Unit 5 – Modeling Processes

Developing with SAP Integration Suite

C_CPI_2404

Agenda

- Modeling Integration Flows
- Learning the Basics
- Using Adapters
- Using Mappings
- Using Adapter Outbound Security
- Performing Exception Handling
- Using Scripting
- Using Adapter Inbound Security
- Using Integration Patterns

Business Scenario - Task flow

Business Scenario – Steps involved

1. Create Integration Package and Integration Flow with Timer
2. Add Content Modifier (mock data – Product list)
3. Add General Splitter (split Products for iteration)
4. Add Content Modifier (add Product ID to Exchange Property)
5. Add Request Reply (is Product ID in database)
6. Add Router (If Product ID exists - CONTINUE, else END)

Using Adapters

- Wide variety of pre-built adapters available
- Support various application, transport protocols, message protocols
- Differentiation made between input and output adapters
- Broadly categorized into 2 groups
  - TCP based
  - Non TCP based

Features of OData Adapter

- Query wizard
  - Navigate the interface to be accessed with metadata document
- Page Processing mode
  - Read entries in multiple pages which are processed sequentially
- Automatically removing namespaces
  - Remove namespaces and prefixes automatically

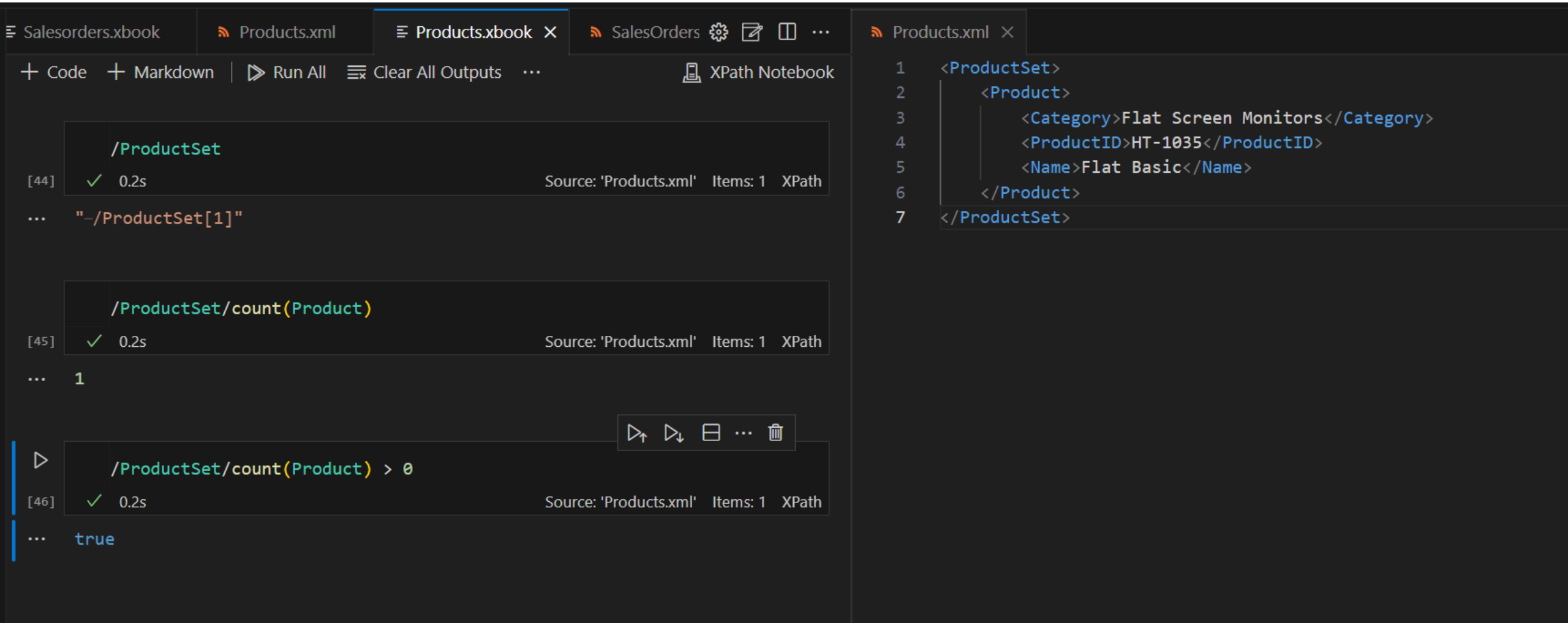
Details of OData Adapter

Example: OData Adapter

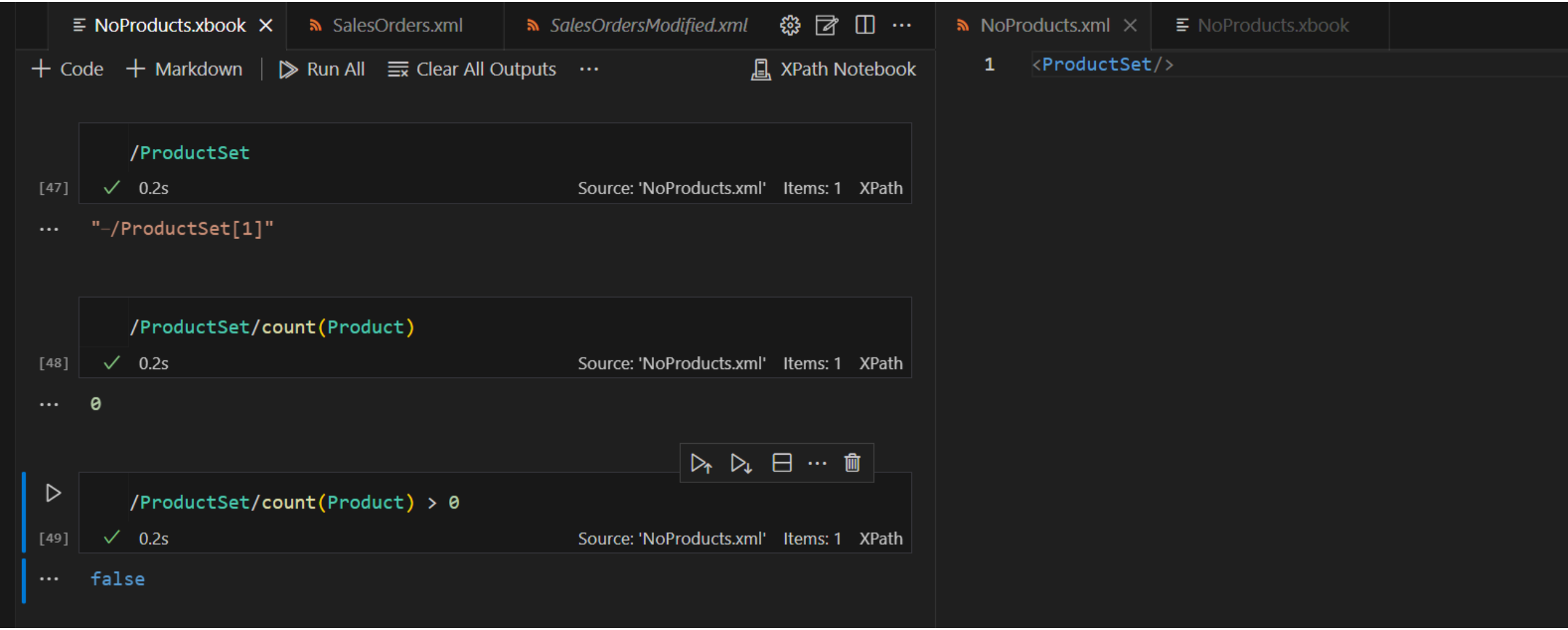
Table 1: Example: Details of an OData Adapter

Detail	Outcome
Category	HTTP based
Transport protocol	TCP/IP
Application protocol	HTTP/HTTPS
Message protocol	Atom Pub as XML or JSON representation

XPath Expressions



XPath Expressions



Business Scenario - Task flow

Business Scenario – Steps involved

7. Add Request Reply (get all Sales Orders for each Product)
8. Add XSLT Mapping (remove namespaces)
9. Add General Splitter (split Sales Order for iteration)
10. Add Content Modifier (add Sales Order ID, Item to Exchange Property)

Business Scenario - Task flow

Business Scenario – Steps involved

11. Add Request Reply (get Sales Header for each Sales Order)
12. Add XSLT Mapping (remove namespaces)
13. Add Content Modifier (add Customer ID to Exchange Property)

HTTP Adapter

SalesOrdersNamespaces.xml | SalesOrders.xml | SalesOrdersModified.xml

```
<feed xmlns="http://www.w3.org/2005/Atom"
    xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metada
    xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices" xml:b
    <id>https://sapes5.sapdevcenter.com/sap/opu/odata/iwbep/GWSAMPLE_BASI
    <title type="text">SalesOrderLineItemSet</title>
    <updated>2023-06-04T07:44:49Z</updated>
    <author>
        <name/>
    </author>
    <link href="ProductSet('HT-1035')/ToSalesOrderLineItems" rel="self" t
    <entry>
        <id>https://sapes5.sapdevcenter.com/sap/opu/odata/iwbep/GWSAMPLE_
        <title type="text">SalesOrderLineItemSet(SalesOrderID='0500000001
        <updated>2023-06-04T07:44:49Z</updated>
        <category term="GWSAMPLE_BASIC.SalesOrderLineItem" scheme="http:/
        <link href="SalesOrderLineItemSet(SalesOrderID='0500000001',ItemP
        <content type="application/xml">
            <m:properties xmlns:m="http://schemas.microsoft.com/ado/2007/
                xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataser
                <d:SalesOrderID>0500000001</d:SalesOrderID>
                <d:ItemPosition>0000000040</d:ItemPosition>
                <d:DeliveryDate>2018-01-07T23:00:00.0000000</d:DeliveryDa
            </m:properties>
        </content>
    </entry>
    <entry>
        <id>https://sapes5.sapdevcenter.com/sap/opu/odata/iwbep/GWSAMPLE_
        <title type="text">SalesOrderLineItemSet(SalesOrderID='0500000007
        <updated>2023-06-04T07:44:49Z</updated>
```

