Sandy Liu
IIT Workshop on ESB Technology
Aug 12, 2010

Using Mule Enterprise Service Bus:
A Non-Expert's Introduction

Agenda

- Introduction to Enterprise Service Bus (ESB)
- Mule: a brief overview
- Application: Health Service Virtual Organization (HSVO)
- Lessons Learned & Discussion

App. Integration Challenges

- Transport: http, stdio, xmpp…
- Data Format: byte stream, xml, soap …
- Invocation Style: sync, async, batch…
- Application Lifecycles

Web Service?

- Mmmm…how about Web Service?
  - Yes, it solves the interoperability problem
  - But…it is a point-to-point solution: *n* components requires *n*(n-1)* interfaces for full communication
  - ESB requires only one single interface to the bus for global communication.

ESB or Not?

- Are you integrating 3 or more applications or services?
- Will you need to plug in more applications in the future? (tipping point: 25+ services?)
- Do you need to use more than one type of communication protocol?
- Do you need message routing capabilities such as forking and aggregating message flows, or content-based routing?
- Do you need to publish services for consumption by other applications?

(Source: MuleSoft founder and CTO Ross Mason)

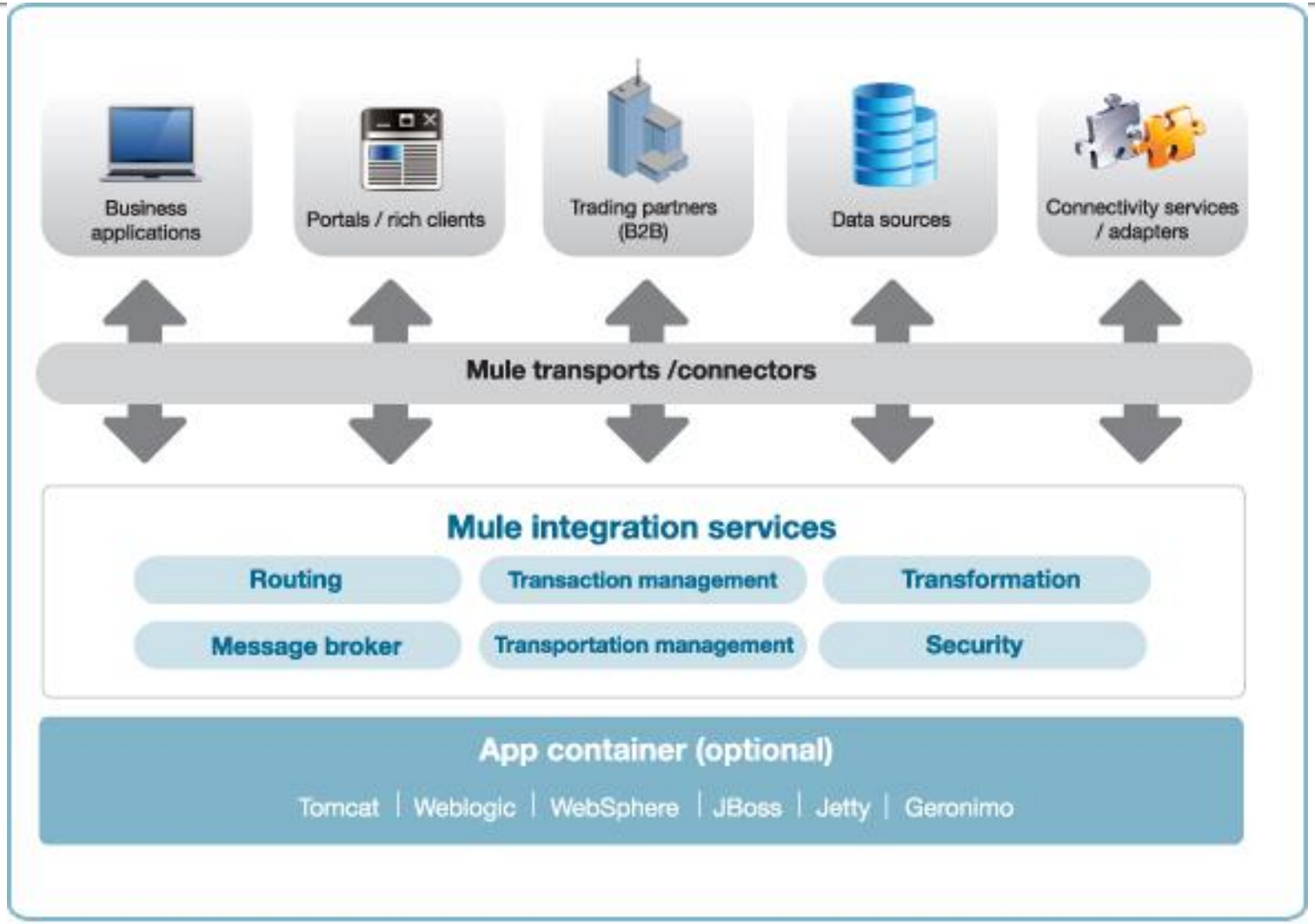
What is an ESB?

- Facilitates application and process integration by providing distributed processing, intelligent routing, security, and dynamic data transformation.
- Provides infrastructure services (middleware) so each application does not have to implement their own solutions independently and in a proprietary manner.
- Encourages loosely-coupled integration

