

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.

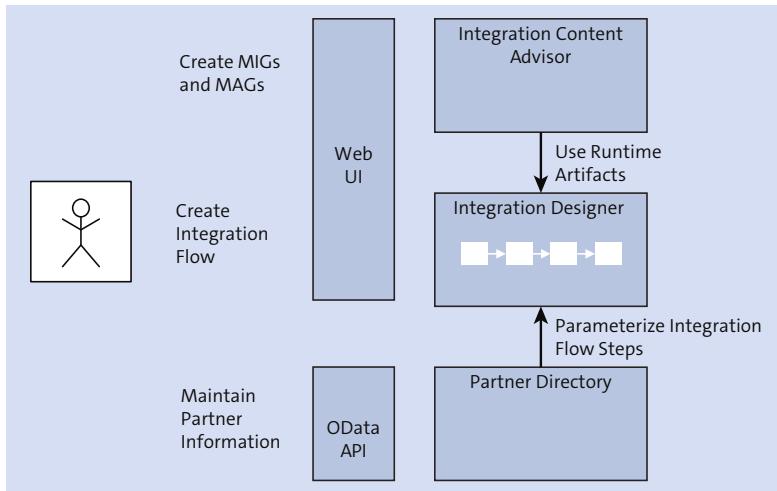


Figure 7.1 Schema of Involved Components in Configuring the B2B Scenario

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.

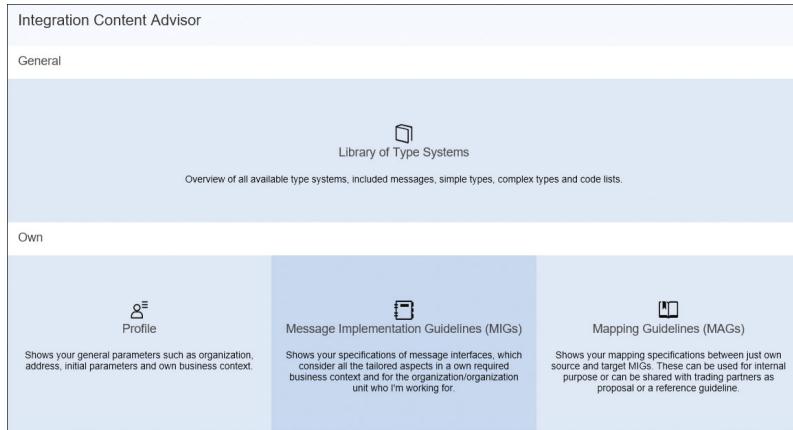


Figure 7.2 ICA Entry Page

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.

Integration Content Advisor / Library of Type Systems /		
Library of Type Systems		
ISO – International Standardization Organization - Codelists and Schemes	Responsible Agency: ISO	
ISO creates documents that provide requirements, specifications, guidelines or characteristics that can be used consistently to ensure that materials, products, processes and services are fit for their purpose. The ISO also in this scope, ISO also develops and maintains international	Provisioned By: GEFEG >	
	License Type: Licensed	
IDoc – SAP Intermediate Documents	Responsible Agency: SAP	
Standard SAP format for electronic data interchange between systems (Intermediate Document). Different message types (such as delivery notes or purchase orders) generally correspond to	Provisioned By: SAP >	
	License Type: Licensed	
UN/CEFACT – Trade Facilitation Recommendations	Responsible Agency: UN/ECE	
UN/CEFACT's trade facilitation recommendations are made to improve the ability of business, trade and administrative organizations, from developed, developing and transitional economies, to exchange products and relevant services effectively. Its principal focus is on facilitating	Provisioned By: GEFEG >	
	License Type: Licensed	
UN/EDIFACT – United Nations Electronic Data Interchange For Administration, Commerce and Transport	Responsible Agency: UN/ECE	
"United Nations/Electronic Data Interchange For Administration, Commerce and Transport (UN/EDIFACT) is the international EDI standard developed under the United Nations. The work of maintenance and further development of this standard is done through the United Nations	Provisioned By: GEFEG >	
	License Type: Licensed	

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.

The screenshot shows the SAP Integration Content Advisor interface. At the top, there is a breadcrumb navigation: Integration Content Advisor / Library of Type Systems / IDoc /. Below the header, the title is "SAP IDoc - SAP Intermediate Documents". A navigation bar below the title includes tabs for OVERVIEW, VERSIONS (1), MESSAGES (47), COMPLEX TYPES (440), SIMPLE TYPES (3411), and CODELISTS (270). The OVERVIEW tab is currently selected and highlighted with a blue underline. The main content area is divided into several sections:

- General Information**: Contains fields for Responsible Agency (SAP – SAP SE), Provisioned By (SAP), Status (Published), and Web Site (http://help.sap.com/saphelp_nw70/helpdata/en/0b/2a6095507d11d18ee90000e8366fc2/frameset.htm).
- Documentation**: Contains a Definition of IDocs and a Copyright Statement pointing to SAP's legal page.
- Administrative Data**: Shows creation and modification details: Created By: SYSTEM, Modified By: SYSTEM, Created On: 24 Jan. 2018 12:00 UTC, Modified On: 24 Jan. 2018 12:00 UTC.

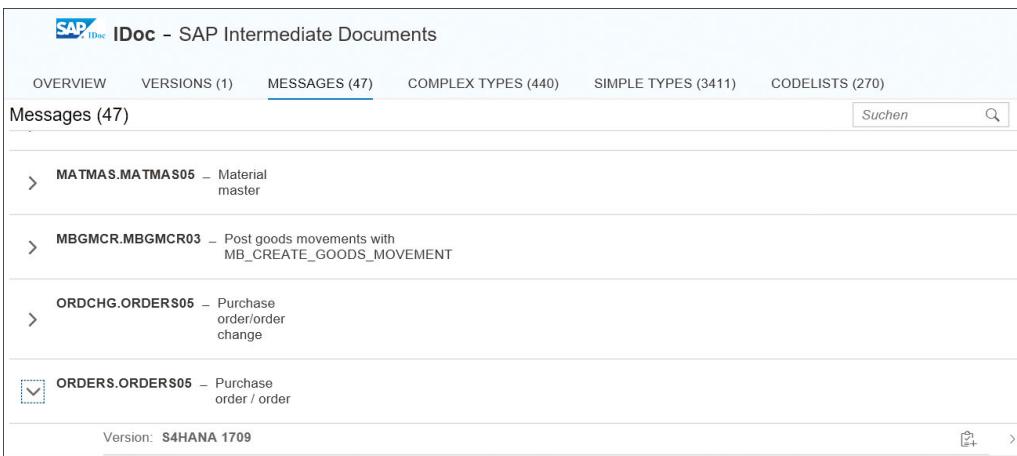
Figure 7.4 SAP IDoc Type System: Overview

- 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**.
- 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.



The screenshot shows the SAP IDoc interface with the title "IDoc - SAP Intermediate Documents". The top navigation bar includes links for OVERVIEW, VERSIONS (1), MESSAGES (47), COMPLEX TYPES (440), SIMPLE TYPES (3411), and CODELISTS (270). The "VERSIONS (1)" tab is selected, indicated by a dashed border. Below the tabs, a search bar contains the text "Suchen" and a magnifying glass icon. A button with an upward arrow is also present. The main content area displays a single version entry: "S4HANA 1709" (Externally Published On: 30 Sep. 2017 12:00 UTC, Number of Messages: 47, Number of Complex Types: 270, Syntax Type: IDOC). To the right of this entry is a note: "The content of this version contains IDocs from newest S/4 HANA Release 1709. As IDocs are developed in a compatible way, the content can also be used for older ERP and S/4 HANA releases." A small navigation arrow is located at the end of this note.

Figure 7.5 Versions for the SAP IDoc Type System



The screenshot shows the SAP IDoc interface with the title "IDoc - SAP Intermediate Documents". The top navigation bar includes links for OVERVIEW, VERSIONS (1), MESSAGES (47), COMPLEX TYPES (440), SIMPLE TYPES (3411), and CODELISTS (270). The "MESSAGES (47)" tab is selected, indicated by a solid blue underline. Below the tabs, a search bar contains the text "Suchen" and a magnifying glass icon. A button with an upward arrow is also present. The main content area lists several message interfaces: "MATMAS.MATMAS05" (Material master), "MBGMCR.MBGMCR03" (Post goods movements with MB_CREATE_GOODS_MOVEMENT), "ORDCHG.ORDERS05" (Purchase order/order change), and "ORDERS.ORDERS05" (Purchase order / order). The "ORDERS.ORDERS05" entry has a blue dashed border around its row, indicating it is currently selected. At the bottom of the list, there is a note: "Version: S4HANA 1709" followed by a small navigation arrow.

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.

Integration Content Advisor / Library of Type Systems / IDoc / ORDERS.ORDERS05 /

Message: **ORDERS.ORDERS05** – Purchase order / order Version: S4HANA 1709

OVERVIEW **STRUCTURE** NOTES (0)

Structure

Node	Constraint	Cardinality	Position	
✓ ORDERS05 – Purchasing/Sales				
> EDI_DC40 – IDoc Control Record for Interface to External System		1..1		
> E1EDK01 – IDoc: Document header general data		1..1		
> E1EDK14 – IDoc: Document Header Organizational Data		0..12		
> E1EDK03 – IDoc: Document header date segment		0..10		

Figure 7.7 Structure of Message ORDERS.ORDERS05

Integration Content Advisor / Library of Type Systems / IDoc / ORDERS.ORDERS05 /

Message: **ORDERS.ORDERS05** – Purchase order / order Version: S4HANA 1709

OVERVIEW **STRUCTURE** NOTES (0)

Structure

Node	Constraint	Cardinality	Position	Primitive...	Syntax D...	Length	Codelist
> EDI_DC40 – IDoc Control Record fo		1..1					
✓ E1EDK01 – IDoc: Document header		1..1					
ACTION – Action code for the wh	0..1	001	String	CHAR	0..3		<input checked="" type="checkbox"/>
KZABS – Flag: order acknowledg	0..1	002	String	CHAR	0..1		<input checked="" type="checkbox"/>
CURCY – Currency	0..1	003	String	CHAR	0..3		<input checked="" type="checkbox"/>

....

DETAILS NOTES (0) Codelist: ISO_4217

Properties **Documentation**

Tag: CURCY	EDI6345_A General ISO code for the currency (e.g. DEM for German Marks (deutschmark)).
Name: Currency	
Cardinality: min : 0 max: 1	

Figure 7.8 Properties of the Field CURCY

- 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).

DETAILS	NOTES (0)	CODELIST: ISO_4217
General Information		Code Values (178)
Identifier: ISO_4217		<input type="button" value="Search"/> <input type="button" value="Up"/>
Name: Currency codes		AED – Dirham
Type System: ISO		AFN – Afghani
Version Mode: Latest		ALL – Lek
Version: 2017		AMD – Dram
Documentation		ANG – Netherlands Antillean Guilder
Definition: Codes for the representation of currencies		AOA – Kwanza
ISO 4217 Edition 8		ARS – Argentine Peso

Figure 7.9 Code List Values for Field CURCY

- 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.
- 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

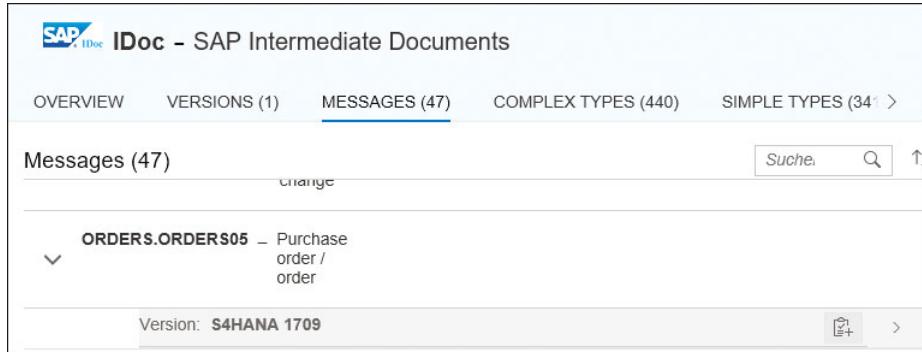
integration flow creation in [Section 7.3](#) and further explore the runtime features SAP Cloud Platform Integration offers for B2B integration.

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.ORDERSO5 message, open the **MESSAGES** tab in the **SAP IDoc** type system, and expand the message **ORDERS.ORDERSO5**. 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](#)).



The screenshot shows the SAP IDoc interface with the following details:

- Header:** SAP IDoc - SAP Intermediate Documents
- Navigation:** OVERVIEW, VERSIONS (1), **MESSAGES (47)**, COMPLEX TYPES (440), SIMPLE TYPES (34)
- Search:** Suchen (Search) with a magnifying glass icon
- Messages List:** Messages (47) with a 'Change' link
- Message Detail:** ORDERS.ORDERSO5 – Purchase order / order
- Version Information:** Version: S4HANA 1709
- Action Buttons:** Create (New MIG icon), Back, Forward

Figure 7.10 Creating a New MIG for ORDERS.ORDERSO5

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.

Create New Message Implementation Guideline

General Information

*Name: BookB2BMIG_ORDERS.ORDERS05
 *Direction: Out ▾
 Version: 1.0
 Status: Draft
 Message... : ORDERS.ORDERS05
 Type System: IDoc
 Type Syst... : S4HANA 1709

Documentation

Summary: B2B Integration: MIG for SAP IDoc ORDERS.ORDERS05

* Own Business Context +
 Business... : Create Order ▾

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.

The screenshot shows the SAP Integration Content Advisor interface for defining a Message Implementation Guideline (MIG). The title bar indicates the path: ... / IDoc / BookB2BMIG_ORDERS.ORDERS05 / and the message implementation guideline name: BookB2BMIG_ORDERS.ORDERS05. The version is listed as Version: 1.0. The top navigation bar includes buttons for Export, Activate, Get Proposals, Save, Cancel, and Delete.

The main area displays the structure of the MIG. The 'STRUCTURE' tab is selected, showing an overview of the nodes defined in the MIG. The table has columns for Node, Constraint, Cardinality, Position, Primitive..., Syntax Da..., Length, and Codelist.

Node	Constraint	Cardinality	Position	Primitive...	Syntax Da...	Length	Codelist
ORDERS05 – Purchasing/Sales		1..1					
EDI_DC40 – IDoc Control Recd		1..1					
E1EDK01 – IDoc: Document header		1..1					
ACTION – Action code for the document		0..1	001	String	CHAR	0..3	<input checked="" type="checkbox"/>
KZABS – Flag: order acknowledgement		0..1	002	String	CHAR	0..1	<input checked="" type="checkbox"/>
CURCY – Currency		0..1	003	String	CHAR	0..3	<input checked="" type="checkbox"/>
HWAER – EDI local currency		0..1	004	String	CHAR	0..3	
WKURS – Exchange rate		0..1	005	String	CHAR	0..12	
ZTERM – Terms of payment		0..1	006	String	CHAR	0..17	
KUNDEUINR – VAT Registration Number		0..1	007	String	CHAR	0..20	
EIGENUINR – VAT Registration Number		0..1	008	String	CHAR	0..20	
BSART – Document Type		0..1	009	String	CHAR	0..4	

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 define **Fixed Value** in the **Properties** section ([Figure 7.14](#)). Those values will be mapped automatically later in the MAG configuration.

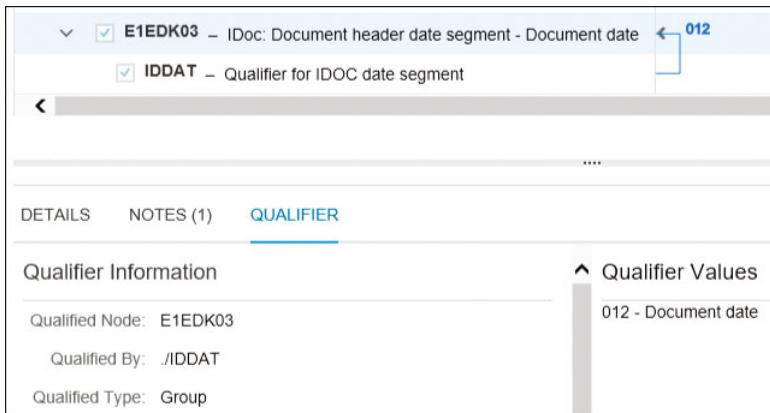


Figure 7.13 Qualified Field in the MIG

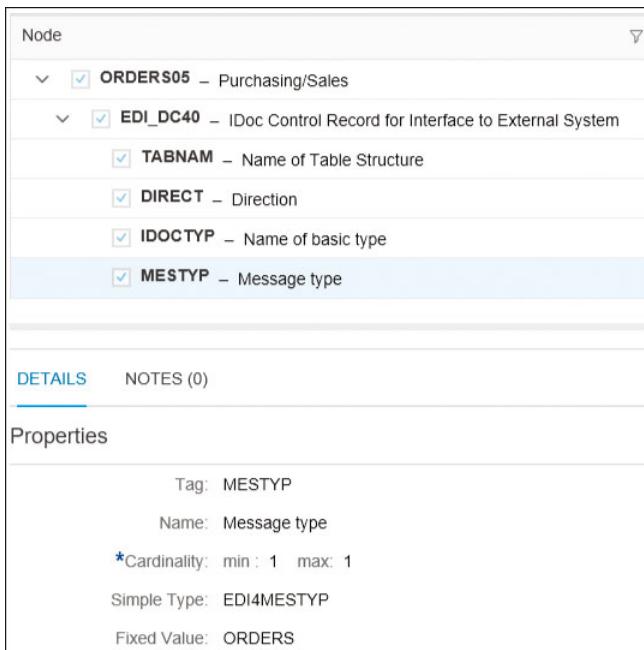


Figure 7.14 Defining a Fixed Value for the Message Type

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:	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:	Target: BookB2BMIG_ORDERS.ORDERS05		Suchen 			
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.

The screenshot shows the SAP Integration Content Advisor interface. At the top, there's a breadcrumb trail: ... / Mapping Guidelines / Mapping BookB2BMIG_PurchaseOrder to BookB2BMIG_ORDERS.ORDERS05 /. To the right are buttons for Export, Activate, Edit, Delete, and Copy, with Version: 1.0 displayed.

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: 695692bea4734130b85f624ab0eef19b	URL: /shell/ica/mags/undefined/magVersions/695692bea47341...
Created By: D 9	Modified By: C 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).

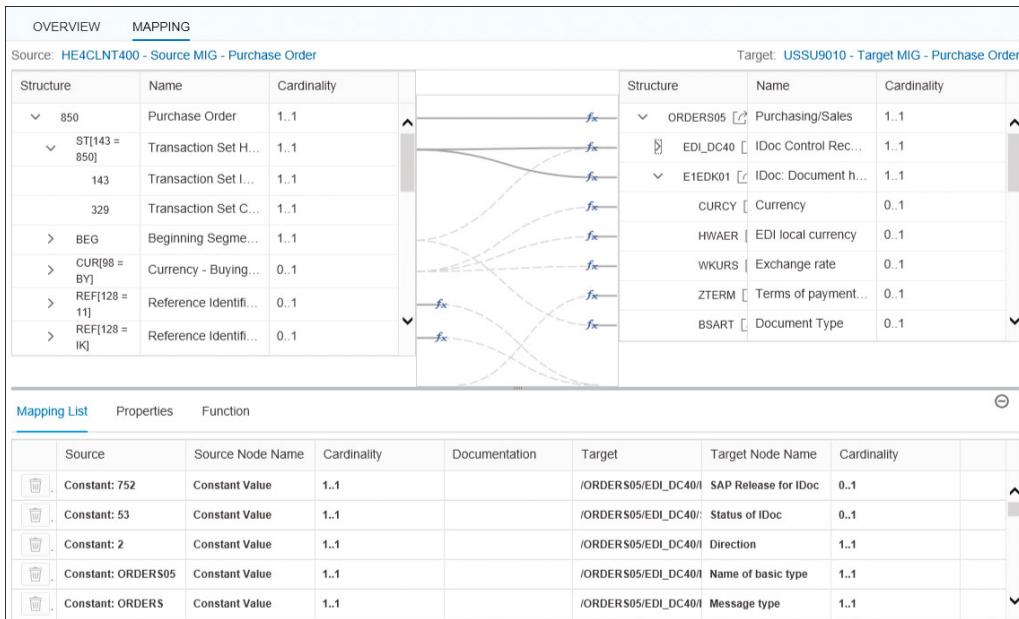


Figure 7.18 Creating Mapping between Source and Target MIGs

- 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.

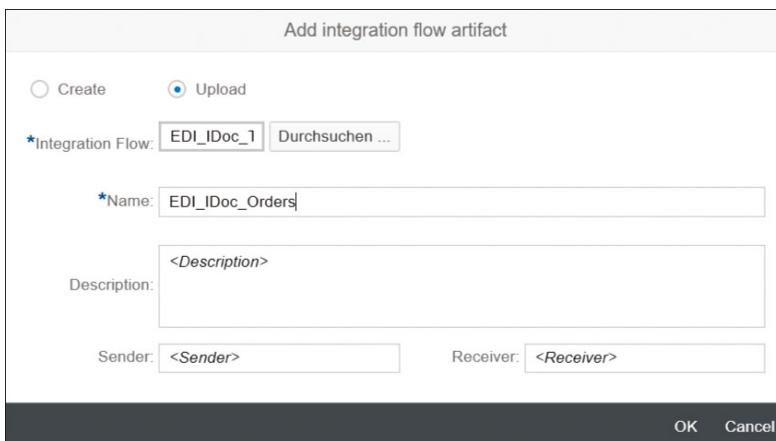


Figure 7.19 Creating an Integration Flow from a Template

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.

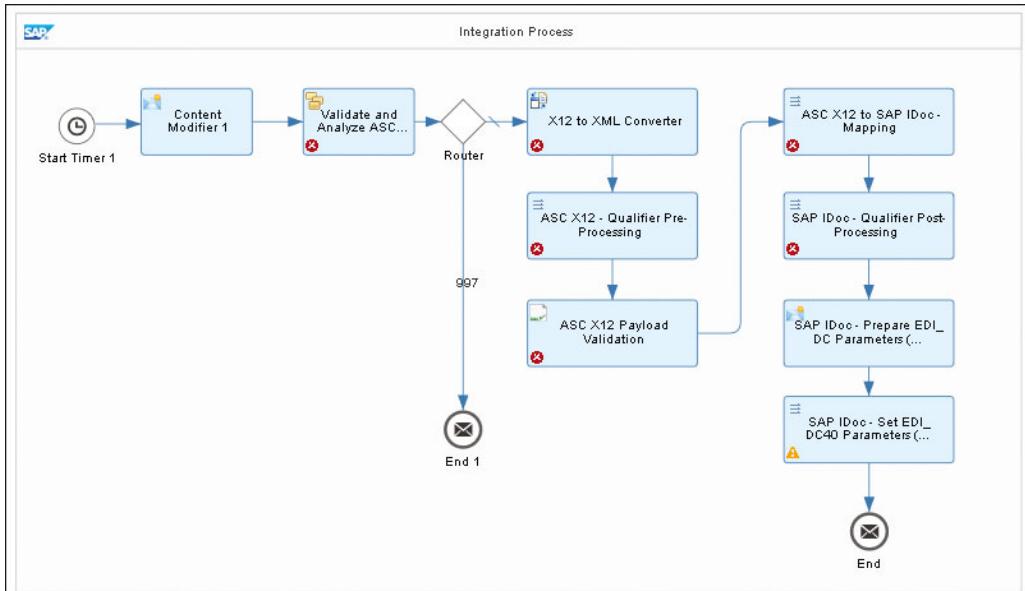


Figure 7.20 Created Integration Flow Based on a Template

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.

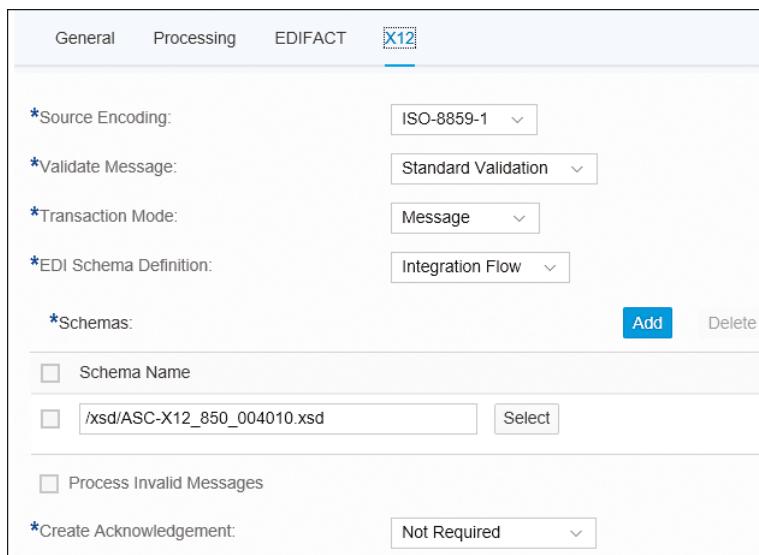


Figure 7.21 Configuring the EDI Splitter

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.

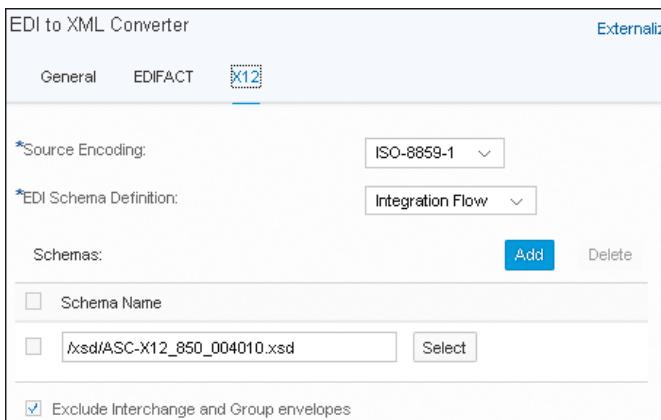


Figure 7.22 Configuring the EDI to XML Converter

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.

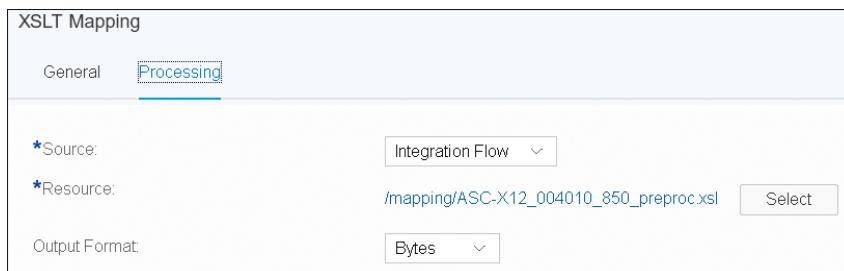


Figure 7.23 Configuring X12 Preprocessing

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.

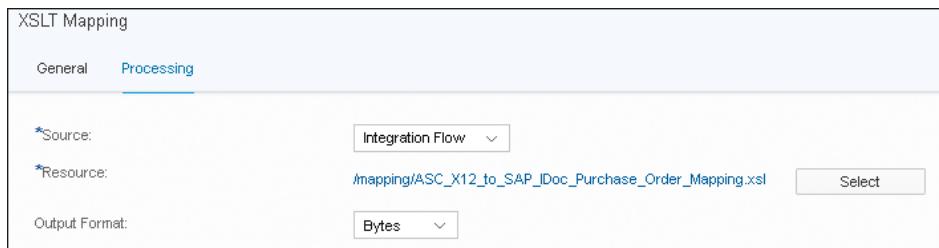


Figure 7.25 Configuring XSLT Mapping

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.

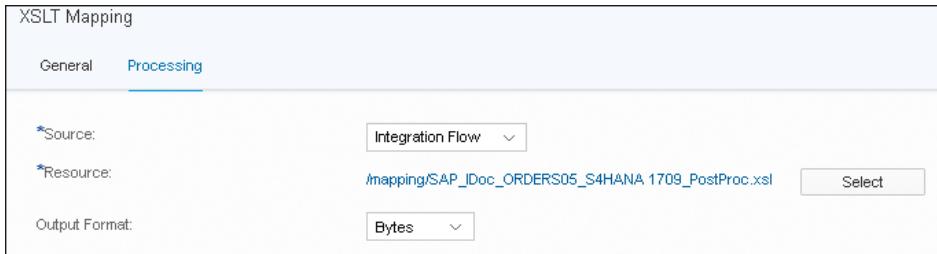


Figure 7.26 Configuring the IDoc Postprocessing

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](#)).

Attachments:				Add	Delete
<input type="checkbox"/>	Name	Mime-Type	Source	Header Name	
<input type="checkbox"/>	ORDERS.ORDERS05.xml	application/xml	Body		

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**.

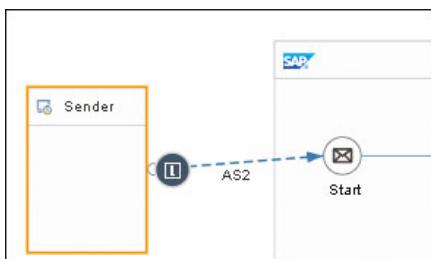


Figure 7.28 Integration Flow with AS2 Adapter

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.

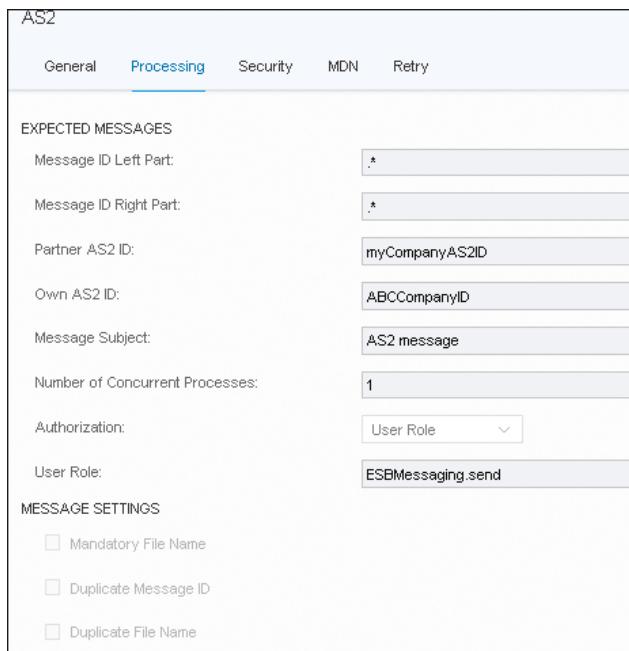


Figure 7.29 AS2 Sender Adapter Configuration

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`.

Integration Content (17)		Filter by Name or ID	Search	Clear	Help
Name	Status				
Book Integration Flow	Started				
Integration Flow					
Book Integration Flow with Splitter	Error				
Integration Flow					
JMS Resources	Started				
Integration Flow					
AS2Sender	Started				
Integration Flow					

Figure 7.30 Endpoint of the AS2 Sender Adapter A

AS2Sender [Restart](#) [Undeploy](#) [...](#)

Deployed On: Mar 07, 2018, 15:42:51 ID: AS2Sender
 Deployed By: D 9 Version: 1.0.0

[Endpoints](#) [Status Details](#) [Artifact Details](#) [Log Configuration](#)

<https://com.sap.cloud.gateway.as2/as2> . [mand.com/as2/as2](#)
 Own AS2 ID: "ABCCompanyID"; Partner AS2 ID: "myCompanyAS2ID"; Message Subject: "AS2 message"
[Edit](#)

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).

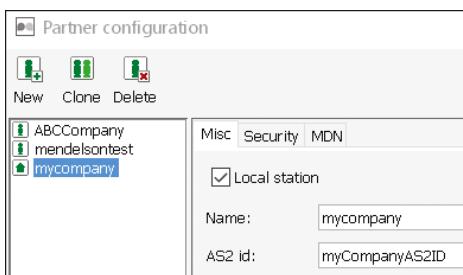


Figure 7.32 Own Partner Configuration in Mendelson

- 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.

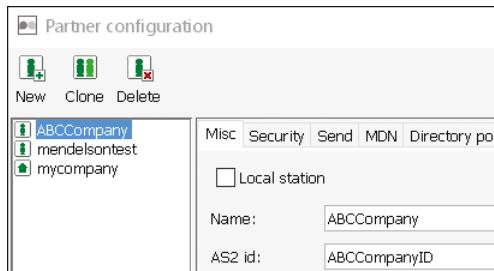


Figure 7.33 Partner Configuration for the SAP Cloud Platform Integration Tenant in Mendelson

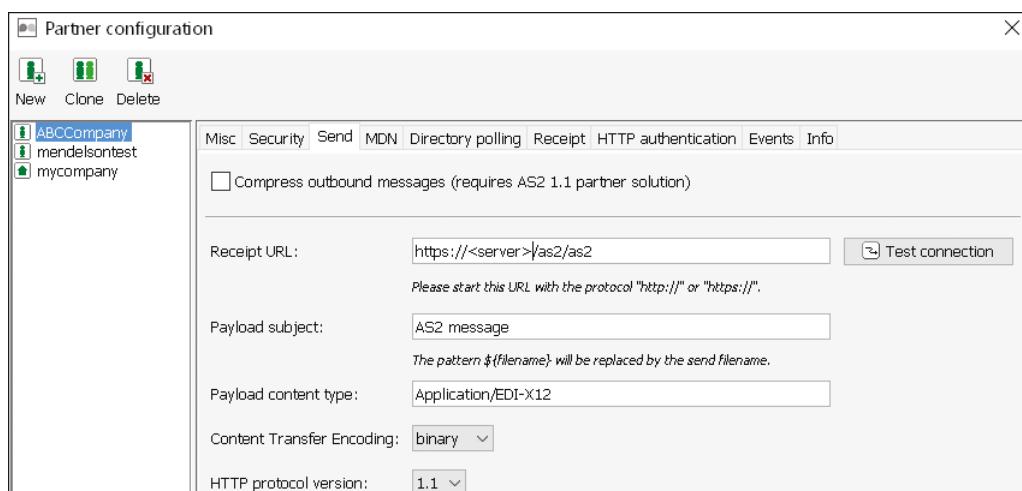


Figure 7.34 Configuring Endpoint and Payload Subject in Mendelson

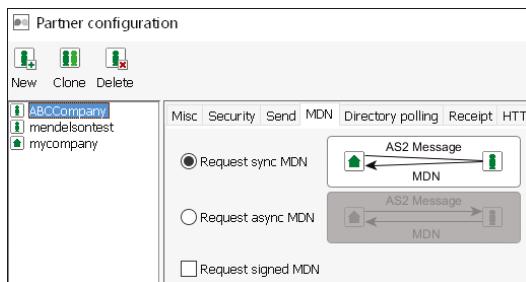


Figure 7.35 MDN Configuration in Mendelson

- 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](#).

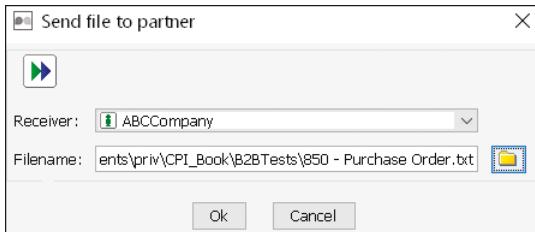


Figure 7.36 Sending a Test Message in Mendelson

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.

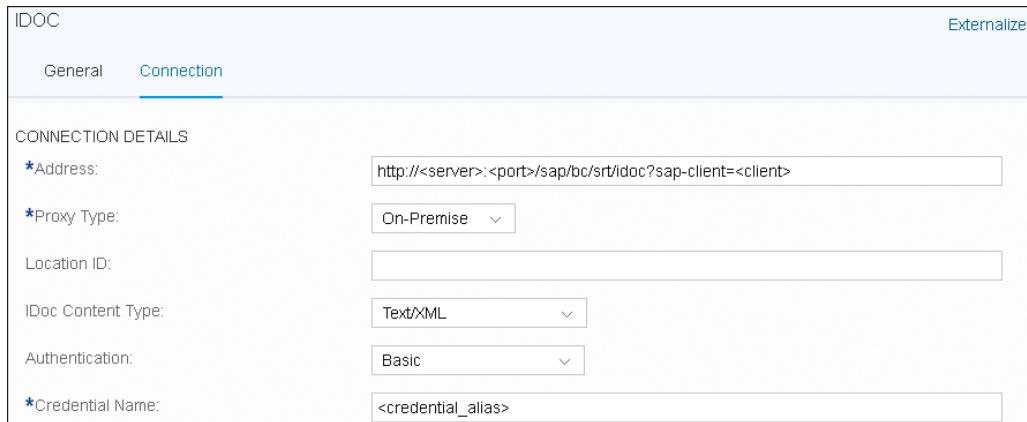


Figure 7.37 Configuring the IDoc Channel

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.

Status	Virtual Host	Internal Host	Check Result	Pro...	Back-end Type	Actions
<input checked="" type="checkbox"/>	[REDACTED]	[REDACTED]	<input checked="" type="checkbox"/> Reachable	HT...	ABAP System	
<input checked="" type="checkbox"/>	[REDACTED]	[REDACTED]	<input checked="" type="checkbox"/> Reachable	HT...	ABAP System	

Enabled	Status	URL Path	Access Policy	Actions
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	/	Path and all sub-paths	

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.

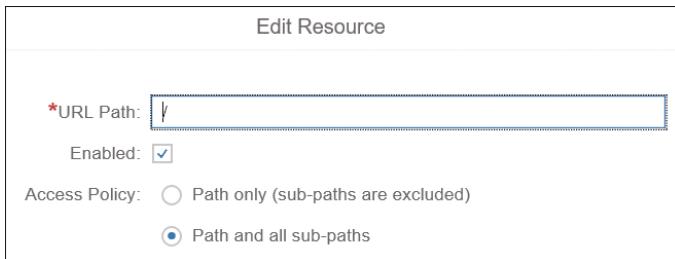


Figure 7.39 Edit Resource Dialog

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).

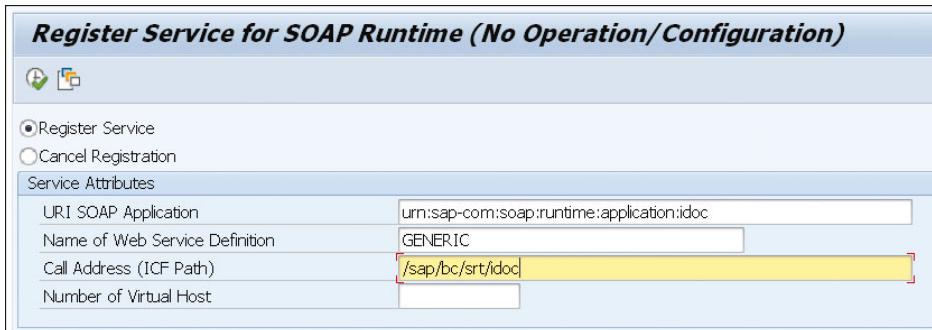


Figure 7.40 Registering the HTTP Service for IDoc Processing

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 WEO5, 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.

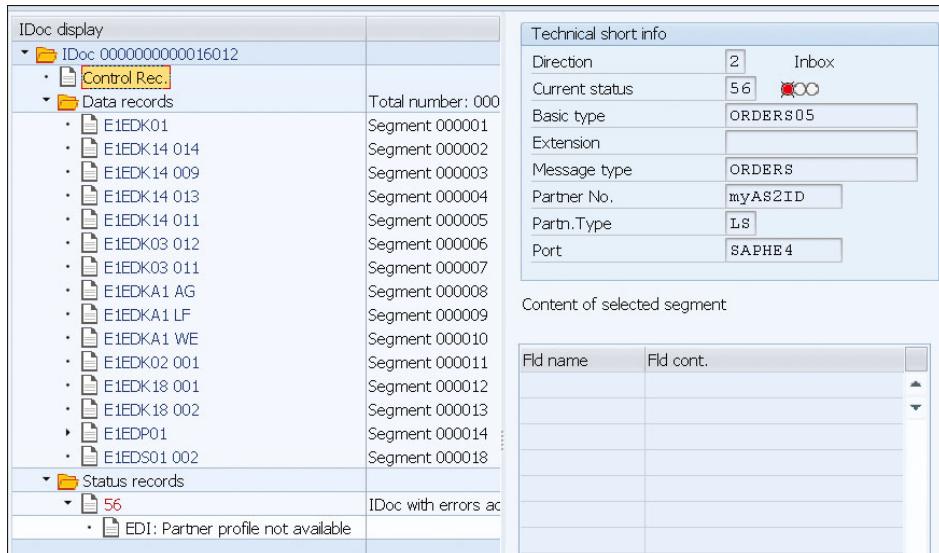


Figure 7.41 IDoc Monitoring in the Receiver System

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.

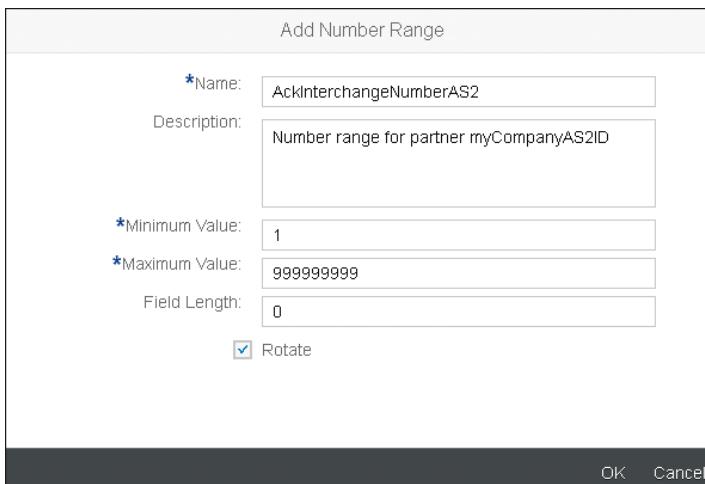


Figure 7.42 Defining the NRO

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.

*Create Acknowledgement:	Required
*Interchange Number:	Number Range
*Number Range:	AckInterchangeNumberAS2
<input type="checkbox"/> Exclude AK3 and AK4	

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.

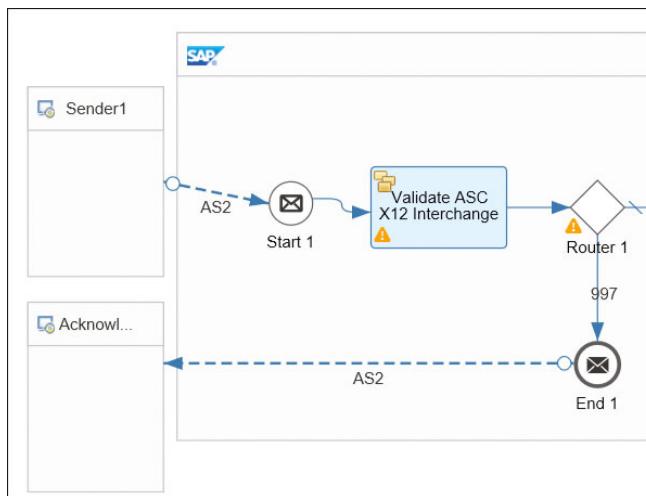


Figure 7.44 Adding the Receiver of the Acknowledgement

AS2	
	General Connection Processing Security MDN
RECIPIENT INFORMATION	
*Recipient URL:	<input type="text" value="http://11.11.11.11:8080/as2/HttpReceiver"/>
URL Parameters Pattern:	<input type="text"/>
*Proxy Type:	<input type="text" value="On-Premise"/>
Location ID:	<input type="text"/>
Authentication Type:	<input type="text" value="None"/>

Figure 7.45 Configuring the Connection Tab in AS2 Receiver Channel

Partner configuration	
<input type="button" value="New"/> <input type="button" value="Clone"/> <input type="button" value="Delete"/>	
<input checked="" type="checkbox"/> ABCCompany <input checked="" type="checkbox"/> mendelsontest <input checked="" type="checkbox"/> mycompany	<input type="button" value="Misc"/> <input type="button" value="Security"/> <input type="button" value="MDN"/> MDN URL: <input type="text" value="http://11.11.11.11:8080/as2/HttpReceiver"/> <small>Please start this URL with the protocol "http://" or "https://".</small>

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.

The screenshot shows the SAP Cloud Platform Integration configuration interface for an AS2 receiver channel. The top navigation bar includes tabs for General, Connection, Processing (which is selected), Security, and MDN. A blue link labeled "Externalize" is located in the top right corner. The main section is titled "MESSAGE INFORMATION" and contains the following fields:

- File Name: [empty input field]
- Message ID Left Part: [empty input field]
- Message ID Right Part: [empty input field]
- *Own AS2 ID: ABCCompanyID
- *Partner AS2 ID: myCompanyAS2ID
- *Message Subject: 997
- *Own E-mail address: my.mail@gmx.net
- *Content-Type: application/edi-x12
- Custom Headers Pattern: [empty input field]
- *Content Transfer Encoding: binary

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**.

Mapping Virtual To Internal System						
Status	Virtual Host	Internal Host	Check Result	Prot...	Back-end Type	Actions
	10.16.75.71:8080	10.16.75.71:8080	Reachable	HTTP	Non-SAP System	
	localhost:8080	localhost:8080	Reachable	HTTPS	ABAP System	
	vm1:8080	vm1:8080	Reachable	HTTPS	ABAP System	

Resources Accessible On 10.16.75.71:8080				
Enabled	Status	URL Path	Access Policy	Actions
<input checked="" type="checkbox"/>		/as2/HttpReceiver	Path only (sub-paths are excluded)	

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).

Transactions		News and updates			
Partner	Message details	Filter	Toggle refresh	Columns	Delete
Timestamp	Local stat... Partner	Message id		Payload	
4/24/18 2:59 PM mycompany	ABCCompany	mendelson_opensource_AS2-1524574794962-12@myCompanyAS2ID_ABCCompanyID		850 - Purchase Order - Technically incorrec...	
4/24/18 2:59 PM mycompany	ABCCompany	e9a2ff2b-f286-4d7-95c8-2f11ec6519@ABCCompanyID		ABCCompanyID_File	
4/24/18 3:52 PM mycompany	ABCCompany	mendelson_opensource_AS2-1524577973612-14@myCompanyAS2ID_ABCCompanyID		850 - Purchase Order - Technically incorrec...	
4/24/18 3:52 PM mycompany	ABCCompany	087390224-2f8e-4a3-be0f-120a7ab73c13@ABCCompanyID		ABCCompanyID_File	

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.

Message details					
186cf42e-5636-40f2-b7ce-d61f79b8e469@ABCCompanyID					
Date	Ref No	Sig...	Enc...	Sender	AS2 server
4/24/18...	186cf42e-5636-40f2-b7ce-d61f79b...	No s...	No e...	10.96.58.227	AHC/1.0
Log of this message instance		Raw data (unencrypted)		Message header	Transferred payload
ISA*00* *00* *SN*USS9010 *SN*myAS2ID *180424*0950*U*00403*00000004*0*T*>~ GS*FA*SAPPRT*SAPPRT*20180424*0950*1*X*004010~ ST*997*0001~ AK1*PO*000000001~ AK2*850*540000087~ AK5*A~ AK9*A*1*1*1~ SE*6*0001~ GE*1*1~ IEA*1*000000004~					

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.

The screenshot shows the SAP Cloud Platform Integration message monitoring interface. On the left, a table lists three messages under the heading 'Messages (3)'. The first message, 'X12_IDoc_Orders' (status: Completed, timestamp: Apr 24, 2018, 16:36:22), has a duration of 620 ms. The second message, 'X12_IDoc_Orders' (status: Completed, timestamp: Apr 24, 2018, 16:35:58), has a duration of 1 min 9 sec. The third message, 'X12_IDoc_Orders' (status: Completed, timestamp: Apr 24, 2018, 15:52:54), has a duration of 540 ms. On the right, a detailed view for the first message is shown. It includes the message ID 'X12_IDoc_Orders', last updated at 'Apr 24, 2018, 16:37:28', and tabs for Status, Properties, Logs, and Attachments. The Attachments tab is selected, displaying two entries: 'MDN Attachment' (text/xml, modified at Apr 24, 2018, ...) and 'Splitter Validation Error Document' (text/xml, modified at Apr 24, 2018, ...).

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

Date	Ref No	Sig...	Enc...	Sender	AS2 server
4/24/1...	087a9224-2f8e-4af3-bc0f-120a7ab73c13@ABCCompanyID	No s...	No e...	10.96.58.227	AHC/1.0

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

```
ISA*00* *00* *SN*USSU9010 *SN*myAS2ID *180424*0152*U*00403*00000006*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*I*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.

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	<input checked="" type="checkbox"/>

CODELIST : 353

Selected Codelist: 353

General Information

Identifier: 353

Name: Transaction Set Purpose Code

Code Values (1)

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.

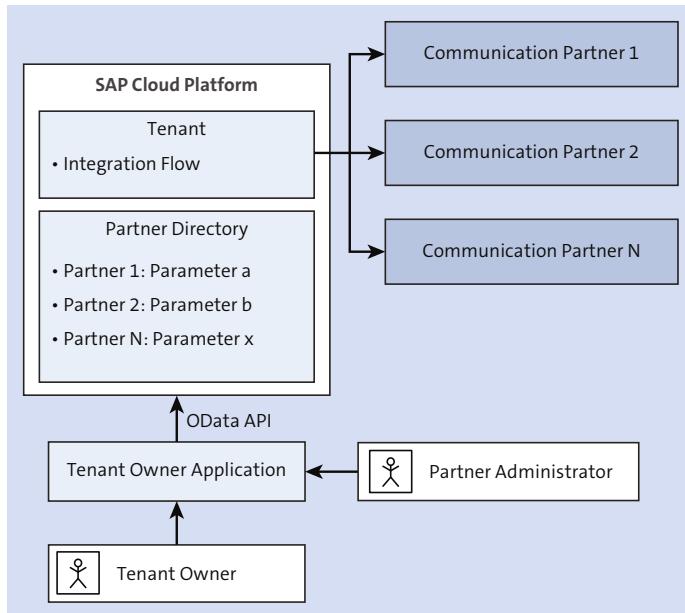


Figure 7.55 Usage of Partner Directory

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](#).

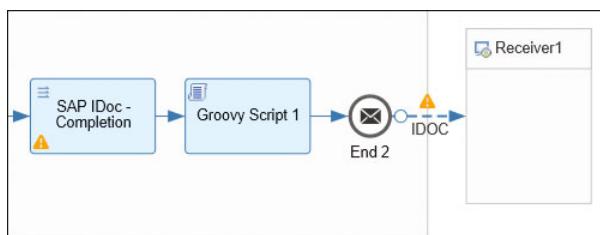


Figure 7.56 Adding a Groovy Script

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.ITAapiFactory;
def Message processData(Message message) {
    def service = ITAapiFactory.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 (`AuthGroupAdministrator`) 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](#)).

The screenshot shows the Postman interface with the following configuration:

- Method:** GET
- URL:** `https://<TMN-host>/api/v1`
- Authorization:** Basic Auth (selected)
- Headers:** A new header `X-CSRF-Token` is added with the value `Fetch`.
- Params:** None
- Send:** Send button
- Save:** Save button
- Authorization Tab Details:**
 - Type:** Basic Auth
 - Username:** User
 - Password:** (Redacted)
 - Show Password:**
- Notes:** A note states: "Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)"
- Buttons:** Preview Request button

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.

Body	Cookies (3)	Headers (12)	Test Results	Status: 200 OK	Time: 425 ms	Size: 5.73 KB
		<p>Cache-Control → no-cache,no-store,must-revalidate</p> <p>Connection → close</p> <p>Content-Length → 5333</p> <p>Content-Type → application/atomsvc+xml;charset=utf-8</p> <p>DataServiceVersion → 1.0</p> <p>Date → Sat, 28 Apr 2018 08:10:38 GMT</p> <p>Expires → Thu, 01 Jan 1970 00:00:00 UTC</p> <p>Pragma → no-cache</p> <p>Server → SAP</p> <p>Set-Cookie → JSESSIONID=4B78A9D89996707080C973A70AB674DA02ED7ADF7E7B4960AA740B8818701AC6; Path=/api; Secure; HttpOnly</p> <p>Strict-Transport-Security → max-age=31536000; includeSubDomains; preload</p> <p>X-CSRF-Token → F91E9FA98151760AA8C30B7EC544464E</p>				

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.

POST		https://<TMN-host>/api/v1/StringParameters	Params	Send	Save	
Authorization		Headers (3)	Body	Pre-request Script	Tests	
					Cookies Code	
Key	Value		Description	...	Bulk Edit	Presets ▾
<input checked="" type="checkbox"/> X-CSRF-Token	xxx					
<input checked="" type="checkbox"/> Accept	application/json					
<input checked="" type="checkbox"/> Content-Type	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](#)).

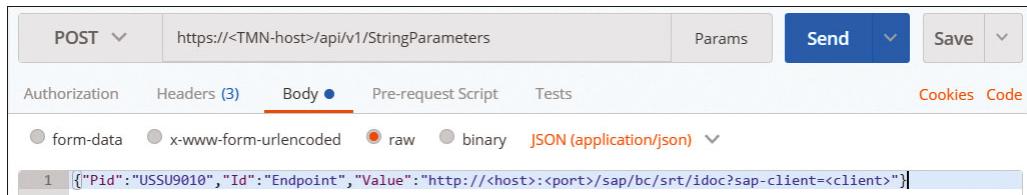


Figure 7.62 Body in the POST Request to Create the Endpoint Parameter

```

1  {
2   "d": {
3     "_metadata": {
4       "id": "https://<host>:<port>/sap/bc/srt/idoc?sap-client=<client>",
5       "id": "Endpoint",
6       "uri": "https://<host>:<port>/sap/bc/srt/idoc?sap-client=<client>",
7       "type": "com.sap.hci.api.StringParameter"
8     },
9     "Pid": "USSU9010",
10    "Id": "Endpoint",
11    "LastModifiedBy": "I...@",
12    "LastModifiedTime": "/Date(1524905240181)/",
13    "CreatedBy": "I...@",
14    "CreatedTime": "/Date(1524905240181)/",
15    "Value": "http://<host>:<port>/sap/bc/srt/idoc?sap-client=300"
16  }

```

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.

Chapter 8

SAP Cloud Platform Integration Operations

As an expert service provider, SAP ensures the support and maintenance of the hardware and software needed to keep SAP Cloud Platform Integration running optimally. However, there are a few operational tasks that need to be understood and performed by consumers themselves. This chapter explains the operational tasks to be performed by the customer to optimize running integration scenarios.

In [Chapter 4](#), [Chapter 5](#), and [Chapter 6](#), you learned how to implement simple and complex integration scenarios using SAP Cloud Platform Integration. When you've successfully developed, configured, and deployed integration flows in SAP Cloud Platform Integration, your business will be ready to use and trigger them during runtime.

As integration scenarios are being used at runtime, the customer needs to keep maintaining them to ensure that the organization continues to reap the benefits that these integration scenarios provide. For you as a customer, maintenance is an ongoing job that includes regularly monitoring integration flows and temporary data as well as managing users and security artifacts.

To gain access to the details of integration flow instances during runtime, you need to use SAP Cloud Platform Integration's monitoring capabilities. The monitoring features give customers insight into running integration flows to ensure they are correctly performing the job they were built for and are helping the business achieve its goals.

This chapter will focus on what is required to properly manage, track, and detect issues in running integration scenarios in your SAP Cloud Platform Integration tenant. Additionally, the features and tools to support the ongoing maintenance and operation jobs are explored. Let's start by giving you the big picture on the operational activities needed for your SAP Cloud Platform Integration tenant.

8.1 Operations: Overview

In traditional on-premise platforms, the customer is responsible for the entire maintenance and operation of the platform. For instance, this is the case for customers running an SAP Process Orchestration installation. SAP Process Orchestration is SAP Cloud Platform Integration's sister on-premise integration platform. The customer is solely responsible for maintaining the hardware and software, and monitoring the integration scenarios that are built on top of it.

SAP Cloud Platform Integration, as a cloud-based platform, brings a number of benefits to the customer when it comes to maintenance and operational aspects. As a service provider, SAP eliminates most of the headaches that come from the day-to-day support and maintenance of the SAP Cloud Platform Integration platform. Customers can thus use their energy on the core business at hand: developing, running, and monitoring integration scenarios.

Although most people are aware of the operational benefits of using a cloud-based platform such as SAP Cloud Platform Integration, not everyone has a clear view of the demarcation line between what operational activities are taken care of by SAP as a service provider and which ones still need to be covered by the customer. [Table 8.1](#) provides an overview of tasks to consider as part of SAP Cloud Platform Integration's day-to-day operations. The table also provides a responsibility matrix covering the tasks both performed at SAP and by a SAP Cloud Platform Integration customer.

Task	Responsibility	Description
Manage users and roles	Customer	Maintain users and assign roles to them to clearly define what actions they can perform on the SAP Cloud Platform Integration tenant.
Hardware and infrastructure maintenance	SAP Team	SAP is responsible for ensuring that SAP Cloud Platform Integration runs properly. This includes performing "health checks" in areas covering the operating system, memory, CPU, and so forth.

Table 8.1 Responsibility Matrix of Operational Tasks

Task	Responsibility	Description
Software updates and maintenance	SAP Team	Software updates are performed on a regular basis by SAP while ensuring near-zero downtime. The customer's tenant is upgraded automatically, and the newly installed features can immediately be used by all customers. The SAP team ensures that the updates performed don't negatively affect running integration scenarios.
SAP content update	SAP Team	Updates to the prepackaged content in the content catalog are regularly performed by SAP.
Update of consumed SAP content	Customer	Updates to the content packages in the content catalog need to be consumed by the customer, either manually or automated. Refer to Chapter 3 for more details.
Monitoring of run-time messages and bug fixes	Customer	Customers are responsible for monitoring, fixing, and maintaining their own integration scenarios and content.
Manage SAP keys	SAP Team	SAP-owned keys are updated by SAP before they expire. We'll look at this later in this chapter.
Manage security material	Customer	Customers deploy and maintain security artifacts needed for their integration flows. Certificates are a good example of such artifacts.

Table 8.1 Responsibility Matrix of Operational Tasks (Cont.)

SAP Cloud Platform Integration is constantly being developed and maintained; SAP releases updates for the platform about once a month. This short update cycle ensures that you continually receive new features and improvements to your SAP Cloud Platform Integration platform. During these updates, there is no downtime for customers. [Table 8.1](#) contains a number of activities that are performed by SAP. However, this chapter mainly focuses on those activities listed that fall under the responsibility of the customer.

Now that you're aware of the activities required to maintain your SAP Cloud Platform Integration tenant, this section will take you through each of the tools and features that are available to perform operational tasks. We'll cover the following aspects:

- Monitoring integration components and message processing
- Security material management
- Monitoring message stores, such as data stores and Java Message Service (JMS) queues
- Monitoring locks created during message processing
- Checking log files

Let's start with how to monitor your SAP Cloud Platform Integration tenant.

8.2 Monitoring Integration Content and Message Processing

After an integration flow has been deployed to the SAP Cloud Platform Integration tenant, it's time to monitor all running integration artifacts to make sure that they are correctly doing the job for which they were built. This section will focus on looking at what is required to properly manage, track, and detect issues on running integration flows. The monitoring of deployed artifacts is part of the ongoing job of maintaining and improving your organization's IT environment. Besides monitoring the runtime artifacts deployed on your tenant, this section also focuses on monitoring the message processing in the cluster's runtime.

Open the Web UI's homepage at <http://<server>:<port>/itspaces>. The monitoring screens can be accessed via the **Monitor** menu item on the left menu, as indicated in [Figure 8.1](#).

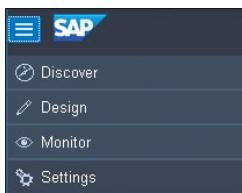


Figure 8.1 Accessing SAP Cloud Platform Integration's Monitoring Features

After clicking the **Monitor** tab, you're taken to a screen with a number of tiles displaying various statistics, numbers and statuses pertaining to the messages that have been processed by the runtime engine of SAP Cloud Platform Integration. [Figure 8.2](#) presents an overview of the main monitor screen. From this screen, also known as the monitoring dashboard, you can access the different monitors available for integration content and message monitoring.

A tile is a block in a page that filters messages or artifacts corresponding to a particular status (e.g., completed messages). As shown in [Figure 8.2](#), the screen contains six main tile groupings, namely:

- **Monitor Message Processing**
- **Manage Integration Content**
- **Manage Security**
- **Manage Stores**
- **Access Logs**
- **Manage Locks**

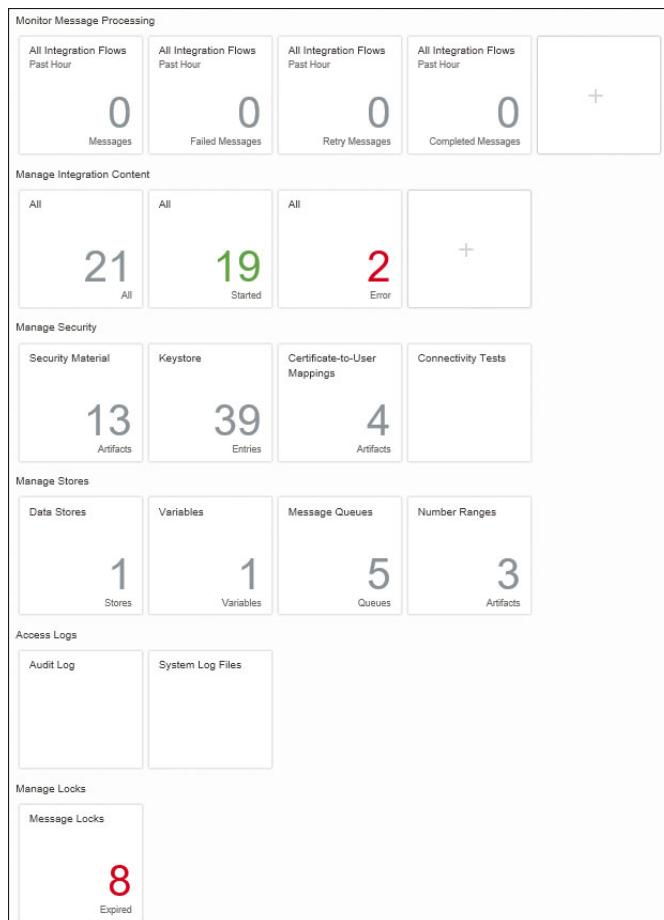


Figure 8.2 Overview of the Monitoring Dashboard

Let's first explore the **Manage Integration Content** section's set of tiles because we need to understand the monitoring options for the integration content to relate them to the monitoring of the message processing.

8.2.1 Manage Integration Content

The tiles in the **Manage Integration Content** section of the screen display the statuses of the different integration artifacts that have been deployed on the tenant. Deployed integration artifacts include, for instance, integration flows and OData services. Referring to the screen shown in [Figure 8.2](#), you can see that a total of 21 integration artifacts have been deployed (first tile in the **Manage Integration Content** section). Out of those 21, it shows that 19 have successfully started (second tile), and 2 have failed (third tile). You can click on the second tile to view all integration artifacts that have been successfully deployed on your SAP Cloud Platform Integration tenant, as shown in [Figure 8.3](#).

The screenshot shows the SAP Cloud Platform Integration Content Monitor interface. At the top left, there is a breadcrumb navigation: Overview / Manage Integration Content. Below the header, there is a search bar labeled "Integration Content (15)" with a placeholder "Filter by Name or ID" and a magnifying glass icon. To the right of the search bar are three icons: a refresh symbol, a gear symbol, and a copy symbol.

Name	Status
ValueMappingForOrders	Started
Value Mapping	
Book Integration Flow with Splitter	Started
Integration Flow	
AS2Sender	Started
Integration Flow	
Splitter_in_local_process	Started
Integration Flow	
InboundFlow	Started
Integration Flow	
OutboundFlow	Started
Integration Flow	
Transaction Management with JDBC	Started
Integration Flow	
Transaction Management for JMS	Started
Integration Flow	
Exceptions	Started
Integration Flow	
XSLT_Mapping_Test	Started
Integration Flow	
Variables	Started
Integration Flow	

On the right side of the table, there are four buttons: **Restart**, **Undeploy**, and **Download**. Below the table, there is a summary section for the selected artifact, "Book Integration Flow with Splitter". It shows the deployment details: Deployed On: Jan 25, 2018, 16:16:28, ID: Book_Integration_Flow_with_Splitter, Deployed By: [redacted], Version: 1.0.0. There are also tabs for **Endpoints**, **Status Details**, **Artifact Details**, and **Log Configuration**. The **Status Details** tab contains a message: "The Integration Flow is deployed successfully." The **Artifact Details** tab has links to "Monitor Message Processing" and "View Integration Flow". The **Log Configuration** tab includes a dropdown menu for "Log Level" with options "Info" and "Error".

Figure 8.3 Manage Integration Content Monitor

As depicted in [Figure 8.3](#), the monitor presents the deployed artifacts in a table on the left side of the monitor with status and artifact type. [Table 8.2](#) contains a detailed description of those attributes and the possible values.

Attribute	Description
Name	The name of the integration artifact that has been deployed.
Type	The type of integration content, for example, Value Mapping , Integration Flow , or OData service .
Status	<p>The deployment status of the integration artifact:</p> <ul style="list-style-type: none"> ■ Starting: The integration artifact is starting, but it can't be executed yet. ■ Started: The integration artifact is ready to be used. ■ Error: The integration artifact is in error state; it needs manual action, for example, a change of the artifact and redeployment. ■ Stopping: The artifact is stopping, for example, after undeployment. It can't be executed anymore.

Table 8.2 Attributes of the Integration Artifact on the Left Side of the Monitor

The content of the table is sorted by the time of the last deployment, which means that the artifact deployed at the latest is shown on top. To customize the list of displayed artifacts, you can either search for specific artifacts by artifact name or ID, or you can filter the table content by attributes such as status or artifact type. To sort or filter the content of the table, select the **Table Settings** icon  at the top of the table. On the subsequent screen, you can define how the table entries are to be sorted by specifying an attribute and whether the entries are to be sorted for that attribute in ascending or descending order. You can also filter the table entries for certain attributes.

On the right side of the monitor the details of the selected artifact are shown. At the top, the most important details about the deployed integration artifact are provided, such as ID, version, and who deployed it when. The description of those attributes is summarized in [Table 8.3](#).

Attribute	Description
Deployed On	Specifies date and time at which the last deployment was performed.

Table 8.3 Attributes of the Integration Artifact on the Top of the Right Side of the Monitor

Attribute	Description
Deployed By	Specifies the user who deployed the artifact.
ID	The technical ID of the integration artifact. This ID is necessary, for example, when searching for errors in the log files.
Version	The version number of the deployed integration artifact.

Table 8.3 Attributes of the Integration Artifact on the Top of the Right Side of the Monitor (Cont.)

The following actions, among others, can be performed on each artifact:

■ **Undeploy**

Remove an artifact that was previously deployed in the tenant. To undeploy an integration artifact, select the artifact in the table on the left of the monitor, and choose **Undeploy**.

■ **Restart**

Restart if there was a problem with the integration artifact that was solved in the meantime, such as the credentials weren't deployed on the tenant. Such errors don't need a redeployment; they just need to restart the integration artifact. If changes in the integration artifact were required to solve the error, the integration artifact needs to be redeployed from design view, and restart isn't sufficient then. Note that the **Restart** action isn't always available; it depends on the status of the artifact and the type of error.

■ **Download**

Download the content of an integration artifact. The artifact is then stored on the local machine in form of a Java Archive (JAR) file. Downloading integration content can be useful as a backup method. The backed-up data can then, for instance, be reimported or transported into another SAP Cloud Platform Integration tenant. Note that read-only integration artifacts can't be downloaded.

The actions available on the top-right corner of the monitor depend on the status of the artifact and whether it's a read-only artifact.

Below the header information, the view is divided into different sections. You can navigate between them using the tabs, which are shown on top:

■ **Endpoints**

The HTTP-based sender adapters, such as SOAP or HTTP, expose endpoints, which

can be called to start the runtime processing of a specific artifact. In the **Endpoints** section, all accessible endpoints are listed, which are exposed by the selected artifact. We already used this option in some of the sample scenarios to retrieve the URL to be called to trigger the integration flows using a SOAP sender adapter.

■ Status Details

This section provides the overall status of the integration artifact. For started integration artifacts, you'll see the message that the integration artifact is deployed successfully (refer to [Figure 8.3](#)). But if the integration artifact is in error status, the error details are provided as depicted in [Figure 8.4](#). You also have the option to download the log by clicking the **Download** icon , for example, to attach it to a support ticket.



The screenshot shows a "Status Details" section. A red box highlights the message "The Integration Flow is not operational.". Below it, under "Error Details", there is a download icon and the error stack trace: "[CAMEL][IFLOW][EXCEPTION] : java.lang.RuntimeException: Soap 1.1 endpoint already registered on address /Book_Demo".

Figure 8.4 Error Details in the Manage Integration Content Monitor

■ Artifact Details

This section provides two important links. When selecting the **Monitor Message Processing** link, the message monitor opens and shows all messages processed for the selected artifact in the last hour. Note that this link isn't available for value mappings. The second link, **View Integration Flow**, opens the selected artifact in its modeler in read-only mode. You can, for example, use this view to explore the configured steps and adapters in an integration flow and check the properties defined. You can't change the configuration and redeploy the artifact from this view, however. The view is meant for monitoring only. If changes to the integration artifact are required, you need to change to the design view and adjust it there.

■ Log Configuration

You can configure different log levels for the integration flows and OData services deployed on the tenant in the **Log Configuration** section. This section isn't available for value mappings.

Let's take a more detailed look at the log levels available for integration flows and the consequences of changing them.

8.2.2 Log Configuration

To monitor the integration flows most efficiently in all phases of developing a new scenario, different log levels are available. With this configuration option, you can, for example, monitor integration flows during development and test phases with a high log level, which provides details about runtime execution in all steps and adapters, and observe payloads and headers after each step. Later, when the integration flow runs in a productive scenario, you won't need such detailed logs anymore.

As depicted in [Figure 8.5](#), the following **Log Levels** are provided by SAP Cloud Platform Integration

- **None**

No message processing log (MPL) is written.

- **Info**

This is the default log level set when a new integration flow is deployed . With this log level, only the most important information about the runtime execution of a message is logged, such as start and stop time and the overall status. Nevertheless, if the message encounters an error during runtime execution, the last steps are logged in detail in order to analyze the error. This is the log level you should use for integration flows running in a productive scenario.

- **Debug**

All flow steps and adapter calls are logged in detail, even on successful execution of a message. This log level is mostly used during development and testing of a new scenario. This log level should not be used in productive scenarios due to the negative impact on performance and data volume stored in the database.

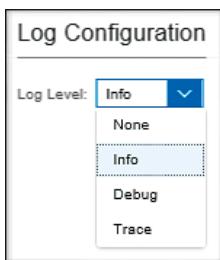


Figure 8.5 Log Levels for Log Configuration

- **Trace**

This is the most detailed log level available. With this log level, payloads and headers are logged after each processing step. **Trace** is activated only for a short period to analyze issues in scenario execution in detail. The maximum time the log level **Trace** can be activated for is 10 minutes to avoid overloading the database with too much data. After this time, the log level is reset automatically to the log level that was configured before activating **Trace**.

During the first deployment of a new integration flow, the log level is set to **Info**. Then, you can configure the log level as desired in the **Manage Integration Content** monitor. Subsequent deployments of the same integration flow don't reset the log level; the setting is kept.

As soon as you change the log level in the **Log Configuration** dropdown, it's active; no additional save action and no restart of the integration flow is required. The next MPL of a message is written with the new log level.

Log and Trace Configuration: Additional Features

The logging and tracing feature will be extended by additional features, such as a dedicated log level adapter trace. The most recent features are documented in the "Troubleshooting Message Processing in the CPI Web Application" blog in the SAP Community (www.sap.com/community.html).

Now you know how to monitor your integration content and how to define the log level for deployed integration flows. With this information, we can dive into the details about monitoring the message processing in the SAP Cloud Platform Integration runtime.

8.2.3 Monitor Message Processing

As depicted earlier in [Figure 8.2](#), within the monitoring dashboard, the top section of the screen includes a list of tiles that all relate to message processing. After clicking on any of the tiles in this section, you're taken to a new page, which looks like the one shown in [Figure 8.6](#).

The screenshot shows the SAP Cloud Platform Integration Monitor interface. At the top, there are filters for Time (Past 24 Hours), Status (All), Artifact (Book Integration Flow with Splitter), and an ID search bar. Below the filters, a message list titled 'Messages (8)' shows eight entries for the 'Book Integration Flow with Splitter'. Each entry includes the artifact name, status (e.g., Retry, Completed), timestamp, and duration. To the right of the message list, detailed logs for the selected artifact ('Book Integration Flow with Splitter') are displayed. The logs tab is active, showing an error message: 'An error occurred during message processing and a retry has been started automatically.' It also shows processing time (4 sec 814 ms) and error details related to a MailConnectException. Below the logs, the Properties tab is shown, displaying message ID, correlation ID, artifact name, artifact ID, and artifact type. The Logs tab is also present, showing a table of log runs with columns for number, started at, duration, log level, process ID, and status.

Figure 8.6 Monitor Message Processing

By default, the screen displays two views:

- A left view, with a list of messages matching the tile configuration. It's possible to select any of the listed messages to further view its log details in the right view.
- A right view, with the message details of the selected entry on the left side of the screen. **Status**, **Properties**, and **Logs** are provided and offer further navigation to the MPL, MPL attachments, and the integration flow.

As shown in [Figure 8.6](#), it's also possible to further filter the messages by changing the filtering settings. Messages can be filtered based on **Time**, **Status**, **Artifact**, **Message**, **Correlation**, or **Application ID**. [Table 8.4](#) provides a description of these filter attributes.

Filter Name	Description
Time	The date and time at which the message was triggered.

Table 8.4 List of Filter Attributes in the Monitor

Filter Name	Description
Status	The status of the MPL entry in the monitor. Possible values include the following: <ul style="list-style-type: none"> ■ All ■ Failed ■ Retry ■ Completed ■ Processing ■ Escalated
Artifact	The name of the integration flow or OData service as created in your design workspace.
Message ID	A unique identifier for a specific MPL across the tenant. This attribute is often used in a filter when troubleshooting a specific integration flow instance with a known message ID.
Correlation ID	A unique identifier for a specific group of messages that belong together. This attribute is relevant for scenarios where different MPLs belong to one end-to-end process, for example, in a JMS scenario where a message is stored in a JMS queue using one MPL and consumed in a second MPL.
Application ID	Only relevant when the SAP_ApplicationID header element has been specified using either the content modifier or script step.

Table 8.4 List of Filter Attributes in the Monitor (Cont.)

On the left side of the screen, you get the overall number of messages matching the selection criteria. Using the arrows at the top of the **Messages** table, you can navigate between the different pages of the monitor to check all the messages available ([Figure 8.7](#)).

**Figure 8.7** Navigation in the Message Monitor

For each message entry on the left view, the processing time, the status of the message, and the configured log level are displayed. Upon selection of any entry in the left view, further details in respect to the message are displayed on the right side of the screen:

- At the top, the time of the last update in the log for this message is shown.
- The **Status Details** section provides the overall status of the message processing, together with the overall processing time. If an error occurred during processing of the message, the error details are shown ([Figure 8.8](#)).

The screenshot shows a user interface for managing messages. At the top, there are three tabs: 'Status' (which is selected and underlined in blue), 'Properties', and 'Logs'. Below the tabs, a red rectangular box contains the text 'Message processing failed.' In the main content area, the text 'Processing Time: 17 ms' is displayed. Under the heading 'Error Details', there is a stack trace:

