software AG

Natural 4 Enhancements Variables
July 24, 2007

Presented by: James Bando
Advisory Systems Engineer
Software AG, Inc.

STAY ON TOP OF TECHNOLOGY

53 74 61 79 20 6f 6e 20 74 6f 70 20 6f 66 20

74 65 63 68 6e 6f 6c 6f 67 79

Size of Variables

- Alphanumeric variable size limit changed from 253 bytes to 1 GB (2**30)

```
1 #ALPHA      (A1073741824)
```

- Binary variable size limit changed from 126 bytes to 1 GB (2*30)

```
1 #BINARY     (B1073741824)
```

- Unicode variables limit is 2**29 characters

```
1 #UNICODE   (U536870912)
```

```
* V4VARSIZ VARIABLES OF LARGE SIZE
DEFINE DATA LOCAL
1 #LARGE (A500) INIT FULL LENGTH <'X'>
END-DEFINE
PRINT #LARGE
END
```

```
Page      1                                                  06-11-14  17:11:02
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXX
```

Size of Data Elements

- ❑Size of single data element (array or indexed group) has been increased from 32 KB to 1 GB

```
* V4ARRSIZ  V4 array greater than 32K
DEFINE DATA LOCAL
1 #A (A70/1:1000)  /* Array 70,000 bytes
END-DEFINE
MOVE ALL 'A' TO #A (1)
MOVE ALL 'B' TO #A (2)
MOVE ALL 'C' TO #A (999)
MOVE ALL 'D' TO #A (01000)
WRITE '   #A(1)' #A (1)
WRITE '   #A(2)' #A (2)
WRITE ' #A(999)' #A (999)
WRITE '#A(1000)' #A (01000)
END
```

```
Page      1                                                  06-11-14  16:54:56

   #A(1) AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
   #A(2) BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
 #A(999) CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
#A(1000) DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
```

Array Sizes

❑Maximum number of occurrences (2**30)

```
* V4ARROCC  V4 OCCURRENCES UP TO 1,073,741,824
DEFINE DATA LOCAL
1 #A (A1/1073741824) /* ARRAY OCCURRENCES 1GB
END-DEFINE
MOVE 'A' TO #A (1000000)
MOVE 'B' TO #A (1000001)
DISPLAY #A (1000000:1000001)
END
```

```
Logon accepted to library YSAJGB.
NEXT V4ARROCC
Page     1                                                   06-11-14  17:00:02
#A
--
A
B
NEXT FIN
NAT9995 Natural session terminated normally.
```

Dynamic Variables

- **Dynamic variables are defined without a length**
  - Alphanumeric: `1 #FULL-NAME   (A) DYNAMIC`
  - Binary: `1 #PICTURE     (B) DYNAMIC`
  - Unicode: `1 #NATIVE-NAME (U) DYNAMIC`
- **Lengths of a dynamic variables change based on the values assigned to them**
- **Many statements and APIs use dynamic variables**
  - Example: Used in REQUEST DOCUMENT statement for received page since the page size is never known until run-time.
- **(NAT414) Natural RPC stubs that contain dynamic variables in their PDAs can be generated**
- **Define in GLOBAL, LOCAL, PARAMETER, INDEPENDENT, CONTEXT, OBJECT data areas**

Dynamic Variables

- **Can be initialized**

```
1 #HOMETOWN  (A) DYNAMIC INIT <'YORKTOWN'>
```

- **Can be named constants**

```
 1 #CORP-NAME  (A) DYNAMIC CONST <'ACME, INC.'>
```

- **Cannot be redefined**
- **Cannot be part of a redefinition**
- ***LENGTH(dyn-var) returns length**
- **Can be used with SUBSTRING()**
  - Accessing beyond *LENGTH results in a run-time error
  - Example

```
IF *LENGTH(#DYNA) >= 1 AND *LENGTH(#DYNB) >=3
   IF SUBSTRING(#DYNA,1,1) = 'X'
     MOVE SUBSTRING(#DYNA,1,1) TO SUBSTRING(#DYNB,3,1)
```

