Final Summary – SAP Integration Suite

Developing with SAP Integration Suite

C_CPI_15

Key Summary Points – Unit 1

- Request driven architecture (Synchronous)
- Event driven architecture (Synchronous + Asynchronous)
  - Pull variant, Push variant
- Combination

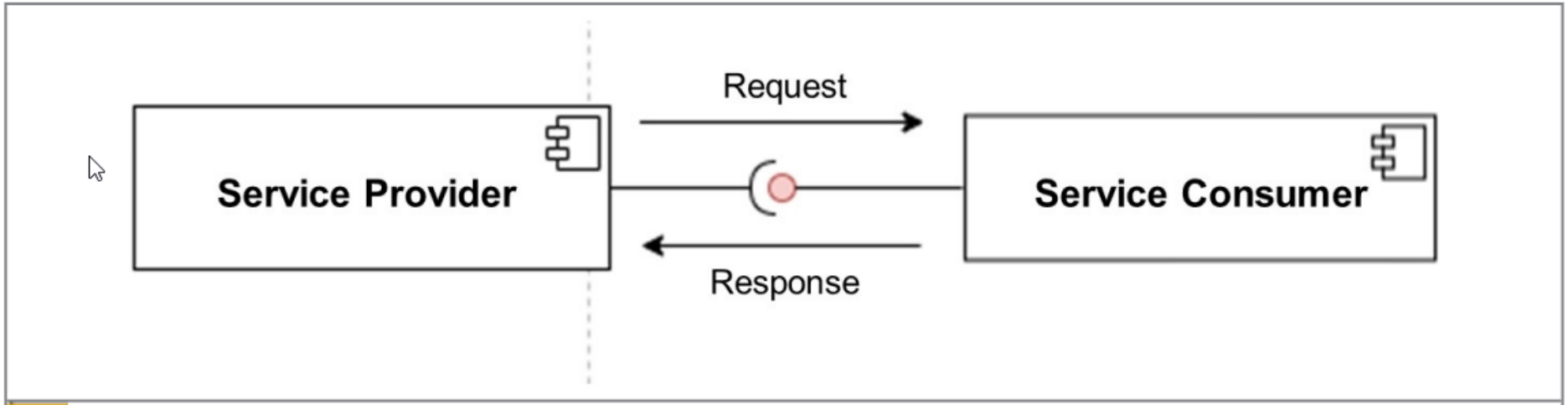


Figure 6: Request Driven Architecture

Key Summary Points – Unit 1

Pull Variant

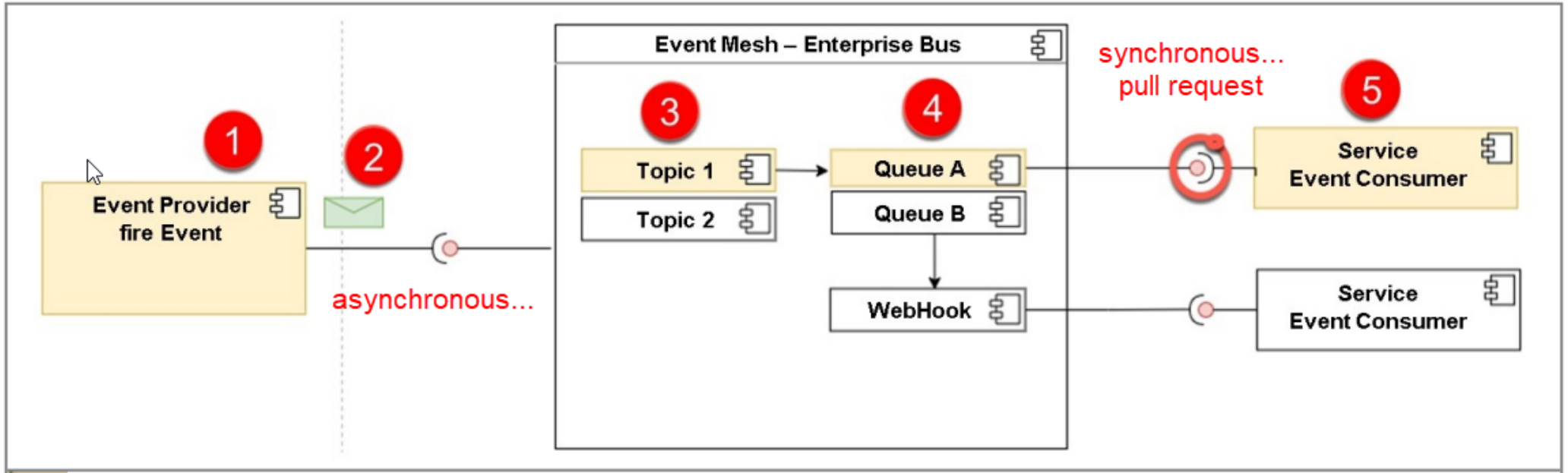


Figure 7: Pull Variant

Key Summary Points – Unit 1

Push Variant

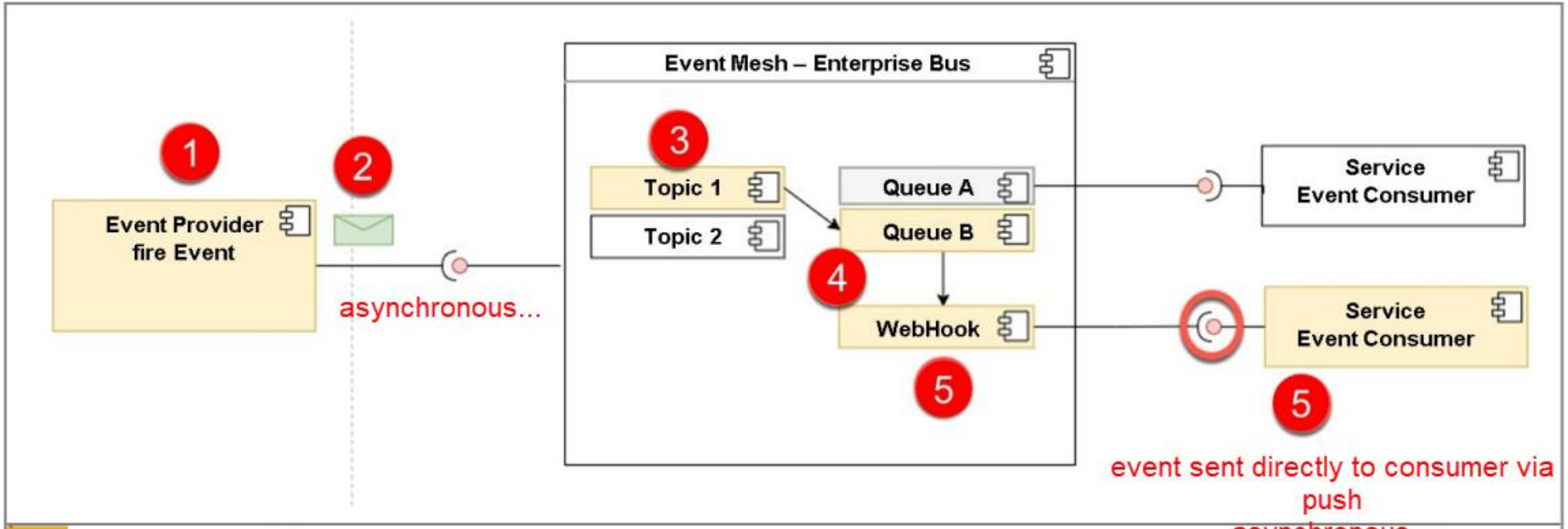


Figure 8: Push Variant

Key Summary Points – Unit 1

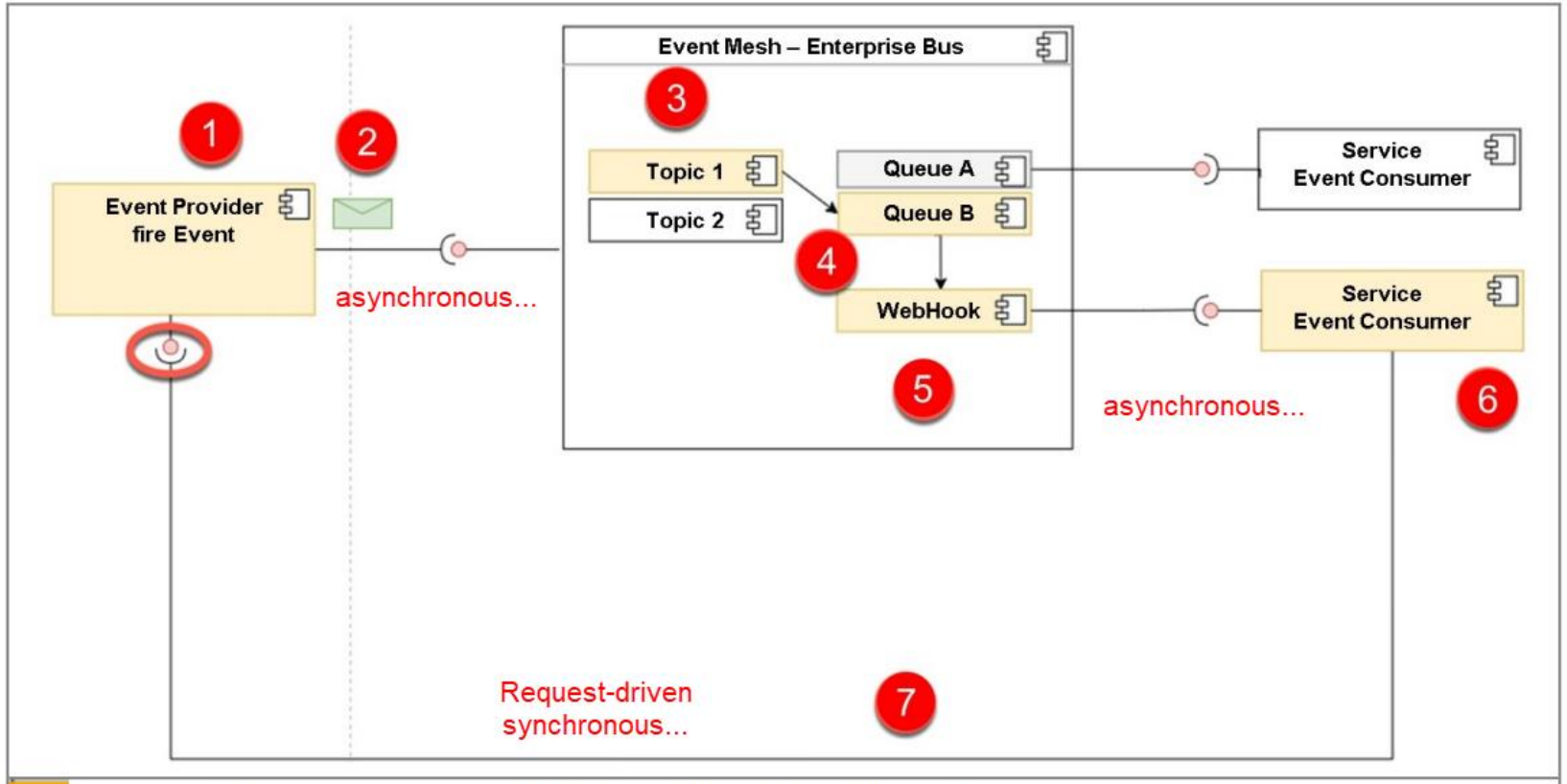


Figure 9: Combination of Request-Driven and Event-Based Architecture

Key Summary Points – Unit 1

Q2. Which are the guiding constraints that defines the REST architectural style?

A High-Availability

✓ Client-Server-Architecture

✓ Cache-Ability

✓ Stateless

✓ Correct

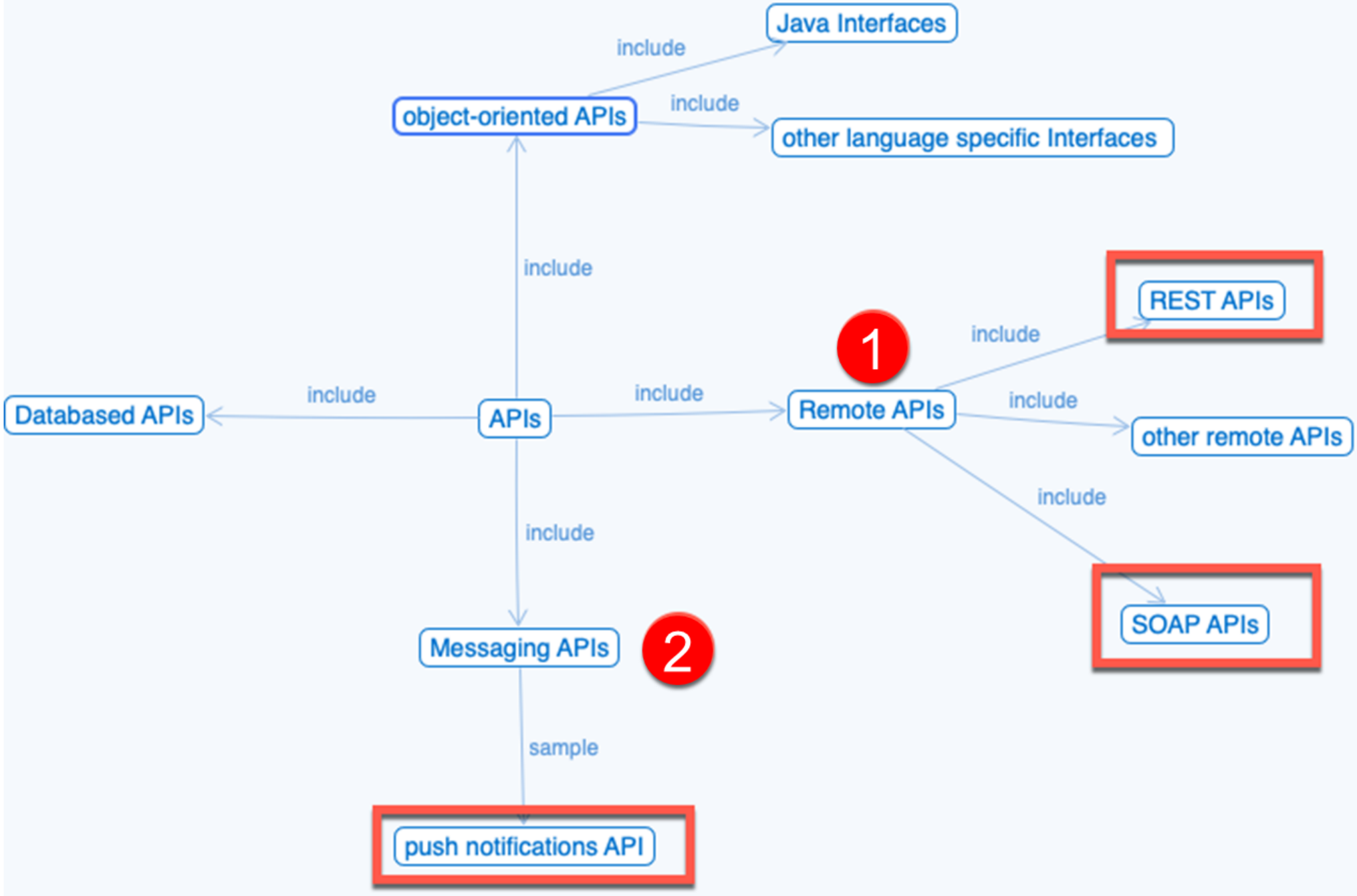
Correct. The guiding constraints that defines the REST architectural style are: Stateless, Client-Server-Architecture, and Cache-Ability.

