

# Week 1 Unit 5: Working with Data in SAP HANA Cloud Integration

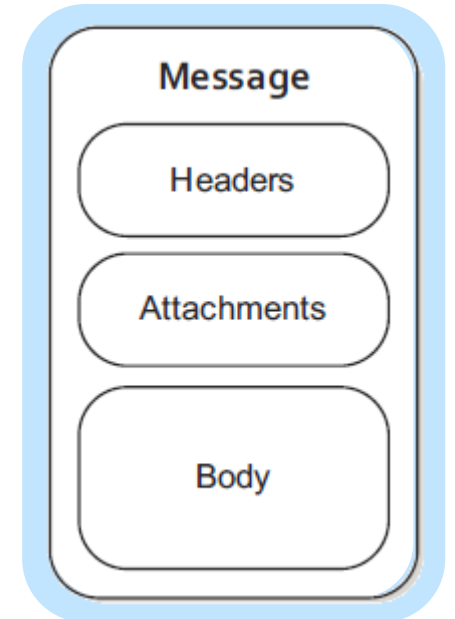
# Working with Data in SAP HANA Cloud Integration

## Camel's message model\* (1)

### 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



\* Taken from the book: "Camel in Action" (2011) by Claus Ibsen and Jonathan Anstey, Manning Publications Co., ISBN 978-1-935182-36-8

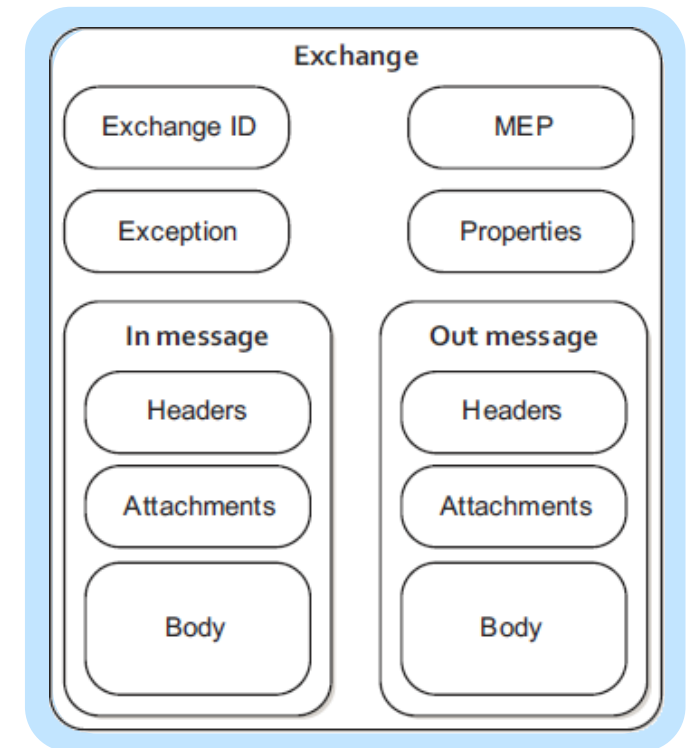
# Working with Data in SAP HANA Cloud Integration

## Camel's message model\* (2)

### 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



\* Taken from the book: “Camel in Action” (2011) by Claus Ibsen and Jonathan Anstey, Manning Publications Co., ISBN 978-1-935182-36-8