```
com.sap.it.rt.adapter.as2.api.exception.InboundProcessingException:  
com.sap.it.rt.adapter.as2.api.exception.ChannelConfigException: Signed  
MDN requested, but no private key was configured in AS2 sender channel  
for MDN signing
```

Figure 8.8 Status Details for an Erroneous Message

- In the **Properties** section, the message ID, the correlation ID, and the details of the deployed integration artifact are displayed, including the name, the ID, and the type of the artifact, together with a link that opens the artifact in display mode.
- The **Logs** section contains information about the **Log Level** and the **Process ID** for the message processing. The **Process ID** indicates the ID of the runtime node the message was executed on; this information is important to find the related system log if an error needs to be analyzed in detail ([Section 8.5.2](#)). The **Log Level**, for example **Debug**, is shown as a link, and, if selected, it will open the detailed steps of the message processing in a new window. There the processing steps are shown in a table at the top, and the message traversal path is depicted in the model below. We'll discuss this in more detail later in this section. The textual representation of the **Message Processing Log** can be accessed using the **Open Text View** link.
- If MPL attachments are created during message processing, an additional tab, **Attachments**, is available. There the MPL attachments are listed in a table showing the name of the attachment, the type, and the time the attachment was modified last. MPL attachments can, for example, be created during message processing using the script step.

Let's check out the available sections in more detail.

Properties

In the **Properties** section, the message ID for this specific message processing and the properties of the deployed integration flow are listed. [Table 8.5](#) describes each of these attributes.

Attribute	Description
Message ID	ID of the selected MPL that can be used as search criteria in message monitoring and in log files.
Correlation ID	ID of a group of MPLs that belong together and can be used as search criteria in message monitoring.
Artifact Name	Shows the name of the integration artifact.
Artifact ID	The technical artifact ID of the integration flow or the OData project.
Artifact Type	Can be either Integration Flow or OData Project .

Table 8.5 List of Attributes in the Properties Section

If the artifact for this message is still deployed on the tenant, the name of the artifact is shown as a link (refer back to [Figure 8.6](#)). Selecting the link opens the artifact in read-only mode ([Figure 8.9](#)). You can use this view to explore the configured steps and adapters in the integration flow and check the defined properties.

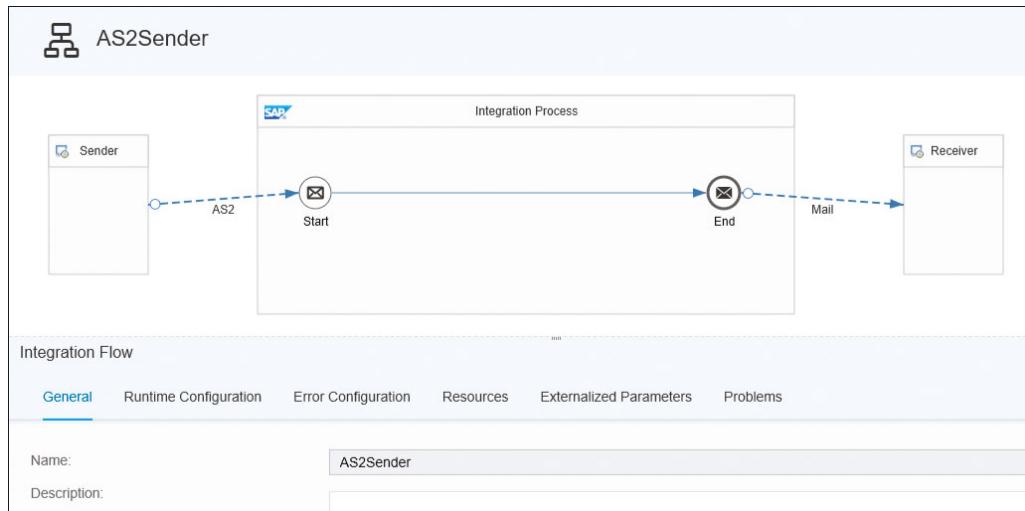


Figure 8.9 Integration Flow in Read-Only Mode

Note that it isn't possible to change the configuration and redeploy the integration flow in this view. The view is meant for monitoring only. If changes to the integration flow are required, you need to change to the design view and adjust the integration flow there.

Logs

Let's take a deeper look at the **Logs** section (refer to [Figure 8.6](#)). There the MPL can be explored in visual and in textual representation.

The **Logs** tab may contain a table listing several message processing runs, as shown previously in [Figure 8.6](#). This is the case, for example, when JMS queues are used for temporary message storage, such as in the XI or AS2 adapter, or if the message is retried by the JMS sender adapter. Then, for each message processing run, one line appears in the table. However, in most of the scenarios, the **Logs** tab looks similar to [Figure 8.10](#). Because only a single message processing run is executed, the data isn't provided in a table.



Figure 8.10 Logs Tab for a Single Message Processing Run

The log level is displayed that was used to write the MPL. Remember that we learned which log levels exist and how to set the log level in [Section 8.2.1](#). The **Log Level** value is shown as a link in the **Logs** tab. As depicted in [Figure 8.11](#), selecting the link opens a new window that provides a table with all the processing steps and the graphical model of the integration flow. Note that the details shown depend on the log level being used; we'll elaborate more on this in the next couple of pages.

As depicted in [Figure 8.12](#), for erroneous messages, the error is highlighted in the table with an **Error** icon ⓘ, which shows the error details when clicked.

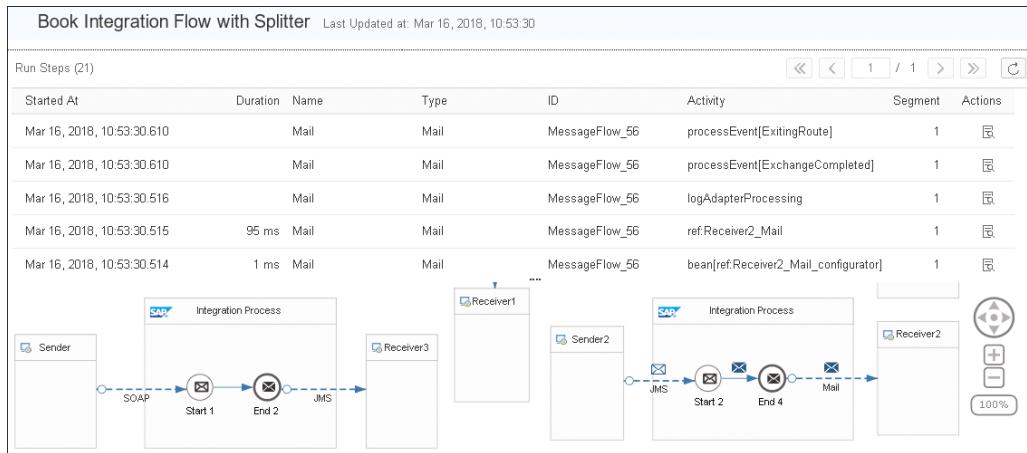


Figure 8.11 Visual Representation of the Message Processing Log

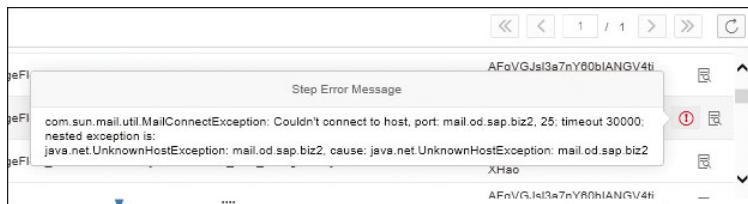


Figure 8.12 Visual Representation of the Message Processing Log for an Erroneous Message

The table at the top of the window contains all executed processing steps sorted by execution time, starting with the last step executed. The different columns of the table are described in detail in [Table 8.6](#).

Attribute	Description
Started At	Time the step was started.
Duration	Overall time of the execution of the step.
Name	Name of the flow step or adapter channel.
Type	Type of the flow step or adapter, for example, Mail.
ID	ID of the flow step or adapter in the integration flow model. All steps with the same ID belong to the same flow step or adapter in the integration flow.

Table 8.6 Description of the Columns in the Message Processing Log

Attribute	Description
Activity	Detailed activity executed by the flow step or adapter. There are typically several activities executed by one flow step or adapter, which are grouped by the model step ID.
Segment	ID of the processing segment in the runtime. This value gets especially interesting if multiple branches are executed for one message processing because then multiple processing segments are available. This is the case, for example, if splitter or multicast patterns are used.
Actions	Actions for the single step. Currently you can show the step details using the Show Step Details  icon.

Table 8.6 Description of the Columns in the Message Processing Log (Cont.)

At the bottom of the screen, the model of the integration flow is shown with the traversal path for the message processing. As depicted in [Figure 8.11](#), envelope icons are shown in the model  to indicate the path this specific message has taken during processing. For steps causing an error, the envelope is shown in red .

Icons for Path Traversal Depend on the Log Level

Note that the filled envelopes are shown only if log level **Trace** was used for the message processing, indicating that a full trace with payloads and headers is available in the details. If trace level **Info** or **Debug** was used, no payload data is available in the detailed log data. In this case, the envelopes indicating the traversal path aren't filled. Blue envelopes  show the traversal path for successfully executed steps, and a red envelope  points out that there was an error executing this step.

When you select the **Show Step Details** icon  for one specific processing step in the table, a new window opens. As depicted in [Figure 8.13](#), the screen shows multiple tabs for the selected processing step. The **Log** tab contains a table with key/value pairs written by this processing step.

In the **Configuration** tab, the defined settings in the flow step or adapter can be explored. The configuration tabs available for the flow step or adapter are shown in read-only mode. In [Figure 8.14](#), the **Configuration** tabs for the **Mail** receiver adapter are **General**, **Connection**, and **Security**. You can navigate between the tabs and discover all configurations defined for this flow step or adapter.

Book_Integration_Flow_with_Splitter - Mail						
Log	Configuration	Message Content				
<table border="1"><thead><tr><th>Key</th><th>Value</th></tr></thead><tbody><tr><td>Description</td><td>Sending mail message to receiver</td></tr></tbody></table>			Key	Value	Description	Sending mail message to receiver
Key	Value					
Description	Sending mail message to receiver					

Figure 8.13 Log of a Single Processing Step

Book_Integration_Flow_with_Splitter - Mail Started at: Jun 11, 2018, 17:53:42.010

Log	Configuration	Message Content																		
<table border="1"><thead><tr><th>General</th><th>Connection</th><th>Security</th></tr></thead><tbody><tr><td>Name: <input type="text" value="Mail"/></td><td colspan="2"></td></tr><tr><td colspan="2">CHANNEL DETAILS</td><td>ADAPTER DETAILS</td></tr><tr><td>Direction: <input type="text" value="Receiver"/></td><td>Adapter Type: <input type="text" value="Mail"/></td><td></td></tr><tr><td>System: <input type="text" value="Receiver2"/></td><td>Transport Protocol: <input type="text" value="SMTP"/></td><td></td></tr><tr><td>Description: <input type="text"/></td><td>Message Protocol: <input type="text" value="None"/></td><td></td></tr></tbody></table>			General	Connection	Security	Name: <input type="text" value="Mail"/>			CHANNEL DETAILS		ADAPTER DETAILS	Direction: <input type="text" value="Receiver"/>	Adapter Type: <input type="text" value="Mail"/>		System: <input type="text" value="Receiver2"/>	Transport Protocol: <input type="text" value="SMTP"/>		Description: <input type="text"/>	Message Protocol: <input type="text" value="None"/>	
General	Connection	Security																		
Name: <input type="text" value="Mail"/>																				
CHANNEL DETAILS		ADAPTER DETAILS																		
Direction: <input type="text" value="Receiver"/>	Adapter Type: <input type="text" value="Mail"/>																			
System: <input type="text" value="Receiver2"/>	Transport Protocol: <input type="text" value="SMTP"/>																			
Description: <input type="text"/>	Message Protocol: <input type="text" value="None"/>																			

Figure 8.14 Configuration Details of a Single Processing Step

The **Message Content** tab, if available, provides the payload and header data written during execution of the flow step or adapter (Figure 8.15). The screen is divided into two sections, the **Header** section at the top and the **Payload** section at the bottom. In the **Header** section, all the headers available during execution of this step or adapter are shown. In Figure 8.15, for example, you see all the headers set by the mail receiver adapter. In the **Payload** section, the corresponding payload sent to the mail server is shown.

Book_Integration_Flow_with_Splitter - Mail Started at: Jun 11, 2018, 17:53:42.010

[Log](#) [Configuration](#) [Message Content](#)

Message after Receiver Adapter [Download](#)

Header

Name	Value
Content-Transfer-Encoding	quoted-printable
Content-Type	text/plain; charset=UTF-8
From	mail@mail.sap.hana.ondemand.com
Message-ID	<1839525528.11.1528732422019@mail.od.sap.biz>
MIME-Version	1.0
SAP_MessageProcessingLogID	AFsemwW2pw1Lm-C09in3qOD6Ph1b
Subject	Message to Receiver2, CorrID AFsemwlser-J29ZVB4QDyJG44Zid
To	mandy.krimmel@sap.com

Payload

Encoding: UTF-8

```
From: mail@mail.sap.hana.ondemand.com
To: mandy.krimmel@sap.com
Message-ID: <1839525528.11.1528732422019@mail.od.sap.biz>
Subject: Message to Receiver2, CorrID AFsemwlser-J29ZVB4QDyJG44Zid
MIME-Version: 1.0
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: quoted-printable
SAP_MessageProcessingLogID: AFsemwW2pw1Lm-C09in3qOD6Ph1b
SAP_MplCorrelationId: AFsemwlser-J29ZVB4QDyJG44Zid
SAP_PregeneratedMpId: AFsemwUShTmf2c9p4i-wTh5sNFb
```

Figure 8.15 Message Content Sent by the Mail Receiver Adapter

With these details, it's possible to know exactly how the headers and payload changed during execution of this specific step. This can especially be useful if you need to understand a specific error situation in the scenario execution. Using the **Download** link, you have the option to download the payload and the headers.

Message Content Not Available?

Note that the message content for a specific step is only available if log level **Trace** was defined for this message processing.

Details Available Depend on the Defined Log Level

Depending on the log level set for the integration flow, different details are available in the MPL:

- **None**

The MPL isn't available at all.

- **Info**

Only the traversal path is available in the model combined with the **Configuration** tab. If the message processing ended with the status **Failed**, the last processing steps are listed, and the **Properties** and **Configuration** tabs are shown in the details of the processing steps with the respective data.

- **Debug**

The traversal path is available in the model. In addition, all processing steps are listed, and the **Properties** and **Configuration** tabs are shown in the details of the processing steps with the respective data.

- **Trace**

The traversal path is available in the model. In addition, all processing steps are listed, and the **Properties**, **Message Content**, and **Configuration** tabs are shown in the details of the processing steps with the respective data.

In addition to the graphical representation of the MPL, a textual representation is available that contains the details of all executed steps during processing of the message in one single file. You can select the **Open Text View** link in the message processing monitor (refer back to [Figure 8.6](#)) to open the textual log view.

As depicted in [Figure 8.16](#), the main attributes of the message processing are shown, such as the start and stop time, overall status, and message guide. If log level **Info** was set for this message processing, no additional details are shown, and the single steps and adapters aren't logged.

If there was an error when processing the message, a more detailed log is available for the last steps before the error occurred ([Figure 8.17](#)), even for log level **Info**. This log provides integration developers enough information to easily spot problems or errors at hand. Aside from exploring the contents of the log, you can also download the log using the **Download** button at the top-right corner of the page.

Note that for log level **Debug**, all steps and adapters are always logged, independent of whether the message was processed successfully or ended with an error.

[Download](#)

Artifact Name: Book Integration Flow with Splitter	Status: Completed	Processing Time: 54 ms
Last Updated at: Mar 16, 2018, 08:38:26	Log Level: Info	

Log

```

Message Processing Log:
StartTime      = Fri Mar 16 07:38:26.296 UTC 2018
StopTime       = Fri Mar 16 07:38:26.349 UTC 2018
OverallStatus  = COMPLETED
MessageGuid   = AFqrdfHL4T7npa0mEdPBpUAEMZ66U
ChildCount    = 0
ChildrenCounter = 14
ContextName   = Book_Integration_Flow_with_Splitter
CorrelationId = AFqrdfHKabs5ofD2vOWepxFNm0TsT
IntermediateError = false
Node          = vsa3925238
ProcessId     = 171e30214e302bfa80d912983df432436cf349ea
SenderId       = Sender_SOAP

```

Figure 8.16 Log of a Completed Message

Artifact Name: AS2Sender	Status: Retry	Processing Time: 8 d 21 h
Last Updated at: Mar 16, 2018, 13:34:10	Log Level: Info	

Log MDN Attachment

```

com.sun.mail.util.MailConnectException: Couldn't connect to host, port: mail
nested exception is:
java.net.UnknownHostException: mail.od.sap.biz2, cause: java.net.UnknownHost

```

The message processing log is truncated. Only the first and the last 28 procurements are omitted.

Message Processing Log (last processing run):

```

StartTime      = Fri Mar 16 12:34:10.384 UTC 2018
StopTime       = Fri Mar 16 12:34:10.392 UTC 2018
OverallStatus  = RETRY
MessageGuid   = AFqf-z_1BMwSTZ5DPMUwJlax5_C
ChildCount    = 0
ChildrenCounter = 10
ContextName   = AS2Sender
CorrelationId = AFqf-z_GaPDwv1TFQPsk_UZWmbWo
IntermediateError = false
Node          = vsa3885215
ProcessId     = b0828a3e3760a2c918e559f9302a95dc290e3283

```

Segment:

```

Entering Camel route Process_1:
StartTime      = Fri Mar 16 12:34:10.385 UTC 2018
ChildCount    = 1
ModelStepId   = MessageFlow_4

```

Exchange ID - vsa3885215-46121-1521109630151-19-2 created in Endpoint[jms://AS2_AS2Sender_evelName=CACHE_CONNECTION&chunkReceiveTimeout=10000&chunking=true&concurrentConnectionFactory=&connectionFactory=&defaultTaskExecutorType=Threadpool&maxReceiveTimeout=5000&transacted=true&transactionManager=&jmsTransactionManager=&]
StartTime = Fri Mar 16 12:34:10.385 UTC 2018
Status = PROCESSING
ChildCount = 2
ModelStepId = MessageFlow_4

Figure 8.17 Log of a Message in Error Status

As stated earlier, the log details are often referred to as the *MPL*. The textual MPL represents a well-structured tree of log information. The MPL structure consists of two main components: the top and bottom sections. The top section contains the properties and metadata of the message as a whole, similar to a header section. This is the section available with log level **Info**. The bottom section contains one or more branches with entries for each step of a particular integration flow. This section is only available with log level **Debug** or if an error occurred during processing of the message.

The MPL structure always includes a predetermined set of attributes, which are listed in [Table 8.7](#) and [Table 8.8](#). Becoming familiar with these attributes will enable you to better understand the logs and therefore improve your ability to troubleshoot issues. Note that not all properties listed in the tables are present in every MPL. Depending on the status of a message, only a few properties might be shown in the MPL. For instance, the **Error** attribute is only present if a message is in **Failed** status, as you can see in [Figure 8.17](#).

Property	Description
Error	Specifies the error of a particular step in the integration flow. Note that this attribute is displayed at the top of the log in red color and is only available in the MPL of messages that are in a failed state.
StartTime	The time that message processing started.
StopTime	The time that message processing ended.
OverallStatus	The status of the message processing, as discussed earlier in the chapter (see Table 8.4).
ChildCount	The serial number of the current processing step. For the overall overview at the top of the MPL, this number is always 0.
ChildrenCounter	The total number of message processing steps executed.
ContextName	The name of the integration flow.

Table 8.7 Properties Contained in the MPL Header

Property	Description
CorrelationId	The ID that identifies correlated messages. Messages can be correlated, for example, when different integration flows on the same tenant communicate with each other, for example, using JMS queues. A correlation ID is a base64-encoded ID that is generated in this case by the first integration process and stored in the message header. As part of the message header, the CorrelationId is then propagated across all related integration flows.
CustomHeaderProperties	Shown if you specify your own headers via the script application programming interface (API) in the script step.
IntermediateError	If, during message processing, an error occurred, or message processing needed more than one minute, the value of this property is set to true.
LastErrorModel-StepId	ID of the step that caused the error. Note that this attribute is only available in the MPL of messages that are in a failed state.
MessageGuid	A key that identifies the message uniquely in the database.
Node	The host name of the runtime node that processed the message.
Process ID	The ID of the runtime process that executed the message. This information is useful to identify the system log file, in which details to the message processing are logged.
ReceiverId	The name of the receiver as configured in the integration flow using the header SAP_Receiver, for example, in a content modifier or script step.
SenderId	Specifies the name of the sender as configured in the integration flow using the header SAP_Sender, for example, in a content modifier or script step.
Id	Displayed in the MPL header if the ID has been defined using the header SAP_ApplicationID in the integration flow, for example, in a content modifier or script step.
MessageType	Displayed in the MPL header if the message type has been defined using the header SAP_MessageType in the integration flow, for example, in a content modifier or script step.

Table 8.7 Properties Contained in the MPL Header (Cont.)

Property	Description
Segment	Indicates that the following steps have been processed in the same processing segment. If a split or multicast step is used, the steps belonging to one subroutine are grouped together within one segment.
StartTime	The time that each step of the integration flow started.
StopTime	The time that each step of the integration flow stopped.
Status	The status of each step of the integration flow.
ChildCount	The serial number of the current processing step.
ModelStepId	The ID of a particular step in the integration flow. This ID is used to specify the relation between a modeled step (in the integration flow) and an MPL entry. The integration flow model steps are fragmented in the Camel runtime environment into several processing steps.
StepID	The ID of a particular step in the log.

Table 8.8 Properties Contained in the MPL for Each Step

Attachments

In addition to viewing the log, it's also possible to access MPL attachments written during message processing. As depicted earlier in [Figure 8.6](#), in the **Attachments** tab, all MPL attachments are listed in a table showing the name of the MPL attachment, the type, and the time the attachment was modified last. The name of the attachment is shown as a link. When you select this link, the attachment opens in a new window. In [Figure 8.18](#), for example, an MPL attachment with the name **MDN Attachment** is shown.

MPL attachments are written by some adapters, such as the AS2 adapter. But you can also create MPL attachments yourself with the help of a script step. Refer to [Chapter 9, Section 9.4](#), for details about using Java APIs in scripts.

[Download](#)

Artifact Name: AS2Sender Status: Completed Processing Time: 3 d 33 min 33 sec 150 ms
Last Updated at: Jan 26, 2018, 13:13:33 Log Level: Info

[MDN Attachment](#) [Log](#)

```
-----_Part_0_913428445.1516707600430
Content-type: text/plain
Content-Transfer-Encoding: 7bit

Message received and processed at local time Tue Jan 23 11:40:00 UTC 2018, internal message ID
-----_Part_0_913428445.1516707600430
Content-Type: message/disposition-notification
Content-Transfer-Encoding: 7bit

Reporting-UA: SAP AS2 Adapter
Original-Recipient: rfc822; ABCCompanyID
Final-Recipient: rfc822; ABCCompanyID
Original-Message-ID: <mendelson_opensource_AS2-1516707599552-7@myCompanyAS2ID_ABCCompanyID>
Disposition: automatic-action/MDN-sent-automatically; processed

-----_Part_0_913428445.1516707600430--
```

Figure 8.18 Accessing the MPL Attachment

Using MPL Attachments

MPL attachments can be created in the script step using the `MessageLog` API. You should use this option with care and avoid writing the whole payload into the MPL attachment because this can cause out-of-memory errors in the worker node, which can lead to unavailability of the whole scenario.

Important recommendations for using MPL attachments can be found in the “[Avoid Storing Payloads in the Message Processing Log](#)” blog in the SAP Community (www.sap.com/community.html).

Now that you know how to monitor messages through SAP Cloud Platform Integration’s monitors, let’s explore how you can manage and customize the tiles on your monitoring dashboard.

8.2.4 Managing Tiles

As mentioned earlier, a tile in the monitor page is a block in a page that filters messages or artifacts corresponding to a particular status. Each one of these tiles can be clicked to display a list of messages or artifacts matching the status. The message and artifact tiles are presented and grouped by statuses. For messages, the following tiles are available by default:

- All Messages
- Failed Messages
- Retry Messages
- Completed Messages

Note

By default, all tiles in the **Monitor Message Processing** section use a one-hour period, which means, for instance, that the **Failed Messages** tile only shows failed messages of the last hour. However, this can be changed easily by choosing another value from the **Time** dropdown menu, as shown in [Figure 8.19](#).

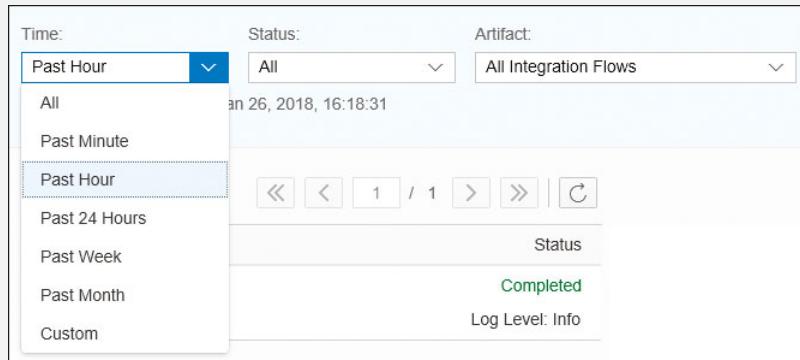


Figure 8.19 Changing Timeframe Values

You may think of these tiles as preconfigured message or artifact filters, or as shortcuts to access the messages or artifacts quickly that you're looking for. Furthermore, you can also remove or move the default tiles to suit your needs. To move the position of a tile, proceed as follows:

1. Mouse over the tile to be moved.
2. Drag and drop the tile to the desired location.

To completely delete or remove an existing tile, follow these steps:

1. Right-click on the concerned tile.
2. From the resulting context menu, select the **Delete** option, as shown in [Figure 8.20](#).

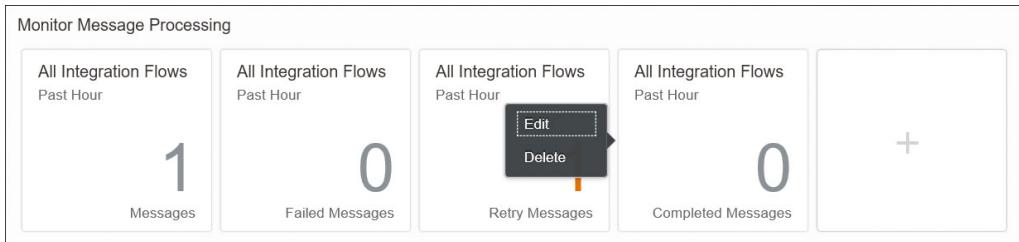


Figure 8.20 Editing and Managing Monitor Message Processing Tiles

If you often apply a particular filter to messages or artifacts, you can also create a new tile for it. For the sake of illustration, imagine that you want to create a tile to view all escalated messages. Follow these steps:

1. Click on the empty tile on the right side of the screen with the + sign.
2. A new **Tile Settings** popup screen appears, from which you can specify the filtering criteria to be applied on past runtime messages. [Figure 8.21](#) shows how to select the **Escalated** option from the **Status** dropdown menu.

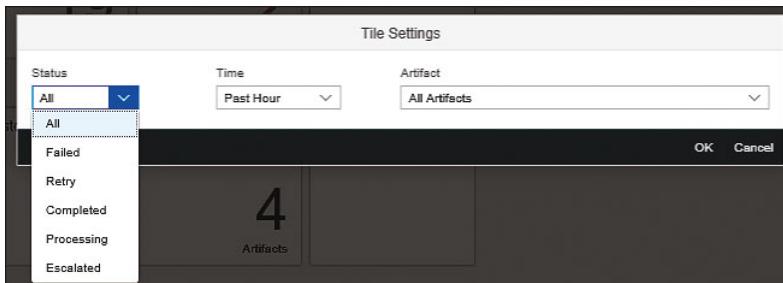


Figure 8.21 Creating a New Message Monitoring Tile

Note that in the **Tile Settings** popup screen for messages, the following filtering options are available:

- **Status**

Different message statuses can be selected, such as **Failed**, **Retry**, **Completed**, and so on.

- **Time**

Different filtering timeframes are available to be selected, including **Past Minute**, **Past Hour**, **Past 24 Hours**, **Past Week**, and so on.

- **Artifact**

All available integration flows and OData services on the target runtime system and tenant are listed. By selecting a particular integration flow or OData service on your tile settings, you can further restrict the filtering of messages to entries of the selected integration flow or OData service, as depicted in [Figure 8.22](#).

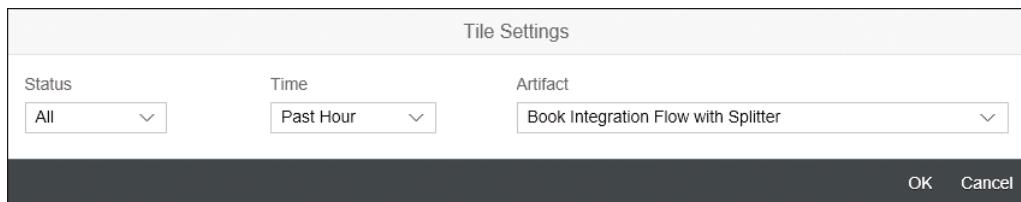


Figure 8.22 Selecting a Specific Integration Flow in the Tile Settings

As an administrator of your SAP Cloud Platform Integration tenant, it's important to gain a good understanding of what each message status in the monitor means. Currently, five statuses are available, as described in [Table 8.9](#).

Status	Description
Failed	The message hasn't been delivered to the receiver, and message processing failed and ended in a dead end. Message processing has ultimately failed, and no more retries are possible.
Escalated	The message runs into an escalation end event as configured in the integration flow. There are several escalation categories available for configuration, for example Receiver not reachable, Receiver not found, Not authorized to invoke the receiver, and so on. An error is raised back to the sender.
Retry	The message processing went into Error status, and an automatic retry was triggered.

Table 8.9 Possible Message Statuses in the Message Monitor

Status	Description
Processing	The message is currently being processed.
Completed	The message has been successfully delivered to the receiver.

Table 8.9 Possible Message Statuses in the Message Monitor (Cont.)

Now that you understand how to monitor integration artifacts and messages processed by the artifacts we'll explore how to maintain the security artifacts for your integration content.

8.3 Manage Security

When dealing with integration scenarios, security is an important topic. To consume other services, you'll most likely need to use security objects such as certificates, user names, passwords, and so forth. The types of objects needed to secure your interfaces are recognized as *security artifacts*. [Table 8.10](#) presents the complete list of security artifacts available in SAP Cloud Platform Integration.

Note

We mainly focus in this chapter on how to manage these artifacts. For more information on the security concepts behind the artifacts, see [Chapter 10](#).

Type	Description
SSH Known Hosts	Contains the trusted hosts that need to be specified when the tenant is connected with a remote component using Secure Shell (SSH).
OAuth2 Credentials	Contains the client ID and client secret of the client you're connecting to using OAuth 2.0 authentication.
PGP Public Keyring	Contains the public keys required when the exchanged messages are digitally signed or encrypted using Open Pretty Good Privacy (Open PGP). See Chapter 10, Section 10.4 .

Table 8.10 Integration Artifact Types Related to Secure Messaging

Type	Description
PGP Secret Keyring	Contains the private keys required when the exchanged messages are digitally signed or encrypted using Open PGP. See Chapter 10, Section 10.4 .
Secure Parameter	Contains credentials to be used together with specific authentication options, such as OAuth. To find out how to use such an artifact when setting up a scenario with OAuth authentication, see Chapter 10, Section 10.4.5 .
User Credentials	Contains username and password information for basic authentication. We introduced this artifact type already with the first scenario described in this book (Chapter 2, Section 2.3.4).
Keystore artifacts	In the keystore, the keys and certificates required to enable secure connection of the tenant with other components based on public key certificates (type X.509) are maintained. See also Chapter 10, Section 10.5 for more details.
Certificate-to-User Mapping	Client certificates mapped to users for role-based authorization. See Chapter 10, Section 10.4.4 , to find out how to use such an artifact when setting up a secure connection.

Table 8.10 Integration Artifact Types Related to Secure Messaging (Cont.)

The security-related artifacts can be maintained in the monitoring dashboard in the **Manage Security** section. As depicted in [Figure 8.23](#), three tiles are available in this section for the different security artifacts: one for the **Keystore** material, such as certificates and key pairs; one for the **Certificate-to-User Mappings**; and one for the additional **Security Material**, such as credentials, PGP keyrings, and secure parameters.

In addition to the security-related tiles, a tile for **Connectivity Tests** is available, in which you have the option to execute several outbound connectivity tests to test that the security material is maintained correctly and the connection can be executed successfully.

Let's explore the different tiles in more detail. We'll start with the **Security Material** tile.

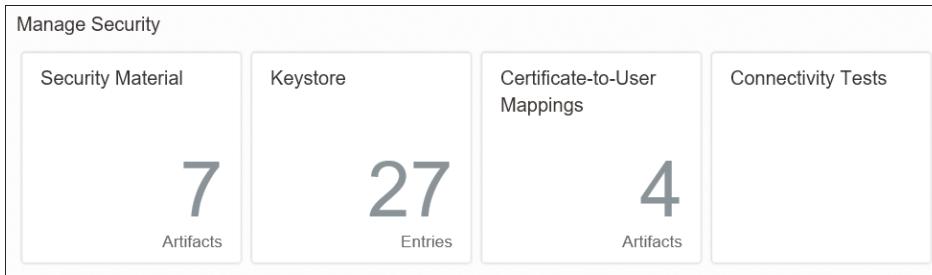


Figure 8.23 Manage Security Section in the Monitoring Dashboard

8.3.1 Maintain Security Material

SAP Cloud Platform Integration provides the security material tile to enable users to manage the tenant's security material. The tile can be accessed via the **Security Material** link, as shown previously in [Figure 8.23](#).

After clicking on the link, you're redirected to a page similar to [Figure 8.24](#). The table displays a list of the different security materials that have been deployed on the tenant and shows the details of the security artifacts in different columns. [Table 8.11](#) provides descriptions of the columns.

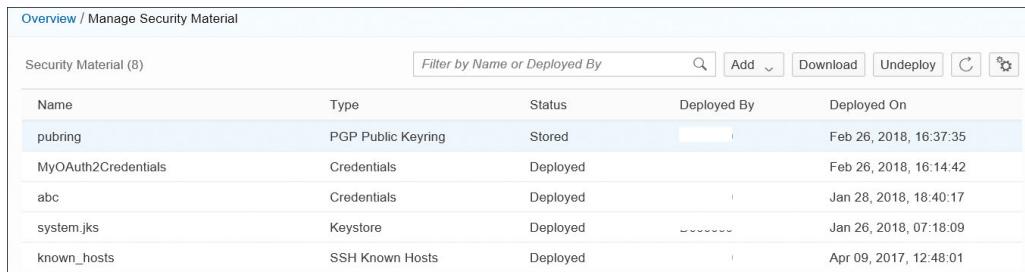
Column Name	Description
Name	Contains the name of the security artifact.
Type	<p>Specifies the type of the security artifact. The possible values for this field are as follows:</p> <ul style="list-style-type: none"> ■ Credentials: Used for user credentials, OAuth2 credentials, and secure parameters. ■ PGP public keyring: Contains the public keys used for PGP encryption and signature verification. ■ PGP secret keyring: Contains the private and public keys used for PGP decryption and signing. ■ SSH known hosts: Contains the public keys of the connected SFTP servers. ■ Keystore: Used for certificates and private keys

Table 8.11 Attributes of a Security Material

Column Name	Description
Status	Specifies the states of the deployed artifact on the SAP Cloud Platform Integration server. Possible values include the following: <ul style="list-style-type: none"> ■ Error: Security artifact has an error. ■ Deployed: Security artifact is successfully deployed on the worker node. ■ Deploying: Security artifact is being deployed on the worker node. ■ Stored: Security artifact is stored on the tenant management node but not yet deployed on the worker node.
Deployed By	The name of the user who performed the deployment.
Deployed On	The date and time when the deployment was performed.

Table 8.11 Attributes of a Security Material (Cont.)

If you're not satisfied with the layout of the table shown in [Figure 8.24](#), you can change and customize it using the **Table Settings** icon .

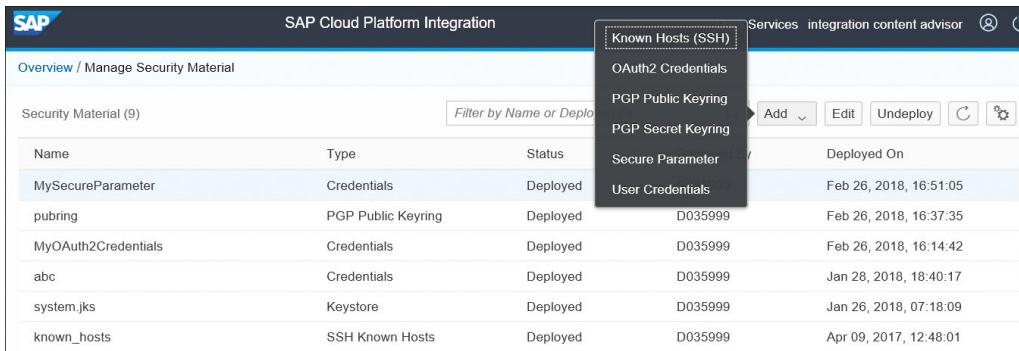


Overview / Manage Security Material				
Security Material (8)		Filter by Name or Deployed By <input type="text"/> <input type="button" value="Search"/> <input type="button" value="Add"/> <input type="button" value="Download"/> <input type="button" value="Undeploy"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/>		
Name	Type	Status	Deployed By	Deployed On
pubring	PGP Public Keyring	Stored		Feb 26, 2018, 16:37:35
MyOAuth2Credentials	Credentials	Deployed		Feb 26, 2018, 16:14:42
abc	Credentials	Deployed		Jan 28, 2018, 18:40:17
system.jks	Keystore	Deployed	System	Jan 26, 2018, 07:18:09
known_hosts	SSH Known Hosts	Deployed		Apr 09, 2017, 12:48:01

Figure 8.24 Manage Security Material

Let's now explore what is needed to add or deploy a new security artifact to your tenant. Proceed as follows:

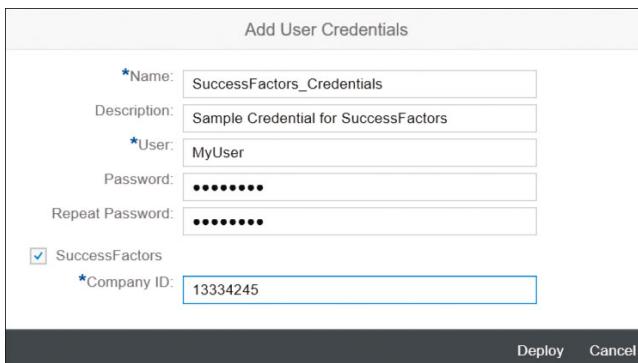
1. At the top-right side of the screen, select the **Add** button, as shown in [Figure 8.25](#).
2. Select the desired type of artifact (e.g., **User Credentials**), and fill in the requested details, as shown in [Figure 8.26](#).
3. If the desired artifact is a credential to be used for an SAP SuccessFactors-related integration scenario, select the **SuccessFactors** checkbox, and fill in the **Company ID** field. The **Company ID** represents the client instance used to connect to the SAP SuccessFactors system. Refer to [Figure 8.26](#) as an example.



The screenshot shows the SAP Cloud Platform Integration interface. In the top right corner, there is a context menu with several options: Known Hosts (SSH), OAuth2 Credentials, PGP Public Keyring, PGP Secret Keyring, Secure Parameter, and User Credentials. The 'Known Hosts (SSH)' option is currently selected. Below the menu, a table lists various security artifacts. One row, 'known_hosts', is highlighted.

Name	Type	Status	Deployed On
MySecureParameter	Credentials	Deployed	Feb 26, 2018, 16:51:05
pubring	PGP Public Keyring	Deployed	Feb 26, 2018, 16:37:35
MyOAuth2Credentials	Credentials	Deployed	Feb 26, 2018, 16:14:42
abc	Credentials	Deployed	Jan 28, 2018, 18:40:17
system.jks	Keystore	Deployed	Jan 26, 2018, 07:18:09
known_hosts	SSH Known Hosts	Deployed	Apr 09, 2017, 12:48:01

Figure 8.25 Adding a New Security Artifact



The screenshot shows the 'Add User Credentials' dialog box. It contains fields for Name, Description, User, Password, Repeat Password, and Company ID. The 'SuccessFactors' checkbox is checked. At the bottom, there are 'Deploy' and 'Cancel' buttons.

*Name:	SuccessFactors_Credentials
Description:	Sample Credential for SuccessFactors
*User:	MyUser
Password:	*****
Repeat Password:	*****
<input checked="" type="checkbox"/> SuccessFactors	
*Company ID:	13334245

Figure 8.26 Sample Security Credentials for an SAP SuccessFactors Scenario

- Select **Deploy** at the bottom of the **Add User Credentials** dialog. The new security artifact is deployed and added to the list.

Keystore Entry in the List of Security Material

You may have noticed that an entry for the keystore exists in the table of security artifacts, but you can't add a keystore using the **Add** action. This is because a dedicated monitor exists for managing the keystore entries, which can be accessed via the **Keystore** tile.

Credential artifacts can be changed and edited by performing the following steps:

- Select a credential entry from the table of security artifacts. An **Edit** button then appears on the top-right corner ([Figure 8.27](#)).

Security Material (9)					<input type="text" value="Filter by Name or Deployed By"/>	<input type="button" value="Search"/>	<input type="button" value="Add"/>	<input type="button" value="Edit"/>	<input type="button" value="Undeploy"/>	<input type="button" value="C"/>	<input type="button" value="S"/>
Name	Type	Status	Deployed By	Deployed On							
SuccessFactors_Credentials	Credentials	Deployed	9	Feb 26, 2018, 18:34:23							
pubring	PGP Public Keyring	Deployed	3	Feb 26, 2018, 16:37:35							

Figure 8.27 Selecting a Credential Artifact for Editing

- Click **Edit** to change the properties of the credential artifact. Figure 8.28 shows an example of a user credential being edited.

Edit User Credentials

*Name:	SuccessFactors_Credentials
Description:	Sample Credential for SuccessFactors
*User:	MyNewUser
Password:	*****
Repeat Password:	*****
<input checked="" type="checkbox"/> SuccessFactors	
*Company ID:	13334245

Deploy Cancel

Figure 8.28 Updating Security Material of Type User Credential

PGP keyrings and SSH known hosts can be downloaded from the list by using the **Download** button, as shown in Figure 8.29. Furthermore, all artifact types (except the **Keystore**) can be removed from the tenant using the **Undeploy** button.

Security Material (9)					<input type="text" value="Filter by Name or Deployed By"/>	<input type="button" value="Search"/>	<input type="button" value="Add"/>	<input type="button" value="Download"/>	<input type="button" value="Undeploy"/>	<input type="button" value="C"/>	<input type="button" value="S"/>
Name	Type	Status	Deployed By	Deployed On							
SuccessFactors_Credentials	Credentials	Stored		Feb 26, 2018, 18:36:13							
pubring	PGP Public Keyring	Deployed		Feb 26, 2018, 16:37:35							

Figure 8.29 Selecting a PGP Keyring for Downloading

Let's now discuss how you can manage the certificates and key pairs in the keystore monitor.

8.3.2 Manage the Keystore

Using the second tile in the **Manage Security** section, the **Keystore** tile, you can maintain the certificates and private key pairs deployed on the tenant.

Preinstalled Artifacts in the Keystore

When provisioning a new tenant, SAP preinstalls the following artifacts in the key-store:

- One key pair with the alias `sap_cloudintegrationcertificate`
- Some SAP-owned root certificates, which enable communication with other SAP cloud systems, such as SAP Ariba and SAP Customer Relationship Management (SAP CRM)

Those certificates and key pairs can be used by customers to set up secure HTTP connections to backend systems using client certificates; for scenarios using message-level security; to sign or decrypt messages using PKCS7, XML, or simple signer; or in WS-Security.

More details about the different security features SAP Cloud Platform Integration offers are provided in [Chapter 10](#).

To open the keystore monitor, select the **Keystore** tile. As depicted in [Figure 8.30](#), four tabs are available, and the most important tab **Current** is opened when selecting the **Keystore** tile. It shows all certificates and key pairs currently deployed on the tenant in a table. The expired artifacts are highlighted in red to bring them to your attention because they usually need to be updated to guarantee successful scenario execution.

Overview / Manage Keystore						
Entries (13)		Type	Owner	Valid Until	Last Modified At	Actions
dewdfgwp00822.wdf.sap.corp (sapnetca)	Certificate	Tenant Administrator	Feb 05, 2015, 14:08:09	May 24, 2017, 14:39:24	Edit	
id_rsa	Key Pair	Tenant Administrator	Jul 23, 2023, 13:10:53	Nov 11, 2016, 09:48:26	Edit	
sap	Key Pair	Tenant Administrator	Jun 27, 2018, 10:54:19	Jul 13, 2017, 17:18:34	Edit	
sap_global_root_ca	Certificate	Tenant Administrator	Apr 26, 2032, 17:46:27	Nov 11, 2016, 09:48:26	Edit	
sap_passport_ca	Certificate	Tenant Administrator	Apr 01, 2018, 12:00:00	Nov 11, 2016, 09:48:26	Edit	
sapnetca	Certificate	Tenant Administrator	Jul 18, 2015, 14:00:00	May 24, 2017, 14:39:24	Edit	
sapnetca_g2 (sap_global_root_ca)	Certificate	Tenant Administrator	Mar 17, 2025, 10:34:51	Nov 11, 2016, 09:48:26	Edit	
sgw-a1s-05 (sap_se webadmin ca)	Certificate	Tenant Administrator	Jan 01, 2038, 01:00:01	Jan 08, 2018, 09:02:12	Edit	
vmw4998.wdf.sap.corp (sapnetca)	Certificate	Tenant Administrator	Jan 28, 2015, 17:04:41	May 24, 2017, 14:39:24	Edit	
sap_baltimore cybertrust root	Certificate	 SAP	May 13, 2025, 01:59:00	May 24, 2017, 14:39:24	Edit	
sap_myccn2	Key Pair	 SAP	Oct 23, 2018, 14:10:25	Nov 03, 2017, 13:57:27	Edit	

Figure 8.30 Keystore Monitor

The table on the screen provides the header details of the certificates and key pairs in different columns. Table 8.12 describes the columns.

Column Name	Description
Alias	Contains the alias name of the certificate or key pair. The alias name is shown as a link, and selecting it opens the details of the certificate or key pair, as shown in <u>Figure 8.31</u> . Note that the prefix sap_ is reserved for SAP-owned key pairs and certificates.
Type	Specifies the type of the keystore artifact. The possible values for this field are as follows: <ul style="list-style-type: none"> ■ Certificate ■ Key Pair For more details, see <u>Chapter 10, Section 10.5.2</u> .
Owner	Identifies the owner of the keystore artifact. The possible values are as follows: <ul style="list-style-type: none"> ■ Tenant Administrator: Those artifacts are maintained by the customer's tenant administrator. ■ SAP: SAP-owned artifacts are maintained by SAP, and they can't be changed or deleted by the customer. This is also indicated by the Lock icon .
Valid Until	Validity of the certificate or key pair. As shown in <u>Figure 8.30</u> , if the artifact is already expired, the validity timestamp is shown in red.
Last Modified At	The date and time when the artifact was changed.
Actions	The actions available differ for the different artifacts and also depend on the owner of the artifacts: <ul style="list-style-type: none"> ■ For Certificates: Delete, Download, Rename, Update. ■ For Key Pairs: Delete, Download Certificate, Download Certificate Chain, Rename, Update. ■ For id_rsa/id_dsa keys: Delete, Download Certificate, Download Certificate Chain, Download Public OpenSSH Key, Rename, update. ■ For SAP-owned certificates: Download. ■ For SAP-owned Key Pairs: Download Certificate, Download Certificate Chain. Detailed descriptions of the actions can be found in the “Keystore Monitor Now Available for Tenant Administrator” blog in the SAP Community (www.sap.com/community.html).

Table 8.12 Attributes of Keystore Artifacts

id_rsa/id_dsa Keys in the Keystore Monitor

The key pair aliases id_dsa and id_rsa are reserved for key pairs used in setting up secure SFTP connections. This is described in detail in the “How to Set Up a Secure Connection to an SFTP Server” blog in the SAP Community (www.sap.com/community.html).

On the top of the table, actions for the whole keystore are available. See [Table 8.13](#) for details of those actions.

Action	Description
Filter	Filter for specific artifacts. The filter is executed for the alias name field.
Back Up	Back up all certificate and key pairs owned by the tenant administrator. Backups for SAP-owned artifacts are executed by SAP and aren't visible in the keystore monitor.
Add	Add entries from keystore files, single certificate, and key pairs. Note that you're not allowed to add artifacts with the prefix sap_ because this prefix is reserved for SAP-owned key pairs and certificates.
Download	Download the public content of the keystore. The downloaded keystore file is named <i>PublicContentKeystore.jks</i> and is saved without a password. It can be opened and maintained by any external keystore editor. Note that private key pairs aren't downloaded; for security reasons, private keys don't leave the tenant.
Reload	Reload the content of the page.
Settings	Define specific table settings such as sorting and filtering owner and artifact types.

Table 8.13 Actions for Keystore

As you saw in [Figure 8.30](#), the alias name of the certificate or key pair is shown as a link. When you click the link, the details of the selected entry are provided in a new window. In [Figure 8.31](#), for example, you see the details of a key pair. On the left side of the screen, the certificate chain, also known as the chain of trust, is shown as a tree with the key pair at the bottom and one or more certificates above. For a signed key pair, you usually have the key pair at the bottom, one or more intermediate certificates, and the root certificate at the top that identifies the root certificate authority

(CA). You can click all the entries in the tree to get the details of all the involved certificates.

Certification Path		Rename	Download	Delete
✓ SAP Global Root CA	General	Fingerprints	Administration	
✓ SAPNetCA_G2				
pi.r.sap.corp	Alias: piouser_old2 Keystore: Current Type: Key Pair Owner: Tenant Administrator Version: 3 Serial Number: 0x37E5 Subject DN: CN=pi.r.sap.corp, O=SAP, C=DE Issuer DN: CN=SAPNetCA_G2, O=SAP, L=Waldorf, C=DE Key Type: RSA Key Size: 2048 Signature Algorithm: SHA256withRSA Valid From: Jun 08, 2015, 14:48:45 Valid Until: Jun 08, 2017, 14:48:45			
Fingerprints				
SHA-1: 6B:D7:E8:D7:0C:8D:19:1C:9D:02:D3:04:FE:06:46:70:F5:8C:DE:F3 SHA-256: 1A:66:6B:AD:79:D8:1E:9B:30:22:8F:C8:B1:8C:27:AB:C5:DF:89:A3:90:53:08:A6:01:74:EE:02:23:54:D0:A8 SHA-512: 5A:0B:80:88:10:0F:38:45:9F:E2:08:BE:50:DB:FB:57:21:0F:B5:F9:10:12:B7:29:DC:A9:D2:DC:E9:E4:19:5E: 7B:B5:9E:D1:A5:93:1FA8:E0:FB:83:4B:33:86:9B:B4:88:E9:12:01:C7:09:8C:FB:E3:E7:38:58:B1:B6:1D				
Administration				
Created By:	Created On:			
[User]	Jan 26, 2018, 07:18:09			
Last Modified By:	Last Modified On:			
[User]	Jan 26, 2018, 07:18:09			

Figure 8.31 Details of a Keystore Artifact

On the right side of the screen, the details of the certificate or key pair are provided, such as **Subject DN**, **Issuer DN**, **Key Type**, **Signature Algorithm**, and **Valid From/Valid Until** (see also [Chapter 10, Section 10.5.2](#)). The unique fingerprint of the artifact, also known as a thumbprint, is provided in the **Fingerprints** tab. The fingerprint is shown in hexadecimal format and can be used to check that the key wasn't being injected by an attacker during key exchange.

The **Administration** tab shows the user who created and modified the artifact and the time of the creation and change.

Note

More details about the security and especially the keystore artifacts together with processes for how to handle security material are provided in [Chapter 10](#).

On the right top of the screen, the same actions are offered as in the single line actions in the table containing all certificates and key pairs, such as **Delete**, **Rename**, and **Download**.

Keystore Monitor: Additional Features

The keystore monitor will be further extended in the future with additional features, such as creating private keys and handling certificate signing requests. The “Keystore Monitor Now Available for Tenant Administrator” blog in the SAP Community (www.sap.com/community.html) is regularly updated with the new features.

Backup

For secure system management, it's important to be able to restore certificates and key pairs if there are problems. To support you in performing this task, SAP Cloud Platform Integration offers the **Back Up** option in the keystore monitor. With this option, a backup can be created for all keystore artifacts owned by the tenant administrator.

No Backup for Single Certificates or Key Pairs

Note that when creating a backup, ALL certificates and key pairs from the keystore are backed up; there is no option to back up single artifacts of the keystore.

The backup can be explored in the **Backup** tab in the keystore monitor. As depicted in [Figure 8.32](#), the table available in the **Backup** tab contains all certificates and key pairs backed up.

Current	Backup	New SAP keys (2)	SAP Key History
Entries (8) - Jul 26, 2017, 15:49:12			
Alias	Type	Owner	Valid Until
dewdfgwp00822.wdf.sap.corp (sapnetca)	Certificate	Tenant Administrator	Feb 05, 2015, 14:08:09
id_rsa	Key Pair	Tenant Administrator	Jul 23, 2023, 13:10:53
			Last Modified At
			May 24, 2017, 14:39:24
			Nov 11, 2016, 09:48:26

Figure 8.32 Backup Tab in Keystore Monitor

At the top of the table, the timestamp shows when the backup was performed. Note that there is always only one backup kept, which means the next backup will overwrite the backup created before.

The **Backup** tab provides the same artifact details for the artifacts in the backup as the **Current** tab for the currently active artifacts (see [Table 8.12](#)), except that the **Actions** column isn't available because artifacts from the backup can't be changed. As in the **Current** tab, the alias name is shown as a link, which allows you to access the details of the artifact in a new window. You can use this view to explore the content of the backup, but you can't modify the artifacts in the backup.

The only action available in the **Backup** view is **Restore**. Using **Restore**, the active key-store is overwritten by all the certificates and key pairs from the backup. This option helps you switch back to the old keystore, for example, if changes in the keystore caused communication errors in running integration scenarios.

Restore Overwrites Active Keystore with Backup

Note that when using **Restore**, the whole keystore is replaced, meaning newly created artifacts in the active keystore that don't exist in the backup are removed. Only the SAP-owned entries are kept; they aren't touched by the backup and restore operations.

It's recommended to always create a backup before making any changes to keystore artifacts, such as renaming aliases or overwriting key pairs and certificates. Using the backup, you then have the option to switch back to the last keystore version that worked correctly.

SAP Keys

As explained already you're not allowed to change SAP-owned key pairs and certificates, but you can use them in your integration scenarios. When such SAP-owned

artifacts expire, SAP is responsible for renewing them. The renewal process differs depending on the artifact type:

- For SAP-owned root certificates, the process is easy: SAP adds the new root certificate to the keystore, and it's active immediately. There is no need for customers to react on the update.
- If the SAP-owned key pair expires, the process is more complex because SAP can't just update the key pair as this would lead to errors in existing scenarios. If the SAP Cloud Platform Integration tenant gets a new key pair, all backends connected using this key need to be updated as well.

To address the need to involve customers in updating associated keys in the involved backends, the keystore monitor offers the **New SAP keys** tab. As depicted in [Figure 8.33](#), the table in the **New SAP keys** tab contains the new key pairs uploaded by SAP. Refer to [Table 8.14](#) for a description of the columns in the table.

Current	Backup	New SAP keys (2)	SAP Key History
Entries (2)			<div style="display: flex; align-items: center;"> Filter by Alias <input type="text"/> </div>
Alias	Valid From	Valid Until	Last Modified At
sap_mync	Oct 23, 2017, 12:27:45	Oct 23, 2018, 12:27:45	Jan 12, 2018, 08:57:18

Figure 8.33 New SAP Keys Tab in the Keystore Monitor

Column Name	Description
Alias	Contains the alias name of the new SAP key pair. The alias name is shown as a link, and selecting it opens the details of the key pair.
Valid from	Start time the key pair is valid from.
Valid Until	Validity of the new key pair.
Last Modified At	The date and time when the new key pair was changed.
Actions	Actions available for the new SAP key pair: <ul style="list-style-type: none"> ■ Activate ■ Download Certificate ■ Download Certificate Chain

Table 8.14 Attributes of SAP Key Pairs in the New SAP Keys Tab

The new SAP key pair isn't active yet in the keystore, but it's available to trigger the update process from the customer's side. The tenant administrator needs to coordinate the overall renewal process as described in [Chapter 10](#), [Chapter 10.5.3](#), as well as in the “Activate SAP Keys in Keystore Monitor” blog in the SAP Community (www.sap.com/community.html).

During activation of the new SAP key pair, a backup of the old SAP key pair with the respective alias is stored in the **SAP Key History** tab to revert the change, if necessary. The **SAP key history** tab lists all SAP key pairs that were active in the tenant ([Figure 8.34](#)). See [Table 8.15](#) for a description of the columns in the table.

Current	Backup	New SAP keys (2)	SAP Key History
Entries (1)			<input type="text" value="Filter by Alias"/> Filter icon Reset icon Up icon
Alias	Active From	Active Until	Actions
sap_my(cn)2	Nov 03, 2017, 13:52:06	Nov 03, 2017, 13:57:27	Copy icon

Figure 8.34 SAP Key History Tab in the Keystore Monitor

Column Name	Description
Alias	Contains the alias name of the backed up SAP key pair. The alias name is shown as a link, and selecting it opens the details of the key pair.
Active from	Time the key pair was active from.
Active Until	Time the key pair was active until.
Actions	Actions available for the backed up SAP key pair: <ul style="list-style-type: none"> ■ Add to New SAP keys ■ Download Certificate ■ Download Certificate Chain

Table 8.15 Attributes of SAP Key Pairs in the SAP Key History Tab

The most important action in the **SAP Key History** tab is the **Add to New SAP Keys** action. With this option, you can revert to the old SAP key pair. When you select this option, the SAP key pair is written back to **New SAP Keys**, and you can reactivate it from there. When reverting to the old SAP key pair, keep in mind that some backends may have already activated the new certificate. Therefore, use this option with care!

Automatic Activation of New SAP Key Pair

If the new SAP key pair wasn't activated by the tenant administrator before the key pair expired, a job activates the new SAP key automatically on behalf of the tenant administrator. This is done to ensure that the tenant has a valid key pair in the key-store.

For more information on the lifecycle management for keys, read [Chapter 10, Section 10.5.3](#).

Now that you're aware of how to manage security material such as credentials, certificates, and key pairs, you need to understand the mapping of certificates to users to be able to use client certificates for inbound authorization.

8.3.3 Maintain Certificate-to-User Mappings

In a more commonly known on-premise corporate environment, systems live in the limited network of your organization. With a controlled network, it can be argued that there are relatively few threats, and that the user account/password model works well to secure your platforms. However, the Internet being a potentially hostile environment, cloud-based platforms are more prone to user account/password attacks. A certificate-to-user mapping provides a good alternative solution to the user/password model by using public-key (certificates) technology. Certificates are much more difficult to hack than password-based systems.

SAP Cloud Platform Integration uses certificate-to-user mappings to map a certificate that has been issued to a user's account. SAP Cloud Platform Integration can then use public-key technology to authenticate a user through his certificate. The result is the same as if a user had applied a username and password, but the approach is more secure. With this approach, a user presents a certificate to SAP Cloud Platform Integration, and the platform looks at the mapping to determine which user account should be logged on.

The usage and setup of certificate-to-user mappings in scenarios using this option for inbound authorization are described in detail in the "How to Set Up a Secure HTTP Inbound Connection with Client Certificates" blog in the SAP Community (www.sap.com/community.html).

Now that you understand the idea behind using certificate-to-user mappings, the question is how to create them. From within the **Manage Security** section of the

monitoring dashboard (refer back to [Figure 8.23](#)), one of the included tiles is **Certificate-to-User Mappings**. This tile provides access to all certificate-to-user mappings defined and deployed in your tenant. After clicking on this tile, you're redirected to a page with a list of available certificate-to-user mappings, as shown in [Figure 8.35](#).

Certificate-to-User Mappings (4)						
User Name	Subject DN	Issuer DN	Serial Number	Valid Until	Modified By	Modified At
MyCommunicationUser	CN=ldcj...p,OU=...SYS,L=Waldorf,O=SAP,AG,ST=Baden-Baden,C=DE	CN=I...sap.corp,OU=...SYS,L=Waldorf,O=SAP,AG,ST=Baden-Baden,C=DE	2198626846	Apr 16, 2028, 06:16:27	D...J9	May 17, 2017, 16:03:57
picouser	CN=picouser...corp,OU=...Infra,O=SAP,C=DE	CN=SAPNetCA_G2,O=SAP,L=Waldorf,C=DE	82466	Jun 20, 2019, 11:22:57	Di...j	Jan 26, 2018, 18:40:51

Figure 8.35 Listing Certificate-to-User Mappings

A certificate-to-user mapping comprises a number of attributes, which are listed in [Table 8.16](#).

Attribute Name	Description
User Name	The unique identifier of the user to which the certificate needs to be mapped
Subject DN	The unique identifier for the object being secured with information about the object being certified, including common name, organization, organization unit, and country codes, among others
Issuer DN	The name of the party that signed the certificate
Serial Number	A number that uniquely identifies the certificate, which is issued by the CA
Valid Until	The date and time the certificate expires
Modified By	The user that last changed the certificate-to-user mapping artifact
Modified At	The date and time the certificate-to-user mapping artifact was most recently changed

Table 8.16 Attributes of a Certificate-to-User Mapping

On the page presented in [Figure 8.35](#), you can edit mappings using the **Edit** button or delete them using the **Delete** button. Furthermore, you can add new certificate-to-user mappings to your tenant by following these steps:

1. Select the **Add** button at the top-right corner of the screen (refer to [Figure 8.35](#)).
2. Specify the **User Name** to which the certificate needs to be mapped (see [Figure 8.36](#)).

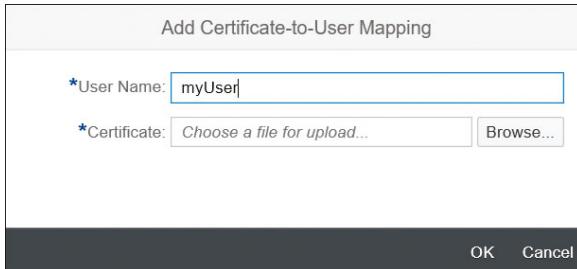


Figure 8.36 Adding a New Certificate-to-User Mapping

3. Click the **Browse** button, and search for the certificate file (you're looking for a *.cer file) on your local file system. The details of the certificate, as indicated in [Table 8.16](#), are automatically populated based on the imported certificate.
4. Select **OK**.

Note

For more information on how to use a certificate-to-user mapping when setting up a secure connection, see [Chapter 10](#), [Chapter 10.3.5](#).

With the maintenance of the certificate-to-user mappings, we finish the section about managing security material. However, as you saw already in the monitoring dashboard (refer to [Figure 8.23](#)), there is one more tile in the **Manage Security** section: **Connectivity Tests**. Let's check out how this can help us test the security material.

8.3.4 Test Outbound Connectivity

Using the tests available in the **Connectivity Tests** tile, you can check that the security material deployed can be used successfully to establish secure connections to the connected backends, mail servers, and SFTP servers.

As depicted in [Figure 8.37](#), when you select the **Connectivity Test** tile, a new window opens that has tabs for the different connection options:

■ TLS

This test tries to establish an HTTPS connection via Transport Layer Security (TLS) to a backend. HTTPS is used by several receiver adapters, such as SOAP, IDoc, and HTTP, to connect to the backends.

■ SSH

This test tries to establish a connection via SSH to an SFTP server. The SSH protocol is used by the SFTP sender and receiver adapters to communicate with the SFTP server.

■ SMTP

This test tries to establish a Simple Mail Transfer Protocol (SMTP) connection to a mail server. This is the protocol used by the mail receiver adapter to send messages to the mail server. You used this tool already when setting up the first integration flow covered in this book ([Chapter 2, Section 2.3.6](#)).

■ IMAP

This test tries to establish a connection via the Internet Message Access Protocol (IMAP) to a mail server. This is one of the protocols used by the mail sender adapter to poll messages from mail servers.

■ POP3

This test tries to establish a connection via Post Office Protocol version 3 (POP3) to a mail server. This is the second protocol that can be used by the mail sender adapter to poll messages from mail servers.

TLS	SSH	SMTP	IMAP	POP3
Request				
*Host: <input type="text" value="www.example.org"/>				
*Port: <input type="text" value="443"/>				
<input type="checkbox"/> Authenticate with Client Certificate				
<input checked="" type="checkbox"/> Validate Server Certificate				
Send				

Figure 8.37 Connectivity Tests

Let's take a look at the different options.

Test HTTPS Connection

The **TLS** test tries to establish an HTTPS connection via TLS to the server specified. For the **TLS** test, the **Host** and **Port** fields of the server to connect to must be set. In [Figure 8.38](#), a test is executed against **google.com**. You see that the response indicates an error during the Secure Sockets Layer (SSL) handshake because no valid certificate can be found in the keystore.

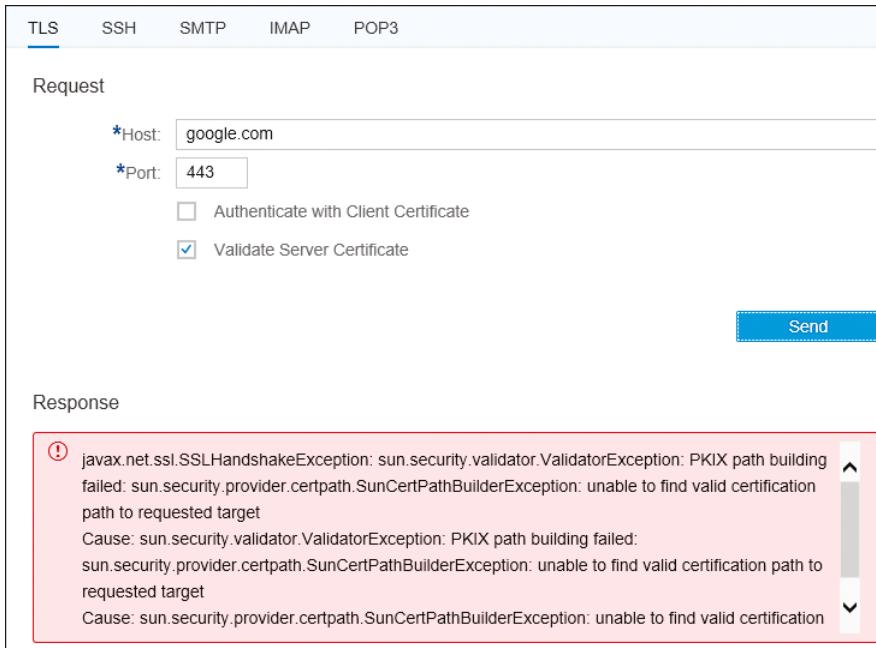


Figure 8.38 TLS Connection Test with SSL Handshake Error

You see that the default setting is to validate the server certificate, which is always the case when a TLS connection is established during runtime. But in the connection test, you have the option to deselect the **Validate Server Certificate** checkbox. By using the option to establish a connection without validation of the server certificate, you can check that the server can be reached at all, and the certificates are displayed that the server provides for validation. In [Figure 8.39](#), you can see that the TLS test without validation check is executed successfully, and the certificates are displayed in the **Response** view.

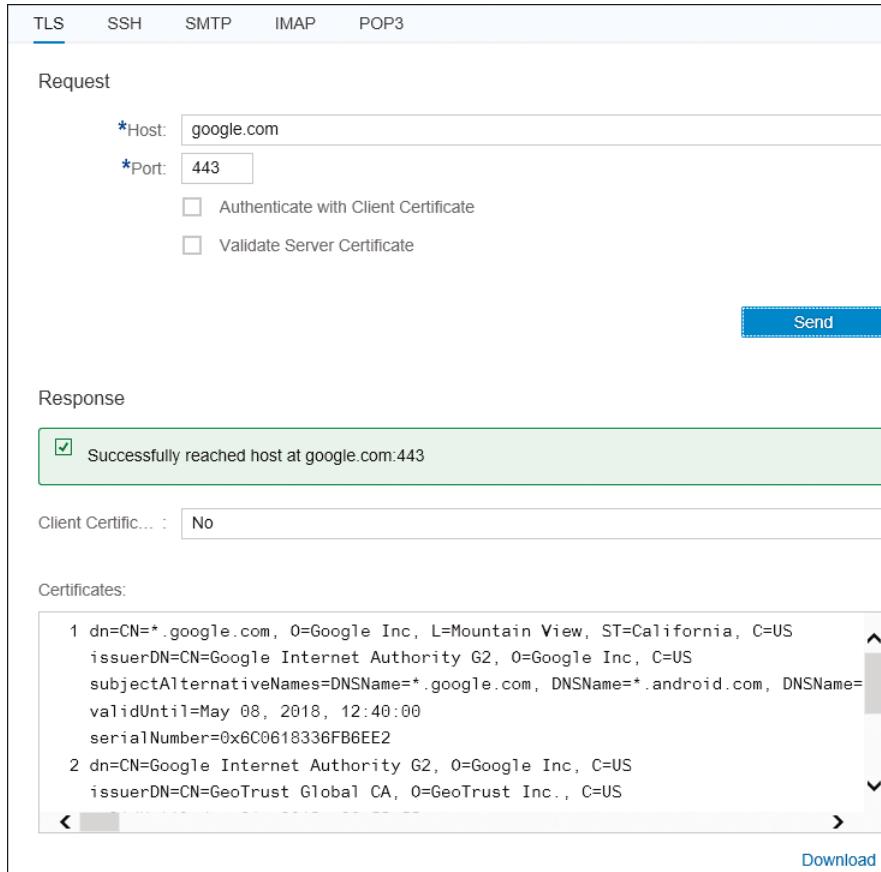


Figure 8.39 TLS Connection Test without Validating Server Certificate

You don't need much more to get the TLS connection established successfully: you can either get the trusted root certificate from the backend server or simply download the certificates shown in the TLS connection test via the **Download** button at the bottom of the screen. The **Download** action creates a *certificates.zip* file in the download folder of your local PC. The file contains all certificates from the connection test. Extract the *.cer file of the root CA, in this case, the GeoTrust Global CA, and add it to the keystore in the keystore monitor. Execute the test again with the **Validate Server Certificate** checkbox checked. The test should now pass successfully because the certificate can be validated by SAP Cloud Platform Integration.

The second checkbox, **Authenticate with Client Certificate**, can be used to test the client-certificate based authentication at the backend. As shown in [Figure 8.40](#), when you select the checkbox, a new entry field **Alias** appears, where you can enter a key-store alias for the key pair you want to use for authentication. If the **Alias** field is left empty, the system tries to find a valid key pair in the keystore and uses it.

After execution of the test, the **Response** section indicates whether the client certificate-based authentication was used and shows the **Alias** that was used.

The screenshot shows the TLS connection test configuration and its successful execution. In the 'Request' section, the host is set to 'dc.....sap.hana.ondemand.com' and the port to '443'. The 'Authenticate with Client Certificate' checkbox is checked, and the 'Alias' field contains 'picouser'. The 'Validate Server Certificate' checkbox is also checked. A 'Send' button is present. In the 'Response' section, a green box indicates success: 'Successfully reached host at dc.....sap.hana.ondemand.com:443'. Below this, the 'Client Certificate Used' field is set to 'Yes' and the 'Alias' field is set to 'picouser'.

Figure 8.40 TLS Connection Test with Client Certificate Authentication

You've now learned that the TLS test can be used to test the transport layer security against backends connected via HTTPS. With this, you have to option to test the connection and authentication settings used in the integration flow channels of all adapters using HTTPS, such as SOAP, HTTP, IDoc, and XI.

Test the Connection to the SFTP Server

To test the connection to an SFTP server, the **SSH** test can be used. The **Host** and **Port** fields of the SFTP server need to be entered, together with the **Authentication** option to be used when connecting to the SFTP server ([Figure 8.41](#)).

Using **Authentication** option **None**, you can test that the SFTP server can be reached at all, and you can download the host key of the SFTP server using the **Copy Host Key** button at the bottom of the **Response** screen. This host key needs to be added to the known hosts file and deployed in the **Manage Security Material** tile to be able to successfully establish an SSH connection to the SFTP server. This is described in detail in the “How to Setup Secure Connection to sftp Server” blog in the SAP Community (www.sap.com/community.html).

The screenshot shows the SAP Fiori interface for managing security. The top navigation bar has tabs for TLS, SSH, SMTP, IMAP, and POP3. The SSH tab is currently selected. Below the tabs is a 'Request' section with fields for Host (l sap.corp), Port (22), and Timeout (10000). The Authentication section shows a radio button for 'None' selected. A 'Send' button is located below the request section. The 'Response' section displays a green box indicating a successful connection: 'Successfully reached host at l sap.corp:22'. It also shows the Host Key Type as 'ssh-rsa' and the Host Key Fingerprint as 'bd:03:bd:67:46:60:1b:4b:55:11:9e:0b:ff:77:8b:3b'. A 'Copy Host Key' button is located at the bottom right of the response section.

Figure 8.41 SSH Connection Test without Authentication

Authentication: None

Note, that the **Authentication** option **None** is only available in the SSH test to be able to download the host key without authentication. This option isn't available in the SFTP adapter because either public key or user/password authentication is mandatory in runtime.

If either **Public Key** or **User Credentials** are used for **Authentication**, two new checkboxes appear (Figure 8.42):

- **Check Host Key**

Selecting this checkbox causes the SSH test to check that the host key is deployed

in the known hosts file and that the SAP Cloud Platform Integration tenant trusts this SFTP server. In runtime, this is mandatory; otherwise, messages cannot be polled from the SFTP server nor sent to the SFTP server.

■ Check Directory Access

Using this checkbox, you can test whether the user connecting to the SFTP server has the authorization to access its directories. If you enter a specific directory in the **Directory** field, you get all files in this directory listed in the **Response** screen.

Request					
TLS	SSH	SMTP	IMAP	POP3	
Host:					
<input type="text" value="sap.corp"/>					
Port:					
<input type="text" value="22"/>					
Timeout (in ms):					
<input type="text" value="10000"/>					
Authentication:					
<input type="radio"/> None <input checked="" type="radio"/> Public Key <input type="radio"/> User Credentials					
User Name:					
<input type="text" value="sftp"/>					
<input checked="" type="checkbox"/> Check Host Key <input checked="" type="checkbox"/> Check Directory Access					
Directory:					
<input type="text" value="mercury/sftptest"/>					
Send					
Response					
<div style="background-color: #c8f7e4; padding: 5px;"> <input checked="" type="checkbox"/> Successfully reached host at sap.corp:22 </div>					
Host Key Type:					
<input type="text" value="ssh-rsa"/>					
Host Key Fingerprint:					
<input type="text" value="bd:03:bd:67:46:60:1b:4b:55:11:9e:0b:ff:77:8b:3b"/>					
Copy Host Key					
Directory:					
<pre>drwxr-xr-x 4 sftp users 4096 Aug 9 2017 . drwxr-xr-x 2 sftp users 9531392 Jan 26 2017 Out drwxr-xr-x 14 sftp users 4096 Mar 29 2017 .. drwxr-xr-x 2 sftp users 4096 Aug 11 2016 .archive -rw-r--r-- 1 sftp users 8625 Feb 20 12:15 test</pre>					

Figure 8.42 SSH Connection Test with Public Key Authentication

Now that you know how to test the connection to SFTP servers and how to download the host key, you can successfully set up and test scenarios using the SFTP sender or receiver adapter.

Test the Connection to the Mail Server

If you need to analyze issues connecting to a mail server, either for sending messages to the mail server or for polling messages from the mail server, there are three different tests to assist you. While the **SMTP** test checks the outbound connection to send messages to the mail server, the **IMAP** and **POP3** tests check whether messages can be polled from the mail server.

As depicted in [Figure 8.43](#) for executing the **SMTP** test to the mail server, you specify the **Host** and **Port** of the mail server, and then you select the **Protection** mechanism the mail server supports. With the **Authentication** option set to **None**, you can test whether the mail server can be reached and the secure connection can be established.

The screenshot shows the 'Request' configuration for an SMTP test. The 'Host' is set to 'mail.gmx.net', 'Port' to '587 (SMTP / STARTTLS)', and 'Protection' to 'STARTTLS Optional'. The 'Authentication' dropdown is set to 'None'. The 'Validate Server Certificate' checkbox is checked, while 'Check Mail Addresses' is unchecked. A 'Send' button is visible. Below the request section, the 'Response' area displays an error message in a red-bordered box:

```
① javax.mail.MessagingException: Could not convert socket to TLS
Cause: javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested
target
Cause: sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested
target
```

Figure 8.43 SMTP Connection Test with an Error

In [Figure 8.43](#), you see that an error occurred during the SSL handshake when connecting to the mail server. As in the TLS test, you have the option to execute the **SMTP** test without validating the server certificate by deselecting the **Validate Server Certificate** checkbox. Then you use the **Download** option for the certificates and add the root CA to the keystore using the keystore monitor. Afterward, the **SMTP** test should pass successfully. The **Response** screen displays whether the server certificate is valid.

If **Authentication** with user and password is used, either **Plain User/Password** or **Encrypted User/Password**, a new entry field named **Credential Name** appears where the name of the deployed credential needs to be entered (Figure 8.44). The **Response** screen displays whether the authentication with the credentials was successful.

The screenshot shows the SAP Cloud Platform Integration Operations interface for testing an SMTP connection. The top navigation bar includes tabs for TLS, SSH, SMTP, IMAP, and POP3, with the SMTP tab selected. The main area is divided into two sections: 'Request' and 'Response'.

Request:

- *Host: mail.gmx.net
- *Port: 587 (SMTP / STARTTLS)
- Protection: STARTTLS Optional
- Authentication: Plain User/Password (selected)
- *Credential Name: GMXUser
- Validate Server Certificate
- Check Mail Addresses
- *From: r_gmx.de
- To: m.k@sap.com

A blue 'Send' button is located at the bottom right of the Request section.

Response:

A green box highlights the message: "Successfully reached host at mail.gmx.net:587". Below this, detailed server information is listed:

- Server Version: gmx.com (mrgmx002) Nemesis ESMTP Service ready
- STARTTLS: Supported
- STARTTLS Connection: Established
- Authentication: Successful
- Server Certificate: Valid
- Mail From: Supported
- Mail To: Supported

Figure 8.44 SMTP Connection Test with Checking of Mail Addresses

A useful option in the SMTP test is to check the mail addresses used in the mail receiver channel. When selecting the **Check Mail Addresses** checkbox, two new entry fields are displayed: **From** and **To**. There you enter the mail addresses used in your integration flow in the mail receiver channel. The **SMTP** test checks if the mail server supports those addresses and shows the check result in the **Response** screen (Figure 8.44).

The **IMAP** and **POP3** tests support similar features as the **SMTP** test, except the check for the mail addresses. This is simply because this option isn't relevant for the mail sender adapter where messages are polled from a mail server.

In Figure 8.45, the **IMAP** test to *gmx.net* is shown as a sample. You see that the **Host** and **Port** of the mail server need to be entered together with the **Protection** mechanism, **Authentication**, and **Credential Name**.

Request

- *Host: `imap.gmx.net`
- *Port: `993 (IMAPS)`
- Protection: `IMAPS`
- Authentication: Encrypted User/Password Plain User/Password
- *Credential Name: `GMXUser`
- Validate Server Certificate
- List Folders
- Check Mailbox Content
- *Folder: `INBOX`

Response

Successfully reached host at `imap.gmx.net:993`

```

Server Version: IMAP server ready H migmx001 3700075029 IMAP-1N96ub-1efg5K3WAZ-015j4v
STARTTLS: Not Supported
Authentication: Successful
Server Certificate: Valid
Mails in Folder: All: 84, Unread: 8
Certificates:
1 dn=CN=imap.gmx.net, EMAILADDRESS=server-certs@fnd1.de, L=Montabaur, ST=Rhineland-Palatinate, O=f&1 Mail & ...
  issuerDN=CN=TeleSec ServerPass DE-2, STREET=Untere Industriestr. 20, L=Netphen, OID.2.5.4.17=57250, ST=Nordrhein-Westfalen
  subjectAlternativeNames=DNSName=imap.gmx.net, DNSName=imap.gmx.de
  validUntil=Sep 28, 2018, 01:59:59
  serialNumber=0x5226F7B1283F0E25
2 dn=CN=TeleSec ServerPass DE-2, STREET=Untere Industriestr. 20, L=Netphen, OID.2.5.4.17=57250, ST=Nordrhein-Westfalen
  issuerDN=CN=Deutsche Telekom Root CA 2, OU=T-TeleSec Trust Center, O=Deutsche Telekom AG, C=DE
  < >
  Download
Folders:
Archiv
Entwürfe
Gelöscht
Gesendet
INBOX
OUTBOX
  
```

Figure 8.45 IMAP Connection Test with Folders

For the **IMAP** test, the following check features are available:

- **Validate Server Certificate**

As in the **SMTP** test, deselecting this checkbox can test whether the mail server can

be reached at all. Furthermore, the provided mail server certificates can be downloaded and added to the keystore.

■ List Folders

If selected, the test lists all folders in the mail server for the user connected using the credentials. As depicted in [Figure 8.45](#), the **Folders** are shown at the end of the **Response** screen.

■ Check Mailbox Content

If selected, a new entry field named **Folder** appears, where the folder in the mail server must be entered. The test checks how many mail messages are available in the specified folder and how many of them are **Unread**. The result of the check is shown in the **Response** screen.

The **POP3** test is very similar to the **IMAP** test. As depicted in [Figure 8.46](#), the **Request** and **Response** details look almost identical to the **IMAP** test.

The screenshot shows the SAP Cloud Platform Integration Operations interface for performing a POP3 connection test. The top navigation bar includes links for TLS, SSH, SMTP, IMAP, and POP3, with POP3 currently selected. The main area is divided into Request and Response sections.

Request:

- *Host: pop.gmx.net
- *Port: 995 (POP3S)
- Protection: POP3S
- Authentication: Encrypted User/Password Plain User/Password
- *Credential Name: GMXUser
- Validate Server Certificate
- Check Mailbox Content

Response:

Successfully reached host at pop.gmx.net:995

Server Version: +OK POP server ready H migmx106 1N3ryq-1ejDD13d9k-010f1E

STARTTLS: Not Supported

Authentication: Successful

Server Certificate: Valid

Mails in Inbox: 84

Certificates:

```
1 dn=CN=pop.gmx.net, EMAILADDRESS=server-certs@1und1.de, L=Montabaur, ST=Rhineland-Palatinate, C=DE
issuerDN=CN=TeleSec ServerPass DE-2, STREET=Untere Industriestr. 20, L=Netphen, O=Deutsche Telekom Root CA 2, OU=T-TeleSec Trust Center, O=Deutsche Telekom AG
subjectAltName=DNSName=pop.gmx.net, DNSName=pop.gmx.de
validUntil=Sep 28, 2018, 01:59:59
serialNumber=0x9E937046E79E212A
2 dn=CN=TeleSec ServerPass DE-2, STREET=Untere Industriestr. 20, L=Netphen, O=Deutsche Telekom Root CA 2, OU=T-TeleSec Trust Center, O=Deutsche Telekom AG
```

Figure 8.46 POP3 Connection Test

The following check features are available for the POP3 test:

- **Validate Server Certificate**

As in the **SMTP** and **IMAP** tests, deselecting this checkbox can test whether the mail server can be reached at all. Furthermore, the provided mail server certificates can be downloaded and added to the keystore.

- **Check Mailbox Content**

The test checks how many mail messages are available in the **Inbox** folder. The result of the check is shown in the **Response** screen.

Note

POP3 doesn't support different folders, so only the **Inbox** folder is relevant. Because of this, the check for mailbox content doesn't offer an option to specify a dedicated folder but provides the mail messages contained in the **Inbox** folder. In addition, POP3 doesn't distinguish between read and unread mail, so all mail messages in the **Inbox** folder are shown.

Now that you know how to manage the security-relevant artifacts in the SAP Cloud Platform Integration tenant and how to test connections using transport-level security, we'll discuss some monitors that are relevant only in specific scenarios. Let's first look into scenarios using temporary data, such as data stores and JMS queues.

8.4 Manage Temporary Data

In many scenarios, temporary data needs to be stored; this data can be either whole messages, parts of messages, error messages, or configuration data. The main storage options available in SAP Cloud Platform Integration are data stores and JMS queues.

In the **Manage Stores** section in the monitoring dashboard, you find the monitors to handle the temporary data ([Figure 8.47](#)).

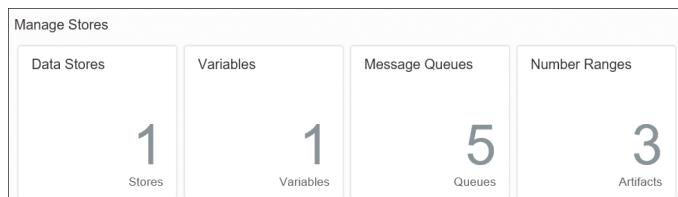


Figure 8.47 Manage Stores Section in the Monitoring Dashboard

8.4.1 Monitor Data Stores

The first tile in the **Manage Stores** section shows the **Data Stores** used in the deployed integration flows. Data stores are used by the following flow steps and adapters:

- **Data Store flow steps**

The following four flow steps work on data contained in the data store: **Write**, **Get**, **Select**, and **Delete**. Refer to [Chapter 5, Section 5.4.1](#), for more information about use cases for the data store.

- **XI adapter**

The XI adapter temporarily stores messages for exactly-once (EO) scenarios, either in a data store or in a JMS queue. If storage in a data store is configured in the XI adapter, those messages appear in the **Manage Data Stores** monitor.

As depicted in [Figure 8.48](#) and [Figure 8.49](#), the data store monitor consists of two sections: the master table on the left and the details table on the right. The master table on the left shows all data stores on the SAP Cloud Platform Integration tenant that contain messages. [Table 8.17](#) provides a detailed description of the attributes.

Data Stores without Messages Not Visible

Note that only data stores containing data are shown in the monitor because a data store is only created and kept in runtime as long as there are entries available in that specific data store. As soon as all data/messages are consumed from the data store, it's removed.

Name	Count	Status
Order	3	1 Overdue
Book_Integration_Flow_with_Splitter	1	

Figure 8.48 Manage Data Stores Monitor A

The screenshot shows a monitor interface for managing data stores. At the top, it says "Order Global" and has a "Delete" button. Below that, it displays "Overdue Entries: 1". A table follows, with a header row "Entries (3)" and columns "ID", "Status", "Due At", "Created At", and "Retain Until". The data rows are:

ID	Status	Due At	Created At	Retain Until
56d7fbcb-ad5f-44ca-9fd4-c3b91901fc9b	Waiting	Mar 06, 2018, 16:06:51	Mar 05, 2018, 16:06:51	Jun 03, 2018, 17:06:51
484489af-c98f-4fdc-bd3d-7ef1094ec54a	Overdue	Mar 06, 2018, 15:53:35	Mar 05, 2018, 15:53:35	Jun 03, 2018, 16:53:35

Below the table are several buttons: "Filter by ID" (with a magnifying glass icon), "Delete" (with a trash can icon), "Download" (with a download icon), and three other icons (refresh, gear, and list).

Figure 8.49 Manage Data Stores Monitor B

Attribute	Description
Name	The name of the data store as configured in the data store flow step. If the data store is used by an XI adapter, the data store name is generated per the following pattern: <participant>_<channel name>.
Visibility	The visibility of the data store as configured in the data store flow steps. The possible values are as follows: <ul style="list-style-type: none"> ■ Global: The data store can be used globally by all integration flows deployed on the tenant. ■ Integration Flow Name: The data store is used by the specified integration flow only. If the data store is used by an XI adapter, the integration flow name shown has the following pattern: <integration flow>/XI.
Number of Entries	The number of entries currently contained in the data store.
Number of Overdue Entries	The number of entries that already should have been processed but are still stored in the data store. This value is shown in red.

Table 8.17 Attributes of Data Stores on the Left Side of the Monitor

When you select a data store on the left side of the screen, all entries in this specific data store are shown on the right side in the details table. [Table 8.18](#) describes the attributes of single data store entries and their possible values.

Column	Description
ID	The ID of the entry in the data store. The entry ID can be specified in one of the following ways: <ul style="list-style-type: none">■ It can be specified in the data store Write step (e.g. by using a specific unique ID from the message payload).■ If it isn't specified in the data store Write step, a unique ID is generated by the runtime when the entry is written to the data store.■ The XI adapter generates a unique ID when the message is written to the data store.
Status	The status of the data store entry. The possible statuses are as follows: <ul style="list-style-type: none">■ Waiting: This is the status after the entry was written to the data store. The message/data is waiting to be consumed by another process or integration flow.■ Overdue: If the message wasn't consumed in the due time specified in the data store Write step, the message get the status Overdue. For such entries, the tenant administrator should check why the data isn't consumed (e.g., maybe the consuming integration flow doesn't run anymore) and solve the problem.
Due At	The date and time the entry is expected to be consumed latest. This time is configured in the data store Write step in the configuration field Retention Threshold for Alerting . If the message is processed by the XI adapter, the due time is set to two days by the adapter.
Created At	The date and time the entry was created in the data store.
Retain Until	The date and time the entry will be deleted. This time is configured in the data store Write step in the configuration field Expiration Period , which is 90 days per default. If the message is processed by the XI adapter, the expiration period is automatically set to 90 days by the adapter.

Table 8.18 Attributes of Data Store Entries

For the entries in the data stores, SAP Cloud Platform Integration offers the following actions at the top of the details table:

- **Filter by ID**

Filter for a specific ID. This can be very useful if the ID is specified in the Write step to be set from a unique ID from the incoming message payload. In this case, you can filter for this ID.

- **Delete**

Remove selected entries from the data store if not needed anymore.

- **Download**

Download selected entries. An *<ID>.zip* file is created in the download folder of your local PC containing the payload stored in the data store.

- **Reload** 

Reload the content of the details table.

- **Table Settings** 

Sort the entries and filter by the **Status** column.

- **Multi-select Mode** 

Select multiple entries at once. This option is useful when you need to delete multiple entries. Note that the *Download* action isn't available for multiple entries.

Now that you understand the data store monitor, we'll look at the variables monitor. You'll notice that several attributes are identical because the variables are technically also stored in the same data store table.

8.4.2 Monitor Variables

In some scenarios, variables are used to store configuration data, which is read and updated during message processing. Variables are used by the following flow steps:

- **Write Variables**

The **Write Variables** flow step is used to write or update variables in the runtime. Variables can have local or global visibility, which means that a variable can be read and updated only by the same integration flow or by all integration flows deployed on this tenant.

- **Content Modifier**

The **Content Modifier** flow step can read the value of a global or local variable and use it to set headers or properties.

The second tile in the **Manage Stores** section provides the monitoring options for the variables used in runtime. When you select the **Variables** tile, a table containing all

variables defined in the SAP Cloud Platform Integration tenant appears ([Figure 8.50](#)). [Table 8.19](#) provides a detailed description of all the columns of the table.

Overview / Manage Variables					
Variables (2)			Filter by Variable Name or Integration Flow		
Name	Visibility	Integration Flow	Updated At	Retain Until	Actions
globVar	Global		Apr 12, 2017, 17:18:49	May 17, 2018, 17:18:49	 
localVar	Integration Flow	Variables	Apr 12, 2017, 17:18:49	May 17, 2018, 17:18:49	 

Figure 8.50 Manage Variables Monitor

Column	Description
Name	The name of the variable as defined in the Write Variables flow step. The name is shown as a link. Selecting the link opens the variable in text format.
Visibility	The visibility of the variable as defined in the Write Variables step. The possible values are as follows: <ul style="list-style-type: none"> ■ Global: The variable can be used globally by all integration flows deployed on the tenant. ■ Integration Flow: The variable is used by the integration flow only.
Integration Flow	If Visibility is Integration Flow , in this column the integration flow using this variable is shown.
Updated At	The date and time the variable was last changed.
Retain Until	The date and time the variable will be deleted. This time is set to 400 days after the creation of the variable, but it's also updated along with any update of the variable. This means the variable is deleted 400 days after its last update.
Actions	Actions available for variables are as follows: <ul style="list-style-type: none"> ■ Download  : This option downloads the variable as a <code><variable>_<integration flow>.zip</code> file, which contains the variable in a <code>headers.prop</code> file, to the download folder of your local PC ■ Delete  : This option deletes the variable.

Table 8.19 Attributes of Variables

When you select the name of the variable, which is shown as a link, a popup opens providing the content of the variable in text format (Figure 8.51). You can also download the content using the **Download** button.



Figure 8.51 Content of a Variable

The two storage options described, data stores and variables, are based on the database of the SAP Cloud Platform Integration tenant. The third storage option for temporary data is a JMS queue based on the JMS Message Broker connected to the SAP Cloud Platform Integration tenant. For this option, the third tile in the **Manage Stores** section is important.

8.4.3 Maintain Message Queues

The **Message Queues** tile in the **Manage Stores** section shows the JMS queues used in the deployed integration flows. JMS queues are used by the following adapters:

- **JMS adapter**

The JMS adapter is used to directly process messages to and from JMS queues: the **JMS receiver** adapter writes messages to JMS queues, and the **JMS sender** adapter polls messages from JMS queues. Refer to [Chapter 5, Section 5.4](#), where we introduced reliable messaging using JMS queues.

- **AS2 adapter**

The **AS2** sender adapter temporarily stores messages during runtime processing in JMS queues.

- **XI adapter**

The **XI** sender and receiver adapter temporarily stores messages for EO scenarios, either in a data store or in a JMS queue. If storage in a JMS queue is configured in the **XI** adapter, those messages appear in the **Manage Message Queues** monitor.

Manage Message Queues Monitor Not Visible?

JMS messaging is available only with the SAP Cloud Platform Integration, Enterprise Edition, or if a JMS messaging license is purchased separately. If your SAP Cloud Platform Integration system isn't running with the Enterprise license, and no JMS Messaging is purchased, the JMS adapter doesn't appear in the list of available adapters.

Furthermore, you need to get a JMS Message Broker provisioned for your SAP Cloud Platform Integration tenant. The provisioning is triggered using a self-service in the account cockpit. Details about the provisioning of a JMS Message Broker can be found in the documentation for SAP Cloud Platform Integration (https://help.sap.com/viewer/product/CLOUD_INTEGRATION/Cloud) and in the “Provision Message Broker” blog in the SAP Community (www.sap.com/community.html).

The **Message Queues** tile is visible in the monitoring dashboard only after successful JMS Message Broker provisioning.

As depicted in [Figure 8.52](#) and [Figure 8.53](#), the **Manage message queues** monitor consists of two sections: the master table on the left and the details table on the right. The master table on the left side of the monitor shows all JMS queues available in the JMS Message Broker. [Table 8.20](#) provides a detailed description of the attributes shown for JMS queues.

The screenshot shows the 'Overview / Manage Message Queues' page. A blue header bar at the top displays the title. Below it, a status indicator says 'JMS Resources Ok.' with a 'Details' link. The main area features a master table with the following data:

Queues (2)	Ou	Actions
		✖️ 🔎
Name	Entries	Size (in MB)
Outbound_Queue1	4	0.010
Outbound_Queue2	0	0.000

Figure 8.52 Manage Message Queue Monitor A

Messages (4)		
<input type="text"/> Filter by Message ID <input type="button"/>		<input type="button"/> Retry <input type="button"/> Delete <input type="button"/> Download <input type="button"/> <input type="button"/> <input type="button"/> <input type="button"/>
JMS Message ID	Message ID	Status
ID:10.120.41.225b651163399ce10b0:3	AFqvpisUVUo9uzWJtCITWMmZAe50	Failed
Due At: Mar 21, 2018, 12:59:39		
Created At: Mar 19, 2018, 12:59:39		
Retain Until: Jun 17, 2018, 14:00:07		
Retry Count: 1178		
Next Retry On: May 07, 2018, 11:31:02		

Figure 8.53 Manage Message Queue Monitor B

Attribute	Description
Name	The name of the JMS queue as configured in the JMS adapter. If the JMS queue is used by an AS2 or XI adapter, the queue name is generated automatically as per the following pattern: AS2_<Integration Flow Name>_<Channel Name>_<Guid> or XI_<Integration Flow Name>_<Channel Name>_<Guid>.
Entries	The number of entries currently contained in the JMS queue.
Size	The overall size of all messages contained in the JMS queue.

Table 8.20 Attributes of JMS Queues on the Left Side of the Monitor

Automatic Queue Creation

The JMS queues are created automatically in the JMS messaging instance during deployment of the first integration flow using a new JMS queue name. Unlike the data store monitor, JMS queues show up in the monitor also if there are no messages available in the queue.

The actions available for queues can be accessed via the **Actions** button on top of the queues table ([Figure 8.54](#)). The following actions are available:

- **Retry**

Restarts all messages in the selected queue.

- **Delete**

Deletes the selected queue. Be careful with this option: it deletes the queue with all

the messages in the queue. To re-create the queue, the integration flow needs to be redeployed.

■ **Where-Used**

Shows the integration flows using the selected queue.

■ **Check**

Some consistency checks for JMS queue usage are executed; for example, it's checked if some queues aren't used by any integration flows anymore. A detailed description of the checks and the check results can be found in the "Checks in JMS Message Queue Monitor" blog in the SAP Community (www.sap.com/community.html).

■ **Reload** 

Reloads the content of the table containing the JMS queues.

■ **Sort** 

Offers sorting options for the available columns.

Queues (11)			Filter by Name	Actions	M
Name	Entries	Size			
AS2_AS2Sender_AS2_04509	0	0.00	Retry		
a0d_cc37_337f_9709_ee3c35	0	0.00	Delete		
etlk212			Where-Used		
Inbound_Queue	1	0.00	Check		
InbPreProcess_Queue	0	0.00	Reload		
OOM_Final	0	0.00	Sort		

Figure 8.54 Actions for Queues

Queues Aren't Deleted Automatically

JMS queues aren't deleted automatically during undeployment of the integration flow because there may still be messages in the JMS queue, and deleting the queue would delete them as well, which would lead to data loss. Only the owner of the scenario knows if the JMS queue can be deleted with all its content or if it's still required.

Upon selecting a specific queue on the left side of the screen, all messages in this specific queue are shown on the right side in the details table. [Table 8.21](#) describes the attributes of single messages in the JMS queue and their possible values.

Column	Description
JMS Message ID	The ID of the entry in the JMS queue. The JMS Message ID is a unique ID generated by the runtime when the entry is written to the JMS queue.
Message ID	The ID of the MPL. The Message ID is shown as a link, and selecting the link opens the MPL in the message processing monitor.
Status	<p>The status of the message. The possible statuses are as follows:</p> <ul style="list-style-type: none"> ■ Waiting: This is the status after the entry was written to the JMS queue. The message is waiting to be consumed by another process or integration flow. ■ Failed: The last processing of the message ended with an error. Follow the link to the MPL to get the details of the error. ■ Blocked: The message caused several runtime node outages and isn't processed anymore. The dead letter queue feature is described in detail in the “Configure Dead Letter Handling in JMS Adapter” blog in the SAP Community (www.sap.com/community.html). ■ Overdue: The message gets this status if it wasn't consumed in the due time. For such entries, the tenant administrator should check why the message isn't consumed (e.g., maybe the consuming integration flow doesn't run anymore) and solve the problem.
Due At	The date and time the message is expected to be consumed latest. This time is configured in the JMS receiver adapter in the configuration field Retention Threshold for Alerting . If the message is processed by the AS2 or XI adapter, the due time is set to 2 days by the adapter.
Created At	The date and time the entry was created in the JMS queue.
Retain Until	The date and time the message will be deleted. This time is configured in the JMS receiver adapter in the configuration field Expiration Period , which is 90 days per default. If the message is processed by the AS2 or XI adapter, the expiration period is automatically set to 90 days by the adapter.
Retry Count	The numbers of retries executed for this message.
Next Retry On	The date and time the message will be retried next.

Table 8.21 Attributes of JMS Queue Entries

The following actions are available on the top of the details table for messages stored in JMS queues:

- **Filter**

Filters for a specific message ID, which can be either the JMS message ID or the MPL ID.

- **Retry**

Restarts the selected messages.

- **Delete**

Deletes selected messages from the JMS queue. Only use this option if the messages aren't needed anymore.

- **Download**

Downloads selected entries. A *<ID>.zip* file is created in the download folder of your local PC containing the message stored in the JMS queue.

- **Reload** 

Reloads the content of the details table.

- **Table Settings** 

Allows sorting of the entries and filtering by the **Status** column.

- **Multi-select Mode** 

Allows you to select multiple entries at once. This option is useful when you need to delete multiple entries. Note that the **Download** action isn't available for multiple entries.

Retry of Blocked Messages

When triggering a retry for messages with status **Blocked**, keep in mind that those messages got the blocked status because they could not be processed due to the fact that the runtime node crashed multiple times. Because of this, it's very likely that the message caused the crash. Another retry may cause another runtime node crash.

JMS Resource Check

JMS resources in the connected JMS Message Broker instance are limited. Therefore, you need to monitor them carefully, especially if the load for scenarios using JMS queues increases. If the JMS resources are exhausted, you'll encounter errors during runtime processing. This needs to be avoided. Details about the JMS resources available with the SAP Cloud Platform Integration licenses can be found in the "JMS Resource and Size Limits" blog in the SAP Community (www.sap.com/community.html).

To monitor the JMS resources, an info bar is available at the top of the message queues monitor screen that shows the result of the JMS resource check (refer back to [Figure 8.52](#)). Three severity levels are available for the info bar:

- **Info** ⓘ

The blue info message tells you that the used JMS resources are within the purchased limits. No action is necessary in this case.

- **Warning** ⚠

The orange message indicates that at least one of the JMS resources is in a critical state.

- **Error** ⓘ

If at least one JMS resource is exhausted, a red error message is shown.

When you click the **Details** link, the different JMS resources with the current statuses are shown ([Figure 8.55](#)). If dedicated resources are critical, the tenant administrator together with the scenario owner should take actions to bring the resources back to normal. If JMS resources are exhausted, message processing is affected, and immediate action is required.

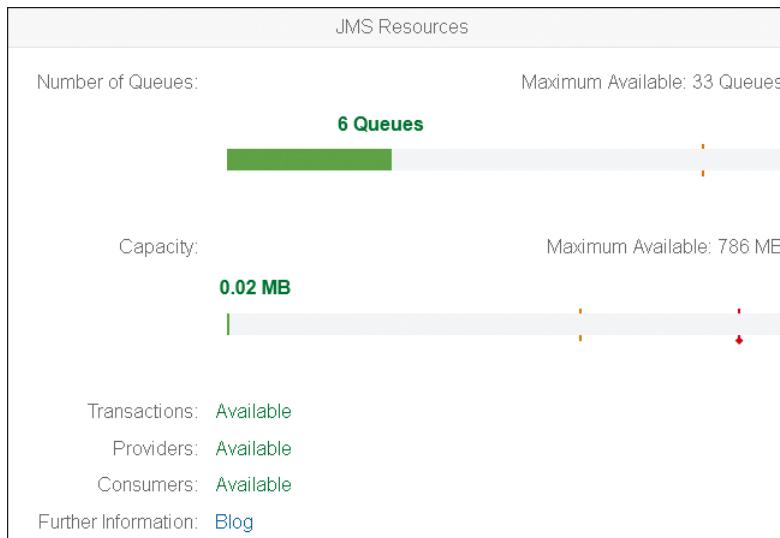


Figure 8.55 Details of JMS Resources

Let's take a detailed look at the **JMS Resources** screen as shown, what the options mean, and what needs to be done if they are critical or exhausted:

■ Number of Queues

This is the number of JMS queues created on the JMS Message Broker during integration flow deployment. If the limit is reached on the JMS Message Broker, deployments of integration flows using new JMS queues will fail because the new JMS queue can't be created on the JMS Message Broker. What needs to be done now? You simply have to clean up the JMS queues if new queues are required. Use the **Check** action (refer to [Figure 8.54](#)) to find out if there are unused queues and delete them. Furthermore, you can check if all integration flows using JMS queues are still required; if not, undeploy them, and delete the corresponding queues.

■ Capacity

There is an overall storage capacity available in the JMS Message Broker instance connected to your tenant for all the queues. If 100% of the available capacity is used, no messages can be stored in the JMS queues anymore, and the processing of the messages will end in runtime errors. To avoid this, check why there are so many messages in the JMS queues and why the messages aren't consumed. For example, the consuming integration flow might not run, or dedicated queues aren't needed anymore because they just contain messages from an old scenario. Check and correct the problem as soon as the **Critical** limit is reached to avoid downtimes in the deployed scenarios.

■ Consumers

Consumers are created in the JMS Message Broker from **JMS** sender adapters to consume messages from the JMS queue. For each JMS queue used in a **JMS** sender channel, as many consumers are created as concurrent processes are configured in the **JMS** sender channel in the **Number of Concurrent Processes** parameter (see [Chapter 5, Section 5.4.1](#)). If several runtime nodes are started in the SAP Cloud Platform Integration cluster, those many consumers are created from each runtime node. If the value of consumers gets **Critical**, you need to reduce the parallelism, either by reducing the **Number of Concurrent Processes** in the **JMS** sender channels or by reducing the number of runtime nodes started in the cluster.

■ Providers

Providers are created by the **JMS** receiver adapter in the JMS Message Broker to store messages in a JMS queue. As many providers are created as messages are send to the JMS queue in parallel. If the value of providers reaches the **Critical** limit, the parallelism of the inbound processing is too high. You need to reduce the number of parallel inbound calls from sender systems sending messages to scenarios using JMS queues.

■ Transactions

To consistently process messages in SAP Cloud Platform Integration, JMS transactions are required in the JMS Message Broker to roll back the processing if there are errors (see [Chapter 5, Section 5.4.2](#), for detailed information). As many transactions are created in the JMS Message Broker as consumers and providers are under processing in parallel. If the limit for transactions gets **Critical**, you need to reduce the parallelism for the consumer and/or provider connections.

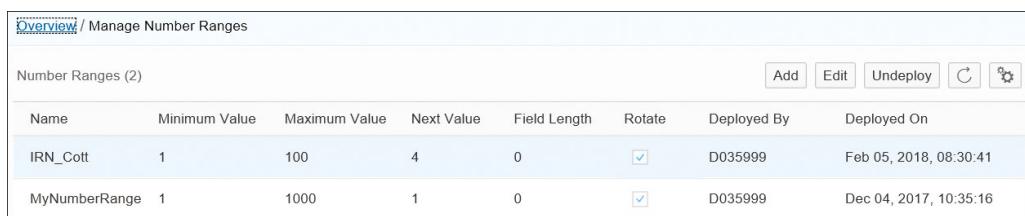
If the mentioned actions don't solve the JMS resource shortage, you may need to purchase more JMS messaging units and add them to the JMS Message Broker instance connected to the tenant to increase the JMS resources increased in the JMS Message Broker.

The last tile in this section is the **Number Ranges** tile, which was used already in [Chapter 7](#), when we configured a complete B2B integration.

8.4.4 Maintain Number Ranges

As explained in detail in [Chapter 7, Section 7.3.4](#), number ranges are required to define unique interchange numbers for each document for Electronic Data Interchange (EDI) processing. To do this, a number range object (NRO) can be used that defines the length of the number and the minimum and maximum value. The NRO is defined and monitored in the **Number Ranges** monitor and used in runtime when an integration flow is executed.

Selecting the **Number Ranges** tile opens the list of defined NROs in a table, as shown in [Figure 8.56](#). [Table 8.22](#) provides a detailed description of the available attributes of the NROs.



Number Ranges (2)							
Name	Minimum Value	Maximum Value	Next Value	Field Length	Rotate	Deployed By	Deployed On
IRN_Cott	1	100	4	0	<input checked="" type="checkbox"/>	D035999	Feb 05, 2018, 08:30:41
MyNumberRange	1	1000	1	0	<input checked="" type="checkbox"/>	D035999	Dec 04, 2017, 10:35:16

Figure 8.56 Number Ranges Monitor

Column	Description
Name	Name of the NRO. This name is used in the integration flow configuration to refer to the NRO.
Minimum Value	Minimum value; should be greater or equal 0.
Maximum Value	Maximum value allowed in this NRO.
Next Value	The next value used in runtime when invoking the NRO.
Field Length	Field length of the number. Leading zeros are added to the current value to achieve a fixed field length in runtime. This means if the current value is 5, the unique number used in runtime is 0005 for a field length of 4. If Field Length is 0, no leading zeros are added. Maximum value allowed for this field is 99.
Rotate	If this flag is set, and the number range reaches the specified maximum value, the current value resets to the specified minimum value.
Deployed By	The user who deployed the NRO.
Deployed On	Date and time the NRO was deployed.

Table 8.22 Attributes of Number Range Objects

The following actions are available for NROs at the top of the table:

- **Add**
Opens the **Add Number Range** dialog to define and deploy a new NRO ([Figure 8.57](#)).
- **Edit**
Opens the **Edit** dialog to adjust the definitions of the selected NRO.
- **Undeploy**
Removes the NRO from the tenant.
- **Reload** 
Reloads the content of the table.
- **Table Settings** 
Allows sorting of the entries and filtering by the name of the NRO or by the user who deployed the NRO.

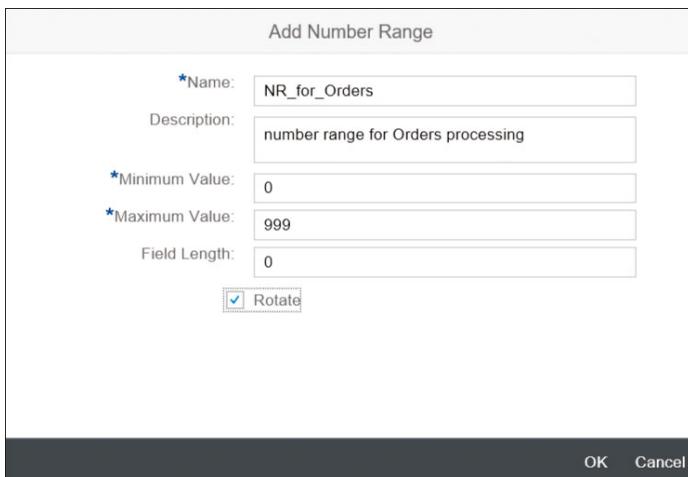


Figure 8.57 Add Number Range Dialog

With this, we complete the **Manage Stores** section and start to explore the **Access Logs** section in the monitoring dashboard where access to different logs is provided.

8.5 Access Logs

As depicted in Figure 8.58 the **Access Logs** section offers access to different logs created in SAP Cloud Platform Integration. Logs get important, for example, when specific errors need to be analyzed that occur during runtime processing. Often detailed logs are requested by SAP support to analyze specific error situations. Therefore, it's important to know which logs are available and what they contain.

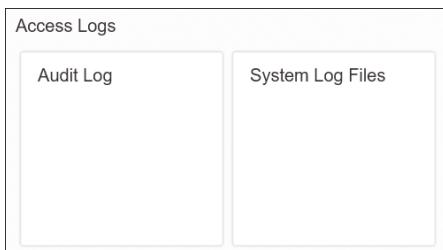


Figure 8.58 Access Logs Section in Monitoring Dashboard

There are two tiles available: the **Audit Log** and **System Log Files**. Whereas the audit log is more relevant for auditing purposes, the provided system logs are mainly required for error analysis.

8.5.1 Monitor Audit Log

Audit logs are security-relevant records. SAP Cloud Platform Integration needs to log all system changes in the SAP Cloud Platform Integration tenant. The changes logged are configuration changes, deployments, or deletions of security artifacts; log level changes; and so on. All those changes are logged in the audit log. The monitoring dashboard offers access to the audit log via the **Audit Log** tile (refer to [Figure 8.58](#)).

Selecting the **Audit Log** tile opens a table in a new screen showing all audit logs of the last hour ([Figure 8.59](#) and [Figure 8.60](#)). However, you can adjust the **Time Range** at the top of the table to see audit logs for a longer time period. In [Figure 8.59](#), for example, you see the audit logs for the **Past 24 Hours**. See [Table 8.23](#) for a detailed description of the attributes for audit log entries.

Audit Logs (3)						
Time	Action	Object Type	Object Name	User	Source	Filter by Object Name, User or Source
Mar 09, 2018, 02:15:00	Delete	Message Store Entries	messageStoreEntries	SAP	SAP	<input type="button" value="Filter"/>
Mar 08, 2018, 22:08:11	Change	Configuration Parameter	Name=MplLogLevel;Ns=Book_Integration_Flow_with_Splitter;Type=RBusiness	SAP	SAP	<input type="button" value="C"/> <input type="button" value="↑"/>
Mar 08, 2018, 12:37:48	Create	X.509 Certificate	biz	SAP	SAP	

Figure 8.59 Audit Log Monitor A

Audit Logs (3)			
Object Name	User	Source	Filter by Object Name, User or Source
messageStoreEntries	SAP	SAP	<input type="button" value="Filter"/>
Name=MplLogLevel;Ns=Book_Integration_Flow_with_Splitter;Type=RBusiness	SAP	SAP	<input type="button" value="C"/> <input type="button" value="↑"/>
biz	SAP	SAP	

Figure 8.60 Audit Log Monitor B

Retention Time

The retention time of audit logs is 30 days, after which the audit log is automatically deleted.

Column	Description
Time	The date and time the audit log entry was created.
Action	<p>The action performed on the system. Possible values are as follows:</p> <ul style="list-style-type: none"> ■ Create: Creation of artifacts or configurations, for example, certificates in the keystore. ■ Change: Changing of configuration parameters, for example, log levels of an integration flow. ■ Delete: Deletion of artifacts, for example, certificates, variables, or messages. ■ Read: Read access to artifacts, for example, to message payloads using the trace feature or to messages or variables in temporary storages.
Object Type	Type of the object that was changed, for example, Message , Variable , Configuration Parameter , or X509 Certificate .
Object Name	Name or ID of the object that was changed, for example, ID of a message, name of a X.509 certificate, or name of a variable.
User	User who triggered the change. If an SAP user triggered the change, SAP is displayed.
Source	IP address the change was triggered from. If an SAP user triggered the change, SAP is displayed.

Table 8.23 Attributes of Audit Log Entries

You can sort the entries using the **Sort**  icon or filter by **Object Name**, **User**, or **Source** using the **Filter** field at the top of the table.

Now let's look at the second tile in the **Access Logs** section, which provides access to system logs.

8.5.2 Check System Log Files

Sometimes, the error messages shown in the MPL aren't sufficient to fully understand the root cause of an error or SAP support requires detailed information for analyzing an error. In such cases, you need to have a look into the system logs written during message processing on the runtime node. The **System Log Files** tile provides access to those technical system logs.

Selecting the **System Log Files** tile opens a table containing the most important log files of the runtime nodes running in your cluster ([Figure 8.61](#)) in the **Log Files** tab. [Table 8.24](#) provides a detailed description of the attributes of the system log files.

Log Files (51)					Filter by Name
Name	Log Type	Updated At	Size	Actions	
ljs_trace_f43632e_2018-03-09.log	CP Default Trace	Mar 09, 2018, 09:24:34	3.858 KB		
ljs_trace_0930e62_2018-03-09.log	CP Default Trace	Mar 09, 2018, 09:24:34	3.862 KB		
http_access_0930e62_2018-03-09.log	HTTP Access Log	Mar 09, 2018, 09:24:29	27 KB		

Figure 8.61 Monitor System Logs Screen

Retention Time

The retention time of the system log files is seven days. After that, the system logs are automatically deleted.

Column	Description
Name	Name of the log file. The name has the following pattern: <log type>_<process ID>_<date>.log. The process ID is shown in the message processing monitor, so that it's easy for you to identify the system log in which details about the message processing are logged.

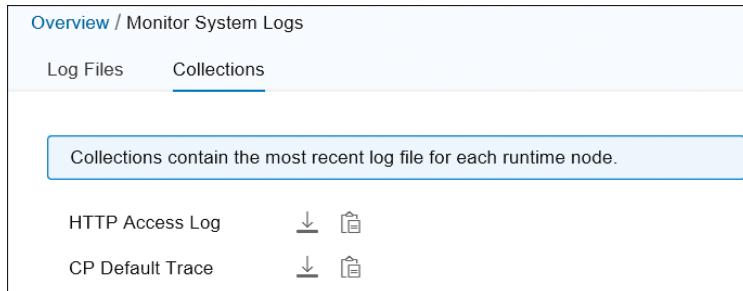
Table 8.24 Attributes of Log Files

Column	Description
Log Type	Type of the log file. Possible values are as follows: <ul style="list-style-type: none">■ CPI Default Trace: System trace file.■ HTTP Access Log: Log file containing the HTTP inbound requests. One log file is available for each runtime node.
Updated At	Date and time the log file was updated.
Size	Size of the log file.
Actions	Actions available for the log file: <ul style="list-style-type: none">■ Download  : Downloads the log file.■ Copy Download URL  : Copies the download URL to the clipboard.

Table 8.24 Attributes of Log Files (Cont.)

You can sort the entries using the **Sort**  icon or filter by **Name** using the **Filter** field at the top of the table.

In the **Collections** tab, you can access collections of the system log files ([Figure 8.62](#)). You can download  the collections as an archive or copy the download URL . Downloading the collection, you get a *.zip archive containing the latest log files.

**Figure 8.62** Collections Tab in System Logs

With this, we complete the **Access Logs** section and continue with the next, very specific section that handles locks created during message processing.

8.6 Manage Locks

On the SAP Cloud Platform Integration tenant, locks are written by some components to ensure efficient and consistent message processing in the runtime. To monitor those message processing locks, you can use the **Message Locks** tile in the **Manage Locks** section of the monitoring dashboard (Figure 8.63).

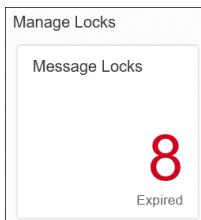


Figure 8.63 Manage Locks Section in the Monitoring Dashboard

Monitor Message Locks

During message processing in SAP Cloud Platform Integration, some adapters write entries into an in-progress repository to avoid the same message being processed multiple times in parallel, for example, by different runtime nodes, or to avoid large messages leading to out-of-memory issues on runtime nodes again and again.

The following adapters write such message processing locks:

- **SFTP sender adapter**

To prevent double processing of files by the **SFTP sender adapter**, a lock entry is written to the in-progress repository each time a file is processed by a runtime node. As long as this lock entry exists, no other runtime node can access the file. After message processing, independent of whether the processing ended in **Completed** or **Failed**, the lock is removed by the runtime. If the runtime node crashes during processing of the message, for example, caused by an out-of-memory error, the message processing lock isn't removed, and upon restart of the runtime node, the message is retried by the adapter. If the runtime node crashes two more times during processing of this message, the message is taken out from processing, and the message lock remains in the in-progress repository. Manual action is required by the tenant administrator together with the scenario owner to check why this file could not be processed, for example, maybe it's too large to be processed by the integration flow. In this case, it should be removed from the polling directory and be split into smaller files. The lock can then be released.

- **JMS sender adapter**

The **JMS** sender adapter uses the in-progress repository only if the **Dead Letter Queue** handling is switched on (for details, refer to [Chapter 5, Section 5.4.1](#)). If the runtime node crashes during processing of the message, for example, caused by an out-of-memory error, the message processing lock isn't removed, and upon restart of the runtime node, the message is retried by the JMS sender adapter. If the runtime node crashes two more times during processing of this message, the message is taken out from processing and marked as **Blocked** in the queue monitor ([Section 8.4.3](#)). The message lock in the message lock monitor is removed.

- **AS2 sender adapter**

Because the **AS2** sender adapter uses JMS queues to store the message during processing, the same behavior as explained for the JMS sender adapter holds true for the AS2 sender adapter. The in-progress repository is only used if the **Dead Letter Queue** handling is switched on. The message is marked as **Blocked** in the queue monitor, and the message lock is removed from the message lock monitor if the runtime node crashed multiple times during processing of this message.

- **XI sender and receiver adapters**

The XI adapters, both sender and receiver, can use either JMS queues or data stores for temporary storage of the messages during processing.

- If JMS queues are used, the same behavior as explained for the **JMS** and **AS2** sender adapters holds true for the **XI** adapter as well. The in-progress repository is only used if the **Dead Letter Queue** handling is switched on. The message is marked as **Blocked** in the queue monitor, and the message lock is removed from the message lock monitor if the runtime node crashed multiple times during processing of this message.
- If the data store is used as temporary storage, the in-progress repository is always used to avoid a message crashing the runtime node again and again; there is no configuration option available. If the runtime node crashes three times during processing of this message, the message is taken out of processing, and the message lock remains in the in-progress repository. Those entries stay in the message locks monitor until you manually release them. Before releasing the lock, check the corresponding message in the data store monitor, for example, to see if it's too large to be processed by the integration flow. In this case, you need to request the sender of this message to split the message and send smaller messages instead. The message in the data store can then be removed, and the lock in the lock monitor released.

You can monitor the in-progress repository entries in the **Message Locks** tile. When you select the **Message Locks** tile, a table containing all currently set message processing locks is shown (Figure 8.64). Table 8.25 provides a detailed description of the available attributes of the message processing locks.

Message Locks (8)				
Component	Source	Entry	Created At	Expires At
JMS	JMS:QueueForRetry	ID:10.120.32.173892915d361975350:1	Jul 12, 2017, 11:38:55	Jul 12, 2017, 11:55:35
JMS	JMS:QueueForRetry	ID:10.120.32.173892915d361975350:2	Jul 12, 2017, 11:56:59	Jul 12, 2017, 12:13:39

Figure 8.64 Message Locks Monitor

Column	Description
Component	Component that set the lock. Possible values are as follows: <ul style="list-style-type: none"> ■ SFTP: The lock was set by the SFTP sender adapter. ■ JMS: The lock was set by the JMS component; this is the case for JMS adapter, AS2 adapter, and XI adapter when JMS queues are used for temporary storage. ■ XI: The lock was set by the XI adapter when using the data store for temporary storage.
Source	Identifies the source of the lock. Possible values are as follows: <ul style="list-style-type: none"> ■ For SFTP: <sftp user>@<sftp server> ■ For JMS: JMS:<queue name> ■ For XI: XI_<integration flow name>.<participant name>_<channel name>
Entry	Identifies the locked object, for example, the message or file name: <ul style="list-style-type: none"> ■ For SFTP: directory1/dir2/test.xml ■ For JMS: ID:<JMS message ID> (following the link opens the JMS Message in the queue monitor) ■ For XI: <Entry ID from data store>
Created At	Date and time the lock was set.
Expires At	Date and time possible retries expire. After this time, no retry is triggered anymore. The message can only be retried by releasing the lock.

Table 8.25 Attributes of Message Locks

The following actions are available for message processing locks at the top of the table:

■ **Filter**

Filters the table by specifying values for the **Source** or the **Entry** column. Using this option, you can, for example, search for a specific message ID or a file name.

■ **Release**

Releases the lock and so triggers another retry of the message or file.

■ **Reload** 

Reloads the content of the table.

■ **Table Settings** 

Allows sorting of the entries and filtering by the **Component** column.

■ **Multi-select Mode** 

Allows you to select multiple entries at once. This option is useful when you need to delete multiple entries.

Releasing Locks Can Lead to a Runtime Node Crash

When releasing locks, keep in mind that those messages got the lock because they could not be processed due to multiple runtime node crashes. Because of this, it's very likely that the message caused the crash. Releasing the lock triggers another retry of the file or message and so may cause another runtime node crash.

With this we complete the section about message processing locks and also end the journey through all the monitors available in SAP Cloud Platform Integration. You should now feel confident to operate the SAP Cloud Platform Integration tenant and understand all the monitors available.

8.7 Summary

No one wants to build a castle that only lasts one day. After spending a lot of time building integration scenarios to help your business reach its integration goals, you'll want to ensure that the business continues to enjoy its benefits for a long time. Your castle, as such, requires regular checks and maintenance work to stay in optimal condition. The same applies for your SAP Cloud Platform Integration platform, which requires maintenance by monitoring the running integration flows and the data stored temporarily.

Given that SAP Cloud Platform Integration is a cloud-based platform, most operational tasks (e.g., health checks) are performed by SAP, as the service provider. This chapter focused on operational activities performed by customers. In the chapter, we explored the different features and monitors that can be used in SAP Cloud Platform Integration for monitoring and operational purposes. The monitors were discussed in detail and the required scenarios were outlined as well.

After reading this chapter, you should be able to monitor all integration flows deployed in the tenant and use the logging and tracing feature to analyze the runtime of the integration flows. You're now well equipped to manage the different security artifacts, such as certificates, that are needed for your integration flows. In addition, you know how to monitor temporary data, such as messages and variables, stored in the SAP Cloud Platform Integration tenant's database or in JMS queues.

With this we have covered the lifecycle of developing integration scenarios in the web designer of Cloud Integration and monitoring them in the monitoring dashboard. In the next chapter we will take you into the world of using the APIs offered by SAP Cloud Platform Integration to develop and monitor integration content.

Chapter 9

Application Programming Interfaces

SAP enables the consumption and provision of application programming interfaces (APIs) to expose different functionalities to the outside world. The chapter dives into the specifics of an API within the context of SAP Cloud Platform Integration, presents currently available features, and explores the APIs that are currently available for customers.

In today's connected world, pretty much everything is at our fingertips. From traditional desktops, mobile phones, and connected devices (e.g., watches), we're able to purchase goods, write articles, and book flight tickets. But how does data move from application A to B, for instance, when booking a flight ticket or a car? The hidden enablers in most cases are known as application programming interfaces (APIs).

An API within the context of integration is an interface through which data exchange is made possible between applications. APIs are used to expose functionalities (or programmable interfaces) to the outside world through a service path or URL.

Simply put, an API takes a request from the caller, performs a task on a server application, and returns a response to the caller. The application providing the API is known as a *service provider*. And the application or user using the API is generally called a *service consumer*.

This chapter introduces the Java and OData APIs provided by SAP Cloud Platform Integration and explains how to use them. The chapter further explores SAP Cloud Platform API Management and how to use it together with SAP Cloud Platform Integration.

9.1 Introduction

There is a lot to be said about APIs, and this topic deserves its own book. The descriptive introduction in this section is only intended to give you a brief overview.

After having read the introductory information, if you're familiar with the topic of integration, you might be wondering what the difference is between an API and a traditional web service. [Table 9.1](#) points some differences out between an API and a web service.

Aspects	Web Service	API
Network	Needs a network connection for its operation	Can also operate offline
Protocols	SOAP, REST, XML-RPC	SOAP, REST, but can also communicate via CURL
Exposed via	XML over HTTP	JAR, DLL, XML, or JSON over HTTP

Table 9.1 Comparing Web Services to APIs

Furthermore, besides the differences mentioned in [Table 9.1](#), all web services can be considered APIs, but not all APIs can be considered web services.

A web service is a type of API that almost always operates over HTTP (hence the **web** part of the name). However, some web services, such as the Simple Object Access Protocol (SOAP), can use alternate transports, for example, Simple Mail Transfer Protocol (SMTP). The official W3C definition mentions that web services don't necessarily use HTTP, but this is almost always the case and is usually assumed unless mentioned otherwise.

On the other hand, it can be argued that every bit of function ever created, be it in a Dynamic Link Library (DLL), Java Archive (JAR), web service, or plain code is an API. APIs can use any type of communication protocol and aren't limited to HTTP like web services are.

Now that you have a good high-level awareness of what APIs are, let's explore the Java APIs that are currently provided by SAP Cloud Platform Integration.

9.2 Java APIs Provided by SAP Cloud Platform Integration

SAP Cloud Platform Integration provides a number of Java-based APIs to access and control the processing of messages on your tenant. At the time of publishing this

book, it's possible to use Groovy or Java Script as the programming language to access these APIs. Accessing the functionality provided by these APIs in a script step or while creating a user-defined mapping function can be handy ([Section 9.3](#)).

Furthermore, the APIs can be used while developing a custom SAP Cloud Platform Integration adapter (using the Adapter Development Kit [ADK]). Note, however, that if there is a custom adapter, Java is used as a development language. We discuss the ADK in [Chapter 6](#).

The existing APIs can be classified under the following categories:

- **Generic API**

Complete and parent set of APIs covering various features. These APIs are kept under the package `com.sap.it.api`. See [Table 9.2](#) for a list of interfaces contained in this package.

- **Message API**

Provides APIs to access properties of a message. These APIs are kept under the package `com.sap.it.api.msg`. See [Table 9.2](#) for a list of interfaces contained in this package.

- **Script API**

Provides APIs to control scripts.

- **Mapping API**

Provides APIs to control mappings. These APIs are contained under the package `com.sap.it.api.mapping`. See [Table 9.2](#) for a list of some commonly used packages and their corresponding interfaces.

Package	Interface	Description
<code>com.sap.it.api.msg</code>	<code>ExchangePropertyProvider</code>	Provides access to the properties of a message exchange.
<code>com.sap.it.api.msg</code>	<code>MessageSizeInformation</code>	Provides information about the size of a message.
<code>com.sap.it.api.mapping</code>	<code>MappingContext</code>	Mapping context object to be provided to mapping user-defined functions (UDFs).

Table 9.2 API Packages and Interfaces

Package	Interface	Description
com.sap.it.api.mapping	Output	Class used in advanced UDFs (execution type All values of Context or All values of a Queue) to return the result of a function.
com.sap.it.api.mapping	ValueMappingApi	Used to execute value mapping with the given parameters.
com.sap.gateway.ip.core.customdev.util	Message	Accesses the exchanged message. The API provides an extensive set of functionalities, including manipulation of attachment, reading and changing payload, and retrieval of message properties such as size, header, and so on.
com.sap.it.script.logging	ILogger	Performs different operations on the logs (e.g., writing message logs).
com.sap.it.public.generic.api	ITApiException, Key-storeService, Secure-StoreService, UserCredential	Global API covering a wide range of functionalities, including; access key storage services, access the deployed user credentials, and access exception object.
com.sap.it.api.pd	PartnerDirectoryService	Performs different operations on the Partner Directory parameter values, the partner IDs, the alternative partner IDs, and the authorized users of a partner.
com.sap.it.api.pd	BinaryData	Container for binary data relevant for Partner Directory binary parameters.

Table 9.2 API Packages and Interfaces (Cont.)

For a full list of interfaces and classes, refer to the JavaDocs at: <http://bit.ly/2LFY6V7>.

Note

Note that the APIs summary of packages listed in [Table 9.2](#) relates to API version 2.12.0.

To illustrate the usage of the APIs, let's next look at UDFs in a mapping.

9.3 Using the Java API in a User-Defined function

Imagine that you need a message mapping to perform a complex transformation between a source and target message. Furthermore, imagine that none of the existing standard functions can fulfill the needed logic. That is when a user-defined function (UDF) comes to the rescue.

A UDF is a custom function that is built to cater to special mapping needs that can't be expressed by the mapping functions predefined in the mapping editor. In SAP Cloud Platform Integration, a UDF can be built using Groovy or Java Script. UDF generally uses mapping-related APIs listed in [Table 9.2](#) to transform the messages. (To see an illustration of a UDF in use, go to the example in [Chapter 4, Section 4.3](#).)

In this section, we've built an integration flow that consumes an external OData service. After invoking the OData service, we received a response that looks like the one presented in [Figure 9.1](#).

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header/>
  <soap:Body>
    <nsl:OrderShippingDetails_MT xmlns:nsl="http://hci.sap.com/demo">
      <orderNumber>10249</orderNumber>
      <customerName>Toms Spezialitäten</customerName>
      <shipCity>Münster</shipCity>
      <shipStreet>Luisenstr. 48</shipStreet>
      <shipPostalCode>44087</shipPostalCode>
      <shipCountry>DE</shipCountry>
      <shipDate>1996-07-10T00:00:00.000</shipDate>
    </nsl:OrderShippingDetails_MT>
  </soap:Body>
</soap:Envelope>
```

Figure 9.1 Response of the Integration Flow of [Chapter 4](#)

Let's hypothetically imagine that, with regard to the consumer application, you would prefer not to have the empty spaces in the element `shipStreet` between the street name and the house number. Furthermore, the consumer requires the street field to be appended with the order number as a postfix, and the entire field should be returned in uppercase. Looking at the example in [Figure 9.1](#), it means that the value `Luisenstr. 48` should be transformed to `LUISENSTR.48_10249` instead.

You can implement this requirement by using a combination of predefined functions in the mapping. However, for the sake of illustration, let's try to achieve this requirement using a UDF by following these steps:

1. Using the **Design** section of the SAP Cloud Platform Integration Web UI, navigate to the concerned package, and open the integration flow.
2. Click the **Edit** button.
3. Select the **Message Mapping** step that should be enhanced with the UDF, as shown in [Figure 9.2](#).

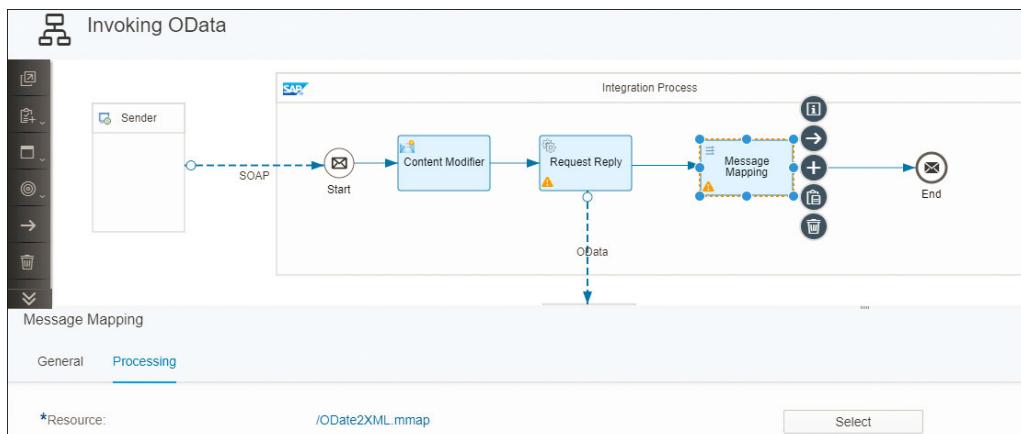


Figure 9.2 Selecting the Message Mapping Step to Be Enhanced with a UDF

4. Select the **Processing** tab, and click on the [/ODate2XML.mmap](#) link (see [Figure 9.2](#)) to open the mapping editor.
5. Select an element on the target message structure (e.g., the `shipStreet` field as shown in [Figure 9.3](#)). Notice that a section called **Functions** appears on the bottom-left corner of the mapping editor (see [Figure 9.3](#)).

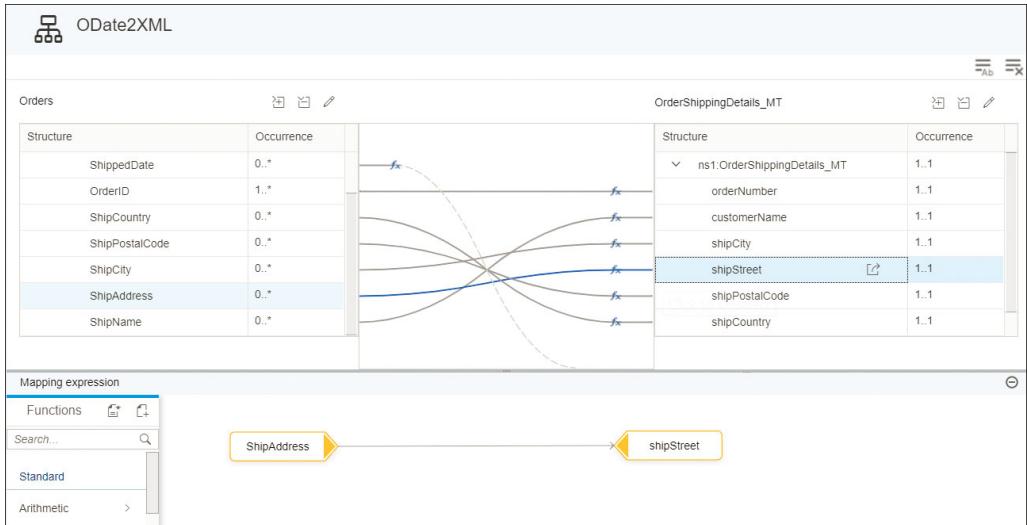


Figure 9.3 Mapping Editor with the Details of the Mapping to Be Enhanced

6. Let's now create a new UDF using the **Create** icon next to the **Functions** box in the mapping editor.
7. Specify a name for the UDF to be created (see [Figure 9.4](#)).

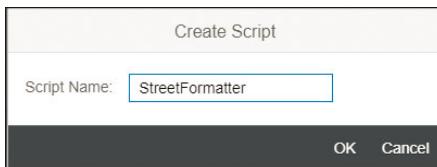


Figure 9.4 Providing a Name for the UDF

You're redirected to a UDF editor with some standard code (see [Figure 9.5](#)).

Note

As you notice on the top part of [Figure 9.5](#), the mapping API has been imported by the statement `import com.sap.it.api.mapping.*;`. This is the same package that was discussed in [Table 9.2](#).

Note that [Figure 9.5](#) also has includes a sample source code to showcase what is possible. The sample code is included between the /* and */ characters.

Cloud Integration Book Package / Invoking OData / ODate2XML / StreetFormatter.groovy /

StreetFormatter.groovy

```
1 import com.sap.it.api.mapping.*;
2
3 /*Add MappingContext parameter to read or set headers and properties
4 def String customFunc1(String P1,String P2,MappingContext context) {
5     String value1 = context.getHeader(P1);
6     String value2 = context.getProperty(P2);
7     return value1+value2;
8 }
9
10 Add Output parameter to assign the output value.
11 def void custFunc2(String[] is,String[] ps, Output output, MappingContext context) {
12     String value1 = context.getHeader(is[0]);
13     String value2 = context.getProperty(ps[0]);
14     output.addValue(value1);
15     output.addValue(value2);
16 }*/
17
18 def String customFunc(String arg1){
19     return arg1
20 }
```

Figure 9.5 Standard Groovy Script Code Provided When Creating a UDF

8. Rename the Groovy method (e.g., “streetFormatterFunc”), and adapt the source code to suit your need (see the example in [Figure 9.6](#)). Line 8 of [Figure 9.6](#) indicates that the code provided is using the Java API to retrieve the OrderNo from the message header.

Design / Cloud Integration Book Package / Invoking OData / ODate2XML / StreetFormatter.groovy /

OK Cancel ?

StreetFormatter.groovy

```
1 import com.sap.it.api.mapping.*;
2
3
4 def String streetFormatterFunc(String arg1,MappingContext context){
5     String newInput = arg1.replace(" ","").toUpperCase(); //remove empty space and make upper case.
6     String order = context.getHeader("OrderNo"); //Retrieve order ID for the message header
7
8     return newInput + "_" + order;
9 }
10
11
12
```

Figure 9.6 Adapted Code to Remove Empty Spaces and Turn the Text to Uppercase

9. Click on the **OK** button in the top-right corner of the screen in [Figure 9.6](#).

10. Return to the mapping editor, and find the newly created **StreetFormatter** UDF under the **Custom** section, as shown in [Figure 9.7](#).

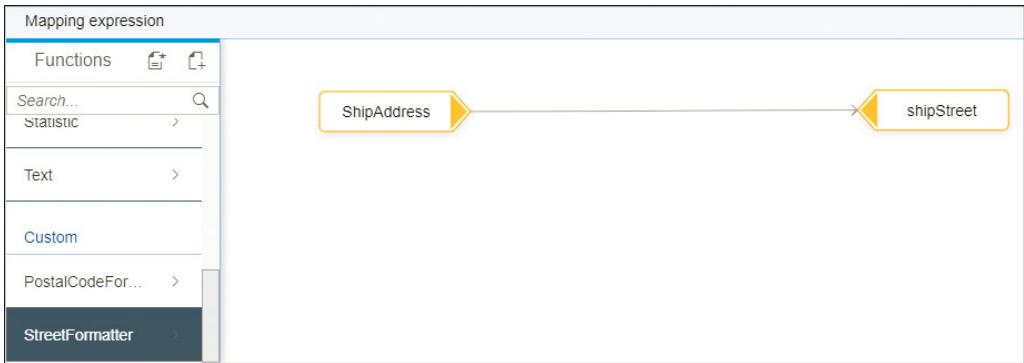


Figure 9.7 The Newly Created UDF Now Available in the Mapping

11. Click on the **StreetFormatter** UDF to see its method name. In our example, it's called **streetFormatterFunc** (see [Figure 9.8](#)). Note that the **streetFormatterFunc** comes from the name that we provided for the Groovy method in [Figure 9.6](#).
12. Let's now use the new function by dragging it between **ShipAddress** and **shipStreet**, as shown in [Figure 9.8](#).

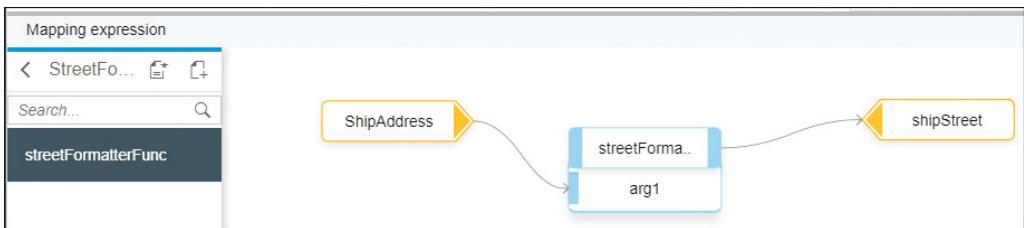
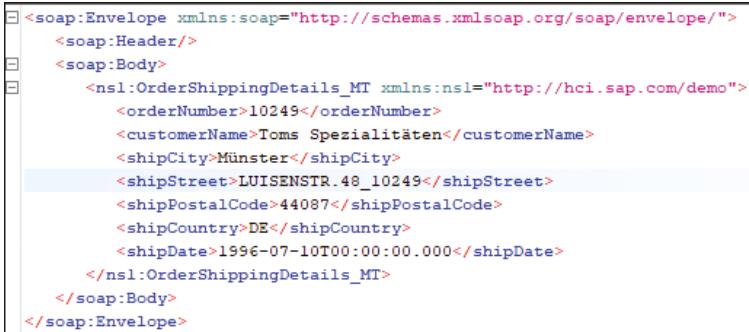


Figure 9.8 Inserting the UDF in Your Mapping Logic

13. Save and deploy the integration flow.
14. Test the service using SoapUI. You get a response with the field **shipStreet** without spaces, in uppercase, and prefixed with the order number, as shown in [Figure 9.9](#).

A screenshot of a code editor showing a SOAP response XML. The XML is well-structured with nested elements. It includes namespaces like 'soap' and 'ns1', and specific fields such as 'orderNumber', 'customerName', 'shipCity', 'shipStreet', 'shipPostalCode', 'shipCountry', and 'shipDate'. The XML is color-coded for readability, with different colors for tags, attributes, and text values.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header/>
  <soap:Body>
    <ns1:OrderShippingDetails_MT xmlns:ns1="http://hci.sap.com/demo">
      <orderNumber>10249</orderNumber>
      <customerName>Toms Spezialitäten</customerName>
      <shipCity>Münster</shipCity>
      <shipStreet>LUISENSTR. 48_10249</shipStreet>
      <shipPostalCode>44087</shipPostalCode>
      <shipCountry>DE</shipCountry>
      <shipDate>1996-07-10T00:00:00.000</shipDate>
    </ns1:OrderShippingDetails_MT>
  </soap:Body>
</soap:Envelope>
```

Figure 9.9 Response Returned by the Integration Flow after Adding the UDF

You now know how to create a UDF that uses SAP Cloud Platform Integration Java APIs to perform transformation logic in a mapping. Let's now move to the next section where we look at using the script step in SAP Cloud Platform Integration.

9.4 Using the Script Step

SAP Cloud Platform Integration provides a script step that enables you to write different custom scripts to perform a wide range of activities and utilize the Java APIs that we explored earlier in [Section 9.2](#). The scripting feature opens the door to the developer's imagination to do pretty much anything. Note, however, that scripting should be used with due diligence and caution to avoid unnecessary overheads and performance issues.

At the time of publishing this book, the script step supports Groovy and JavaScript. Groovy is a Java-syntax-compatible object-oriented programming language for Java platforms. To learn more about Groovy, go to <http://groovy-lang.org>.

JavaScript is a dynamic, weakly typed, prototype-based, and multiparadigm programming language for the Web. A large number of websites use it. To learn more about JavaScript, go to www.w3schools.com/js.

Both languages are relatively easy to learn, and there are plenty of resources on the Internet that you can use as reference.

SAP Cloud Platform Integration provides a Script API in a form of a JAR file. Using this JAR, a Java developer can easily import the library in his development tool of preference and inspect the provided methods. The Script API JAR can be downloaded from <https://tools.hana.ondemand.com/#cloudintegration>. On that web page, search for

the **Using Script API** section, as shown in [Figure 9.10](#). Then click on the **Download** link to save the JAR on your local file system.

Using Script API

The Script API enables Eclipse to display the provided methods in content assist. You need to download the Script API JAR and add it to your integration project's build path to enable usage of methods it provides in groovy scripts.

For detailed information on how to use the Script API, see [Using Script API Methods in Groovy Scripts](#).

Script API JAR : [Download](#)

Figure 9.10 Where to Download the Script API JAR

At the time of publishing this book, the JAR file is called *cloud.integration.script.apis-1.36.1*.

To better illustrate the usage of the Script API, let's use a sample scenario in the next section.

9.4.1 Target Scenario

Let's reuse the example from [Chapter 4, Section 4.3](#), to illustrate the use of the script step. In that example, we invoked an external OData service from our integration flow. [Figure 9.11](#) shows the integration flow that was built for it.

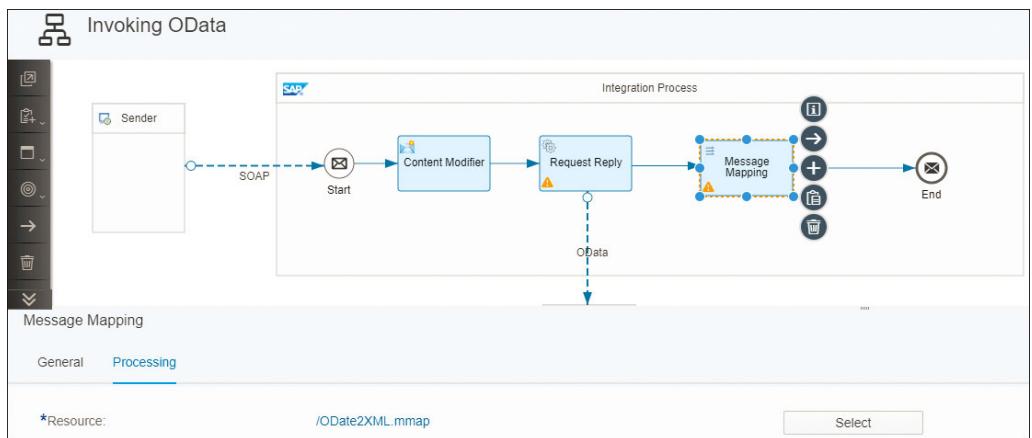


Figure 9.11 Invoking an OData Service

Imagine that the organization providing this integration flow is currently running a lucky draw on every incoming message. As an integration developer, you're

requested to change the integration flow by generating a random number to be associated to each call. This random number will be used by your organization later to pick the lucky winner.

Keeping the solution simple, we can create a script that generates a random number and uses the API to save it as a message header.

9.4.2 Enhancing the Integration Flow

Let's now change our integration flow. Note that we won't detail every single step because you're already familiar with editing integration flows. Follow these steps:

1. From the **Design** section of the SAP Cloud Platform Integration Web UI, navigate to the correct package and integration flow.
2. In the opened integration flow, click on the **Edit** button.
3. Select a **Script** step  from the palette (see [Figure 9.12](#)). You find this shape in the palette under **Message Transformers** .

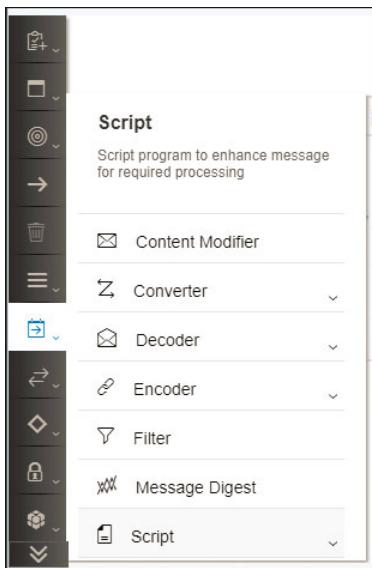


Figure 9.12 Adding the Script Step to the Integration Flow

4. Another menu appears from the palette with a choice of **JavaScript** or **Groovy-Script**. Click on **GroovyScript** (see [Figure 9.13](#)).

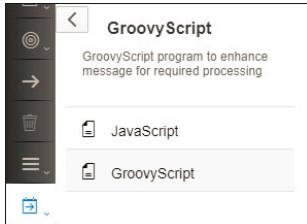


Figure 9.13 Selecting GroovyScript from the Palette

5. Place it in your integration flow, right after the **Start** icon
6. The **Script Editor** automatically opens with some sample script, as shown in [Figure 9.14](#). Note that this is the same editor that we used when creating a UDF in [Section 9.3](#).

The screenshot shows the SAP Integration Designer's Script Editor window. The title bar says 'script1.groovy'. The editor displays the following Groovy script:

```

1 /*
2  * The integration developer needs to create the method processData
3  * This method takes Message object of package com.sap.gateway.ip.core.customdev.util
4  * which includes helper methods useful for the content developer:
5  * The methods available are:
6  * public java.lang.Object getBody()
7  * public void setBody(java.lang.Object exchangeBody)
8  * public java.util.Map<java.lang.String,java.lang.Object> getHeaders()
9  * public void setHeaders(java.util.Map<java.lang.String,java.lang.Object> exchangeHeaders)
10 public void setHeader(java.lang.String name, java.lang.Object value)
11 public java.util.Map<java.lang.String,java.lang.Object> getProperties()
12 public void setProperties(java.util.Map<java.lang.String,java.lang.Object> exchangeProperties)
13     public void setProperty(java.lang.String name, java.lang.Object value)
14 */
15 import com.sap.gateway.ip.core.customdev.util.Message;
16 import java.util.HashMap;
17 def Message processData(Message message) {
18     //Body
19     def body = message.getBody();
20     message.setBody(body + "Body is modified");
21     //Headers
22     def map = message.getHeaders();
23     def value = map.get("oldHeader");
24     message.setHeader("oldHeader", value + "modified");
25     message.setHeader("newHeader", "newHeader");
26     //Properties
27     map = message.getProperties();
28     value = map.get("oldProperty");
29     message.setProperty("oldProperty", value + "modified");
30     message.setProperty("newProperty", "newProperty");
31     return message;
32 }
33

```

Figure 9.14 Sample Code Included in the Groovy Script Editor

7. Change the `processData` method part of the script in [Figure 9.14](#) to adapt the script to your requirements. In our case, we used the code shown in [Figure 9.15](#). Note that the `processData` method takes a `Message` object as input and also returns a `Message`

object as an output. If you’re a programmer, the script presented in [Figure 9.15](#) is self-explanatory. It also includes comments in plain English for those less familiar with Groovy. The code generates a random number between 0 and 1000. The generated random number is then added as a message header named `LuckyNumber`. The Groovy Script in [Figure 9.15](#) can be downloaded from the book homepage.



