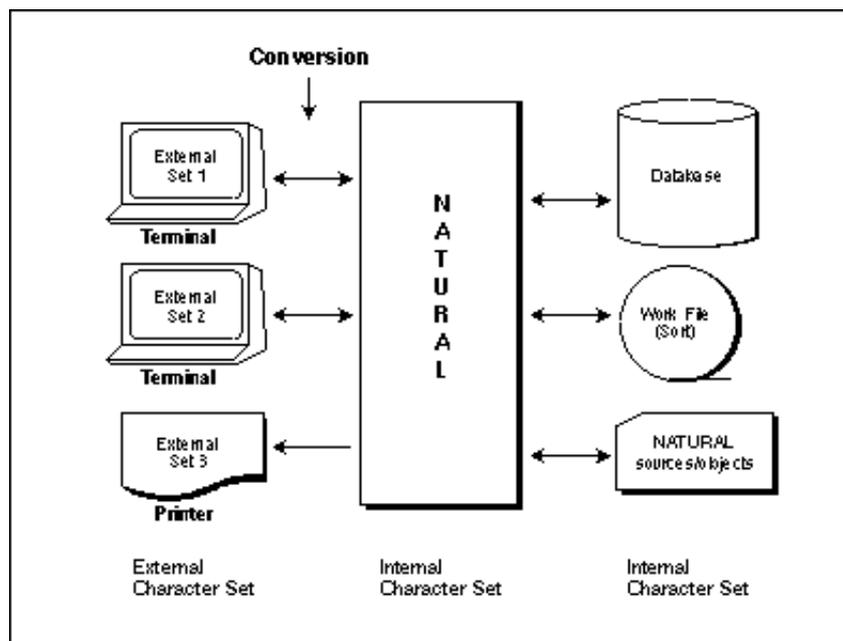
- terminals and printers with different character sets, all communicating with the same Natural environment;
- several Natural environments sharing one database and located on different platforms;
- language-specific characters in Natural identifiers, object names and library names;
- upper-/lower-case translation of language-specific characters;
- language-specific characters in an operand compared with a mask definition.

Character Sets that are Supported

Natural supports any single-byte character set that conforms to the ASCII character set in the lowest seven bits.

Natural distinguishes between an internal character and several external character sets; the internal character set is used by Natural itself.

As illustrated below, conversion between the internal and an external character set is performed after the input from a terminal and before the output to a terminal or printer. There is no conversion to an external character set available for work file I/Os, database I/Os and reading/writing of Natural objects.



Internal Character Set

By default, Natural uses the internal character set "ISO8859_1". If the default character set does not meet your requirements, you can choose either one of the predefined character sets provided by Natural or any other standard character set.

Note: Problems may occur if you run computers with different internal character sets sharing the same database or if you try to exchange data or programming objects between such computers.

External Character Sets

You are able to define an external character set for any terminal and printer.

For a terminal, the name of its character set is defined by the TCS entry in the terminal database, for example: ":TCS = usascii:".

You can also use the OpenVMS logical NATTCHARSET/the UNIX environment variable \$NATTCHARSET which overrides all TCS settings.

If neither a TCS entry nor the logical NATTCHARSET/the environment variable \$NATTCHARSET is defined, no conversion is performed during terminal I/O.

The character set name for a printer can be defined in the printer profile.

Character-Set Conversion

Natural performs the conversion between the internal character set and an external character set by using tables. For the most common character sets, these tables are predefined. You can modify existing tables and also define new ones.

The internal character set is used in the following situations:

- when checking identifiers,
- when translating from upper to lower case or vice versa,
- when classifying characters.

Identifier Checking

Natural checks identifiers (that is, user-defined variables in source programs), names of Natural objects and names of Natural libraries by using check tables. By modifying these tables, you can, for example, allow language-specific characters. In addition, you can redefine the characters "+", "#", and "&", which have a special meaning when used as the first character in variable names.

Upper-/Lower-Case Conversion

Natural performs upper-/lower-case conversion if you use one of the following:

- the %U terminal command,
- the AD = T field attribute,
- the EXAMINE TRANSLATE statement.

By modifying the translation tables, you can support a specific internal character set.

Configuration File NATCONV.INI

All check, translation and classification tables used by Natural to support a user-specific character set reside in the configuration file NATCONV.INI.

You can modify NATCONV.INI to support local or application-specific character sets.

In a standard application, NATCONV.INI need not and should not be modified, because this could lead to serious inconsistencies, in particular if Natural objects and database data are already present.