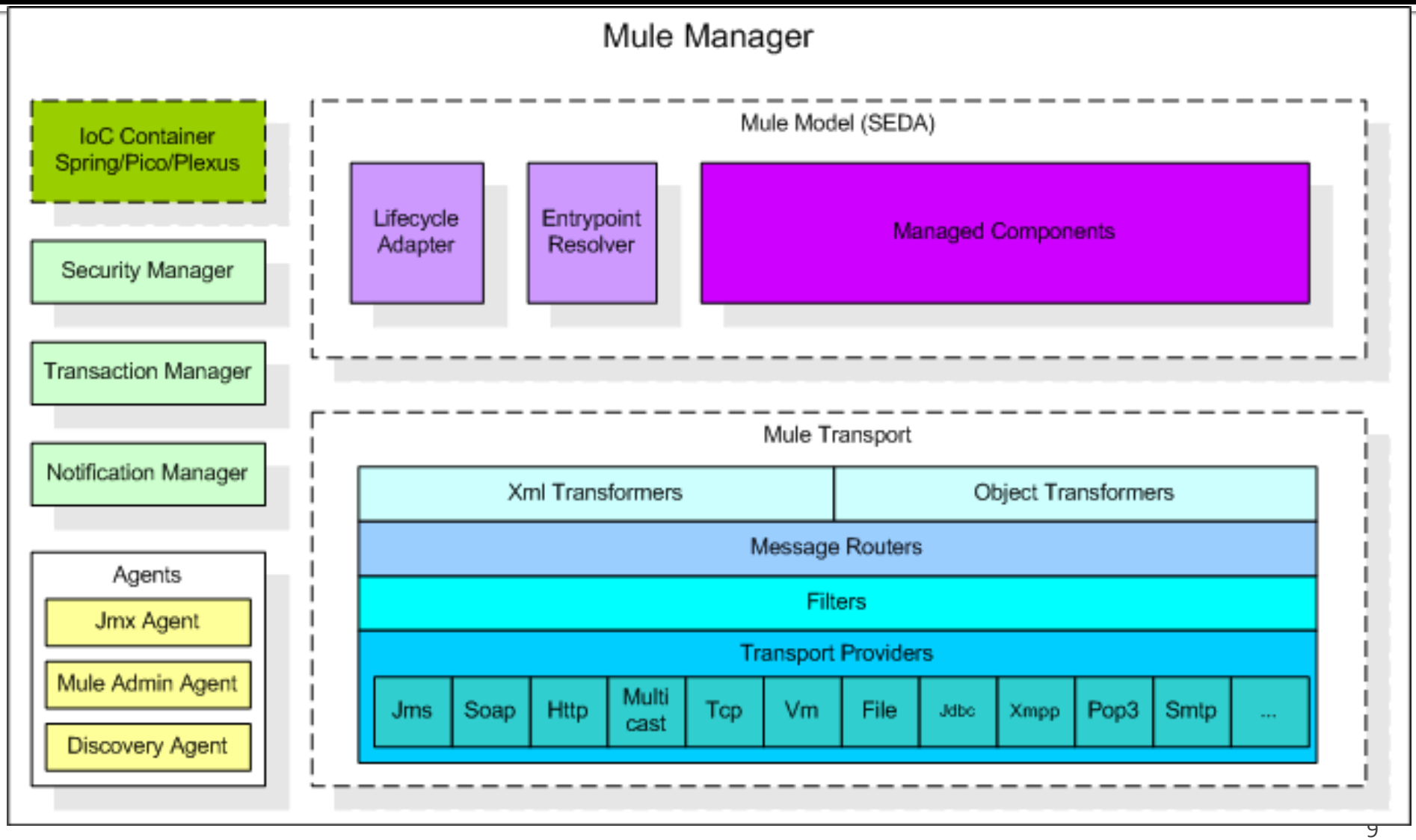
What is Mule?

- An opensource lightweight event-driven ESB
- Written in Java (thus you can have POJO as a service component).
- Supports many open standards
- Supports a big collection of transports
- Utilized Spring for configuration
- Uses the SOA model: modular services
- Easy to install & run
  - "mule start -config hello-config.xml"

