SAP Cloud Integration –

Integration Flow SAP SOAP to ASC X12 - Outbound

Table of Contents

1. Introduction 3
2. Usage Policy and Copyright Statement: ASC X12 and SAP SE 3
3. Integration Flow 4
3.1 Basic Concepts 4
3.2 Sender Channel 4
3.3 Start Event 4
3.4 Read SAP SOAP Parameters 4
Content Modifier 4
3.5 SOAP Pre-Processing 5
XSLT Mapping 5
3.6 SOAP to X12 Mapping 5
XSLT Mapping 5
3.7 X12 Extended Validation (optional) 5
XML Validator 5
3.8 X12 Post-processing 6
XSLT Mapping 6
3.9 Envelope Handling 6
Content Modifier 6
3.10 Transformation Script 8
3.11 XML To EDI Converter 9
Converter 9
3.12 End Event 9
3.13 Receiver Channel 10

1. Introduction

The SAP BTP includes the SAP Cloud Integration, which offers diverse approaches to connect your IT systems with other cloud or on premise system landscapes. This makes cloud integration simple and reliable. Hence it is SAP's strategic integration platform for SAP Cloud customers. It provides out-of-the-box connectivity across cloud and on-premise solutions. Since the SAP Cloud Integration is operated by SAP, you don't need to worry about basic activities. Additionally, SAP is offering prepackaged integration content as reference templates, that allows customers to quickly realize new business scenarios. This drastically reduces integration project lead times and lowers resource consumption significantly.

This document gives an overview about the outbound SOAP to ASC X12 template flow of SAP Cloud Integration in combination with SAP Integration Advisor (IA). It explains how exported runtime artefacts from SAP IA can be imported into the flow and how the flow can be configured.

We assume the reader is an integration developer and is familiar with SAP Cloud Integration.

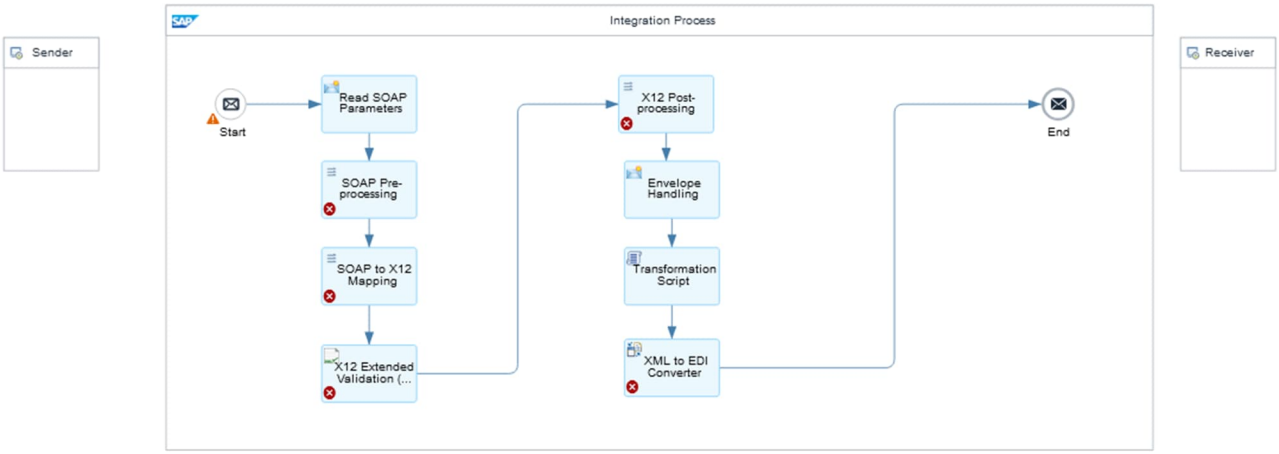
2. Usage Policy and Copyright Statement: ASC X12 and SAP SE

For downloading and using one of the provided ASC X12 message XSD file a valid license for the respective X12 standard is required. Consumers have to be in compliance with ASC X12 IP Usage Policies (http://store.x12.org/store/ip-use).

