

Re: Extracting data from an XML to put into a constant

have to set the value to a
variable
instead.

[...]

'you cannot ...' seems incorrect to me. I
thought a script could
build strings containing almost any sequence
of valid VBScript
statements and use the Execute or
ExecuteGlobal statements to do
almost anything that can be done during
normal execution of
predefined scripts.

[...]

–Paul Randall

While it's not wrong/technically correct to say that you can
create
constants at runtime by Execute[Global] strings like

[...]

works as expected. But users/advocates of such hacks will
come to
grieve
(and deserve it). The problem with 'on the fly Consts' is that
you
can't
use them where you would like to use them most:

```
ExecuteGlobal "Const cnUB = 9"  
WScript.Echo cnUB  
On Error Resume Next  
Dim aFix( cnUB )  
WScript.Echo Err.Description  
On Error GoTo 0
```

this will break at *compile time*, because "Dim aFix(cnUB
)" is
compiled
(fails to compile) before 'ExecuteGlobal "Const cnUB = 9"
is
executed.

[...]

Re: Extracting data from an XML to put into a constant

Thank you for posting sample code to demonstrate the problem. But it actually demonstrates a problem that has nothing to do with Execute[Global]. The following script causes the same error.

```
Const cnUB = 9
WScript.Echo cnUB
On Error Resume Next
Dim aFix( cnUB )
WScript.Echo Err.Description
On Error GoTo 0
```

The Dim statement expects an integer constant which is not quite the same as a Constant with an integer value.

I would like to see sample code that demonstrates problems with using Execute[Global].

–Paul Randall

Thanks, Paul, for pointing out my error. I forgot/repressed that VBScript doesn't accept even 'real' Consts when dimming a fixed array. But the special treatment of 'real' Consts ('moving up' to start of scope) that isn't emulated by Execute–Consts can be seen from this code

```
Dim sSecs1 : sSecs1 = 10 * cnSecsPerMin
Dim sSecs2 : sSecs2 = 10 * cnSecsPerHour
ExecuteGlobal "cnSecsPerMin = 60"
Const cnSecsPerHour = 3600
WScript.Echo "sSecs1:", sSecs1
WScript.Echo "sSecs2:", sSecs2
```

and output:

```
cscript execonst.vbs
sSecs1: 0
sSecs2: 36000
```

In my opinion, compiling/executing code generated/loaded at runtime (Exec[Global] and Eval in VBScript) is a valuable feature of scripting languages – suited for special purposes and with their own set of problems (e.g. security). I use ExecGlobal regularly to load 'libraries' (from a CPP host application, where the src attribute of a script tag can't be used as in .wsf, .html, or .hta files), but I'd consider creating

Re: Extracting data from an XML to put into a constant

variables and/or constants by this method as a second best practice (to be replaced by using Dictionaries or Classes with restricted access to members most of the time).

OK, now I understand what you are trying to demonstrate; I think this script demonstrates it better:

```
'Start of code
MsgBox "PrescanConstant = " & PrescanConstant & vbCrLf & _
"GlobalExecConstant = " & GlobalExecConstant & vbCrLf & _
"OrdinaryVariable = " & OrdinaryVariable

Dim GlobalExecConstant
ExecuteGlobal "GlobalExecConstant = 2"
MsgBox "PrescanConstant = " & PrescanConstant & vbCrLf & _
"GlobalExecConstant = " & GlobalExecConstant & vbCrLf & _
"OrdinaryVariable = " & OrdinaryVariable

Dim OrdinaryVariable: OrdinaryVariable = 3
MsgBox "PrescanConstant = " & PrescanConstant & vbCrLf & _
"GlobalExecConstant = " & GlobalExecConstant & vbCrLf & _
"OrdinaryVariable = " & OrdinaryVariable

Const PrescanConstant = 1
'End of code
```

Like subroutine and function definitions, the value of constants is done during the prescan of a script. This means that the subroutine or function can be called from the first line of code, and that the value of a constant can also be used from the first line of code. Ordinary variables and constants defined with execute statements have no value until their defining statement is executed at run time.

But sometimes, the a constant's value is difficult to come by when you write the code, but a simple subroutine can determine the names of the constants and their values. This is what I do in do in the DefineConstants routine I posted last year; it access the typlib to get the info and uses ExecuteGlobal to do the definitions. So an ExecuteGlobal statement executed prior to the constant's being used is a good solution, especially for a person of limited resources who just wants to make use of objects already installed on their computer.

–Paul Randall