Dynamic Variables

```
* V4DYNVAR    DEMO DYNAMIC VARIABLES
DEFINE DATA LOCAL
1 #FULL-NAME (A) DYNAMIC
1 EMPLOYEES VIEW OF EMPLOYEES
2 NAME
2 FIRST-NAME
2 MIDDLE-I
END-DEFINE
READ (2) EMPLOYEES BY NAME
DISPLAY FIRST-NAME NAME MIDDLE-I
COMPRESS NAME INTO #FULL-NAME
PRINT 'LENGTH:' *LENGTH(#FULL-NAME) '=' #FULL-NAME
COMPRESS #FULL-NAME ',' INTO #FULL-NAME LEAVING NO
PRINT 'LENGTH:' *LENGTH(#FULL-NAME) '=' #FULL-NAME
COMPRESS #FULL-NAME FIRST-NAME MIDDLE-I INTO #FULL-NAME
PRINT 'LENGTH:' *LENGTH(#FULL-NAME) '=' #FULL-NAME
SKIP 1
END
```

```
FIRST-NAME              NAME         MIDDLE-I
-------------------- -------------------- --------
KEPA                 ABELLAN
LENGTH:           7 #FULL-NAME: ABELLAN
LENGTH:           8 #FULL-NAME: ABELLAN,
LENGTH:          13 #FULL-NAME: ABELLAN, KEPA

ROBERT               ACHIESON             W
LENGTH:           8 #FULL-NAME: ACHIESON
LENGTH:           9 #FULL-NAME: ACHIESON,
LENGTH:          18 #FULL-NAME: ACHIESON, ROBERT W
```

Dynamic Variables

- Example: Use a dynamic variable to determine the last non-blank character in a variable
- Traditional

```
* V4DYNTS1 Find last non-blank character, traditional method
1 #I      (I4)
1 #DATA   (A80) INIT <'This is some data'>
1 #LENGTH (I4)
FOR #I = 80 1 -1  /*check each byte until non-blank is found
  IF SUBSTR(#DATA,#I,1) NE ' '
    MOVE #I TO #LENGTH
    ESCAPE BOTTOM IMMEDIATE
  END-IF
END-FOR
```

Average elapsed time: 100 microseconds

- Using a dynamic variable

```
* V4DYNTS2 Find last non-blank character, dynamic variable method
1 #DATA   (A80) INIT <'This is some data'>
1 #DYNAMIC(A) DYNAMIC
1 #LENGTH (I4)
COMPRESS #DATA INTO #DYNAMIC
MOVE *LENGTH(#DYNAMIC) TO #LENGTH
```

Average elapsed time: 24 microseconds

Dynamic Variables

- ❑ (NAT413) Dynamic variables can be used as control break fields and with system functions

```
1 #FULL-NAME    (A) DYNAMIC
         :
FIND EMPLOYEES WITH NAME = 'S' THRU 'S' - H'FF'
END-ALL
SORT   BY NAME FIRST-NAME USING KEYS
  COMPRESS NAME ',' INTO #FULL-NAME LEAVING NO
  COMPRESS #FULL-NAME SUBSTR(FIRST-NAME,1,1) INTO #FULL-NAME
  AT BREAK OF #FULL-NAME
    DISPLAY 'Name' OLD(#FULL-NAME)                 (AL=20)
      'Count' COUNT(#FULL-NAME)
  END-BREAK
END-SORT
```

```
        Name          Count
-------------------- --------
SMITH, L                    1
SMITH, M                    2
SMITH, S                    4
SMITH, T                    2
SMITH, W                    2
SMITH-MANSON, F             1
```

