How to use time and date

BY SAGEDEV · PUBLISHED FEBRUARY 20, 2018 · UPDATED SEPTEMBER 20, 2018

Let’s see today how to use time and date in Sage X3.

To get the current date in a Date variable, use the date$ function.

In the same way to have current date and time in a Datetime variable, use the datetime$ function (only from V7 release).

The established form to represent date and time with a string is:

YYYY-MM-DDThh:mm:ssZ #attention to the letters T and Z

Typically date and time have “YYYY-MM-DDThh:mm:ss” format

```
num$(date$)              #To get today's date string
num$(time$)              #To get the current time as a string
num$(datetime$)          #To get today's date and time as a string
gdat$(DAY,MONTH,YEAR)    #To get a date of a day, month and year
gdatetime$(str)          #To get a Datetime variable from a string
year(date)               #To get the year of a date
month(date)              #To get the month number of a date
month$(date)             #To get the month name of a date in the current language
day(date)                #To get the day number of the month (1-31)
day$(date)               #To get the name of the day of the week
dayn(date)               #To get the day number of the week (1=lunedì, 7=domenica)
week(date)               #To get the week number (1-53)
nday(date)               #To get the number of days passed since January, 1600\
nday$(NUM)               #To get a date from 1 Jan + a NUM of days
eomonth(date)            #To get the last day of a month
```

You can also perform simple operations:

Local Date DATE

DATE=[1/2/2015]+5 #DATE = [06/02/2015]

DATE=[1/2/2015]-1 #DATE = [31/01/2015]

An example:

Local date LASTDATE

LASTDATE = eomonth([1/2/2015])

Infbox num$(LASTDATE) #will display the date [28/02/2015]

To obtain a date as a string with a determined format:

format$("D:YYYYMMDD[_]hhmmss",date$)

December 31, 2018 at 12.30 and 56 seconds will correspond to “20181231_123056”.

Note that square brackets are used to insert a space or other constant elements:

FORMAT = "DD[ ]MM[ ]YY"

To get the date and time as a formatted string, you can also use AFNC.FDH.

With this call it uses the formatting contained in the global variable GFMDAT.

Suppose that today is January 19, 2018, and GFMDAT contains “DD [/] MM [/] YY”

func AFNC.FDH(date$,"121531") # restituisce "19/01/18 12:15:31"

func AFNC.FDH(date$,"") # returns "19/01/18 17:08:27"

func AFNC.FDH(date$,"15:32") # returns "19/01/18 15:32:00"

func AFNC.FDH(date$,"15") # returns "19/01/18 15:00:00"

func AFNC.FDH(date$,"15:") # returns "19/01/18 15:00:00"

func AFNC.FDH(date$,"error-string") # returns "19/01/18 "

func AFNC.FDH(date$+1,"x") # returns "20/01/18 "

It should be remembered that there are several global variables containing formatting strings,

all preset according to the predefined parameters.

They are useful to maintain a standard that adapts according to various users, sites, etc.

The order day-month-year of these formats depends exactly on the location and user settings.

The variables are these:

GFMDAT = standard date, with 2 caratteri for the day, 2 for the month , 2 for the year

GFMDAT4 = date with year of 4-digit

GFMDAT3 = date with month of 3 letters

GFMDAT9 = date with month written in letters

GFMMOI = month-year

GFMMOI4 = month-year with year of 4-digit

GFMJOU = day-month

Example :

```
######################################################## OUVRE

$OUVRE
Gosub OUVRE_FIC From ZIMPAUTO

If clalev([F:ABR])=0 : Local File "@X3.ABATRQT" [ABR] : Endif
Return

######################################################## DEBUT

$DEBUT
Raz [M:ZIA] #Empty the field [M:ZIA]
[M:ZIA]ZDATFIN = date$ #Todays's date
[M:ZIA]ZDATDEB = date$-GZIAFILNBJ #Todays's date – GZIAFILNBJ days
Affzo [M:ZIA]20 #Refresh
Gosub AFFICHE_TAB_MODELE
Gosub AFFICHE_TAB_RESULTAT
Gosub AFFICHE_REQUETE
Return
```

To go in a new line in the middle of a adonix code instruction

BY SAGEDEV · PUBLISHED JUNE 12, 2018 · UPDATED SEPTEMBER 20, 2018

To go in a new line in the middle of a code instruction just put & as first character (without any spaces before) of the new line.

