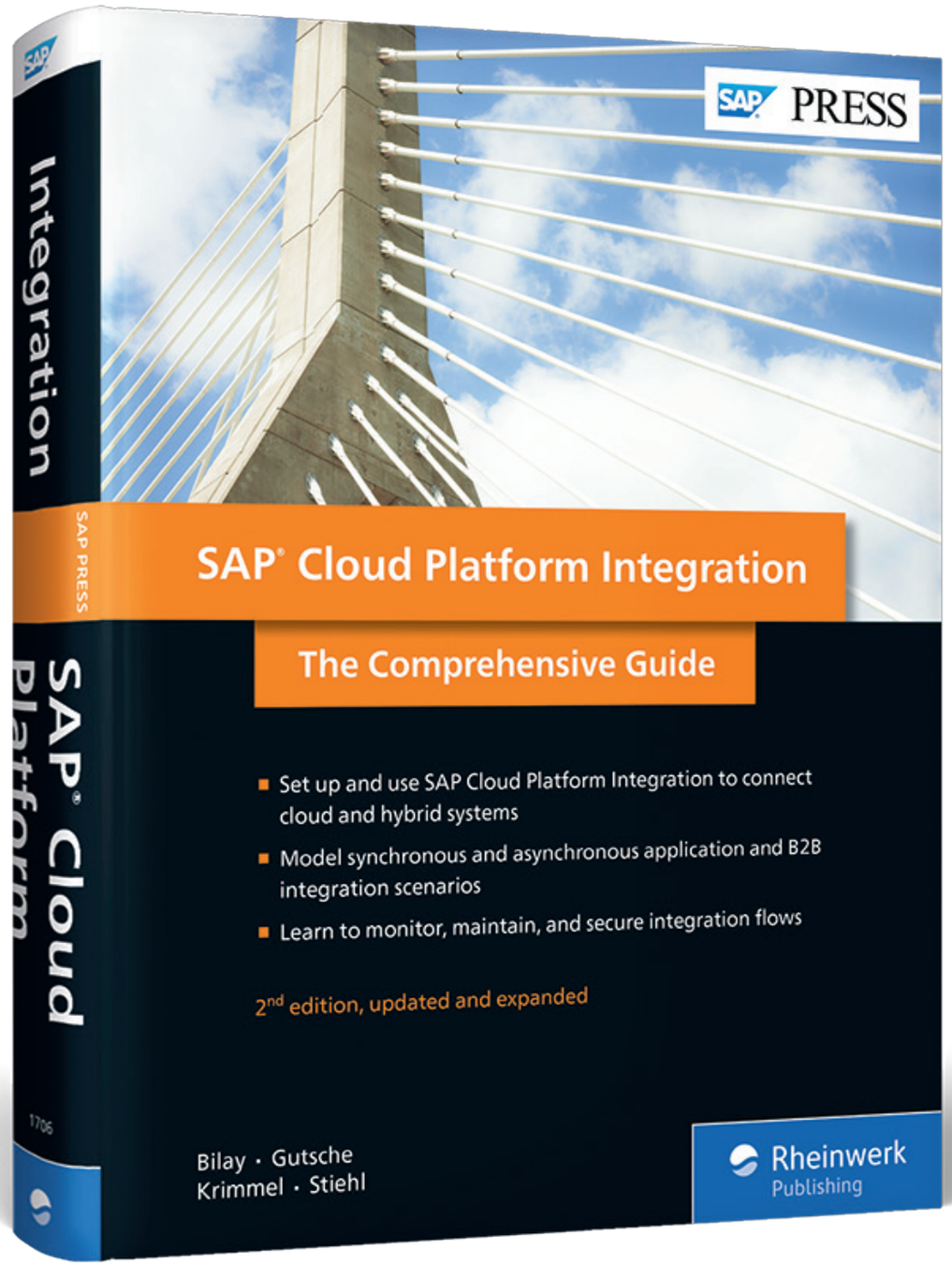
SAP PRESS
First-hand knowledge.
Rheinwerk Publishing



Browse the Book

Understand the capabilities of SAP Cloud Platform Integration to support business-to-business (B2B) integration. This chapter guides you along an end-to-end scenario through the usage of specific SAP components and tools such as the Integration Content Advisor, the Partner Directory, and more.

"B2B Integration with SAP Cloud Platform Integration"

Contents

Index

The Authors

John Mutumba Bilay, Peter Gutsche, Mandy Krimmel, Volker Stiehl

SAP Cloud Platform Integration: The Comprehensive Guide

792 Pages, 2018, $89.95
ISBN 978-1-4932-1706-9

www.sap-press.com/4650

Chapter 7

B2B Integration with SAP Cloud Platform Integration

The end-to-end flow of a business-to-business (B2B) integration project includes defining and implementing interfaces for different partners, creating mappings between those interfaces, and, finally, setting up the integration scenarios. With the B2B capabilities provided with the SAP Cloud Platform Integration, Enterprise Edition, SAP supports you throughout your entire B2B integration project. In this chapter, you'll get to know the B2B features provided in SAP Cloud Platform Integration and learn how to use them in your B2B integration project.

In the previous chapters, you learned how to implement simple as well as more complex integration scenarios using SAP Cloud Platform Integration. But SAP Cloud Platform Integration can also support you in simplifying, streamlining, and configuring complex business-to-business (B2B) integration processes.

Integration between different businesses, for example, between a manufacturer and a wholesaler, is known as B2B integration. B2B integration typically relies on a variety of industry standards for electronic business document exchange, including Accredited Standards Committee X12 (ASC X12), United Nations Electronic Data Interchange for Administration, Commerce and Transport (UN/EDIFACT), and SAP Intermediate Document (IDoc).

B2B integration projects are known to be long-running and complex projects that imply tedious and time-consuming tasks. Throughout such a project, the following tasks need to be fulfilled:

- Defining interfaces for the involved partners
- Creating mappings between the interfaces
- Maintaining partner-specific configuration data
- Creating integration content

- Deploying and testing the integration content
- Monitoring the integration scenario

This chapter will describe the B2B capabilities available with the SAP Cloud Platform Integration, Enterprise Edition, that can help you execute those integration tasks.

7.1 B2B Capabilities in SAP Cloud Platform Integration: Overview

To support you in your B2B integration projects SAP Cloud Platform Integration provides a set of B2B-specific capabilities, such as a web-based application to define interfaces and mappings, B2B-specific adapters and flow steps, and a persistency to store configuration data for different business partners.

Table 7.1 gives you a complete overview of the available B2B capabilities in SAP Cloud Platform Integration. Note that this list reflects the set of capabilities available at the time of publishing the book. More B2B-specific adapters and flow steps are on the roadmap and will be provided in future releases. For the most recent list of B2B-specific features, check out the online documentation for SAP Cloud Platform Integration at *https://help.sap.com/viewer/product/CLOUD_INTEGRATION/Cloud*.

Capability	Description
Integration Content Advisor (ICA)	Web-based application that facilitates the definition of interfaces based on industry standards and the configuration of mappings between those interfaces. You can export documentation and generated runtime artifacts from this tool. The generated runtime artifacts can be used in integration flows.
Library of type systems	A collection of Electronic Data Interchange (EDI) standard interfaces provided by agencies that maintain the B2B standards. You can access these libraries from the ICA. Each of the type systems is developed and maintained by the agency that owns it. For example, the SAP IDoc type system is developed and maintained by SAP. The external libraries need to be purchased separately.
Partner Directory	Repository to store configuration data for different business partners. Application Programming Interfaces (APIs) are available to maintain the data in the Partner Directory. The configuration data from the Partner Directory can be used in integration flow configuration.

Table 7.1 B2B Capabilities in SAP Cloud Platform Integration

Capability	Description
Number range objects (NROs)	Artifact to define unique interchange numbers for each EDI document. NROs can be used in scripts and in the EDI splitter in integration flow configuration.
AS2 Sender and Receiver Adapter	Adapter in integration flow designer to exchange business documents with your partner using the AS2 (Applicability Statement 2) protocol. Can be used to encrypt, decrypt, sign, and verify documents.
AS4 Receiver Adapter	Adapter in the integration flow designer to exchange business documents with your partner using the AS4 (Applicability Statement 4) protocol.
EDI splitter	Variant of the splitter step in the integration flow designer that splits, validates, and acknowledges inbound bulk EDI messages.
EDI to XML Converter	Converter step in the integration flow designer that transforms a message from EDI format to XML format. You can convert EDIFACT and ASC X12 formats.
XML to EDI Converter	Converter step in the integration flow designer that transforms a message from XML format to EDI format. You can convert to EDIFACT and ASC X12 formats.

Table 7.1 B2B Capabilities in SAP Cloud Platform Integration (Cont.)

B2B Capabilities Only Available in SAP Cloud Platform Integration, Enterprise Edition

Some of the B2B capabilities are only available in your tenant if you've purchased the SAP Cloud Platform Integration, Enterprise Edition. Otherwise, you can't use the ICA or the B2B-specific flow steps and adapters when designing integration flows.

What can we do with those capabilities, and how do they help us in our tasks throughout the B2B integration project? We give answers to these questions in the following sections. We'll explain the B2B capabilities in detail and show how to use most of them throughout one sample B2B scenario.

In the sample B2B scenario, the sender posts EDI messages to SAP Cloud Platform Integration using the AS2 protocol. SAP Cloud Platform Integration receives and acknowledges receipt of the message, transforms it into an IDoc message, and sends

it to the receiver using the IDoc adapter. At the end of the chapter, we'll make the integration flow dynamic so that the configuration data is read from the Partner Directory, where the partner-specific configuration data is stored. See Figure 7.1 for an overview of the different configuration steps necessary to set up the sample scenario.

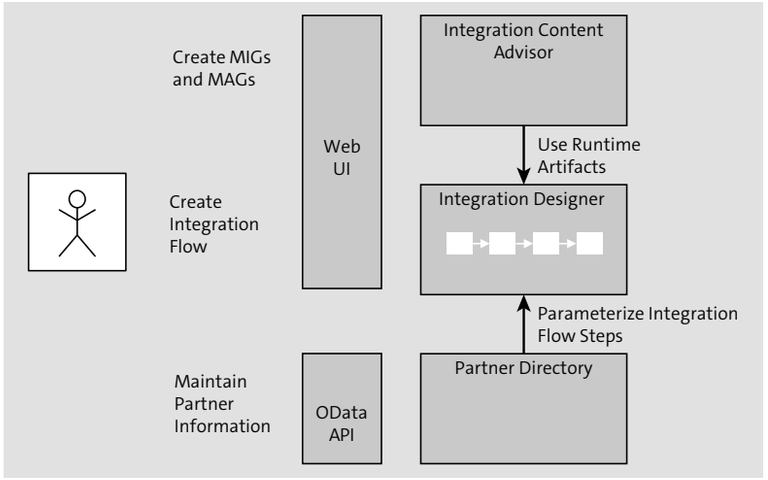


Figure 7.1 Schema of Involved Components in Configuring the B2B Scenario

Let's get started with creating the interfaces and mappings in the ICA.

7.2 Defining Interfaces and Mappings in the Integration Content Advisor

The starting point to set up a new B2B integration scenario is to define the interfaces required by the business partners and to create mappings between those interfaces. Until now, this is one of the biggest challenges for B2B integration projects because it means connecting and managing a potentially large number of business partners with a wide variety of different business requirements. Defining and implementing these interfaces based on the standards for electronic business document exchange usually requires tedious manual effort.

To overcome those efforts, SAP offers the Integration Content Advisor (ICA), a cloud-based design-time solution that accelerates the implementation of B2B scenarios. The ICA unifies all the required tasks for creating integration content based on a comprehensive knowledge base.

The ICA's design time is based on the following main pillars:

- **Library of type systems**
  The ICA includes a set of B2B industry standard libraries containing a collection of messages/message interfaces, associated complex and simple types, and code lists used in the messages. Such a collection is referred to as type system.
- **Message implementation guidelines (MIGs)**
  One main focus of the ICA is assisting in the writing of interface specifications. The specifications provide instructions and constraints for implementing a certain message interface using a B2B standard message provided by the type system in a certain business context. These specifications determine the behavior and the use of each B2B standard message, including limitations or customizations. They contain the definitions of mandatory elements and occurrences, property definitions for each element, permitted code lists and code values, and, finally, the definition of validation constraints and business rules.
- **Mapping guidelines (MAGs)**
  A mapping guideline is the detailed specification of a mapping from a source MIG to a target MIG in a given business context. The focus is on the description of each mapping entity across the corresponding elements, so that business domain experts understand the reason and meaning of the mappings. All technical aspects are implicitly calculated and derived into the technical artifacts, which is the fourth pillar.
- **Automatically generated runtime artifacts**
  The runtime artifacts generated by the ICA are required for preprocessing and postprocessing, conversion, detailed validation, or even the transformation (mapping) from source to target message. The ICA generates a number of artifacts based on XML Schema Definition (XSD) Version 1.0 and Extensible Stylesheet Language Transformation (XSLT) Version 2.0, which can be directly used in a prepared integration flow in the designer of SAP Cloud Platform Integration.

Now that you understand the main parts of the ICA, let's check how interfaces and mappings are created in the ICA.

7.2.1 Create Message Implementation Guidelines

The first task when using the ICA is to create the message interfaces that the involved partners require for the scenario. A message interface is usually defined based on a B2B standard interface.

B2B integration relies on a variety of industry standards, also known as B2B standards, for electronic business document exchange. At the time of publishing this book, the ICA supports the following standards in its type systems:

- **ASC X12 (*www.x12.org*)**
  This standard, maintained by the American National Standards Institute Accredited Standards Committee X12 (ANSI ASC X12), is one of the commonly used EDI standards for electronic data exchange, mainly in the United States. The ICA offers several hundred messages with the corresponding complex types, simple types, and code lists in many versions.
- **UN/EDIFACT (*www.unece.org/cefact/edifact/welcome.html*)**
  This standard is maintained and further developed through the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT). It's widely used in Europe. The ICA offers around 200 messages with the corresponding complex types, simple types, and code lists in many versions (in syntax version S3).
- **UN/CEFACT**
  Additional code lists are offered separately in ICA and are maintained by UN/CEFACT (*www.unece.org/cefact.html)*. Eight additional code lists are available in multiple versions.
- **ISO (International Organization for Standardization) (*www.iso.org*)**
  ISO develops and maintains international standardized code lists and identifier schemas. The ICA offers five code lists in versions 2004, 2012, and 2017.
- **SAP IDoc**
  This is the SAP-owned document format for business transaction data transfers. The ICA offers the IDoc versions of the SAP S/4HANA 1709 release. The most commonly used IDoc messages with the corresponding complex types, simple types, and code lists are offered.

Prerequisites for Using the Integration Content Advisor

To use the ICA, you need to get an ICA application provisioned for your SAP Cloud Platform Integration tenant, and you must assign certain user roles to the users who want to access the application.

The provisioning is done via self-service from the SAP Cloud Platform cockpit as described in the documentation for SAP Cloud Platform Integration (*https://help.sap.com/viewer/product/CLOUD_INTEGRATION/Cloud*) in the topic "Subscribing to Integration Content Advisor From SAP Cloud Platform Cockpit." After the provisioning, you'll find the URL to the ICA application in the **Details** section of the SAP Cloud Platform cockpit.

To call the ICA application URL, you need to assign the user roles `Guidelines.ReadWrite` and `TypeSystem.Read` to the users who want to access the ICA application. This is described in detail in the documentation for SAP Cloud Platform Integration (*https://help.sap.com/viewer/product/CLOUD_INTEGRATION/Cloud*) in the topic "Assigning Users for Integration Content Advisor for SAP Cloud Platform Integration."

To use the non-SAP type systems, such as UN/EDIFACT, you need to purchase additional licenses. As long as the license isn't purchased for a specific type system, it's shown as **Unlicensed** in the ICA.

In our sample scenario, we're going to use the 850 Purchase Order message from the ASC X12 type system as the inbound message and the ORDERS.ORDERS05 IDoc from SAP IDoc type system as the outbound message. We'll take a look at those messages in the ICA and get to know how MIGs and MAGs can be created based on it.

Note

We won't explain step by step how to create the MIGs and the MAGs for the sample scenario as this would fill a book on its own. We'll describe the overall process and the features the ICA offers to create MIGs and MAGs. We'll also explore how to export the runtime artifacts, which are then used in the integration flow configuration. The generated runtime artifacts from ICA needed for the B2B sample scenario are provided with the book downloads at *www.sap-press.com/4650*.

Because the sample scenario uses the 850 Purchase Order message from ASC X12 type system as the inbound message and the ORDERS.ORDERS05 IDoc from SAP IDoc type system as the outbound message, we need the B2B standard message templates from those two type systems as the starting point for creating our own MIG. Afterwards, we tailor the MIG to suit the scenario-specific requirements.

Search for Standard Messages in Type Systems

To create MIGs, you first need to explore the messages provided by the ICA and select the ones you require for the scenario. To do this, execute the following steps:

1. Launch the ICA application from the URL provided in the SAP Cloud Platform cockpit. The ICA entry page opens. As depicted in Figure 7.2, it's divided into different

sections: in the upper **General** section, you have the option to navigate to the **Library of Type Systems**, and in the lower **Own** section, you can check your **Profile** and create your own MIGs and MAGs.

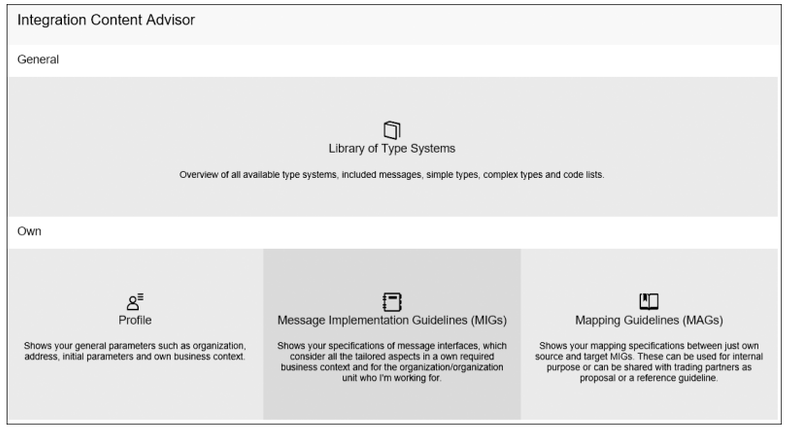


Figure 7.2 ICA Entry Page

2. Select the **Library of Type Systems** link to get the overview of the type systems available in your ICA (Figure 7.3). By default, only the **SAP IDoc** and the **UN/CEFACT** type systems are shown as **Licensed**. All the other non-SAP type systems are available as well but are shown as **Unlicensed** if you haven't purchased them. You'll get an error message when trying to access the details of the unlicensed type systems.