Modifications are necessary if you want to do any of the following:

- use an internal character set other than the default one,
- use a terminal or printer whose character set is still not supported by NATCONV.INI,
- allow or disallow the use of certain characters in identifiers,
- support local characters when evaluating the MASK option.

Any modifications of NATCONV.INI should be well considered and carefully performed, because otherwise, problems might occur that are difficult to locate.

NATCONV.INI is subdivided in sections and subsections. It contains five defined sections with the following names:

- CHARACTERSET-DEFINITION
- CHARACTERSET-TRANSLATION
- CASE-TRANSLATION
- IDENTIFIER-VALIDATION
- CHARACTER-CLASSIFICATION

CHARACTERSET-DEFINITION

This section defines the name of the internal character set. The default is ISO8859_1. If you choose a different character set, subsections for this character set must be contained in the following sections.

CHARACTERSET-TRANSLATION

This section contains the tables required for the conversion between the internal character set and external character sets. If you use, for example, a terminal with an entry in SAGtermcap of ":TCS = ASCII_GERMAN:" and if ISO8859_1 is used as internal character set, the following two subsections must be contained in this section:

- [ISO8859_1->ASCII_GERMAN]
- [ASCII_GERMAN->ISO8859_1]

CASE-TRANSLATION

This section contains the tables required for the conversion from upper to lower case. This conversion is done within the internal character set. If, for example, the internal character set is "ISO8859_5", the following two subsections must be contained in this section:

- [ISO8859_5->UPPER]
- [ISO8859_5->LOWER]

IDENTIFIER-VALIDATION

This section contains the tables required for the validation of identifiers, object names and library names. It contains a subsection for each defined internal character set. The special characters "#" (for non-database variables), "+" (for application-independent variables) and "&" (for dynamic source generation) can be redefined in this section. In addition, the set of valid first and subsequent characters for identifiers, object names and library names can be modified.

CHARACTER-CLASSIFICATION

This section contains the tables required for the classification of characters, which, for example, are used when evaluating the MASK option. It contains a subsection for each defined internal character set.

The section CHARACTERSET-DEFINITION as well as each subsection contain lines which describe how characters are to be converted and which characters are related with which attributes.

These lines are represented as follows:

```

line      ::= key = value
key       ::= name_key | range_key
name_key  ::= keyword{ CHARS }
keyword   ::= INTERNAL-CHARACTERSET | NON-DB-VARI | DYNAMIC-SOURCE |
            GLOBAL-VARI | FIRST-CHAR | SUBSEQUENT-CHAR |
            LIB-FIRST-CHAR | LIB-SUBSEQUENT-CHAR | ALTERNATE-CARET
            ISASCII | ISALPHA | ISALNUM | ISDIGIT | ISXDIGIT |
            ISLOWER | ISUPPER | ISCNTRL | ISPRINT | ISPUNCT |
            ISGRAPH | ISSPACE
range_key ::= hexnum |
            hexnum-hexnum
value     ::= val {, val }
val       ::= hexnum |
            hexnum-hexnum
hexnum    ::= xhexdighexdigit |
            Xhexdighexdigit
    
```

Notes:

If "range_key" variable is specified on the left-hand side, the number of values specified on the right-hand side must correspond to the number of values specified in the key range, unless only one value is specified on the right-hand side, which is then assigned to each element of the key range.

When the "name_key" variable is specified on the left-hand side and the corresponding list of character codes does not fit in one line, it can be continued on the next line by specifying "name_key = " again. You must not start the lines with leading blanks or tabulators.

Examples of Valid Lines:

x00-x1f = x00	All characters between "x00" and "x1f" are converted to "x00".
x00-x7f = x00-x7f	All characters between "x00" and "x7f" are not converted.
x00-x08 = x00,x01-x07,x00	The characters "x00" and "x08" are converted to "x00" and characters between "x01" and "x07" are not converted.
ISALPHA = x41-x5a,x61-x7a,xc0-xd6,xd8 ISALPHA = xd9-xf6,xf8-xff	The attribute ISALPHA is assigned to all characters specified in these two lines.

Examples of Invalid Lines:

x41 = 'A'	All characters must be specified in hexadecimal format.
0x00-0x1f = 0x00	Hexadecimal values have to be specified in either of the following ways: <i>xdigitdigit</i> <i>Xdigitdigit</i>
x00-x0f = x00,x01	The number of specified values does not correspond to the number of elements in the key range.