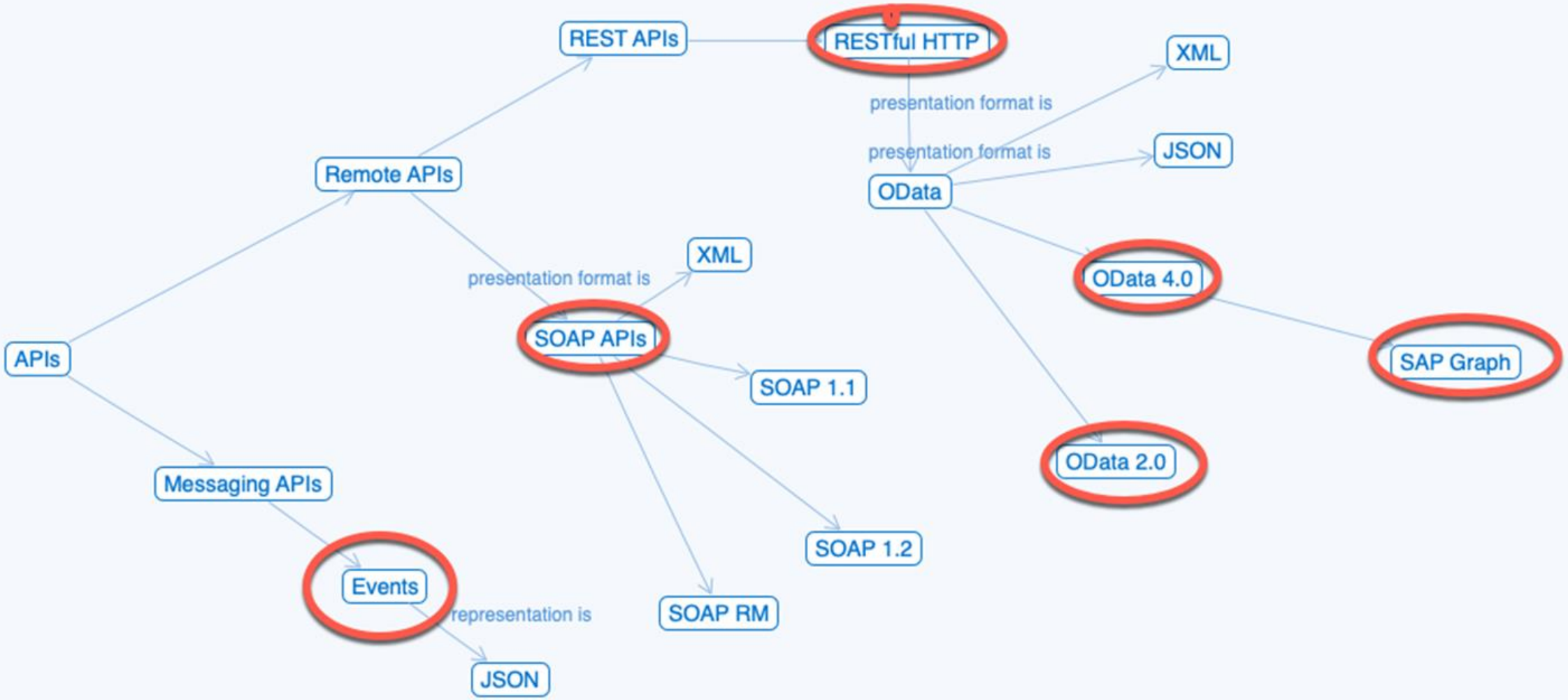
REST

- Architectural properties
  - Simplicity of uniform interface
  - Scalability, modifiability, reliability etc.
- Architectural constraints (6)
  - Client server architecture
  - Stateless
  - Cache ability
  - Layered system
  - Code on demand (optional)
  - Uniform interface
- Uses standard HTTP methods and supports many media types

OData

- Architectural constraints
  - Resource identification
  - Fixed documents
    - Service document
      - Lists entity sets, functions, singletons
    - Metadata document
      - Describes types, sets, functions, actions
  - Dynamic resources
  - Resource operation
  - Querying
  - Resource representation


Java Interfaces
include
object-oriented APIs
include
other language specific Interfaces
include
REST APIs
1
include
Databased APIs
include
APIs
include
Remote APIs
include
other remote APIs
include
include
SOAP APIs
Messaging APIs
2
sample
push notifications API


APIs
Remote APIs
REST APIs
RESTful HTTP
presentation format is
XML
presentation format is
JSON
OData
OData 4.0
SAP Graph
OData 2.0
presentation format is
XML
SOAP APIs
SOAP 1.1
SOAP 1.2
SOAP RM
Messaging APIs
Events
representation is
JSON

Key Summary Points – Unit 1

Q3. Where can you configure the virtual host alias?

A Discover

B Design

✓ Settings ->Integrations ->Configure

D Configure ->Settings ->Integrations

✓ Correct

Correct. You can configure the virtual host alias here: Settings ->Integrations ->Configure.

Key Summary Points – Unit 1

- API Provider
  - Provides an interface
- API Consumer
  - Consumes the interface

SOAP, REST

Type of API	Description language
SOAP	Web Services Description Language (WSDL)
REST	Open API • Used in API management • Interface definition language for describing, producing, consuming and visualizing RESTful web services RAML

Contract between API Provider and API consumer

- Implementation first approach
  - Implementation created first by API Provider
  - Contract generated automatically which is used by API Consumer
- Contract first approach
  - Contract created first
  - Both API Provider and API Consumer can simultaneously start working against the contract

Key Summary Points – Unit 2

SAP Integration Strategy

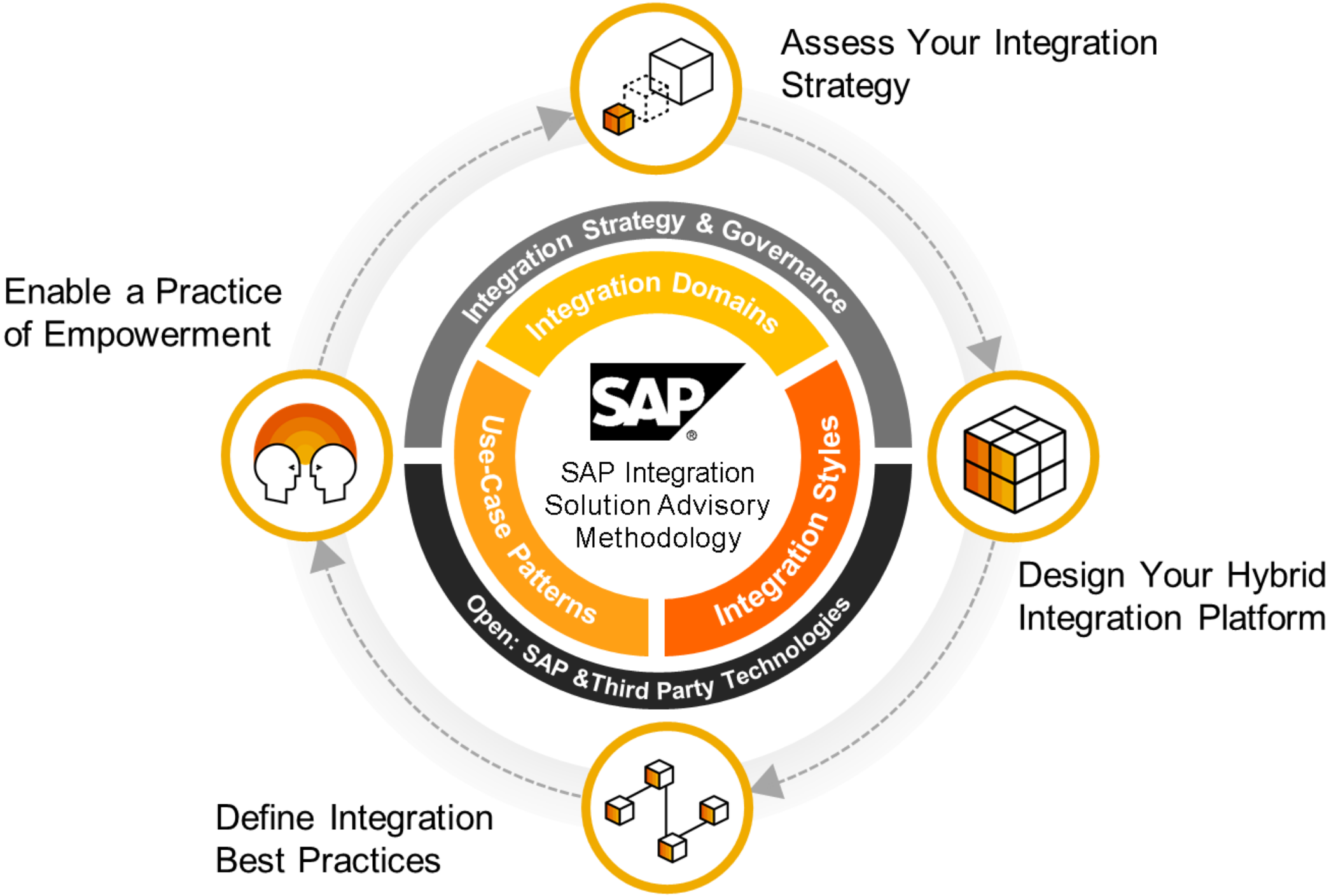
- Predefined integration
  - Prebuilt integrations in SAP Business Accelerator Hub
- Open integration
  - Integration to SAP software, partner software and 3rd party software
  - Open Connectors
- Holistic integration
  - Covers most flavors of cloud and hybrid integration
- AI driven integration
  - AI techniques for integration scenarios
  - Integration Advisor

Secure & compliant
Open & flexible
Unified & simple

Broad ecosystem
Enterprise-grade & scalable
Business-centric

Business Technology Platform

App Dev	Automation	Integration	Data and Analytics	AI
→ Visual Low-Code / No-Code experience	→ Workflow Management	→ Process integration	→ Database	→ Pre-built business AI models
→ Pro-code tooling	→ Robotic Process Automation	→ API led integration	→ Data management	→ MLOps
→ Digital experience	→ Process monitoring & analytics	→ Event based integration	→ Data warehouse	→ Responsible AI
→ DevOps	→ Automated document processing	→ Data integration	→ Analytics & planning	


Assess Your Integration Strategy
Design Your Hybrid Integration Platform
Define Integration Best Practices
Enable a Practice of Empowerment
Integration Strategy & Governance
Integration Domains
Use-Case Patterns
Integration Styles
Open: SAP &Third Party Technologies
SAP
SAP Integration Solution Advisory Methodology

Key Summary Points – Unit 2

The core capacities are as follows:

Cloud-Integration

Seamless integration of everything and everyone (A2A/B2B) in real time.

API-Management

Make your data and processes available as APIs. Manage the E2E lifecycle.

Integration Assessment

Tool support for ISA-M to define and execute an integration strategy for companies.

Integration Advisor

Accelerate the implementation and maintenance of B2B scenarios through machine learning.

Trading Partner Management

Accelerate onboarding and maintenance of B2B integration scenarios with trading partners.

Open Connectors

Accelerate connectivity to non-SAP applications.

Summary

One divides core capabilities, add-on capabilities, and finally add-on capabilities. The core capabilities are implemented in the Integration Suite. The most important are the API management and the cloud integration.

Key Summary Points – Unit 2

The add-on capacities are as follows:

Master Data Integration

Ensure a consistent view of master data within an integrated intelligent suite and its ecosystem.

SAP Data Intelligence

Extract, transform, and load ETL scenarios for data lakes and data warehouses.

Event Mesh

Event-based integrations with predefined events from SAP applications.

Connectivity

Establish secure connectivity between cloud applications and On-Premise systems.

SAP Graph

Unified API for accessing SAP-managed data that can be used to create new extensions and applications using SAP data.

Alert Notification

Provides a common API for providers to publish alerts and for consumers to subscribe to these alerts.

Cloud Transport Management

Management of software products between accounts in different environments by transporting them over different terms.

Internet of Things

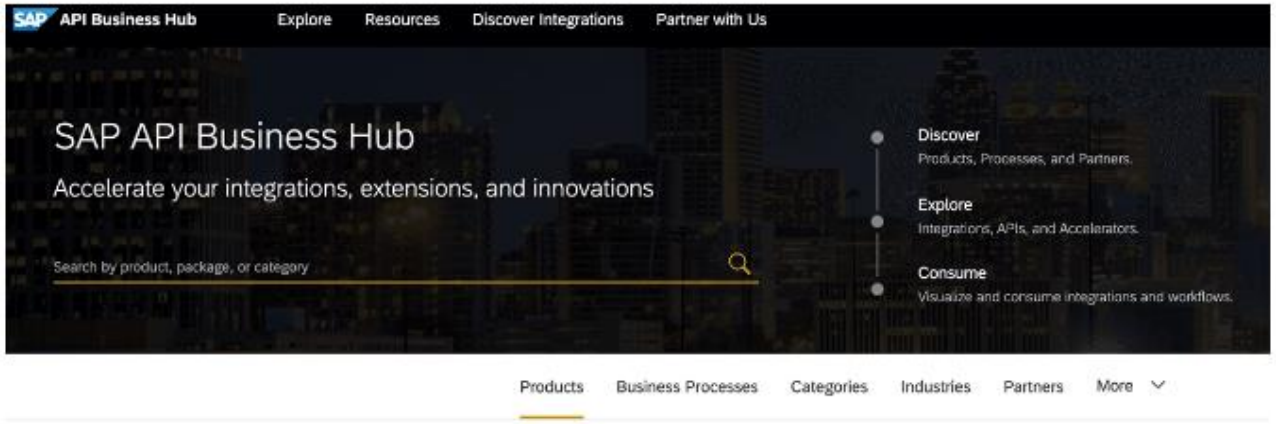
Bring raw sensor data into the context of business objects and use the data in analytical or transactional business applications.

Key Summary Points – Unit 2

On-top capacities are as follows:

SAP Business Accelerator Hub

Jump start for integration projects with APIs, packaged integration content and adapters.



Key Summary Points – Unit 3

Q10. What are the reasons for using policies in API management?

- ✓ Access Control
- B Identity Management
- C Data Management

✓ Correct

Correct. The reason for using policies in API management is Access Control.

Key Summary Points – Unit 3

Q8. Where can you download standardized, reusable policy templates?

