

Debugging in HealthShare

Article

[Daniel Kutac](#) · May 15, 2020



3m read

Debugging in HealthShare

Hi, if you develop production with HealthShare, you may have noticed there is a nice tracing feature available - HS.Util.Trace.Operations. This feature allows visually tracing debug information, in structured way. It complements good old logging macros like \$\$\$LOGINFO, \$\$\$TRACE and alike.

I use this feature a lot. However, it has limited functionality, it works nice with classes that extend XML Adapter, streams or data types, but it doesn't work well with arbitrary objects that do not extend from XML Adapter.

I decided to implement a subclass of the trace helper class HS.Util.Trace.Helper and add the missing functionality.

here is the code I added:

```
Elseif tVal.%Extends("%Stream.Object") {
    Do tStream.Write("<![CDATA[")
    Set tSC = tStream.CopyFrom(tVal)
    Do tStream.Write("]]>")
    Do tVal.Rewind()
}
else {
    // DK - arbitrary object
    set tInitIO=$io
    set %Stream=##class(%Stream.TmpCharacter).%New()
    use tInitIO::("^"_$zname)
    do ##class(%Device).ReDirectIO(1)
    zw tVal
    If ##class(%Device).ReDirectIO(0) Use tInitIO
    do %Stream.Rewind()
    Do tStream.Write("<![CDATA[")
    set tSC=tStream.CopyFrom(%Stream)
    Do tStream.Write("]]>")
}
```

I have used a nice feature of redirecting output to variable, - added following class method to my trace helper class.

```
ClassMethod redirects() [ Private, ProcedureBlock = 0 ]
{
    #; Public entry points for redirection
    wstr(s) Do %Stream.Write(s) Quit
    wchr(a) Do %Stream.Write($char(a)) Quit
    wnl Do %Stream.Write($char(13,10)) Quit
    wff Do %Stream.Write($char(13,10,13,10)) Quit
    wtab(n) New chars Set $piece(chars," ",n+1)=" " Do %Stream.Write(chars) Quit
    rstr(len,time) Quit ""
    rchr(time) Quit ""
}
```

you can see the sample output in the following picture.

Debugging in HealthShare

Published on InterSystems Developer Community (<https://community.intersystems.com>)

The screenshot displays the Visual Trace interface. On the left, a sequence diagram shows the flow of a request through four components: Services (EnLib.Testing Service), Processes (EnLib.Testing Process), Operations (X.Operation Test), and HS.Util.Trace Operations. The diagram includes four numbered steps: [1] Request, [2] Request, [3] Request, and [4] NULL. On the right, the XML output is shown, starting with `<?xml version="1.0" ?>` and including a `<Request>` element with various attributes and nested `<Item>` elements. The XML content is as follows:

```
<?xml version="1.0" ?>
<!-- type: HS.Util.Trace.Request id: 100596 -->
<Request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:s="http://www.w3.org/2001/XMLSchema">
  <HSCoreVersion>17</HSCoreVersion>
  <CurrentClass>X.Operation.Test</CurrentClass>
  <CurrentMethod>DoTest</CurrentMethod>
  <Comment>test arbitrary object</Comment>
  <Items>
    <Item>
      <ItemName>tMsg</ItemName>
      <ItemValue><![CDATA[tVal=<OBJECT REFERENCE>[5@X.Data.Test]
-----
general information
-----
oref value: 5
class name: X.Data.Test
reference count: 4
-----
attribute values
-----
DOB = 57564
Name = "Pepina Skořdopole"
Status = 1
%Concurrency = 1 <Set>
-----
]]></ItemValue>
    </Item>
  </Items>
</Request>
```

All you need in your business component, is to add your trace helper subclass to the list of base classes and call `..HSTrace()` method or its macro equivalent `$$$HSTRACE` with your desired parameters.

Hope you find this little improvement useful.

Dan

[#Debugging](#) [#Deployment](#) [#HealthShare](#)

30 1 0 2 254

Log in or sign up to continue

Add reply

Source URL: <https://community.intersystems.com/post/debugging-healthshare>