Dynamic Variables

❑Optional Statements to Manipulate Dynamic Variables

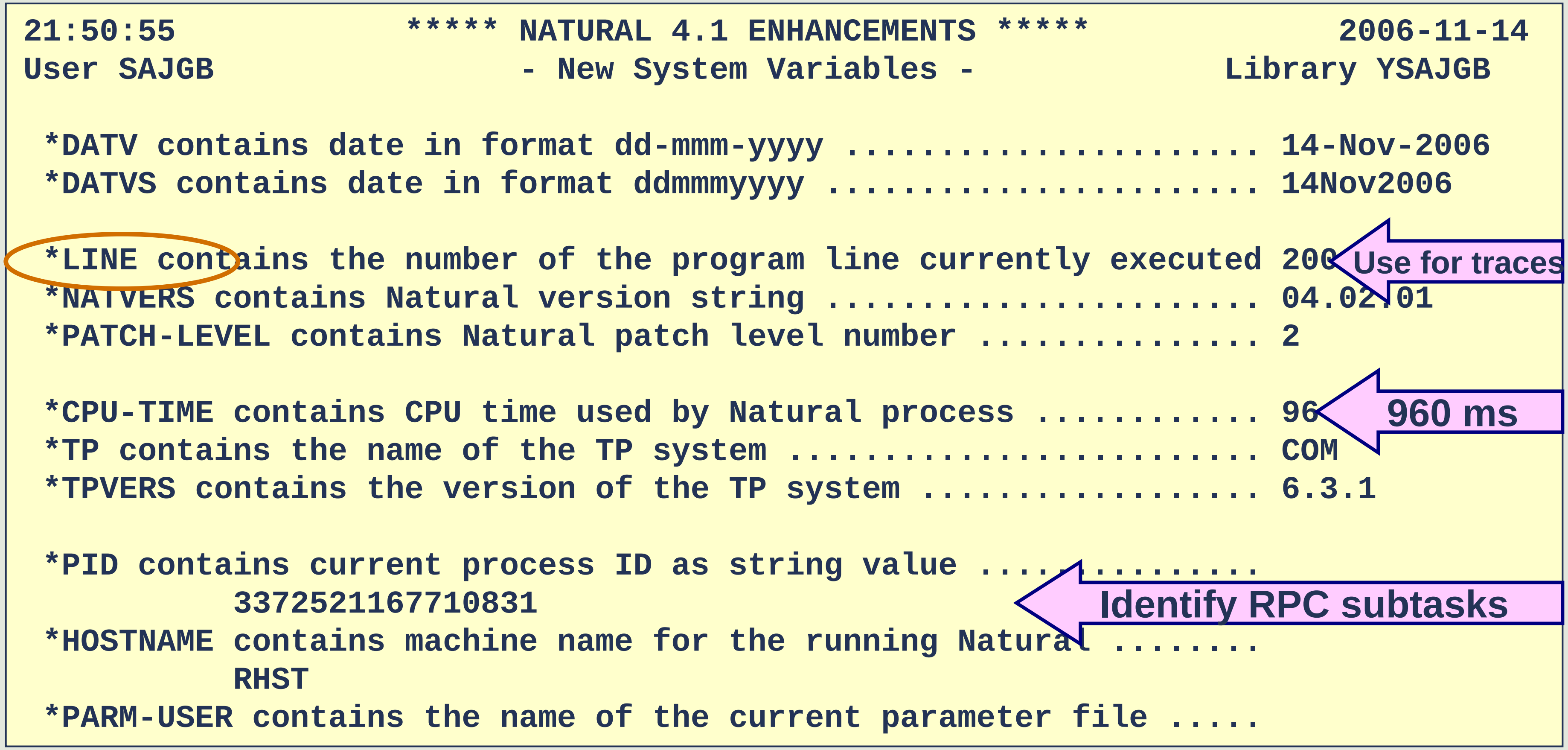
- **`EXPAND DYNAMIC`** *`dynamic-variable`* **`TO`** *`new-size`*
  - Tells Natural, "Get *new-size* of physical memory from the operating system because I expect to use it."
  - Only needed to improve performance
  - Can eliminate many memory requests to the operating system if the dynamic variable size increases a lot
  - *LENGTH is never changed
- **`REDUCE DYNAMIC`** *`dynamic-variable`* **`TO`** *`new-size`*
  - Can give back memory to the operating system when you are done with it
  - Can be used to truncate the length
  - *LENGTH is reduced if *new-size* is less than current *LENGTH
- **`RESIZE DYNAMIC`** *`dynamic-variable`* **`TO`** *`new-size`*
  - Use to either expand or reduce a dynamic variable