Sample NATCONV.INI File:

```

*****
[CHARACTERSET-DEFINITION]

# defining the internal character set for NATURAL

INTERNAL-CHARACTERSET = ISO8859_1

[CHARACTERSET-DEFINITION-END]
*****

*****
[CHARACTERSET-TRANSLATION]

# translation tables between internal and external character set

#-----
[ISO8859_1->USASCII]
#translate ISO8859_1 to USASCII code

x00-x7F = x00-x7F
x80-xBF = x3F
xC0-xCF = x41,x41,x41,x41,x41,x41,x41,x43,x45,x45,x45,x45,x49,x49,x49,x49
xD0-xDF = x44,x4E,x4F,x4F,x4F,x4F,x4F,x3F,x4F,x55,x55,x55,x55,x59,x50,x73
xE0-xEF = x61,x61,x61,x61,x61,x61,x61,x63,x65,x65,x65,x65,x69,x69,x69,x69
xF0-xFF = x64,x6E,x6F,x6F,x6F,x6F,x6F,x3F,x6F,x75,x75,x75,x75,x79,x70,x59

[ISO8859_1->USASCII-END]
#-----

#-----
[USASCII->ISO8859_1]
#translate USASCII to ISO8859_1 code

x00-xFF = x00-xFF

[USASCII->ISO8859_1-END]
#-----

[CHARACTERSET-TRANSLATION-END]
*****

```

```

*****
[CASE-TRANSLATION]

# translation tables for lower/uppercase conversion of the internal
characterset
#-----
[ISO8859_1->LOWER]
#translate ISO8859_1 to lowercase

x00-x40 = x00-x40
x41-x5A = x61-x7A
x5B-xBF = x5B-xBF
xC0-xDE = xE0-xF6,xD7,xF8-xFE
xDF-xFF = xDF-xFF

[ISO8859_1->LOWER-END]
#-----

#-----
[ISO8859_1->UPPER]
#translate ISO8859_1 to uppercase

x00-x60 = x00-x60
x61-x7A = x41-x5A
x7B-xDF = x7B-xDF
xE0-xFF = xC0-xD6,xF7,xD8-xDE,xFF

[ISO8859_1->UPPER-END]
#-----

[CASE-TRANSLATION-END]
*****

```

```

#####
[IDENTIFIER-VALIDATION]

# tables for validation of identifiers

#-----
[ISO8859_1]

# special characters
#
# non DB variable
#      '#'
NON-DB-VARI    = x23

# dynamic source generation
#      '&'
DYNAMIC-SOURCE = x26

# global variable
#      '+'
GLOBAL-VARI    = x2B

# valid first characters
#      '#' '&' '+' 'A'-'Z' 'a'-'z'
FIRST-CHAR     = x23,x26,x2B,x41-x5A,x61-x7A

# valid subsequent characters

#      '#' '$' '&' '-' '/' '0'-'9'
SUBSEQUENT-CHAR = x23,x24,x26,x2D,x2F,x30-x39

#      '@' 'A'-'Z' '_' 'a'-'z'
SUBSEQUENT-CHAR-1 = x40,x41-x5A,x5F,x61-x7A

# valid first characters for library names
#      'A'-'Z' 'a'-'z'
LIB-FIRST-CHAR  = x41-x5A,x61-x7A

# valid subsequent characters for library names
#      '-' '0'-'9' 'A'-'Z' '_' 'a'-'z'
LIB-SUBSEQUENT-CHAR = x2D,x30-x39,x41-x5A,x5F,x61-x7A

[ISO8859_1-END]
#-----

[IDENTIFIER-VALIDATION-END]
#####

```



```
# lowercase characters
#      'a'-'z' language specific lowercase characters
#      vvvvvvvvvvvvvvvvv
ISLOWER = x61-x7A,xDF-xF6,xF8-xFF

# uppercase characters
#      'A'-'Z' language specific uppercase characters
#      vvvvvvvvvvvvvvvvv
ISUPPER = x41-x5A,xC0-xD6,xD8-xDE

# control characters
#
ISCNTRL = x00-x1F,x7F-x9F

# printable characters
#
ISPRINT = x20-x7E,xA0-xFF

# special characters
#
ISPUNCT = x21-x2F,x3A-x40,x5B-x60,x7B-x7E,xA1-xBF,xD7,xF7

# graphical characters
#
ISGRAPH = x21-x7E,xA1-xFF

# spacing characters
#
ISSPACE = x09-x0D,x20

[ISO8859_1-END]
#-----

[CHARACTER-CLASSIFICATION-END]
#*****
```

NATTERMCAPI Utility

- General Settings
- Invoking NATTERMCAPI
- File
- Edit
- Search
- Test
- Options
- Help
- Terminal Capabilities - Overview
- Terminal Capabilities - Sorted by Name

The NATTERMCAPI utility is used to create, modify and test terminal capabilities described in the terminal database SAGtermcap.

