Configuration Guide
TPM Integration Packages with Custom Extensions

As a substitute of “Cloud Integration - Trading Partner Management V2”

Published by “B2B Integration Factory”

July 2024

Table of contents

Disclaimers

The “B2B Integration Factory – [Integration Packages|Pre-Packaged B2B Integration Content]” are Community Content and this is provided as a standalone component under the Apache License, v. 2.0 and, is not part of any SAP product. SAP does not support Community Content. Please raise any issues under the associated GitHub project.

The B2B Integration Factory cannot guarantee this. There is also no guarantee that this content will be continually maintained.

You can find the official release plans for SAP product development on SAP Roadmap Explorer. Here you can also see when the respective features that are considered in the B2B Integration Factory – Integration Packages will be released as a standard feature in the SAP Integration Suite.

If one of these feature is required in the SAP Integration Suite in a timely manner but not listed in the SAP Roadmap Explore, we therefore ask you to create a “Feature Improvement” on SAP Customer Influence – Campaign: SAP Integration Suite or vote for it, if exactly same “Feature Improvement” is already available. All “Feature Improvements” and especially their votes have a significant influence on the prioritization of the features to be delivered in SAP Integration Suite.

Prerequisites

You need a SAP Integration Suite license in where the following capabilities are at least enabled:

1. Cloud Integration
2. Trading Partner Management
3. Integration Advisor
4. API Management

You should undeploy all integration flows as well as the reusable groovy scripts from the integration package “Cloud Integration - Trading Partner Management V2”.

You should also delete this integration package “Cloud Integration - Trading Partner Management V2” including all the content of this integration package.

You should download all the B2B Integration Factory packages that are listed in chapter List of Integration Packages.

Introduction

Thank you for using our [B2B Integration Factory] Integration Packages. These packages offer additional features that are currently not yet supported by the B2B Capabilities @ SAP Integration Suite and the standard integration package "Cloud Integration - Trading Partner Management V2". These features are implemented through extensions in the integration flows or through scripts incorporated into the flows. The extensions are regularly reviewed against the released standard features in the B2B Capabilities @ SAP Integration Suite and will be removed once a standard feature covers the exact same use case. The B2B Integration Factory will also be adding further extensions to the Integration Packages in upcoming releases based on demand. You can influence this decision by submitting a request on the provided GITHub page for the [B2B Integration Factory] Integration Packages.

Below is a list of all the [B2B Integration Factory] Integration Packages. All the packages do belong together, therefore, it is recommended to copy all the packages into the dlocal tenant. Please note that there are two different types of responsibilities for integration packages:

1. Integration packages developed, maintained, and supported by SAP are classified with "SAP" in the list below. The B2B Integration Factory takes bug fixes very seriously and implements them promptly upon error reports via GITHub. However, response times may not adhere to SAP Standard Support. We do not recommend customers to make changes to these packages. If customers choose to do so, they are fully responsible for upgrades and fixing any errors.
2. Integration packages that are classified with "Customer" consist of sample integration flows developed by SAP. These sample integration flows demonstrate how additional customer specific extensions can be implemented and connected through ProcessDirect at the respective extension points in the main integration flows. These sample integration flows are managed and supported by SAP. However, they can be enhanced or copied by the customer to create new integration flows which shifts full responsibility to the customer.

Whether you choose the "SAP" or "Customer" integration packages, they are designed to work well together, ensuring high compatibility and interoperability. Using these packages allows you to customize your B2B processing to meet your specific needs, even if the standard does not provide such features yet. If you use the [B2B Integration Factory] Integration Packages, you will also receive information about when the standard will cover these additional features and how you can upgrade to the standard version.

List of Integration Packages

Integration Package	Description	Artifacts	Responsibility
[B2B Integration Factory] Cloud	Main Integration Package which substitutes the standard main integration package: "Cloud Integration -	7	SAP

Integration - Trading Partner Management	Trading Partner Management V2". It covers the generic integration flows for "SAP Cloud Integration - Trading Partner Management" including additional and relevant extension points as well as custom features which are not provided by the standard solution at this point of time. It contains the script collections and integration flows needed for processing messages between B2B partners dynamically based on configurations done in Trading Partner Management or in additional integration flows that are connected via the provided extension points. This Integration Package should be used, if at least one of the provided custom extension is required.		
[B2B Integration Factory] Communication Receiver Flows (Custom)	This integration package provides custom communication receiver flows, which can be connected via ProcessDirect at the receiver communication step of a TPA --> Business Transaction Activity. SAP provides some sample communication flows such as for sending target interchange/message payloads via Email or storing them at a SFTP server. It also includes an integration flow, called "Step 3b - B2B Simulation.Receiver" which is necessary if you want to do an end-to-end simulation via an API tool in where you would like to see the result (target interchange/message payloads) in the HTTP response. Furthermore, you'll find a template, which you can use for the creating of your own communication receiver flow.	4	Company
[B2B Integration Factory] Communication Sender Flows	This integration package provides sender communication flows for the TPM B2B communication, which are provided by SAP. These sender communication flows are connected to the Step 1b "Step 1b - Sender Interchange Header Extraction Flow".	3	Company
[B2B Integration Factory] Extended Interchange Processing Flows	This integration package provides all the flows and scripts which are necessary for fulfilling all the requirements that are not supported by the standard yet.		
[B2B Integration Factory] Pre and	This integration package is used to collect all the pre- and post-processing related flows, which are required if a Customized Pre- or Post-Processing is enabled in a	n	Company

Post-Processing Flows (Custom)	TPA --> Business Transaction Activity. This integration package provides a number of example integration flows as well as a templates, which can		
[B2B Integration Factory] Interchange Assembly Flows	This package includes all integration flows, which are responsible for the final assembly of target (receiver) interchange/message payloads according to the conventions of the considered type system.	7	Company
[B2B Integration Factory] Interchange Extraction Flows	This integration package provides all integration flows for extracting the relevant parameters and key fields from the headers depending of the supported type systems. It also includes the "Extraction Value Mapping" entries.	6	Company

Integration Flows within the Integration Packages

[B2B Integration Factory] Cloud Integration - Trading Partner Management

Integration Flow	Description	Configurable Parameters
Extended Groovy Scripts	Extended Script collection with all additional scripts that are required to support additional requirements, which are not part of the Reusable Script Collection yet	N/A
Reusable Groovy Scripts	Script collection aligned to SAP standard script collection from "Cloud Integration - Trading Partner Management V2". It provides scripts with the needed reusable functionalities (events, PartnerDirectory lookup, log, B2B monitor, number ranges...)	N/A
Step 1 - Sender AS2 MDN Flow	Receives AS2 MDNs (Message Disposition Notifications), checks if this MDN correlates with a submitted AS2 message and	Sender: Address: /tpm/b2b/mdn

	submits the MDN status to the B2B monitoring.	Remark: This is the suffix of the address on where the MDNs should be sent.
Step 1b - Sender Interchange Header Extraction Flow	Gets interchange/message payload from sender communication flow via ProcessDirect, identifies type system, calls the appropriate extraction flows or steps and writes the payload together with the extracted header parameters into message queue for the further processing in step 2. In case of in case of XML based type systems, it calls separate extraction flows.	Sender: Address: /tpm/sender/com/flow Remark: This is the ProcessDirect address on where the sender communication flows must hand over the messages so that these will get further processed by the Step 1b flow. Receiver: Step_2_Flow Queue Name: SAP_TPM_INBOUND_Q Remark: This is the message queue name of the message queue in where the Step 1b flow hands over the messages to the Step 2 flow. More: KeyChange_EVM_Name: ${header.SAP_EDI_Document_Standard} KeyChange_EVM_SourceParameters: SAP_EDI_Sender_ID,SAP_EDI_Message_Type KeyChange_EVM_TargetParameters: SAP_EDI_Extracted_Keys_Change_ ProcessDirectAddress Remark: These are the relevant parameters for dynamically calling further extension flows who should a modification of the extracted key fields of an EDI (ASC X12, UN/EDIFACT or TRADACOMS) specific interchange payload . These parameters belong to the extension [B2BIFACTORY-86] Modification of EDI extracted key values.

Step 2 - Interchange Processing Flow	Picks incoming interchange/message payloads from inbound queue (SAP_TPM_INBOUND_Q) and processes it according the dynamically obtained artefacts from PD. For this purpose, the extracted key values from step 1b are used to calculate a unique PD identifier (PID). This PID is then used to obtain the corresponding artifacts such as parameters, schemas and scripts from the PD into the runtime and process them accordingly. Processing includes, for example, pre- and post-processing, mapping, as well as the assembly of the whole target interchange/message payload. This assembled payload is then transferred to step 3 via a further message queue.	Sender: Queue Name: SAP_TPM_INBOUND_Q Remark: This is the message queue name of the message queue in where Step 2 flow picks the messages which were persisted by the Step 1b flow. Receiver: Queue Name: SAP_TPM_OUTBOUND_Q Remark: This is the message queue name of the message queue in where Step 2 flow hands over the messages to Step 3 flow.
Step 2b - Interchange Processing Flow - Base-Overlay Mapping Process	Base-Overlay Mapping is divided into two mapping steps, the base and the overlay-mapping. The base mapping generates the common parts of a target payload, which are relevant by the most of the trading partners. And the overlay-mapping generates a delta part of the target payload which considers the individual requirements per trading partner. Both outputs are then merged into a single final target payload. Both base and overlay mappings map to the same source and target message types based on the same type systems. However, there is a difference in the versions of the business partner-relevant message structures (the so called Overlay MIGs). Therefore, the base mapping	Sender: Address: /tpm/process/base-overlay-mapping Remark: This is the ProcessDirect address, which will be called when a base-overlay mapping process is required.

	considers at the trading partner side a structure that is based on the latest agreed version, and the overlay mapping considers here the trading partner-specific version.	
Step 3 - Receiver Communication Flow	Picks the assembled interchange/message payloads from outbound queue (SAP_TPM_OUTBOUND_Q) and sends them to the final receiver (either trading partner or company system) by the configured communication protocol.	Sender: Queue Name: TPM_OUTBOUND_Q Receiver: For receiver: Trigger_BT_Response (triggering the response of a business transaction) Address: /tpm/sender/com/flow Remark: This is the message queue name of the message queue in where Step 2 flow picks the messages which were persisted by the Step 1b flow.

[B2B Integration Factory] Communication Sender Flows

Integration Flow	Description	Configure
Step 1a - AS2.Sender	Receives messages via AS2 protocol, identifies type system and writes payload and header parameters into message queue.	Sender: Address: /ms2/tpm/b2b/as2 (Default addess: /tpm/b2b/as2) Receiver: Address: /tpm/sender/com/flow
Step 1a - B2B Simulation.Sender	Simulates the end-to-end flow for the processing of an interchange via a B2B business transaction activity using HTTPS. It returns the result back to the sender via a synchronous HTTP response.	Sender: Address: /tpm/b2b/simulate Receiver: Address: /tpm/sender/com/flow

Step 1a - HTTP.Sender	Receives messages via HTTP/HTTPS protocol and provides a response in synchronous call, if needed.	Sender: Address: /tpm/b2b/http Receiver: Address: /tpm/sender/com/flow More: For fetching the correlation data of a synchronous HTTP response Data Store name: SAP_B2B_HTTP_ SynchronousResponsePayloads
Step 1a - IDOC.Sender	Receives messages via IDOC protocol, identifies type system and writes payload and header parameters into message queue.	Sender: Address: /ms2/tpm/b2b/idoc/ User Role: ESBMessaging.send Receiver: Address: /tpm/sender/com/flow
Step 1a - SFTP.Sender	Receives messages via SFTP protocol, identifies type system and writes payload and header parameters into message queue.	Sender: Address: /tpm/b2b/sftp/dummy Receiver: Address: /tpm/sender/com/flow More: Directory: dummy File_name: <empty> User_name: dummy_user Private_Key_Alias: dummy_private_key_alias
Step 1a - SOAP.Sender	Receives messages via SOAP protocol, identifies type system and writes payload and header parameters into message queue.	Sender: Address: /ms2/tpm/b2b/soap

		Receiver: Address: /tpm/sender/com/flow
Step 1a - TEMPLATE.Sender	Receives messages via any supported communication protocol, identifies type system and writes payload and header parameters into message queue	

[B2B Integration Factory] Communication Receiver Flows (Custom)

Integration Flow	Description	Configure
Step 3b - B2B Simulation.Receiver	This iflow represents the endpoint that communicates with the sender iflow, accepts incoming data, transforms it if required, and routes it to the appropriate place within the B2B system.	Receiver: Address: /tpm/b2b-simulation/receiver
Step 3b - SFTP.Receiver	Picks interchanges from outbound queue and sends them to the final receiver via SFTP. Here it is TP specific because a custom header pattern must be passed which is per default not required	Receiver: Address: /tpm/sftp/receiver
Step 3b - eMail.Receiver	Picks interchanges from outbound queue and sends them to the final receiver via eMail. Here it is TP specific because a custom header	Receiver: Address: /tpm/email/receiver

	pattern must be passed which is per default not required	
Step 3b - TEMPLATE.Receiver	Picks interchanges from outbound queue and sends them to the final receiver by the agreed communication protocols. With this Template receiver iflow, you can create your own iflow using the supported communication protocol.	

[B2B Integration Factory] Extended Interchange Processing Flows

Integration Flow	Description	Configure
UN-EDIFACT Extracted Key Value Change	This Flow is responsible to change the extracted UN-EDIFACT key values, which is needed to match the unique PID.	Sender: Address: /tpm/un-edifact/extraction/substitute/key-values
UN-EDIFACT Functional Acknowledgement Interchange Modification	This Flow is responsible to modify the functional acknowledgement to fit the requirement of the specific partner	Sender: Address: /tpm/un-edifact/functional-acknowledgement-modification
UN-EDIFACT Interchange Modification	This flow is responsible to modify the EDI Interchange payload before they are processed by the EDI Splitter step	Sender: Address: /tpm/un-edifact/Interchangemodification

[B2B Integration Factory] Pre and Post-Processing Flows (Custom)

Integration Flow	Description	Configure
Post-Processing . _BASE_ . 04 - Invoice - Outbound. UN-EDIFACT	Transforms the Target Structure to the Overlay Target Structure Version for Both Base as well as Overlay. And this is applicable for all BASE Outbound Scripts	Sender: Address: /tpm/post-processing/_base_/04-invoice-outbound/un-edifact
Pre-Processing . _BASE_ . 01 - Purchase Order - Inbound . UN-EDIFACT	Transforms the Overlay Structure to the BASE Structure Version	Sender: Address: /tpm/pre-processing/_base_/01-purchaseOrder-inbound/un-edifact

[B2B Integration Factory] Interchange Assembly Flows

Integration Flow	Description	Configure
ASC-X12 Interchange Assembly Process	Assembles the ASC X12 interchange payload and inserts the TPA Business Transaction Activity parameters into the relevant ASC X12 header elements.	Sender: Address: /tpm/assembly-flow/ASC_X12
CSV Interchange Assembly Process	Assembles the CSV message payload and inserts the TPA Business Transaction Activity parameters into the relevant header elements.	Sender: Address: /tpm/assembly-flow/CSV
GS1XML Interchange Assembly Process	Assembles the GS1XML interchange payload and inserts the TPA Business Transaction Activity parameters into the relevant header elements.	Sender: Address:/tpm/assembly-flow/GS1-XML

SAP_S4HANA_OnPremise_SOA Interchange Assembly Process	Assembles the SAP SOAP message payload and inserts the TPA Business Transaction Activity parameters into the relevant header elements.	Sender: Address: /tpm/assembly-flow/SAP_S4HANA_OnPremise_SOA
SAP-IDoc Interchange Assembly Process	Assembles the SAP IDOC message payload and inserts the TPA Business Transaction Activity parameters into the relevant header elements.	Sender: Address: /tpm/assembly-flow/SAP_IDoc
TRADACOMS Interchange Assembly Process	Assembles the TRADACOMS message payload and inserts the TPA Business Transaction Activity parameters into the relevant header elements.	Sender: Address: /tpm/assembly-flow/TRADACOMS
TradeXML Interchange Assembly Process	Assembles the TradeXML message payload and inserts the TPA Business Transaction Activity parameters into the relevant header elements.	Sender: Address: /tpm/assembly-flow/TradeXML
UN-EDIFACT Interchange Assembly Process	Assembles the UN/EDIFACT interchange payload and inserts the TPA Business Transaction Activity parameters into the relevant header elements.	Sender: Address: /tpm/assembly-flow/UNEDIFACT
TEMPLATE Interchange Assembly Process	This template iflow	

[B2B Integration Factory] Interchange Extraction Flows

Name	Description	Configure
_ Get flat based Type System	This iflow identifies the flat based payload based on syntax type, which can't be identified by the EDI extractor or the XPath expression.	Sender: Address: /tpm/extract/flat/type-system
_ Get XML based Type System	This external iflow is called, if the payload is based on	Sender: Address: /tpm/extract-type-system-flow/xml

	XML and a type system is not set by the step 1a flow.	
Extraction Value Mapping	This value mapping includes all value mappings for the diverse type system related extraction rules. For e.g. map several source sender ids to one target sender id which is for calculation of the PID	See separate chapter

	of the corresponding PD entry.	
CSV-Custom Interchange Extraction Flow	This flow extracts the necessary parameters from the relevant CSV header elements and provides it as exchange header parameters. It also calls the Extraction Value Mapping for substituting parameters that	Sender: Address:/tpm/extraction-flow/{{SAP_EDI_Document_Standard}}

	will fit to the expected PD String	
GS1XML Interchange Extraction Process	This flow extracts the necessary parameters from the relevant GS1XML header elements and provides it as exchange header parameters. It also calls the Extraction Value Mapping for substituting param	Sender: Address: /tpm/extraction-flow/GS1-XML Remark: Below listed are the extraction values which would be passed as exchange header parameters in the iflow. More: SAP_B2B_EVM_Name: GS1XML SAP_B2B_EVM_SourceParameters: SAP_EDI_Message_Version,SAP_EDI_Message_Business_Scope SAP_B2B_EVM_TargetParameters: SAP_EDI_Message_Version,SAP_EDI_Message_Type,SAP_EDI_Message_Type,SAP_EDI_Receiver_Partner_Type,SAP_EDI_Sender_ID_Qualifier,SAP_EDI_Sender_Partner_Type SAP_EDI_Document_Standard: SAP_S4HANA_Cloud_SOA SAP_EDI_Header_Version: //*:StandardBusinessDocumentHeader[1]/*:HeaderVersion[1] SAP_EDI_Interchange_Control_Number //*:StandardBusinessDocumentHeader[1]/*:DocumentIdentification[1]/*:InstanceIdentifier[1] SAP_EDI_Message_Business_Scope: //*:StandardBusinessDocumentHeader[1]/*:BusinessScope[1]/*:Scope[1]/*:Identifier[1] SAP_EDI_Message_Type: //*:StandardBusinessDocumentHeader[1]/*:DocumentIdentification[1]/*:Type[1]

	eters that will fit to the expected PD String.	SAP_EDI_Message_Version: //*:StandardBusinessDocumentHeader[1]/ *:DocumentIdentification[1]/*:TypeVersion[1] SAP_EDI_Receiver_ID: //*:StandardBusinessDocumentHeader[1]/*:Receiver[1]/ *:Identifier[1] SAP_EDI_Receiver_ID_Qualifier: //*:StandardBusinessDocumentHeader[1]/*:Receiver[1]/ *:Identifier[1]/@Authority SAP_EDI_Receiver_Partner_Type: //*:StandardBusinessDocumentHeader[1]/*:Receiver[1]/ *:ContactInformation[1]/*:ContactTypeIdentifier[1] SAP_EDI_Sender_ID: //*:StandardBusinessDocumentHeader[1]/*:Sender[1]/ *:Identifier[1] SAP_EDI_Sender_ID_Qualifier: //*:StandardBusinessDocumentHeader[1]/*:Sender[1]/ *:Identifier[1]/@Authority SAP_EDI_Sender_Partner_Type: //*:StandardBusinessDocumentHeader[1]/*:Sender[1]/ *:ContactInformation[1]/*:ContactTypeIdentifier[1]
SAP-IDOC Interchange Extraction Process	This flow extracts the necessary parameters from the relevant SAP IDOC EDI_DC40	Sender: Address: /tpm/extraction-flow/SAP_Idoc Remark: Below listed are the extraction values which would be passed as exchange header parameters . More: SAP_B2B_EVM_Condition SAP_EDI_Sender_ID SAP_B2B_EVM_Name

	header elements and provides it as exchange header parameters. It als calls the Exraction Value Mapping for substituting parameters that these will fit to the expected PD String.	SAP_IDoc_Default SAP_B2B_EVM_SourceParameters SAP_EDI_Message_Version,SAP_EDI_Message_Code,SAP_EDI_Message_Type SAP_B2B_EVM_TargetParameters SAP_EDI_Message_Version,SAP_EDI_Message_Type,SAP_EDI_Sender_ID,SAP_EDI_GS_Sender_ID, SAP_EDI_Sender_Partner_Type,SAP_EDI_GS_Receiver_ID,SAP_EDI_Receiver_Partner_Type SAP_EDI_Archiving_Indicator (//*:EDI_DC40/*:MANDT)[1] SAP_EDI_Client (//*:EDI_DC40/*:MANDT)[1] SAP_EDI_Customer_Extension (//*:EDI_DC40/*:CIMTYP)[1] SAP_EDI_Direction if ((//*:EDI_DC40/*:DIRECT)[1] = '1') then 'OUTBOUND' else if ((//*:EDI_DC40/*:DIRECT)[1] = '2') then 'INBOUND' else () SAP_EDI_GS_Receiver_ID (//*:EDI_DC40/*:RCVPOR)[1] SAP_EDI_GS_Sender_ID (//*:EDI_DC40/*:SNDPOR)[1] SAP_EDI_Idoc_Type (//*:EDI_DC40/*:IDOCTYP)[1] SAP_EDI_Interchange_Control_Number (//*:EDI_DC40/*:DOCNUM)[1] SAP_EDI_Interchange_Date (//*:EDI_DC40/*:CREDAT)[1] SAP_EDI_Interchange_DateTime replace(concat((//*:EDI_DC40/*:CREDAT)[1], (//*:EDI_DC40/*:CRETIM)[1]), '^(\d{4})(\d{2})(\d{2})(\d{2})(\d{2})(\d{2})$', '$1-$2-$3T$4:$5:$6') SAP_EDI_Interchange_Time (//*:EDI_DC40/*:CRETIM)[1] SAP_EDI_Message_Code

		(//*:EDI_DC40/*:MESCOD)[1]
		SAP_EDI_Message_Function
		(//*:EDI_DC40/*:MESFCT)[1]
		SAP_EDI_Message_Number
		(//*:EDI_DC40/*:DOCNUM)[1]
		SAP_EDI_Message_Type
		concat(if((//*:EDI_DC40/*:CIMTYP)[1] != '') then concat((//*:EDI_DC40/*:CIMTYP)[1], '.') else (), (//*:EDI_DC40/*:MESTYP)[1], if((//*:EDI_DC40/*:IDOCTYP)[1] != '') then concat('.', (//*:EDI_DC40/*:IDOCTYP)[1]) else ())
		SAP_EDI_Message_Version
		(//*:EDI_DC40/*:DOCREL)[1]
		SAP_EDI_Output_Mode
		if ((//*:EDI_DC40/*:OUTMOD)[1] = '4') then 'collect' else if ((//*:EDI_DC40/*:OUTMOD)[1] = '3') then 'collectAndSubsystem' else if ((//*:EDI_DC40/*:OUTMOD)[1] = '1') then 'passSubsystem' else 'pass'
		SAP_EDI_Processing_Priority_Code
		(//*:EDI_DC40/*:EXPRSS)[1]
		SAP_EDI_Receiver_ID
		(//*:EDI_DC40/*:RCVPRN)[1]
		SAP_EDI_Receiver_Logical_Address
		(//*:EDI_DC40/*:RCVLAD)[1]
		SAP_EDI_Receiver_Partner_Function
		(//*:EDI_DC40/*:RCVPFC)[1]
		SAP_EDI_Receiver_Partner_Type
		(//*:EDI_DC40/*:RCVPRT)[1]
		SAP_EDI_Receiver_Routing_Address
		(//*:EDI_DC40/*:RCVSAD)[1]
		SAP_EDI_Sender_Group_Reference_Number
		(//*:EDI_DC40/*:REFGRP)[1]
		SAP_EDI_Sender_ID
		(//*:EDI_DC40/*:SNDPRN)[1]

