Replacing a commercial integration platform with an open source ESB

Magnus Larsson| magnus.larsson@callistaenterprise.se | Cadec 2010-01-20

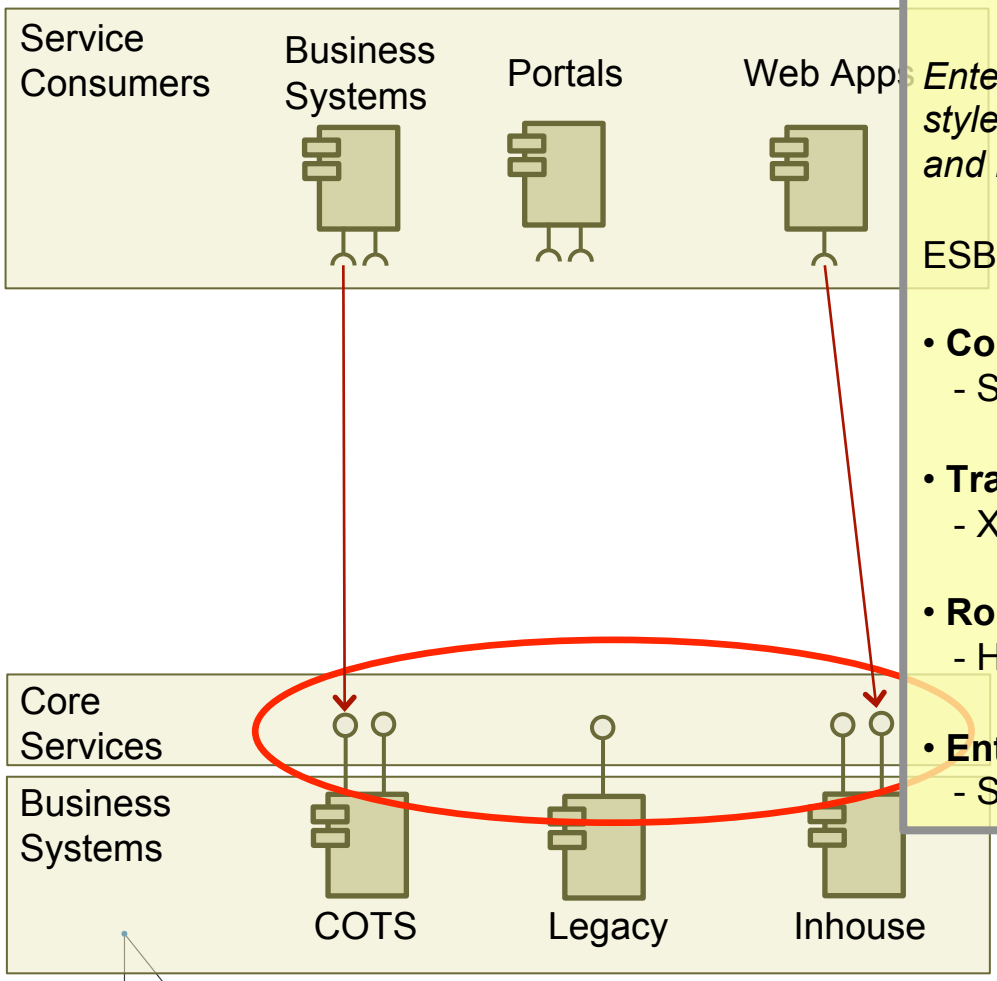
CALLISTA
Enterprise

Agenda

- The customer
- Phases
  - Problem definition
  - Proof of concepts
  - Establish platform
  - Migration of system landscape
- Summary

CALLISTA Enterprise

Recap: What is an ESB?



From CADEC 2009 (Open Source SOA):

Enterprise Service Bus (ESB) is an architectural style that support integrations in a loosely coupled and implementation independent way

ESB products typically handles:

- **Connectivity**
  - SOAP, Rest, Messaging, Database, FTP…
- **Transformation**
  - XML, CSV, Fixed Position…
- **Routing**
  - Header and/or Content based
- **Enterprise Integration Patterns**
  - Splitting, Aggregation, Resequencing…

CALLISTA Enterprise

About the customer

- Volvofinans Bank AB
  - Business Concept
    ”to actively support sale of the products marketed in Volvohandeln on the Swedish market through product financing and sale financing and with good profitability.”
  - Employees: 175
    - » IT department: 35
  - Customers
    - » Loan and Leasing: 230.000
    - » Credit Card: 1.100.000

CALLISTA Enterprise

VOLVOFINANS

Where are we?

- The customer
- Phases
  - Problem definition
  - Proof of concepts
  - Establish platform
  - Migration of system landscape
- Summary

CALLISTA Enterprise

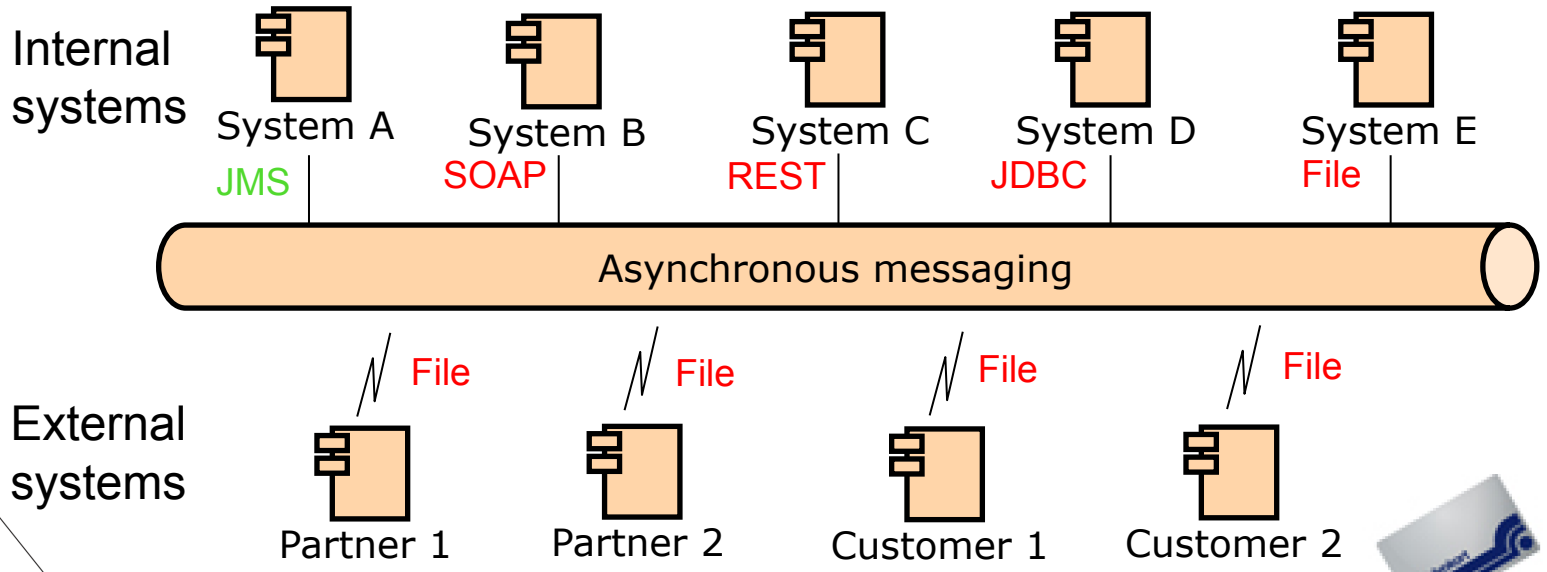
Problem definition

- An Enterprise Architecture workshop identified a number of concerns with the existing commercial integration platform
  - High complexity
    - Development
    - Runtime
    - Cause high cost and long delivery time for development and maintenance
  - High license cost (cpu based)
    - Forced a centralized hub-like usage
    - Preferred a federated (distributed) usage

