

This small but powerful module enables the use of custom variables -kind of LVars- for storing and retrieving data across the entire gaugeset of an aircraft panel. Like Lvars, this variables can have any name, can hold numeric values of any kind and, unlike LVars, can also handle string values. Besides, they can handle *\*standard\* multidimension array-style manipulation of data*, a powerful tool that is not possible to work with through the native LVarset unless a complex group of macros is suited for the task.

XMLVARS supports storing and retrieving a total of 1024 different variables, with names up to 64 char, numeric values of type FLOAT64 and string values up to 128 char. It can minimize the use of repetitive code in complex gauges like FMS, Navigation Display, TCAS and similar as one of its strongest points is the ability to maintain variable values across the different gauges, like Lvars do (and most gpsvars don't). However, unlike LVars, those values still remain even after an aircraft reload, aircraft change or flight reset; something that could be useful for debugging purposes. In this case, making a RELOAD\_USER\_AIRCRAFT in FSX equals to make RELOAD\_PANELS in FS9 (a powerful event for developers that was lost in FSX).

XMLVARS was tested in FSX and FS9 (FS2004) under Windows Vista and Windows 7 64 bits.

## **Installation in FSX**

First, copy **XMLVars.dll** directly in FSX main folder. Next, locate `dll.xml` file which should be in:

`C:\Users\username\AppData\Roaming\Microsoft\FSX\ path`

or

`C:\Documents and Settings\username\Application Data\Microsoft\FSX\ path`

Edit the file and add the following within `<SimBase.Document>`

```
<Launch.Addon>
  <Name>XMLVars</Name>
  <Disabled>False</Disabled>
  <ManualLoad>False</ManualLoad>
  <Path>XMLVars.dll</Path>
  <DllStartName>module_init</DllStartName>
  <DllStopName>module_deinit</DllStopName>
</Launch.Addon>
```

Finally launch FSX; if the above steps were well done a couple of windows asking for permission to install XMLVars.DLL will show; accept all for the module to remain installed.

## **Installation in FS9 (FS2004)**

Copy **XMLVars9.dll** directly in FS9 (FS2004) Modules folder.

## **XMLVARS constant identifiers**

A group of *constant identifiers* are used to communicate between an XML gauge and the module's internal database. They are the same for both FS9 and FSX versions:

**(C:XMLVARS:StoreVarName,string)** - Write only

This identifier is used to store the name of a custom variable in a record of the database.

For example, 'MyCustomVar' (>C:XMLVARS:StoreVarName,string) will search the database for a record ID equal to 'MyCustomVar'; if found then the index to that record is set in the module, otherwise the var name is added to the database and the corresponding index value is set in the module. A variable name may contain any alphanumeric character (case insensitive) except '"' (single quote) and '"' (double quote); char ">" and spaces are ignored. So, 'MyCustomVar' and '>MYCUSTOMVAR' are considered the same variable name.

**(C:XMLVARS:SearchVarName,string)** - Write only

This identifier is used to search for the name of a custom variable in the database.

For example, 'MyCustomVar' (>C:XMLVARS:SearchVarName,string) will search the database in a similar way as in the above example, but in this case if not found the index will point to EOF.

**(C:XMLVARS:NumberValue,number)** - Read/Write

**(C:XMLVARS:StringValue,string)** - Read/Write

These identifiers are used to store/retrieve the value of a custom var in/from a record of the database.

(C:XMLVARS:NumberValue,number)

holds numeric values and

(C:XMLVARS:StringValue,string)

strings of up to 128 alphanumeric chars (including spaces and excluding single and double quotes)

For example,

(A:FLAPS HANDLE POSITION,percent) (>C:XMLVARS:NumberValue,number)

will store the content of an FSX variable into a database's record in the current index position.

'This is a test' (>C:XMLVARS:StringValue,string)

will store a string of text using the same logic.