		SAP_EDI_Sender_Logical_Addess (//*:EDI_DC40/*:SNDLAD)[1] SAP_EDI_Sender_Message_Reference_Number (//*:EDI_DC40/*:REFMES)[1] SAP_EDI_Sender_Partner_Function (//*:EDI_DC40/*:SNDPFC)[1] SAP_EDI_Sender_Partner_Type (//*:EDI_DC40/*:SNDPRT)[1] SAP_EDI_Sender_Routing_Address (//*:EDI_DC40/*:SNDSAD)[1] SAP_EDI_Serialization (//*:EDI_DC40/*:SERIAL)[1] SAP_EDI_Standard_Flag (//*:EDI_DC40/*:STD)[1] SAP_EDI_Standard_Message_Type (//*:EDI_DC40/*:STDMES)[1] SAP_EDI_Standard_Version (//*:EDI_DC40/*:STDVRS)[1] SAP_EDI_Status (//*:EDI_DC40/*:STATUS)[1] SAP_EDI_Transmission_File (//*:EDI_DC40/*:REFINT)[1] SAP_EDI_Usage_Indicator (//*:EDI_DC40/*:TEST)[1]
SAP-SOAP Interchange Extraction Process	This flow extracts the necessary parameters	Sender: Address: /tpm/extraction-flow/ SAP-SOAP Remark: Below listed are the extraction values which would be passed as exchange header parameters.

	from the relevant SOAP header elements and provides it as exchange header parameters. It also calls the Extraction Value Mapping for substituting parameters that these will fit to the expected PD String.	More: SAP_B2B_EVM_Name SAP-SOAP SAP_B2B_EVM_SourceParameters SAP_EDI_Sender_ID,SAP_EDI_Receiver_ID SAP_B2B_EVM_TargetParameters SAP_EDI_Sender_ID,SAP_EDI_Receiver_ID SAP_EDI_Interchange_Control_Number //*:MessageHeader[1]/*:ID[1] SAP_EDI_Interchange_DateTime //*:MessageHeader[1]/*:CreationDateTime[1] SAP_EDI_Message_Business_Scope //*:MessageHeader[1]/*:BusinessScope[1]/*:ID[1] SAP_EDI_Message_Number //*:MessageHeader[1]/*:ID[1] SAP_EDI_Message_Type local-name(/node()[1]) SAP_EDI_Message_Version 1809_FPS02 SAP_EDI_Receiver_ID //*:MessageHeader[1]/*:RecipientParty[1]/*:InternalID[1] SAP_EDI_Receiver_ID_Agency_ID //*:MessageHeader[1]/*:RecipientParty[1]/*:InternalID[1]/@schemeAgencyID SAP_EDI_Receiver_ID_Qualifier //*:MessageHeader[1]/*:RecipientParty[1]/*:InternalID[1]/@schemeID SAP_EDI_Sender_ID //*:MessageHeader[1]/*:SenderBusinessSystemID[1] SAP_EDI_Sender_ID_Agency_ID //*:MessageHeader[1]/*:SenderParty[1]/*:InternalID[1]/@schemeAgencyID SAP_EDI_Sender_ID_Qualifier

		//*:MessageHeader[1]/*:SenderParty[1]/*:InternalID[1]/@schemeID SAP_EDI_Usage_Indicator if(//*:MessageHeader[1]/*:TestDataIndicator[1] = 'true') then 'test' else 'productive'
TradeXML Interchange Extraction Process	This flow extracts the necessary parameters from the relevant TradeXML header elements and provides it as exchange header parameters. It also calls the Extraction Value Mapping for substituting parameters that	Sender: Address: /tpm/extraction-flow/SAP_S4HANA_Cloud_SOA Remark: Below listed are the extraction values which would be passed as exchange header parameters in the iflow. More: SAP_B2B_EVM_Name TradeXML SAP_B2B_EVM_SourceParameters SAP_COM_SND_Adapter_Type,SAP_EDI_Message_Type SAP_B2B_EVM_TargetParameters SAP_COM_SND_Adapter_Type,SAP_EDI_Document_Standard,SAP_EDI_Message_Version,SAP_EDI_Sender_ID,SAP_EDI_Receiver_ID,SAP_EDI_Trigger_Response SAP_EDI_Document_Standard SAP_S4HANA_Cloud_SOA SAP_EDI_Interchange_Control_Number //*:TransportEnvelope/*:UniqueTransportID SAP_EDI_Interchange_DateTime replace(concat(//*:TransportEnvelope/*:CreationDate/*:Year, '-', //*:TransportEnvelope/*:CreationDate/*:Month, '-', //*:TransportEnvelope/*:CreationDate/*:Day, 'T', //*:TransportEnvelope/*:CreationDate/*:Hour, ':', //*:TransportEnvelope/*:CreationDate/*:Minute,':', //*:TransportEnvelope/*:CreationDate/*:Second), '^(\d{4})(\d{2})(\d{2})(\d{2})(\d{2})(\d{2})$', '$1-$2-$3T$4:$5:$6') SAP_EDI_Message_Type concat('TradeXML_', //*:TransportEnvelope/*:DocumentClassification) SAP_EDI_Message_Version 'TRADEXML_2.0.0'

	these will fit to the expect ed PD String.	SAP_EDI_Receiver_ID //*:TransportEnvelope/*:Routing/*:To SAP_EDI_Sender_ID //*:TransportEnvelope/*:Routing/*:From

[B2B Integration Factory] Extended Groovy Scripts

Name	Description
BusinessDocumentSentEvent	This script sets the Completed state in the B2B Monitoring in case of successful interchange.
callPDandGetCustomParameters	This script calls the PD for getting the custom parameters which can be used in the customized pre-/post-processing flows.
callPDforBaseOverlayMappingProcess	This script calls the PD for getting the relevant parameters for the Base-Overlay mapping process.
callPDforDefaultMappingProcess	This script calls the PD for getting the relevant parameters for the default mapping process.
callPDforInterchangeAssembly	This script calls the PD for getting the relevant parameters for inserting it into the related interchange and message headers.
callPDforReceiverCommunication	This script calls the PD for getting the relevant parameters from the Partner Directory.
ExtractCustomParameterFromPDInReceiver	This script is a extended callPDwithHeaderParameters.groovy script, which is optimized for the BOM (Base-Overlay Mapping) approach to extract custom Parameter search attributes.

ExtractCustomXpathFromPDInReceiver	This script is a extended callPDwithHeaderParameters.groovy script, which is optimized for the BOM (Base-Overlay Mapping) approach to extract custom Xpath search attributes from the Receiver Interchange.
ExtractCustomXpathFromPDInSender	This script is a extended callPDwithHeaderParameters.groovy script, which is optimized for the BOM (Base-Overlay Mapping) approach to extract custom Xpath search attributes from the Sender Interchange.
extractionValueMapping	This value mapping includings all value mappings for the diverse type system related extraction rules. For example map several source sender ids to one target sender id which is for calculation of the PID of the corresponding PD entry.
getAS2HeaderParameters	This script is used in the customized AS2 receiver communication flow for getting the necessary AS2 communication parameters from Partner Directory and providing these as Camel exchange properties.
MS3_BusinessDocumentSentEventBeforeTransmission	This script is an extended BusinessDocumentSentEventBeforeTransmission.groovy script, which is optimized for the BOM (Base-Overlay Mapping) approach.
MS3_BusinessDocumentSplitEvent	This script is an extended BusinessDocumentSplitEvent.groovy script, which is optimized for EDI Bulk Messaging B2B Monitoring
MS3_callPDForReceiverCommunication	This script is a extended callPDForReceiverCommunication.groovy script, which is optimized for the BOM (Base-Overlay Mapping) approach.

MS3_callPDwithHeaderParameters	This script is a extended callPDwithHeaderParameters.groovy script, which is optimized for the BOM (Base-Overlay Mapping) approach.
MS3_callPDwithHeaderParametersForFunctionalAcknowledgement	This script is a extended callPDwithHeaderParametersFor FunctionalAcknowledgement.groovy script, which is used to get the camel parameters to send the message back to the sender
MS3_FunctionalAcknowledgementSentEvent	This script is an extended FunctionalAcknowledgementSentEvent.groovy, which sets the Completed state in the B2B Monitoring in case of successful acknowlegement and swaps sender and receiver information.
SelectAssemblyProcessDirectAddress	This script is used for creating the type system related process direct address, so that the corresponding ASSEMBLY_XSLT will be used for the assembly of the whole interchange
setControlNumbers	This script sets the number ranges based on these settings of the number types in the corresponding TPA
setControlNumbersforFunctionalAcknowledgement	This script sets the number ranges based on these settings of the number types in the corresponding TPA for the Functional Acknowledgment
SimulationLogging	This script is used for writing attachments with the interim state of payloads into the MPL Log, if the simulation flag is enabled
Wait for some seconds	This script is responsible for waiting a predefined time before the next step will be called

[B2BIFACTORY-84] Syntax Type and Type System related Routing

Introduction

Change Log

Date	Release	Responsible	Change comment
17.06.2024	1.0.0	MÖ	Added section How to Test

Requirement

It should be possible to route according to the set syntax type (header.SAP_AS2MessageContentType) and type system (header.SAP_EDI_Document_Standard) so that the sender payloads will be dynamically extracted according to the specific requirements of the identified type system. The syntax type defines the syntax representation of the payload, such as XML, JSON, ASN.1, or flat structures using a comma-separated version or values with fixed lengths. The syntax type is usually expressed by the MIME types (IANA Media Types).

Relevant for the B2B are the following IANA Media Types:

- application/EDI-consent [RFC1767] for syntax representation of TRADACOMS
- application/EDIFACT [RFC1767] for ISO 9735 (UN/EDIFACT) syntax representation
- application/EDI-X12 [RFC1767] for ASC X12 syntax rules
- application/json for JSON based payloads.
- application/xml [RFC7303] for payloads based on XML syntax representation.
- text/xml [RFC7303] for payloads based on XML syntax representation.

Solution

At the beginning of the Step 1b flow there should be a router with different routes for how the parameters for type system and syntax type are set by the Step 1a flow:

If the type system is not set, but the syntax type is set to any supported EDI syntax (EDIFACT, ASC X12 or TRADACOMS)

- If the type system is already set by the Step 1a integration flow
- If the syntax type is a XML based type
- If the syntax type is another kind of type such as flat (CSV or fixed length)

How to configure?

You should set at least the value of the exchange header parameter SAP_AS2MessageContentType in the Step 1a communication sender flow. The Step 1a consulting flows from the [B2B Integration Factory] Communication Sender Flows package already consider the setting of the exchange header parameter SAP_AS2MessageContentType using the content modifier step (e.g., the content modifier "Set Adapter Type" in step a AS2 sender flow).

If required, and if it is possible, you can set the type system header.SAP_EDI_Document_Standard by the appropriate custom flow: "Step1a - Communication Sender Flow". If the exchange header parameter SAP_EDI_Document_Standard is set, then the sender payload will be directly handed over to the appropriate "Interchange Extraction Flow" (see chapter).

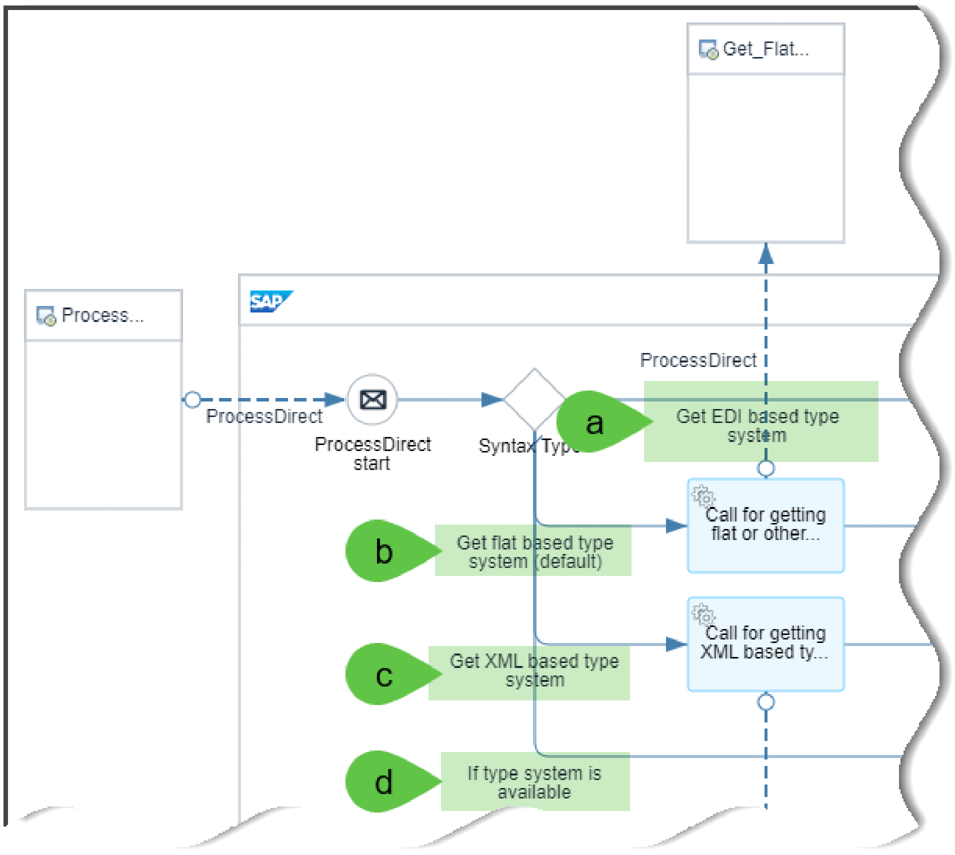
Following SAP_AS2MessageContentType values are considered yet:

application/EDI.*	EDI related syntax and therefore EDI related type systems
application/xml	XML syntax and therefore XML related type system

If either a header.SAP_EDI_Document_Standard or header.SAP_AS2MessageContentType is not set, then this interchange payload will be handed over to a process direct related flow, which is responsible to get a flat based type system (see [B2BIFACTORY-88] Selection of flat based type systems). This flow should have the specific logic for identifying at least the specific header.SAP_AS2MessageContentType and also the header.SAP_EDI_Document_Standard.

How is it implemented?

The distinction of the different syntax types is made in:
Step 1b - Sender Interchange Extraction Flow


Get_Flat...
Process...
ProcessDirect
ProcessDirect
ProcessDirect start
Syntax Type
a
Get EDI based type system
Call for getting flat or other...
b
Get flat based type system (default)
Call for getting XML based ty...
c
Get XML based type system
d
If type system is available

Router conditions:

a. If syntax type (${header.SAP_EDI_Document_Standard} = '') is not set, but the syntax type (AS2MessageContentType) contains one of the MIME Media Type that starts with: "application/EDI" such as:
    a. application/EDI-consent
    b. application/EDIFACT
    c. application/EDI-X12
b. If type system (${header.SAP_EDI_Document_Standard} = '') and the syntax type (${header.AS2MessageContentType}) are not set
c. If type system (${header.SAP_EDI_Document_Standard} = '') is not set, but the syntax type (${header.AS2MessageContentType}) contains one of the MIME Media Type with the term "xml"
d. If a type system is aleady set by the step 1a flow in where the ${header.SAP_EDI_Document_Standard} must not be empty.

[B2BIFACTORY-85] Split of Sender Payloads with Several EDI Interchanges

Introduction

Change Log

Date	Release	Responsible	Change comment
25.06.2024		TO	Added How to Test section & corrected process direct address

Requirement

It could be possible that one sender payload (communication message body) has more than one EDI interchange.

Example of a sender interchange payload in UN/EDIFACT:

```
UNA:+.?
'UNB+UNOC:3+8422416777772:14+8720389000331:14+240120:2340+EW834703975'UNH+EW31020603+INVR
PT:D:96A:UN:EAN004'BGM+78+01-083...
UNA:+.?
'UNB+UNOC:3+8422416777772:14+8720389000331:14+240120:2338+EW761882365'UNH+EW26171410+INVR
PT:D:96A:UN:EAN004'BGM+78+01-0010...
UNA:+.?
'UNB+UNOC:3+8422416777772:14+8720389000331:14+240120:2337+EW212073070'UNH+EW23913354+INVR
PT:D:96A:UN:EAN004'BGM+78+01-0748...
UNA:+.?
'UNB+UNOC:3+8422416777772:14+8720389000331:14+240120:2337+EW884275898'UNH+EW98108858+INVR
PT:D:96A:UN:EAN004'BGM+78+01-0647...
UNA:+.?
'UNB+UNOC:3+8422416777772:14+8720389000331:14+240120:2337+EW502431048'UNH+EW37012605+INVR
PT:D:96A:UN:EAN004'BGM+78+01-0150...
UNA:+.?
'UNB+UNOC:3+8422416777772:14+8720389000331:14+240120:2338+EW39066522'UNH+EW49275495+INVRP
T:D:96A:UN:EAN004'BGM+78+01-0696...
UNA:+.?
'UNB+UNOC:3+8422416777772:14+8720389000331:14+240120:2338+EW921723170'UNH+EW11514822+INVR
PT:D:96A:UN:EAN004'BGM+78+01-0480...
```

Solution

These exchanges must be split in the step 1b integration flow via an additional route which is doing a split of the sender EDI payload. The identification of when a split should be taken should be considered.

How to configure?

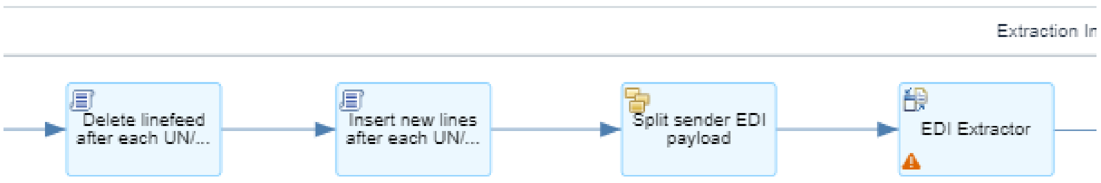
Not applicable

${header.SAP_EDI_Split_Sender_Per_Interchange}

How is it implemented?

The splitting of several interchanges in one sender payload into a camel message per interchange will happen in:

Step 1b - Sender Interchange Extraction Flow


Extraction In
Delete linefeed after each UN/...
Insert new lines after each UN/...
Split sender EDI payload
EDI Extractor

Additional integration flow step:

a. **Groovy Script:** Delete linefeed after each UN/EDIFACT segment
This Groovy Script removes all linefeeds such as \r and \n from the payload.

b. **Groovy Script:** Insert new lines after each UN/EDIFACT interchange
The Groovy Script inserts a line break after each UN/EDIFACT interchange.

c. **General Splitter:** Split sender EDI payload
This general splitter splits the payload into an entry per interchange, if there is a line break after each interchange. This general splitter runs in parallel mode.

[B2BIFACTORY-86] Modification of EDI extracted key values

Introduction

Change log

	Release	Responsible	Change comment
2024.06.25		TO	Added section How to Test & corrected process direct address

Requirement

In some cases, the extracted key values by the EDI extractor might not be sufficient or correct for getting an unambiguous differentiation for obtaining different business transaction activities. For example: A trading partner has different plants located in different countries, but this trading partner is using the same sender id (GLN) for all these plants. This trading partner also needs different TPA / Business Transaction Activities with different MAGs for these plants. To distinguish, it is necessary to extract further values from the sender interchange payload, such as the country code or the plant identifier.

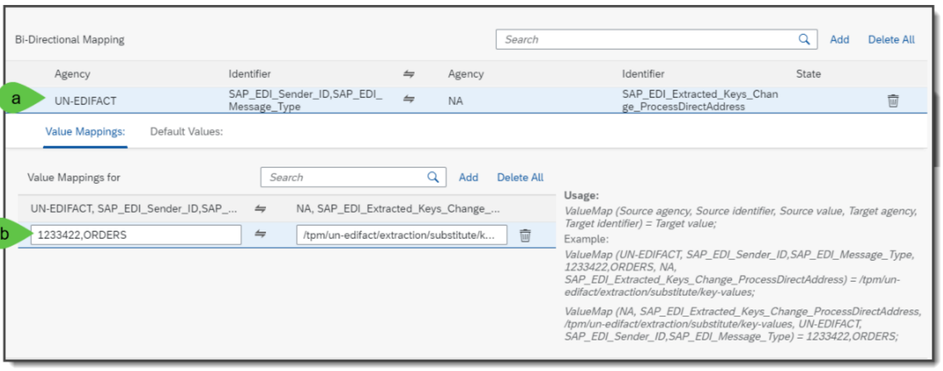
Solution

Provide in Step 1b flow an extension point that allows to change or add extracted key values via a custom flow, which is connected via ProcessDirect.

How to configure?

Navigate Design > Integrations and APIs and go to the package **[B2B Integration Factory] Interchange Extension** Flows. Select "Configure" for Extraction Value Mapping. The configuration for whom (trading partner) and for which EDI based message type should be configured via the "Extraction Value Mapping" in the table "<Type System>" (such as: UN-EDIFACT, ASC_X12, or TRADACOMS). The user should enter the trading partner's sender ID as set in the sender's interchange header and the message type. Furthermore, the user should enter the ProcessDirect related address as target value. This ProcessDirect related address should point to a Custom integration flow for changing the extracted key values. This substitution might happen via a custom-made Groovy Script which will be called in this custom integration flow.

The Extraction Value Mapping should look like following:



Agency	Identifier	⇋	Agency	Identifier	State
UN-EDIFACT	SAP_EDI_Sender_ID,SAP_EDI_Message_Type	⇋	NA	SAP_EDI_Extracted_Keys_Change_ProcessDirectAddress	

UN-EDIFACT, SAP_EDI_Sender_ID,SAP_...	⇋	NA, SAP_EDI_Extracted_Keys_Change_...
1233422,ORDERS	⇋	/tpm/un-edifact/extraction/substitute/k...

Usage:
ValueMap (Source agency, Source identifier, Source value, Target agency, Target identifier) = Target value;
Example:
ValueMap (UN-EDIFACT, SAP_EDI_Sender_ID,SAP_EDI_Message_Type, 1233422,ORDERS, NA, SAP_EDI_Extracted_Keys_Change_ProcessDirectAddress) = /tpm/un-edifact/extraction/substitute/key-values;

ValueMap (NA, SAP_EDI_Extracted_Keys_Change_ProcessDirectAddress, /tpm/un-edifact/extraction/substitute/key-values, UN-EDIFACT, SAP_EDI_Sender_ID,SAP_EDI_Message_Type) = 1233422,ORDERS;

a. Bi-Directional Mapping (Extraction Value Mapping (EVM) Table):

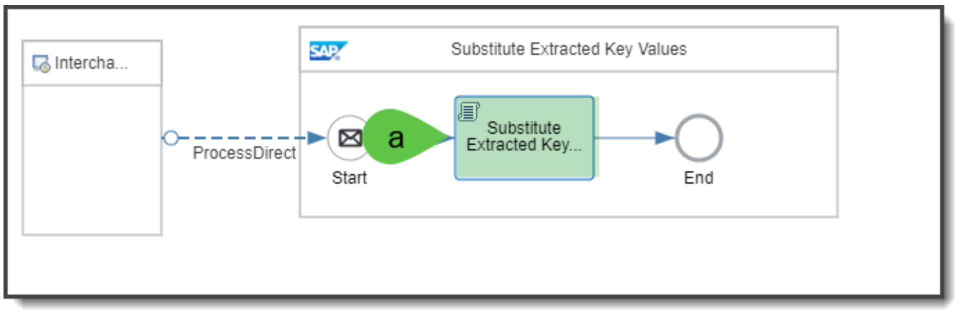
Source Agency: **Extraction Value Mapping (EVM) Name:**	<Type System ID>
Source Identifier: **Source Parameters**	SAP_EDI_Sender_ID,SAP_EDI_Message_Type
Target Agency: **Conditions**	NA
Target Identifier: **Target Parameter**	SAP_EDI_Extracted_Keys_Change_ProcessDirectAddress

b. Value Mappings for:

Left hand side: **Source Values:**	<Sender ID>,<Message Type> e.g.: 1233422,ORDERS
Right hande side: **Target Values:**	/tpm/un-edifact/extraction/substitute/key-values

Custom integration flow for changing the extracted key values

The following flow and groovy script shows, how for e.g., another key value from RFF and CUX segment can be extractions. The extraction part is based on a regular expression.