Dynamic Variables

❑Expanding, reducing, resizing

```
Page      1                                                  06-12-05   12:00:27

MOVE 'abcdef' TO #DYN
After moving data to #DYN: abcdef     Length           6

EXPAND DYNAMIC #DYN TO 300
After expanding #DYN to 300: abcdef     Length           6

REDUCE DYNAMIC #DYN TO 4
After reducing #DYN to 4: abcd     Length           4

RESIZE DYNAMIC #DYN TO 300
After resizing #DYN to 300: abcd     Length           4

RESIZE DYNAMIC #DYN TO 3
After resizing #DYN to 3: abc     Length           3
```

New System Variables



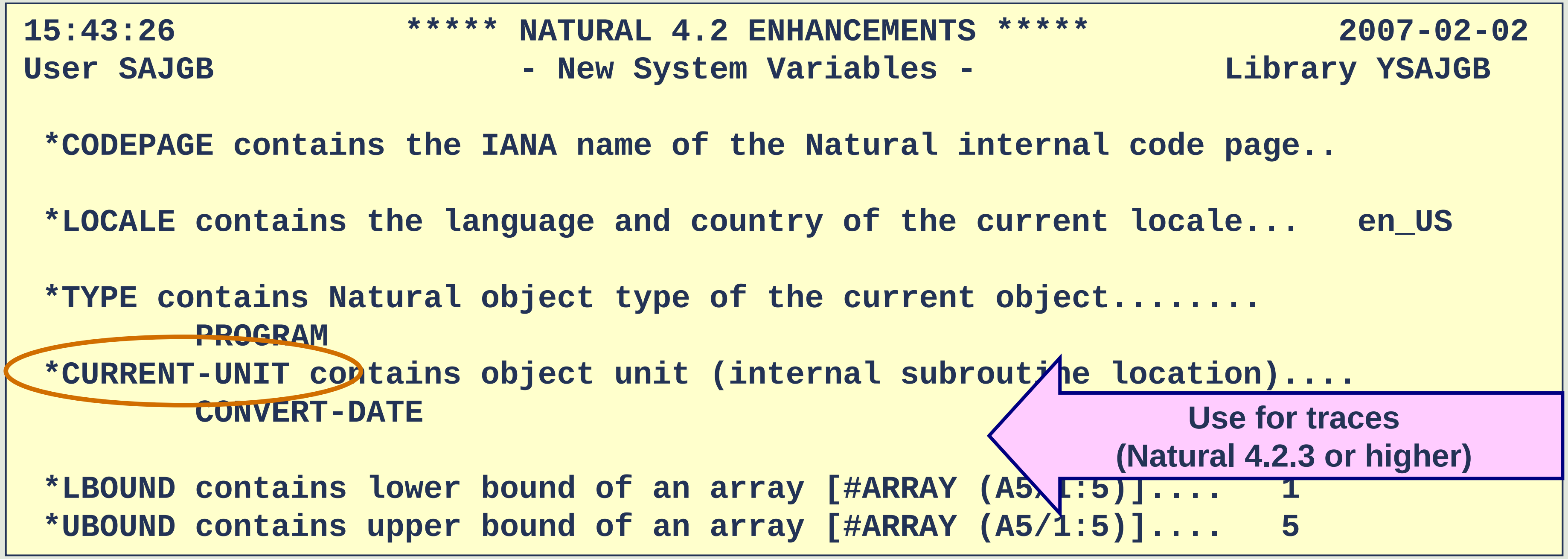
```
21:50:55              ***** NATURAL 4.1 ENHANCEMENTS *****              2006-11-14
User SAJGB                  - New System Variables -                Library YSAJGB

 *DATV contains date in format dd-mmm-yyyy ....................... 14-Nov-2006
 *DATVS contains date in format ddmmmyyyy ........................ 14Nov2006

 *LINE contains the number of the program line currently executed 200
 *NATVERS contains Natural version string ........................ 04.02.01
 *PATCH-LEVEL contains Natural patch level number ................ 2

 *CPU-TIME contains CPU time used by Natural process ............. 96
 *TP contains the name of the TP system .......................... COM
 *TPVERS contains the version of the TP system ................... 6.3.1

 *PID contains current process ID as string value ................
           3372521167710831
 *HOSTNAME contains machine name for the running Natural ........
           RHST
 *PARM-USER contains the name of the current parameter file .....
```