A SAP API Business Hub Enterprise

B Enterprise Hub for APIs

✓ SAP Business Accelerator Hub

✓ Correct

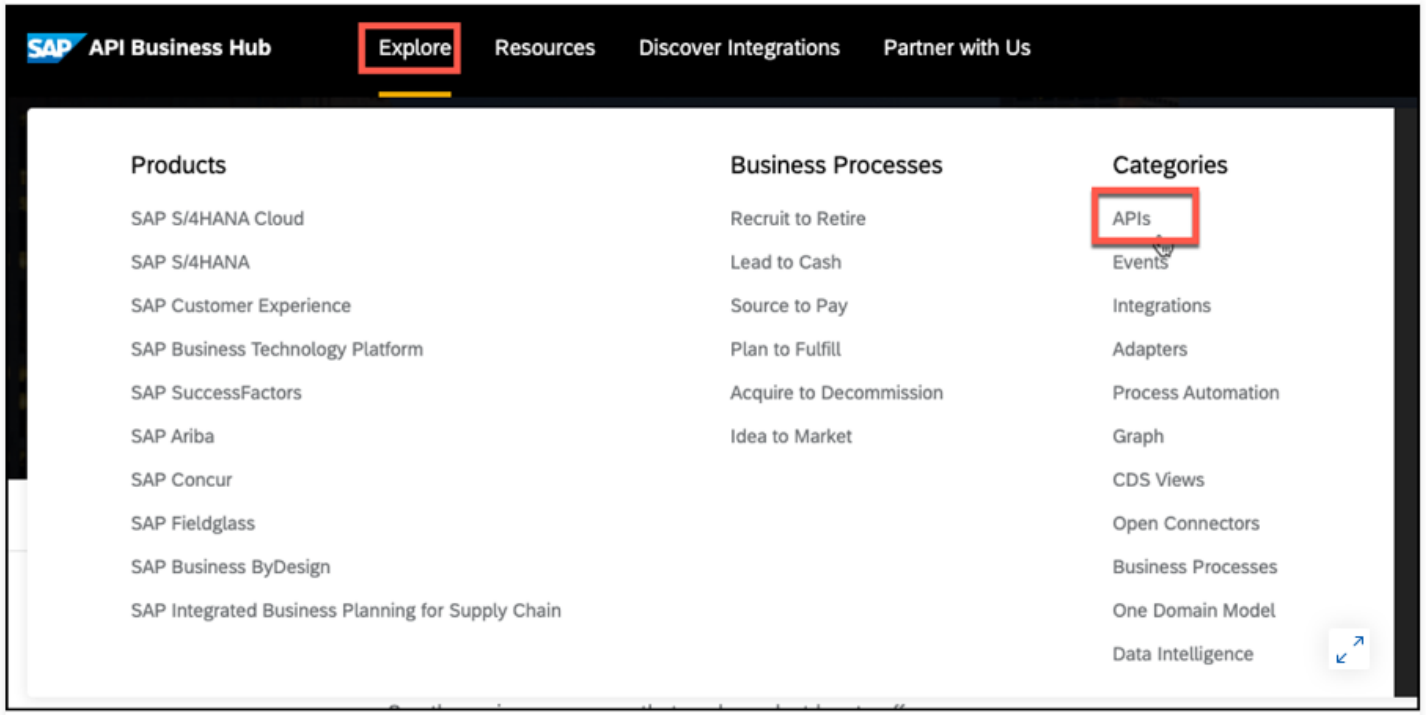
Incorrect. You can you download standardized, reusable policy templates from the SAP Business Accelerator Hub.

Key Summary Points – Unit 3

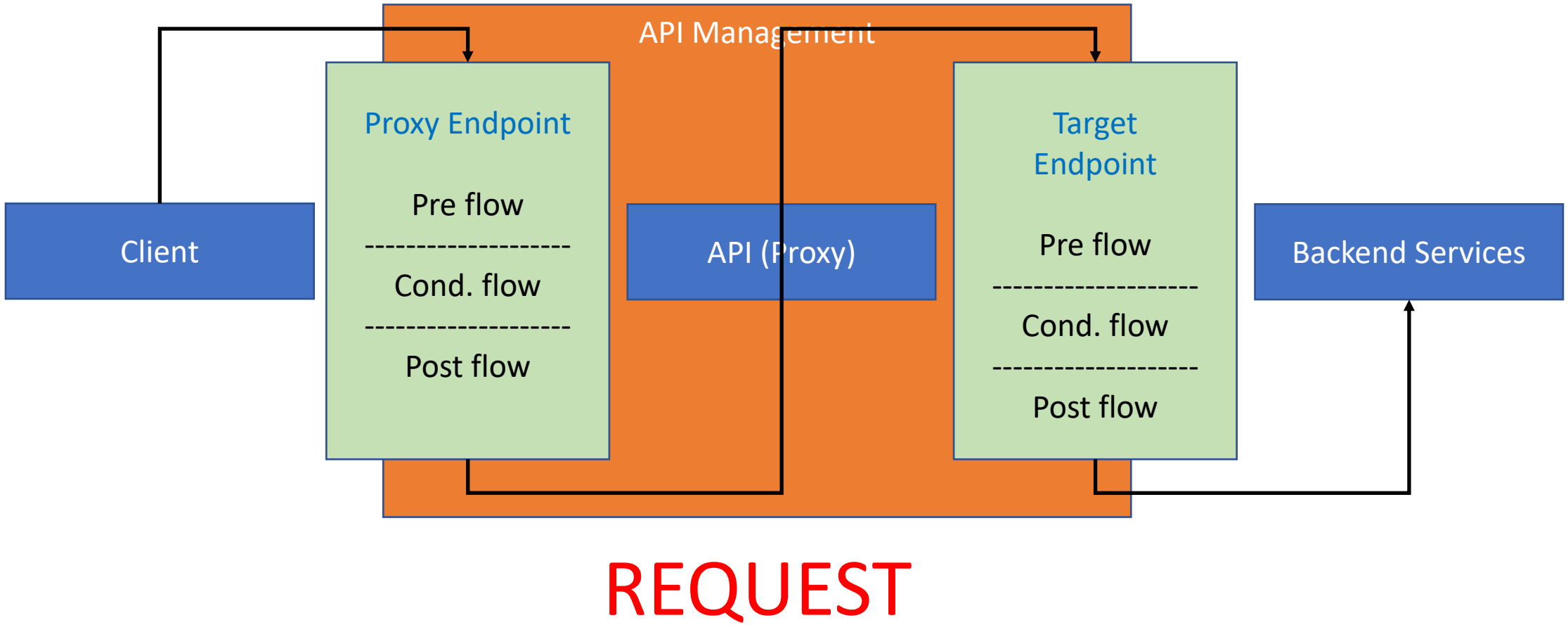
Use predefined policies

There are predefined sets of policies for specific applications. These can be found in the SAP Business Accelerator Hub.

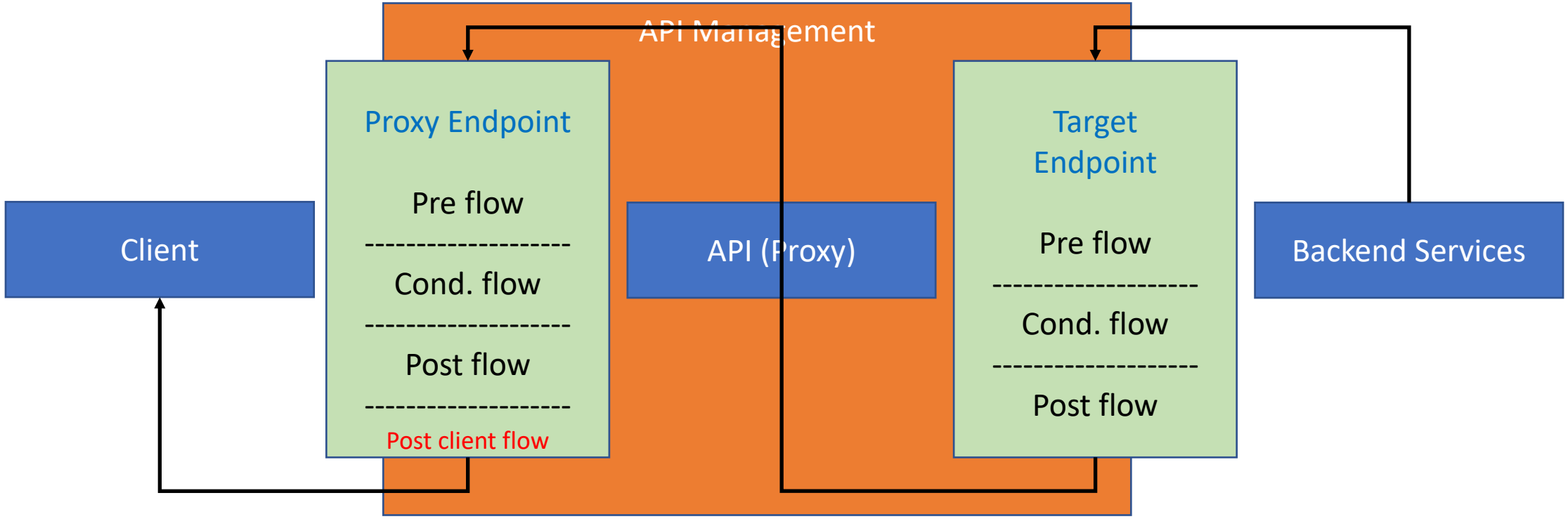
Navigate to https://api.sap.com/ to *Explore* → *APIs*.



Flows – Where should I apply my policies ?



Flows – Where should I apply my policies ?



RESPONSE

Key Summary Points – Unit 3

Q9. Which Role Collections do you need to use the API Business Hub Enterprise?

A AuthGroupAPIADMINDesigner,AuthGroupAdministrator

✓ AuthGroup.API.Admin, AuthGroup.API.ApplicationDeveloper

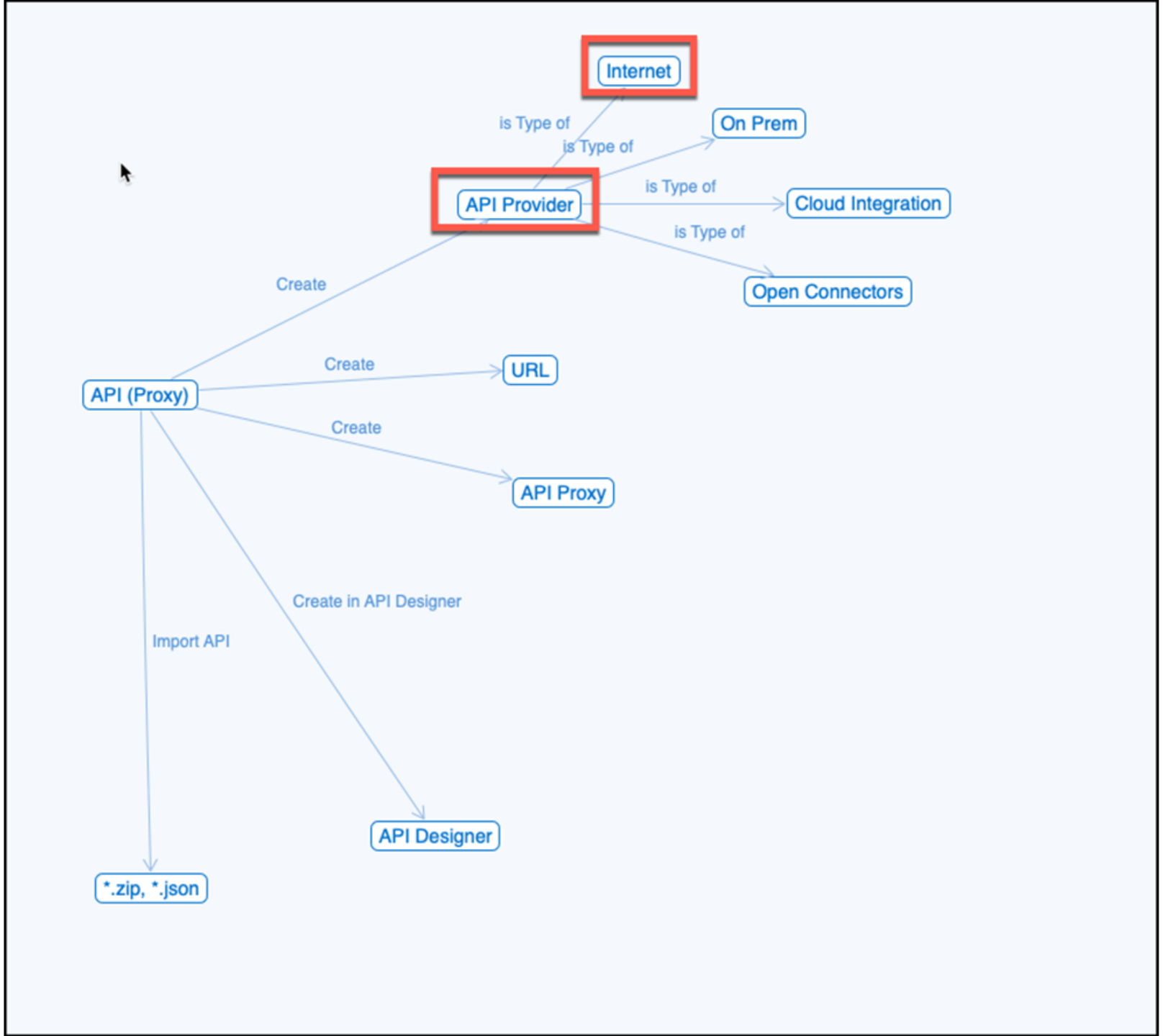
C AuthgroupHeadofManager,AuthgroupChildhoodCaseManager

✓ Correct

Correct. To use the API Business Hub Enterprise you need the Role Collections AuthGroup.API.Admin, and AuthGroup.API.ApplicationDeveloper.