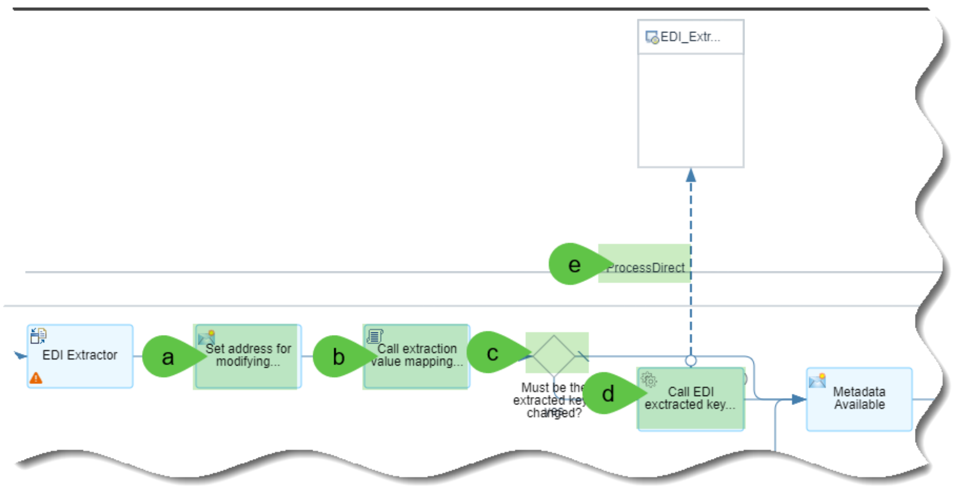
Groovy Script

```
import com.sap.gateway.ip.core.customdev.util.Message;
import org.apache.commons.lang.StringEscapeUtils;
import java.util.HashMap;
import java.util.regex.Pattern;

def Message processData(Message message) {

    //Headers
    def headers     = message.getHeaders();
    def contentType = headers.get("SAP_EDI_Document_Standard")as String;
    def version     = headers.get("SAP_EDI_Message_Version")as String;
    def syntax      = headers.get("SAP_EDI_Syntax_Version_Number")as String;
    def release     = headers.get("SAP_EDI_Message_Release")as String;
    def body        = message.getBody(java.lang.String) as String;

    def sndSenderIdQualifier = headers.get("SAP_EDI_Sender_ID_Qualifier")as String;
    def sndReceiverIdQualifier = headers.get("SAP_EDI_Receiver_ID_Qualifier")as String;

    if (sndSenderIdQualifier == "ZZ") {
       sndSenderIdQualifier = "ZZZ";
       message.setHeader("SAP_EDI_Sender_ID_Qualifier", sndSenderIdQualifier);
    }
    if (sndReceiverIdQualifier == "ZZ") {
       sndReceiverIdQualifier = "ZZZ";
       message.setHeader("SAP_EDI_Receiver_ID_Qualifier", sndReceiverIdQualifier);
       message.setHeader("EDI_Receiver_ID_Qualifier", sndReceiverIdQualifier);
    }

    version = version + "." + release + " S" + "3";
    message.setHeader("SAP_EDI_Message_Version", version);

    // /Remove unwanted new lines
    def body_temp = body.replace("\n", "");
    def compsiteSeparator, dataElementSeparator, decimalSeparator, releaseCharacter,
repet, segmentSeparator;

    try {
        // Obtain syntax delimiters from interchange which should work if UNA segment
present
            (_, compsiteSeparator, dataElementSeparator, decimalSeparator,
releaseCharacter, repet, segmentSeparator) = (body_temp =~ (/UNA([^A-Z])([^A-Z])([^A-
Z])([^A-Z])([^A-Z])?([^A-Z])/))[0]
        } catch (Exception e) {
        // When reading from UNA fails, assume default meta
            (_, compsiteSeparator, dataElementSeparator, decimalSeparator,
releaseCharacter, repet, segmentSeparator) = ["", ":", "+", ".", "?", "", "'"];
     }

     // Construct regex
     segmentSeparator = "\\" + segmentSeparator;                 // '
     releaseCharacter = "\\" + releaseCharacter;                 // ?
     compsiteSeparator = "\\" + compsiteSeparator;               // :
     dataElementSeparator = "\\" + dataElementSeparator;         // +
     // Regex for extracting value from first RFF+ZZZ segment and write it to the 
exchange header parameter: TPM_SND_CountryCodeIdentifier.
     def ref_zzz_value = "";
     try {
            ref_zzz_value = (body_temp =~ 
/RFF${dataElementSeparator}ZZZ${compsiteSeparator}([A-Z]{2})${segmentSeparator}/)[0][1];
            message.setHeader("TPM_SND_CountryCodeIdentifier", ref_zzz_value);
     } catch (Exception e) {
            message.setHeader("TPM_SND_CountryCodeIdentifier", ref_zzz_value );
     }

     // Regex for extracting value from first CUX+2 segment and write it to the exchange 
header parameter: TPM_SND_CurrencyCode.
     def cux_value = "";
     try {
            cux_value = (body_temp =~ 
/CUX${dataElementSeparator}2${compsiteSeparator}([A-Z]{3})${compsiteSeparator}[0-
9]{1,2}${segmentSeparator}/)[0][1];
            message.setHeader("TPM_SND_CurrencyCode", cux_value);
     } catch (Exception e) {
            message.setHeader("TPM_SND_CurrencyCode", cux_value);
     }

    return message;
}
```

How is it implemented?

The updating of key values (camel exchange parameters) happens is proceeded in:

Step 1b - Sender Interchange Extraction Flow Consulting


EDI_Extr...
e
ProcessDirect
EDI Extractor
a
Set address for modifying...
b
Call extraction value mapping...
c
Must be the extracted key changed?
d
Call EDI exctracted key...
Metadata Available

Additional integration flow steps:

a. Content Modifier: Set address for modifying EDI interchange This flow step is used to set the required parameters for the next step (b) which is doing an extraction value mapping call by these parameters. These are:

Parameter	Value	Comment
SAP_B2B_EVM_Name	${SAP_EDI_Document_Standard}	Sets the selected type system as EVM name, which is reflected as (source) agency in the EVM table.
SAP_B2B_EVM_SourceParameters	SAP_EDI_Sender_ID,SAP_EDI_Message_Type	Sets the source values for which an extension flow should be found. Starting point: SAP_EDI_Sender_ID SAP_EDI_Message_Type
SAP_B2B_EVM_TargetParameters	SAP_EDI_Extracted_Keys_Change_ProcessDirectAddress	Provides the address of the ProcessDirect related integration flow, which should be used for substituting the extracted values.

b. Call extraction value mapping for getting ProcessDirect address
ProcessDirect related integration flow which should be used for substituting the extracted key values.

c. Router: Must be the extracted keys changed?
This router decides, if the custom flow for changing extracted keys must be called or not. It will be called if the property SAP_EDI_ExtractedKeyChange_ProcessDirect_Flow is not empty.

d. Request Reply: Call EDI exctracted keys change flow
This request reply should call the external flow via ProcessDirect and hands over the returned exchange headers with the changed key values to the next step.

e. ProcessDirect: ProcessDirect_TPM_EDI_ExtractedKeyChange_Flow
This Process Direct calls the the external custom flow provided by the exchange property:
SAP_EDI_ExtractedKeyChange_ProcessDirect_Flow for changing the extracted key values. See details about the custom flow in chapter:

[B2BIFACTORY-87] Selection of XML based Type Systems

Introduction

Change log

Date	Release	Responsible	Change comment
2024.06.20		TO	Added section How to Test & corrected process direct address

Requirement

If the payload is based on XML and a type system is not set by the step 1a flow (see route option c in chapter: 1.) Syntax Type and Type System related Routing) then this XML based type system must be explicitly identified.

Solution

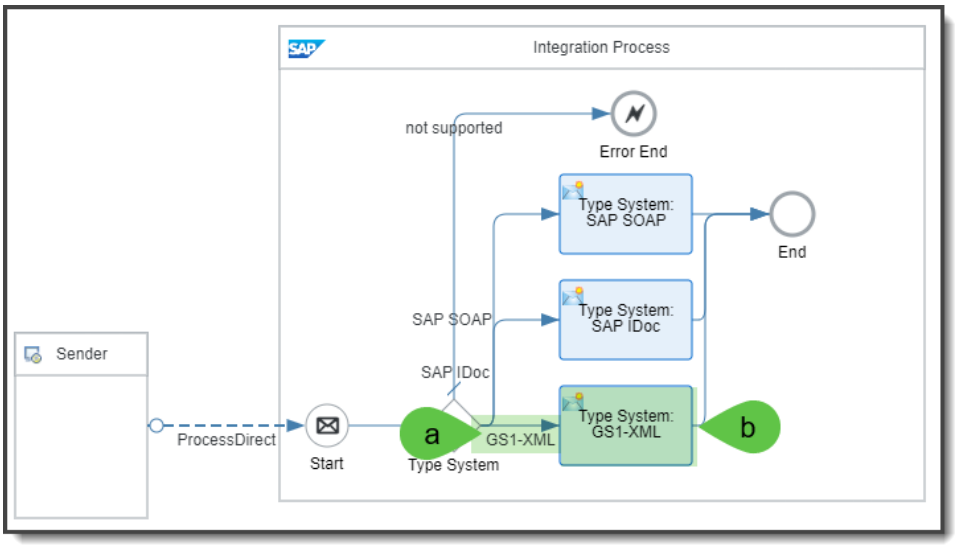
The Step 1b integration flow should refer to an external flow, if the type system and the syntax type are not set (this is the default route) by the Step 1a integration flow. In this process direct connected flow, it should be possible that the user can define his own routes and conditions. Especially this flow should have some predefined routes and conditions for the already supported type systems.

How to configure?

The user should enter an additional route + content modifier into the provided integration flow _ Get XML based Type System. This route should have a XPath expression for an unambiguous matching of the additionally required type system. Furthermore, this route should point to a Content Modifier in where the specific type system will be set via a Camel exchange header attribute.

_ Get XML based Type System

This integration flow is responsible for getting the type system id as ${header.SAP_EDI_Document_Standard} using the router. This integration flow is in the integration package [V3] Interchange Extraction Flows and can be modified by the user. This integration flow should have predefined routing steps for the supported XML based standard type systems such as SAP SOAP, SAP IDOC, and GS1-XML. Further routing steps can be added by the user.



Each routing step should provide:

a. Route: <Name of Type System>
Should have an unambiguous Xpath based condition expression for identifying the type system such as:

Order	Router Name	Condition Expression
1	GS1-XML	boolean(/node()/*:StandardBusinessDocumentHeader[1 = last()])
2	SAP IDoc	boolean(/node()/*:IDOC[1][1=last()])
4	SAP SOAP	boolean(/node()/MessageHeader[1 = last()])
3	not supported	

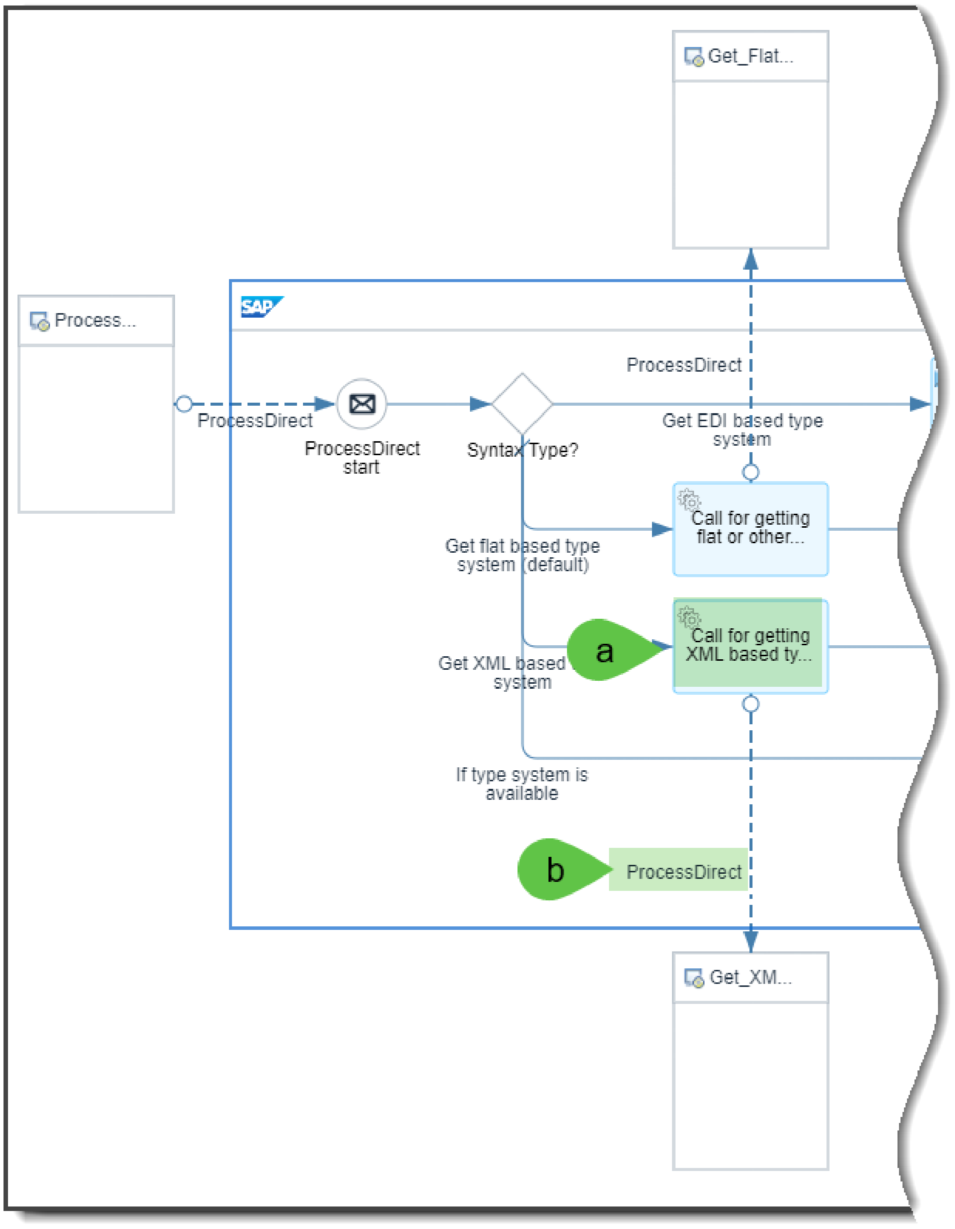
b. Content Modifier: Set values for identifying XML based type system.
This flow steps sets the exchange header parameter SAP_EDI_Document_Standard accordingly.
Following exchange header parameter should be predefined:

Router Name	SAP_EDI_Document_Standard
GS1-XML	GS1-XML
SAP IDoc	SAP_IDoc
SAP SOAP	SAP_S4HANA_OnPremise_SOA
not supported	Leads to an error because XML based type system is not identified

How is it implemented?

The setting of a XML based type system (header.SAP_EDI_Document_Standard) happens in:

Step 1b - Sender Interchange Extraction Flow


Get_Flat...
Process...
ProcessDirect
ProcessDirect
ProcessDirect start
Syntax Type?
Get EDI based type system
Call for getting flat or other...
Get flat based type system (default)
a
Call for getting XML based ty...
Get XML based system
If type system is available
b
ProcessDirect
Get_XM...

Additional integration flow steps:

a. Request-Reply: Call for getting XML based type system
It calls an external integration flow for getting the type system id as ${header.SAP_EDI_Document_Standard} for the XML based type system

b. ProcessDirect: ProcessDirect_Get_XML_TypeSystem_Flow_Call
The address for calling the external integration flow with the name "_ Get XML based Type System" is: /tpm/extract-type-system-flow/xml.

[B2BIFACTORY-89] Extraction of key fields of XML or Flat based Sender Interchange Payloads

Introduction

Change log

Date	Release	Responsible	Change comment
2024.06.21		TO	Added section How to Test

Requirement

It is necessary to extract the necessary key values from the sender interchange payloads which are required for the finding of the appropriate PD entry in the partner directory for further processing in Step 2. The following key fields should at least be extracted: sender ID, sender ID qualifier (optional), receiver ID, receiver ID qualifier (optional), message type, and message version. These must be written to the appropriate Camel exchange header attributes so that they can be used for the calculation of the PID (Partner Directory Identifier) in Step 2 flow. Depending on the structures of the XML or Flat based standard these key values are placed on different locations in form of elements or attributes. Furthermore, it could be possible that the standard key fields are not sufficient of an unambiguous obtaining of the required PD entry. Therefore, it is required to define extraction rules per required type system which can be individually changed if this is required by the user (customer).

Solution

The solution is an external integration flow per required type system which covers the individual extraction rules. The appropriate external integration flow should be called via ProcessDirect depending on the type of system which was identified by the flow steps as described in chapter 4.) Selection of XML based Type Systems and 5.) Selection of flat based Type Systems. SAP should provide a predefined set of this so-called extraction integration flows for all supported type systems. It should be possible that these predefined integration flows can be individually modified by the customer. It should also provide a template and a best practice guide so that the customer can define his own extraction integration flows.

How to configure?

- There should be an external integration flow (so called integration extraction flow) per required type system in the integration package: [B2B Integration Factory] Interchange Extraction Flows. SAP should provide some default extraction flows for all supported type systems.
- There are two different types of extraction flows:

- Interchange extraction flows for XML based type system in which the extraction of the key values will be via the XPath expressions in the Content Modifier
- Interchange extraction flows of flat based type systems which should have a conversion step (CSV to XML or Fixed to XML) in front of the extraction of the key values. The extraction will be then similar to the XML based type system.

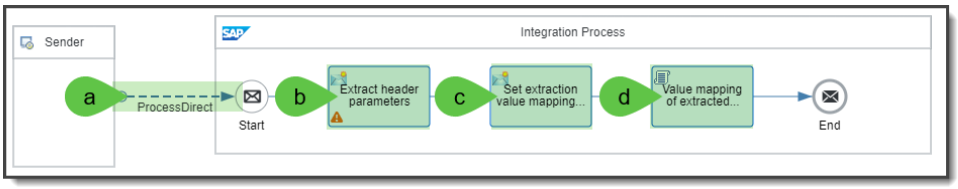
Interchange extraction flows for XML based type system

In case of a new XML based type system, you must create an additional integration flow in the package [B2B Integration Factory] Interchange Extraction Flows, which extracts the required key fields from the (message) header part of the XML based payload and writes it into the required Camel exchange header parameters. If you need a substitution of further change of these extracted values, you can also add the two flow of the “Extraction Value Mapping” as described in chapter “EVM - Extraction Value Mapping”.

Following Camel exchange header parameters should be generated with the corresponding extracted values:

Header Parameter	Mandatory/Optional	Meaning
SAP_EDI_Message_Type	Mandatory	The type of the message that is used for the XML based message payload
SAP_EDI_Message_Version	Mandatory	The version of the message type.
SAP_EDI_Sender_ID	Mandatory	The identifier of the sender of this message payload
SAP_EDI_Sender_ID_Qualifier	Optional	The qualifier of the identifier scheme on which the sender identifier is based on, such as GS1.
SAP_EDI_Receiver_ID	Mandatory	The identifier of the sender of this message payload
SAP_EDI_Receiver_ID_Qualifier	Optional	The qualifier of the identifier scheme on which the receiver identifier is based on, such as GS1.

The following example shows you how an XML based interchange extraction flow should look like:


Sender
Integration Process
a
ProcessDirect
Start
b
Extract header parameters
c
Set extraction value mapping...
d
Value mapping of extracted...
End

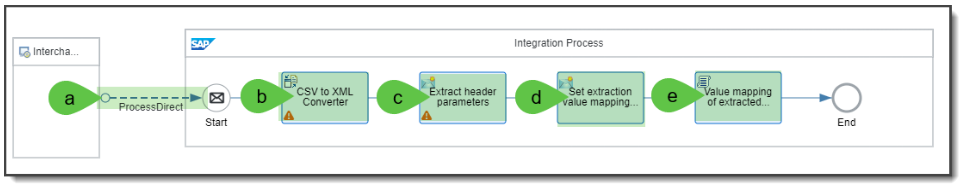
(a) ProcessDirect (Mandatory): It is quite important to set the ProcessDirect address, which will be called from Step 1b flow, when it comes to the extraction step. This ProcessDirect address is: /tpm/extraction-flow/{{SAP_EDI_Document_Standard}}
This means, the already identified "SAP_EDI_Document_Standard" from the flow steps in "Step 1b" before is used for calling this extraction integration flow.
(b) Content Modifier "Extract header parameters" (Mandatory): You can do this extraction via the "Content Modifier" in where you allocate the values via XPath expressions. Further parameters can be optionally extracted, in case if these are necessary for further processing or decision making.
(c) Content Modifier "Set extraction value mapping key fields" (Optional): If it is necessary, you can also change or substitute the extracted values by the "Extraction Value Mapping". The chapter EVM - Extraction Value Mapping explains you, how you can set up and configure it.
(d) Groovy Script "Value mapping of extracted header parameters" (Optional): This flow is required when an Extraction Value Mapping must be called.

Interchange extraction flows of flat based type system

Similar like the XML based type systems, you need a separate extraction flow per supported flat based type system. If this flat based type system is based on CSV syntax, it is recommended to convert the message payload into the internally used XML syntax by using the CSV to XML flow step (b). Once this message is converted into XML, you can use at least a content modifier for extracting or setting the required values similar like described in chapter Interchange extraction flows for XML based type system. Following Camel Exchange Header parameters with their key values are also required:

Header Parameter	Mandatory/Optional	Meaning
SAP_EDI_Message_Type	Mandatory	The type of the message that is used for the XML based message payload
SAP_EDI_Message_Version	Mandatory	The version of the message type.
SAP_EDI_Sender_ID	Mandatory	The identifier of the sender of this message payload
SAP_EDI_Sender_ID_Qualifier	Optional	The qualifier of the identifier scheme on which the sender identifier is based on, such as GS1.
SAP_EDI_Receiver_ID	Mandatory	The identifier of the sender of this message payload
SAP_EDI_Receiver_ID_Qualifier	Optional	The qualifier of the identifier scheme on which the receive identifier is based on, such as GS1.

The following example shows you how a CSV based interchange extraction flow should look like:


Intercha...
Integration Process
a
ProcessDirect
Start
b
CSV to XML Converter
c
Extract header parameters
d
Set extraction value mapping...
e
Value mapping of extracted...
End

(a) ProcessDirect (Mandatory): Quite important, is to set the ProcessDirect address, which will be called from Step 1b flow, when it comes to the extraction step. This ProcessDirect address is: /tpm/extraction-flow/{{SAP_EDI_Document_Standard}}
This means, the already identified "SAP_EDI_Document_Standard" from the flow steps in "Step 1b" before is used for calling this extraction integration flow.

(b) CSV To XML Converter (Mandatory): This flow step is required for converting the CSV data into the XML syntax. For this purpose, you need a XML schema that represents the output in XML syntax and furthermore you have to set the XPath location of the target element which should be generated per line in the CSV input payload.

(c) Content Modifier "Extract header parameters" (Mandatory): You can do this extraction via the "Content Modifier" in where you allocate the values via XPath expressions. Further parameters can be optionally extracted, in case if these are necessary for further processing or decision making.

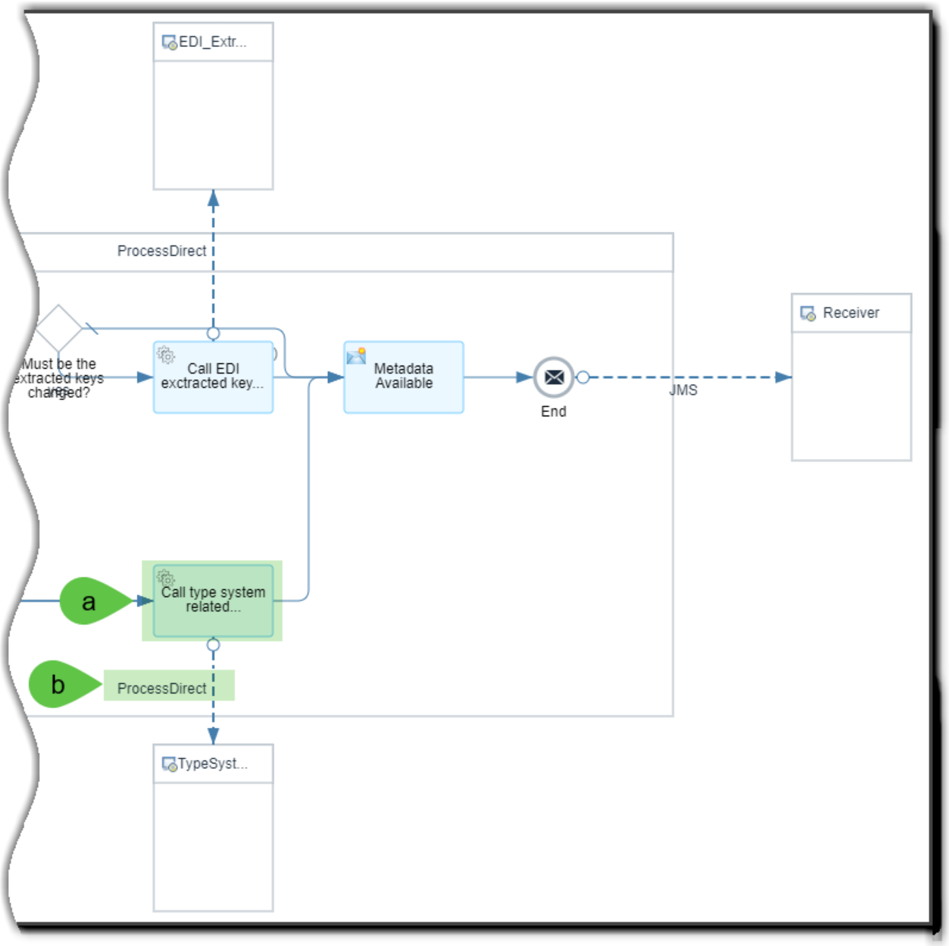
(d) Content Modifier "Set extraction value mapping key fields" (Optional): If it is necessary, you can also change or substitute the extracted values by the "Extraction Value Mapping". The chapter EVM - Extraction Value Mapping explains you, how you can set up and configure it.