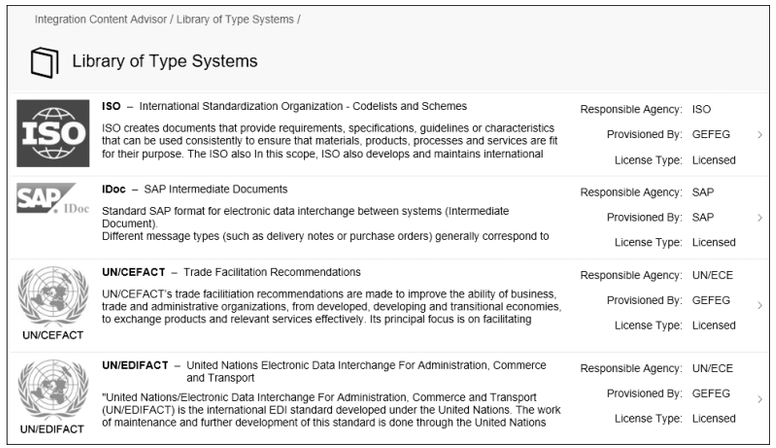


Figure 7.3 Type Systems in ICA

3. Let's first explore the SAP IDoc type system, as our outbound interface will be based on it. Select the **SAP IDoc** type system. A new window opens providing the details, as depicted in Figure 7.4. The **OVERVIEW** tab shows **General Information**, such as the **Responsible Agency** and the **Status**, as well as further **Documentation** about the type system. In addition, creation and modification information is shown in the **Administrative Data** area.

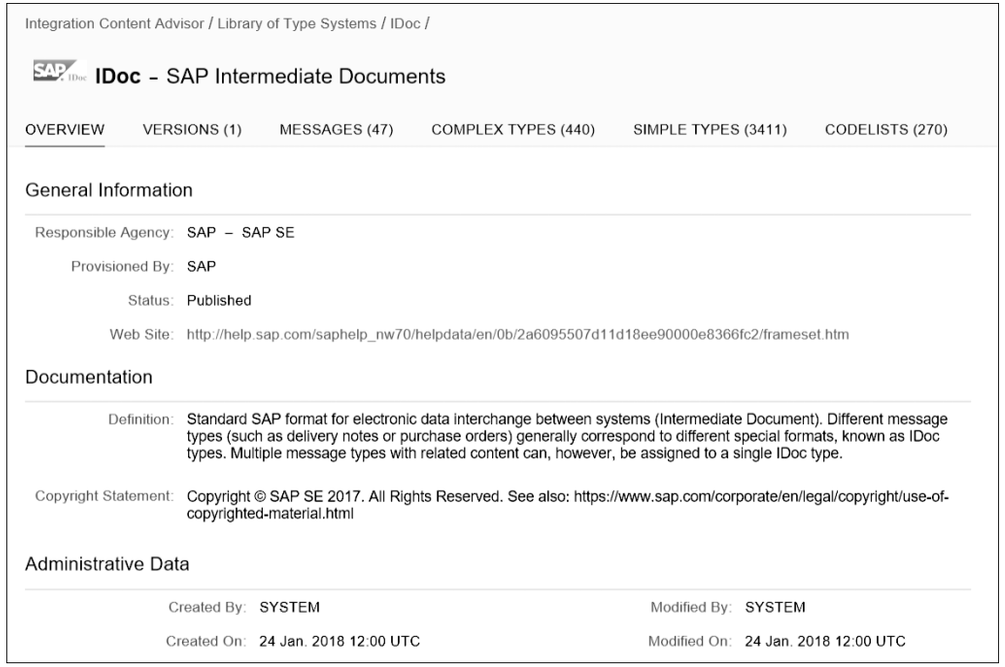


Figure 7.4 SAP IDoc Type System: Overview

4. In the **VERSIONS** tab, all available versions in the SAP IDoc type system are shown (Figure 7.5). As mentioned already, the only version in the SAP IDoc type system available at the time of publishing this book is **S/4HANA Release 1709**.
5. The **MESSAGES** tab lists all available messages provided by this type system in a tree structure. As depicted in Figure 7.6, the **SAP IDoc** type system contains the most commonly used IDoc messages. Opening one of them in the tree structure shows the versions available for this message. As only one version is available in the SAP IDoc type system, only one version is shown for the selected message. In other type systems, several versions appear for one message.

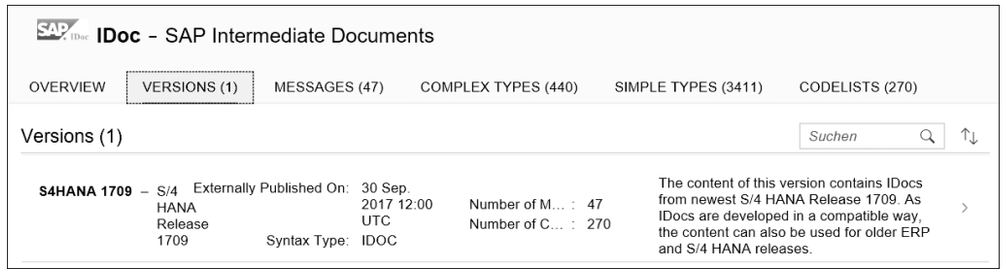


Figure 7.5 Versions for the SAP IDoc Type System

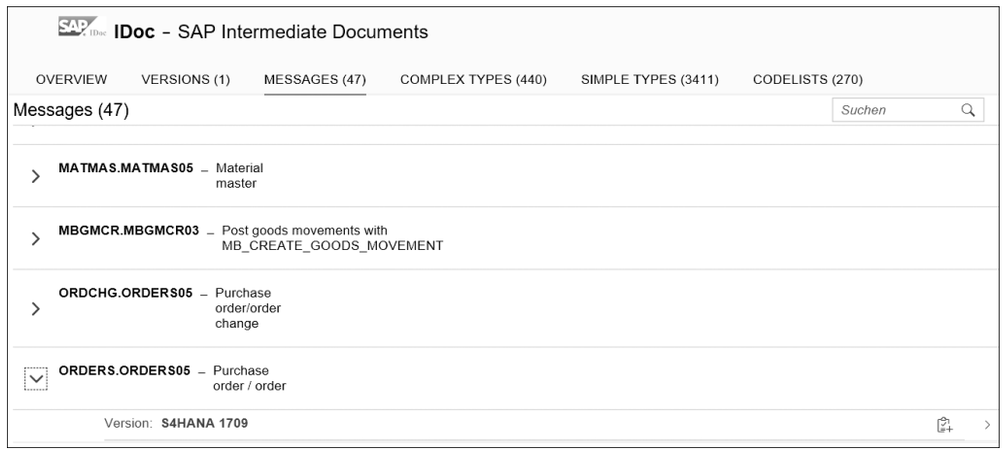


Figure 7.6 Message Interfaces in SAP IDoc Type System

6. When you click the row containing the **Version** information for the selected message, the detailed structure of the message is shown in a new window. In Figure 7.7, you find the structure of the **ORDERS.ORDERS05** message we're going to use in the scenario we want to set up. You can use this view to explore the message's structure.
7. You can drill down into single fields by opening the tree structure. Selecting dedicated fields opens the **Properties** view for the selected field below the tree structure. In Figure 7.8, for example, the details of the field **CURCY** are shown. The **DETAILS** tab provides information such as the **Tag**, **Name**, **Cardinality,** and **Documentation** for the field.

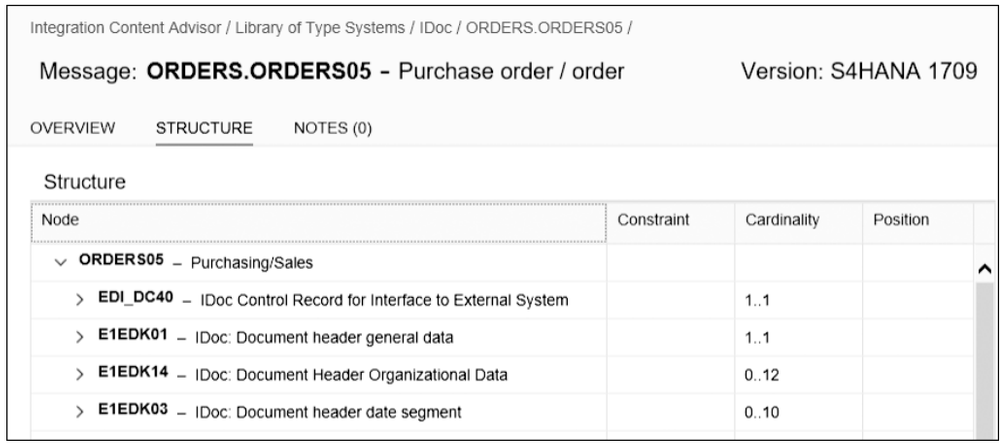


Figure 7.7 Structure of Message ORDERS.ORDERS05

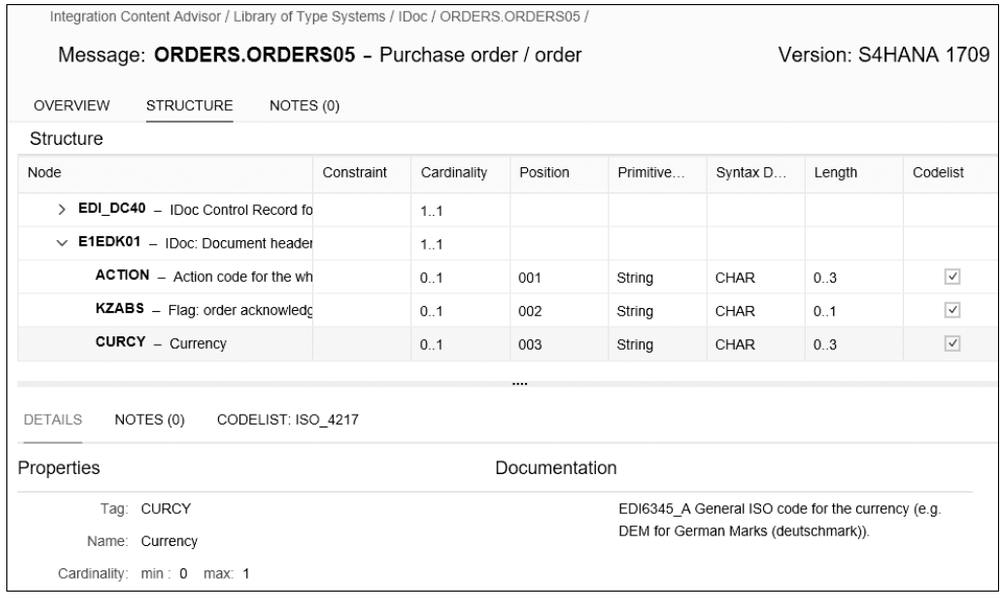


Figure 7.8 Properties of the Field CURCY

8. The **CODELIST** tab is very important because it shows the values of the linked code list. You notice already from the name of the tab that the code list **ISO_4217** is linked, which means that the values from this code list apply for this field. Note

that you can navigate to the **CODELIST** tab for this field only if the ISO type system is licensed because the linked code list is an ISO code list (name: `ISO_xxx`). If a code list from the SAP IDoc type system is linked, or the ISO type system was purchased, you can navigate to the **CODELIST** tab where you find the allowed values in a table (Figure 7.9).

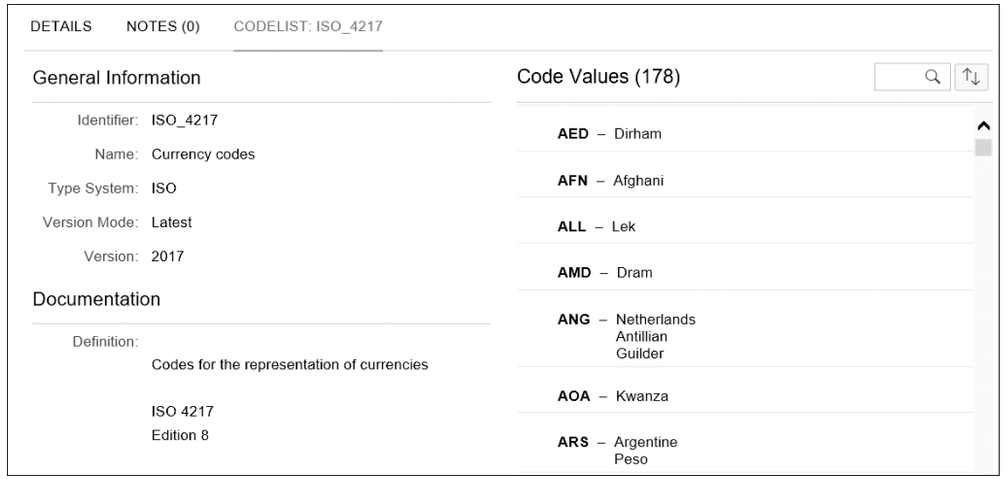


Figure 7.9 Code List Values for Field CURCY

9. In the **COMPLEX TYPES** and **SIMPLE TYPES** tabs, you can navigate to all data types used in the messages in this type system. Note that this is the same information shown when you inspect the properties of a selected field in the tree structure of a specific message. The same holds true for the **CODELISTS** tab: you can navigate to the available code lists either from the type system or when checking the properties of dedicated fields.
10. Get familiar with the type system browser, and navigate to the message **850 - Purchase Order** in version **004010** in the ASC X12 type system. This is the message interface we're going to use for our inbound message.

No License for ASC X12 Messages?

If you don't have a license for the ASC X12 type system to explore its messages but you want to set up the B2B sample scenario, you can download the runtime content—the mappings and XSDs—that the ICA would generate from the book downloads at *www.sap-press.com/4650*. Using this content, you can continue with the

integration flow creation in Section 7.3 and further explore the runtime features SAP Cloud Platform Integration offers for B2B integration.

You're now familiar with browsing messages in the available type systems and know how to explore the structure of single messages. Let's now create the MIGs for the messages.

Create Message Implementation Guidelines

To create MIGs, execute the following steps:

1. To create a MIG based on the ORDERS.ORDERS05 message, open the **MESSAGES** tab in the **SAP IDoc** type system, and expand the message **ORDERS.ORDERS05**. In the line containing the version information select the **Create a New MIG** icon to create a new MIG for this message (Figure 7.10).

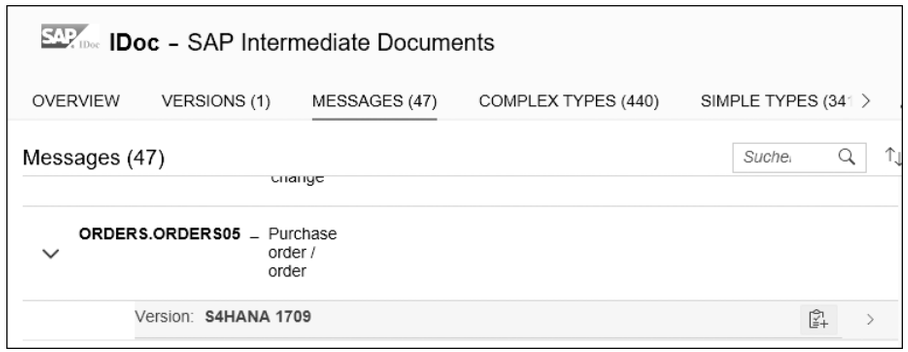


Figure 7.10 Creating a New MIG for ORDERS.ORDERS05

2. On the MIG creation screen, enter a **Name** for your MIG, select the **Direction**, and specify the **Business Context**, as depicted in Figure 7.11 and described here:
   - The **Direction** dropdown list offers three values: **In**, **Out**, and **Both**. You can choose whether the interface is used in inbound or outbound direction or if it can be used in both directions in the context of a B2B transaction. Setting the correct direction will make the proposal service more precise. We choose **Out** as the **Direction** for our MIG because we want to use this message interface as the target interface.
   - During creation of a new MIG, the **Version** is set to **1.0** with the **Status** as **Draft**.

- The fields **Message Type**, **Type System**, and **Type System Version** are prefilled based on the selected template message and can't be changed.
- You can use the **Documentation** field to add a description for the MIG. This text is visible as short text in the MIG overview list.
- In the **Business Context** field, you specify the business context in which the interface will be used. Use the + icon to add a business context. Note that you can define multiple entries. Based on the business context, you'll be provided with further options in the dropdown list. For example, add the business context **Business Process** with the value **Create Order**. The defined business context is used by the proposal service to provide optimal proposals.

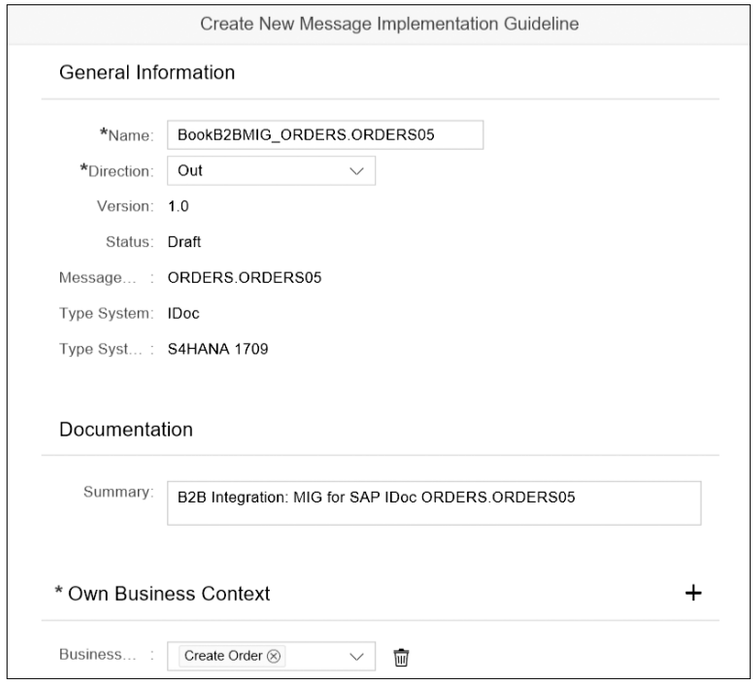


Figure 7.11 Creating the MIG for ORDERS.ORDERS05

3. Click **Create,** and then the MIG editor opens.
4. Choose **Edit** in the upper-right corner to switch to edit mode. In the **STRUCTURE** tab, the whole message structure of the used template message is shown (Figure 7.12). In this view, you select the fields you need for your scenario. By default, all

mandatory fields are preselected. In addition to the preselected fields, you can select additional fields required for your scenario.

