Unit 4 – Managing Cloud Integration

Developing with SAP Integration Suite

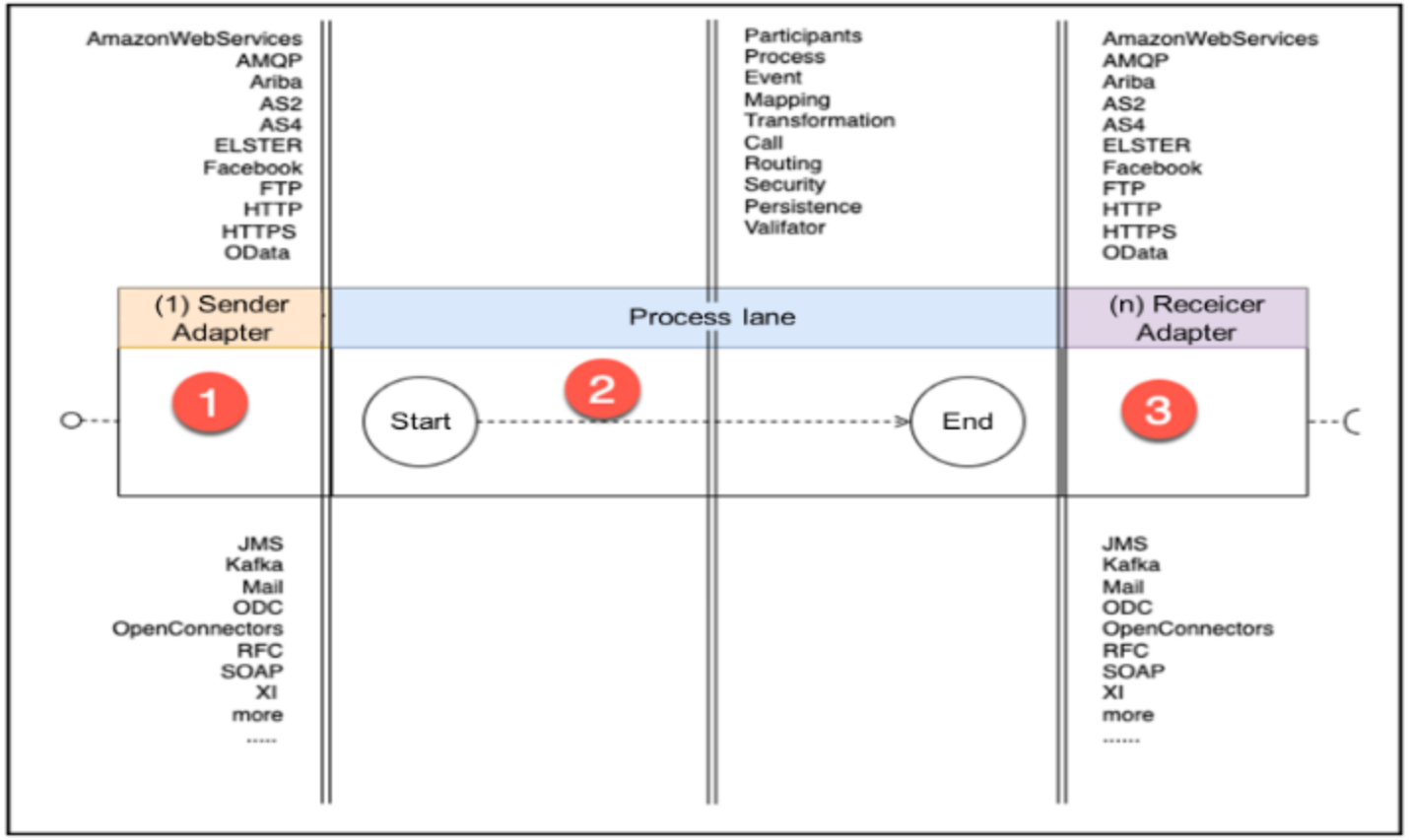
C_CPI_15

Agenda

- Introducing Cloud Integration
- Business Scenario
- Explaining Development Cycle
- Message Monitoring and Logging
- Camel Data Model and Simple Expression Language

Introducing Cloud Integration

- Supports end-to-end process integration through exchange of messages
- Based on open source framework Camel from Apache Software Foundation
- Core capabilities of SAP BTP Integration Suite
- Low Code / No Code approach