SalesOrdersNamespaces.xbook | NoProducts.xml | NoProducts.xbook

+ Code + Markdown | Run All | Clear All Outputs ... XPath Notebook

```
//content/m:properties/d:SalesOrderID
```

[53] ✓ 0.2s Source: 'SalesOrdersNamespaces.xml' Items: 2 XPath

```
[
    "-/feed[1]/entry[1]/content[1]/m:properties[1]/d:SalesOrderID[1]",
    "-/feed[1]/entry[2]/content[1]/m:properties[1]/d:SalesOrderID[1]"
]
```

```
//content/m:properties/d:ItemPosition
```

[54] ✓ 0.2s Source: 'SalesOrdersNamespaces.xml' Items: 2 XPath

```
[
    "-/feed[1]/entry[1]/content[1]/m:properties[1]/d:ItemPosition[1]",
    "-/feed[1]/entry[2]/content[1]/m:properties[1]/d:ItemPosition[1]"
]
```

Namespaces are not automatically removed in HTTP adapter...
So your XPath needs to take that into account

Mappings

- Message Mapping
  - Mapping editor provides tools to map XML or JSON messages
- XSLT Mapping
  - Language designed for transforming XML docs to other formats
  - Stylesheet is processed by an XSLT processor (Xalon or Saxon)
- Mapping with scripting
- Operation Mapping from Enterprise Service Repository (On-Premise)

Mappings

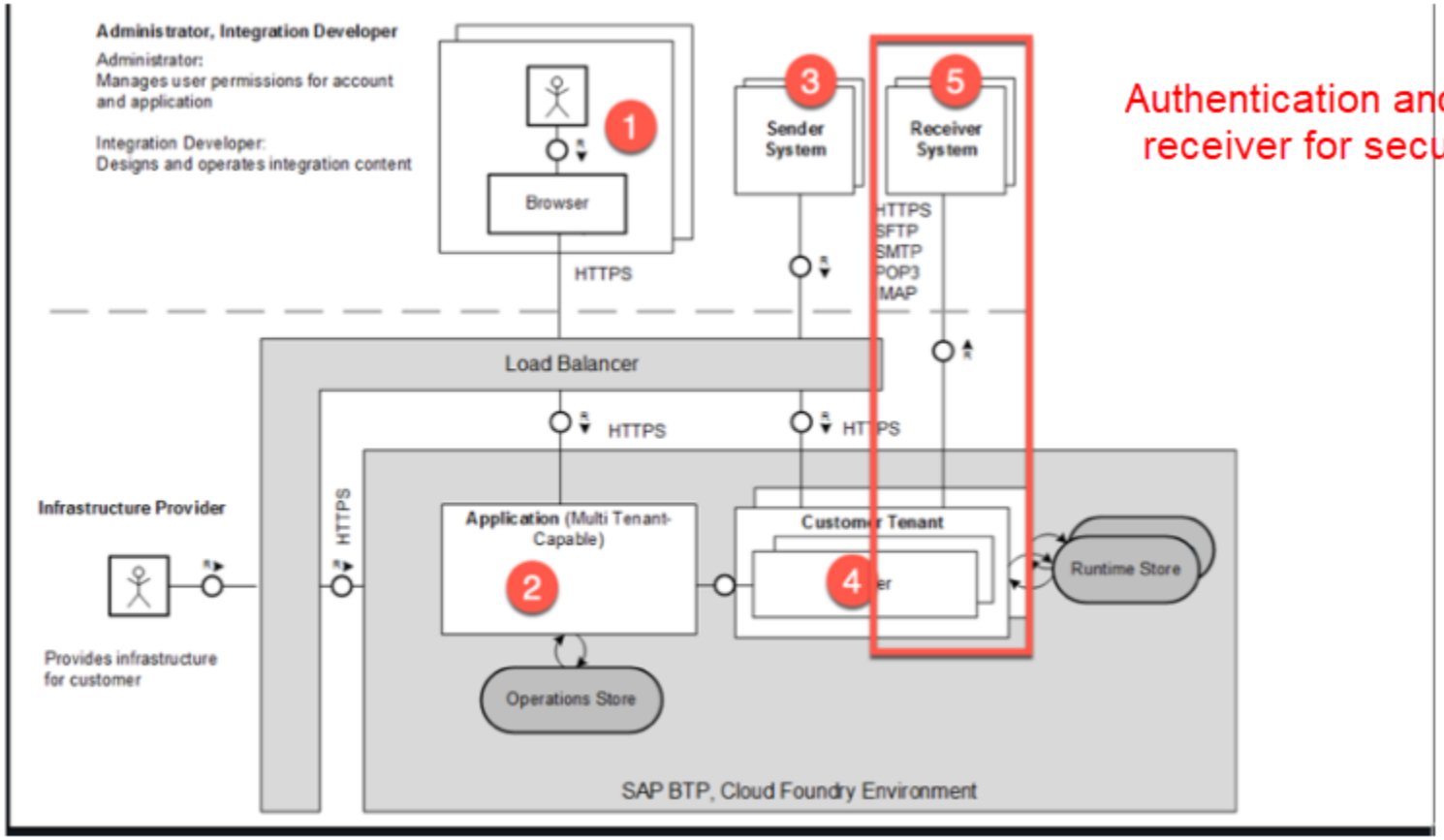
- Process of converting source format into different target formats
- Message Mapping offers context handling, UDF, testing functions
- XSLT Mapping requires XML as input
  - Can create more target formats
  - Useful for creating attachments
- Mapping via scripting offers most flexibility

Business Scenario - Task flow

Business Scenario – Steps involved

14. Add Data Store (write Customer ID to Data Store)

Using Adapter Outbound Security



Authentication and Authorization with receiver for secure communication

Example of a Direct Receiver and Sender Adapter

Options for Authentication / Authorization

- Basic
- Client Certificate
- None
- OAuth2 Client Credentials
- OAuth2 SAML Bearer Assertion

