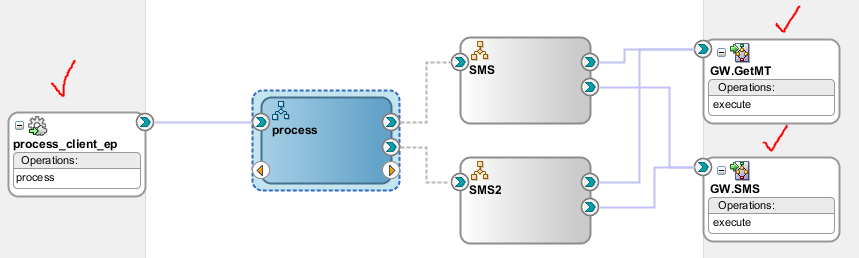
# **Bước 1: Bố trí đầu tiên là cấu trúc của project SMS**

Update Service -> SOAP

Cấu trúc của Schema: Trong file NEOSchema ->

Trong đó res chứa tập các pool trong khi rs chỉ chứa tập các ps(tập danh sách của các param)

Đầu vào chính là req, đầu ra chính là các res nhé.



Phân tích đến bước này con như xong bước 1:

Các giá trị đầu vào của process là req và res

Đầu vào của các REST URL là: rq và rs, dùng cho cả GW.GetMT và GW.SMS.

# Bước 2: Phân tích cái GW.GetMT và GW.SMS

Lấy MT và Gửi MT đi

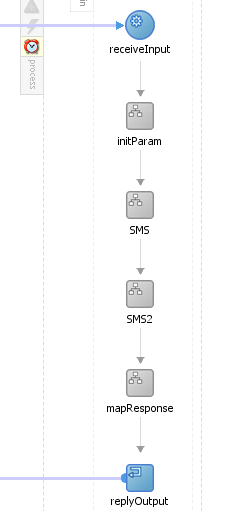
<http://localhost:18080/smsmt/get> POST

Thông số chỉnh lại localhost: thành địa chỉ server khác để lấy được kết quả.

Trong những khối này chỉ mục đích duy nhất là chứa các yêu cầu REST từ một service nào đó, nhận giá trị trả về và xử lý chúng.

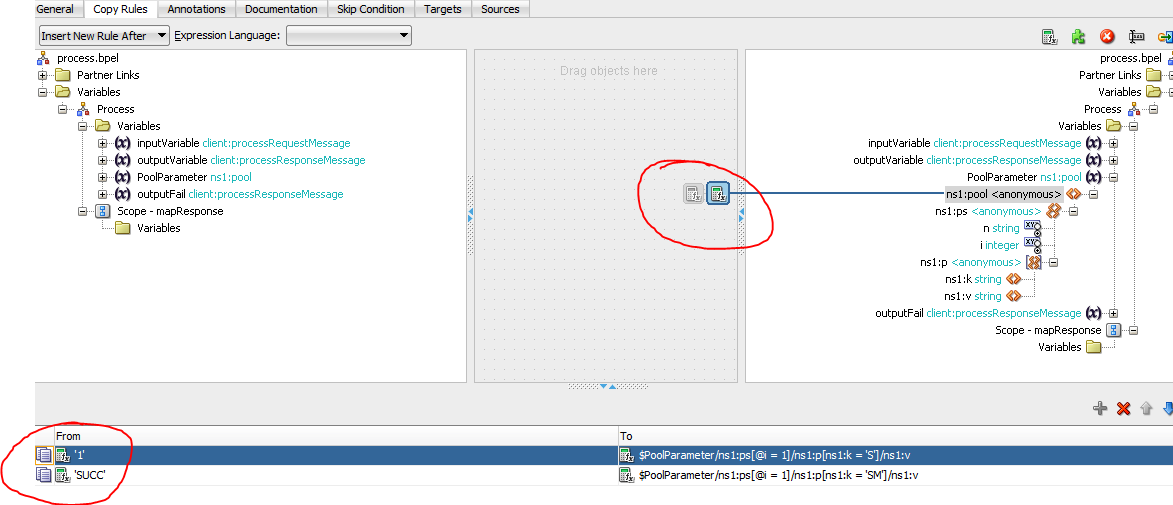
URL trong phần **GW.SMS** đang không trả về kết quả của đường dẫn -> đường dẫn không hoạt động.