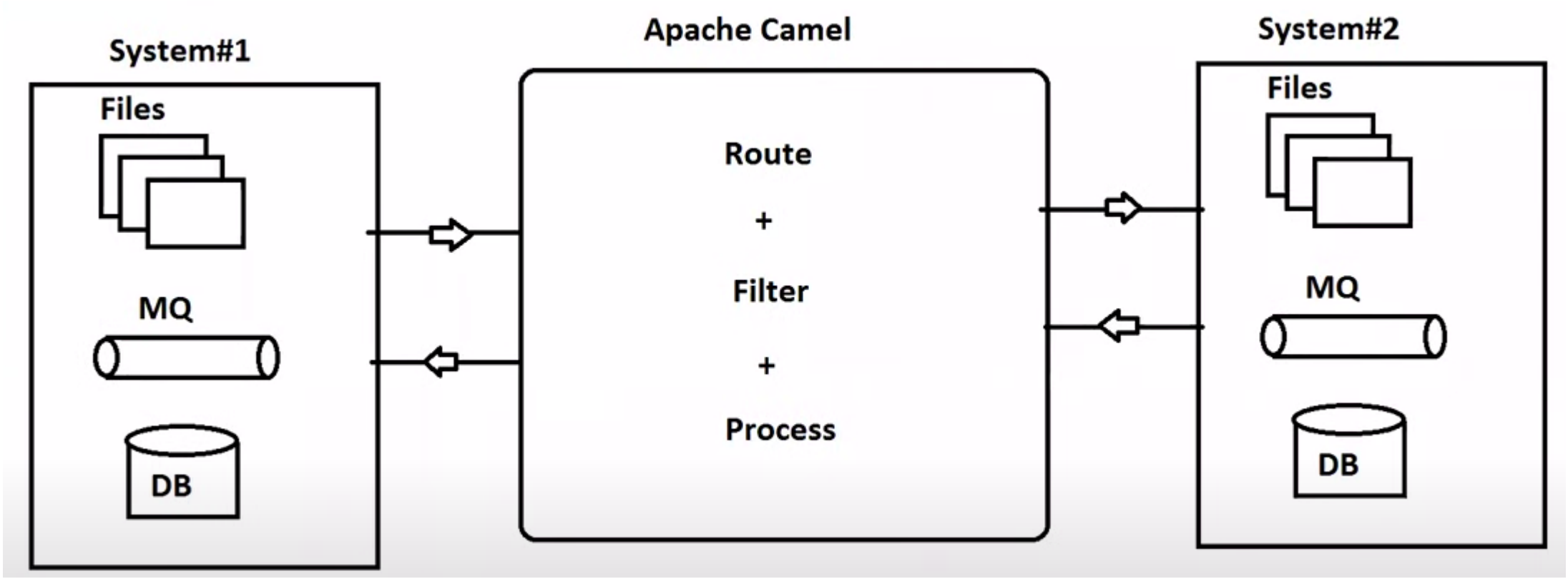
Key Features of Cloud Integration

- 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

Apache Camel - Challenge

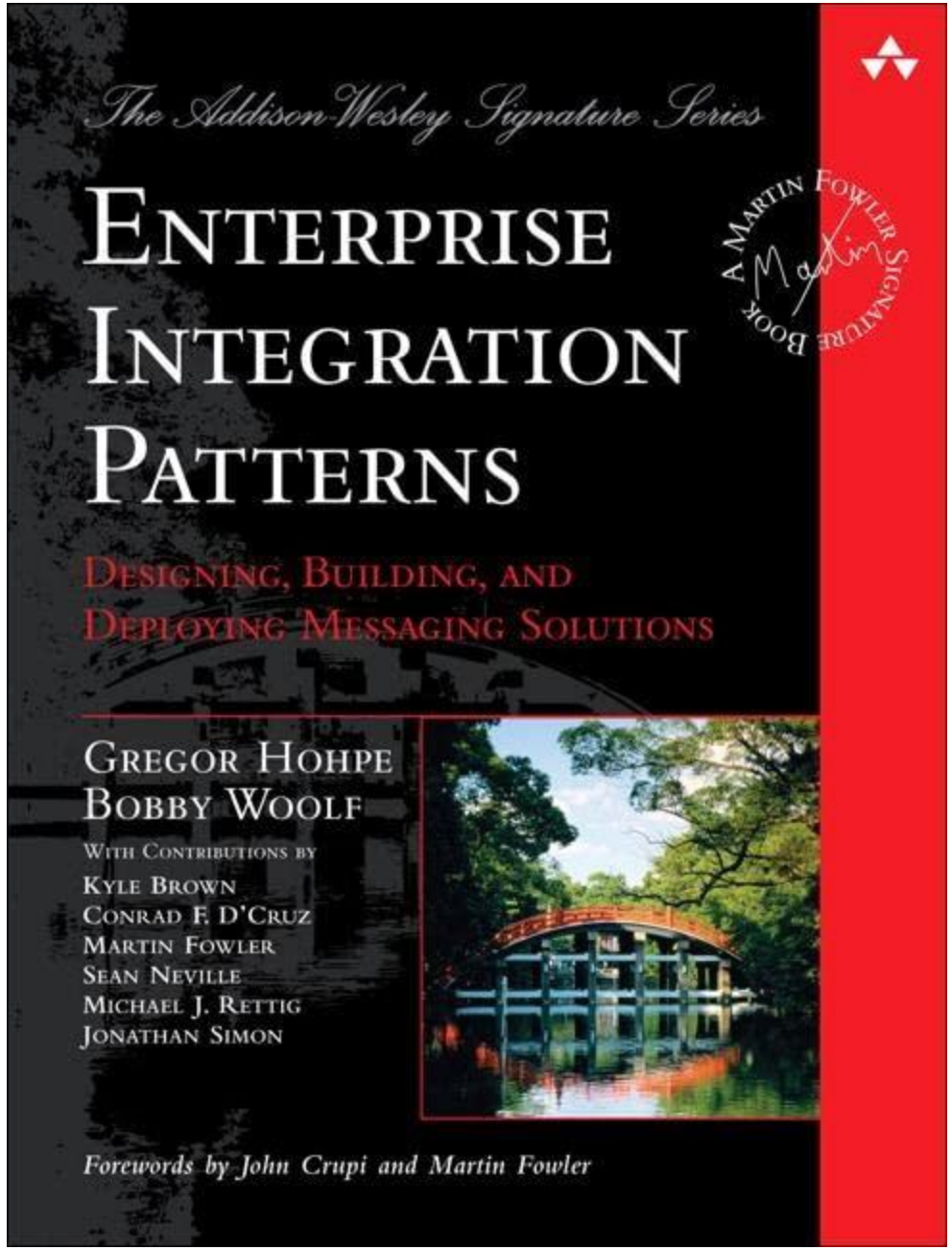
- Companies have data in various systems
- Need to move data between various systems
- Writing a program entails understanding…
  - Protocols of the various systems
  - Data formats of the various systems
  - And so much more