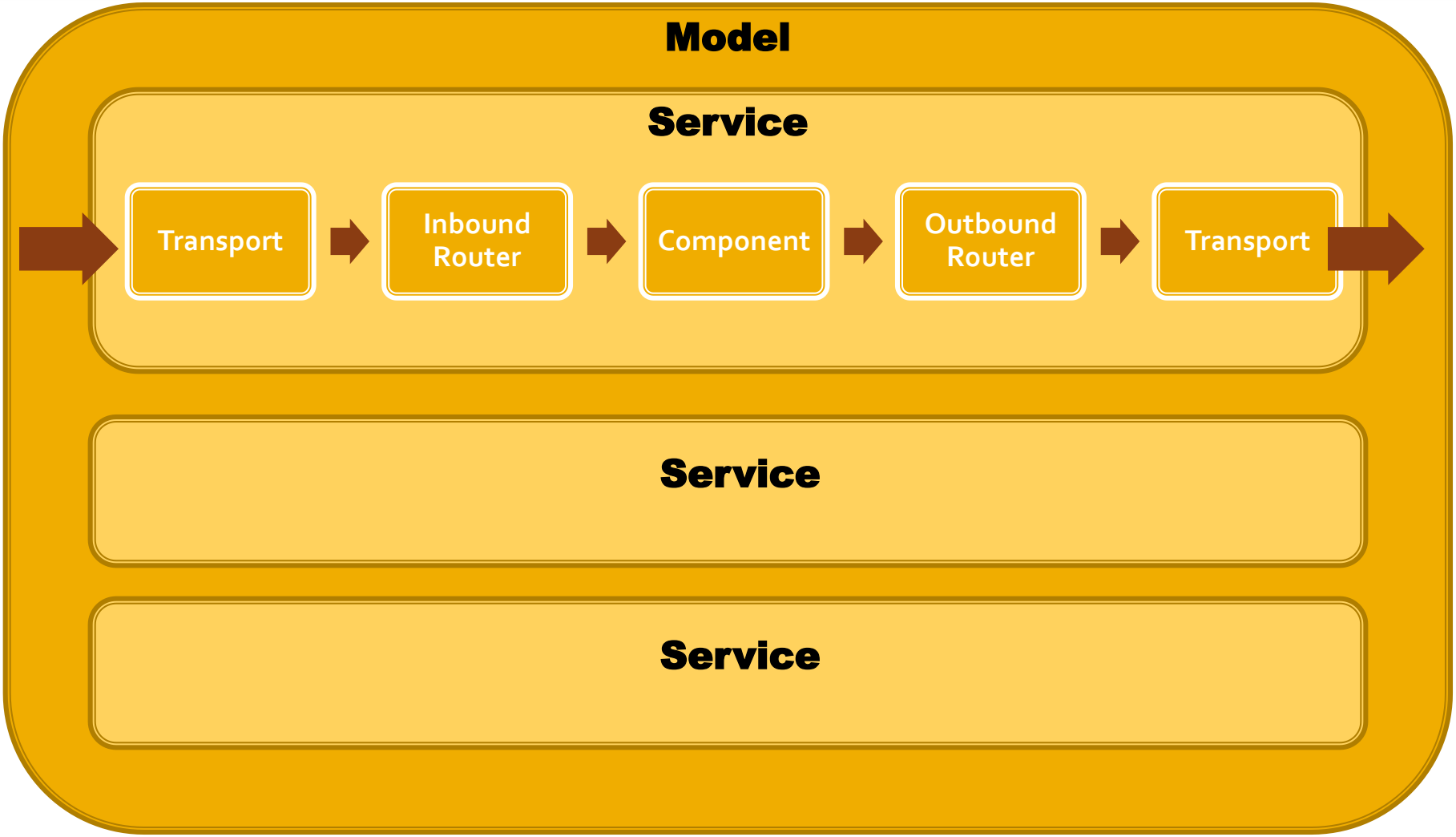
Mule: a high-level view



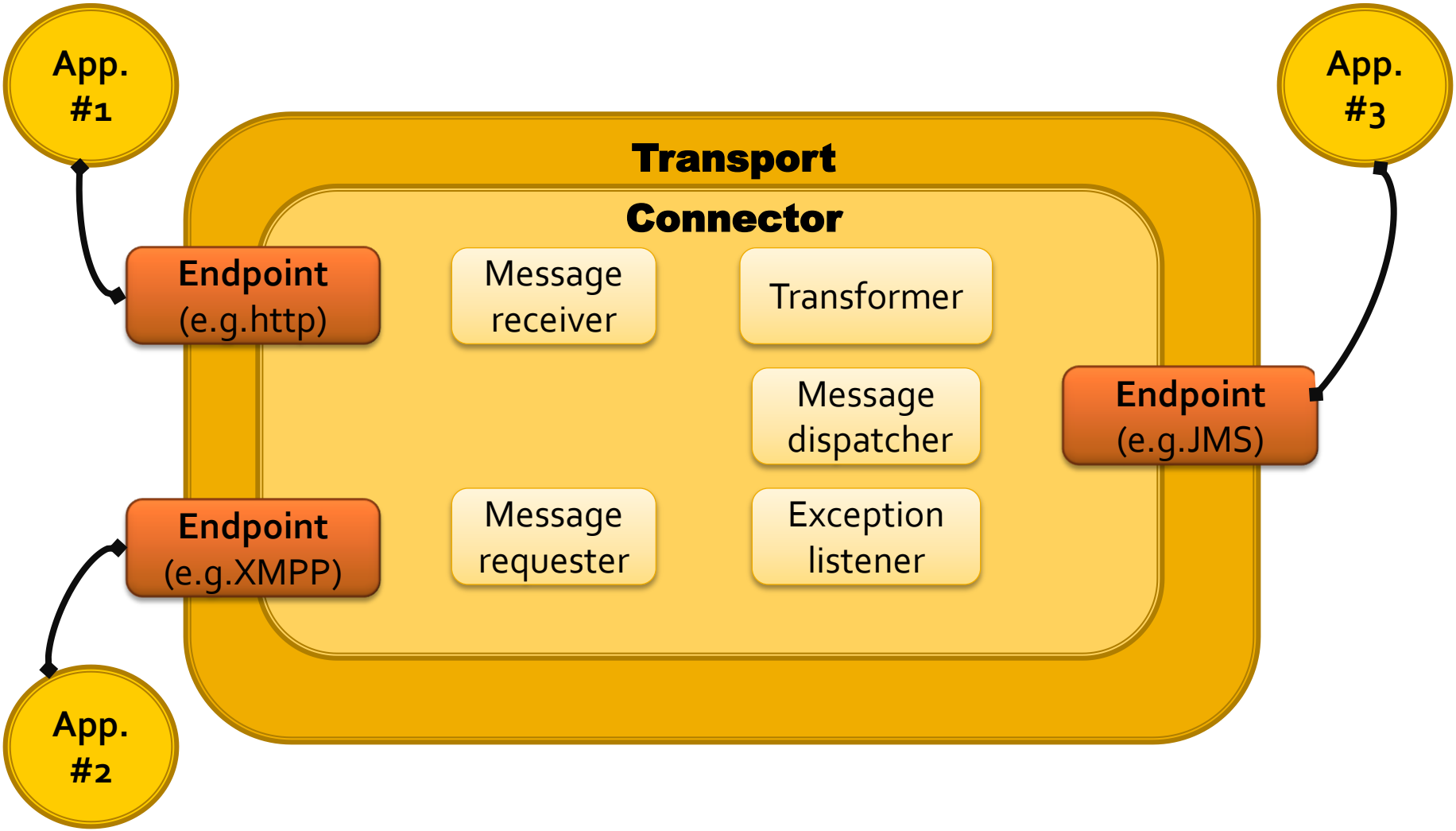
Mule: Technical Components



Model: Runtime Env. For Services