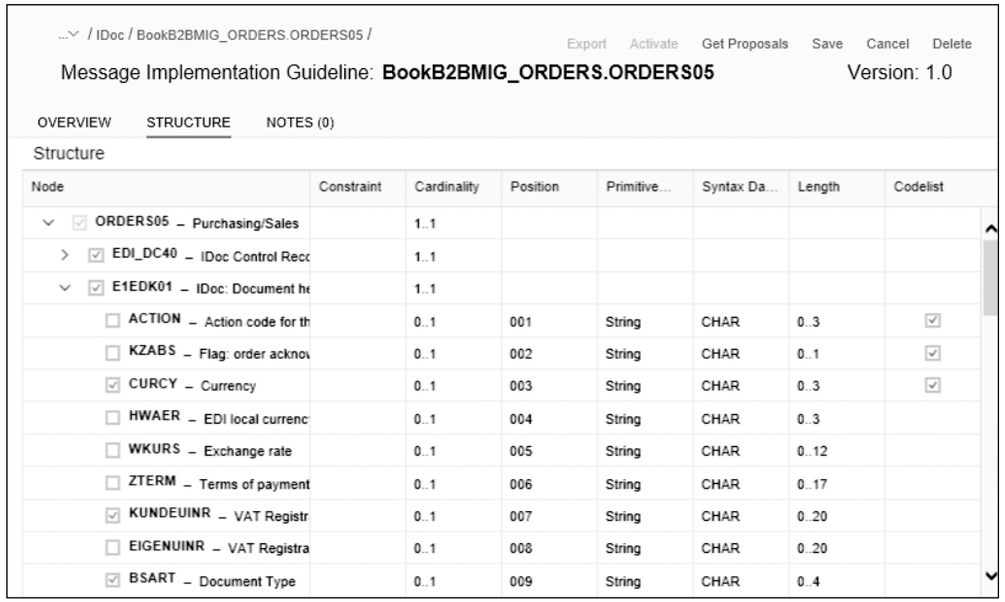


Figure 7.12 Structure of the MIG for ORDERS.ORDERS05

5. Using the **Get Proposals** button on the top-right corner of the screen, you activate a proposal indicator that displays which fields might be most relevant for you. This is calculated based on the other available MIGs. Note that this is just a proposal that can help you define the MIG more quickly, but you don't have to accept the fields proposed.
6. Using the context menu option **Qualify Node**, you can qualify a node by specifying a qualifier marker and qualifier value. The qualifier is then shown in the structure as an arrow with the defined qualifier value (see Figure 7.13). Using qualifiers helps to simplify the MAG creation in the next step.
7. For fields that have fixed values in your scenario, you can a define **Fixed Value** in the **Properties** section (Figure 7.14). Those values will be mapped automatically later in the MAG configuration.

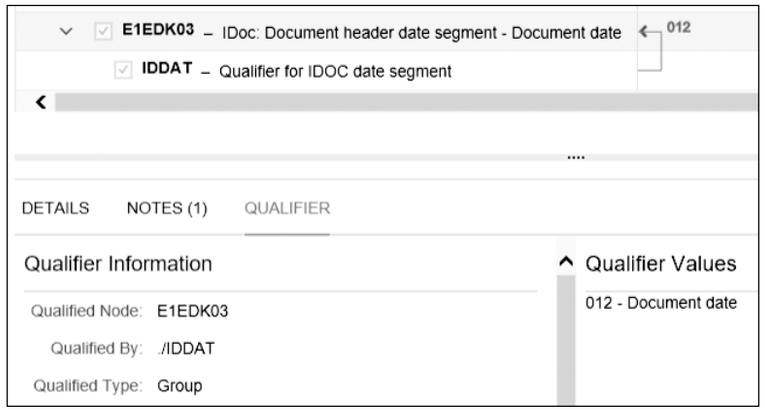


Figure 7.13 Qualified Field in the MIG

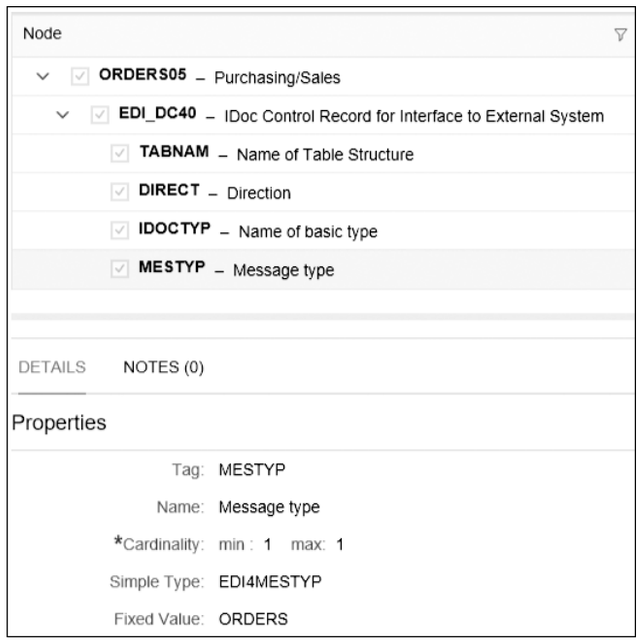


Figure 7.14 Defining a Fixed Value for the Message Type

8. In the **Properties** section, you can also change further field properties (e.g., the cardinality and the length of fields), define example values, and select code values from the linked code lists.

9. After you've selected the required fields, save the MIG using the **Save** button in the upper-right corner of the screen. Note that the activation of the MIG using the **Activate** button should be done only after creating and testing the MAG because this will save the MIG as the main version. Then no more changes are allowed to this version to protect it against unwanted changes; a new version would have to be created instead.
10. After creating a MIG, you can export the runtime artifacts using the **Export** button in the upper-right corner of the screen. A **.zip* file is created in your local download folder containing the **.xsd* and **.xsl* descriptions for the MIG. Those files can be imported into integration flows for the sake of performing validations and transformations.
11. Following the same procedure, you can create a MIG for the inbound interface based on the message 850 - Purchase Order with version 004010 from the ASC X12 type system. Select direction **In** when creating the MIG, and choose the same business context used for the ORDERS.ORDERS05 message: **Business Process** with the value *Create Order*. Note that it isn't mandatory to define the MIG to continue with the sample scenario because the generated runtime artifacts are provided in the book downloads as mentioned before.
12. Keep the mandatory fields and also define additional fields you require for the scenario. Use the proposal service as described before, and define qualifiers and constants as required.
13. You can save the MIG and export the runtime artifacts as described before.

Now that the MIGs are created, you can create a MAG based on them.

7.2.2 Configure Mapping Guidelines

After defining the MIGs for the source and target message interface, a mapping between the fields of those interfaces needs to be created. This step is executed in the MAG editor.

To create the MAG for the scenario, execute the following steps:

1. Start the MAG editor from the ICA entry screen (Figure 7.2). Select the **Mapping Guidelines** section in the lower section on the right side of the screen. A table containing all existing MAGs opens. In your case, the table may still be empty because no MAGs are created yet.

2. At the top of the table, select the **Create a New MAG** icon to create a new MAG. The MAG creation wizard opens.
3. In the first screen of the wizard, select the source MIG (Figure 7.15). Choose the MIG created for the **850 Purchase Order** message, and select **Next** at the bottom of the screen.

Select Source MIG

Source: BookB2BMIG_PurchaseOrder Target: ... Suchen

Name	Type System	Version	Message	Direction	Version
#Demo - ConElChi - Purchase Order MIG (Do not delete)	ASC_X12	004010	850	In	1.0
#Demo - HiTechPro - Purchase Order MIG (Do not delete)	SAP_IDoc	S4HANA 1...	ORDERS.ORDERS05	Out	1.0
(Backup) HE4CLNT400 - Source MIG - Purchase Order	ASC_X12	004010	850	In	1.0
BookB2BMIG_PurchaseOrder	ASC_X12	004010	850	In	1.0

Figure 7.15 Selecting the Source MIG

4. In the **Select Target MIG** screen, select the MIG you created based on **ORDERS.ORDERS05** (Figure 7.16), and click **Create**.

Select Target MIG

Source: BookB2BMIG_PurchaseOrder Target: BookB2BMIG_ORDERS.ORDERS05 Sucher

Name	Type System	Version	Message	Direction	Version	Status
#Demo - ConElChi - Purchase Order MIG (Do not delete)	ASC_X12	004010	850	In	1.0	Draft
#Demo - HiTechPro - Purchase Order MIG (Do not delete)	SAP_IDoc	S4HANA 1...	ORDERS.ORDERS05	Out	1.0	Draft
(Backup) HE4CLNT400 - Source MIG - Purchase Order	ASC_X12	004010	850	In	1.0	Draft
BookB2BMIG_ORDERS.ORDERS05	SAP_IDoc	S4HANA 1...	ORDERS.ORDERS05	Out	1.0	Draft

Figure 7.16 Selecting the Target MIG

5. The MAG editor opens. As depicted in Figure 7.17, the **OVERVIEW** tab contains information about the MAG and the selected source and target MIGs.

... / Mapping Guidelines / Mapping BookB2BMIG_PurchaseOrder to BookB2BMIG_ORDERS.ORDERS05 /

Export Activate Edit Delete Copy

Mapping Guidelines: **Mapping BookB2BMIG_PurchaseOrder to BookB2BMIG_ORDERS.ORDERS05** Version: 1.0

OVERVIEW MAPPING

General Information

Name: Mapping BookB2BMIG_PurchaseOrder to BookB2BMIG_ORDERS.ORDERS05
Version: 1.0
Status: Draft

Source and Target MIGs

Source		Target	
MIG:	BookB2BMIG_PurchaseOrder	MIG:	BookB2BMIG_ORDERS.ORDERS05
Version (Status):	1.0 (Draft)	Version (Status):	1.0 (Draft)
Message Type:	Purchase Order	Message Type:	Purchase order / order
Type System:	ASC X12	Type System:	IDoc
Type System Version:	004010	Type System Version:	S4HANA 1709

Documentation

Summary: Enter summary here...
Definition: Enter definition here...

Source Business Context

Business Process: Create Order

Target Business Context

Business Process: Create Order

Administrative Data

UUID: 695692bea4734130b85f824ab0eef19b URL: /shell/ica/mags/undefined/magVersions/695692bea47341...
Created By: D 9 Modified By: D 9
Created On: 18 Apr. 2018 03:11 UTC Modified On: 18 Apr. 2018 03:11 UTC

Figure 7.17 Overview Tab for Mapping Guidelines

6. Select the **MAPPING** tab to open the mapping editor. In this view, you map the fields from the source MIG to the target MIG. As depicted in Figure 7.18, draw a line from **Source** field to **Target** field to create the mapping between the fields. In the **FUNCTION** tab, you can define functions for the mapping of specific fields, and the defined code snippets are then generated into the XSLT mapping. You can also use the proposal service by selecting **Get Proposal** to get some mappings proposed in the table. We won't describe how to map all fields step by step because this would fill pages. Instead, we point you to the downloadable XSL transformation file provided with the book downloads (*www.sap-press.com/4650*).

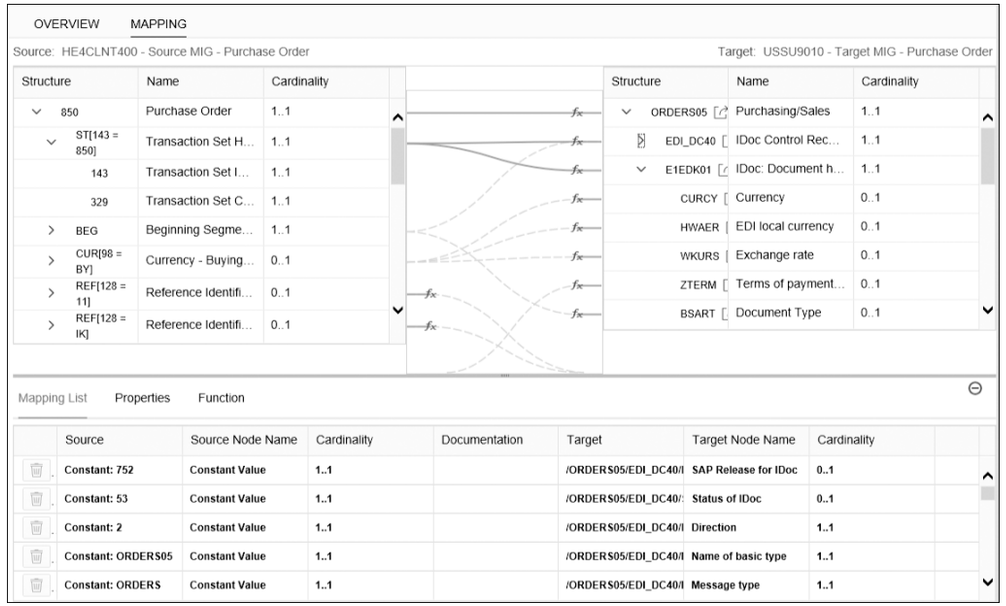


Figure 7.18 Creating Mapping between Source and Target MIGs

7. After finishing the mapping, save the MAG. Activate it after successful testing to protect it against unwanted changes.

Most Recent Features in Integration Content Advisor

To find the newest features in ICA, check the blogs in the SAP Community (*www.sap.com/community.html*), including "Integration Content Advisor: Discover B2B/A2A Standard libraries."

After defining the desired interfaces and mappings in the ICA, you can generate the runtime artifacts. You can use these runtime artifacts in the integration content, for example, in a mapping step of an integration flow.

7.2.3 Generate the Runtime Content

To use the mapping and the message interface definitions at runtime, you need to export the runtime artifacts from ICA.

To export the runtime artifacts, use the **Export** button in the upper-right corner of the MAG editor screen. A*.zip* file is created in your local download folder containing the **.xsd* and **.xsl* descriptions of the source and target MIG and an **.xsl* file containing the mapping between the source and target MIG.

Those artifacts are required in the integration flow configuration. Note that the runtime artifacts to be used in the sample scenario are provided with the book downloads at *www.sap-press.com/4650*. Download the runtime artifacts from the book download page, and continue with creating the integration flow based on a template.

7.3 Configure a B2B Scenario with AS2 Sender and IDoc Receiver Adapters

After defining the message interfaces for a B2B scenario and creating the mappings, the usual task for a content developer is to create the integration flow that handles the processing of the messages for this scenario.

In this section, we'll create the integration flow and use the generated message definitions and mappings within its processing steps. We create a sample scenario with the following processing steps:

1. An ASC X12 Purchase Orders message is received by the AS2 sender adapter.
2. The message content is validated.
3. A 997 acknowledgement is sent back to the sender.
4. After successful validation, the message is transformed into a SAP IDoc ORDERS.ORDERS05 message.
5. At the end of the processing, the IDoc message is send via the IDoc adapter to a receiver backend.

Let's get started.

7.3.1 Create an Integration Flow Using a Template

At first, we need to create the integration flow. To enable easy configuration, SAP provides predefined templates for the different B2B integration patterns. Those templates are offered in the Integration Content Catalog in the EDI Integration Templates for Integration Content Advisor package. In this package, you find all the predefined templates published by SAP for setting up B2B scenarios.

Integration Templates

At the time of publishing this book, not all the integration templates are final yet, and changes to the templates are still expected. Because of this we provide the template *EDI_IDoc_Template.zip* used for the sample scenario in the book downloads at *www.sap-press.com/4650*. Please download this template to set up the sample scenario.

Create a new integration flow in the integration designer perspective. In the creation wizard, select **Upload**, select the template *EDI_IDoc_Template.zip* file you downloaded from the book downloads, and enter a **Name**, as shown in Figure 7.19. Select **OK** to create the integration flow.

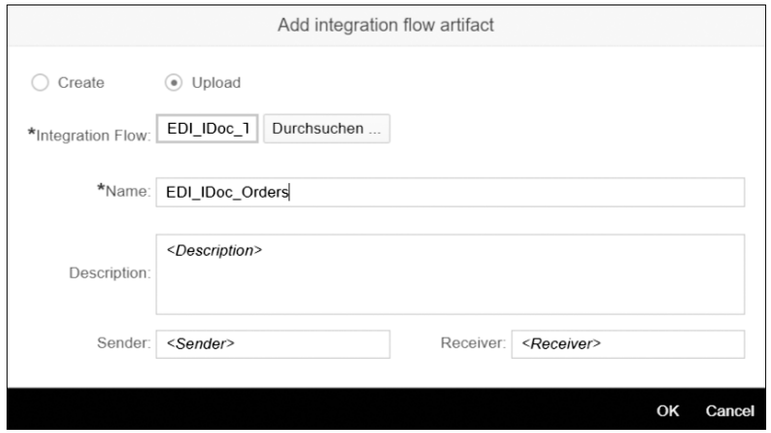


Figure 7.19 Creating an Integration Flow from a Template

The created integration flow looks like the one in Figure 7.20. There are lots of flow steps configured, and several of them have error markers because the flow isn't configured yet; we haven't yet added the generated runtime content from ICA.

In the next sections, we'll explore all of these steps and how to configure them to get the scenario running. Let's start from left to right as this is also the message processing order.

