Natural

System Variables

Version 9.3.3

October 2026

ADABAS & NATURAL

This document applies to Natural Version 9.3.3 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 1992-2026 Software GmbH, Darmstadt, Germany and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software GmbH product names are either trademarks or registered trademarks of Software GmbH and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software GmbH and/or its subsidiaries is located at https://softwareag.com/licenses.

Use of this software is subject to adherence to Software GmbH's licensing conditions and terms. These terms are part of the product documentation, located at https://softwareag.com/licenses and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software GmbH Products / Copyright and Trademark Notices of Software GmbH Products". These documents are part of the product documentation, located at https://softwareag.com/licenses and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software GmbH.

Document ID: NATWIN-NNATVARI-933-20261009

Table of Contents

Preface vii
1 About this Documentation 1
Document Conventions 2
Online Information and Support 2
Data Protection 3
2 About Natural System Variables 5
3 Application-Related System Variables 7
*APPLIC-ID 9
*APPLIC-NAME 9
*COM 9
*CONTROL 10
*CONVID 10
*COUNTER (r) 11
*CPU-TIME 11
*CURRENT-UNIT 11
*DATA 12
*DIALOG-ID 12
*EDITOR 12
*ERROR-LINE 13
*ERROR-NR 13
*ERROR-TA 14
*ETID 14
*EVENT 14
*ISN (r) 15
*LBOUND 15
*LENGTH (field) 17
*LEVEL 17
*LIBRARY-ID 18
*LINE 18
*LINEX 18
*LOAD-LIBRARY-ID 19
*NUMBER (r) 19
*OCCURRENCE 20
*PAGE-EVENT 22
*PAGE-LEVEL 23
*PROGRAM 23
*REINPUT-TYPE 23
*ROWCOUNT 24
*STARTUP 24
*STEPLIB 26
*SUBROUTINE 26
*THIS-OBJECT 27
*TYPE 27

*UBOUND 28
4 Date and Time System Variables 31
Usage 32
DAT - Date System Variables 32
TIM - Time System Variables 33
Example of Date and Time System Variables 34
5 Input/Output-Related System Variables 37
*CURS-COL 38
*CURS-FIELD 38
*CURS-LINE 39
*CURSOR 40
*LINE-COUNT 40
*LINESIZE 41
*LOG-LS 41
*LOG-PS 41
*PAGE-NUMBER 41
*PAGESIZE 42
*PF-KEY 42
*PF-NAME 43
*WINDOW-LS 43
*WINDOW-POS 44
*WINDOW-PS 44
6 JSON-Related System Variables 45
*PARSE-LEVEL (r) for JSON 46
*PARSE-INDEX (r) for JSON 46
*PARSE-TYPE (r) for JSON 47
7 Natural Environment-Related System Variables 49
*BROWSER-IO 50
*DEVICE 50
*GROUP 51
*HARDCOPY 51
*INIT-USER 51
*LANGUAGE 52
*NATVERS 55
*NET-USER 56
*PARM-USER 56
*PATCH-LEVEL 56
*PID 57
*SCREEN-IO 57
*SERVER-TYPE 58
*UI 58
*USER 59
*USER-NAME 59
*UUID 59
8 System Environment-Related System Variables 61

*CODEPAGE 62
*HARDWARE 62
*HOSTNAME 62
*INIT-ID 63
*INIT-PROGRAM 63
*LOCALE 63
*MACHINE-CLASS 64
*OPSYS 64
*OS 65
*OSVERS 65
*TP 65
*TPSYS 65
*TPVERS 66
*WINMGR 66
*WINMGRVERS 67
9 XML-Related System Variables 69
*PARSE-COL (r) for XML 70
*PARSE-LEVEL (r) for XML 70
*PARSE-NAMESPACE-URI (r) for XML 70
*PARSE-ROW (r) for XML 71
*PARSE-TYPE (r) for XML 71

Preface

This documentation describes the Natural system variables.

The documentation for the system variables is grouped by functions:

Application-Related System Variables	System variables, which are useful in conjunction with a Natural application: name of the library to which the user is logged on, current library ID, information required in the event of an error, type or name of the Natural object which is currently executed, etc.
Date and Time System Variables	Date and time system variables that may be specified in the statements `COMPUTE`, `DISPLAY`, `MOVE`, `PRINT`, `WRITE` and in logical condition criteria.
Input/Output-Related System Variables	System variables, which contain input or output related information, such as current cursor position, line number of the current line within the current page, physical line or page size.
JSON-Related System Variables	System variables, which are available in conjunction with the `PARSE JSON` statement.
Natural Environment-Related System Variables	System variables which are relevant in conjunction with the Natural environment: device type/mode from which Natural has been invoked, user ID of the user, user ID as taken from the Natural Security logon, language indicator (language code), Natural version, etc.
System Environment-Related System Variables	System variables relating to the operating system used: name of the hardware platform, machine or machine class on which Natural is running, name or version number of the operating system, name or version of the TP subsystem under which Natural is running, name or version of the window manager being used, etc.
XML-Related System Variables	System variables, which are available in conjunction with the `PARSE XML` statement.
System Variables in Alphabetical Order	Descriptions of all system variables in alphabetical order.

1 About this Documentation

- Document Conventions 2
- Online Information and Support 2
- Data Protection 3

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at **https://documentation.softwareag.com**.

Product Training

You can find helpful product training material on our Learning Portal at **https://learn.softwareag.com**.

Tech Community

You can collaborate with Software GmbH experts on our Tech Community website at **https://techcommunity.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software GmbH news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://hub.docker.com/publishers/softwareag** and discover additional Software GmbH resources.

Product Support

Support for Software GmbH products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

2 About Natural System Variables

Natural system variables contain information about the current Natural session, such as: the current library, the user and terminal identification; the current status of a loop processing; the current report processing status; the current date and time. They may be referenced at any point within a Natural program.

See also:

- *System Variables* in the *Programming Guide*
- *Example of System Variables and System Functions* in the *Programming Guide*

3 Application-Related System Variables

- *APPLIC-ID 9
- *APPLIC-NAME 9
- *COM 9
- *CONTROL 10
- *CONVID 10
- *COUNTER (r) 11
- *CPU-TIME 11
- *CURRENT-UNIT 11
- *DATA 12
- *DIALOG-ID 12
- *EDITOR 12
- *ERROR-LINE 13
- *ERROR-NR 13
- *ERROR-TA 14
- *ETID 14
- *EVENT 14
- *ISN (r) 15
- *LBOUND 15
- *LENGTH (field) 17
- *LEVEL 17
- *LIBRARY-ID 18
- *LINE 18
- *LINEX 18
- *LOAD-LIBRARY-ID 19
- *NUMBER (r) 19
- *OCCURRENCE 20
- *PAGE-EVENT 22
- *PAGE-LEVEL 23
- *PROGRAM 23
- *REINPUT-TYPE 23
- *ROWCOUNT 24
- *STARTUP 24

- *STEPLIB 26
- *SUBROUTINE 26
- *THIS-OBJECT 27
- *TYPE 27
- *UBOUND 28

*APPLIC-ID

Format/length:	A8
Content modifiable:	No

This system variable contains the ID of the library to which the user is currently logged on.

*APPLIC-NAME

Format/length:	A32
Content modifiable:	No

Under Natural Security

If Natural Security is installed, this system variable contains the name of the library to which the user is logged on. If the user is logged on via a special link, it contains the link name instead. If Natural Security is not installed, this system variable contains the name `SYSTEM`.

The general option `Set *APPLIC-NAME always to library name` can be set so that `*APPLIC-NAME` always contains the library name, regardless of whether the user is logged on via a special link or not. See *Set *APPLIC-NAME always to library name* in the *Natural Security* documentation.

*COM

Format/length:	A128
Content modifiable:	Yes

This system variable designates a communication area which can be used to process data from outside a screen window.

Normally when a window is active, no data can be entered on the screen outside the window. However, if a map contains `*COM` as a modifiable field, that field will still be available for the user to enter data when a window is currently on the screen. Further processing can then be made dependent on the content of `*COM`. This allows you to implement user interfaces where a user can always enter data in the command line, even when a window with its own input fields is active.

