SAP PI/XI: Generating Sequence Number Between Multiple Instances of Mapping Execution

Applies to:

SAP XI 3.0, PI 7.0 and 7.1.

Summary

The paper discusses about how to obtain sequence number between multiple executions of the same or different mappings in a lightweight fashion.

Author: Sudip Kumar Paul

Company: Accenture

Created on: 15 October 2010

Author Bio

Sudip Kumar Paul is an SAP certified Net weaver (SAP XI/PI) consultant at Accenture. He has more than 6 years of experience in SAP Net weaver, Java/J2EE and middleware technologies and more than 5 years in SAP XI/PI. Was part of Netweaver team at SAP.

Table of Contents

Introduction

Let's run through a very common scenario which consultants come across while dealing with the transformation or mapping step in SAP XI/PI.

If there is a scenario where it requires executing mapping more than once and with each run you should be able to obtain a number which is next in sequence. This requires the last sequence information to be remembered till the point the mapping gets executed again. To make the problem more understandable lets look at a business scenario.

Example: Business scenario

Payment runs (TCODE: F110) happens in a SAP FI system and the payment information needs to be send across to a third party say in a file(or some other format). Now every day when there is a payment run there file generated should contain a sequence number identifying the file so that the third party application can process these files in the correct order.

The Problem

The problem is SAP PI/XI as such does provide any mean to have number ranges or generate sequence between multiple executions of mappings. It only provides a mean to have a sequence within a mapping execution.

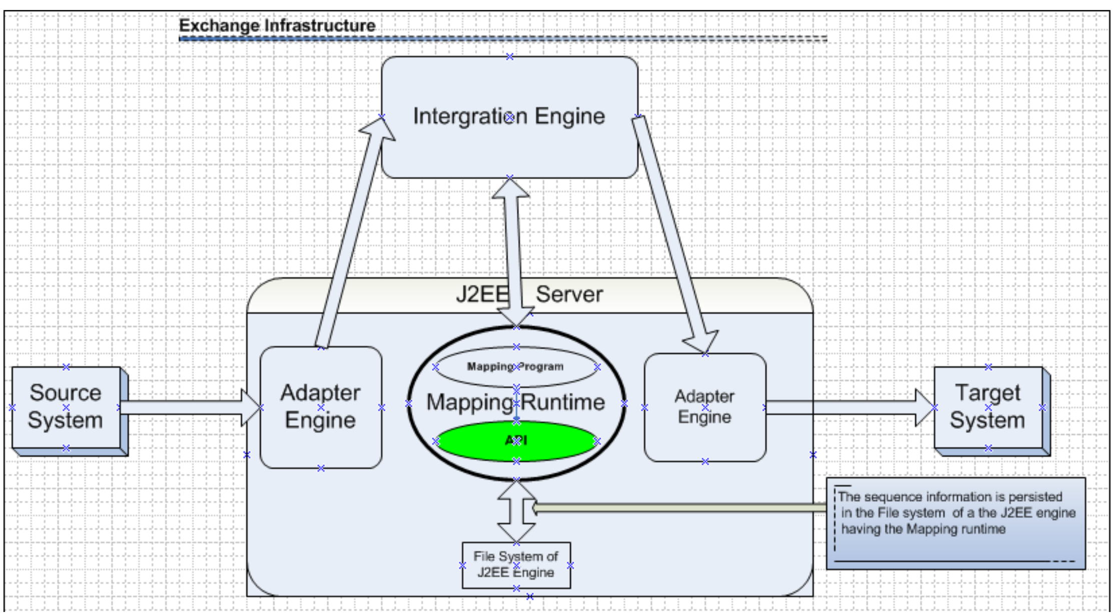
Now the question is how to do this in SAP XI/PI without doing any of the heavyweight steps like RFC, DB or web service look-up?

The Solution

Architectural Explanation

The solution involves introducing some kind of data source which can be accessed from the mapping runtime. Then the idea is to have access to a file data source (Properties file in this case) through an API interface which can then be reused. The java class "NumberGenerator" (see below for the code snippet) uses the mapping runtime to persist the sequencing information in native file system of the SAP J2EE server node instance in the form of properties file. The same information is re-read and updated during consecutive execution of mappings.