Business Scenario - Task flow

Business Scenario – Steps involved

15. Add Exception Subprocess (barebone)
16. Add Groovy Script (read exception messages to payload)

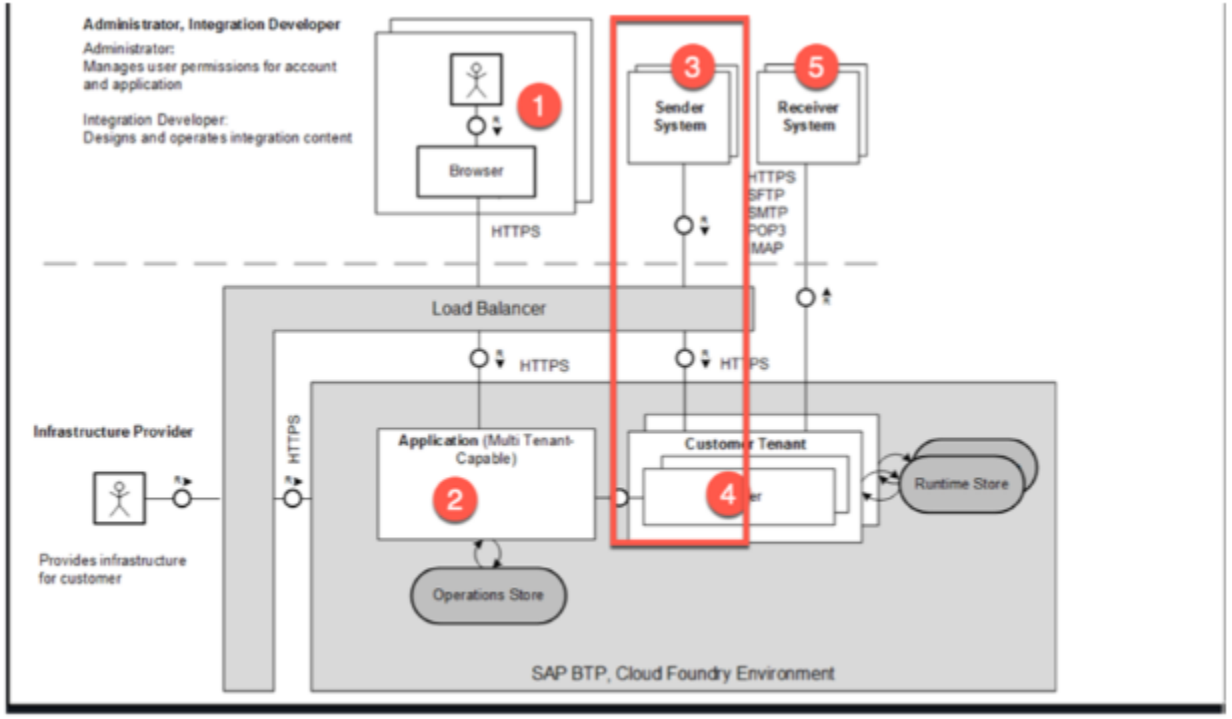
Business Scenario - Task flow

Business Scenario – Steps involved

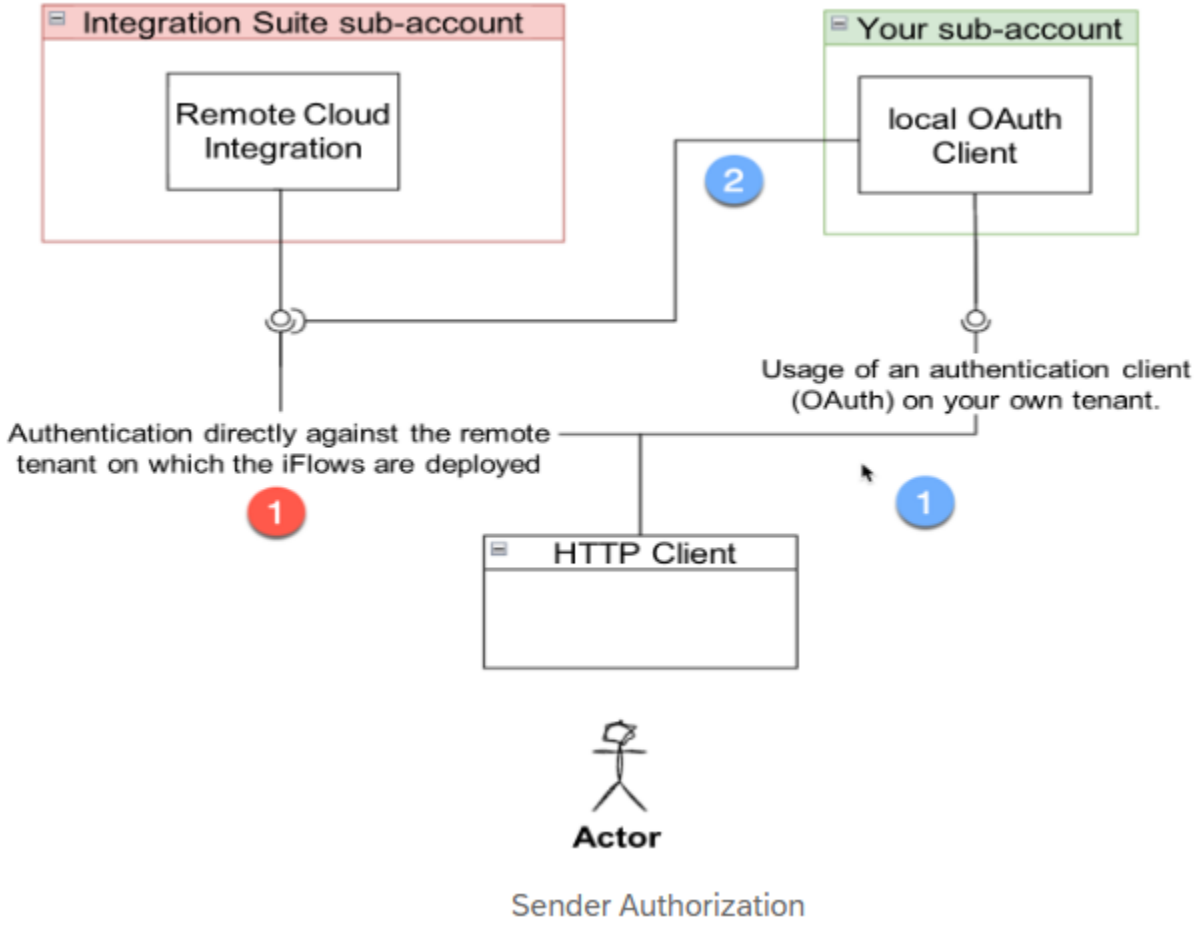
17. Add Message Start (remove Timer)
18. Add SOAP adapter (asynchronous call)
19. Remove $top clause (power of asynchronous call)

Using Adapter Inbound Security

- Certificates between sender and load balancer for HTTPS connection
- Sender's authorization validated against Integration flow endpoint



Authorization of sender



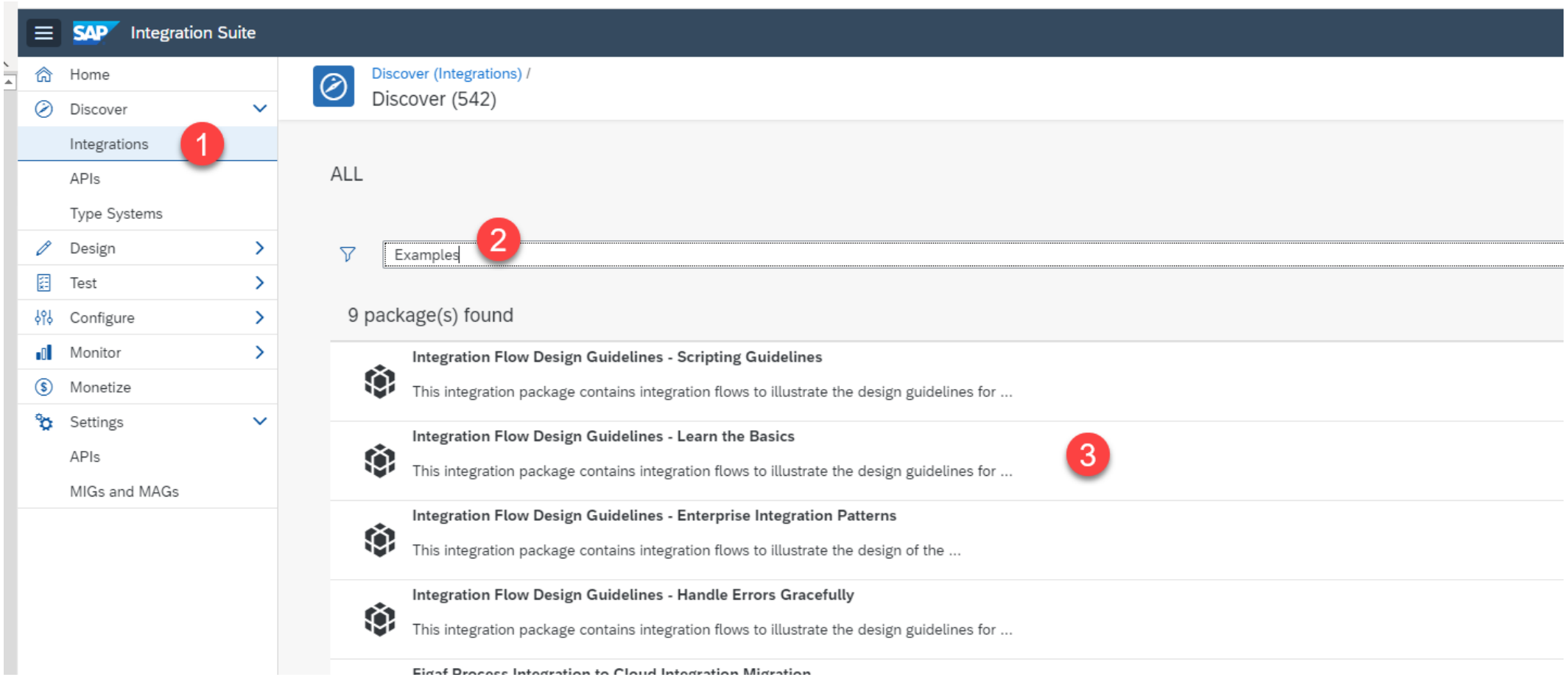
Authentication against remote endpoint

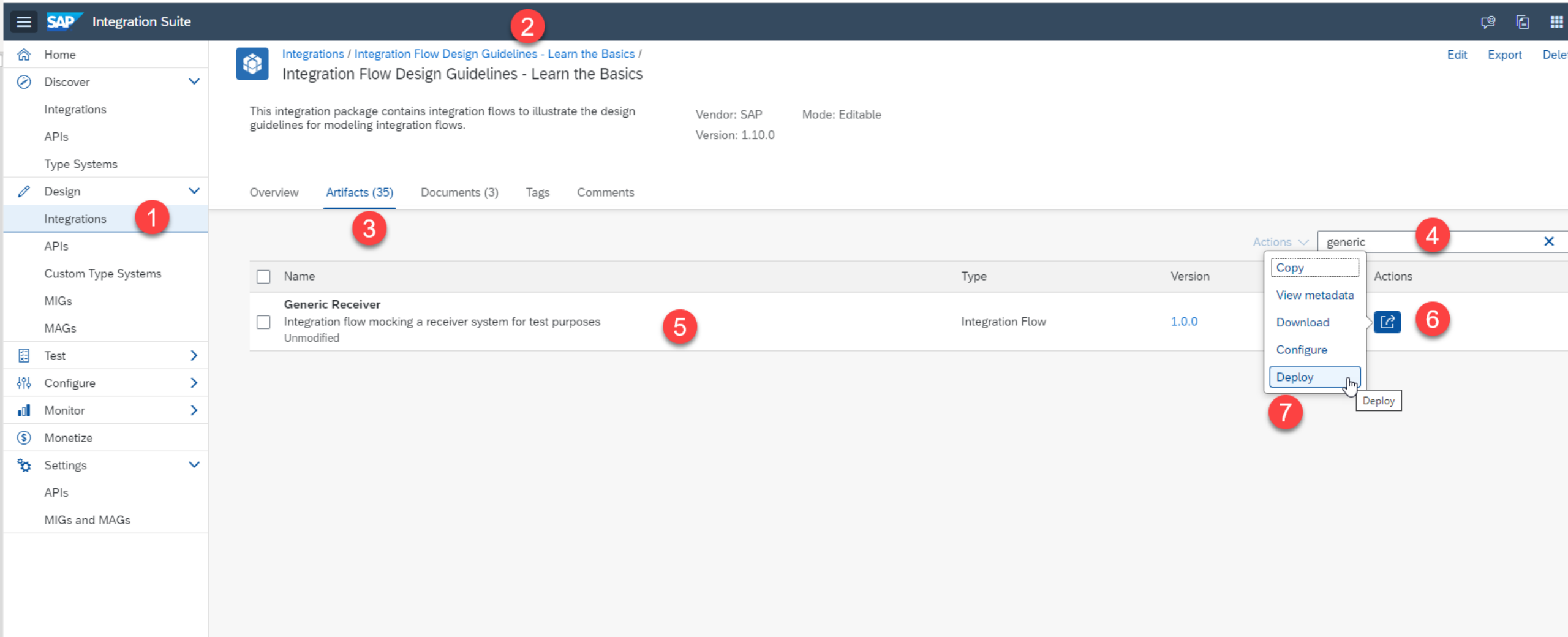
- Assign user role ESBMessaging.send
- Not recommended for production use