Copyright Statement for XML Schema Representation generated by SAP SE:

3. Integration Flow

Integration Flow	
Name	SAP SOAP to X12 - Outbound
Description	SAP SOAP to X12 - Outbound Template



3.1 Basic Concepts

With the SAP Integration Advisor one can create MIG (message implementation guidelines) and MAG (mapping guidelines). These can be exported as SAP Cloud Integration runtime artifacts (zip file containing *xslt and *xsd files). The flow templates contain steps serving as containers for the exported runtime artifacts (where the runtime artifacts can be imported into). E.g. the runtime artifacts exported from the MIG and MAG of the SAP Integration Advisor can be used as follows: schemas (xsd) can be used in EDI Splitter, EDI to XML Converter, XML to EDI Converter, XML Validator (extended validation) ; stylesheet transformations (xslt files) in XSLT Mapping.

Furthermore, it is necessary to define and customize the communication adapters as well as the required information of the interchange envelope and header structures (e.g. in the content modifier).

3.2 Sender Channel

Sender channel is configured by the customer. In case of SOAP outbound scenario, an SOAP adapter should be used.

3.3 Start Event

The Start Message event is triggered by the sending system.

3.4 Read SAP SOAP Parameters

Content Modifier

In the flow template, the Message Header is empty because the information which can be stored in header variables depends on the SOAP message. So in case you need to retrieve information from a SOAP payload (different than the one we use in this example), please use a construct similar to the

example shown here (for the SOAP message SupplierInvoiceRequest, present in the type system 'SAP S/4HANA Cloud SOAP' of the 'SAP Integration Advisor').

In this example we store in camel variables the values (from the input message) to be used later during the envelope handling (content modifier, 3.9.). In our example we used Xpath expressions from the SupplierInvoiceRequest message (from the SAP Cloud type system of the SAP Integration Advisor).

Content Modifier	*Description: From the source SOAP, parameters are extracted.*
Message Header (this is only an example, the your Xpath expressions can be different depending on the SOAP message used).	

Action	Name	Type	Data Type	Value	Default
Create	Sender	XPath	java.lang.String	//MessageHeader/SenderParty/InternalID	
Create	Receiver	XPath	java.lang.String	//MessageHeader/RecipientParty/InternalID	

3.5 SOAP Pre-Processing

XSLT Mapping

Mapping	*In this step, the SOAP is preprocessed via an XSLT mapping.*
Name	<*SourceMIGName*>__preproc.xsl
Resource	*Runtime artefact from SAP IA. Located in the MIG source folder within the exported zip file.*
Type	XSLT Mapping
Output Format	XML

3.6 SOAP to X12 Mapping

XSLT Mapping

Mapping	*Mapping step where the SOAP message is transformed into the ASC X12 message via XSLT.*
Name	*<MAGName>.xsl*
Resource	*Runtime artefact from SAP IA. Located at the root folder of the exported zip file.*
Type	XSLT Mapping
Output Format	XML

3.7 X12 Extended Validation (optional)

XML Validator

Mapping	*XML Validation step where the result of the XSLT mapping is validated against the ASC X12 extended validation XSD. Supports XSD 1.1 version.*
Name	*<SourceMIGName>__RD.xsd*
Resource	*Runtime artefact from SAP IA. Located in the MIG target folder within the exported zip file.*

Type	XSLT Mapping
Output Format	XML

This validation step is optional but recommended in case the generated payload needs to be validated against a xsd. E.g. for target message 810 4010, N1.98 qualified “BY” (Buying Party), N4.26 (country code) is restricted using a codelist only to value “DE”.

If you don’t want to execute validation of the message, simply remove this flow step from your integration flow.