Apache Camel



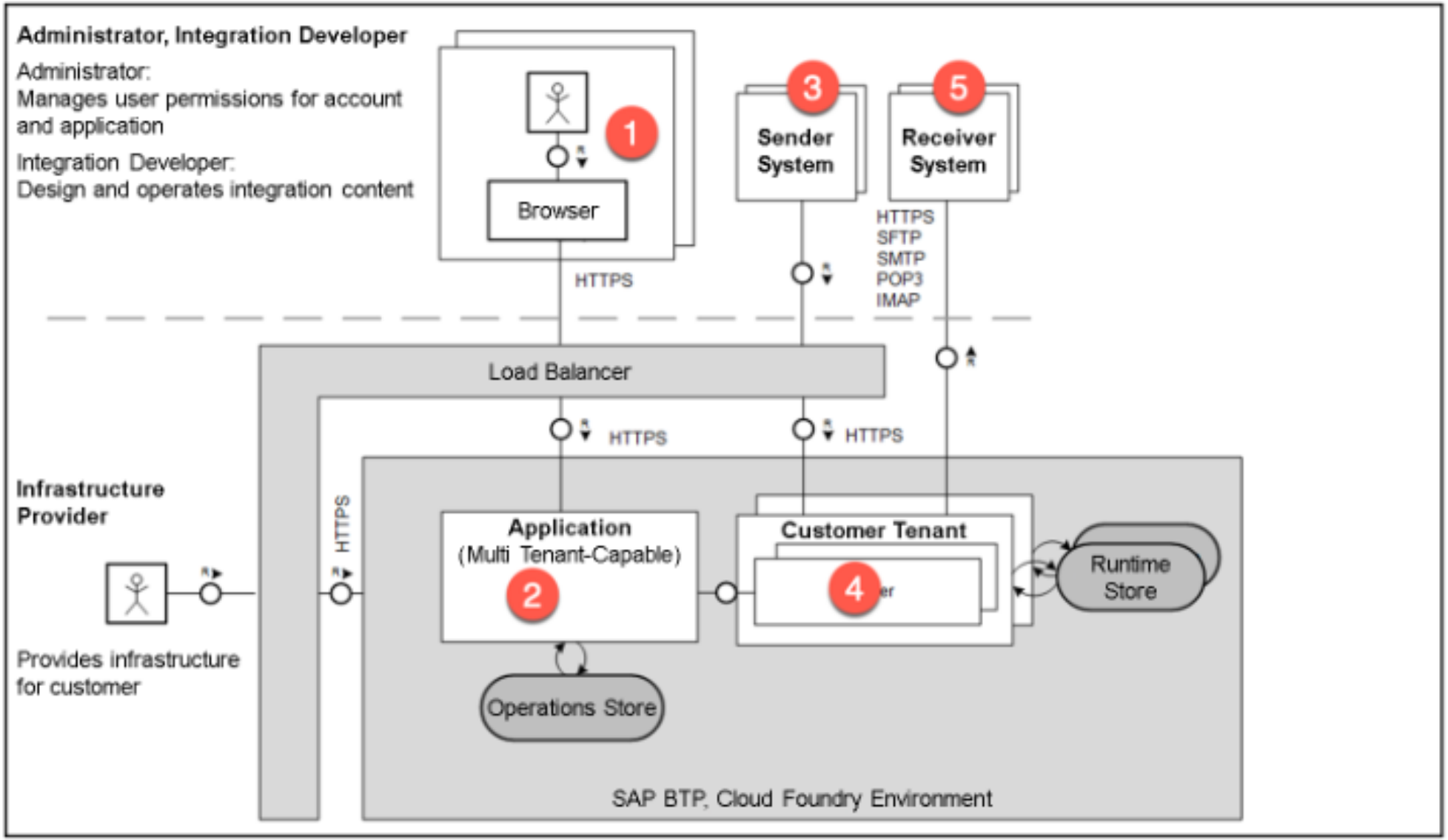
Apache Camel

- Java library that helps you write integrations and run them
- Define your integration flows
  - Where you want to pull data from
  - What you can or cannot do with the data
  - Where the data needs to go
  - And a lot more…
- Comes with built-in set of patterns you can use in integration flows
  - Splitter pattern – split the message based on how you want
  - Content based routing pattern – route messages based on content
  - And a lot more…