# Working with Data in SAP HANA Cloud Integration

## Understanding Camel's message model (1)

### Goals

Store data in the message header and in the properties of the exchange

Retrieve data from header and properties to build the reply message

#### Input Message

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

#### Output Message

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

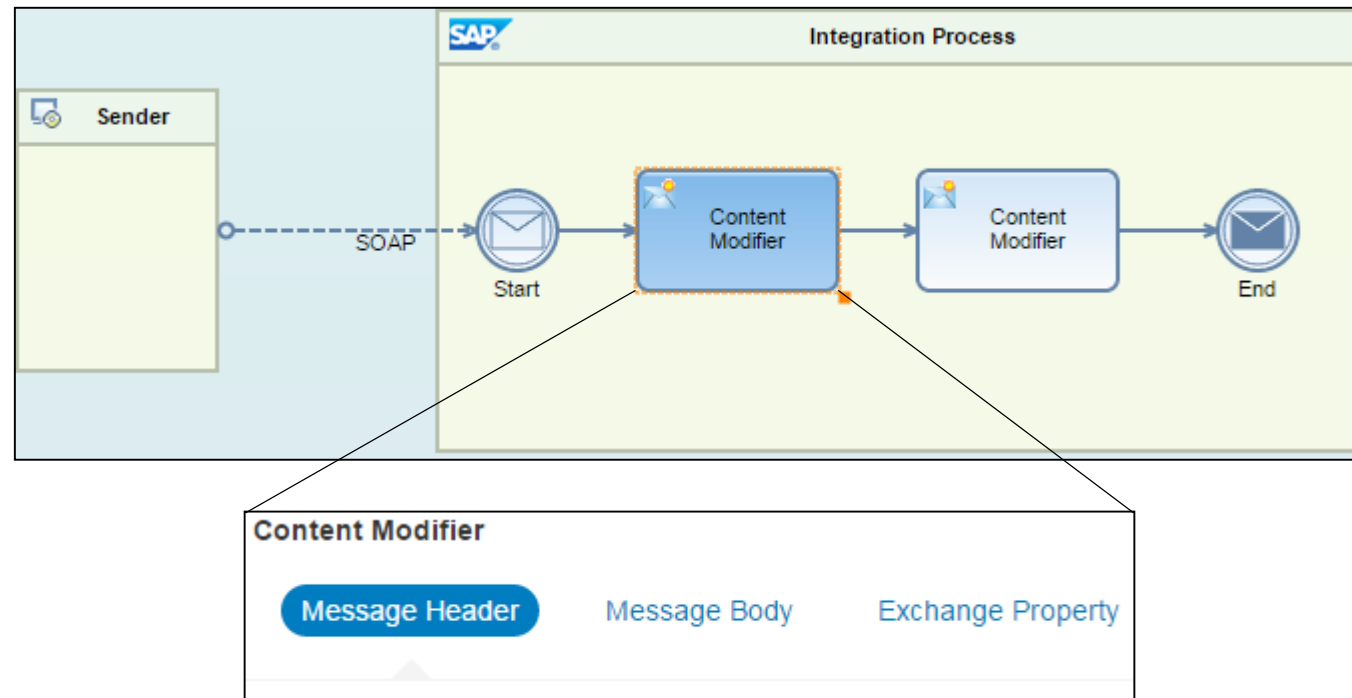
# Working with Data in SAP HANA Cloud Integration

## Understanding Camel's message model (2)

### Goals

Store data in the message header and in the properties of the exchange

Retrieve data from header and properties to build the reply message



# Working with Data in SAP HANA Cloud Integration

## Understanding Camel's message model (3)

### Goals

Store data in the message header and in the properties of the exchange

Retrieve data from header and properties to build the reply message

**Content Modifier**

**Message Header** Message Body Exchange Property

Header Add Delete

<input type="checkbox"/> Name:	Type	Value	
<input type="checkbox"/> OrderNo	XPath <span>▼</span>	//orderNumber	<span>🗑️</span>
	java.lang.String		

**Content Modifier**

Message Header Message Body **Exchange Property**

Properties Add Delete

<input type="checkbox"/> Name:	Type	Value	
<input type="checkbox"/> msg	Expression <span>▼</span>	\${in.body}	<span>🗑️</span>

Using Camel's  
Simple Expression Language

<http://camel.apache.org/simple.html>

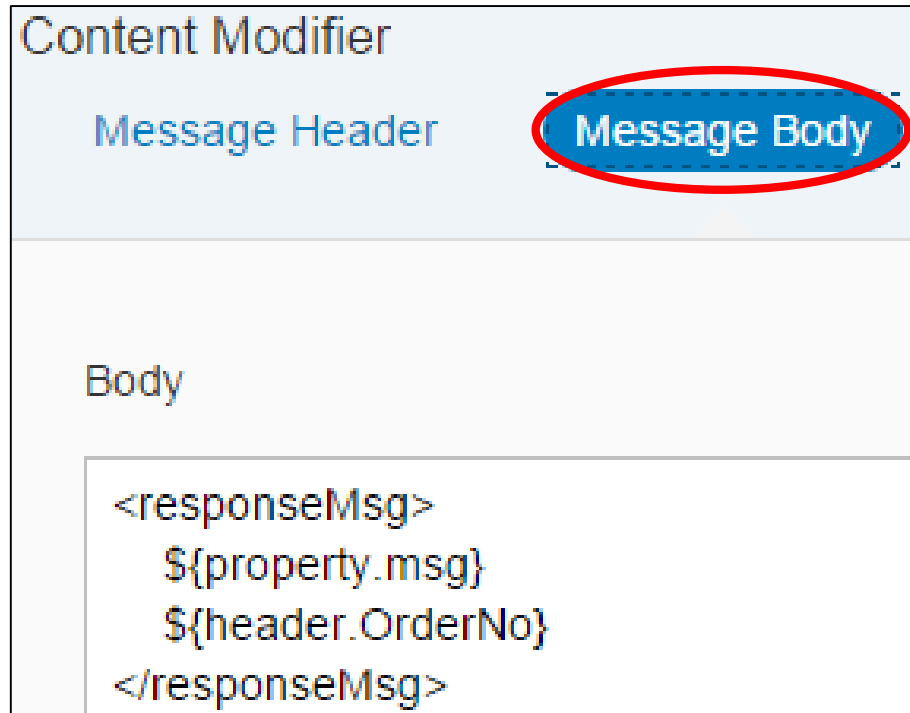
# Working with Data in SAP HANA Cloud Integration