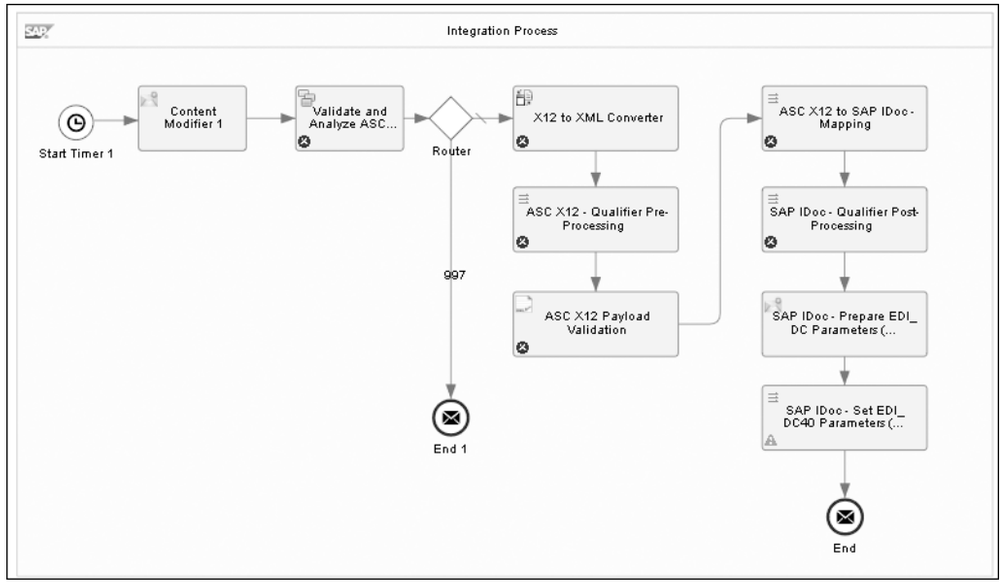


Figure 7.20 Created Integration Flow Based on a Template

Trigger Message Using Timer Start Event

In the imported template, the first two steps are a **Start Timer** event configured to run once at deployment of the integration flow and a **Content Modifier** that sets an inbound payload. Those steps were explicitly added to the original SAP template for the sample scenario to make the scenario configuration easier for you. Usually, the SAP template starts with a Message Start event without a sender adapter because the payload could be received by different adapters, such as SOAP or AS2, and continues directly with the EDI splitter.

With the **Start Timer** event, it's easy to trigger the first test message without having to configure a sender. Later, in Section 7.3.2, we'll change this configuration when we configure the AS2 sender adapter to receive messages for this scenario.

In the **Content Modifier** step, a sample payload is set in the **Message Body** tab. This is the same sample payload as provided in the *850 - Purchase Order.txt* file provided in the book downloads at *www.sap-press.com/4650*.

Validate EDI Messages Using EDI Splitter

In the next step of the processing, **Validate and Analyze ASC X12 Interchange**, the incoming EDI message is validated. We need to define based on which XSD a technical validation of the inbound message will be done. The EDI splitter is used for this as it can split and validate UN/EDIFACT and ASC X12 EDI messages based on configured XSD schemas. You need to use the generated XSD schema from ICA for the X12 Purchase Order definition. Use the generated content provided in the book downloads at *www.sap-press.com/4650*.

As our inbound message is an X12 message, open the **X12** tab in the flow step and configure it as depicted in Figure 7.21. Select **ISO-8859-1** as **Source Encoding**, and define that a **Standard Validation** of the message will be executed. Click on the **Add** button to add the schema `ASC-X12_850_004010.xsd` to do the technical validation based on the XSD for the standard X12 EDI message. Use the **Upload from File System** option in the selection dialog to add the XSD to the integration flow.

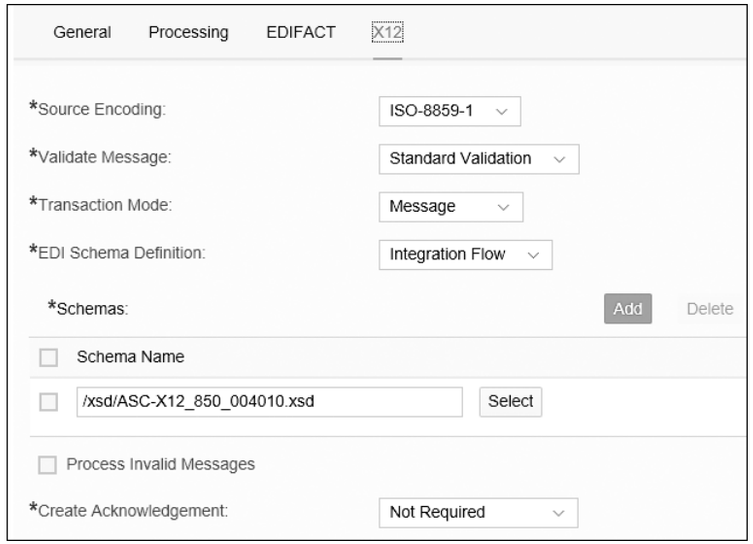


Figure 7.21 Configuring the EDI Splitter

Note that for now, we set **Create Acknowledgement** to **Not Required**. The acknowledgement handling will be configured in Section 7.3.4.

We leave the **Router** and **End Message** event after the EDI splitter as they are for now; they are used when we configure acknowledgement handling in Section 7.3.4.

Detailed Documentation of the Used Integration Flow Steps

For the sake of simplicity, we won't explain all configuration options of the integration flow steps in detail. Refer to the "Define EDI Splitter," "Define EDI to XML Converter," "Validating Message Payload against XML Schema," and "Create XSLT Mapping" sections in the documentation for SAP Cloud Platform Integration at *https://help.sap.com/viewer/product/CLOUD_INTEGRATION/Cloud*.

Convert EDI Messages to XML Format Using the EDI to XML Converter

In the **X12 to XML converter** step in the template, the conversion of the EDI message to XML is executed using the EDI to XML converter, which can convert UN/EDIFACT and ASC X12 EDI messages. As our inbound message is an X12 message, open the **X12** tab, and configure it as shown earlier in Figure 7.22.

Define **ISO-8859-1** as **Source Encoding**, and add the schema `ASC-X12_850_004010.xsd` to do the conversion to XML based on the MIG created for the ASC X12 source message. Use the XSD already uploaded in the last step.

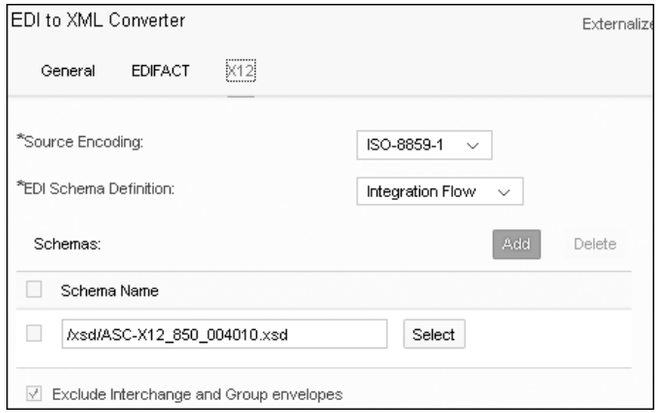


Figure 7.22 Configuring the EDI to XML Converter

After this step, the EDI message is available as an XML representation in the runtime, so that additional conversions, validations, and mapping to the target structure can be done.

Configure ASC X12 Qualifier Preprocessing

In the **ASC X12 Qualifier Pre-processing** step, qualifier suffixes are added to the XML based on the MIG definition from ICA in order to perform a content validation of the message in the next step. The preprocessing is executed via the XSLT mapping generated by the ICA.

In the **PROCESSING** tab, select the mapping `ASC-X12_004010_850_preproc.xsl` generated from ICA and provided in the book downloads (Figure 7.23). Use the **Upload from File System** option in the selection dialog to add the mapping to the integration flow.

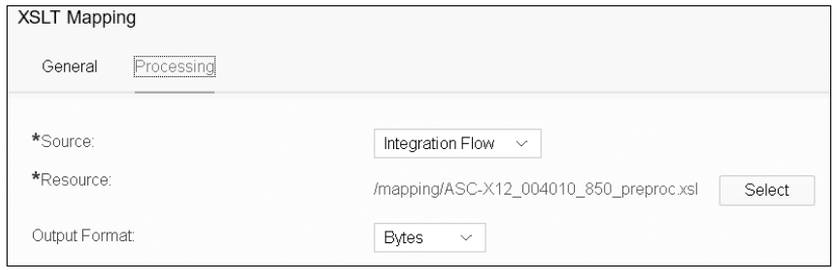


Figure 7.23 Configuring X12 Preprocessing

After this step, the real payload validation can be executed based on the defined qualifiers and qualifier values.

Configure XML Validator

In the **ASC X12 Payload Validation** step, the payload validation of the inbound message is done using the XML validator. The validation is done against the "Russian doll" (RD) XSD generated from ICA for the source MIG. (RD style means that the XSD schema structure mirrors the XML document structure.) For the content validation, this XSD is required because it contains the constraints defined in the MIG and provides a high-precision validation of each segment of the payload supporting qualifiers and code lists.

In the **Validation** tab of the XML validator step, select the `ASC-X12_850_004010_RD.xsd` representing the schema of the ASC X12 850 Purchase Order in RD format, as depicted in Figure 7.24. Use the **Upload from File System** option in the selection dialog to add the XSD to the integration flow.

If the validation isn't successful during runtime, an error is raised. If the validation passes successfully, the XML is transformed to the IDoc XML format in the next step.

Figure 7.24 Configuring the XML Validator

Configure Mapping from EDI to IDoc Format

The **ASC X12 to SAP IDoc – Mapping** step converts the X12 message into SAP IDoc format using the XSLT mapping generated by the ICA.

To configure this, in the **Processing** tab of the XSLT mapping step (Figure 7.25), select the `ASC_X12_to_SAP_IDoc_Purchase_Order_Mapping.xsl` file generated by the ICA and provided in the book downloads. Use the **Upload from File System** option in the selection dialog to add the mapping to the integration flow.

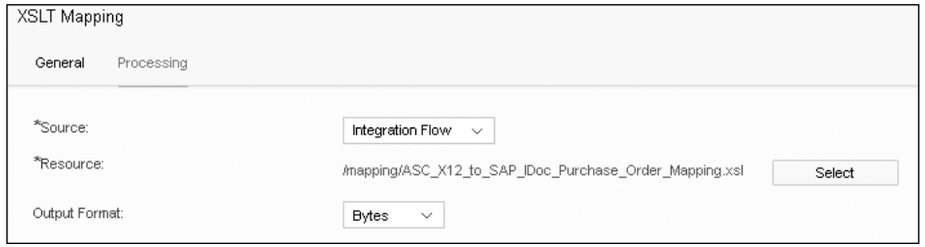


Figure 7.25 Configuring XSLT Mapping

After the transformation to the IDoc XML format, the postprocessing steps for the IDoc format have to be executed.

Configure IDoc Postprocessing

For the IDoc postprocessing, first the qualifier suffixes are removed because they aren't required in the final IDoc payload. Then, the IDoc control record EDI_DC40 needs to be defined.

To configure the IDoc qualifier postprocessing in the **SAP IDoc – Qualifier Post-Processing** step, in the **Processing** tab, select the XSLT mapping `SAP_IDoc_ORDERS05_`

S4HANA 1709_PostProc.xsl file generated by the ICA and provided in the book downloads (Figure 7.26). Use the **Upload from File System** option in the selection dialog to add the mapping to the integration flow.

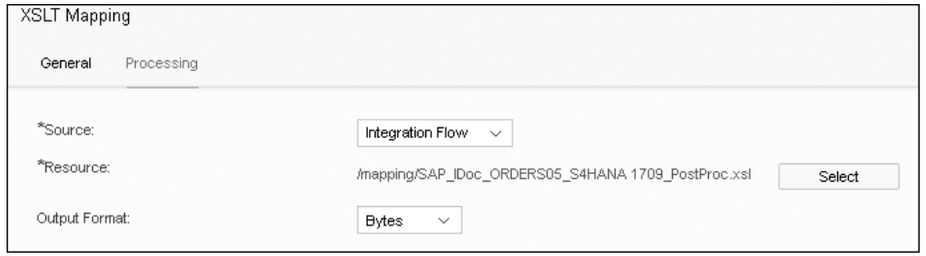


Figure 7.26 Configuring the IDoc Postprocessing

To configure the IDoc control record, a set of properties is defined in the **Content Modifier** step named **SAP IDoc - Prepare EDI_DC Parameters**. In the **Properties** tab, you can define the values that will be generated into the EDI_DC IDoc control record. Those values are required for the IDoc configuration in the receiver system (see also Section 7.3.3). Most of the properties are filled automatically from the source payload using an XPath expression; some are set to constants, and some are filled from headers. For the sample scenario, you may keep the default values or adjust them, if required. The following values are of special interest because they define the processing in the receiver system:

- **SAP_IDoc_EDIDC_SNDPOR**
  Port of the sender; if you don't change it, the value **SAPABC** is used.
- **SAP_IDoc_EDIDC_SNDPRT**
  Partner type of the sender; if you don't change it, the value **LS** is used.
- **SAP_IDoc_EDIDC_SNDPRN**
  Partner number of the sender; if you don't change it, the value **myAS2ID** from the inbound request is used.
- **SAP_IDoc_EDIDC_RCVPOR**
  Port of the receiver; if you don't change it, the value **SAPABC** is used.
- **SAP_IDoc_EDIDC_RCVPRT**
  Partner type of the receiver; if you don't change it, the value **LI** is used.
- **SAP_IDoc_EDIDC_RCVPRN**
  Partner number of the receiver; if you don't change it, the value **USSU9010** from the inbound request is used.

After defining the values for the IDoc control record, those values have to be taken over into the EDI_DC control record. This is done in the XSLT mapping named **SAP IDoc - Set EDI_DC40 Parameters**, which is using a predefined XSLT mapping to insert the defined properties into the IDoc payload.

With this last processing step in the integration flow, you've completed the configuration of the validations and mappings of the EDI message. Now we can configure the receiver of the message.

Send the Message Using Mail Receiver Adapter

In the template, no receiver channel is configured because the message could be sent out using different adapters. The usual one for an IDoc message is the IDoc adapter, but for our first sample execution, let's use the **Mail** adapter. With this, you can easily send the first test message to your mail account to check how the validations and mappings are executed. Configure the **Mail** adapter as described in Chapter 5, Section 5.3.2. Configure the message body as attachment, so that the IDoc message is sent as an attachment in the email (Figure 7.27).

Figure 7.27 Mail Receiver Adapter's Attachment Configuration

With this configuration, you'll receive the mapped IDoc ORDERS.ORDERS05 message as email attachment in your mail account.

Test the Integration Flow

After you've configured all the steps and adapters, save the integration flow, and deploy it on the SAP Cloud Platform Integration tenant. In monitoring, check that the integration flow started and the message was processed successfully. In your mail account, you should have received an email with the mapped IDoc message.

Now that you've successfully executed the scenario with all its validation and mapping steps, you can configure a real sender adapter, enabling the ASC X12 850 Purchase Order message to be sent to the integration flow from a sender system.

7.3.2 Configure AS2 Sender Channel to Receive EDI Messages

As in a real-life B2B scenario, messages are sent from a sender system to the SAP Cloud Platform Integration tenant, so we now need to replace the timer start event with a real sender configuration. In our sample scenario, we'll use the **AS2** sender adapter to receive the 850 Purchase Order messages via the AS2 protocol.

Open the integration flow in edit mode, and delete the timer **Start** event and the **Content Modifier** that defined the sample payload. Add a **Start** message event and a **Sender** participant from the palette. Connect the **Start** message event with the first flow step in the integration flow and the **Sender** participant with the **Start** message event (Figure 7.28). In the adapter selection screen, select the **AS2** adapter with message protocol **AS2**.

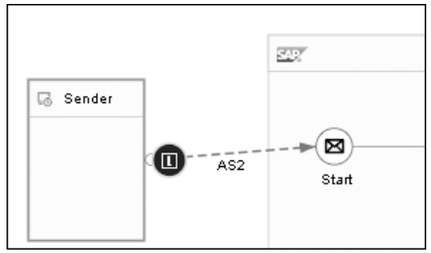


Figure 7.28 Integration Flow with AS2 Adapter

Configure the **AS2** sender adapter's **Processing** tab, as shown in Figure 7.29.

> **Detailed Documentation of the AS2 Sender Adapter**
>
> Note, that for the sake of simplicity, we won't describe all possible configuration options in the **AS2** sender adapter in detail in the book. You can refer to the "Configure Communication Channel with AS2 Adapter" section in the documentation for SAP Cloud Platform Integration at *https://help.sap.com/viewer/product/CLOUD_INTEGRATION/Cloud*.

The most important settings are the configurations for the **Expected Messages**: **Message ID Left Part**, **Message ID Right Part**, **PARTNER AS2 ID**, **Own AS2 ID**, and **Message Subject** because those parameters define the expected inbound message. The combination of the parameters must be unique across all the integration flows deployed on the tenant. You'll need the defined settings when setting up the sender simulation tool in the next section.

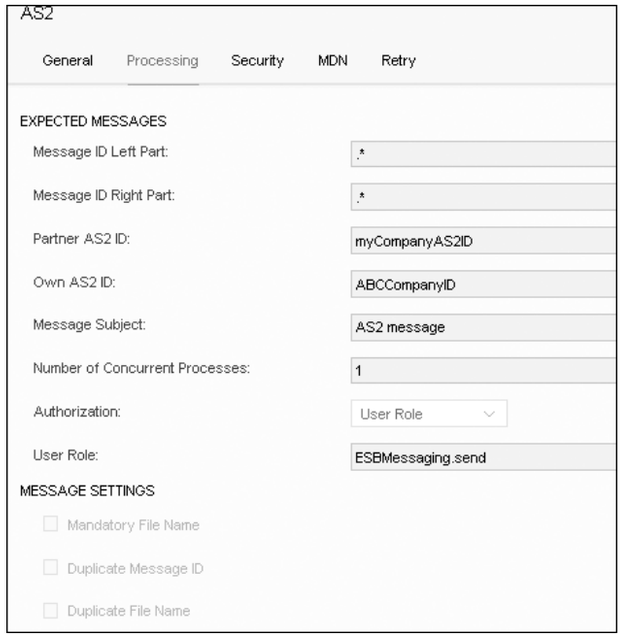


Figure 7.29 AS2 Sender Adapter Configuration

> **AS2 Sender Adapter Uses JMS Queues for Storage**
>
> Because the **AS2** sender adapter is using JMS queues to temporarily store messages for retry in error cases, you need to get a JMS Message Broker provisioned to successfully use it in an integration flow. If no broker is provisioned, the integration flow using the **AS2** sender adapter won't start, and an error stating that no broker is available will appear in the **Manage Integration Content** monitor.
>
> Check the "Provision Message Broker" blog in the SAP Community (*www.sap.com/community.html*) about the provisioning of a JMS Message broker for your tenant.