```

1 * /*
2  * Generate a random numner and add it to the message header
3  */
4 import com.sap.gateway.ip.core.customdev.util.Message;
5 import java.util.HashMap;
6 def Message processData(Message message) {
7
8     //Generate a random Number
9     int max = 1000;
10    int min = 0;
11    Random r = new Random();
12    def value = r.nextInt((max - min) + 1) + min;
13    //Properties
14    map = message.getProperties();
15    message.setHeader("LuckyNumber", value);
16
17    return message;
18 }

```

Figure 9.15 Final Result of the Groovy Script

- Click on the **OK** button in the top-right corner of [Figure 9.15](#) to return to the integration flow. [Figure 9.16](#) depicts the final look of the integration flow extended with the **Script** step.

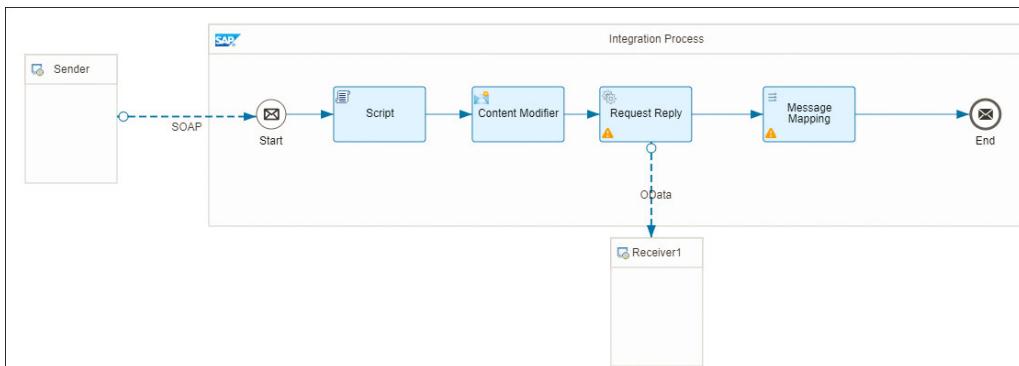


Figure 9.16 Enhanced Overview of the Integration Flow

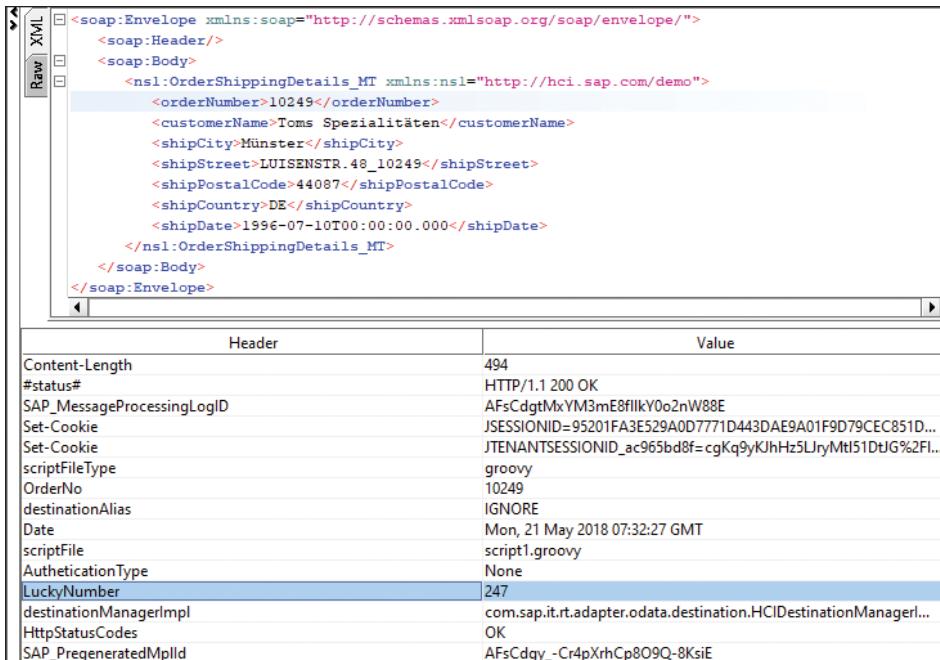
- Save and deploy the integration flow.

You're now ready to send a message via SoapUI. After you've triggered the message, the randomly generated number is returned in the header of the response message. Notice the `LuckyNumber` in the header section of the response in [Figure 9.17](#) and [Figure 9.18](#). This is exactly what we asked the script to do (see code in line 15 of [Figure 9.16](#)).



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope">
  <soapenv:Header/>
  <soapenv:Body>
    <demo:OrderNumber_MT xmlns:demo="http://</>">
      <orderNumber>10249</orderNumber>
    </demo:OrderNumber_MT>
  </soapenv:Body>
</soapenv:Envelope>
```

Figure 9.17 Returned Response Headers A



Header	Value
Content-Length	494
#status#	HTTP/1.1 200 OK
SAP_MessageProcessingLogID	AFsCdgtMxYM3mE8fllkV0o2nW88E
Set-Cookie	JSESSIONID=95201FA3E529A0D7771D443DAE9A01F9D79CEC851D...
Set-Cookie	JTENANTSESSIONID_ac965bd8f=cgKq9yKJhHz5LuryMt51DtJG%2Fl...
scriptFileType	groovy
OrderNo	10249
destinationAlias	IGNORE
Date	Mon, 21 May 2018 07:32:27 GMT
scriptFile	script1.groovy
AuthenticationType	None
LuckyNumber	247
destinationManagerImpl	com.sap.it.rt.adapter.odata.destination.HCIDestinationManager...
HttpStatusCodes	OK
SAP_PregeneratedMplId	AFsCdgy_-Cr4pXrhCp8O9Q-8KsiE

Figure 9.18 Returned Response Headers B

Congratulations, you can now use the **Script** step to perform different tasks using the APIs. Let's now move to discuss the OData API.

9.5 OData API

Besides the Java APIs, SAP Cloud Platform Integration also allows the developer to access various aspects of the platform using an OData API. The APIs are provided in the form of Representational State Transfer (REST) APIs that use the Open Data Protocol (OData) as a technical protocol. As a result, these APIs use the well-known HTTP methods of GET, POST, PUT, PATCH, and DELETE.

Note

Note that at the time of publishing this book, OData specification version 2.0 is supported by the SAP Cloud Platform Integration OData APIs. To read more about the OData V2 specification, go to www.odata.org/documentation/odata-version-2-0.

The OData APIs can be accessed using the HTTP URL of the form:

https://<host>/api/v1/<resource>?\$<property1>=< property1_value>&\$<property2>=< property2_value>

In this URL, note the following:

- <*host*>
Represents the URL address of the tenant management node.
- <*resource*>
Represents the path of the entity types to be called. Some entity resources are presented later in [Table 9.4](#). For example, you can use the resource `MessageProcessingLogs` to address the message processing log (MPL).
- <*property1*>
Represents the name of the property to be queried. For example, you can use the property `count` to return the total number of MPLs. It's always prefixed by the \$ symbol. Note that the property field is optional. Furthermore, you're able to add as many properties as you need by using the character & between them.

The OData APIs are protected by basic authentication (username and password) and require the API client to enable HTTP cookies. Furthermore, to use an API, you need

to have the correct authorization group assigned to your user. [Table 9.3](#) lists and describes the authorization groups needed for different API actions.

API Action	Authorization Group
Ability to display message overview	AuthGroup.Administrator or AuthGroup.IntegrationDeveloper
Undeploy integration content	AuthGroup.IntegrationDeveloper
Download a message	AuthGroup.BusinessExpert

Table 9.3 Required Authorization Groups and Description

The SAP Cloud Platform Integration OData APIs are structured around entity types or resources. Every entity type contains a number of properties. A property can also refer to another entity type, which means it's possible to start with one entity type and navigate to another entity type. Therefore, it's important to properly understand the entity types, their tasks, and their relationships with each other. [Table 9.4](#) lists all available entity types and describes what they are used for.

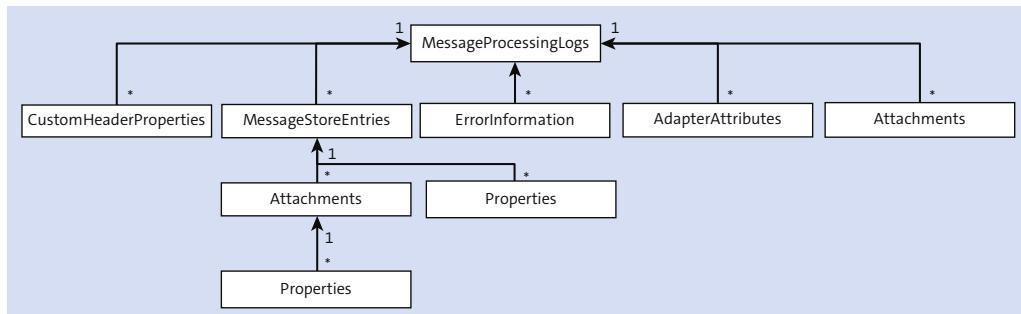
Entity Types	Task Description
MessageProcessingLog	Reads an MPL
MessagePropcressingLogCustomHeaderProperty	Reads custom header properties of the MPL
MessageProcessingLogErrorInformation	Reads error information for a message
MessageProcessingLogAdapterAttribute	Reads adapter-specific attributes
MessageProcessingLogAttachment	Reads an MPL attachment
MessageStoreEntry	Reads a message from the message store
MessageStoreEntryProperty	Reads a header property of a message from the message store
MessageStoreEntryAttachment	Reads an attachment of a message from the message store

Table 9.4 Entity Types and Their Tasks

Entity Types	Task Description
MessageStoreEntryAttachmentProperties	Reads properties of message attachments from the message store
IntegrationRuntimeArtifact	Reads properties of deployed integration content
PartnerDirectory	Accesses the Partner Directory, creates entries, and helps manage them
LogFile	Accesses all current (nonarchived) log files
LogFileArchives	Accesses all archived log files

Table 9.4 Entity Types and Their Tasks (Cont.)

As an illustration of how each entity type relates to the others, [Figure 9.19](#) depicts an entity model around the *MessageProcessingLogs* entity type to help you better grasp how to use the APIs related to the monitoring of message flows.

**Figure 9.19** Entity Model Diagram for MessageProcessingLogs

As specified by the SAP documentation (here depicted in [Figure 9.19](#)), the *MessageProcessingLogs* contain a number of subentities, including *CustomHeaderProperties*, *MessageStoreEntries*, *ErrorInformation*, *AdapterAttributes*, and *Attachments*. Descriptions of what each subentity does are provided in [Table 9.4](#).

Additionally, the OData APIs provided by SAP Cloud Platform Integration make use of common query options. These query options can be used on different entity types to perform specific actions on them. However, not all options are supported by each entity type. [Table 9.5](#) describes some common query options.

Option	Description
\$filter	Retrieves a set of entries based on the resource entity and the filter expression used in the Uniform Resource Identifier (URI)
\$metadata	Retrieves the data model and structure of all resources
\$select	Retrieves a subset of information on the entities identified by the resource path section of the URI
\$top	Returns a subset of n top records from the resource used in the URI
\$count	Returns the number of entries that matches the resource specified in the URI or the filter-specified criteria
\$inlinecount	Indicates that the response contains a count of the number of entries in the collection of records identified by the resource path section of the URI
\$value	Retrieves specific values of an entity resource specified by a Global Unique Identifier (GUID)
\$skip	Skips n records in the collection returned according to the resource path section of the URI
\$expand	Retrieves related and correlated entities for a given navigation property in line with the entities being retrieved
\$orderby	Specifies the sorting of the returned collection by one or more values

Table 9.5 Commonly Used Query Options

Note

The OData API provided by SAP Cloud Platform Integration limits the number of entries in the response to a maximum of 1,000 entries for each call. This limitation feature protects the performance of the SAP Cloud Platform Integration runtime environment and avoids the negative consequences of queries returning huge amounts of data.

Queries with more than 1,000 entries are capped, and a **Next** link element is added to the response, which can be used to initiate the return of the additional entries.

Later in the chapter, we'll explore APIs and entities related to the following aspects:

- Monitoring MPL
- Deployed integration content
- Log files
- Message store
- Security material
- Partner Directory

The APIs listed can easily be tested and explored using the SAP API Business Hub. Let's discuss it next.

9.5.1 SAP API Business Hub

From the SAP Cloud Platform 1802 release, the OData APIs are documented in the SAP API Business Hub, which is the central catalog of all SAP and partner APIs for developers to build sample apps, extensions, and open integrations with SAP. The SAP API Business Hub landing page is at <https://api.sap.com>, as shown in [Figure 9.20](#).

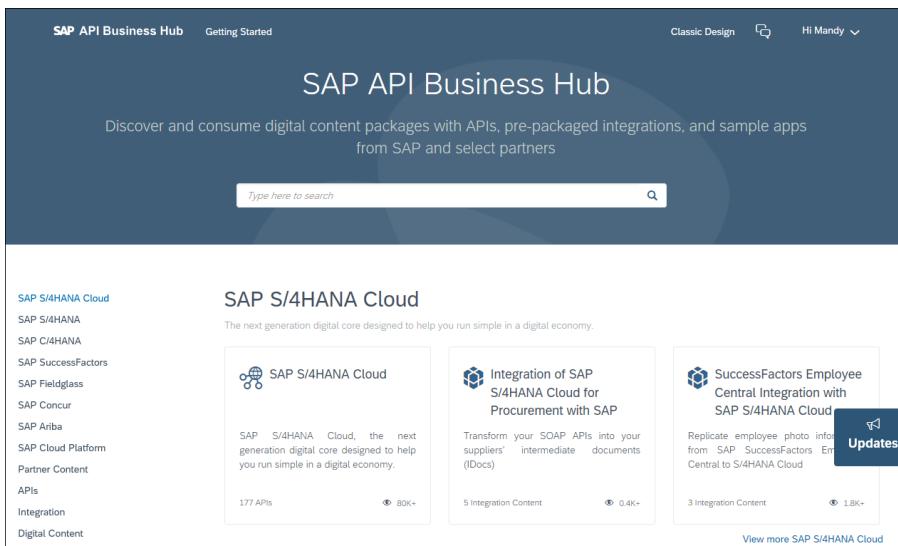


Figure 9.20 Landing Page of SAP API Business Hub

From the page presented in [Figure 9.21](#), you can click on the **View more APIs** link to move to another page enabling you to select different packages (see [Figure 9.22](#)).

The screenshot shows the SAP API Business Hub APIs section. On the left, there's a sidebar with a tree view of categories: SAP S/4HANA Cloud, SAP S/4HANA, SAP SuccessFactors, SAP Fieldglass, SAP Concur, SAP Ariba, SAP Cloud Platform, Partner Content, APIs, Integration, and Digital Content. The main area is titled "APIs" and has a sub-header "Explore API content from SAP and partners." It features three cards: "SAP Digital Manufacturing Cloud - Manufacturing Network" (2 APIs, 0.1K+), "SAP Translation Hub" (1 API, 3.5K+), and "Connect to SAP Concur" (1 Policy Template, 1K+). A "View more APIs" link is at the bottom right.

Figure 9.21 Landing Page of SAP API Business Hub, APIs Section

The screenshot shows the SAP API Business Hub main page. The title is "SAP API Business Hub" with the subtitle "Discover and consume digital content packages with APIs, pre-packaged integrations, and sample apps from SAP and select partners". Below is a search bar. The main content is titled "APIs" with a "Sort By: Recently Updated" dropdown. It lists six API packages in a grid:

- SAP Hybris Marketing Cloud**: Enables marketers to understand the real-time intent of individual customers. (2 APIs, 0.5K+)
- SAP Cloud Platform**: This API product contains available SAP Cloud Platform APIs, which can be used in the context of an SAP Cloud Platform account. (7 APIs, 6.7K+)
- SAP Cloud Platform Integration**: SAP Cloud Platform Integration supports end-to-end process integration. (5 APIs, 0.8K+)
- SAP Cloud Platform Business Rules**: Externalize business decisions from applications using Business Rules service. (2 APIs, 0.8K+)
- SAP Watch List Screening**: Screen business partners' names and addresses against sanctioned party lists. (2 APIs, 0.1K+)
- SAP Cloud Platform 3D Visualization**: A set of tools and services that provide 3D capabilities for cloud solutions. (4 APIs, 0.2K+)

Figure 9.22 List of Available API Packages

Select the package called **SAP Cloud Platform Integration**. If you can't find it you can filter packages based on different categories to see a list of all possible APIs (see [Figure 9.23](#)).

The screenshot shows the SAP Cloud Platform Integration API list page. At the top, there is a title 'SAP Cloud Platform Integration' with a globe icon. Below it, a sub-header states 'SAP Cloud Platform Integration supports end-to-end process integration.' There are two tabs: 'APIs' (selected) and 'Details'. A search bar with placeholder 'Type here to filter' and a magnifying glass icon is located above the list. Below the search bar, it says 'Showing 5 of 5 APIs'. The page displays five API artifacts:

API Name	Description	Version	ODATA
Partner Directory	Get, write or delete data of the Partner Directory. This content can be used to parameterize integration flows.	Version 1.0.0	ODATA
Message Store	Get data from Message Store (such as message payloads, headers, properties, or attachments) for processed messages.	Version 1.0.0	ODATA
Log Files	Get an overview of HTTP and trace log files.	Version 1.0.0	ODATA
Message Processing Logs	Get an overview of the messages processed on a tenant and get the details for individual messages.	Version 1.0.0	ODATA
Integration Content	Get deployed integration content or deploy/undeploy integration artifacts.	Version 1.0.0	ODATA

Figure 9.23 Artifacts of the OData APIs in the SAP Cloud Platform Integration Package

From the web page shown in Figure 9.23, you can explore the different APIs. Using the SAP API Business Hub, you can test and try out the different APIs directly without having to implement any code or use a third-party REST client such as Postman (www.getpostman.com). For APIs implementing the GET HTTP method, you can also use a simple browser.

The SAP API Business Hub has two different approaches to performing tests:

- API sandbox
- Your SAP Cloud Platform Integration tenant

Each of these two testing approaches are explored next.

API Sandbox

If you don't have access to a SAP Cloud Platform Integration tenant to test with, you can always use an API sandbox provided by SAP. The API sandbox is filled with test data and presents a quick way to get a feel for the way the APIs operate. Note that only operations using the `GET` method are supported in the API sandbox. Operations needing write access are forbidden because you need to log in before you can call operations requiring write access in the API sandbox.

For instance, let's use the Log Files API to illustrate testing using the API sandbox approach. For that, follow these steps:

1. Click on **Log Files** from the page presented earlier in [Figure 9.23](#).
2. From the resulting page, select the **RESOURCE** tab. The page lists all operations of the API. Note that the **API Endpoint** field is automatically assigned with a sandbox-related URL.
3. Assuming we want to test the **LogFileArchives** API, we'll need to click on the **Show/Hide** link to the right of **LogFileArchives** ([Figure 9.24](#)).

The screenshot shows the SAP API Management interface for the Log Files API. At the top, there is a share icon and the title "Log Files". Below the title, a subtitle reads "Get an overview of HTTP and trace log files." There are two buttons: "Show API Key" and "Download SDK".

Below these buttons, there are two tabs: "API References" (which is selected) and "Details".

In the "API Environment" section, the dropdown menu is set to "Sandbox".

The main content area is titled "LogFile Archives". It contains two "GET" operations:

- The first operation is "/LogFileArchives". It includes "Try out" and "Code Snippet" buttons.
- The second operation is "/LogFileArchives(Scope='Scope',LogFileType='LogFileType',NodeScope='worker')". It also includes "Try out" and "Code Snippet" buttons.

At the bottom of the page, there are links for "Show/Hide", "List Operations", and "Expand Operations".

Figure 9.24 List of Entity Types Available for the Log Files API

4. From the resulting page, click on the GET button to the left of `/LogFileArchives`.
5. You get a page similar to the one presented in [Figure 9.25](#). Click on the Try it out! button. Note that if your API requires input parameters, they will be listed in this page. You'll then need to fill in the required parameters as a minimum.

The screenshot shows the SAP API Business Hub interface for the Log File Archives API. At the top, there are buttons for 'Show/Hide', 'List Operations', and 'Expand Operations'. Below that, a navigation bar shows a 'GET' method and the endpoint '/LogFileArchives'. To the right are 'Try out' and 'Code Snippet' buttons.

Description: Get all log file collections.

Response Class (Status 200):

Model	Model Schema
<pre>{ "d": { "results": [{ "Scope": "all", "LogFileType": "http", "NodeScope": "worker", "ContentType": "application/zip" }] } }</pre>	

Response Content Type: application/json

Parameters:

Parameter	Value	Description	Parameter Type	Data Type	Actions
-----------	-------	-------------	----------------	-----------	---------

Response Messages:

HTTP Status Code	Reason	Response Model	Headers
default	Error	Model Model Schema <pre>{ "error": { "code": "string", "message": { "lang": "string", "value": "string" }, "details": [{ "code": "string", "value": "string" }] } }</pre>	

Try out button

Figure 9.25 The SAP API Business Hub Try it Out! Page

From the page presented in [Figure 9.25](#), a number of attributes and functionalities are available, as described in [Table 9.6](#).

Properties	Description
Description	Describes the operation.
Parameters	Used to add the header or query parameters for the request message of the API.
Response Messages	Documents the possible response messages, including the different HTTP status codes and example response messages. After calling the API, both response body and response header are also returned.

Table 9.6 Properties in the SAP API Business Hub's Try It Out! Page

- After clicking on the **Try it out!** button, a response header and body are returned on the page. [Table 9.7](#) lists the functionalities and features available in the page shown in [Figure 9.25](#).

Functionality	Description
Code Snippet	Retrieve the documentation and code snippet describing how to invoke this API. Currently a snippet in the following languages and technologies is supported: JavaScript, Java, Swift, Curl, ABAP, and SAPUI5 (see Figure 9.26). This snippet gives you a head start with your implementation without the need to start from scratch.
Download API	Download the API in one of the following formats: JavaScript Object Notation (JSON), YAML Ain't Markup Language (YAML), or Entity Data Model Designer (EDMX).
Show API Key	Use to retrieve the API key in the sandbox host URL to try out APIs.
Download SDK	Download a prepopulated Java Software Development Kit (SDK) for the API. Note that this is a full Java project that you can use.

Table 9.7 Functionalities in the SAP API Business Hub's Try It Out! Page

The screenshot shows a 'Code Snippet' dialog box. On the left, there's a sidebar with language options: Javascript, Java, Swift, Curl, ABAP, and SAPUI5. The Javascript option is selected. The main area contains sample code for making an API request using XMLHttpRequest. Below the code are two buttons: 'Copy and Close' and 'Cancel'.

```

Javascript
var data = null;
var xhr = new XMLHttpRequest();
xhr.withCredentials = false;

xhr.addEventListener("readystatechange", function () {
  if (this.readyState === this.DONE) {
    console.log(this.responseText);
  }
});

//setting request method
//API endpoint for API sandbox
xhr.open("GET", "https://sandbox.api.sap.com:443/cpi/api/v1/LogFileArchives");

```

Figure 9.26 Code Snippet to Consume the API in Different Languages

Let's next look at how to configure the SAP API Business Hub to test against your own SAP Cloud Platform Integration tenant.

Your SAP Cloud Platform Integration Tenant

As previously discussed, SAP API Business Hub also makes it possible to test against your own tenant. For that, you'll need to configure SAP API Business Hub to point to your tenant by changing the API endpoint. Follow these steps:

1. Click on the **Configure Environments** link on the top-right top side of the page shown earlier in [Figure 9.25](#).
2. If you're not already logged in, you're presented with a **Login Required** popup ([Figure 9.27](#)). Click on the **Login** button, and provide your credentials. If you're already logged in, proceed to step 4.

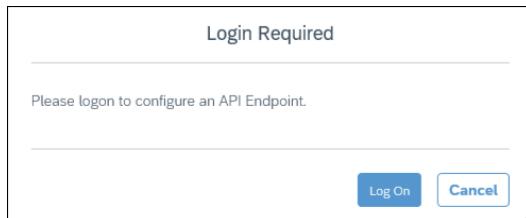


Figure 9.27 Login Page When Configuring the API Endpoint

3. You're redirected back to the main page again (refer to [Figure 9.25](#)). Click on the **Configure Environments** link one more time.
4. Select the region-specific host of your SAP Cloud Platform Integration tenant from the list presented in [Figure 9.28](#) for the starting URL. You just need to compare your tenant URL with one of the entries presented in [Figure 9.28](#) to find the right match.

Enter a **Display Name for Environment**, **Account Short Name**, **SSL Host**, and your user credentials. Refer to [Table 9.8](#) for more details about these attributes.

Configure Environments

Create New Environment

Sandbox	Starting Url*
	<input type="text" value="https://(Account Short Name)-tmn.(SSL Host).ap1.hana.ondemand.com/api/v1"/>
	Display Name for Environment*
	<input type="text" value="Development"/>
	Account Short Name*
	<input type="text" value="p001"/>
	SSL Host*
	<input type="text" value="hci"/>
	Resulting URL
	<input type="text" value="https://p001-tmnhci.ap1.hana.ondemand.com/api/v1"/>
	Authentication Type*
	Basic Authentication
	Username*
	<input type="text" value="myuser"/>
	Password*
	<input type="password" value="***"/>
<input type="checkbox"/> Apply this environment to all APIs in this package that are not yet configured	

Figure 9.28 Selecting the Region of Your Tenant

Column	Description
Display Name for Environment	Enter a human-readable alias name. Example: Development, Test, or Acceptance.
Account Short Name	This is your tenant ID, which can be found between the // and -tmn in your tenant URL. The tenant URL is always of the format: <code>https://{Account Short Name}-tmn.{SSLHost}.{Region}.hana.ondemand.com</code> .
SSL Host	This can be retrieved from your tenant URL. The tenant URL is always of the format: <code>https://{Account Short Name}-tmn.{SSL-Host}.{Region}.hana.ondemand.com</code> . Example: For <code>{SSL Host}</code> , use hci.
Username	The username of your SAP Cloud Platform Integration tenant account.
Password	The password of your SAP Cloud Platform Integration tenant account.

Table 9.8 Attributes of the Configure API Endpoint Page

And voila! From this point, you can perform the API calls against your own tenant. Let's now discuss the CSRF token handling in the next section.

9.5.2 Cross-Site Request Forgery Token Handling

Cross-site request forgery (CSRF or XSRF) is a type of security attack or malicious exploit that occurs when unauthorized commands are transmitted from a user that the web application trusts. CSRF exploits the trust that a site has in a user's browser, such as cookies associated with your bank.

Within the context of APIs, this means that a CSRF attacker can execute an action on the target application via an API without the knowledge and permissions of the consumer application. CSRF attacks have the following characteristics:

- They generally concern sites that rely on a user's identity.
- They exploit the site's trust in that identity.
- They trick the consumer application into sending HTTP requests to a target site.
- They involve HTTP requests that change application data.

To prevent CSRF attacks, some OData APIs provided by SAP Cloud Platform Integration require CSRF token validation. The CSRF token is mostly required for the APIs that need permission to write and change objects via the POST, PUT, and DELETE HTTP operations. There aren't enough pages in this book to go into the details of how CSRF tokens work, so for further reference, consult the many resources online, including <https://docs.spring.io/spring-security/site/docs/current/reference/html/csrf.html>.

When an API uses a CSRF token, calls made to the API without a CSRF are rejected. As a result, you need to retrieve a CSRF token first before invoking such an API. Let's now explore how to fetch a CSRF token.

You can use any API client of your choice, but for illustration purposes, we'll use Postman (www.getpostman.com/). You've already installed it from the scenario in [Chapter 7](#). To retrieve the CSRF token, you'll need to use the following endpoint: `https://<TMN-host>/api/v1`.

Here, `TMN-host` represents the tenant host of SAP Cloud Platform Integration. As shown in [Figure 9.29](#), select **GET** as the HTTP method, and specify the OData API endpoint. You also need to select **Basic Auth** and enter your credentials.

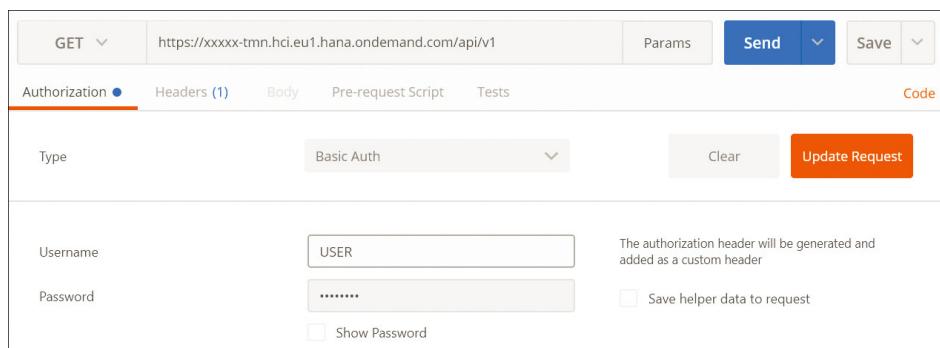


Figure 9.29 Configuring Endpoint and Authorization in Postman

In the **Headers** tab, add a new key named **X-CSRF-Token** with the value `Fetch` to request an X-CSRF token ([Figure 9.30](#)).

Authorization		Headers (1)	Body	Pre-request Script	Tests
Key	Value				
<input checked="" type="checkbox"/> X-CSRF-Token	Fetch				
New key	Value				

Figure 9.30 Adding the X-CSRF-Token Header

Click on the **Send** button (in top-right corner of [Figure 9.29](#)) to trigger the request. The response message includes a number of headers, including the CSRF token, which can be identified by **X-CSRF-Token** in the **Headers** tab ([Figure 9.31](#)).

Body	Cookies	Headers (10)	Test Results
<pre>cache-control → no-cache,no-store,must-revalidate content-length → 5325 content-type → application/atomsvc+xml;charset=utf-8 dataserviceversion → 1.0 date → Sat, 19 May 2018 12:56:55 GMT expires → Thu, 01 Jan 1970 00:00:00 UTC pragma → no-cache server → SAP strict-transport-security → max-age=31536000; includeSubDomains; preload x-csrf-token → 0016004867FBFF66EDF0312A2E8EDA68</pre>			

Figure 9.31 X-CSRF-Token in the Headers Tab

The value of the CSRF token can then be used in the header of your next OData API request. Furthermore, the body of the response message is filled with the list of entity types that the CSRF token can be used with. In other words, the scope of the CSRF token is limited to the entity types listed in the response (see [Figure 9.32](#)).

In the next sections, we explore the following different API categories:

- Monitoring message flows using the API
- Managing deployed integration content using the API
- Managing log files using the API
- Managing the message store using the API
- Managing security material using the API
- Managing the Partner Directory using the API.

Let's start with the message flow APIs.

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <service xml:base="https://p0262-tmn.hci.eu1.hana.ondemand.com:443/api/v1/" xmlns="http://www.w3.org/2005/Atom"
3    :atom="http://www.w3.org/2005/Atom">
4    <workspace>
5      <atom:title>Default</atom:title>
6      <collection href="Configurations">
7        <atom:title>Configurations</atom:title>
8      </collection>
9      <collection href="LogFileArchives">
10        <atom:title>LogFileArchives</atom:title>
11      </collection>
12      <collection href="IntegrationRuntimeArtifacts">
13        <atom:title>IntegrationRuntimeArtifacts</atom:title>
14      </collection>
15      <collection href="MessageProcessingLogCustomHeaderProperties">
16        <atom:title>MessageProcessingLogCustomHeaderProperties</atom:title>
17      </collection>
18      <collection href="CertificateResources">
19        <atom:title>CertificateResources</atom:title>
20      </collection>
21      <collection href="KeystoreEntries">
22        <atom:title>KeystoreEntries</atom:title>
23      </collection>
24    </workspace>
25  </service>

```

Figure 9.32 Response Body Associated with the X-CSRF Token

9.5.3 Monitoring Message Flows Using the API

When it comes to APIs related to monitoring message flows, the following entity types play a key role:

- **MessageProcessingLogs**

Entity responsible for MPLs. This is the main and parent entity. From this entity, you can navigate to all other ones.

- **MessageProcessingLogAdapterAttributes**

Encapsulates the adapter attributes of the MPL of a specified message entry. It includes details such as the type of adapter used in the related integration flow.

- **MessageProcessingLogAttachments**

Contains attachments of the MPL related to a specified message entry.

- **MessageProcessingLogCustomHeaderProperties**

Contains custom header properties of MPLs related to a specified message entry.

- **MessageProcessingLogErrorInformation**

Contains error information for the message related to a specified message entry.

Many APIs are included in these entities. Describing all of them in detail would go beyond the scope of this chapter. However, you can find extensive details and

examples in the SAP documentation at https://help.sap.com/viewer/product/CLOUD_INTEGRATION/Cloud. You can also explore these APIs via the SAP API Business Hub at <https://api.sap.com/shell/discover/contentpackage/CloudIntegrationAPI/api/MessageProcessingLogs>.

Figure 9.33 shows the entities and APIs available for MPLs. You can test and explore each one of them using the testing approach that was explored in [Section 9.5.1](#).

The screenshot shows the SAP API Business Hub interface for the 'Message Processing Logs' entity. At the top, there are tabs for 'OVERVIEW' and 'RESOURCE'. Below the tabs, a note states: 'SAP API Business Hub allows you to select and configure an API Endpoint, so you can test APIs against the data available in your tenancy. By default, API sandbox is selected to enable the experience.' An 'API Endpoint' dropdown is set to 'Sandbox (https://sandbox.api.sap.com:443/cpi/api/v1)' with a 'Configure' button next to it. The main area lists several API operations:

- MessageProcessingLogs**:
 - GET /MessageProcessingLogs**: Get all message processing logs. [Code Snippet]
 - GET /MessageProcessingLogs('{MessageGuid}')**: Get message processing log by message Guid. [Code Snippet]
 - GET /MessageProcessingLogs/\$count**: Get number of all message processing logs. [Code Snippet]
- MessageProcessingLogAdapterAttributes**:
 - Show/Hide | List Operations | Expand Operations
- MessageProcessingLogAttachments**:
 - Show/Hide | List Operations | Expand Operations
- MessageProcessingLogCustomHeaderProperties**:
 - Show/Hide | List Operations | Expand Operations
- MessageProcessingLogErrorInformation**:
 - Show/Hide | List Operations | Expand Operations

Figure 9.33 Entity Types and APIs in the MPLs

Let's use an example scenario to illustrate the usage and some of the functionalities of APIs related to the monitoring of message flows.

Assume that your organization uses many integration platforms, including SAP Process Orchestration, SAP Cloud Platform Integration, and other third-party platforms. You've been asked to build a custom dashboard monitoring solution to see, at one glance, statistics and lists of failing messages in the various integration platforms. The advantage of such a dashboard is that you don't need to log in to each of these platforms individually. You just need to log in to the custom dashboard, and you can see errors occurring in all integration platforms.

The question that comes to mind is how you can programmatically retrieve the information from the MPLs of SAP Cloud Platform Integration and find the entries with errors. This is where the Monitoring Message Flows APIs come to the rescue.

To solve this challenge, you need an API that retrieves all MPLs in error for the last hour. That requires us to use the `MessageProcessingLogs` entity. One potential solution is to use the OData endpoint:

```
https://<tenant>/api/v1/MessageProcessingLogs?$inlinecount=allpages&$filter=Status eq 'FAILED' and LogStart gt datetime'2018-04-29T12:00:00' and LogEnd lt datetime'2018-04-29T13:00:00'&$expand=AdapterAttributes
```

Let's examine this OData endpoint with the help of [Table 9.9](#) to understand what's happening. (Note that some attributes used in [Table 9.9](#) were also previously explained in [Table 9.5](#).)

API Endpoint Element	Description	Example
MessageProcessing-Logs	Retrieves MPL entries.	MessageProcessingLogs
inlinecount	Indicates that the response should contain a count of the number of entries in the returned collection.	allpages
filter	Filters the result based on various criteria. As mentioned in the example columns, we're filtering all messages that have the status Failed . Additionally, we filter all message logs that have been created between 29/04/2018 at 12:00:00 and 29/04/2018 at 13:00:00.	Status eq 'FAILED' and LogStart gt datetime'2018-04-29T12:00:00' and LogEnd lt datetime'2018-04-29T13:00:00'
expand	Retrieves correlated entities for a given navigation. In our case, we also want to retrieve adapter-specific attributes.	AdapterAttributes

Table 9.9 Attributes Included in OData Endpoint to Retrieve Entries with Errors

After calling the OData endpoint that solves our challenge, you get the response message presented in [Listing 9.1](#).

```
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata" xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices" xml:base="https://p0262-tmn.hci.eu1.hana.ondemand.com:443/api/v1/">
  <id>
    https://p0262-tmn.hci.eu1.hana.ondemand.com:443/api/v1/MessageProcessingLogs
  </id>
  <title type="text">MessageProcessingLogs</title>
  <updated>2018-04-29T12:58:12.413Z</updated>
  <author>
    <name/>
  </author>
  <link href="MessageProcessingLogs" rel="self" title="MessageProcessingLogs"/>
  <m:count>1</m:count>
  <entry>
    <id>
      https://p0262-tmn.hci.eu1.hana.ondemand.com:443/api/v1/
      MessageProcessingLogs('AFrlv7POJUYGSI2bXJAGRQ74HMyP')
    </id>
    <title type="text">MessageProcessingLogs</title>
    <updated>2018-04-29T12:58:12.413Z</updated>
    <category term="com.sap.hci.api.MessageProcessingLog" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/>
    <link href="MessageProcessingLogs('AFrlv7POJUYGSI2bXJAGRQ74HMyP')/edit" rel="edit" title="MessageProcessingLog"/>
    <link href="MessageProcessingLogs('AFrlv7POJUYGSI2bXJAGRQ74tiHyP')/CustomHeaderProperties" rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/CustomHeaderProperties" title="CustomHeaderProperties" type="application/atom+xml;type=feed"/>
    <link href="MessageProcessingLogs('AFrlv7POJUYGSI2bXJAGRQ74HMyP')/MessageStoreEntries" rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/MessageStoreEntries" title="MessageStoreEntries" type="application/atom+xml;type=feed"/>
    <link href="MessageProcessingLogs('AFrlv7POJUYGSI2bXJAGRQ74tiHyP')/ErrorInformation" rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/ErrorInformation" title="Errorinformation" type="application/
```

```

atom+xml;type=entry"/>
<link href="MessageProcessingLogs('AFrlv7POJUYGSI2bXJAGRQ74HMyP')/
AdapterAttributes" rel="http://schemas.microsoft.com/ado/2007/08/dataservices/
related/AdapterAttributes" title="AdapterAttributes" type="application/
atom+xml;type=feed"/>
<m:inline>...</m:inline>
</link>
<link href="MessageProcessingLogs('AFrlv7POJUYGSI2bXJAGRQ74tiHyP')/
Attachments" rel="http://schemas.microsoft.com/ado/2007/08/dataservices/
related/Attachments" title="Attachments" type="application/atom+xml;type=
feed"/>
<link href="MessageProcessingLogs('AFrlv7POJUYGSI2bXJAGRQ74HMyP')/Runs" rel=
"http://schemas.microsoft.com/ado/2007/08/dataservices/related/Runs" title=
"Runs" type="application/atom+xml;type=feed"/>
<content type="application/xml">
  <m:properties>
    <d:MessageGuid>AFrlv7POJUYGSI2bXJAGRQ74HMyP</d:MessageGuid>
    <d:CorrelationId>AFrlv7MGmzLrpFr0xq0olvAcig5Y</d:CorrelationId>
    <d:ApplicationMessageId m:null="true"/>
    <d:ApplicationMessageType m:null="true"/>
    <d:LogStart>2018-04-29T12:50:59.723</d:LogStart>
    <d:LogEnd>2018-04-29T12:51:00.12</d:LogEnd>
    <d:Sender>Sender_SOAP</d:Sender>
    <d:Receiver m:null="true"/>
    <d:IntegrationFlowName>Invoking_OData</d:IntegrationFlowName>
    <d>Status>FAILED</d>Status>
    <d:AlternateWeblink>...</d:AlternateWeblink>
    <d:IntegrationArtifact m:type="com.sap.hci.api.IntegrationArtifact">
      <d:Id>Invoking_OData</d:Id>
      <d:Name>Invoking OData</d:Name>
      <d>Type>INTEGRATION_FLOW</d>Type>
    </d:IntegrationArtifact>
    <d:LogLevel>INFO</d:LogLevel>
    <d:CustomStatus>FAILED</d:CustomStatus>
  </m:properties>
</content>
</entry>
</feed>
```

Listing 9.1 Response of the OData Endpoint Call

When examining the response message presented in [Listing 9.1](#), you'll notice an element named entry, which represents a log entry in the message monitor. Note that if multiple entries are returned, they are sorted in descending order (with the oldest entry on the top) by default. The entries returned here can also be found in SAP Cloud Platform Integration' monitoring.

In addition, observe that the entry shown in [Listing 9.1](#) has the MessageGuid with a value AFrlv7POJUYGSI2bXJAGRQ74HMyP. The same entry can also be found from the **Monitor Message Processing** section of SAP Cloud Platform Integration in [Figure 9.34](#) next to the **Message ID** field. Also note that the response message has a field count, which specifies the number of returned entries. Furthermore, remember the field IntegrationFlowName, which has the value Invoking_OData and will be of use later in the chapter.

The screenshot shows the SAP Cloud Platform Integration interface for monitoring message processing. On the left, a table lists several messages with their artifact names, timestamps, and statuses. One message, 'Invoking OData', is highlighted and shown in more detail on the right. The detailed view includes the artifact name, last update time, and a properties tab. The properties tab displays the following information:

Message ID:	AFrlv7POJUYGSI2bXJAGRQ74HMyP
Sender:	Sender_SOAP
Artifact Name:	Invoking_OData
Artifact ID:	Invoking_OData
Artifact Type:	Integration Flow

Figure 9.34 MPLs in SAP Cloud Platform Integration

The entry returned in the OData API call contains a number of properties as shown earlier in [Listing 9.1](#). [Table 9.10](#) lists the `MessageProcessingLogs` properties and their descriptions.

Property	Description
MessageGuid	GUID of the message that the processing log concerns.
CorrelationId	GUID of the correlated messages.
ApplicationMessageId	GUID specific to a particular application. Think about it as an identifier set for the sake of identification by an external application. This value can be set using a Content Modifier step and assigning a value to the <code>SAP_ApplicationID</code> header element.

Table 9.10 Properties of the `MessageProcessingLogs`

Property	Description
ApplicationMessageType	Property to represent a type of message as known by a business application. Use a Script step in the integration flow to set this property.
LogStart	Date and time that the writing of the log started.
LogEnd	Date and time that the writing of the log ended.
Sender	Identifier of the sender system.
Receiver	Identifier of the receiver system.
IntegrationFlowName	Name of the integration flow.
Status	Status of the message processing. Currently the following statuses are possible: COMPLETED , PROCESSING , RETRY , ERROR , ESCALATED , and FAILED .
AlternateWebLink	Link used to directly open the MPL on this monitoring entry.
IntegrationArtifact/Id	Technical name or ID of the integration flow.
IntegrationArtifact/Name	Name of the integration flow. This is identical to the IntegrationFlowName property.
IntegrationArtifact/Type	Type of artifact that this message processing concerns. Example: INTEGRATION_FLOW

Table 9.10 Properties of the MessageProcessingLogs (Cont.)

In the next sections, we carry on with our OData API journey by exploring APIs that relate to deployed integration content.

9.5.4 Managing Deployed Integration Content Using the API

Using the OData APIs provided by SAP Cloud Platform Integration, you can query the content of integration artifacts deployed on a tenant. The APIs that access the deployed integration content revolve around the following entity types:

- **IntegrationRuntimeArtifact**

Manages all deployed integration artifacts in the tenant. It's also possible to use the **POST** method to deploy an artifact from the file system. Additionally, an already deployed artifact can be undeployed using the **DELETE** method.

- `IntegrationRuntimeArtifactsErrorInformation`
Holds error information of a specific deployed integration artifact.
- `CSRF Token Handling`
Holds the CSRF token for this session. The CSRF token is only required for write access (as discussed in [Section 9.5.2](#)).

There are too many APIs included in the preceding entities to discuss them all in this chapter. However, you can find extensive details and examples in SAP documentation at https://help.sap.com/viewer/product/CLOUD_INTEGRATION/Cloud. You can also explore these APIs via the SAP API Business Hub at <https://api.sap.com/shell/discover/contentpackage/CloudIntegrationAPI/api/IntegrationContent>.

[Figure 9.35](#) shows the entities and APIs available. You can test and explore each one of them using the testing approaches that were explored in [Section 9.5.1](#).

The screenshot displays the SAP API Business Hub interface for the 'Integration Content' entity. At the top, there are tabs for 'OVERVIEW' and 'RESOURCE'. Below the tabs, a message states: 'SAP API Business Hub allows you to select and configure an API Endpoint, so you can test APIs against the data available in your tenancy. By default, API sandbox is selected to enable the experience.' A dropdown menu for 'API Endpoint' is set to 'TEST (https://p0262-trn.hci.eu1.hana.ondemand.com/api)'. The main content area is divided into sections: 'IntegrationRuntimeArtifacts', 'IntegrationRuntimeArtifactsErrorInformation', and 'CSRF Token Handling'. The 'IntegrationRuntimeArtifacts' section contains three API entries: a blue 'GET' button for '/v1/IntegrationRuntimeArtifacts' (labeled 'Get all deployed integration artifacts'), a green 'POST' button for '/v1/IntegrationRuntimeArtifacts' (labeled 'Deploy an integration artifact'), and a red 'DELETE' button for '/v1/IntegrationRuntimeArtifacts('{id}')' (labeled 'Undeploy an integration artifact'). Below these is another blue 'GET' button for '/v1/IntegrationRuntimeArtifacts('{id}')' (labeled 'Get a deployed integration artifact by id'). The other two sections ('IntegrationRuntimeArtifactsErrorInformation' and 'CSRF Token Handling') have their own 'Show/Hide', 'List Operations', and 'Expand Operations' buttons.

Figure 9.35 Entity Types and APIs Related to Deployed Integration Content

Let's further enhance the example scenario of [Section 9.5.3](#) to illustrate the usage of functionalities related to deployed integration content. In the previous section, we retrieved an entry with error using the APIs of MPLs. Imagine that after retrieving and displaying the entry with error in your custom dashboard, you also want to see details of the related deployed integration content. You're interested to know the name, status, and version on the deployed content, as well as the user who deployed it and when.

To solve this challenge, you need to use the `IntegrationRuntimeArtifact` entity. One solution is to use the following OData endpoint: `https://<tenant>/api/v1/IntegrationRuntimeArtifacts('Invoking_OData')`

Note the value `Invoking_OData` was retrieved from the field `IntegrationFlowName` within the content node depicted in [Listing 9.1](#).

The resulting response of the preceding OData endpoint is shown in [Figure 9.36](#).

```
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
      xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices" xml:base="https://p0262-tmn.hci.eu1.hana.ondemand.com:443/api/v1/">
  <id>
    https://p0262-tmn.hci.eu1.hana.ondemand.com:443/api/v1/IntegrationRuntimeArtifacts('Invoking_OData')
  </id>
  <title type="text">IntegrationRuntimeArtifacts</title>
  <updated>2018-04-29T15:49:08.406Z</updated>
  <category term="com.sap.hci.api.IntegrationRuntimeArtifact" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/>
  <link href="IntegrationRuntimeArtifacts('Invoking_OData')/$value" rel="edit" title="IntegrationRuntimeArtifact"/>
  <link href="IntegrationRuntimeArtifacts('Invoking_OData')/ErrorInformation" rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/ErrorInformation" type="application/octet-stream" title="ErrorInformation" type="application/atom+xml;type=entry"/>
  <content type="application/octet-stream" src="IntegrationRuntimeArtifacts('Invoking_OData')/$value"/>
  <m:properties>
    <d:Id>Invoking_OData</d:Id>
    <d:Version>1.0.0</d:Version>
    <d:Name>Invoking_OData</d:Name>
    <d>Type>INTEGRATION_FLOW</d>Type>
    <d:DeployedBy>S0011540061</d:DeployedBy>
    <d:DeployedOn>2018-04-28T21:11:25.951</d:DeployedOn>
    <d>Status>STARTED</d>Status>
  </m:properties>
</entry>
```

Figure 9.36 Response of the OData Endpoint Call

Note that every `<entry>` element returned in the response of [Figure 9.36](#), represents an artifact in SAP Cloud Platform Integration. In our case, we only have one entry returned. The `properties` element of the response message of [Figure 9.36](#) includes a number of attributes to describe the deployed integration content. These properties are listed and described in [Table 9.11](#).

Attributes	Description
<code>Id</code>	Technical identification of the integration content.
<code>Version</code>	Latest version of the integration content when deployed.
<code>Name</code>	Name of the integration
<code>Type</code>	Type of artifact that this message processing concerns. Possible values include <code>INTEGRATION_FLOW</code> , <code>VALUE_MAPPING</code> , <code>DATA_INTEGRATION</code> , and <code>ODATA_SERVICE</code> .
<code>DeployedBy</code>	Name of the user who deployed the content.

Table 9.11 Properties Available for the Deployed Integration Content OData API

Attributes	Description
DeployedOn	Date and time that the integration content was last deployed.
Status	Current status of deployed integration content. Possible values include STARTED, STARTING, and ERROR.

Table 9.11 Properties Available for the Deployed Integration Content OData API (Cont.)

In the next section, we explore APIs that relate to log files.

9.5.5 Managing Log Files Using the APIs

Using the OData APIs provided by SAP Cloud Platform Integration, you can query log files on a tenant. Note that there are two types of log files:

- **Default trace**

These log files include processing information of a technical nature.

- **HTTP access logs**

These log files include information about all inbound HTTP requests arriving in SAP Cloud Platform Integration.

The APIs that facilitate the access to log files revolve around two main entity types:

- **LogFileArchives**

Used for all archived log files.

- **LogFiles**

Used for all current (nonarchived) log files.

These entities include a number of APIs. We won't explore them all in this section, but we'll look at a scenario to showcase what's possible. To get a full description of the different APIs, refer to the SAP documentation at https://help.sap.com/viewer/product/CLOUD_INTEGRATION/Cloud. You can also explore these APIs via the SAP API Business Hub at <https://api.sap.com/shell/discover/contentpackage/CloudIntegrationAPI/api/LogFiles>.

Figure 9.37 shows the entities and APIs available. You can test and explore each one of them using the testing approaches we explored in Section 9.5.1.

Let's use the example scenario to illustrate the usage of functionalities relating to log files. In Section 9.5.3, we retrieved an entry with an error using the MPL API. Imagine that you want to further troubleshoot the error from your custom dashboard. For

that, you need to download a copy of all log files of type HTTP around the time period that the error occurred. Note that the error occurred around 12:51 based on the Log-Start field shown earlier in [Listing 9.1](#).

The screenshot shows the SAP API Business Hub interface for the 'Log Files' entity. At the top, there are tabs for 'OVERVIEW' and 'RESOURCE'. The 'RESOURCE' tab is selected, showing the following details:

- API Endpoint:** Sandbox (<https://sandbox.api.sap.com:443/cpi/api/v1>)
- LogFileArchives** Entity Type:
 - GET** /LogFileArchives: Get all log file collections. (Code Snippet)
 - GET** /LogFileArchives(Scope='{Scope}',LogFileType='{LogFileType}',NodeScope='worker'): Get log file collections by scope and type in a compressed file. (Code Snippet)
- Show/Hide**, **List Operations**, and **Expand Operations** buttons are available for this entity.
- LogFiles** Entity Type:
 - GET** /LogFiles: Get all log files. (Code Snippet)
 - GET** /LogFiles(Name='{Name}',Application='{Application}')
 - Get log files by name and application. (Code Snippet)
- Show/Hide**, **List Operations**, and **Expand Operations** buttons are available for this entity.

Figure 9.37 Entity Types and APIs Related to Log Files

To solve this challenge, we can use the `LogFileArchives` entity by invoking the following OData endpoint:

`https://<tenant>/api/v1/LogFileArchives(Scope='all',LogFileType='http',NodeScope='worker')/$value?modifiedAfter=2018-04-29T12:50:00Z`

Let's now examine this OData endpoint to understand what is happening with the help of [Table 9.12](#).

Endpoint Attribute	Description
LogFileArchives	Entity used to retrieve an archived log file.
Scope	Indicates which scope/type of log files you want to download. Possible values include <code>all</code> to download all existing HTTP log files, and <code>latest</code> to only retrieve the latest HTTP log files.
LogFileType	Filters the result based on the type of log file. Possible values include <code>http</code> and <code>Trace</code> .

Table 9.12 Attributes Included in the OData Endpoint to Log Archive Entries

Endpoint Attribute	Description
NodeScope	Specifies that we're only interested to retrieve log files from runtime nodes (also referred to as worker nodes).
value	Specifies that the next parameters in the URL will contain parameter values.
modifiedAfter	Specifies the time after which the filtered log file was changed.

Table 9.12 Attributes Included in the OData Endpoint to Log Archive Entries (Cont.)

Note that you need to have the role `IntegrationOperationServer.read` assigned to your user to call log file APIs.

In the next section, we'll explore APIs that relate to the message store.

9.5.6 Managing Message Store Entries Using APIs

Using the OData APIs provided by SAP Cloud Platform Integration, we can access the tenant's message store entries. In scenarios with the requirement to persist messages, message content can be written and saved in the message store using the **Persist Message** step of an integration flow. You can then access the stored message and analyze it at a later point in time. However, note that a message is stored on the runtime node for 90 days. After this time, the message is automatically deleted.

For each entry of the message store, you can retrieve its properties, headers, payload, and attachments. The APIs that access message store revolve around four main entity types:

- **MessageStoreEntries**
Used to manage message store entries.
- **MessageStoreEntryProperties**
Used to manage properties of message store entries.
- **MessageStoreEntryAttachments**
Used to manage attachments from a specific message store entry.
- **MessageStoreEntryAttachmentProperties**
Used to manage properties of an attachment in the message store.

Referring to the entity model diagram presented earlier in [Figure 9.19](#), notice that there is a direct relationship between a message store entry (represented by the

entity type `MessageStoreEntries`) and `MessageProcessingLogs`. This relationship means that for every entry in the processing log with an attachment, you can try to retrieve its related message store entries (if available).

In this section, we're not going to explore all APIs included in the entities listed, but we'll look at a scenario to showcase what is possible. To get a full description of the different APIs, refer to the SAP documentation at https://help.sap.com/viewer/product/CLOUD_INTEGRATION/Cloud. You can also explore these APIs via the SAP API Business Hub at <https://api.sap.com/shell/discover/contentpackage/CloudIntegrationAPI/api/MessageStore>.

[Figure 9.38](#) shows the entities and APIs available. You can test and explore each one of them using the testing approaches we explored in [Section 9.5.1](#).

The screenshot displays the SAP API Business Hub interface for the `Message Store` entity. At the top, there are two tabs: `OVERVIEW` and `RESOURCE`, with `RESOURCE` being the active tab. Below the tabs, a note states: "SAP API Business Hub allows you to select and configure an API Endpoint, so you can test APIs against the data available in your tenancy. By default, API sandbox is selected to enable the experience." A dropdown menu labeled "API Endpoint" is set to "Sandbox (<https://sandbox.api.sap.com:443/cpi/api/v1>)".

The main content area is organized into sections for different entity types:

- MessageStoreEntries**: Contains three API operations:
 - GET /MessageProcessingLogs('{MessageGuid}')/MessageStoreEntries**: Description: Get message store entries by message Guid. | [Code Snippet](#)
 - GET /MessageStoreEntries('{MessageStoreEntryId}')**: Description: Get message store entry by Id. | [Code Snippet](#)
 - GET /MessageStoreEntries('{MessageStoreEntryId}')/\$value**: Description: Get message payload from message store by entry Id. | [Code Snippet](#)
- MessageStoreEntryAttachments**: Contains three API operations:
 - GET /MessageStoreEntries/{MessageStoreEntryId}/Attachments**: Description: Get message attachments by entry Id. | [Code Snippet](#)
 - GET /MessageStoreEntries/{MessageStoreEntryId}/Attachments/{AttachmentId}**: Description: Get message attachment by Id. | [Code Snippet](#)
 - PUT /MessageStoreEntries/{MessageStoreEntryId}/Attachments/{AttachmentId}**: Description: Update message attachment by Id. | [Code Snippet](#)
- MessageStoreEntryAttachmentProperties**: Contains three API operations:
 - GET /MessageStoreEntries/{MessageStoreEntryId}/Attachments/{AttachmentId}/Properties**: Description: Get message attachment properties by Id. | [Code Snippet](#)
 - PUT /MessageStoreEntries/{MessageStoreEntryId}/Attachments/{AttachmentId}/Properties**: Description: Update message attachment properties by Id. | [Code Snippet](#)
 - DELETE /MessageStoreEntries/{MessageStoreEntryId}/Attachments/{AttachmentId}/Properties**: Description: Delete message attachment properties by Id. | [Code Snippet](#)
- MessageStoreEntryProperties**: Contains three API operations:
 - GET /MessageStoreEntries/{MessageStoreEntryId}/Properties**: Description: Get message store entry properties by Id. | [Code Snippet](#)
 - PUT /MessageStoreEntries/{MessageStoreEntryId}/Properties**: Description: Update message store entry properties by Id. | [Code Snippet](#)
 - DELETE /MessageStoreEntries/{MessageStoreEntryId}/Properties**: Description: Delete message store entry properties by Id. | [Code Snippet](#)

Figure 9.38 Entity Types and APIs Related to the Message Store

Note that you need to have the role `esbmessagestorage.read` assigned to your user to call message store APIs.

Consider the message aggregation scenario that we explored in [Chapter 4, Section 4.6](#). In that scenario, we aggregated correlated messages. Imagine that after the aggregation is finished, we want to persist the final aggregated payload in the message

store. We can write the payload to the message store using the **Persist Message** step to our integration flow.

To achieve that, follow these steps:

1. In the **Design** section of SAP Cloud Platform Integration, open the integration flow that you created in [Chapter 4, Section 4.6](#).
2. Switch to edit mode by using the **Edit** button in the top-right corner of the integration flow screen.
3. Add the **Persist Message** step to the integration flow. The **Persist** step can be found in the palette on the left, as shown in [Figure 9.39](#).

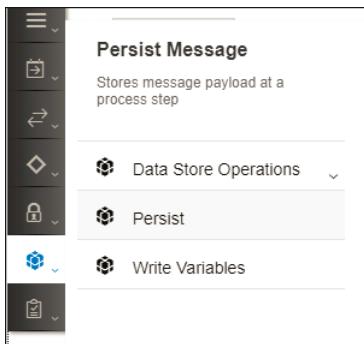


Figure 9.39 Selecting the Persist Step from the Palette

4. Ensure that the **Step ID** of the newly added step is unique. The final integration flow should look like the one presented in [Figure 9.40](#) and [Figure 9.41](#). Note that according to our integration flow, the message store is only populated after all messages have been collected because the **Persist** step comes after the **Aggregator** step.

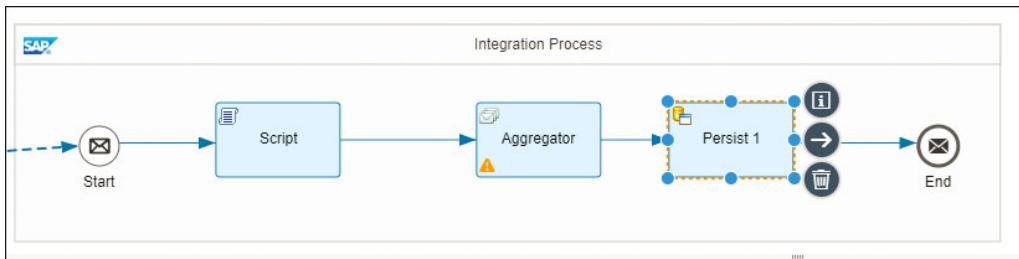


Figure 9.40 Overview of the Integration Flow Extended with a Persist Step A

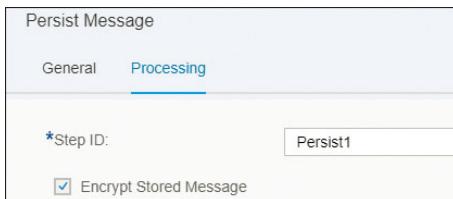


Figure 9.41 Overview of the Integration Flow Extended with a Persist Step B

5. Save and deploy the integration flow.

Now let's assume that in your custom dashboard application that you started building in [Section 9.5.3](#), you want to be warned if a failed message flow contains entries in the message store. In such a case, you can retrieve the payload of the entry in the message store.

To retrieve the payload of the entry in the message store, you'll need to call a number of OData APIs in a particular sequence:

1. Use `MessageProcessingLogs` to retrieve the list of failing messages. You already know how to query for failing messages from our discussion in [Section 9.5.3](#).
2. Use the message `Guid` of the message retrieved from the first call to make a second call to query if there are message store entries for messages with the specified message `Guid`. For that, the following endpoint can be used:

`https://<tenant>/api/v1/MessageProcessingLogs('<Guid>')/MessageStoreEntries`

The response of the API call is shown in [Listing 9.2](#).

```
<content type="application/octet-stream" src="MessageStoreEntries("sap-it-
res%3Amsg%3Aac965bd8f%3Ab694f69-73a5-4430-b681-1673c963fd4c")/$value"/>
```

Listing 9.2 Response of the OData API Call

3. Retrieve the payload of the entry in the message store. Looking at the entry returned in [Listing 9.2](#), notice the `src` attribute of the `content` element. The value of this attribute provides details regarding how we can retrieve the payload. In this example, we can use the following link to retrieve the payload:

`https://<tenant>/api/v1/MessageStoreEntries('sap-it-
res%3Amsg%3Aac965bd8f%3Ab694f69-73a5-4430-b681-1673c963fd4c')/$value`

Note that in this URL, the value between `/v1/` and `/$value` is copied from the `src` attribute of [Listing 9.2](#). The API returns the aggregated payload as depicted in [Figure 9.42](#).

```
<?xml version="1.0" encoding="UTF-8"?><multipack:Messages xmlns:multipack="http://sap.com/xi/XI/SplitAndMerge"><multipack:Message1>
<OrderItem xmlns:demo="http://hci.sap.com/demo" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <orderNumber>AA2345</orderNumber>
  <Item>
    <ItemNo>1</ItemNo>
    <Quantity>1</Quantity>
    <Unit>1</Unit>
    <LastStatus>false</LastStatus>
  </Item>
</OrderItem><OrderItem xmlns:demo="http://hci.sap.com/demo" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <orderNumber>AA2345</orderNumber>
  <Item>
    <ItemNo>2</ItemNo>
    <Quantity>5</Quantity>
    <Unit>1</Unit>
    <LastStatus>false</LastStatus>
  </Item>
</OrderItem><OrderItem xmlns:demo="http://hci.sap.com/demo" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <orderNumber>AA2345</orderNumber>
  <Item>
    <ItemNo>3</ItemNo>
    <Quantity>25</Quantity>
    <Unit>1</Unit>
    <LastStatus>true</LastStatus>
  </Item>
</OrderItem></multipack:Message1></multipack:Messages>
```

Figure 9.42 Aggregated Payload Returned by the Message Store API

Now that you know how to use the message store API, let's explore the OData APIs related to security materials.

9.5.7 Managing Security Material Using the API

Using the OData APIs provided by SAP Cloud Platform Integration, we can access the tenant's keystore (X.509 certificates and key pairs). This API contains a lot of features that can't all be explored in this section.

To get a full description of the different APIs, refer to the SAP documentation via https://help.sap.com/viewer/product/CLOUD_INTEGRATION/Cloud.

To give you an impression of how to use them, let's work with a sample scenario to illustrate its usage. Assume that you want your keystore entries to be automatically backed up at the end of each month. Given that you don't want to perform this activity manually every month, you're looking for ways to automate the process via your custom dashboard application.

You can easily achieve this automation task by getting your custom application to call the Security Material API. To be more specific, there is an API that enables you to back up all keystore entries via the endpoint: <https://<tenant>/api/v1/KeystoreResources>.

Note that you'll need to use a POST method for this request. Listing 9.3 shows an example request. It's also possible to include the query option indicated in Table 9.13 on the request.