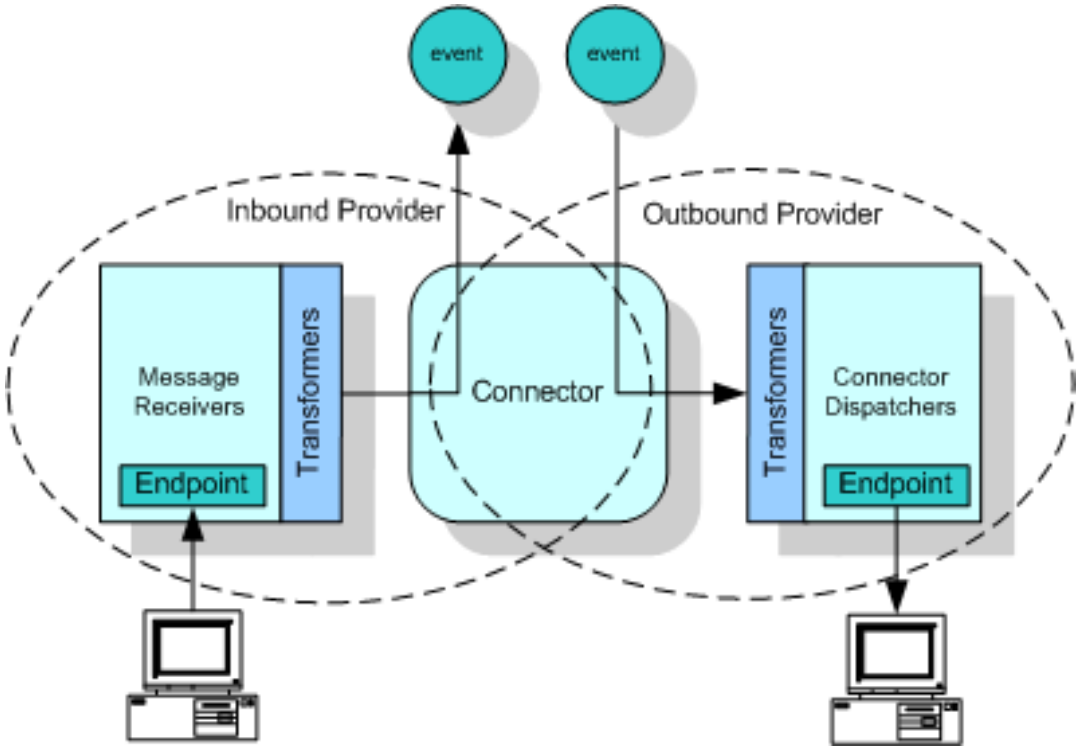


Mule: Transport

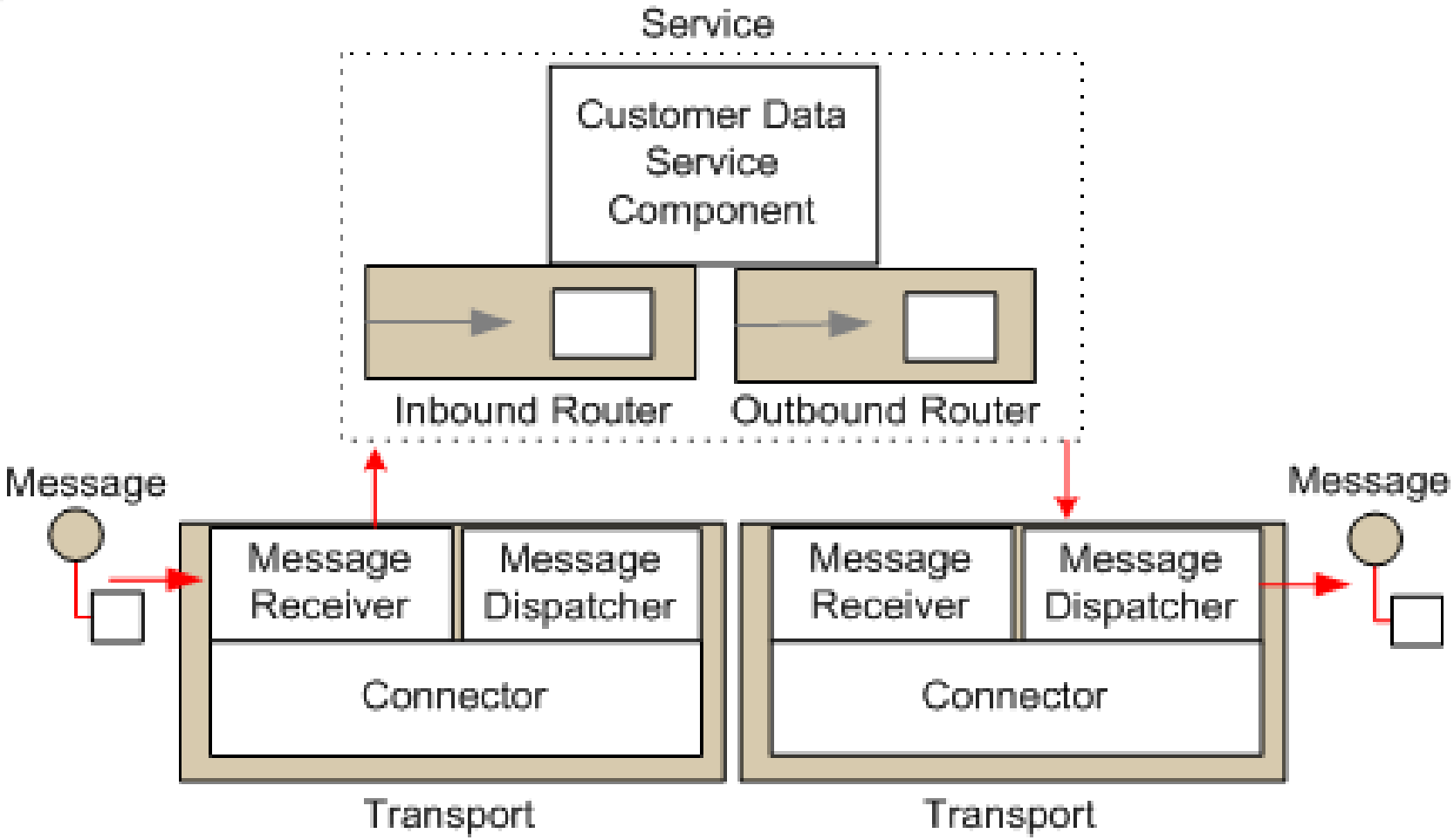


Mule: an Event-driven ESB

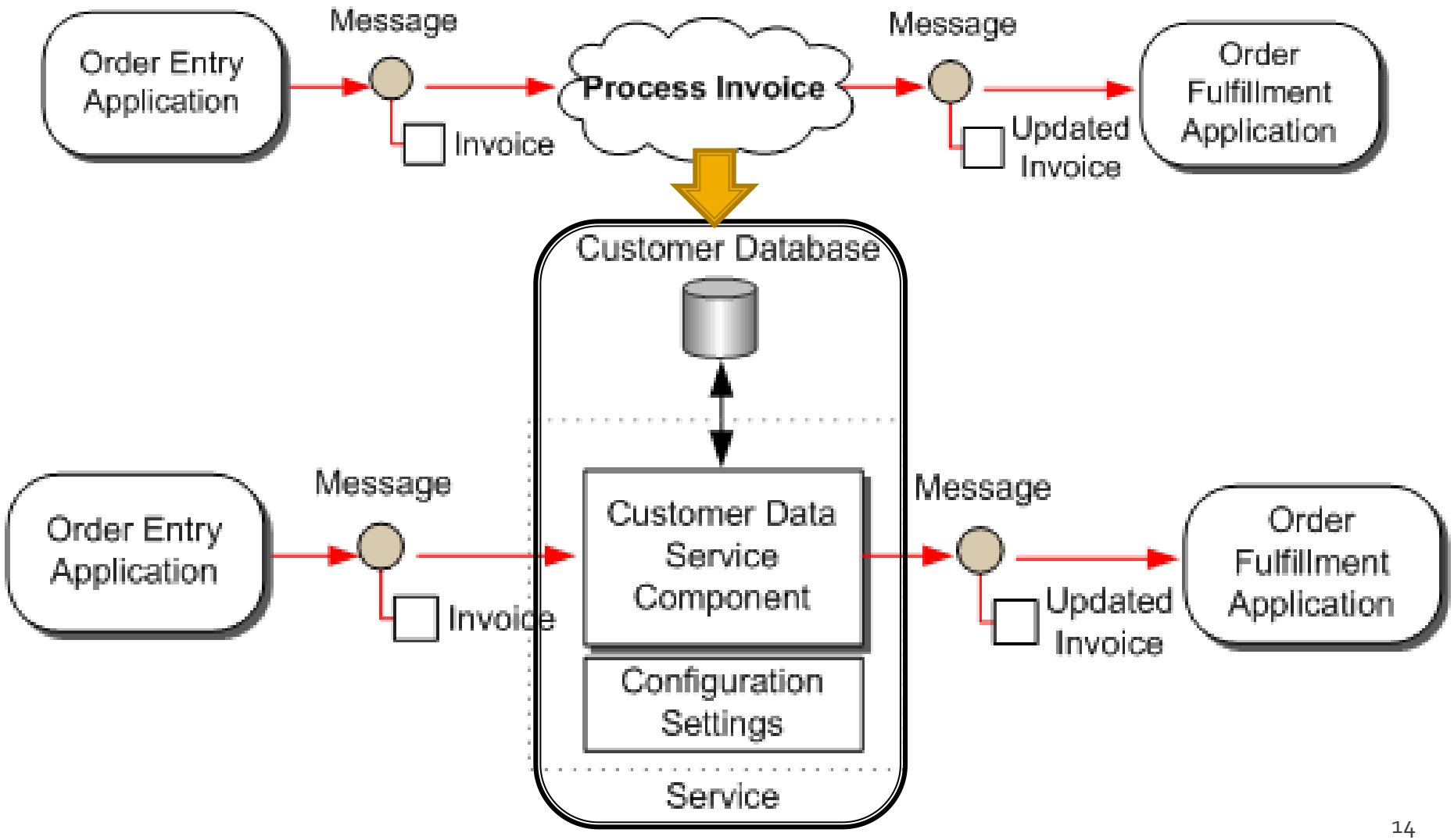
A request is associated with a session object, which carries all the necessary context for the processing of a message while it transits through the service.



