delaware XML to CSV converter

General Introduction

SAP Cloud Platform Integration has a built-in XML to CSV converter, but it has a very basic functionality. As experienced SAP Process Integration/SAP Process Orchestration consultants we are used to working with the XML to CSV conversion which is built-in in SAP Process Integration/SAP Process Orchestration.
The SAP Process Integration/SAP Process Orchestration solution has a lot more functionality than the SAP Cloud Platform Integration one hence the idea to recreate the SAP Process Integration/SAP Process Orchestration one ourselves which we can reuse at several Cloud Platform Integration projects.
This guide will explain how to use the custom XML to CSV converter in a SAP Cloud Platform Integration project and how it is developed based on the SAP Process Integration/SAP Process Orchestration solution.
Topics which won't be covered in this guide:

- How to use the Cloud Platform Integration XML to CSV converter
- How to use the SAP Process Integration/SAP Process Orchestration XML to CSV converter
    - The custom solution is based on the SAP Process Integration/SAP Process Orchestration solution and the author assumes that the SAP Process Integration/SAP Process Orchestration solution is known by the reader.
- How the code is developed

Adding the Java code to the project

The converter module is written in Java and can be used in SAP Cloud Platform Integration as an external. When adding this JAR to a groovy script, its classes are available in the groovy script.
The code is written and archived using JAVA 7 (the version currently used by SAP Cloud Platform Integration). The code also uses the JDOM 2.0 JAR.
Both JAR files should be added to the project.

The used JARs:


jdom-2.0.6.jar

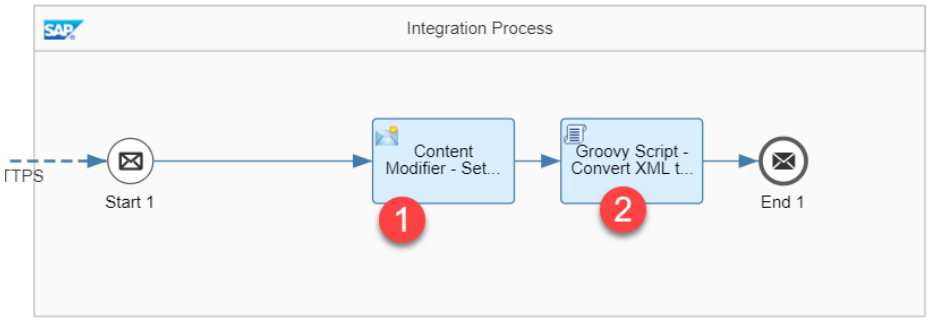

XMLToFlatFileConverter_Java7.jar

Configuring the converter logic

There are 2 steps to be added to the integration flow to configure the functionality of the converter.

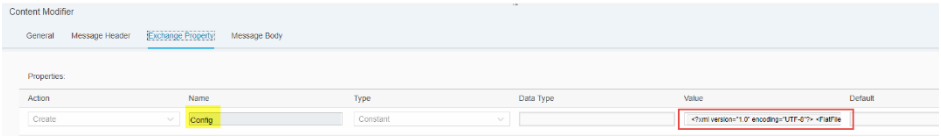
1. Add content modifier with the configuration XML.
2. Add groovy script which reads the input, the configuration and runs the converter module.

Process



Content modifier

An exchange property is added to the flow. The name is should be the same as the one used in the groovy script. The value is the configuration XML which will be elaborated later.



Groovy script

The script will generate the flat file base on the input and the configuration.
Note that the XMLToFlatFileConverter JAR should be added to the imports.

ConvertXMLToFlatFile.groovy

```groovy
/*
 The integration developer needs to create the method processData
 This method takes Message object of package com.sap.gateway.ip.core.customdev.util
which includes helper methods useful for the content developer:
The methods available are:
    public java.lang.Object getBody()
    public void setBody(java.lang.Object exchangeBody)
    public java.util.Map<java.lang.String,java.lang.Object> getHeaders()
    public void setHeaders(java.util.Map<java.lang.String,java.lang.Object> exchangeHeaders)
    public void setHeader(java.lang.String name, java.lang.Object value)
    public java.util.Map<java.lang.String,java.lang.Object> getProperties()
    public void setProperties(java.util.Map<java.lang.String,java.lang.Object> exchangeProperties)
       public void setProperty(java.lang.String name, java.lang.Object value)
*/
import com.sap.gateway.ip.core.customdev.util.Message;
import java.util.HashMap;
import com.delaware.conversion.XMLToFlatFileConverter;
def Message processData(Message message) {
    //Get Source XML
    def source = message.getBody(java.lang.String) as String;

    //Get Configuration
    def map = message.getProperties();
    def config = map.get("Config");

    //Convert source using configuration
    def converter = new XMLToFlatFileConverter();
    def flatfile = converter.run(config, source);

    //Set message body
    message.setBody(flatfile);

    return message;
}
```

Below is the script which was used for this project. It is possible to add extra functionality to the script if needed for your integration flow.