The configurations in the **Security**, **MDN**, and **Retry** tabs can be left with the default settings. For our simple sample scenario, we don't use signing, encryption, or asynchronous message disposition notification (MDN). A synchronous MDN is sent back to the receiver as a receipt to acknowledge that the message was successfully received. A retry will be executed every minute if there is an error. Check out the documentation for SAP Cloud Platform Integration to get more details about the configuration options for those features.

Sample Scenario with Signing, Encryption, and Asynchronous MDN

If you want to extend the simple sample scenario by using signing, encryption, and asynchronous MDN, refer to the detailed "B2B Capabilities in SAP Cloud Platform Integration – Part 2" blog in the SAP Community (*www.sap.com/community.html*).

Save and deploy the integration flow. The integration flow is now ready to be called from the sender system.

Get the Endpoint URL

To call this integration flow, the sender needs to know the URL where to send the message to. The endpoint URL can be retrieved the same way as we've explained in several places in the book for the SOAP adapter. Open the **Manage Integration Content** monitor, and select the deployed integration flow. The endpoint URL is shown in the **Endpoints** section in the details screen on the right (Figure 7.30 and Figure 7.31) and has the structure *https://<runtime node>/as2/as2*.

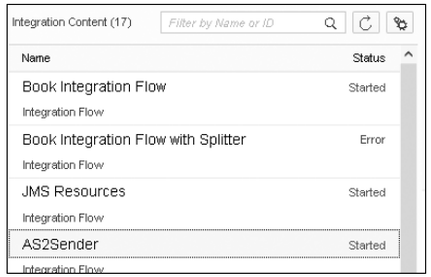


Figure 7.30 Endpoint of the AS2 Sender Adapter A

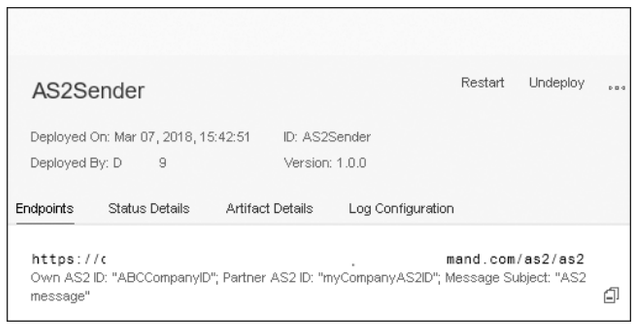


Figure 7.31 Endpoint of the AS2 Sender Adapter B

With this change, the sender needs to trigger the scenario execution by sending a message to the endpoint. For that, you need to configure a sender backend or application to send a message via the AS2 protocol to the **AS2** sender adapter.

Configure the Mendelson Tool to Send AS2 Test Messages

In our sample scenario, we'll send the test message from the open-source Mendelson AS2 tool (*http://as2.mendelson-e-c.com*). Mendelson AS2 can be used to simulate AS2 partners sending test messages via AS2 to the SAP Cloud Platform Integration tenant.

Install and configure Mendelson AS2 as described in the "B2B Capabilities in SAP Cloud Platform Integration – Part 1" blog in the SAP Community (*www.sap.com/community.html*). Make sure the following configured values are matching (fields are case-sensitive):

- The defined **AS2 id** for the local Mendelson AS2 partner configuration (Figure 7.32) needs to match with the **Partner AS2 ID** in the **AS2** sender channel (refer to Figure 7.29).

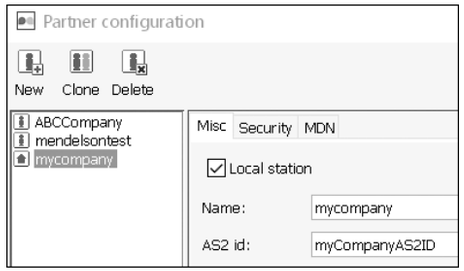


Figure 7.32 Own Partner Configuration in Mendelson

- The defined **AS2 id** for the AS2 partner created for the SAP Cloud Platform Integration tenant (Figure 7.33) needs to match the **Own AS2 ID** in the **AS2** sender channel (refer to Figure 7.29).
- In the **Send** tab of the configured AS2 partner, enter the endpoint URL retrieved from the **Endpoints** section in the **Manage Integration Content** monitor (Figure 7.34).
- The defined **Payload Subject** in the **Send** tab of the configured AS2 partner that was created for the SAP Cloud Platform Integration tenant (Figure 7.34) needs to match the **Message Subject** configured in the **AS2** sender channel (refer to Figure 7.29).
- In the **MDN** tab, select **Request sync MDN**, as shown in Figure 7.35.

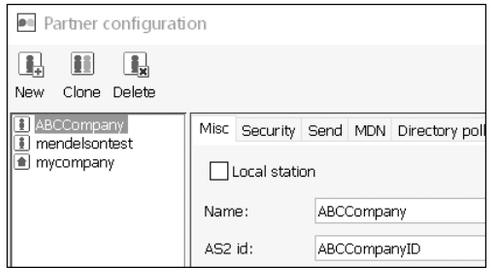


Figure 7.33 Partner Configuration for the SAP Cloud Platform Integration Tenant in Mendelson

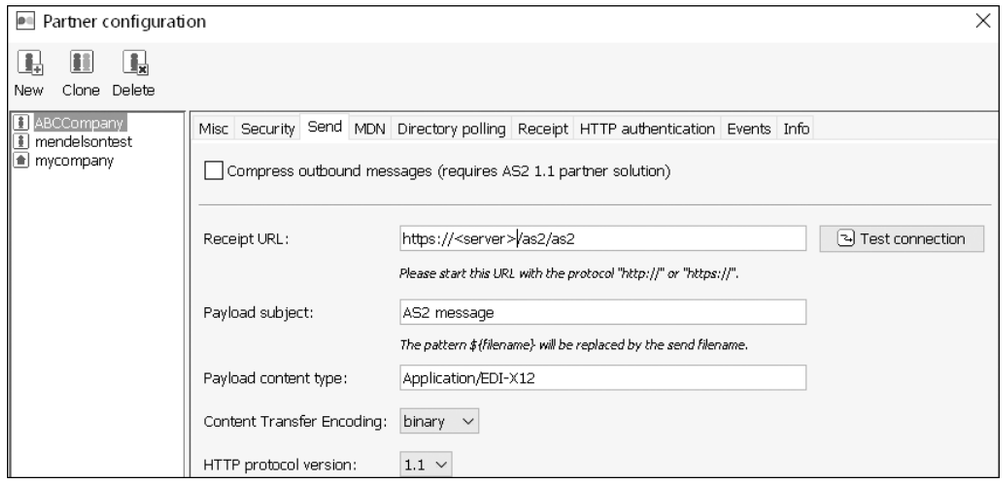


Figure 7.34 Configuring Endpoint and Payload Subject in Mendelson

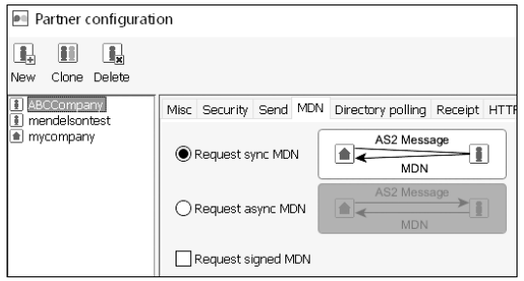


Figure 7.35 MDN Configuration in Mendelson

- In the **HTTP Authentication** tab, select **Use HTTP Authentication to Send AS2 Messages**. Furthermore, enter the **Username** and **Password** of the SAP Cloud Platform Integration user you want to use to log in to the SAP Cloud Platform Integration tenant.
- Import the SSL certificate from SAP Cloud Platform Integration tenant's keystore monitor as described in the blog *B2B Capabilities in SAP Cloud Platform Integration – Part 1* in the SAP Community (*https://www.sap.com/community.html*) to establish the HTTPS connection.

Now that you've set up and configured the AS2 partner, you can use the **Test Connection** option in the **Send** tab to test the connection to the SAP Cloud Platform Integration tenant. The test should pass successfully, and then you can continue with sending a real message to the integration flow.

Trigger a Test Message

To trigger a message from the Mendelson AS2 tool use the sample message *850 - Purchase Order.txt* provided in the book downloads at *www.sap-press.com/4650*. Download the message, and save it to your local workstation.

Trigger sending the message in Mendelson AS2 by choosing **File • Send File to Partner**. Use the configured AS2 partner, and select the downloaded sample message, as shown in Figure 7.36.

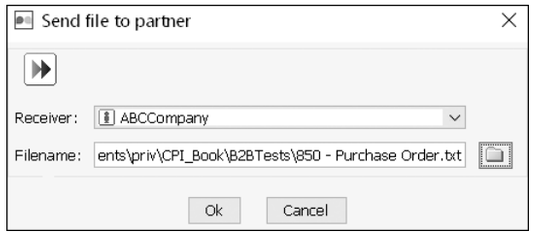


Figure 7.36 Sending a Test Message in Mendelson

Select **OK** to send the message. Then, check in the SAP Cloud Platform Integration's message monitoring to verify that the request was successfully received and processed. Check in your mail account to see that you received the mapped ORDERS.ORDERS05 IDoc message.

Now you've successfully set up a real AS2 sender to your integration flow, so you're able to send messages to the **AS2** sender adapter in your integration flow. The next step in the scenario setup is to send the message to a real IDoc receiver.

7.3.3 Add an IDoc Receiver

To configure the receiver of your scenario as an IDoc receiver, you need to set up an IDoc receiver adapter, which sends the ORDERS.ORDERS05 IDoc to the receiver backend, and you have to configure the receiver backend for IDoc inbound processing.

> **No IDoc Receiver Backend Available?**
>
> If you have no receiver backend available or don't want to set up the SAP Cloud Platform Connectivity service to connect to the on-premise backend, you may skip this section and keep the mail receiver channel. You'll still be able to continue with the sample scenario creating the acknowledgement in the next section.

For configuring the connection to the IDoc receiver, open the integration flow in edit mode, and delete the **Mail** receiver channel. Draw a new line between the **End** message event and the **Receiver.** Then select the **IDoc** adapter in the adapter selection screen.

Configure the IDoc adapter channel as depicted in Figure 7.37. In the **Address** field of the **Connection** tab, set the URL to call the IDoc processing of the receiver system. The URL is constructed as follows: *http://<server>:<port>/sap/bc/srt/idoc?sap-client=<client>*, where *server* and *port* are the server and the HTTP(S) port of the receiver system, respectively, and the *client* is the ABAP client in the system you want to post the IDoc to.

As **Proxy Type**, you probably have to select **On-Premise** because you want to connect to an on-premise system, which is usually not accessible from the Internet. To configure this connection, you have to set up and configure SAP Cloud Platform Connectivity, which will be described in the next section. If your on-premise system can be called from the Internet, you choose **Internet** as the **Proxy Type**, and then you don't have to set up SAP Cloud Platform Connectivity and can skip the next section.

As the **IDoc Content Type**, select **Text/XML** to send out the IDOC in XML format.

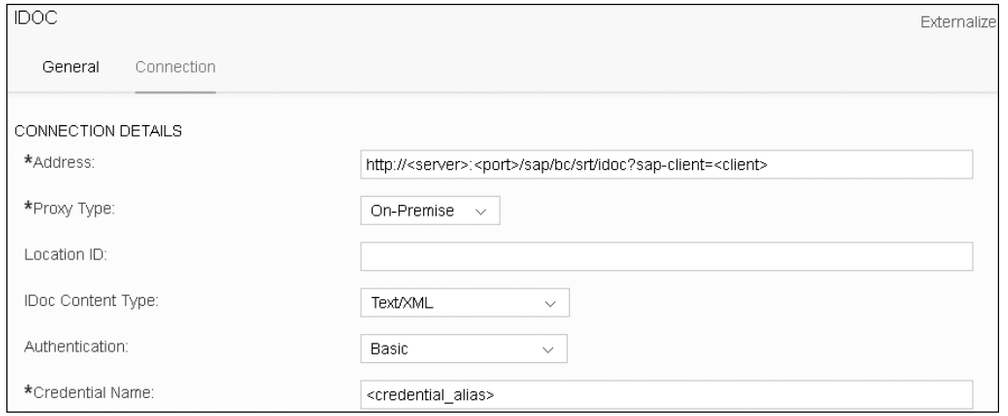


Figure 7.37 Configuring the IDoc Channel

> **Only HTTP Allowed If the Proxy Type On-Premise Is Used**
>
> Note that you need to define the address in the IDoc channel starting with *http://* if the connection is configured via SAP Cloud Platform Connectivity. However, the port you define can be either an HTTP or an HTTPS port as configured in the SAP Cloud Platform Connectivity configuration in the next section. This is because the connection to SAP Cloud Platform Connectivity is always done using a secure HTTP tunnel. The connection to the backend itself is then established via HTTPS or HTTP as configured in the system mapping in the SAP Cloud Platform Connectivity configuration. Further details can be found in the "Using SAP Cloud Platform Cloud Connector with SAP Cloud Platform Integration" blog in the SAP Community (*www.sap.com/community.html*).

If you want to use basic authentication to connect to the receiver backend, deploy the user credentials in the **Security Artifacts** monitor as already described in several scenarios in the book. You then use this security artifact and configure it in the **Credential Name** field of the IDoc channel.

Configure SAP Cloud Platform Connectivity

As already mentioned, you need to set up and configure SAP Cloud Platform Connectivity to set up a secure connection to an on-premise system. The detailed installation and configuration procedure is described in the online documentation for SAP Cloud Platform at *https://help.sap.com/viewer/p/CP* in the **Cloud Connector** section

and in the "Using SAP Cloud Platform Cloud Connector with SAP Cloud Platform Integration" blog in the SAP Community (*www.sap.com/community.html*). After you've done the installation and initial configuration of SAP Cloud Platform Connectivity, you can connect SAP Cloud Platform Connectivity to your SAP Cloud Platform Integration account. Create and configure the subaccount in the subaccount dashboard of the SAP Cloud Platform Connectivity configuration per the description in the online documentation.

Now that your SAP Cloud Platform Integration tenant is connected to SAP Cloud Platform Connectivity, you can create the cloud to on-premise system mapping for your IDoc backend in the SAP Cloud Platform Connectivity configuration for your subaccount. In the **Cloud To On-Premise** section (Figure 7.38), add a new system mapping using the + icon at the top of the upper table.

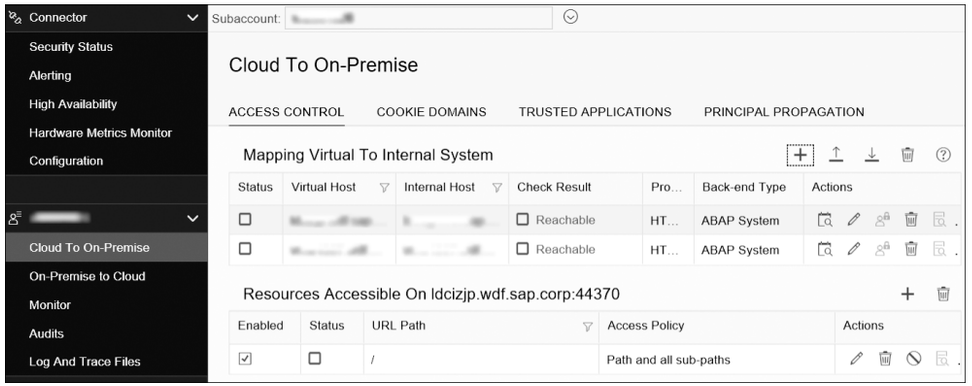


Figure 7.38 Adding System Mapping in SAP Cloud Platform Connectivity

In the **Add System Mapping** wizard, configure the connection to your ABAP receiver system via HTTP or HTTPS, and enter the hostname and the HTTP or HTTPS port of your IDoc receiver system. As principal type, select **None** if you want to forward the credentials entered in the IDoc channel.

After defining the system mapping to your receiver system, execute the availability check using the icon. Your system should then appear as **Reachable** in the **Check Result** column (see Figure 7.38).

For this newly created system mapping, you then need to define the accessible resource using the + icon at the top of the lower table. In the **Add Resource** dialog, either enter "/" as the **URL Path** and allow access to all subpaths (see Figure 7.39) or define the specific **URL Path** as "/sap/bc/srt/idoc" as configured in the IDoc channel.

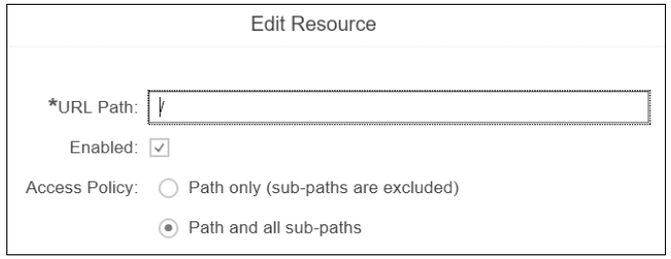


Figure 7.39 Edit Resource Dialog

Client Certificate-Based Authentication Using SAP Cloud Platform Connectivity

If you want to use client certificate-based authentication to the receiver system, you need to set up the client certificate in SAP Cloud Platform Connectivity as described in the "HCI: Integrate On-Premise ERP with HCI IDoc Adapter Using HANA Cloud Connector & Client Authentication" blog in the SAP Community (*www.sap.com/community.html*).

After you've set up SAP Cloud Platform Connectivity to connect your SAP Cloud Platform Integration tenant to your receiver system, you need to configure the IDoc processing in the receiver system.

Configure IDoc Processing in the Receiver System

To receive and process the IDoc in an SAP system based on an Application Server ABAP (AS ABAP), multiple configuration steps are required: you have to define logical system settings, set up ports, and configure partner profiles. As these are basic IDoc configuration steps, we won't explain them in detail here in this book. You can refer to the detailed documentation in Transaction SALE in your receiver backend.

Note that for the sample scenario, it isn't urgently required to configure all the IDoc configurations if you don't want to get the order processed in the application. It's sufficient to activate the HTTP service to receive IDocs via HTTP, and then you can monitor the IDoc in the system's IDoc runtime. The IDoc will be in error state, but from the connectivity point of view, the IDoc is received by the receiving system.