CALLISTA Enterprise

Problem definition

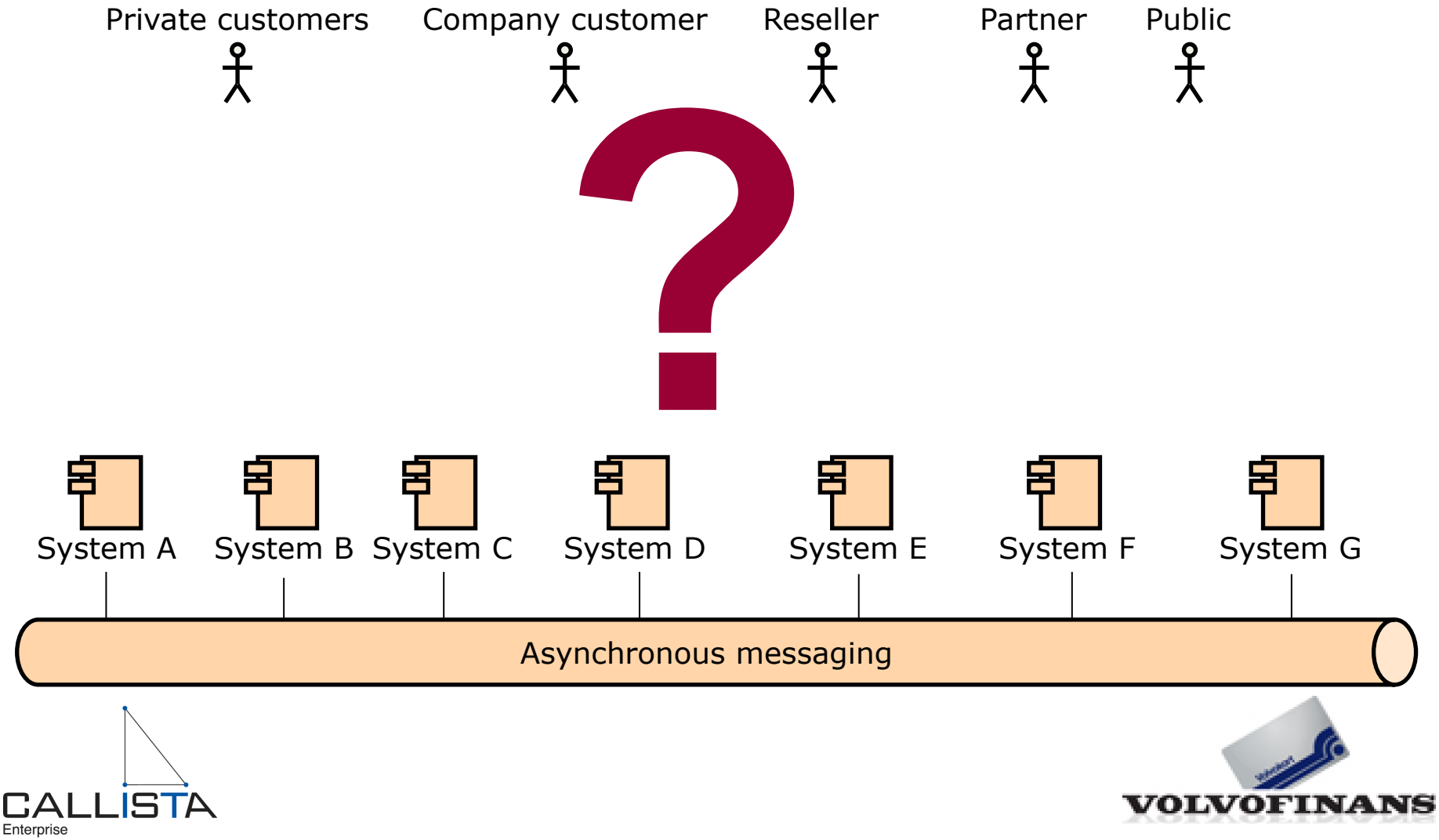
- Concerns with existing integration platform, cont…
  - Low flexibility
    - » All integrations needs to be message queuing “a like”
      - File transfers, db export/imports, synchronous services all needs to be fit into a fire and forget model…
      - Drives complexity and results in non robust solutions



CALLISTA Enterprise

VOLVOFINANS

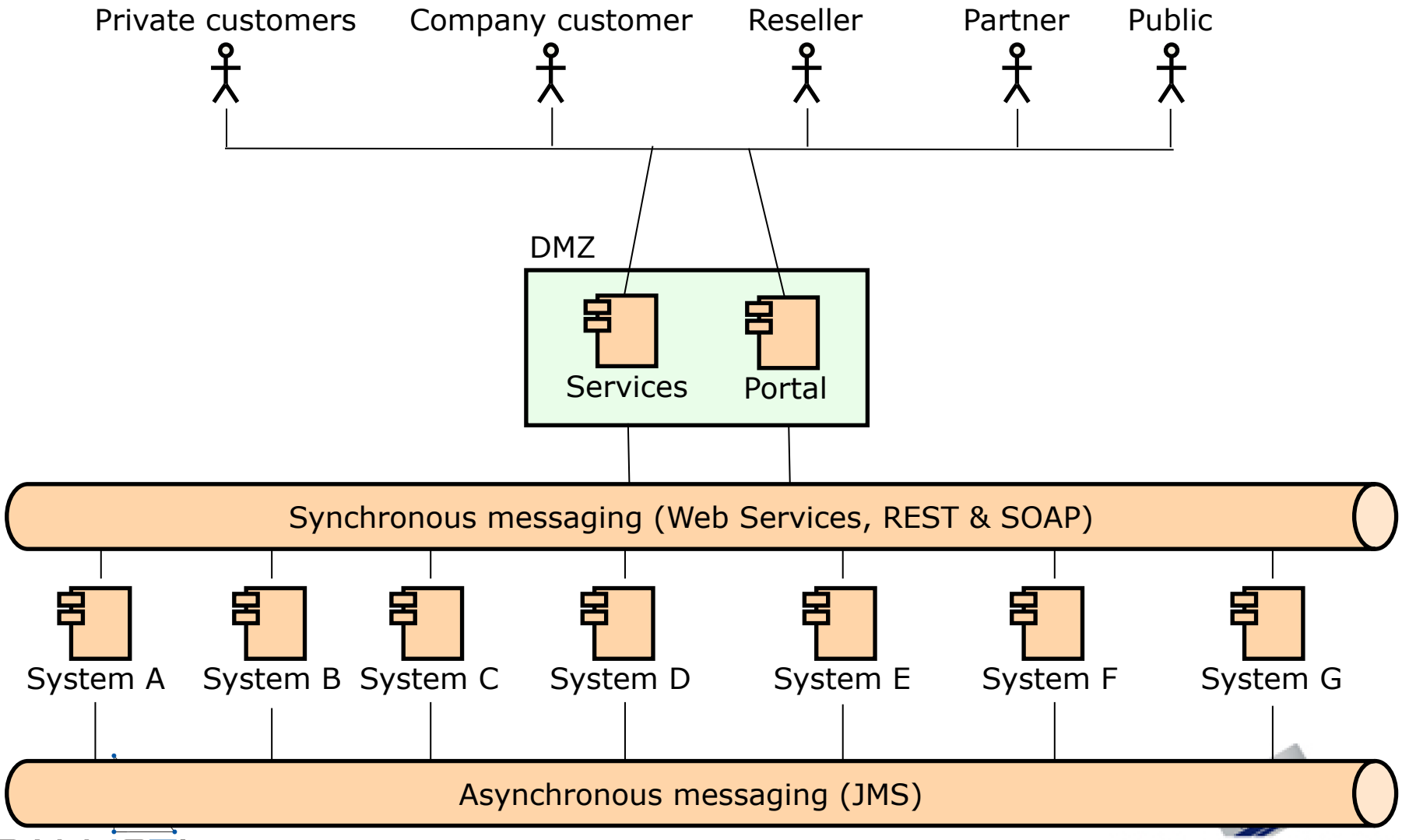
Problem definition -
Service enabling the system landscape