As SAGtermcap contains various terminal types and there is no standard terminal type definition, Software AG does not assume any responsibility for the completeness and the correctness of these terminal types. A terminal type standard used for Digital Equipment Corporation's VT terminals is ANSI X3.64 (corresponds to ISO 6429).

General Settings

Environment Variables

For the NATTERMCAPI utility, you can set the following environment variables:

TERM	System environment variable used for the currently active terminal type.
NATTERM	Natural environment variable used for the Natural terminal type. If this environment variable is set, it overwrites the setting of TERM.
NATTCAP	Natural environment variable used to specify a different terminal database than SAGtermcap.
COLUMNS	Used for terminal-screen width. If this environment variable is not set, the current screen width is used. If the capability "co" is defined (means: has a value not equal to 0), this value is taken instead of the environment variable COLUMNS.
LINES	Used for terminal screen page size. If this environment variable is not set, the current screen page size is used. If the capability "li" is defined (means: has a value not equal to 0), this value is taken instead of the environment variable LINES.

Terminal Mode

NATTERMCAPI is a screen I/O application. Therefore, it needs a terminal database and a terminal type to display menus and dialogs on the screen. By default, it uses the same terminal database and type as Natural does.

Natural retrieves the terminal type as follows:

1. It takes the contents of the environment variable NATTERM.
2. If NATTERM is not set, it takes the contents of the system environment variable TERM.

Natural retrieves the terminal database as follows:

1. It takes the contents of the environment variable NATTCAP.
2. If NATTCAP is not set, it retrieves the database name from the NATTCAP entry in the local configuration file Natural.INI.
3. If this terminal database could not be found, NATTERMCAPI tries to locate a database named "SAGtermcap" in the current directory.

To avoid this automatism, NATTERMCAPI offers a predefined terminal database in the dynamic parameter TERMCAP with a limited number of terminal types. These types can be accessed by specifying the dynamic parameter DISPLAY.

Key Definitions

The NATTERMCAPI utility has the following predefined function keys:

Key	Explanation
CTRL+A	Inserts ANSI definitions.
CTRL+E	Evaluates keys automatically.
CTRL+N	Inserts non-graphic characters for frames.
CTRL+P	Gets help.
CTRL+V	Tests capabilities.

To modify the predefined function keys, choose the Options > Key Assignments. You can only specify control keys (CTRL+A to CTRL+Z).

Dynamic Parameters

The following dynamic parameters can be specified:

Dynamic Parameter	Explanation
DISPLAY	<p>The DISPLAY parameter is used to define the terminal type for the NATTERMCP utility itself. If no DISPLAY parameter is specified, the Natural terminal database SAGtermcap is used and the same terminal type tracking mechanism as for Natural is in effect.</p> <p>Format: DISPLAY = {#vt100 #vt100NG #vt220 #vt220ng wyse60 #tty <other>}</p> <p>Alternatives:</p> <p>#vt100 - Use the terminal entry 'DEC vt100' from the internal database.</p> <p>#vt100ng - Similar to #vt100, but graphic line characters will be replaced by single characters such as '-', ' ' and '+'.</p> <p>#vt220 - Use the terminal entry 'DEC vt220' from the internal database.</p> <p>#vt220ng - Similar to #vt220, but graphic line characters will be replaced by single characters such as '-', ' ' and '+'.</p> <p>#wyse60 - Use the terminal entry 'wyse60' from the internal database.</p> <p>#tty - Use the terminal entry 'tty' from the internal database. The tty terminal works in a line-oriented way without using escape control sequences. Only a few functions are available if this terminal entry is selected.</p> <p><other> - Use any other terminal type in the terminal database, for example 'xterm'.</p>
EDIT	<p>The parameter EDIT is used to view and/or modify a specific terminal capability. If the capability is found, the associated dialog is displayed and the cursor is positioned in the specified field. If the capability can not be found, an error message will be displayed and NATTERMCP terminates.</p> <p>Format: EDIT = capability</p> <p>Capability: Any terminal capability known in Natural can be specified. A list of capabilities can be found in the sections Terminal Capabilities - Overview and Terminal Capabilities - Sorted by Name.</p>
EXIT	<p>This parameter is used to terminate the utility after all parameters have been processed.</p> <p>Example: NATTERM EDIT = PF10 EXIT After modifying the function key PF10, the utility terminates immediately.</p>