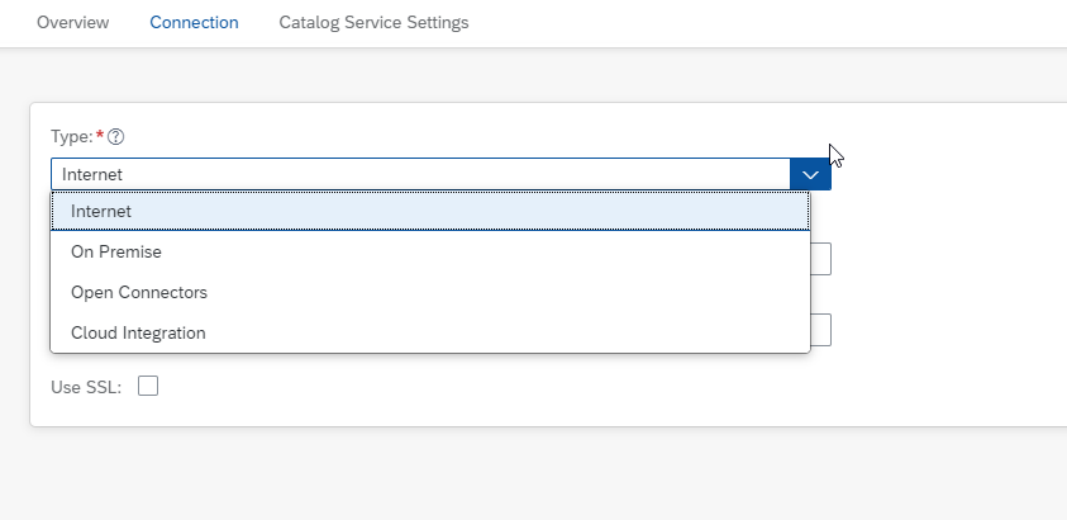
Components of SAP API Management

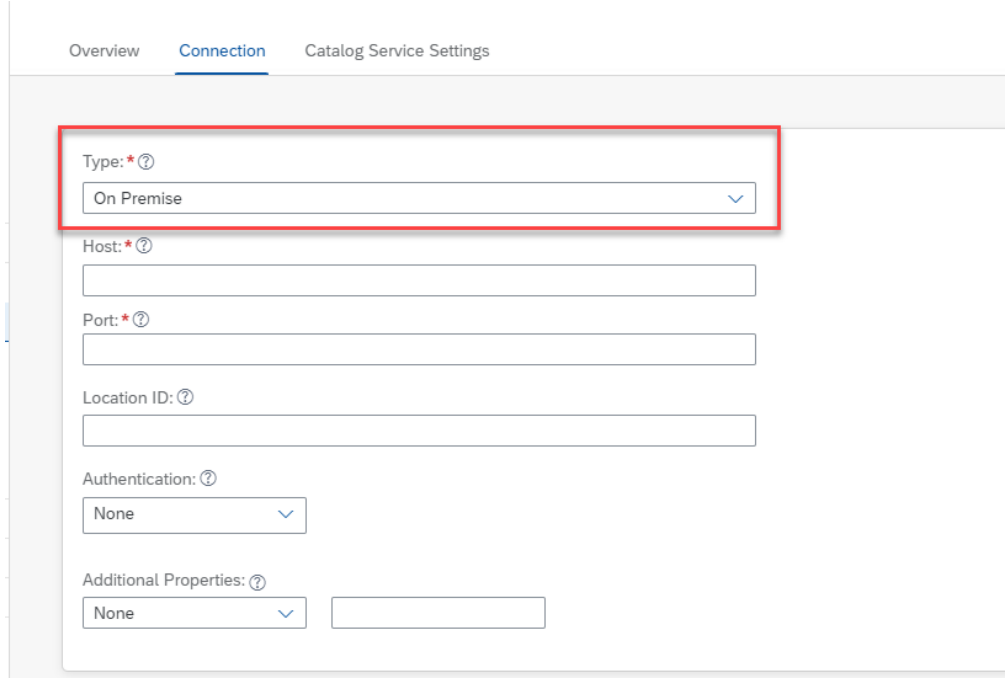
- API Provider
  - Concept in API Management that defines connection details for existing services
- API (Proxy)
  - Managed facades for existing services (sits in front of the existing service)
  - Applications connect to API (proxy)
- Policies
  - Provides capabilities to define behavior of an API (proxy)
- Product
  - Bundle and publish API (proxies) as a Product for consumption
- Application
  - Consumes the Product (bundle of API proxies) using api key and secret


Internet
is Type of
On Prem
is Type of
API Provider
is Type of
Cloud Integration
is Type of
Open Connectors
Create
Create
URL
API (Proxy)
Create
API Proxy
Create in API Designer
Import API
API Designer
*.zip, *.json

Demo: API Provider (5 different sources)

- Open Connectors
- Through Cloud Connector to SAP On-Premise backends
- Cloud Integration
- APIs from internet
- SAP Business Accelerator Hub (API Business Hub)





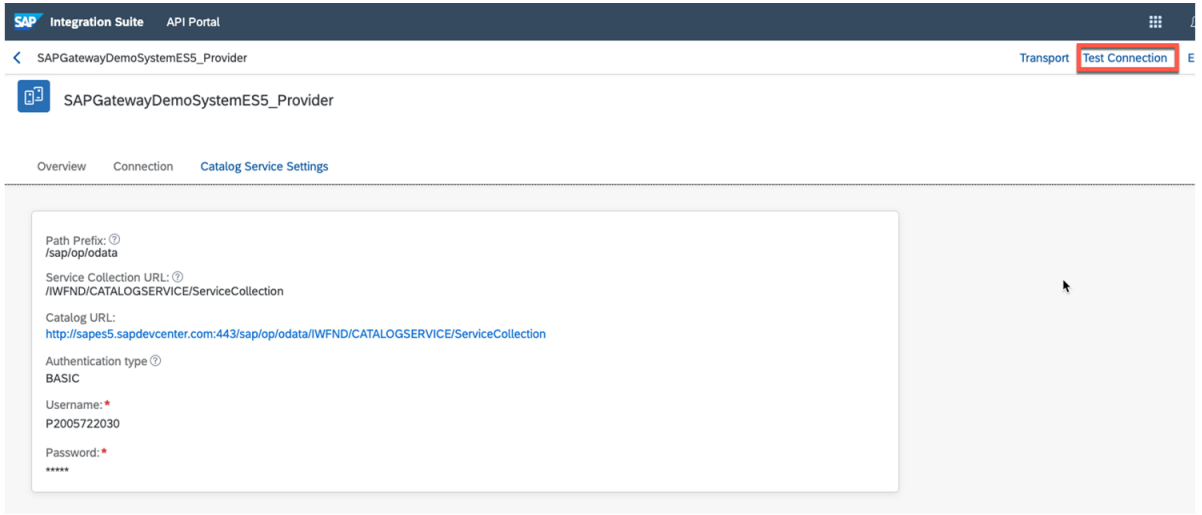
Demo: API Provider

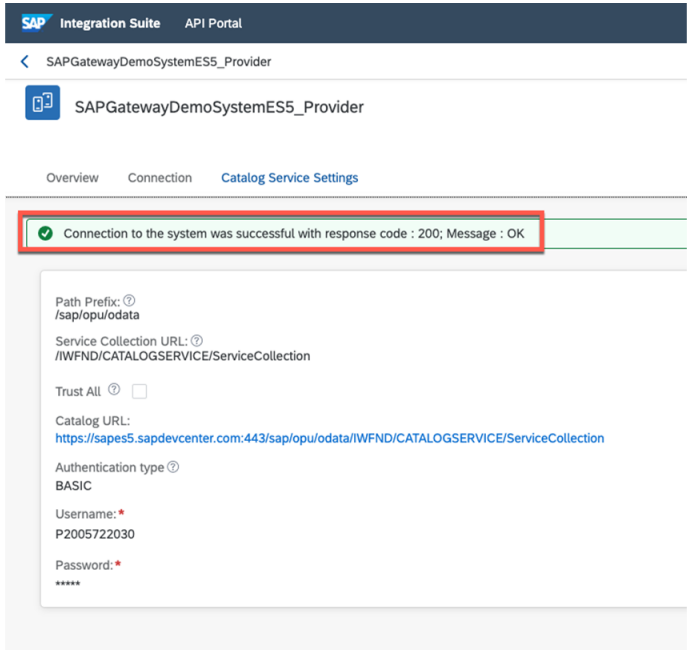
Field Name	Input
Type	Internet
Host	sapes5.sapdevcenter.com
Port	443
Use SSL	Checked
Path Prefix	/sap/opu/odata
Service Collection URL	/IWFND/CATALOGSERVICE/ServiceCollection
Authentication Type	Basic
Username	*Credentials only used to create API Provider*
Password	*Not for the actual call of API*

Key Summary Points – Unit 3

Summary

An API provider encapsulates access to APIs from various sources. More than 260 third-party REST-based APIs are connected through the Open Connector. SAP backend systems such as SAP S/4HANA On-Prem or ECC/PI/PO can be connected through the Cloud Connector. SOAP APIs can also be made available through the Cloud Integration. Ultimately, almost all APIs can be connected. The procedure for connecting a foreign API is wizard-controlled.





Key Summary Points – Unit 3

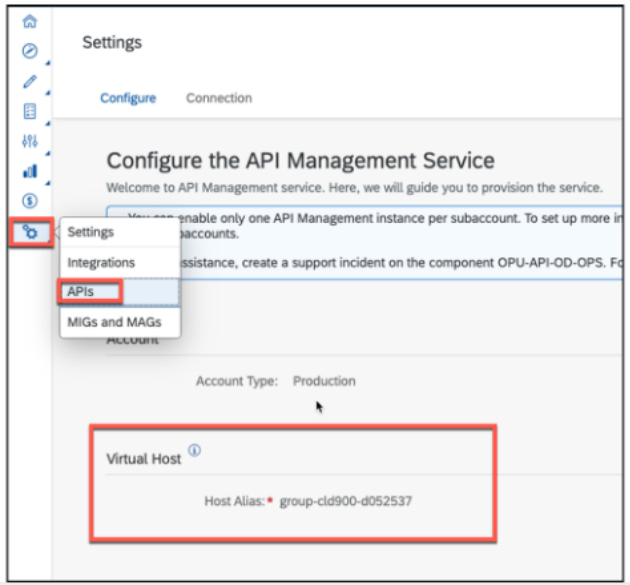
API URL - Proxy URL (No. 1)

At No. 1, you can see the new URL (proxy URL) with which you can now call the original source API. The *URL* consists of the following elements:

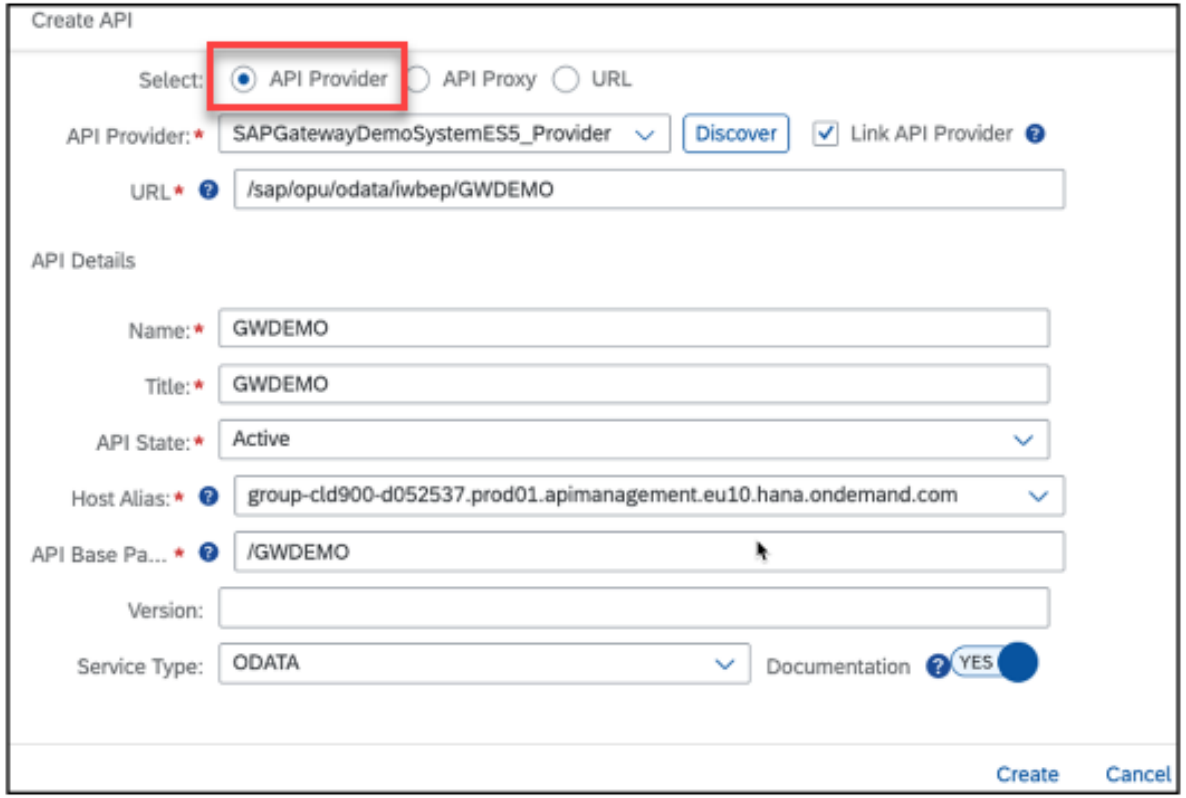
- API URL: https://group-cld900-d052537.prod01.apimanagement.eu10.hana.ondemand.com:443/GWSAMPLE_BASIC
- Application protocol: https
- Virtual Host: group-cld900-d052537
- API Host: prod01.apimanagement.eu10.hana.ondemand.com
- API Port: 443
- API Name: GWSAMPLE_BASIC

Virtual Host

The virtual host was created during the provisioning of API management and can be changed at any time using *Settings → APIs*.



