Article
Michael Cohen · Jan 26, 2017 3m read

How to use %Library.Global.Export() to export a global subtree

The Management Portal allows you to Export one or more globals to a file that you can then Import into that or another namespace. However, the Management Portal can only be used to export *entire* globals. For exporting selected nodes or subtrees within a global, a different utility is necessary. This utility is the Export() classmethod in the %Library.Global class, which can export an entire global but also has the ability to export selected nodes or subtrees.

For example, consider this global named ^DATA:
```
^DATA(1)="ONE"
^DATA(1,2)="twelve"
^DATA(1,2,3)=123
^DATA(1,"two")="three"
^DATA(2)="TWO"
^DATA(2,"two")="2two"
```

1. Let ' s say you only want to export the ^DATA(1) and ^DATA(2) nodes, not the subtrees. You can export just the top level nodes ^DATA(1) and ^DATA(2) as follows:
```
s gbl="DATA(1),DATA(2)"
s file="C:\temp\DATA1and2.gof"
w ##class(%Library.Global).Export(,gbl,file)
```

This prints:
```
Exporting to GO/GOF format started on ...
Exporting global: ^DATA(1)
Exporting global: ^DATA(2)
Export finished successfully.
1
```

If you examine the output file with a text editor, you will note that it includes just the top level nodes ^DATA(1) and ^DATA(2) as expected (with additional doc and control characters):
```
1 Export of 2 globals from Namespace MVFormat=5.V~17 Jan 2017 10:13 PM Cache ^DATA(1) ONE ^DATA(2) TWO
```

2. Now, let ' s say you want to export the ^DATA(1 and ^DATA(2 subtrees. You can do this simply by omitting the closing parentheses, as follows:
```
s gbl="DATA(1,DATA(2"
s file="C:\temp\DATA1and2subtrees.gof"
w ##class(%Library.Global).Export(,gbl,file)
```

This prints:
```
Exporting to GO/GOF format started on ...
Exporting global: ^DATA(1
Exporting global: ^DATA(2
Export finished successfully.
1
```

If you examine the output file with a text editor, you will find the entire ^DATA(1 and ^DATA(2 subtrees:
```
1 Export of 2 globals from Namespace MVFormat=5.V~17 Jan 2017 10:20 PM Cache ^DATA(1) ONE
 ^DATA(1,2) twelve ^DATA(1,2,3) 123 ^DATA(1,"two") three ^DATA(2) TWO ^DATA(2,"two") 2two
```

1 Export of 2 globals from Namespace MVFormat=5.V~17 Jan 2017 10:20 PM Cache ^DATA(1)

3. Up to this point, we have been specifying the variable gbl as a string which contains a comma-delimited list of global names. Unfortunately, because the comma is used as a delimiter to separate globals to be exported, this approach does not work if you want to export a global which is more than one subscript level deep. For example, let ' s say you want to export the ^DATA(1,2) and ^DATA(2,"two") nodes. If you follow steps similar to what we did in Example #1, the export will not work

If you specify this as:
s gbl="DATA(1,2),DATA(2,""two"")"

The Export() classmethod will export the ^DATA(1 subtree and then give an error trying to export a global named ^2.

To solve this problem, Export() also allows you to specify the desired global names as subscripts in a local array, passed by reference. So, to export these globals you can do the following:
k gbl
s gbl("DATA(1,2)")=""
s gbl("DATA(2,""two"")")=""
s file="C:\temp\DATA12and2two.gof"
w ##class(%Library.Global).Export(,.gbl,file)

This prints:
Exporting to GO/GOF format started on 01/17/2017 22:39:41...
Exporting global: ^DATA(1,2)
Exporting global: ^DATA(2,"two")
Export finished successfully.
1

NOTE: You must include the "." before "gbl" as shown above, causing the variable to be passed by reference.

If you examine the output file with a text editor, you will find the ^DATA(1,2) and ^DATA(2,"two") nodes as expected:
1 Export of 2 globals from Namespace MVFormat=5.V~17 Jan 2017 10:39 PM Cache
^DATA(1,2) twelve ^DATA(2,"two") 2two

4. Similarly, if you wanted to export the ^DATA(1,2) and ^DATA(2,"two") subtrees, you would again pass the array by reference but this time omit their closing parentheses, as follows:
k gbl
s gbl("DATA(1,2")=""
s gbl("DATA(2,""two""")=""
s file="C:\temp\DATA12and2twosubtrees.gof"
w ##class(%Library.Global).Export(,.gbl,file)
...

And as a final note, when you Import globals, if there is already existing data in those globals, the Import only sets/replaces the nodes that it includes, but leaves other nodes unchanged. If that is not your intent, you might want to kill the existing global before doing the Import().

#Object Data Model #Tips & Tricks #Caché