

## 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.

The screenshot displays the Visual Trace interface. On the left, a sequence diagram shows the flow of a request through four components: EnLib.Testing Service, EnLib.Testing Process, X.Operation Test, and HS.Util.Trace Operations. The request is initiated at [1] on 2020-05-15 10:21:12.663, passes through [2] on 2020-05-15 10:21:12.668, and [3] on 2020-05-15 10:21:12.671, finally reaching [4] on 2020-05-15 10:21:12.672. The request is labeled 'Request' and 'Request' at various points. On the right, the XML view shows the request details, including the request ID (100596), the current class (X.Operation.Test), and the current method (DoTest). The XML also includes a comment 'test arbitrary object' and a list of items, with the first item being a message (tMsg) containing a class name (X.Data.Test) and a reference count (4). The XML is expanded to show the attribute values, including DOB (57564), Name ('Pepina Skořdopole'), Status (1), and %Concurrency (1).

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

Hope you find this little improvement useful.

Dan

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

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