Here an example:

```
    [M:YTX]YFIELD1(nolign-1) = [F:YTM]YFIELD2 + " "
& + " " +func LIB.FUNCTION_EXAMPLE2(NUMBER1,NUMBER2,NUMBER3,NUMBER4)
& + func LIB.FUNCTION_EXAMPLE("PARAM1","PARAM2","PARAM3","PARAM4")
```

In Eclipse this wrapping is automatically made when you exceed the max length accepted.

As? Just compiling with pressing F7 key!

Loops and conditions

BY ENRICO LIDACCI · PUBLISHED FEBRUARY 6, 2018 · UPDATED SEPTEMBER 24, 2018

Conditions

If Then Else

The following scripts are equivalent:

```
If I=1 Then J=2 Else J=3 : Endif
```

```
If I=1  :  J=2 Else J=3 : Endif  # the ":" allow to write one instruction after the other
```

```
If I=1
 J=2
Else J=3
Endif
```

In the case of multiple conditions placed in and, the evaluations are made in the written order, and the program stops at the first false condition found; in the case below, if ConditionA is false the other two conditions are not evaluated (therefore, in case of instructions, they are not executed):

```
If CondizioneA & CondizioneB and CondizioneC
```

Examples:

<> mathematical operator means “different from”;for example if you want check if a string is not null:

```
If [M:YCE1]YSEZFIL <> ""
Endif
```

The exclamation mark corresponds to not

```
If !clalev([F:YADI])
 Local File ATABDIV [F:YADI]
Endif
```

Inline conditions

To put an “inline” condition, for example in the selection fields of an object, you can use stringstring$

or to make a value appear only for a certain transaction: string$(GFLAG='YOFQ',[F:POH]YPOHORE)

Loops

You can use two forms of loops,

For…Next

and

While…Wend

with different syntaxes based on the cases.

See below some examples

Loop on numerical variable

The step if not specified is 1.

For I = 1 To 13 Step 2.5 : Infbox num$(I) : Next I : Infbox 'FIN='+num$(I) # displays : 1 3.5 6 8.5 11 FIN=13.5

For I = 15 To 20 : Infbox I : Next I : Infbox 'FIN='+num$(I) # displays : 15 16 17 18 19 20 FIN=21

For I = 15 To 11 Step -1 : Infbox I : Next I : Infbox 'FIN='+num$(I) # displasy : 15 14 13 12 11 FIN=10

Loop on alfanumerical variable

For CHN='A','EF','X','ZZZ' : Infbox CHN : Next CHN : Infbox 'FIN='-CHN # displays : A EF X ZZZ FIN=ZZZ

```
For USER="John","Matteo","Lucas"
  Call PARAMUSER(USER,OTHER) From YLIB
  ...
Next USER
```

Loop on table

```
For [F:ITM] Where [F:ITM]YCAT='COD'
 ...
```

```
Next
While INDEX>0
 ...
Wend
```

Break

It is possible to use the break keyword to interrupt the cycle:

```
For [F:ITM]
   ...
   Break
Next
```

Multiple conditions

The Case statement can be used to manage multiple conditions:

```
Case YI
When 1 : Infbox '1'
When 2
   YFILE = "C:\TEMP\test2.pdf"
   Infbox '2'
When 3
   Local Char V2(250)
   YFILE = "C:\TEMP\test3.pdf"
   Infbox '3' - YFILE - V2
When Default
Endcase
```

REPLACE function for strings in Sage X3 adonix

BY SAGEDEV · PUBLISHED DECEMBER 4, 2017 · UPDATED SEPTEMBER 20, 2018

Adonix does not provide an instruction for replacing a string part.