The Addison-Wesley Signature Series
A Martin Fowler Signature Book
Enterprise Integration Patterns
Designing, Building, and Deploying Messaging Solutions
Gregor Hohpe
Bobby Woolf
With Contributions by
Kyle Brown
Conrad F. D'Cruz
Martin Fowler
Sean Neville
Michael J. Rettig
Jonathan Simon
Forewords by John Crupi and Martin Fowler

SAP Cloud Integration

- Adds enterprise features to Apache Camel
- Engineered for cloud
  - Multitenancy
  - Rolling software updates
  - Horizontal scalability
- Strong focus on security including data isolation
- Used by SAP SaaS solutions
  - SAP S/4HANA
  - SAP SuccessFactors and so on…



The Entire Implementation

Demo: Building sample integration

- Building integration from scratch
- Predefined integration content

Business Scenario

- Company A sells goods to customers
- Some products cannot be delivered on time
- Inform customers who ordered these products about delay

Task flow


Send a list of Product IDs (SOAP)
Is Product ID in DB ?
Do this for all products
Get all Sales Orders for Product ID
Do this for all products in DB
Get Sales Header for Sales Order
Do this for all Sales Orders
Select Customer ID and write to Data Store
Do this for all Sales Headers
Laptop
iPhone
TV
Laptop
iPhone
TV
Laptop (Sales orders)
• 501
• 502
TV (Sales orders)
• 601
• 602
• 603
501 => Header 1
502 => Header 2
601 => Header 3
602 => Header 4
603 => Header 5
Header 1 => Customer 1
Header 2 => Customer 2
Header 3 => Customer 3
Header 4 => Customer 4
Header 5 => Customer 5
Write to Data Store

Demo of working solution…

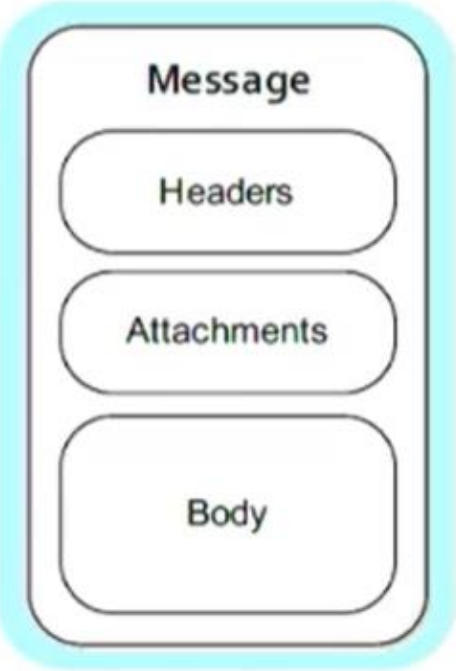
- Demo of the working solution
- Explore Cloud Integration
  - Discover area
  - Design area
  - Monitor area
  - Settings area

