Artículo
Muhammad Waseem · 13 oct, 2021 Lectura de 4 min

División de mensajes ORU mediante ObjectScript y DTL

A lo largo de los años, me he encontrado con la necesidad de crear varios mensajes HL7 basados en un solo mensaje entrante. Por lo general, toman la forma de un pedido o son el resultado de un laboratorio. Cada vez que he afrontado el reto, he intentado empezar de cero, con la convicción de que el intento anterior podría haberse hecho mejor.

Recientemente, volvió a surgir la necesidad y pude crear una solución de la que no me avergonzaba. Mi principal preocupación era que siempre me encontraría enterrado en un BPL, o usaría ObjectScript e intentaría editar mensajes usando el método SetValueAt para la clase de mensaje HL7.

Problema
Cuando el Sistema A procesa múltiples pedidos para un solo paciente, el resultado vendrá en un solo mensaje con ORCgrp repetido con los segmentos OBR y OBX contenidos en este. El sistema B solo puede recibir un único OBR por mensaje.

Estrategia
Desarrollar un proceso de ObjectScript que dividirá un solo mensaje en varios, en función del número de repeticiones de ORCgrp, y hacerlo manipulando únicamente el mensaje HL7 con una DTL.

Ejecución

El Código (parte 1)
Para empezar, necesitamos una clase que extienda Ens.BusinessProcess y espere un mensaje HL7 bajo pedido:

```
Class Demo.Processes.MessageSplitter Extends Ens.BusinessProcess{
 Method OnRequest(pRequest As EnsLib.HL7.Message) As %Status{
    Quit $$$OK
    }
}
```

El siguiente paso es recorrer el mensaje en función del número de repeticiones de ORCgrp. Para ello, necesitaremos 2 cosas:

1. El número de repeticiones
2. Un bucle For

Para obtener el número de repeticiones, podemos tomar el recuento del mensaje usando el siguiente código:

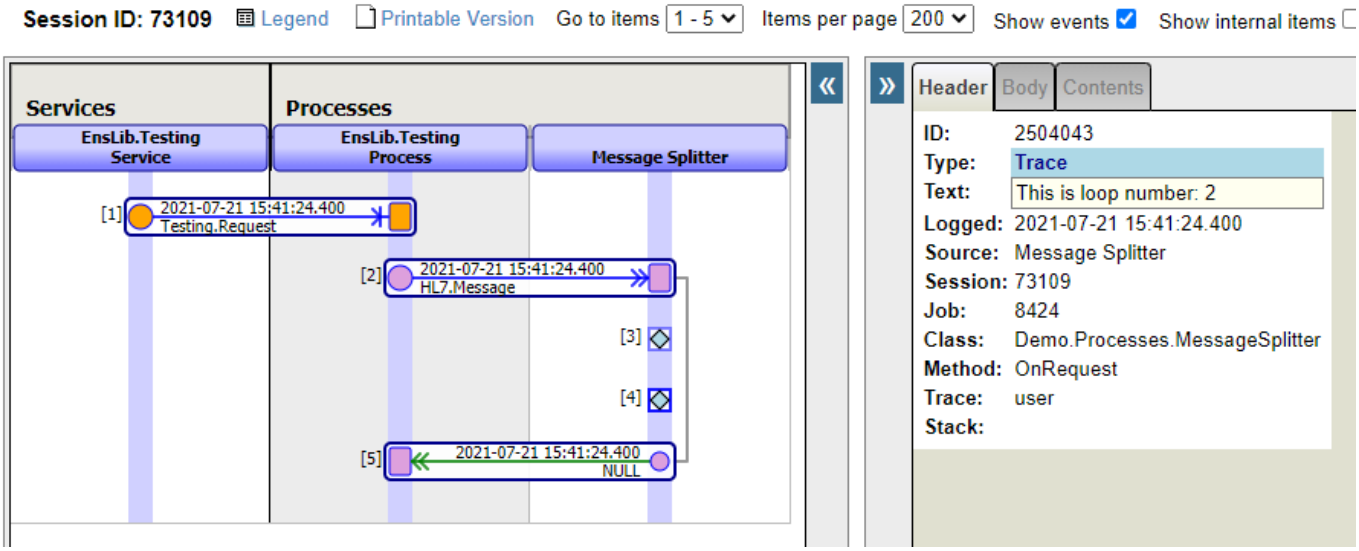
```
Set ORCCount = pRequest.GetValueAt("PIDgrpgrp(1).ORCgrp(*)")
```

Esto establecerá la variable "ORCCount" en el número de repeticiones.