The only existent instruction is

ctrans(STR,"AB","C")

that substitutes into the STR string all occurrences of ‘A’ and ‘B’ letters into ‘C’ letter.

Note that it does not replace string “AB”, but all occurrences of ‘A’ letter and all occurrences of ‘B’ letter.

We remedy this miss implementing a REPLACE function and a his wrapper too,

so we can manage with the same code both simple strings both clobs.

```
#
#File YSAGEDEV
#

################################################################################
# Author: SageDev.it
# It substitutes all occurrences of OLD with NEW
# STR is a clob
################################################################################
Funprog REPLACE(STR, OLD, NEW)
   Value Clbfile STR()
   Value Char OLD()
   Value Char NEW()
   Local Integer LENOLD
```

```
   Local Integer LENNEW

   #length of string to substitute
   LENOLD= len(OLD)
   #length of string that will substitute OLD string
   LENNEW= len(NEW)

   If LENOLD<=0 or LENOLD>len(STR)
      #the string to substitute is empty or it is longer than initial string, so we do nothing
      End STR
   Endif

   Local Integer INDEX
   Local Integer INDEXSTART

   #We search the string to substitute
   INDEX=instr(1,STR,OLD)

   While INDEX>0
      #INDEX>0 this means that we have found an occurrence of OLD
      #We substitute OLD occurrence with the NEW string
      STR =left$(STR, INDEX-1) + NEW + right$(STR, INDEX+LENOLD)

      #We calc the index to which the NEW string just inserted ends
      INDEXSTART=INDEX+LENNEW

      #We search the string to substitute starting from the end of last found occurrence
      INDEX=instr(INDEXSTART,STR,OLD)
   Wend
End STR
################################################################################
```

```
################################################################################
# Author: SageDev.it
# Wrapper of REPLACE(STR, OLD, NEW)
# It substitutes all OLD occurrence with NEW
# the STR parameter is a CHAR(), not a clob
################################################################################
Funprog REPLACESTR(STR, OLD, NEW)
   Value Char STR()
   Value Char OLD()
   Value Char NEW()

   Local Char RET(250)
   Local Clbfile STRCLB(1)
   STRCLB = STR
   RET = func YSAGEDEV.REPLACE(STRCLB, OLD, NEW)
End RET
################################################################################
```

This code is part of YSAGEDEV library

I hope it helps you!

Trace

You can only have one active trace so to create a new trace you should check there is no active trace.

The good way to manage a trace :

```
################################################################################


local integer CheckTraceCreated
If (GTRACE ="") then
        CheckTraceCreated = 1
        call OUVRE_TRACE("NameOfTheTrace") from LECFIC
endif
call ECR_TRACE("Your own debug information", 0) from GESECRAN
 If (CheckTraceCreated=1) then
        call FERME_TRACE from LECFIC
        call LEC_TRACE from LECFIC
endif
end
################################################################################
```

Tables Join in Sage X3 adonix with Link instruction

BY SAGEDEV · PUBLISHED SEPTEMBER 12, 2018 · UPDATED SEPTEMBER 20, 2018

The instruction that Sage X3 makes available to make a join between two tables is Link.

Here we see how to create a join between the TAB1 table and the TAB2 table;

TAB1 is the table with more detail,

The link is based on the TAB2 table.

Key index is KEY0, it's made up of 2 fields which in this case correspond to the fields FIELD1TAB2 and FIELD2TAB2 of TAB2:

Link [F:TAB1] With [F:TAB2]KEY0~=FIELD1TAB2;FIELD2TAB2 As [YJOI]

& Where [F:TAB1]FIELD1=VALUE1 and [F:TAB1]FIELD2=VALUE2

& Order By [F:TAB1]FIELD1 Asc;[F:TAB1]FIELD2 Asc

Attention: the join statement must be written all on one line, or alternatively you can use the character & as seen here.