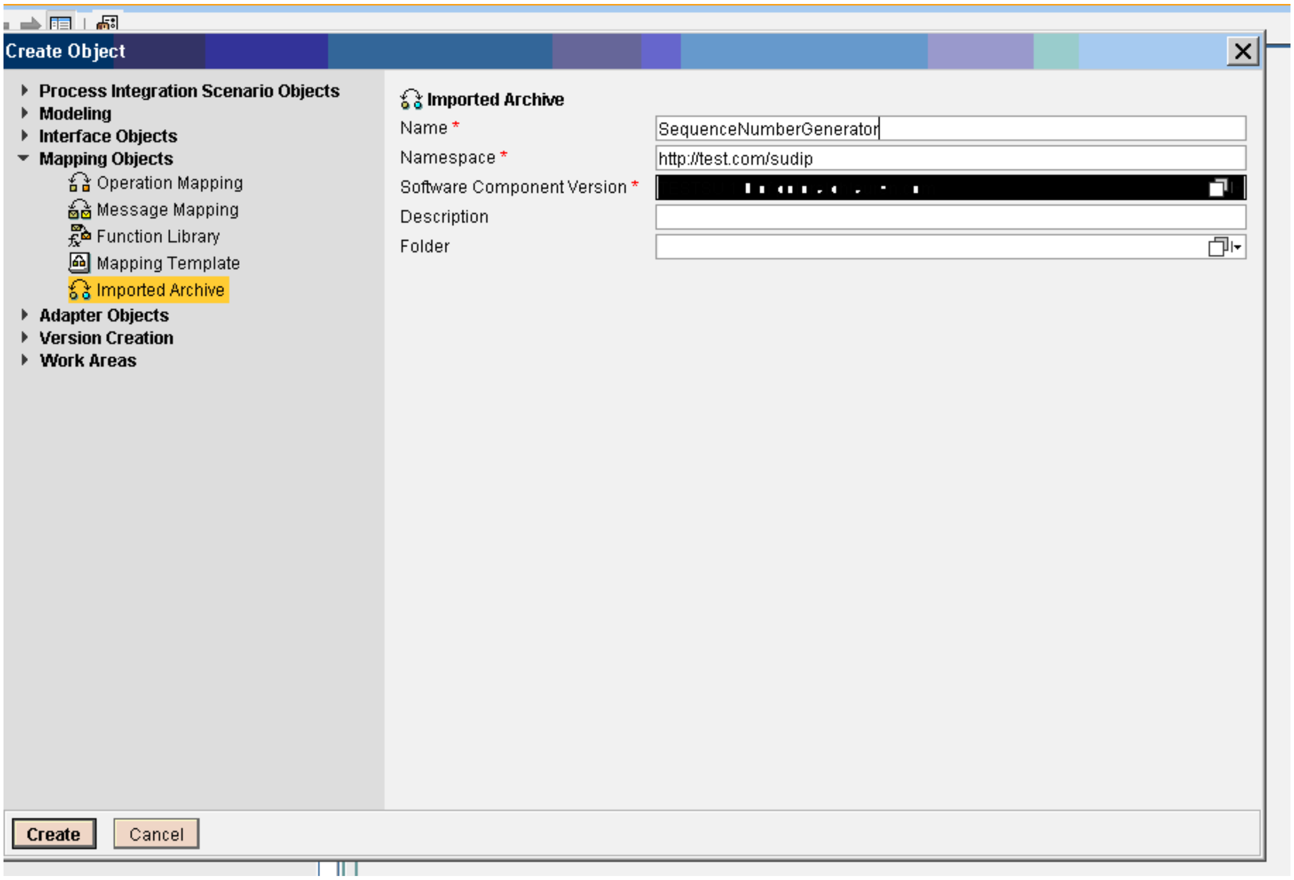
The architecture looks as follows,



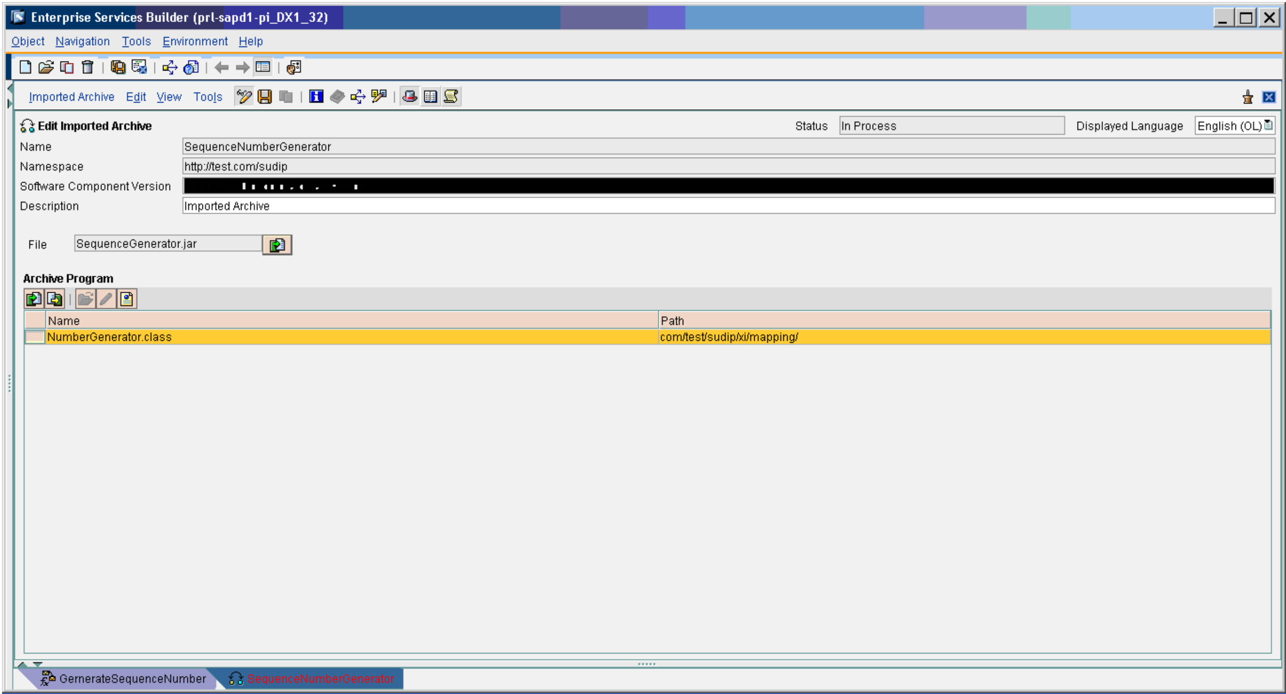
How to use:

The steps illustrated below shows how to use this