(e) Groovy Script "Value mapping of extracted header parameters" (Optional): This flow is required when an Extraction Value Mapping must be called.

How is it implemented?

The call of the integration flows for extracting the key fields is implemented in:

Step 1b - Sender Interchange Extraction Flow


EDI_Extr...
ProcessDirect
Receiver
Must be the extracted keys changed?
Call EDI exctracted key...
Metadata Available
End
JMS
a
Call type system related...
b
ProcessDirect
TypeSyst...

Additional integration flow steps:

a. Request-Reply: Call type system related extraction flow.
It calls an external integration flow for doing the type system related extraction of the key values.

b. ProcessDirect: ProcessDirect_Get_Flat_TypeSystem_Flow_Call
The address for calling the external integration, which is /tpm/extraction-flow/${header.SAP_EDI_Document_Standard} whereby the SAP_EDI_Document_Standard must be set by the steps before.

[B2BIFACTORY-90] Further processing of received functional acknowledgement interchanges

Introduction

Change log

Date	Release	Responsible	Change comment
2024.06.20		VD	Changed Screenshot, added additional flow step section.

Requirement

The results from trading partner's received functional acknowledgement interchanges should not be just used for displaying the final status in the B2B Monitoring. Especially these results should be mapped to a target message such as SAP IDOC SYSTAT or ALEAUD for submitting it to the target business application.

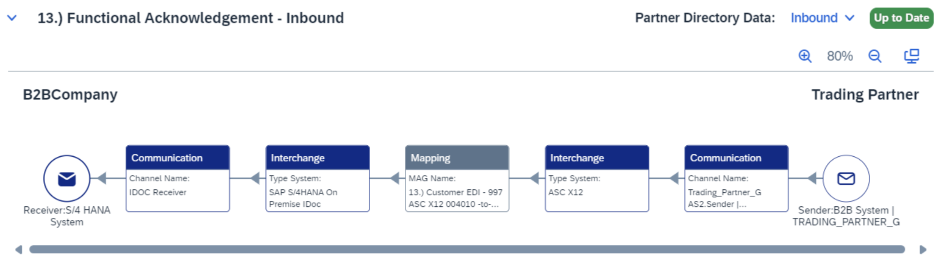
Solution

In the TPA --> Outbound Business Transaction Activity in where the functional acknowledgement is requested, a parameter should be set that signals that a further processing is required. This parameter will be additionally stored in the entry of data store, which is used for building the correlation between submitted target interchange payloads and received functional acknowledgements. The Step 2 flow should now check in the correlation phase if this parameter is set. If this is the case, the whole interchange should be handed over to the main processing flow such as for doing mapping and handing to the target system.

How to configure?

Set in the section “Activity Parameters” of specific outbound Business Transaction Activity in where a functional acknowledgment is requested the custom parameter:
SAP_EDI_SND_Received_Funct_Ack_Processing = true.

Furthermore, the user must create in the same TPA a further business transaction which has a business transaction activity for transforming the functional acknowledgement into the required target message type such as SAP IDOC SYSTAT (see following figure).



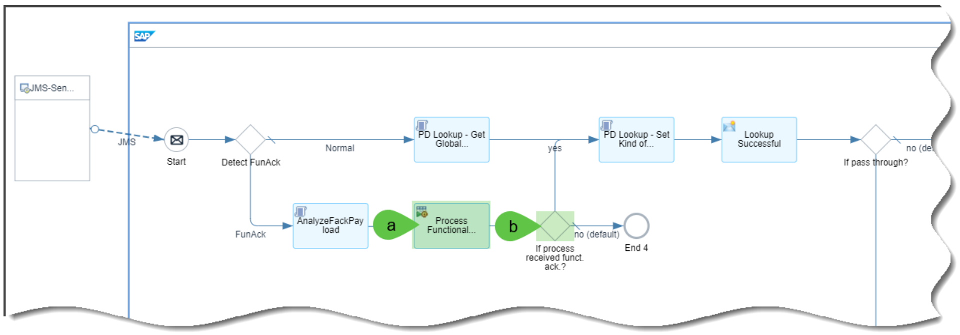
13.) Functional Acknowledgement - Inbound
Partner Directory Data: Inbound
Up to Date
80%
B2BCompany
Trading Partner
Receiver:S/4 HANA System
Communication
Channel Name: IDOC Receiver
Interchange
Type System: SAP S/4HANA On Premise IDoc
Mapping
MAG Name: 13.) Customer EDI - 997 ASC X12 004010 -to-...
Interchange
Type System: ASC X12
Communication
Channel Name: Trading_Partner_G AS2.Sender |...
Sender:B2B System | TRADING_PARTNER_G

How is it implemented?

The extensions for the further processing of a functional acknowledgement is implemented:

Step 2 - Interchange Processing Flow

→ Main Interchange Processing Flow



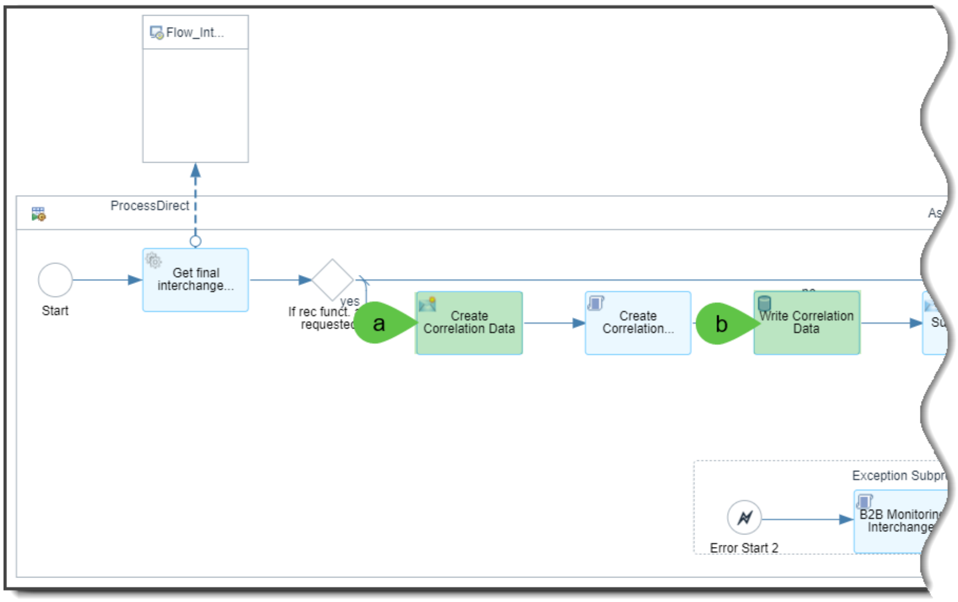
Additional integration flow steps:

(a) Process Call: "Process Functional Acknowledge"

(b) This local integration process is extended so that received functional acknowledgement interchanges can be further processed, for e.g. for the mapping to SAP Idoc SYSTAT messages. It will especially return the exchange property SAP_EDI_SND_Received_Funct_Ack_Processing, if a further processing is requested. See details in → Local Integration Process: Functional Acknowledge Interchange

(c) Router: If process received funct. ack.?
Hands over the received functional ack interchange (for e.g. UN/EDIFACT CONTRL or ASC X12 997) to the PD Lookup step, if the exchange property SAP_EDI_SND_Received_Funct_Ack_Processing is "true"

→ Local Integration Process: Assemble Receiver Interchange

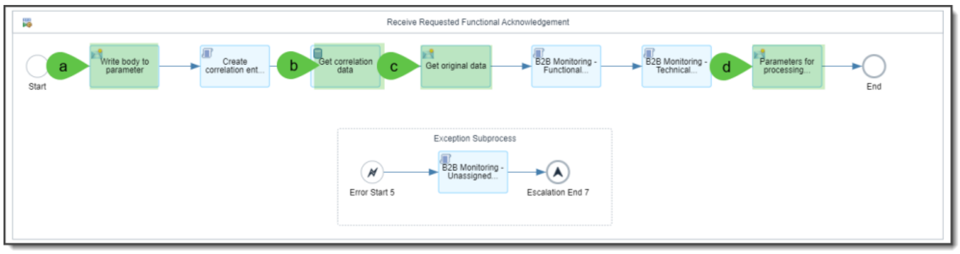


Changes in following flow steps:

a. Content Modifier: Create Correlation Data
The parameter "SAP_EDI_SND_Received_Funct_Ack_Processing" is inserted in the "<BusinessTransactionActivity" XML body as defined in "Message Body":

```
<BusinessTransactionActivity>
    <AgreementID>${property.Agreement_ID}</AgreementID>
    <BusinessTransactionID>${property.Transaction_ID}</BusinessTransactionID>
    <BusinessDocumentID>${property.Document_ID}</BusinessDocumentID>

<InterchangeControlNumber>${property.SAP_EDI_REC_Interchange_Control_Number}</Interchange
ControlNumber>

<SenderTradingPartnerID>${property.SAP_TPA_SND_Trading_Partner_ID}</SenderTradingPartnerI
D>

<ReceiverTradingPartnerID>${property.SAP_TPA_REC_Trading_Partner_ID}</ReceiverTradingPart
nerID>
    <InterchangeDateTime>${header.SAP_EDI_Interchange_DateTime}</InterchangeDateTime>

<FunctionalAckRequest>${property.SAP_EDI_REC_Acknowledgement_Request}</FunctionalAckReque
st>

<FunctionalAckProcessingIndicator>${property.SAP_EDI_SND_Received_Funct_Ack_Processing}</
FunctionalAckProcessingIndicator>
</BusinessTransactionActivity>
```

→ Local Integration Process: Functional Acknowledge Interchange


Receive Requested Functional Acknowledgement
Start
a
Write body to parameter
Create correlation ent...
b
Get correlation data
c
Get original data
B2B Monitoring - Functional...
B2B Monitoring - Technical...
d
Parameters for processing...
End
Exception Subprocess
Error Start 5
B2B Monitoring - Unassigned...
Escalation End 7

Additional integration flow steps:

(a) Content Modifier: Write body to parameter
The received functional acknowledgement interchange payload will be written into an exchange property so that this can be taken back into the body, once the correlated data ??

(b) Content Modifier: Get Correlation Data
Read the XML Payload stored from the data store "SAP_BTA_CorrelationDataStore"

(c) Content Modifier: Get Original Data
Writes the payload back into the Camel exchange body

(d) Content Modifier: Parameters for processing funct. ack.
This content modifier obtains the value of the element FunctionalAckProcessingIndicator in the obtained XML body "BusinessTransactionActivity" from the data store and writes it into the Camel exchange property "SAP_EDI_SND_Received_Funct_Ack_Processing"

[B2BIFACTORY-92] Pass Through

Introduction

Change log

Date	Release	Responsible	Change comment
2024.06.24		TO	Added section How to Test

Requirement

It is the requirement that specific sender interchange payload should be just submitted to a target system without further processing. This exchange should be monitored in the B2B monitoring using the metadata provided by the TPM.

Solution

There should be a router before the main processing steps which directly passes the data to the sub integration flow "Assemble Receiver Interchange".

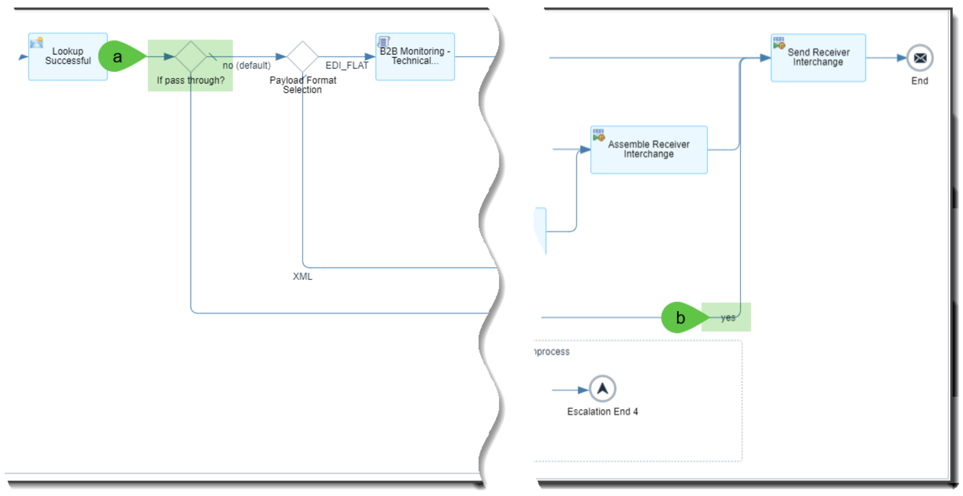
How to configure?

The user should set the custom parameter: SAP_EDI_Processing_Type = "PASS_THROUGH" in the business transaction activity in where a pass through of the sender interchange payload is required.

How is it implemented?

Step 2 - Interchange Processing Flow Consulting

→ Main Interchange Processing Flow



Additional integration flow steps:

(a) Router: If pass through?
The router routes the sender interchange payload via the pass-through route, if the ${header.SAP_EDI_Processing_Type} = 'PASS_THROUGH'

(b) Assemble Receiver Interchange
The pass-through route

[B2BIFACTORY-93] Sender EDI Interchange Custom Processing

Introduction

Change log

Date	Release	Responsible	Change comment
2024.06.25		VD	Added Additional Flow steps section

Requirement

Some EDI interchange payloads can't be sufficiently processed by the EDI Splitters. This content leads to an EDI Splitter error. Typical examples are:

- UN/EDIFACT:
    - Unsupported syntax representation
    - Sender/receiver interchange identifier qualifier: ZZZ instead of ZZ
- ASC X12:
    - Subset in the version

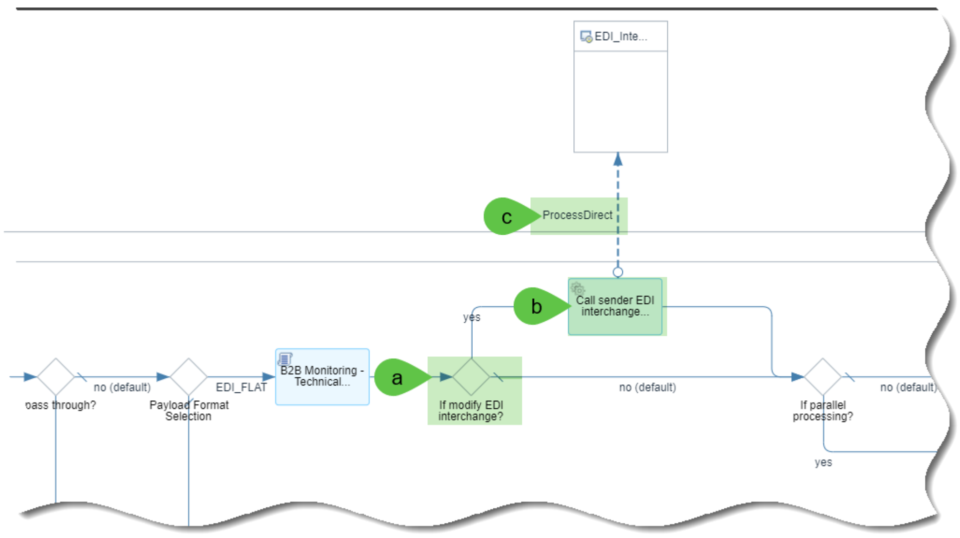
Solution

These EDI interchange payloads will be manipulated (changed) via a ProcessDirect based custom processing step before the EDI Splitter step. Using this ProcessDirect integration flow, it is possible to adjust the payload so that they can be processed by the EDI Splitter.

How is it implemented?

Step 2 - Interchange Processing Flow

→ Main Interchange Processing Flow



Additional integration flow steps:

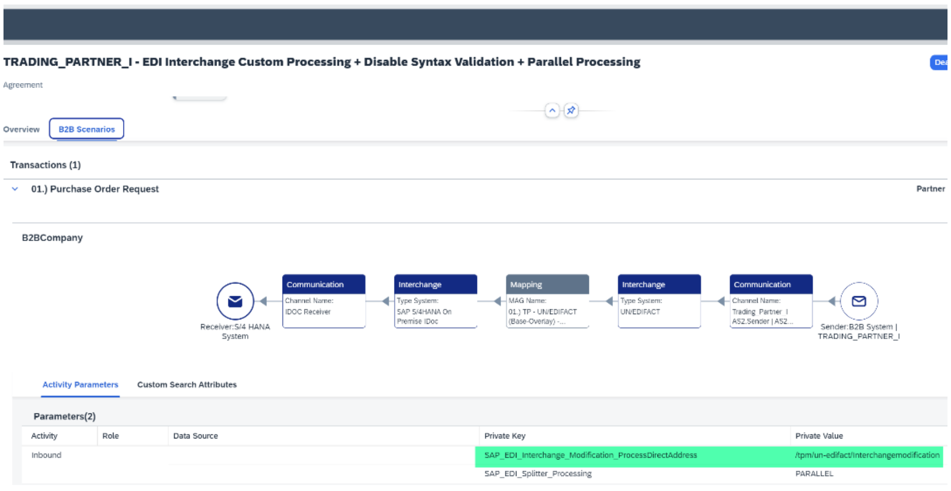
a. Router: If modify EDI interchange?
   This router will route the EDI Interchange payload to the flow step Request Reply: Call sender EDI interchange payload modification flow if the Camel exchange property SAP_EDI_Interchange_Modification_ProcessDirectAddress is not empty.

b. Request Reply: Call sender EDI interchange payload modification
   It calls an external integration flow through process direct and returns the response to the router "If Parallel Processing?"

c. ProcessDirect: SAP_EDI_Interchange_Modification_ProcessDirectAddress
   The address for calling the external integration flow with the name "UN-EDIFACT Interchange Modification" is **/tpm/un-edifact/Interchangemodification**. This iflow is in the package "[B2B Integration Factory] Extended Interchange Processing Flows". It calls the ProcessDirect related integration flow via the address as defined in the Camel exchange Property ${property.SAP_EDI_Interchange_Modification_ProcessDirectAddress}

How to configure?

You should set in the section “Activity Parameters” of the relevant Business Transaction Activity the following customer parameter:
SAP_EDI_Interchange_Modification_ProcessDirectAddress ::= ProcessDirect address of the flow, which is responsible for the preceding handling of the interchange. Process Direct address of the flow is **/tpm/un-edifact/Interchangemodification**


TRADING_PARTNER_I - EDI Interchange Custom Processing + Disable Syntax Validation + Parallel Processing

Agreement

Transactions (1)

01.) Purchase Order Request

Parameters(2)

Activity	Role	Data Source	Private Key	Private Value
Inbound			SAP_EDI_Interchange_Modification_ProcessDirectAddress	/tpm/un-edifact/Interchangemodification
			SAP_EDI_Splitter_Processing	PARALLEL

[B2BIFACTORY-94] Disable Sender EDI Interchange Syntax Validation

Introduction

Change log

Date	Release	Responsible	Change comment
2024.06.25		VD	Updated how to configure, Additional Flow steps section

Requirement

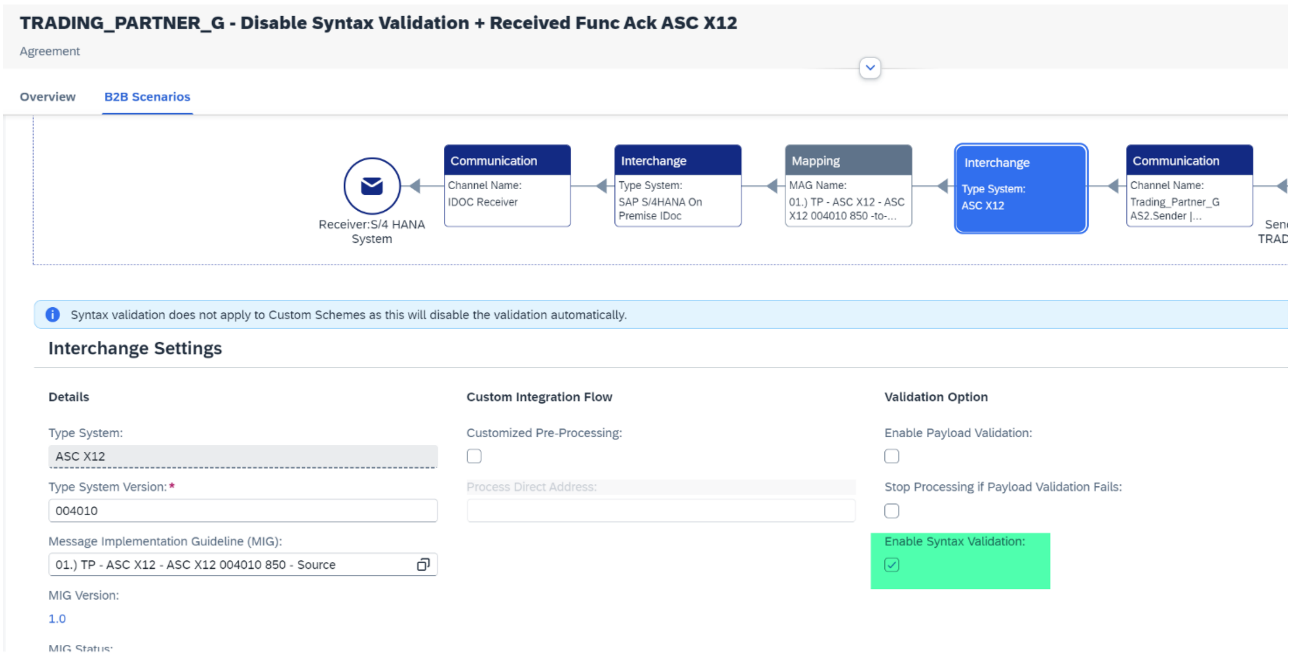
Some sender EDI interchange payloads have an incorrect setting of the counters in the trailer interchange or message segments. These normally lead to an error and processing stop by the EDI Splitter step. Usually, if these errors did not stop the complete process of the EDI interchange processing, this payload could be processed further. It is this step that you want to force in some cases.

Solution

Provide in the Step 2 integration flow an additional route to an EDI Splitter in where the validation is disabled.

How to configure?

By default, the parameter **Enable Syntax Validation** is always set to **true** when the TPA is configured. This value is pushed to Partner directory and the Iflow step 2 fetches this reads this parameter from the partner directory. So if you want the payload to be validated, check the **Enable Syntax validation** checkbox in TPA. If you do not want the payload to be validated, just uncheck the **Enable Syntax validation** checkbox by editing the TPA Sender Interchange and unchecking the parameter.



TRADING_PARTNER_G - Disable Syntax Validation + Received Func Ack ASC X12

Agreement

Overview | B2B Scenarios

Syntax validation does not apply to Custom Schemes as this will disable the validation automatically.