3.8 X12 Post-processing

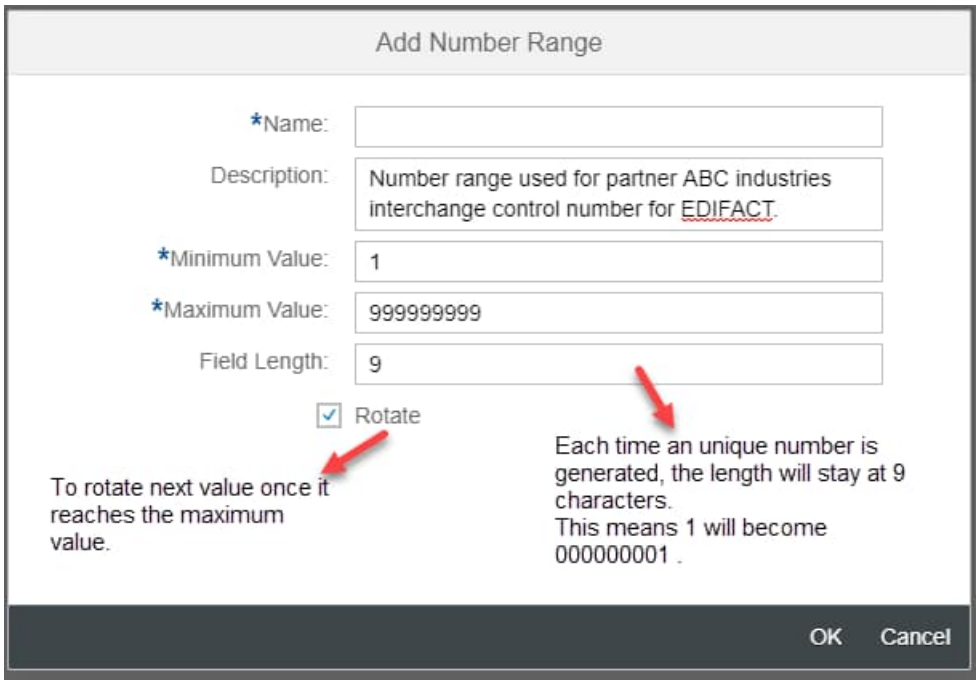
XSLT Mapping

Mapping	*The qualifiers within the target ASCX12 message are removed via an XSLT mapping.*
Name	*<TargetMIGName>__postproc.xsl*
Resource	*Runtime artefact from SAP IA. Located in the MIG target folder within the exported zip file.*
Type	XSLT Mapping
Output Format	XML

3.9 Envelope Handling

Content Modifier

In case the requirement is to generate a new Interchange Control Number each time you are sending an ANSI X12 message, you can use Number Ranges.
To configure Number Ranges, navigate to Monitor->Manage Stores->Number Ranges.
Example of a Number Range object:
A number range can be used to insert unique sequence numbers.



The rest of this paragraph lists the technical details of this Content Modifier used for X12 envelope handling.

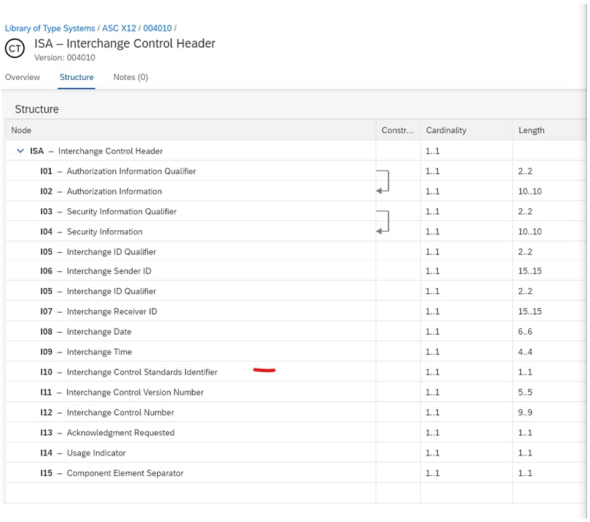
The body of the message is extracted into the header variable *ExtractedMessage.*