Dynamic Parameter	Explanation
HELP	<p>With parameter HELP, you can get help about a specific capability or about using the NATTCAP utility.</p> <p>Format: HELP = {CAP USAGE capability}</p> <p>Alternatives: CAP - Displays help for all capabilities sorted by capability name. USAGE - Displays all dynamic parameters in NATTERMCAPI. capability - Displays help for a specific capability.</p>
REPORT	<p>With this parameter, a text file with a detailed description of the current terminal entry is created by default in the Natural TMP directory.</p> <p>Format: REPORT{= name} If no name is specified, terminal_entry.txt is used.</p>
SAVE	<p>Saves all modifications of the current terminal entry.</p> <p>Format: SAVE{= name}</p>
TERM	<p>This parameter is used to read in a different terminal entry. If this parameter is not specified, the current Natural terminal type is used (NATTERM or TERM).</p> <p>Format: TERM = entry_name where entry_name is any type of given terminal included in the terminal database.</p>
TERMCAP	<p>This parameter is used to work with a different terminal database. If this parameter is not specified, the current Natural terminal database is used (NATTCAP).</p> <p>Format: TERMCAP = database_name where database_name is the database path and file name.</p>
TEST	<p>With the parameter TEST, NATTERMCAPI calls the specified test dialog.</p> <p>Format: TEST = {CONSISTENCY COLORS GRAPHICS KEYS VIDEO}</p> <p>Choices: CONSISTENCY - Checks whether the function keys are uniquely defined. COLORS - All available colors are displayed with sample text. GRAPHICS - A single line and a double line box are displayed. KEYS - A text on any pressed key will be displayed. The dialog can be terminated by pressing one of the letters 'E', 'Q', 'X', or '.'. VIDEO - Displays video attributes such as blinking, underlined and reversed video.</p>

Special Control Codes

The following table gives an overview of unprintable characters, as well as characters that have a special meaning in terminal capability syntax:

Control Code	Explanation
\E	Escape character
\b	Backspace character
\n	New line
\r	Carriage return character
\t	Tab character
\xxx	Octal value of xxx; must be three characters
\072	The character ':'; NATTERMCP uses ':' as internal separator
^x	Control-x, where x is any letter

Example:

If the function key PF10 is to be defined as F10 on a DEC VT220 terminal, the code of F10 is:

```
<ESCAPE>[21~
```

Specify the following for capability PF10:

```
\E[21~
```

If the character "~" (tilde) is not available on the keyboard, use the octal value of tilde instead. Tilde is defined as octal 176. The alternative specification is then:

```
\E[21\176
```

Terminal Copy Capabilities

Terminal copy capabilities (TCs) are capabilities transferred from another terminal entry, like the #include directive of a C program. However, if capabilities are already defined in the current entry, the transferred capabilities are ignored. This makes the entries more efficient, not only by reducing redundancies, but also by ensuring that related entries are kept consistent. Capabilities read from a terminal copy entry are marked "[TC]" to the right of the input field. Additionally, the name of the entry from where this capability is transferred is shown in the top right corner of the menu, above the terminal name.

Once a terminal capability has been modified, it loses the link to the transferred terminal entry and the modification is made to the current terminal entry. To display the current terminal entry without any terminal copy capabilities, select Options > Terminal Capabilities, Hide option.

Example:

Assume TERM is set to "vt100" and the vt100 (vt220) entry in the terminal database looks as follows:

vt100 entry:

```
ti = \E =
```

```
ESC = \E
```

```
ETO = 300
```

```
tc = vt220
```

vt220 entry:

```
ti = \E[0m
```

```
cr = \r
```

The combined terminal entry for the terminal type vt100 would be:

vt100:

ti = \E = /* taken from the original vt100 entry

ESC = \E /* taken from the original vt100 entry

ETO = 300 /* taken from the original vt100 entry

cr = \r /* transferred from vt220 entry

The capability "ti(= \E[0m)" from the vt220 terminal is ignored, because "ti" is already defined in the vt100 entry.

Invoking NATTERMCP

To invoke the NATTERMCP utility, at the operating system prompt enter "NATTERMCP" for UNIX or "nat51termcap" for OpenVMS. A menu will be displayed with the name of the currently active terminal entry in the top right-hand corner of the screen.