functionality with an XI/PI graphical mapping using a User Defined Function.

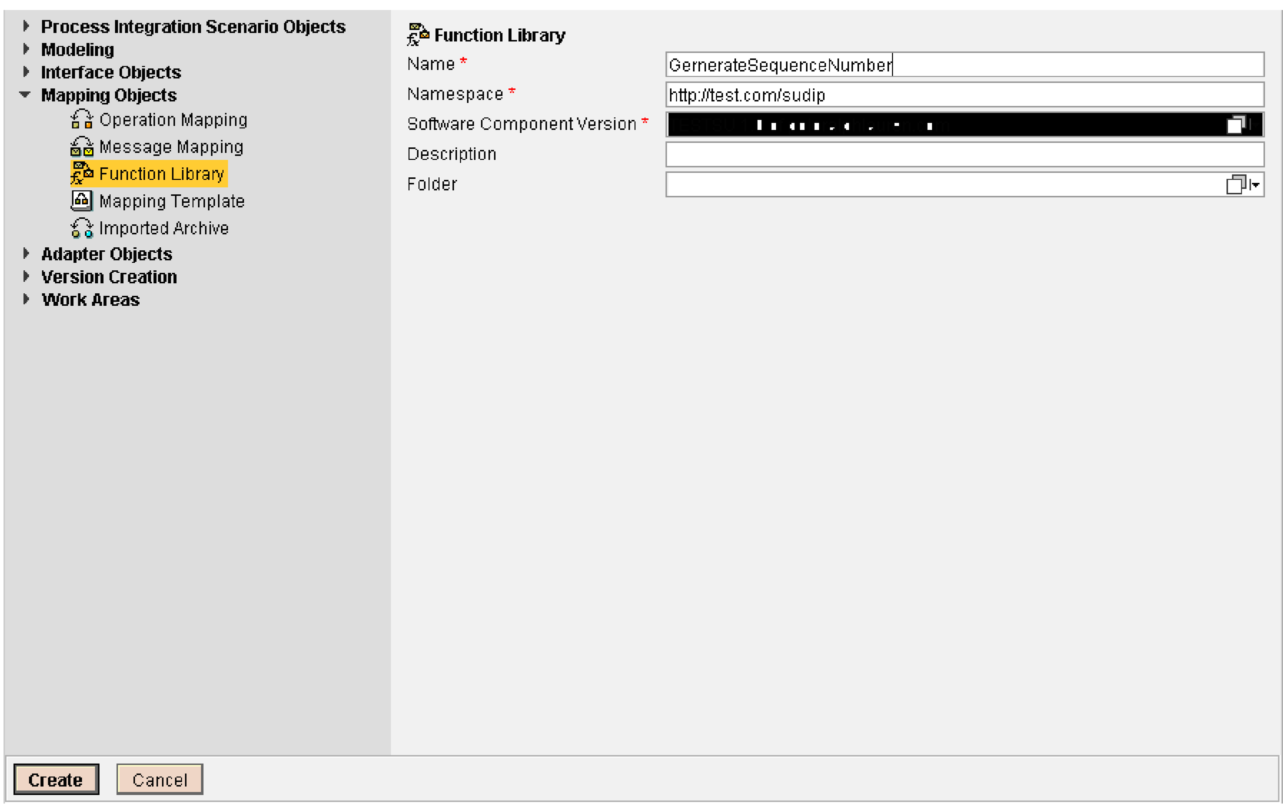
1. Create an Import Archive object in the appropriate namespace.