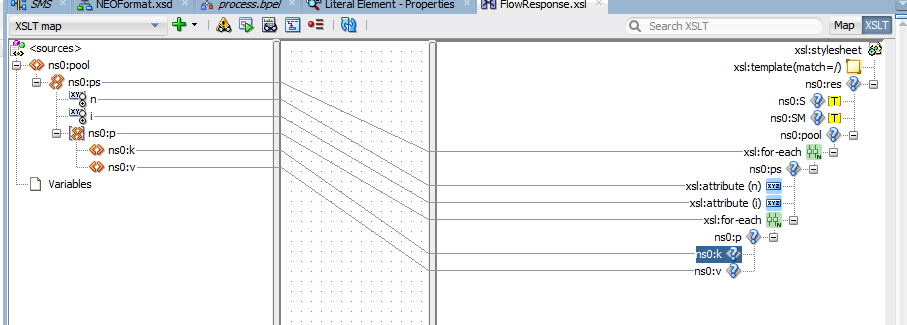
<http://localhost:18080/smsmt/send> POST hiện tại chưa quan tâm đến cái send MT này



Những phần sử lý như sau: initParam: khởi tạo những tham số cần thiết.

mapResponse: Map lại những phản hồi của hệ thống

Khi thành công sẽ gán cho giá trị vào từng giá trị tương ứng, mục đích chính là gán thôi, ngoài ra không có gì khác.

Khi nhận thành công thì dùng Transaction các giá trị của nó tương ứng. 

Map từng phần -> code bằng tay không kéo thả, gây ra nhiều dữ liệu không mong muốn như các giá trị id chẳng hạn.

# Bước 0: Tạo PoolParameter:

Việc tạo poolparameter cho việc lưu dữ liệu một cách đóng gói.

|  |  |
| --- | --- |
|  | **Edit Variable – PoolParameter**  Name: PoolParameter  Type: Element  Namespace:  Local Part: chính là lấy từ NEOSchemas đưa ra ngoài. |

# **Bước 3: Phân tích phần initParam:**

|  |  |
| --- | --- |
|  | FlowInitParam: Khởi tạo nên tham số cho hệ thống.  Giá trị nhận từ InputVariable được map đến giá trị cho PoolParameter  Có thể khởi tạo thêm các p tương ứng muốn dùng.  Chọn them từng cặp tương ứng -> chọn cặp nào nguồn ở đâu thì nằm trong phần select của đường dẫn đến nó.  FlowUniqueParam: Đồng nhất tham số hệ thống  Nhận giá trị từ cái PoolParameter đến cái PoolParameter tương ứng.  Mục đích chính làm cho truyền những cặp giá trị tương ứng vào Poolparameter  GetInstanceName: Lấy InstanceName |

Phân tích phần FlowInitParam:

|  |  |
| --- | --- |
|  | Từ nguồn đích:  Đến nguồn variable:  Mapper file: FlowInitParam  Trong đó tạo mới một PoolParameter cho mục đích lưu những dữ liệu cần thiết.  Nội dung file: FlowInitParam được quy định theo cách cấu hình. |

Phần FlowUniqueParam

|  |  |
| --- | --- |
|  | Biến nguồn từ Poolparameter  Biến đích đến: PoolParameter  Lý do: xử lý xoay quanh cái poolParameter này, những giá trị được lưu vào Pool, xử lý logic cũng trong pool, xuất dữ liệu ra ngoài cũng thông qua pool này.  Key trong phần FlowUniqueParam có nghĩa gì?  <xsl:key name="product" match="/ns0:pool/ns0:ps[@i = 1]/ns0:p" use="ns0:k"/>  <ns0:pool>  <ns0:ps n="Global" i="1">  <xsl:for-each select="/ns0:pool/ns0:ps[@i = 1]/ns0:p[generate-id() = generate-id(key('product',ns0:k)[1])]">  <ns0:p>  <ns0:k>  <xsl:value-of select="ns0:k"/>  </ns0:k>  <ns0:v>  <xsl:value-of select="ns0:v"/>  </ns0:v>  </ns0:p>  </xsl:for-each>  </ns0:ps>  </ns0:pool> |

Phần cấu hình của getInstanceName: -> phần getInstanceName

|  |  |
| --- | --- |
| Edit Java Embedding  getInstanceName | try {  String instanceName = System.getProperty("weblogic.Name");  setVariableData("PoolParameter", "/ns1:pool/ns1:ps[@i = 1]/ns1:p[ns1:k = 'INST']/ns1:v", instanceName);  }catch(Exception e){  setVariableData("PoolParameter", "/ns1:pool/ns1:ps[@i = 1]/ns1:p[ns1:k = 'INST']/ns1:v", "unkhow instance");  } |
|  |  |