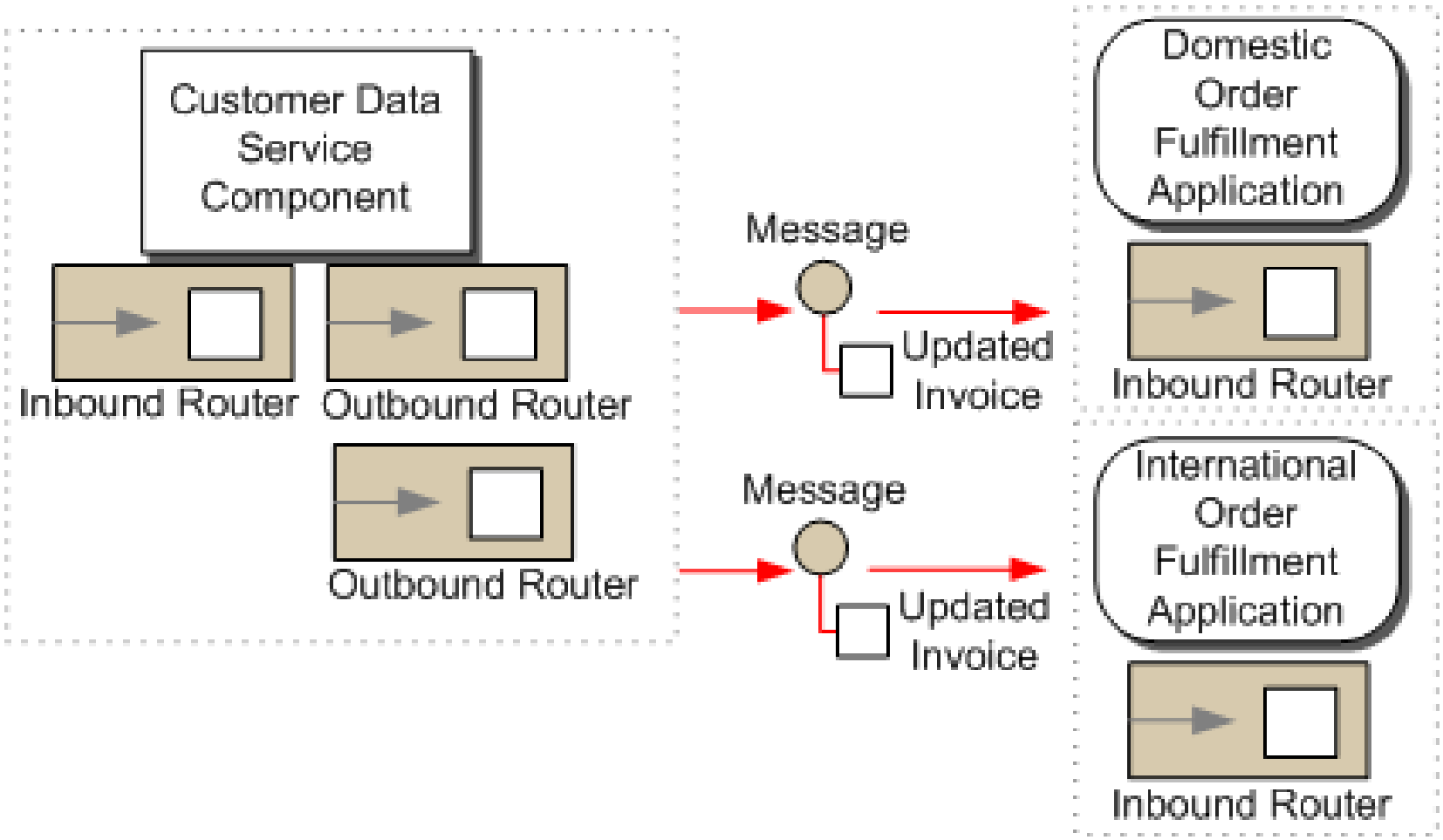
Mule: Architecture



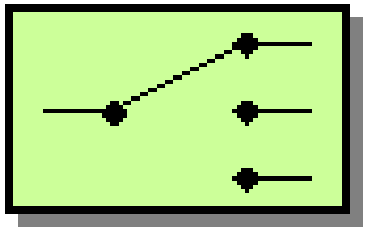
Mule: Processing Data w/ Service



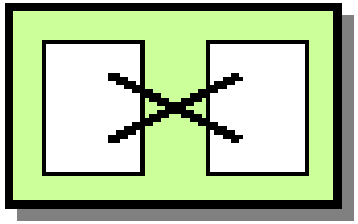
Routing Msg. through Services



Mule: Routers & Transformers



Message Routers are used to control how events are sent and received by components in the system. Mule defines *Inbound routers* that apply to events as they are received and *outbound routers* that are invoked when an event is being dispatched.



Message Transformer transforms message payloads (data) to and from different types.

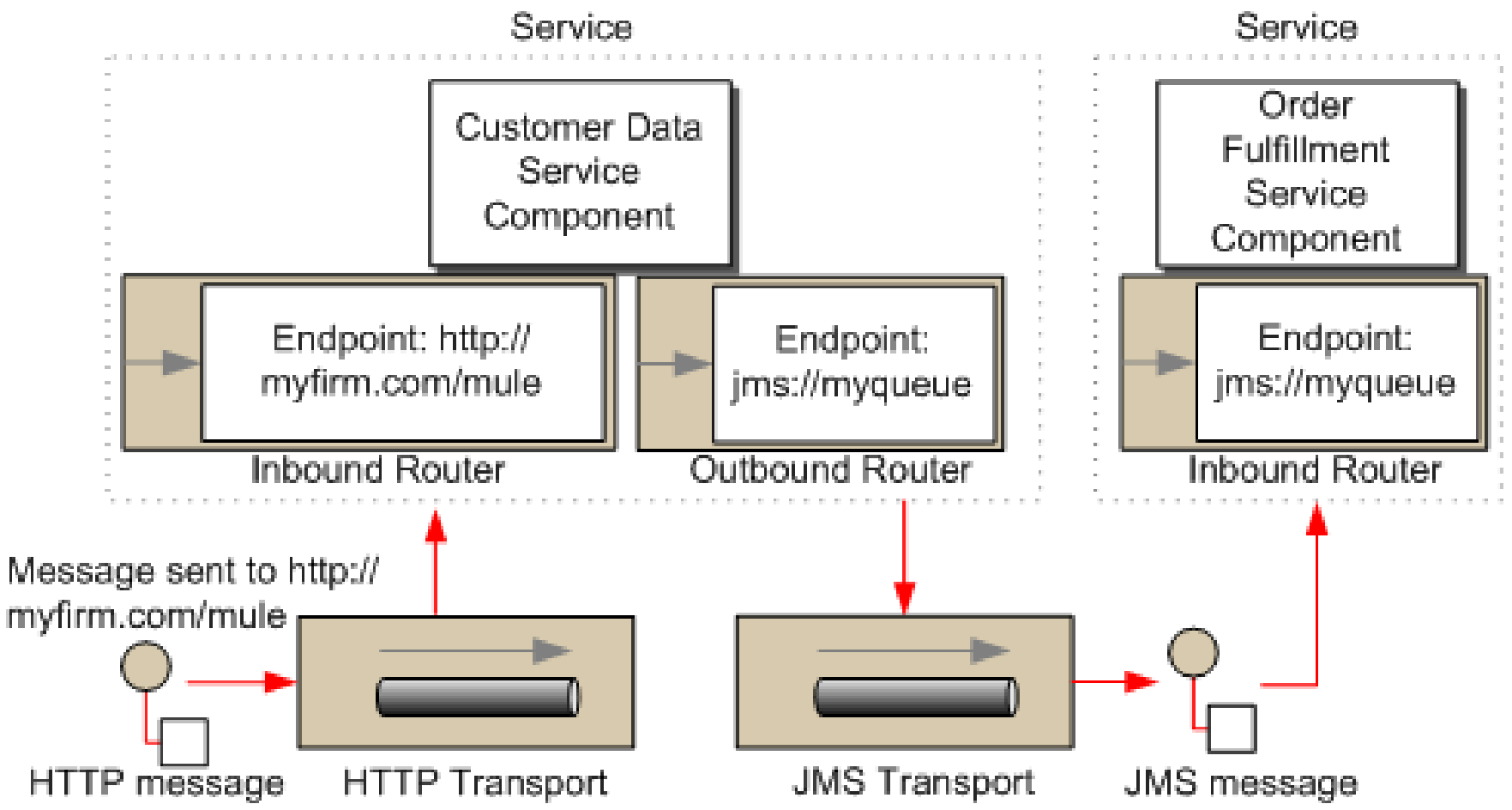
Mule: Available Routers

Inbound Routers	Outbound Routers	Async-Reply Routers	Catch-all Strategies
No Router	Pass-through	Single	Forwarding
Selective Consumer	Filtering	Collection	Custom Fowarding
Idempotent Receiver	Recipient List Routers	Custom	Logging
Idempotent Secure Hash Receiver	Multicasting		Custom
Collection Aggregator	Chaining		
Message Chunking Aggregator	List Message Splitter		
Custom Correlation Aggregator	Filtering XML Message Splitter		
Correlation Resequencer	Expression Splitter Router		
Forwarding	Message Chunking Router		
WireTap	Exception Based Routers		
Custom	Template Endpoint		
	Round Robin Message Splitter		
	Custom		