Interchange Settings

Details

Type System: ASC X12

Type System Version:* 004010

Message Implementation Guideline (MIG): 01.) TP - ASC X12 - ASC X12 004010 850 - Source

MIG Version: 1.0

MIG Status:

Custom Integration Flow

Customized Pre-Processing: ☐

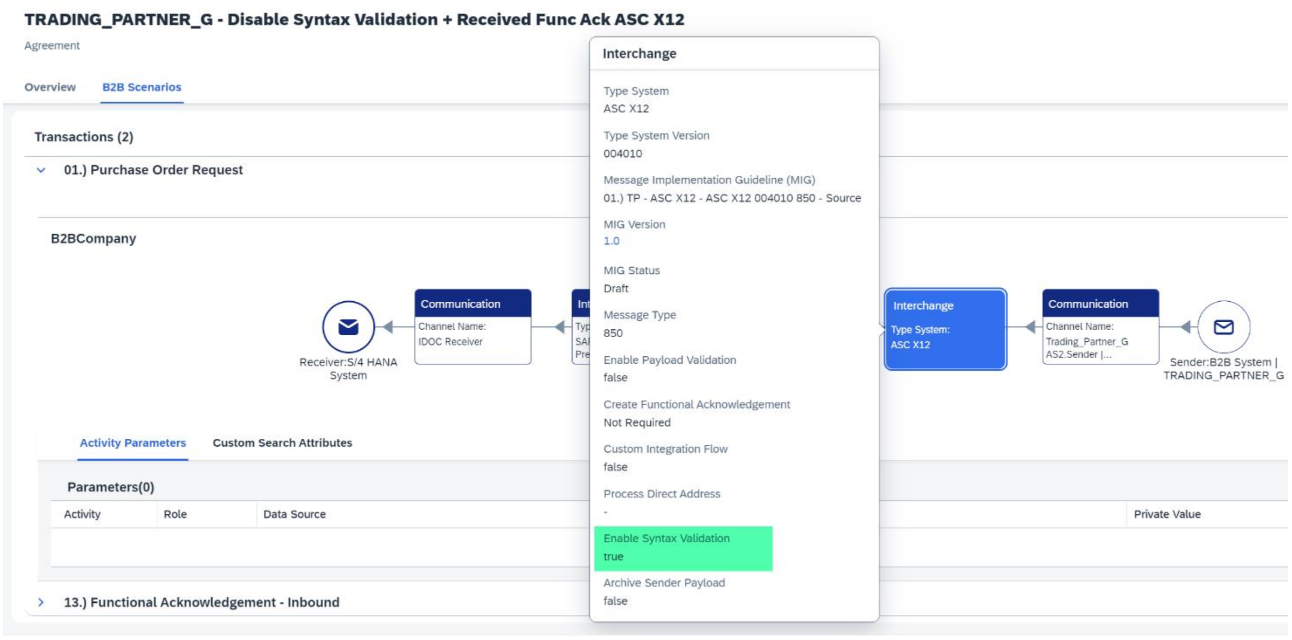
Process Direct Address:

Validation Option

Enable Payload Validation: ☐

Stop Processing if Payload Validation Fails: ☐

Enable Syntax Validation: ✓



TRADING_PARTNER_G - Disable Syntax Validation + Received Func Ack ASC X12

Agreement

Overview | B2B Scenarios

Transactions (2)

01.) Purchase Order Request

B2BCompany

Activity Parameters | Custom Search Attributes

Parameters(0)

Activity	Role	Data Source	Private Value

13.) Functional Acknowledgement - Inbound

Interchange

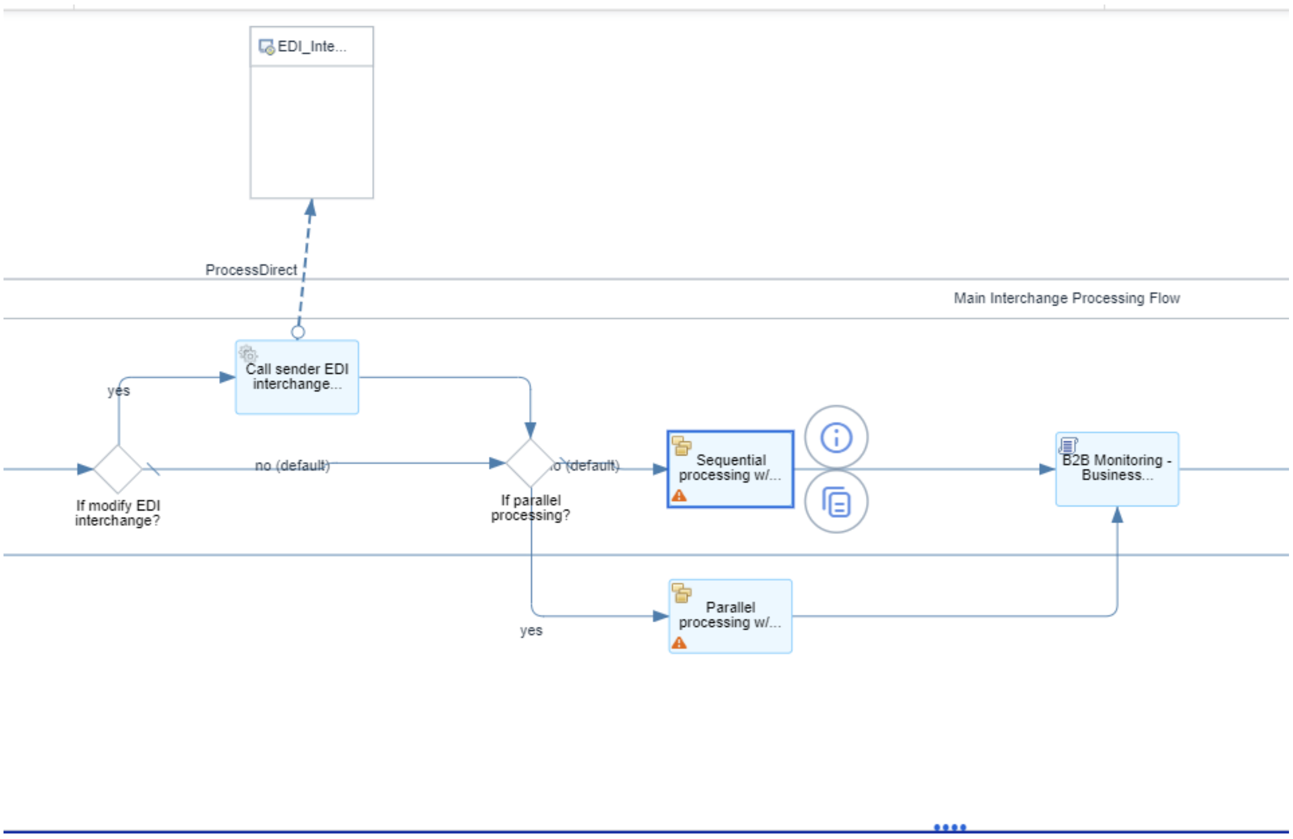
Type System	ASC X12
Type System Version	004010
Message Implementation Guideline (MIG)	01.) TP - ASC X12 - ASC X12 004010 850 - Source
MIG Version	1.0
MIG Status	Draft
Message Type	850
Enable Payload Validation	false
Create Functional Acknowledgement	Not Required
Custom Integration Flow	false
Process Direct Address	-
Enable Syntax Validation	true
Archive Sender Payload	false

How is it implemented?

Step 2 - Interchange Processing Flow

→ Main Interchange Processing Flow

It is an integral part of the newest release of the flow step “EDI Splitter”


EDI_Inte...
ProcessDirect
Main Interchange Processing Flow
Call sender EDI interchange...
yes
no (default)
If modify EDI interchange?
no (default)
If parallel processing?
Sequential processing w/...
B2B Monitoring - Business...
yes
Parallel processing w/...

Additional Integration Flow Steps:

Router: **If Parallel processing?** Here it would be checked whether the payload should be processed in a sequential or parallel way with/without validation.

a. Router condition : If Parallel Processing ?-> No (default)
The Routing condition ${property.SAP_EDI_Splitter_Processing} = 'PARALLEL' if false, then the payload is routed to the EDI Splitter flow step 'Sequential processing '.

b. Router condition : If Parallel Processing ?-> Yes
The Routing condition ${property.SAP_EDI_Splitter_Processing} = 'PARALLEL' if true, then the payload is routed to the EDI Splitter flow step 'Parallel processing '.

c. EDI Splitter

- Parallel Processing :
- Sequential Processing

[B2BIFACTORY-96] B2B Monitoring - Business Document Split Event

Introduction

Change log

Date	Release	Responsible	Change comment
2024.06.24		TO	Added section How to Test

Requirement

If a sender EDI interchange payload has been split, this is not displayed correctly in B2B monitoring. But users want to see in the B2B monitoring an entry for this sender EDI interchange payload and additional entries for each split EDI message, which will get further processed.

Solution

Provide in Step 2 integration flow an additional flow step "Business Document Split Event" after the EDI Splitter step.

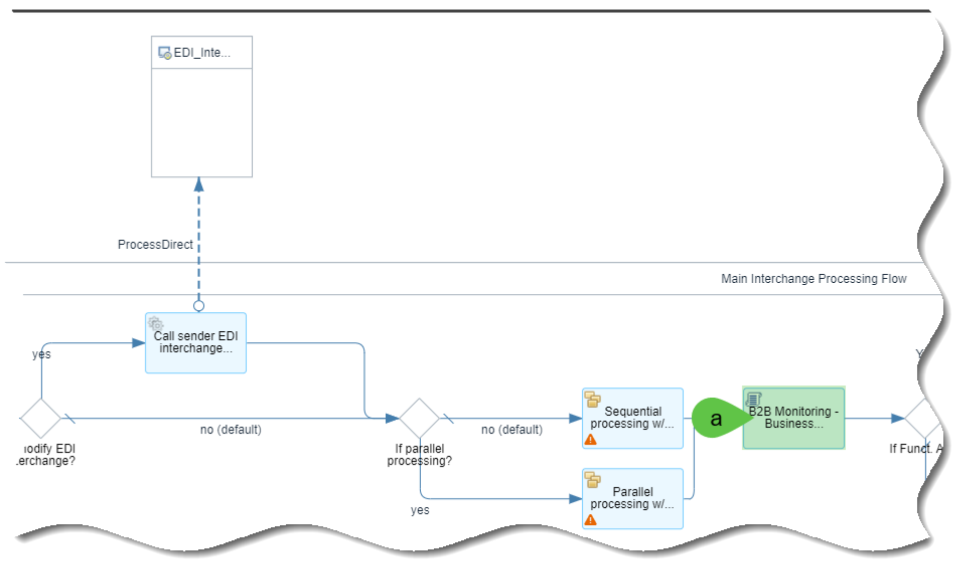
How to configure?

Not applicable

How is it implemented?

Step 2 - Interchange Processing Flow Consulting

→ Main Interchange Processing Flow


EDI_Inte...
ProcessDirect
Main Interchange Processing Flow
Call sender EDI interchange...
yes
no (default)
odify EDI erchange?
If parallel processing?
no (default)
Sequential processing w/...
Parallel processing w/...
yes
a
B2B Monitoring - Business...
If Funct. A

Additional integration flow steps:

a. Groovy Script: B2B Monitoring - Business Document Split Event

[B2BIFACTORY-97] Sender XML Interchange Split Processing

Introduction

Change log

Date	Release	Responsible	Change comment
2024.06.24		TO	Added section How to Test

Requirement

Some XML based sender interchange payloads contain some messages that must be split before they can be processed further according to the specifications of the MAG (Mapping Guidelines).

Solution

There should be in the Step 2 flow a split step before the main processing steps for XML based sender interchange payloads into single messages. Where should be split is defined by a XPath expression.

How to configure?

You should set in the section "Activity Parameters" of the relevant Business Transaction Activity the following customer parameters:

SAP_XML_Interchange_Split_Indicator ::= true

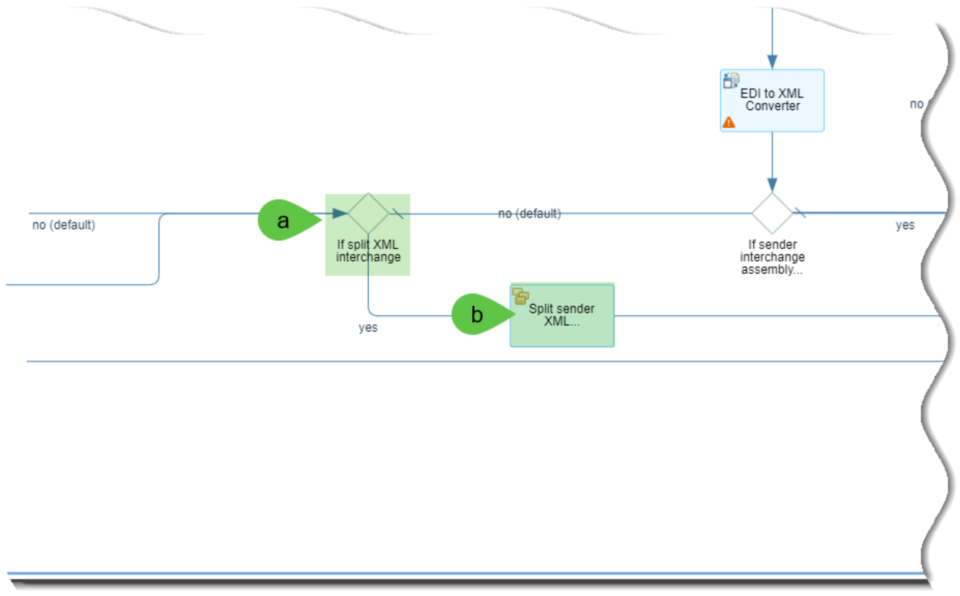
SAP_XML_Interchange_Split_Path ::= <XPath>

if split of a XML based sender interchange payload is required

How is it implemented?

Step 2 - Interchange Processing Flow Consulting

→ Main Interchange Processing Flow



Additional integration flow steps:

a. Router: If split XML interchange
b. General Splitter: Split sender interchange payload

[B2BIFACTORY-98] Set Receiver Interchange Control Numbers

Introduction

Change log

Date	Release	Responsible	Change comment
2024.06.24		TO	Added section How to Test

Requirement

There is often a requirement that the control numbers which will be set for the receiver interchange payloads can be also used in Mapping Guidelines for the mapping of these control numbers to other leaf nodes apart of the corresponding trailer leaf nodes.

Solution

In the step 2 integration flow there should be the flow step for setting control numbers before the main processing steps which are responsible for the mapping.

How to configure?

You should set in the section “Activity Parameters” of the relevant Business Transaction Activity the following customer parameters:

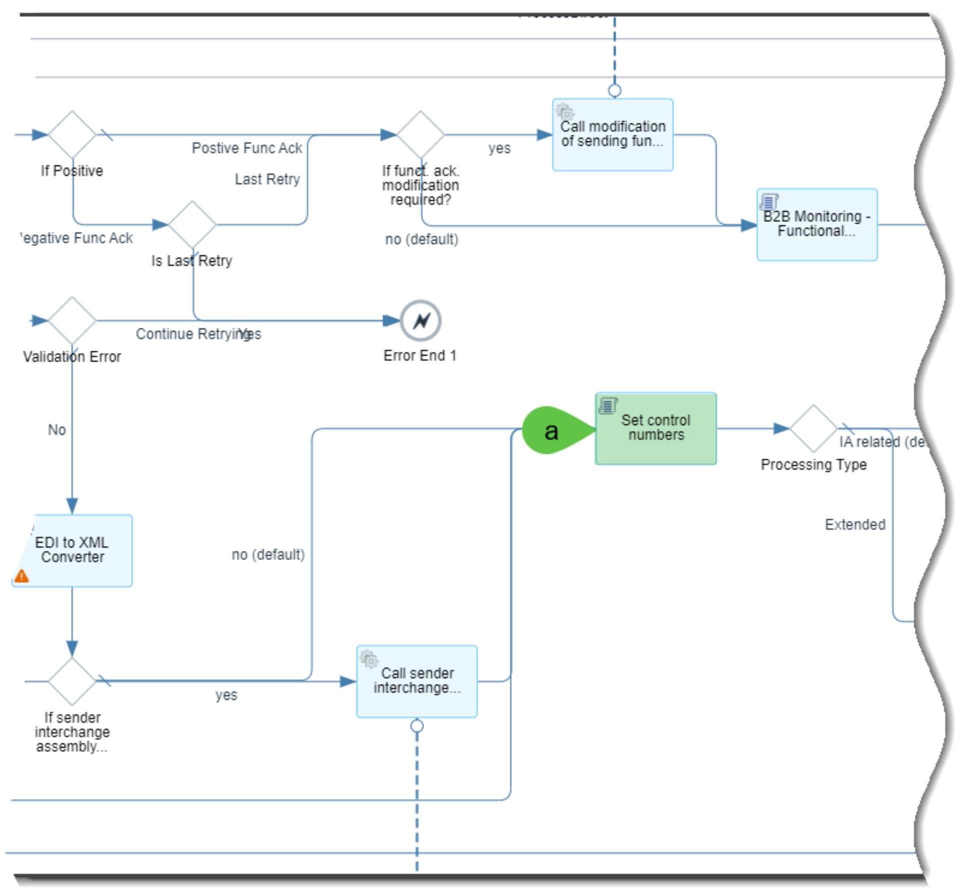
SAP_EDI_REC_Group_Control_Number_Type ::= <Functional Group Number Range Object>, if a sequential number from the corresponding number range object for the functional group is required

SAP_EDI_REC_Interchange_Control_Number_Type ::= <Message Number Range Object>, if a sequential number from the corresponding number range object for the message is required

How is it implemented?

Step 2 - Interchange Processing Flow

→ Main Interchange Processing Flow



Changed integration flow step:

a. Groovy Script: Set control numbers

```
import com.sap.gateway.ip.core.customdev.util.Message;
import java.util.HashMap;
import groovy.json.*;
import com.sap.it.api.pd.PartnerDirectoryService;
import com.sap.it.api.ITApiFactory;
import com.sap.it.api.nrc.NumberRangeConfigurationService;

def Message processData(Message message) {
def NRCS = ITApiFactory.getApi(NumberRangeConfigurationService.class, null);

def headers                                =   message.getHeaders();
def sndDocumentStandard             =   headers.get("SAP_EDI_Document_Standard");
def sndMessageVersion               =   headers.get("SAP_EDI_Message_Version");
def sndMessageRelease               =   headers.get("SAP_EDI_Message_Release");
def sndMessageType                  =   headers.get("SAP_EDI_Message_Type");
def sndInterchangeControlNumber     =   headers.get("SAP_EDI_Interchange_Control_Number");
def sndGroupControlNumber           =   headers.get("SAP_EDI_GS_Control_Number");
// change to SAP_EDI_Message_Control_Number
def sndMessageNumber                =   headers.get("SAP_EDI_Message_Control_Number");
```

```
         def properties                      =   message.getProperties();
         def recInterchangeControlNumberType =
properties.get("SAP_EDI_REC_Interchange_Control_Number_Type");
         def recGroupControlNumberType       =
properties.get("SAP_EDI_REC_Group_Control_Number_Type"); // (b)
         def recMessageNumberType            =
properties.get("SAP_EDI_REC_Message_Number_Type");

         def recInterchangeControlNumber = "";
         def recGroupControlNumber       = "";
    def recMessageNumber             = "";

         if (recInterchangeControlNumberType == null){
        recInterchangeControlNumberType = "ICN_DEFAULT";
    }
    try {
             recInterchangeControlNumber =
NRCS.getNextValuefromNumberRange(recInterchangeControlNumberType, null);
                    recGroupControlNumber       = recInterchangeControlNumber;
                    recMessageNumber            = recInterchangeControlNumber;

             if (recGroupControlNumberType != null){
                     recGroupControlNumber =
NRCS.getNextValuefromNumberRange(recGroupControlNumberType, null); // (c)
             }

             if (recMessageNumberType != null){
                     recMessageNumber = NRCS.getNextValuefromNumberRange(recMessageNumberType,
null);
             }
    } catch(Exception e){
        throw new IllegalStateException("Error reading number range " +
recInterchangeControlNumberType);
    }

    message.setProperty("SAP_EDI_REC_Interchange_Control_Number", recInterchangeControlNumber);
    message.setProperty("SAP_EDI_REC_Group_Control_Number", recGroupControlNumber);
    message.setProperty("SAP_EDI_REC_Message_Number", recMessageNumber);
    message.setProperty("SAP_EDI_SND_Document_Standard", sndDocumentStandard);
    message.setProperty("SAP_EDI_SND_Standard_Version", sndMessageVersion);
    message.setProperty("SAP_EDI_SND_Standard_Release", sndMessageRelease);
    message.setProperty("SAP_EDI_REC_Standard_Message_Type", sndMessageType);
    message.setProperty("SAP_EDI_SND_Interchange_Control_Number", sndInterchangeControlNumber);
    message.setProperty("SAP_EDI_SND_Group_Control_Number", sndGroupControlNumber);
    message.setProperty("SAP_EDI_SND_Message_Number", sndMessageNumber);

    return message;
}
```

[B2BIFACTORY-99] Customized Mapping Process

Introduction

Change log

Date	Release	Responsible	Change comment
2024.06.27		TO	Updated description for additional flow steps. **No test case available but extension point is available**

Requirement

Users require that their own mapping procedures can be performed instead of the standard process that is based on a single set of Integration Advisors generated MIG and MAG processing scripts. In this case, it should be possible to disable the mandatory use of MIGs and MAGs. Examples for this customized mapping process are:

- The use of CPI's mmaps
- The use of custom made XSLTs
- The customized base-/overlay-approach

Solution

The step 2 flow should provide a route to an external flow via ProcessDirect in where a custom mapping integration flow can be assigned. This route should be in front of the call of the local integration process: "Default Mapping".

How to configure?

Set in the section “Activity Parameters” of specific outbound Business Transaction Activity in where a functional acknowledgement is requested the custom parameter:

SAP_EDI_Processing_Type ::= EXTENDED

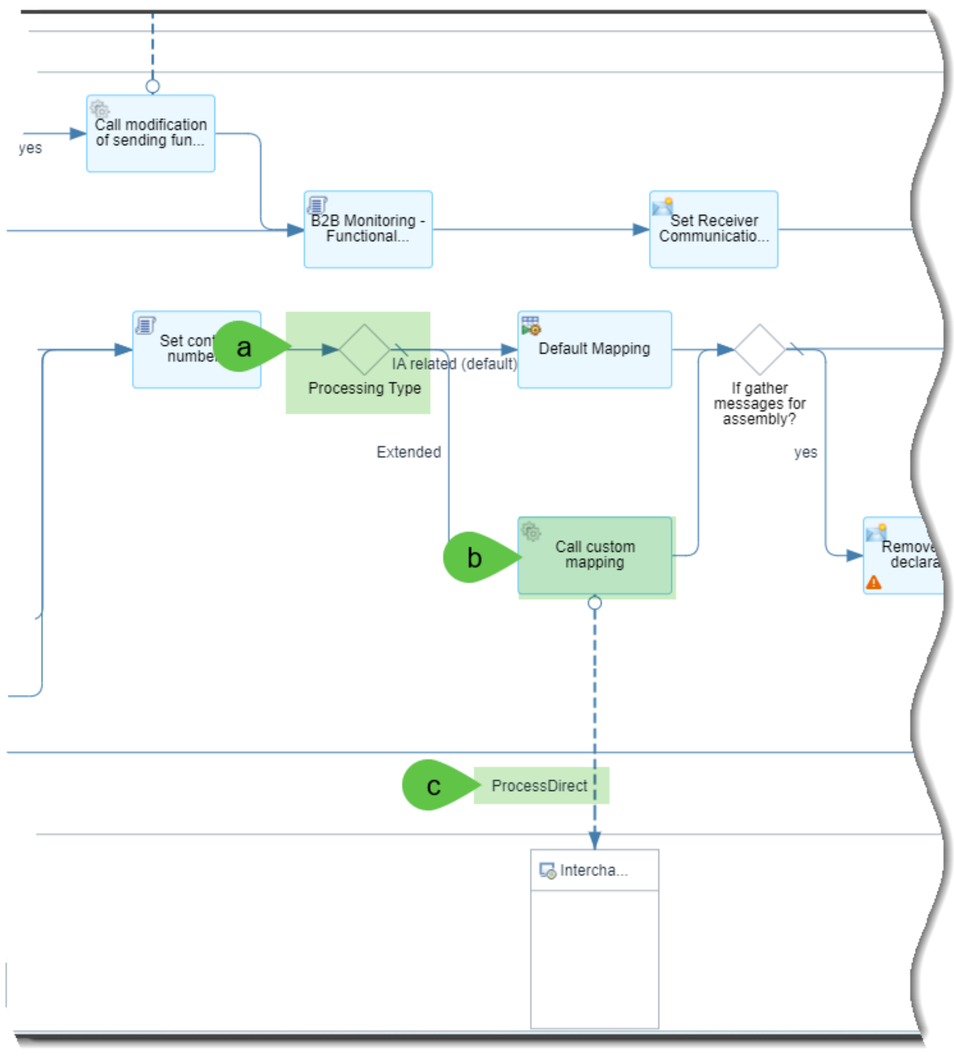
SAP_B2B_Custom_Mapping_ProcessDirectAddress ::= <ProcessDirectAddress> for the process direct related integration flow, in where the external (custom) mapping (may via mmap) is implemented.

Furthermore, you must reference to a source/target Dummy MIG as well a Dummy MAG in this Business Transaction Activity.

How is it implemented?

Step 2 - Interchange Processing Flow Consulting

→ Main Interchange Processing Flow



Additional integration flow steps:

a. Router: Processing Type
   This router will route the payload to the flow step Request Reply: custom mapping flow if the Camel header parameter SAP_EDI_Processing_Type is equal to EXTENDED.

b. Request Reply: Call custom mapping
   This flow step calls the external integration flow via process direct and returns the result to the router "Processing Type"

c. ProcessDirect: ProcessDirect_Custom_Mapping
   It calls the ProcessDirect related integration flow via the address as defined in the Camel exchange property SAP_B2B_Custom_Mapping_ProcessDirectAddress

[B2BIFACTORY-100] Returning Functional Ack. Interchange Modification

Introduction

Change log

Date	Release	Responsible	Change comment
2024.06.27		TO	Updated description for additional flow steps. **No test case available but extention point is available**

Requirement

The from the EDI Splitter generated UN/EDIFACT or ASC X12 interchange with the functional acknowledgement is somehow limited so that this does not quite often fit to the requirements of the trading partner. For e.g., required segments or data elements are missing, or it is not based on the trading partner's requested version.