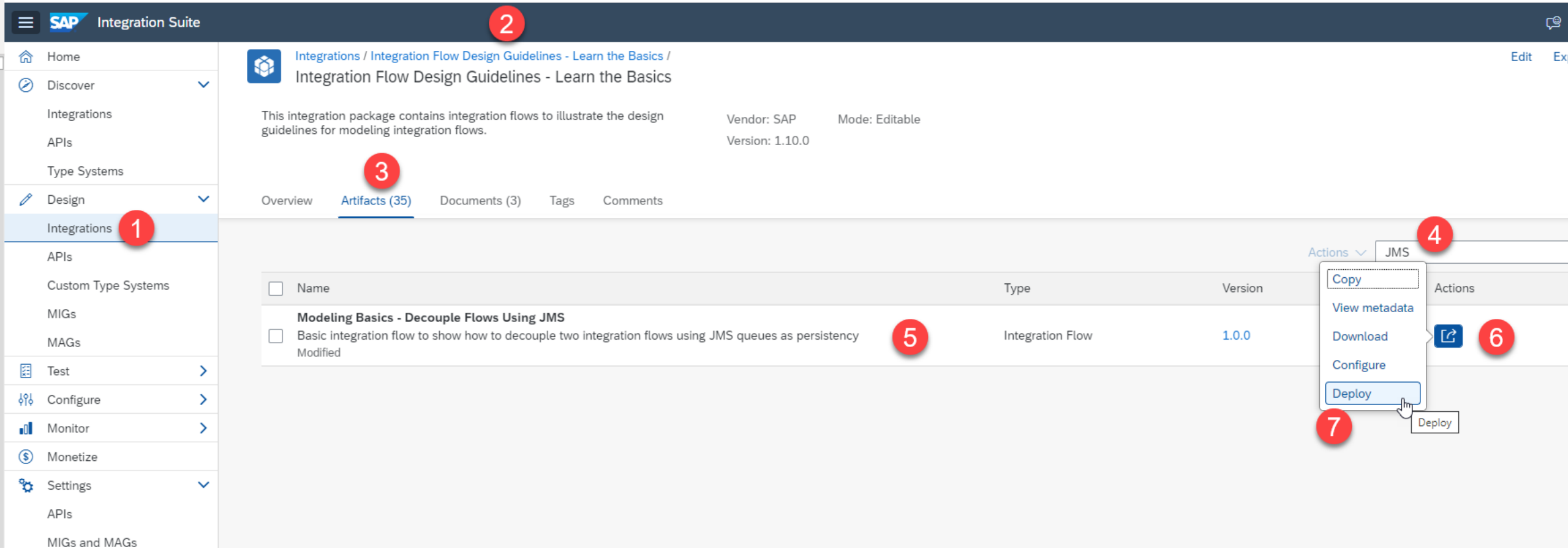
Authentication (Oauth) client on your own Tenant

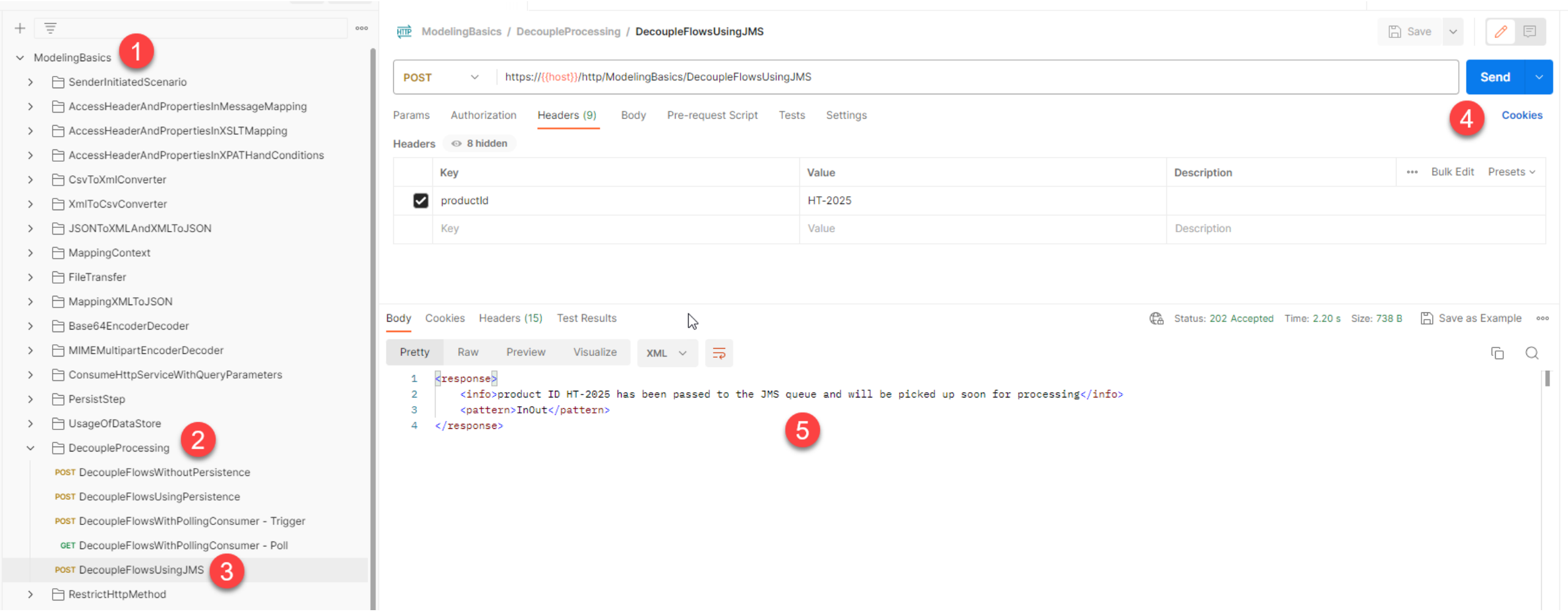
- Set up Process Integration Runtime instance
- Supports
  - Authorization code
  - Client credentials
  - Password
  - Refresh Token
  - SAML2 Bearer
  - JWT Bearer

Sample Integration Flows




SAP Integration Suite
Home
Discover
Integrations
APIs
Type Systems
Design
Integrations
APIs
Custom Type Systems
MIGs
MAGs
Test
Configure
Monitor
Monetize
Settings
APIs
MIGs and MAGs
1
2
Integrations / Integration Flow Design Guidelines - Learn the Basics /
Integration Flow Design Guidelines - Learn the Basics
Edit
Export
This integration package contains integration flows to illustrate the design guidelines for modeling integration flows.
Vendor: SAP
Mode: Editable
Version: 1.10.0
Overview
Artifacts (35)
Documents (3)
Tags
Comments
3
Actions
generic
4
Name
Type
Version
Actions
Generic Receiver
Integration flow mocking a receiver system for test purposes
Unmodified
5
Integration Flow
1.0.0
6
Copy
View metadata
Download
Configure
Deploy
Deploy
7


SAP Integration Suite
2
Home
Discover
Integrations
APIs
Type Systems
Design
Integrations
1
APIs
Custom Type Systems
MIGs
MAGs
Test
Configure
Monitor
Monetize
Settings
APIs
MIGs and MAGs
Integrations / Integration Flow Design Guidelines - Learn the Basics /
Integration Flow Design Guidelines - Learn the Basics
Edit
Ex
This integration package contains integration flows to illustrate the design guidelines for modeling integration flows.
Vendor: SAP
Mode: Editable
Version: 1.10.0
3
Overview
Artifacts (35)
Documents (3)
Tags
Comments
4
Actions
JMS
Name
Type
Version
Actions
Modeling Basics - Decouple Flows Using JMS
Basic integration flow to show how to decouple two integration flows using JMS queues as persistency
Modified
5
Integration Flow
1.0.0
6
Copy
View metadata
Download
Configure
Deploy
7
Deploy


ModelingBasics
1
SenderInitiatedScenario
AccessHeaderAndPropertiesInMessageMapping
AccessHeaderAndPropertiesInXSLTMapping
AccessHeaderAndPropertiesInXPATHandConditions
CsvToXmlConverter
XmlToCsvConverter
JSONToXMLAndXMLToJSON
MappingContext
FileTransfer
MappingXMLToJSON
Base64EncoderDecoder
MIMEMultipartEncoderDecoder
ConsumeHttpServiceWithQueryParameters
PersistStep
UsageOfDataStore
DecoupleProcessing
2
POST DecoupleFlowsWithoutPersistence
POST DecoupleFlowsUsingPersistence
POST DecoupleFlowsWithPollingConsumer - Trigger
GET DecoupleFlowsWithPollingConsumer - Poll
POST DecoupleFlowsUsingJMS
3
RestrictHttpMethod
ModelingBasics / DecoupleProcessing / DecoupleFlowsUsingJMS
Save
POST
https://{{host}}/http/ModelingBasics/DecoupleFlowsUsingJMS
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
4
Cookies
Headers
8 hidden
Key
Value
Description
Bulk Edit
Presets
productId
HT-2025
Key
Value
Description
Body
Cookies
Headers (15)
Test Results
Status: 202 Accepted
Time: 2.20 s
Size: 738 B
Save as Example
Pretty
Raw
Preview
Visualize
XML
1 <response>
2 <info>product ID HT-2025 has been passed to the JMS queue and will be picked up soon for processing</info>
3 <pattern>InOut</pattern>
4 </response>
5

Key Summary Points – Unit 5

Q2. Which object do you use to transform message structure into a specific target structure?