CALLISTA Enterprise

VOLVOFINANS

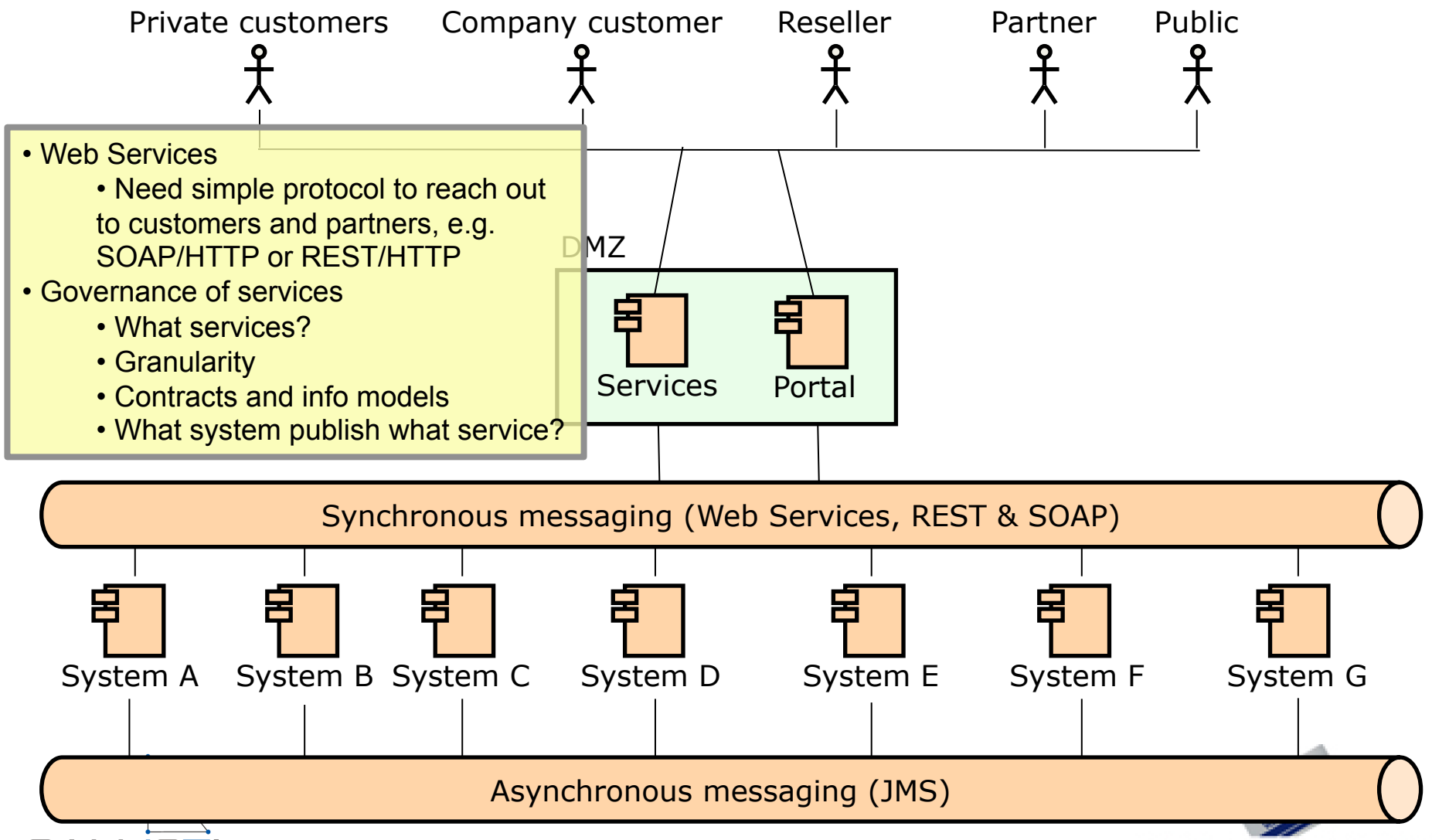
Problem definition - Service enabling the system landscape



CALLISTA Enterprise

VOLVOFINANS

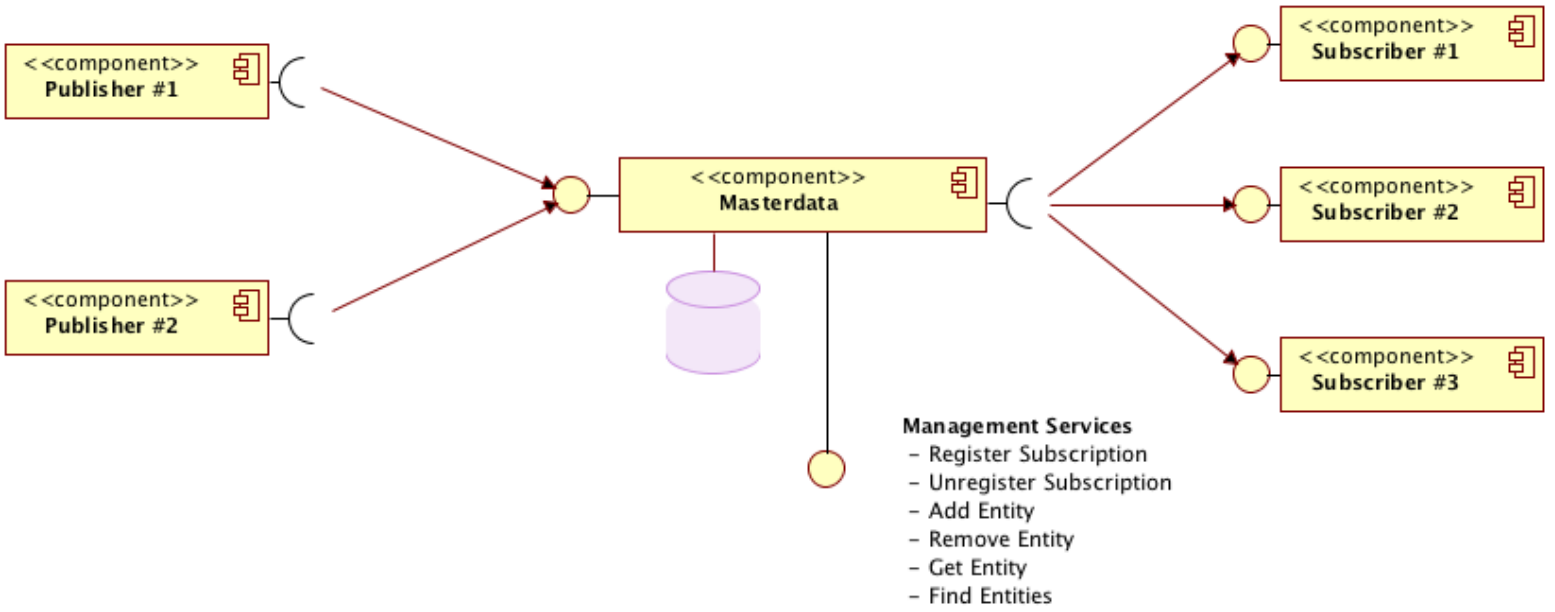
Problem definition - Service enabling the system landscape



CALLISTA Enterprise

VOLVOFINANS

Problem Definition - Masterdata



- Publish and Subscribe not only over JMS but also JDBC and file
- Dynamic registration of both data and subscribers
- Complex filter functions required
- Many types of patterns and protocols involved

CALLISTA Enterprise

VOLVOFINANS

Problem definition - Output

- Output
  - Need for better functional support in areas of
    - Service enabling the system landscape
    - File transfer and database export/import
    - Building masterdata solutions
  - Need to reduce complexity and cost for development of integrations
  - Decision
    - Test if an open source ESB can complement the existing commercial integration platform

CALLISTA Enterprise

Where are we?

- The customer
- Phases
  - Problem definition
  - Proof of concepts
  - Establish platform
  - Migration of system landscape
- Summary

CALLISTA Enterprise

Proof of Concepts – Selection of open source ESB

- Mule ESB was selected
  - The leading open source ESB in terms of
    - Functionality
    - Installed base
  - Support offering from MuleSoft
  - Based on Spring framework
    - All application development at the customer is already based on Spring
      - Customer already familiar with Spring and its tooling
    - Great potential for reduced complexity and cost!