```
{"Name": "backup_admin_system"}
```

Listing 9.3 Example Request Body to Back Up Keystore Entries

Query Option	Description
returnKeystoreEntries	Possible values include true and false. When set to true, the KeystoreEntry instances that have been backed up are returned in the response. Note that this is an optional query that is defaulted to false.

Table 9.13 Possible Query Option for Renaming an Alias

Because the keystore OData API is protected against CSRF attacks, you first have to fetch an X-CSRF token before you can make this API call. We've already explored how to Fetch X-CSRF token in [Chapter 7, Section 7.4.3](#).

Let's now explore APIs that relate to managing the Partner Directory in the next section.

9.5.8 Managing the Partner Directory Using the API

[Chapter 7, Section 7.4](#) discussed and explained the notion of the tenant Partner Directory. We explained that during a business-to-business (B2B) project, the SAP Cloud Platform Integration owner might decide to build an application where the partners involved in the scenario can maintain their specific configuration data.

As things stand at the time this book is published, Partner Directory information can only be maintained via the OData API. The HTTP addresses required to make outbound calls to the partner systems are examples of the type of data stored in the Partner Directory.

Assuming that the Partner Directory notions are clear to you, we'll now focus on the usage of the Partner Directory OData APIs provided by SAP Cloud Platform Integration. These APIs access the Partner Directory, create entries, and help manage them. The APIs revolve around the following entity types:

- AlternativePartners
- AuthorizedUsers
- BinaryParameters
- Partners

- StringParameters
- UserCredentialParameters
- CSRF Token Handling

Updated details about these APIs can be found via the API Business Hub at <https://api.sap.com/api/PartnerDirectory>.

Figure 9.43 shows the entities and APIs available. You can test and explore each one of them using the testing approaches we explored in Section 9.5.1.

The screenshot displays the SAP API Business Hub interface for the Partner Directory. At the top, there's a navigation bar with a logo and the text "Partner Directory". Below it, two tabs are visible: "OVERVIEW" and "RESOURCE", with "RESOURCE" being the active tab. The main content area lists several entity types, each with a "Show/Hide" button, a "List Operations" button, and an "Expand Operations" button. The listed entity types are: AlternativePartners, AuthorizedUsers, BinaryParameters, and Partners. Under the Partners section, there are two API entries: a blue "GET /v1/Partners" entry with "Get all partners." and "Code Snippet" buttons, and a red "DELETE /v1/Partners('{PId}')" entry with "Delete partner." and "Code Snippet" buttons. Below these are additional entity types: StringParameters, UserCredentialParameters, and CSRF Token Handling, each with their own "Show/Hide", "List Operations", and "Expand Operations" buttons.

Figure 9.43 Entity Types and APIs Related to the Partner Directory

To get an impression of how to use these APIs, refer to the sample scenario in Chapter 7, Section 7.4. The scenario used the OData API to store an endpoint URL and a credential alias in Partner Directory. It's therefore not necessary to repeat it in this section.

Notes

It's important to be aware of the following Partner Directory limitations:

- The number of AlternativePartners in the tenant is limited to a maximum of 1,000,000.

- The number of `AuthorizedUsers` in the tenant is limited to a maximum of 500,000.
- The number of `BinaryParameters` in the tenant is limited to a maximum of 400,000.
- The number of `StringParameters` in the tenant is limited to a maximum of 3,000,000.

9.6 Using SAP Cloud Platform Integration with SAP Cloud Platform API Management

In the past, APIs were mostly only known to programmers, but in today's digital era, even business executives are aware of them and their financial impact. In a digitized world, companies are generating revenue by exposing APIs like any other service they offer to their business partners, suppliers, and customers.

Companies the likes of Amazon, Facebook, Twitter, Netflix, Uber, and Google are generating huge revenues based on their APIs. So, chances are high that APIs will play a key role in the digital transformation journey of your organization. Today, APIs are managed like traditional products!

SAP Cloud Platform API Management (referred to as SAP API Management in the rest of this chapter) can help in the digital transformation journey by providing simple, scalable, and secure access to your organization's digital assets through APIs. SAP API Management enables developer communities to consume and discover your organization's APIs. Refer to <http://bit.ly/2uLnEFV> to read more about SAP Cloud Platform API Management:

Some of its key capabilities include the following, just to name a few:

- Standardized and consistent way to provision APIs via REST, OData, and SOAP
- Real-time and historic analytics on usage, error, monitoring, and traffic of APIs
- High security standard for the APIs to prevent against attacks such as Denial-of-Service attacks (DoS), cross-site scripting (XSS), cross-site request forgery (CSRF), and so on
- Robust traffic management of APIs
- Full API Lifecycle Management

- Management, discovery, testing, subscription, and consumption of APIs by the developer community
- Monetization of APIs

Figure 9.44 depicts the positioning of SAP API Management within your landscape. As shown by the figure, different applications can consume APIs via SAP API Management, which then acts as a gateway. It also proxies the calls to the backend systems (which are either on-premise or cloud-based systems). SAP API Management connects to these backend systems via various protocols, such as SOAP, REST, OData, and so on.

Figure 9.44 shows the different personas involved with SAP API Management:

- **External applications (Mobile, Web, etc.)**
These applications consume the APIs provided by SAP API Management.
- **Apps Developer**
This developer is responsible to make the external applications consume APIs.
This developer needs to be able to discover existing APIs and figure out how to easily consume them.
- **API Developer**
This person is responsible to design and implement the API via SAP API Management.
- **API Admin, Owners**
These people are responsible to administer and manage the APIs via monitoring, analyzing, and monetizing processes.

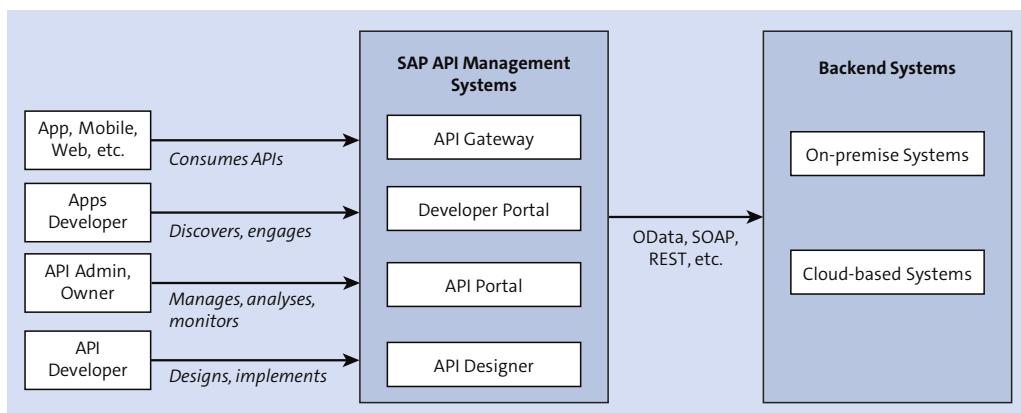


Figure 9.44 Positioning of SAP API Management and Its Personas

This positioning presented by [Figure 9.44](#) also means that SAP API Management can proxy services provided by SAP Cloud Platform Integration, which is the main subject of this section.

SAP API Management is a big topic that deserves its own book. In the next sections, we briefly explore how SAP API Management can be used to publish APIs from services provided by SAP Cloud Platform Integration in a secure manner. To find out more about SAP API Management, follow different tutorials on the SAP community page (www.sap.com/community.html).

Note that SAP API Management also sits on top of SAP Cloud Platform and is included in the SAP Cloud Platform Integration, Enterprise Edition, as discussed in [Chapter 1, Section 1.4](#). It's also possible to register for a free trial account at <https://account.hanatrial.ondemand.com/#/home/welcome>.

You might be wondering what the difference is between SAP API Management and SAP API Business Hub. SAP API Management enables any organization to expose its own APIs. It also allows their business partners to discover and consume these APIs in a secure manner. As for SAP API Business Hub, it allows you to discover, explore, and test the APIs offered by SAP.

One of the first things that needs to happen after obtaining your SAP API Management tenant is to establish a connection to your SAP Cloud Platform Integration tenant. Let's explore how that can be achieved next.

9.6.1 Establish a Connection between SAP Cloud Platform Integration and SAP API Management

Note that connecting SAP API Management to SAP Cloud Platform Integration is a one-time action. After that, this connection can be reused.

To connect SAP API Management to SAP Cloud Platform Integration, follow these steps:

1. Log in to your SAP API Management tenant via the following URL (if there is a trial account): <https://account.hanatrial.ondemand.com/cockpit>.
The link to the productive account is given in the tenant provisioning mail received from SAP when getting a new tenant.
2. Navigate to the **Services** section on the left menu (see [Figure 9.45](#)).
3. Click on the **API Management** link under the **Integration** section (see the right panel of [Figure 9.45](#)).

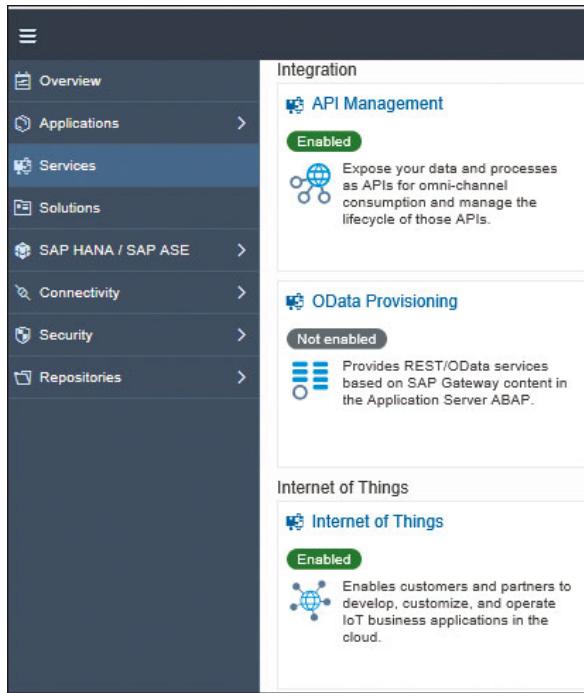


Figure 9.45 Navigating to the SAP API Management Service

4. On the next screen, select the **Access API Portal** link (see [Figure 9.46](#)).

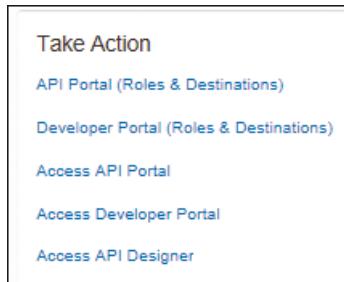


Figure 9.46 Landing Page of SAP API Management with Overview of Possible Actions

5. The next screen presents an overview of the API Portal, which contains information such as API traffic, error, usage, performance, applications, and so on. From here, select the **Develop** link on the menu located in the left panel (see [Figure 9.47](#)).

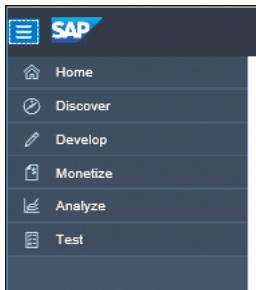


Figure 9.47 Overview of the API Portal

6. On the next page, select the **API PROVIDERS** tab at the top (see [Figure 9.48](#)). Click the **Create** button to add a new API Provider. An API provider represents the back-end system that will receive and execute the call.

A screenshot of the SAP API Portal's main interface. At the top, there's a navigation bar with the SAP logo and links for Home, Discover, Develop, Monetize, Analyze, and Test. Below the navigation bar, the main content area has a title "API Portal". In the center, there's a table-like structure with tabs for "API PROVIDERS (0)", "CERTIFICATES (0)", "APIS (0)", "PRODUCTS (0)", "APPLICATIONS (0)", and "POLICY TEMPLATES (0)". The "API PROVIDERS" tab is currently selected. To the right of the tabs, there's a blue "Create" button. Below the tabs, there are columns for "Name", "Created By", "Created On", and "Changed By". A message "No data" is displayed. The left side of the screen has a sidebar with the same navigation links as the top bar.

Figure 9.48 API PROVIDERS Tab

7. On the next page, provide a name and description for the new API provider (see [Figure 9.49](#)).

A screenshot of a dialog box titled "Add API Provider". At the top left is a back arrow and the title "Add API Provider". At the top right are "Save" and "Cancel" buttons. The main area contains a section for "CloudIntegration" with a "Name" field containing "CloudIntegration". Below this is a "Description" field with the text "Connection to Cloud Integration". At the bottom of the dialog, there are three tabs: "OVERVIEW" (which is selected), "CONNECTION", and "CATALOG SERVICE SETTINGS".

Figure 9.49 Providing a Name and Description

8. Provide the hostname, authentication, and other details related to your SAP Cloud Platform Integration tenant, as shown in [Figure 9.50](#) and [Figure 9.51](#).

The screenshot shows the 'Add API Provider' interface for a tenant named 'CloudIntegration'. The 'CONNECTION' tab is selected. The configuration includes:

- *Name: CloudIntegration
- *Type: Internet
- *Host: -ifimap.hcisbp.eu1.hana.ondemand.com
- *Port: 443
- Use SSL: checked
- Trust Store: empty field
- Key Store Certificate: empty field

Figure 9.50 Connection Details to the SAP Cloud Platform Integration Tenant

9. Finally, click the **Save** button in the top-right corner of [Figure 9.51](#).

The screenshot shows the 'CloudIntegration' tenant configuration. The 'CATALOG SERVICE SETTINGS' tab is selected. The 'Catalog Service Settings' section contains:

- Path Prefix: empty field
- Service Collection URL: empty field
- Trust All: unchecked checkbox
- Catalog URL: https://-ifimap.hcisbp.eu1.hana.ondemand.com:443

The 'Authentication' section contains:

- Type: Basic
- *UserName: S001
- *Password: masked input field

Figure 9.51 Authentication Details for the SAP Cloud Platform Integration Tenant

The connection between SAP API Management and SAP Cloud Platform Integration is now ready. In the next section, we explore how to expose an existing integration flow as a REST API.

9.6.2 Provision Application Programming Interfaces

There is a lot to learn around the topic of provisioning APIs through SAP API Management. Our intention isn't to provide a guide for SAP API Management in this section, but rather to give you a glimpse of what it can do and how it can work in combination with SAP Cloud Platform Integration to create APIs.

To illustrate the provision of an API via SAP API Management, let's once again use the integration flow example that we built in [Chapter 4, Section 4.3](#). We already used this integration flow in [Section 9.3](#) when exploring the **Script** step (refer to [Figure 9.11](#)). This scenario currently exposes a SOAP endpoint. Our goal will be to provide this SOAP service as a REST-based API in SAP API Management and apply restrictions to it by limiting the number of calls per minute. This approach of wrapping an existing service as an API is known as API Proxy. REST APIs can accept both JSON or XML as the payload. To simplify, we stick to XML for our scenario. The final end-to-end scenario is presented in [Figure 9.52](#). Because the integration flow already exists in SAP Cloud Platform Integration, we only need to expose it as an API.

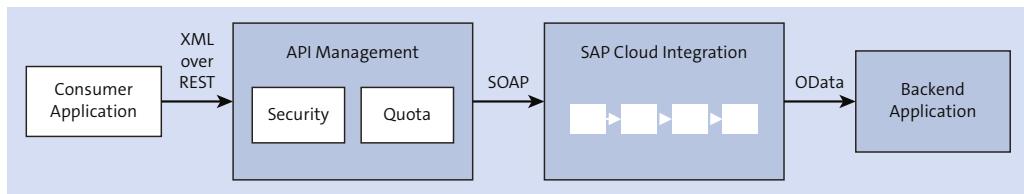


Figure 9.52 End-to-End Overview of Scenario

Let's recall from [Chapter 4, Section 4.3](#), that the endpoint of the concerned integration flow was of the form: https://<tenant>/cx/CPBook_Demo_OData.

We can now start the provisioning of our API by first navigating to the landing page of SAP API Management as shown in [Figure 9.53](#) [Figure 9.54](#). We already described how to get to this page in [Section 9.6.1](#).

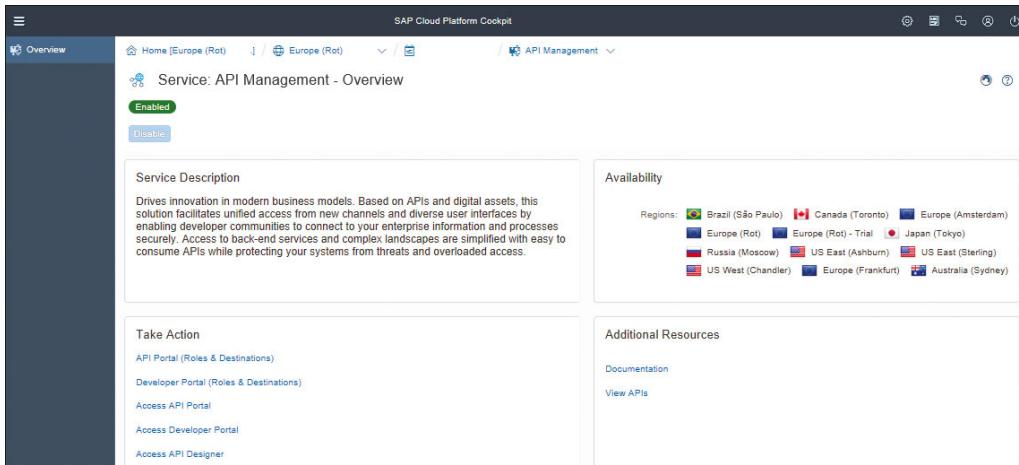


Figure 9.53 Main Page of SAP API Management



Figure 9.54 SAP API Management, Take Action Options

The page presented in [Figure 9.53](#) includes the following options (under **Take Action**):

- **API Portal (Roles & Destinations)**

Used to assign roles to users who are allowed to create APIs. It also enables the creation of destinations to different API providers. A destination is an object that contains connection details to a remote system or application. The connection details include the URL of the remote system or service, authentication type, and the user credentials. It's defined once and reused throughout the system.

- **Developer Portal (Roles & Destinations)**

Used to assign roles to users interested in discovering and consuming APIs. They are also commonly referred to as developers from an SAP API Management point of view. Additionally, from this option, it's possible to create destinations to different API providers.

■ Access API Portal

Enables access to various monitoring matrices about APIs, applications, and products. This is also the place where you can design and create new APIs.

■ Access Developer Portal

Enables the developer community to access and subscribe to available APIs in SAP API Management.

■ Access API Designer

An API editor that includes rich capabilities to import existing open APIs, download APIs, generate equivalent HTML output views, and validate open API syntax.

From the page in [Figure 9.53](#), click on the **Access API Portal** link. You're redirected to a page that looks like [Figure 9.55](#).

From this page, you can access monitors and statistics about API performance, traffic, usage, and errors. Furthermore, you can create new APIs, API providers, and applications.

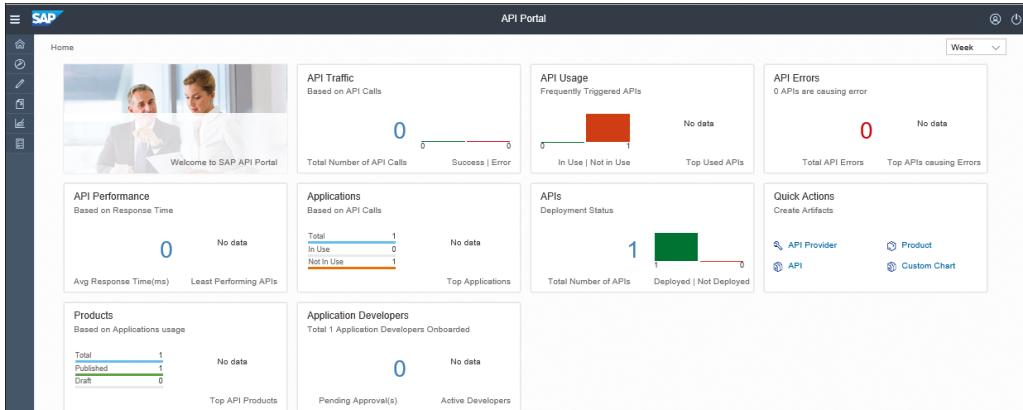


Figure 9.55 API Portal Landing Page

Let's go on and create an API by clicking on the **API icon**  , which can be found on in the **Quick Actions** tile of [Figure 9.55](#). A page pops up that allows you to fill in the details of the API proxy. The API proxy needs to point to the integration flow in SAP Cloud Platform Integration. Specify the details as shown in [Figure 9.56](#) and [Figure 9.57](#). Note that both [Figure 9.56](#) and [Figure 9.57](#) are part of the same page.

Create API

API Provider: CloudIntegration Link API Provider [?](#)

*URL [?](#) /cxif/CPI_Book_Demo_OData

API Details

*Name: ShipDetails

*Title: Retrieve Shipping details

Description:

B I U S | █ █ █ █ █ | Verdana ▾ 11pt ▾ A ▾ A ▾
✖ ↻ ⌂ | █ █ █ █ █ | 🔍 🔍

Call this API to retrieve all relevant Shipping Details given an order number as an input

Create Cancel

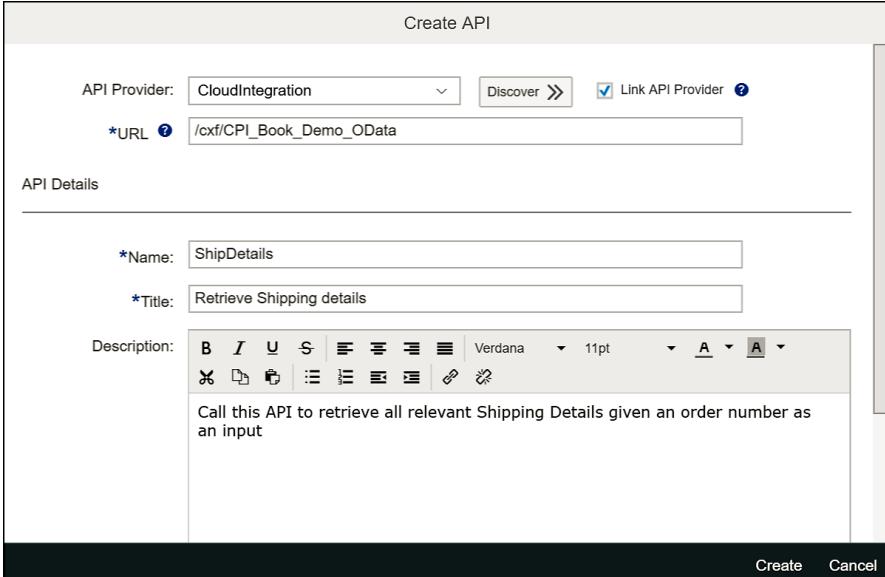


Figure 9.56 Adding Details of the API Proxy to Be Created (Top of Screen)

Create API

*Name: ShipDetails

*Title: Retrieve Shipping details

Description:

B I U S | █ █ █ █ █ | Verdana ▾ 11pt ▾ A ▾ A ▾
✖ ↻ ⌂ | █ █ █ █ █ | 🔍 🔍

Call this API to retrieve all relevant Shipping Details given an order number as an input

*Host Alias: [?](#) s0004426257trial-trial.apim1.hanatrial.ondemand.com

*API Base Path: [?](#) /ShipmentDetails

Service Type: REST

Create Cancel

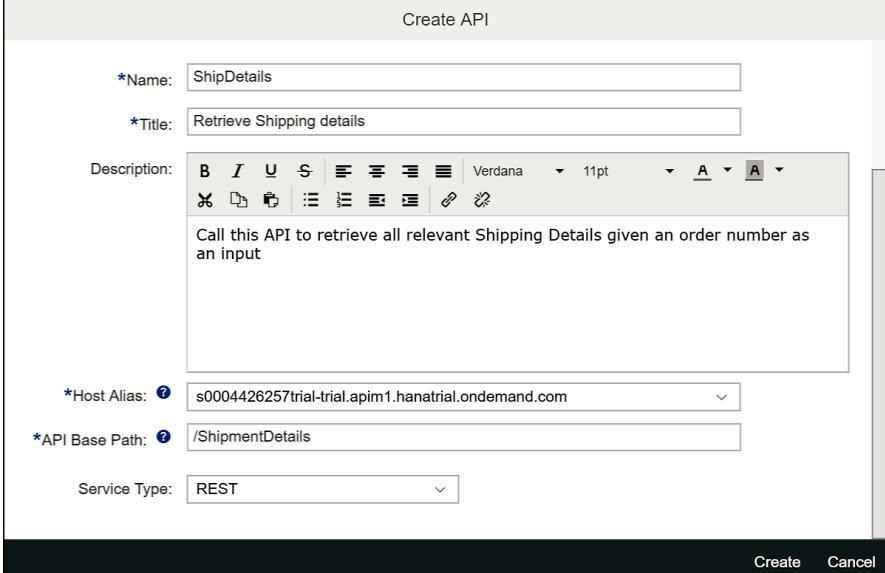


Figure 9.57 Adding Details of the API Proxy to Be Created (Bottom of Screen)

Table 9.14 provides a description of the fields used in Figure 9.56 and Figure 9.57.

Field	Description
API Provider	Specifies the API provider that we created in <u>Section 9.6.1</u> to connect to the SAP Cloud Platform Integration tenant.
URL	The last part of the integration flow's endpoint, which can be found from integration artifact monitoring, as discussed in <u>Chapter 4, Section 4.1</u> .
Name	Meaningful name for the API.
Title	Title for the API.
Description	Description for the API.
Host Alias	Automatically populated with the host details of our SAP API Management tenant. Leave this field with its default value.
API Base Path	Specifies the base path to be used as part of the endpoint for the API.
Service Type	Possible values include REST, SOAP, and ODATA.

Table 9.14 API Proxy Fields

In Figure 9.58, you can add the resource by clicking on the **Add** button. An API can have multiple resources, and a resource represents an endpoint.

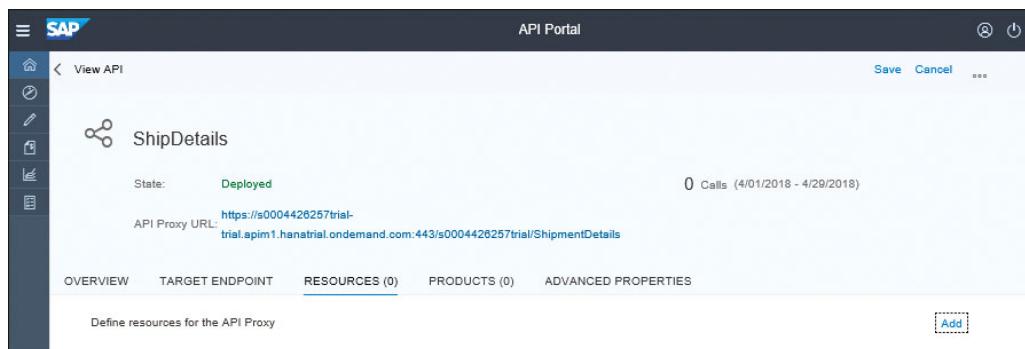


Figure 9.58 Resource Tab of the API Proxy

From the next screen (Figure 9.59), enter the name of the resource as “Shipment”. Select the **POST** checkbox as the supported HTTP method, and specify a description. Then click on the **Add** button.

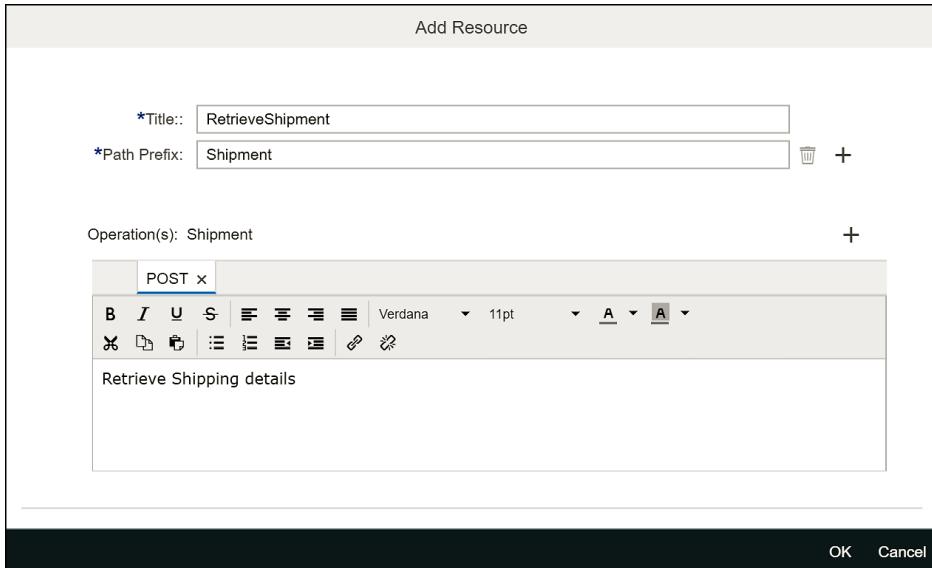


Figure 9.59 Adding a Resource to an API

Let's now add some policies to our API. An API policy is a module that implements a specific API behavior. Policies are built to let you add common management capabilities to an API such as security, rate-limiting, transformation, and mediation. You can access the policy editor by clicking the **Policies** button on the top-right corner of the screen (see [Figure 9.60](#)).

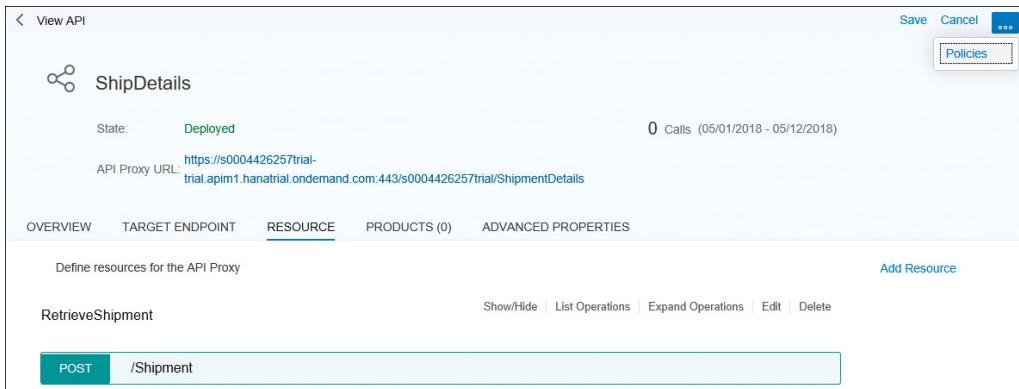


Figure 9.60 Navigating to the Policy Editor

You're redirected to the policy editor page from where a wide variety of policies can be added to fulfill your requirements (see [Figure 9.61](#)).

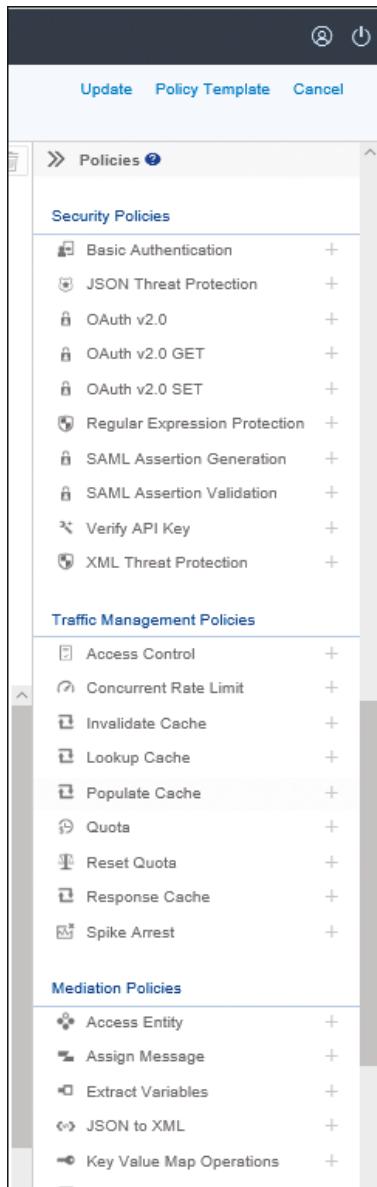


Figure 9.61 An Overview of the Policy Editors

At the time of publishing this book, the following categories of policy are included in the editor:

- **Security**

Includes various policies aimed at protecting your API. You can add basic authentication, different versions of OAuth, SAML, XML thread protection, verify API keys, and so on.

- **Traffic management**

Helps you regulate your API traffic using techniques such as caching, quotas, access control, concurrent rate limit, and so on.

- **Mediation**

Actions and scripts, such as extraction of variables and conversion of JSON to XML (and vice versa), are bundled in this category.

Note that any number of policies included in this editor can be mixed together and used in combination to achieve your desired requirements. The rectangular shaped image included in the middle of [Figure 9.61](#) represents a policy flow in SAP API Management. It defines a processing pipeline and the order of execution of the included policies.

The flow has two main execution paths: the **PreFlow** and **PostFlow**. The **PreFlow** is the top part of the rectangle and represents the actions to be executed first before the control is passed to the API provider (SAP Cloud Platform Integration in our case). The **PostFlow**, on the other hand, is the bottom part of the rectangle and represents the actions to be performed after the API provider has been called. These are, for instance, actions to be performed before sending the response back to the API consumer.

For simplicity, let's add a policy to check API calling quotas. Assuming that our backend system can only accept a limited number of calls per minute, let's use SAP API Management to limit API consumption to a maximum of two API calls per minute. This could be a good way to regulate traffic by protecting our backend system from being flooded with calls. It makes sense to add the quota policy in the **PreFlow**. Therefore, it prevents calls to SAP Cloud Platform Integration if there is a quota violation.

Let's start by selecting **PreFlow** in the top-left corner under the **Flows** section (refer to [Figure 9.61](#)). Then click on the plus icon  next to **Quota** on the right panel, under the **Traffic Management Policies** section. You are then presented with a pop-up to provide the name of the policy, as indicated by [Figure 9.62](#).

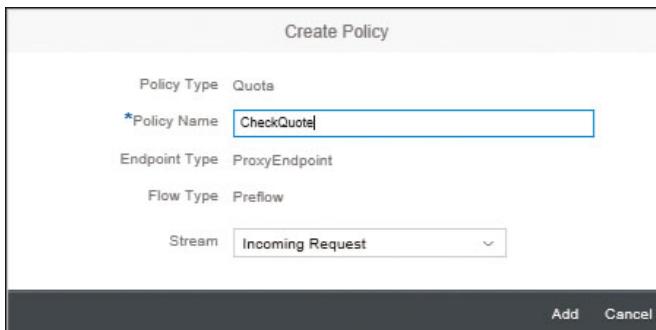


Figure 9.62 Adding the Quote Policy

Beside providing a name for the policy, select the **Incoming Request** value for the **Stream** dropdown, as shown in [Figure 9.62](#). Then click on the **Add** button. In the next screen, specify the maximum number of allowed API calls per minute in the tag element **Allow count**, as depicted in [Figure 9.63](#). In our case, we need to use the value 2 to fulfill our previously stated requirement.

```

1  
2  <Quota async="false" continueOnError="false" enabled="true" type="calendar" xmlns="http://www.sap.com/apimgt">
3    <!-- specifies the number of requests allowed for the API Proxy -->
4    <Allow count="2"/>
5    <!-- the interval of time for which the quota should be applied -->
6    <Interval>1</Interval>
7    <!-- used to specify if a central counter should be maintained and continuously synchronized across all message processors -->
8    <Distributed>true</Distributed>
9    <!-- Use to specify the date and time when the quota counter will begin counting,
10       regardless of whether any requests have been received from any apps -->
11   <StartTime>2015-2-11 12:00:00</StartTime>
12   <!-- if set to true, the distributed quota counter is updated synchronously. This means that
13       the update to the counter will be made at the same time the API call is quota-checked -->
14   <Synchronous>true</Synchronous>
15   <!-- Use to specify the unit of time applicable to the quota. Can be second, minute, hour, day, or month -->
16   <TimeUnit>minute</TimeUnit>
17 </Quota>
```

Figure 9.63 Configuration of CheckQuota

Click on the **Update** button in the top-right corner to update the API with the added policies (see [Figure 9.64](#)).

To find more information about any policy, refer to the SAP Documentation at <https://help.sap.com/viewer/product/CP/Cloud> (and search for **SAP Cloud Platform API Management**).

In the next screen, click on the **Save** button. Well done, you're now ready to consume the API in the next section.

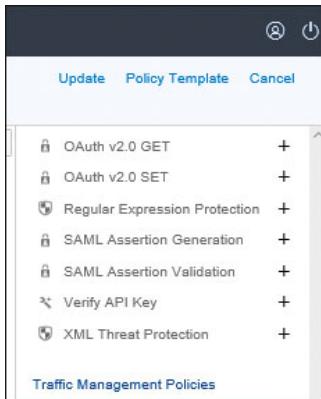


Figure 9.64 Updating the API with the Flow of Policies

9.6.3 Consume the Application Programming Interface

Now that the API is ready, we can test it using any REST client of your preference (e.g., Postman). You can also perform a test directly in SAP API Management by navigating to the test tool via the **Test** icon , as shown at the bottom of the left menu in [Figure 9.65](#).

The screenshot shows the SAP API Portal interface. At the top, there is a navigation bar with icons for Home, View API, Policies, Debug, Edit, and more. The main area displays an API named 'ShipDetails'. The 'OVERVIEW' tab is selected. The API is listed as 'Deployed'. The 'API Proxy URL' is shown as <https://s0004426257trial.apim1.hanatrial.ondemand.com:443/s0004426257trial/ShipmentDetails>. Below this, there are sections for 'Title', 'API Proxy URL', 'Description', 'Version', 'Created By', 'Changed By', and 'Changed On'. The 'TARGET ENDPOINT', 'RESOURCE', 'PRODUCTS (0)', and 'ADVANCED PROPERTIES' tabs are also visible.

Figure 9.65 Overview of the Created API

You're then redirected to a new test page from where you should select **ShipDetails** on the left of the screen shown in [Figure 9.66](#). Note that, by default, an endpoint ending with **/SWAGGER_JSON** is selected (see [Figure 9.66](#)). You'll need to select the other endpoint from the dropdown list. The correct endpoint will end with the prefix used while creating the resource (in our scenario, **/Shipment**).

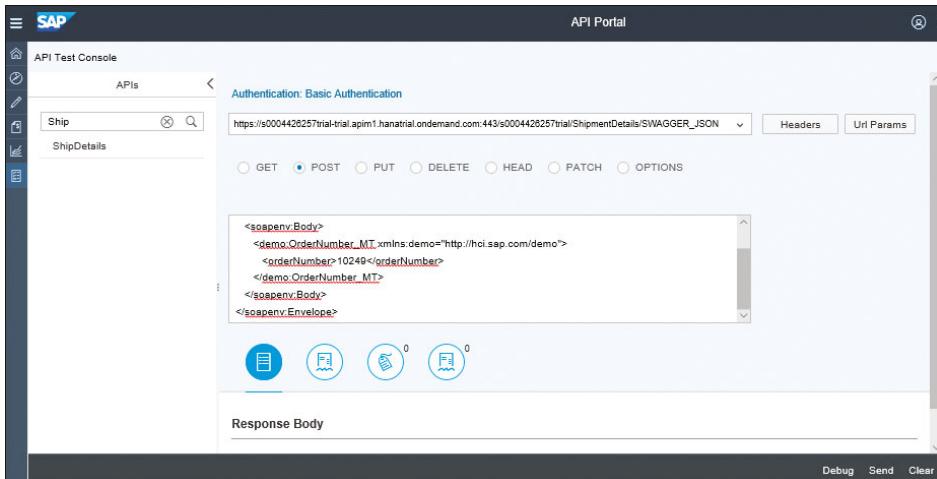


Figure 9.66 Setup of the API Test Tool

Note

If you're using an external REST client such as Postman, you'll need an endpoint to post the API call to. The endpoint of the newly created API can be obtained via the dropdown field in [Figure 9.66](#).

After selecting the correct endpoint, specify the HTTP method as **POST** by selecting its radio button, provide the desired request XML payload, and change the authentication method to basic authentication. The authentication can be changed by clicking the **Authentication** link on the top-left corner of [Figure 9.66](#).

We can now trigger the call by clicking on the **Send** button at the bottom-right corner of the screen. In the background, SAP API Management performs a check to validate whether we're still within our quota limits. Because this is the first message, the call is accepted and sent to SAP Cloud Platform Integration, which in turn returns a valid response, as shown in the **Response Body** section of [Figure 9.67](#).

Quickly perform the same call two more times to exceed our quota limit. You'll then get a quota violation error, as shown in [Figure 9.68](#). This error insinuates that we've exceeded the quota of two calls within the same minute.

The screenshot shows the API Test Console interface. At the top, there's a navigation bar with 'APIs' on the left and 'API Test Console' in the center. Below the navigation is a header for 'Authentication: Basic Authentication' with a URL input field containing 'https://s0004426257trial.trial.apim1.hanatrial.ondemand.com:443/s0004426257trial/ShipmentDetails/Shipment'. There are tabs for 'Headers' and 'Url Params'. Below the URL are method selection buttons: GET, POST (selected), PUT, DELETE, HEAD, PATCH, and OPTIONS. The main content area displays the XML request body:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:demo="http://hci.sap.com/demo">
<soapenv:Header>
<soapenv:Body>
<demo:OrderNumber_MT>
<orderNumber>10249</orderNumber>
</demo:OrderNumber_MT>
```

Below the request body are four circular icons with blue outlines: a document icon (highlighted in blue), a map icon, a clipboard icon, and another map icon. To the right of these icons are the numbers '23' and '0'. The 'Response Body' section shows the XML response body:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope">
<soap:Header></soap:Header>
<soap:Body>
<ns1:OrderShippingDetails_MT xmlns:ns1="http://hci.sap.com/demo">
<orderNumber>10249</orderNumber>
<customerName>Tom Spezialitaten</customerName>
<shipCity>Münster</shipCity>
<shipStreet>LUISENSTR.48</shipStreet>
<shipPostalCode>44887</shipPostalCode>
<shipCountry>DE</shipCountry>
<shipDate>1996-07-18T00:00:00.000</shipDate>
</ns1:OrderShippingDetails_MT>
</soap:Body>
</soap:Envelope>
```

At the bottom of the response body section, it says 'Status Code: 200 OK | Response Time: 2.52 secs | Content Length: 488 Bytes'. On the far right, there are buttons for 'Debug', 'Send' (disabled), and 'Clear'.

Figure 9.67 The Test Result of Our API Call

This screenshot shows the same API Test Console interface after exceeding the quota. The 'Response Body' section now contains a JSON object representing an error response:

```
{
  "fault": {
    "faultstring": "Rate limit quota violation. Quota limit exceeded. Identifier : _default",
    "detail": {
      "errorcode": "policies.ratelimit.QuotaViolation"
    }
  }
}
```

Figure 9.68 Response When Quota Is Exceeded

Congratulations, you successfully provided, called, and consumed the API. However, it's important to remember that this was a simplistic scenario. In a real-life scenario, you should consider the following additional aspects:

- **Authentication in SAP API Management**

By default, the API exposed can be consumed without authentication. If you want to protect the API, you'll need to add the relevant policy (e.g., OAuth, API key, etc.).

- **Authentication in SAP Cloud Platform Integration**

In our scenario, we're simply passing along to SAP Cloud Platform Integration whatever header was provided during the API call. This means that the username/password details used while calling the API are also forwarded and used for authentication in SAP Cloud Platform Integration. You could decide to use the basic authentication security policy to provide the login details for SAP Cloud Platform Integration.

- **Payload**

Note that the sample request message used in [Figure 9.66](#) includes the entire SOAP message. Given that we're exposing a REST API, we could decide to only use the SOAP body as input. But that will require us to perform some logic to extract the relevant data from the incoming message and construct the SOAP message to send to SAP Cloud Platform Integration. This is necessary because the service that we're consuming in SAP Cloud Platform Integration is of type SOAP.

- **Error message**

Note that the error response returned by the API when the quota was exceeded (in [Figure 9.68](#)) is in JSON format. In a real-life scenario, you'll want to convert the response to XML before returning it to the consumer. As a hint, consider the JSON2XML policy.

These points aren't tackled in this chapter given that it falls out of the chapter scope.

You now understand how to use SAP API Management to wrap services available in SAP Cloud Platform Integration and expose them as APIs, as well as how to enforce different policies.

9.7 Summary

This chapter introduced you to the API capabilities and features of SAP Cloud Platform Integration. An overview of Java-based APIs was given, and you've also learned about how to use them in UDFs. The chapter then explored a number of OData APIs

available in SAP Cloud Platform Integration and used some examples to illustrate their usage.

Lastly, the chapter explored how to use SAP Cloud Platform Integration in combination with SAP API Management to provision APIs. Even though this isn't intended to be a chapter on SAP API Management, you also learned how to add policies to the APIs using a scenario.

In the next chapter, we'll explore different security topics in SAP Cloud Platform Integration.

Chapter 10

SAP Cloud Platform Integration Security

Using an integration platform in the cloud implies that your data is processed on servers outside the boundaries of your organization and thus beyond your influence. This raises the question of how secure the data hosted by the software provider actually is and how it can be protected. This chapter summarizes the measures taken by SAP to protect your data at the highest level and shows what you can do to maximize the security level of your integration scenarios.

A fundamental characteristic of using cloud software as compared to on-premise software is the fact that you, the customer, hand over some of the responsibility for your data to the software provider. In particular, using an integration platform-as-a-service (iPaaS) implies that your data is processed on servers outside your organization. Your data is also stored on servers hosted by the software provider. Therefore, the topic of security is of a fundamental and existential importance.

This chapter will show that your data is in safe hands with SAP. It answers two questions: How secure is the data passing through SAP Cloud Platform Integration during the execution of an integration scenario, and what measures are implemented by SAP to exclude the risk of malicious actions taken on your data?

In [Section 10.1](#) and [Section 10.2](#), we'll cover the following:

- Security mechanisms that are already in place (i.e., inherently provided by the technical infrastructure)
- Security levels that are guaranteed by the processes associated with the development and use of SAP Cloud Platform Integration

Section 10.3 covers the topic of user management that comes first when a customer starts working with SAP Cloud Platform Integration. Here you learn how to set up dedicated permissions for people involved in an integration project.

Section 10.4 shows how you can use the tools provided with SAP Cloud Platform Integration to configure integration scenarios with the highest possible protection of the involved data. Finally, Section 10.5 explains how to manage the lifecycle of keys in the tenant keystore, a central component required to set up secure integration scenarios.

The tutorials provided in this chapter focus in the security-related aspects of the described integration scenarios. They won't repeat each step required to set up the scenario because after walking through the previous chapters, you're now already an integration expert who knows how to design various integration flows. Exceptions are those tutorials where we introduce certain security-related integration flow steps (Section 10.4.7) or adapters that we haven't introduced yet (e.g., the Twitter adapter in Section 10.4.5).

10.1 Technical System Landscape

In this section, we'll discuss the security level that is imposed by the technical system landscape. We'll first cover the security of the software architecture and the network design. Then, we provide a quick glance into the physical data security, provided by the fact that the virtual machines (VMs) that process your messages are operated at SAP data centers. Next, we discuss how data *at rest*—that is, data stored at various steps during the processing of a message—is protected by the infrastructure. Finally, we close this section by showing how data protection and privacy are ensured when using SAP Cloud Platform Integration.

10.1.1 Architecture

In this section, we'll focus on those aspects of the architecture that make SAP Cloud Platform Integration a secure cloud-based integration platform.

In Chapter 2, Section 2.1, we showed that a clustered, virtual design establishes SAP Cloud Platform Integration as an integration platform that is shared by many participants and allows flexible allocation of resources for different participants. We also

introduced the basic concept of tenant isolation, which ensures that the platform resources assigned to different participants are strictly isolated from each other. Each participant has an SAP Cloud Platform subaccount and tenant assigned to it. On each tenant, an individual (virtual) integration runtime—a tenant cluster—is installed, which is strictly separated from those of the other tenants.

In other words, data and processes related to different customers are kept apart. However, in a more general sense, a customer can set up different tenant clusters for different purposes (e.g., test and production). The architecture of SAP Cloud Platform Integration makes sure that the processes and data belonging to those different use cases are strictly isolated from each other.

We'll now take a closer look at the architecture and show how tenant isolation is achieved. We'll also show which concepts are in place for secure communication between the involved participants and technical components.

In [Chapter 2, Section 2.1](#) (see Figure 2.7), we introduced the architecture of SAP Cloud Platform Integration and provided a glance of the internal structure of the integration platform. We'll now address the following additional questions:

- How is user access to SAP Cloud Platform Integration restricted, and how is the platform protected against unauthorized access?
- How is *data at rest* (stored data) protected?
- How is *data in transit* (data exchanged between the participants and SAP Cloud Platform Integration) protected?
- How is secure communication implemented between the participants and the involved components of the cluster?

[Figure 10.1](#) shows the high-level architecture view from [Chapter 2](#), enriched with information on the security protocols used for the involved communication paths. In addition to this, [Figure 10.1](#) also adds the Secure Shell File Transfer Protocol (SFTP) server for the use cases where SFTP is chosen as transport-level security (which we skipped in [Chapter 2](#) for the sake of simplicity).

We'll walk you through this figure and explain the security-related aspects of the architecture.

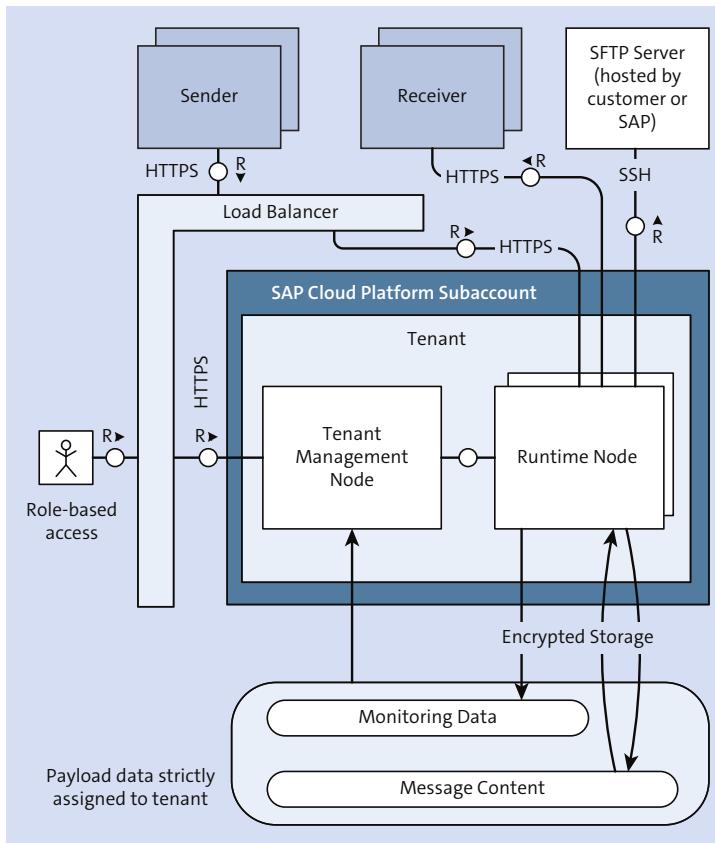


Figure 10.1 High-Level Architecture with Technical Connections, Transport-Level Security Options, and Persistence Steps

Role-Based User Access and User Management

Let's begin with the interaction points for dialog users. Access to SAP Cloud Platform Integration in the context of specific tasks (e.g., deploying integration content on a tenant) is always controlled by authorizations based on user-to-role assignments. User-to-role assignments are accomplished through the SAP Cloud Platform cockpit. You can access this application through the URL provided to you by SAP in the email informing you about the details of the tenant you requested (see [Chapter 2](#)).

With regard to user management and authorization, all platform resources are strictly isolated: in other words, a user with specific permissions on a test cluster might not have the same permissions on a productive cluster.

Each customer is given access to one or more subaccounts (by default, to one test and one productive account). The clustered design of SAP Cloud Platform Integration and the role-based access to the platform make sure that, first, each subaccount is reserved for a dedicated set of users who are completely isolated from the user setup associated with other subaccounts of the platform. In larger companies, the users assigned to a test account will be associated with persons other than those assigned to the productive subaccount.

Secondly, the SAP Cloud Platform user management allows you to assign to the users of each subaccount granular roles or authorization groups (specific groups of roles) that are tailored to the different sets of tasks that come into play in an integration project. We cover this topic in more detail in [Section 10.3](#).

Secure Data Storage

The separation of data belonging to different tenants is based on the fact that although different tenants of SAP Cloud Platform Integration might physically share one common database, each tenant stores its data in a separate database schema or Java Message Service (JMS) queue. This ensures that data is strictly separated and isolated per tenant.

In [Section 10.1.3](#), we focus in more detail on this aspect.

Secure Communication of the Technical Components

The separation of platform resources covers more than data storage—tenant isolation is also related to the processing of data and the way the various components involved in an integration scenario communicate with each other.

Let's take a closer look at how the connections between these components are secured. The SAP Cloud Platform Integration runtime is established as a cluster of multiple nodes of different types, with each node type responsible for specific tasks. These nodes (VMs) have various connections to other components:

- **HTTPS connections to remote components**

A common protocol that can be used both for inbound communication (when SAP Cloud Platform Integration is addressed by an incoming request) and for outbound communication (when SAP Cloud Platform Integration sends a message to a receiver) is HTTPS. This protocol comes with Transport Layer Security (TLS).

For inbound HTTPS requests, we first need to differentiate between requests from the user interfaces (UIs) to the tenant management node (TMN) (e.g., when a user

connects to the tenant using the Web UI) and, second, requests from external sender systems (to get messages processed). In both cases, inbound communication is forwarded to the runtime node by a load balancer. As a result, the load balancer terminates incoming TLS requests and establishes new ones.

- **Intra-cluster communication**

The intra-cluster communication (between TMNs and runtime nodes) is accomplished by a messaging service.

In [Section 10.4.2](#), we cover how HTTPS-based on TLS works in more detail.

Additional Protocols Supported

The following additional protocols are supported by SAP Cloud Platform Integration:

- Communication between SAP Cloud Platform Integration and external components can also be secured using SFTP. When using SFTP, an SFTP server needs to be connected to the tenant (through a dedicated SFTP adapter).
- Another set of protocols (not depicted in [Figure 10.1](#)) is also supported when exchanging e-mails using SAP Cloud Platform Integration:
 - For connections using the mail sender adapter: Post Office Protocol version 3 over TLS/SSL (POP3S) and Internet Message Access Protocol over TLS/SSL (IMAPS)
 - For connections using the mail receiver adapter: Simple Mail Transfer Protocol Secure (SMTPS)

For more information, see [Table 10.3](#).

10.1.2 Network Infrastructure

In this section, we cover the security aspects of the *network design*. We'll also expand our focus a bit and show how SAP manages the SAP Cloud Platform Integration clusters in a secure way.

Let's consider that a network is a setup of physical machines or VMs that communicate with each other in a well-defined and controlled way. A network comprises different *network segments*, and all components within one network segment are on the same trust level. The trust level determines what kind of communication is allowed, based on the implemented protocols and security settings. As such, the specific design of the network determines how the various components of the SAP Cloud Platform Integration infrastructure are arranged in a network and protected by various measures, such as firewalls.

Figure 10.2 shows the network design of SAP Cloud Platform Integration from a bird's-eye perspective. For simplicity, the SFTP use case is left out in this figure.

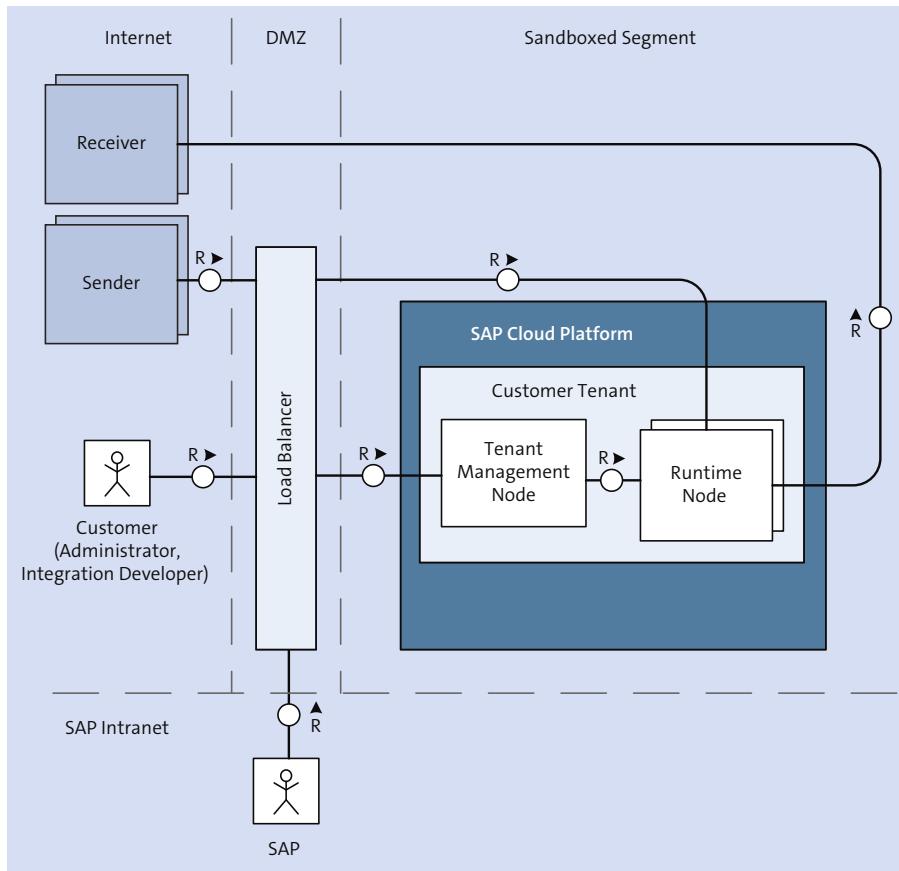


Figure 10.2 High-Level Network Design of SAP Cloud Platform Integration

As already shown in Figure 10.1, external components (sender and receiver systems as well as dialog users on the customers' side) access the SAP Cloud Platform Integration platform from the Internet. In terms of the network view, the Internet is a large and untrusted network. The components of SAP Cloud Platform Integration that process sensitive customer data can't directly be called by components from the Internet, as the load balancer is interconnected and terminates each inbound TLS request. After the load balancer has established a new TLS request, the external call is forwarded to a VM of the SAP Cloud Platform Integration cluster that actually processes

the request. As you saw in [Section 10.1.1](#), this pattern applies both for sender components that call SAP Cloud Platform Integration from the Internet and for dialog users that connect to SAP Cloud Platform Integration through a web browser (when, e.g., a user designs integration flows with the Web UI). In terms of the network design, the load balancer resides in the Demilitarized Zone (DMZ). It's common practice for organizations not to expose their external-facing services directly to the Internet and instead locate these services in a DMZ.

The VMs that process customer data aren't only shielded from the outside world. As shown in [Figure 10.2](#), the VMs of a SAP Cloud Platform Integration cluster reside in a separate network segment: the *sandboxed segment*. Sandboxing means, in its most general sense, separating IT processes from each other. In terms of SAP Cloud Platform Integration, *sandboxed* means that, to a certain extent, each VM runs isolated from the others in its own sandbox or micro-network. In other words, each VM of the platform is shielded from other VMs in the same segment, as sandboxes can't interfere with or even "see" each other. This design makes sure that all components in the sandboxed segment are strictly isolated from each other.

We've restricted our statement above by saying *to a certain extent*: to be more precise, we have to limit and refine our consideration with regard to the fact that a TMN can interfere with those runtime nodes that are assigned to it. When a user requests monitoring data through the Web UI, the associated TMN needs to communicate with the runtime node, which is in charge of processing the message, to retrieve the data. However, runtime nodes themselves can't interfere with other runtime nodes, and nodes assigned to different tenants are isolated from each other.

10.1.3 Data Storage Security

As an integration platform, SAP Cloud Platform Integration acts like a *transit place* for data, as its task is essentially to receive, process, and forward messages. However, different kind of data associated with message processing can also be stored at various steps during message processing. In this section, we summarize by which measures *data at rest* is protected at a maximum level.

As explained in [Chapter 2, Section 2.1.2](#), the following types of data can be stored during runtime (compare also with Table 2.2):

- **Message content**

Integration developers can configure dedicated steps of an integration flow to store message content. The message can either be stored permanently (by default,

90 days) using the Message Store step, or temporarily (for a few seconds), to make it available for subsequent processing steps, using the Data Store step (or by using JMS queues).

- **Monitoring data**

The message processing log (MPL) records the actually executed processing steps for a message. It can be accessed using the monitoring application of the Web UI by users that have the appropriate permission.

The tenant isolation concept ensures that data belonging to different participants is strictly isolated from each other.

Message content stored in the database can be encrypted with an encryption key, which is generated automatically and is unique for each tenant (using Advanced Encryption Standard [AES] and a key length of 256 bits). The encryption key is stored in a different database than the encrypted data. To increase security, key rotation is supported (so that keys can be changed periodically or whenever they are compromised).

In addition, with regard to access permissions, data stored in different tenants is strictly isolated: an administrator on tenant A won't have permission to access data stored on tenant B. This rule also applies to SAP internal employees in charge of setting up, maintaining, and updating customer clusters. Furthermore, SAP will have no access to data stored in customer tenants.

In the case of support, customers can temporarily grant restricted permissions to dedicated SAP employees to allow them to execute tasks such as error analysis (typically assigning read permissions using authorization group `AuthGroup.ReadOnly`; [Section 10.3](#) for more details).

In any case, the principle of *least privilege* is always applied, which means that users are limited to the minimum set of privileges (permissions) required to perform a necessary task.

10.1.4 Data Protection and Privacy

Data protection is always related to legal requirements and privacy concerns. In this section, we first give a brief summary of where sensitivity of data is to be considered in an integration project and, secondly, provide information on specific measures taken to protect such data within SAP Cloud Platform Integration.

As a general rule, customer data processed by and stored within the SAP Cloud Platform Integration infrastructure is classified as *confidential*, in the sense that it requires (and receives) the highest level of protection.

Which Data Needs to Be Protected

Table 10.1 provides examples of which kind of sensitive data is to be considered at various steps in the SAP Cloud Platform Integration lifecycle.

Phase	Kind of Data
Message processing	Data contained in customer messages and processed on an SAP Cloud Platform Integration VM at runtime is usually business data that can contain personal information, such as names or address data. The measures to protect this data have been explained in <u>Section 10.1.3</u> .
Monitoring	The MPL records the executed processing steps for a message. Therefore, information about customer activities can be derived out of it (e.g., through the frequency of message processing). This data is only accessible per tenant (i.e., it's subject to tenant isolation) and for users with dedicated permissions.
Audit log	Certain events and system changes (e.g., access to message content or the deployment of integration artifacts) are logged during the operation of SAP Cloud Platform Integration. Audit logs are generated both for administrators at SAP that monitor the operations of a cluster and for tenant administrators. In both cases, the access to audit log data is protected by specific roles. For more information, see the following section.
Dialog user logs in to the SAP Cloud Platform Integration Web UI	When customers log in to their tenant through the Web UI, they register to the SAP ID Service. In this case, certain data is collected, and personal data (names and email addresses) also comes into play. SAP ID service is a special SAP Cloud Platform tenant (managed by SAP) that is used as a default identity provider (IdP) for SAP Cloud applications (see also <u>Section 10.3</u>).

Table 10.1 Examples of Customer Data Stored during the SAP Cloud Platform Integration Lifecycle

We'll now provide an overview of measures that are taken by SAP to protect this data.

Audit Log

To increase data protection, data accesses and other incidents are recorded in an audit log. Audit logs are generated, first, for administrators at SAP to enable them to monitor those incidents and prevent malicious usage of the same. These audit logs provide a chronological record of events such as the following:

- Data read access
- Security-critical incidents that might affect the confidentiality, integrity, or availability of the system (e.g., starting or stopping a VM, failed logins, or changes of critical system parameters)
- Configuration changes to the system (e.g., integration flow changes or content deployment tasks)

Logging such data enables SAP to perform regular audits to meet the requirements of regulatory compliance.

Audit logs are generated for each tenant, which means that log data related to different customers is separated. In addition, strict access control is imposed, and no log modifications by malicious users are possible. Logs are retained for 18 months and can be handed out to customers upon request. Components subject to audit logs are the VMs of a cluster and the load balancer.

Audit logs are, secondly, also generated for tenant administrators, and they record all security-relevant changes in their (the administrator's) tenant cluster. Such information is retained for 30 days and then deleted automatically. To display such audit logs, the user needs to have specific roles assigned. More information on how a tenant administrator can access these audit logs can be found in [Chapter 8, Section 8.5.1](#).

European General Data Protection Regulation

SAP Cloud Platform Integration has established the necessary processes and the infrastructure to comply with the new European *General Data Protection Regulation* (GDPR) that came into action on May 25, 2018.

European customers can opt in to having their data treated according to a European Union (EU) access policy. In such a case, customer data is processed and stored exclusively at a data center located in the European Union, which is subject to European data protection and privacy laws. Back up of data is also accomplished using a data center located in the European Union.

Personal data of such customers will only be accessible to members of a dedicated European SAP Cloud Platform Integration Operations team, and not to anyone outside Europe.

Data Protection in the EU

More information about data protection in the EU can be found here:

- <http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32016R0679>
- https://ec.europa.eu/info/law/law-topic/data-protection_en

10.1.5 Physical Data Security

Now that we've covered the security aspects inherently provided by the architecture, the network infrastructure, and data protection guidelines, let's take a closer look at *physical data security*. After this, we focus on how the processes related to an integration project are protected.

The VMs that process customer messages run on servers located in various regions worldwide. At the time of this book's release, SAP Cloud Platform Integration is exclusively running in the SAP Cloud Platform Neo environment. Data exchanged in the course of integration scenarios is technically stored and processed within an SAP infrastructure, namely, on servers located in SAP data centers. Customer data is also stored there for a certain time. SAP data centers are world-class and, as such, meet the highest security standards. To mention a few examples, SAP data centers rely on redundant power supplies, physical access is protected by biometric-access control mechanisms, and there is 24-hour surveillance. A security and facility support team is onsite 24/7, and locations are monitored by hundreds of surveillance cameras with digital recording. Buildings are protected against fire by ceilings, walls, and doors that provide 90 minutes of fire resistance. All of these measures are checked and audited on a regular basis.

Customers can assure themselves of the high security standards by requesting a guided tour for visitors. For more information, visit www.sapdatacenter.com.

10.2 Processes

In this section, we focus on various processes around the development, provisioning, and usage of SAP Cloud Platform Integration. We'll show that these processes fulfill the highest security standards.

First, SAP has certified that the development, maintenance, and operations of SAP Cloud Platform Integration comply with the requirements of standards such as SAP Cloud Platform ISO/IEC 27001, SAP Cloud Platform SOC 1 (ISAE3402), SAP Cloud Platform SOC 2, and SAP Cloud Platform: ISO/IEC 22301.

You can find the certificates at <https://www.sap.com/about/cloud-trust-center/cloud-certification-compliance/compliance-finder.html>. On this page, filter for the **Solution/Area SAP Cloud Platform Integration**.

This certification is under surveillance and renewed annually.

We'll now walk you through the security-relevant aspects of the processes that are in place.

10.2.1 Software Development Process

SAP Cloud Platform Integration software, as all SAP software, is developed in compliance with the SAP Security Development Lifecycle (SDLC), which helps build security into the software from the beginning and includes the following features:

- Test-driven development, including source code reviews, architecture audits, and security scans. In addition, quality gates have to be passed through on a monthly basis. This includes scanning the source code for security issues (checking the source code, identifying possible security issues and gaps, and helping developers fix any issues that may arise).
- Threat modeling techniques are selectively applied, which means that possible security gaps in individual components of the architecture are identified and modeled in advance. That way, possible vulnerabilities of the system can be anticipated and taken into consideration when designing the software.
- Open-source components being used are scanned for security vulnerabilities based on a risk assessment process.

The requirements imposed by the SDLC on the software development process are dynamically adapted and updated regularly based on publicly accessible sources that provide information on known software vulnerabilities (e.g., *Common Weakness Enumeration* or *Common Vulnerabilities and Exposures*).

10.2.2 Provisioning and Operating SAP Cloud Platform Integration Clusters by SAP

Let's now take a closer look at how the SAP Cloud Platform Integration platform is provided and administered by SAP. The provisioning of accounts and tenants for new customers and the setup and operation of the associated VMs is performed by the SAP Cloud Platform Integration Operations team.

SAP Cloud Platform Integration is managed and administered by software-as-a-service (SaaS) administrators from a dedicated team at SAP (these were indicated as the user icon with the title **SAP** in [Figure 10.2](#)). Members of this team are in charge of setting up and operating the individual tenant clusters of various customers. Tasks they are responsible for include starting and stopping cluster nodes, as well as performing monthly software updates.

Requests from the SAP-internal SaaS administrators associated with these tasks are again forwarded by the load balancer component. Based on the network design shown earlier in [Figure 10.2](#), the components of the customers are also maximally isolated from the component that internally manages them.

SAP takes certain measures to minimize the risk of malicious actions on the part of SAP-internal personnel who have access to the SAP Cloud Platform Integration platform.

The SAP Cloud Platform Integration Operations team is on duty around the clock. The team members act in the role of SaaS administrator. Their tasks also include the continuous operation of the customers' VMs and monthly software updates.

Although the team is in charge of administering and operating customer clusters, and of acting on alerts (in case of unexpected system behavior), the permissions granted to these persons are reduced to the bare minimum needed to perform operational tasks—the principle of *least privilege*. SAP Cloud Platform Integration Operations team members, for example, have no permissions to access the content of messages processed on a customer tenant. Only a defined group of people has the required access rights and permissions to start and stop VMs. Monthly audits of the process make sure that the access rights of the involved persons are constantly monitored, reviewed, and strictly controlled.

All system-related activities of the team members are logged and can be checked. If a customer requires support, selected development experts at SAP can be granted temporarily restricted access to a customer cluster to debug the problem.

10.2.3 Setting Up Secure Connections between the Tenant and Remote Systems

The prerequisite of reliably operating an integration scenario is to set up secure connections between the sender and receiver systems and the SAP Cloud Platform Integration tenant.

A core task is the implementation of the required technical trust relationships between the remote systems (sender and receiver) and the tenant. In most cases, this includes the generation of digital keys and the associated configuration of the keystores of the remote systems and the tenant. Usage of public key technology always requires the exchange of public keys between the administrators of the connected sender and receiver systems and the tenant administrator (as explained in [Section 10.4.1](#)).

Whereas the provisioning and operations of VMs are always kept in the hands of SAP, the responsibility for the tenant cluster lies with the customer (tenant administrator). The tenant administrator is responsible for setting up processes and communication channels that guarantee a reliable way of exchanging security-related material with the administrators of the associated systems.

In other words, the tenant administrator is mainly responsible for managing the required security material for the tenant. However, specific digital keys are still provided by SAP. As keys have a limited validity period, they need to be updated on a regular basis by SAP. Note, however, that each change or renewal of a key pair entails the update of the associated public keys implemented in the keystores of the connected remote systems. Therefore, the keystore management functions provided by SAP Cloud Platform Integration make sure that the tenant administrator still keeps control over the activation of updated SAP keys on the tenant (as explained in [Section 10.5](#)).

10.3 User Administration and Authorization

In the architecture overview of [Section 10.1.1](#), we showed where user management and authorization come into play in the lifecycle of a SAP Cloud Platform Integration project. In this section, you'll learn the basic concepts of user management and authorizations related to SAP Cloud Platform Integration. We'll show you how to manage authorizations for persons involved in an integration project.

10.3.1 Technical Aspects of User Management

Users and authorizations for integration project teams are managed separately for each customer account within SAP Cloud Platform. However, note that SAP Cloud Platform has no built-in user management component (as compared, e.g., to the User Management Engine that comes with the Java stack of SAP Process Orchestration). Instead, SAP Cloud Platform delegates authentication and user management to another system dedicated to managing information about user identities, also referred to as *IdP*.

By default, SAP Cloud Platform uses the SAP ID service, which is a tenant that exposes an SAP-operated IdP (see <https://cloudplatform.sap.com>). The connection and trust relationship of your subaccount with the SAP ID Service is preconfigured when you register to SAP Cloud Platform. The SAP ID Service also supports Single Sign-On (SSO), which allows users to log on once and then receive seamless access to all deployed applications.

When you initially access SAP Cloud Platform, you'll register at SAP ID Service. SAP ID Service manages users for other SAP websites, such as SAP Community. Therefore, when you have such a user (e.g., an S-user), you're already registered with SAP ID Service, so no further actions are required. You can also use a custom IdP (using an SAP Cloud Platform tenant with SAP Cloud Platform Identity Authentication service).

10.3.2 Personas, Roles, and Permissions

To manage authorizations for users involved in integration projects, SAP Cloud Platform provides predefined roles that allow you to give subaccount users permissions related to their tasks. According to the main tasks for integration projects, these roles are grouped in authorization groups shown in [Table 10.2](#).

Authorization Group	Description
AuthGroup.BusinessExpert	Enables a business expert to perform tasks such as monitoring integration flows and monitoring message content stored in temporary storages
AuthGroup.Administrator	Enables the tenant administrator to perform administrative tasks on the tenant cluster, for example, deploying security content and integration flows

Table 10.2 Authorization Groups for Integration Team Members

Authorization Group	Description
AuthGroup.Integration-Developer	Enables an integration developer to display, download, and deploy artifacts (e.g., integration flows)
AuthGroup.ReadOnly	Enables a user to display integration content and to monitor messages
AuthGroup.SystemDeveloper	Enables a system developer to perform tasks required for system support (e.g., restarting subsystems of the tenant cluster, and software development tasks on VMs of the tenant cluster)

Table 10.2 Authorization Groups for Integration Team Members (Cont.)

The tenant administrator (who is the first to access the tenant and perform user and authorization management tasks) assigns the relevant authorization groups to the users that are associated with people involved in integration projects and are in charge of specific tasks (e.g., developing integration content).

In addition to assigning authorization groups to users, you can also define permissions on a more detailed level by assigning individual elementary roles. For example, to display audit log entries (as shown in [Chapter 8, Section 8.5.1](#)), the following roles have to be assigned to the user: `IntegrationOperationServer.read` and `AuditLog.Read`, whereas to display the MPL (as explained in [Chapter 8, Section 8.5.1](#)), only the role `IntegrationOperationServer.read` has to be assigned to the user.

For a detailed documentation of the roles that are available for the different tasks associated with an integration project, go to the online documentation of SAP Cloud Platform Integration at https://help.sap.com/viewer/product/CLOUD_INTEGRATION/Cloud and search for **Tasks and Required Roles**.

So far, we've only considered users associated to persons involved in an integration project. However, we also need to define authorizations for certain technical users (i.e., users associated with components or systems that are to be connected to SAP Cloud Platform Integration). For these technical users, a specific role needs to be assigned: `ESBMessaging.send`. Note that this role needs to be assigned to the runtime node, not, like all other predefined roles, to the TMN. You can also define custom roles for this purpose as explained in [Section 10.3.3](#).

10.3.3 Managing Users and Authorizations for a SAP Cloud Platform Integration Subaccount

When you've requested a tenant, you'll receive an email from SAP that contains the details of your tenant, as well as how to access your subaccount (see [Chapter 2](#)). This mail contains all information you need to access your tenant cluster as well as SAP Cloud Platform cockpit. The first step after your tenant is provided by SAP is to go to SAP Cloud Platform cockpit and define the authorizations of all people in your organization that work in your integration project.

After you've logged in to SAP Cloud Platform cockpit, selected your region, your global account, and then your subaccount, and you'll get to the UI shown in [Figure 10.3](#).

The screenshot shows the SAP Cloud Platform Cockpit interface. The left sidebar has a dark blue background with white icons and text. It includes sections for Overview, Applications, Services, Solutions, Virtual Machines, SAP HANA / SAP ASE, Connectivity, Security, Repositories, Resource Consumption, and Members. The 'Overview' section is currently selected. The main content area has a light gray background. At the top, there's a breadcrumb navigation: Home [Europe (Rot)] / Europe (Rot) / D0349 Integration Account / d0262. Below the breadcrumb, the title 'SAP Cloud Platform Cockpit' is displayed. The main content is divided into several sections: 'System Status' which includes Java, HTML5, and Database Systems status; 'Favorite Applications' which indicates no favorite applications have been defined; and 'Subaccount Information' which shows details for the selected subaccount 'd0262'. The subaccount information includes fields for Display Name (d0262), Name (a2ee4d384), Members (2), Subscriptions (2 Subscribed), and Services (21 available).

Figure 10.3 SAP Cloud Platform Cockpit When Subaccount Is Selected

As the tenant administrator, you can now start managing users and authorizations for the account. This process comprises two steps:

1. Define the members of the account. These are all people who are supposed to work as tenant administrators.
2. Assign authorization groups to the users of all (additional) persons that are supposed to perform tasks related to the tenant (e.g., integration developers).

Adding Members to the Account

In the SAP Cloud Platform cockpit navigation area of your subaccount ([Figure 10.3](#)), click **Members**, and add the users who will have the role of tenant administrator.

1. On the **Add Members** page, enter the user ID, and select the role to be assigned to the user. We recommend selecting the roles **Administrator (predefined role)** and **Developer (predefined role)** ([Figure 10.4](#)).

Figure 10.4 Adding Members to a Subaccount

The **Administrator** role gives the added user ID the full permissions of a tenant administrator. It allows you to manage subaccount members, as well to manage authorizations, OAuth settings, and other tasks.

The **Developer** role provides permissions to perform typical developer tasks such as starting or stopping applications.

Don't use any of the other roles.

These roles are predefined on SAP Cloud Platform and define authorizations for subaccount users. They aren't related to the authorization groups and roles that are required for specific tasks in an integration project ([Table 10.2](#)). Those are assigned in an additional step, which we'll explain next.

2. Click **Add Members**.

Defining Authorizations for Integration Team Members

After you've defined the subaccount members (tenant administrators), you'll assign authorization groups or roles to users associated with the people who are supposed to work in an integration project (e.g., for integration developers). Follow these steps:

1. In the navigation area (refer to [Figure 10.3](#)), choose **Security • Authorizations**.

2. On the **Authorization Management** page, enter the **User** for whom you want to define authorizations, and choose **Assign** (Figure 10.5).

The screenshot shows the 'Authorization Management' interface. At the top, there are tabs for 'Users', 'Groups', and 'Token'. Below the tabs, there's a section titled 'Assign Web Roles and Groups to Individual Users' with a user icon and a lock icon. A search bar for 'User: *' and a 'Show Assignments' button are present. Under 'Roles', the 'Assign' button is highlighted with a black box. To the right, under 'Groups', the 'Assign' button is also highlighted. Below these sections are filters for 'Subaccount', 'Application', 'Role', and 'Actions', along with a 'Group' filter.

Figure 10.5 Authorization Management Page

3. On the **Assign roles to user <user name>** screen, choose the authorization group or role that is to be assigned in the **Role** field (Figure 10.6).

The screenshot shows the 'Assign roles to user' dialog. It has fields for 'Subaccount' (set to 'abcd'), 'Application' (set to '<tenant name>tmn'), and 'Role' (set to 'AuthGroup.IntegrationDeve'). A callout box points from the 'Role' dropdown to the label 'Authorization group or role'.

Figure 10.6 Assigning Authorization Groups or Roles to a User

Note that when assigning authorization groups or roles to dialog users associated with an integration project, you select a TMN (ending with “tmn”) for the **Application** field (as dialog users access the SAP Cloud Platform Integration platform through the TMN; refer to [Figure 10.1](#)).

4. Click **Save**.
5. Repeat these steps for all users and authorization groups to be assigned.

You also can create user groups for all users who should get identical authorizations (see the right side of [Figure 10.5](#)).

Creating and Assigning Roles for Technical Users to Process Messages

To enable users associated with technical systems to process messages on the tenant, you need to define the relevant permissions on the level of the runtime nodes.

SAP provides a predefined role for that purpose, which you already learned about in [Chapter 2, Section 2.3.3](#) (Figure 2.16), and at several other places within this book: `ESB-Messaging.send`. This role is defined on the runtime node level. To authorize a sending application to process messages on the tenant (more specifically, on the runtime node), you need to assign this role to the technical user associated with the sender. Use the same procedure as described previously for managing user-to-role assignments for dialog users, with the exception that as **Application**, you select a runtime node (ending with “iflmap”; see [Figure 10.7](#)). Remember, in contrast to dialog users, users associated with technical systems access the SAP Cloud Platform Integration platform through the runtime node (as shown in [Figure 10.7](#)).



Figure 10.7 Assigning a Role to a Technical User for the Sender System

However, you can also define custom roles, for example, to specify permissions on a more fine-granular level. We'll now show how to define a new user-to-role assignment for the runtime node (to define permissions to process messages on the runtime node). In [Section 10.4.4](#), when discussing the inbound connection, we already mentioned the option to use custom roles to define permissions for inbound calls. Follow these steps:

1. In SAP Cloud Platform cockpit, select your subaccount as shown earlier.
2. In the left navigation area, click **Subscriptions**.
3. Under **Application**, click the one ending with **iflmap** (which identifies your runtime node).
4. In the left navigation area, click **Roles**.
5. Choose **New Role**, and enter the name of your role (e.g., “MyRole”), as shown in [Figure 10.8](#).
6. Choose **Assign**, and enter the **User ID** (of the user which is to be associated with the sending application), as shown in [Figure 10.9](#).

The screenshot shows the SAP Cloud Platform Cockpit interface. The left sidebar has 'Overview', 'Destinations', and 'Roles' selected. The main content area is titled 'Roles (All: 1)'. It shows a table with one row for 'MyRole', which is categorized as 'Custom'. Other roles like 'ESBMessaging.send' are listed as 'Predefined'. A 'New Role' button is visible at the top of the table.

Name	Type	Actions
ESBMessaging.send	Predefined	
MyRole	Custom	

Figure 10.8 Defining a Custom Role

This screenshot continues from Figure 10.8. It shows the 'Assign' tab for the 'MyRole' custom role. Under 'Individual Users', the user 'd030770' is listed with an 'Unassign' button. There is also a 'Groups' section with an 'Assign' button and a 'Group:' dropdown.

User ID	Actions
d030770	

Figure 10.9 Assigning a Custom Role to a User

You've now learned how to manage users and authorizations, including how to define the permissions initially for your integration team members.

Now, from a security perspective, you can start working with SAP Cloud Platform Integration. In the next sections, we show all options of data and data flow security and how your integration team members can use the tools of SAP Cloud Platform Integration to set up scenarios with a certain security level.

10.4 Data and Data Flow Security

In [Section 10.1](#) and [Section 10.2](#), you learned about the security measures that are already in place when you begin working with SAP Cloud Platform Integration—given by the architecture, the network design, the processes associated with SAP Cloud Platform Integration, and other aspects. [Section 10.3](#) introduced you to the topic of user management.

In this section, we'll discuss the options that you, as the customer, have to maximize the security of the solution by configuring the way data is exchanged between the components in the customer landscape and SAP Cloud Platform Integration. In other words, we'll focus on how *data in transit* (i.e., on its way between the involved parties) can be protected.

SAP Cloud Platform Integration provides you with a variety of options to protect data in transit on the following levels:

- **On the transport protocol level**

Establish a secure communication channel between remote systems and the involved SAP Cloud Platform Integration components.

- **On the message level**

Further protect the exchanged messages via digital encryption and signing.

This topic, which deals with concepts of secure communication and protecting digital data, is known as *cryptography*. In [Section 10.4.1](#), we'll provide a brief overview of the basic terms and concepts of cryptography. Note that this is general knowledge, not exclusively related to SAP Cloud Platform Integration. Those who are familiar with the topic of cryptography can, therefore, skip [Section 10.4.1](#) and continue with [Section 10.4.2](#).

In [Section 10.4.2](#) and [Section 10.4.3](#), respectively, we provide an overview of the transport-level security options and of the authorization and authentication options supported by SAP Cloud Platform Integration. In [Section 10.4.4](#) and [Section 10.4.5](#), two tutorials will follow that show you how to establish secure connections between your tenant and remote systems and how to develop a simple integration flow with a specific authentication option, OAuth. In [Section 10.4.6](#), we discuss the message-level security options provided by SAP Cloud Platform Integration. In [Section 10.4.7](#), finally, we explain how message-level security can be implemented using SAP Cloud Platform Integration (including a step-by-step tutorial that you can easily reproduce).

10.4.1 Basic Cryptography in a Nutshell

Cryptography, in the broadest sense, deals with methods and techniques that help protect data against unauthorized access. One basic measure of protecting data against unauthorized access is to encrypt it. Encrypting means to transform a *clear text* (which is readable by everyone) into a *secret text*, prior to sending it to a communication partner. This transformation is done on the sender side by a mathematical operation (based on an *encryption key*). The dedicated receiver needs to know the inverse operation (the *decryption key*) to transform the secret text back into the clear text.

Another measure to protect a message on its way between a sender and receiver is to apply a *digital signature* so that the receiver can be sure that the message has been sent by the trusted sender (this is also known as data integrity).

The simplest and most obvious approach (which is also referred to as *symmetric key technology*) is that sender and receiver use the same key, and a critical requirement to ensure a seamless and protected communication of sender and receiver is that the key is securely exchanged between both parties prior to the actual data exchange. It's obvious that this requirement can't be met in the digital age, as the number of potential communication partners and communication paths for a given channel (e.g., email) is on a large scale and dynamically changes in the short term.

To overcome this challenge, *asymmetric* (or *public*) *key technologies* have been developed, which always require two different key types: a public key and a private (or secret) key. Both key types are generated together and are related to each other based on a mathematical operation such that the public key can be easily calculated from the private key. However, it's impossible to perform the reverse operation and derive the private key from the public key (at least, when considering computing capacities available today). Mathematically, this approach is based on one-way functions, which we won't explain further here. For more information, refer to *Cryptography and Public Key Infrastructure on the Internet* (Wiley, 2003).

For the encryption use case, the asymmetric approach works in the following way: the intended receiver of a message generates a public-private key pair and shares the public key with the sender. The sender then uses the public key to encrypt the message. The receiver then uses its private key (which is always kept with the receiver and never shared with any other party) to decrypt the message.

For digital signatures, the inverse pattern is applied. The sender signs a message using its private key (which is always kept with the sender) and the receiver (and many other potential receivers) can verify the signature by using the public key,

which they have been provided with by the sender. We show how this works in detail in [Section 10.4.6](#).

The private key must never be shared with anyone else obviously, or it would allow others to sign messages in your name or to decrypt content that is destined solely for you.

Because of the inherent mathematical concepts (one-way functions), public keys, on the other hand, can potentially be shared through unsecure channels (e.g., email). Any malicious party who receives the public key by mistake has no opportunity to decrypt or sign any message with it. Note, however, that certain additional measures should be undertaken to enable the receiver of a public key to validate the authenticity of the same, as outlined later.

Hybrid Approaches

Symmetric methods typically use simple bit operations to transform a clear text bit sequence into a secret text bit sequence. In asymmetric methods, however, sophisticated mathematical operations come into play (which use modulo mathematics). Because of this, asymmetric methods are computationally intensive and, therefore, not well-suited to operate on larger bit sequences that come along with large business messages exchanged in integration scenarios.

To overcome this issue, it's more feasible to apply hybrid approaches, which combine asymmetric and symmetric methods. In encryption scenarios, the most common pattern is as follows (see [Figure 10.10](#)).

The sender encrypts the (potentially large-volume) content of a message using a *symmetric* encryption key. Thereafter, the sender encrypts the symmetric encryption key with a public key and sends the encrypted symmetric key (along with the encrypted message content) to the receiver. Before the message exchange between sender and receiver has been initiated, the public key has been generated by the receiver as part of an *asymmetric* key pair and shared with the sender without risk. With the associated private key, the receiver, as soon as he has received the encrypted symmetric key (and the encrypted message content), decrypts it and, finally, uses the revealed secret key to decrypt the message content.

This way, we avoid applying asymmetric key operations to the whole message content.

Hybrid approaches are applied in the transport-level security option TLS, described in [Section 10.4.2](#), as well as in the message-level security options described in [Section 10.4.6](#). In these sections, we'll show in more detail how these approaches work.

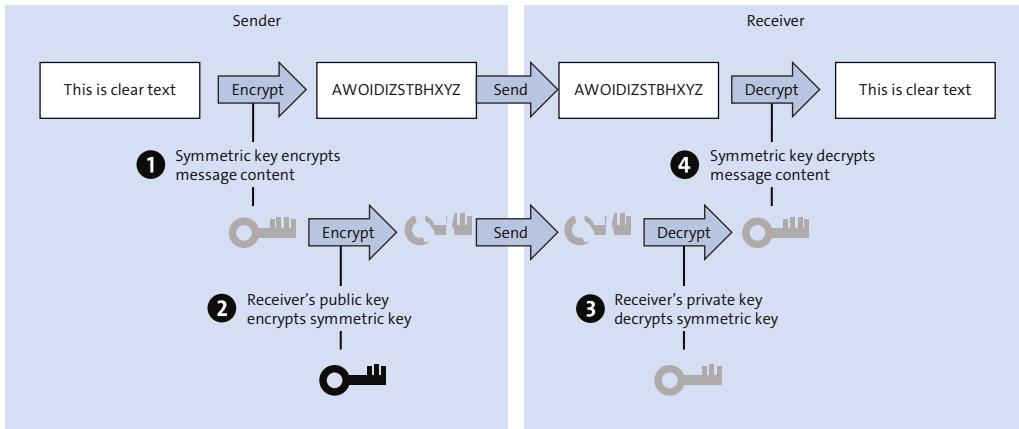


Figure 10.10 Hybrid Usage of Symmetric and Asymmetric Key Technology When Encrypting and Decrypting Message Content

Keystores

To introduce another important term, note that keys (public and private keys) are stored in *keystores* owned by the involved parties. How a keystore is implemented and its characteristics depend on the type of system that implements the services of the sender and receiver.

Figure 10.11 illustrates the setup of components for two parties exchanging encrypted messages based on public key technology. In the shown general setup, sender and receiver separately generate their own public-private key pair and import it into their keystore. In a subsequent step, both participants share the corresponding *public* key with the communication partner. Both partners also import the corresponding foreign public key into their keystores.

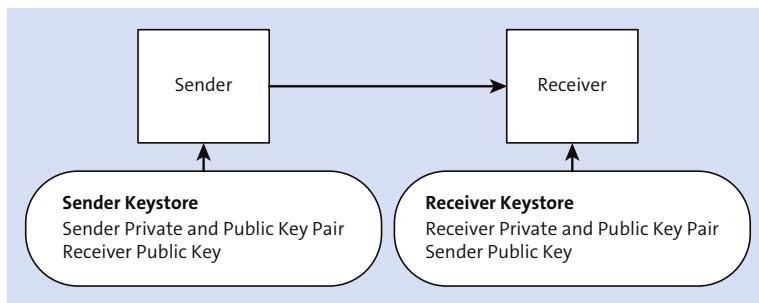


Figure 10.11 Keystores Containing the Required Key Material

Although there is no common term for the key storage of all possible kinds of systems, simplicity, we'll always talk about *keystores* throughout this book. We'll also shorten the term *public and private key pair* to *private key pair* or *key pair*.

Keystore Types for SAP Cloud Platform Integration

In scenarios described in this book, the tenant is always one communication partner. Therefore, a keystore needs to be deployed on the tenant that contains the required key material. The characteristics of the keystore also depend on the chosen security standard (as will be shown in [Section 10.4.2](#) and [Section 10.4.6](#)).

To give an example, when you use X.509 certificates, you utilize a Java keystore, whereas when you use OpenPGP, you'll need two different "keystore" types: a PGP Public Keyring and a PGP Secret Keyring ([Section 10.4.6](#) and [Section 10.4.7](#)).

To give an example for a remote system that can be connected to a tenant: if it's an SAP system based on Application Server ABAP (AS ABAP), the required keys are maintained with the Trust Manager in the Personal Security Environment, which takes over the role as the keystore.

Although we've stated that public keys can potentially be exchanged on unsecure channels (whereas private keys must never be shared with another party), a malicious party can nevertheless misuse this fact and send a public key to another party, pretending to be the owner of the public key. So, the question is, how can the authenticity of a public key be guaranteed to further increase security?

To answer this, we'll briefly discuss *digital certificates*. As certificates are of significant importance for the whole topic of security, we put the definition of this term and concept in an info box.

Digital Certificate

A digital certificate is a public key that is signed by a trusted authority (usually referred to as a certification authority (CA)). This way, the identity and trustworthiness of the public key owner can be confirmed. Taken in short, a certificate couples an identity with a public key.

There are many options to build trust based on certificates. One example is the X.509 standard, which comes into play, for example, when two communication partners protect their communication channel using TLS ([Section 10.4.2](#)). X.509 supports the usage of PKIX-certificates (Public Key Infrastructure X.509) that allow you to build up certificate chains. These are hierarchical trust models, which include many CAs on

different levels, where the CA on the higher level signs the certificate of the correspondingly lower level, and so forth. The CA on the top level is referred to as the root CA. This model is called a certificate chain.

An example of a CA is GeoTrust (www.geotrust.com). One root certificate issued by GeoTrust and supported by the load balancer is *GeoTrust Global CA*.

Another, alternative trust model is the *Web of Trust*. In such a model, communication partners mutually confirm the authenticity of each other's public keys, building a network of partners rather than a hierarchical structure. The Web of Trust model can be used in conjunction with the security standard OpenPGP. [Figure 10.12](#) shows the difference between the two trust models.

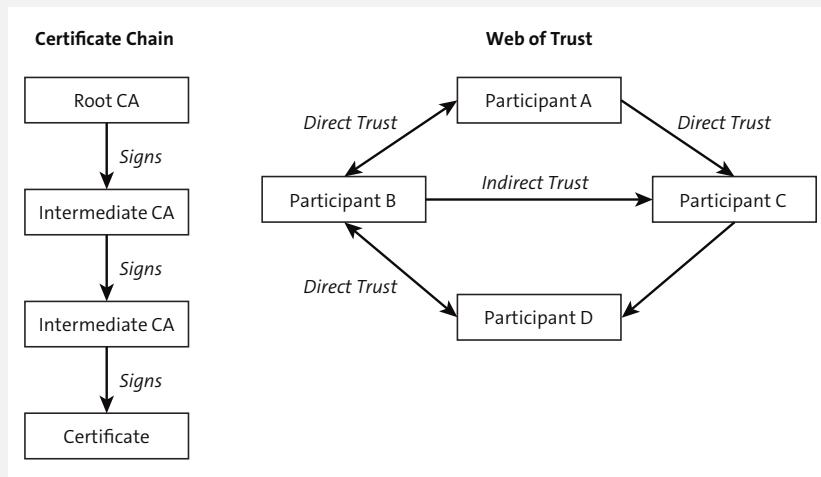


Figure 10.12 Certificate Chain (Left) Compared to a Web of Trust (Right)

X.509 Certificates

As this certificate type is the most commonly used for transport-level security, we would like to discuss a few basic concepts and terms. X.509 certificates allow you to implement trust models based on certificate chains. A set of elements is required to specify an X.509 certificate, although we won't go into detail about that in this book. We only want to point out the **Issuer** and the **Subject** field, which are required to understand the following:

■ **Issuer**

Specifies the CA (that issued and signed the certificate).

■ **Subject**

Specifies the entity that is associated with the public key of the certificate. If the certificate is used to authenticate a client calling a server, the subject usually identifies the client.

Both entries, **Issuer** and **Subject**, are uniquely defined by a distinguished name (DN), which is composed of a set of attributes such as company name, country identification, and so on. The format of DNs is defined by the specification RFC 5280 (see <https://tools.ietf.org/html/rfc5280>). That way, DNs are guaranteed to be unique, and using them ensures that, first, the certificate issuer can uniquely be identified (to make sure the certificate isn't tampered with information from a malicious party) and, second, the entity associated with the certificate certainly can be considered to be the one it's intended to be.

Certificates and their management are discussed in more detail in [Section 10.5](#).

You're now equipped with the basic concepts and terms of cryptography. In [Section 10.4.2](#) and [Section 10.4.6](#), we explain in detail how to apply cryptographic concepts to set up secure message exchange scenarios based on certain security standards.

10.4.2 Transport-Level Security Options

SAP Cloud Platform Integration provides various connectivity options—methods of connecting remote systems with different technical characteristics to SAP Cloud Platform Integration. The available adapters imply one of the transport protocols listed in [Table 10.3](#), each with different options to secure the communication channel.

Transport Protocol	Description
Hyper Text Transfer Protocol (HTTP) over Transport Layer Security (TLS), also referred to as HTTPS	<p>TLS is a protocol for secure communication over a computer network that is widely used on the Internet. As TLS is the enhancement of SSL. Note that these terms are often used synonymously.</p> <p>The protocol works with the following adapter types: SOAP, IDoc, SAP SuccessFactors, and HTTP.</p>

Table 10.3 Transport Protocols and Associated Security Options

Transport Protocol	Description
SSH File Transfer Protocol: Secure File Transfer Protocol (SFTP)	This protocol has been developed for the Secure Shell (SSH) and allows secure transfer of files. SSH is a network protocol that allows you to set up a secure connection to a remote computer. This protocol works with the SFTP adapter.
Simple Mail Transfer Protocol Secure (SMTPS)	Simple Mail Transfer Protocol enables a computer to exchange emails with a mail server. With SMTPS, SMTP connections can be secured by SSL or TLS. The protocol works with the mail receiver adapter.
Post Office Protocol version 3 over TLS/SSL (POP3S)	This protocol enables email clients to retrieve emails from an email server using the Internet Protocol (IP). POP3S works with the mail sender adapter.
Internet Message Access Protocol over TLS/SSL (IMAPS)	This protocol enables email clients to retrieve emails from an email server using a TCP/IP connection. IMAPS works with the mail sender adapter.

Table 10.3 Transport Protocols and Associated Security Options (Cont.)

As TLS is by far the most commonly used transport-level security option, we'll add a few remarks on this protocol.

Transport Layer Security

TLS uses a hybrid approach of asymmetric and symmetric key technology: the asymmetrical approach is used to encrypt a symmetric session key at the beginning of the connection setup, while the latter is then used to actually encrypt and decrypt the data as long as the TLS connection (session) is active.

TLS uses a hierarchical trust model (based on X.509 certificates). During the connection setup, certificates between the client and server are exchanged, and the authenticity of these certificates is validated based on the provided signatures by certification authorities.

With TLS, SAP Cloud Platform Integration offers different options for how a client authenticates itself against the server. We'll discuss the authentication options in [Section 10.4.3](#).

10.4.3 Authentication and Authorization

Next to measures to protect the exchanged messages by digital encryption and signature, other important aspects that impact the security of a system of components that communicate with each other are authentication and authorization.

Before presenting the options offered by SAP Cloud Platform Integration, we'll briefly clarify and distinguish the terms *authentication* and *authorization*.

Authentication and Authorization

Authentication verifies the identity of something (or someone) in a communication workflow. It checks, for example, if the person associated with a user (that connects to a server) is who he claims to be.

In information technology, authentication workflows typically relate to scenarios where a client (to be authenticated) requests access to some kind of protected resource hosted on a server. In the context of SAP Cloud Platform Integration, the term *protected resource* covers not only data but also message processing capabilities (e.g., implemented on a VM of a tenant cluster).

After authentication, an *authorization* check verifies what the authenticated entity is allowed to do in the connected server system.

In many cases, in an authorization check, user-to-role assignments of the (authenticated) user are investigated in the server system. For inbound communication (where SAP Cloud Platform Integration acts as server), the user-to-role assignments are defined by the tenant administrator (as shown earlier in [Section 10.3](#)).

Basic Authentication

This is the simplest option, where authentication is based on user credentials (username and password). When you configure basic authentication for inbound communication (where a client sends messages to SAP Cloud Platform Integration), the credentials of the technical user associated with the client are forwarded to SAP Cloud Platform Integration in the message header (through a secure channel, e.g., using HTTPS). The identity of the client is then verified based on the credentials stored at SAP.

SAP doesn't recommend using basic authentication for productive scenarios, as it's less secure than using client certificates.

Client Certificate Authentication

Using this option, the authentication step is accomplished based on digital certificates. Instead of credentials, the client forwards a digital certificate to the server. The authentication of the client is then done by checking the certificate.

Client certificate authentication is always the option of choice for productive scenarios, as certificates guarantee a higher level of security (they rely on a trust relationship, which is difficult to break, as shown in [Section 10.4.1](#)).

OAuth

Until now, we've only talked about authentication workflows that include two parties: a client (requesting access to a protected resource) and a server (that hosts the resource). In such scenarios, the protected resource is owned by either the client or the server.

We'll now talk about a different, more sophisticated authentication pattern, where the owner of the protected resource isn't necessarily identical to either client or server. Due to this, three parties (or roles) come into play:

- A user who owns the protected resources
- A client, which is typically an application
- A server (or resource server), which is typically a service provider that hosts the protected resources

That is what OAuth is about. It enables the user (as owner of the *protected resource*), to grant the *client* access to the protected resource (hosted by the *resource server*). The access granted to the client is typically *restricted*—no full access rights will be given. While doing that, you, the resource owner, can keep your credentials private. [Figure 10.13](#) shows the basic, simplified setup of OAuth.

As an example, you can think of Twitter as a resource server and a Twitter user as the owner of his Twitter stream (which is the resource). We'll show in [Section 10.4.5](#) how SAP Cloud Platform Integration (as the client application) can be granted access to Twitter content on behalf of a Twitter user. For that purpose, the corresponding integration flow uses a Twitter receiver adapter.

In OAuth 2.0, an additional role is added to the picture—the *authorization server*—which is the component that is finally granting access to the protected resource (and that is issuing the required access tokens, see more below). So far and in [Figure 10.13](#),

the authorization server wasn't identified. For simplicity, we assumed that authorization server and resource server (that hosts the resource) were identical, which is also a possible scenario.

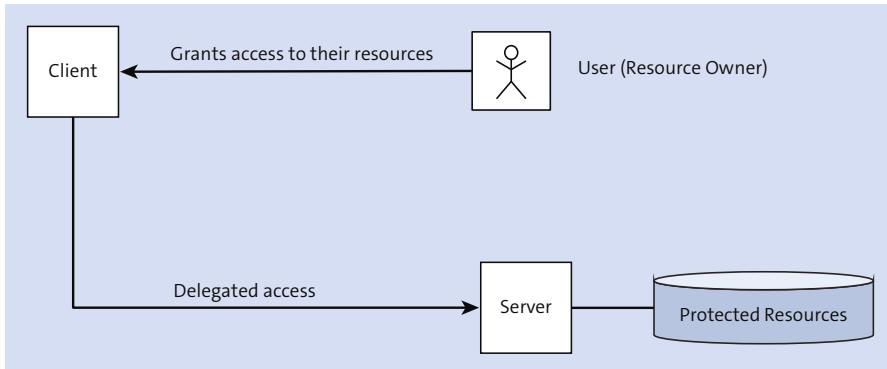


Figure 10.13 OAuth Scenario: Resource Owner Delegates Access to Protected Resources (Hosted on a Server) to a Client

To enable client access to the protected resource, the owner doesn't need to share his own credentials with the client. However, OAuth comes with a more complex setup of credentials than that used in the other authentication scenarios discussed earlier. In addition to the resource owner's own credentials (necessary to access his own resource), two additional kinds of credentials come into play. [Table 10.4](#) provides a summary of the credentials.

Credential Type	Description
Resource owner's credentials	Enable the resource owner to log on to the resource server to access and manage his protected resources.
Client credentials	<p>Identify the client (at the resource server's side). Client credentials are composed of the <i>consumer key</i> and the <i>consumer secret</i>.</p> <p>In OAuth 2.0 terminology, client credentials are a specific implementation of an <i>authorization grant</i>. An authorization grant is a credential that is used by the client to obtain an access token (see the next table entry).</p>

Table 10.4 Credential Types Used with OAuth

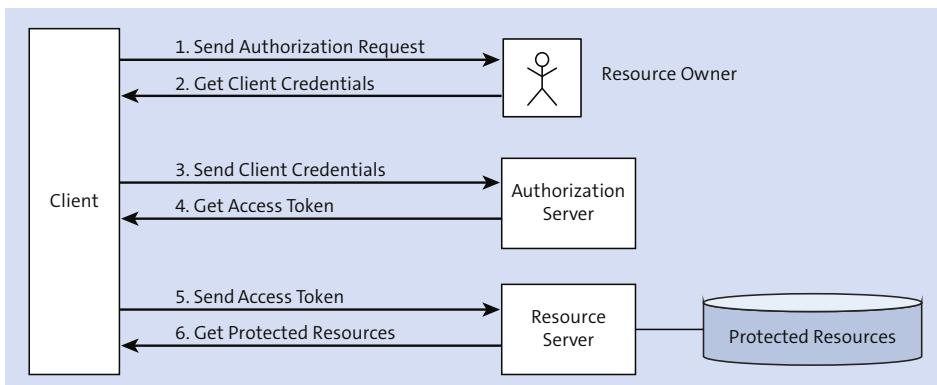
Credential Type	Description
Token credentials	<p>Authorize the client (on behalf of the resource owner) to access the resource.</p> <p>The token credentials identify the resource owner at the server's side. They can be revoked by the resource owner at any time.</p> <p>Token credentials are composed of <i>access token</i> and <i>access token secret</i>.</p>

Table 10.4 Credential Types Used with OAuth (Cont.)

In the sets of credentials summarized in [Table 10.4](#), the secret is a piece of information exclusively shared between the client and server. Using the two parts of a credential pair in combination helps protect those credentials from being compromised.

A characteristic of the token credentials (as compared to the client credentials) is that they can be revoked by the resource owner at any time. This allows you to adapt authorization workflows flexibly to changing requirements without the need to reconfigure the set of client credentials or to renew the resource owner's own credentials.

With these terms in mind, a typical OAuth authentication workflow comprises the following steps, now considering that resource server and authentication server are separate components. [Figure 10.14](#) is geared to the general description of the OAuth 2.0 workflow in the OAuth 2.0 specification that you can find at <https://tools.ietf.org/html/rfc6749>.

**Figure 10.14** OAuth 2.0 Authentication Workflow (with Client Credentials)

The workflow comprises the following sequence of steps:

1. The client requests authorization from the resource owner to access his protected resource.
2. The resource owner provides the client with client credentials (as a specific type of authorization grant as indicated in [Table 10.4](#)).
3. The client connects to the authorization server and presents the client credentials.
4. The authorization server authenticates the client, validates the client credentials, and, if they are valid, provides the client with an access token.
5. Authenticating itself against the resource server with the access token, the client requests the protected resource from the resource server.
6. The resource server validates the access token and, if it's valid, grants the client access to the protected resource.

An important aspect of this workflow is that the resource owner at no point in time has to share his credentials with anyone.

OAuth Terminology

OAuth is available in two versions:

- OAuth 1.0 (<https://tools.ietf.org/html/rfc5849>)
- OAuth 2.0 (<https://tools.ietf.org/html/rfc6749>)

Version 1.0 uses slightly different terminology for the credentials than version 2.0 does: *consumer key* and *consumer secret* (in version 1.0) are summarized as *client credentials* in version 2.0, whereas *access token* and *access token secret* (in version 1.0) are referred to as *token credentials* in version 2.0. In other words, the terms *consumer* and *client* are used synonymously.

Note that the social media adapters of SAP Cloud Platform Integration (Twitter and Facebook adapters), which we'll discuss in [Section 10.4.5](#), use the terms associated with version 1.0. To make it easy for you to understand how these adapters work, we mentioned both versions of the terms and set them into the proper context in [Table 10.4](#).

This is the briefly explained OAuth authentication workflow. In [Section 10.4.5](#), we'll show you how OAuth works together with the SAP Cloud Platform Integration Twitter adapter.

Principal Propagation

This authentication option is based on a setup where the identity of a user is transferred along all relevant communication paths of an integration scenario—from the sender to the receiver.

We show you now which authentication options are supported for HTTPS-based communication both for inbound and outbound communication. In particular, we also explain which combinations of authentication and authorization are supported by the SAP Cloud Platform Integration system when acting as server (inbound communication).

Inbound Authentication and Authorization (for HTTPS Communication)

When configuring inbound HTTPS communication (where SAP Cloud Platform Integration is to receive messages from a sender system), you'll notice that certain sender adapters that support this protocol provide the option to choose an **Authorization** option (rather than an **Authentication** option). We're talking here, namely, about the SOAP, IDoc, and HTTPS adapters. You can check this by creating, for example, a sender SOAP 1.x channel as shown in our first tutorial in this book in [Chapter 2, Section 2.3.4](#) (see Figure 2.27). For your convenience, we show the related UI property in [Figure 10.15](#). Two authorization options are available: **User Role** and **Client Certificate**.



Figure 10.15 Authorization Options Offered for Certain Sender Adapters That Support HTTPS

As we're here talking about SAP Cloud Platform Integration as the server that is requested by a sender (client) system, you (as integration developer who defines an integration scenario) can specify how the user associated with the sender is authorized to “do things” on the SAP Cloud Platform Integration platform. The possible options depend on the technical capabilities of the SAP Cloud Platform Integration platform.

With the selection of an *authorization* option for inbound communication, you determine already which *authentication* option can be used along with the selected authorization option.

Table 10.5 points out the difference between the authorization options.

Authorization Option	Description
User Role	<p>For the user associated with the sender system, the authorizations are checked based on user-to-role assignments defined for the tenant using SAP Cloud Platform cockpit.</p> <p>SAP recommends using this authorization option together with a <i>certificate-to-user mapping</i>. In this case, you specify the client certificates in a Certificate-to-User Mappings artifact (which is deployed on the tenant). In this artifact, you assign a user to the certificate.</p> <p>When the sender calls SAP Cloud Platform Integration, it authenticates itself against SAP Cloud Platform Integration based on the specified client certificate. In a subsequent step, authorization is checked based on user-to-role assignments defined for the user, which is derived from the certificate-to-user mapping.</p> <p>Already in the first tutorial (Chapter 2, Section 2.3), you learned about the role ESBMessaging.send, which defines permissions for a sender system to process messages on the tenant.</p> <p>We show how to set up such a scenario in Section 10.4.4. For certain reasons that are explained in Section 10.4.4, this is the recommended option.</p>
Client Certificate	<p>Using this option, you specify the required certificate in the integration flow (in particular, in the sender adapter).</p> <p>At runtime, the sender authenticates itself against SAP Cloud Platform Integration based on a client certificate.</p> <p>In a subsequent step, the permission of the sender to execute the integration flow is checked by evaluating the DN of the sender's client certificate.</p> <p>Note that SAP recommends that you don't use this option any longer because of certain disadvantages (e.g., that changing certificates would always mean redeployment and, consequently, downtimes of the integration flow).</p>

Table 10.5 Authorization Options for Inbound HTTPS Communication

When you select **Client Certificate** as the **Authorization** option, you inherently determine as *authentication* method the option *client certificate*, whereas the alternative **Authorization** option, **User Role**, can be combined with different authentication options. According to [Table 10.5](#), the recommended way to go is combining this authorization option with the usage of a certificate-to-user mapping, which comes along with *client certificate* authentication.

However, note that with **User Role** authorization, the following other *authentication* options can be used for inbound communication:

- Basic authentication, which we don't recommend to use for productive scenarios
- OAuth, which allows the configuration of specific scenarios such as principal propagation

To find out more about the supported options, check out the documentation of SAP Cloud Platform Integration at https://help.sap.com/viewer/product/CLOUD_INTEGRATION/Cloud and search for **Connecting a Customer System to Cloud Integration**.

Outbound Authentication (for HTTPS Communication)

The other way around—when talking about outbound communication (where SAP Cloud Platform Integration acts as client)—SAP Cloud Platform Integration can't offer any choices about how the user associated with the outbound request is authorized to execute certain actions in the receiver system.

Therefore, as integration developer, you can't specify any authorization options. This is intuitively plausible because how permissions of a calling entity are checked can only be defined by the technical capabilities of the server (in the outbound communication case, this is the receiver system). Because SAP Cloud Platform Integration (as client in this case) can't impose which technical capabilities are offered by the receiver system, it would not make any sense that the SAP Cloud Platform Integration platform allows you to specify any authorization option in a receiver adapter.

However, in a receiver adapter, you can specify the **Authentication** option supported by the client (SAP Cloud Platform Integration, in this case). You can briefly verify this by creating a receiver channel that supports HTTP communication (e.g., a receiver SOAP adapter; see [Figure 10.16](#)).

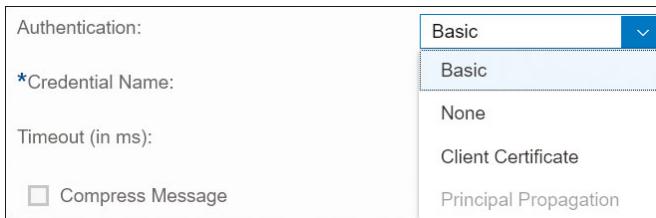


Figure 10.16 Authentication Options Offered for Receiver Adapters That Support HTTPS

Specifying an **Authentication** option is reasonable because SAP Cloud Platform Integration can provide the required artifacts for each authentication option.

Table 10.6 summarizes the different options and provides information on the related integration artifacts that are to be considered when configuring such a communication option.

Authentication Option	Description
Basic	<p>Authentication of SAP Cloud Platform Integration against a receiver system is based on user credentials (username and password).</p> <p>When you configure basic authentication for outbound communication, you need to complement the related receiver adapter setting by defining a security artifact that contains the credentials (a User Credentials artifact as shown, for example, in Chapter 2, Section 2.3.4).</p>
Client Certificate	<p>Authentication of SAP Cloud Platform Integration against a receiver system is based on a client certificate.</p> <p>A client certificate (including public and private key) and receiver server root certificate, which is accepted by the receiver, need to be part of the Keystore deployed on the tenant. In the receiver adapter settings of the integration flow, the private key alias of the certificate can be specified to indicate a specific key pair to be used for this step. If you don't specify a private key alias, any fitting key in the keystore is used. For a whole setup, Section 10.4.4.</p>

Table 10.6 Outbound Authentication Options (for HTTPS-Based Communication)

Authentication Option	Description
Principal Propagation	<p>In this case, the tenant authenticates itself against the receiver system by forwarding the identity (principal) of the user (associated with the inbound request) to the SAP Cloud Platform Connectivity service and from there to the receiver system (which can be, e.g., an on-premise SAP system).</p> <p>Consequently, this option can only be selected when you've chosen On-Premise for the Proxy Type option, meaning you've configured outbound connectivity to an on-premise system through the SAP Cloud Platform Connectivity (see the upcoming info box).</p> <p>Setting up a scenario with this authentication option requires comprehensive configuration steps at the inbound and outbound side of SAP Cloud Platform Integration, as well as in SAP Cloud Platform Connectivity and the receiver backend system. A detailed step-by-step tutorial would go beyond the scope of this book.</p>
OAuth (when using Twitter or Facebook adapter)	<p>SAP Cloud Platform Integration calls Twitter or Facebook using OAuth authentication mechanisms.</p> <p>A Secure Parameter artifact is required to store the OAuth credentials. For more information, Section 10.4.5.</p> <p>This authentication option isn't offered for the other HTTP-based adapters and therefore is also not shown in the dropdown list in Figure 10.16.</p>

Table 10.6 Outbound Authentication Options (for HTTPS-Based Communication) (Cont.)

Authentication option **None** (as shown in [Figure 10.16](#)) isn't considered in the table. If this option is selected, no authentication is required for the tenant when calling a receiver system.

Note that to have permission to deploy security-related artifacts, your user needs to have assigned dedicated roles, for example, the authorization group `AuthGroupAdministrator` (we covered the topic of roles and authorization groups in detail in [Section 10.3](#)).

Proxy Type

The following adapter settings are relevant in the context of configuring **Authentication** setting **Principal Propagation**.

In most HTTP-based adapters (e.g., the SOAP and IDoc adapter), you find the attribute **Proxy Type**. In the scenarios in this book, we always kept the default setting of this attribute as **Internet**, which makes sure that the tenant can connect to another system through the Internet (e.g., over HTTP).

The other option for the **Proxy Type** attribute is **On Premise**. Using this option, you enable the tenant to connect to an on-premise system through SAP Cloud Platform Connectivity.

When setting up such a scenario, you also need to install an additional component, referred to as the *cloud connector*, in your on-premise landscape that acts as proxy for those requests that try to access your on-premise system coming from the Internet.

If you use multiple cloud connector instances in your system landscape, you also need to specify a **Location ID**. With this attribute, you can identify the cloud connector instance you want to use for your connection.

You might have noticed that when you select **On Premise** as **Proxy Type**, **Authentication** option **Client Certificate** is deactivated. This expresses the fact that when using SAP Cloud Platform Connectivity, this authentication option isn't supported in the respective receiver adapter. If client certificate authentication is nevertheless required for such a connection, you need to configure this authentication option when setting up SAP Cloud Platform Connectivity.

For more information, consult the documentation for SAP Cloud Platform at <https://help.sap.com/viewer/p/CP> in the **Cloud Connector** section and the "Using SAP Cloud Platform Cloud Connector with SAP Cloud Platform Integration" blog in the SAP Community (www.sap.com/community.html).

10.4.4 Securely Connecting a Customer System to SAP Cloud Platform Integration (through HTTPS)

Having introduced the transport-level security options in [Section 10.4.2](#) and authorization and authentication options in [Section 10.4.3](#), we can now discuss how to establish a secure connection between a remote system and SAP Cloud Platform Integration. For simplicity, we'll focus on one—and the most common—option: using HTTPS over TLS and client certificate authentication. We discuss this topic separately for inbound communication (when a sender sends a request to SAP Cloud Platform Integration) and for outbound communication (when SAP Cloud Platform

Integration sends a request to a receiver) because the setup of components and the sequence of tasks differ considerably, depending on the communication direction.

For inbound communication, we show how to set up **User Role** authorization in combination with a certificate-to-user mapping.

For both directions, we first outline the target picture (setup of components and key stores) and then explain the steps required to achieve this setup.

Note that we don't provide an end-to-end tutorial on how to set up a specific integration flow. Instead, we'll focus on those steps that are relevant to set up a secure connection both for inbound and outbound direction (which are mainly certain steps in the configuration of the sender and receiver adapter). We also don't show how to configure certain sender or receiver systems in detail, as this would require a complete integration scenario and a specific technical landscape (for specific kinds of sender and receiver systems). To find more information on such a complete setup for integration packages predefined by SAP (including the configuration of certain SAP systems at the sender and receiver side), refer to the integration guides that you can find in the **Documents** section of the Integration Content Catalog for certain integration packages (see [Chapter 3](#)).

When showing the connection setup-related steps for the sender and receiver adapters, we show (as an example) how to do this for a SOAP 1.x adapter. However, the steps work also for most other adapters that support HTTP communication.

Inbound Communication

For inbound communication, the recommended option is to use certificate-to-user mappings. In [Chapter 8, Section 8.3.3](#), we explained the corresponding integration artifact and how to manage it, and we already briefly introduced it in [Table 10.5](#) of this chapter. Next, we'll show you how to include a certificate-to-user mapping in the connection setup for inbound communication.

[Figure 10.17](#) shows how the components interact with each other at runtime (on top) and the required security setup to realize this behavior (bottom).

To begin the target setup, a characteristic feature of inbound HTTP connections is that the load balancer terminates each TLS request from the sender and establishes a new TLS connection to the tenant, which is then processing the request (refer to [Figure 10.1](#)). The sender system connects to the load balancer via TLS and verifies the load balancer certificate. Vice versa, the load balancer verifies if the certificate sent by the sender system is valid (as we've chosen client certificate authentication of the

sender system). For the discussed security option, the certificates are stored in a key-store on the load balancer component (which is a component maintained by SAP). It's essential that the client certificate installed on the sender system is signed by one of the CAs that are supported by the load balancer. SAP has published a list of such CAs in the online documentation as shown later. If the validation is successful, the load balancer forwards the client certificate to the tenant.

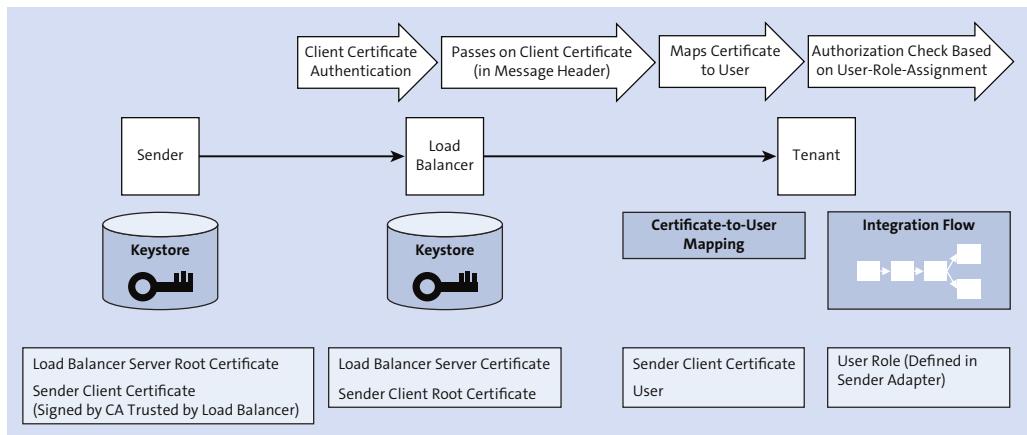


Figure 10.17 Setup of Components Required to Establish a Secure Inbound Connection Using HTTPS, Certificate-to-User Mapping, and User Role Authorization

In a subsequent step, the tenant evaluates (for the certificate) if a certificate-to-user mapping is defined (as deployed in a **Certificate-to-User Mappings** artifact on the tenant). If that is the case, the tenant checks if the sender associated with the user (as derived from the certificate-to-user mapping) is authorized to process the integration flow on the tenant by doing the following:

- Evaluating the user-to-role assignments (as defined on the subaccount for the runtime node application)
- Checking the **User Role** specified in the sender adapter of the integration flow

If the check is successful, the message is processed on the runtime node according to the integration flow settings.

You can find a step-by-step description of the required steps to set up such a connection in the “Cloud Integration – How to Set Up Secure HTTP Inbound Connection with Client Certificates” blog in SAP Community: www.sap.com/community.html. This blog

outlines the necessary steps in detail. In the following subsections, we provide a summary of the key aspects.

Configuring the Sender System

Sender system configuration includes generating a client certificate and sending the corresponding certificate signing request (CSR) to a CA for a signature. The administrator of the sender system needs to make sure that it's signed by a CA that is also supported by the load balancer. Afterwards, he applies the CA reply and imports the signed certificate into the sender keystore.

Note

How the certificates are installed and the type of keystore used on the sender side depends on the kind of sender system, which isn't covered here.

Because the load balancer is controlled centrally and administered by SAP, no configuration actions are required for this component. The administrator of the sender system has to make sure that the certificates installed on the sender system are compatible with those installed on the load balancer in terms of the hierarchical trust model.

You'll find a list of root certificates actually supported by the load balancer in the documentation of SAP Cloud Platform Integration at https://help.sap.com/viewer/product/CLOUD_INTEGRATION/Cloud under **Connecting a Customer System to Cloud Integration • Concepts of Secure Communication • Basics • HTTPS-Based Communication • Load Balancer Root Certificates Supported by SAP**.

Configuring the Inbound Authorization in the Related Integration Flow

We assume that as the integration developer, you use a SOAP 1.x sender adapter (see Chapter 2, Section 2.3.4, Figure 2.27).

For **Authorization**, keep the setting **User Role** to make sure that for the user associated with the calling sender, the permissions are checked based on user-to-role assignments by the SAP Cloud Platform Integration framework. In the **User Role** field, you can keep the entry **ESBMessaging.send**. This role is predefined by SAP to authorize a sender (the SOAP client) to call your tenant. Alternatively, you can specify a custom role here. With this option, you can restrict access to SAP Cloud Platform Integration on the level of individual integration flows (e.g., in case certain integration flows are only to be called by specific senders).

To use custom roles, you also need to define this custom role for the runtime node and assign it to the relevant user in the SAP Cloud Platform cockpit (as explained earlier in [Section 10.3.3](#)).

When you've finished the integration flow design, deploy the integration flow on the tenant.

Defining and Deploying a Certificate-to-User Mappings Artifact

Open the **Monitor** application of the Web UI, and click the **Certificate-to-User Mappings** tile under **Manage Security**. To add a new certificate-to-user mapping, choose **Add**. In the next dialog, enter the **User Name** and click **Browse** to look for the signed certificate, which you've exported from the sender keystore prior to this activity, as shown in [Figure 10.18](#).

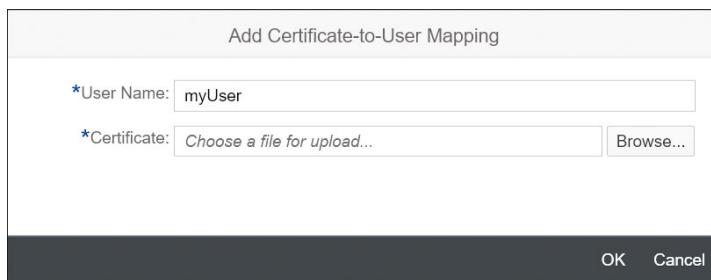


Figure 10.18 Defining a Certificate-to-User Mapping

You need to import the client certificate that the sender system uses to authenticate itself against the load balancer.

Click **OK** to deploy the newly defined certificate-to-user mapping on the tenant. For more information on the **Certificate-to-User Mappings** artifact, read [Chapter 8, Section 8.3.3](#).

Configuring the User-to-Role Assignment

To finish the setup, you need to assign the user specified in the certificate-to-user mapping to the role specified in the sender channel of the integration flow in the SAP Cloud Platform cockpit. Only then is the sender allowed to send messages to the integration flow.

The steps when using the predefined role **ESBMessaging.send** were already shown in [Chapter 2, Section 2.3.3](#). For your convenience, we again show the related dialog in [Figure 10.19](#).

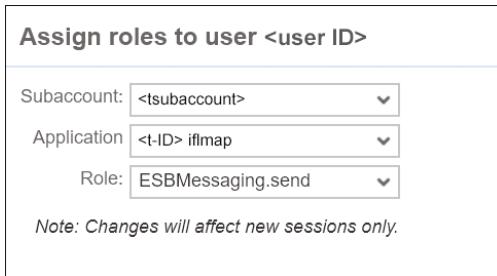


Figure 10.19 Assigning ESBMessaging.send to the Runtime Node in SAP Cloud Platform Cockpit

As mentioned before, you can also create your own roles in this scenario. This gives you the option to specify individual permissions for each integration flow (sender adapter).

Let's assume you define a custom role called **MyRole** for the runtime node (as explained in [Section 10.3.3](#)) and assign it to the user associated with the inbound request (let's say **MyUser**). If you use a custom role, the previously described tasks of defining the inbound authorization (in the sender adapter) and of defining the certificate-to-user mapping are slightly different:

- In the sender adapter (e.g., the SOAP 1.x channel), you define the following settings: as **Authorization**, specify **User Role**, and as **User Role**, select **MyRole**.
- In the **Certificate-to-User Mappings** artifact, you enter the user "MyUser" (to be mapped from the chosen certificate).

Having configured these settings, the integration flow can only be processed when the certificate associated with the inbound call is mapped to user MyUser and when in SAP ID service the role MyRole is assigned to user MyUser.

Client Certificate Authorization

In an alternative setup, without certificate-to-user mapping, inbound communication can be defined so that the permissions of the sender are checked based on the subject/issuer DN of a client certificate, which is specified directly in the integration flow.

In this case, the tenant checks if the sender is authorized to call the tenant by comparing the certificate forwarded by the load balancer with the one specified in the relevant integration flow.

This option has certain disadvantages:

- Each time you change (and redeploy) the integration flow, a brief downtime is caused by this.
- Each time when the client certificate is renewed, the integration flow needs to be redeployed (which also causes downtime).

Therefore, SAP doesn't recommend using **Client Certificate** authorization.

Outbound Communication

We now consider the outbound side where the tenant sends a message to a remote receiver system using an HTTP connection. Figure 10.20 shows how the components interact with each other at runtime (on top) and the required security setup to realize this behavior (bottom).

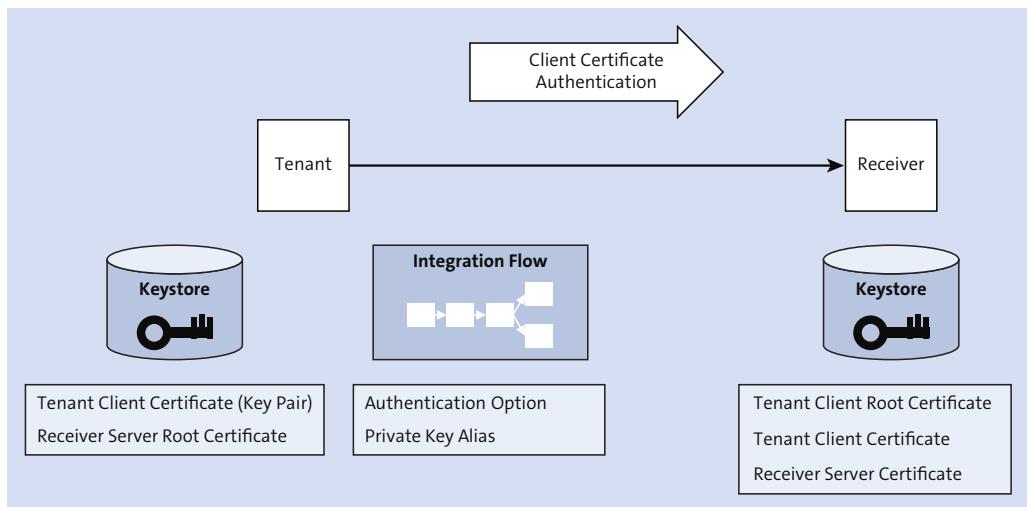


Figure 10.20 Setup of Components Required to Establish a Secure Outbound Connection Using HTTPS and Client Certificate Authentication

To begin the target setup, the tenant connects to the receiver via TLS and verifies the receiver certificate. Vice versa, the receiver verifies if the certificate sent by the tenant is valid. These steps are performed based on the installed certificates, both on the tenant and in the receiver system. For the discussed security option, the certificates of the tenant are stored in a Java keystore deployed on the tenant. Similar to the

inbound case, the identity and permissions of the tenant are checked in the receiver system (based on the settings in the receiver system).

You can find a step-by-step description of the required steps to set up such a connection in the “Cloud Integration – How to Set Up Secure Outbound HTTP Connection Using Keystore Monitor” blog in SAP Community at www.sap.com/community.html. This blog outlines the necessary steps in detail. In the following subsections, we summarize the key aspects.

Configuring the Receiver System

Configuring the receiver system requires first creating a server certificate (private key pair) and then importing it into the receiver keystore. The server certificate can be a certificate chain where the top-level certificate is a root certificate issued by a dedicated CA. The administrator of the receiver system needs to download the root certificate from the receiver keystore and make it available to the tenant administrator.

Additionally, the receiver keystore needs to contain the tenant client certificate and the tenant client root certificate. To enable the receiver administrator to perform the necessary steps, the tenant administrator needs to export the certificate chain of the tenant client certificate from the tenant keystore, extract the root certificate and the client certificate out of it, and hand both over to the receiver system administrator.

Configuring the Tenant Keystore

With the provisioning of your tenant cluster, SAP has already provided a tenant keystore with a number of certificates. You can find the content of the keystore by opening the **Monitor** application of the Web UI and selecting the **Keystore** tile (under **Manage Security**) (see also [Chapter 8](#), [Section 8.3.2](#), and [Section 10.5](#) in this chapter).

To enable the tenant to trust the receiver system, the tenant administrator needs to import the (server) root certificate of the receiver, which is to be provided by the administrator of the receiver system, into the tenant keystore.

To enable trust the other way around (so that the receiver trusts the tenant), a client certificate is also required as part of the tenant keystore (private public key pair). All tenants provided by SAP contain already a key pair. You can use this certificate to configure this step.

Figure 10.21 shows a part of the certificate list of the tenant keystore accessible in the **Monitor** application under **Manage Security** in the **Keystore** tile.

Entries (42)					
Alias	Type	Owner	Valid Until	Last Modified At	Actions
addtrust external ca root	Certificate	Tenant Administrator	May 30, 2020, 12:48:38	Jan 31, 2018, 07:25:35	[Edit]
baltimore cybertrust root	Certificate	Tenant Administrator	May 13, 2025, 01:59:00	Jan 31, 2018, 07:25:35	[Edit]
certum ca	Certificate	Tenant Administrator	Jun 11, 2027, 12:46:39	Jan 31, 2018, 07:25:35	[Edit]
certum level iv ca (certum ca)	Certificate	Tenant Administrator	Mar 03, 2024, 13:54:25	Jan 31, 2018, 07:25:35	[Edit]
comodo high-assurance secure server ca (addtrust external ca root)	Certificate	Tenant Administrator	May 30, 2020, 12:48:38	Jan 31, 2018, 07:25:35	[Edit]
cybertrust public sureserver sv ca (baltimore cybertrust root)	Certificate	Tenant Administrator	Sep 08, 2020, 19:34:08	Jan 31, 2018, 07:25:35	[Edit]
digitalocean root ca	Certificate	Tenant Administrator	Nov 10, 2021, 01:00:00	Jan 31, 2018, 07:25:35	[Edit]

Figure 10.21 Certificates Contained in the Tenant Keystore

Importing a certificate to the tenant keystore was already described in [Chapter 2, Section 2.3.6](#). [Section 10.5](#) will provide more details on certificate management for the tenant.

Configuring and Deploying the Integration Flow

In addition to the previous steps, you need to specify the required security settings in the related integration flow and the associated receiver adapter.

If you use an HTTP receiver adapter to connect the tenant to the receiver system, specify the following settings:

- **Authentication**

Select the **Client Certificate** option.

- **Private Key Alias**

Enter the alias of the signed tenant client certificate that you intend to use for authentication when connecting to the associated receiver. The alias is used to point to a specific key pair in the tenant keystore. If you leave this field empty, any valid key pair will be used for this step.

Figure 10.22 shows the authentication options that are available in the HTTP-based receiver adapters (e.g., we show the UI of the HTTP receiver adapter, **Connection** tab).

Finally, deploy the integration flow on the tenant.



Figure 10.22 Authentication Options Available in the HTTP Receiver Adapter

Outbound Connectivity Test Tool

To test your security configuration, SAP Cloud Platform Integration provides an outbound connectivity test tool for the various protocols, among them also for HTTP connections using TLS.

When setting up the first integration flow described in this book ([Chapter 2, Section 2.3.6](#)), you already got to know this tool, in that case, for the SMTP protocol.

This tool is described in detail in [Chapter 8, Section 8.3.4](#).

10.4.5 Setting Up a Scenario Using OAuth with the Twitter Adapter

In [Section 10.4.3](#), we introduced the concept of OAuth, which allows a resource owner to grant client applications restricted access to his resources. Social networks, such as Facebook or Twitter, provide APIs for client applications that support OAuth.

SAP Cloud Platform Integration offers adapters to connect a tenant with Facebook and Twitter to write data to these platforms or to read data from them. Both adapter types work in the same way. In this section, we'll show you how to set up a simple integration scenario using the Twitter adapter.

The Twitter receiver adapter allows an SAP Cloud Platform Integration tenant (in terms of OAuth it's the *client*) to access Twitter (the *resource server*) on behalf of a Twitter user (the *resource owner*). After gaining access, the tenant can either read or post tweets on Twitter (via the resource owner's account). You can use this adapter, for example, to implement market analysis scenarios.

Technically, the tenant calls the Twitter API. As a prerequisite, the integration developer has to prepare the tenant so that it can call Twitter on behalf of a specific Twitter user. To do that, the integration developer obtains a set of client credentials from Twitter that will be used to authenticate the client (tenant) in the Twitter API.

In [Section 10.4.3](#), we described a common OAuth authentication scenario in which every time the client requests access to the server, the resource owner is asked by the server to grant the client access to the protected resources. This implies a user dialog where the resource owner needs to log in to the server with his credentials. For SAP Cloud Platform Integration scenarios, this workflow isn't feasible, as each time message processing is triggered, it would require a resource owner to log on to Twitter and confirm that the required token credentials are being generated for the client.

To address this aspect, the Twitter adapter uses OAuth in a more system-centric way. The integration developer (who is identical to the resource owner, i.e., the Twitter account user) provides the tenant with all credentials (client and token credentials) when defining the Twitter adapter settings in the integration flow.

[Figure 10.23](#) is derived from [Figure 10.13](#), and shows the general OAuth setup adapted to the Twitter adapter scenario.

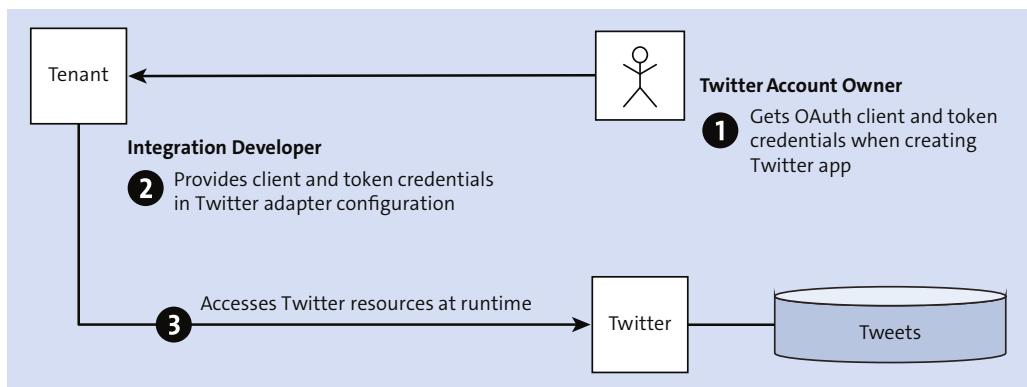


Figure 10.23 OAuth Setup of Components When Using the Twitter Adapter

Designing an Integration Flow Using Twitter Adapter

[Figure 10.24](#) shows our target integration flow. A **Start Timer** event (introduced in [Chapter 6, Section 6.1.2](#)) starts the message flow. A **Request-Reply** step (see [Chapter 4, Section 4.3](#)) calls a receiver (**Receiver2**, which represents Twitter) and reads data from it. Finally, the enriched message is sent to an email server (**Receiver1**) using the **Mail** receiver adapter. The **Mail** receiver adapter has been used in several scenarios throughout this book, so it doesn't need further explanation (e.g., [Chapter 2, Section 2.3.4](#)). We'll keep our description short and focus on the aspects that are specific to how Twitter and OAuth come into play.

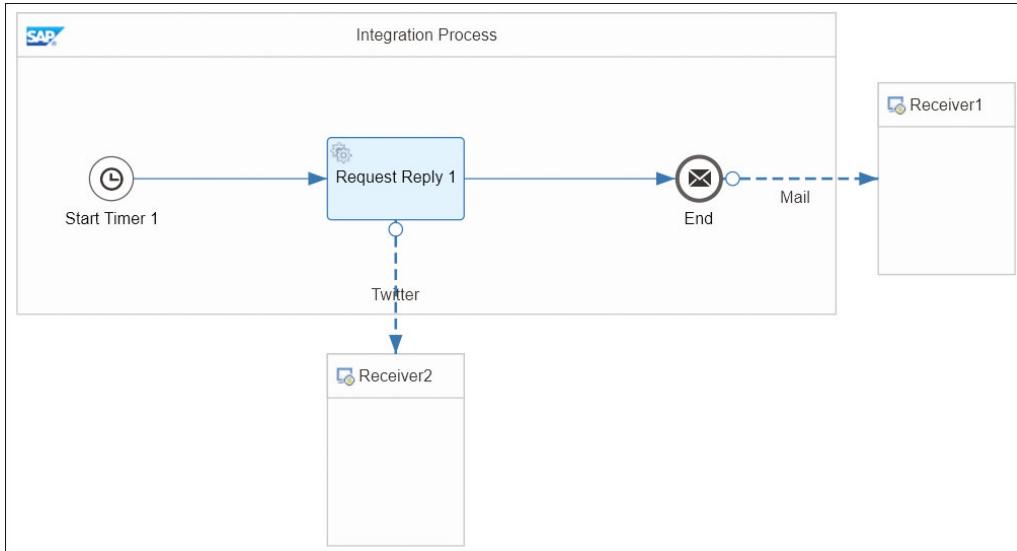


Figure 10.24 Integration Flow with the Twitter Adapter

Tasks Related to the Twitter API

Let's assume that you (the resource owner) use your own Twitter account for this scenario. To prepare to use the Twitter account in an SAP Cloud Platform Integration scenario, perform the following steps:

1. Go to <https://dev.twitter.com/apps>, and create an app by choosing **Create New App**.
2. Provide a **Name** and a **Description**. For **Website**, enter any test URL.
3. Leave the **Callback URL** field empty.
4. Confirm the Developer Agreement, and select **Create Your Twitter Application**. On the following page ([Figure 10.25](#)), certain information is displayed.
5. Choose **Keys and Access Tokens**.

Under **Application Settings**, you find your **Consumer Key** and **Consumer Secret**. Under **Your Access Token**, you find your **Access Token** and **Access Token Secret**.

Cloud Integration App

Details **Settings** Keys and Access Tokens Permissions

 Test app to connect Cloud Integration to Twitter
<http://mysite.com>

Organization

Information about the organization or company associated with your application. This information is optional.

Organization	None
Organization website	None

Application Settings

Your application's Consumer Key and Secret are used to [authenticate](#) requests to the Twitter Platform.

Access level	Read and write (modify app permissions)
Consumer Key (API Key)	RKG0v9pcABaRSE314edWdWAh (manage keys and access tokens)
Callback URL	None
Callback URL Locked	No
Sign in with Twitter	Yes
App-only authentication	https://api.twitter.com/oauth2/token
Request token URL	https://api.twitter.com/oauth/request_token
Authorize URL	https://api.twitter.com/oauth/authorize
Access token URL	https://api.twitter.com/oauth/access_token

Figure 10.25 UI of the Twitter App

Now you have all four credentials that you need for further configuration of our scenario. In the next step, you'll define a separate security artifact for each of the four credentials and deploy them on the tenant. When configuring the integration flow, you'll refer to these artifacts in the Twitter adapter configuration UI.

Creating and Deploying the OAuth Security Artifacts

To provide the tenant with the credentials, you need to create and deploy an individual **Secure Parameter** artifact for each of the four OAuth credentials. It's important to point out that during this step, only you, the resource owner, can see the actual values of the keys and access tokens (when creating and deploying the artifacts). In the same way as creating and deploying a **User Credentials** artifact (see [Chapter 2, Section 2.3.4](#)), you'll give each of these entities an alias, which will be used during the subsequent integration flow configuration to complete the Twitter adapter settings. That way, no one aside from you knows the keys and access tokens. Others who use the Web UI on the same tenant will only see the credential names (aliases), keeping confidential information protected.

For the following process, we propose the aliases listed in [Table 10.7](#) for the four credential types.

Credential	Alias
Consumer key	Twitter_ConsumerKey
Consumer secret	Twitter_Secret
Access token	Twitter_AccessToken
Access token secret	Twitter_TokenSecret

Table 10.7 Proposed Aliases for the OAuth Credentials

Repeat the following steps for each of the four credentials: consumer key, consumer secret, access token, and token secret.

1. Connect to your tenant using the Web UI, and choose **Monitor**.
2. Click the **Security Material** tile in the **Manage Security** section.
3. Choose **Add • Secure Parameter**.
4. For **Name**, enter an alias (which you later have to enter into the corresponding field of the Twitter adapter). We recommend that you use the aliases proposed in [Table 10.7](#).
5. Paste the corresponding credential from your Twitter app into the fields for **Secure Parameter** and **Repeat Secure Parameter**.
6. Click **Deploy**.

The **Secure Parameter** artifact has now been deployed and is displayed in the table under **Manage Security Material**.

Configuring the Twitter Adapter

With the following steps, you create a receiver communication channel with Twitter adapter:

1. Using the Web UI, create an integration flow with the elements depicted in [Figure 10.24](#).
2. For the **Mail** adapter, you can use the settings proposed in [Chapter 2, Section 2.3.4](#).
3. For the **Start** step, use a **Timer** event. You can leave the **Run Once** checkbox selected.
4. Configure the Twitter adapter ([Figure 10.26](#)).

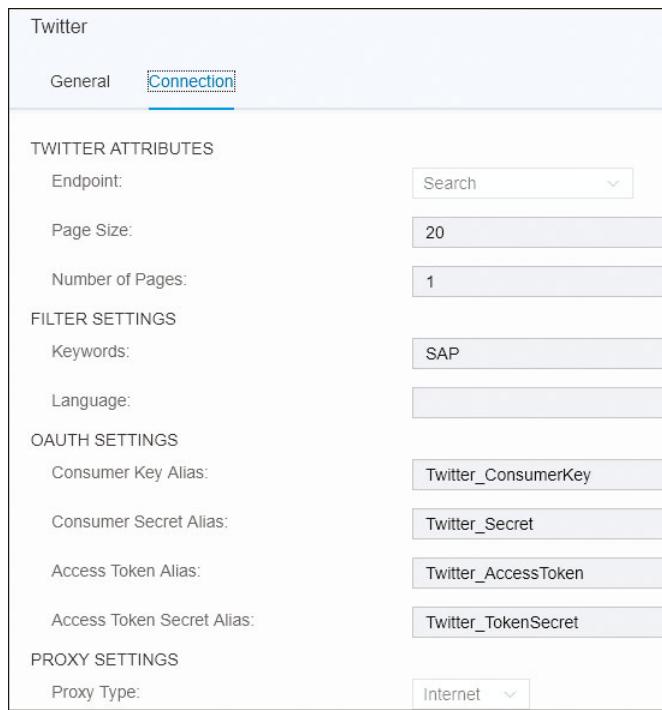


Figure 10.26 Twitter Adapter Settings

For **Endpoint** select **Search**, which enables the tenant to extract information from Twitter (based on certain criteria).

There are two other options in the **Endpoint** dropdown list:

- **Send Tweet**: Enables the tenant to send a message to Twitter (to the account of the resource owner associated with the credentials configured under **OAuth Settings**).
- **Send Direct Message**: Enables the tenant to send a message to a user who doesn't necessarily have to be identical to the resource owner. When choosing the **Send Direct Message** option, you specify this user in an additional **User** field that is hidden when one of the other two options for **Endpoint** is selected.

Select **Search** because our integration flow starts with a timer, and it makes no sense to send an empty message to Twitter.

5. **Page Size** allows you to specify the maximum number of tweets per page, and **Number of Pages** allows you to specify the number of pages that the tenant is supposed to consume.
6. In the **Keywords** field, you can either enter keywords or Twitter queries to filter Twitter content (queries are explained at <https://dev.twitter.com/rest/public/search>).
7. With **Language**, obviously, you can specify the search language (e.g., **EN** for English).
8. Under **OAUTH SETTINGS**, enter the aliases of the **Secure Parameter** artifacts deployed in the previous step (see also [Table 10.7](#)).
9. Save and deploy the integration flow.

As soon as you've deployed the integration flow, message processing is triggered (according to the settings in the timer **Start** event). Go to your email account and check if a message has been received. It should contain tweets according to the criteria you provided in the Twitter adapter.

OAuth for Inbound Connections

This tutorial showed how to set up a scenario using OAuth at the *outbound* side. There are also several options to configure OAuth for *inbound* connections ([Section 10.4.3](#)). For more information, check out the documentation of SAP Cloud Platform Integration at https://help.sap.com/viewer/product/CLOUD_INTEGRATION/Cloud under **Connecting a Customer System to Cloud Integration • Configuring Inbound Communication**.

10.4.6 Message-Level Security Options

On top of establishing a secure transport channel as explained in [Section 10.4.2](#), you can further protect the messages to be exchanged with digital encryption and digital signatures. Applying these options increases the security level of your scenario because even if the transport channel is broken, the messages still can't be read by malicious parties. Signing a message allows a recipient to validate if the message has been received from the expected sender, increasing data integrity.

Why Decrypt Messages on the SAP Cloud Platform Integration Tenant?

You might ask yourself why not always pass encrypted messages through SAP Cloud Platform Integration and thus keep the exchanged message a *black box* for the integration middleware (unreadable for anyone in the unlikely case that any malicious party receives unauthorized access to the infrastructure at the software provider's side). Why is there any need to decrypt a message on the tenant at all? The reason is that various integration patterns require the runtime components to access the message content to process the message correctly.

One example for this is the content-based router (CBR) (see [Chapter 5, Section 5.1](#)). Consider a scenario where a message sent from one company is routed to one, or many, possible banks (receivers), and the actual receiver depends on the value of the bank identifier code (BIC) in the message. To process the routing step in the intended way, the message has to be decrypted on the tenant prior to the routing step to reveal the BIC.

How Digitally Encrypting and Decrypting a Message Works

SAP Cloud Platform Integration supports various message-level security standards that each work in different ways. However, there is a general pattern. All of these options use a hybrid approach of asymmetric and symmetric key technologies ([Section 10.4.1](#)). We'll briefly show this pattern for both use cases: encryption/decryption and signing/verifying.

For encryption/decryption, the process was already explained in the info box in [Section 10.4.1](#) and illustrated in [Figure 10.10](#), so we'll only briefly repeat the steps.

The sender uses a (typically randomly generated) symmetric key to encrypt the message content. Additionally, the sender uses the public key of the receiver (which has been provided by the receiver in advance) to encrypt the symmetric key. The encrypted symmetric key along with the encrypted message content is sent to the

receiver. The receiver then uses its private key (which is associated with the public key that was used by the sender) to decrypt the symmetric key. With the recovered symmetric key, the receiver decrypts the message content.

How Digitally Signing/Verifying of a Message Works

When signing/verifying a message, the usage of public and private keys is inverted in the following way: the sender uses its private key to generate a signature out of the message content, and the receiver uses the associated public key (which has been provided by the sender in advance) to verify the signature.

As we mentioned in [Section 10.4.1](#), the usage of asymmetric keys is, in general, computationally intensive. Therefore, applying the digital signing process on the complete content of a message can have a negative impact on overall performance because message sizes can vary from a few kilobytes to several megabytes. To overcome this problem, a *hash function* is applied to the message content prior to the signing process. A hash function allows you to calculate an expression of fixed size from an input (which can be of any size). The input, in our case, is the message content. The output is referred to as a *hash value* (other, synonymous terms are *digest*, *footprint*, and *fingerprint*). The calculation is accomplished in a fully reproducible way, which means performing the same hash algorithm on the same data will lead to the same hash value. However, the inverse operation isn't possible, which means the data can't be reproduced out of the hash value (this makes a difference to compression algorithms that allow, based on the compressed data, recovery of the uncompressed original). The resulting small size hash value is then subject to the signing process, as shown in [Figure 10.27](#).

Here, the asymmetric key pair of the sender comes into play in the following way: The sender's private key is used to encrypt the hash value. The transferred hash value can be considered the digital signature and is sent along with the message content to the receiver. On the receiver side, the public key (associated to the sender's private key) is used to decrypt the hash value. In a separate step that is independent from the decryption of the hash value, the receiver uses the hash algorithm to calculate the hash value directly out of the message content and compares the result with the hash value that has been decrypted using the public key. If both values are identical, the signature is verified.

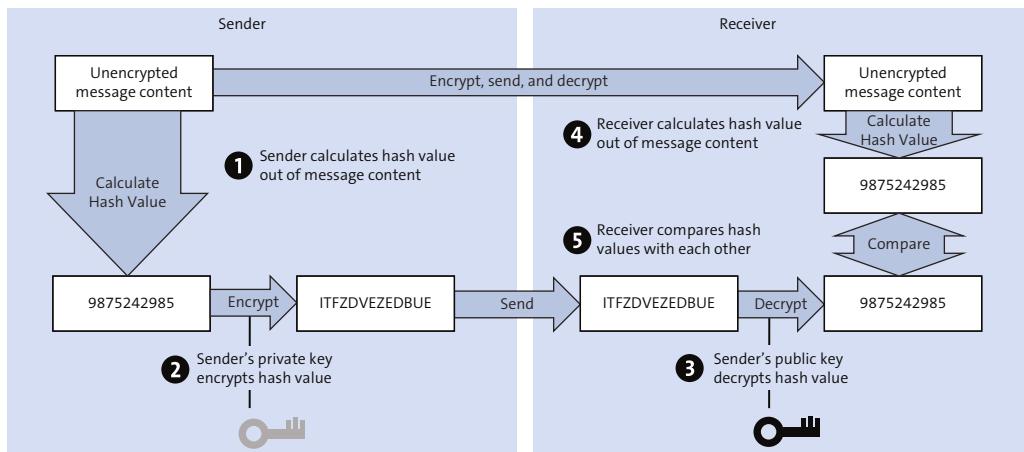


Figure 10.27 Combining Hash Functions with Asymmetric Key Usage When Signing and Verifying a Message

Overview of Supported Standards

Table 10.8 summarizes the message-level security standards supported by SAP Cloud Platform Integration and lists the supported options.

Standard	Options and References
PKCS#7/CMS Enveloped Data and Signed Data	<p>PKCS stands for <i>Public Key Cryptography Standard</i> and CMS stands for <i>Cryptographic Message Syntax</i> (see https://tools.ietf.org/html/rfc2315).</p> <p>This standard provides the following options:</p> <ul style="list-style-type: none"> Encrypting the message content (and, vice versa, decrypting it) Signing a message (and, vice versa, verifying it) A combination of encrypting and signing a message (and, vice versa, the combination of decrypting and verifying it)

Table 10.8 Message-Level Security Options Supported by SAP Cloud Platform Integration

Standard	Options and References
OpenPGP	<p>OpenPGP stands for <i>Open Pretty Good Privacy</i> (see https://tools.ietf.org/html/rfc4880).</p> <p>This standard provides the following options:</p> <ul style="list-style-type: none"> ■ Encrypting the message content (and decrypting it) ■ A combination of encrypting and signing a message (and the combination of decrypting and verifying it)
XML Signature	<p>XML Signature is also referred to as <i>XMLDSig</i>, <i>XML-DSig</i>, or <i>XML-Sig</i>. It's defined by a W3C recommendation (see www.w3.org/TR/xmlsig-core/).</p> <p>This standard allows for signing a message and verifying it.</p>
XAdES	<p>XML Advanced Electronic Signatures (XAdES) is an enhancement of XML Signature, which can be used in the context of the European Union Directive 1999/93/EC. It allows you to use signatures in electronic contracts within the European Union (see www.w3.org/TR/XAdES/).</p> <p>This standard allows for signing a message.</p>
WS-Security	<p>Web Services Security (WS-Security) is an extension to SOAP that allows you to apply security to web services (see www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss).</p> <p>This standard provides the following:</p> <ul style="list-style-type: none"> ■ Signing a SOAP body and verifying it ■ Encrypting the message content and decrypting it <p>The following adapters support WS-Security:</p> <ul style="list-style-type: none"> ■ SOAP 1.x sender and receiver adapter ■ AS4 receiver adapter
S/MIME	<p>S/MIME allows you to securely transfer Multipurpose Internet Mail Extensions (MIME) data—in other words, emails. It's used with most common email programs (see https://tools.ietf.org/html/rfc5751).</p> <p>This standard allows you to use digital signatures and to encrypt emails.</p>

Table 10.8 Message-Level Security Options Supported by SAP Cloud Platform Integration (Cont.)

Now you know which security options are supported by SAP Cloud Platform Integration at the transport level and the message level. In [Section 10.4.4](#), we showed you how to set up a secure connection with SAP Cloud Platform Integration based on TLS; in [Section 10.4.5](#), we showed you how to set up a scenario using OAuth with the Twitter adapter; and in [Section 10.4.7](#), we'll show you how to set up a scenario that includes digital encryption (using the OpenPGP standard).

10.4.7 Designing Message-Level Security Options in an Integration Flow

In this section, we'll show you how a message can be protected on the application level by digital encryption and signatures. We've given an overview of the options and concepts in [Section 10.4.6](#).

As we did for the transport-level security considerations in [Section 10.4.4](#), we'll here also distinguish between two communication directions (from the perspective of the tenant):

- **Inbound communication**

The sender sends an -encrypted or signed message to SAP Cloud Platform Integration, and this message needs to be decrypted or verified on the tenant.

- **Outbound communication**

The tenant encrypts or signs a message and sends it to a receiver where it's decrypted or verified.

We'll summarize the key tasks of how to set up message-level security scenarios for both use cases. We then finish this section with a tutorial that shows you how to set up a simple integration flow, which includes decrypting and encrypting a message.

Before we discuss the two use cases, let's continue with some general remarks. As for transport-level security, you also need to create digital keys and establish the corresponding key storages. You've learned in [Section 10.4.4](#) that SAP provides you the tenant already with a Java keystore deployed on it, and that to implement certain transport-level security options, this keystore needs to contain dedicated keys.

The key material for the different message-level security options must be stored in different types of key storages. Accordingly, different security artifact types have to be deployed on the tenant. In [Table 10.9](#), we summarize the artifact types that are relevant to store keys for message-level security.

Security Standard	Related Artifact Types
PKCS#7 WS-Security XML Signature	<p>You use X.509 keys, the same type of security material used for TLS. Therefore, you can import additional keys for message-level security into the same Java keystore that is used for transport-level security (and which you can access in the Monitor application under Manage Security in the Keystore tile; see Chapter 8, Section 8.3.2, and Section 10.5 in this chapter).</p> <p>However, make sure that you use different keys for message-level security than for securing the transport channel.</p> <p>Keys for message-level security don't need to be signed by a CA; they can be self-signed.</p>
OpenPGP	<p>You use specific PGP keys. In addition, the terminology is slightly different when compared to the other security standards. For example, in the context of PGP, <i>private keys</i> are referred to as <i>secret keys</i>. You also need dedicated key storages:</p> <ul style="list-style-type: none"> ■ Use a PGP Public Keyring to store a PGP public key (associated with the sender or receiver system connected to the tenant). ■ Use a PGP Secret Keyring to store a private-public key pair associated with the tenant. <p>You can access the corresponding artifacts in the Monitor application under Manage Security in the Security Material tile).</p>

Table 10.9 Security Artifacts Required to Store Keys for Certain Message-Level Security Scenarios

Most importantly, to configure message-level security options, you have to add dedicated steps to the integration flow, which we'll discuss next.

Inbound Communication

[Figure 10.28](#) shows how the components interact with each other at runtime (at the top) and the required security setup to realize this behavior (on the bottom).

In this use case, the components interact with each other in the following way:

- The sender does the following:
 - Encrypts the message content using the tenant's public key, which is associated with the tenant's private key
 - Signs a message using its own private key

- The tenant receives the message and then does the following:
 - Decrypts the message content using its own private key
 - Verifies the message using the sender's public key, which is associated with the sender's private key

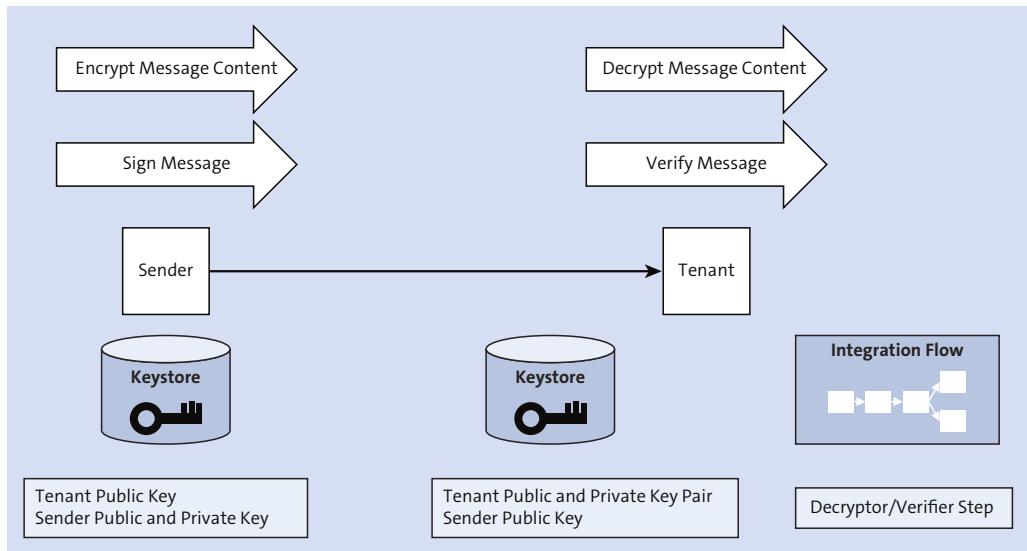


Figure 10.28 Configuring Message-Level Security for Inbound Communication

You might notice that, other than in the transport-level security case, the load balancer isn't depicted in Figure 10.28. It doesn't play a role because we consider the application level (or message level), not the transport level. The termination and reestablishment of the TLS request takes place only on the transport level. The message is encrypted on its way between the sender and tenant (across the load balancer).

To implement this setup, both on the side of the sender system and the SAP Cloud Platform Integration platform, perform the following tasks:

- Generate key pairs for both the tenant and the sender system, and set up the keystores, as shown in Figure 10.28.
- Import the tenant's public key into the sender's keystore.
- Import the sender's public key into the tenant's keystore.
- Configure the decryptor and verifier steps in the integration flow.

Note that you can also configure a combination of decryption and verification on one decryptor step for certain security standards. However, we won't cover this use case.

Outbound Communication

[Figure 10.29](#) shows how the components interact with each other at runtime (on top), and the required security setup to realize this behavior (at the bottom).

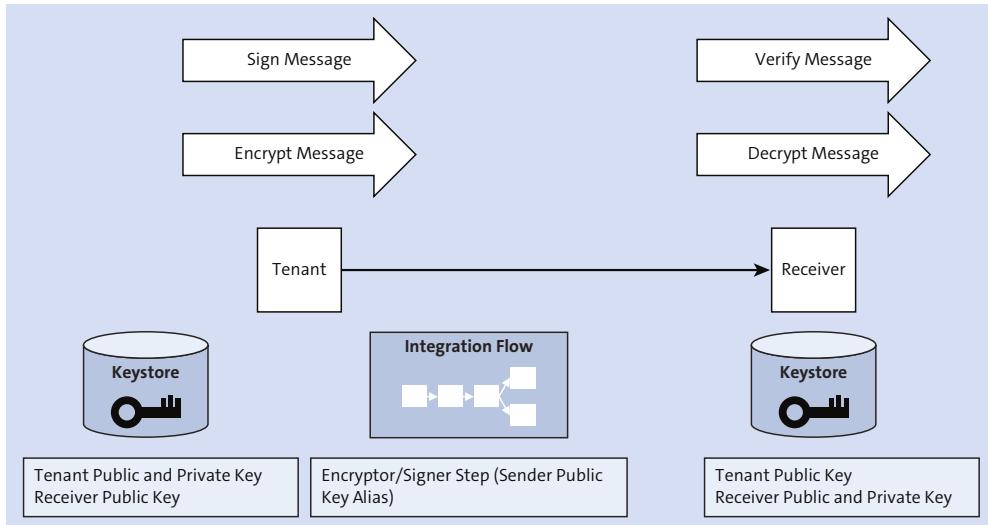


Figure 10.29 Configuring Message-Level Security for Outbound Communication

In this use case, the components interact with each other in the following way:

- The tenant does the following:
 - Encrypts the message content using a public key associated with the receiver's private key
 - Signs a message using its own private key
- The receiver does the following:
 - Decrypts the message content using its own private key
 - Verifies the message using the public key associated with the tenant's private key

To set up this scenario, you have to generate key pairs for the tenant and for the receiver, and you have to configure the relevant integration flow steps at the tenant's side.

Note that the encryptor also provides the option to combine encryption/signing.

Setting Up a Simple Integration Flow with Decryption/Encryption Steps

To complete our section on message-level security, we'll show you how to set up an integration flow that decrypts an incoming (encrypted) message, encrypts it again, and sends the newly encrypted message as an email to a receiver. A person at the receiver side then manually decrypts the received message. For simplicity, we won't consider message signing/verifying.

We'll demonstrate how this works using the OpenPGP standard. For the management of PGP keys and the manual encryption and decryption steps in the scenario, we propose you use Kleopatra, which is a publicly available software that is easy to install. Kleopatra allows you to manage PGP keys and to sign and encrypt (as well as verify and decrypt) files and text strings. You can download Kleopatra from www.gpg4win.de/index.html.

Some Remarks about OpenPGP

When used in the context of SAP Cloud Platform Integration, OpenPGP not only uses different key storages than the other security standards often use for message-level security (i.e., PKCS#7) (see [Table 10.9](#)), but it also comes with a slightly different terminology: instead of *private key*, in the context of Open PGP, the term *secret key* is used. For more information, you can go to the documentation of SAP Cloud Platform Integration at https://help.sap.com/viewer/product/CLOUD_INTEGRATION/Cloud and search for [How OpenPGP Works](#).

Here's the scenario: As sender, we use an email server from which SAP Cloud Platform Integration polls emails (according to the mail sender adapter settings). An email that is due to be read by SAP Cloud Platform Integration contains an encrypted message (as clear text, we use the string sequence Paul Smith). The decryptor step decrypts the message and sends the clear text to an email account through the mail receiver adapter. The received email will contain the message as clear text.

For simplicity, we propose that in this scenario, you use the same email server and email account as sender and receiver. This means that the integration flow will poll an email with an encrypted message from the email account and send it to the same email account.

When we've executed this scenario, we'll enhance it by an additional encryptor step so that the message is encrypted again after the decryptor step and is then sent

through the mail adapter to an email account. In that variant of the scenario, the receiver of the mail manually decrypts the received text using Kleopatra.

We've shown the mail receiver adapter several times throughout this book (see, e.g., [Section 2.3](#)). Therefore, we keep the related steps short and focus on the aspects of message encryption/decryption and the management of the required keys.

In this tutorial, you'll successively take over the roles of all participants: the sender, who encrypts the message; the integration developer, who creates the integration flow and deploys it on the tenant; and, finally, the receiver, who decrypts the message manually.

[Figure 10.30](#) shows the involved components and keys.

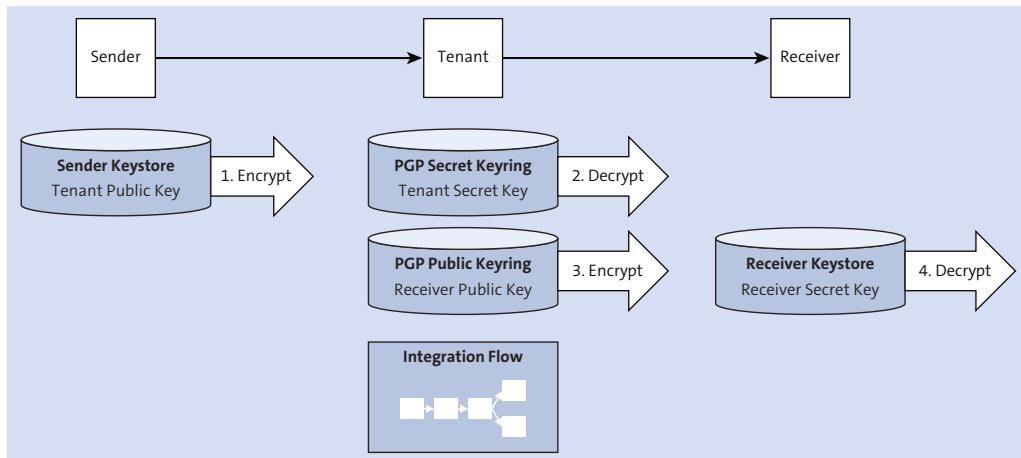


Figure 10.30 Components of the Demo Example

As already mentioned, when using OpenPGP, the required keys for the tenant are to be stored separately in a PGP secret keyring and a PGP public keyring, respectively. Both entities must be separately deployed on the tenant.

This is how it works:

- The sender uses a public key (which is associated with the tenant's secret key) to encrypt the message (manually).
- The tenant (in a **Decryptor** step) decrypts the message using its own secret key (from the PGP secret keyring), and thereafter encrypts it again (within an

Encryptor step) using the public key (from the PGP public keyring), which is associated with the receiver's secret key.

- Finally, the receiver receives the encrypted message and decrypts it (manually) using his own secret key.

As explained earlier, we start with the following integration flow that only contains a **Decryptor step** (Figure 10.31).

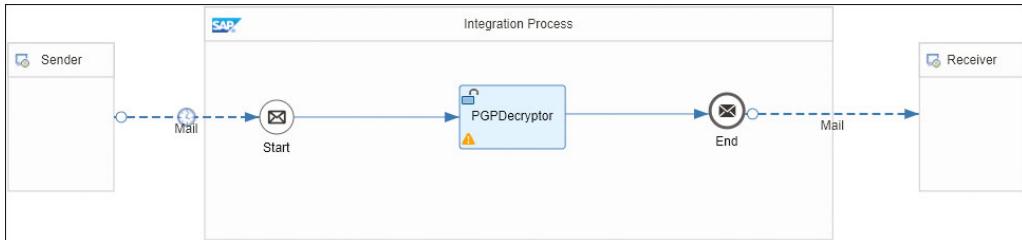


Figure 10.31 Integration Flow with Decryptor Step and Email Sender and Receiver

As a prerequisite, install the free software Kleopatra for key management. Download the software (www.gpg4win.de/index.html). When asked, make sure that the software package to download includes Kleopatra.

To start setting up the scenario, let's first take the role of the integration developer, who creates the private-public key pair (using Kleopatra) and deploys it as a PGP secret keyring on the tenant. The secret key is used by the decryptor step to decrypt the encrypted message.

1. Open Kleopatra.
2. Under **File**, choose **New Key Pair** (see Figure 10.32).



Figure 10.32 Creating a New Key Pair with Kleopatra

3. In the following dialog, click **Create a Personal OpenPGP Key Pair**.
4. Enter a name (e.g., "Tenant", to indicate that this is the tenant's key pair) and email address, and click **Next**.

5. Select **Create**.
6. Enter a passphrase to protect the key. You need to remember the passphrase for a later step when deploying the PGP secret keyring on the tenant.
7. Choose **Finish**. The newly created key pair is shown in a list ([Figure 10.33](#)).

Name	E-Mail	User-IDs	Valid From	Valid Until	Details	Key-ID
Tenant	admin@cpi.com	certified	25.02.2018		OpenP... 604AAFBB	

Figure 10.33 The New Key Pair Shown in a List

8. Select the key (**Tenant**), and in the context menu, choose **Export Secret Keys** (see [Figure 10.34](#)).
9. Save the key on your computer (as a *.gpg* file).

Figure 10.34 Exporting Secret Keys from Kleopatra

You now need to deploy the key pair generated with the preceding steps on the tenant:

1. Open the **Monitor** application of the Web UI, and under **Manage Security**, click the **Security Material** tile.
2. Select **Add • PGP Secret Keyring**, and choose **Next** (Figure 10.35).

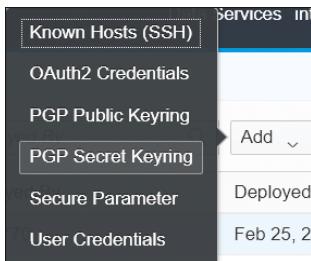


Figure 10.35 Adding a PGP Secret Keyring Artifact

3. Browse for the secret key *.gpg* file on your local disk, enter the secret key passphrase, and choose **Deploy**. The newly deployed artifact is shown in the **Security Material** overview (Figure 10.36).

Overview / Manage Security Material		
Security Material (9)		
Name	Type	Status
secring	PGP Secret Keyring	Deployed
Twitter_TokenSecret	Credentials	Deployed
Twitter_AccessToken	Credentials	Deployed

Figure 10.36 The Newly Deployed Artifact Displayed in the Artifact Overview

Now switch to the role of the sender to manually encrypt the clear text with the tenant's public key. In real-life scenarios, this is performed automatically by the software deployed on the sender system. Follow these steps:

1. To encrypt your text, start Kleopatra, and then enter the clear text (e.g., "Paul Smith") in a text editor.
2. Copy the text to the clipboard.
3. In Kleopatra, choose **Clipboard • Encrypt** (Figure 10.37).

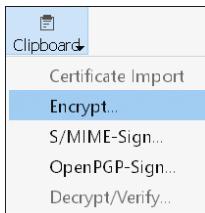


Figure 10.37 Choosing the Encrypt Option for the Clipboard in Kleopatra

4. Keep the option **OpenPGP** selected, and click **Add Recipient**.
5. Imagine that you, as a sender, communicate with many recipients who have shared their public keys with you. In this step, select the key from that recipient (the tenant, in our case) to whom you'd like to send an encrypted message (see [Figure 10.38](#)).

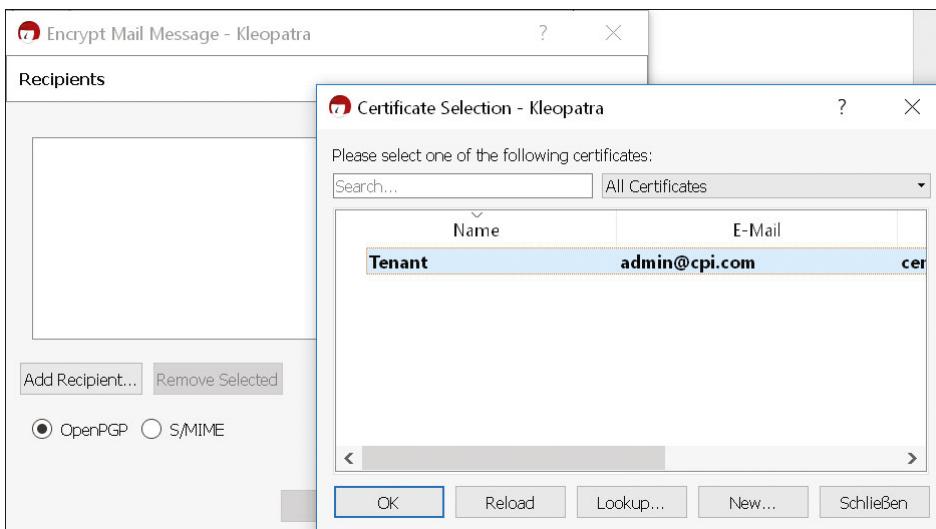


Figure 10.38 Selecting the Key to Be Used to Encrypt the Clear Text

6. Select the key/recipient, and choose **OK**.
7. When the message **Encryption Succeeded** is displayed, click **OK** ([Figure 10.39](#)).
8. Paste the text from the clipboard to a text editor. The encrypted message will look like what you see in [Figure 10.40](#).

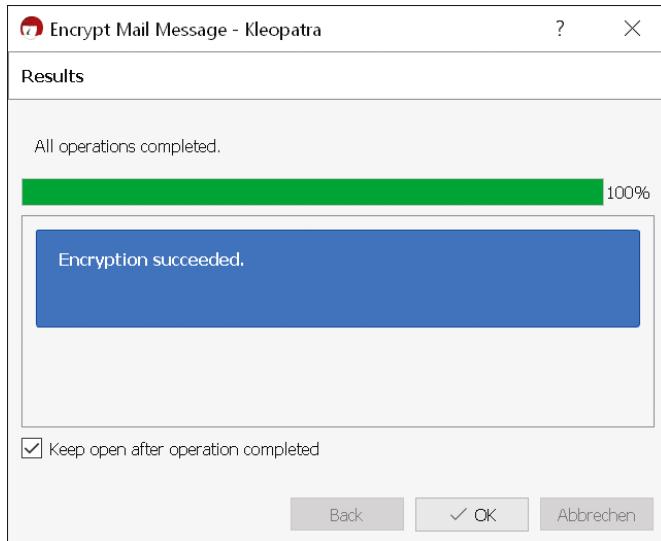


Figure 10.39 Success Message When Kleopatra Has Encrypted the Clear Text