- ✓ XSLT Mapping
- B Message Mapping
- C Value Mapping
- D Content Modifier

✓ Correct

Correct. You use the XSLT Mapping to transform message structure into a specific target structure.

Key Summary Points – Unit 5

Q3. Where can user credentials be configured for secure authentication?

A Monitor → API → Manage Security → Manage Security Material

✓ Monitor → Integrations → Manage Security → Manage Security Material

C Monitor → Integrations → Manage Security → User Role

✓ Correct

Correct. You configure user credentials here: Monitor → Integrations → Manage Security → Manage Security Material.

Key Summary Points – Unit 5

Q6. What role do you need to assign to yourself in order to send a message to your configured endpoint?

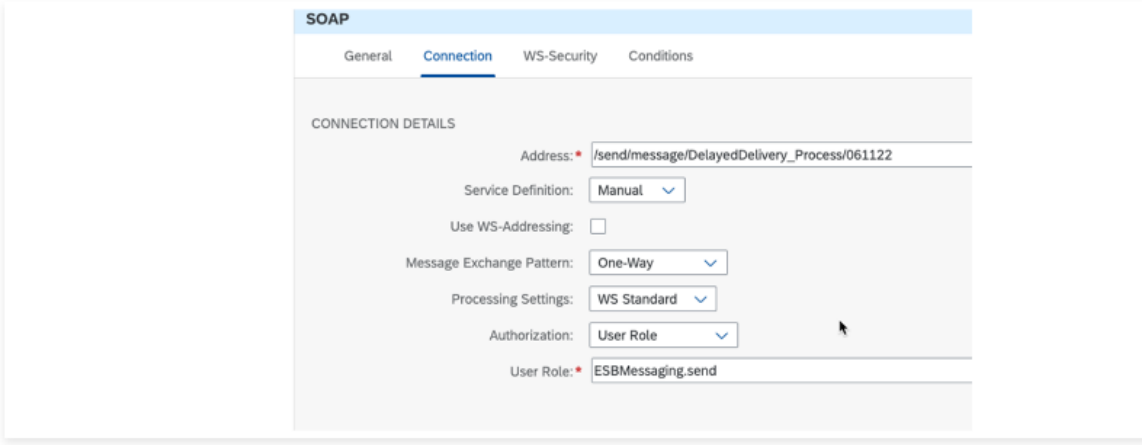
- ✓ ESBMessaging.send
- B Send.To.Endpoint
- C ESB.Messaging.Send
- D HTTP.ESBMessaging.Send

✓ Correct

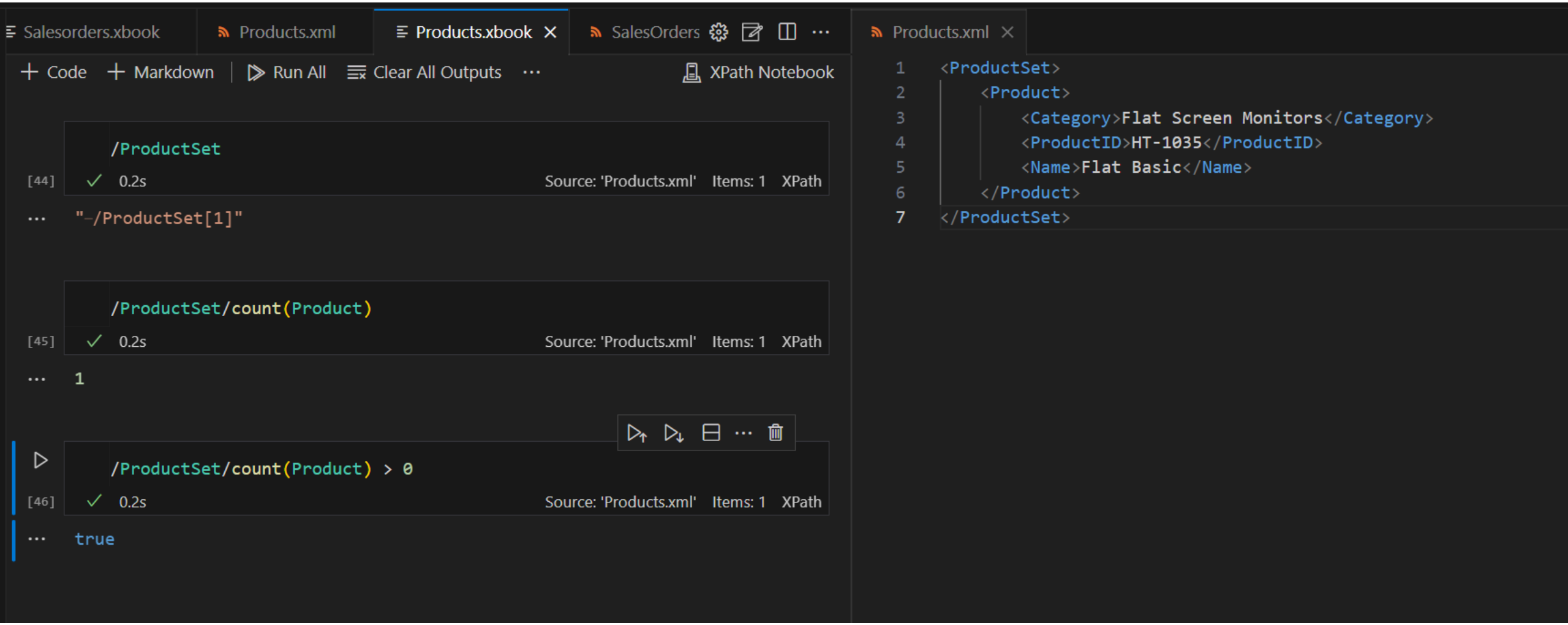
Correct. In order to send a message to your configured endpoint you need to assign the role: ESBMessaging.send.

Key Summary Points – Unit 5

Field Name	Input Data
Address	/send/message/DelayedDelivery_Process/timestamp (must be unique)
Service Definition	Manual
Message Exchange Pattern	One-Way (starting asynchronous)
Processing Settings	WS standard
Authorization	User Role
User Role	ESBMessaging.send



XPath Expressions



HTTP Adapter

SalesOrdersNamespaces.xml | SalesOrders.xml | SalesOrdersModified.xml

```
<feed xmlns="http://www.w3.org/2005/Atom"
    xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metada
    xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices" xml:b
    <id>https://sapes5.sapdevcenter.com/sap/opu/odata/iwbep/GWSAMPLE_BASI
    <title type="text">SalesOrderLineItemSet</title>
    <updated>2023-06-04T07:44:49Z</updated>
    <author>
        <name/>
    </author>
    <link href="ProductSet('HT-1035')/ToSalesOrderLineItems" rel="self" t
    <entry>
        <id>https://sapes5.sapdevcenter.com/sap/opu/odata/iwbep/GWSAMPLE_
        <title type="text">SalesOrderLineItemSet(SalesOrderID='0500000001
        <updated>2023-06-04T07:44:49Z</updated>
        <category term="GWSAMPLE_BASIC.SalesOrderLineItem" scheme="http:/
        <link href="SalesOrderLineItemSet(SalesOrderID='0500000001',ItemP
        <content type="application/xml">
            <m:properties xmlns:m="http://schemas.microsoft.com/ado/2007/
                xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataser
                <d:SalesOrderID>0500000001</d:SalesOrderID>
                <d:ItemPosition>0000000040</d:ItemPosition>
                <d:DeliveryDate>2018-01-07T23:00:00.0000000</d:DeliveryDa
            </m:properties>
        </content>
    </entry>
    <entry>
        <id>https://sapes5.sapdevcenter.com/sap/opu/odata/iwbep/GWSAMPLE_
        <title type="text">SalesOrderLineItemSet(SalesOrderID='0500000007
        <updated>2023-06-04T07:44:49Z</updated>
```

SalesOrdersNamespaces.xbook | NoProducts.xml | NoProducts.xbook

+ Code + Markdown | Run All | Clear All Outputs ... XPath Notebook

```
//content/m:properties/d:SalesOrderID
```

[53] ✓ 0.2s Source: 'SalesOrdersNamespaces.xml' Items: 2 XPath

```
[
    "-/feed[1]/entry[1]/content[1]/m:properties[1]/d:SalesOrderID[1]",
    "-/feed[1]/entry[2]/content[1]/m:properties[1]/d:SalesOrderID[1]"
]
```

```
//content/m:properties/d:ItemPosition
```

[54] ✓ 0.2s Source: 'SalesOrdersNamespaces.xml' Items: 2 XPath

```
[
    "-/feed[1]/entry[1]/content[1]/m:properties[1]/d:ItemPosition[1]",
    "-/feed[1]/entry[2]/content[1]/m:properties[1]/d:ItemPosition[1]"
]
```

Namespaces are not automatically removed in HTTP adapter...
So your XPath needs to take that into account