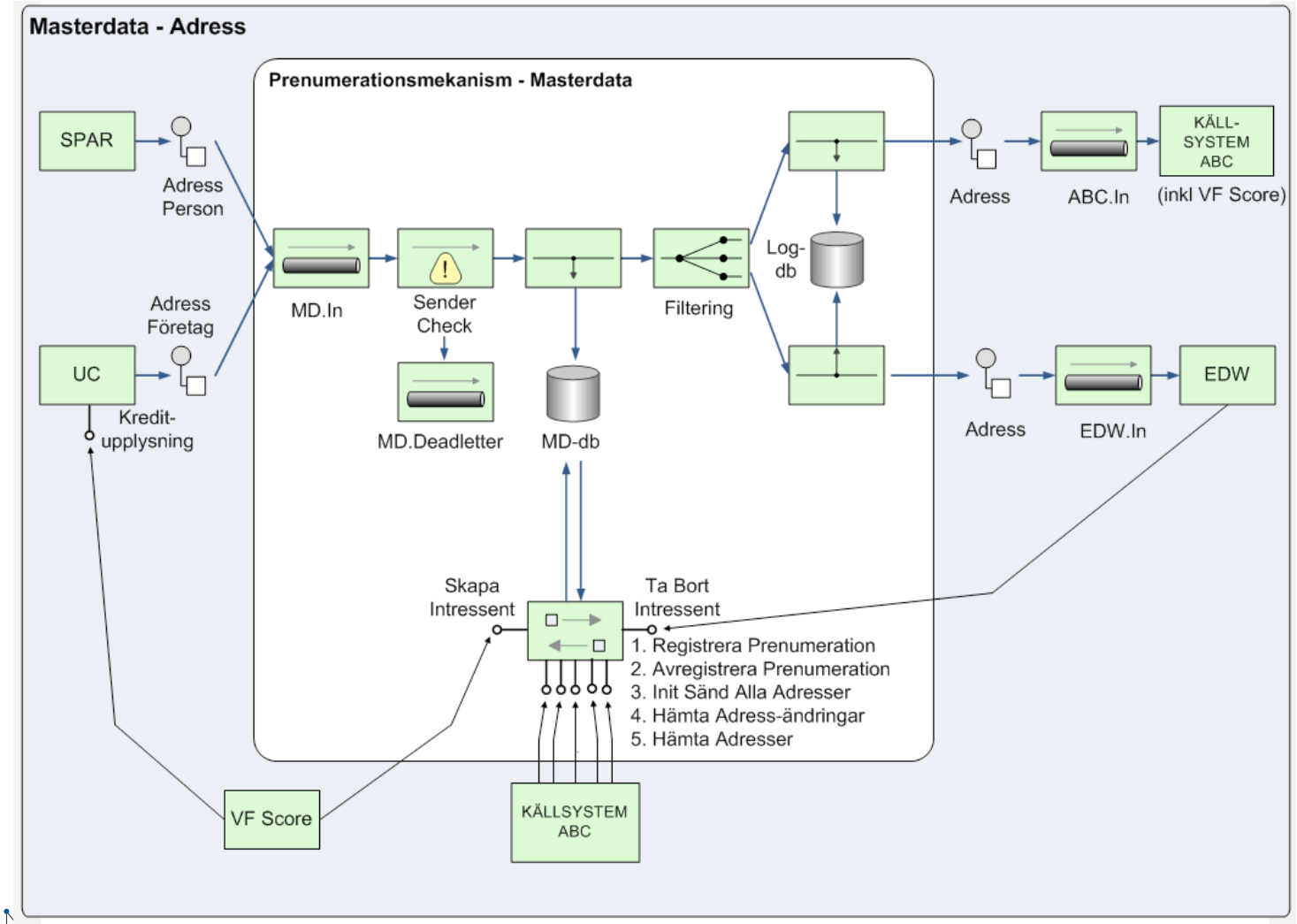
CALLISTA Enterprise

Proof of Concepts

- Masterdata
  - Tests both many integration patterns and protocols
    - Synchronous and asynchronous
    - Store and forward, publish/subscribe and request/reply
    - JMS, JDBC, SOAP/HTTP, REST/HTTP
    - Routing, transformation, logging
- Large file transfer
  - Transfer files over 1GB without blowing the heap or adding complexity in terms of split and aggregation of files
    - Streaming capabilities key feature
  - Strong requirements on security as well