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

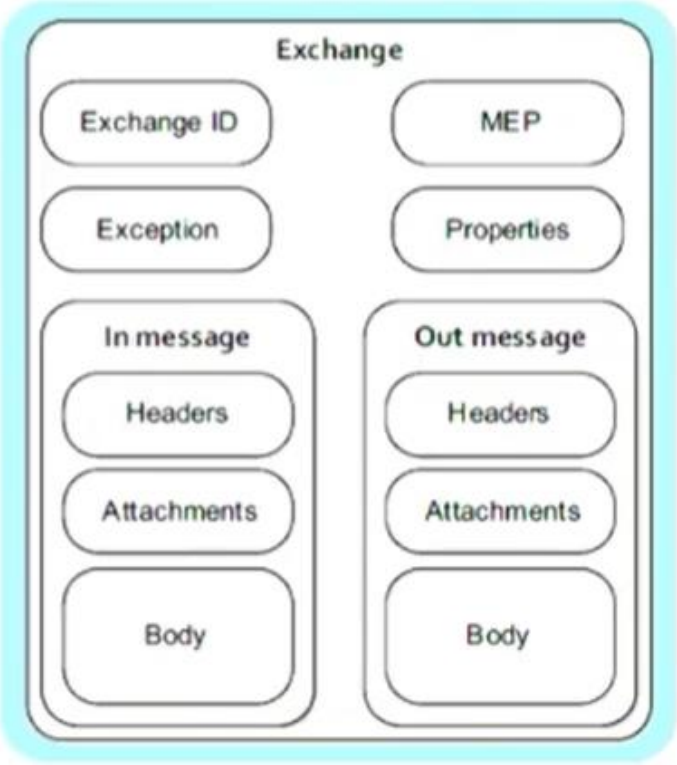


Basic concepts of Cloud Integration flow…

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



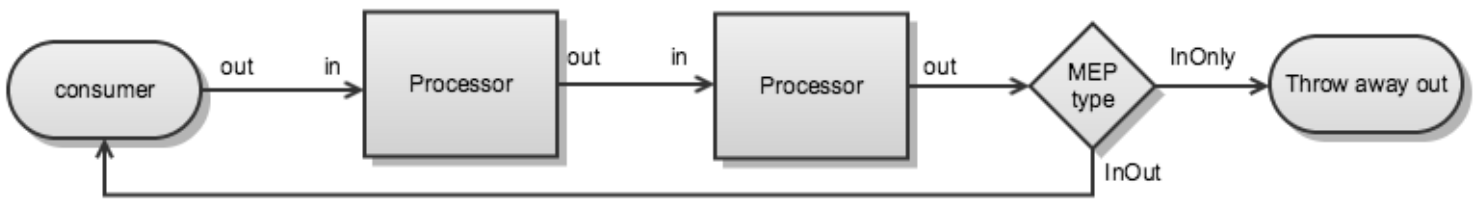
MESSAGE EXCHANGE PATTERNS AND THE EXCHANGE OBJECT

The Camel API is influenced by APIs such as JBI specification, CXF which defines a concept called Message Exchange Patterns (MEP for short).

The MEP defines the messaging style used such as one-way (InOnly) or request-reply (InOut), which means you have IN and optionally OUT messages. This closely maps to other APIs such as WS, WSDL, REST, JBI and the likes.

The Exchange API provides two methods to get a message, either `getIn` or `getOut`. Obviously the `getIn` gets the IN message, and the `getOut` gets the OUT message.

FLOW OF AN EXCHANGE THROUGH A ROUTE



- The out message from each step is used as the in message for the next step
- if there is no out message then the in message is used instead
- For the InOut MEP the out from the last step in the route is returned to the producer. In case of InOnly the last out is thrown away

TIP

Beware of `getOut` to check if there is an out message

`exchange.getOut` creates an out message if there is none. So if you want to check if there is an out message then you should use `exchange.hasOut` instead.

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}

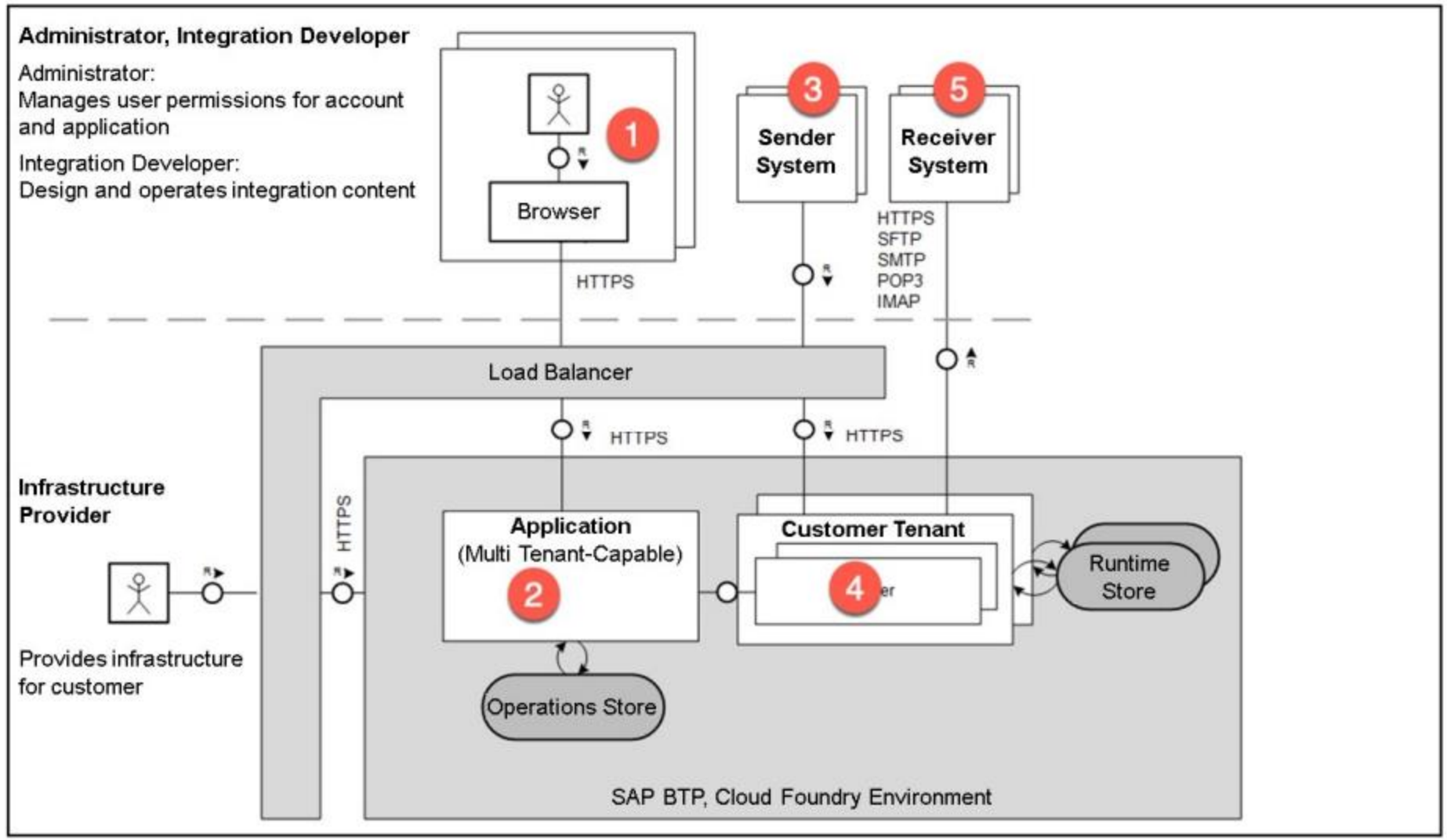


Figure 78: The Entire Implementation

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)

