Article
Randy Pallotta · May 27, 2016 · 3m read

Removing all trailing delimiters for fields and segments in HL7 messages

Two fairly common requests we receive from HL7 customers are (1) how to remove all trailing delimiters for fields and segments in HL7 messages and (2) how to "find and replace" for an entire HL7 message (as opposed to one segment/field at a time). The code sample below shows a custom function that solves for item 1 and by extension item 2 above. In other words the same approach can be used for finding/replacing any sequence of chars in an entire HL7 message, with some tweaks to the custom function.

Here's an example of how to actually call the function from DTL. I recommend doing this as the last line of code in any DTL where it is used. Note that "target" is used in both the value and property of the code below (as opposed to the DTL "source" message).

```
<assign value='##class(ISCSample.Custom.Function.ConversionScrub).ConversionScrub(target)' property='target' action='set' />
```

Here's the sample class and classmethod:

```
Class ISCSample.Custom.Function.ConversionScrub Extends Ens.Rule.FunctionSet
{

/// This function removes trailing pipes "¦" and/or up-carats "^" from the given HL7 message.
///
/// Per the HL7 standard, empty fields at the end of a segment need not be sent at all.
/// This also applies to empty sub-components at the end of a field.
/// For example, PID¦1¦12345¦¦DOE^JOHN^^^^¦¦¦¦¦¦¦¦¦¦ is equivalent to PID¦1¦12345¦¦DOE^JOHN.
///
/// We created this function to help with our file compares, so that Ensemble mimics the
/// behavior of Cerner Open Engine which removes trailing pipes automatically. By calling
/// this function, we ensure that Ensemble produces the same output as Cerner Open Engine.
ClassMethod ConversionScrub(pHL7 As EnsLib.HL7.Message) As EnsLib.HL7.Message
{
  set tSegCount = pHL7.SegCount
  set tCount = pHL7.SegCount
  while tCount > 0 {
  set tSegment = pHL7.GetValueAt(tCount)

  ; Remove trailing up carats
  while ..StartsWith(($REVERSE(tSegment)),"^") {
  set tLength = ..Length(tSegment)
  set tSegment = ..SubString(tSegment,1,(tLength-1))
  do pHL7.SetValueAt(tSegment,tCount)
  }

  ; Remove trailing up carats within a field
  set tIndex = 0
```

```
    while $FIND(tSegment,"^¦",tIndex) > 0 {
    set tSegment = $Replace(tSegment,"^¦","¦")
    do pHL7.SetValueAt(tSegment,tCount)
    }

    ; Remove trailing pipes
    while ..StartsWith(($REVERSE(tSegment)),"¦") {
    set tLength = ..Length(tSegment)
    set tSegment = ..SubString(tSegment,1,(tLength-1))
    do pHL7.SetValueAt(tSegment,tCount)
    }

    ; Remove empty segment
    if ..Length(tSegment) = 3 {
    do pHL7.SetValueAt("",tCount,"remove")
    }
    set tCount = tCount-1
    }
quit pHL7
}

}
```

[#Code Snippet](#) [#DTL](#) [#HL7](#) [#Ensemble](#)