To receive IDoc documents via HTTP, you need to register the IDoc service in the SOAP runtime using Transaction SRTIDOC. Run the transaction, and select **Execute** to activate the HTTP-based IDoc service (Figure 7.40).

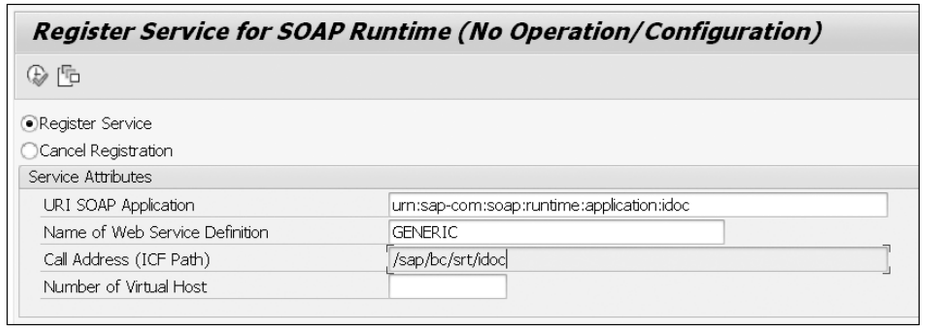


Figure 7.40 Registering the HTTP Service for IDoc Processing

Now your receiver system can receive IDoc documents via HTTP from the SAP Cloud Platform Integration tenant.

Let's try it out.

Trigger Scenario Execution

To start the processing, trigger a message from the Mendelson AS2 test tool. Then, check in the SAP Cloud Platform Integration's message monitoring that the message was processed successfully.

To check if the IDoc was received by the receiver backend, search for the ORDERS05 IDoc in the IDoc monitoring. Call Transaction WE05, and search for IDoc documents with basic type ORDERS05. If you haven't executed all the IDoc-specific configurations, the ORDERS IDoc should appear in error status, as shown in Figure 7.41. The error **56 EDI: Partner profile not available** indicates that the partner profile isn't available for further processing of the IDoc. This shows that the IDoc is successfully received in the receiver system with the settings we've defined, but the IDoc-specific configuration is missing. If you want, you can configure the partner profile in Transaction WE20 and continue with configuring the IDoc inbound processing so that the order is processed in the system and finally an invoice is sent back.

Because the IDoc-specific settings aren't in scope of this B2B sample scenario, we skip the configuration of the IDoc processing and continue with the configuration of acknowledgement handling in SAP Cloud Platform Integration.

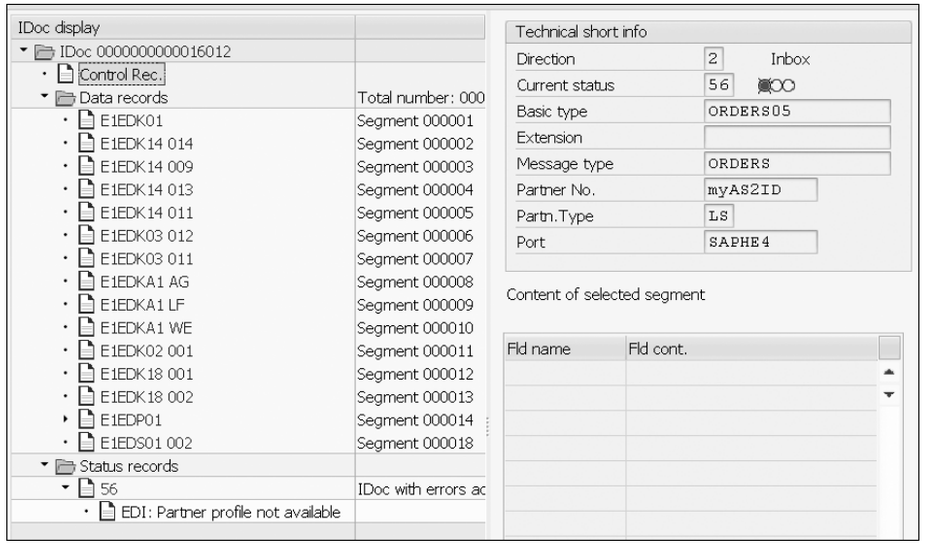


Figure 7.41 IDoc Monitoring in the Receiver System

7.3.4 Configure Acknowledgement Handling

When sending an EDI message, the sender usually expects a functional acknowledgement, also known as a 997 acknowledgement. The acknowledgement is used to notify that the message was received and validated and so can be further processed.

To address this requirement, SAP Cloud Platform Integration offers the option to validate the incoming EDI message and generate an acknowledgement in the **EDI Splitter** flow step.

This section will describe how to configure the acknowledgement in the integration flow. We'll extend the integration flow so that after the functional validation, an acknowledgement will be generated and sent via the **AS2** receiver adapter back to the original sender of the EDI message.

Let's get started.

Define a Number Range Object

Let's first configure a number range object (NRO) that will be required in the next configuration step to define the unique interchange number. Using number ranges,

the runtime generates unique IDs for a specific NRO. Those unique IDs can be used in steps, such as the **EDI Splitter** step, or in scripts. For a brief introduction of NRO, refer to Table 7.1.

In the SAP Cloud Platform Integration's monitoring dashboard, select the **Number Ranges** tile in the **Manage Stores** section (for more details of the monitor, refer to Chapter 8, Section 8.4.4). At the top of the table, select **Add** to define a new NRO. Give the NRO a unique **Name**, and define the **Minimum Value** and **Maximum Value** (Figure 7.42). Furthermore, select **Rotate** so that the numbers will start with the minimum value again when the maximum is reached.

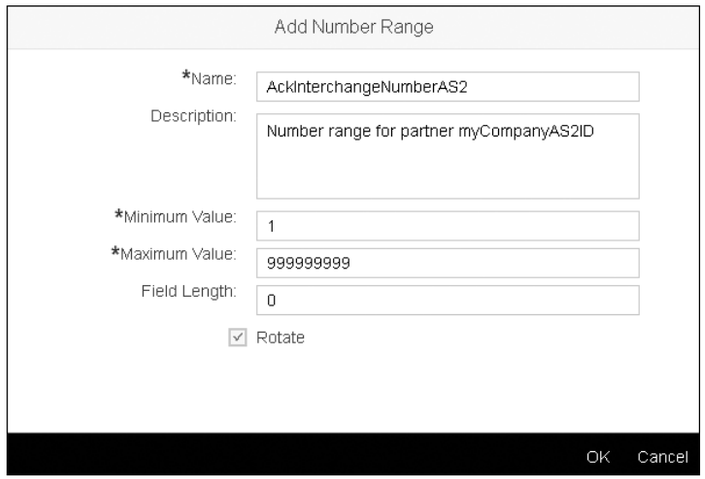


Figure 7.42 Defining the NRO

Configure Acknowledgement Handling in the Integration Flow

After the NRO is created, we can configure the acknowledgement handling in the EDI splitter and define the outbound processing for the acknowledgement.

As already indicated, the EDI splitter splits the incoming EDI bulk messages into single EDI messages. However, it can also validate them and generate an acknowledgement for the whole interchange containing the validation result.

To activate the acknowledgement creation, open the **EDI Splitter** step in the integration flow, and select **Required** in the **Create Acknowledgement** dropdown in the **X12** tab (Figure 7.43). One additional configuration option appears in which you need to

define whether the **Interchange Number** for the acknowledgement is taken from the inbound EDI message (**Use From EDI Message** option) or whether it's generated using an NRO (**Number Range** option). Usually, the interchange number is taken from the inbound message, but to demo the NRO feature in SAP Cloud Platform Integration, select the **Number Range** option for our sample scenario. After selecting this option, an input field for the **Number Range** is shown where you enter the name of the number range you created in the last step.

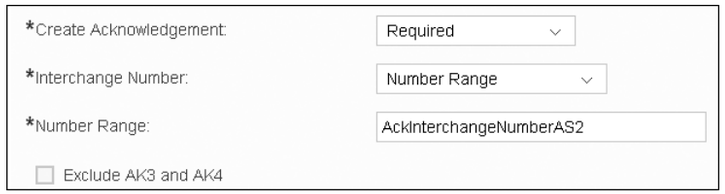


Figure 7.43 Configuring Acknowledgement Handling in EDI Splitter

The **Exclude AK3 and AK4** checkbox defines whether you want to get the detailed error information in the acknowledgement in the segments AK3 and AK4 in case of an error during validation. We don't select this flag, so that we get the detailed validation error later in our test.

Now that we've configured that an acknowledgement will be sent, we have to configure its receiver.

Configure the AS2 Receiver Adapter

As we received the inbound message in our scenario via the **AS2** adapter, we'll also send the acknowledgement back to the sender using the AS2 protocol. For this, we also use the **AS2** receiver adapter.

To configure the receiver of the acknowledgement, add another **Receiver** participant to the integration flow, and connect the **End** message event coming from the router to the new receiver, as depicted in Figure 7.44. Select the **AS2** adapter in the adapter selection dialog.

Configure the **AS2** receiver adapter as depicted in Figure 7.45. In the **Connection** tab, in the **Recipient URL** field, enter the URL of the local HTTP receiver from the AS2 Mendelson client. To get this URL, check in the **MDN URL** field in the **MDN** tab of your local AS2 partner configured in the AS2 Mendelson client (Figure 7.46). Make sure the correct IP address of your local system is entered there.

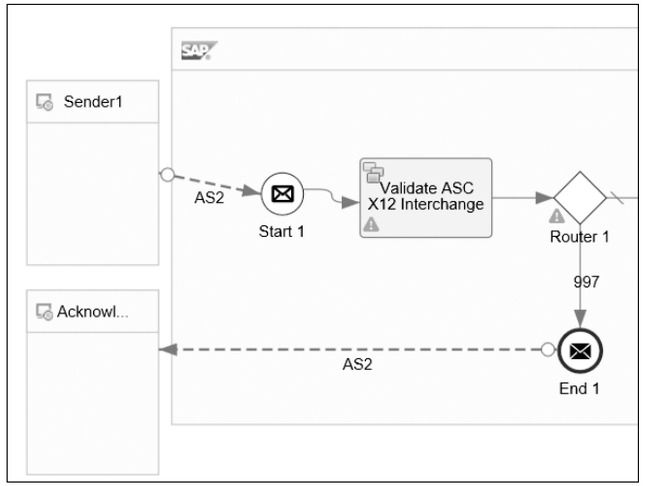


Figure 7.44 Adding the Receiver of the Acknowledgement

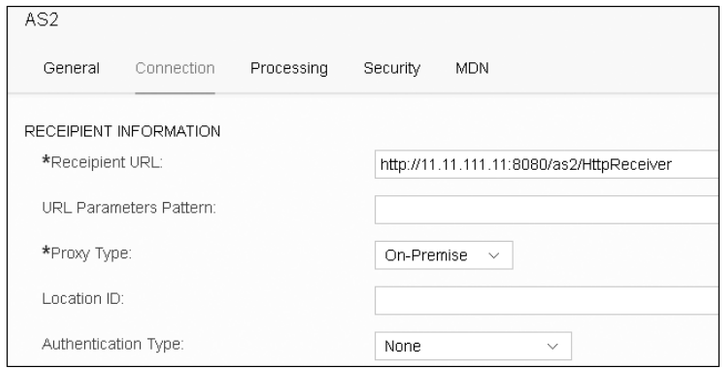


Figure 7.45 Configuring the Connection Tab in AS2 Receiver Channel

Figure 7.46 Getting the URL of the Mendelson Receiver

As **Proxy Type**, you most likely have to configure **On-Premise** because the system where the AS2 Mendelson client is installed isn't reachable from the Internet. Because of this, you need to set up the connection using SAP Cloud Platform Connectivity whose configuration is done in the next section.

No SAP Cloud Platform Connectivity Configured?

To set up that an acknowledgement is sent back via **AS2** receiver adapter, you need to configure SAP Cloud Platform Connectivity to connect to the local Mendelson AS2 tool. If you don't want to set up SAP Cloud Platform Connectivity for this scenario, you could also use a **Mail** receiver adapter and send the acknowledgement to your mailbox for test purposes.

In addition to the connection details, you configure the specific AS2 processing settings in the **Processing** tab. As depicted in Figure 7.47, configure the AS2-specific settings. For the sample scenario, you enter the following mandatory settings:

- **Own AS2 ID**
  Specify the same ID as used in the **AS2** sender channel. This is the AS2 ID identifying the SAP Cloud Platform Integration system.
- **Partner AS2 ID**
  Specify the ID of the partner receiving the acknowledgement. It should also match the ID used in the **AS2** sender channel.
- **Message Subject**
  Define **997** to indicate that this is a 997 acknowledgement.
- **E-Mail Address**
  Enter an email address. Note that this address is required per the AS2 protocol but not used at runtime.
- **Content-type**
  Define the content type of your acknowledgement. We set **application/edi-x12** because it's an ASC X12 message.

For a detailed explanation of the configuration fields in the **AS2** receiver channel, refer to the "Configure Communication Channel with AS2 Adapter" section in the documentation for SAP Cloud Platform Integration at *https://help.sap.com/viewer/product/CLOUD_INTEGRATION/Cloud*.

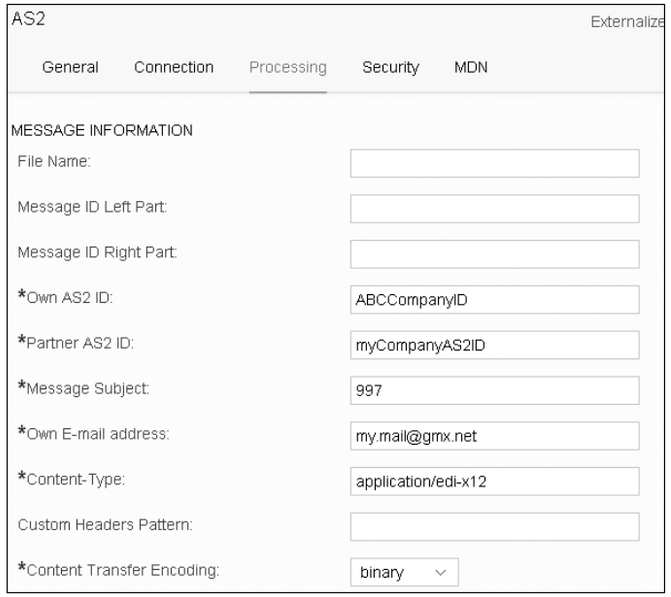


Figure 7.47 Configuring the Processing Tab in the AS2 Receiver Channel

You can leave the default values in the configuration options in the **Security** and **MDN** tabs because we don't want to use signature and encryption in the sample scenario. We also don't request an MDN for the acknowledgement. If you want to extend the sample scenario, refer to the "B2B Capabilities in SAP Cloud Platform Integration – Part 2" blog in the SAP Community (*www.sap.com/community.html*).

Save and deploy the integration flow.

Configure SAP Cloud Platform Connectivity

As already indicated, the connection to the HTTP URL of the AS2 Mendelson tool will probably have to be established using SAP Cloud Platform Connectivity because the system AS2 Mendelson is running on can't be reached from the Internet.

In Section 7.3.3, you've already seen how to set up SAP Cloud Platform Connectivity and how to connect it to your SAP Cloud Platform Integration tenant. There we showed you how to configure the connection to an SAP system based on AS ABAP to send the IDoc messages to. Now you have to configure a connection to your local system, where the AS2 Mendelson tool is running.

Log on to SAP Cloud Platform Connectivity. In the **Cloud To On-Premise** section (Figure 7.48), add a new system mapping using the + icon at the top of the upper table. In the **Add System Mapping** wizard, configure the connection to a **Non-SAP System** via HTTP, and enter the IP address and the HTTP port of your local system. As **Principal Type**, select **None**.

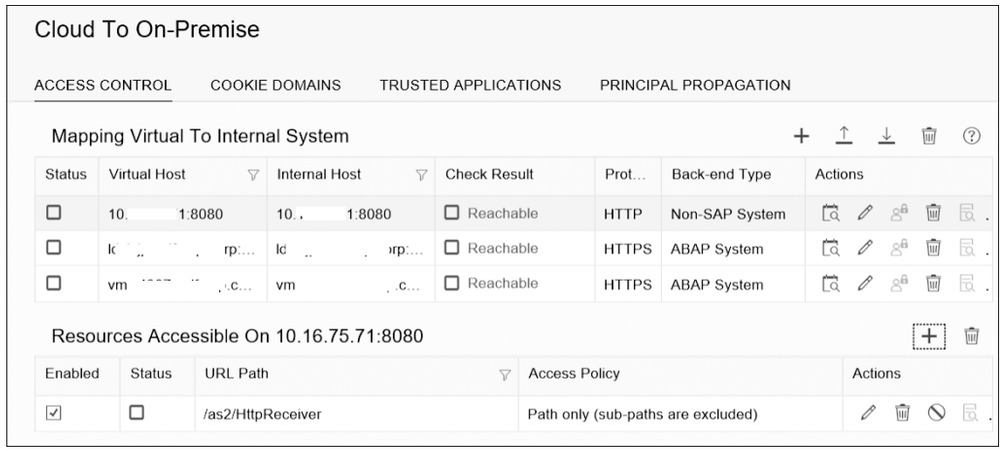


Figure 7.48 Adding Mapping to the Local System in SAP Cloud Platform Connectivity

After defining the system mapping to your receiver system, execute the availability check using the icon. Your system should then appear as **Reachable** in the **Check Result** column.

For this newly created system mapping, you then need to define the accessible resource using the + icon at the top of the lower table. In the **Add Resource** dialog, either enter "/" as the **URL Path** and allow access to all subpaths, or define the specific **URL Path** as "/as2/HttpReceiver", as configured in the AS2 channel.

Now the configuration of the acknowledgement handling is completed. You're ready to run your scenario.

Run the E2E Scenario

Trigger the scenario from your AS2 Mendelson tool as described before. Use the sample message `850 - Purchase Order.txt` as EDI test message. In the SAP Cloud Platform Integration's monitoring, you should see one message with **Completed** status. The receiver system should still receive the ORDERS IDoc. So far, there's no difference. But

now, the AS2 Mendelson tool should indicate that it got back an acknowledgement message as a response to the request (Figure 7.49).