## Understanding Camel's message model (4)

### Goals

Store data in the message header and in the properties of the exchange

Retrieve data from header and properties to build the reply message



Header	Value
OrderNo	10249
Date	Tue, 04 Aug 2015 11:07:08 GMT
Content-Length	307
#status#	HTTP/1.1 200 OK
Set-Cookie	JSESSIONID=ECE81E7C389FBCBD005C67303100...
Set-Cookie	JTENANTSESSIONID_aaio005=VeKqeXrB2Qa14i...
Set-Cookie	BIGipServeriflmapdm102aaio005avtaiod-aaio005...
Content-Type	text/xml; charset= UTF-8
Server	SAP

# Working with Data in SAP HANA Cloud Integration

## Understanding Camel's message model (5)

### Goals

Store data in the message header and in the properties of the exchange

Retrieve data from header and properties to build the reply message

Channel

General Adapter Specific

General

WS-Security

CONNECTION DETAILS

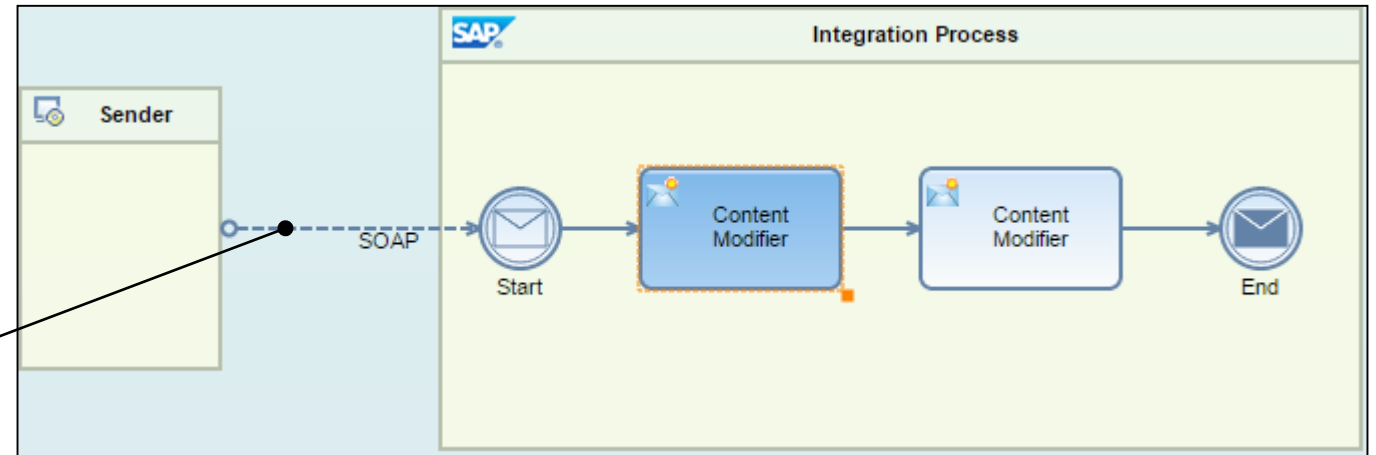
\*Address: /HCI\_Week\_1\_Unit\_5

URL to WSDL:  Select

Service:

Endpoint:

\*Processing Settings: Standard ▼





# Working with Data in SAP HANA Cloud Integration

## Demo

**DEMO**

Working with Data in  
SAP HANA Cloud  
Integration



# Working with Data in SAP HANA Cloud Integration

## What you've learned in this unit

---

- SAP HANA Cloud Integration relies on the Apache Camel integration framework and because of this inherits its data model
- Camel's data model differentiates between a message and an exchange
- How to store data in the header area of a message and in the properties area of an exchange
- How to retrieve data from the different locations and create a response message in the message body
- How to configure a SOAP channel





# Thank you

Contact information:

[open@sap.com](mailto:open@sap.com)

open**SAP**

# © 2016 SAP SE or an SAP affiliate company. All rights reserved.

---

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. Please see <http://global12.sap.com/corporate-en/legal/copyright/index.epx> for additional trademark information and notices.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors.

National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP SE or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP SE or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, which speak only as of their dates, and they should not be relied upon in making purchasing decisions.