Note: Although `*COM` can be used as a modifiable field in an `INPUT` statement, it is *not* treated as an input field, but as a system variable; that is, any input entered into the `*COM` field will

be taken as it is, without any input processing (e.g. conversion to upper case) being performed on it. Once *COM has been displayed on the screen via an INPUT statement, every subsequent INPUT or REINPUT statement will cause the current content of *COM to be displayed.

See also:

Dialog Design in the *Programming Guide*

- *Processing Data Outside an Active Window*
- *Positioning the Cursor to *COM - the %T* Terminal Command*
- *Copying Data from a Screen*

*CONTROL

Format/length:	Handle
Content modifiable:	No

This system variable contains the handle of the dialog element for which the current event has been triggered.

For details on events, see *Event-Driven Programming Techniques* in the *Programming Guide*.

*CONVID

Format/length:	I4
Content modifiable:	Yes

This system variable contains the conversation ID of the current conversational remote procedure call (RPC). This ID is set by an OPEN CONVERSATION statement.

Via an OPEN CONVERSATION statement, a client can get a server for exclusive use to execute a number of services (subprograms) within one server process. This exclusive use is called conversation. The OPEN CONVERSATION statement is used to open a conversation and specify the subprograms to be involved in this conversation. When an OPEN CONVERSATION statement is executed, it assigns a unique ID which identifies the conversation to the system variable *CONVID.

Several conversations can be open at the same time. To switch from one open conversation to another, you assign the corresponding conversation ID to *CONVID.

For further information on Natural RPC, see the *Natural RPC (Remote Procedure Call)* documentation.

*COUNTER (r)

Format/length:	P10
Content modifiable:	Yes

This system variable contains the number of times a processing loop initiated by a FIND, READ, HISTOGRAM or PARSE statement has been entered.

(r) notation after *COUNTER is used to indicate the statement label or source-code line number of the FIND, READ, HISTOGRAM or PARSE statement. If (r) is not specified, *COUNTER represents the number of times the currently active processing loop has been entered.

*COUNTER is not incremented if a record is rejected as a result of the criteria specified in a WHERE clause. *COUNTER is incremented if a record is rejected as a result of an ACCEPT/REJECT statement.

*CPU-TIME

Format/length:	I4
Content modifiable:	No

*CPU-TIME contains the CPU time currently used by the Natural process in units of 10 ms.

*CURRENT-UNIT

Format/length:	A32
Content modifiable:	No

This system variable contains the name of the currently executed unit. This is

- the function name in case of the object type “function”,
- the inline subroutine name if an inline subroutine is performed,
- the external subroutine name in case of the object type “subroutine”, see also *SUBROUTINE,
- the object name in case of all other object types (program, subprogram, map, dialog, etc.); see also *PROGRAM.

The contents of *CURRENT-UNIT will always be in upper case.

*DATA

Format/length:	N3
Content modifiable:	No

This system variable contains the number of data elements in the Natural stack which are available to the next `INPUT` statement as input data. `*DATA` will contain `0` when the stack is empty. A value of `-1` indicates the next element in the stack is a command or the name of a Natural transaction.

The settings of the Natural profile/session parameters `IA` (Input Assign Character) and `ID` (Input Delimiter Character) at the time of execution of the `STACK` statement are used to determine the `*DATA` value.

*DIALOG-ID

Format/length:	I4
Content modifiable:	No

This system variable contains the ID of the current instance of a dialog.

For details on dialogs, see *Event-Driven Programming Techniques* in the *Programming Guide*.

*EDITOR

Format/length:	L
Content modifiable:	No

This system variable indicates whether the Natural program, data area, and map editors are enabled and can be used.

It can contain one of the following values:

Value	Description
TRUE	The editors are enabled.
FALSE	The editors are not enabled and cannot be used.

*ERROR-LINE

Format/length:	N4
Content modifiable:	No

This system variable contains the source-code line number of the statement that caused an error.

`*ERROR-LINE` is reset to `0` when a Level 1 program starts executing.

*ERROR-NR

Alternatively, you may specify `*ERROR`.

Format/length:	N7
Content modifiable:	Yes

This system variable contains the error number of the error which caused an `ON ERROR` condition to be entered.

Only error numbers in the range from 0 to 9999 are supported.

Normally, `*ERROR-NR` contains the Natural *system* error number which caused an error condition to be entered; however, when a `REINPUT WITH TEXT *nnnn` statement is executed, the *application-specific* message number *nnnn* is placed into `*ERROR-NR`.

You may modify the content of this system variable via a Natural program; however, not within an `ON ERROR` statement block.

`*ERROR-NR` is reset to `0` when a Level 1 program starts executing.

*ERROR-TA

Format/length:	A8
Content modifiable:	Yes

This system variable contains the name of the error transaction program which is to receive control in the event of an error condition.

For further information, see *Using an Error Transaction Program* in the *Programming Guide*.

*ETID

Format/length:	A8
Content modifiable:	No

This system variable contains the current identifier of transaction data for Adabas. The default value is one of the following:

- the value of the Natural profile parameter `ETID`,
- the value from the security profile of the currently active user (applies only under Natural Security).

*EVENT

Format/length:	A32
Content modifiable:	No

This system variable contains the name of the current event.

For details on events, see *Event-Driven Programming Techniques* in the *Programming Guide*.

*ISN (r)

Format/length:	P10
Content modifiable:	Yes

This system variable contains the Adabas internal sequence number (ISN) of the record currently being processed within a processing loop initiated by a FIND or READ statement.

(*r*) notation after *ISN is used to indicate the label or statement number of the statement in which the FIND or READ was issued. If (*r*) is not specified, *ISN represents the ISN of the record currently being processed in the currently active processing loop.

For the HISTOGRAM statement, *ISN contains the number of the occurrence in which the descriptor value last read is contained (*ISN = 0 if the descriptor is not contained within a periodic group).

Database-Specific Information:

SQL Databases	*ISN cannot be used.
Tamino	*ISN contains the XML object ID.

*LBOUND

Format/length:	I4
Content modifiable:	No

*LBOUND contains the current lower boundary (index value) of an array for the specified dimension(s) (1, 2 or 3) or for all dimensions (asterisk (*) notation).

Syntax:

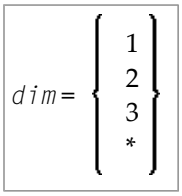
*LBOUND (*operand1* [,*dim*])

Operand Definition Table:

Operand	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
operand1			A			A	U	N	P	I	F	B	D	T	L	C	G	O	yes	no

operand1 is the array for which the lower boundary is specified. The index notation of the array is optional. As index notation only the complete range notation * is allowed for each dimension.

dim is the dimension number for which the current lower boundary is returned:



If no dimension is specified, the lower bound of the first dimension is returned.

If 1, 2 or 3 is specified, the lower bound of the first, second or third dimension is returned.

If * is specified, the lower bound of all the defined dimensions are returned, that is,

- 1 in case of an one dimensional array,
- 2 in case of a two dimensional array,
- 3 in case of three dimensional array.

If an X-array is not allocated and the lower bound of the specified dimension of this X-array is the variable index bound, that is, it is represented by an asterisk (*) character in the index definition, the lower bound of the specified dimension is undefined, and access to `*LBOUND` leads to a runtime error. In order to avoid the runtime error, `*OCCURRENCE` may be used to check against zero occurrences:

```
DEFINE DATA LOCAL
  1 #XA(A5/1:*)
END-DEFINE
IF *OCCURRENCE (#XA) NE 0 AND *LBOUND(#XA) > 10
  THEN ...
```

Examples:

```
DEFINE DATA LOCAL
  1 #I  (I4)
  1 #J  (I4/1:3)
  1 #XA (A5/10:*,20:*)
END-DEFINE
#I  := *LBOUND(#XA)       /* lower bound of 1st dimension is 10
#I  := *LBOUND(#XA,1)     /* lower bound of 1st dimension is 10
#I  := *LBOUND(#XA,2)     /* lower bound of 2nd dimension is 20
#J(1:2):= *LBOUND(#XA,*)  /* lower bound of all dimensions
                          /* #J(1) is 10 and #J(2) is 20
END
```

See also *UBOUND and *OCCURRENCE.