Details of OData Adapter

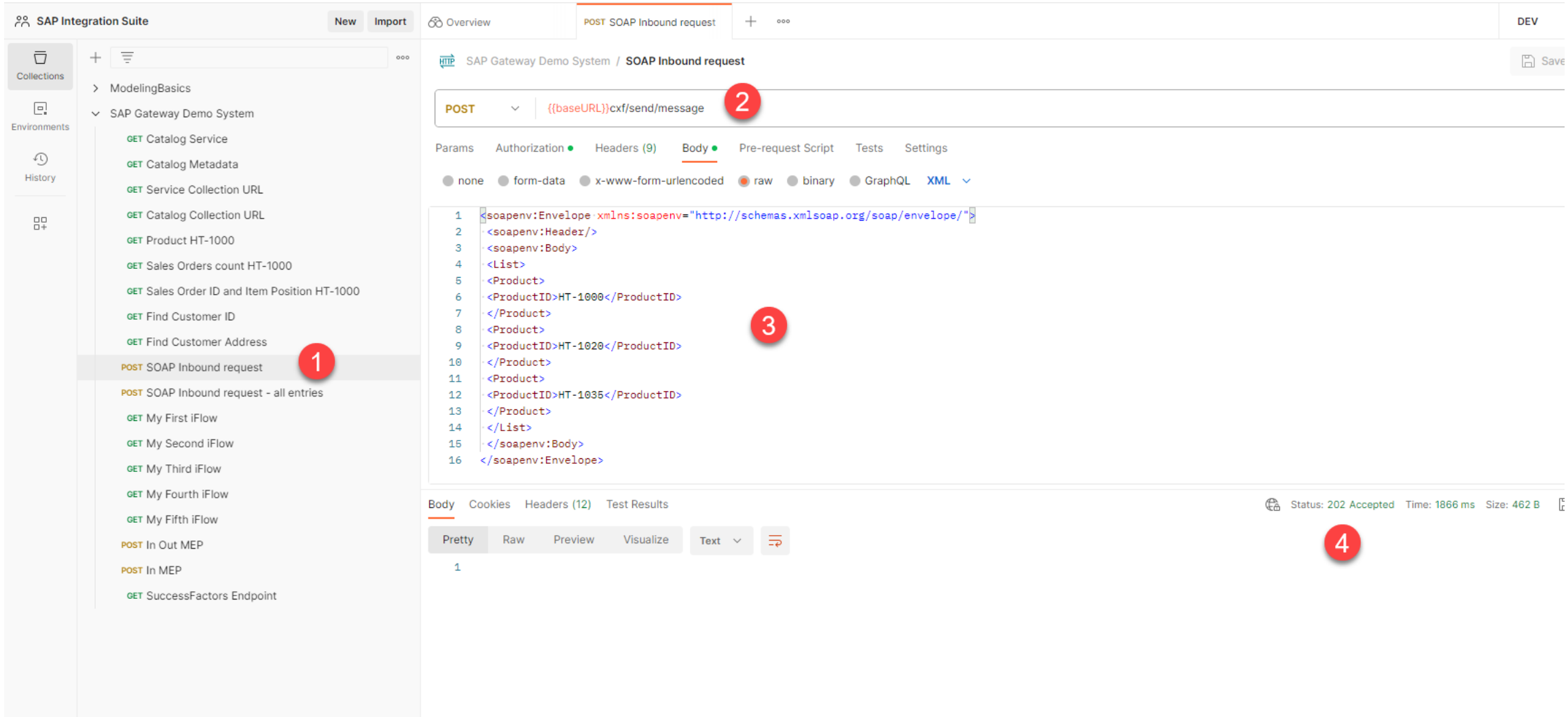
Example: OData Adapter

Table 1: Example: Details of an OData Adapter

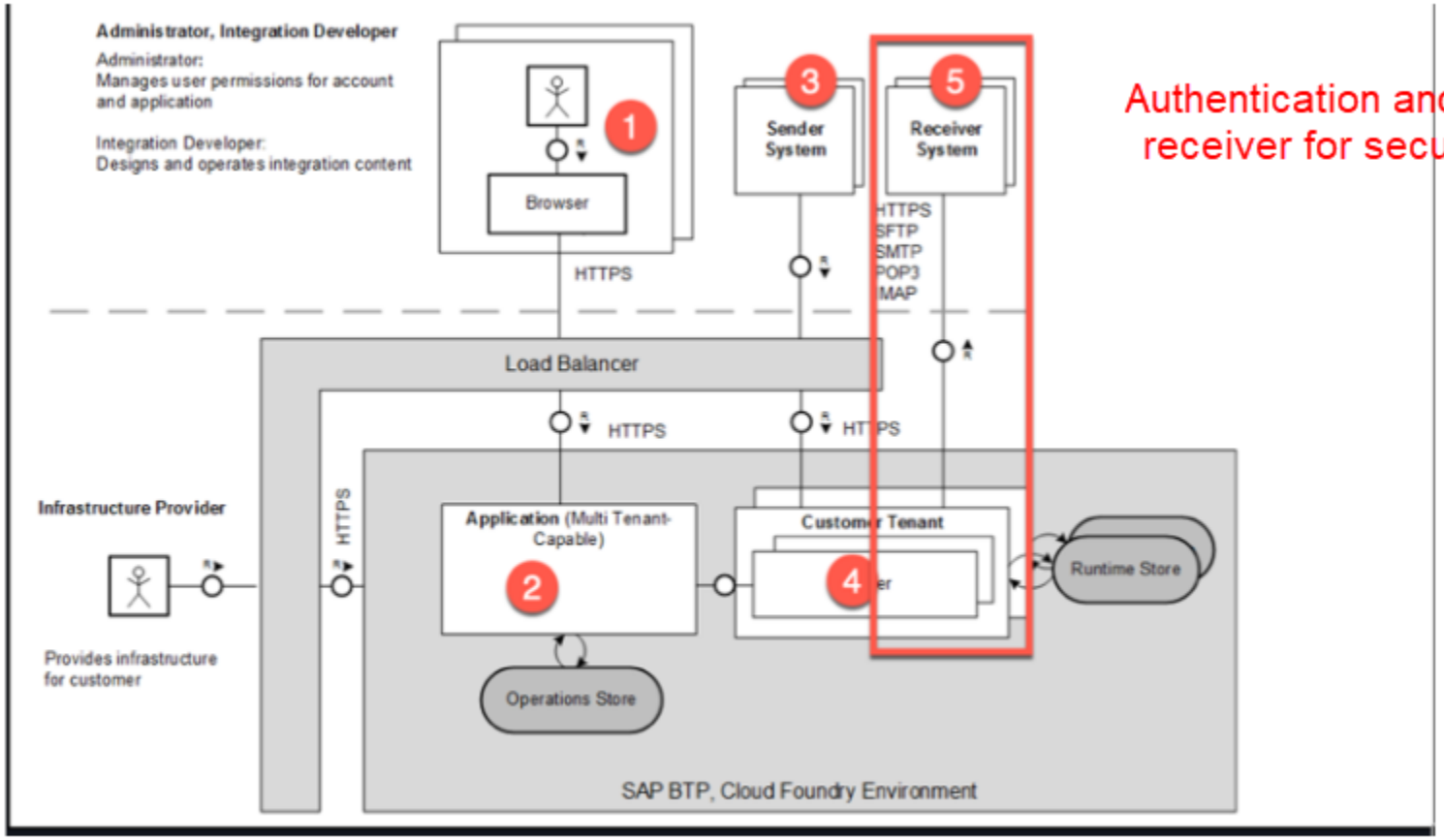
Detail	Outcome
Category	HTTP based
Transport protocol	TCP/IP
Application protocol	HTTP/HTTPS
Message protocol	Atom Pub as XML or JSON representation

Features of OData Adapter

- Query wizard
  - Navigate the interface to be accessed with metadata document
- Page Processing mode
  - Read entries in multiple pages which are processed sequentially
  - Overcome challenges with large number of entries
- Automatically removing namespaces
  - Remove namespaces and prefixes automatically


SAP Integration Suite
New
Import
Overview
POST SOAP Inbound request
DEV
Collections
Environments
History
ModelingBasics
SAP Gateway Demo System
GET Catalog Service
GET Catalog Metadata
GET Service Collection URL
GET Catalog Collection URL
GET Product HT-1000
GET Sales Orders count HT-1000
GET Sales Order ID and Item Position HT-1000
GET Find Customer ID
GET Find Customer Address
POST SOAP Inbound request
POST SOAP Inbound request - all entries
GET My First iFlow
GET My Second iFlow
GET My Third iFlow
GET My Fourth iFlow
GET My Fifth iFlow
POST In Out MEP
POST In MEP
GET SuccessFactors Endpoint
SAP Gateway Demo System / SOAP Inbound request
Save
POST
{{baseURL}}cxf/send/message
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
XML
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Header/>
<soapenv:Body>
<List>
<Product>
<ProductID>HT-1000</ProductID>
</Product>
<Product>
<ProductID>HT-1020</ProductID>
</Product>
<Product>
<ProductID>HT-1035</ProductID>
</Product>
</List>
</soapenv:Body>
</soapenv:Envelope>
Body
Cookies
Headers (12)
Test Results
Status: 202 Accepted
Time: 1866 ms
Size: 462 B
Pretty
Raw
Preview
Visualize
Text
1
2
3
4