The menu provides the following options:

Option	Explanation
File	Creates, reads, saves and deletes a terminal entry.
Edit	Views and sets terminal capabilities.
Search	Searches for a specific capability by name.
Test	Tests capabilities.
Options	Modifies the default key definition and shows or hides terminal copy capabilities. Terminal copy capabilities are capabilities included from another terminal entry.
Help	Provides help on each capability and on the usage of the dynamic parameters.

File

When you select File, a selection list containing the following functions is displayed:

Function	Explanation
New	Creates a new terminal entry in the current terminal database.
Read	Reads a terminal entry from the terminal database.
Save	Saves terminal capabilities to the current terminal entry.
Save As	Saves terminal capabilities to a different or new terminal entry.
Delete	Removes the current terminal entry from the terminal database.
Generate Report	Generates a text file including information about the description, aliases and capabilities of the current terminal entry. The text file will be stored by default in the Natural TMP directory as " <i>terminal-name.txt</i> ", for example "xterm.txt".
Move	Moves the terminal entry physically to the top of the terminal database. If a terminal is on top of the database, the access time during the terminal initialization will be improved.
Import Database	Allows working with a database other than Natural's SAGtermcap.
Export Database	Saves the whole database and all terminal entries with a different path and/or name than Natural's SAGtermcap.
Properties	Displays detailed information about the terminal database, terminal entry, environment variables and display type.
Exit	Exits the NATTERMCAPI utility.

Edit

When you select Edit, you get the following different types of input fields:

- boolean, where only "ON" or "OFF" may be specified;
- numeric, where only digits (0 to 9) may be specified;
- string, where 32 alphanumeric characters may be specified, with the exception of terminal capabilities "te" and "ti", for which 132 characters may be specified;
- description, where 132 alphanumeric characters may be specified.

For further information on the individual Natural capabilities to be edited, see the sections Terminal Capabilities - Overview and Terminal Capabilities - Sorted by Name.

Search

When you select Search, a selection list containing all capabilities sorted by name is displayed. After you have selected a capability, the corresponding dialog within the Edit option is displayed.

Test

When you select Test, a selection list containing the following functions is displayed:

Function	Explanation
Colors	Tests all foreground and background colors.
Consistency	Searches for inconsistent key definitions. A list of affected keys is displayed if they are not unique.
Keys	Displays the name of a pressed key. Leave this functions by pressing one of the following letters : 'E', 'Q', 'X', or '.'.
Line Graphics	Tests the graphic line capabilities used for drawing window frames.
Video Attributes	Tests all video atributes.

Options

When you select Options, a selection list containing the following functions is displayed:

Function	Explanation
Terminal Copy Capabilities	Shows or hides the terminal capabilities included from a different terminal entry specified by the capability "tc".
Key Assignments	Modifies the default key assignments of the utility.

Help

When you select Help, a selection list containing the following functions is displayed:

Function	Explanation
Help Topics	Invokes a detailed help section for a given topic, such as NAME, REPORT, TC or TEST.
Capabilities	Invokes a detailed help section for each capability.
Usage	Displays information on how the dynamic parameters are used.
About	Displays product information.

Terminal Capabilities - Overview

Colors

Name	Description
ct	Color terminal
bgbla	Screen background
fgblu	Foreground color blue; Natural color definition CD = BL
fggre	Foreground color green; Natural color definition CD = GR
fgmag	Foreground color pink; Natural color definition CD = PI
fgred	Foreground color red; Natural color definition CD = RE
fgcya	Foreground color turquoise; Natural color definition CD = TU
fgwhi	Foreground color white; Natural color definition CD = NE
fgyel	Foreground color yellow; Natural color definition CD = YE
bgblu	Background color blue
bggre	Background color green
bgmag	Background color pink
bgred	Background color red
bgcya	Background color turquoise
bgwhi	Background color white
bgyel	Background color yellow
ctres	Reset colors; Natural attribute definition AD = D

Cursor Keys and Modes

Name	Description
kd	Cursor key down (in normal mode)
kl	Cursor key left (in normal mode)
kr	Cursor key right (in normal mode)
ku	Cursor key up (in normal mode)
DK	Cursor key down (in application mode)
LK	Cursor key left (in application mode)
RK	Cursor key right (in application mode)
UK	Cursor key up (in application mode)
@7	Cursor key end
kh	Cursor key home
CKNO	Normal cursor key mode
CKAP	Application cursor key mode
cm	Cursor motion
CNL	Cursor next line
ve	Cursor visible
vi	Cursor invisible

Description and Comments

These fields can be used to describe the terminal entry or to add some comments.