*LENGTH (field)

Format/length:	I4
Content modifiable:	No

This system variable returns the currently used length of a field defined as dynamic variable in terms of code units; for A and B format the size of one code unit is 1 byte and for U format the size of one code unit is 2 bytes (UTF-16). *LENGTH*(field)* applies to dynamic variables only.

See also *Value Space Currently Used for a Dynamic Variable* in the *Programming Guide*.

*LEVEL

Format/length:	N2
Content modifiable:	No

This system variable contains the level number of the program, subprogram, external subroutine, map, helproutine or dialog which is currently active. Level 1 is a main program. If higher levels occur during runtime (maximum = 512), then the content of *LEVEL will be 99.

*LEVEL does not apply to inline subroutines.

See also *Using an Error Transaction Program* in the *Programming Guide*.

*LIBRARY-ID

Format/length:	A8
Content modifiable:	No

This system variable contains the current library ID (as specified by the user in the `LOGON` command).

This variable is the equivalent of the variable `*APPLIC-ID`.

*LINE

Format/length:	I4
Content modifiable:	No

It contains the number of the line currently executed in a Natural object.

*LINEX

Format/length:	A100
Content modifiable:	No

Returns the line number of the statement currently executing, with all line numbers of used `INCLUDE` levels.

When used in the main source of a program (not inside a copycode), it returns only the number of the line in which it is referenced. In this case, the data is the same as what is returned by `*LINE`, except the format is (A100).

If `*LINEX` is used in a (nested) `INCLUDE` structure, all the line numbers starting from the first `INCLUDE` up to the last `INCLUDE` and the line number of the statement are returned in this field, separated from each other by a slash.

Example:

```
....
3200
3210 INCLUDE COPY01
     0010 ...
     0020 ...
     ....
     0200 INCLUDE COPY02
          0010 ...
          0020 ...
          ....
          0050 PRINT *LINEX
```

The output produced by the PRINT statement is 3210/0200/0050 and represents the complete line number path needed to exactly locate the position where *LINEX was referenced initially.

*LOAD-LIBRARY-ID

Format/length:	A8
Content modifiable:	No

This system variable contains the library ID from where the current executed object was loaded.

*NUMBER (r)

Format/length:	P10
Content modifiable:	Yes

This system variable contains either of the following:

- the number of records which were selected as a result of a FIND statement (as a result of the WITH clause);
- the number of values selected as a result of a HISTOGRAM statement;

(*r*) notation after *NUMBER is used to indicate the statement label or source-code line number of the associated statement. If (*r*) is not specified, *NUMBER automatically refers to the innermost active FIND or HISTOGRAM processing loop by default.

Note: When *NUMBER is used in conjunction with a FIND statement and the Adabas file accessed is protected by the Adabas facility Security By Value, *NUMBER will contain 9999999999, if more than 1 record was found. If 1 record was found, *NUMBER will contain 1. If no record was found, *NUMBER will contain 0.

Database-Specific Information:

SQL Databases	For SQL databases, *NUMBER only contains the number of rows found, when used with a FIND NUMBER or with a HISTOGRAM statement without a WHERE clause. In all other cases, *NUMBER will not contain the number of rows found: *NUMBER will be 0 if no rows have been found; any value other than 0 indicates that rows have been found, but the value will have no relation to the number of rows actually found.
Tamino	When used with a FIND NUMBER statement without a WHERE clause, *NUMBER will contain the number of rows found. Otherwise, when applied to an XML database, *NUMBER will not contain the number of rows found: *NUMBER will be 0 if no rows have been found. Any value other than 0 indicates that rows have been found. However, the value will have no relation to the number of rows actually found. If a FIND NUMBER with a WHERE clause is used, the number of rows found is returned in *COUNTER.

*OCCURRENCE

Format/length:	I4
Content modifiable:	No

This system variable provides the current number of occurrences of an array. It can be applied to all kinds of array fields, with a fixed or variable number of occurrences.

This covers:

- static arrays with a constant number of occurrences

  Example: (1:5)
- X-Arrays with an alterable number of occurrences

  Example: (1:*)
- parameter arrays, defined as (1:V)

Syntax:

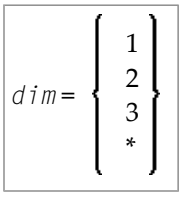
*OCCURRENCE (*operand1* [,*dim*])

Operand Definition Table:

Operand	Possible Structure					Possible Formats													Referencing Permitted	Dynamic Definition
operand1			A			A	U	N	P	I	F	B	D	T	L	C	G	O	yes	no

operand1 is the array for which the number of occurrences is returned. The index notation of the array is optional. If supplied, only the complete range notation * is allowed for each dimension, such as `*OCC(#X(*))` or `*OCC(#Y(*,*))`.

dim is the dimension number for which the current number of occurrences is returned:



Explanation:

1	One-dimensional array. This is the default, if *dim* is not specified.
2	Two-dimensional array.
3	Three-dimensional array.
*	All dimensions defined for the corresponding array apply.

In a parameter data area, you can use the index notation `(1:V)` to define an array with a variable number of occurrences (see the `DEFINE DATA` statement). The current number of occurrences of such an array is determined at runtime. With `*OCCURRENCE`, you can ascertain the current number of array occurrences.

Examples:

```
DEFINE DATA
PARAMETER
  1 #PARR  (I2/1:V)
LOCAL
  1 #FARR  (I2/1:5)
  1 #XARR1 (I2/1:*)
  1 #XARR2 (I2/1:*,1:*)
  1 #I     (I2)
  1 #J     (I2)
END-DEFINE
FOR #I = 1 TO *OCC(#PARR)       /* Parameter array
  WRITE 2X #I
END-FOR
FOR #I = 1 TO *OCC(#FARR)       /* Fixed array
  WRITE 4X #I
END-FOR
EXPAND ARRAY #XARR1 TO (1:4)
```

```
FOR #I = 1 TO *OCC(#XARR1)      /* X-Array
  WRITE 6X #I
END-FOR
EXPAND ARRAY #XARR2 TO (1:3,1:4)
FOR #I = 1 TO *OCC(#XARR2,1)    /* X-Array
  FOR #J = 1 TO *OCC(#XARR2,2)
    WRITE 8X #I #J
  END-FOR
END-FOR
END
```

See also the example programs OCC1P and OCC2P.

Concerning X-arrays, *OCCURRENCE contains the current number of occurrences:

```
DEFINE DATA LOCAL
  1 #I   (I4)
  1 #J   (I4/1:3)
  1 #XA  (A5/1:*,1:*)
END-DEFINE
EXPAND ARRAY #XA TO (1:10,1:20)
#I      := *OCC(#XA)    /* #I=10
#I      := *OCC(#XA,1)  /* #I=10
#I      := *OCC(#XA,2)  /* #I=20
#J(1:2) := *OCC(#XA,*)  /* #J(1)=10  #J(2)=20
END
```

*PAGE-EVENT

Format/length:	U(dynamic)
Content modifiable:	No

This system variable contains the name of the current event delivered from Natural for Ajax.

It is used for rich GUI programming with the PROCESS PAGE statement. For further information, see the *Natural for Ajax* documentation.

*PAGE-LEVEL

Format/length:	I4
Content modifiable:	No

This system variable contains the level of the active PROCESS PAGE MODAL statement blocks.

If no PROCESS PAGE MODAL is active, the value of *PAGE-LEVEL is 0.

Note: If the value of *PAGE-LEVEL is greater than 0, no output to Report 0 via an INPUT, PRINT, WRITE or DISPLAY statement is possible.

*PROGRAM

Format/length:	A8
Content modifiable:	No

This system variable contains the name of the Natural object that is currently being executed.

*REINPUT-TYPE

Format/length:	A16
Content modifiable:	No

This system variable indicates whether the application is in a state that allows execution of a REINPUT or PROCESS PAGE UPDATE statement.

An application can use an INPUT and (if running with Natural for Ajax) a PROCESS PAGE USING statement to perform an input/output processing. Under certain conditions, an application may return and re-execute these I/O statements with a REINPUT or PROCESS PAGE UPDATE statement.