Poniendo esto junto con un bucle For y un seguimiento para ver algunos resultados, se ve así:

```
Method OnRequest(pRequest As EnsLib.HL7.Message) As %Status{
    Set ORCCount = pRequest.GetValueAt("PIDgrpgrp(1).ORCgrp(*)")
    For i=1:1:ORCCount {
        $$$TRACE("This is loop number: "_i)
    }
    Quit $$$OK
}
```

Pasar un mensaje con dos repeticiones ORCgrp a través de este proceso desde dentro de una producción nos da:



Como puedes ver, obtenemos dos rastros.

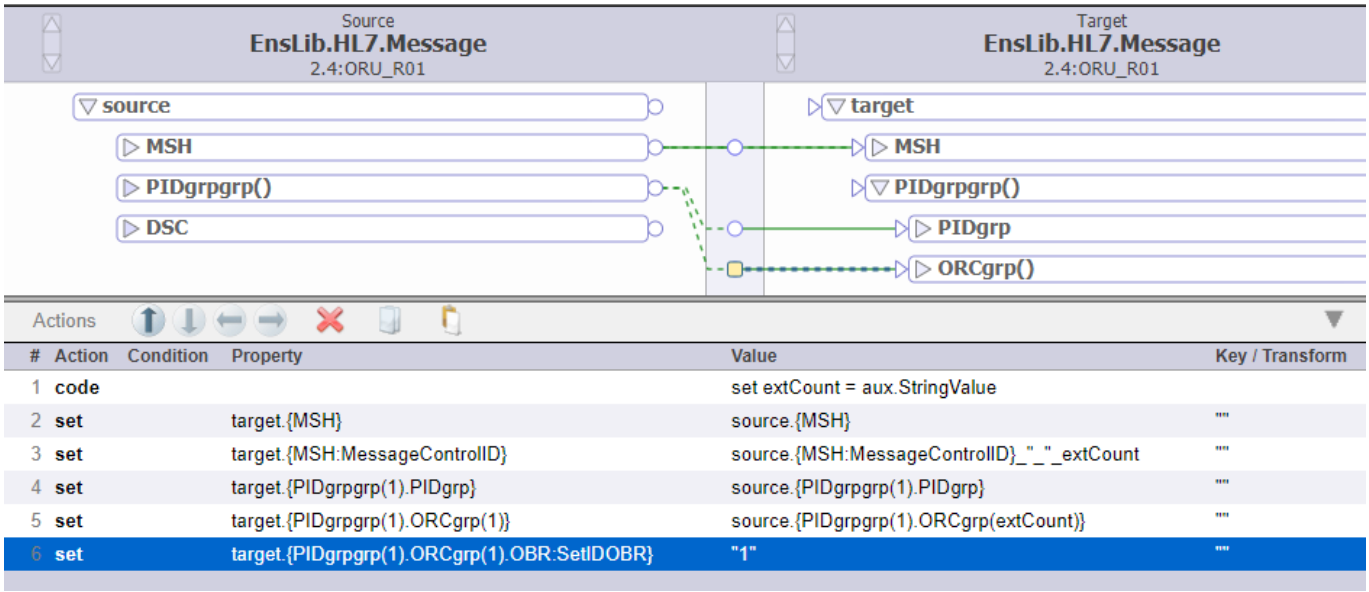
Ahora, desde aquí, necesitamos poder llamar a una transformación, y también poder decirle a esa transformación qué iteración del ORCgrp debería estar devolviendo. Para esto, usaremos el parámetro "aux" (que a veces es ignorado) para las clases de transformación.

La transformación

La transformación en sí puede ser muy simple. Todo lo que queremos para esto es:

1. Copiar el encabezado MSH mientras haces que MessageControlID sea único para tu mensaje dividido
2. Establecer el primer ORCgrp del mensaje de destino en la iteración en la que estamos desde la fuente
3. Establecer el Target SetIDOBR en "1", ya que siempre será el primero en el mensaje de destino

Para esto, nuestra transformación debería verse un poco así:



Pero espera, ¿de dónde obtiene aux.StringValue los datos?

Para averiguarlo, tenemos que volver al código...