Solution

There should be in step 2 integration flow a routing decision to an external flow that can be connected via ProcessDirect. This routing decision should be in the route of the functional acknowledgement after the EDI Splitter step.

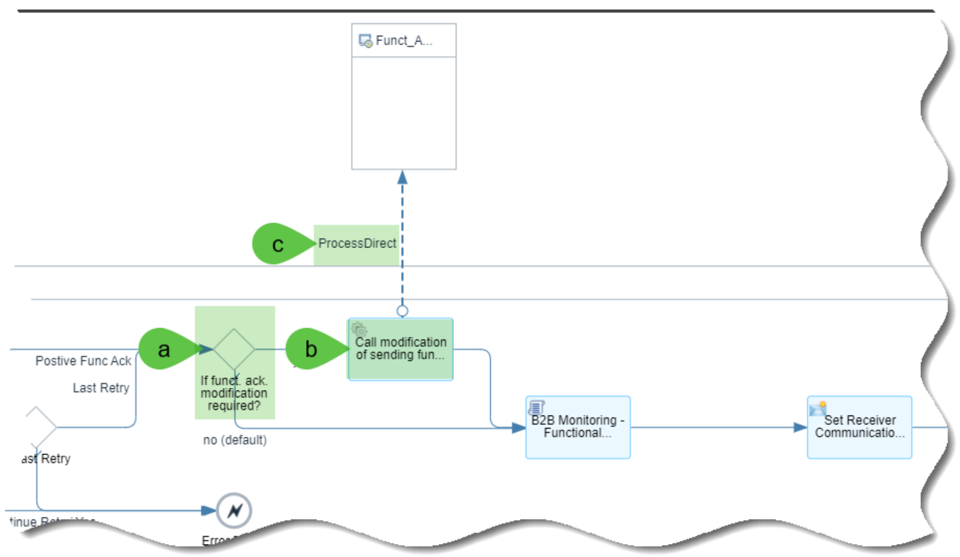
How to configure?

You should set in the section “Activity Parameters” of the relevant Business Transaction Activity the following customer parameters:
SAP_EDI_SendingFunctAck_Modification_ProcessDirectAddress !!= <ProcessDirectAddress> in where the process direct related integration flow for further manipulation of the functional acknowledgement is located.

How is it implemented?

Step 2 - Interchange Processing Flow Consulting

→ Main Interchange Processing Flow



Funct_A...
c
ProcessDirect
a
b
Call modification of sending fun...
Postive Func Ack
Last Retry
If funct. ack. modification required?
no (default)
B2B Monitoring - Functional...
Set Receiver Communicatio...
ast Retry

Additional integration flow steps:

a. Router: If funct. ack. modification required?
   This router will route the payload to the flow step Request Reply: Call modification of sending funct. ack if the Camel exchange property SAP_EDI_SendingFunctAck_Modification_ProcessDirectAddress is not null

b. Request Reply: Call modification of sending funct. Ack.
   This flow step calls the external integration flow via process direct and returns the result to the router "If funct. ack. modification required?"

c. ProcessDirect: ProcessDirect_SendFunctAck_ModificationIt calls the ProcessDirect related integration flow via the address as defined in the Camel exchange property SAP_EDI_SendingFunctAck_Modification_ProcessDirectAddress

[B2BIFACTORY-101] Receiver Interchange Gather Bulk Processing

Introduction

Change log

Date	Release	Responsible	Change comment
2024.06.27		TO	Updated description for additional flow steps. Added how to test section

Requirement

Split sender interchange payloads into single messages made via EDI Splitter or the additional XML based splitter step (see: 14.) Sender XML Interchange Split Processing) should be in some cases collected and assembled to a single receiver interchange payload.

Solution

This should be possible in Step 2 flow via an additional route after the mapping step. The route decision will call the required flow steps for gathering the messages into a single receiver interchange payload and setting the sufficient root tag.

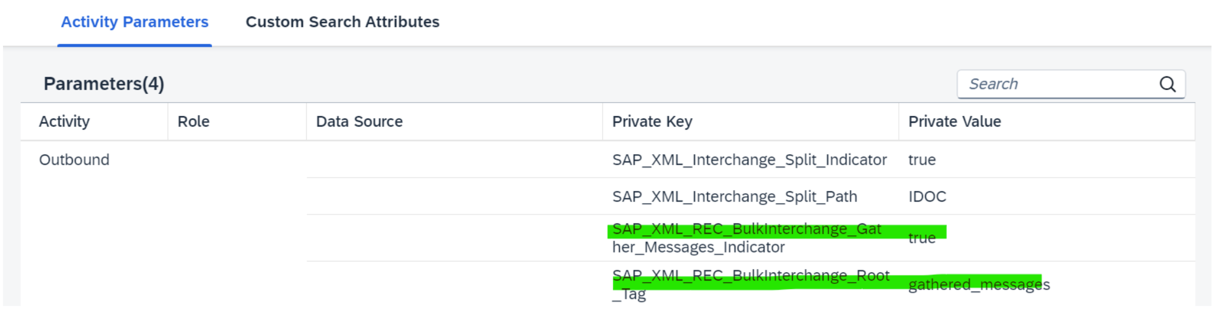
How to configure?

You should set in the section "Activity Parameters" of the relevant Business Transaction Activity the following customer parameters:

SAP_XML_REC_BulkInterchange_Gather_Messages_Indicator !!= true, if the split messages should be gathered for a single bulk receiver interchange.

SAP_XML_REC_BulkInterchange_Root_Tag !!= <RootTag>. This should be root tag for the bulk receiver interchange.

Example:


Activity Parameters | Custom Search Attributes

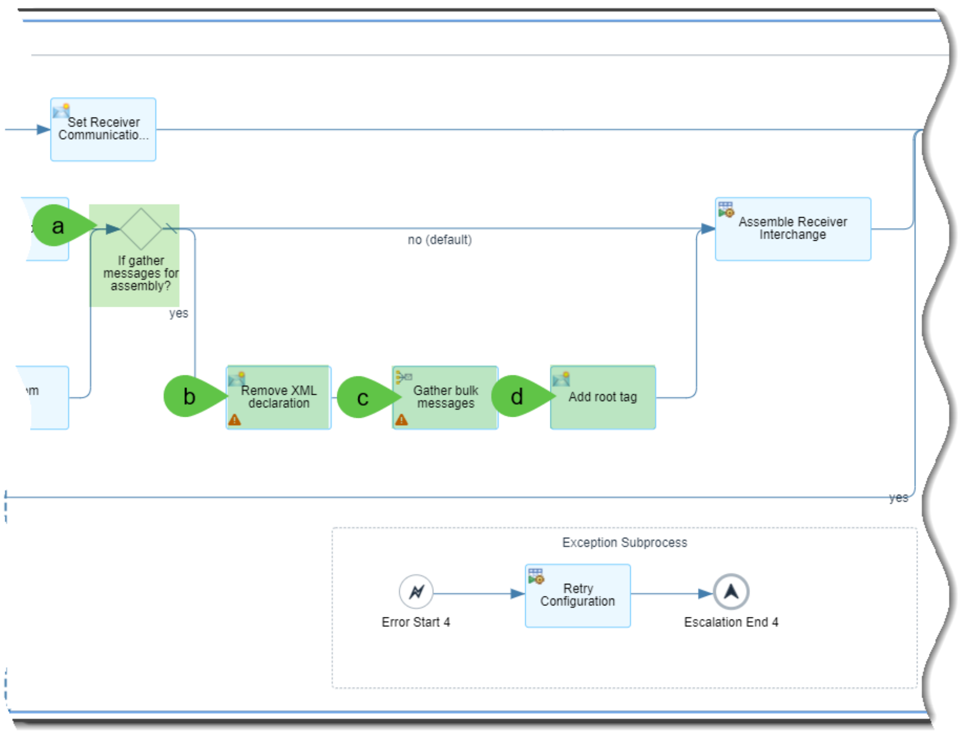
Parameters(4)

Activity	Role	Data Source	Private Key	Private Value
Outbound			SAP_XML_Interchange_Split_Indicator	true
			SAP_XML_Interchange_Split_Path	IDOC
			SAP_XML_REC_BulkInterchange_Gather_Messages_Indicator	true
			SAP_XML_REC_BulkInterchange_Root_Tag	gathered_messages

How is it implemented?

Step 2 - Interchange Processing Flow Consulting

→ Main Interchange Processing Flow



Additional integration flow steps:

a. Router: If gather messages for assembly?
This router will route the payload to the flow step if the Camel exchange property SAP_XML_REC_BulkInterchange_Gather_Messages_Indicator is true.

b. Content Modifier: Remove XML declaration.
This XML modifier removes the XML declaration from payload.

c. Gather: Gather bulk messages
This gather step merges interchanges into a single interchange that have been split in previous steps.

d. Content Modifier: Add root tag.

This content modifier inserts the Camel exchange property SAP_XML_REC_BulkInterchange_Root_Tag as a root tag into the payload.

[B2BIFACTORY-115] Receiver Interchange Assembly Processing

Introduction

Change log

	Release	Responsible	Change comment
2024.06.27		TO	Updated description for additional flow steps. **No test case available but extention point is available**

Requirement

Just as the individual declaration and extraction of custom required sender interchange payloads on the sender side, it should be possible to declare custom type system on the receiver side which also allow a custom-made assembly of the receiver interchange payload.

Solution

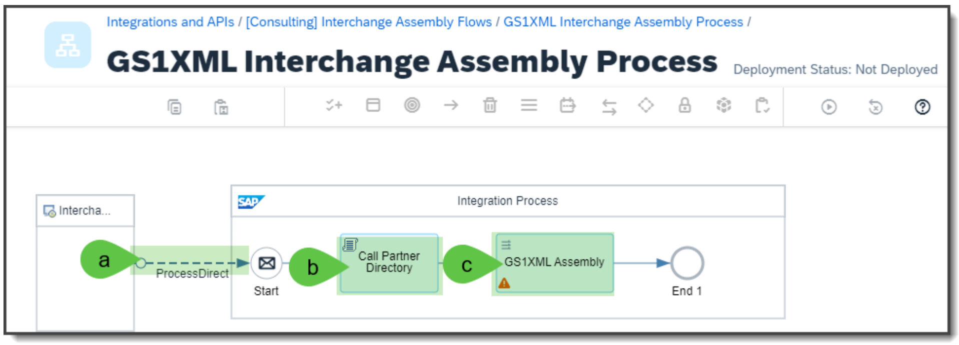
Instead of obtaining a fixed set of assembly XSLTs that are provided via the bootstrap content, it is recommended that the assembly is outsourced in separate integration flows which will be called in the Step 2 integration flow --> local integration process: "Assemble Receiver Interchange" via a ProcessDirect step which calls this separate assembly integration flow via the type system name which should be a part of the ProcessDirect address.

How to configure?

If you want to process a non-supported type system, you should set in the section “Activity Parameters” of the relevant Business Transaction Activity the following customer parameter:
SAP_EDI_REC_Document_Standard ::= <Type System Value> of the to be processed type system

There should be a sperate assembly integration flow per supported receiver type system in the integration package: [Consulting] Interchange Assembly Flows.

Example:


Integrations and APIs / [Consulting] Interchange Assembly Flows / GS1XML Interchange Assembly Process /
GS1XML Interchange Assembly Process
Deployment Status: Not Deployed
Intercha...
Integration Process
a
ProcessDirect
Start
b
Call Partner Directory
c
GS1XML Assembly
End 1

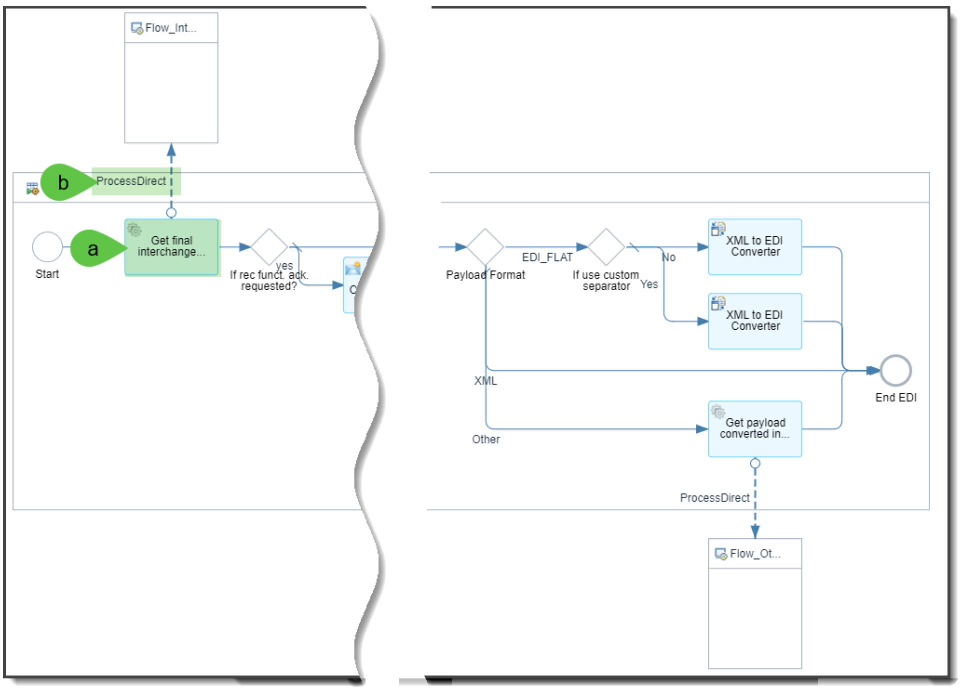
The steps are:

a. ProcessDirect (Mandatory): The ProcessDirect address should be: /tpm/assembly-flow/{{SAP_EDI_REC_Document_Standard}} in where the SAP_EDI_REC_Document_Standard represents the externalized parameter for the setting of the type system (for e.g. GS1-XML), which has to be processed by this flow.
b. Groovy Script “Call Partner Directory” (Mandatory): This script is necessary for obtaining the TPA –-> Business Transaction Activity relevant parameters from the partner directory, so that these can be inserted into the header by the next flow step.
c. XSLT Mapping “<Type System> Assembly” (Mandatory): This XSLT script is responsible to insert the values obtained from Partner Directory (see step (b)) into the appropriate locations of the receiver interchange/message header-

How is it implemented?

Step 2 - Interchange Processing Flow Consulting

→ Assemble Receiver Interchange



Additional integration flow steps:

(a) Request Reply: Get final interchange assembly.
This flow step calls the external integration flow via process direct

(b) ProcessDirect: ProcessDirect_Assembly_Flow_Address
It calls the ProcessDirect related integration flow via the address as defined as /tpm/assembly-flow/${property.SAP_EDI_REC_Document_Standard}

[B2BIFACTORY-116] XML to Other Syntax Conversion

Introduction

Change log

Date	Release	Responsible	Change comment
2024.06.27		VD	Added Additional Flow steps section and changed the processdirect address

Requirement

It is required that receiver interchange payload must be converted to a different syntax representation (e.g., CSV). Very often, receiver interchange payloads that are based on custom type systems should be converted to flat file representation such as CSV or fixed length.

Solution

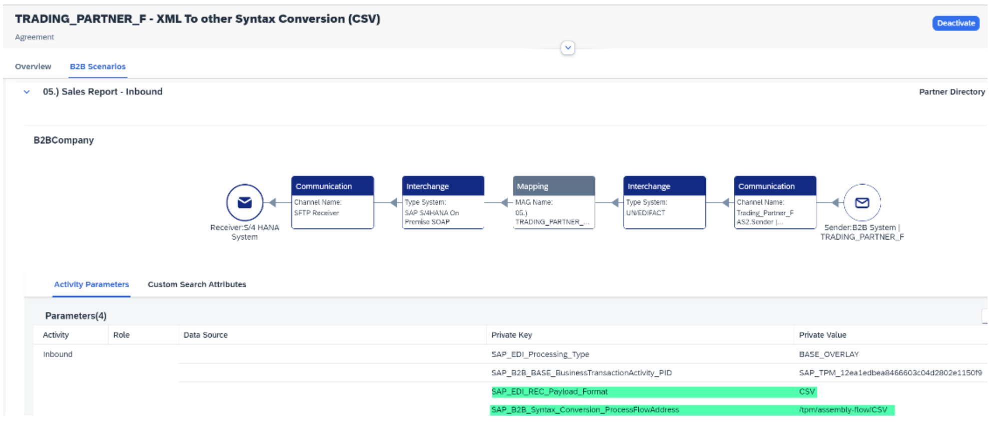
In the Step 2 integration flow --> local integration process: "Assemble Receiver Interchange" the router which routes the receiver interchange payloads to the EDI conversion should be extended with another route. This route should call an external integration flow via ProcessDirect in where the conversion step should take place.

How to configure?

If you want to convert into another syntax than the supported syntax representations, you should set in the section “Activity Parameters” of the relevant Business Transaction Activity the following customer parameters:
SAP_EDI_REC_Payload_Format ::= <Syntax Type> which should be other than EDI_FLAT or XML
SAP_B2B_Syntax_Conversion_ProcessFlowAddress ::=
ProcessDirect address of the flow, which calls the process direct related integration flow in where the syntax conversion should happen.

Activity Paramaters configuration in TPA:

SAP_EDI_REC_Payload_Format should be set to **CSV**
SAP_B2B_Syntax_Conversion_ProcessFlowAddress should be set **/tpm/assembly-flow/CSV** (Receiver/syntax-conversion)



TRADING_PARTNER_F - XML To other Syntax Conversion (CSV)

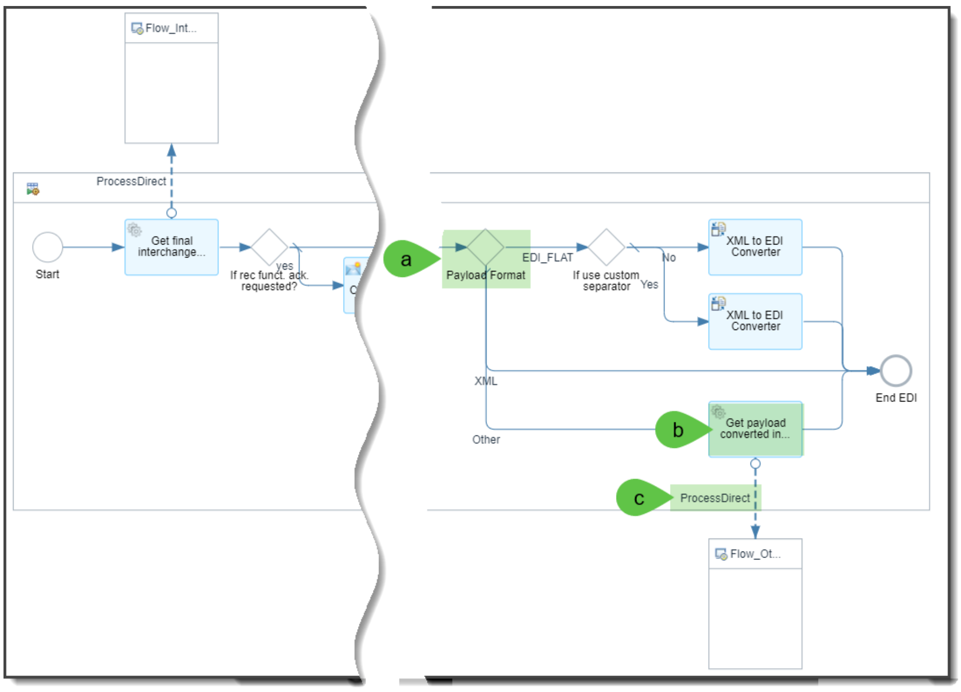
Agreement

Parameters(4)

Activity	Role	Data Source	Private Key	Private Value
Inbound			SAP_EDI_Processing_Type	BASE_OVERLAY
			SAP_B2B_BASE_BusinessTransactionActivity_PID	SAP_TPM_12ea1edbea8466603c04d2802e1150f9
			SAP_EDI_REC_Payload_Format	CSV
			SAP_B2B_Syntax_Conversion_ProcessFlowAddress	/tpm/assembly-flow/CSV

How is it implemented?

Step 2 - Interchange Processing Flow
→ Assemble Receiver Interchange



Additional integration flow steps:

a. Router: Payload Format
Router Conditions:

- XML: If the Receiver Interchange Payload is in XML format
  The exchange property ${property.SAP_EDI_REC_Payload_Format} is XML then the payload is routed through this route
- Other: If the Receiver Interchange Payload is not in EDI_FLAT, XML format

The exchange property ${property.SAP_EDI_REC_Payload_Format} is other than EDI_FLAT and XML then the payload is routed through this route to an external iflow to carry out the conversion.

- EDI_Flat: If the Receiver Interchange Payload is in EDI_Flat format
  The exchange property ${property.SAP_EDI_REC_Payload_Format} is EDI_FLAT format (CSV or fixed length) then the payload is routed through this route to another Router "If use Custom Seperator"

b. Request Reply: This flow step calls an external Integration Flow via Processdirect and gets payload which is in Other format (not in XML or EDI_Flat format) converted in CSV.

c. ProcessDirect: ProcessDirect_SyntaxConversion
This flow step call an external integration flow which converts the Other Format to Flat File representation such as CSV or Fixed Length.

[B2BIFACTORY-104] Toggle Sender- and Receiver-Name in Returning Functional Acknowledgement Interchange

Introduction

Change log

Date	Release	Responsible	Change comment
2024.06.28		VD	Updated Additional Flow steps section, how it is implemented and how to test section

Requirement

It is required that for the CONTRL message in the B2B Monitoring the Interchange Parameter are swapped:

Solution

In Step 3 Integration Flow --> Integration Process --> B2B Monitoring Functional Ack Sent.groovy to fetch the required header parameters and swap them.

How to configure?

Not applicable

How is it implemented?