The value returned by this system variable indicates whether or not such a re-executing statement is possible at this position. It returns one of the following values:

Value	Description
(blanks)	The application can neither perform a REINPUT nor a PROCESS PAGE UPDATE statement.
REINPUT	The application can perform a REINPUT, but no PROCESS PAGE UPDATE statement.
UPDATE	The application can perform a PROCESS PAGE UPDATE, but no REINPUT statement.

*ROWCOUNT

Format/length:	I4
Content modifiable:	No

This system variable contains the number of rows that were deleted, updated or inserted by one of the Natural SQL statements "searched" DELETE, "searched" UPDATE or INSERT (with *select-expression*), respectively. *ROWCOUNT always refers to the last executed one of these statements.

*STARTUP

Format/length:	A8
Content modifiable:	Yes

The program whose name is contained in this system variable will be executed whenever Natural would otherwise display the command input prompt (NEXT prompt or direct command line/window).

- Activation of *STARTUP
- Deactivation of *STARTUP
- Avoiding Undesirable Results when Using *STARTUP

Activation of *STARTUP

Initial Setting without Natural Security

If Natural Security is not used, the value of the NATPARM parameter STARTUP applies.

Initial Setting with Natural Security

If Natural Security is used and a logon to a library is executed, *STARTUP contains the name of the program which has been entered in Natural Security as the startup transaction in the security profile of the respective library (except in batch mode; see also the *Natural Security* documentation).

Setting at Runtime

You can assign a program name to *STARTUP which always overwrites the content of *STARTUP.

Deactivation of *STARTUP

Perform the following steps to deactivate the program contained in *STARTUP:

- Depending on the context in which *STARTUP is used, there are several ways to properly terminate a startup program without causing a program loop or Natural error NAT9969, respectively. Choose one of the following options:
  1. Set the *STARTUP system variable to a blank value (for example, by using the RESET *STARTUP statement) and finish the startup afterwards.
  2. Terminate the session.
  3. Force a logoff or a logon to another library.
- If the *STARTUP program issues a MORE prompt, enter a command or another input to interrupt the program.
- If the *STARTUP program issues any output, request a command prompt by issuing a Natural %% terminal command.

  A %% command deactivates the startup program in either a non-security or Natural Security environment in which command mode is allowed for the current library.

Avoiding Undesirable Results when Using *STARTUP

When you define a startup program, consider that values returned for the system variables *DEVICE, *SERVER-TYPE or *SCREEN-IO can affect the program result. For example, perform the following steps to avoid undesirable results:

- In batch mode (*DEVICE or *SCREEN-IO), include a FETCH or STACK COMMAND statement in the startup program; otherwise, a program loop or Natural error NAT9969 can occur.
- If you map a Natural Development Server environment (*SERVER-TYPE) from NaturalONE and select a library with an active *STARTUP, make sure that the startup program does not perform screen I/O and terminates properly in the respective library (a logon to another library is not allowed). You can use the application programming interface USR4218N to find out whether NaturalONE is your current system environment. See also the description of startup transactions in the section *Using an Existing Natural Development Server Environment*, the section *Frequently Asked Questions* and the *Glossary* in the *NaturalONE* documentation.

***STARTUP under Natural Security**

In a Natural Security environment in which command mode is prohibited for the current library, %% will cause the program whose name is contained in *STARTUP to be invoked.

The startup program must contain a FETCH or STACK COMMAND statement if the NEXT/MORE line is not allowed in the security profile of the library where the program resides. See also the corresponding option setting described in *Security Options* in the section *Library Maintenance* in the *Natural Security* documentation.

When a Natural runtime error occurs which is caused by a startup transaction (*STARTUP), Natural's error processing might lead to the startup transaction being executed again. This would cause an error-loop situation. To prevent such a loop, the general option **Logoff in error case if *STARTUP is active** is available. See *Logoff in Error Case if *STARTUP is Active* in the *Natural Security* documentation.

*STEPLIB

Format/length:	A8
Content modifiable:	No

This system variable contains the name of the steplib library which has been concatenated to the Natural library to which the user is currently logged on.

If Natural Security is not active, *STEPLIB contains the *STEPLIB name specified with the profile parameter STEPLIB in the parameter file used.

If Natural Security is active, the value may be defined in the security profile of a given library.

Note: The database ID and file number of the *STEPLIB library are derived from its name. Apart from the library SYSTEM, libraries with the name SYS*xxx* are assumed to be in FNAT and other libraries are assumed to be in FUSER.

*SUBROUTINE

Format/length:	A32
Content modifiable:	No

This system variable contains the name of the external subroutine that is currently being executed. The contents of *SUBROUTINE will always be in upper case.

*THIS-OBJECT

Format/length:	HANDLE OF OBJECT
Content modifiable:	No

This system variable contains a handle to the currently active object. The currently active object uses `*THIS-OBJECT` to either execute its own methods or pass a reference to itself to another object.

`*THIS-OBJECT` only contains an actual value when a method is being executed. Otherwise it contains `NULL-HANDLE`.

*TYPE

Format/length:	A32
Content modifiable:	No

This system variable contains the type of the Natural object which is currently executed.

Valid values of `*TYPE`:

Value	Object Type
`PROGRAM`	Program
`FUNCTION`	Function
`SUBPROGRAM`	Subprogram
`SUBROUTINE`	Subroutine
`HELPROUTINE`	Helproutine
`MAP`	Map
`ADAPTER`	Adapter
`DIALOG`	Dialog

*UBOUND

Format/length:	I4
Content modifiable:	No

UBOUND contains the current upper boundary (index value) of an array for the specified dimension(s) (1, 2 or 3) or for all dimensions (notation).

Syntax:

*UBOUND (*operand1* [,*dim*])

Operand Definition Table:

Operand	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
operand1			A			A	U	N	P	I	F	B	D	T	L	C	G	O	yes	no

operand1 is the array for which the upper boundary is specified. The index notation of the array is optional. As index notation only the complete range notation * is allowed for each dimension.

dim is the dimension number for which the current upper boundary is returned:

dim = { 1 | 2 | 3 | * }

If no dimension is specified, the upper bound of the first dimension is returned.

If 1, 2 or 3 is specified, the upper bound of the first, second or third dimension is returned.

If * is specified, the upper bound of all the defined dimensions are returned, that is

- 1 in case of an one dimensional array,
- 2 in case of a two dimensional array,
- 3 in case of three dimensional array.

If an X-array is not allocated and the upper bound of the specified dimension of this X-array is the variable index bound, that is, it is represented by an asterisk (*) character in the index definition, the upper bound of the specified dimension is undefined, and access to *UBOUND leads to a runtime

error. In order to avoid the runtime error, *OCCURRENCE may be used to check against zero occurrences:

```
DEFINE DATA LOCAL
  1 #XA(A5/1:*)
END-DEFINE
IF *OCCURRENCE (#XA) NE 0 AND *UBOUND(#XA) > 10
  THEN ...
```

Examples:

```
DEFINE DATA LOCAL
  1 #I  (I4)
  1 #J  (I4/1:3)
  1 #XA (A5/*:10,*:20)
END-DEFINE
#i  := *UBOUND(#XA)        /* upper bound of 1st dimension is 10
#i  := *UBOUND(#XA,1)      /* upper bound of 1st dimension is 10
#i  := *UBOUND(#XA,2)      /* upper bound of 2nd dimension is 20
#j(1:2):= *UBOUND(#XA,*)   /* upper bound of all dimensions
                           /* (1st and 2nd)
                           /* #J(1) is 10 and #J(2) is 20
```

See also *LBOUND and *OCCURRENCE.

4 Date and Time System Variables

- Usage 32
- *DAT* - Date System Variables 32
- *TIM* - Time System Variables 33
- Example of Date and Time System Variables 34

Usage

The date and time system variables listed below may be specified in the following places:

- statements:
  - COMPUTE
  - DISPLAY
  - MOVE
  - PRINT
  - WRITE
- logical condition criteria

The contents of date and time system variables as generated by Natural are *non-modifiable*, which means that in a Natural program you cannot assign another value to any of them.

DAT - Date System Variables

All date system variables contain the current date. The format of the date is different for each date variable, as indicated below.