Key Summary Points – Unit 4

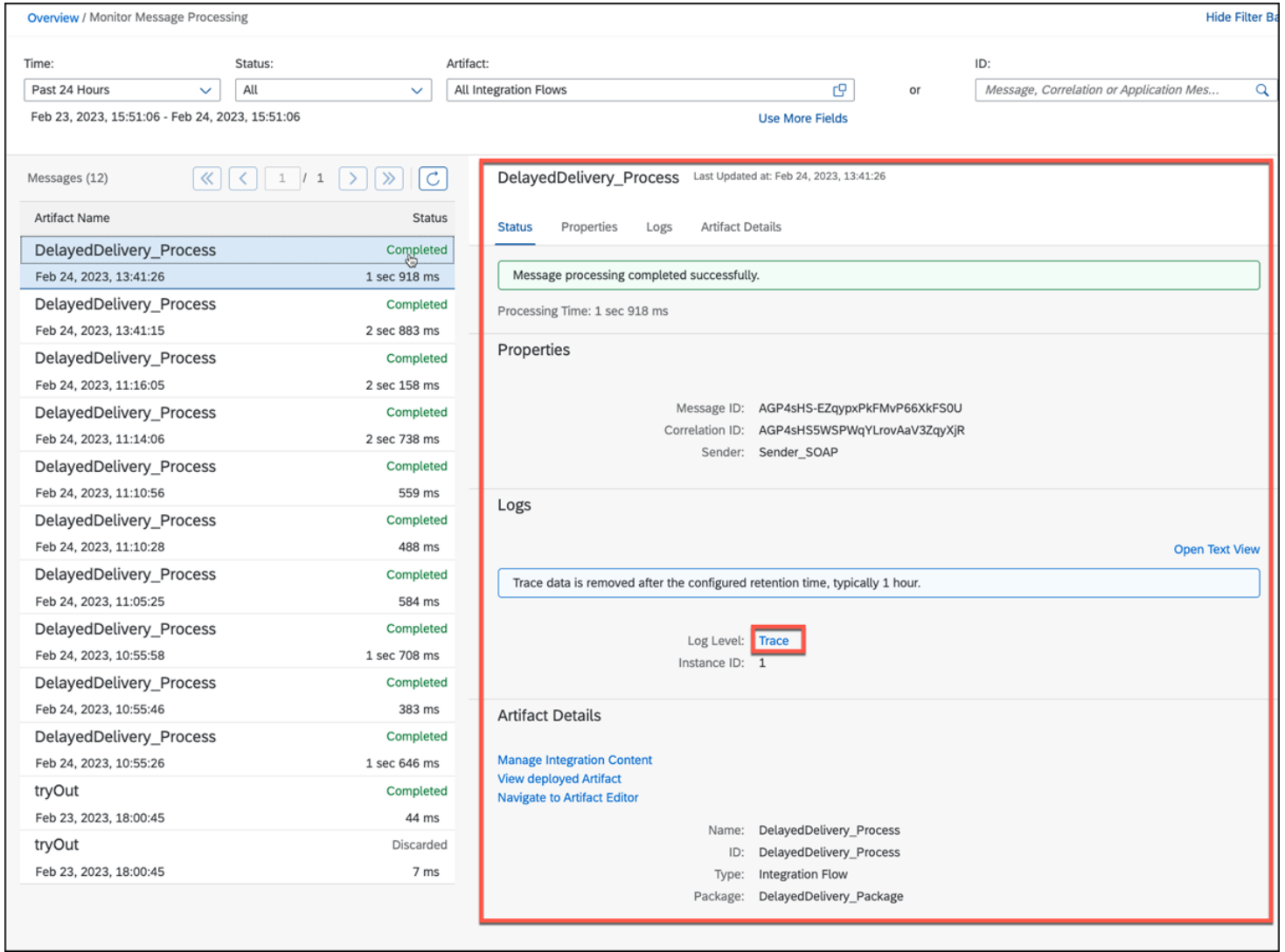
Q2. What needs to be enabled to work in debugging mode within the monitor?

- ✓ 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.