Using Adapter Outbound Security



Authentication and Authorization with receiver for secure communication

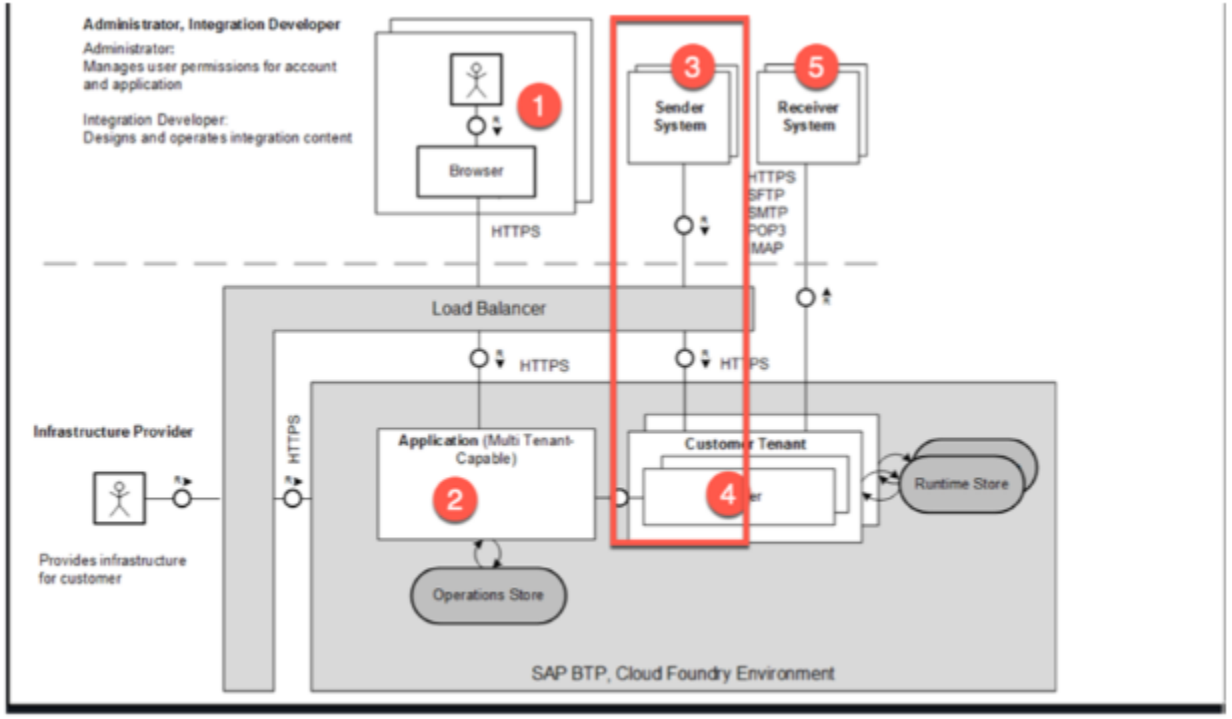
Example of a Direct Receiver and Sender Adapter

Options for Authentication / Authorization

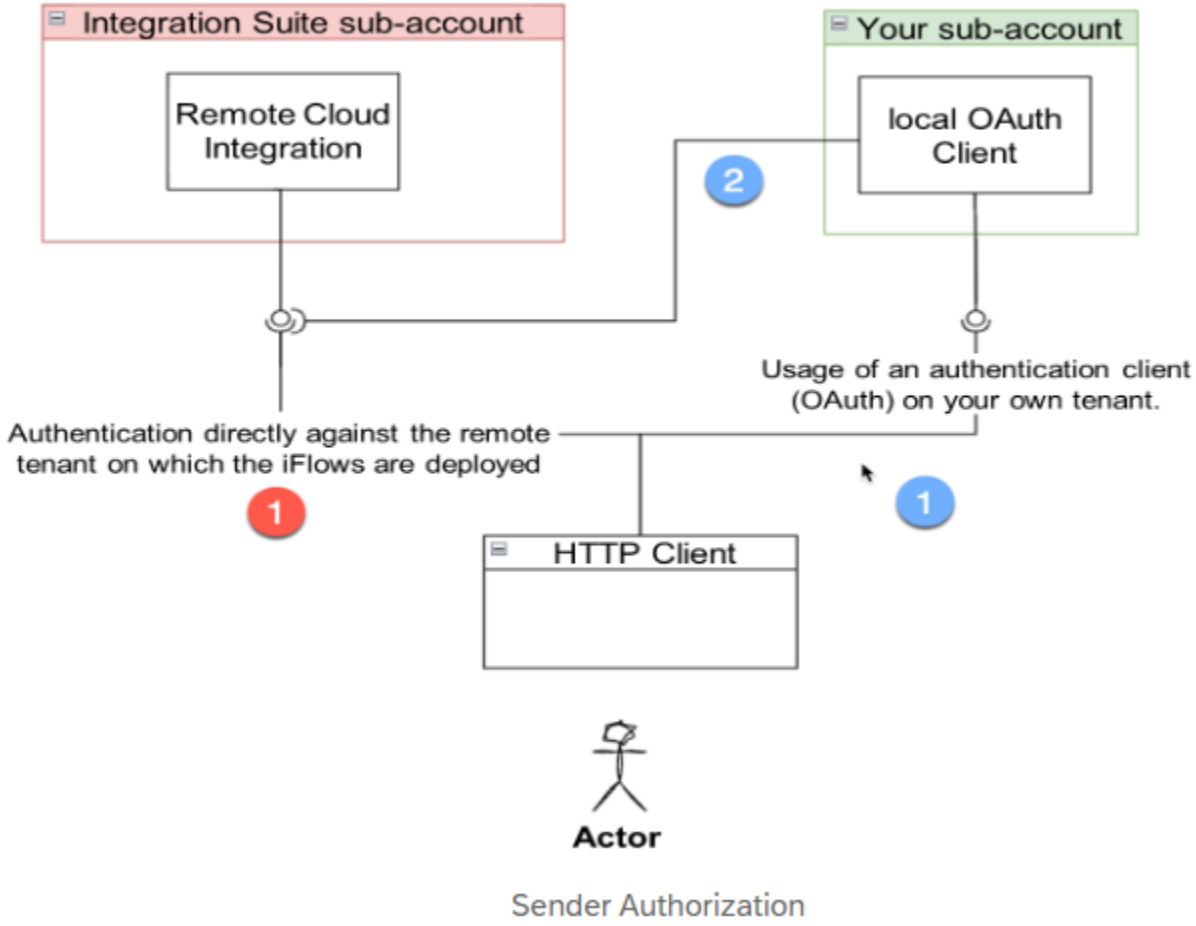
- Basic
- Client Certificate
- None
- OAuth2 Client Credentials
- OAuth2 SAML Bearer Assertion

Using Adapter Inbound Security

- Certificates between sender and load balancer for HTTPS connection
- Sender's authorization validated against Integration flow endpoint



Authorization of sender



Sender Authorization

Authentication against remote endpoint

- Assign user role ESBMessaging.send
- Not recommended for production use

Authentication (OAuth) client on your own Tenant

- Set up Process Integration Runtime instance
- Supports
  - Authorization code
  - Client credentials
  - Password
  - Refresh Token
  - SAML2 Bearer
  - JWT Bearer

Usage of an Authentication (OAuth) Client on your own Tenant

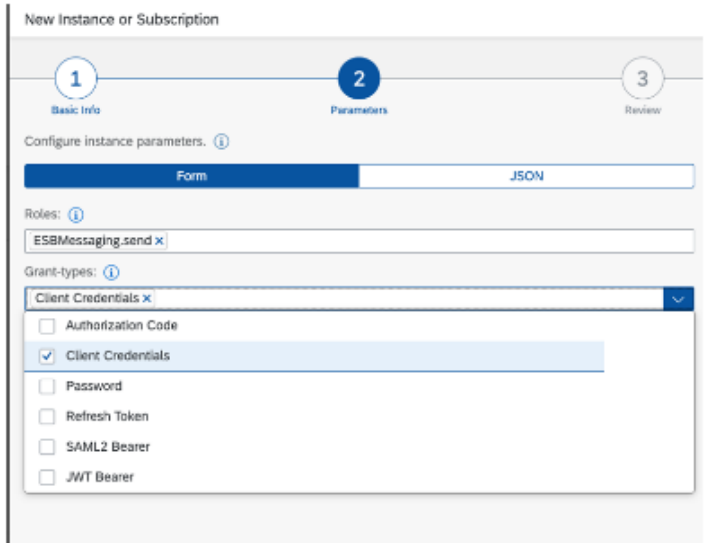
The method of directly calling an integration flow via the role-based approach shown uses personalized users and basic authentication, which are not suitable for productive purposes. For better authentication methods, we need to use a self-configured OAuth2.0 client that can be created on our own subaccount.

To accomplish this, we need to set up a Process Integration Runtime instance on our subaccount, and associate it with the integration flow plan. This instance can then be customized with various client credentials. These correspond to No. 1 and No. 2, marked in blue in the picture above.