Use for traces

960 ms

Identify RPC subtasks

New System Variables

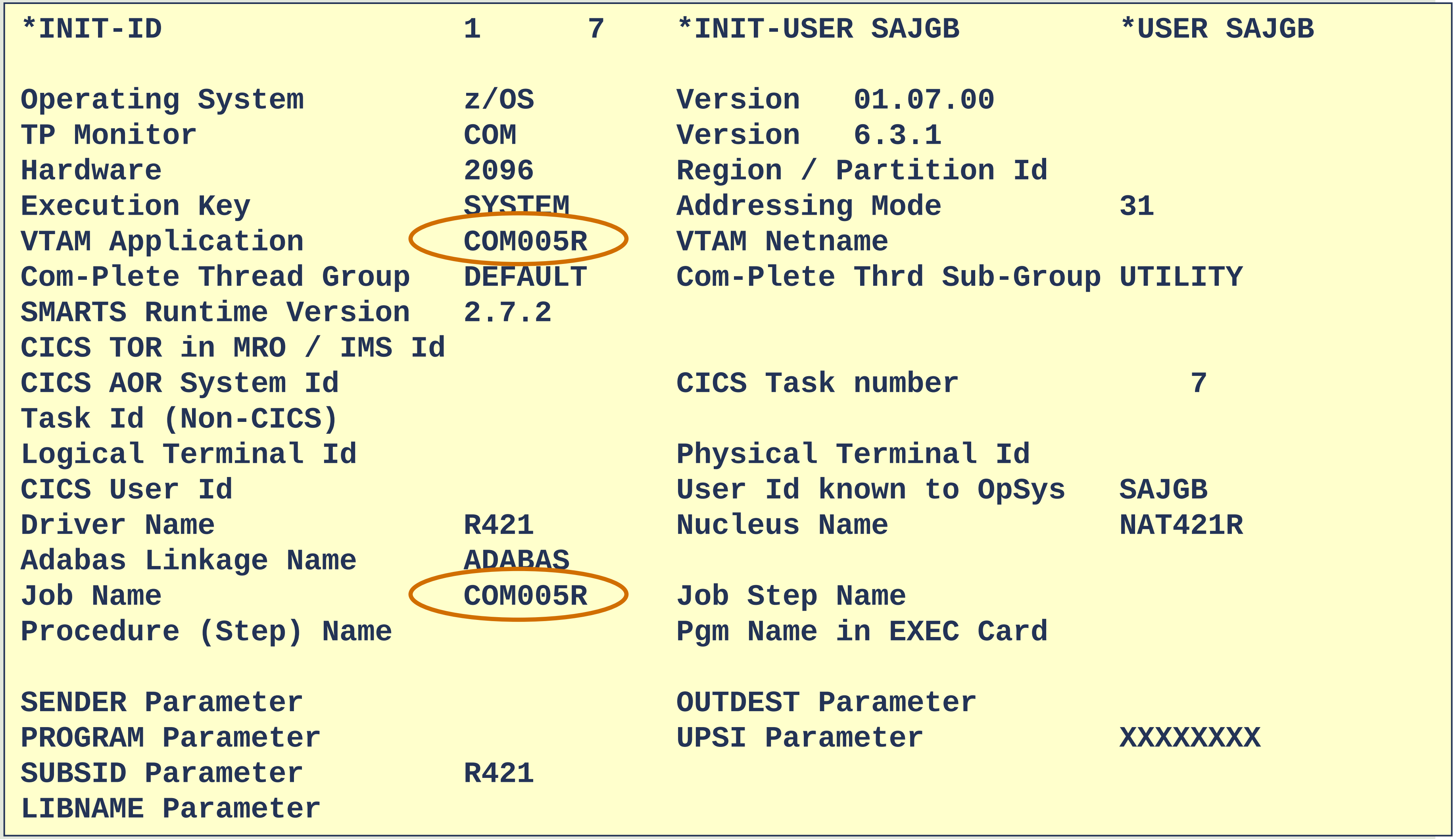


```
15:43:26            ***** NATURAL 4.2 ENHANCEMENTS *****             2007-02-02
User SAJGB                - New System Variables -             Library YSAJGB

 *CODEPAGE contains the IANA name of the Natural internal code page..

 *LOCALE contains the language and country of the current locale...   en_US

 *TYPE contains Natural object type of the current object........
         PROGRAM
 *CURRENT-UNIT contains object unit (internal subroutine location)....
         CONVERT-DATE

 *LBOUND contains lower bound of an array [#ARRAY (A5/1:5)]....   1
 *UBOUND contains upper bound of an array [#ARRAY (A5/1:5)]....   5
```