Date Variable	Format/Length	Date Format*
*DATD	A8	DD.MM.YY
*DAT4D	A10	DD.MM.YYYY
*DATE	A8	DD/MM/YY
*DAT4E	A10	DD/MM/YYYY
*DATG	A15	DD*monthname*YYYY (Gregorian date)
*DATI	A8	YY-MM-DD
*DAT4I	A10	YYYY-MM-DD
*DATJ	A5	YYJJJ (Julian date)
*DAT4J	A7	YYYYJJJ (Julian date)
*DATN	N8	YYYYMMDD
*DATU	A8	MM/DD/YY
*DAT4U	A10	MM/DD/YYYY
*DATV	A11	DD-MON-YYYY
*DATVS	A9	DDMONYYYY
*DATX	D	internal date format

* D = day, J = Julian day, M = month, Y = year, MON = leading three bytes of the month's name as in *DATG

TIM - Time System Variables

At runtime, the content of a time system variable is evaluated anew each time the variable is referenced in a Natural program. The format of the time is different for each time variable, as indicated below.

Time Variable	Format/Length	Explanation
*TIMD (*r*)	N7	Can only be used in conjunction with a previous SETTIME statement. Contains the time that has elapsed after the SETTIME statement was executed (in the format HHIISST (*)). (*r*) represents the statement label or source-code line number of the SETTIME statement used as the basis for *TIMD.
*TIME	A10	Contains the time of day in the format HH:II:SS.T (*).
*TIME-OUT	N5	Contains the number of seconds remaining before the current transaction will be timed out (only available with Natural Security). *TIME-OUT is 0 if transaction mode has not been entered. Transaction mode is entered with the execution of a FIND, READ or GET statement that reads a database record for the purpose of updating or deleting the record. *TIME-OUT is reset to 0 when an END TRANSACTION or BACKOUT TRANSACTION statement is executed.
*TIMESTMP	B8	Machine-internal store clock value as provided by the Assembler instruction STCK (Store Clock format). For more information about the meaning of this value, please refer to the IBM documentation. The processing of store clock values is described in *Programming Guide* > *Further Programming Aspects* > *Processing of Store Clock Values*.
*TIMESTMPX	B16	Machine-internal store clock value as provided by the Assembler instruction STCKE (Store Clock Extended format). For more information about the meaning of this value, please refer to the IBM documentation. The processing of extended store clock values is described in *Programming Guide* > *Further Programming Aspects* > *Processing of Store Clock Values*.
*TIMN	N7	Contains the time of day in format HHIISST (*).
*TIMX	T	Contains the time of day in internal time format.

* H = hour, I = minute, S = second, T = tenth of a second.

Example of Date and Time System Variables

```
** Example 'DATIVAR': Date and time system variables
************************************************************************
DEFINE DATA LOCAL
1 #DATE (D)
1 #TIME (T)
END-DEFINE
*
WRITE NOTITLE
  'DATE IN FORMAT DD.MM.YYYY  '   *DAT4D /
  'DATE IN FORMAT DD/MM/YYYY  '   *DAT4E /
  'DATE IN FORMAT DD-MON-YYYY '   *DATV  /
  'DATE IN FORMAT DDMONYYYY   '   *DATVS /
  'DATE IN GREGORIAN FORM     '   *DATG  /
  'DATE IN FORMAT YYYY-MM-DD  '   *DAT4I /
  'DATE IN FORMAT YYYYDDD     '   *DAT4J /
  'DATE IN FORMAT YYYYMMDD    '   *DATN (AD=L) /
  'DATE IN FORMAT MM/DD/YYYY  '   *DAT4U /
  'DATE IN INTERNAL FORMAT    '   *DATX (DF=L) ///
  'TIME IN FORMAT HH:II:SS.T  '   *TIME /
  'TIME IN FORMAT HHIISST     '   *TIMN (AD=L) /
  'TIME IN INTERNAL FORMAT    '   *TIMX /
  'TIME IN STORE CLOCK FORMAT '   *TIMESTMP /
  'TIME IN EXT. STCK FORMAT   '   *TIMESTMPX /
*
MOVE *DATX TO #DATE
ADD 14 TO #DATE
WRITE 'CURRENT DATE'              *DATX (DF=L) 3X
      'CURRENT DATE + 14 DAYS   ' #DATE (DF=L)
*
MOVE *TIMX TO #TIME
ADD 100 TO #TIME
WRITE 'CURRENT TIME'              *TIMX 5X
      'CURRENT TIME + 10 SECONDS' #TIME
*
END                                                  ↩
```

Output of program DATIVAR:

```
DATE IN FORMAT DD.MM.YYYY   05.05.2023
DATE IN FORMAT DD/MM/YYYY   05/05/2023
DATE IN FORMAT DD-MON-YYYY  05-May-2023
DATE IN FORMAT DDMONYYYY    05May2023
DATE IN GREGORIAN FORM      05May      2023
DATE IN FORMAT YYYY-MM-DD   2023-05-05
DATE IN FORMAT YYYYDDD      2023125
DATE IN FORMAT YYYYMMDD     20230505
DATE IN FORMAT MM/DD/YYYY   05/05/2023
DATE IN INTERNAL FORMAT     2023-05-05


TIME IN FORMAT HH:II:SS.T   13:26:06.4
TIME IN FORMAT HHIISST      1326064
TIME IN INTERNAL FORMAT     13:26:06
TIME IN STORE CLOCK FORMAT  DD400D6D2A5C700E
TIME IN EXT. STCK FORMAT    00DD400D6D2A5C800E00000000000000

CURRENT DATE 2023-05-05   CURRENT DATE + 14 DAYS    2023-05-19
CURRENT TIME 13:26:06     CURRENT TIME + 10 SECONDS 13:26:16
```

5 Input/Output-Related System Variables

- *CURS-COL 38
- *CURS-FIELD 38
- *CURS-LINE 39
- *CURSOR 40
- *LINE-COUNT 40
- *LINESIZE 41
- *LOG-LS 41
- *LOG-PS 41
- *PAGE-NUMBER 41
- *PAGESIZE 42
- *PF-KEY 42
- *PF-NAME 43
- *WINDOW-LS 43
- *WINDOW-POS 44
- *WINDOW-PS 44

*CURS-COL

Format/length:	P3
Content modifiable:	Yes (however, a negative value must not be assigned)

This system variable contains the number of the column in which the cursor is currently positioned.

The cursor position is defined within the currently active window, regardless of its physical placement on the screen, starting with position 1/1 from the upper left corner of a logical page.

If the value of *CURS-COL is negative, this indicates that the cursor is outside the active window. If *CURS-COL is negative, *CURS-LINE will also contain a negative value. In this case, the absolute values of both system variables indicate the position of the cursor on the physical screen.

Note: The message line, function-key lines and infoline/statistics line are not counted as data lines on the screen.

See also *Dialog Design*, *Column-Sensitive Processing* in the *Programming Guide*.

*CURS-FIELD

Format/length:	I4
Content modifiable:	No

This system variable returns an identification of the field in which the cursor is currently positioned. The value returned is an internal representation of the field address.

*CURS-FIELD cannot be used by itself, but only in conjunction with the POS function. You may use them to check if the cursor is currently positioned in a specific field and have processing performed depending on that condition. See the POS function for details.

If the cursor is not in a field or if no REINPUT is possible, *CURS-FIELD contains 0.

In Natural for Ajax applications, *CURS-FIELD identifies the operand that represents the value of the control that has the input focus. You may use *CURS-FIELD in conjunction with the POS function to check for the control that has the input focus and perform processing depending on that condition.

Note: *CURS-FIELD cannot distinguish between two different variables that start at the same storage position (REDEFINE variables) since the internal field address returned by *CURS-FIELD is the same for both.

The value of *CURS-FIELD serves only as internal identification of the field and cannot be used for arithmetic operations. If *CURS-FIELD identifies an occurrence of an X-array (an array for which at least one bound in at least one dimension is specified as extensible), the value of *CURS-FIELD may change after the number of occurrences for a dimension of the array has been changed using the EXPAND, RESIZE or REDUCE statements.

See also *Dialog Design*, *Field-Sensitive Processing* in the *Programming Guide*.

*CURS-LINE

Format/length:	P3
Content modifiable:	Yes (however, a negative value or 0 must not be assigned)

This system variable contains the number of the line in which the cursor is currently positioned.