# **Bước 4: Phân tích phần mapResponse:**

|  |  |
| --- | --- |
|  | mapSuccess: Truyền giá trị vào khi thành trả về kết quả thành công, khi thành công thì truyền giá trị vào:    Gán biến cho 2 giá trị này thôi, ngoài ra không có gì nữa.  @i là gì?  Ns1:k = ‘S’ -> status  Ns1:k = ‘SM’ -> status message  FlowResponse:  Truyền từ PoolParameter đến outputVariable. Truyền giá trị tương ứng cho đầu ra.  Các phần cần phân tích: |

Phân tích các thành phần trong FlowResponse:

|  |  |
| --- | --- |
|  | FlowResponse: mục đích chứa phản hồi từ Response về ->  Nội dung:  <xsl:template match="/">  <ns0:res>  <ns0:S>1</ns0:S>  <ns0:SM>SUCC</ns0:SM>  <ns0:pool>  <xsl:for-each select="/ns0:pool/ns0:ps">  <ns0:ps>  <xsl:attribute name="n">  <xsl:value-of select="@n"/>  </xsl:attribute>  <xsl:attribute name="i">  <xsl:value-of select="@i"/>  </xsl:attribute>  <xsl:for-each select="ns0:p">  <ns0:p>  <ns0:k>  <xsl:value-of select="ns0:k"/>  </ns0:k>  <ns0:v>  <xsl:value-of select="ns0:v"/>  </ns0:v>  </ns0:p>  </xsl:for-each>  </ns0:ps>  </xsl:for-each>  </ns0:pool>  </ns0:res>  </xsl:template> |

|  |  |
| --- | --- |
|  | Source nguồn của biến xuất phát từ PoolParameter -> giá trị của pool ->  outputVariable -> giá trị biến đầu ra  Thông qua file Mapper File: FlowResponse.xsl |
| <xsl:template match="/">  <ns0:res>  <ns0:S>1</ns0:S>  <ns0:SM>SUCC</ns0:SM>  <ns0:pool>  <xsl:for-each select="/ns0:pool/ns0:ps">  <ns0:ps>  <xsl:attribute name="n">  <xsl:value-of select="@n"/>  </xsl:attribute>  <xsl:attribute name="i">  <xsl:value-of select="@i"/>  </xsl:attribute>  <xsl:for-each select="ns0:p">  <ns0:p>  <ns0:k>  <xsl:value-of select="ns0:k"/>  </ns0:k>  <ns0:v>  <xsl:value-of select="ns0:v"/>  </ns0:v>  </ns0:p>  </xsl:for-each>  </ns0:ps>  </xsl:for-each>  </ns0:pool>  </ns0:res> | Giá trị trong file FlowResponse.xsl  Truyền giá trị cho S và SM cho từng giá trị đó  Giá trị của attribute name tương ứng -> value-of select  For each ns0:p |

# **Bước 5: Phân tích cục SMS đầu tiên: Trong đây chứa phần Call1**

Trong đây không có các Variable gì cả?

Được thiết kế bằng một cục sequence -> mục đích cho việc thực hiện thao tác theo cách tuần tự, quá trình quay trở lại khi thực hiện thành công. 

* Các giá trị ns1, ns2, ns3 chỉ là các namespace tương ứng trện thống -> tạo một thì sẽ là 1, càng nhiều namespace thì các ns1 càng nhiều.

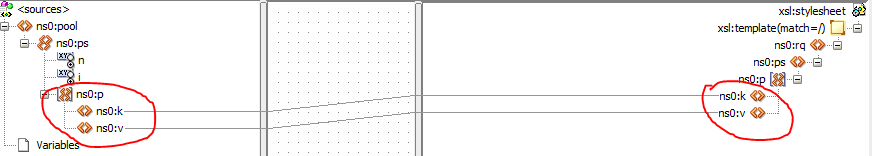
Giá trị trong Subprocess SMS:

|  |  |
| --- | --- |
|  | PoopParameter: element -> ns1:pool  TYPE: Simple Type: String  OutFail: Message Type -> ns2:processResponse… |
|  |  |

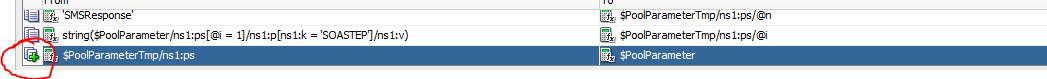
# **Bước 6: Phân tích cục SMS thứ 2: Trong phần Call2**

Trong đây không có các Variable gì cả?