ConvertXMLToFlatFile.groovy

The configuration XML

As mentioned earlier, the configuration XML is based on the SAP Process Integration/SAP Process Orchestration solution. There is an additional tab where you can add some parameters (name + value).

The Configuration XML also consists of parameters with a particular name and value.

```
<?xml version="1.0" encoding="UTF-8"?>
<FlatFileConfig>
    <Parameter>
        <Name>contentType</Name>
        <Value>text/plain;charset=utf-8</Value>
    </Parameter>
    <Parameter>
        <Name>recordSetStructure</Name>
        <Value>Person</Value>
    </Parameter>
    <Parameter>
        <Name>Person.addHeaderLine</Name>
        <Value>2</Value>
    </Parameter>
    <Parameter>
        <Name>Person.fieldSeparator</Name>
        <Value>;</Value>
    </Parameter>
    <Parameter>
        <Name>Person.endSeparator</Name>
        <Value>nl</Value>
    </Parameter>
</FlatFileConfig>
```

The used parameters are taken from the SAP Process Integration/SAP Process Orchestration FFC guide.

https://help.sap.com/saphelp_nwpi711/helpdata/en/44/686e687f2a6d12e10000000a1553f6/content.htm?no_cache=true

Below you can also find the example configuration XML and the XSD schema of the XML.


config.xml

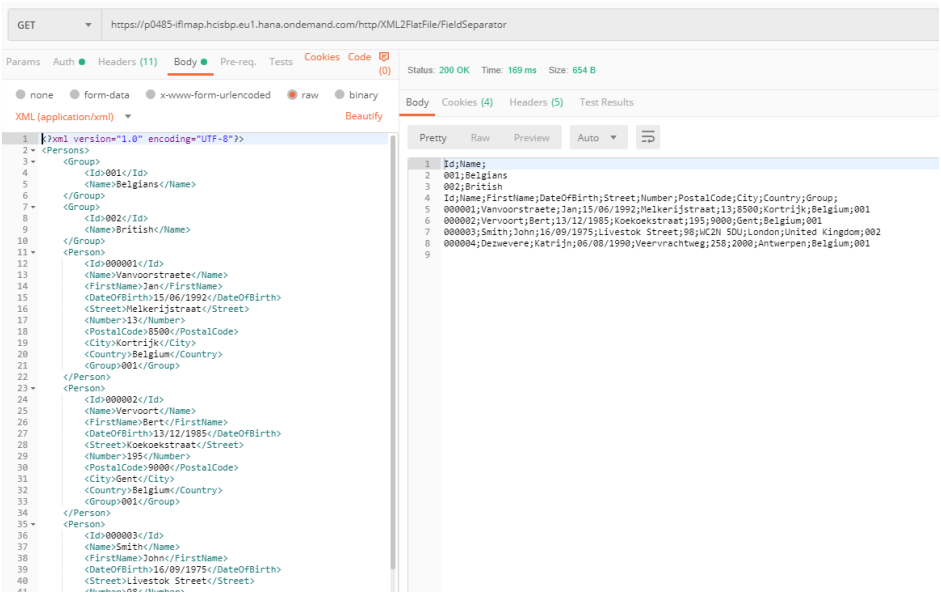

Config.xsd

When the XML is finished, add it to the exchange property of the Integration Flow.

Example

Below postman project contains 2 example requests which are sent to 2 different integration flows. One uses the fixed field length to convert the XML to a flat file, the other one uses a field separator.
The message body is the same for both requests but the response is different due to the different configuration XML in the integration flows. The integration flows are setup in package Utility XML to Flat File Converter by delaware

DLW_CPI_XML2FlatF
ile.postman_collecti



Summary

The Utility XML to Flat File Converter by delaware is a custom module to convert XML messages to flat files in CPI based on the SAP Process Integration/SAP Process Orchestration solution. The reason to develop a custom solution was the lack of functionality in the SAP Cloud Platform Integration solution.