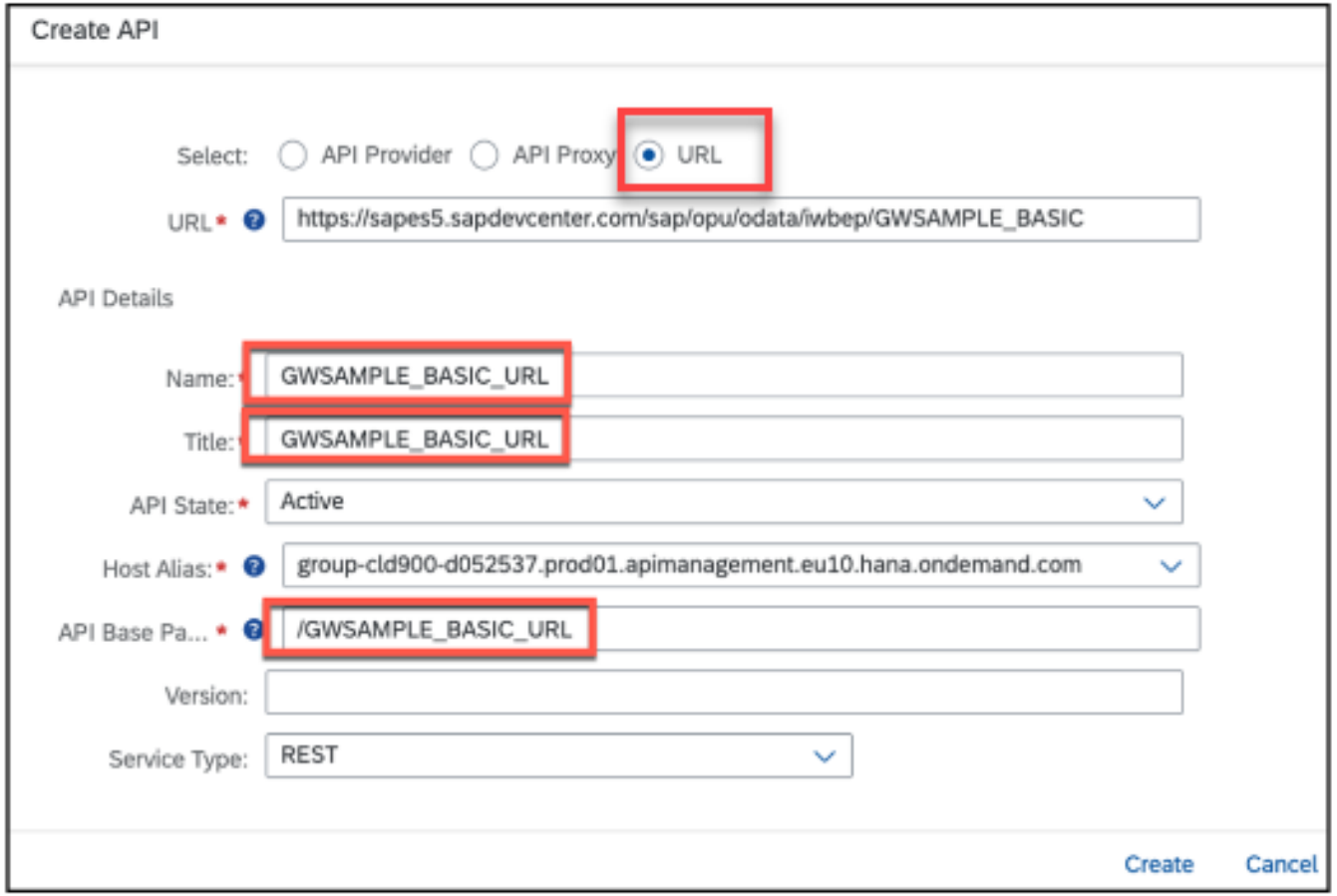
Key Summary Points – Unit 3



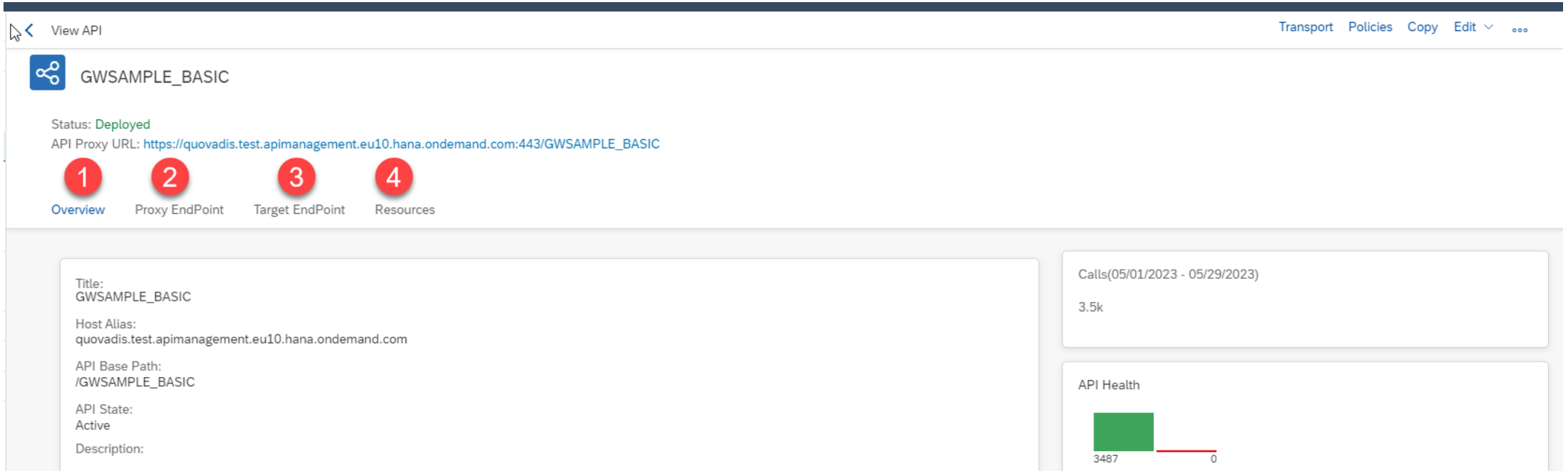
When you finish creating this API (Proxy), it has to be deployed so that it can be used. After that, the API (proxy) is ready for testing. The *service type* is automatically defined. In this case, it is OData.

Key Summary Points – Unit 3

In this case, you must enter the data manually (marked). The *Service Type* can only be *REST* or *SOAP*.



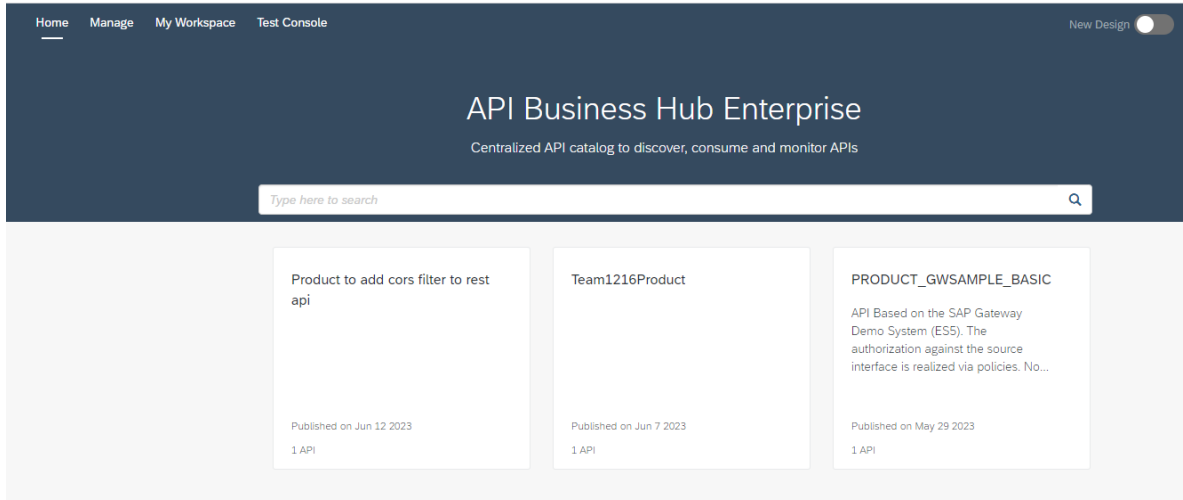
Editing APIs



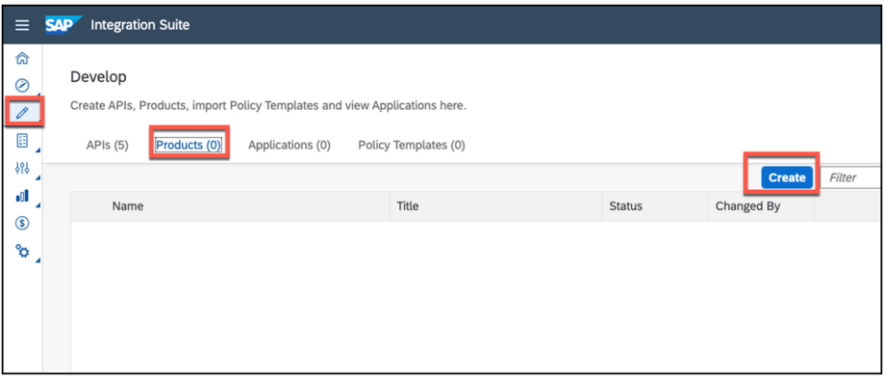
Creating a Product

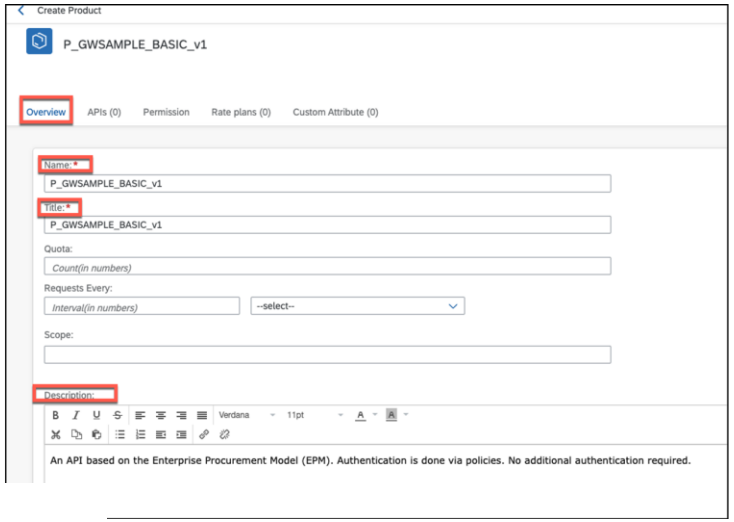
- Products are artifacts that appear in the SAP API Business Hub Enterprise Portal
- Bundle and publish one or more API (proxies) as a Product for consumption

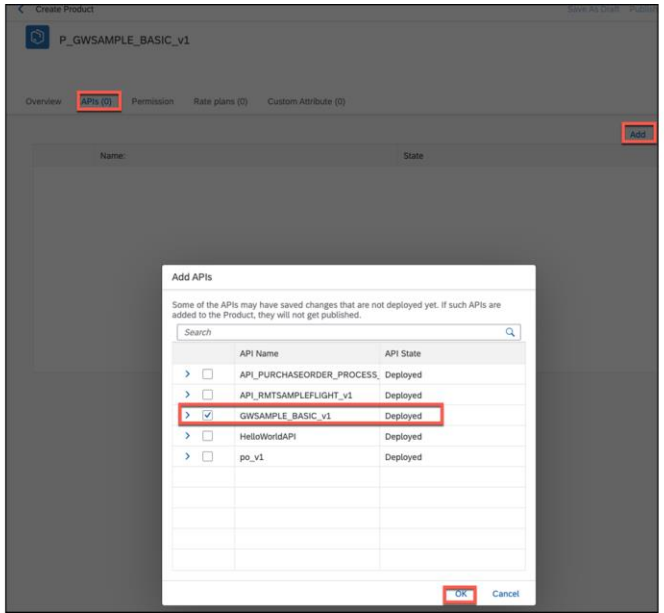
- Role Collections
  - AuthGroup.API.Admin
  - AuthGroup.API.ApplicationDeveloper



Key Summary Points – Unit 3

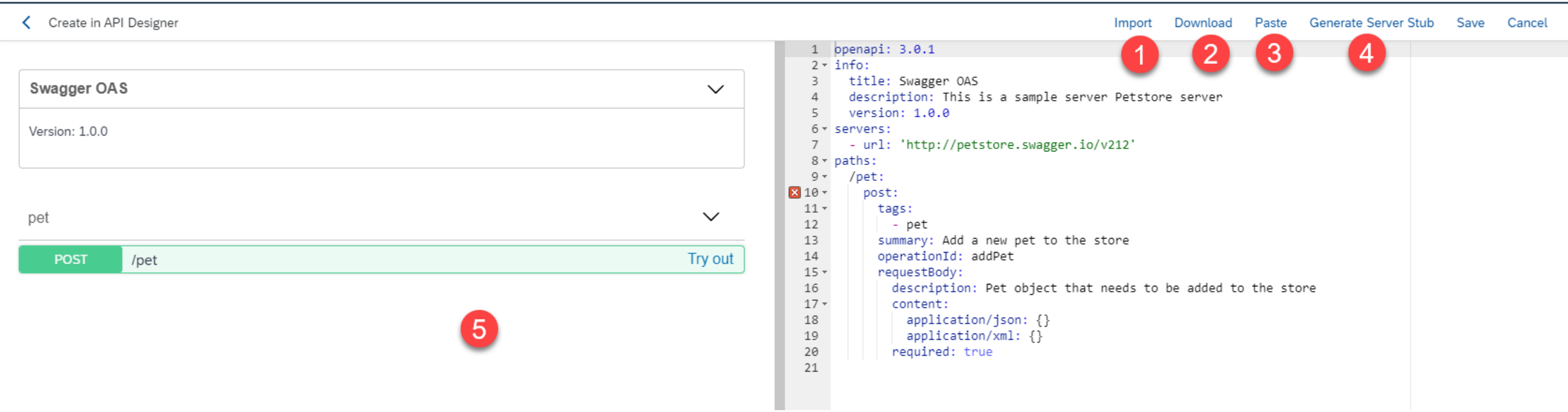






Key Summary Points – Unit 3

- API Designer
  - Visualization of openAPI specification is done using swagger UI
  - Swagger UI is an open source JavaScript framework to make APIs tangible



Key Summary Points – Unit 4

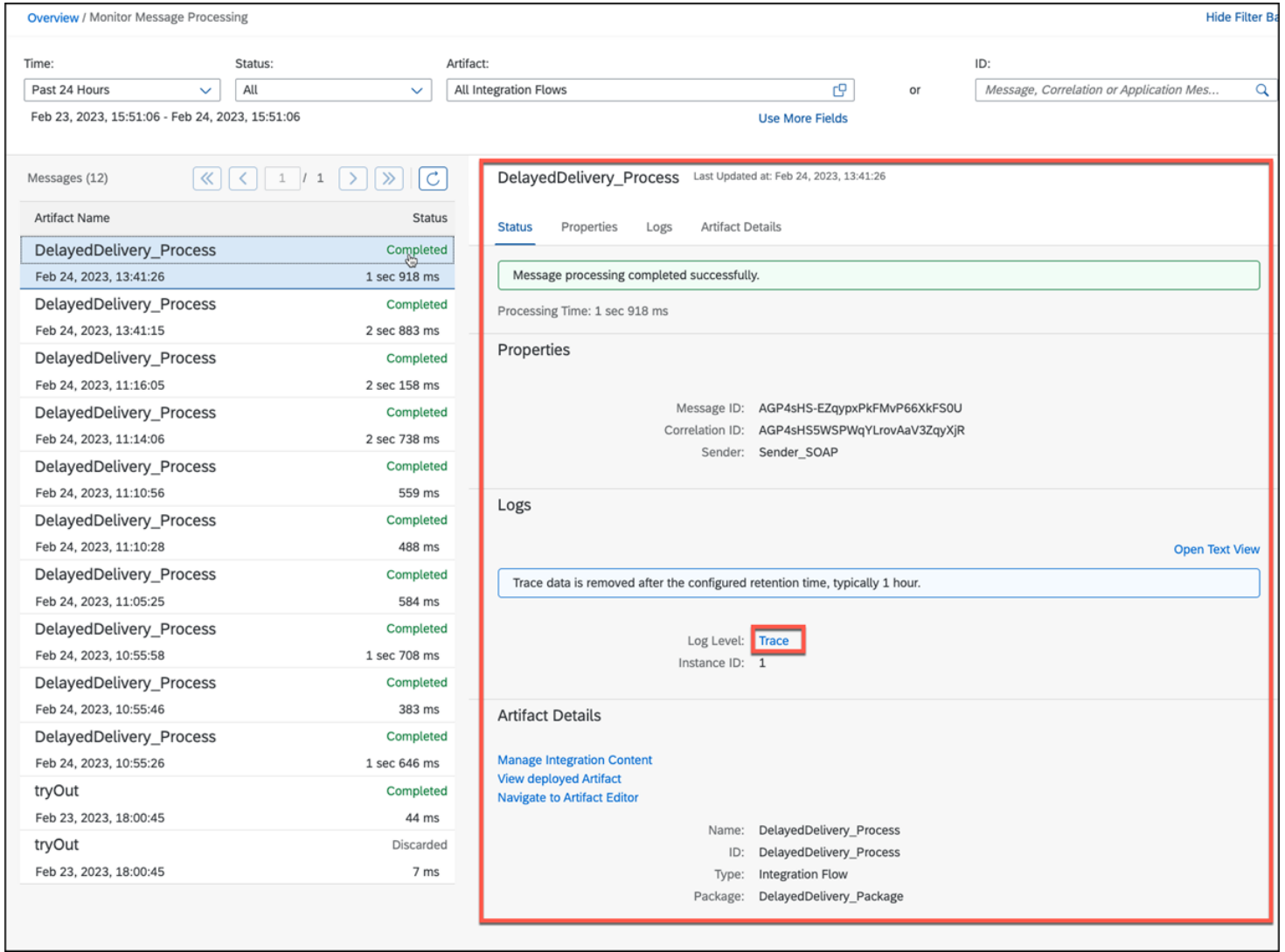
Q2. What needs to be enabled to work in debugging mode within the monitor?