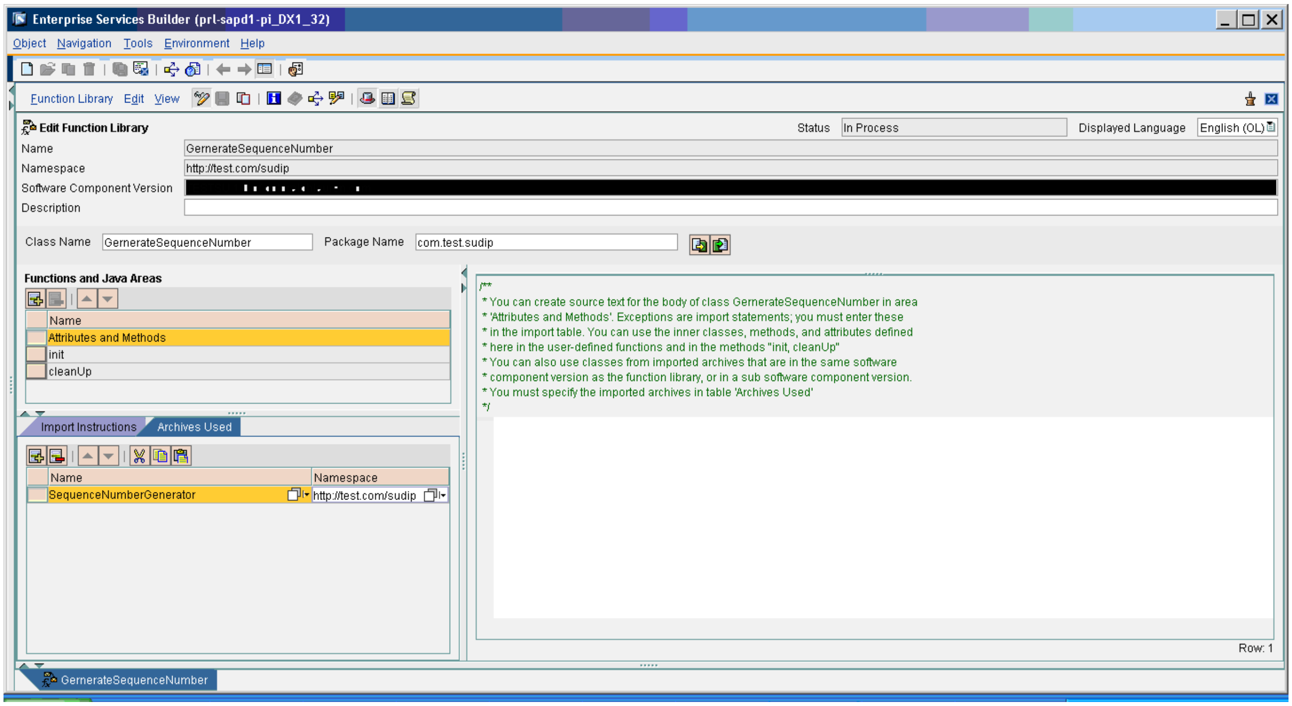


2. Import the Jar file containing the API implementation.

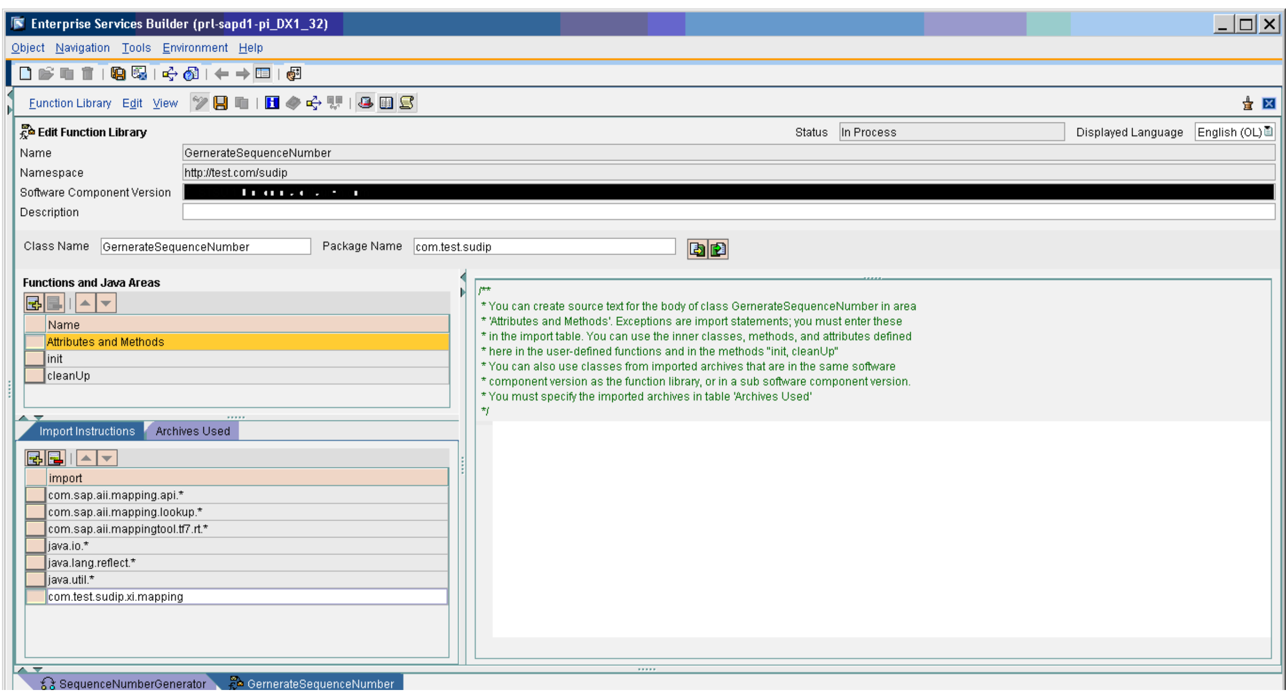


3. Create a reusable function library which can be used in the graphical mapping and refer the imported archive as shown.





4. Declare the import statement for this class here for using it in the mapping and select the created archive containing the api