The cursor position is defined within the current active window, regardless of its physical placement on the screen, starting with position 1/1 from the upper left corner of a *logical* page.

Note: The message line, function-key lines and infoline/statistics line are not counted as data lines on the screen.

*CURS-LINE may also contain one of the following values:

Value	Cursor Position
0	On the top or bottom horizontal frame line of a window.
-1	On the Natural message line.
-2	On the Natural infoline/statistics line.
-3	On the upper function-key (number) line.
-4	On the lower function-key (name) line.

If the value of *CURS-COL is negative, which indicates that the cursor is outside the active window, *CURS-LINE will also contain a negative value. In this case, the *absolute* values of both system variables indicate the position of the cursor on the *physical* screen.

See also *Dialog Design*, *Line-Sensitive Processing* in the *Programming Guide*.

*CURSOR

Format/length:	N6
Content modifiable:	No

This system variable contains the position of the cursor on the input screen at the time the ENTER key or a function key is pressed.

Note: Instead of *CURSOR, it is recommended that the system variables *CURS-LINE and *CURS-COL be used. *CURSOR only continues to be available for compatibility with previous Natural versions.

*LINE-COUNT

Format/length:	P5
Content modifiable:	No

This system variable contains the line number of the current line within the current page.

This variable is used by Natural to determine the line number for the next line of the report.

The value of *LINE-COUNT is incremented by 1 for each line to be output. The value is updated during the execution of a WRITE, SKIP, DISPLAY, PRINT or INPUT statement and contains the number of the last line on the page that has been output.

An EJECT or NEWPAGE statement causes *LINE-COUNT to be reset to 1 (except in the case of NEWPAGE WITH TITLE, where the value of *LINE-COUNT depends on the number of lines output as title).

The maximum line number permitted is 250.

If multiple reports are being produced by the program, (*rep*) notation after *LINE-COUNT is used to specify the report identification for which the current line number is being requested.

*LINESIZE

Format/length:	N7
Content modifiable:	No

This system variable contains the physical line size of the I/O device from which Natural was invoked (if the TP system is able to provide such).

*LOG-LS

Format/length:	N3
Content modifiable:	No

This system variable contains the line size of the logical page that is output with the primary report.

`*LOG-LS` is only applicable to the primary report, not to any additional report.

*LOG-PS

Format/length:	N3
Content modifiable:	No

This system variable contains the page size of the logical page that is output with the primary report.

`*LOG-PS` is only applicable to the primary report, not to any additional report.

*PAGE-NUMBER

Format/length:	P5
Content modifiable:	Yes

This system variable contains the current value for page number of an output report.

If multiple reports are being produced by the program, `(rep)` notation after `*PAGE-NUMBER` is used to specify the report identification for which the current page number is being requested.

This variable is defined by Natural at the time formatting for the report is started. Therefore, the parameter has no meaning until the first FORMAT, WRITE, or DISPLAY statement for any given report has been issued. This variable may be modified by a Natural program.

This variable is used by Natural to determine the page number for the next page of the report. The value is always incremented by 1 for the next page initiated by WRITE, DISPLAY, SKIP or NEWPAGE statements. EJECT does not cause *PAGE-NUMBER to be incremented.

*PAGESIZE

Format/length:	N7
Content modifiable:	No

This system variable contains the physical page size of the I/O device from which Natural was invoked (if the TP subsystem is able to provide such).

*PF-KEY

Format/length:	A4
Content modifiable:	No

This system variable contains the identification of the key which was pressed last.

*PF-KEY can contain one of the following values:

Value	Description
PA1 to PA3	Program Attention keys 1 to 3.
PF1 to PF48	Program Function keys 1 to 48.
ENTR	ENTER key.
CLR	CLEAR key.
PEN	Light pen.
PGDN	PAGE DOWN key.
PGUP	PAGE UP key.

*PF-KEY only contains the identification of a key if that key is currently sensitive; otherwise *PF-KEY will contain ENTR.

Notes:

1. When a page break occurs, *PF-KEY changes to ENTR. This applies to all environments (terminal or non-terminal).
2. When you compare the content of *PF-KEY with a range of values, remember that *PF-KEY contains an alphanumeric value.

See also:

- SET KEY statement (for effects on the contents of *PF-KEY)
- *Processing Based on Function-Keys* in the *Programming Guide*
- *%K and %KP - Simulate PF- and PA-Key* (terminal commands which set *PF-KEY) in the *Terminal Commands* documentation

*PF-NAME

Format/length:	A10
Content modifiable:	No

This system variable contains the name of the function key that was pressed last, that is, the name as assigned to the key with the NAMED clause of the SET KEY statement.

This allows you to perform processing depending on a specific function name, not a specific key. For example, if you wish to allow users to invoke help by pressing either PF1 or PF13, you assign the name HELP to the keys PF1 and PF13 and make the invoking of help dependent on *PF-NAME='HELP': the help will then be invoked no matter whether the user presses PF1 or PF13 to invoke it.

See also *Dialog Design*, *Processing Based on Function-Key Names* in the *Programming Guide*.

*WINDOW-LS

Format/length:	N3
Content modifiable:	No

This system variable contains the line size of the logical window (without frame). See also the DEFINE WINDOW statement.

*WINDOW-POS

Format/length:	N6
Content modifiable:	No

This system variable contains the position which corresponds to the upper left corner of the window. See also the `DEFINE WINDOW` statement.

The position is counted in characters across multiple lines, beginning with `0` (upper left corner).

*WINDOW-PS

Format/length:	N3
Content modifiable:	No

This system variable contains the page size of the logical window (without frame). See also the `DEFINE WINDOW` statement.

6 JSON-Related System Variables

- *PARSE-LEVEL (r) for JSON 46
- *PARSE-INDEX (r) for JSON 46
- *PARSE-TYPE (r) for JSON 47

This document covers the following topics:

The system variables, which are available when using the PARSE JSON statement, are only valid in the context of the current loop.

*PARSE-LEVEL (r) for JSON

Format/length:	I4
Content modifiable:	No

This system variable contains the level of currently nested elements.

(r) notation after *PARSE-LEVEL is used to indicate the statement label or source-code line number of the PARSE JSON statement. If (r) is not specified, *PARSE-LEVEL represents the level where the parser in the currently active processing loop is working at.

*PARSE-INDEX (r) for JSON

Format/length:	I4
Content modifiable:	No

This system variable contains the index of currently nested elements.

(r) notation after *PARSE-LEVEL is used to indicate the statement label or source-code line number of the PARSE JSON statement. If (r) is not specified, *PARSE-INDEX represents the index of the array element where the parser in the currently active processing loop is working at.

Note: System variable *PARSE-INDEX is only applicable to array elements and is specific to the PARSE JSON statement.

*PARSE-TYPE (r) for JSON

Format/length:	A1
Content modifiable:	No

This Natural system variable is automatically created for each `PARSE JSON` statement issued.

This system variable contains the type of the delivered data.

`(r)` notation after `*PARSE-TYPE` is used to indicate the statement label or source-code line number of the `PARSE JSON` statement. If `(r)` is not specified, `*PARSE-TYPE` represents the type of the delivered data in the currently active processing loop.

Possible values for ASCII-based systems are:

<	Start of an Object Data-Structure
>	End of an Object Data-Structure
(	Start of an Array Data-Structure
)	End of an Array Data-Structure
A	String type Data-Element
I	Numeric type Data-Element
N	Null type Data-Element
L	Boolean type Data-Element
K	Key (or) Data-Name

7 Natural Environment-Related System Variables

- *BROWSER-IO 50
- *DEVICE 50
- *GROUP 51
- *HARDCOPY 51
- *INIT-USER 51
- *LANGUAGE 52
- *NATVERS 55
- *NET-USER 56
- *PARM-USER 56
- *PATCH-LEVEL 56
- *PID 57
- *SCREEN-IO 57
- *SERVER-TYPE 58
- *UI 58
- *USER 59
- *USER-NAME 59
- *UUID 59

*BROWSER-IO

Format/length:	A8
Content modifiable:	No

This system variable indicates that the application is running in a web browser. An application can run in a web browser either via the Natural Web I/O Interface or by using Natural for Ajax. An application that is running with the Natural Web I/O Interface can use maps. An application that is running with Natural for Ajax can use both maps and rich GUI pages (using the `PROCESS PAGE` statement).