Step 3 - Receiver Communication Flow

→Integration Process: B2B Monitoring - Functional Acknowledgement Sent Event

```
import org.osgi.framework.FrameworkUtil;
import com.sap..ip.core.customdev.util.Message;
import com.sap.it.op.b2b.monitor.api.*;
import com.sap.it.op.b2b.monitor.api.events.*;
import java.nio.charset.StandardCharsets;

def Message processData(Message message) {

    def headers = message.getHeaders();
    def tpaReceiverTpName = message.getProperty("SAP_TPA_REC_Trading_Partner_Name");
    def tpaSenderTpName   = message.getProperty("SAP_TPA_SND_Trading_Partner_Name");
    def recMsgType = headers.get("SAP_EDI_Message_Type");
    def recDocumentStandard = headers.get("SAP_EDI_Document_Standard");
    def recInterchangeControlNumber = headers.get("SAP_EDI_Interchange_Control_Number");

    //get B2BMonitoring APIs
    def bundleContext =
FrameworkUtil.getBundle(Class.forName("com.sap.gateway.ip.core.customdev.util.Message")).
getBundleContext();
    def serviceRef =
bundleContext.getServiceReference(Class.forName("com.sap.it.op.b2b.monitor.api.B2BMonitor
ingApi"));
    B2BMonitoringApi api = (B2BMonitoringApi) bundleContext.getService(serviceRef);
    def payloadValidationMessage = message.getProperty("SAP_XmlValidationResult");

    //create event and assign to MPL
    FunctionalAcknowledgementSentEvent functionalAckSentEvent =
api.createFunctionalAcknowledgementSentEvent();

functionalAckSentEvent.setMonitoringReference(headers.get("SAP_MessageProcessingLogID"));
    functionalAckSentEvent.setMonitoringReferenceType(MonitoringReferenceType.MPL);

    //update Business Document of the envelope message
    // BusinessDocument documentFA =
functionalAckSentEvent.createUpdateBusinessDocument(message.getProperty("EnvelopeDocument
_ID"));
    BusinessDocument documentFA =
functionalAckSentEvent.createUpdateBusinessDocument(message.getProperty("Document_ID"));
    documentFA.setSenderTradingPartnerName(tpaReceiverTpName);
    documentFA.setReceiverTradingPartnerName(tpaSenderTpName);
    documentFA.setReceiverDocumentStandard(recDocumentStandard);
    documentFA.setSenderMessageType(recMsgType);
    documentFA.setReceiverMessageType(recMsgType);
    documentFA.setReceiverInterchangeControlNumber(recInterchangeControlNumber);

    if(payloadValidationMessage != null){
        if(payloadValidationMessage.toString().indexOf("fatal") != -1){
            documentFA.setProcessingStatus(ProcessingStatus.FAILED);
        }
        else{
            documentFA.setProcessingStatus(ProcessingStatus.COMPLETED);
        }
    }
    else{
        documentFA.setProcessingStatus(ProcessingStatus.COMPLETED);
    }

    //update Functional Acknowledgement
    def fAckId = message.getProperty("SAP_FuncAck_ID");
    FunctionalAcknowledgement functionalAck =
functionalAckSentEvent.createUpdateFunctionalAcknowledgement(fAckId);

    //set Functional Acknowledgment Status
    switch (message.getProperty("SAP_FuncAck_Status")) {
        case "ACCEPTED":
            functionalAck.setStatus(FunctionalAcknowledgementStatus.ACCEPTED);
            break;
        case "REJECTED":
            functionalAck.setStatus(FunctionalAcknowledgementStatus.REJECTED);
            break;
        default:
            break;
    }
    def body = message.getBody(java.lang.String) as String;
    functionalAck.setPayload(body.getBytes(StandardCharsets.UTF_8));

    //set Functional Acknowledgment Transmission Status
    functionalAck.setTransmissionStatus(TransmissionStatus.COMPLETED);

    //Submit Functional Acknowledgment
    functionalAckSentEvent.submit();

    return message;
}
```

[B2BIFACTORY-105] Step 3 - Receiver Communication Flow - Extended → Integration Process: AS2 Receiver Channel and (SFTP) Receiver Channel

Introduction

Change log

Date	Release	Responsible	Change comment
2024.07.11		TO	**No test case available but extention point is available**

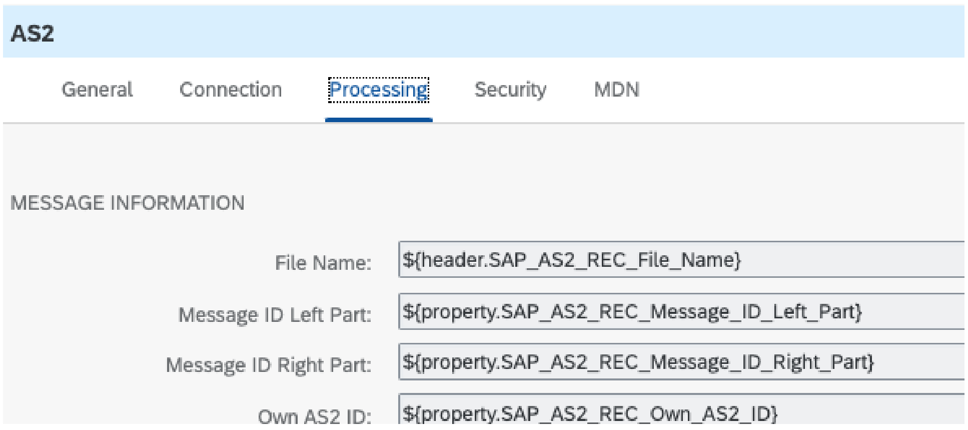
Requirement

The FileName is mostly generated dynamically via expressions which must be resolved first via a Custom Integration Flow. In this the FileName must be passed as a header to Step 3 Flow

Solution

FileName should be declared as header

FileName must be a header not property



MESSAGE INFORMATION

File Name:	${header.SAP_AS2_REC_File_Name}
Message ID Left Part:	${property.SAP_AS2_REC_Message_ID_Left_Part}
Message ID Right Part:	${property.SAP_AS2_REC_Message_ID_Right_Part}
Own AS2 ID:	${property.SAP_AS2_REC_Own_AS2_ID}

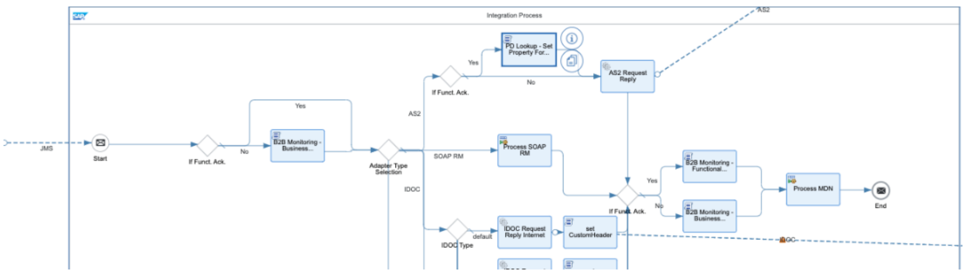
How to configure?

TBD

How is it implemented?

Step 3 - Receiver Communication Flow

→Integration Process: Functional Acknowledge PD Lookup



++ Func Ack: partnerID must be: "SAP_TPM_PARTNER_ID"

def partnerID = headers.get("SAP_TPM_PARTNER_ID");

[B2BIFACTORY-106] Set Custom Search Attributes (with multiple values)

Introduction

Change log

Date	Release	Responsible	Change comment
2024.06.20		VD	Created. **No test case available but extention point is available**

Requirement

Custom search attributes in B2B scenarios can be used to enhance the search experience for users and help them find exactly what they are looking for. These attributes can be tailored to the specific needs of B2B buyers and sellers, such as industry-specific product features, technical specifications, pricing options, and more. By allowing users to filter search results based on these custom attributes, businesses can provide a more personalized and efficient search experience. Additionally, custom search attributes can also be used to gather valuable data on customer preferences and behaviour, which can be leveraged to further optimize the B2B buying and selling process.

Solution

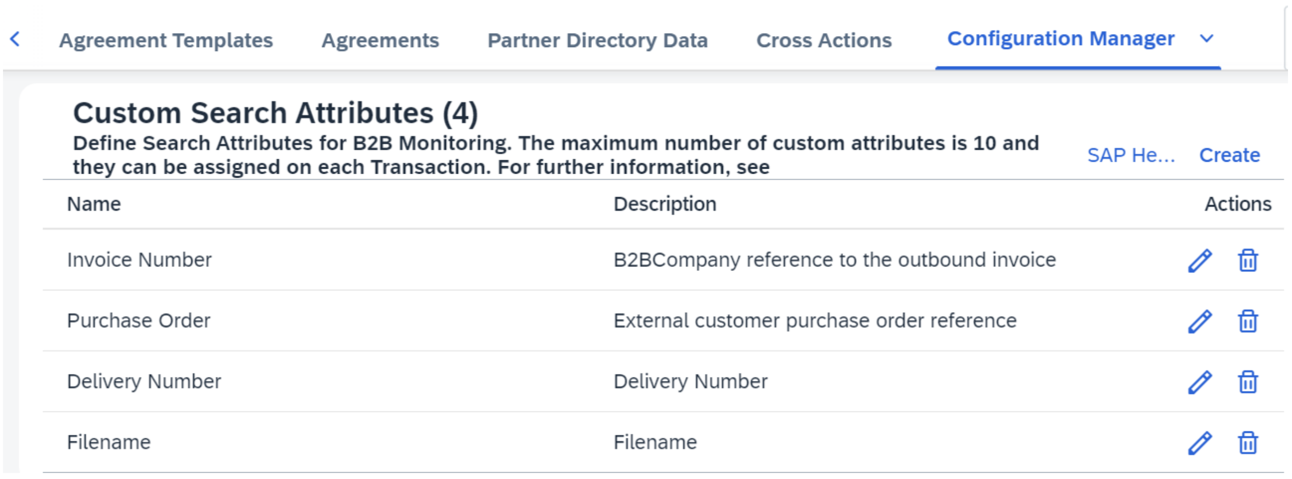
TBD

How to configure?

Custom Search Attributes can be maintained for both Inbound as well as Outbound Transactions.

To configure custom Search Attributes in Configuration Manager

1. Login into the application and navigate to **Design > B2B Scenarios.**
2. Select the **Configuration Manager** tab and choose Create to create custom search attribute.(**Note**: **Maximum of 10 custom search attributes can be created**).
3. Enter a meaningful name in the **Name** field and provide a description in the **Description** field.
4. Choose Save. The custom search attribute has been created successfully.
5. After creating custom search attributes, these attributes need to be assigned to the transactions for them to reflect in B2B Monitor.



Custom Search Attributes (4)

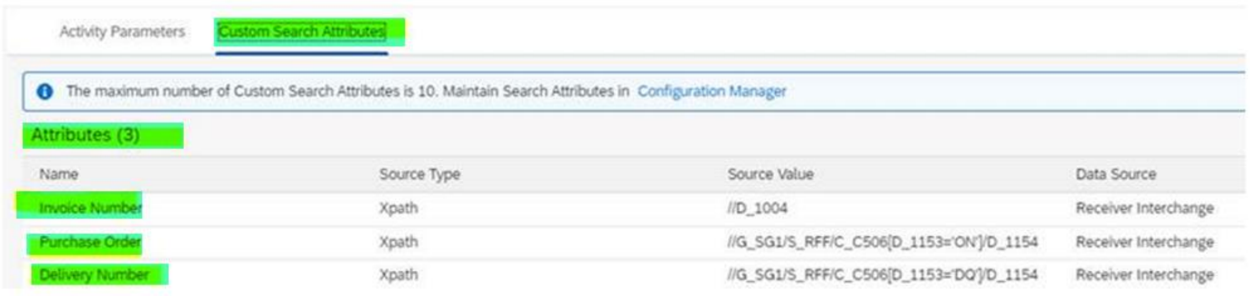
Define Search Attributes for B2B Monitoring. The maximum number of custom attributes is 10 and they can be assigned on each Transaction. For further information, see

Name	Description	Actions
Invoice Number	B2BCompany reference to the outbound invoice	
Purchase Order	External customer purchase order reference	
Delivery Number	Delivery Number	
Filename	Filename	

To add custom Search Attributes to the transaction

1. Navigate to the **Custom Search Attributes** tab and choose Add.
2. In the resulting Dialog, maintain the following fields:

Field	Description
Name	Select a value from the list of attributes that are created via Configuration Manager
Source Type	Select whether the source type is **Parameter** or **XPATH**
Source Value	Enter the source value for the name. If **Source type** is **Parameter** , then enter the parameter here. If the **Source type** is **XPath**, then enter the XPath of the source value.
Data Source	Select whether the data source should be **Sender Interchange** or **receiver Interchange**
Business Transaction Activity	Select the direction of the business transaction activity from the drop-down list

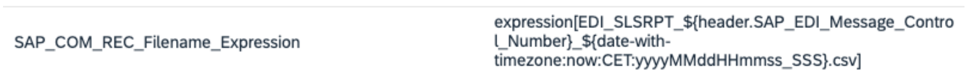


The maximum number of Custom Search Attributes is 10. Maintain Search Attributes in Configuration Manager

Attributes (3)

Name	Source Type	Source Value	Data Source
Invoice Number	Xpath	//D_1004	Receiver Interchange
Purchase Order	Xpath	//G_SG1/S_RFF/C_C506[D_1153='ON']/D_1154	Receiver Interchange
Delivery Number	Xpath	//G_SG1/S_RFF/C_C506[D_1153='DQ']/D_1154	Receiver Interchange

If the Source Type is Parameter , the custom search attribute should be maintained in the Trading Partner Agreement under Activity Parameters tab in B2B Scenarios.

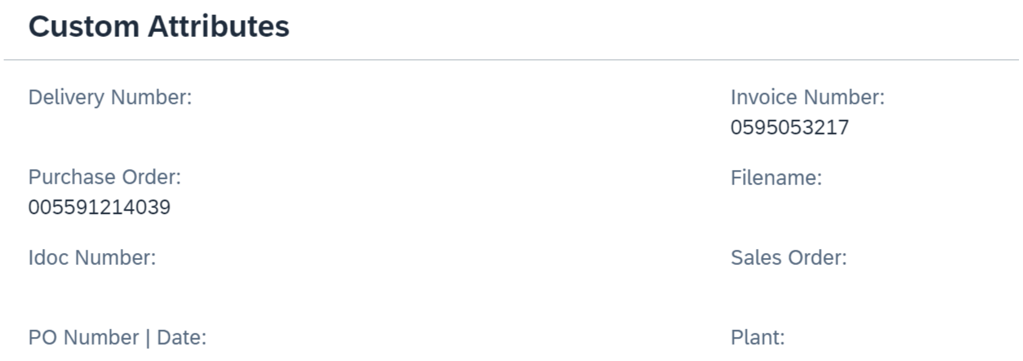


SAP_COM_REC_Filename_Expression	expression[EDI_SLSRPT_${header.SAP_EDI_Message_Control_Number}_${date-with-timezone:now:CET:yyyyMMddHHmmss_SSS}.csv]

Once the custom search attributes are created and defined, they should be used in the necessary B2B transactions in an agreement. These attributes are then pushed into Partner directory once the agreement is activated.

What will be the result?

The incoming payload consists of the search values in XPath format and when the payload is parsed, the values are read and displayed in the B2B monitor.



Custom Attributes

Delivery Number:	Invoice Number: 0595053217
Purchase Order: 005591214039	Filename:
Idoc Number:	Sales Order:
PO Number | Date:	Plant:

How is it implemented?

TBD

[B2BIFACTORY-108] Base-Overlay Approach

Introduction

Change log

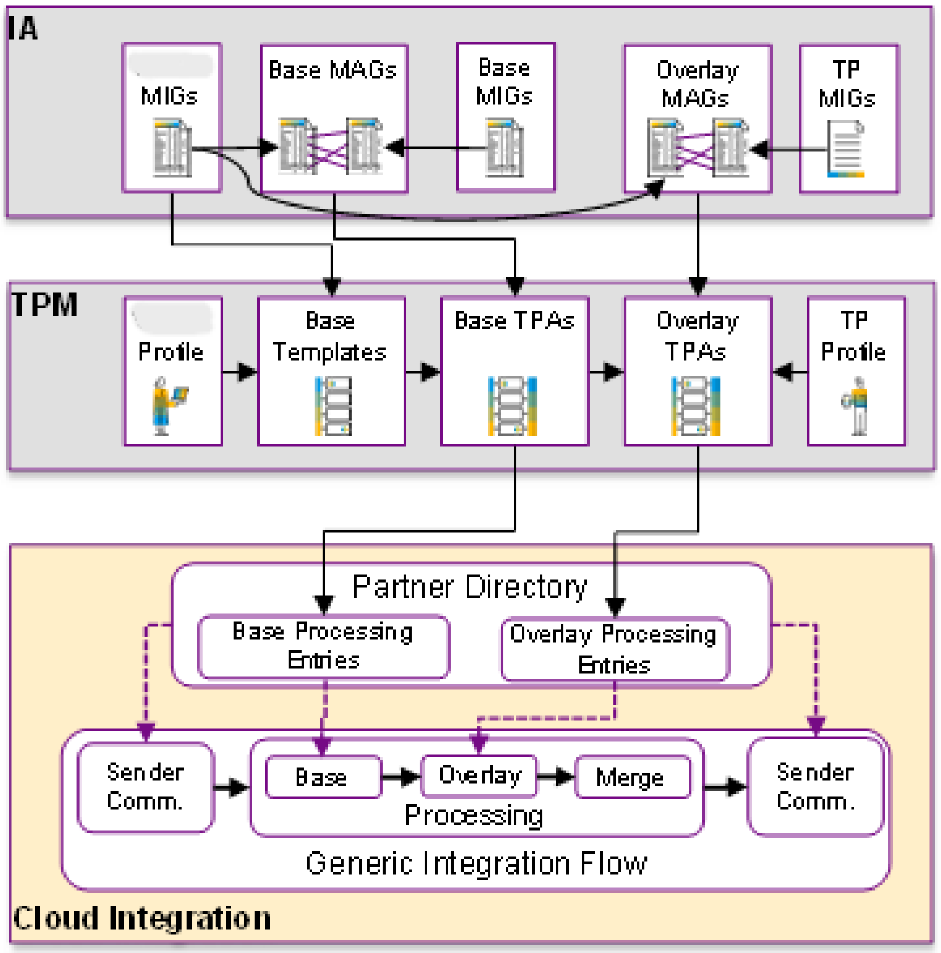
Date	Release	Responsible	Change comment
2024.06.20		TO	Created.

Requirement

To minimize the maintenance efforts, the Base Overlay Approach divides the TPAs and their related design time artifacts into two parts – Base and Overlay:

- Base-TPAs – Covers the common requirements across all Trading Partners that request the same B2B standard (such as UN/EDIFACT) but in the latest supported version of the Company (such as D.10B). There must be one Base-TPA per requested B2B standard. The provided Base-TPA related runtime processing entries in the Partner Directory are used for processing and mapping in front of the Overlay-TPA related processing entries. All changes in the Base-TPA and its related design artifacts will have an immediate effect on all Overlay-TPAs that are based on this Base-TPA. For this purpose, the updated base related runtime processing entries must be pushed one time into the Partner Directory.
- Overlay-TPAs – Covers the Trading Partner's specific requirements in the given B2B standard of the leading Base-TPA (such as in UN/EDIFACT), but in its required version (such as D.96A). The specific requirements should be the delta changes in comparison to the Base-TPA design time artifacts. The provided Overlay-TPA related runtime processing entries in the Partner Directory are used for processing and mapping after the Base-TPA related processing entries. The results of the Overlay related processing entries overrule the results of the Base processing entries. Updates in Overlay-TPAs will just belong to the overlay related processing entries and will not influence any other processing entry.

The following figure displays an abstract architecture picture of Base and Overlay related design time artifacts and their runtime related processing entries.

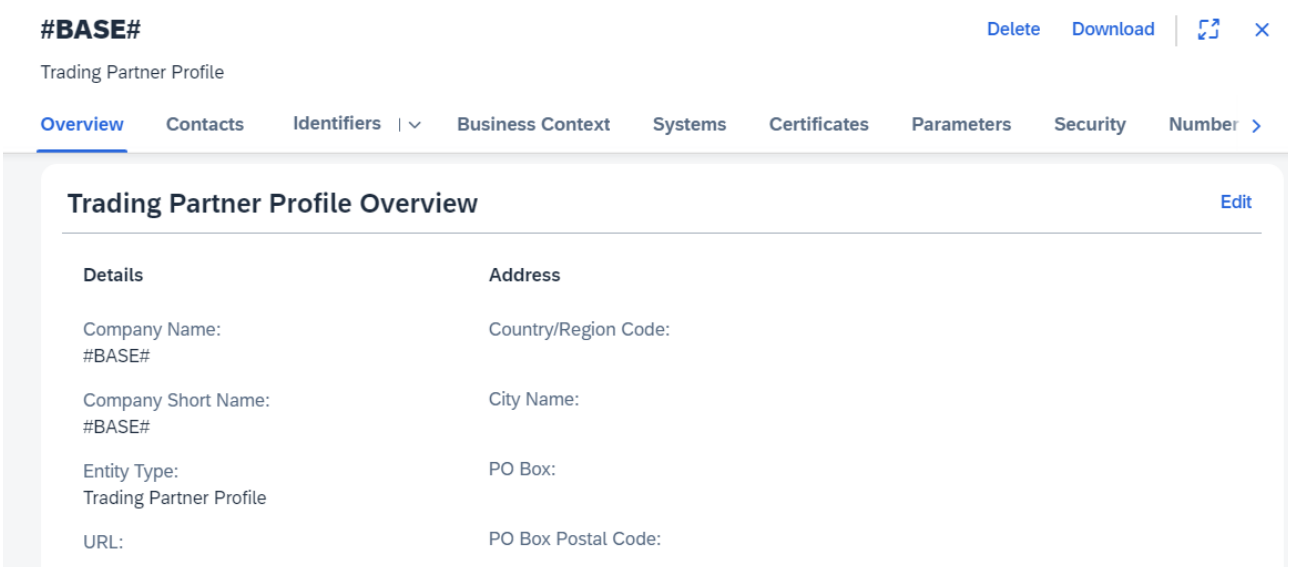


How to configure?

You should create a BASE TPA and an Overlay TPA as described in the following steps.

Step 1: Create a BASE Trading Partner Agreement

1. Navigate to **Design > B2B Scenarios > Partner Profiles**.
2. Choose **Create > Trading Partner** to create a Base Trading Partner Profile (TPP). This Base TPP is a dummy TPP and required for a Base TPA.
    a. Navigate to **Overview** tab and enter #BASE# as company name



Trading Partner Profile Overview

Edit

Details

Company Name:
#BASE#

Company Short Name:
#BASE#

Entity Type:
Trading Partner Profile

URL:

Address

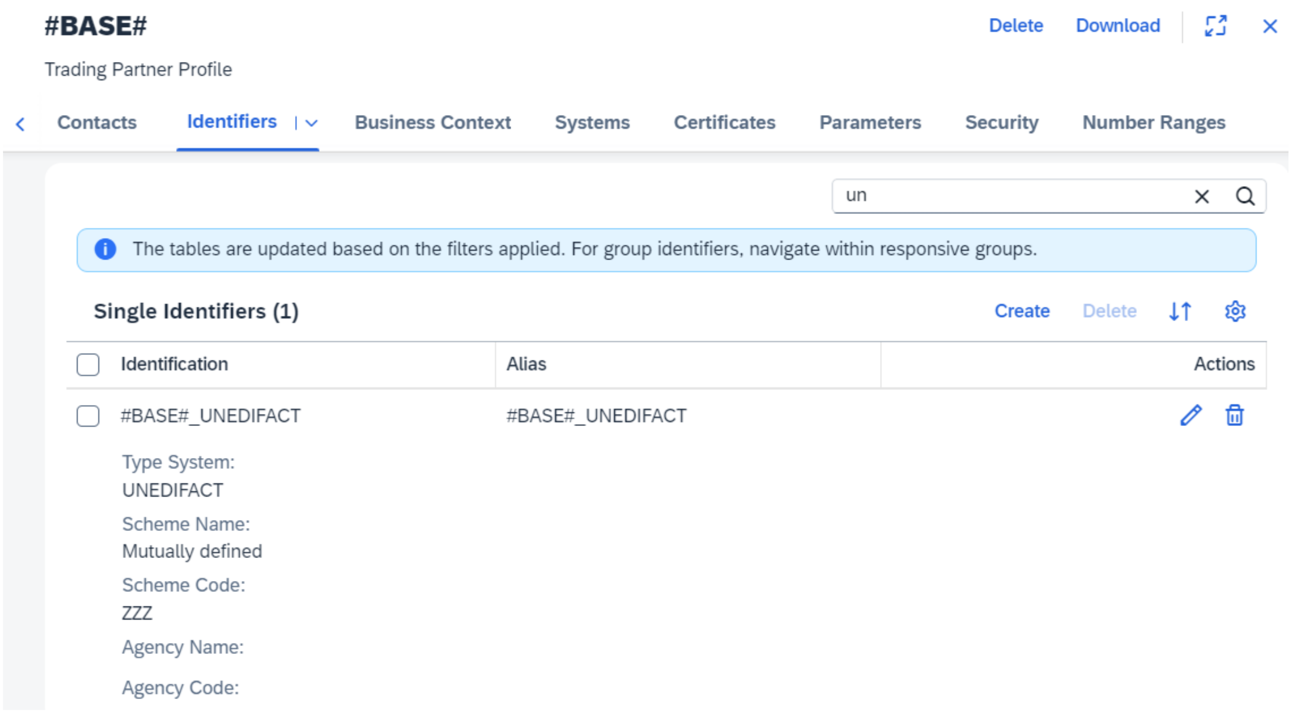
Country/Region Code:

City Name:

PO Box:

PO Box Postal Code:

b. Navigate **Identifiers** tab. Click **Create** and enter dummy values as the BASE identifiers for each type system (in this example, one for UN/EDIFACT and the other one for IDOC identifier created).



The tables are updated based on the filters applied. For group identifiers, navigate within responsive groups.