Likewise,

(C:XMLVARS:NumberValue,number) (>L:MyLocalVar,number)

will retrieve a value stored in the database, in the current index position, and assign it to (L:MyLocalVar,number); and

<String>%(C:XMLVARS:StringValue,string) %!s!</String>

will display a text stored in the current index position.

**(C:XMLVARS:NumberOfRecords,number)** - Read only

This identifier is used to retrieve the total number of records that have a variable name assigned. Useful for developing purposes.

**(C:XMLVARS:Reset)** - Write only

This identifier is used to clear the database and reset all the variable names to (empty) and values to default ( 0 for numbers and (empty) for strings).

### **Proper usage**

The correct syntax for each kind of operation is:

*-To store a numeric value:*

```
'MyVarName' (>C:XMLVARS:StoreVarName,string)
nValue (>C:XMLVARS:NumberValue,number)
```

*-To store a text string:*

```
'MyVarName' (>C:XMLVARS:StoreVarName,string)
'This is a text' (>C:XMLVARS:StringValue,string)
```

*-To retrieve a numeric value:*

```
'MyVarName' (>C:XMLVARS:SearchVarName,string)
(C:XMLVARS:NumberValue,number)
```

If 'MyVarName' doesn't exist in the database, an empty (0) value is retrieved.

*-To retrieve a text string:*

```
'MyVarName' (>C:XMLVARS:SearchVarName,string)
(C:XMLVARS:StringValue,string)
```

If 'MyVarName' doesn't exist in the database, an empty (") text is retrieved.

*-To retrieve the total number of records (variables) and store the value into a stack register:*

```
(C:XMLVARS:NumberOfRecords,number) sp0
```

*-To reset the database:*

```
(>C:XMLVARS:Reset)
```

### **Useful macros**

These group of macros make handling the correct code an easy task:

*-Store a text string:*

```
<Macro Name="SS">
    sp49 @1 (>C:XMLVARS:StoreVarName,string)
    l49 (>C:XMLVARS:StringValue,string)
</Macro>
```

*-Store a numeric value:*

```
<Macro Name="NS">
  sp49 @1 (>C:XMLVARS:StoreVarName,string)
  l49 (>C:XMLVARS:NumberValue,number)
</Macro>
```

*-Retrieve a text string:*

```
<Macro Name="SL">
  @1 (>C:XMLVARS:SearchVarName,string)
  (C:XMLVARS:StringValue,string)
</Macro>
```

*-Retrieve a numeric value:*

```
<Macro Name="NL">
  @1 (>C:XMLVARS:SearchVarName,string)
  (C:XMLVARS:NumberValue,number)
</Macro>
```

And the usage:

'My first string' @SS('>MyFirstString') (\* '>' is not mandatory but helps to follow the logic \*)

'Another string saved!' @SS('>My Second String')

@SL('My Second String') @SS('>MYThirdString')

(A:FLAPS HANDLE PERCENT,percent) @NS('>NumericValue')

@NL('NumericValue') (>L:FlapsValue,percent)

## **Operation with arrays**

Working with arrays of values is very simple here:

*-Store an unidimensional string array's text*

```
<Macro Name="SSA">
  sp49 @1 @2 scat (>C:XMLVARS:StoreVarName,string)
  l49 (>C:XMLVARS:StringValue,string)
</Macro>
```

*-Store an unidimensional numeric array's value*

```
<Macro Name="NSA">
  sp49 @1 @2 scat (>C:XMLVARS:StoreVarName,string)
  l49 (>C:XMLVARS:NumberValue,number)
</Macro>
```

*-Retrieve an unidimensional string array's text*

```
<Macro Name="SLA">
  @1 @2 scat (>C:XMLVARS:SearchVarName,string)
  (C:XMLVARS:StringValue,string)
</Macro>
```

*-Retrieve an unidimensional numeric array's value*

```
<Macro Name="NLA">
  @1 @2 scat (>C:XMLVARS:SearchVarName,string)
  (C:XMLVARS:NumberValue,number)
</Macro>
```