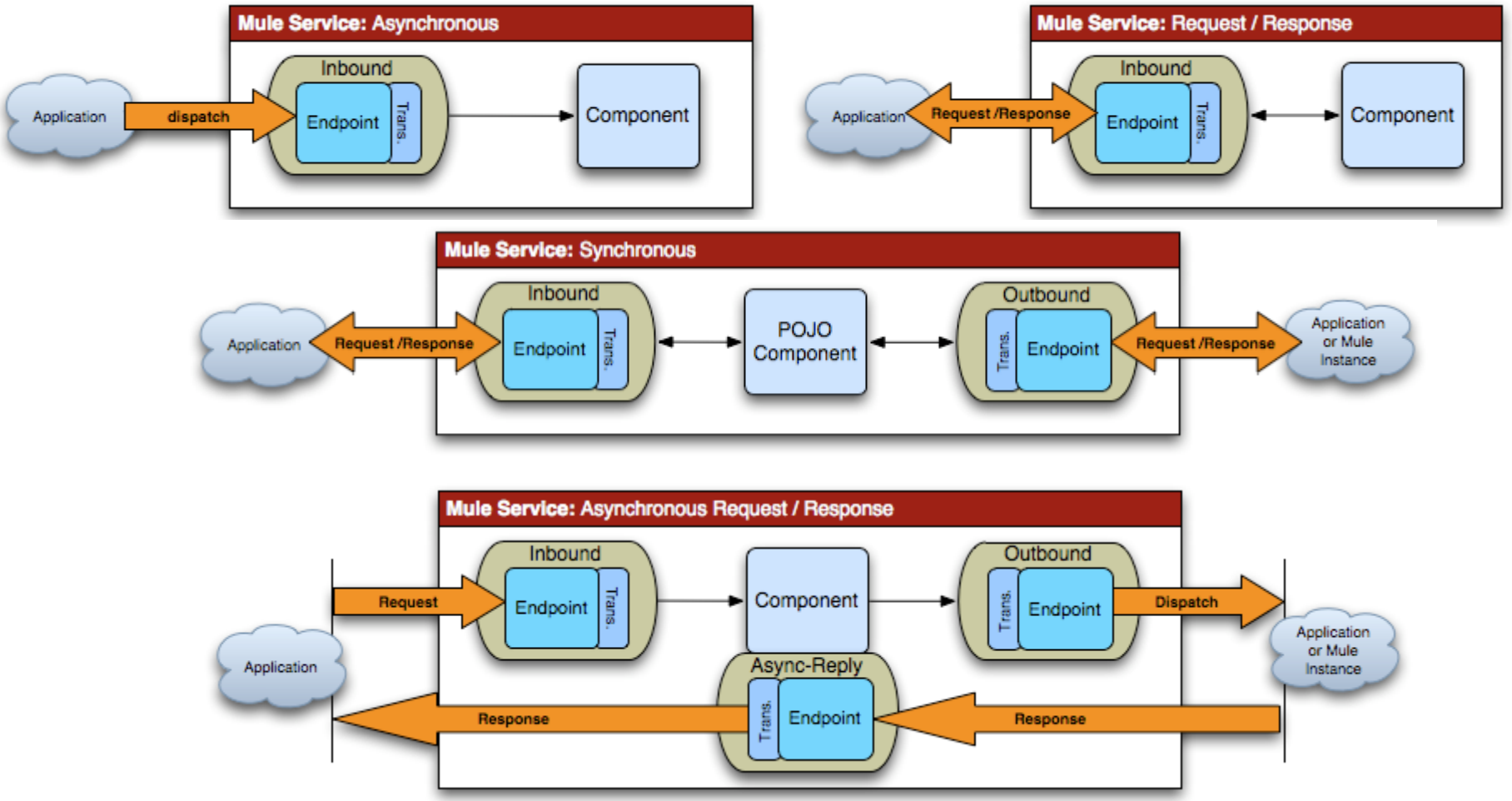
Mule: Available Transformers

- BeanBuilderTransformer
- ByteArrayToHexString
  HexStringToByteArray
- ByteArrayToMuleMessage
  MuleMessageToByteArray
- ByteArrayToObject
  ObjectToByteArray
- ByteArrayToSerializable
  SerializableToByteArray
- ExpressionTransformer
- MessagePropertiesTransformer
- ObjectArrayToString
  StringToObjectArray
- ObjectToInputStream
- ObjectToOutputHandler
- ObjectToString
- StringAppendTransformer
- StringToObjectArray