El Código (parte 2)
Podemos pasar una clase a la transformación a través del parámetro aux, y para este escenario solo voy a usar un contenedor de cadenas. También vamos a enviar la salida de la transformación a un objetivo para que podamos ver los resultados:

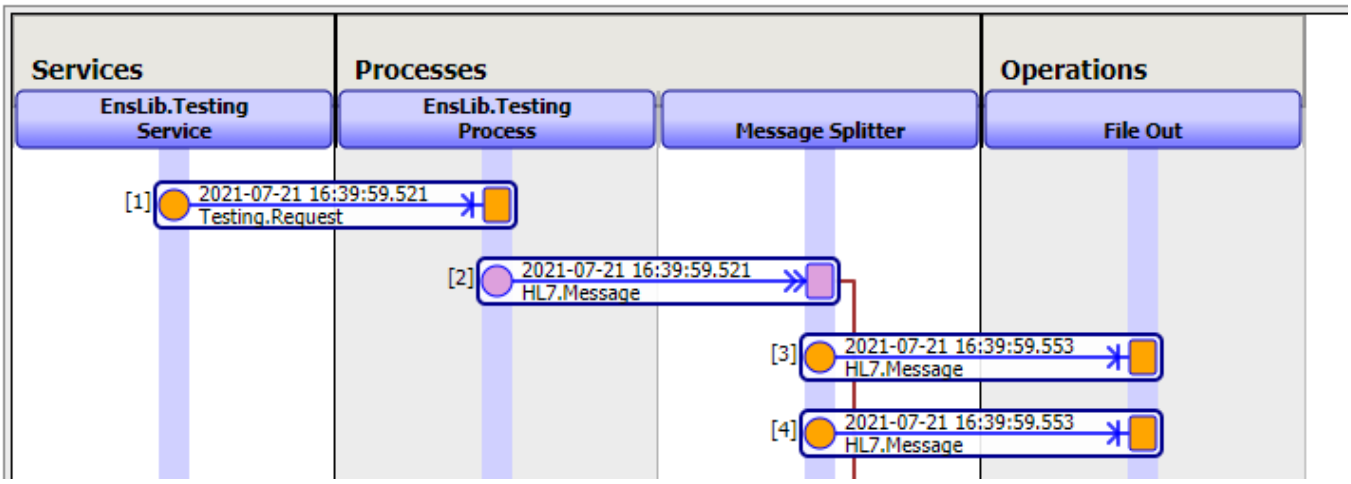
```
Class Demo.Processes.MessageSplitter Extends Ens.BusinessProcess{

Property TargetConfigName As Ens.DataType.ConfigName;
Parameter SETTINGS = "TargetConfigName";

Method OnRequest(pRequest As EnsLib.HL7.Message) As %Status{

    Set ORCCount = pRequest.GetValueAt("PIDgrpgrp(1).ORCgrp(*)")
    For i=1:1:ORCCount {
        Set contextString = ##class(Ens.StringContainer).%New()
        Set contextString.StringValue = i
        $$$QuitOnError(##Class(Demo.Transformations.R01Split).Transform(pRequest,.Spl
itR01,.contextString))
        $$$QuitOnError(..SendRequestAsync(..TargetConfigName,SplitR01,0))
    }
    Quit $$$OK
  }
}
```

Luego, en la producción, establecemos un destino usando la nueva opción de configuración que creamos con la propiedad y el parámetro en la parte superior de la clase, y deberíamos ver algo como esto:



Conclusión

Como dije al principio, siempre me parece que desarrollo una solución para este tipo de problema y luego miro hacia atrás y pienso que podría hacerse mejor. Esta no es una excepción, pero ciertamente es mejor que las iteraciones anteriores.

Para mejorar esto en el corto plazo, añadiré comentarios descriptivos al ObjectScript. A largo plazo, me gustaría poder añadir una configuración para la clase de transformación, para que pueda controlarse desde fuera de ObjectScript.

En general, puedo ver esto como un enfoque que tomaré para desarrollos futuros, y no comenzaré completamente desde cero.

(P.D.: Estoy feliz de compartir el código para esto, aunque solo puedo adjuntar un pdf a este artículo. Parece poco para ser algo para Open Exchange, pero si hay algún interés en que lo cargue allí o en otro lugar, decídmelo).

#DTL #HL7 #Mejores prácticas #ObjectScript #Caché #Ensemble

URL de fuente:https://es.community.intersystems.com/post/divisi%C3%B3n-de-mensajes-oru-mediante-objectscript-y-dtl