Use for traces
(Natural 4.2.3 or higher)

System Variables Now on Mainframe

❑System information available using USR4002N



```
*INIT-ID                 1      7    *INIT-USER SAJGB         *USER SAJGB

Operating System         z/OS        Version   01.07.00
TP Monitor               COM         Version   6.3.1
Hardware                 2096        Region / Partition Id
Execution Key            SYSTEM      Addressing Mode          31
VTAM Application         COM005R     VTAM Netname
Com-Plete Thread Group   DEFAULT     Com-Plete Thrd Sub-Group UTILITY
SMARTS Runtime Version   2.7.2
CICS TOR in MRO / IMS Id
CICS AOR System Id                   CICS Task number             7
Task Id (Non-CICS)
Logical Terminal Id                  Physical Terminal Id
CICS User Id                         User Id known to OpSys   SAJGB
Driver Name              R421        Nucleus Name             NAT421R
Adabas Linkage Name      ADABAS
Job Name                 COM005R     Job Step Name
Procedure (Step) Name                Pgm Name in EXEC Card

SENDER Parameter                     OUTDEST Parameter
PROGRAM Parameter                    UPSI Parameter           XXXXXXXX
SUBSID Parameter         R421
LIBNAME Parameter
```

Extensible Arrays (X-Arrays)

- Number of occurrences are set at run-time
- Either the lower-bound or upper-bound of a dimension can be extensible
- Can be passed to a external objects via a PDA
- Extensible bound can be resized in the subprogram, but there are restrictions
- Multi-dimensional array may have a mixture of fixed and extensible bound dimensions
- Groups may be extensible
- Cannot be redefined
- Cannot be contained in a redefinition
- Many new APIs already use X-arrays

Extensible Arrays (X-Arrays)

❑Defining X-Arrays

```
Data definition of new X-arrays (eXtensible arrays)

DEFINE DATA LOCAL
  1 #ARR1  (A05/1:*)            /* unlimited upper bound
  1 #ARR2  (I04/*:5)            /* unlimited lower bound

  1 #ALPHA (A/-3:*) DYNAMIC     /* X-array with dynamic variable

  1 #DIM   (A10/*:2,5:*)        /* 2-dimensional X-array

  1 #GROUP
    2 #ITEM (I2/1:*)            /* X-array contained in a group
END-DEFINE

Note: Unknown bounds cannot be defined for both upper and lower bounds.
      X-arrays cannot be redefined or contained in a redefinition.
      X-arrays cannot be initialized when defined.
```