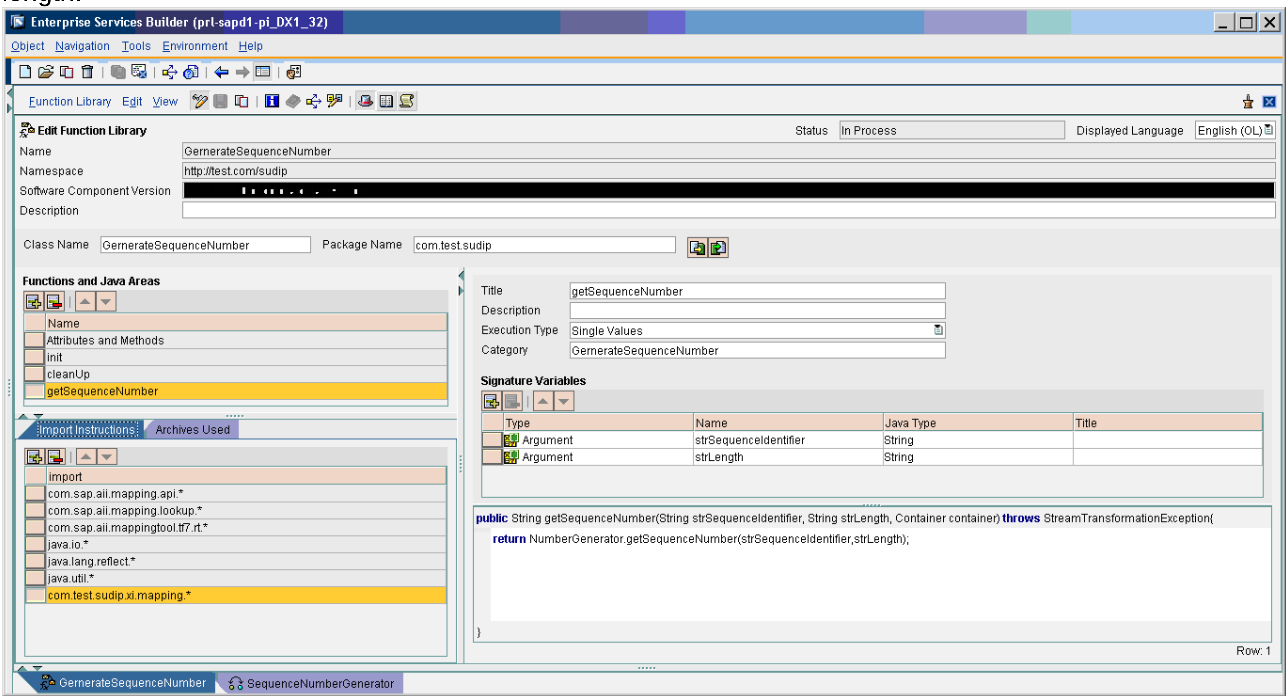


5. Create function in the reusable function library. In this case it is *getSequenceNumber*.

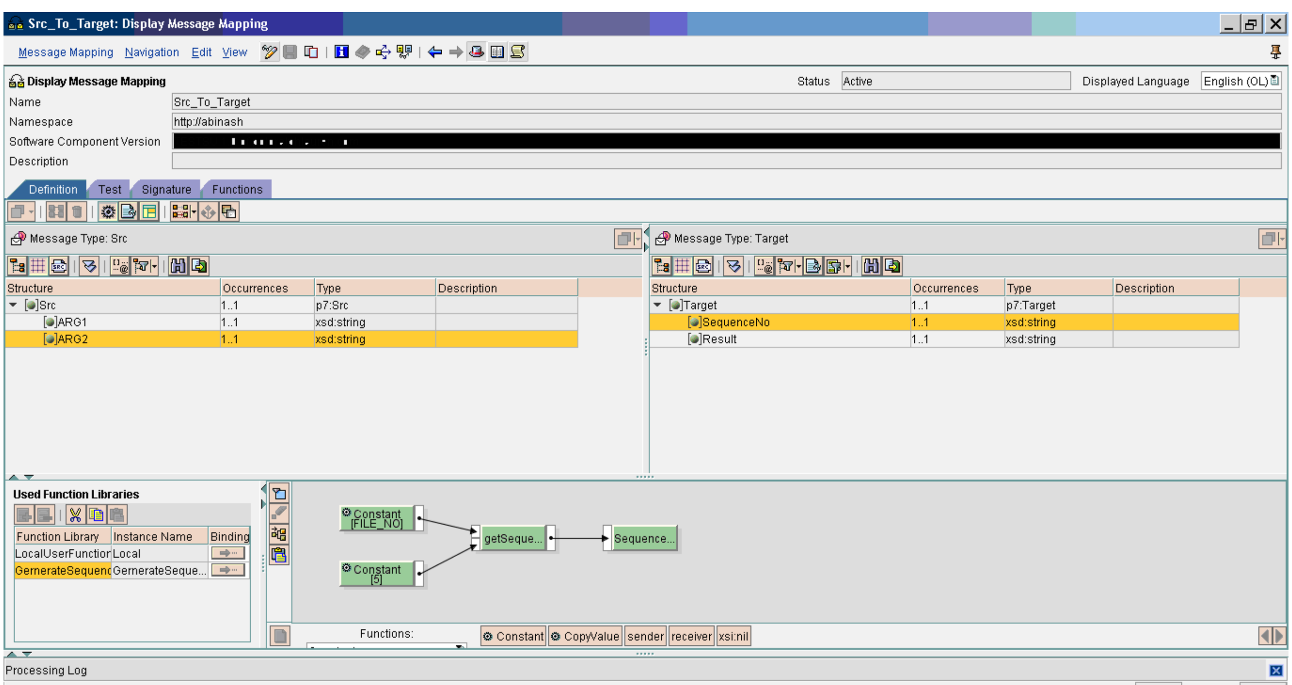
   The function has two arguments,

   strSequenceIdentifier: This is an identifier which identifies the sequence. e.g. in the business which is mentioned as an example above there should be an identifier for the file no, and the same should be used when this particular sequence is required. (In step 6 I use " FILE_NO" as an identifier).

   strLength: The range of the sequence can be specified by this. e.g. if the range required is from 0-999 then 3 should be specified as length here. Similarly for 0-99999, 5 should be specified as length.