- ✓ The log level must be set on trace.
- B The log level must be set on info.
- C The log level must be set on hold.

✓ Correct

Correct. The log level must be set on trace.

Key Summary Points – Unit 4



Key Summary Points – Unit 4

Q3. Where can you discover pre-defined integration content?

A My preferred SAP Consultant dealerstore.

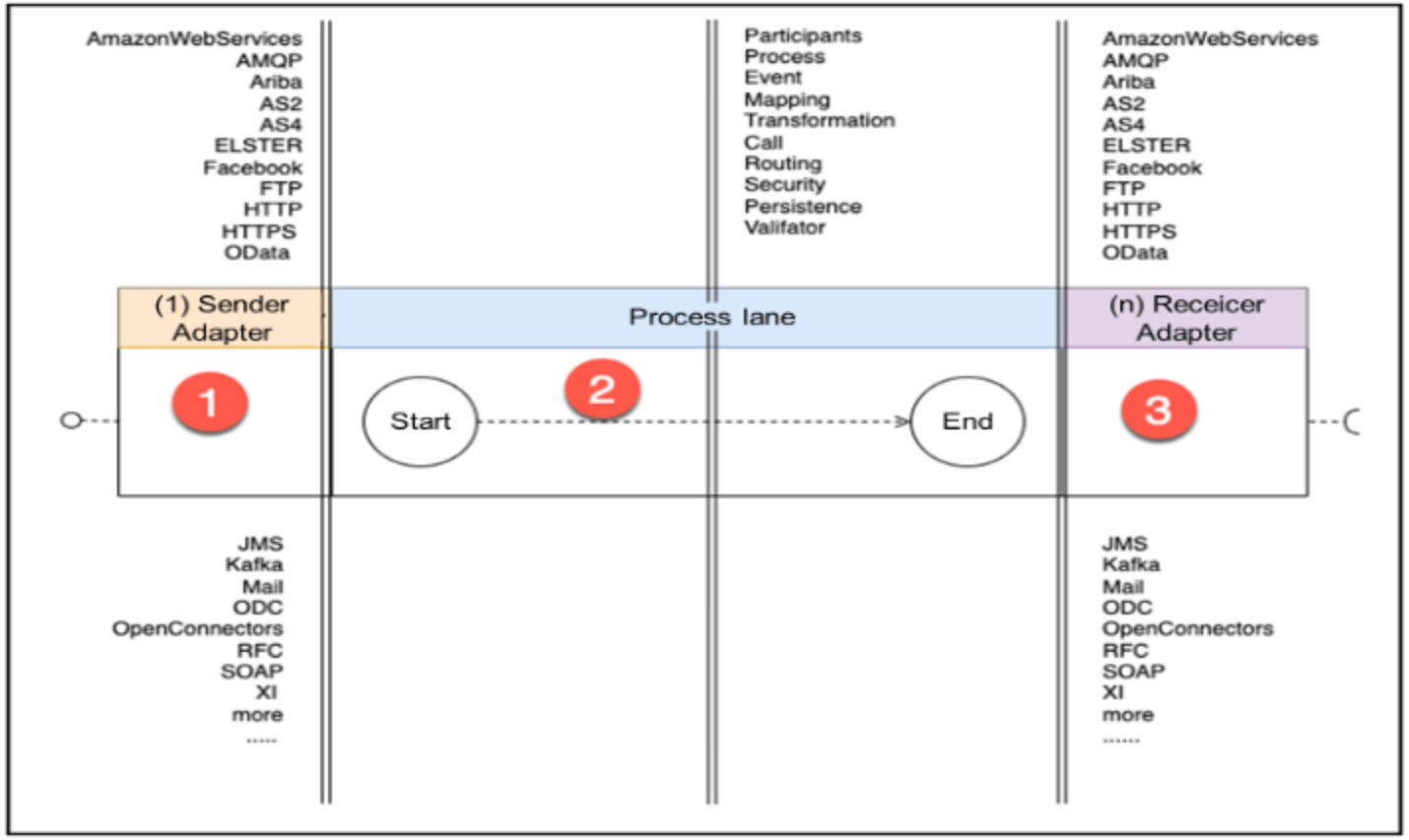
SAP Business Accelerator Hub - New name

API Business Hub or Discovery tab in the Integration Suite.

C API Business Hub Enterprise or Design Tab into the Integration Suite.

Correct

Correct. You can discover pre-defined integration content on the API Business Hub or Discovery tab in the Integration Suite.



Key Features of Cloud Integration

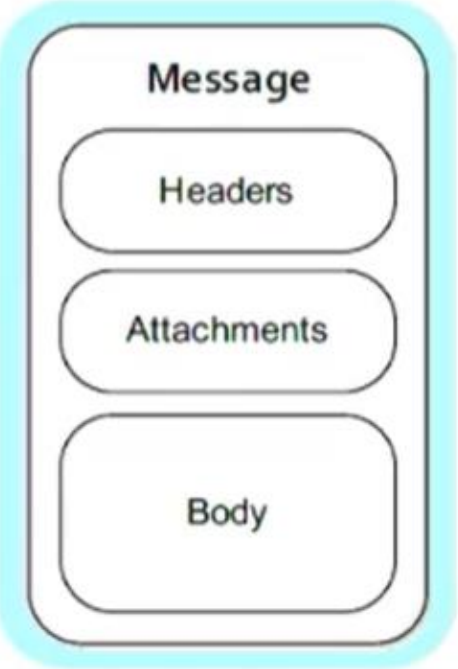
- Integration flow has 0-1 sender adapter
- Message is delivered via an endpoint
  - If an adapter is configured
- Process is started via Start event
- Different ways messages can be processed
- Receiver adapters can be configured
- Message processing can be synchronous or asynchronous

Basic concepts of Cloud Integration flow…

Message

Fundamental entity **containing the data** being carried and routed in Camel

- Messages have a body (a payload), headers, and optional attachments
- Messages are uniquely identified with an identifier of type `java.lang.String`
- *Headers*
  - Headers are values associated with the message
    - ⇨ Sender identifier, hints about content encoding, authentication information,…
  - Headers are name-value-pairs
    - ⇨ Name is a unique, case-insensitive string
    - ⇨ Value is of type `java.lang.Object`
- *Attachments*
  - Optional – typically used for Web service and e-mail components
- *Body*
  - Type: `java.lang.Object` → any kind of content is allowed

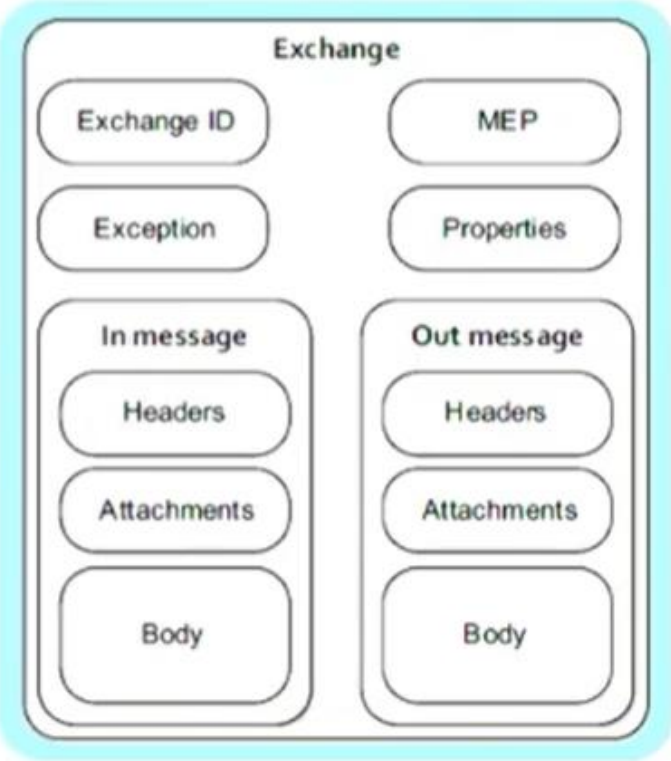


Basic concepts of Cloud Integration flow…

Exchange

The **message's container** during routing

- Provides support for various interaction types between systems, known as Message Exchange Patterns (MEP)
  - `InOnly`: a one-way message (e.g. JMS messaging)
  - `InOut`: a request-response message (e.g. HTTP-based transports)
- *Exchange ID*: a unique ID that identifies the exchange
- *MEP*
  - `InOnly`: exchange contains an "in message" **only**
  - `InOut`: exchange contains an "in message" **and** an "out message" containing the reply message for the caller
- *Exception*: If an error occurs during runtime, the Exception field will be filled
- *Properties*: Similar to message headers, but they last for the duration of the entire exchange; they contain global-level information; you can store and retrieve properties at any point during the lifetime of an exchange



Key Summary Points – Unit 4

Summary

The process of creating an integration flow involves using a graphical editor in the remote cloud integration application. Simulations can be conducted on individual parts or the entire integration flow to verify that values are correctly set in content modifiers, scripts or mappings. Once the integration flow is complete, it is versioned and deployed, resulting in the creation and deployment of a Java application in a runtime. The integration flow can then be executed. The development process can be approached as cycles, where the placement and configuration of components, debugging using trace log levels, and testing are repeated until the desired result is achieved.

Manipulating Exchange Parameters

- Exchange params (including payload): set by incoming messages
- But these params can also be manually manipulated
  - Content Modifier component
  - Groovy SDK
  - JavaScript SDK
  - PDF in Message Mapping
  - XSLT Mapping
  - And more…

Simple Expression Language

- Used to parameterize Exchange Parameters
- General scheme is ${} placeholder containing built-in variable or Exchange parameter
- For example
  - ${in.body}
  - ${property.someproperty}
  - ${header.someheader}

Key Summary Points – Unit 4

Set Exchange Parameters with Groovy SDK

The *com.sap.gateway.ip.core.customdev.util.Message* class offers methods to manipulate the parameters.

```
import com.sap.gateway.ip.core.customdev.util.Message;
    import java.util.HashMap;

    def Message processData(Message message) {
        println "You can print and see the result in the console!
        //Body
        def body = message.getBody(String);
        message.setBody(body + "Body is modified");
        //Headers
        def map = message.getHeaders();
        def value = map.get("oldHeader");
        println "oldHeader value: " +value
        message.setHeader("oldHeader", value + "modified");
        message.setHeader("newHeader", "newHeader");
        //Properties
        map = message.getProperties();
        value = map.get("oldProperty");
        message.setProperty("oldProperty", value + "modified");
        message.setProperty("newProperty", "newProperty");
        return message;
        }

```

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?

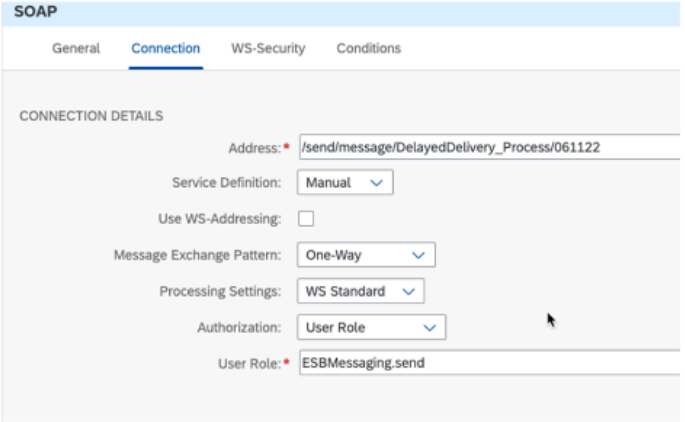
- ✓ ESBMessaging.send
- B Send.To.Endpoint
- C ESB.Messaging.Send
- D HTTP.ESBMessaging.Send

✓ 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

Salesorders.xbook | Products.xml | Products.xbook | SalesOrders

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

```
/ProductSet
```

[44] ✓ 0.2s Source: 'Products.xml' Items: 1 XPath

··· "-/ProductSet[1]"

```
/ProductSet/count(Product)
```

[45] ✓ 0.2s Source: 'Products.xml' Items: 1 XPath

··· 1

```
/ProductSet/count(Product) > 0
```

[46] ✓ 0.2s Source: 'Products.xml' Items: 1 XPath

··· true

Products.xml

```
<ProductSet>
    <Product>
        <Category>Flat Screen Monitors</Category>
        <ProductID>HT-1035</ProductID>
        <Name>Flat Basic</Name>
    </Product>
</ProductSet>
```

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

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 (1)
  - 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

Overview | POST SOAP Inbound request | DEV

SAP Gateway Demo System / **SOAP Inbound request**

SOAP Request...