```
-----BEGIN PGP MESSAGE-----
hQEMA8EPNLAufHwdAQf/YYGQ/J04E21YpFQwi2v5gSdQOJFm0aLYOrmuAsfy7q6E
87xng9mMv0S1E9vnxEz/eAwUUi5DcX7Z0oVBZ6f4H9TwWaCJM7wHjMGYhYZ+Co/4
0C22E8tv+GmAUPjEW94ur0TpVyzXKhtZvELFtSB9GkrZk/sd+0nLcc+j02YUfGpB
fAdqg3U55QIZGxfTmtiBZP3Wi6oN9IbOfHgw4n+tXmrKhmcCoxTlwV0TCpYyV08x
W0m+YwJION+ApuwSBatS+THbrUwXsMMVVUMUVi50giLXA4dfalkoXzh82AM50Z5z
5Cxi973RNNpp931308CGZMFZxrmvRx0Td5K20ZRutJFAaJgQbCgNkXbH94hAmha
uT+lq6V+yqdp0gLfkN3i3YFmZVU1yQWZNdDDn9aYCT7Um62Ro0She6S39tH1VIWE
ACEJNmq3
=gqtZ
-----END PGP MESSAGE-----
```

Figure 10.40 The Encrypted Message (Example)

9. To finish the sender tasks, open the email account from which SAP Cloud Platform Integration should poll emails, create an email, and paste the encrypted message (the text block shown in [Figure 10.40](#)) into the email body.
10. Move the email into the folder (e.g., with the name, *CPI_DEMO*) of your email account from which SAP Cloud Platform Integration should read it (corresponding to the mail sender adapter settings as shown later).

Now let's take the role of the integration developer. We propose that you create a simple integration flow, as shown earlier in [Figure 10.31](#).

Throughout the previous chapters, you've already become an integration development expert, and, as such, we won't go into minute detail again. We'll only briefly summarize the required steps that are specific to this demo example.

Let's propose that the **Mail** sender adapter should poll unread emails from a specific folder (e.g., *CPI_DEMO*) of the receiver email account. If you've already configured a scenario with a *Mail* sender adapter when reading [Chapter 2](#), you can simply reuse the scenario and enhance it with the *Decryptor* step as shown in the following steps. For those who didn't (as we only showed this as a further variant of your first integration flow), we'll show how to create and configure the *Mail* sender adapter. For more information on the mail sender adapter settings, refer to the *Mail Sender Adapter* info box in [Chapter 2, Section 2.3.8](#). Follow these steps:

1. Create a new integration flow, choose the **Edit** mode, and select the **Sender** pool ([Figure 10.41](#)).



Figure 10.41 Connecting the Sender Pool with the Start Message Event to Create the Mail Sender Adapter

2. Click the arrow icon, and drag and drop the cursor to the **Start** message event.
3. As **Adapter Type**, select **Mail**, and as **Transport Protocol**, select **IMAP4**.
4. Open the **Connection** tab.
5. For **Address**, enter the host name of your email provider, followed by the port. Remember that for the mail sender adapter, when using the IMAP4 protocol, only ports 143 or 993 are allowed. We propose to use the same settings as shown in [Figure 10.42](#); in particular, for the **Authentication** option, choose **Plain User/Password** and specify a credential name (in the example, "FirstnameLastname" as already used in the integration flow in [Chapter 2, Section 2.3](#)). We've also shown in [Chapter 2, Section 2.3.5](#), how to create the corresponding **User Credentials** artifact, which contains the user name and password for the email account.

[Figure 10.42](#) shows the settings when using Gmail.

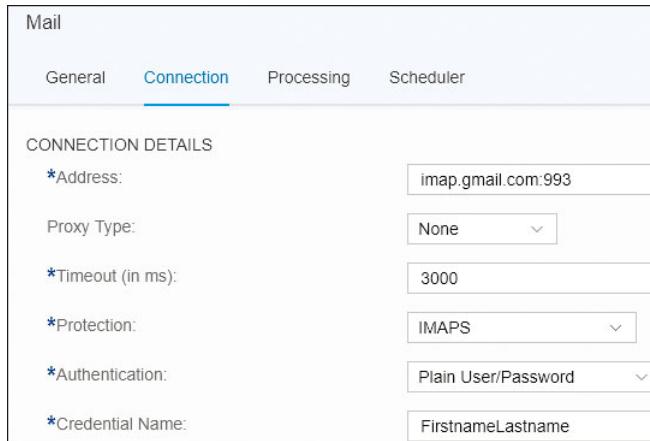


Figure 10.42 Connection Tab of the Mail Sender Adapter

6. Go to the **Processing** tab of the **Mail** sender adapter. Here, you specify how emails should be processed by SAP Cloud Platform Integration.

Figure 10.43 shows example settings to configure the following behavior at run-time: SAP Cloud Platform Integration reads all unread mails (maximum 20 messages per poll) from folder *CPI_DEMO* of your email account. After that, the emails are marked as read (so that with the next poll they aren't read again).

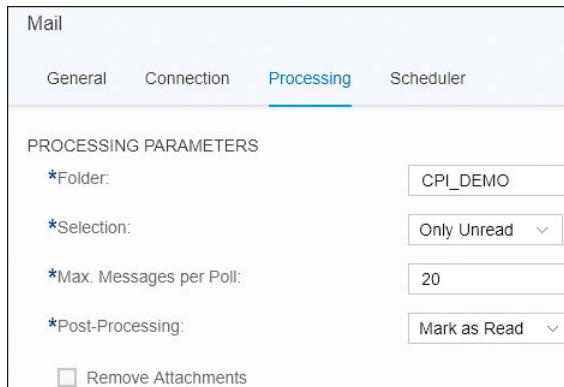


Figure 10.43 Processing Tab of the Mail Sender Adapter

7. Go to the **Scheduler** tab to configure how often emails should be polled. Note that the **Mail** sender adapter contains the same scheduler capabilities as the **Start Timer** event, which was explained in detail in Chapter 6, Section 6.1. We won't

repeat this here. However, we would like to urge you to be careful with these settings and not to spam your email account with the final integration flow. To be on the safe side, you might configure the scheduler so that emails are only polled once (as shown in [Figure 10.44](#)).

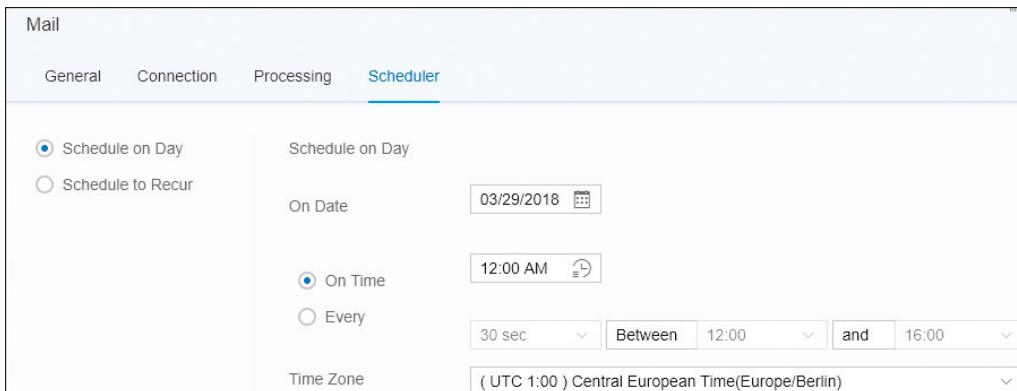


Figure 10.44 Scheduler Tab of the Mail Sender Adapter

8. To add a **PGP Decryptor** step, select the **Security Elements** icon in the palette, and choose **Decryptor • PGP Decryptor** ([Figure 10.45](#)).



Figure 10.45 Adding a PGP Decryptor to the Integration Flow

9. In the settings of the **Decryptor** step, for the **Signatures** field, specify **None Expected**. No further entries are required ([Figure 10.46](#)).



Figure 10.46 Settings of the Decryptor Step (When No Signatures Are Expected)

10. Finally, add the **Mail** receiver adapter to the integration flow.

Use the same settings as shown in [Chapter 2, Section 2.3.3](#).

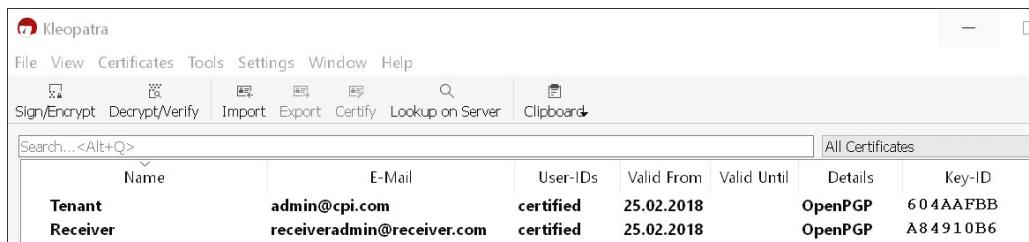
11. Save and deploy the integration flow.

The integration flow will now poll emails as specified in the **Mail** sender adapter. Before we continue to add the **Encryptor** step, let's perform a brief check, and then deploy and run the integration flow that we've built so far.

Let's slip into the role of receiver and open the email account addressed by the mail adapter. When you've selected the appropriate settings in the **Scheduler** tab of the **Mail** sender adapter, you should have received an email with the customer name in clear text (e.g., **Paul Smith**).

To continue adding encryption to the scenario, we first need to create another key pair: the receiver's key pair. Remember that the tenant needs the receiver's public key to encrypt a message. To generate the key pair, use Kleopatra in the same way shown previously for the tenant's key pair. It makes sense to enter "Receiver" in the **Name** field (see [Figure 10.47](#)). Then follow these steps:

1. In Kleopatra, select the newly created key pair (**Receiver**), and choose **Export** in the context menu ([Figure 10.48](#)).



The screenshot shows the Kleopatra interface with a list of certificates. A new entry for 'Receiver' has been added, showing details such as E-Mail, User-IDs, Valid From, Valid Until, Details, and Key-ID.

Name	E-Mail	User-IDs	Valid From	Valid Until	Details	Key-ID
Tenant	admin@cpi.com	certified	25.02.2018		OpenPGP	604AAFB
Receiver	receiveradmin@receiver.com	certified	25.02.2018		OpenPGP	A84910B6

Figure 10.47 Kleopatra Key List Showing the Newly Created Receiver Key Pair



Figure 10.48 Exporting the Receiver's Public Key

2. Save the key on your computer (change the file extension to *.gpg*).

Switch to the role of the integration developer, and follow these steps:

1. Deploy the new public key on the tenant as the **PGP Public Keyring** artifact.
2. Open the **Monitor** application of the Web UI, and under **Manage Security**, click the **Security Material** tile.
3. Choose **Add • PGP Public Keyring**, as shown in [Figure 10.49](#).

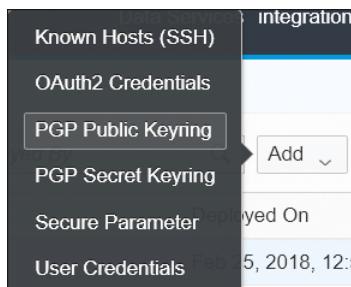


Figure 10.49 Adding a PGP Public Keyring Artifact

4. Browse for the public key *.gpg* file on your local disk, and choose **Deploy**. The artifact is added to the list of deployed artifacts as shown in [Figure 10.50](#).

Overview / Manage Security Material		
Security Material (10)		
Name	Type	Status
pubring	PGP Public Keyring	Stored
secring	PGP Secret Keyring	Deployed
Twitter_TokenSecret	Credentials	Deployed

Figure 10.50 The Newly Deployed PGP Public Keyring Shown in the List of Integration Artifacts

5. To complete the integration flow, add an **Encryptor** (OpenPGP) step to the integration flow.
6. Open the integration flow in the Web UI, click the **Security Elements** icon in the palette, and choose **Encryptor • PGP Encryptor** ([Figure 10.51](#)).



Figure 10.51 Adding a PGP Encryptor Step to the Integration Flow

7. Specify the properties of the **Encryptor** step (Figure 10.52). In the **Encryption User ID of Key(s) from Public Keyring** field, enter the user ID of the related key. To find the user ID, open Kleopatra, select the relevant key used for encryption, and check for the **Name** of the key. In our example, this is the **Receiver** key. For the other parameters, you can leave the default settings.

The screenshot shows the configuration dialog for the PGP Encryptor step. It includes sections for General Details, Advanced Options, and a preview pane. The General Details section contains fields for Name (PGPEncryptor1), Signatures (None), Content Encryption Algorithm (AES), Secret Key Length (128), Compression Algorithm (ZLIB), and checkboxes for Armored and Integrity Protected Data Packet. The Advanced Options section includes a dropdown for Encryption Key User IDs, which lists 'Encryption User ID of Key(s) from Public Keyring' and 'Receiver'.

Figure 10.52 Parameters of the PGP Encryptor to Be Specified When No Signatures Are Included

8. Save and deploy the integration flow.

In the receiver mailbox, you should (depending on the scheduler settings in the mail sender adapter) find an email that contains the encrypted customer name. This looks like Figure 10.40, shown earlier, but the string sequence will be different.

To simulate the receiver's decryption step, follow these steps:

1. Open Kleopatra, and copy the encrypted mail text to the clipboard.
2. Choose **Clipboard • Decrypt/Verify** (refer to [Figure 10.37](#)).
3. As opposed to the encryption case, you don't have to select a key here (there is no **Add Recipient** option) because the receiver's private key is uniquely determined. You need to enter the passphrase for the private key.
4. When you paste the contents of your clipboard to a text editor, you should see the clear text (e.g., **Paul Smith**).

We've now shown how you can test the message encryption and decryption capabilities of SAP Cloud Platform Integration through very simple means. For simplicity, we skipped its signing and verification capabilities.

When you open the **Encryptor** step in the sample integration flow, you also have the option to include signatures (by choosing **Including** as the value for **Signatures**). In this case, you now have to specify additional parameters to define how the message is to be signed ([Figure 10.53](#)):

■ **Digest Algorithm**

This is the algorithm with which the digest (or hash value) is to be calculated out of the message content (refer to [Figure 10.27](#)).

■ **Signer User ID of Key(s) from Secret Keyring**

In addition to the user IDs that have to be specified to find the correct encryption (public) key from the PGP public keyring, in the message signing case, you need to find the correct private key from the PGP secret keyring. Remember that the tenant uses its private key to sign the message.

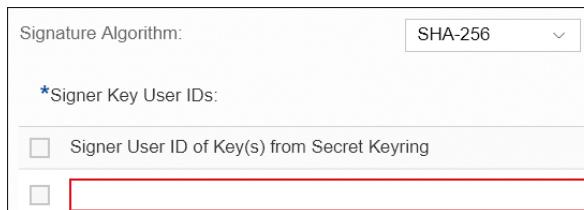


Figure 10.53 Additional Parameters When Including Digital Signatures

To finish this section, note that decrypting a message on the tenant (and encrypting it again before sending it to a receiver) usually only makes sense if you want to perform additional steps (e.g., CBR or mapping) that rely on the original unencrypted

message. In our demo, however, we skip such additional processing steps for simplicity (see [Figure 10.54](#)).

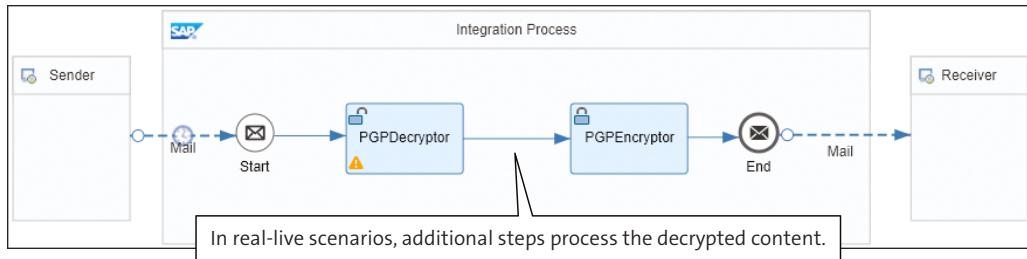


Figure 10.54 Integration Flow Steps That Decrypt an Inbound Message, Modify the Content, and Then Encrypt It Again

SFTP Adapter

Next to the mail sender adapter, SAP Cloud Platform Integration provides another polling adapter, the SFTP sender adapter.

You can modify this scenario by replacing the sender email server with an SFTP server. In this case, you need to store a file on the SFTP server with the encrypted message so that the SFTP sender adapter can poll it. For more information on how to set up a secure connection using SFTP, read the “How to Set Up Secure Connection to an SFTP Server” blog in the SAP Community (www.sap.com/community.html).

10.5 Keystore Management

While reading this book and this chapter in particular, you’ve become familiar with various kinds of security artifacts. Namely, we’ve shown you when you need to define **User Credentials** artifacts ([Chapter 2, Section 2.3.5](#)), **Certificate-to-User Mappings** artifacts ([Section 10.4.4](#)), **Secure Parameter** artifacts ([Section 10.4.5](#)), and **PGP Secret Keyring/PGP Public Keyring** artifacts ([Section 10.4.6](#)).

All aspects about managing these artifacts during the operation of an integration scenario were explained in detail in [Chapter 8, Section 8.3.1](#).

We’ll now elaborate on the **Keystore** artifact that was already introduced in [Section 10.4.4](#) when talking about how to set up secure connections based on HTTP.

10.5.1 Using X.509 Security Material for SAP Cloud Platform Integration

The **Keystore** artifact allows you to manage the content of the tenant keystore. This keystore can contain key material (private/public key pairs and certificates) based on the X.509 standard ([Section 10.4.1](#)) as well as SSH keys.

SSH Keys

You can also use the tenant keystore to manage and store SSH keys that are required when configuring secure connections using SSH with the SFTP adapter, a topic which we haven't explained further in this book. For more information, read the "How to Set Up Secure Connection to an SFTP Server" blog in the SAP Community (www.sap.com/community.html).

Let's focus on X.509 keys. [Table 10.10](#) summarizes in which situations you use X.509 keys and, therefore, need to consider how to deal with the tenant keystore.

Usage	Description
Certificate-based authentication of communication partners when communicating over HTTPS (transport-level security)	Certificates used in such scenarios should be signed by a trusted authority. We've explained this concept in Section 10.4.1 .
Signing/verifying and encrypting/decrypting messages using the PKCS#7 standard (message-level security)	<p>You configure these scenarios by using PKCS7 Signer, PKCS7 Encryptor, PKCS7 Signature Verifier, or PKCS7 Decryptor steps in the integration flow (Section 10.4.6).</p> <p>Certificates used in such scenarios can be self-signed.</p>
Signing/verifying a SOAP body, encrypting/decrypting message content using WS-Security	<p>Certain adapters support dedicated options to sign/verify messages or to encrypt/decrypt message content based on the WS-Security standard, which is an extension to SOAP.</p> <p>The following adapters support these options:</p> <ul style="list-style-type: none"> ■ SOAP 1.x sender and receiver adapter ■ AS2 sender and receiver adapter <p>To implement such scenarios, X.509 keys are also required in the tenant keystore.</p>

Table 10.10 Using X.509 Keys in SAP Cloud Platform Integration

X.509 Keys for Transport-Level Security

As explained in [Section 10.4.3](#), SAP Cloud Platform Integration supports various authentication options, including client certificate authentication. When you choose this authentication option for outbound connections, the tenant keystore needs to contain a client certificate, which is a signed private key pair. When you get your tenant provided by SAP, SAP provides you initially with a keystore that contains one such private key pair, which has the alias `sap_cloudintegrationcertificate` (see the upcoming info box). You can use this key pair to set up an outbound HTTP connection immediately.

As shown in the very first integration flow in [Chapter 2](#), the **Mail** receiver adapter doesn't offer any client certificate authentication option. Instead, authentication is accomplished based on user credentials.

Recall that you had to create and deploy a **User Credentials** artifact to specify these credentials. Nevertheless, we needed to import certain certificates into the tenant keystore because the connection also needed to be protected *the other way around*: the email server needed to set up a trust relationship to the tenant. Therefore, we had to import a root certificate depending on the requirements of the email server. We referred to this certificate as the *receiver server root certificate*. When discussing the setup to establish a secure outbound connection from the tenant to a receiver system, we've already shown how the different security artifacts come into play in the whole picture ([Section 10.4.4](#)). This example showed you that the tenant keystore isn't only required to specify key pairs for client certificate authentication in outbound connections.

To find out more about the keystore content in different security setups, check out the online documentation of SAP Cloud Platform Integration at https://help.sap.com/viewer/product/CLOUD_INTEGRATION/Cloud and choose **Connecting a Customer System to Cloud Integration**.

X.509 Keys for Message-Level Security (PKCS#7 Standard)

As summarized in [Table 10.10](#), the keystore can also contain keys to set up scenarios that include message-level security based on the PKCS#7 standard. As explained in [Section 10.4.1](#), different keys are required to sign and encrypt/verify and decrypt a message. Let's briefly repeat this from the perspective of the tenant:

- To enable a tenant to encrypt a message (using the **PKCS7 Encryptor** step) and, correspondingly, to enable the receiver to decrypt the message, the receiver of the

message needs to generate an X.509 private key pair and share the public part of it with the tenant administrator. The tenant administrator then imports the public key into the tenant keystore. At runtime, the tenant encrypts the message using the public key, and the receiver uses the private key to decrypt it.

- To enable a tenant to decrypt a message encrypted by a sender system (using the **PKCS7 Decryptor** step), the tenant administrator needs to generate an X.509 private key pair and share the public part of it with the administrator of the sender system. At runtime, the sender system uses the public key to encrypt the message, and the tenant uses the private key to decrypt it.

For signing and verifying messages using the **PKCS7 Signer** and **PKCS7 Signature Verifier** steps, the situation is mirror-inverted.

We've shown in [Section 10.4.6](#) how to set up a scenario with encryption and decryption using PGP keys. The keys owned by the tenant in such a case need to be stored in dedicated key storage locations, namely, the **PGP Public Keyring** and the **PGP Secret Keyring**. When you want to set up the same kind of scenario using PKCS#7, you need to use the tenant keystore (**Keystore** artifact) to store both the private and the public keys. Besides that, the same principles apply.

To set up a scenario as summarized earlier in [Figure 10.30](#), but using PKCS#7 keys, you need to establish the security configuration shown in [Figure 10.55](#).

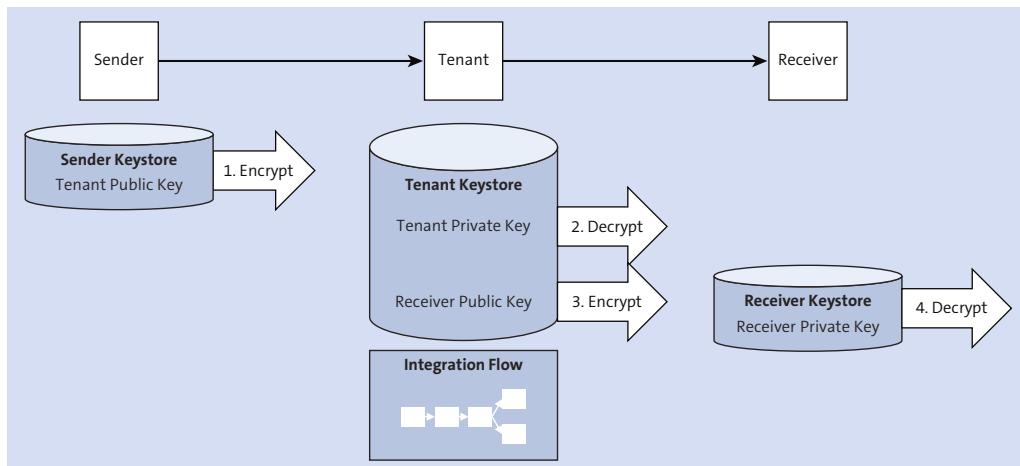


Figure 10.55 Key Setup for Encryption/Decryption Scenario Using PKCS#7 Security Standard

To set up such a scenario, the tenant keystore needs to contain the following:

- One private key pair (**Key Pair** entry)
- One **Certificate** entry that contains the public key of the receiver system (shared by the receiver administrator)

Furthermore, the integration flow uses the following steps:

- **PKCS7 Decryptor**

This uses the private key of the tenant to decrypt the message obtained from the sender.

- **PKCS7 Encryptor**

This uses the public key of the receiver system as specified by the **Receiver Public Key Alias** in the step.

Initial Content of the Tenant Keystore

When you get a tenant from SAP the first time, SAP also provides you with a tenant keystore that contains as initial content the following SAP-owned entries:

- Some SAP-owned CA root certificates that enable you to set up the communication with SAP cloud systems, such as SAP Ariba and SAP SuccessFactors (e.g., with the alias `sap_baltimore cybertrust root` and `sap_digicert global root ca`)
- One signed private key pair (alias: `sap_cloudintegrationcertificate`)

In [Section 10.5.2](#), you'll learn more about how to manage keystore entries, and, in [Section 10.5.3](#), you'll learn in particular how to deal with the lifecycle of SAP-owned keystore entries.

10.5.2 Managing Security Material in the Tenant Keystore

You get access to the tenant keystore in the **Monitor** application of the Web UI (**Keystore** tile under **Manage Security**). The basic functions have already been explained in [Chapter 8, Section 8.3.2](#), in the context of SAP Cloud Platform Integration operations, so we won't repeat this here. However, to remind you of the content, we'll briefly summarize the most important aspects and provide additional details on the content of the keystore. In [Section 10.5.3](#), we'll then show how to manage the lifecycle of keystore entries.

[Figure 10.56](#) shows an excerpt of the tenant keystore.

The screenshot shows a table titled 'Entries (43)' with columns for Alias, Type, Owner, Created At, and Updated At. There are also buttons for Filter by Alias, Back Up, Add, Download, and a refresh icon.

Alias	Type	Owner	Created At	Updated At
verisign class 3 secure server ca - g3 (verisign class 3 public primary certification authority - g5)	Certificate	Tenant Administrator	Feb 08, 2020, 00:59:59	Mar 10, 2018, 13:06:40
verisign class 3 secure server ca - t1 (verisign class 3 public primary certification authority - g5)	Certificate	Tenant Administrator	May 13, 2020, 01:59:59	Mar 10, 2018, 13:06:40
verisign universal root certification authority	Certificate	Tenant Administrator	Dec 02, 2037, 00:59:59	Mar 10, 2018, 13:06:40
sap_cloudintegrationcertificate	Key Pair	SAP	Jun 07, 2019, 01:59:59	Mar 10, 2018, 10:36:04

Figure 10.56 Keystore Entries

The keystore can contain entries owned by SAP and those owned by the tenant administrator. Those owned by SAP are indicated by a lock icon and can't be changed by the tenant administrator. In [Figure 10.56](#), you'll notice the key pair with the alias **sap_cloudintegrationcertificate**, initially provided by SAP as mentioned in the previous info box.

The keystore contains entries of two different types:

■ **Key Pair**

Consists of a private key pair and (commonly) an X.509 certificate chain, unless it's an SSH key, which can be identified by the alias `id_rsa` or `id_dsa`.

■ **Certificate**

Represents an X.509 certificate (in many cases, a root certificate).

As an example for a **Key Pair** entry, let's take a look at the SAP-owned key pair with alias **sap_cloudintegrationcertificate**. You can access the details of a keystore entry by clicking the link in the **Alias** column ([Figure 10.57](#)).

The left side of the figure shows the certificate chain. Note that for **Key Pair** entries, it's common to define them as part of a certificate chain. On top of the certificate chain of the key pair provided by SAP, you notice the root certificate of the CA **VeriSign**.

The details of the **Key Pair** entry are shown on the right side. Note that you can navigate to the details of the intermediate or root certificate by clicking on the corresponding nodes in the certificate chain on the left side; the details on the right then adapt accordingly. The following list describes a few of the attributes here.

The screenshot shows the SAP Cloud Integration Keystore Management interface. The top navigation bar includes 'Overview / Manage Keystore / sap_cloudintegrationcertificate'. A 'Download' button is in the top right. Below the navigation, a 'Certification Path' tree shows two entries: 'VeriSign Universal Root Certification Authority' and 'Symantec Class 3 Secure Server SHA256 SSL CA'. The 'Symantec' entry is expanded, showing its details. The 'General' tab is selected. Key details shown include:

- Alias:** sap_cloudintegrationcertificate
- Type:** Key Pair
- Version:** 3
- Subject DN:** CN=cloudintegcert.eu1.hana.ondemand.com, OU=SAP Trust Community, O=SAP SE, L=Waldorf, ST=Baden Württemberg, C=DE
- Issuer DN:** CN=Symantec Class 3 Secure Server SHA256 SSL CA, OU=Symantec Trust Network, O=Symantec Corporation, C=US
- Key Type:** RSA
- Key Size:** 2048
- Signature Algorithm:** SHA256withRSA
- Valid From:** Jun 05, 2017, 02:00:00
- Valid Until:** Jun 07, 2019, 01:59:59

A 'Fingerprints' section lists the SHA-1 and SHA-256 fingerprints.

SHA-1:
33:9E:70:11:04:D5:61:F7:C1:D0:E6:BB:37:9D:6C:D4:BC:2B:42:63

SHA-256:
D3:14:B1:0F:A1:5F:37:EA:1C:A7:61:7C:9F:7E:68:7D:7B:AF:78:1D:3F:67:E1:65:FE:C2:AE:19:4F:25:EA:14

Figure 10.57 SAP-Owned Key Pair Entry

Note

To find more information on the constituents of an X.509 certificate, check out the X.509 standard as documented in the *Request for Comments 4158* at <https://tools.ietf.org/html/rfc4158>.

■ Alias

The alias is the name by which you can uniquely identify a keystore entry.

When to Refer to an Alias (Examples)

In many cases, you need to specify the alias to tell the system which dedicated keystore entry to use for a certain integration flow step or adapter. We provide two examples:

- When configuring a **PKCS7 Encryptor**, you need to specify the **Receiver Public Key** alias to select the public key that is to be used to encrypt the message for a certain receiver.

- When you configure an **HTTP** receiver adapter and choose **Authentication** option **Client Certificate**, you have the option to specify a **Private Key Alias** to point to that private key pair that is intended to be used to authenticate the tenant when calling a receiver.

- **Serial Number**

This entry is used by the CA to uniquely identify the certificate (within the CA's organization).

- **Subject DN**

The DN of the subject uniquely identifies the entity that is associated with the key-store entry. The DN is composed of a number of attributes that you need to specify when generating the key pair. These attributes comprise the common name (CN), which typically contains the server name of the VM associated with the tenant, and further information to identify the organization associated with the key pair such as Organization (O), Organizational Unit (OU), Country (C), and other attributes (in the example shown in the figure, information about the SAP location in Germany).

- **Issuer DN**

The DN of the issuer of the certificate identifies the authority that issued and signed the certificate (also consisting of several entries such as a Common Name, Organizational Unit, etc.).

- **Signature Algorithm**

This is the algorithm used by the issuer to sign the certificate.

Furthermore, the validity period is displayed (see also [Section 10.5.3](#)).

Note that as it's an SAP-owned entry, so you can only download the entry (**Download** button at the top); no further (changing) actions are possible.

Under **Fingerprints**, you find hexadecimal expressions of hash values calculated out of the public key (using different hash algorithms, e.g., SHA-1 or SHA-256). The tenant administrator can use the fingerprint to verify the trustworthiness of a keystore entry by sharing it with the related communication partner (e.g., the administrator of the connected sender or receiver system) via an independent secure communication channel such as encrypted email.

[Figure 10.58](#) shows a **Certificate** entry owned by the tenant administrator.

Overview / Manage Keystore / verisign universal root certification authority

Certification Path

verisign universal root certification authority	General	Fingerprints	Administration
Alias: verisign universal root certification authority	Type: Certificate	Keystore: Current	
Version: 3	Owner: Tenant Administrator	Serial Number: 0x401AC46421B31321030EBBE4121AC51D	
Subject DN: CN=VeriSign Universal Root Certification Authority, OU=(c) 2008 VeriSign, Inc. - For authorized use only", OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US			
Issuer DN: CN=VeriSign Universal Root Certification Authority, OU=(c) 2008 VeriSign, Inc. - For authorized use only", OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US			
Key Type: RSA	Key Size: 2048	Signature Algorithm: SHA256withRSA	
Valid From: Apr 02, 2008, 02:00:00	Valid Until: Dec 02, 2037, 00:59:59		

Fingerprints

SHA-1:
36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54

SHA-256:
23:99:56:11:27:A5:71:25:DE:8C:EF:EA:61:0D:DF:2F:A0:78:B5:C8:06:7F:4E:82:82:90:BF:B8:60:E8:4B:3C

Figure 10.58 Certificate Entry Owned by the Tenant Administrator

As you see on the left side, this is a root certificate as only one node of a certificate chain is shown.

The attributes have the same meaning as discussed for the **Key Pair** entry after [Figure 10.57](#). Note, however, that **Subject DN** and **Issuer DN** are identical in this case, as the CA is both acting as certificate issuer and as entity associated with the certificate.

[Figure 10.58](#) shows a keystore entry owned by the tenant administrator. Therefore, further actions are supported, such as deleting it or renaming it (changing the alias), as you can see from the **Delete** and **Rename** buttons at the top.

Keystore monitors have been explained in detail in [Chapter 8, Section 8.3.1](#). We'll now focus on the aspect of lifecycle management of security material.

10.5.3 Managing the Lifecycle of Keys Provided by SAP

To increase the security level of integration scenarios, digital certificates used to protect connections have a restricted validity period. Therefore, expiring certificates need to be renewed in regular intervals—in the same way that it's a good idea to change passwords from time to time.

However, keys are a fundamental part of the setup of each integration scenario, and each change in the setup of security material might require certain adaptations in the scenario setup and might disrupt the scenario operations, unless it's done in a well-defined and orchestrated manner.

Therefore, it's of critical importance to manage the lifecycle of keys in a smart way. Namely, as tenant administrator, you need to take certain actions when a key is due to expire and replace it with a new one. You should organize such a task so that there is ideally no or, if it can't be avoided, only a minimal downtime for your integration scenarios. General guidelines of how to renew security material for the various use cases supported by SAP Cloud Platform Integration are documented in the online documentation for SAP Cloud Platform Integration (https://help.sap.com/viewer/product/CLOUD_INTEGRATION/Cloud) under **Operating and Monitoring Cloud Integration • Security Artifact Renewal**.

We'll now focus on the special case when SAP-owned key material is renewed by SAP. SAP renews key pairs regularly (by default, the validity period is two years), and to enable the tenant administrator to manage renewal of SAP-owned keys, there are certain options available. What the tenant administrator is supposed to do in such a case is activate the updated key in the tenant keystore. However, this has to be done in a coordinated manner because other components (i.e., the remote systems that are connected to the tenant and that use the public part of the SAP key pair in their keystores) are affected by such a step.

Renewal of SAP-Owned Root Certificates

As already explained in [Chapter 8, Section 8.3.2](#), there is also the option that SAP updates SAP-owned root certificates. In such a situation, there is no need for the tenant administrator to take any action. After the certificate update, the new certificate is active immediately in the keystore.

The **Monitor** application of the Web UI provides certain options to give the tenant administrator full control over the activation of keys that are updated by SAP. In particular, as already shown in [Chapter 8, Section 8.3.2](#), the **Monitor** application provides four different tabs for the tenant keystore:

■ **Current**

Shows the content of the keystore as currently deployed on the tenant (and actively used in your integration scenarios).

- **Backup**
Shows backed-up keystore entries (see [Chapter 8, Section 8.3.2](#)).
- **New SAP Keys**
Shows new keys provided by SAP in case a dedicated SAP-owned key is due to expire (see upcoming discussion).
- **SAP Key History**
Shows recently renewed SAP keys and provides the option to move a key back to the **New SAP Keys** store.

We won't go through all options of keystore management here, but you can refer to [Chapter 8, Section 8.3.2](#). Instead, we focus on the topic of key activation and show how the tenant administrator deals with the situation when SAP updates an SAP-owned key pair. Furthermore, to illustrate the aspect that there are other components affected by key renewal, we assume (as an example) that the SAP key pair is used in a scenario where a sender sends an encrypted message to SAP Cloud Platform Integration, and the message is decrypted by a **PKCS7 Decryptor** step on the tenant. [Figure 10.59](#) shows where the SAP key pair (the tenant private key pair) is involved.

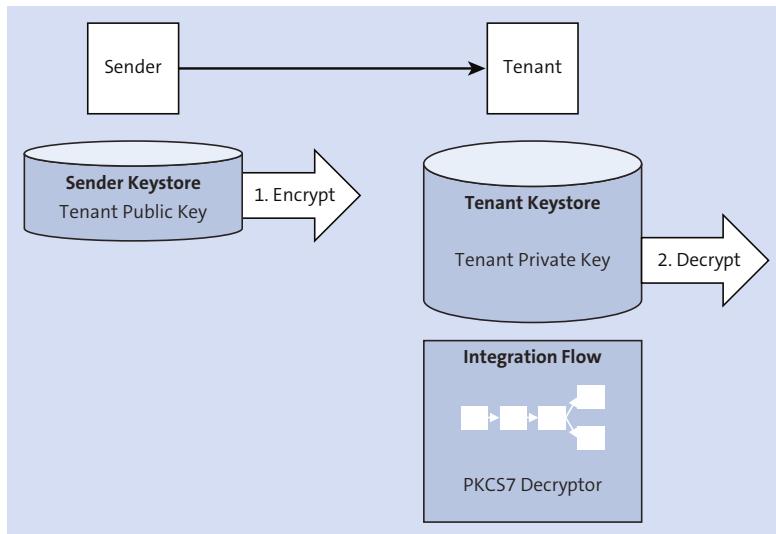


Figure 10.59 Using a Tenant Private Key Provided by SAP in a PKCS7 Decryptor Step

Such a situation requires a coordinated procedure. Let's walk through the procedure of activating the new key now:

1. SAP provides a new key pair for one that is due to expire soon.
2. As tenant administrator, you'll find the new key pair in the **Monitor** application of the Web UI under **Manage Keystore** in the **Keystore** tile in the **New SAP keys** tab ([Figure 10.60](#)).

Entries (1)					<input type="text" value="Filter by Alias"/>			
Alias	Valid From	Valid Until	Last Modified At	Actions				
sap_cloudintegrationcertificate	Jun 05, 2017, 02:00:00	Jun 07, 2019, 01:59:59	Mar 10, 2018, 13:13:21					

Figure 10.60 New Key Provided by SAP Visible in New SAP Keys Tab of the Manage Keystore Monitor

3. Analyze the active integration flows and remote connections. Let's assume you find out that one integration scenario uses a **PKCS7 Decryptor** step, which uses this key pair (as shown earlier in [Figure 10.59](#)). Therefore, you notice that before activating the new key pair, you need to make sure the sender system that sends the corresponding encrypted message gets the public key of the newly provided SAP key pair.
4. Download the certificate from the **New SAP Keys** monitor by selecting the new SAP key (in our example, with the alias `sap_cloudintegrationcertificate`) and choosing **Download Certificate** under **Actions** ([Figure 10.61](#)).

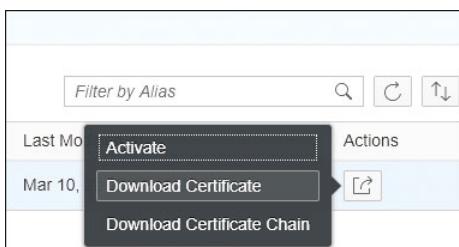


Figure 10.61 Downloading a Certificate from a Key Provided in the New SAP Keys Monitor

5. Save the certificate on your computer (in our example, as a file named `sap_cloudintegrationcertificate.cer`).
6. Provide the administrator of the sender system with the certificate file through a protected channel (e.g., encrypted email) and ask him to update the certificate in

the corresponding keystore of the sender system (to enable the sender to encrypt messages using the new certificate).

Key Renewal without Any Downtime

Note that until now, we've kept quiet about a critical aspect of key renewal. In the example discussed here, we didn't consider that a downtime needs to be considered under certain conditions when renewing keys in all affected systems involved in a secure communication setup.

The tenant administrator should, therefore, find out prior to the activities related to key renewal if the sender system is capable of encrypting messages with the old and the new SAP key at the same time. In such a case, the process of key renewal can be organized so that no downtime is required for the integration scenario. The administrator of the sender system can just import the public key provided by the tenant administrator into the keystore of the sender system and inform the tenant administrator when he has finished this task. The tenant administrator can then activate the new key as shown in the next step.

In the online documentation for SAP Cloud Platform Integration (https://help.sap.com/viewer/product/CLOUD_INTEGRATION/Cloud) under **Operating and Monitoring** **Cloud Integration • Security Artifact Renewal**, you'll find extensive information on this topic and also about those use cases and conditions under which a key renewal task can be organized without any downtime.

7. After the sender administrator has confirmed these actions, you can proceed and activate the new SAP key in the tenant keystore. Go to the **New SAP Keys** monitor, select the new SAP Key (in our example, with the alias **sap_cloudintegrationcertificate**), and choose **Activate** under **Actions** (Figure 10.62).

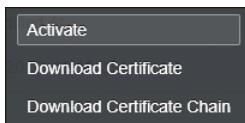


Figure 10.62 Activating a New SAP Key

8. A warning message is shown that reminds you to do the preparatory steps as explained earlier (Figure 10.63).

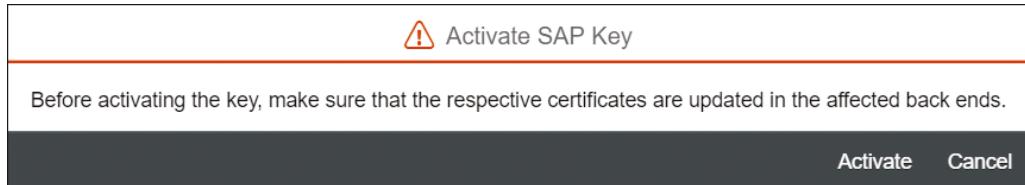


Figure 10.63 Warning Message Reminding You to Update Corresponding Keys in the Connected Remote Systems

- As you've already taken care of the required actions regarding the sender system, you can click **Activate**.

The following happens now behind the scenes: The old key pair (with alias `sap_cloudintegrationcertificate`) is copied from the active keystore (**Current** tab) to the **Key History** keystore. Then, the new key pair (from the **New SAP Keys** keystore) is copied to the active keystore (**Current** tab) and overwrites the old key pair there. Finally, the new key pair is removed from the **New SAP Keys** keystore.

Note

For a detailed description of the process, check out the documentation for SAP Cloud Platform Integration at https://help.sap.com/viewer/product/CLOUD_INTEGRATION/Cloud, and search for **Activating a New SAP Key Pair on the Tenant**.

- Check in the **Key History** tab to see if the old key pair has been added (Figure 10.64).

Overview / Manage Keystore			
Current	Backup	New SAP keys (1)	SAP Key History
Entries (1)			<input type="text" value="Filter by Alias"/> <input type="button" value="Search"/> <input type="button" value="New"/> <input type="button" value="Up"/>
Alias	Active From	Active Until	Actions
<code>sap_cloudintegrationcertificate</code>	Mar 07, 2018, 13:57:26	Mar 10, 2018, 10:36:04	<input type="button" value="Edit"/>

Figure 10.64 Key History Showing the Old Key Pair Replaced by the New One in the Current Keystore

The **SAP Key History** tab provides an important **Add to New SAP Keys** option, which you can use to withdraw the activation of a new key and revert back to the previous SAP key pair (as explained in Chapter 8, Section 8.3.2, Table 8.14).

- Finally, you can verify that in the **New SAP Keys** tab, the newly activated key pair disappeared as expected (Figure 10.65).

The screenshot shows a user interface for managing keystores. At the top, there are tabs: 'Overview / Manage Keystore', 'Current', 'Backup', 'New SAP keys' (which is underlined, indicating it's the active tab), and 'SAP Key History'. Below the tabs is a search bar labeled 'Filter by Alias' with a magnifying glass icon, and buttons for refresh and sort. A table below the search bar has columns: 'Alias', 'Valid From', 'Valid Until', 'Last Modified At', and 'Actions'. The table displays the message 'No data'.

Figure 10.65 Newly Activated Key Pair Finally Removed from the New SAP Keys Monitor

To finish this discussion, let's assume that the key pair to renew is also used in an outbound HTTP connection to implement client certificate authentication (in a setup as shown earlier in Figure 10.20). In this case, prior to activating the new key, download the certificate chain of the SAP key from the **New SAP Keys** keystore (choose **Download Certificate Chain** as the **Action** option), and save it as file (in our example with the name *sap_cloudintegrationcertificate.p7b*) to your computer. From the certificate chain, you finally need to extract the client root certificate and the client certificate, and make both available to the administrator of the receiver system. The receiver administrator needs to import them into the receiver keystore prior to the tenant administrator activating the new key pair.

We've only given a simple example of a situation where a key pair renewed by SAP was used in an existing integration setup. In real-life scenarios, you have to assume that expired keys need to be removed in many more *places* in an integration scenario.

In this section, we only gave a glimpse into the lifecycle management for keys. Depending on the integration scenario, there are different processes that make sure you execute key renewal on all sides of the communication in such a way that down-times are either avoided or kept to a minimum.

We finish this section by mentioning that, in case the tenant administrator hasn't activated an SAP key pair prior to its expiration, a system job activates the key pair automatically (see [Chapter 8, Section 8.3.2](#)).

For a good introduction into how to organize key renewal, read the "Cloud Integration – Activate SAP Keys in Keystore Monitor" blog in the SAP Community (www.sap.com/community.html).

10.6 Summary

Security considerations often come first when businesses consider sourcing out parts of their IT processes into the cloud. This chapter provided an overview of the various measures undertaken to protect customer data processed by SAP Cloud Platform Integration at the highest level. Several security features are already built-in and available via the architecture, the network design, and the way sensitive data is protected at SAP. Furthermore, you've seen how security is taken seriously by the way processes are performed during the lifecycle of an integration project. We showed you how to manage users and authorizations for your SAP Cloud Platform subaccount. Finally, you learned what you, as the customer, can do to configure secure message exchange between your landscape and SAP Cloud Platform Integration. We've shown you, step by step, how to set up a secure connection between remote components and SAP Cloud Platform Integration and how to build integration flows that implement basic security features. We closed this chapter by providing a glimpse into how to manage the lifecycle of security material in the tenant keystore.

In the next chapter, we will provide you with an overview of some productive scenarios that use SAP Cloud Platform Integration.

Chapter 11

Productive Scenarios Using SAP Cloud Platform Integration

Now that you've undertaken a journey through the world of SAP Cloud Platform Integration, you should be familiar with the principles and concepts to begin working with it seriously. You also know that SAP Cloud Platform Integration enables you to build networks of IT applications flexibly on any scale. In this chapter, we'll show you a few examples of how this product can be used productively in real-life scenarios.

In [Chapter 1, Section 1.2](#), we introduced SAP Cloud Platform Integration as a cloud-based integration solution that supports cloud-to-cloud integration and cloud-to-on-premise integration. We also showed that SAP Cloud Platform Integration can be used in combination with on-premise integration solutions (e.g., SAP Process Orchestration). In this chapter, we'll explain a few productive scenarios that each demonstrates one of the cloud-to-cloud and cloud-to-on-premise use cases. Additionally, we'll show you how these scenarios work in real life, in existing landscapes, with actual applications that are connected with each other.

For most of these scenarios, predefined integration content is available in the Integration Content Catalog, which you can use out of the box. That is why [Chapter 3](#), which deals with the Integration Content Catalog, contains information about these scenarios. Therefore, we'll keep the description of those scenarios short in this chapter.

11.1 Integration of SAP Cloud for Customer and SAP ERP

SAP Cloud for Customer is SAP's cloud-based customer relationship management (CRM) solution that helps you, as an SAP customer, improve your own customer interactions.

Using a cloud-based CRM solution often requires that you replicate master data (e.g., account, product, or employee) or transactional data from a connected on-premise SAP ERP application. Transactional data may be replicated, for example, with sales orders, or may be referenced synchronously for the latest updates, such as the latest figures for a customer-specific price. Obviously, both the cloud application and the on-premise system need to be kept in sync, and, therefore, integration scenarios have to be implemented.

[Figure 11.1](#) shows the general setup of the integration of SAP ERP with SAP Cloud for Customer.

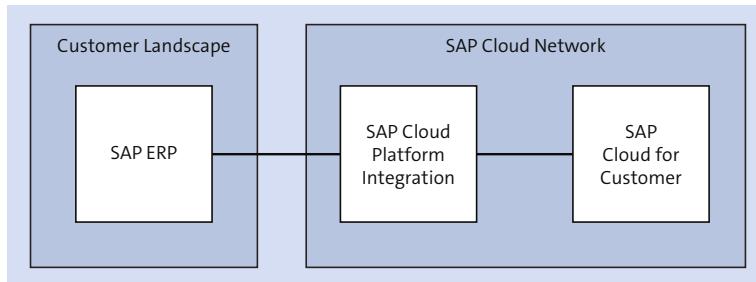


Figure 11.1 Integration of SAP ERP with SAP Cloud for Customer

SAP provides predefined integration content for the integration of SAP Cloud for Customer with SAP ERP in the Integration Content Catalog. An overview of the content and more details on the business use case of this scenario have already been provided in [Chapter 3, Section 3.4.2](#).

11.1.1 Technical Landscape

The integration of SAP Cloud for Customer with SAP ERP is an example of cloud-to-on-premise integration. [Figure 11.2](#) shows the commonly used technical landscape for the scenario. There are, as always, many options, but we'll briefly describe a common setup.

In the proposed setup, the SAP Cloud for Customer application and the integration middleware (SAP Cloud Platform Integration) run in the SAP cloud network, whereas the connected on-premise application (SAP ERP) runs in the customer landscape.

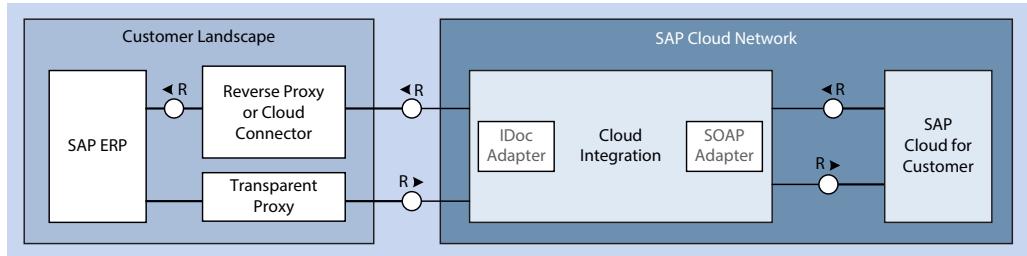


Figure 11.2 Technical Landscape for Integration of SAP Cloud for Customer with SAP ERP (Example Setup)

All components communicate with each other using HTTPS. However, note that connections between the SAP cloud network and components located in the customer landscape require particular security measures: you must ensure that the components in the customer landscape aren't directly accessible from the Internet. For this communication direction, that means when messages are sent from the SAP cloud network to the customer-based SAP ERP system, a component (either a reverse proxy or the SAP Cloud Platform Connectivity service [hereafter, SAP Cloud Platform Connectivity]) in the customer landscape terminates the Transport Layer Security (TLS) requests and reestablishes new ones. That way, it's ensured that the components in the customer landscape are shielded from the open Internet. As a reverse proxy, the SAP Web Dispatcher can be used.

The other way around, a proxy server (or transparent proxy) routes the request from the SAP ERP system to the target component (SAP Cloud Platform Integration) without terminating any TLS request.

SAP Cloud Platform Integration, as message broker interconnected between SAP Cloud for Customer and SAP ERP, typically uses the following connectivity options:

- Simple Object Access Protocol (SOAP) adapter or IDoc adapter (which also uses the SOAP protocol) for connections between SAP ERP and SAP Cloud Platform Integration
- SOAP adapter for connections between SAP Cloud Platform Integration and SAP Cloud for Customer

11.1.2 Example Adapter Configurations

To round off this section, we'll briefly show you how the connections of the scenario depicted in [Figure 11.2](#) can be configured in a productive use case on the SAP Cloud Platform Integration side of the communication. To do this, we've provided screenshots of the four adapters that come into play at the side of the SAP Cloud Platform Integration system.

[Figure 11.3](#) shows an example of an IDoc adapter used for the connection between SAP ERP and SAP Cloud Platform Integration (messages sent from SAP ERP to SAP Cloud Platform Integration).



Figure 11.3 IDoc Adapter for Messages Sent from SAP ERP to SAP Cloud Platform Integration

In the same way in the first integration flow you saw in [Chapter 2](#) for the **SOAP** sender adapter, the **Address** field should define an endpoint address so that SAP ERP can call the integration flow deployed on the SAP Cloud Platform Integration tenant. The final destination, which is to be configured in the SAP ERP system, is then composed of the URL, the runtime node (assigned to the tenant), and this endpoint address (see [Chapter 2, Section 2.3.5](#)). The field **URL to WSDL** was explained in the context of the SOAP adapter in [Chapter 5, Section 5.3.1](#).

[Figure 11.4](#) shows an example of an **IDoc** adapter for the connection between SAP ERP and SAP Cloud Platform Integration for the opposite communication direction (messages sent from SAP Cloud Platform Integration to SAP ERP).

In the **Address** field, you have to specify the host and port of the SAP ERP system, as well as the SAP client. The string `/sap/bc/srt/idoc` is a fixed part of the address to point to the IDoc service of an SAP system.

In the **Proxy Type** field, the **Internet** option is selected as default, which means that the connection is done via a reverse proxy infrastructure (an option used by many customers for such integration scenarios). As an alternative, you can select **On Premise** if you want to use SAP Cloud Platform Connectivity to connect to SAP ERP. For

more information about this attribute, check out the “Proxy Type” info box at the end of [Chapter 10, Section 10.4.3](#).

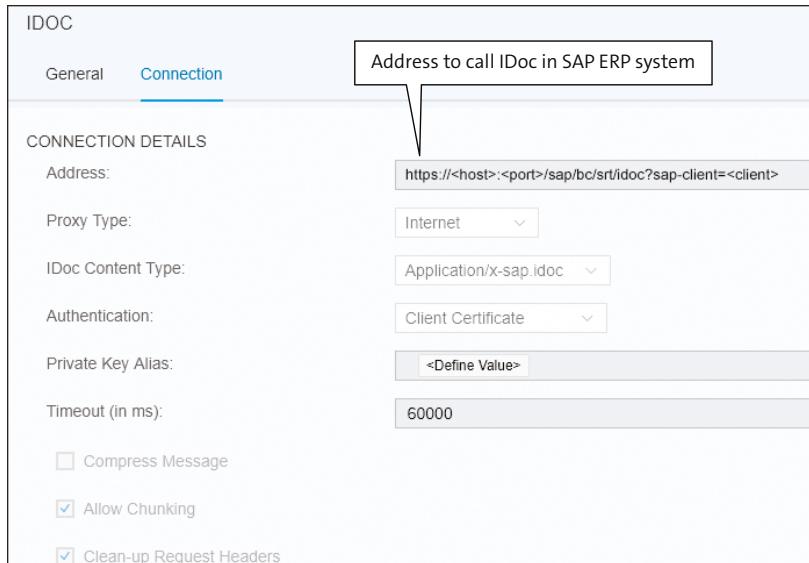


Figure 11.4 IDoc Adapter for Messages Sent from SAP Cloud Platform Integration to SAP ERP

[Figure 11.5](#) shows an example of a SOAP (1.x) sender adapter for the connection between SAP Cloud Platform Integration and SAP Cloud for Customer (messages sent from SAP Cloud for Customer to SAP Cloud Platform Integration):

In the **Address** field, you need to define an endpoint address so that SAP Cloud for Customer can call the integration flow deployed on the SAP Cloud Platform Integration tenant.

In this example, in the **URL to WSDL** field, a WSDL is also specified, allowing SAP Cloud Platform Integration to access information contained in the WSDL to process the message (this was explained in [Chapter 5, Section 5.3.1](#), in our example of an asynchronous message).

[Figure 11.6](#) shows an example of a SOAP (1.x) receiver adapter establishing the connection between SAP Cloud Platform Integration and SAP Cloud for Customer for the opposite communication direction (messages sent from SAP Cloud Platform Integration to SAP Cloud for Customer).

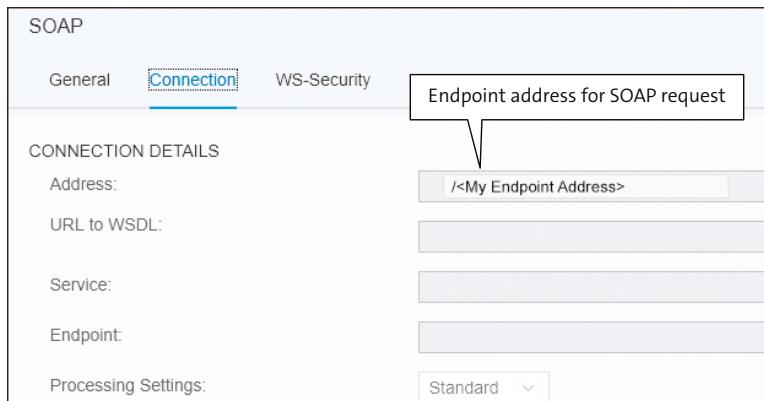


Figure 11.5 SOAP Adapter for Messages Sent from SAP Cloud for Customer to SAP Cloud Platform Integration

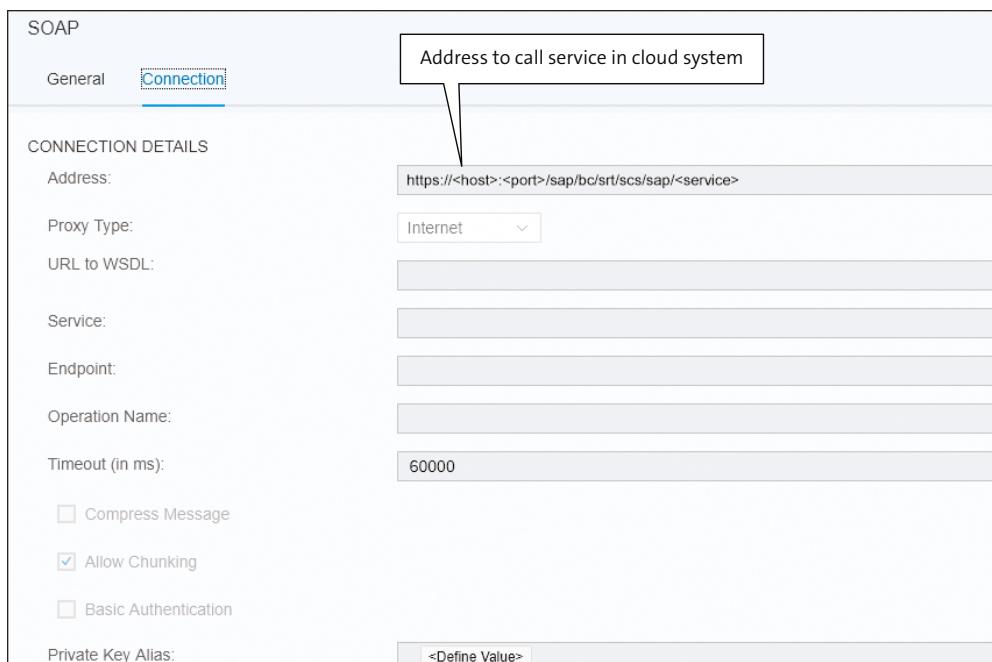


Figure 11.6 SOAP Adapter for Messages Sent from SAP Cloud Platform Integration to SAP Cloud for Customer

In the **Address** field, you must specify the service address in the SAP Cloud for Customer system.

Note that the shown examples ([Figure 11.5](#) and [Figure 11.6](#)) are retrieved from the Integration Content Catalog and reflect earlier versions of the SOAP adapter. In the meantime, updates to this adapter type have been provided so that when you create a new SOAP adapter, you'll get a slightly different user interface (UI). For more information on the versioning concept for integration flow components, see [Chapter 6, Section 6.5](#).

11.2 Integration of SAP Cloud for Customer with SAP S/4HANA Cloud

SAP S/4HANA is SAP's next generation business suite, built on SAP HANA database technology, and providing SAP Fiori UIs. SAP Fiori is a modern UI technology that enables customers to extend the use of SAP applications to tablet computers and smartphones. You can install SAP S/4HANA in your own landscape (on premise) or use it in the public (SAP) cloud.

Customers who prefer to run the IT processes of their enterprise in the cloud can choose SAP S/4HANA Cloud. It covers the areas of logistics, accounting and financials, and sales. The integration package for SAP Hybris Cloud for Customer Integration with SAP S/4HANA Cloud, available in the Integration Content Catalog, provides a set of predefined integration flows and value mappings that facilitate the replication of objects, such as business partner and material, between SAP S/4HANA Cloud and SAP Cloud for Customer.

Compared to the integration of SAP Cloud for Customer and SAP ERP (refer to [Section 11.1](#)), this integration package deals with cloud-to-cloud integration, as illustrated in [Figure 11.7](#).

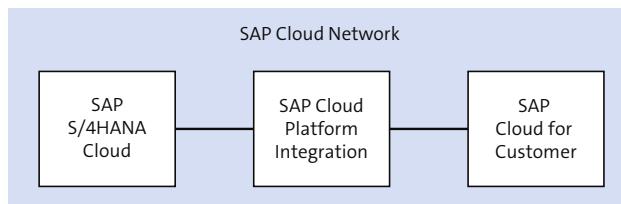


Figure 11.7 Integration of SAP S/4HANA Cloud with SAP Cloud for Customer

11.3 Integration of SAP Marketing Cloud and Various Applications

SAP C/4HANA is an on-demand product family that includes SAP Cloud for Customer (discussed in [Section 11.1](#)), SAP Commerce Cloud, SAP Customer Data Cloud, and SAP Marketing Cloud.

The available integration packages were already presented in [Chapter 3, Section 3.4.3](#) (there, you also find a brief introduction into these solutions).

SAP Marketing Cloud is a cloud solution offered by SAP to help customers optimize their marketing activities (e.g., by developing and executing successful marketing campaigns). As already mentioned in [Chapter 3, Section 3.4.3](#), SAP Marketing Cloud can be integrated with other components such as the following:

- SAP Cloud for Customer (e.g., for master data replication)
- Applications such as SAP S/4HANA Enterprise Management on-premise, SAP Customer Relationship Management (SAP CRM), SAP Cloud for Customer, and SAP ERP
- Social media platforms, such as Twitter or Facebook

We would like to pick out the integration of SAP Marketing Cloud with Twitter as one example for a social media platform. It's obvious that analyzing social media content is invaluable when it comes to getting to know what customers or potential customers think about certain topics. Integrating SAP Marketing Cloud with Twitter allows SAP customers to load data from Twitter into SAP Marketing Cloud and to analyze it further. For example, SAP Marketing Cloud allows you to do sentiment analysis based on Twitter content. In this context, sentiment analysis, in short, is a methodology to analyze social media content (Twitter tweeds) related to a certain topic to determine the attitude of users with regard to this topic. SAP Marketing Cloud can then use the results of this analysis to adapt and improve marketing campaigns.

[Figure 11.8](#) illustrates the components integrated with this scenario.

In [Chapter 10, Section 10.4.5](#), you've already become familiar with the Twitter adapter. The mentioned integration package uses the Twitter adapter to connect Twitter with SAP Cloud Platform Integration. On the other side of the communication, the HTTP adapter is used to connect SAP Cloud Platform Integration with SAP Marketing Cloud.

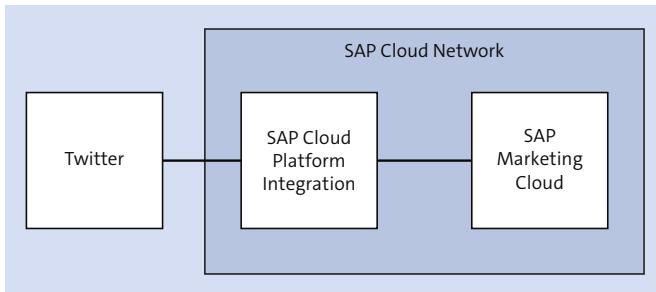


Figure 11.8 Integration of Twitter with SAP Marketing Cloud

You can find the package **SAP Marketing Cloud – Twitter Integration** in the Integration Content Catalog. When you've copied the integration package to your workspace (see [Chapter 3, Section 3.2.2](#)), you can configure the integration flow. The available integration content has been designed in such a way that you only need to configure a few parameters to set the integration scenario into operation—without any further editing and adapting any integration flow (see [Chapter 4, Section 4.2](#), for more details on configuring externalized integration flow parameters). To finish the communication between Twitter and SAP Cloud Platform Integration, the only thing you need to do is to configure your Twitter application programming interface (API) and deploy the required OAuth credentials as **Secure Parameter** artifacts (as we've shown in detail in [Chapter 10, Section 10.4.5](#)).

You can also find a concise and easy-to-follow tutorial on the SAP website at www.sap.com/developer/tutorials/cpi-sentiment-analysis.html.

11.4 Integration of SAP SuccessFactors and SAP ERP

SAP SuccessFactors is cloud-based human capital management (HCM) software that provides tools for recruiting, performance management, talent management, and other employee-centric solutions. It also provides core employee management capabilities (in the Employee Central module).

In many cases, when companies plan to move parts of their HCM processes to the cloud, a phased approach might be the solution of choice. Either the company first moves only parts of the HCM processes to the cloud (e.g., recruitment) and keeps the core functions located in the on-premise landscape, or they prefer to migrate their HCM solution successively to the cloud, based on certain locations. In any case, a

seamless and tight integration between the processes running in the cloud and in the on-premise environment is critical.

As an example of the separation of processes between the on-premise landscape and the cloud, a company might want to run its recruitment processes using SAP SuccessFactors in the cloud, whereas core employee management functions are kept in the on-premise SAP ERP Human Capital Management (SAP ERP HCM) application.

Figure 11.9 shows the involved components at a high level.

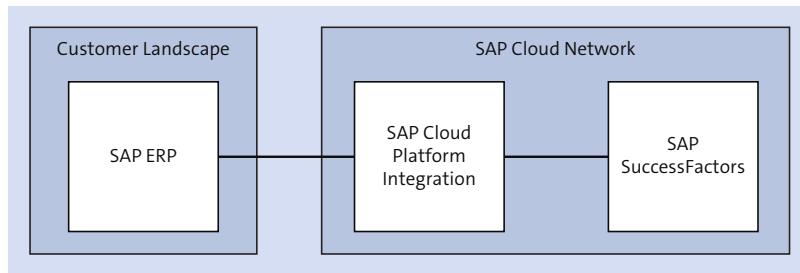


Figure 11.9 Integration of SAP ERP with SAP SuccessFactors

The integration of SAP SuccessFactors with SAP ERP is cloud-to-on-premise.

In the Integration Content Catalog, you can find various integration packages that facilitate the integration of SAP ERP-based processes with the cloud-based HCM processes of SAP SuccessFactors. An overview of the content and more details on the business use case of such scenarios has already been provided in [Chapter 3, Section 3.4.1](#). In this section, we cover additional aspects such as the typical technical landscape setup for such scenarios.

11.4.1 Technical Landscape

Figure 11.10 shows a common setup of an SAP SuccessFactors and SAP ERP integration scenario.

An obvious option to connect the SAP ERP with SAP Cloud Platform Integration is using web services communication (through the SOAP adapter).

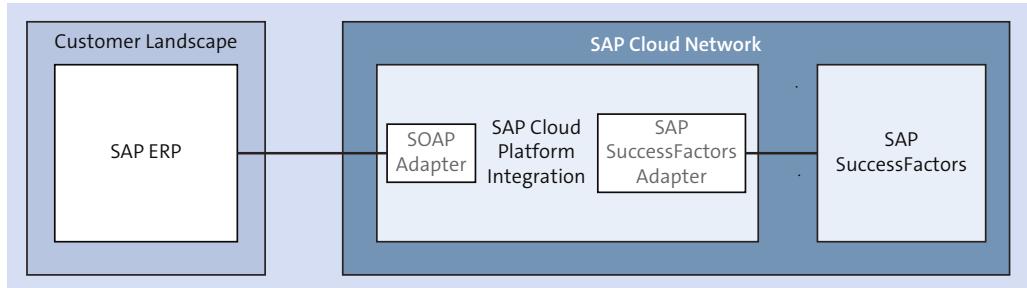


Figure 11.10 Integrating SAP ERP with SAP SuccessFactors through SAP Cloud Platform Integration

For the other side of the communication, SAP SuccessFactors offers various API options to technically integrate with and connect to other systems, including the following:

- **SF API**

This is a SOAP API designed to import or export data to and from SAP SuccessFactors. It allows you to perform create, read, update, delete (CRUD) operations on SAP SuccessFactors entities.

- **OData API**

This API allows you to access SAP SuccessFactors content using OData.

SAP Cloud Platform Integration provides an option to use these APIs in an intuitive and convenient way: the SAP SuccessFactors adapter, which is part of the SAP Cloud Platform Integration standard adapter offering.

11.4.2 SAP SuccessFactors Adapter

The SAP SuccessFactors adapter comes in several variants, depending on which API you want to connect to the SAP SuccessFactors system with (and the communication direction). Table 11.1 provides a list of these variants.

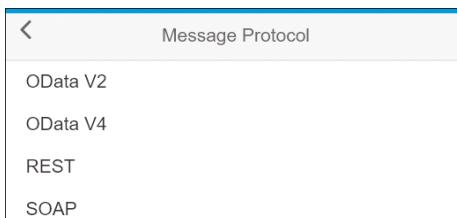
Adapter Variant	Allows You To...
Sender SOAP	Connect to an SAP SuccessFactors sender, and read data from it using web services.

Table 11.1 SAP SuccessFactors Adapter Variants

Adapter Variant	Allows You To...
Sender REST	Connect to an SAP SuccessFactors learning management system (sender), and read data from it through a REST API.
Receiver SOAP	Connect to an SAP SuccessFactors receiver to perform read or write operations on the content using web services.
Receiver REST	Connect to an SAP SuccessFactors receiver to perform read or write operations on the content using a REST API.
Receiver OData V2/V4	Connect to an SAP SuccessFactors receiver to perform read or write operations on the content using OData.

Table 11.1 SAP SuccessFactors Adapter Variants (Cont.)

Each SAP SuccessFactors adapter type provides a dedicated configuration UI to access SAP SuccessFactors entities intuitively. After you've specified an SAP SuccessFactors system to connect to, through the configuration UI of the adapter, you can easily select the entities and define certain operations on them without the need to write any line of code. When you add an SAP SuccessFactors adapter to an integration flow, you'll choose the corresponding variant by selecting a **Message Protocol**, as shown in [Figure 11.11](#), for a SAP SuccessFactors receiver adapter.

**Figure 11.11** Selecting the Message Protocol to Decide Which Adapter Variant to Add (for the SAP SuccessFactors Receiver Adapter)

The detailed properties of the adapter's configuration UI depend on the chosen message protocol. [Figure 11.12](#) shows an example for a SAP SuccessFactors adapter configuration UI for a receiver adapter when you've selected **SOAP** as the **Message Protocol**. Some key properties are indicated and explained next.

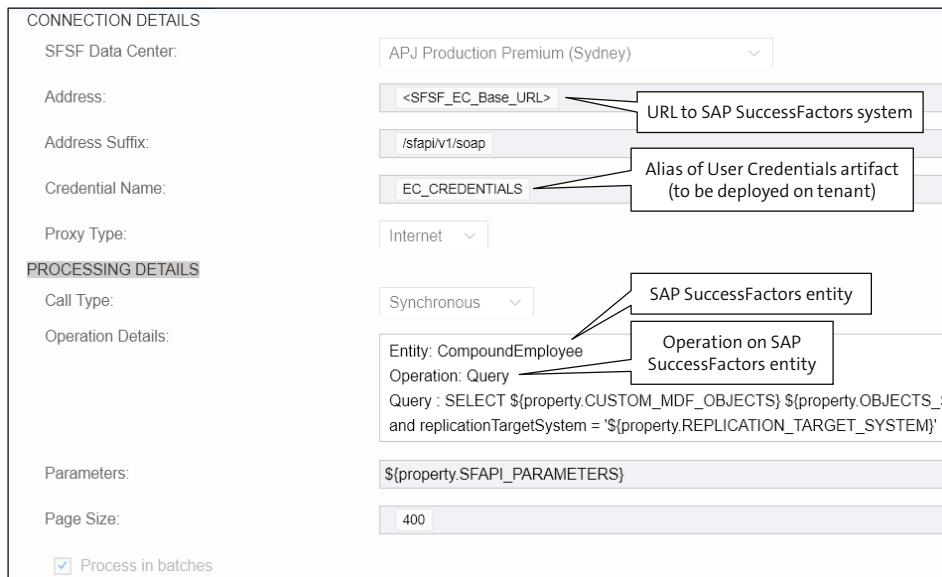


Figure 11.12 SAP SuccessFactors Adapter (Example) from the Integration Content Catalog

Note that the example shown is retrieved from the Integration Content Catalog and reflects the component version 1.1 of the SAP SuccessFactors adapter. In the meantime, updates to this adapter type have been provided so that, when you create a new SAP SuccessFactors receiver adapter (with the SOAP message protocol), you'll get a slightly different UI (with the settings now spread on two different tabs **Connection** and **Processing**). (For more information on the versioning concept for integration flow components, see [Chapter 6, Section 6.5](#).)

When you configure an SAP SuccessFactors sender adapter (to connect to an SAP SuccessFactors sender system), you can specify only query operations. This is because the SAP SuccessFactors adapter acts as a polling adapter, reading information at scheduled intervals from the SAP SuccessFactors system (similar to how the Mail sender adapter reads emails from an email server; see [Chapter 2, Section 2.3.8](#)).

For receiver adapters, all standard CRUD operations of SF API (query, insert, upsert, update) are supported and can be configured ([Figure 11.12](#)).

To configure a secure connection between SAP Cloud Platform Integration and an SAP SuccessFactors sender or receiver system, you'll use the **User Credentials** artifact type, which comprises a username, password, and, specific to SAP SuccessFactors, a

company ID (to indicate the SAP SuccessFactors system you're connecting to). When configuring the connection, you first define a **User Credentials** artifact and deploy it on the tenant, and, second, refer to the alias of the artifact (in the **Credential Name** field) in the SAP SuccessFactors adapter. Compare this process to the one we used for the mail adapter (see [Chapter 2, Section 2.3.5](#)).

[Figure 11.13](#) shows the properties of the **User Credentials** artifact to be deployed on the tenant to configure a secure connection between the tenant and an SAP SuccessFactors system.

The screenshot shows a dialog box titled 'Add User Credentials'. It contains fields for 'Name' (mandatory), 'Description', 'User' (mandatory), 'Password' (mandatory), and 'Repeat Password'. A checked checkbox labeled 'SuccessFactors' is present. Below these fields is a mandatory field for 'Company ID'. At the bottom of the dialog are 'Deploy' and 'Cancel' buttons.

