Question
Dmitry Maslennikov · Apr 30, 2020

%JSON.Adaptor for streams

Okay, we've got a quite useful way to very easily Import and export our objects as JSON, similar to what we already had before for XML.

So, It's a %JSON.Adaptor. But the issue here I faced with, working with Stream properties.

I have an example, when I generate an object, with stream binary stream properties. Export and Import the same, but getting the different resulting objects, depends on the original size of streams.

```
Class User.Test Extends (%JSON.Adaptor, %RegisteredObject)
{

Property Name As %String(%JSONFIELDNAME = "name");

Property HexStream As %Stream.GlobalBinary(%JSONFIELDNAME = "hex_stream", ENCODING =
"hex");

Property Base64Stream As %Stream.GlobalBinary(%JSONFIELDNAME = "base64_stream", ENCOD
ING = "base64");

ClassMethod Test(streamSize = {$system.SYS.MaxLocalLength()})
{
  Set object = ..%New()
  Set object.Name = "testing"
  Set maxLength = $system.SYS.MaxLocalLength()
  If streamSize<0 {
    Set streamSize = maxLength + streamSize
  }

  Set parts = (streamSize \ maxLength) + (streamSize # maxLength > 0)
  For p=1:1:parts {
    Set data = ""
    Set length = $Select(streamSize = maxLength: streamSize, p=parts: streamSize # ma
xLength, 1: maxLength)
    For i=1:1:length {
      Set data = data _ $Char($Random(256))
    }
    Do object.HexStream.Write(data)
    Do object.Base64Stream.Write(data)
  }
  Write !,"original HexStream size = ", $Fnumber(object.HexStream.Size, ",")
  Write !,"original Base64Stream size = ", $Fnumber(object.Base64Stream.Size, ",")

  Do object.%JSONExportToStream(.jsonStream)
  Kill object

  Write !,"Exported JSON size = ", $Fnumber(jsonStream.Size, ",")

  Set newObject = ..%New()
```

```
  Set tSC = newObject.%JSONImport(.jsonStream)
  If $$$ISERR(tSC) {
    Do $System.OBJ.DisplayError(tSC)
    Quit
  }

  Write !
  Write !,"imported object HexStream size = ", $Fnumber(newObject.HexStream.Size, ","
)
  Write !,"imported object Base64Stream size = ", $Fnumber(newObject.Base64Stream.Siz
e, ",")
}

}
```

So, let's to some tests.

Quite small size of the stream. All good.

```
USER>do ##class(User.Test).Test(10000)
original HexStream size = 10,000
original Base64Stream size = 10,000
Exported JSON size = 33,389

imported object HexStream size = 10,000
imported object Base64Stream size = 10,000
```

Let's make it bigger. One of the resulting streams reached the limit. But the import did not fail, it returns that it was OK. Even when we got broken stream at the end.

```
USER>do ##class(User.Test).Test(40000)
original HexStream size = 40,000
original Base64Stream size = 40,000
Exported JSON size = 133,389

imported object HexStream size = 40,000
imported object Base64Stream size = 32,656
```

Let's find when it fails for HexStream as well.

```
USER>do ##class(User.Test).Test($system.SYS.MaxLocalLength()/2)
original HexStream size = 1,820,572
original Base64Stream size = 1,820,572
Exported JSON size = 6,068,773

imported object HexStream size = 1,820,572
imported object Base64Stream size = 32,656

USER>do ##class(User.Test).Test($system.SYS.MaxLocalLength()/2+1)
original HexStream size = 1,820,573
original Base64Stream size = 1,820,573
Exported JSON size = 6,068,779
ERROR #5002: ObjectScript error: <MAXSTRING>%JSONImportInternal+24^User.Test.1
```

So, it looks like, it does not work at all for streams bigger than 1,820,572, half size of MaxLocalLength (hex format just converts binary to a hex string, so, 1 byte to 2 bytes). And if used base64 for JSON, stream can't be bigger

even than just 32656.

Did I miss anything here, and I can make it work? Is there any way, I can override this behavior somehow, rewrite the way, that will eliminate these errors?

I would split such streams in chunks, as an array of smaller string values during Export. And import would just easily convert it to Stream back.

But the issue also looks like in native JSON, which can store such a big value, but with no way to access it without getting <MAXSTRING> error, or I did not find that way.

#JSON #InterSystems IRIS