This system variable may contain one of the following values:

Value	Description
(empty)	The application is not running in a web browser.
WEB	The application is running with the Natural Web I/O Interface. It cannot use the `PROCESS PAGE` statement.
RICHGUI	The application is running with Natural for Ajax. It can use the `PROCESS PAGE` statement.

*DEVICE

Format/length:	A8
Content modifiable:	No

This system variable contains the device type/mode from which Natural has been invoked. It may contain one of the following values:

Value	Description
BATCH	Batch mode.
VIDEO	3270 screen device, PC screen device, VT or X terminal or any type of Linux terminal.
TTY	Teletype or other start/stop device.

*GROUP

Format/length:	A8
Content modifiable:	No

This system variable is applicable under Natural Security only. It contains the ID via which a user is logged on to a protected library, that is, the ID via which the user is linked to the library. This may be either the ID of the group via which the user is linked or the user's own ID (if he or she is linked directly).

`*GROUP` will be blank under the following conditions:

- in the case of a logon to an unprotected library (where no link is used),
- if Natural Security is not active.

*HARDCOPY

Format/length:	A8
Content modifiable:	Yes

This system variable contains the name of the hardcopy device which will be used when the terminal command `%H` is used.

*INIT-USER

Format/length:	A8
Content modifiable:	No

`*INIT-USER` contains the value of the profile parameter `USER` in the parameter file used.

If no value is specified for the `USER` parameter, `*INIT-USER` contains the ID you were requested to enter when starting Natural (default value: `SAGPC`).

*LANGUAGE

Format/length:	I1
Content modifiable:	Yes

This system variable contains the language indicator (language code). This language indicator is used for edit masks of date fields, Natural error messages and user error messages as used in the statements `INPUT` and `REINPUT`.

A one-character code is assigned to each language code; this one-character code is used to replace all ampersand characters (&) in names of language-specific objects (for example, maps, dialogs, helproutines, subprograms). For details on the use of the ampersand character in Natural object names, see the descriptions of the statements `CALLNAT` (*operand1*), `FETCH` (*operand1*), `INCLUDE` (*copycode-name*) and `INPUT` (`USING MAP`) and the session parameter `HE` (*operand1*).

You can specify up to 60 different language codes. The codes are listed below.

The system variable `*LANGUAGE` is set by the Natural profile parameter `ULANG` which determines the language to be used for date edit masks, system messages, user messages, help texts, help routines and multi-lingual maps.

Natural does not differentiate between compile time and run time. It always tries to read the map with the current value of `*LANGUAGE` first and if not found, it then tries to find the map with the default language.

For details on how to use language codes, see also *Character-Based Application User Interfaces* in the *Programming Guide*.

Language Code Assignments

The following languages are assigned to the individual language codes (the right-hand column shows the corresponding one-character codes to be used in names of language-specific objects):

- Left-to-Right Single-Byte Languages with Latin Lower Case
- Left-to-Right Single-Byte Languages without Latin Lower Case
- Bi-directional Single-Byte Languages without Latin Lower Case
- User-Assigned Languages
- Multiple-Byte Languages

- Double-Byte Languages

Left-to-Right Single-Byte Languages with Latin Lower Case

Code	Language	Character Code in Language-Specific Object Names
1	English	1
2	German	2
3	French	3
4	Spanish	4
5	Italian	5
6	Dutch	6
7	Turkish	7
8	Danish	8
9	Norwegian	9
10	Albanian	A
11	Portuguese	B
12	Chinese Latin (Taiwan)	C
13	Czech	D
14	Slovak	E
15	Finnish	F
16	Hungarian	G
17	Icelandic	H
18	Korean	I
19	Polish	J
20	Romanian	K
21	Swedish	L
22	Croatian	M
23	Catalan	N
24	Basque	O
25	Afrikaans	P

Left-to-Right Single-Byte Languages without Latin Lower Case

Code	Language	Character Code in Language-Specific Object Names
26	Bulgarian	Q
27	Greek	R
28	Japanese (Katakana)	S
29	Russian	T
30	Serbian	U

Bi-directional Single-Byte Languages without Latin Lower Case

Code	Language	Character Code in Language-Specific Object Names
31	Arabic	V
32	Farsi (Iran)	W
33	Hebrew	X
34	Urdu (Pakistan)	Y
35	(reserved for future use)	Z
36	(reserved for future use)	a
37	(reserved for future use)	b
38	(reserved for future use)	c
39	(reserved for future use)	d
40	(reserved for future use)	e

User-Assigned Languages

Code	Language	Character Code in Language-Specific Object Names
41	(free for you to assign a language)	f
42	(free for you to assign a language)	g
43	(free for you to assign a language)	h
44	(free for you to assign a language)	i
45	(free for you to assign a language)	j
46	(free for you to assign a language)	k
47	(free for you to assign a language)	l
48	(free for you to assign a language)	m
49	(free for you to assign a language)	n
50	(free for you to assign a language)	o

Multiple-Byte Languages

Code	Language	Character Code in Language-Specific Object Names
51	Hindi	p
52	Malayan	q
53	Thai	r
54	(reserved for future use)	s
55	(reserved for future use)	t
56	(reserved for future use)	u

Double-Byte Languages

Code	Language	Character Code in Language-Specific Object Names
57	Chinese (People's Republic of China)	v
58	Chinese (Republic of China)	w
59	Japanese (Kanji)	x
60	Korean	y

*NATVERS

Format/length:	A8
Content modifiable:	No

This system variable contains the Natural version (excluding the cumulative fix information), for example: 06.02.01.

The cumulative fix information is contained in the system variable `*PATCH-LEVEL`.

For further information, see *Version* in the *Glossary*.

*NET-USER

Format/length:	A253
Content modifiable:	No

The value of `*NET-USER` contains the complete user ID consisting of the domain name and the actual user ID.

The following considerations apply to a NaturalX server.

When a NaturalX server receives an authenticated request, the user ID of this request is passed to the server and placed into `*NET-USER`. (The DCOM function `CoQueryClientBlanket` is used for this purpose.)

After the NaturalX server has processed the request, `*NET-USER` is reset to the value it contained before the request.

A request which is not authenticated has no effect on `*NET-USER`.

*PARM-USER

Format/length:	A253
Content modifiable:	No

This system variable contains the name of the parameter file currently in use.

*PATCH-LEVEL

Format/length:	A8
Content modifiable:	No

This system variable contains the current cumulative fix number. See also the system variable `*NATVERS` and *Version* in the *Glossary*.

*PID

Format/length:	A32
Content modifiable:	No

This system variable contains the current process ID as a string value.

*SCREEN-IO

Format/length:	L
Content modifiable:	No

This system variable indicates whether a screen I/O is possible or not.

It can contain one of the following values:

TRUE	Screen I/O is possible.
FALSE	Screen I/O is not possible.

In an interactive Natural session, `*SCREEN-IO` is initialized with `TRUE`. In a Natural batch session, `*SCREEN-IO` is initialized with `FALSE` (except for a Natural Development Server).

If Natural was started as a Db2 Stored Procedures server (`*SERVER-TYPE=DB2-SP`) or as RPC server (`*SERVER-TYPE=RPC`) `*SCREEN-IO` is set to `FALSE`.

If Natural was started on a Windows platform as DCOM server (`*SERVER-TYPE=DCOM`), `*SCREEN-IO` is set to `FALSE`, while the server is executing a method called by COM/DCOM.

When `*SCREEN-IO` is set to `FALSE` and a statement which requires user interaction is executed, Natural issues error NAT0723.

*SERVER-TYPE

Format/length:	A32
Content modifiable:	No

This system variable indicates the server type Natural has been started as.

It can contain one of the following values:

DB2-SP	Natural Db2 Stored Procedures server
DCOM	NaturalX DCOM server
DEVELOP	Natural development server
RPC	Natural RPC server
WEBIO	Natural Web I/O Interface server

If Natural is not started as a server, `*SERVER-TYPE` is set to blanks.

Note: `*SERVER-TYPE` refers to Natural as a whole, *not* to the Natural program currently being executed (which may run as a client program or as a server program within a server Natural).