# **Bước 7: Nhận giá trị từ một SMS trả về**

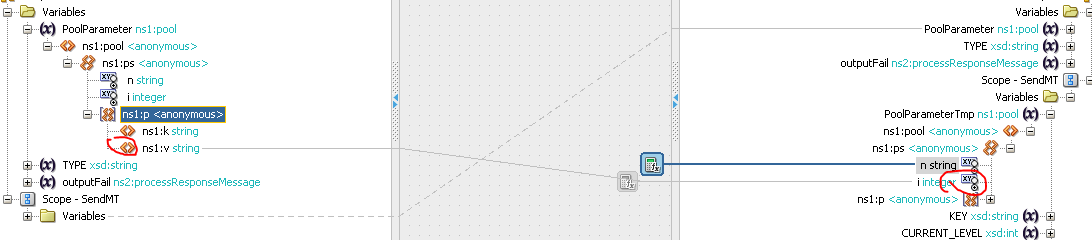


Nhận theo giá trị của key và value.

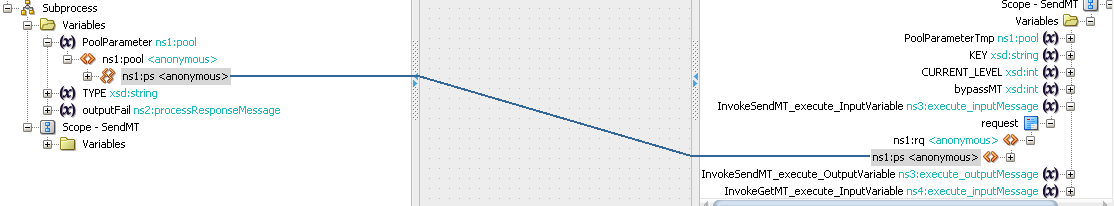


string($PoolParameter/ns1:ps[@i = 1]/ns1:p[ns1:k = 'SOASTEP']/ns1:v)

$PoolParameterTmp/ns1:ps/@i



Map CONTENT SMS:



$PoolParameter/ns1:ps/ns1:p[ns1:k = concat('SMSMT\_',$TYPE,'\_CONTENT')]/ns1:v

$InvokeSendMT\_execute\_InputVariable.request/ns1:ps/ns1:p[ns1:k = 'CONTENT']/ns1:v

Bước tạo mới một Poolparameter trong Subprocess:

Tạo mới một biến Poolprameter trong subprocess -> mục đích chính cho việc nhận và sử lý những giá trị trong sub này.

Những tham số truyền vào trong: SMSRequest.xsl -> được truyền vào SMS.bpel -> SMSRequest ->

Truyền tham số vào request đến Transaction:

<ns0:p>

    <ns0:k>SERVICE\_CODE</ns0:k>

    <ns0:v>

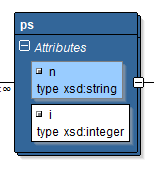
        <xsl:value-of select="/ns0:pool/ns0:ps[@i = 1]/ns0:p[ns0:k = 'SERVICE\_CODE']/ns0:v"/>

    </ns0:v>

</ns0:p>

Giá trị của @i = 1 -> chính là các attribution ->

Giá trị của @n -> name còn @i là index



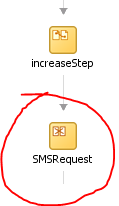
Giá trị cho các tham số trong request được truyền theo các file SMSRequest -> truyền đi để xử lý, bao nhiêu tham số cũng được.

Yêu cầu mới:

Vẫn sử dụng URL GetMT: <http://10.252.10.238:18080/smsmt/get>

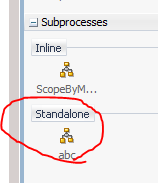
Và đường dẫn SentMT: <http://10.252.10.238:18080/smsmt/send>

<ns0:ps n="Global" i="1"><ns0:ps n="GetSMSMTRequest" i="2"><ns0:ps n="GetSMSMTResponse" i="2"><ns0:ps n="SMSRequest" i="2"><ns0:ps n="SMSResponse" i="2"><ns0:ps n="GetSMSMTRequest" i="3"><ns0:ps n="SMSRequest" i="3"><ns0:ps n="SMSResponse" i="3">Bươc 8: Thêm giá trị trong Requet



Quy tắt đặt tên của Adapter:

GW.GetMT -> …. Mục đích getMT từ GW



Tạo subprocess theo kiểm standalone -> không phải theo kiểm inline