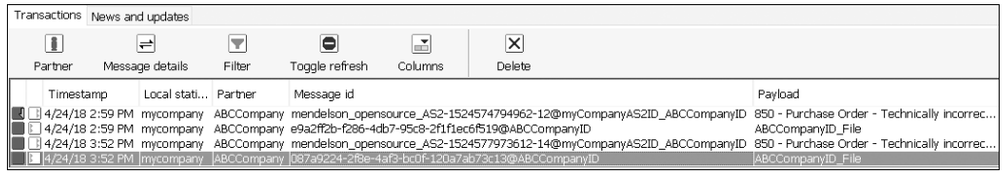


Figure 7.49 Transactions in AS2 Mendelson

On double-clicking the acknowledgement entry, the **Message Details** screen opens. Select the **Transferred Payload** tab to see the received acknowledgement. As the payload is an X12 EDI message, you need to have some knowledge about the structure of a 997 acknowledgement to understand its content. We won't describe this in detail here as this can be found online at several places, but we point you to the segments that indicate whether the message was accepted.

Let's have a look at the 997 acknowledgement we received in Figure 7.50. The first segments provide the header details of the interchange, and the important segment to identify if the message was accepted is `AK5`. `AK5` in this sample acknowledgement shows that the whole transaction sent in the interchange was accepted; this is indicated by the `A` in the `AK5` segment.

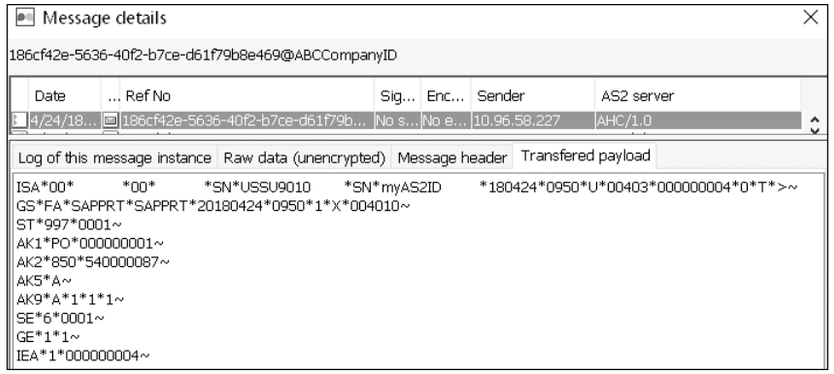


Figure 7.50 997 Acknowledgement for an Accepted EDI Message

Now let's execute the scenario with a message that won't pass the validation. In the AS2 Mendelson tool, select the sample message `850 - Purchase Order - Technically`

`incorrect.txt` (the file is also available with the book downloads) as the EDI test message and send it. In the SAP Cloud Platform Integration's message monitoring, you should still see one message with **Completed** status, but the receiver system should not receive the ORDERS IDoc.

Why is the message completed, and where does the error appear? The logic is that the validation of the EDI inbound message is executed, and if there is an error, the sender is notified via the 997 acknowledgement that the message wasn't accepted. With this, the processing is completed from SAP Cloud Platform Integration's perspective, and the sender needs to correct the message and send it again.

Open the message monitoring, and search for your message. As shown in Figure 7.51, in the **Attachments** tab, you can find an attachment with the name **Splitter Validation Error Document**. This file contains the error information of the validation. The details of the validation error are given back in the 997 acknowledgement in case of a validation error.

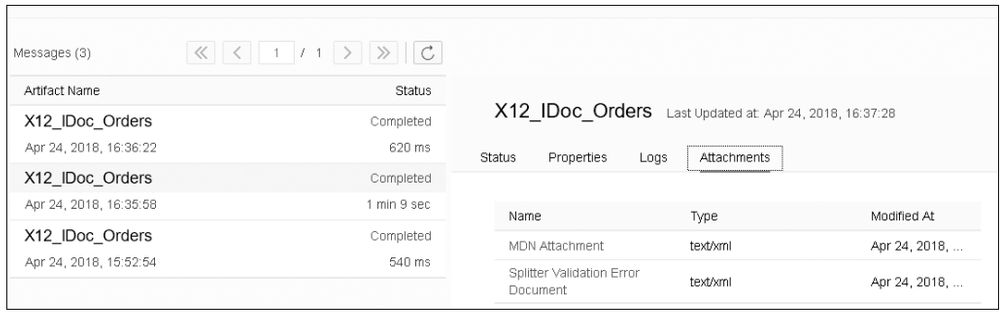


Figure 7.51 Message with Validation Error Attachment

Select the **Splitter Validation Error Document** to get the details as depicted in Figure 7.52. You see that there was a segment error with error code `5`, which means that the data element on position `1` in segment `2` was too long.

Now let's have a look at the 997 acknowledgement in the AS2 Mendelson tool to see if we can find the same details there. Open the received 997 acknowledgement message. It should look like the one shown in Figure 7.53. In segment `AK5`, we see that the message was rejected indicated by the `R`. We also see the error code `5 - One or more segments in error`.

Artifact Name: X12_IDoc_Orders Status: Completed Processing Time: 2 min 39 sec
Last Updated at: Apr 24, 2018, 16:37:28 Log Level: Debug

Log | MDN Attachment | Splitter Validation Error Document

```
<Interchange>
  <DocumentStandard>ASC-X12</DocumentStandard>
  <InterchangeSender>
    <Identification>myAS2ID        </Identification>
    <Qualifier>SN</Qualifier>
  </InterchangeSender>
  <InterchangeReceiver>
    <Identification>USSU9010       </Identification>
    <Qualifier>SN</Qualifier>
  </InterchangeReceiver>
  <InterchangeControlNumber>000000001</InterchangeControlNumber>
  <FunctionalGroup>
    <GroupControlNumber>000000001</GroupControlNumber>
    <GroupSender>
      <Identification>SAPPRT</Identification>
    </GroupSender>
    <GroupReceiver>
      <Identification>SAPPRT</Identification>
    </GroupReceiver>
    <MessageError type="850">
      <MessageControlNumber>540000087</MessageControlNumber>
      <SegmentError>
        <Error>
          <ElementType>DataElement</ElementType>
          <XPath>/Interchange/S_GS/M_850/S_BEG/D_353</XPath>
          <ErrorCode>5</ErrorCode>
          <ErrorText>Data element is too long</ErrorText>
          <SegmentPosition>2</SegmentPosition>
          <DataElementPosition>1</DataElementPosition>
        </Error>
      </SegmentError>
    </MessageError>
  </FunctionalGroup>
</Interchange>
```

Figure 7.52 Splitter Validation Error Document

Message details

087a9224-2f8e-4af3-bc0f-120a7ab73c13@ABCCompanyID

Date	... Ref No	Sig...	Enc...	Sender	AS2 server
4/24/1...	087a9224-2f8e-4af3-bc0f-120a7ab...	No s...	No e...	10.96.58.227	AHC/1.0

Log of this message instance | Raw data (unencrypted) | Message header | Transfered payload

```
ISA*00*          *00*          *SN*USSU9010       *SN*myAS2ID        *180424*0152*U*00403*000000006*0*T*>~
GS*FA*SAPPRT*SAPPRT*20180424*0152*1*X*004010~
ST*997*0001~
AK1*PO*000000001~
AK2*850*540000087~
AK3*BEG*2**8~
AK4*1*353*5~
AK5*R*5~
AK9*R*1*1*0~
SE*8*0001~
GE*1*1~
IEA*1*000000006~
```

Figure 7.53 997 Acknowledgement for a Rejected EDI Message

But where is the detailed error information? For this, you have to check the segments AK3 and AK4. In segment AK3, you find the information regarding which segment of the inbound message caused the validation error (BEG), the count of the segment in error (2), and the error code (8 - Segment has data element errors). To know which data element in the indicated segment caused the error, you need to check segment AK4 of the 997 acknowledgement. The first data element (1) with the X12 data element number 353 caused the error 5 - Data element is too long.

You can now easily relate to the sample message. Open the file 850 - Purchase Order - Technically incorrect.txt in a text editor. Search for the segment BEG on position 2 in the group segment (GS), and check the first data element there (Listing 7.1). The data element reads 022.

```
GS*PO*SAPPRT*SAPPRT*20080404*091606*000000001*X*004010~
ST*850*540000087~
BEG*022*KN*5400000087**20180328~
```

Listing 7.1 Segments in the EDI Message

To check how long this field needs to be, you can easily use the ICA and check the structure of the inbound MIG. There you see that the data element 353 in segment BEG is expected to be exactly two characters long. Furthermore, in the code list, you see that only value 02 is allowed (Figure 7.54). This explains the error because the value in the sample message has three characters.

OVERVIEW | STRUCTURE | NOTES (5)

Structure

Node	Constraint	Cardinality	Position	Primitive Type	Syntax Data Type	Length	Codelist
850 – Purchase Order		1..1					
ST – Transaction Set Header - Purchase Order	850	1..1	010				
BEG – Beginning Segment for Purchase Order		1..1	020				
353 – Transaction Set Purpose Code		1..1	01	Token	ID	2..2	☑

DETAILS | NOTES (0) | CODELIST : 353

Selected Codelist : 353

General Information

Identifier: 353

Name: Transaction Set Purpose Code

Code Values (1) Suchen

02 – Add

Figure 7.54 Data Element 353 in MIG in ICA

You may correct the payload and set `02` instead of `022` and resend the message. Then it should pass the validation, and the IDoc should be sent successfully to the receiver system.

Now that you've configured the complete B2B scenario, you may wonder how to make the integration flow more dynamic, so that different partner-specific configuration settings can be used within the same integration flow. This will be explained in the next section.

7.4 Using the Partner Directory for Partner-Specific Configuration Data

When establishing a communication network between many communication partners, the tenant Partner Directory helps you to simplify the configuration and maintenance of the integration flows. In such scenarios where many communication partners are involved, you don't need to set up specific integration flows for every partner, but you can build a single one or a few integration flows that are then parametrized by partner-specific information stored in the Partner Directory. With this approach, you reduce the numbers of integration flows, which also results in lower maintenance costs of the overall scenario.

7.4.1 Concept of Partner Directory

The design of the Partner Directory is shown in Figure 7.55. The Partner Directory is a tenant-specific database-based component used to store partner-specific configuration data relevant for the scenario execution, such as endpoints, alternative partner IDs, XSLT mappings, XSD definitions, or certificates. This configuration data is used dynamically at runtime when an integration flow is executed.

To allow you to store those parameters in the Partner Directory, SAP Cloud Platform Integration provides a set of OData APIs. At the time of publishing this book, no UI is delivered from SAP Cloud Platform Integration to maintain the configuration data in the Partner Directory. Using the OData APIs, the owner of the tenant, who is the host of the whole B2B scenario, builds an application where the partners involved in the scenario can maintain their specific configuration data.

The different flow steps and adapters in the integration flow configured for the B2B scenario have to be parametrized to read the partner-specific information from the Partner Directory during the runtime of the integration flow.

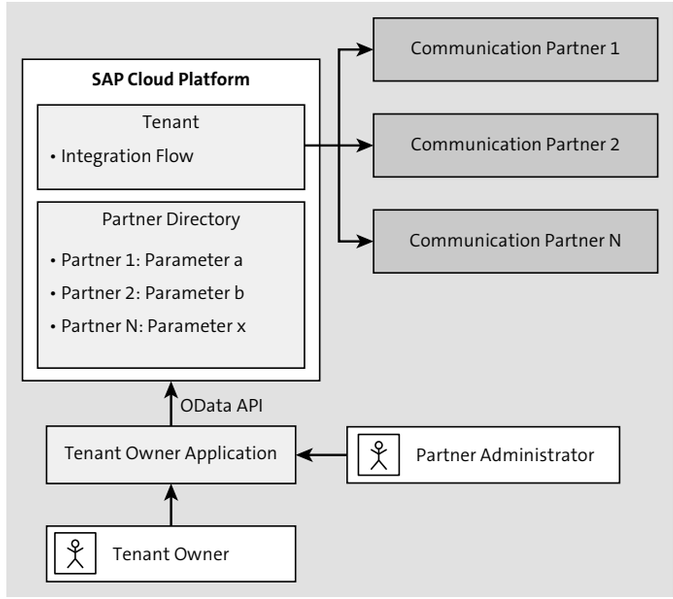


Figure 7.55 Usage of Partner Directory

It's important to understand that because the parameters in the Partner Directory are partner-specific, they have to be read at runtime based on partner-specific values from the incoming request or payload so that the correct configuration is used. The partners, sender and receiver, are usually identified by specific values from the payload.

With the Partner Directory, you can add new communication partners without downtime and without changing or redeploying the integration flows. You can enter attributes of a new partner via the OData API without interrupting the message processing.

Further Details and Sample Scenarios Using the Partner Directory

The Partner Directory offers additional advanced features beside storing simple configuration data:

- XSLT mappings and XSD schemas can be stored.
- Alternative partner IDs can be defined for specific partners.

- Authorized users can be created and used for advanced authorization checks.
- User credential aliases and certificates can be stored and used for authentication and authorization.

Although we don't explain these options in detail here, refer to the "Cloud Integration – Partner Directory – Step-by-Step Example" blog and the referenced blogs in the SAP Community *(www.sap.com/community.html)* to understand the details of those configuration options and to set up advanced scenarios using the Partner Directory.

Now that you understand the idea behind the Partner Directory, let's enhance the sample scenario we've set up by making some attributes dynamic and reading them from the Partner Directory.

7.4.2 Use a Receiver Endpoint URL Dynamically in the Integration Flow

In this section, we'll extend the sample scenario so that specific configuration settings are read from the Partner Directory instead of being defined as fixed values in the integration flow.

To keep the scenario simple, we'll just parameterize the endpoint URL and the corresponding credential alias in the IDoc receiver channel. You could also parametrize XSLT mappings and XSD definitions and make all the partner-specific configurations in the integration flow dynamic, but this is beyond the scope of this chapter. Refer to the SAP Cloud Platform Integration documentation and the referenced blogs.

You Didn't Set Up the IDoc Receiver?

If you kept the mail receiver in your sample scenario and didn't set up the IDoc receiver adapter but want to extend the scenario using dynamic configuration from the Partner Directory, you may parameterize the mail address in the mail receiver channel instead. With this, you're able to continue with the sample scenario using the Partner Directory; just store the mail address in the Partner Directory instead of the IDoc receiver URL.

To use partner-specific attributes from the Partner Directory, we need to extend the integration flow in a way that it first reads that attribute from the Partner Directory and then uses it in a specific integration flow step or adapter. In our sample scenario,

we read the endpoint URL and the credential alias for the EDI receiver partner from the Partner Directory and use it in the IDoc adapter.

Read Configuration Data from the Partner Directory Using the Script Step

To read a specific attribute from the Partner Directory, we need to know the partner ID for which the configuration data is defined and the parameter name that is used in the Partner Directory to store the configuration. For the sample scenario, we use the EDI receiver partner ID `USSU9010` defined in the sample payload, and we'll create two parameters, `Endpoint` and `CredentialAlias`, for this partner in the Partner Directory containing the address and the credential alias, respectively, of the receiver system for the IDoc message.

At runtime, you have to retrieve the receiver partner ID from the incoming message to use it for reading parameters for this partner from the Partner Directory. In the sample scenario, the EDI splitter already does this for us; it reads the EDI receiver partner ID from the incoming EDI message and sets it as header `SAP_EDI_Receiver_ID`. We use this header to retrieve the specific endpoint URL for this receiver partner.

You can easily read parameters from the Partner Directory using a script step, using the Java APIs exposed for the Partner Directory. We won't explain the APIs in detail, but point you to the online documentation (*https://help.sap.com/viewer/product/CLOUD_INTEGRATION/Cloud)* chapter *Accessing Partner Directory Content with the Script Flow Step*.

To extend the sample integration flow, open it in edit mode, and add a **Groovy Script** step from the **Message Transformers** group before the message **End** event, as depicted in Figure 7.56.

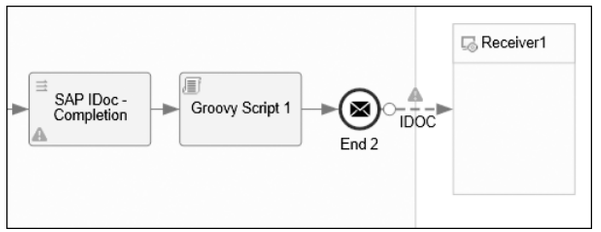


Figure 7.56 Adding a Groovy Script

Create a new groovy script using the **create** action on the right side of the flow step. Copy the coding (Listing 7.2) from the prepared script file *GroovyScript.txt* provided

in the book downloads at *www.sap-press.com/4650* into the **Groovy Script**. Select **OK** to get back to the integration flow configuration.

```
import com.sap.gateway.ip.core.customdev.util.Message;
import java.util.HashMap;
import com.sap.it.api.pd.PartnerDirectoryService;
import com.sap.it.api.ITApiFactory;
def Message processData(Message message) {
       def service = ITApiFactory.getApi(PartnerDirectoryService.class, null);
       if (service == null){
          throw new IllegalStateException("Partner Directory Service not
found");
       }
       def map = message.getHeaders();
       def receiverId = map.get("SAP_EDI_Receiver_ID");
       if (receiverId == null){
          throw new IllegalStateException("Receiver ID is not set in the
header 'SAP_EDI_Receiver_ID'")
       }

     def parameterValue = service.getParameter("Endpoint", receiverId ,
String.class);
     if (parameterValue == null){
        throw new IllegalStateException("Endpoint parameter not found in the
Partner Directory for the partner ID "+receiverId);
     }
     def parameterValueCredential = service.getParameter("CredentialAlias",
receiverId , String.class);
     if (parameterValueCredential == null){
        throw new IllegalStateException("CredentialAlias parameter not found
in the Partner Directory for the partner ID "+receiverId);
     }
      message.setProperty("RECEIVER_Endpoint", parameterValue );
      message.setProperty("RECEIVER_CredentialAlias", parameterValueCredential
);
       return message;
}
```

Listing 7.2 Groovy Script Code for Accessing the Partner Directory

At runtime, the script reads the header `SAP_EDI_Receiver_ID`, which represents the EDI receiver partner, and searches in the Partner Directory for the parameters `Endpoint` and `CredentialAlias` for this partner. The values of those parameters are then set as properties `RECEIVER_Endpoint` and `RECEIVER_CredentialAlias`.