CALLISTA Enterprise

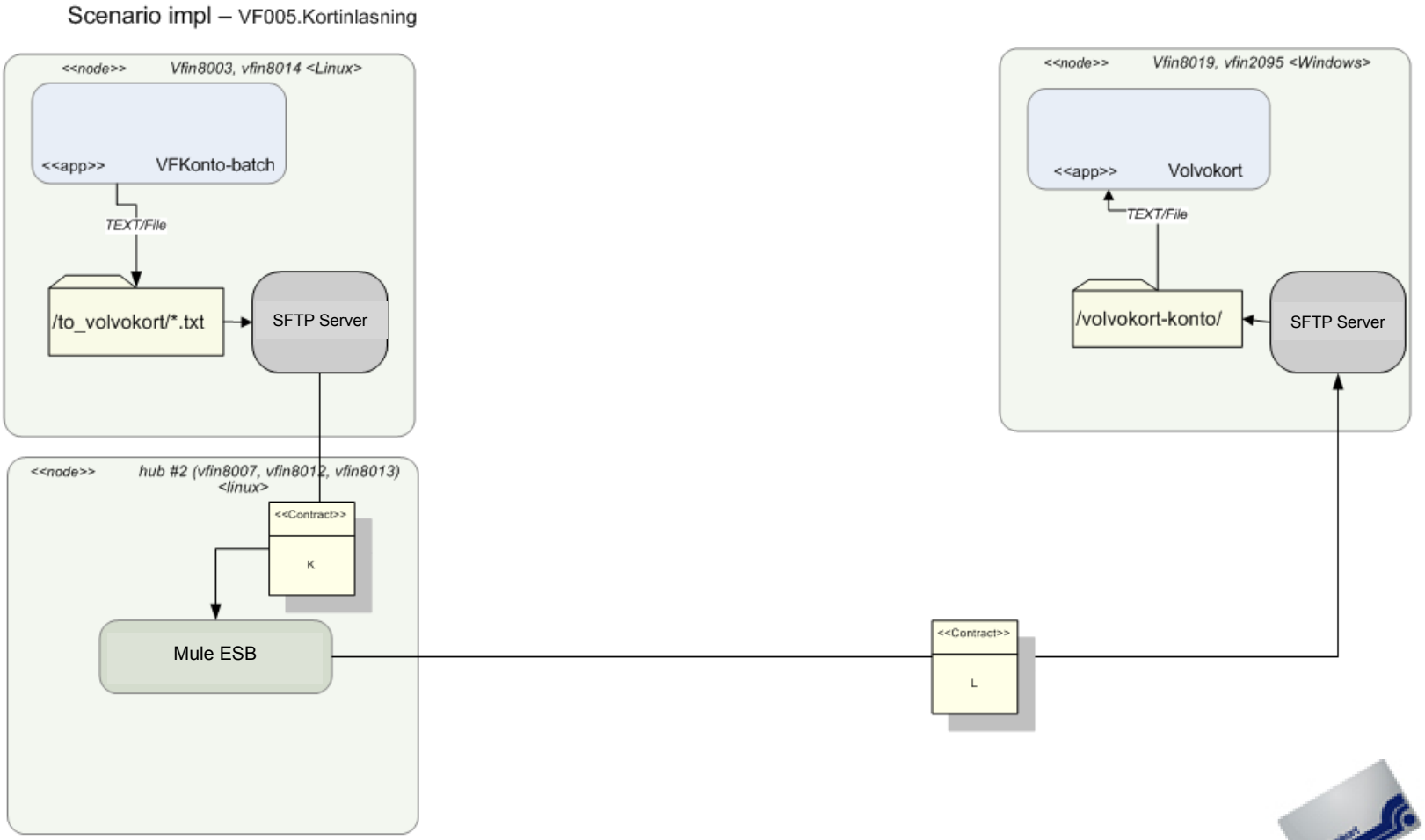
Proof of Concepts - Masterdata



CALLISTA Enterprise

VOLVOFINANS

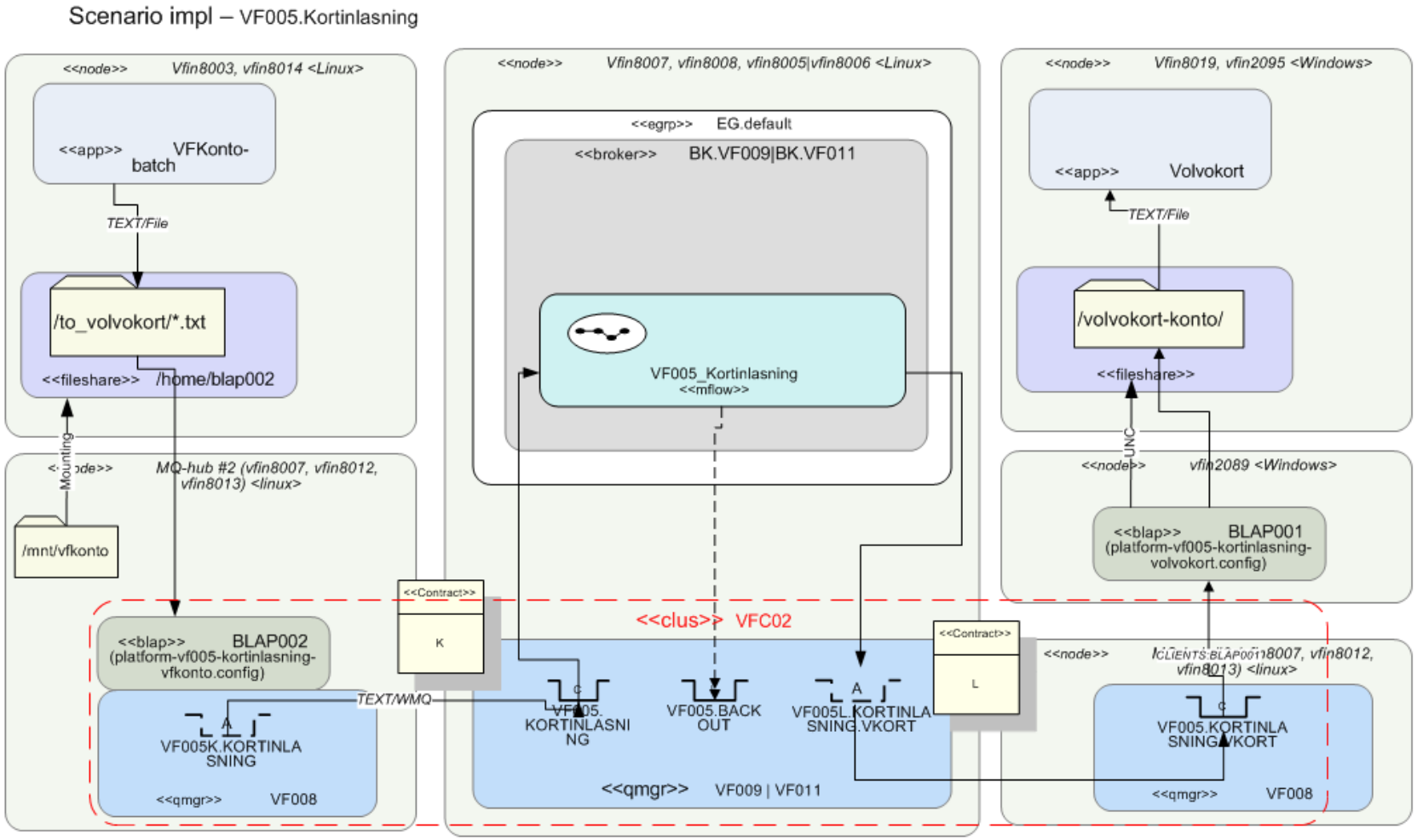
Proof of Concepts – File Transfer



CALLISTA Enterprise

VOLVOFINANS

To be compared with the existing solution…



CALLISTA Enterprise

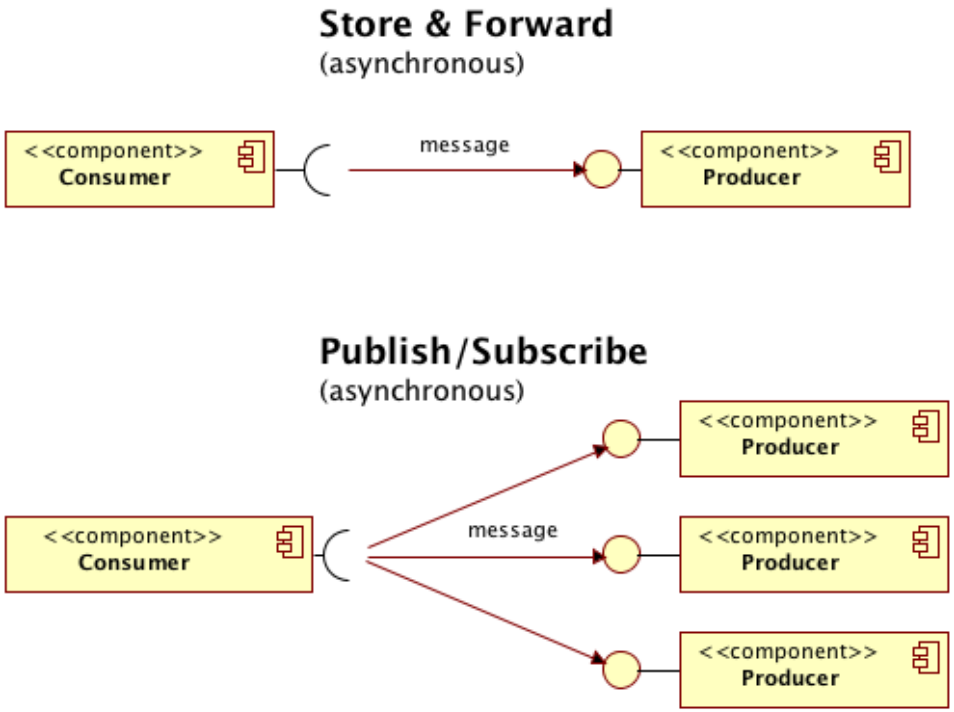
VOLVOFINANS

Proof of Concepts - Output

- Output
  - Mule ESB proved to be able to complement the existing commercial integration platform with the missing functionality and with a substantially reduced complexity and cost
  - Possible to categorize integration requirements in a set of integration patterns
    - Establish an Integration Pattern Catalog!
      - Will potentially simplify decisions and implementations
  - Decision
    - Establish platform and initiate pilot projects

CALLISTA Enterprise

Proof of Concepts – Output - Integration Patterns



CALLISTA Enterprise

VOLVOFINANS

Where are we?

- The customer
- Phases
  - Problem definition
  - Proof of concepts
  - Establish platform
  - Migration of system landscape
- Summary

CALLISTA Enterprise

Establish platform - activities

- Create
  - Integration Pattern Catalog
  - Reference implementations
  - Documentation
    - Template for describing requirements on an integration
    - Guiding documentation
      - Policies, guidelines and detailed instructions
    - Describing the system landscape
      - Baseline, target and roadmaps
  - Frameworks
    - Custom extensions/mechanisms to Mule ESB
      - Error handling, logging, protocol conversion
  - Tools
    - Custom tool for creating a new integration component
      - Structure, naming conventions, maven dependencies
  - Integration Method
    - Puts all the artifacts above into place

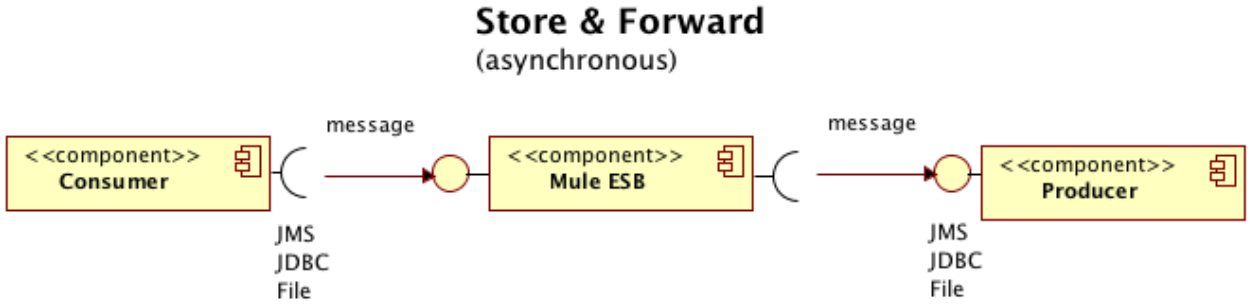
CALLISTA Enterprise

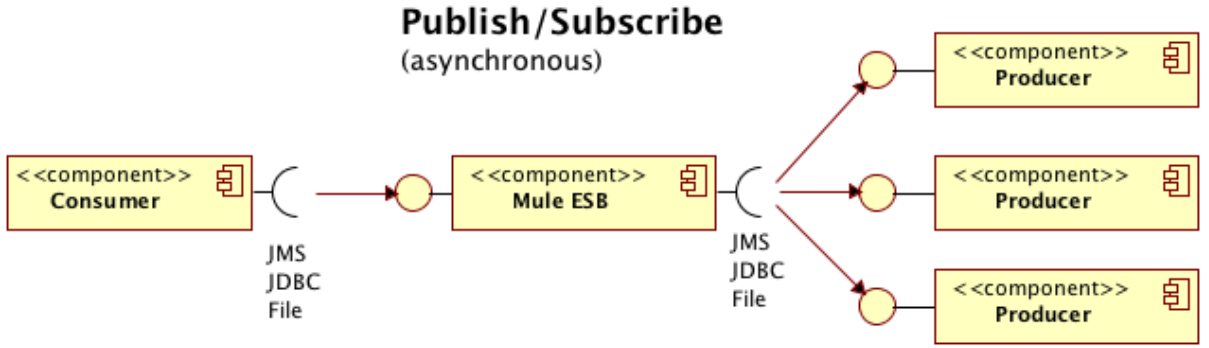
Establish platform - Integration method

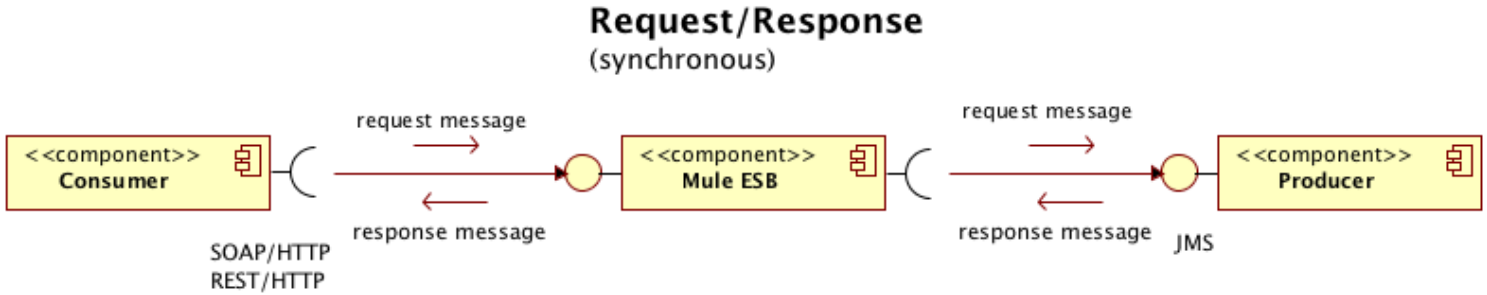
- Integration Method
  - Business Project
    - » Describe business requirements (using the template)
  - IT Governance
    - » Approves the integration (using policies and guidelines)
      - Suggests changes if required
    - » Determine what patterns to use (using the pattern catalog)
    - » Updates the system landscape documentation
  - IT Project
    - » Create new integration component if required
      - Using the custom code generator
    - » Implement integration
      - Using pattern catalog, reference implementations and detailed instructions