Extensible Arrays (X-Arrays)

❑Expand (materialize) an X-array before you use it at run-time

```
DEFINE DATA LOCAL
  1 #MAX        (I4) CONST <5>   /* Constant for maximum upper bound
  1 #ARRAY      (A05/*:#MAX)     /* undefined lower bound
END-DEFINE

Before referencing an X-array, specify the bounds of the X-array.
New occurrences are initialized as appropriate.
EXPAND OCCURRENCES OF ARRAY #ARRAY TO (1:#MAX)   /* new bounds (1:5)
            ---- or ----
EXPAND OCCURRENCES OF ARRAY #ARRAY TO (1:5)  /* new bounds (1:5)
            ---- or ----
EXPAND OCCURRENCES OF ARRAY #ARRAY TO (1:*)  /* asterisk keeps a bound the same

EXPAND OCCURRENCES OF ARRAY #ARRAY TO (-3:*)  /* new bounds (-3:5)

MOVE 'Hello' TO #ARRAY(-2)                    /* X-array can be referenced
```

Extensible Arrays (X-Arrays)

❑Changing Occurrences of X-Arrays

```
More ways to change the bounds of X-arrays.       `

Expand and reset all occurrences to its default value.
EXPAND AND RESET OCCURRENCES OF ARRAY #ARRAY TO (-4:5)

Reduce already specified bounds.
REDUCE OCCURRENCES OF ARRAY #ARRAY TO (3:*)

Resize current bounds.
RESIZE OCCURRENCES OF ARRAY #ARRAY TO (-2:5)

Resize and reset all occurrences to its default value.
RESIZE AND RESET OCCURRENCES OF ARRAY #ARRAY TO (0:*)

Unmaterialize X-array (set back to "no occurrences")
REDUCE ARRAY #ARRAY TO 0          /*cannot use RESIZE to unmaterialize
```

Extensible Arrays (X-Arrays)

❑Retrieve Information About Fixed and X-Arrays

```
Get information about arrays with *OCCURRENCE, *LBOUND, and *UBOUND.

DEFINE DATA LOCAL
  1 #VAR1   (A20/1:*)     /* 1-Dimension X-array
  1 #VAR2   (I04/*:3,2:*) /* 2-Dimension X-array
  1 #COUNTS (P9/1:10)     /* Fixed array
END-DEFINE
*LBOUND (#COUNTS):    1       *UBOUND (#COUNTS):   10     /* Fixed array

*OCCURRENCE (#VAR1):  0       /*One way to tell if XARRAY is "unmaterialized"

EXPAND OCCURRENCES OF ARRAY #VAR1 TO (1:4)
EXPAND OCCURRENCES OF ARRAY #VAR2 TO (-1:3,2:8)

*LBOUND (#VAR1):      1       *UBOUND (#VAR1):      4     /* 1 dim X-array
*LBOUND (#VAR2,1):   -1       *UBOUND (#VAR2,1):    3     /* 1st dimension
*LBOUND (#VAR2,2):    2       *UBOUND (#VAR2,2):    8     /* 2nd dimension
```

Extensible Arrays (X-Arrays)

❑Example: Expand X-Array Based on Data From File

```
DEFINE DATA LOCAL
1 PRODUCTS VIEW OF PRODUCTS
  2 PRODUCT-CODE
  2 PRODUCT-DESC
1 #PRODUCTS (1:*)        /*Don't need to know how many products
  2 #PRODUCT-CODE (A8)
  2 #PRODUCT-DESC (A20)
1 #I (I4) INIT <1>       /*Start out with 1 occurrence
END-DEFINE
READ PRODUCTS       /*Read products from file to build array
  IF *OCCURRENCE(#PRODUCT-CODE) > 0         /*Materialized X-array,
    COMPUTE #I = *UBOUND(#PRODUCT-CODE) + 1 /*so Add 1 more occurrence
  END-IF
  EXPAND ARRAY #PRODUCTS TO (1:#I)          /*Expand X-array
  MOVE PRODUCT-CODE TO #PRODUCT-CODE(#I)    /*Fill array from file
  MOVE PRODUCT-DESC TO #PRODUCT-DESC(#I)    /*Fill array from file
END-READ
IF *OCCURRENCE(#PRODUCT-CODE) > 0 /* If at least 1 product,
  FOR #I 1 *UBOUND(#PRODUCT-CODE) /* display the array of products
    DISPLAY #PRODUCT-CODE(#I) #PRODUCT-DESC(#I) /*All products in X-Array
  END-FOR
END-IF
```