Figure 11.13 User Credentials Artifact for SAP SuccessFactors

For SAP SuccessFactors sender adapters, you can also, for example, select a scheduler to specify that the adapter reads data from the SAP SuccessFactors (sender) system at certain time intervals.

If you're interested to learn more about the integration of SAP SuccessFactors with other applications using SAP Cloud Platform Integration, check out the following E-Bite available from SAP PRESS: *Integrating SAP SuccessFactors with SAP Cloud Platform Integration (SAP HCI)* at www.sap-press.com/4371.

11.5 Integration of SAP Applications with the Ariba Network

The Ariba Network provides a worldwide online marketplace that brings buyers and suppliers together via the Internet and helps them facilitate their procurement processes. Interacting in this network, companies can streamline and accelerate their procurement processes, which include buying, selling, and cash management.

When you keep in mind that in the recent past, the majority of such processes still relied on the exchange of paper-based information, you can imagine that moving the processes to the network and automating them helps accelerate them greatly and considerably reduces manual errors. Both buyers and their suppliers benefit from such a shift. For example, buyers can pay earlier and may benefit from discounts because they are paying faster, and suppliers are receiving their money earlier.

The basic scenario is that a buyer (who uses an SAP ERP system to manage the procurement process) interacts with many suppliers through the Ariba Network. Each supplier is a registered member of the Ariba Network.

The Ariba Network supports different options for buyers to manage the procurement process:

- The buyer's SAP ERP system manages the ordering process, whereas the invoicing process is sourced out to, and automated via, the Ariba Network.
- The entire procure-to-pay process (including the ordering process) is automated using the Ariba Network, and the buyer's SAP ERP system only acts as a system of records.

Let's briefly focus on the first option, where the ordering process is managed by the SAP ERP system. Once set up, a typical scenario comprises the following steps at both the buyer's and supplier's sides:

1. The buyer creates a purchase order to order a product from the supplier's catalog and sends the purchase order to the supplier.
2. The supplier receives the purchase order in its inbox, creates an order confirmation, and sends that confirmation to the buyer.
3. The buyer receives the order confirmation.
4. After the article ships, the supplier creates an advanced shipping notification and sends it to the buyer to inform him that the article is on its way.
5. As soon as the product arrives, the buyer posts a goods receipt and sends it to the supplier.
6. After the goods receipt arrives, the supplier creates an invoice and forwards it to the buyer.

To integrate the buyer's SAP ERP systems with the Ariba Network, SAP provides an Ariba Network Integration for SAP Business Suite Add-On 1.0, which supports the following integration options:

- Setting up a point-to-point connection with the Ariba Network (based on web services)
- Using SAP Process Orchestration as on-premise integration platform
- Using SAP Cloud Platform Integration as the cloud integration solution

For integration using SAP Cloud Platform Integration, SAP provides predefined integration content in the Integration Content Catalog. An overview of the content and more details on the business use case of this scenario have already been provided in [Chapter 3, Section 3.4.2](#).

11.5.1 Technical Landscape

[Figure 11.14](#) shows the technical landscape for integrating a buyer with the Ariba Network based on SAP Cloud Platform Integration.

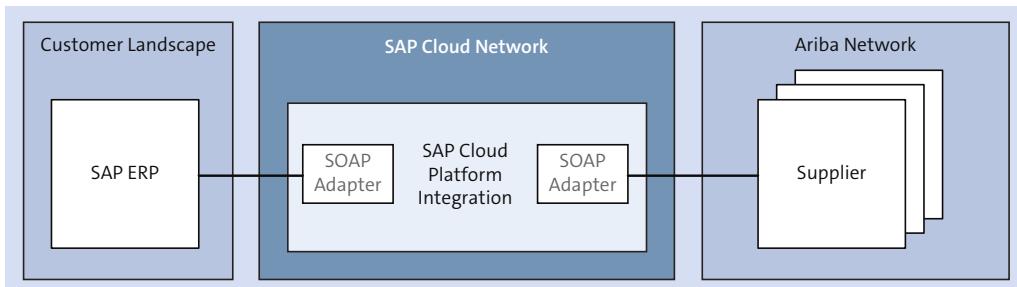


Figure 11.14 Technical Landscape for the Integration of SAP ERP with the Ariba Network

As indicated in [Figure 11.14](#), the connectivity both for the buyer's SAP ERP system and for the Ariba Network is based on the SOAP adapter.

Note that suppliers registered in the Ariba Network can use either their own proprietary applications or an SAP application to manage the procurement process from their side, or they can use a UI provided by the Ariba Network itself.

11.6 Summary

In this chapter, we provided you with a high-level overview of a few productive scenarios that can be implemented using SAP Cloud Platform Integration. We quickly walked you through a few scenarios that allow you to integrate SAP solutions both in

cloud-to-cloud and in hybrid cloud-to-on-premise landscapes. We closed this chapter by briefly touching on Ariba Network integration based on SAP Cloud Platform Integration.

With this, we close our introduction to SAP Cloud Platform Integration. In the next chapter, we provide a final outlook on the evolution of SAP Cloud Platform Integration that users can expect to see in the future.

Chapter 12

Summary and Outlook

In this final chapter, we'll briefly wrap up what you've learned over the course of this book and walk you through some of the enhancements to the SAP Cloud Platform Integration portfolio that you can expect to see in the coming months and years.

Congratulations! You've reached the end of the book, and you're now equipped with all the knowledge required to productively begin your own integration project. However, this isn't the end of your journey into the world of cloud integration: SAP Cloud Platform Integration is updated on a monthly basis, which means that you can expect continuous product improvements. In this chapter, after a brief summary of what you've learned in this book, we'll walk you through a number of key enhancements you can expect to see in the future.

In [Chapter 1](#), we introduced you to the topic of the book by showing you how SAP Cloud Platform Integration fits into the overall cloud strategy of SAP. By outlining various use cases, we gave you an idea about how you can embed SAP Cloud Platform Integration into your own company's digital strategy. In [Chapter 2](#), we brought you up to speed with SAP Cloud Platform Integration's basic terms and concepts and followed up with a description of SAP Cloud Platform Integration's architecture. There, you also learned how to set up and run your first simple integration flow. [Chapter 3](#) introduced you to the Integration Content Catalog, the public show window and shop for SAP's prepackaged integration content. That chapter also introduced the Web UI. [Chapter 4](#), [Chapter 5](#), and [Chapter 6](#) showed you how to use the Web UI to build your own integration scenarios, starting with the simplest integration pattern and, bit by bit, progressing to more complex ones. Each of these chapters provided a rich offering of tutorials that showed you, in detail, how to work with the tool by yourself.

In [Chapter 7](#), we showed you how SAP Cloud Platform Integration supports business-to-business (B2B) scenarios along one end-to-end scenario. [Chapter 8](#) introduced you to another phase of the lifecycle of an integration project: the operations phase of an

integration scenario. You learned how to use the Web UI to monitor messages and integration artifacts.

Chapter 9 provided you with detailed information on how to access SAP Cloud Platform Integration based on an application programming interface (API).

To accommodate the fact that security is a key consideration when moving to the cloud, Chapter 10 introduced you to this topic in detail. You were also provided with a number of tutorials that showed you how to use the Web UI to configure secure scenarios, such as the digital encryption of messages.

Chapter 11 shared a number of examples of how SAP Cloud Platform Integration can be used productively in real-life scenarios. And, finally, you arrived here—at Chapter 12.

In this chapter, we would like to give you an overview of some areas where you can expect to see major enhancements to SAP Cloud Platform Integration in the coming months and years. However, we recommend that you view the following statements as an overview of a roadmap based on today's assumptions, without any claim to be complete. We can't give any guarantee that the mentioned enhancements will be available at a certain point in time, as SAP might change development planning without notice. If you want more reliable information to plan for your own projects and strategies, check the SAP Cloud Platform Integration online documentation, or contact SAP directly. You might also like to check out www.sap.com/products/roadmaps.html. If you search for **Cloud Integration**, you'll find the up-to-date roadmap.

12.1 Multi-Cloud Support

SAP Cloud Platform provides two different development environments: the Neo environment and the Cloud Foundry environment. Both environments are available in different regions worldwide. Note that such an environment to develop, manage, and run applications in the cloud is also referred to as *platform-as-a-service (PaaS)*. A PaaS usually is technically operated on an *infrastructure-as-a-service (IaaS)*, which identifies on-demand access to network, storage, and other resources in the cloud (provided and maintained by a cloud software vendor). For more information on these terms, see the upcoming info box.

We won't go into more detail. However, we would like to point out one major difference between the two environments: Neo is available on an infrastructure that is provided by SAP exclusively, whereas Cloud Foundry, which is the industry standard

PaaS—might run on an IaaS provided by various third-party vendors such as Amazon, Microsoft, or Google. Supporting both environments, SAP Cloud Platform is a *multi-cloud* enterprise cloud platform in such a way that customers can choose freely the underlying cloud infrastructure. For more information on the different environments of SAP Cloud Platform, open its documentation at <https://help.sap.com/viewer/p/CP>, and select **What is SAP Cloud Platform • Environments**. For more information on Cloud Foundry, check out the Cloud Foundry documentation at www.cloudfoundry.org.

You might have asked yourself why, when reading this book, we hardly touched on the topic of SAP Cloud Platform environments, in particular, Cloud Foundry. We briefly mentioned Cloud Foundry only in [Chapter 6, Section 6.6.3](#), when showing you how to set up SAP Cloud Platform's Transport Management Service for integration content transport. Transport Management Service—a service provided on top of SAP Cloud Platform—runs in the Cloud Foundry environment (see, for example, [Figure 6.82](#)).

The answer is that at the time of this book's writing, SAP Cloud Platform Integration exclusively runs in the Neo environment. Therefore, throughout this book, we always made the assumption (without explicitly saying this) that SAP Cloud Platform Integration is used within the context of the Neo environment.

SAP Cloud Platform Integration (running in the Neo environment) is available in various regions worldwide. Physically, this means that SAP Cloud Platform Integration is deployed and operated in various data centers operated by SAP in different regions on the world. SAP continuously increases the number of data centers where SAP Cloud Platform Integration can be operated. To find out more about the actual regional availability, check out the documentation of SAP Cloud Platform at <https://help.sap.com/viewer/p/CP> and search for **Regions**.

However, that isn't the end of the story. We've stated that SAP Cloud Platform already is a multi-cloud enterprise cloud platform supporting multiple environments. Consequently, SAP Cloud Platform Integration will further evolve so that it becomes a multi-cloud service that can be used within the Cloud Foundry environment as well. This is one major enhancement that you can expect in the near future that will imply SAP Cloud Platform Integration can then also be made available on Amazon Web Services (AWS), Google Cloud Platform, or Microsoft Azure.

SAP Cloud Platform will deliver integration flow model compatibility, which means you'll be able to deploy and run your existing integration flows in any SAP Cloud Platform environment without manual adoption.

In particular, the ability to deploy SAP Cloud Platform Integration also on infrastructures provided by hyperscale cloud vendors such as Amazon, Google, or Microsoft will bring a significant increase in flexibility and scalability for the usage of SAP Cloud Platform Integration.

SaaS, PaaS, and IaaS Briefly Explained

We'll briefly shed a light on a couple of terms that often are used in the context of the multi-cloud topic.

The book you have in your hands is about software for process integration (SAP Cloud Platform Integration) that is made available in the cloud as *software-as-a-service (SaaS)*, which means it's delivered through the Internet by a software vendor (SAP) and also operated and updated by the same. By the way, that is why in [Chapter 10, Section 10.2.2](#), we named those people at SAP who are responsible to manage and administer SAP Cloud Platform Integration as *SaaS administrators*.

SAP Cloud Platform Integration is provided as a service on SAP's enterprise cloud platform (SAP Cloud Platform). SAP Cloud Platform is SAP's cloud environment for the development, deployment, and operation of cloud applications. Such a development environment is also generally referred to as a *platform-as-a-service (PaaS)*.

Each software development platform physically runs on servers (located in various data centers in different regions on the world), requiring resources (e.g., for data storage) and access to network components. Software vendors such as SAP, Amazon, Google, or Microsoft provide such infrastructures on demand in the cloud, and such an infrastructure is generally referred to as an *infrastructure-as-a-service (IaaS)*.

12.2 Integration Content Management

With regard to the following topics introduced in [Chapter 6](#), you can expect enhancements:

- **Integration content transport**

As you've learned from [Chapter 6, Section 6.6](#), integration content designed with the Web UI can be transported between different tenants. At the time of this book's writing, the cloud-based transport system, Transport Management Service, is only available as beta software. You can expect that this service will be made generally available within the next few months.

- **Versioning of integration flow components**

In [Chapter 6, Section 6.5](#), you learned about the versioning concept of integration flow components. As of the writing of this book, the following components can be migrated to a newer version: integration flow, integration process, and local integration process. Expect that other components (e.g., individual adapter types) can also be migrated in the future. This will provide you much more flexibility to adapt integration flows on a detailed level to new features provided by SAP.

12.3 Predefined Integration Content

In [Chapter 3](#), you learned about the Integration Content Catalog, which is the channel where SAP provides predefined integration content that you can use out of the box.

SAP continues enhancing the portfolio of predefined integration scenarios to support the integration of different SAP solutions with applications such as SAP SuccessFactors, SAP Ariba, SAP Customer Experience, and SAP S/4HANA.

12.4 New Connectivity Options

The number of adapters provided as part of the product portfolio of SAP Cloud Platform Integration will continuously increase. As of the writing of this book, we see that you can expect the availability of the following new adapters:

- **JDBC adapter**

SAP will make available a Java Database Connectivity (JDBC) adapter that enables you to connect SAP Cloud Platform Integration with a JDBC database and to execute SQL commands on the database.

- **OData V4**

In various tutorials given in this book, we used the OData adapter to retrieve data from an external OData source to further process it in the subsequent steps in an integration flow. The OData adapter currently available with SAP Cloud Platform Integration supports the OData version 2.0 (as described at www.odata.org/documentation/odata-version-2-0). However, there is a newer version of the standard available, version 4.0 (as described at www.odata.org/documentation). Expect that a new adapter supporting OData version 4.0 will be made available soon.

Note that we don't talk here about the OData variant of the SAP SuccessFactors adapter, which already supports both OData versions 2.0 and 4.0.

- **OFTP2**

SAP will make available an adapter to support Odette File Transfer Protocol v2 (OFTP2), which is used to transfer files over the Internet and is widely adopted in the European automotive industry for B2B. Therefore, this additional adapter will significantly enhance the B2B capabilities of SAP Cloud Platform Integration.

For more information on OFTP2, go to www.odette.org/services/oftp2.

12.5 Business-to-Business Support

In [Chapter 7](#), we gave you a detailed introduction to the existing B2B capabilities of SAP Cloud Platform Integration. To name a few components, there are several B2B-related adapters and flow steps (e.g., the AS2 adapter or the EDI splitter), but there are also “larger” components, such as the Integration Content Advisor and the Partner Directory.

SAP will invest in several areas to strengthen the B2B capabilities of the platform.

12.5.1 Further Adapters

SAP will continue to provide new adapters. One example for a new planned adapter that will enhance the B2B capabilities of the platform is the OFTP2 adapter mentioned earlier.

12.5.2 Enhancements of the Integration Content Advisor

The Integration Content Advisor will be enhanced continuously. Expect the following additional features:

- **Support of additional library types**

Integration Content Advisor will enhance its library of type systems, for example, by the additional industry standard VDA (VDA stands for the German Association of the Automotive Industry). Furthermore, SAP Cloud Platform Integration will support the VDA-specific syntax conversion.

- **Support of new formats and schemas**

The Integration Content Advisor will enable exact schema validation based on assertions. This new capability will be supported by the SAP Cloud Platform Integration schema validator based on XML Schema Definition (XSD) 1.1.

12.5.3 Trading Partner Management

Besides these particular enhancements, SAP has a vision to provide a new service—called *Trading Partner Management*—that interested parties can use as a central hub to get connected to other trading partners to build individually defined and agreed-upon B2B scenarios with each other using an open network.

Using this service, interested companies can specify and share their own business requirements as a profile visible for other potential interested parties. This will then be the basis to create connections with new trading partners and share information with them, to invite new parties, and, finally, to deploy and run integration scenarios based on the specifications derived collaboratively.

The already available key components—the Partner Directory and the Integration Content Advisor—will be integrated technically with the Trading Partner Management service so that partner-related information (created in the context of the partner network) will be stored in the Partner Directory of a partners' tenant, and the interfaces and mappings for the agreed B2B scenarios will be created by the Integration Content Advisor.

In other words, expect that SAP will soon develop a “social-like network” for setting up B2B scenarios across trading partners.

In addition, SAP will work on archiving options as well as monitoring capabilities that allow you to monitor message flows end to end—along all parties involved in a B2B scenario.

12.6 SAP Cloud Platform Integration API

The SAP Cloud Platform Integration application programming interface (API) was introduced in [Chapter 9](#). You can expect continuous enhancements of this API. For example, the API will be enhanced to support the management of integration content (design-time objects). Exposing such functions via an API will allow you, for example, to do mass changes in your setup of integration flows (when you need to maintain and keep up to date large numbers of integration flows).

12.7 Connectivity

As you've learned when reading this book, SAP Cloud Platform Integration comes already with a broad spectrum of connectivity options (SAP-developed adapters), which are constantly being updated. Furthermore, SAP Cloud Platform Integration customers can build their own adapters using the Adapter Development Kit (as we've shown in [Chapter 6, Section 6.7](#)).

A new SAP Cloud Platform service, Open Connectors, will allow you to extend the connectivity options of the platform in a complementary way. Open Connectors allows you to access and use a rich set of connectors that are based on REST APIs and provide harmonized access to the remote system with regard to authentication, error handling, and other aspects.

12.8 Security

The security-related features of SAP Cloud Platform Integration will be further expanded. One major part to be covered is the capabilities for the user to manage the tenant keystore. Furthermore, you can also expect SAP to continue getting certified to comply with additional security standards.

12.8.1 Tenant Keystore Management

The tenant keystore is the central component for the setup and maintenance of security material for HTTP connections. In [Chapter 10, Section 10.5](#), we introduced you to the features that allow you to manage the content of the tenant keystore. We particularly showed you how to use the keystore monitor to manage the lifecycle of SAP-owned keys (when being renewed by SAP). The keystore monitor also allows you to add single keystore entries (in the previous releases of SAP Cloud Platform Integration, it was required to upload or remove a complete keystore and to use an external tool such as the KeyStore Explorer to manage the content of the keystore prior to deployment).

However, even with these innovations, for the creation of new keys (owned by the tenant administrator), you still rely on external tools such as KeyStore Explorer. It's planned that in the near future, you can use the keystore monitor of the Web UI to perform all tasks required to create new keys end to end (e.g., generating a new key pair, creating certificate signing requests, etc.).

12.8.2 Compliance with Security Standards

As described in [Chapter 10, Section 10.2](#), SAP Cloud Platform Integration is certified to meet various security standards. SAP is continuously working on additional certifications. Expect that SAP Cloud Platform Integration will soon be certified to meet the requirements of the ISO27018 standard.

12.9 Harmonization of SAP Cloud Platform Integration with Other Services

SAP Cloud Platform Integration is a self-contained service that allows you to set up and operate integration scenarios without the need of any other tool. The Web UI is the main user entry point to design, deploy, and operate integration flows and to monitor the exchange of messages. However, there are certain use cases where it makes sense to combine the usage of SAP Cloud Platform Integration with other services provided on top of SAP Cloud Platform. We gave an example for this when showing how to use SAP Cloud Platform's Transport Management Service together with SAP Cloud Platform Integration to transport integration content across tenants (see [Chapter 6, Section 6.6.3](#)). Another example described in the book is the usage of SAP Cloud Platform Integration together with SAP Cloud Platform API Management (in short: SAP API Management). SAP API Management supports you throughout the whole lifecycle of an API. It allows you, for example, to publish your own APIs or to analyze the usage of APIs. In [Chapter 9, Section 9.6](#), we showed you how to use SAP API Management together with SAP Cloud Platform Integration.

A cloud service that is related to business process integration and automation is SAP Cloud Platform Workflow. Although we haven't touched on this topic in this book, SAP Cloud Platform Workflow is an on-demand solution that helps you automate business processes by providing a graphical modeling tool that allows you to design workflows based on Business Process Model and Notation (BPMN), similar to the integration flow designer of SAP Cloud Platform Integration. Furthermore, you can use this service to manage users and tasks, and to operate, monitor and manage workflows. SAP Cloud Platform Workflow can be connected with other components based on REST interfaces. However, you can also add SAP Cloud Platform Integration to your setup to enhance the connectivity options. For more information on SAP Cloud Platform Workflow, check out the documentation https://help.sap.com/viewer/p/WORKFLOW_SERVICE and the E-Bite *Introducing SAP Cloud Platform Workflow* by Rohit Khan and Rajiv Pandey (SAP PRESS, 2017, www.sap-press.com/4541).

As there are many use cases where it makes sense to use services such as SAP Cloud Platform Workflow and SAP API Management in conjunction with SAP Cloud Platform Integration, SAP is investing efforts to provide a harmonized user experience in the long term. Therefore, expect that these services will converge in the future to provide one single entry point for users who want to access all these services.

12.10 Summary

SAP Cloud Platform Integration is already a mature integration solution with a rich feature portfolio. However, as you see from this chapter, that isn't the end of the journey. SAP Cloud Platform Integration is updated every month, and, successively, many more features will be added to its portfolio. Stay tuned, and check out the available online resources on a regular basis to be informed about new innovations. And, keep enjoying your integration journey with SAP Cloud Platform Integration!

Appendices

A Abbreviations	767
B Literature	775
C The Authors	779

Appendix A

Abbreviations

Abbreviation	Long Text
A2A	Application-to-Application
ABAP	Advanced Business Application Programming
ADK	Adapter Developer Kit
AES	Advanced Encryption Standard
ANSI	American National Standards Institute
API	Application Programming Interface
AS2	Applicability Statement 2
AS4	Applicability Statement 4
ASC	Accredited Standards Committee
B2B	Business-to-Business
BIC	Bank Identifier Code
BizX Suite	Business Execution Suite
BPMN	Business Process Model and Notation
C4C	SAP Cloud for Customer
CA	Certification Authority
CBR	Content-Based Routing
CGI	Common Global Implementation Initiative
CIDX	Chemical Industry Data Exchange
Cloud Connector	SAP Cloud Platform Connectivity service

Abbreviation	Long Text
Cloud Integration	SAP Cloud Platform Integration
CMIS	Content Management Interoperability Services
CMS	Content Management System Cryptographic Message Syntax
CN	Common Name
CPU	Central Processing Unit
CRM	Customer Relationship Management
CRUD	Create, Read, Update, Delete
CSR	Certificate Signing Request
CSRF	Cross-Site Request Forgery
CSV	Comma-Separated Values
CXF	CeltiXFire
DMZ	Demilitarized Zone
DN	Distinguished Name
EAI	Enterprise Application Integration
EDI	Electronic Data Interchange
EDIFACT	Electronic Data Interchange for Administration, Commerce and Transport
EDMX	Entity Data Model XML
ERP	Enterprise Resource Planning
ESB	Enterprise Service Bus
ESR	Enterprise Services Repository
ETL	Extract, Transform, Load
EU	European Union
FSN	Financial Services Network

Abbreviation	Long Text
G&B	Generation & Build Subsystem
GDPR	General Data Protection Regulation
HCM	Human Capital Management
HL7	Health Level 7
HP	Hewlett-Packard
HR	Human Resources
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP Secure
IaaS	Infrastructure as a Service
ICA	Integration Content Advisor
IntaaS	Integration as a Service
IDoc	Intermediate Document
IEC	International Electrotechnical Commission
IMAP	Internet Message Access Protocol
IMAPS	Internet Message Access Protocol Secure
ISO	International Organization for Standardization
IT	Information Technology
JAR	Java Archive
Java SE	Java Platform, Standard Edition
JMS	Java Message Service
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
LDAP	Lightweight Directory Access Protocol
LMS	Learning Management System

Abbreviation	Long Text
MAG	Mapping Guideline
MDN	Message Disposition Notification
MEP	Message Exchange Pattern
MIG	Message Implementation Guideline
MIME	Multipurpose Internet Mail Extensions
MLS	Message Level Security
MPL	Message Processing Log
NRO	Number Range Object
OAuth	Open Standard for Authorization
OData	Open Data Protocol
ODETTE	Organization for Data Exchange by Tele-Transmission in Europe
OFTP	Odette File Transfer Protocol
OMG	Object Management Group
Open PGP	Open Pretty Good Privacy
OSGi	Open Services Gateway initiative
PaaS	Platform as a Service
PGP	Pretty Good Privacy
PKCS	Public-Key Cryptography Standard
PKIX	Public Key Infrastructure X.509
POP3	Post Office Protocol Version 3
POP3S	Post Office Protocol Version 3 Secure
REST	Representational State Transfer
RFC	Remote Function Call (in ABAP) Request for Comments
RN	Runtime Node

Abbreviation	Long Text
RSA	Initial letters of the surnames of Ron Rivest, Adi Shamir, and Leonard Adleman, representing a public-key cryptosystem
S/MIME	Secure/Multipurpose Internet Mail Extensions
SaaS	Software as a Service
SAP C4C	SAP Cloud for Customer
SAP CRM	SAP Customer Relationship Management
SAP ERP	SAP Enterprise Resource Planning
SAP FSN	SAP Financial Services Network
SAP HCM	SAP Human Capital Management
SAP CP	SAP Cloud Platform
SAP ID Service	SAP Identity Service
SAP MM	SAP ERP Material Management
SAP PI	SAP Process Integration
SAP PO	SAP Process Orchestration
SAP RM	SAP Reliable Messaging
SAP S/4HANA	SAP Business Suite 4 SAP HANA
SAP SD	SAP ERP Sales and Distribution
SAP SRM	SAP Supplier Relationship Management
SCA	Static Code Analyzer
SCN	SAP Community Network
SDLC	SAP Security Development Lifecycle
SFAPI	SuccessFactors Application Programming Interface
SFTP	SSH File Transfer Protocol Secure File Transfer Protocol
SHA	Secure Hash Algorithm

Abbreviation	Long Text
SMTP	Simple Mail Transfer Protocol
SMTPS	Simple Mail Transfer Protocol Secure
SOAP	Simple Object Access Protocol
SRM	Supplier Relationship Management
SSH	Secure Shell
SSL	Secure Socket Layer
SSO	Single Sign-On
SWIFT	Society for Worldwide Interbank Financial Telecommunication
TCO	Total Cost of Ownership
TLS	Transport Layer Security
TMN	Tenant Management Node
UDF	User-Defined function
UI	User Interface
UN/CEFACT	United Nations Centre for Trade Facilitation and Electronic Business
UN/EDIFACT	United Nations Electronic Data Interchange for Administration, Commerce and Transport
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VDA	Verband der Automobilindustrie (German Association of the Automotive Industry)
VM	Virtual machine
W3C	World Wide Web Consortium
Web UI	Web User Interface
WS-Security	Web Services Security
WSDL	Web Services Description Language

Abbreviation	Long Text
XAdES	XML Advanced Electronic Signatures
XI	Exchange Infrastructure
XSD	XML Schema Definition
XML	Extensible Markup Language
XMLNS	XML Namespace
XPath	XML Path Language
XSLT	Extensible Stylesheet Language Transformations

Appendix B

Literature

The following sections list the literature you may find helpful in your continued exploration of SAP Cloud Platform Integration.

B.1 Books

- Bilay, J. B., Blanco, R. V., *SAP Process Orchestration: The Comprehensive Guide* (Boston: SAP PRESS, 2017).
- Hohpe, G., Woolf, B., *Enterprise Integration Patterns. Designing, Building, and Deploying Messaging Solutions* (Boston: Addison Wesley, 2004).
- Ibsen, C., Anstey, J., *Camel in Action* (Stamford: Manning Publications, 2011).
- Schmeh, K., *Cryptography and Public Key Infrastructure on the Internet* (Chichester, United Kingdom: John Wiley, 2003).
- Stiehl, V., *Process-Driven Applications with BPMN* (New York: Springer, 2014).
- Wood, J., et al, *Getting Started with SAP HANA Cloud Platform* (Boston: SAP PRESS, 2015).

B.2 SAP Cloud Platform Integration Related Links

- SAP Cloud Platform Integration landing page
<http://scn.sap.com/docs/DOC-40396>
- SAP API Business Hub containing the integration content
<https://api.sap.com/shell/integration>
- SAP Cloud Platform on *help.sap.com*
<https://help.sap.com/viewer/p/CP>
- SAP Cloud Platform Integration on *sap.com*
www.sap.com/products/hana-cloud-integration.html

- SAP Cloud Platform Integration on *help.sap.com*
https://help.sap.com/viewer/p/CLOUD_INTEGRATION
- SAP Cloud Platform Integration tools for adapter development
https://tools.hana.ondemand.com/#cloudintegration
- SAP Cloud Platform Integration Editions and components
https://cloudplatform.sap.com/support/service-description.html#section_11
- SAP Development Tools for Eclipse Oxygen
https://tools.hana.ondemand.com/oxygen
- SAP Data Center Locations
www.sap.com/about/cloud-trust-center/data-center.html
- SAP S/4HANA Cloud on *help.sap.com*
https://help.sap.com/viewer/p/SAP_S4HANA_CLOUD

B.3 General Online Sources

- Accredited Standards Committee X12 (ASC X12)
www.x12.org
- Anstey, J.: *Apache Camel: Integration Nirvana*
http://architects.dzone.com/articles/apache-camel-integration
- Apache Camel:
 - Apache Camel Homepage
http://camel.apache.org
 - Apache Camel Documentation
http://camel.apache.org/documentation.html
 - Apache Camel Simple Expression Language
http://camel.apache.org/simple.html
 - Writing components
http://camel.apache.org/writing-components.html
 - List of available components
http://camel.apache.org/components.html
- Cryptographic Message Syntax
https://tools.ietf.org/html/rfc2315
- Eclipse
www.eclipse.org

- Eclipse Oxygen
www.eclipse.org/oxygen
- Enterprise Integration Patterns
www.enterpriseintegrationpatterns.com
- GeoTrust
www.geotrust.com
- GNU Privacy Guard for Windows (Gpg4win), including Kleopatra (a certificate manager and a universal crypto GUI)
www.gpg4win.de/index.html
- International Standardization Organization (ISO)
www.iso.org
- ISO 20022 standard (defines XML specifications for messages exchanged in the course of financial transactions)
www.iso20022.org
- ISO/IEC 27001:2013 standard (provides requirements for an information security management system [ISMS])
www.iso.org/iso/iso27001
- Mendelson AS2
<http://as2.mendelson-e-c.com>
- Northwind OData service
<http://services.odata.org/Northwind/Northwind.svc>
- OAuth 1.0
<https://tools.ietf.org/html/rfc5849>
- OAuth 2.0
<https://tools.ietf.org/html/rfc6749>
- OAuth 2.0 authorization framework
<http://oauth.net>
- Oorsprong Web services
<http://webservices.oorsprong.org/websamples.countryinfo/CountryInfoService.wso?WSDL>
- Open Pretty Good Privacy (OpenPGP)
<https://tools.ietf.org/html/rfc4880>
- Postman
www.getpostman.com

- RFC 5280 (format of the distinguished name [DN])
<https://tools.ietf.org/html/rfc5280>
- S/MIME
<https://tools.ietf.org/html/rfc5751>
- SoapUI
www.soapui.org
- United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT)
www.unece.org/cefact.html
- United Nations Electronic Data Interchange for Administration, Commerce and Transport (UN/EDIFACT)
www.unece.org/cefact/edifact/welcome.html
- Web Services Security (WS-Security)
www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
- XML Advanced Electronic Signatures (XAdES)
www.w3.org/TR/XAdES
- XML Signature
www.w3.org/TR/xmldsig-core

B.4 SAP Communities

- SAP Cloud Platform Integration
www.sap.com/community/topic/cloud-integration.html
- SAP Process Orchestration
www.sap.com/community/topic/process-orchestration.html

Appendix C

The Authors



John Mutumba Bilay studied computer engineering and finance at the University of Cape Town, South Africa. After completing his studies, he started his career as a software engineer. He currently works as a senior software engineer and enterprise integration consultant at Rojo Consultancy B.V. in the Netherlands. With more than 14 years of international experience in information technology, he has spent the last years primarily focused on integration technologies. His SAP specialities include SAP integration- and process-related technologies, including SAP Process Orchestration and SAP Cloud Platform Integration.

In addition to his daily integration work, he provides integration-related training for SAP and for Rojo Consultancy B.V. John is the author of *SAP Process Orchestration: The Comprehensive Guide* (SAP PRESS, 2017) and one of the co-authors of *Getting Started with SAP HANA Cloud Platform* (SAP PRESS, 2016).



Dr. Peter Gutsche studied physics at Heidelberg University, Ruperto Carola. After completing his Ph.D., he joined SAP in 1999. As a technical author, Peter was involved in many knowledge management projects related to SAP's interface and integration technologies. Today, as a knowledge architect, he is responsible for the product documentation of SAP Cloud Platform Integration and works on documentation concepts for cloud software.

Peter is a seasoned technical author with wide-ranging experience in the fields of SAP Process Integration and SAP Cloud Platform Integration.



Mandy Krimmel studied engineering at Humboldt University, Berlin, Germany. In 1998, she started her professional career at a research institute of the German state of Baden-Württemberg, where she was responsible for the technical evaluation of various EU research projects. 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, helping to shape the development of the SAP Cloud Platform Integration product portfolio.

Mandy is a valued mentor and an active blogger in the SAP Community on various cloud integration-related topics. She is one of the co-authors of *SAP NetWeaver Process Integration* (SAP PRESS, 2010).



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 in 2004. As chief product expert, Volker was responsible for the success of the products SAP Process Orchestration, SAP Process Integration, and SAP HANA Cloud Integration. He left SAP in 2016 and accepted a position as Professor at the Ingolstadt Technical University of Applied Sciences, where he currently teaches business information technology. His research focuses on ways to help companies benefit from digitalization using the process-driven approach.

In September 2011, Volker received his Ph.D. degree from the University of Technology Darmstadt. His thesis was on the systematic design and implementation of applications using BPMN. Volker is also the author of *Process-Driven Applications with BPMN* (Springer, 2014) and a regular speaker at various national and international conferences.

Index

A

Absolute path	269, 273, 276
Access logs	565
Account member	661
Acknowledgement	471, 472, 478
Adapter	140, 145
<i>Ariba adapter</i>	46
<i>Facebook adapter</i>	46
<i>HTTP adapter</i>	46
<i>IDoc (SOAP) adapter</i>	46
<i>JMS adapter</i>	360, 428
<i>mail adapter</i>	46, 354
<i>mail receiver adapter</i>	95
<i>mail sender adapter</i>	109
<i>OData adapter</i>	46, 353
<i>ProcessDirect adapter</i>	378
<i>SFTP adapter</i>	46
<i>SOAP adapter</i>	46
<i>SuccessFactors adapter</i>	46
<i>Twitter adapter</i>	46, 692
<i>type</i>	95, 186, 296
Adapter Development Kit	411, 412
Adapter development, Maven support	418
Adapter-specific endpoints	129
ADK	411, 412
Aggregation algorithm	272
AK3	481
AK4	481
AK5	478, 479
Alias	529, 534, 535
ANSI ASC X12	436
Apache Camel	37, 65, 155, 156, 158, 166, 168, 174, 193, 202, 252, 261, 287, 365, 411
<i>Camel route</i>	67
Application edition	50
Application ID	505
Ariba	137
Ariba Network	48, 141, 144–146, 750
<i>content</i>	144
Artifact	114, 117–119, 125, 126, 128, 129, 132, 133, 147–149, 505
<i>STATUS</i>	499
<i>TYPE</i>	499
<i>user credentials</i>	99
Artifact details	501
AS2 adapter	433, 460, 555, 571
AS2 Receiver	473
AS2 Sender	460
AS2 Sender Adapter	461
AS4 adapter	433
ASC X12	431
Asymmetric key technologies	666
Asynchronous communication	250
Asynchronous decoupling	308
Asynchronous message handling	157, 281, 282
Asynchronous receiver	295, 307
Atom	193, 203
Attachment	156
Audit log	566, 653
Authentication	189, 673
<i>basic</i>	681
<i>basic authentication</i>	673
<i>client certificate</i>	681
<i>client certificate-based</i>	674
<i>inbound</i>	678
<i>OAuth</i>	674, 682
<i>outbound</i>	680
<i>type</i>	163
Authorization	68, 94, 266, 673
<i>client certificate</i>	679, 688
<i>user role</i>	266, 679
Authorization group	647, 658
<i>assign to user</i>	661
<i>business expert</i>	658
<i>integration developer</i>	659
<i>system developer</i>	659
<i>tenant administrator</i>	658
Avoiding hardcoded values	176

B

B2B 431
 B2B capabilities 432
 B2B integration 35, 431
 B2B Scenario 451
 Backup 530, 532
 Basic authentication 98, 163, 172
 Body 98, 165, 166, 254, 256, 287, 372
 BPMN 153–155, 253, 299
 Branch 515, 517
 Bug fixes 495
 Bulk message 271
 Business Process Model and Notation 153

C

Call 306
 Camel component 411, 412
 Capacity 562
 CBR 252–254, 258, 259, 261, 262, 367, 373
 Certificate 522, 523, 527
 chain 530, 670
 distinguished name 671
 Certificate-to-user mapping 523, 527, 536,
 537, 679, 684, 685, 687
 Channel 163
 Check mail addresses 546
 Child process 366–368, 370, 372
 ChildCount 515, 517
 ChildrenCounter 515
 Client certificate 189
 Cloud 39
 computing benefits 32
 on-premise integration 39
 platform 494
 strategy 35
 Cloud Connector 189
 Cloud connector 467
 Cluster 647
 nodes 57
 tenant cluster 59
 Comma-separated values 203
 Communication 34, 647
 inbound 703
 outbound 703

Completed 505, 521, 522
 messages 519

Condition expression 373

Configure-only 117, 120, 129, 133–135

Connection 79

type 162

Connectivity 45

Connectivity test 523, 692

tile 538

Connector 161

Consumers 562

Content

Ariba Network 144

globalization scenarios 146

SAP Cloud for Customer 140

SAP SuccessFactors 138

Content edit 117, 118, 129

conditions 117

Content enricher 183, 195, 201

Content for globalization scenarios 137

Content modifier 154, 164–167, 184, 185

Content publisher 114

Content reviewer 114

Content-based

router 38

routing 250, 252

ContextName 515

Correlation ID 334, 505, 507, 516

Country 123

CPI default Trace 569

CPU 494

Create user credentials artifact 98

Credential name 97, 98, 100, 101

Credentials 523

CSRF 488, 621

CTS+ 405

Custom adapter 341

Customer workspace 126–128, 136

D

Data flows 59, 129

Data integration 86, 148

Data model 153, 158

Data protection 651

dialog user access 652

Data protection (Cont.)	
<i>message content</i>	651
<i>monitoring data</i>	652
Data storage	647
Data store	309, 549, 550
<i>GET</i>	309
<i>SELECT</i>	309
<i>WRITE</i>	309
Database	44
Dead-Letter Queue	318, 571
<i>handling</i>	571
<i>option</i>	571
Debug	502
Default gate	254
Default route	258, 259
Deployment	131, 170, 525
<i>fast</i>	32
Design	116, 128, 133, 147
<i>view</i>	159
<i>workspace</i>	133
Developer Edition	51
Digest	700
Digital certificate	669
Digital encryption	49
Digital signature	49, 700
Discover	116, 122
Documents	125, 149
Download	527
Download artifact	500
Download logs	513
Dynamic configuration	349, 383
eDocument Framework	146
Email	156
Email receiver	99, 295
Empty start event	369
Encryption	462, 666
End event	155
Endpoint	103, 104, 170, 501
Endpoint URL	171
Enricher pattern	263
Enterprise Integration Pattern	37, 252
<i>example</i>	38
Enterprise Services Builder	204, 262
Enterprise Services Repository	202
Error	515, 525
<i>analysis</i>	175
Escalated	505, 520, 521
ESR	202, 204, 215
Evaluation condition	255
Evaluation sequence	261, 262
Exception	157
Exchange	98, 156, 157, 164, 166, 167, 283, 284, 287, 293, 366, 367, 371, 375
Exchange ID	157
Exchange property	157, 158, 164–167, 173, 350, 360, 366, 368, 429
Exclusive gateway	253, 259, 373
Execution sequence	260, 261, 270
Expiration period	552, 559
Expression type	256–258, 269
External CALL	306

E

Eclipse	412, 413
Eclipse Luna	413
EDI splitter	268, 433, 454, 471, 472
EDI to XML Converter	433, 455
Edit mode	161, 618
EDMX	193
eDocument	
<i>Chile</i>	146
<i>Hungary</i>	146
<i>Italy</i>	146
<i>Peru</i>	146
<i>Spain</i>	146

F

Failed	505, 521
<i>messages</i>	519
Failed Messages	519
Failover	61
faultcode	289
faultstring	289
Field mapping	216
File	129, 149
Fingerprints	531
Fire and forget	288
Flexible pipeline	158, 365

G

- Gateway 367
Gather 263, 264, 271–274, 279, 280, 304
Gather/merge pattern 263
General splitter 266, 267
Globalization scenarios 146
 government formats 146
Graphical mapper 216

H

- Hash function 700
Hash value 700
Header ... 156, 158, 164–166, 173, 184, 185, 253,
 256, 257, 271, 276, 350, 360, 366, 429
 variable 367, 373
Health check 49, 494, 574
High availability 61
 failover 61
Host key 543
HTTP 411
HTTP Access Log 569
HTTP session handling 428
HTTPS 49, 539, 647, 671
 connection 647
Hybrid deployment 32

I

- ICA 434
 messages 439
 provisioning 436
 runtime artifacts 435
 user roles 437
id_dsa 530
id_rsa 530
Identity provider 658
IDoc 411, 436, 439
 adapter 739
 splitter 268
IDoc Adapter 466
IMAP 539, 545–547
Industries 123
Info 502
Innovation, fast 33

- InOnly 157, 284, 293, 294
InOut 157, 284, 287
In-progress repository 570
Integration 32, 34
Integration artifact 63
 integration flow 63, 67
Integration bus 36, 37, 43
Integration content 48
 artifacts 125
 consume 126
 deploy 131
 design 74
 documents 125
 perform an update 118
 preconfigured 35
 predefined 48
 tags 125, 126
 terms and conditions 129
 transport 403
Integration Content Advisor ... 71, 72, 432, 434
Integration Content Catalog 39, 48, 72,
 113–118, 121, 122, 124, 127, 137–141, 146,
 147, 150, 151, 575, 641
 accessing 115
 catalog 114
 content 114
 discover 117
 existing package 122
 package 114
 prebuilt integration flows 114
 search 122
 templates 114
 via a publicly accessible URL 115
 via your own tenant 115
 web-based application 115
Integration content monitor 174
Integration developer 114
Integration engine 166, 269, 283, 365
Integration flow 56, 64, 129, 515
 change 130
 channel 64
 component version 388
 connectivity details 130
 creation 84
 definition of 64
 demo example 80

Integration flow (Cont.)	
<i>deployment</i>	67
<i>design</i>	64
<i>development</i>	80
<i>display</i>	130
<i>download</i>	131
<i>history</i>	132
<i>revert to an older version</i>	132
<i>runtime configuration</i>	401
<i>step</i>	64
<i>step types</i>	89
<i>version</i>	132, 133, 135, 136, 140, 145, 391
<i>versions</i>	117, 118, 126, 130–134, 136, 176
Integration flow component upgrading	391
Integration flow development,	
best practices	427
Integration flow modeling	
<i>palette</i>	88
<i>sender adapter</i>	91
Integration flow step	
<i>content modifier</i>	89
<i>decryptor</i>	716
<i>encryptor</i>	718
<i>exception Subprocess</i>	375
<i>exception subprocess</i>	89
<i>general splitter</i>	362
<i>local integration process</i>	89
<i>mapping</i>	89
<i>router</i>	386
Integration middleware	
<i>on-premise</i>	40
Integration package	48
<i>configure</i>	129
<i>create</i>	147
<i>editor</i>	130
<i>latest</i>	122
<i>modify</i>	129
<i>predefined</i>	39
Integration pattern	44
Integration platform	34, 36, 43, 53, 54
<i>resources</i>	55
<i>virtual</i>	57
<i>virtualized</i>	54
Integration process	154
Integration use case	
<i>cloud-to-cloud integration</i>	743
Integration use case (Cont.)	
<i>cloud-to-on-premise integration</i>	738
Integration-as-a-service	43
interchange number	473
Interface determination	158
Intermediate certificate	530
Intermediate error	516
Internet of Things	342
ISO	436
Issuer DN	537
Iterating splitter	267
J	
JavaScript Object Notation	203
JMS	309, 310
<i>Dead-Letter Queue</i>	318
<i>Expiration Period</i>	313
<i>explicit retry</i>	330
<i>monitor</i>	315, 319
<i>queue name</i>	313
<i>receiver</i>	313
<i>RETRY</i>	317
<i>sender</i>	317, 324
JMS adapter	311, 555, 571
JMS Message Broker	310
JMS queue	309
<i>deletion</i>	558
JMS queues	549, 555
<i>deletion</i>	316
JMS resources	560
JMS transaction	
<i>handler</i>	326
Join	304
JSON	203
K	
Key	
<i>lifecycle</i>	729
<i>PGP key pair</i>	709
<i>private</i>	666
<i>public</i>	666
<i>X.509</i>	723
Key pair	523, 527
<i>SAP-owned</i>	726

Keystore	523, 527, 668, 690, 725	Mediation engine	156					
<i>entry, alias</i>	727	Memory	494					
<i>management</i>	721	Mendelson	463					
<i>managing</i>	725	MEP	157, 284, 287, 293, 294					
Keyword	123	Message content	62, 650					
Kleopatra	707	Message exchange	32, 34, 43, 54					
Known hosts	543	Message exchange pattern	157, 265, 284, 286, 287					
L								
Last modified on	537	Message flow	154					
LastErrorModelStepId	516	Message header	368					
Library of type systems	432, 435, 438	Message ID	505, 507					
Line of business	123	Message implementation guidelines	435					
List of entries	262	Message lock	570					
Load balancer	59, 648, 684	Message Locks tile	570, 572					
Local integration process	369–371, 374, 375	Message model	158					
Local process	366, 367, 369, 371	Message monitor	174					
Locks	130, 570	Message monitoring	294					
Log configuration	501, 502	Message processing	54					
Log details	504	Message processing log	61, 290, 294, 506, 508, 515 <i>protection of</i>	652				
Log files tab	568	Message protocol	162					
Log level	502 <i>trace</i>	Message queues	555 <i>monitor</i>	315				
Logs section	508	Message routing	251					
M								
MAG	435, 447	Message size	427					
<i>editor</i>	447	Message statuses	521					
Mail	545	Message type	215					
Manage integration content	498	MessageGuid	516					
Manage Locks section	570	Messaging service	61					
Manage security	523, 536	Metadata	148					
Manage security material	495	Microservice	67					
Manage stores	315, 549, 550, 555	MIG	435 <i>ACTIVATE</i>	447 <i>BUSINESS CONTEXT</i>	444 <i>DIRECTION</i>	443 <i>editor</i>	444 <i>STRUCTURE</i>	444
Manage user and roles	494	ModelStepId	517					
Managing certificate-to-user mappings	536	Modifiable	120					
Managing tiles	519	Modified by	537					
Mapping	158, 201	Modularization	366, 380					
<i>editor</i>	211–213, 215, 216	Monitor	116, 170, 173, 496					
<i>engine</i>	202	Monitor Message Processing	294					
<i>guidelines</i>	435, 447	Monitoring	50, 61, 74, 102, 106					
<i>step</i>	202	<i>dashboard</i>	496					
MDN	461, 462							
Mediated communication	36							

Monitoring (Cont.)

<i>data</i>	651
<i>Integration Content</i>	496
<i>keystore</i>	103
<i>Message Processing</i>	496, 503
<i>runtime messages</i>	495
MPL	290, 427, 508, 515
<i>properties</i>	517
MPL attachment	506, 517, 518
MPL ID	290, 302
Multicast	299, 304
<i>order</i>	305
<i>parallel</i>	299, 300
<i>processing</i>	303
<i>sequential</i>	299, 300, 304
Multimapping	272
Multitenancy	43, 49, 56

N

Namespace	278
Network	648
<i>demilitarized zone</i>	650
<i>sandboxed segment</i>	650
<i>segment</i>	648
New SAP keys	534
Node	516
<i>node failure</i>	61
<i>runtime node</i>	57
<i>tenant management node</i>	57
Notify about update	117, 118, 120
Number of concurrent processes	562
Number or queues	562
Number range	473, 563
Number range object	433, 563

O

OAuth	674, 692
<i>credentials</i>	675, 677
<i>terminology</i>	677
OAuth2 credentials	522
Object management group	153
OData	411
<i>channel</i>	193
<i>connection</i>	187, 203, 279

OData (Cont.)

<i>service</i>	65, 86, 153, 175, 184, 185, 188, 189, 193, 194, 197, 203, 212, 225, 250, 251, 279, 777
OMG	153
One-way	287, 288
<i>message</i>	157, 284
On-premise	31, 39, 114, 467
Open Data Protocol	183
OpenPGP	707
OSGi	429
Out message	157
Out-of-memory	570
OverallStatus	515
Overlapping routing conditions	259
Overview	132
Own content package	147
<i>creating</i>	147
Owner	529

P

Package	159
Package lock	150
Parallel gateway	299, 306
Parallel multicast	299, 300
Parallel processing	270
Parameters	118, 133, 135, 176
Parent process	366, 367, 369–374
Participant	186
Partner directory	432, 482
Passwords	522
Payload	156, 202
Persistence	61
PGP key, user ID	719
PGP keyring	527
PGP public keyring	522, 704
<i>deploy on tenant</i>	718
PGP secret keyring	523, 704
<i>signer user ID</i>	720
PKCS#7/CMS splitter	268
Platform-as-a-service	54
Point-to-point communication	36
Pool	154
POP3	539, 545, 546, 548
Postman	488, 603

Prepackaged integration content	117
118, 147	
<i>import</i>	127
<i>process of consuming</i>	122
<i>update</i>	117
Principal propagation	682
Process call	369–371
Process ID	506, 568
Processing	505, 522
<i>chain</i>	155, 276
<i>settings</i>	292
Product	123
Product profile	74, 395
Professional edition	51
Providers	562
Proxy type	189, 682
Publish content	118
Push-pull pattern	309
Robust One-Way	288
Robust option	288
Role	83, 647, 658, 659
<i>ESBMESSAGING.SEND</i>	93
<i>ESBMessaging.send</i>	83, 659
Root cause analysis	294
Root certificate	530
<i>load balancer</i>	686
Route	155, 157, 204
Routing	202
<i>condition</i>	253, 373
<i>rule</i>	259
<i>sequence</i>	305
Run once	344, 348
Runtime configuration	278, 292
Runtime node	60
Russian doll	456

Q

Query editor ..	183, 189–191, 193, 201, 203, 212
Quick configure	117, 118

R

Receiver	118, 130, 135, 136, 176
<i>determination</i>	158
Relative path	269
Reliable messaging	308
Request message	214, 216
Request-Reply	185–187, 194, 195, 201, 203, 204, 216, 265, 279, 280, 287, 343
Request-response message	157
Resource path	189, 190, 193, 194, 201
Response message	158, 166, 173, 195, 214, 215
Responsibility matrix	494
Restart artifact	500
Restore	533
Result message	167
Retention threshold for alerting	552, 559
Retry	505, 521
<i>messages</i>	519
Robust	292
Robust In-Only	288

S

SaaS administrator	656
SAP BPM	154
SAP Business Process Management	154
SAP Cloud for Customer .	48, 137, 140, 141, 737
<i>content</i>	140
<i>SOAP adapter</i>	140
SAP Cloud Platform	54
<i>account</i>	55
<i>tenant</i>	56
<i>Transport Management Service</i>	405
<i>user management</i>	658
SAP Cloud Platform cockpit	68, 71, 646
SAP Cloud Platform Connectivity	42, 47, 467, 476
SAP Cloud Platform Integration	32, 50, 53, 113, 431, 493, 573–575
<i>architecture</i>	68, 645
<i>connectivity options</i>	139
<i>day-to-day operations</i>	494
<i>monitoring</i>	496
<i>monitoring capabilities</i>	493
<i>operational tasks</i>	494
<i>operations</i>	494
<i>releases updates</i>	495
<i>SAP data center</i>	48, 654
<i>SAP ERP</i>	738

SAP ERP HCM	138	Security standard (Cont.)	
SAP Fiori	743	<i>S/MIME</i>	702
SAP Key History	535	<i>WS-Security</i>	702
SAP Keys	533	<i>XADES</i>	702
SAP MM	144	<i>XML Signature</i>	702
SAP Process Integration	40, 154, 158, 202–204, 215, 216, 218, 262, 272	Send	306
SAP Process Orchestration	40, 396, 494	Sender	118, 130, 134–136, 176
SAP S/4HANA	743	Separation of concerns	366
SAP S/4HANA Cloud Enterprise Edition	743	Sequence flow	168
SAP Solution Manager	154	Sequential Multicast	299, 300, 304
SAP SRM	144	Serial number	537
SAP SuccessFactors	48, 114, 138, 139, 745 <i>adapter</i>	Service Definition	265, 287, 291 <i>WSDL</i>
<i>content</i>	138	Settings	116
<i>discover all packages</i>	139	SFTP	154, 411, 539, 542, 648
<i>employee central</i>	745	Signature	666
<i>integrates onboarding</i>	138	Signing	462
<i>OData API</i>	747	Simple Expression Language	98, 166, 193, 257
<i>query, insert, upsert, update</i>	749	Simple Mail Transfer Protocol	672
<i>SFAPI</i>	747	Simple Object Access Protocol	154
SAP SuccessFactors adapter	747 <i>query operations</i>	SMTP	539, 545, 672
SAP Supplier Relationship Management	144	SOAP	80, 154, 162, 170, 183, 184, 216, 253, 259, 275, 283, 284, 286, 287, 289, 292–295, 307, 341, 366, 411 <i>Processing Settings</i>
SAP Web Dispatcher	739	<i>WSDL</i>	288
SAP_ApplicationID	505	SOAP adapter	162, 163, 752
SAPJMSRetries	331	<i>address</i>	92
Scaling, horizontal	43, 49	SOAP channel	253, 286, 287, 291, 293, 294, 307
Schedule on day	345	SOAP client	82
Schedule to recur	345, 346	<i>authentication</i>	104
Scheduler	135	SOAP data source	341
Script	428	SOAP fault	289, 290
Secure communication, HTTPS	63	SOAP message	103
Secure parameter	523	Software	
Secure Shell (SSH)	49, 672	<i>maintenance</i>	43
Security	44, 63 <i>customer onboarding process</i>	<i>update</i>	49, 62
<i>data storage security</i>	650	<i>upgrade</i>	43
<i>message-level</i>	699	Software updates and maintenance	495
<i>physical</i>	654	Software-as-a-service	33
<i>software development process</i>	655	Splitter	263, 264, 266–271, 273, 274, 276, 279, 280
<i>transport-level</i>	665	<i>pattern</i>	263
Security Material	523, 524	Splitter Validation Error Document	479
Security standard			
<i>OpenPGP</i>	702		
<i>PKCS#7</i>	701		

Splitting tag	267, 269
SRTIDOC	469
SSH	539, 542, 672
SSH key	722
SSH known host	522, 527
Start event	154, 162
Start message event	264
Start Timer event	343
StartTime	515
Status	505, 517, 520
Status details	501, 506
StepID	517
Stop on exception	270
StopTime	515
Stored	525
Streaming	270, 428
Structuring large integration flows	365
Subject DN	537
Subprocess	366–369, 371, 373, 374, 429
Support, hardware and infrastructure	494
Supported platform	123
Symmetric key technology	666
Synchronous communication	183
Synchronous interface	214, 215
Synchronous message handling	157, 281
System Log Files tile	568
T	
Tags	125, 126
Talent management process	138
Templates, copy	128
Temporary data	549
Tenant	116, 122, 129, 131, 133, 136
<i>tenant isolation</i>	56
Tenant administrator	98, 101, 659
Tenant isolation	645, 647
<i>message processing</i>	647
<i>user management</i>	646
Tenant management node	60
Terms and condition	129, 136
Tile settings	520
Tiles	496, 503, 519
Time-based integration flow	343
Timer start event	343, 346, 348, 349, 369
Timer-based message handling	341
Timer-based message transfer	341
TLS	539, 540, 671
Trace	503
Transaction handling	326, 327, 428
Transactions	563
Transport Layer Security	671
Transport node	409
Transport protocol	671
<i>HTTPS</i>	58
Traversal path	510
Troubleshooting	173
Twitter adapter	692, 697
Twitter API	694
U	
UN/CEFACT	436
UN/EDIFACT	436
Undeploy	500
Update available	119
Update package	119
Upgrade	33
URI	184
URL	129, 149, 184, 185, 195
User administration	657
User management	658
User management, dialog user	646
User role	266
Username	522, 537
User-to-role assignment	646
V	
Valid until	537
Validate server certificate	540
Value mapping	86, 129, 148
Variables	553
Variables tile	553
Version history	132
Versioning	388
Virtual machine	56
<i>Java Virtual Machine</i>	56
<i>Java Virtual Machine instance</i>	56
Visibility	551, 554

W

- Web of trust 670
Web service 156
Web UI 70, 366, 412
 Monitoring 75
Working with lists 262
Write Variables 553
WSDL 82, 93, 165, 203, 204, 212, 218, 262,
 285, 287, 291–293
WS-Standard 288, 290, 292

X

- X-CSRF Token 488
XI adapter 550, 555, 571
XML 193, 203, 204, 256–258, 262, 263, 272,
 278, 293, 372–374
 envelope 273
 schema definition 203
 validator 293
XML to EDI Converter 433
XML Validator 456, 457
XPath 166
 expression 256, 269, 273, 274, 276,
 278, 279
XSD 193, 203, 204, 212, 218
XSLT mapping 457

Service Pages

The following sections contain notes on how you can contact us.

Praise and Criticism

We hope that you enjoyed reading this book. If it met your expectations, please do recommend it. If you think there is room for improvement, please get in touch with the editor of the book: willj@rheinwerk-publishing.com. We welcome every suggestion for improvement but, of course, also any praise!

You can also share your reading experience via Twitter, Facebook, or email.

Supplements

If there are supplements available (sample code, exercise materials, lists, and so on), they will be provided in your online library and on the web catalog page for this book. You can directly navigate to this page using the following link: www.sap-press.com/4650. Should we learn about typos that alter the meaning or content errors, we will provide a list with corrections there, too.

Technical Issues

If you experience technical issues with your e-book or e-book account at SAP PRESS, please feel free to contact our reader service: support@rheinwerk-publishing.com.

About Us and Our Program

The website <http://www.sap-press.com> provides detailed and first-hand information on our current publishing program. Here, you can also easily order all of our books and e-books. Information on Rheinwerk Publishing Inc. and additional contact options can also be found at <http://www.sap-press.com>.

Legal Notes

This section contains the detailed and legally binding usage conditions for this e-book.

Copyright Note

This publication is protected by copyright in its entirety. All usage and exploitation rights are reserved by the author and Rheinwerk Publishing; in particular the right of reproduction and the right of distribution, be it in printed or electronic form.

© 2018 by Rheinwerk Publishing, Inc., Boston (MA)

Your Rights as a User

You are entitled to use this e-book for personal purposes only. In particular, you may print the e-book for personal use or copy it as long as you store this copy on a device that is solely and personally used by yourself. You are not entitled to any other usage or exploitation.

In particular, it is not permitted to forward electronic or printed copies to third parties. Furthermore, it is not permitted to distribute the e-book on the Internet, in intranets, or in any other way or make it available to third parties. Any public exhibition, other publication, or any reproduction of the e-book beyond personal use are expressly prohibited. The aforementioned does not only apply to the e-book in its entirety but also to parts thereof (e.g., charts, pictures, tables, sections of text).

Copyright notes, brands, and other legal reservations as well as the digital watermark may not be removed from the e-book.

Digital Watermark

This e-book copy contains a **digital watermark**, a signature that indicates which person may use this copy. If you, dear reader, are not this person, you are violating the copyright. So please refrain from using this e-book and inform us about this violation. A brief email to info@rheinwerk-publishing.com is sufficient. Thank you!

Trademarks

The common names, trade names, descriptions of goods, and so on used in this publication may be trademarks without special identification and subject to legal regulations as such.

All of the screenshots and graphics reproduced in this book are subject to copyright © SAP SE, Dietmar-Hopp-Allee 16, 69190 Walldorf, Germany. SAP, the SAP logo, ABAP, Ariba, ASAP, Concur, Concur Expenses, Concur Tript, Duet, SAP Adaptive Server Enterprise, SAP Advantage Database Server, SAP Afaria, SAP ArchiveLink, SAP Ariba, SAP Business ByDesign, SAP Business Explorer, SAP BusinessObjects, SAP BusinessObjects Explorer, SAP BusinessObjects Lumira, SAP BusinessObjects Roambi, SAP BusinessObjects Web Intelligence, SAP Business One, SAP Business Workflow, SAP Crystal Reports, SAP Early-Watch, SAP Exchange Media (SAP XM), SAP Fieldglass, SAP Fiori, SAP Global Trade Services (SAP GTS), SAP GoingLive, SAP HANA, SAP HANA Vora, SAP Hybris, SAP Jam, SAP Max-Attention, SAP MaxDB, SAP NetWeaver, SAP PartnerEdge, SAPPHIRE NOW, SAP Power-Builder, SAP PowerDesigner, SAP R/2, SAP R/3, SAP Replication Server, SAP S/4HANA, SAP SQL Anywhere, SAP Strategic Enterprise Management (SAP SEM), SAP SuccessFactors, The Best-Run Businesses Run SAP, TwoGo are registered or unregistered trademarks of SAP SE, Walldorf, Germany.

Limitation of Liability

Regardless of the care that has been taken in creating texts, figures, and programs, neither the publisher nor the author, editor, or translator assume any legal responsibility or any liability for possible errors and their consequences.