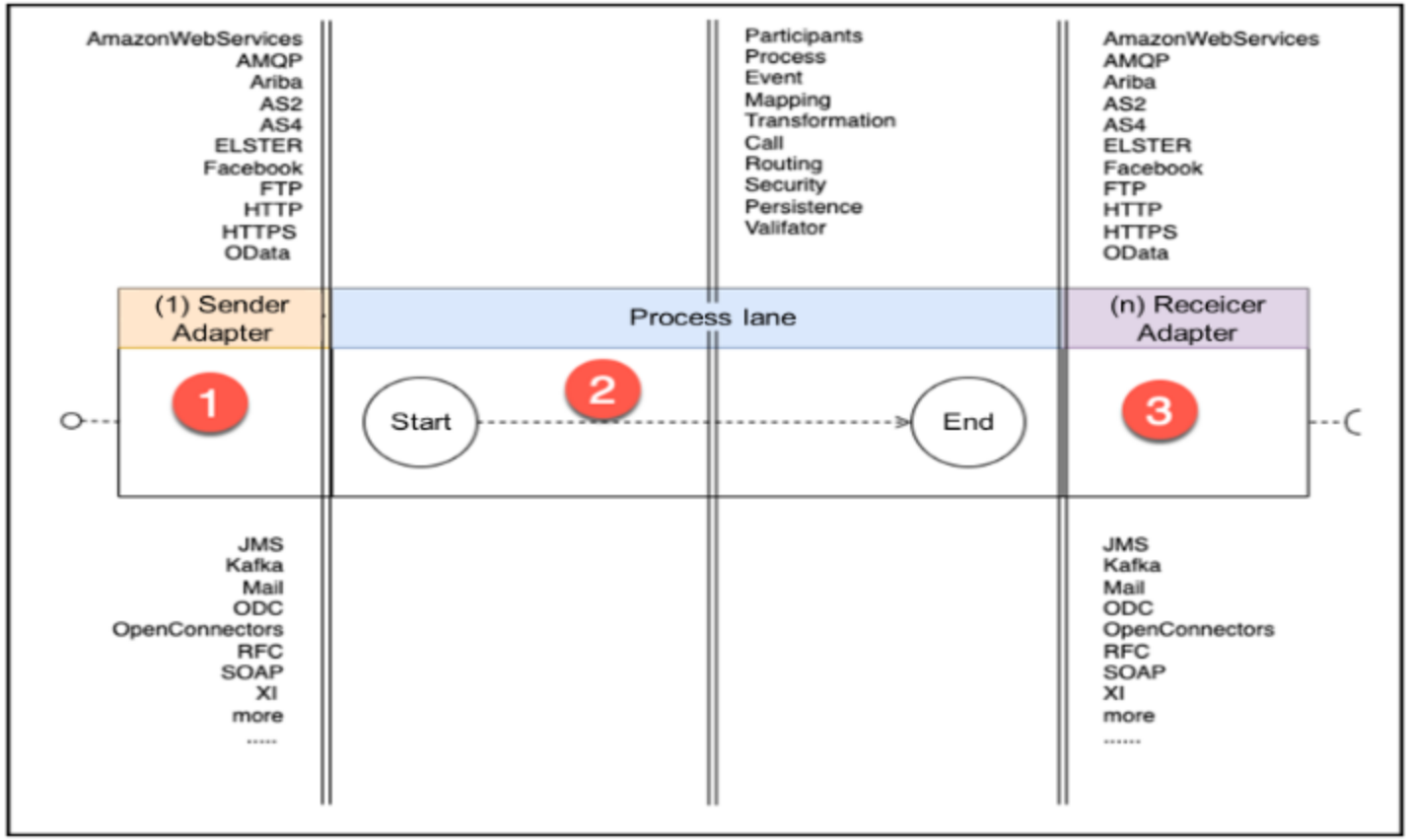
SAP Business Accelerator Hub - New name

✓ 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

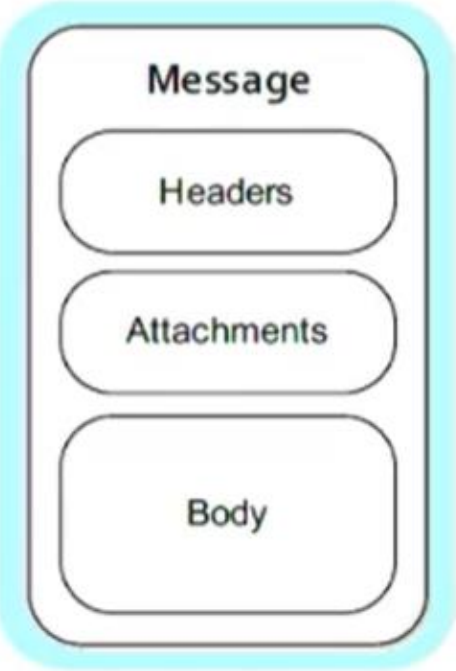
- Integration flow has 0-1 sender adapter
- Message is delivered via an endpoint
  - If an adapter is configured
- Process is started via Start event
- Different ways messages can be processed
- Receiver adapters can be configured
- Message processing can be synchronous or asynchronous