Longer Constants

- ❑Alphanumeric and hexadecimal constants increased from 253 characters to 2**30 characters

```
* V42CONS  Longer constants in V42
DEFINE DATA LOCAL
1 #PARGRAPH1 (A327) INIT <
  'We the People of the United States, in Order to fo'
 -'rm a more perfect Union, establish Justice, insure'
 -' domestic Tranquility, provide for the common defe'
 -'nce, promote the general Welfare, and secure the B'
 -'lessings of Liberty to ourselves and our Posterity'
 -', do ordain and establish this Constitution for th'
 -'e United States of America.'
 >
END-DEFINE
PRINT #PARGRAPH1
```

Longer Constants

❑Hexadecimal Constant > 253 Bytes

```
MOVE
 H'E68540A3888540D7859697938540968640A3888540E49589A3'
-H'858440E2A381A385A26B40899540D69984859940A396408696'
-H'99944081409496998540978599868583A340E4958996956B40'
-H'85A2A381829389A28840D1A4A2A38983856B408995A2A49985'
-H'4084969485A2A3898340E399819598A4899389A3A86B409799'
-H'96A589848540869699 40A38885408396949496954084858685'
-H'9583856B40979996949 6A38540A38885408785958599819340'
-H'E68593868199856B4081958440A28583A4998540A3888540C2'
-H'9385A2A2899587A240968640D389828599A3A840A3964096A4'
-H'99A28593A585A2408195844096A49940D796A2A3859989A3A8'
-H'6B40849640969984818995408195844085A2A381829389A288'
-H'40A38889A240C39695A2A389A3A4A38996954086969940A388'
-H'8540E49589A3858440E2A381A385A240968640C19485998983'
-H'814B'
TO #PARGRAPH1
PRINT #PARGRAPH1
END
```

Systems Engineering Services Offerings

- ❑ **Some Mentoring Workshops**
  - ➢ IBM Assembler Programming for Administrators (5 days)
  - ➢ SuperNatural: A Practical Guide (4 days)
  - ➢ Natural for Unix Internals (3 days)
  - ➢ A Hands-On Introduction to Linux for the Unix Neophyte (2 days)
  - ➢ Tuning Natural for DB2 Applications (2 days)
  - ➢ Using Natural for DB2 Tools (1 day)
- ❑ **Adabas and/or Natural Performance and Tuning (duration varies)**
- ❑ **Temporary DBA Assistance (duration varies)**
- ❑ **For more mentoring workshops and information, go to ...**
  http://www.softwareagusa.com/webedreg/mentoring/index.asp

Systems Engineering Services Offerings

❑ **State Agency**

- ➢ Challenge: State agency spent 21 man-days per month producing invoices from reports and sending them to vendors.
- ➢ Solution: Three-day SES mentoring workshop taught them how to write queries to download invoice data directly to PC spreadsheet.
- ➢ Benefit: Invoices are produced and sent to vendors in 1.5 man-days.

❑ **Services Company**

- ➢ Challenge: Problems with customer bill production.
- ➢ Solution: One-day SES mentoring workshop showed them how to leverage the systems they already had in place and automate the manual processes.
- ➢ Benefits: Significantly reduced the time and money spent for bill creation and mailing. Bill format and bill accuracy greatly improved. Reduced error handling and customer complaints.

Questions and Answers