The keyword that defines the link between the two tables is With that corresponds with "ON" in an SQL statement:

SELECT * FROM TABA JOIN TABB JOIN ON TABA.FIELDA = TABB.FIELDB

To express a condition in FULL JOIN, use the ~= operator (tilde equal)

To express a condition in LEFT JOIN use the = operator (equal)

Before joining you must declare / open the tables you want to use,

and it is possible to declare the same table twice, with different abbreviations.

For example, the detail table is TABDETAIL and the and the table to be joined is TAB2:

this is the syntax (as already said everything goes on the same line, in brackets the optional parts):

LINK TABDETAIL

With CONDITION_JOIN[,CONDITION_JOIN]...[,CONDITION_JOIN]

As [JOIN_NAME]

[Where CONDITION_WHERE]

[Order By EXPRESSION_ORDER]

CONDITION_JOIN

[TAB2]KEY_NAME ~= EXPRESSION_LIST #used = with the LEFT JOIN

[TAB2]KEY_NAME(INDEX_VALUE) ~= EXPRESSION_LIST #used = with the LEFT JOIN

There must be at least one CONDITION_JOIN and maximum 11.

Up to 12 tables can be put in JOIN.

KEY_NAME is the name of a table index, so to speak those defined in the “Index” tab in the GESATB function.

As we can see, the join condition is conditioned to the use of fields that belong to at least one table index,

therefore you can not set a join on any field.

JOIN_NAME

In practice we give a name to the join class thus created, which can be used with a statement;

for example a FOR statment:

For [JOIN_NAME]

...

Next

Then any WHERE conditions and any sorting with ORDER BY must be added.

Let’s see some examples.

Inner join example

We try to recover the tax code of customers whose code starts with ‘AAA’.

We retrieve the tax code from the CRN field of the BPs table.

Not all BPs are customers, so we set up an inner join to get only records that match the BPCUSTOMER table.

```
If !clalev([F:BPR]) : Local File BPARTNER [BPR] : Endif
If !clalev([F:BPC]) : Local File BPCUSTOMER [BPC] : Endif

Link [F:BPR] With [F:BPC]BPC0~=BPRNUM As [JOIN]
& Where left$([F:BPR]BPRNUM,5)='A0001'

For [JOIN]
Infbox [F:BPR]BPRNUM-'-'-[F:BPR]CRN
Next
```

If in the join condition remove the tilde ~ all BPs will be displayed, even those who are not customers.

Left join example with link one-to-many

We look for all the articles with their articles-site.

An article can have multiple matches in the table of articles-site (one to many relationship).

However, we also want to display all the articles that do not have any site articles.

```
If !clalev([F:ITM]) : Local File ITMMASTER [ITM] : Endif
If !clalev([F:ITF]) : Local File ITMFACILIT [ITF] : Endif

Link [F:ITM] With [F:ITF]ITF0(1)=ITMREF As [JOIN]
& Where left$([F:ITM]ITMREF,7)='ABCDFEG'

For [JOIN]
  Infbox [F:ITM]ITMREF-[F:ITF]STOFCY
Next
```

In this way, an article without an article-site will still be displayed.

Note that in this case we used the syntax [TAB2] KEY_NAME (INDEX_VALUE):

in fact we have set the join only on the first field of the ITF0 key which is as follows:

ITF0 = & gt; ITMREF STOFCY +

The problem is that the link statement requires the use of a key in the join condition,

and you have to make do with the keys present or create a new one.

Transaction

For any script changing fields value in database using write / rewrite / delete, you should encapsulate change in a transaction block.

At the start declare a variable to store if there is any error

```
Local Integer WERR : WERR = 0
```

Declare a transaction on the table you wish to modify

```
Trbegin [F:Table1],[F:Table2]  #Declare tables impacted
```

After each write / rewrite / delete check if everything is ok and store the result in case of error

```
If(fstat<>0)Then
  WERR = 1
Endif
```

At the end of you code commit or rollback depending the value of the result

```
If(WERR=0)Then
  Commit
  Call MESSAGE("READ/WRITE/REWRITE/DELETE OK !") From GESECRAN
Else
  Rollback
  Call ERREUR("ERROR [" + num$(WERR) + "]") From GESECRAN
Endif
```

If any error occurs variable WERR will have the value 1 so at the end there is an error, and no commit should be done. Of course you could adjust this code to your need.