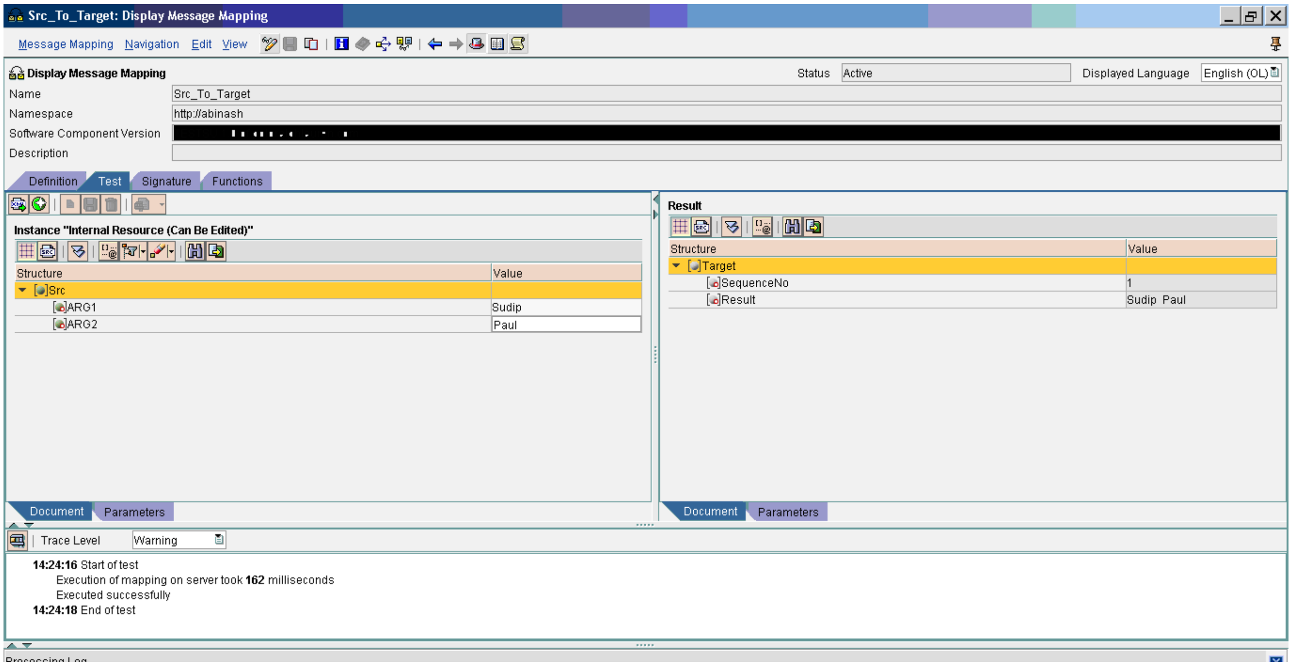


6. Add the function library to the mapping in which this is required.

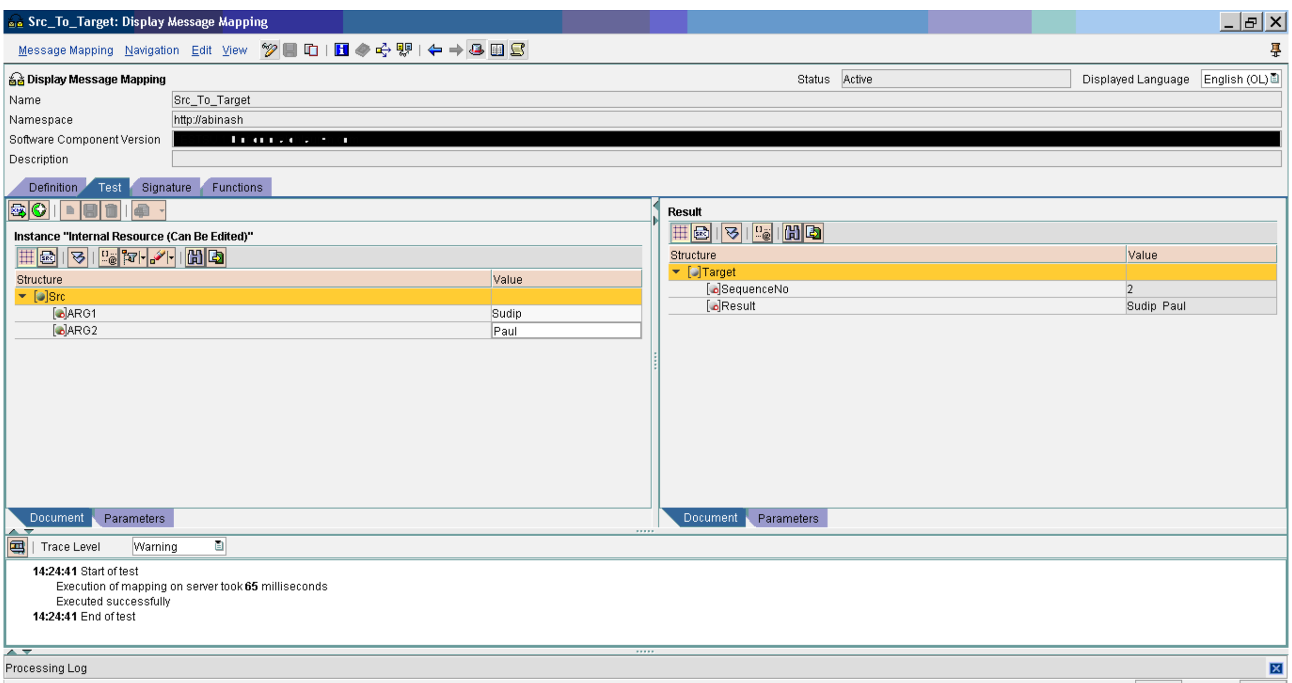


7. This can be tested as follows by executing the mapping in the test tool multiple times. Every time this is executed the next value in sequence is obtained. See the field “SequenceNo” in the target.

First Execution



Second Execution



Using in Java and XSLT mapping

If this required to be used in a java mapping or an XSLT mapping then this can be directly imported via the import statement in the code.

Code Snippet

public static String getSequenceNumber(String formatName,String strLength)

```
            {
                    int ilength = 0;
             Properties properties = new Properties();
             File propertiesfile = new File("Sequence.properties");
                    if(strLength != null && strLength.length() > 0)
                    {
                        ilength = Integer.parseInt(strLength);
                    }
                    else
                    {
                        return "Error:Specify length";
                    }
             try {
                    propertiesfile.createNewFile();
                    properties.load(new FileInputStream(propertiesfile));
             }
                    catch (IOException e)
                    {
                    e.printStackTrace();
                    return "Error:File not read";
             }
             //get the current value of sequence
             String number = properties.getProperty(formatName);
             if (number == null)
                    {
                    number = "0";
             }
             //increment the sequence
             String counter = String.valueOf(Integer.parseInt(number) + 1);
             //compare if the format value length exceed it limits
                    if(counter.length() > ilength)
                {
                    //start the counter again
                    counter = "1";
                }
             //set the current value of sequence
             properties.setProperty(formatName, counter);
             // Write properties file.
             try {
                    //store the values in file
                    properties.store(new FileOutputStream(propertiesfile), null);
             } catch (IOException e) {
                    return "Error:File not updated";
             }
             return counter;
      }
```

Jar

SequenceGenerator.jar

Related Content

https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/20ab7ccf-e78a-2a10-71b4-80fc764bafb0

http://www.sdn.sap.com/irj/scn/weblogs?blog=/pub/wlg/2171

http://www.sdn.sap.com/irj/scn/weblogs?blog=/pub/wlg/2781

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.