Now that we've read the parameters from the Partner Directory, we can use it in the IDoc receiver channel in the next step.

Dynamic Configuration in the IDoc Receiver Channel

In the sample scenario, we've currently configured the **Address** field in the **IDoc** receiver channel with the URL to the **IDoc** endpoint in the **Receiver** system, and we configured a fixed credential **Alias**. As we want to set these parameters dynamically from the properties defined in the **Groovy Script**, we have to change this configuration.

To change the configuration, open the **IDoc** receiver channel. First copy the URL currently defined in **Address** field and the **Credential Name** into a notepad because we'll need them later when we define the parameters in the Partner Directory. Change the settings to `${property.RECEIVER_Endpoint}` and `${property.RECEIVER_CredentialAlias}`, as shown in Figure 7.57, to read the address and the credential alias dynamically from the properties defined in the script (you can refer to Chapter 6, Section 6.2, for dynamic configuration).

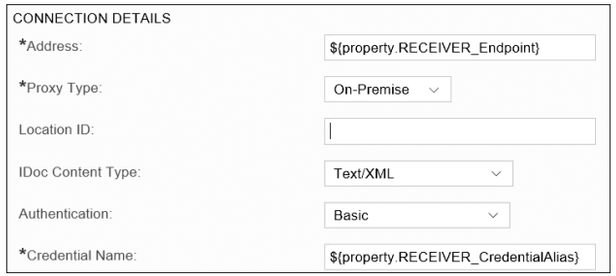


Figure 7.57 Configure Address and Credential Name in IDoc Receiver via Property

Save and deploy the integration flow. Now the endpoint address and the credential alias will be dynamically determined during runtime. If you were to send a message to your integration flow using the AS2 Mendelson tool, the message would end with the error `Endpoint parameter not found in the Partner Directory for the partner ID USSU9010` in the script step. The reason is that we haven't yet defined the endpoint in the Partner Directory. We'll do the necessary configuration in the next step.

7.4.3 Store the Partner-Specific Endpoint URL in Partner Directory

Now, to get the scenario running successfully, we have to add the `Endpoint` and `CredentialAlias` parameters for the partner `USSU9010` to the Partner Directory. As already mentioned, usually the partner would enter this configuration parameter using the application the tenant owner offers. But for our sample scenario, we'll create the entry directly via the OData API. The OData API can be called by the tenant administrator (`AuthGroup.Administrator`) or by a user with the role `AuthGroup.PartnerDirectoryConfigurator`.

Because the Partner Directory OData API is protected against cross-site request forgery (CSRF) attacks, you first have to fetch an X-CSRF Token before you can make create, change, or delete requests to entries in the Partner Directory. Refer to documentation for SAP Cloud Platform Integration (found at *https://help.sap.com/viewer/product/CLOUD_INTEGRATION/Cloud*) in the OData API topic for detailed information. In addition, refer to Chapter 9 for details on API availability and usage.

Fetch X-CSRF Token

The easiest way of calling OData APIs in the SAP Cloud Platform Integration tenant is using Postman (*www.getpostman.com*). Download, install, and start it, and you're ready to trigger your first request.

Use a **Get** request to the OData API root URL *https://<TMN-host>/api/v1* on the tenant management node (TMN). Select **Basic Auth**, and enter your credentials in the **Authorization** tab, as shown in Figure 7.58. In the **Headers** tab, create a new key `X-CSRF-Token` with the value `Fetch` to request for an X-CSRF Token (Figure 7.59).

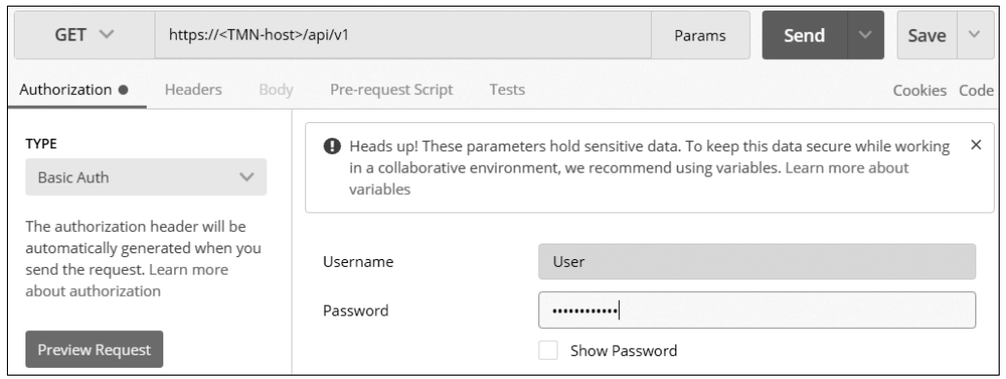


Figure 7.58 Configuring Authorization in Postman

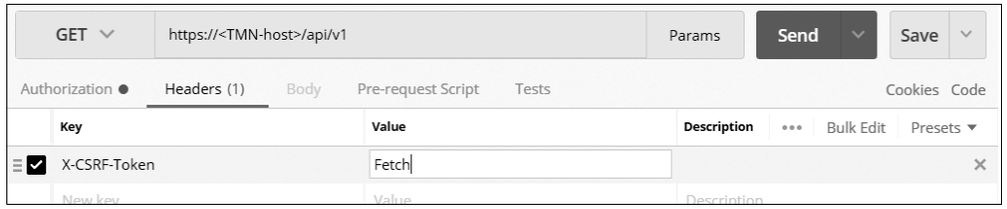


Figure 7.59 GET Request for an X-CSRF Token in Postman

Select **Send** to trigger the request. In the response, you receive a list of all available APIs in the **Body** tab. But more important for us is the very last header `X-CSRF-Token` in the **Headers** tab (Figure 7.60). This is the header we have to provide in our subsequent **PUT** request. Copy the value to use it in the next request.

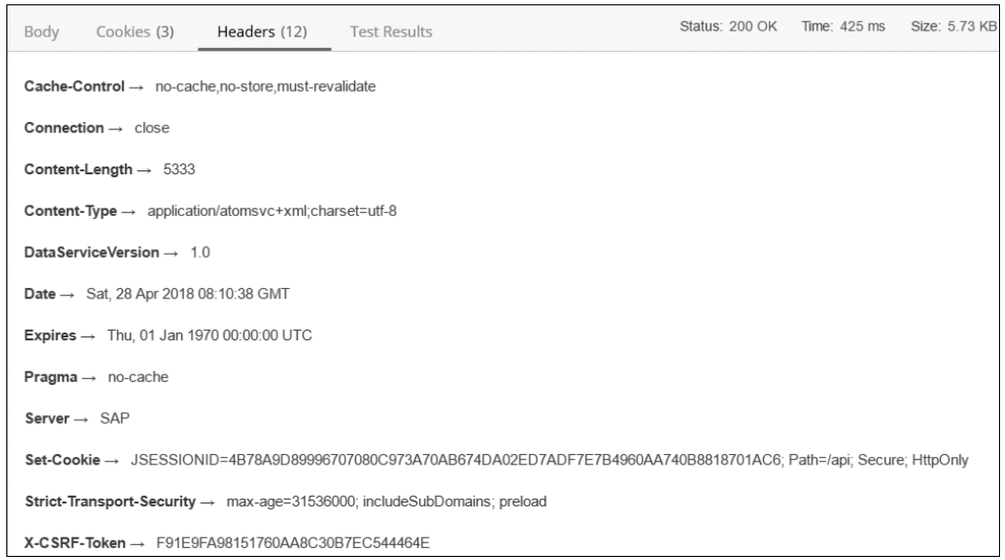


Figure 7.60 Received X-CSRF Token in the Headers Tab

Store the Endpoint URL and Credential Alias in the Partner Directory

Now that you've retrieved the X-CSRF Token, you can use it to trigger a POST request to store the endpoint URL of the IDoc receiver and the credential alias to the Partner Directory.

Create a new request and select **Post** as method. Enter the URL "https://<TMN-host>/api/v1/StringParameters" to create a simple string parameter in the Partner Directory. In the **Headers** tab, create three headers, as shown in Figure 7.61:

- `X-CSRF-Token`
  As the **Value**, enter the token you received in the last step. Note that the token is only valid for 30 minutes; afterwards, you need to retrieve a new token as described in the last step.
- `Accept`
  As the **Value**, select `application/json`.
- `Content-Type`
  As the **Value**, select `application/json`.

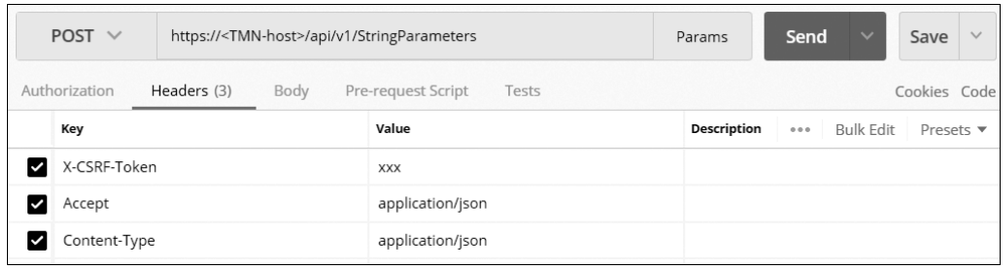


Figure 7.61 Required Headers in a POST Request to Create Parameters in the Partner Directory

In the **Body** tab, you need to provide the details for the POST request. Select **Raw** and **JSON(application/json)** to post the request in JSON format. In the entry field, enter the details of the partner and the parameter you want to create. For the endpoint URL, you enter the following JSON request: `{"Pid":"USSU9010","Id":"Endpoint","Value":"http://<host>:<port>/sap/bc/srt/idoc?sap-client=<client>"}`, as depicted in Figure 7.62. As the **URL**, enter the real URL to your IDoc receiver system, which you copied from the IDoc channel into the notepad in the previous section.

Select **Send** to post the request. With this request, a new parameter `Endpoint` is created in the Partner Directory for the partner `USSU9010` with the URL *http://<host>:<port>/sap/bc/srt/idoc?sap-client=<client>*. If the call was successful, you receive the details of the created entry in the response (Figure 7.63).

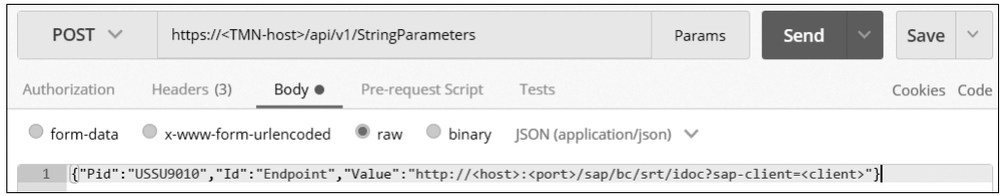


Figure 7.62 Body in the POST Request to Create the Endpoint Parameter

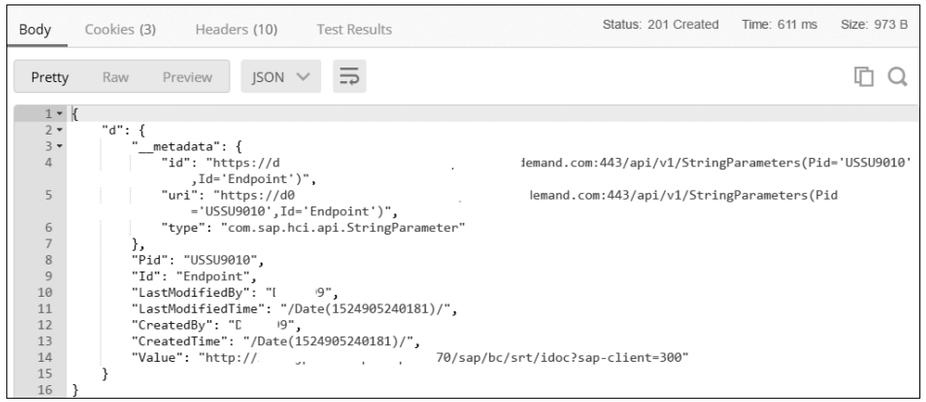


Figure 7.63 Response for a Successful POST Request

Execute another POST request for the parameter `CredentialAlias`. In the **Body** field, use the following JSON request: `{"Pid":"USSU9010","Id":"CredentialAlias","Value":"<credential alias>"}`. As the **Credential** alias, enter the alias of the credentials as deployed for this receiver.

With this last step, you finished the configuration of the scenario so that the endpoint address and the credential alias are retrieved dynamically from the Partner Directory during runtime. Let's test to see if it works.

Run Scenario

From your AS2 Mendelson test client, trigger a new message to the integration flow. The message should be sent successfully, and an acknowledgement should be received.

In the SAP Cloud Platform Integration's message monitoring, the message should be in status **Completed**, and the IDoc should be successfully received by the receiver backend.

Error Sending Acknowledgement?

If your message is in status **Retry** in the message monitor, you have to check the error message. If the error message is `Remote server returned response code 502` and `error message Bad Gateway`, most probably the acknowledgement can't be sent because the IP address of your AS2 Mendelson HTTP server to receive the acknowledgement changed. This happens because local machines usually get new IP addresses dynamically when they connect to a network. To fix this problem, get your new IP address, and enter it in the SAP Cloud Platform Connectivity's configuration in the **Internal Host** field. Keep the **Virtual Host** as is because SAP Cloud Platform Connectivity maps the virtual host (used in the **AS2** receiver channel) to the internal host (refer to Section 7.3.4 for how to configure SAP Cloud Platform Connectivity).

Now you've successfully extended your sample scenario to fetch the endpoint address and the corresponding credential alias dynamically. Now you can easily add a second receiver partner to the same scenario by just adding another partner with its endpoint URL and the credential alias to the Partner Directory. In addition, you would have to deploy the credentials for the new receiver backend. If a message for this partner is received, it would automatically be routed to the new receiver using the newly deployed credentials.

7.5 Summary

You're now able to configure MIGs and MAGs in the ICA and can configure B2B scenarios based on available B2B templates with the adapters and flow steps available in SAP Cloud Platform Integration. You understand the B2B acknowledgement handling and are able to interpret 997 acknowledgements. You also learned to define partner-specific attributes in the Partner Directory and how to use them in the integration flow. With this knowledge, you're now well equipped to configure your own B2B scenarios.

With this chapter we complete the design and configuration of integration scenarios in the web designer of SAP Cloud Platform Integration and come to another important part in the lifecycle of integration content, the monitoring. In the next chapter we will work you through the monitoring capabilities of SAP Cloud Platform Integration.

Contents

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

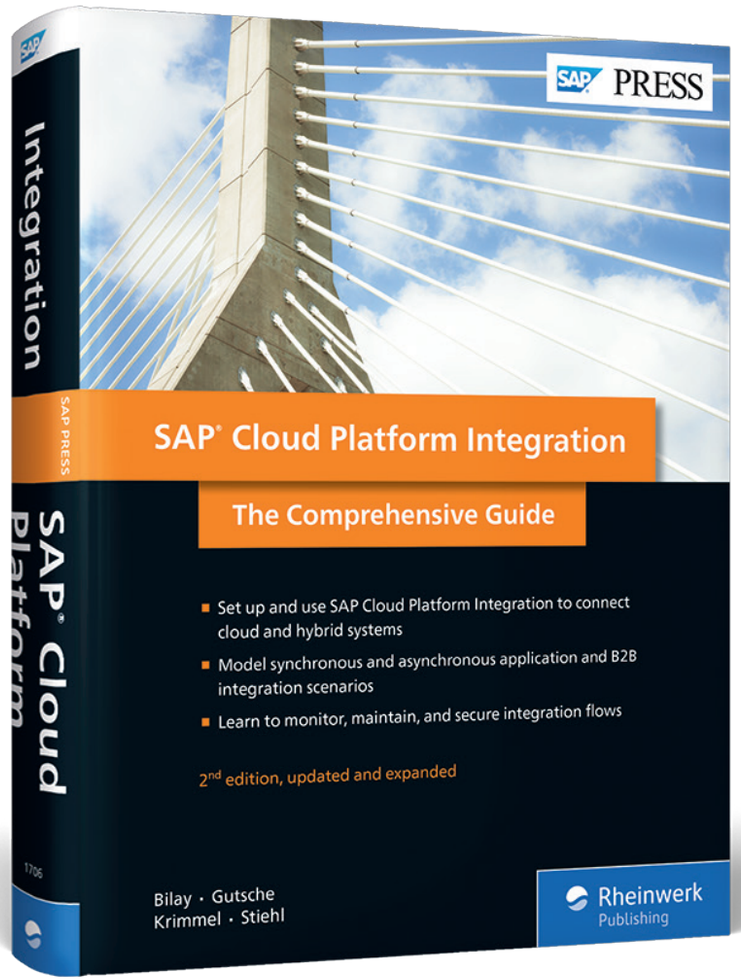
T

U

V

W

X



John Mutumba Bilay, Peter Gutsche, Mandy Krimmel, Volker Stiehl

SAP Cloud Platform Integration: The Comprehensive Guide

792 Pages, 2018, $89.95
ISBN 978-1-4932-1706-9

www.sap-press.com/4650

John Mutumba Bilay studied computer engineering and finance at the University of Cape Town, South Africa. He currently works as a senior software engineer and enterprise integration consultant at Rojo Consultancy B.V. in the Netherlands. His SAP specialties include SAP integration- and process-related technologies.

Dr. Peter Gutsche studied physics at Heidelberg University, Ruperto Carola. After completing his Ph.D., he joined SAP in 1999. Today, as a knowledge architect, he is responsible for the product documentation of SAP HANA Cloud Integration and works on documentation concepts for cloud software.

Mandy Krimmel studied engineering at Humboldt University, Berlin, Germany. In 2001, Mandy joined SAP, working on various integration-related projects, including SAP Process Integration and SAP Cloud Platform Integration. In her current role as product owner, she is responsible for the design, architecture, and development of various cloud integration components.

Prof. Dr. Volker Stiehl studied computer science at the Friedrich-Alexander-University of Erlangen-Nuremberg. After 12 years as a developer and consultant at Siemens, he joined SAP as the chief product expert responsible for the success of SAP Process Orchestration, SAP Process Integration, and SAP HANA Cloud Integration. He currently teaches business information technology at the Ingolstadt Technical University of Applied Sciences.