Editing Keys

Name	Description
BC	Backspace key
BCVT	Alternative backspace key
BT	Backtab key
CR	Carriage return key
DC	Delete character key
ESC	Escape key
ETO	Escape timeout value in milliseconds
KDEL	Delete to end of field key
KL	Insert or overstrike mode key
KN	Page down (next) key
KP	Page up (previous) key
NLFF	Next line first field key
PD	Alternative page down (next) key
PU	Alternative page up (previous) key
TA	Tab key
%1	Help key
&2	Refresh key

Initialization and Reset

Name	Description
TE	Additional sequence after termination
TECL	Clear screen after termination
TERA	Reset video attributes after termination
TENL	Cursor next to line after termination
TECV	Cursor visible after termination
TECI	Cursor invisible after termination
TEAK	Application keypad after termination
TENK	Numeric keypad after termination
TEAC	Application cursor key mode after termination
TENC	Normal cursor key mode after termination
TEDB	Dark background after termination
TI	Additional initialization sequence
TICL	Clear screen after initialization
TIRA	Reset attributes after initialization
TIGR	Enable line graphics after initialization
TICI	Cursor invisible after initialization
TICV	Cursor visible after initialization
TIAK	Application keypad after initialization
TINK	Numeric keypad after initialization
TIAC	Application cursor key mode after initialization
TINC	Normal cursor key mode after initialization
TIDB	Dark background after initialization
TILB	Light background after initialization
TIIM	Insert mode after initialization
TIOM	Overstrike mode after initialization

Keypad Keys for Mathematical Operations

Keypad	Description
KP01	Single Null
KP1	One
KP2	Two
KP3	Three
KP4	Four
KP5	Five
KP6	Six
KP7	Seven
KP8	Eight
KP9	Nine
KPADD	Add
KPSUB	Subtract
KPDIV	Divide
KPMUL	Multiply
KPTS	Thousand Separator
KPDP	Decimal Point
KPRES	Result

Line Graphics

Name	Description
eA	Enable line graphics
as	Graphics mode on
ae	Graphics mode off
G1	Single upper right corner character
G2	Single upper left corner character
G3	Single lower left corner character
G4	Single lower right corner character
GH	Single horizontal bar character
GV	Single vertical bar character
GD1	Double upper right corner character
GD2	Double upper left corner character
GD3	Double lower left corner character
GD4	Double lower right corner character
GDH	Double horizontal bar character
GDV	Double vertical bar character

Miscellaneous

Name	Description
bl	Audio bell
vb	Visual bell
cl	Clear screen
ce	Clear to end of line
ks	Keypad mode application
ke	Keypad mode numeric
xi	Scroll glitch
TCS	External terminal/printer character set, for more information, see the section Support of Different Character Sets , section External Character Sets.
tc	Terminal copy

Name and Aliases

A name and up to 30 aliases can be defined for each entry.

PA and PF Keys

Key	Description
PA1	Attention key PA1
PA2	Attention key PA2
PA3	Attention key PA3
PF1	Function key PF1
PF2	Function key PF2
...	
PF47	Function key PF47
PF48	Function key PF48

Right-To-Left Support

Name	Description
RTLFL	Right-to-left language toggle key for fields
RTLSS	Right-to-left screen toggle key

Screen Dimension and Appearance

Name	Description
li	Number of screen rows (if not specified, take the current screen size)
co	Number of screen columns (if not specified, take the current screen size)
DAR	Dark background, light text
LIG	Light background, dark text

Video Attributes

Name	Description
mb	Blinking on; Natural attribute definition AD = B
adc	Cursive/italics on; Natural attribute definition AD = C
md	Intensified (bold) on; Natural attribute definition AD = I
mr	Reversed on; Natural attribute definition AD = V
us	Underline on; Natural attribute definition AD = U
me	Reset attributes; Natural attribute definition AD = D
so	Standout mode on
xs	Standout glitch
BR	Blinking off
adc0	Cursive/italics off
HR	Intensified off
mr0	Reversed off
ue	Underlined off
se	Standout mode off