CALLISTA Enterprise

Integration Pattern Catalog







CALLISTA Enterprise

VOLVOFINANS

Integration Pattern Catalog

Logical Pattern	Implementation Pattern
Store & Forward	JMS
	JDBC
	SFTP
Publish/Subscribe	JMS
	JDBC
	SFTP
Request/Reply	SOAP/HTTP
	JMS

Implementation Patterns described by

- Name
- Intent
- Motivation
- Structure
- Collaboration
- Consequences
- Reference Implementation
- Detailed Instructions

CALLISTA Enterprise

VOLVOFINANS

Establish platform – Key Question

- Shall all integration be established as full blows services?
  - Or can we in some cases allow point to point integrations? (through the ESB)
- Establish full blows services drives initial cost and time…
- Point to point tends to manifest the stovepipes…

CALLISTA Enterprise

Establish platform – Service or point to point?

- Characteristics of a service
  - + The cornerstone in a service oriented architecture
    - + Enabler of reuse, loose coupling, ease of use and so on…
    - – Requires careful design in terms of
      - Canonical Message Formats (non application specific formats)
        - Based on a common information model…
      - Granularity
      - Reuse
      - Composability
      - Ownership
      - Security
    - – Higher initial cost and longer development time
  - + Once reuse and composite services increase cost and time for development is lowered

CALLISTA Enterprise

Establish platform – Service or point to point?

- Characteristics - point to point (through the ESB)
  - Typically
    - Only two parts involved
      - Or a few well defined parts
    - No routing nor transformation required
    - Store and forward pattern
      - File or database protocols
    - Involves large information volumes (e.g. large files)
  - \+ Very simple to implement (low cost, short development time)
  - − Manifest the stovepipes and cause unwanted tight coupling…
    - \+ An ESB in-between prepares for a future evolution into a service

CALLISTA Enterprise

Integration Pattern Catalog - examples

- Examples
  - Store & Forward
    - Point to point (through the ESB)
    - SFTP
  - Request/Reply
    - Service based on CMF
    - SOAP/HTTP Consumers & JMS Providers
    - Handling state change in provider - Idempotent Receiver

CALLISTA Enterprise

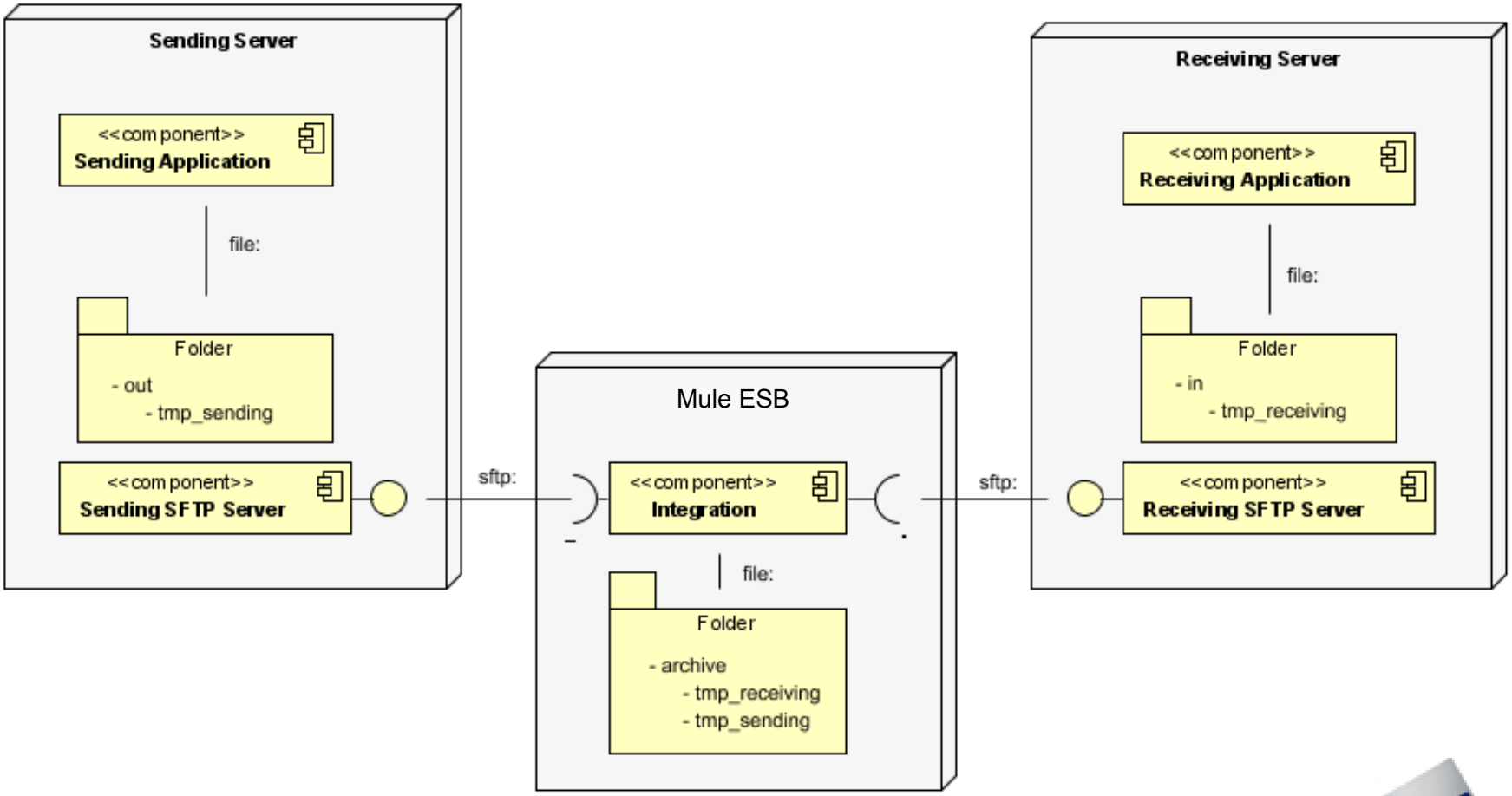
Store & Forward pattern - SFTP

- Secure transfer of files using SFTP (FTP over SSH)
- Identification based on Public Keys (PKI)
  - No need to handle tons of server specific passwords
  - Only one public key to distribute
- ESB responsible for the file transfer
  - Applications only read and write to local file system
- Mule can stream large files over SFTP