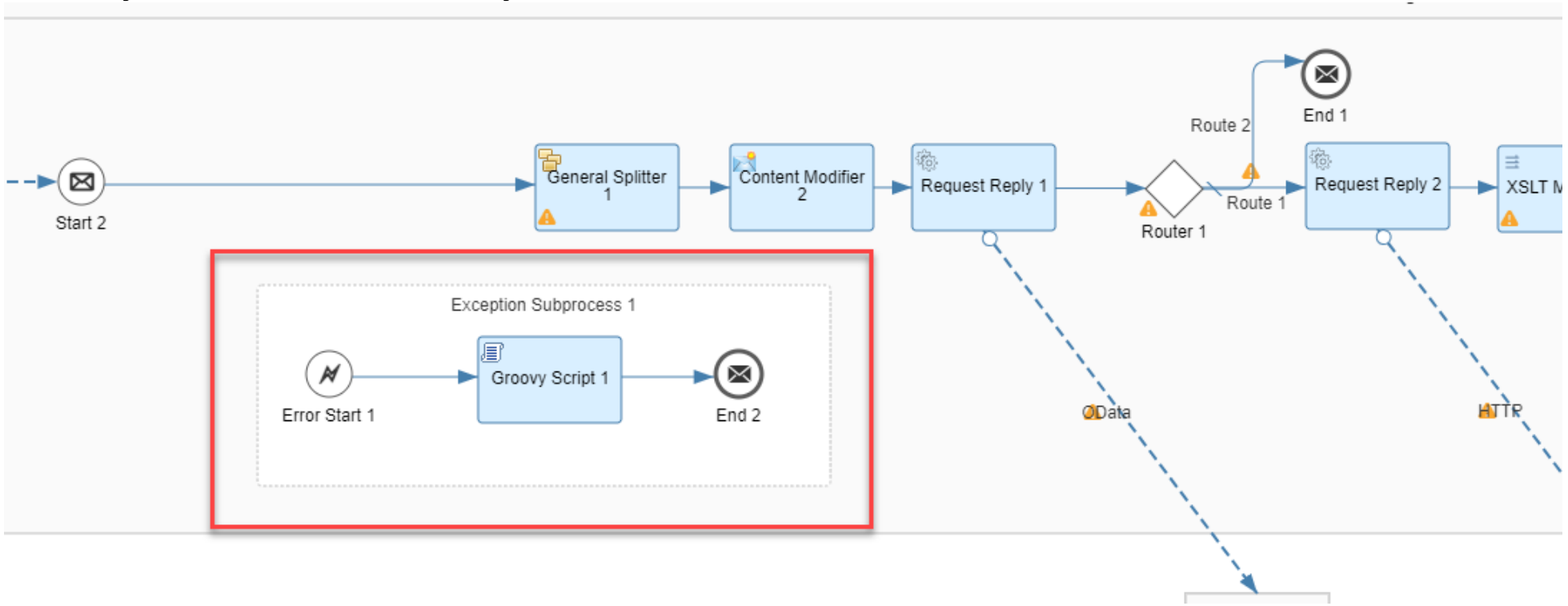
You can choose the following grand-types:

- Authorization Code
- Client Credentials
- Password
- Refresh Token
- SAML2 Bearer
- JWT Bearer

Selection of grand types when configuring the local *Process integration Runtime* instance.



Key Summary Points – Unit 5



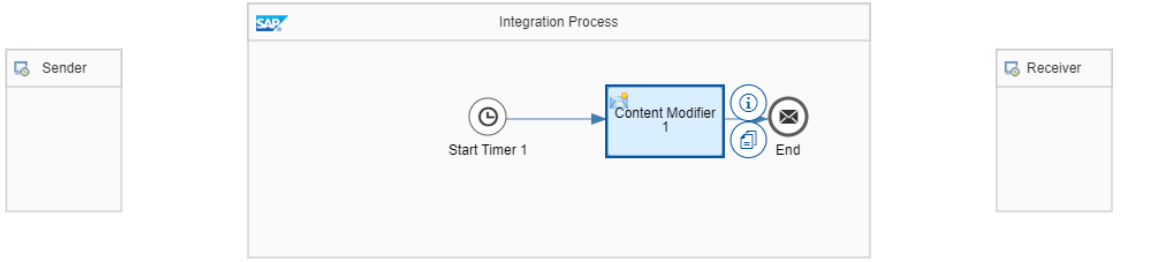
Summary

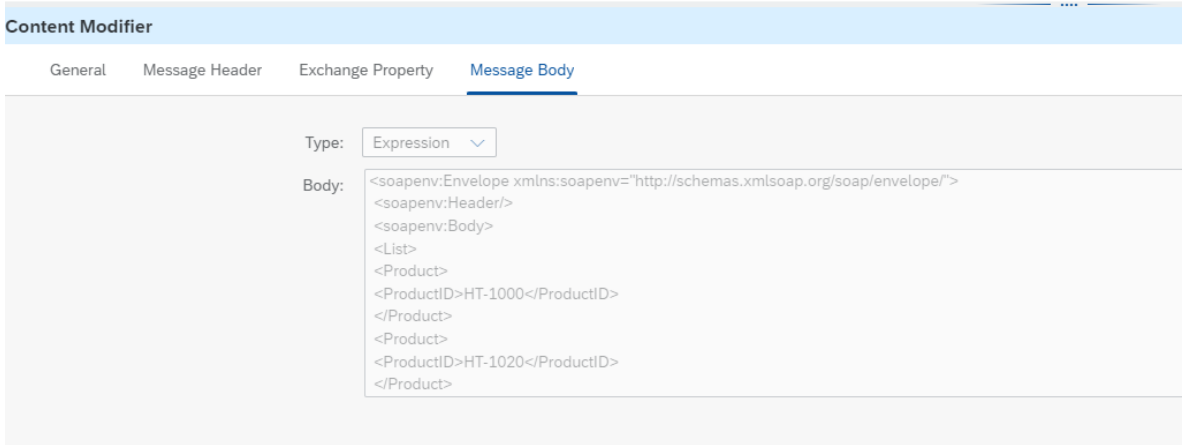
A special error subprocess can intercept an unexpected error using an Exception Start Event. After interception, various processing steps can be implemented. For instance, it would be appropriate to store process values or message content following an error. Additionally, informing the sender about the error can also be configured.

Key Summary Points – Unit 5

Developer Test with Real Deployment and Debugging of your Integration Flow

Before examining the integration flow, it needs to be deployed in the monitoring environment. The graphical model is converted into a Java application and placed in the runtime, allowing the integration flow to be started. If the deployment is successful, the integration flow will either execute immediately if a timer event is used, or it will wait for an incoming message. Cloud integration offers a trace log level that provides insight into the processing of each integration flow component.





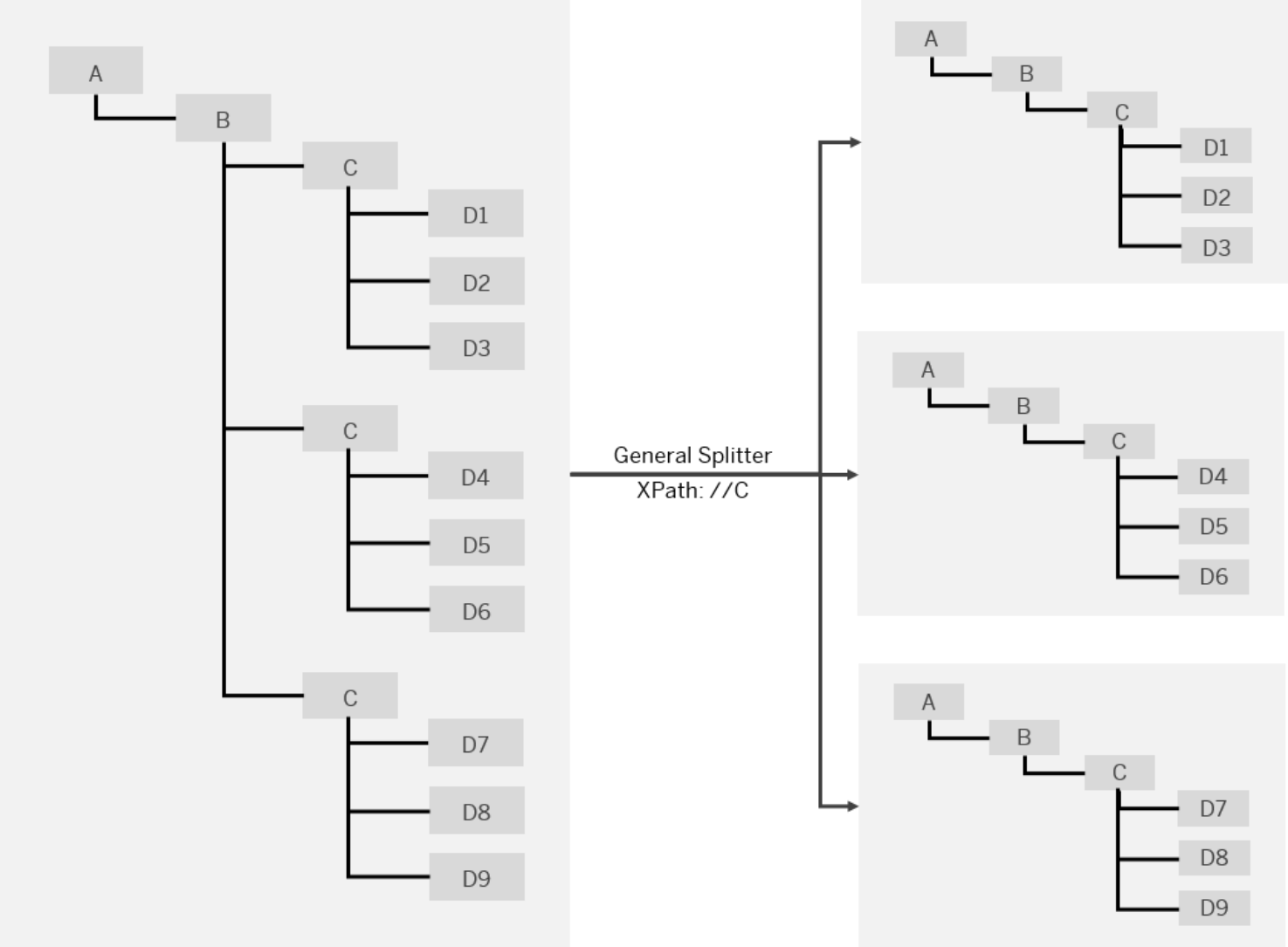
Key Summary Points – Unit 5

Show Integration Patterns

The following integration patterns are included in the example package:

- Aggregator
- Composed Message Processor
- Content-Based Routing
- Content Enricher
- Content Filter
- Message Filter
- Recipient List
- Resequencer
- Scatter-Gather
- Splitter
- Quality of Service Exactly Once

Key Summary Points – Unit 5



Key Summary Points – Unit 5