Terminal Capabilities - Sorted by Name

Name	Description
%1	Help key
&2	Refresh key
@7	Cursor key end
adc	Cursive/italic on; Natural attribute definition AD = C
adc0	Cursive/italic off
ae	Graphics mode off
as	Graphics mode on
bc	Backspace key
bcvt	Alternative backspace key
bgbla	Screen background
bgblu	Background color blue
bgcya	Background color turquoise
bggre	Background color green
bgmag	Background color pink
bgred	Background color red
bgwhi	Background color white
bgyel	Background color yellow
bl	Audio bell
BR	Blinking off
bt	Backtab key
ce	Clear to end of line
CKAP	Application cursor key mode
CKNO	Normal cursor key mode
cl	Clear screen
cm	Cursor motion
CNL	Cursor next line
co	Number of screen columns; if not specified, take the current screen size
cr	Carriage return key
ct	Color terminal
ctres	Reset colors; Natural attribute definition AD = D
DAR	Dark background, light text
dc	Delete character key
DK	Cursor key down (in application mode)

Name	Description
eA	Enable line graphics
ESC	Escape key
ETO	Escape timeout value
fgblu	Foreground color blue; Natural color definition CD = BL
fgcya	Foreground color turquoise; Natural color definition CD = TU
fggre	Foreground color green; Natural color definition CD = GR
fgmag	Foreground color pink; Natural color definition CD = PI
fgred	Foreground color red; Natural color definition CD = RE
fgwhi	Foreground color white; Natural color definition CD = NE
fgyel	Foreground color yellow; Natural color definition CD = YE
G1	Single upper right corner character
G2	Single upper left corner character
G3	Single lower left corner character
G4	Single lower right corner character
GD1	Double upper right corner character
GD2	Double upper left corner character
GD3	Double lower left corner character
GD4	Double lower right corner character
GDH	Double horizontal bar character
GDV	Double vertical bar character
GH	Single horizontal bar character
GV	Single vertical bar character
HR	Intensified off
kd	Cursor key down (in normal mode)
kE	Delete to end of field key
ke	Keypad mode numeric
kh	Cursor key home
kI	Insert or overstrike mode key
kl	Cursor key left (in normal mode)
kN	Page down (next) key
kP	Page up (previous) key
KP01	Single Null
KP1	One
KP2	Two
KP3	Three

Name	Description
KP4	Four
KP5	Five
KP6	Six
KP7	Seven
KP8	Eight
KP9	Nine
KPADD	Add
KPSUB	Subtract
KPDIV	Divide
KPMUL	Multiply
KPTS	Thousand Separator
KPDP	Decimal Point
KPRES	Result
kr	Cursor key right (in normal mode)
ks	Keypad mode application
ku	Cursor key up (in normal mode)
li	Number of screen rows; if not specified, take the current screen size
LIG	Light background, dark text
LK	Cursor key left (in application mode)
mb	Blinking on; Natural attribute definition AD = B
md	Intensified (bold) on; Natural attribute definition AD = I
me	Reset attributes; Natural attribute definition AD = D
mr	Reversed on; Natural attribute definition AD = V
mr0	Reversed off
NLFF	Next line first field key
PA1	Attention key PA1
PA2	Attention key PA2
PA3	Attention key PA3
PD	Alternative page down (next) key
PF1	Function key PF1
PF2	Function key PF2
...	
PF47	Function key PF47
PF48	Function key PF48
PU	Alternative page up (previous) key

Name	Description
RK	Cursor key right (in application mode)
RTLFL	Right-to-left language toggle key for fields
RTLS	Right-to-left screen toggle key
se	Standout mode off
so	Standout mode on
ta	Tab key
tc	Terminal copy
TCS	External terminal/printer character set.
te	Additional sequence after termination
TEAC	Application cursor key mode after termination
TEAK	Application keypad after termination
TECI	Cursor invisible after termination
TECL	Clear screen after termination
TECV	Cursor visible after termination
TEDB	Dark background after termination
TELB	Light background after termination
TENC	Normal cursor key mode after termination
TENK	Numeric keypad after termination
TENL	Cursor next to line after termination
TERA	Reset video attributes after termination
ti	Additional initialization sequence
TIAC	Application cursor key mode after initialization
TIAK	Application keypad after initialization
TICI	Cursor invisible after initialization
TICL	Clear screen after initialization
TICV	Cursor visible after initialization
TIDB	Dark background after initialization
TIGR	Enable line graphics after initialization
TIIM	Insert mode after initialization
TILB	Light background after initialization
TINC	Normal cursor key mode after initialization
TINK	Numeric keypad after initialization
TIOM	Overstrike mode after initialization
TIRA	Reset attributes after initialization
ue	Underlined off

Name	Description
UK	Cursor key up (in application mode)
us	Underline on; Natural attribute definition AD = U
vb	Visual bell
ve	Cursor visible
vi	Cursor invisible
xi	Scroll glitch
xs	Standout glitch