CALLISTA Enterprise

VOLVOFINANS

Store & Forward pattern - SFTP

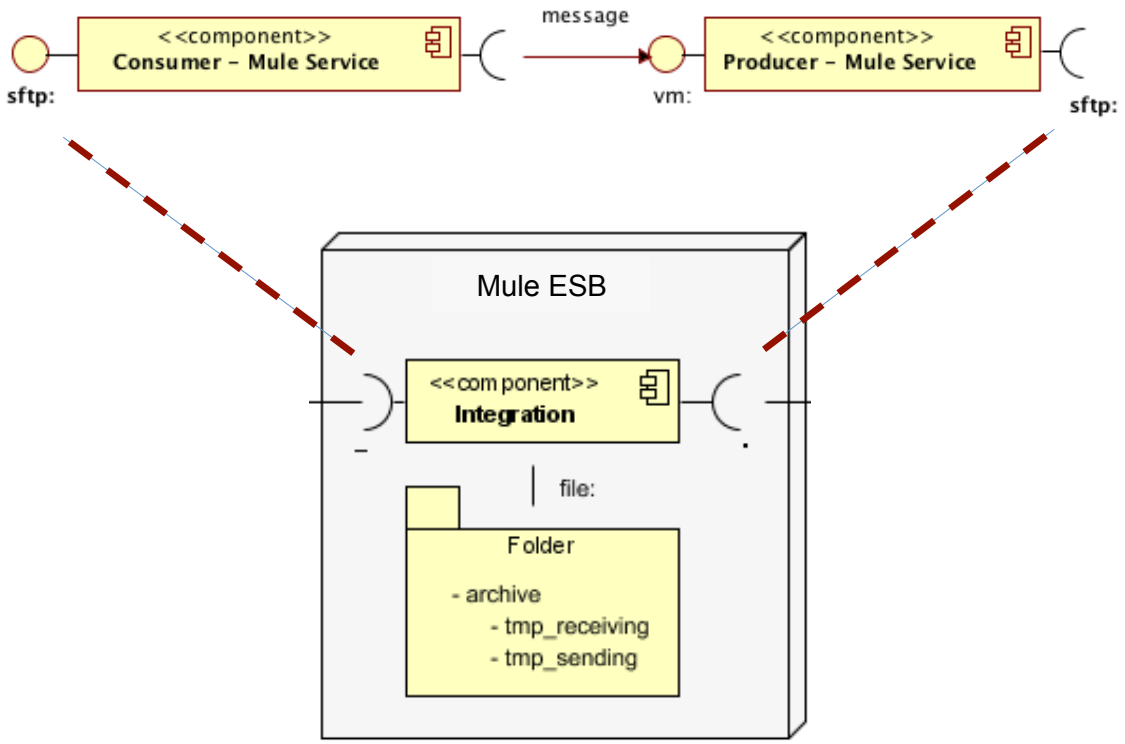


CALLISTA Enterprise

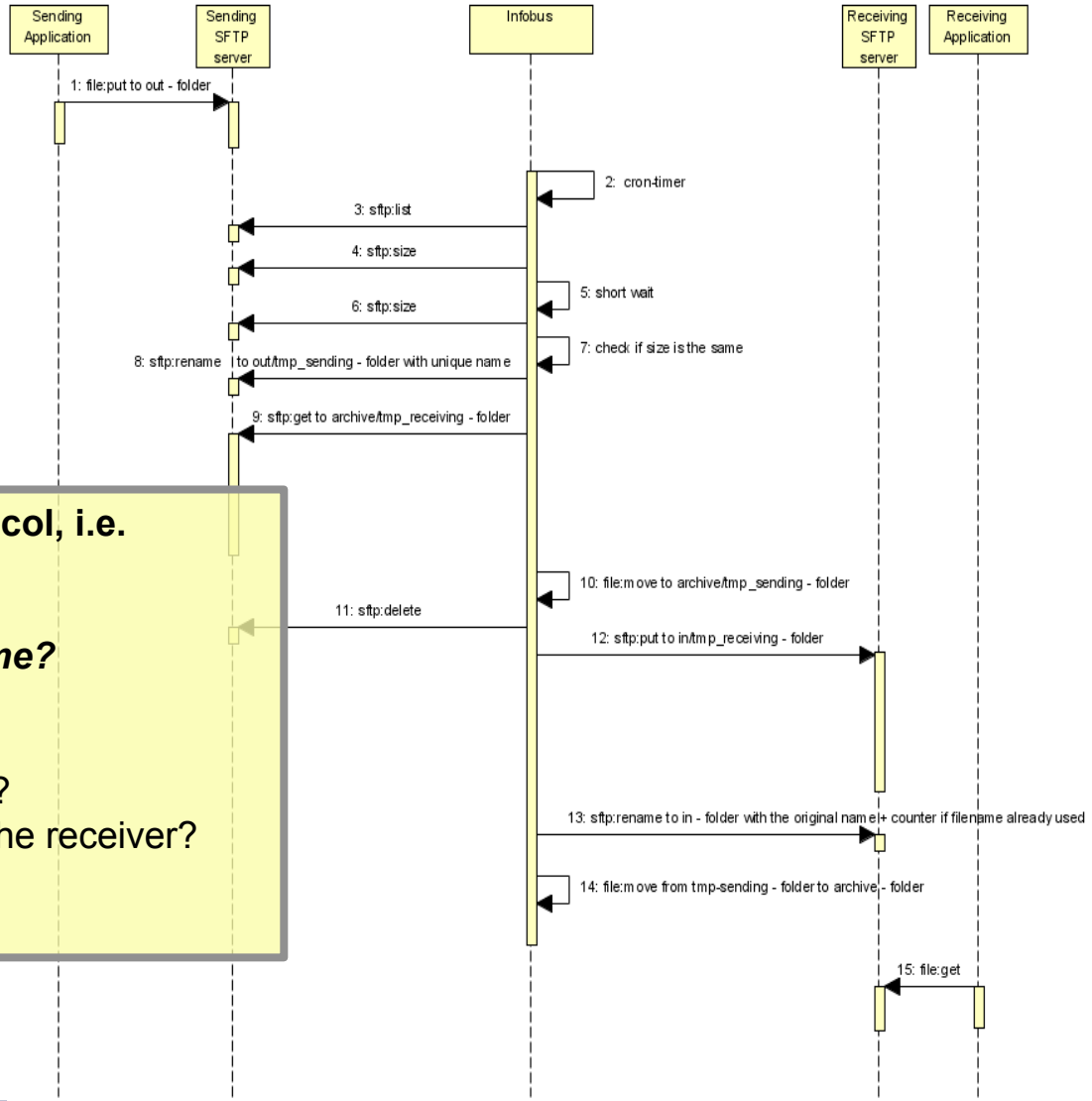
VOLVOFINANS

Store & Forward pattern - SFTP

- Are we building stovepipes now?
  - I.e. with tight coupling between the consumer and the provider?
- No, inside Mule ESB consumer and producer services are separated
  - Each side can be replaced transparently



Store & Forward pattern - SFTP



Robust usage of the SFTP protocol, i.e.

how to guarantee that a file is transferred one and only one time?

- Is the input file complete?
- How to avoid duplicate receiving?
- What if the filename is in use at the receiver?
- Archive-function?
- How to recover from crash?

CALLISTA Enterprise

Store & Forward pattern – SFTP – code example

- Declaring a sftp-connector
  - Using PKI keys for authentication
  - Custom error handler for error logging

```
<sftp:connector
  name="sftp-connector"
  identityFile="${SFTP_IDENTITYFILE}"
  passphrase="${SFTP_IDENTITYFILE_PASSPHRASE}">

  <custom-exception-strategy class="se.volvofinans.commons.mule.ExceptionListener"/>
</sftp:connector>
```

- Declaring a sftp-event-listener for logging

```
<spring:bean
  name="sftpTransportNotificationLogger"
  class="se.volvofinans.commons.mule.SftpTransportNotificationListenerImpl"/>

<notifications>
  <notification-listener ref="sftpTransportNotificationLogger"/>
</notifications>
```

CALLISTA Enterprise

VOLVOFINANS

Store & Forward pattern – SFTP – code example

- Consuming a sftp-file

```
<sftp:inbound-endpoint
  address="sftp://${VFKONTO_USERNAME}@${VFKONTO_HOST}${VFKONTO_FOLDER}"
  pollingFrequency="${VFKONTO_POLLING_MS}”
  tempDir=”sending”
  useTempFileTimestampSuffix="true"
  archiveDir="${ARCHIVE_FOLDER}”>

  <file:filename-wildcard-filter pattern="${VFKONTO_FILENAME_FILTER}"/>
</sftp:inbound-endpoint>
```

- Producing a sftp-file

```
<sftp:outbound-endpoint
  address="sftp://${VK_USERNAME}@${VK_HOST}${VK_FOLDER}"
  outputPattern="#[ORIGINALNAME]"
  tempDir="receiving”
  duplicateHandling="addSeqNo"/>
```

CALLISTA Enterprise

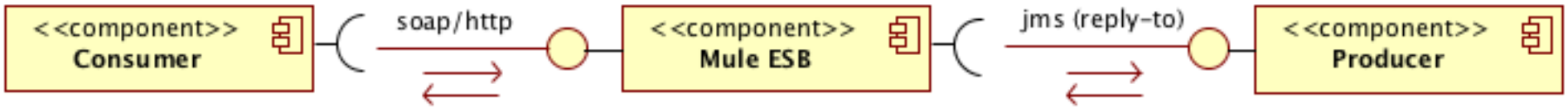
VOLVOFINANS

Integration Pattern Catalog - examples

- Examples
  - Store & Forward
    - Point to point (through the ESB)
    - SFTP
  - Request/Reply
    - Service based on CMF
    - SOAP/HTTP Consumers & JMS Providers
    - Handling state change in provider - Idempotent Receiver

CALLISTA Enterprise

Request/Reply pattern – SOAP/HTTP ←→ JMS



- Requirements on lightweight internal and external consumers
  - Support SOAP/HTTP Web Services on consumer side
- Producers prefer reliable protocols such as JMS (i.e. guaranteed delivery)
- ESB responsible for the protocol transformation
  - Synchronous ←→ Asynchronous
  - Error handling
    - » Jms Reply message with error info → SOAP Fault