Example :

```
################################################################################
$OK
#On va sauvegarder les lignes du tableau
#L'opération va se faire à l'intérieure d'une transaction
Local Integer WERR : WERR = 0
```

```
Local Integer WNUMLIG : WNUMLIG = 0
Trbegin [F:ZSREP]
#On commence par supprimer toute les lignes de la table concernant l'article de prestation
Delete [F:ZSREP] Where [F:ZSREP]ZARNUM=[M:ZDSREP]ZARNUM and
[F:ZSREP]ZCODCHARG=[M:ZDSREP]ZCODCHARG and
[F:ZSREP]ZCODCLTCHARG=[M:ZDSREP]ZCODCLTCHARG
If(fstat=0)Then
  #On va maintenant parcourir le tableau pour sauvegarder les lignes
  For WNUMLIG=0 To [M:ZDSREP]NBLIG-1
    [F:ZSREP]ZARNUM       = [M:ZDSREP]ZARNUM
    [F:ZSREP]ZCODCHARG    = [M:ZDSREP]ZCODCHARG
    [F:ZSREP]ZCODCLTCHARG = [M:ZDSREP]ZCODCLTCHARG
    [F:ZSREP]ZSITE        = [M:ZDSREP]ZSITE(WNUMLIG)
    [F:ZSREP]ZTARIF       = [M:ZDSREP]ZTARIF(WNUMLIG)
    [F:ZSREP]ZCLTFOU      = [M:ZDSREP]ZCLTFOU(WNUMLIG)
    [F:ZSREP]ZQTYMIN      = [M:ZDSREP]ZQTYMIN(WNUMLIG)
    [F:ZSREP]ZQTYMAX      = [M:ZDSREP]ZQTYMAX(WNUMLIG)
    Write[F:ZSREP]
    If(fstat<>0)Then
      WERR = 1
      Break
    Endif
  Next
Else
  WERR = 1
Endif
If(WERR=0)Then
  Commit
Else
  Rollback
  Call ERREUR("Impossible de sauvegarder les lignes !!") From GESECRAN
Endif
```

Return

##

Manipulating Table

The L4G code can be used to modify behavior in the ERP or perform an independent process. Within the ERP behavior, tables are directly accessible and usable, but not all of them. If there is a need to add tables, they must be declared as follows:

"If clalev([F:ABBREVIATION])=0: Local File Table Name [ABBREVIATION]: Endif".

At the end of the code, close the tables that were opened:

"If clalev([F:ABBREVIATION])<>0: Close File [F:ABBREVIATION]: Endif".

It is customary to assign the table name to the variable, but as always in L4G, the table's abbreviation must be specified.

Remember, if you need to read, write, rewrite, delete a table you should open and close the table. Moreover use transaction for any modification.

Example :

```
#############################################################################
Subprog APQTY(AMTLOC1,ZMKSTAT)
Variable Decimal AMTLOC1
Variable Shortint ZMKSTAT

# si on est sur les commandes EPALIS, on ne fait rien
If(GFONCTION="GESSOH" and GFLAG="EPI")Then
   End
Endif

# Si Webservices, on ne fait rien
If GWEBSERV : End : Endif

If clalev([F:ZAUS])=0 : Local File AUTILIS [F:ZAUS] : Endif
```

```
Read [F:ZAUS]CODUSR=GUSER
If fstat
   GMESSAGE="Erreur de lecture de la fiche Utilisateur pour controle prix de vente."
   ZMKSTAT=2
Elsif [F:ZAUS]ZCTLPRIV=2
   If AMTLOC1=0
      If GZCTRLGROPRI<>0
         AMTLOC1 = GZCTRLGROPRI
      Endif
   Endif
Endif

Close Local File [F:ZAUS]

End
##############################################################################
```