*UI

Format/length:	A16
Content modifiable:	No

This system variable indicates the type of user interface being used:

CHARACTER	Character-oriented user interface.
GUI	Graphical user interface.

*USER

Format/length:	A8
Content modifiable:	No

This system variable contains the user ID as taken from the Natural Security logon.

If the profile parameter `AUTO=ON` (Automatic Logon) is set or if Natural Security is not active, the value of `*USER` is identical to that of `*INIT-USER`.

*USER-NAME

Format/length:	A32
Content modifiable:	No

If Natural Security is installed, this variable contains the name of the user who is currently logged on to Natural.

If Natural Security is not active, the default is `SYSTEM`.

*UUID

Format/length:	B16
Content modifiable:	No

This system variable provides a Universally Unique Identifier (UUID).

The UUID is a B16 unique value generated each time the function is called. Here, the most significant part is stored in `[0]`, and the least significant part is stored in `[15]`.

The `CoCreateGuid()` function is used to generate a UUID. It calls the RPC function `UuidCreate()`, which creates a globally unique 128-bit identifier (GUID).

To a very high degree of certainty, `CoCreateGuid()` returns a unique value. No other invocation, on the same system or on any other system, whether networked or not, should return the same value. A Version 4 (V4) UUID is generated.

8 System Environment-Related System Variables

- *CODEPAGE 62
- *HARDWARE 62
- *HOSTNAME 62
- *INIT-ID 63
- *INIT-PROGRAM 63
- *LOCALE 63
- *MACHINE-CLASS 64
- *OPSYS 64
- *OS 65
- *OSVERS 65
- *TP 65
- *TPSYS 65
- *TPVERS 66
- *WINMGR 66
- *WINMGRVERS 67

*CODEPAGE

Format/length:	A64
Content modifiable:	No

This system variable returns the IANA name of the default code page which is internally used by Natural for conversions to and from Unicode and which is set by the Natural profile parameter `CP`.

Note: `*CODEPAGE` is also the default if a code page is not specified in a `MOVE ENCODED` statement.

Example:

```
windows-1252
```

*HARDWARE

Format/length:	A16
Content modifiable:	No

This system variable contains the name of the hardware platform on which Natural is running. This value is supplied by the operating system.

*HOSTNAME

Format/length:	A64
Content modifiable:	No

The name of the machine Natural runs on.

*INIT-ID

Format/length:	A8
Content modifiable:	No

`*INIT-ID` contains the value `PC_WIN`.

*INIT-PROGRAM

Format/length:	A8
Content modifiable:	No

`*INIT-PROGRAM` contains the value `Natural`.

In a Natural Development Server environment using Com-plete or Com-plete/SMARTS, `*INIT-PROGRAM` is set according to the Com-plete/SMARTS startup option `INSTALLATION`. The default content is `********`.

*LOCALE

Format/length:	A8
Content modifiable:	No

This system variable contains the language and country of the current locale, which specifies the Unicode collation sequence.

Example:

```
en_US
```

*MACHINE-CLASS

Format/length:	A16
Content modifiable:	No

This system variable contains the name of the machine class on which Natural is running.

It can contain one of the following values:

MAINFRAME
PC
UNIX

*OPSYS

Format/length:	A8
Content modifiable:	No

This system variable contains the Natural name of the operating system that is being used.

It can contain one of the following values:

ATT_OSX	MVS/ESA	WNT-X86
AVIION	NCR 3000	
BULL/BOS	OS	
DEC-OSF/	RS_6000	
DPS300	SCO	
DRS 6000	SINIX_52	
FUJI M73	SINIX_54	
HP_HPUX	SUN_SOLA	
MSDOS	SUN_SUNO	
LINUX	WNT-X86	

Note: Instead of `*OPSYS`, it is recommended that the system variables `*MACHINE-CLASS`, `*HARDWARE` and `*OS` be used, as they allow a more precise distinction of the environment in which Natural is running.

*OS

Format/length:	A32
Content modifiable:	No

This system variable contains the name of the operating system under which Natural is running. This value is supplied by the operating system and may be subject to change.

*OSVERS

Format/length:	A16
Content modifiable:	No

This system variable contains the version number of the operating system under which Natural is running. This value is supplied by the operating system and may be subject to change.

*TP

Format/length:	A8
Content modifiable:	No

It contains the name of the TP subsystem under which Natural is running. This value is supplied by the operating system and may be subject to change.

*TPSYS

Format/length:	A8
Content modifiable:	No

This system variable contains the Natural name of the TP monitor or environment that is being used.

It can contain one of the following values:

```
AIM/DC
CICS
```

COMPLETE
IMS TM
OS/400
SERVSTUB (Natural Development Server)
TSO
TSS

On z/OS, *TPSYS will be blank in batch mode.

On Windows and Linux platforms, *TPSYS will be NONE.

*TPVERS

Format/length:	A8
Content modifiable:	No

It contains the version of the TP subsystem under which Natural is running. This value is supplied by the operating system and may be subject to change.

If no TP monitor is used, *TPVERS will be blank.

*WINMGR

Format/length:	A16
Content modifiable:	No

If a graphical user interface is used, this system variable contains the name of the window manager being used (for example, MOTIF or PM).

If a character-oriented user interface is used, *WINMGR will be blank.

The type of user interface is indicated by the value of the system variable *UI.

*WINMGRVERS

Format/length:	A16
Content modifiable:	No

If a graphical user interface is used, this system variable contains the version number of the window manager being used.

If a character-oriented user interface is used, `*WINMGRVERS` will be blank.

The type of user interface is indicated by the value of the system variable `*UI`.

9 XML-Related System Variables

- *PARSE-COL (r) for XML 70
- *PARSE-LEVEL (r) for XML 70
- *PARSE-NAMESPACE-URI (r) for XML 70
- *PARSE-ROW (r) for XML 71
- *PARSE-TYPE (r) for XML 71

These system variables, which are available when using the PARSE XML statement, are only valid in the current loop context.

*PARSE-COL (r) for XML

Format/length:	I4
Content modifiable:	No

This system variable contains the column where the parser is currently working at.

(r) notation after *PARSE-COL is used to indicate the statement label or source-code line number of the PARSE XML statement. If (r) is not specified, *PARSE-COL represents the column where the parser in the currently active processing loop is working at.

*PARSE-LEVEL (r) for XML

Format/length:	I4
Content modifiable:	No

This system variable contains the level of currently nested elements.

(r) notation after *PARSE-LEVEL is used to indicate the statement label or source-code line number of the PARSE XML statement. If (r) is not specified, *PARSE-LEVEL represents the level where the parser in the currently active processing loop is working at.

*PARSE-NAMESPACE-URI (r) for XML

Format/length:	A (dynamic)
Content modifiable:	No

This system variable contains the namespace URI of the current element/attribute, if the element/attributes belong to a namespace. If the NAME (*operand3*) value of the PARSE XML statement is empty, then there is also no namespace and *LENGTH(*PARSE-NAMESPACE-URI) is set to 0.

(r) notation after *PARSE-NAMESPACE-URI is used to indicate the statement label or source-code line number of the PARSE XML statement. If (r) is not specified, *PARSE-NAMESPACE-URI represents the namespace URI of the current element/attribute in the currently active processing loop.

*PARSE-ROW (r) for XML

Format/length:	I4
Content modifiable:	No

This system variable contains the row where the parser is currently working at.

(*r*) notation after *PARSE-ROW is used to indicate the statement label or source-code line number of the PARSE XML statement. If (*r*) is not specified, *PARSE-ROW represents the row where the parser in the currently active processing loop is working at.

*PARSE-TYPE (r) for XML

Format/length:	A1
Content modifiable:	No

This Natural system variable is automatically created for each PARSE XML statement issued.

This system variable contains the type of the delivered data.

(*r*) notation after *PARSE-TYPE is used to indicate the statement label or source-code line number of the PARSE XML statement. If (*r*) is not specified, *PARSE-TYPE represents the type of the delivered data in the currently active processing loop.

Possible values for ASCII-based systems are:

?	Processing instruction (but not first <?XML ... ?>).
!	Comment.
C	CDATA section.
T	Starting tag.
@	Attribute (on z/OS: § or @, depending on session code page and terminal emulation).
/	Closing tag.
$	Parsed data.