CALLISTA Enterprise

VOLVOFINANS

Request/Reply pattern – SOAP/HTTP ←→ JMS

- Synchronous SOAP/HTTP inbound endpoint

```
<service name="getkontoinformation">

  <inbound>
    <cxf:inbound-endpoint
      address="${GETKONTOINFORMATION_URL}"
      wsdlLocation="classpath:/schemas/sd/konto/v1/GetKontoinformationService.wsdl"
      serviceName="GetKontoinformationService"
      namespace="urn:se.volvofinans.sd.konto.getkontoinformation.wsdl:v1"
      proxy="true"
      synchronous="true"
      responseTransformer-refs="createSoapFaultIfException"
    />
  </inbound>
```

Message definitions based on CMF's via classpath

Error handling, creation of SOAP faults

CALLISTA Enterprise

VOLVOFINANS

Request/Reply pattern – SOAP/HTTP ←→ JMS

- Asynchronous JMS outbound endpoint

```
<outbound>
  <pass-through-router>
    <outbound-endpoint
      address="jms://${GETKONTOINFORMATION_REQUEST_QUEUE}"
      transformer-refs="objectToJmsMsg"
      synchronous="true"/>
      <reply-to address="jms://${GETKONTOINFORMATION_REPLY_QUEUE}" />
  </pass-through-router>
</outbound>

<async-reply timeout="${TIMEOUT}">
  <inbound-endpoint
    address="jms://${GETKONTOINFORMATION_REPLY_QUEUE}"
    transformer-refs="jmsMsgToObject"/>
  <single-async-reply-router/>
</async-reply>

</service>
```

Adds reply-to header to the jms message

Waits for an asynch. reply jms mesage

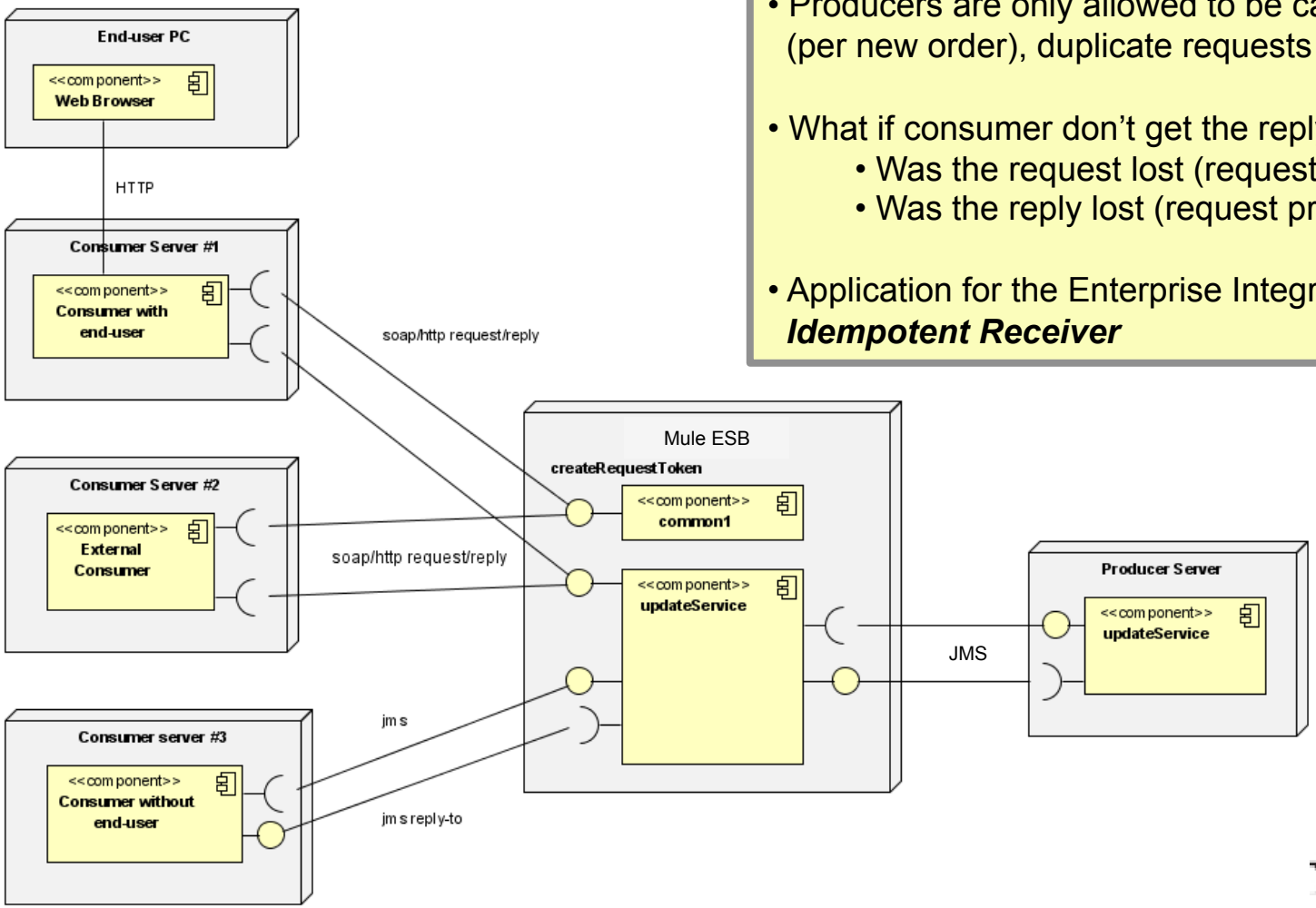
CALLISTA Enterprise

VOLVOFINANS

Request/Reply pattern – HTTP and state change

Unreliable protocols and producers that change state
(e.g. `createOrder()` that return the new orderNumber)

- Producers are only allowed to be called once (per new order), duplicate requests must discarded
- What if consumer don't get the reply?
  - Was the request lost (request not processed)?
  - Was the reply lost (request processed)?
- Application for the Enterprise Integration Pattern ***Idempotent Receiver***



If we had more time…

- …we could have discussed development aspects such as
  - Building a standard integration including a test in minutes
    - Using skeleton generator + detailed instructions
    - Extending the base functionality to meet more advanced requirements
  - Test
    - Testing integrations "*out of container*"
      - I.e. not require deploy before test
    - Unit testing asynchronous mechanisms is tricky
      - Use of Mule's Event Notifications and Doug Lea's java.util.concurrent.CountDownLatch makes it much easier!
  - Handling of Canonical Message Formats (CMF) and common XML Schemas
    - CMF's are a key to success for service oriented integration!
    - Use of XML Schema Catalog to be able to load from classpath

CALLISTA Enterprise

If we had more time…

- …and runtime aspects such as
  - Logging
    - Listen to Mule events (info + error) and log using a custom log4j jms appenders
  - Monitoring and management
    - Through Mule's JMX api
  - Deployment
    - Using Tcat server
      Tomcat v6 + management features packaged by MuleSoft

CALLISTA Enterprise

Establish platform - output

- Output
  - We are ready to go live!
  - A new platform based on Mule ESB is in place!
    - Easier to use, less complexity, higher flexibility
  - Integration approval process + integration pattern catalog
    - Significantly reduce time for decision making and implementation
  - A strategic decision is taken
    - Replace (i.e. not only complement) the existing commercial integration platform with Mule ESB!

CALLISTA Enterprise

Where are we?

- The customer
- Phases
  - Problem definition
  - Proof of concepts
  - Establish platform
  - Migration of system landscape
- Summary

Migration of system landscape

- Three phases
  1. Support initial projects to deliver business values
     - File transfer projects
     - Service enabling project
     - Masterdata project
  2. Large scale usage
  3. Shutdown of existing integration platform

CALLISTA Enterprise

Where are we?

- The customer
- Phases
  - Problem definition
  - Proof of concepts
  - Establish platform
  - Migration of system landscape
- Summary

CALLISTA Enterprise

Summary

- A new platform based on Mule ESB is in place!
  - Easier to use, less complexity, higher flexibility
    - Simple standard solutions for common requirements
    - Extendable to meet more advanced requirements
  - Shared development environment with application development
    - Easier for developers to work cross applications and integrations
  - Lowered development cost and time

- Integration approval process + integration pattern catalog
  - Reduced time for decision making and implementation

CALLISTA Enterprise

Summary

- Do you already have a commercial integration platforms in place?
  - Then it's prime time to challenge it!
  - Start to complement/replace it with a Open Source ESB **NOW!**
- If not…
  - Start to explore Open Source ESB's **NOW!**

CALLISTA Enterprise

Questions?

CALLISTA
Enterprise