Save

POST {{baseURL}}cxf/send/message (2)

Params | Authorization | Headers (9) | Body | Pre-request Script | Tests | Settings

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL | XML

(3)

```
<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

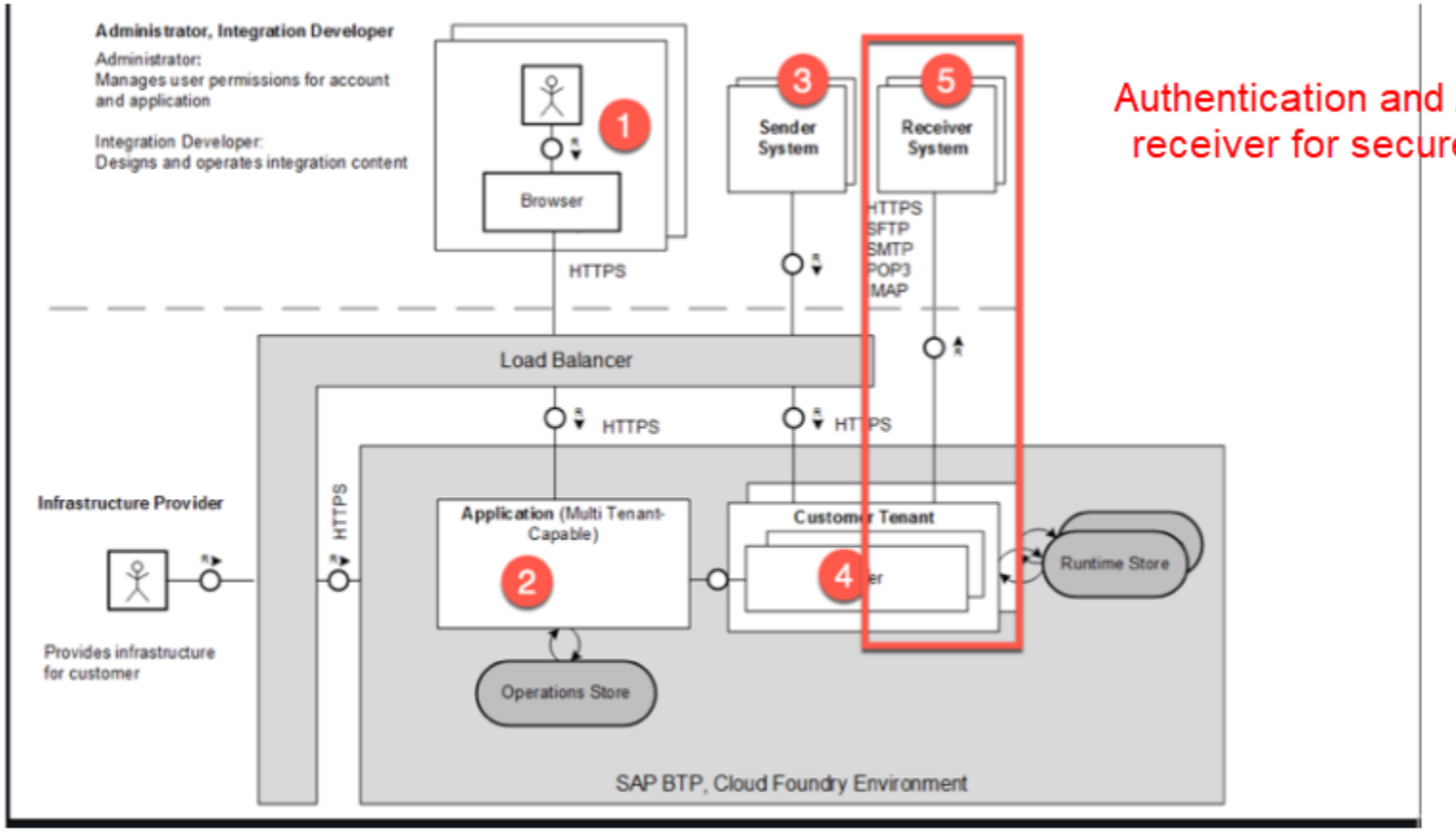
Status: 202 Accepted | Time: 1866 ms | Size: 462 B

(4)

Pretty | Raw | Preview | Visualize | Text

1

Using Adapter Outbound Security



Authentication and Authorization with receiver for secure communication

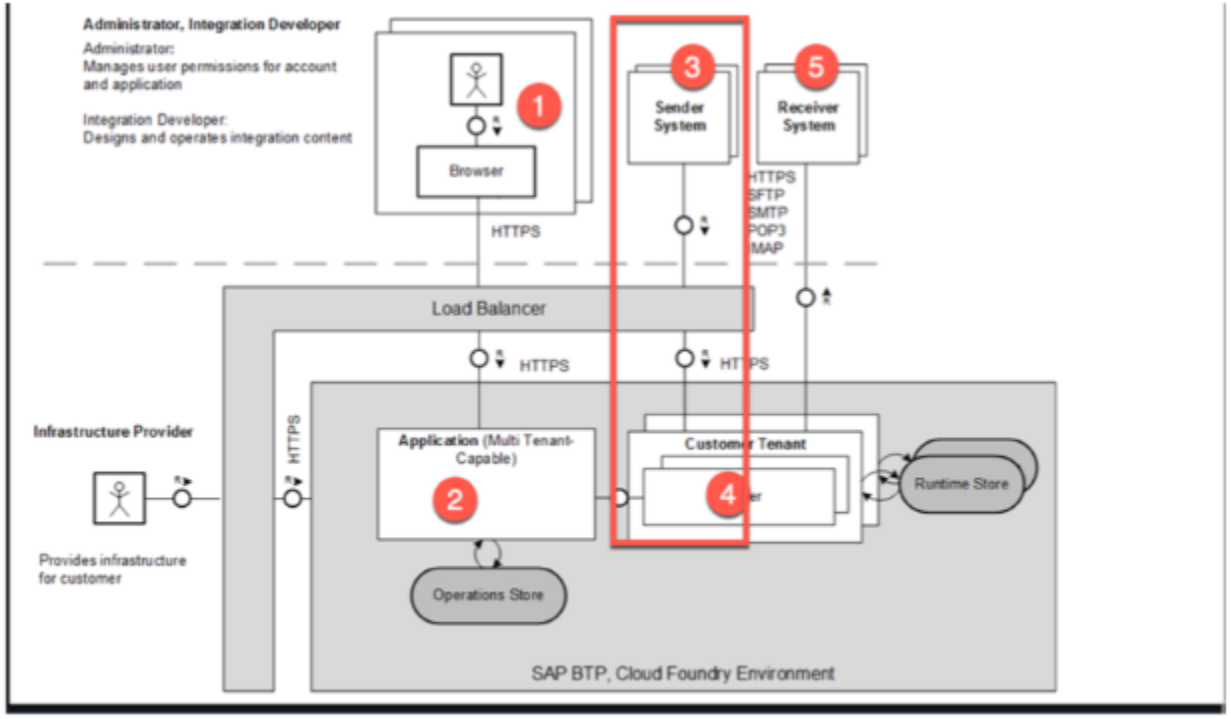
Example of a Direct Receiver and Sender Adapter

Options for Authentication / Authorization

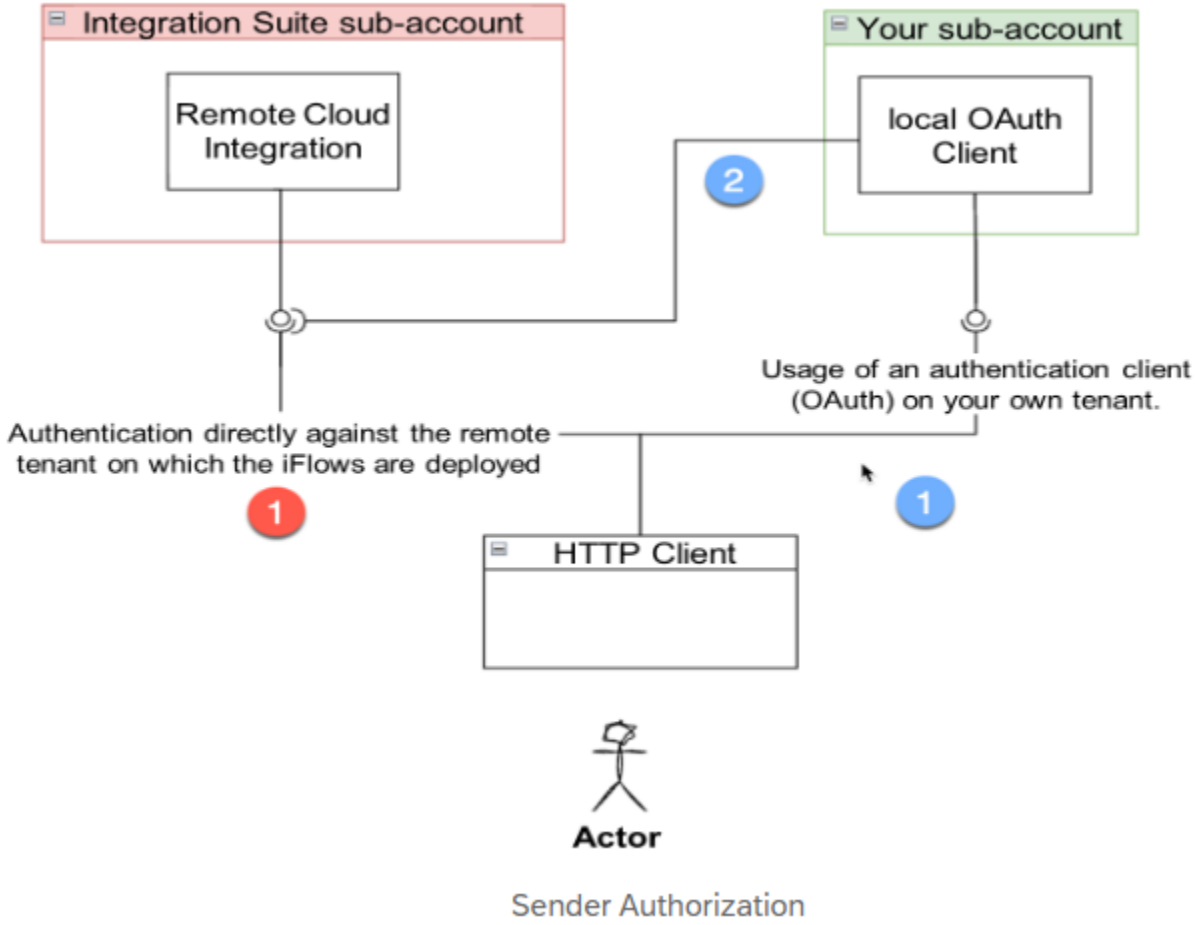
- Basic
- Client Certificate
- None
- OAuth2 Client Credentials
- OAuth2 SAML Bearer Assertion

Using Adapter Inbound Security

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



Authorization of sender



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

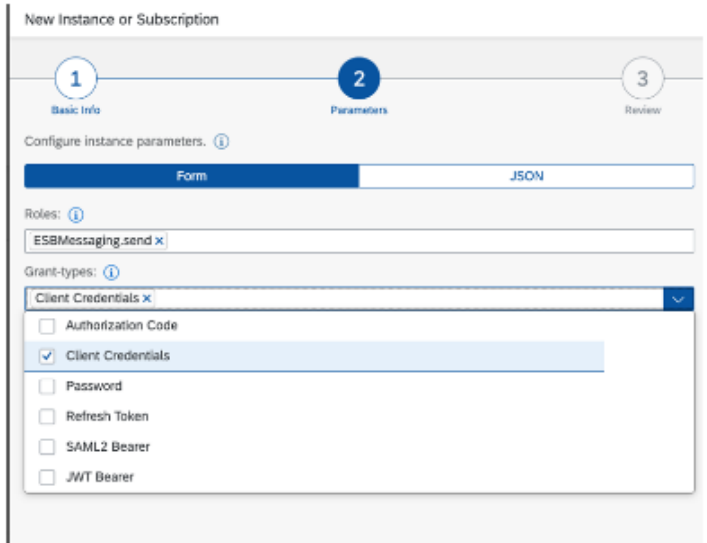
The method of directly calling an integration flow via the role-based approach shown uses personalized users and basic authentication, which are not suitable for productive purposes. For better authentication methods, we need to use a self-configured OAuth2.0 client that can be created on our own subaccount.

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.

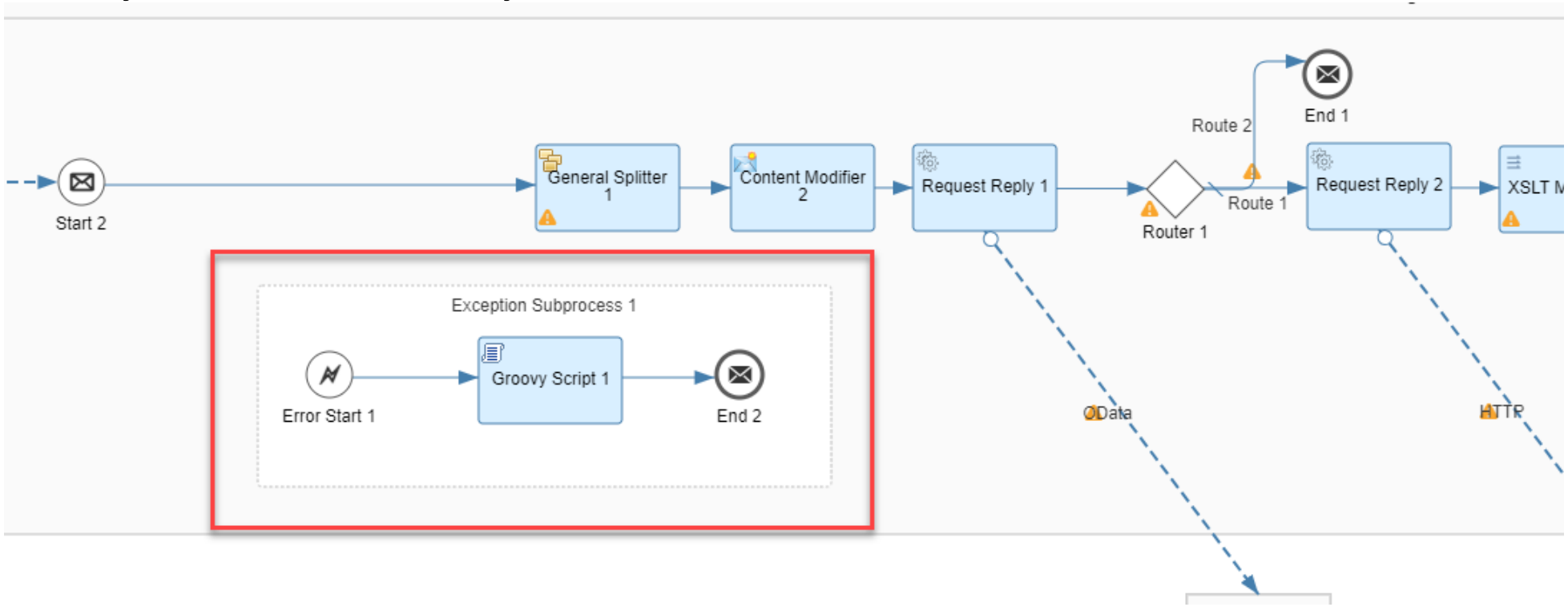
You can choose the following grand-types:

- 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



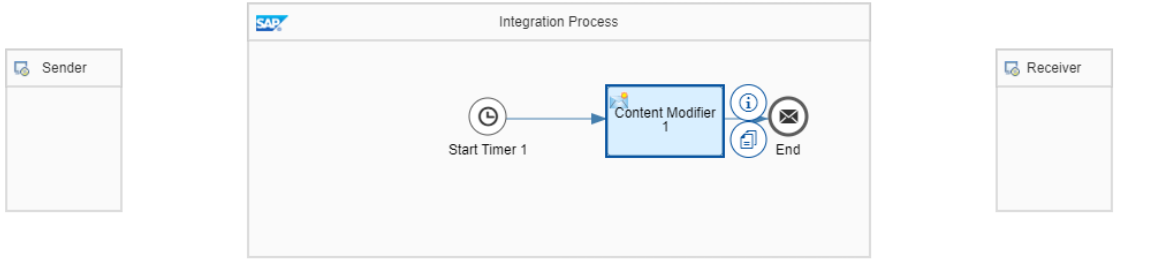
Summary

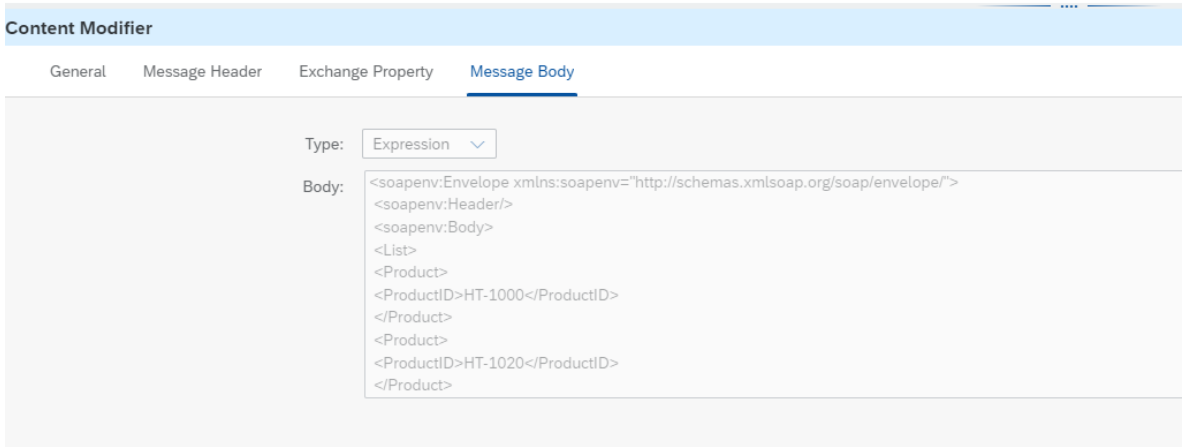
A special error subprocess can intercept an unexpected error using an Exception Start Event. After interception, various processing steps can be implemented. For instance, it would be appropriate to store process values or message content following an error. Additionally, informing the sender about the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





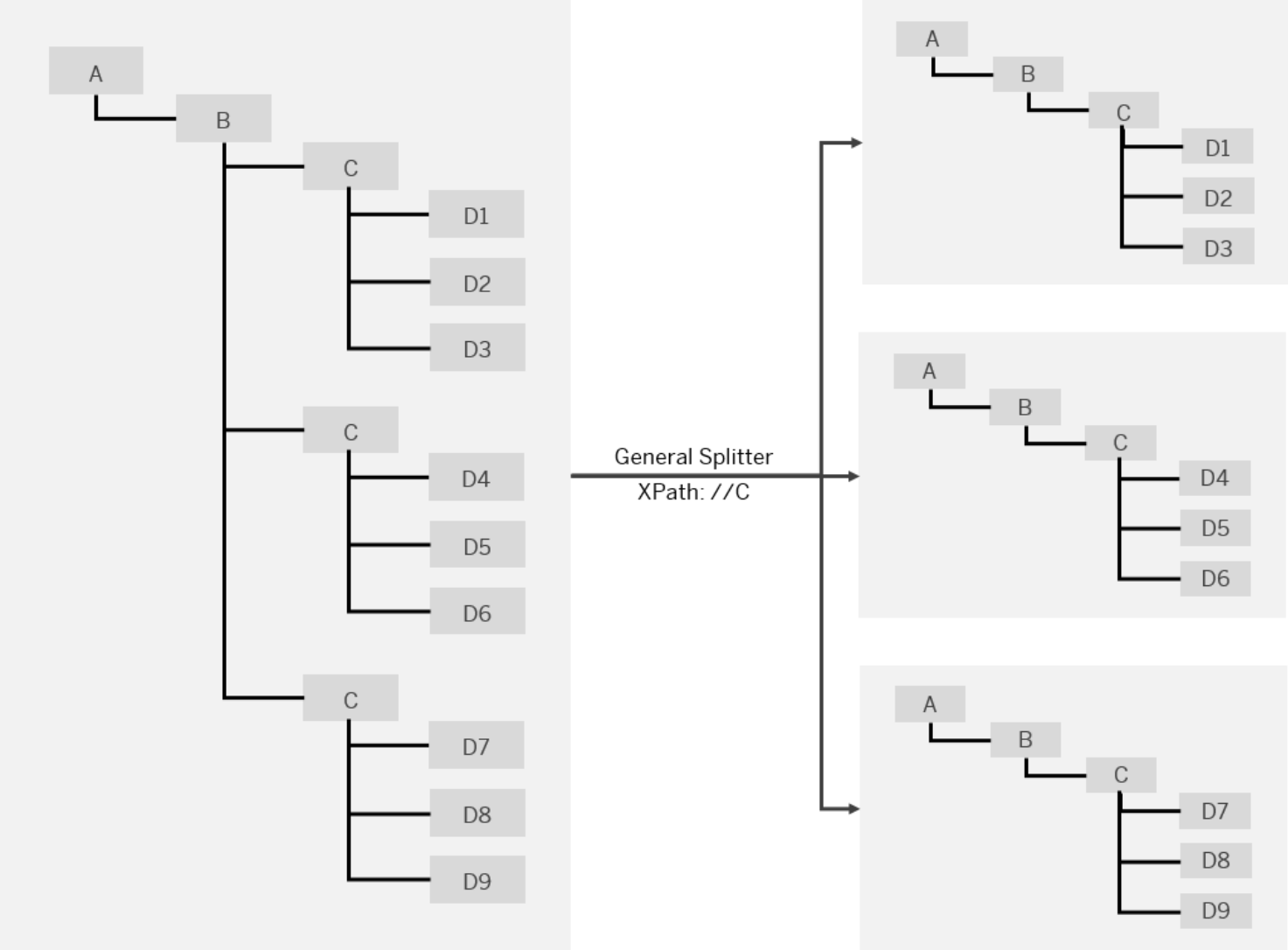
Key Summary Points – Unit 5

Show Integration Patterns

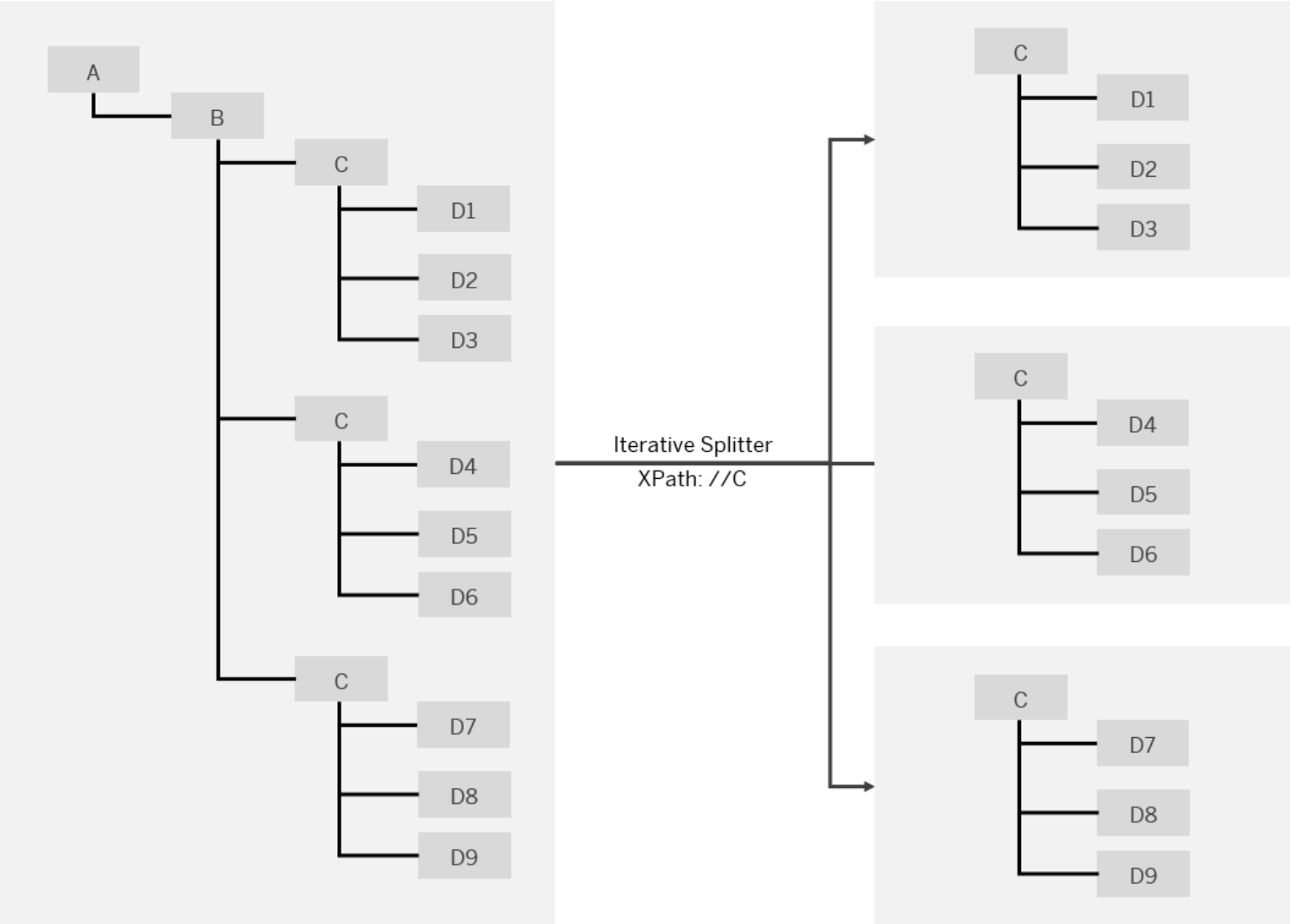
The following integration patterns are included in the example package:

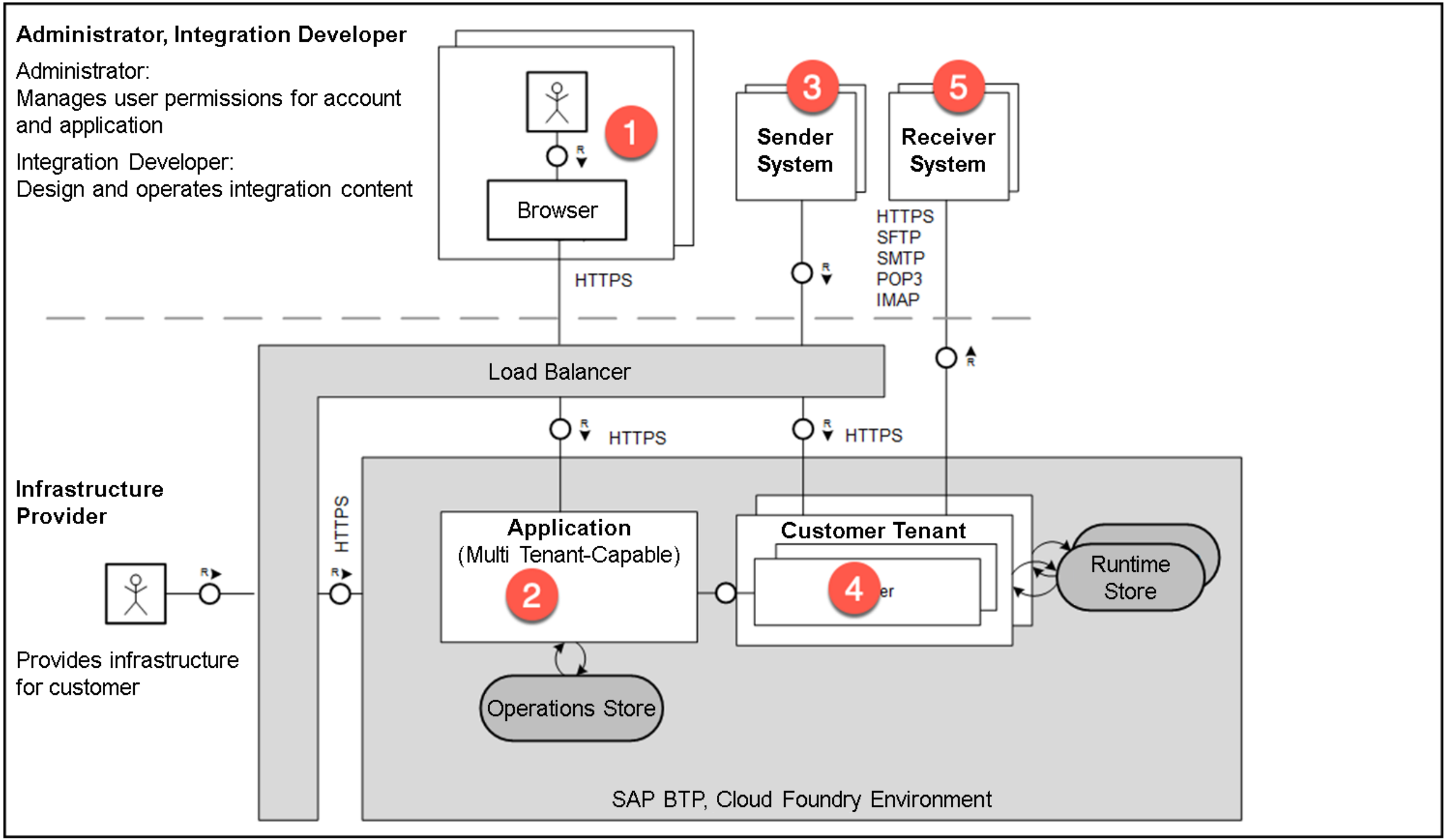
- Aggregator
- Composed Message Processor
- Content-Based Routing
- Content Enricher
- Content Filter
- Message Filter
- Recipient List
- Resequencer
- Scatter-Gather
- Splitter
- Quality of Service Exactly Once

Key Summary Points – Unit 5



Key Summary Points – Unit 5




Administrator, Integration Developer
Administrator:
Manages user permissions for account and application
Integration Developer:
Design and operates integration content
1
Browser
HTTPS
3
Sender System
5
Receiver System
HTTPS
SFTP
SMTP
POP3
IMAP
Load Balancer
HTTPS
HTTPS
Infrastructure Provider
Provides infrastructure for customer
HTTPS
Application
(Multi Tenant-Capable)
2
Customer Tenant
4
Runtime Store
Operations Store
SAP BTP, Cloud Foundry Environment