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

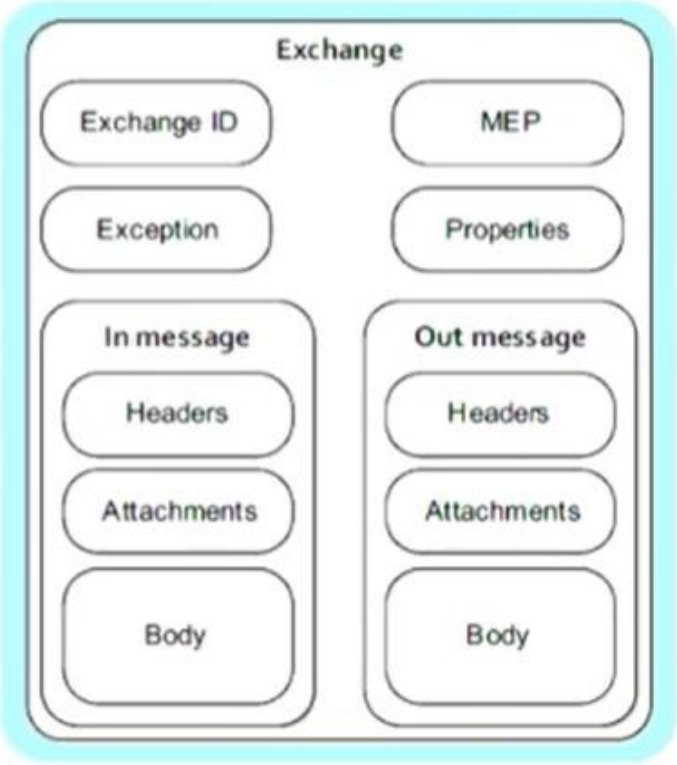


Basic concepts of Cloud Integration flow…

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

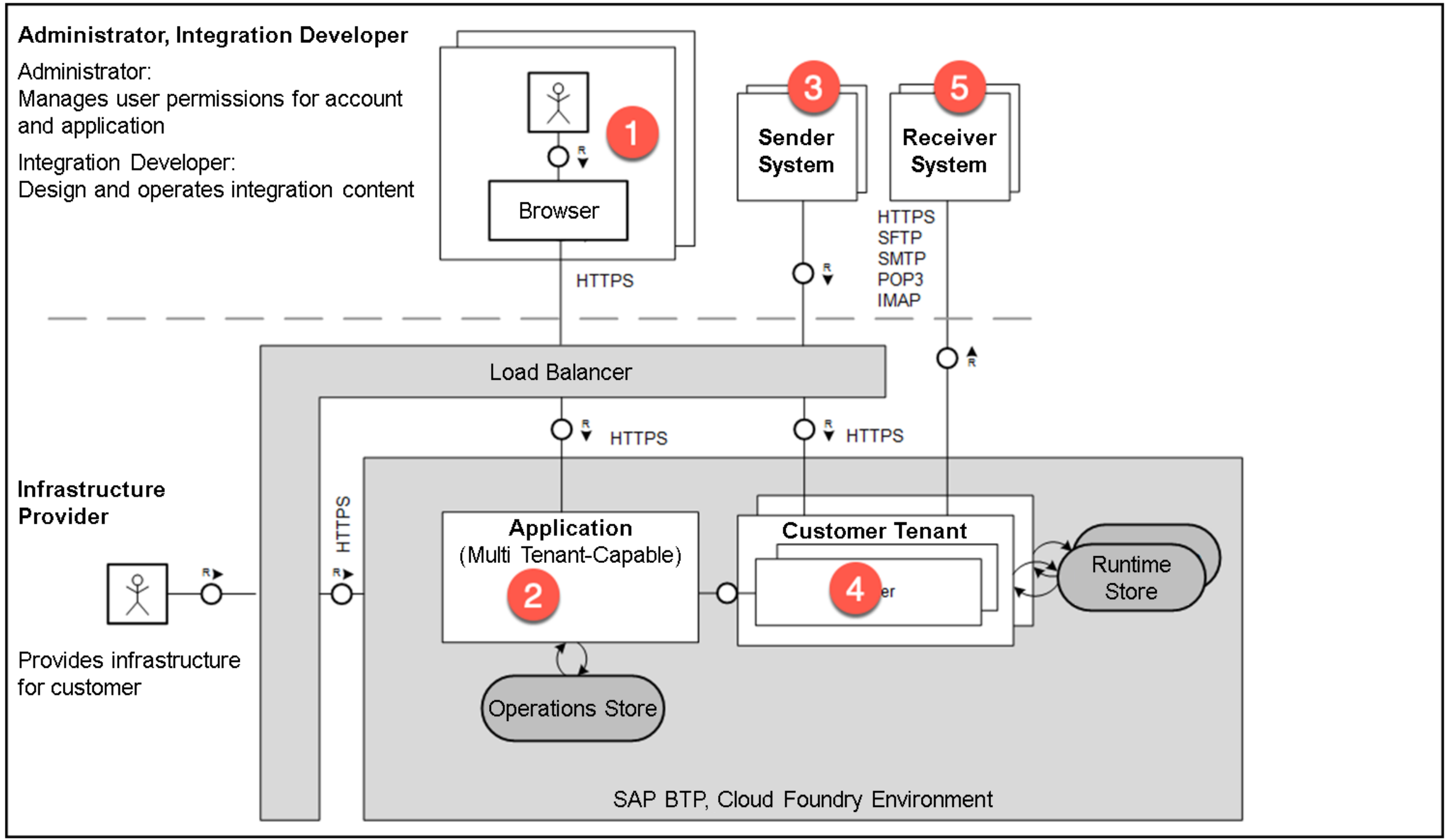
Set Exchange Parameters with Groovy SDK

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;
        }

```


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