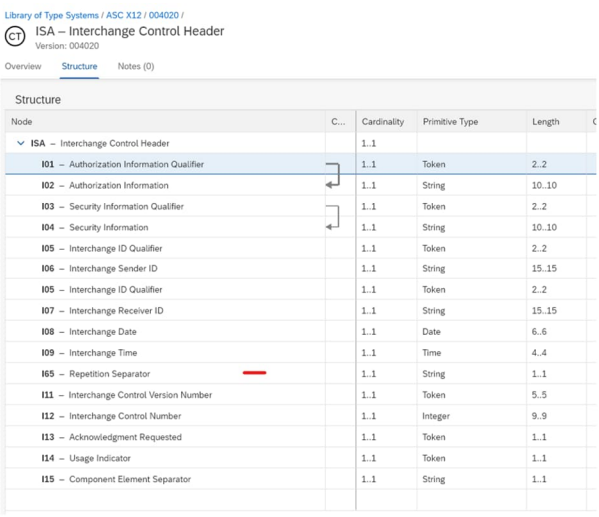
Example: The customer could to set the value for the XPath variable in the following format:
//<TargetMessageTypeName>
Example values are provided here:

Content Modifier					
Message Header (example)					
Action	Name	Type	Data Type	Value	Default
Create	ExtractedMessage	XPath	org.w3c.dom.NodeList	//M_850	
or					
Create	ExtractedMessage	XPath	org.w3c.dom.NodeList	@*|node()	

Example values for the ASC X12 Interchange header are provided in the following table:

Content Modifier					
Exchange Property (example)					
Action	Name	Type	Data Type	Value	Remark
Create	InterchangeControlNumber	Number Range		ICN_X12	Number Range Object
Create	GroupControlNumber	Number Range		GCN_X12	Number Range Object (can be different from ICN)
Create	InterchangeTime	Expression		${date:now:HHmm}	Current system time
Create	InterchangeDate	Expression		${date:now:yyMMdd}	Current system date
Create	Time	Expression		${date:now:HHmm}	
Create	Date	Expression		${date:now:yyyyMMdd}	





The picture above shows the difference between the ISA segments for X12 4010 or below and X12 4020 and above.

3.10 Transformation Script

We use a groovy script to adjust the Interchange Control Reference to the value set above with the Number Range (3.9.).

Content of the script is:

Groovy Script

import com.sap.gateway.ip.core.customdev.util.Message;
import java.util.HashMap;
def Message processData(Message message) {
  def body = message.getBody(new java.lang.String().getClass());
 	
  //Properties
  def properties = message.getProperties();
 	
  def interchangeControlNumber = properties.get("InterchangeControlNumber");
  def groupControlNumber = properties.get("GroupControlNumber");
 	
  def iCNRegex = (/<D_I12>.*?<\/D_I12>/);
  def gCNRegex = (/<D_28>.*?<\/D_28>/);
 	
  def updatedICN = body.replaceAll(iCNRegex, "<D_I12>${interchangeControlNumber}</D_I12>");
  def updatedGCN = updatedICN.replaceAll(gCNRegex, "<D_28>${groupControlNumber}</D_28>");
 	
  message.setBody(updatedGCN);
 	
  return message;
}

3.11 XML To EDI Converter

Converter

XML To EDI Converter	
General	
Name	XML to EDI Converter
X12	
Source Encoding	e.g. UTF-8
EDI Schema Definition	Integration Flow
Schema Name	*ASC-X12_<TransactionSet>_<MessageVersion>.xsd* *Runtime artefact from SAP IA. Located in the MIG target folder within the exported zip file.*

3.12 End Event

The End Message event shoud be connected with the receiving system.

3.13 Receiver Channel

Receiver channel is configured by the customer.