Single Identifiers (1)

Identification	Alias	Actions
#BASE#_UNEDIFACT	#BASE#_UNEDIFACT	

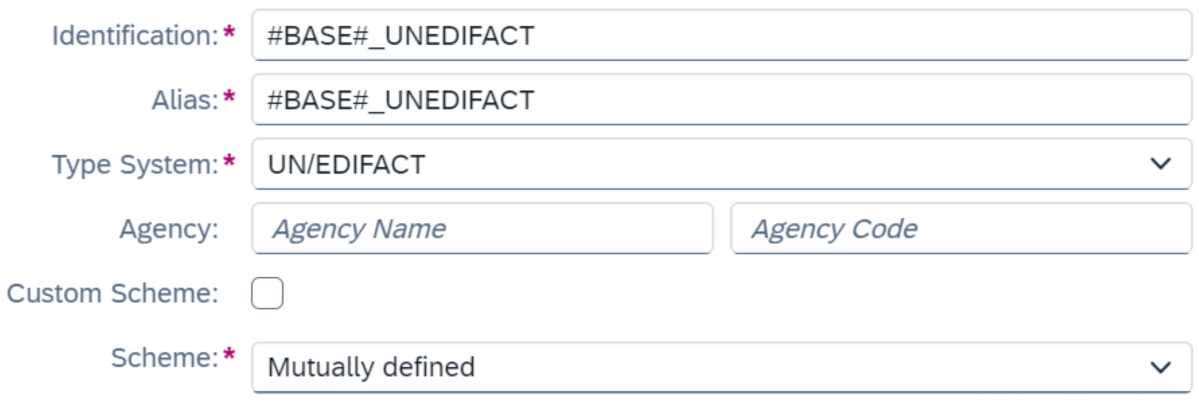
Type System:
UNEDIFACT

Scheme Name:
Mutually defined

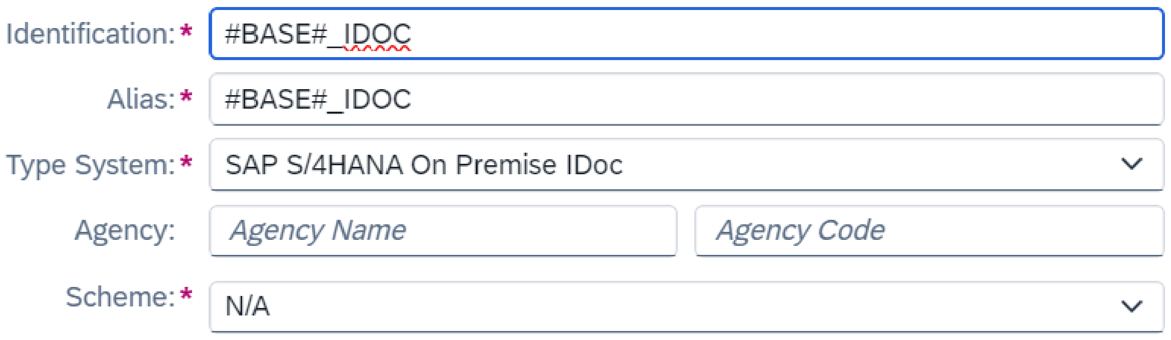
Scheme Code:
ZZZ

Agency Name:

Agency Code:

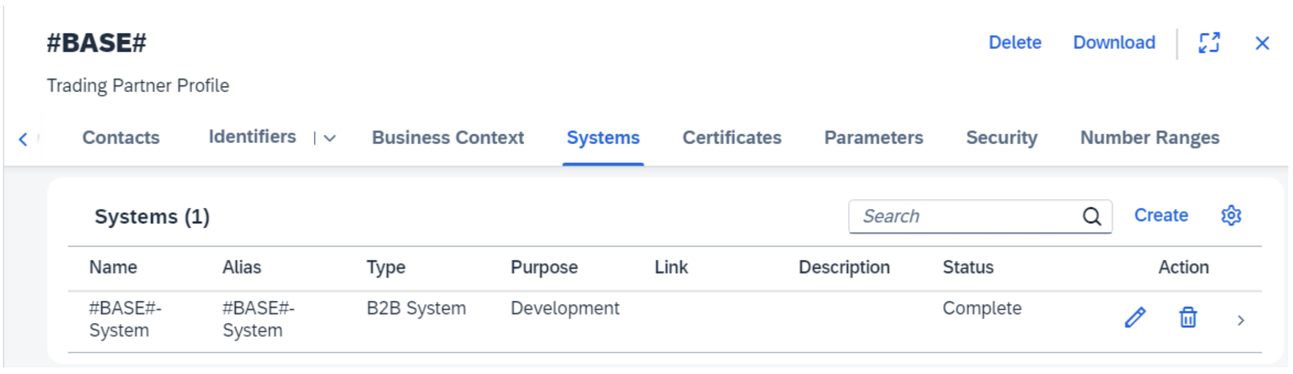


Identification: *	#BASE#_UNEDIFACT
Alias: *	#BASE#_UNEDIFACT
Type System: *	UN/EDIFACT
Agency:	Agency Name / Agency Code
Custom Scheme:	☐
Scheme: *	Mutually defined



Identification: *	#BASE#_IDOC
Alias: *	#BASE#_IDOC
Type System: *	SAP S/4HANA On Premise IDoc
Agency:	Agency Name / Agency Code
Scheme: *	N/A

c. Navigate to **Systems** tab. Create a B2B system, configure type systems, and create a dummy sender and receiver channel like below screenshots.

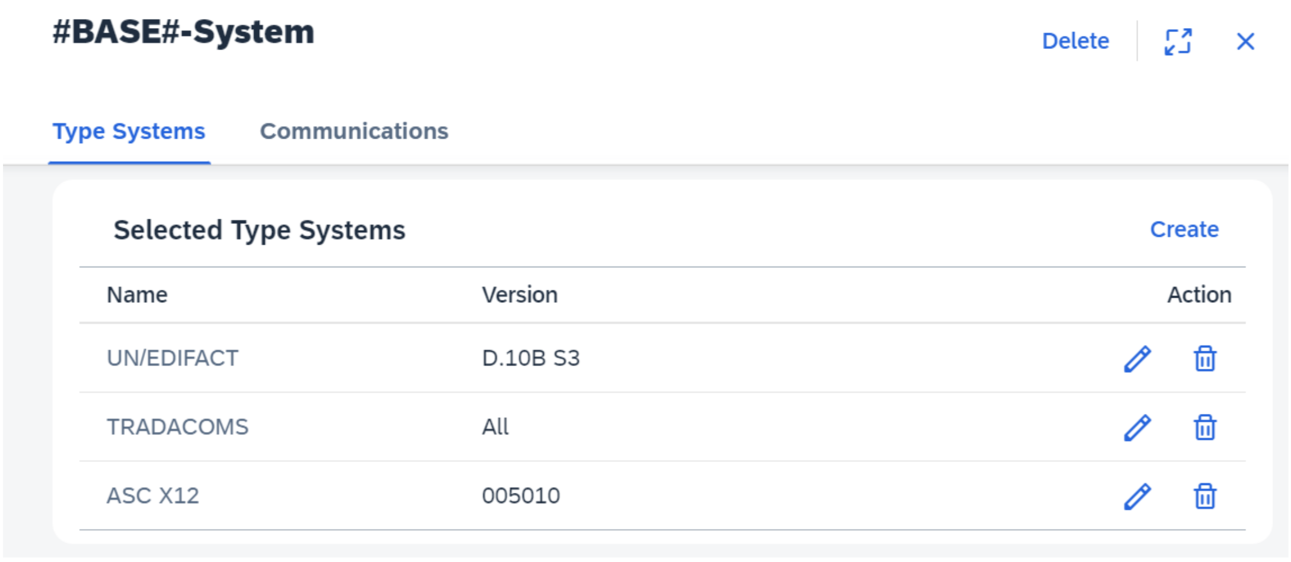


#BASE#

Trading Partner Profile

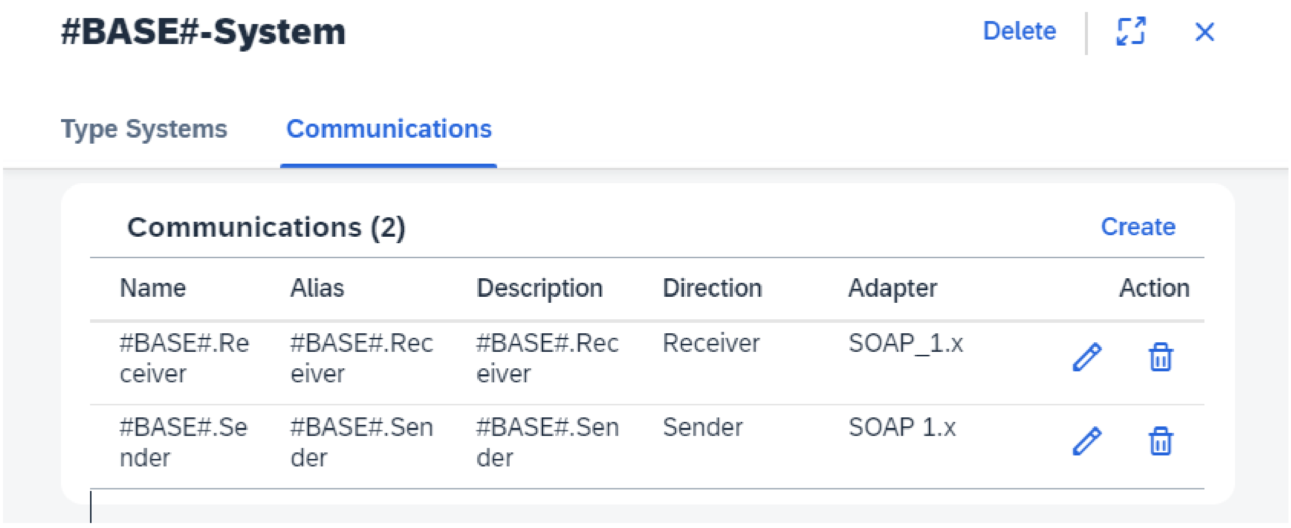
Systems (1)

Name	Alias	Type	Purpose	Link	Description	Status	Action
#BASE#-System	#BASE#-System	B2B System	Development			Complete	



Selected Type Systems

Name	Version	Action
UN/EDIFACT	D.10B S3	
TRADACOMS	All	
ASC X12	005010	



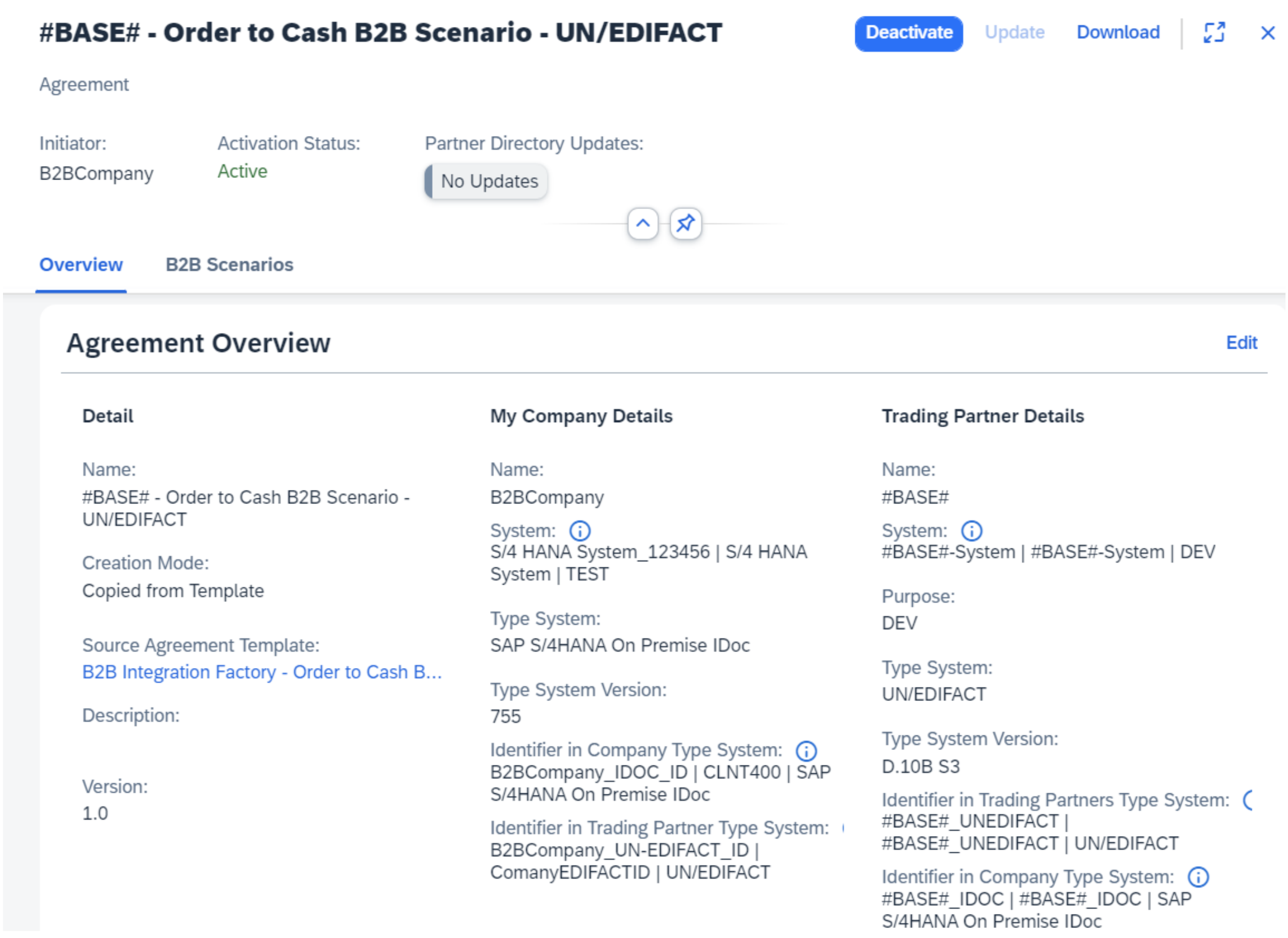
Communications (2)

Name	Alias	Description	Direction	Adapter	Action
#BASE#.Receiver	#BASE#.Receiver	#BASE#.Receiver	Receiver	SOAP_1.x	
#BASE#.Sender	#BASE#.Sender	#BASE#.Sender	Sender	SOAP 1.x	

3. Navigate to **Design > B2B Scenarios > Agreements** and choose **Create** to create a Base Trading Partner Agreement (TPA). Choose a TPA template and select #BASE# from Trading Partner drop down. If you haven't created a Base TPP, you don't see #BASE# in the drop down.

Note: This Base TPA should cover the base related integration content such as company MIGs for all defined business transaction activities as defined in the TPA template.

a. In **Overview** tab, maintain Trading Partner Details with system, type system, type system version and identifiers as you configured in BASE TPP in the previous steps.


#BASE# - Order to Cash B2B Scenario - UN/EDIFACT

Deactivate | Update | Download

Agreement

Initiator: B2BCompany

Activation Status: Active

Partner Directory Updates: No Updates

Overview | B2B Scenarios

Agreement Overview

Edit

Detail

Name: #BASE# - Order to Cash B2B Scenario - UN/EDIFACT

Creation Mode: Copied from Template

Source Agreement Template: B2B Integration Factory - Order to Cash B...

Description:

Version: 1.0

My Company Details

Name: B2BCompany

System: S/4 HANA System_123456 | S/4 HANA System | TEST

Type System: SAP S/4HANA On Premise IDoc

Type System Version: 755

Identifier in Company Type System: B2BCompany_IDOC_ID | CLNT400 | SAP S/4HANA On Premise IDoc

Identifier in Trading Partner Type System: B2BCompany_UN-EDIFACT_ID | ComanyEDIFACTID | UN/EDIFACT

Trading Partner Details

Name: #BASE#

System: #BASE#-System | #BASE#-System | DEV

Purpose: DEV

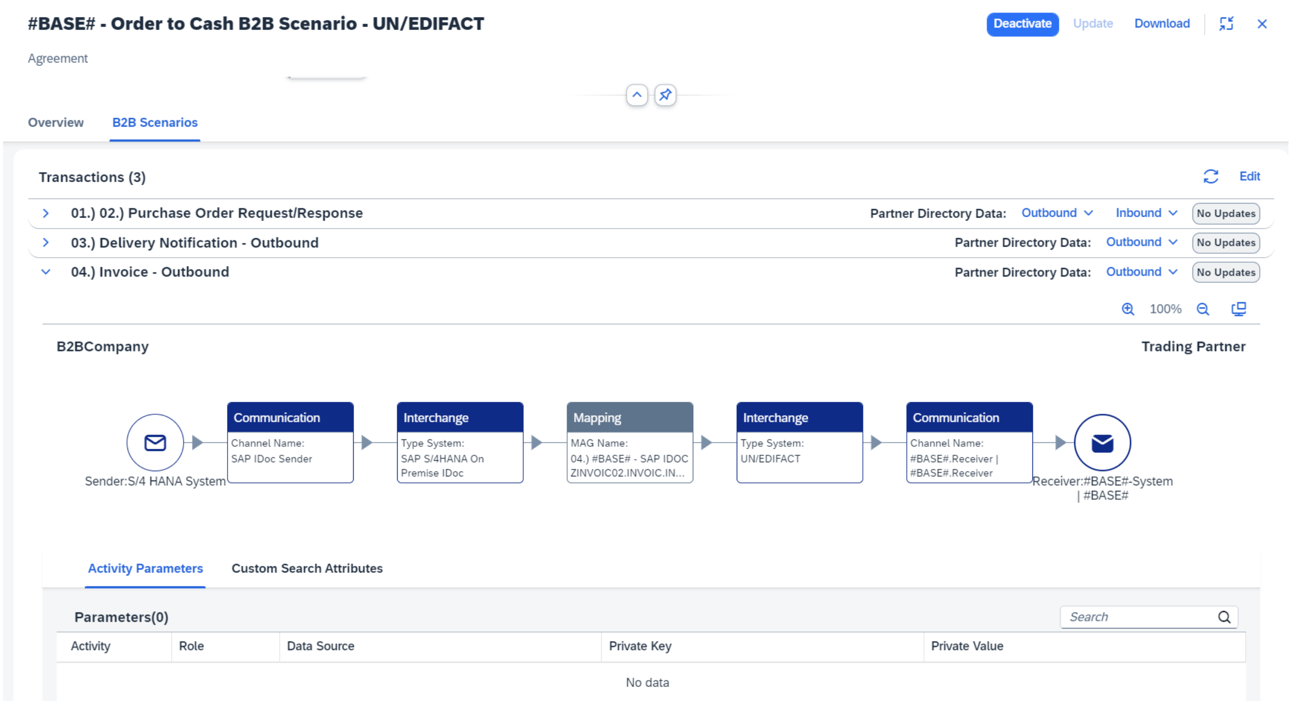
Type System: UN/EDIFACT

Type System Version: D.10B S3

Identifier in Trading Partners Type System: #BASE#_UNEDIFACT | #BASE#_UNEDIFACT | UN/EDIFACT

Identifier in Company Type System: #BASE#_IDOC | #BASE#_IDOC | SAP S/4HANA On Premise IDoc

4. For each defined business transaction activity, you should now:
    a. Create a Base MIG based on a message type with latest agreed version of the necessary type system.
    b. Initially create a Base MAG between this Base MIG and the corresponding Company MIG as described in chapter Base MAG.
    c. Create the base custom pre- or post-processing integration flows as described in chapter Base Custom Pre- and Post-Processing. These are especially relevant for the transformation of incompatible versions of message types.
    d. Assign all the created integration content from step a. - c. at the business transaction activity


#BASE# - Order to Cash B2B Scenario - UN/EDIFACT

Agreement

Transactions (3)

01.) 02.) Purchase Order Request/Response	Partner Directory Data: Outbound Inbound	No Updates
03.) Delivery Notification - Outbound	Partner Directory Data: Outbound	No Updates
04.) Invoice - Outbound	Partner Directory Data: Outbound	No Updates

B2BCompany → Trading Partner

Sender:S/4 HANA System → Communication (Channel Name: SAP IDoc Sender) → Interchange (Type System: SAP S/4HANA On Premise IDoc) → Mapping (MAG Name: 04.) #BASE# - SAP IDOC ZINVOIC02.INVOIC.IN...) → Interchange (Type System: UN/EDIFACT) → Communication (Channel Name: #BASE#.Receiver | #BASE#.Receiver) → Receiver:#BASE#-System | #BASE#

Parameters(0)

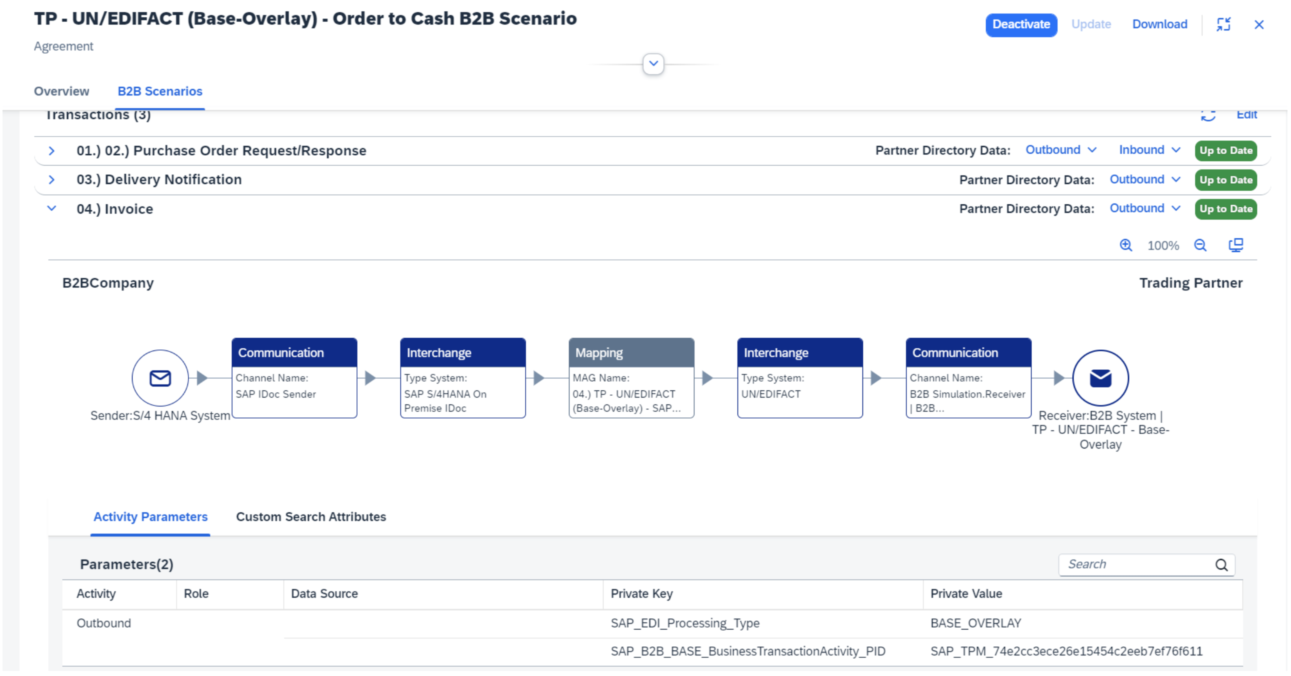
Activity	Role	Data Source	Private Key	Private Value

No data

5. Activate the Base TPA, if you are finished with all business transaction activities.

Step 2: Create an Overlay Trading Partner Agreement

1. Create an Overlay Trading Partner Profile (TPP).
2. Create an Overlay Trading Partner Agreement (TPA).
3. For each defined business transaction activity, you should now:
    a. Create an Overlay MIG based on the agreed type system and its version.
    b. Initially create an Overlay MAG between this Overlay MIG and the corresponding Company MIG
    c. Create and assign the trading partners's custom pre- or post-processing integration flows as described in chapter Overlay Custom Pre- and Post-Processing. These are especially relevant for the transformation of incompatible versions of message types.
    d. Assign all the created integration content from step a. - c. at the business transaction activity
4. In the Overlay TPA, the following custom activity parameters should be maintained
    - **SAP_EDI_Processing_Type = BASE_OVERLAY**
    - **SAP_B2B_BASE_BusinessTransactionActivity_PID = <your BASE PID >** (e.g. SAP_TPM_95e8354a6ad07f3758fd8e13aef52b4e)


TP - UN/EDIFACT (Base-Overlay) - Order to Cash B2B Scenario

Agreement

	Partner Directory Data			
01.) 02.) Purchase Order Request/Response	Partner Directory Data:	Outbound	Inbound	Up to Date
03.) Delivery Notification	Partner Directory Data:	Outbound		Up to Date
04.) Invoice	Partner Directory Data:	Outbound		Up to Date

100%
B2BCompany
Trading Partner
Sender:S/4 HANA System
Communication
Channel Name: SAP IDoc Sender
Interchange
Type System: SAP S/4HANA On Premise IDoc
Mapping
MAG Name: 04.) TP - UN/EDIFACT (Base-Overlay) - SAP...
Interchange
Type System: UN/EDIFACT
Communication
Channel Name: B2B Simulation.Receiver | B2B...
Receiver:B2B System | TP - UN/EDIFACT - Base-Overlay
Activity Parameters
Custom Search Attributes
Search

Parameters(2)

Activity	Role	Data Source	Private Key	Private Value
Outbound			SAP_EDI_Processing_Type	BASE_OVERLAY
			SAP_B2B_BASE_BusinessTransactionActivity_PID	SAP_TPM_74e2cc3ece26e15454c2eeb7ef76f611

5. Activate the Overlay TPA, if you are finished with all the business transaction activities.
6. Test the activated Overlay TPA using the prepared test cases in the test bed and refine it according the deltas that are shown as result of each test case.

How is it implemented?

Step 2b – Interchange Processing Flow – Base-Overlay Mapping Process