And the usage:

```
'My first array value' @SSA('>MyFirstArray:',1)
'My second array value' @SSA('>MyFirstArray:',2)
123.45 @NSA('>MyFirstArray:',2)
```

Second parameter supports an integer/register (I0,I1,I2,etc) value but NOT a local/sim variable name (LVar,AVar)

Then

@SLA('MyFirstArray:',1) displays 'My first array value' as text and 0 as numeric value  
@NLA('MyFirstArray:',2) displays 'My second array value' as text and 123.45 as numeric value

Finally, some interesting code:

```
<Update>

  1 sp0
  :5
  'I repeat the same ' 10 scat ' times!'
  @SSA('>MYFIRSTArray:',10)
  10 ++ d @NS('>MYFIRSTArray_Last) sp0
  10 400 &lt; if{ g5 }

</Update>
```

Then

```

<FormattedText .....>
  <String>
    %(@NL('MYFIRSTArray_Last) s2 0 !=)
    %{if}
    %(1 sp1)
    %{loop}
    %(@SLA('MYFIRSTArray:',l1))%!s!
    %(l1 ++ s1 l2 &lt;)
    %{if}\n%{end}
    %{next}
    %{end}
  </String>
</FormattedText>

```

## Displays

```

'I repeat the same 1 times!'
'I repeat the same 2 times!'
'I repeat the same 3 times!'
'I repeat the same 4 times!'

```

etc, etc.

## **UNIVERSAL macros**

For those who prefer not to deal with so many different macros, a bunch of generic ones is also available:

### *-Single Variable macro*

```

<Macro Name="V">
  @1 d 0 symb '>' scmi 0 ==
  if{ (>C:XMLVARS:StoreVarName,string) (>C:XMLVARS:@2Value,@2) }
  els{ (>C:XMLVARS:SearchVarName,string) (C:XMLVARS:@2Value,@2) }
</Macro>

```

### *-Unidimensional Array macro*

```

<Macro Name="Va1">
  @1 d 0 symb '>' scmi 0 ==
  if{ @3 scat (>C:XMLVARS:StoreVarName,string) (>C:XMLVARS:@2Value,@2) }
  els{ @3 scat (>C:XMLVARS:SearchVarName,string) (C:XMLVARS:@2Value,@2) }
</Macro>

```

### *-Bidimensional Array macro*

```

<Macro Name="Va2">
  @1 d 0 symb '>' scmi 0 ==
  if{ @3 scat ':' scat @4 scat (>C:XMLVARS:StoreVarName,string)
    (>C:XMLVARS:@2Value,@2) }
  els{ @3 scat ':' scat @4 scat (>C:XMLVARS:SearchVarName,string)
    (C:XMLVARS:@2Value,@2) }
</Macro>

```

And the usage:

```
'Any Text' @V('>MYFIRSTstring',string)
or
nNumber/LVar/Avar @V('>MYFIRSTNumber',number)
for storing a single variable
```

```
@V('MYFIRSTstring',string)
or
@V('MYFIRSTNumber',number)
for retrieving a single variable
```

```
'Any Text' @Val('>MYFIRSTstring:',string,n)
for storing a unidimension string array var, where n could be a static integer number or
a register ID (I0,I1,etc)
```

```
nNumber/LVar/Avar @Va2('>MYFIRSTNumber:',number,n1,n2)
for storing a bidimension number array var, where n1 and n2 can be static integer
numbers or register IDs (I0,I1,etc)
```

## **Credits**

Robbie McElrath for helping with the FS9 version of XMLVars.dll.  
Robert "Bob" McElrath for testing the module in FS9.

## **Comments**

XMLVARS can be installed and used on any commercial and non-commercial panel.  
XMLVARS cannot be distributed nor sold as a standalone product.

*Use this software at your own risk.*

© Tom Aguilo  
taguilo1@hotmail.com  
March, 2012