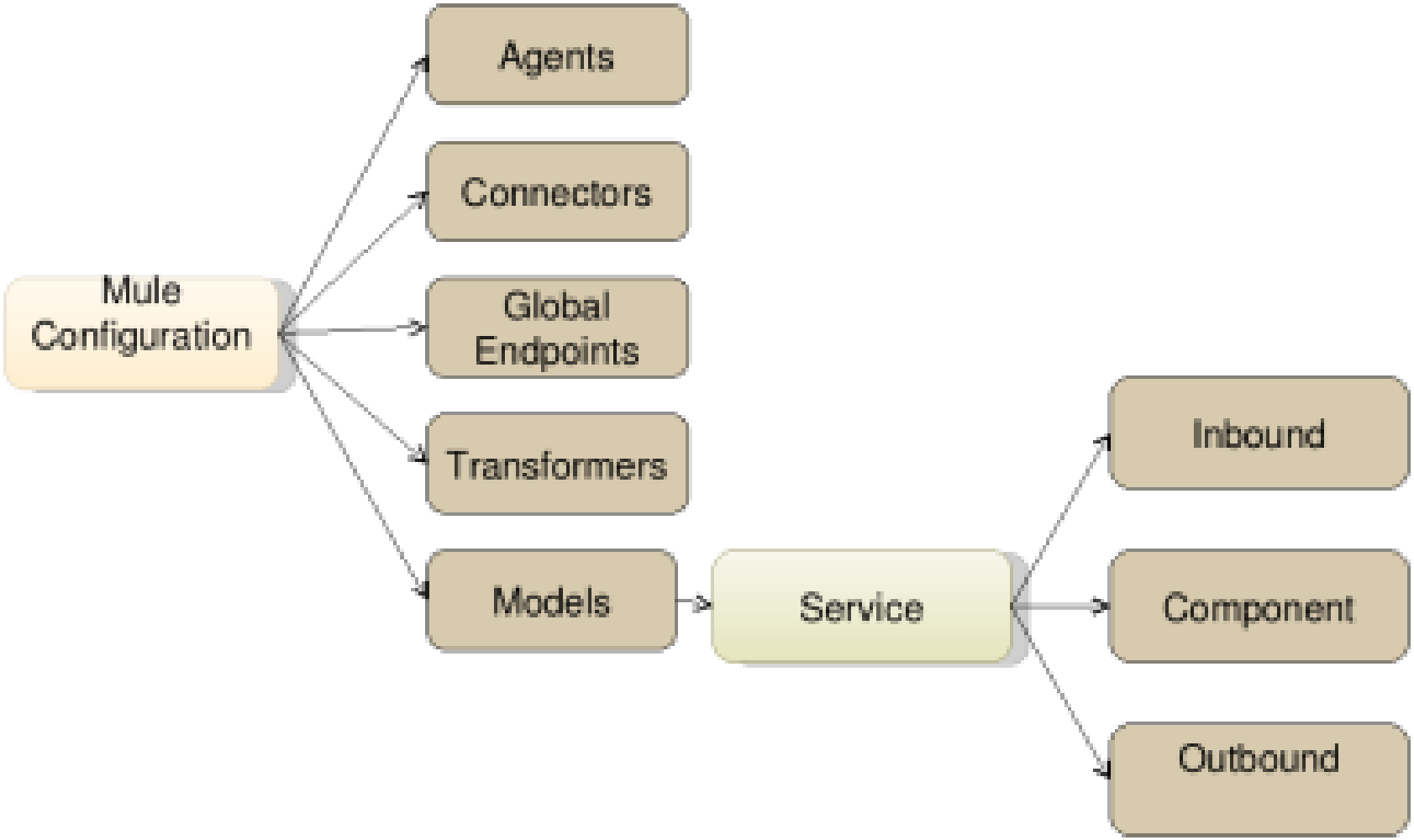
Wiring Everything Together



Mule: Messaging Styles



Mule: Config XML Structure



Config File Snippets

```
<service name="FileToJmsBridge">
        <inbound>
                <file:inbound-endpoint path="/data/in">
                <file:filename-wildcard-filter pattern="*.txt"/>
                        </inbound-endpoint>
        </inbound>
        <!– Optional  to configure a component here -->
        <outbound>
                <pass-through-router>
                <jms:outbound-endpoint
topic="receivedFiles"/>
                </pass-through-router>
        </outbound>
</service>
```

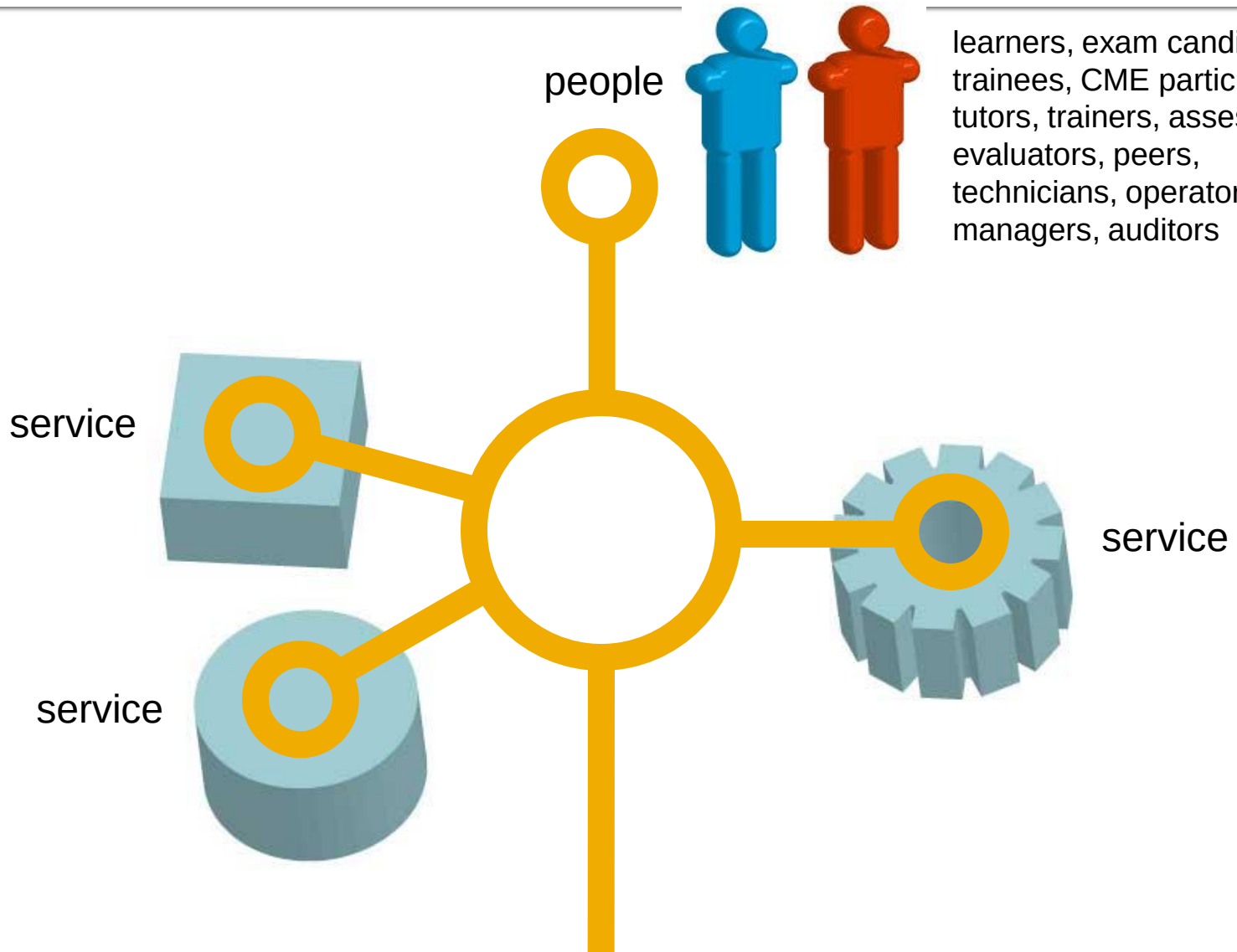
Licensing: CPAL

- Common Public Attribution License
  - A free software license approved by the OSI in 2007.
  - Based on the Mozilla Public License.
  - Non-viral
- Commercial version: the "Enterprise Edition"
  - Community vs. Enterprise Edition: http://www.mulesoft.org/mule-community-vs-mule-enterprise

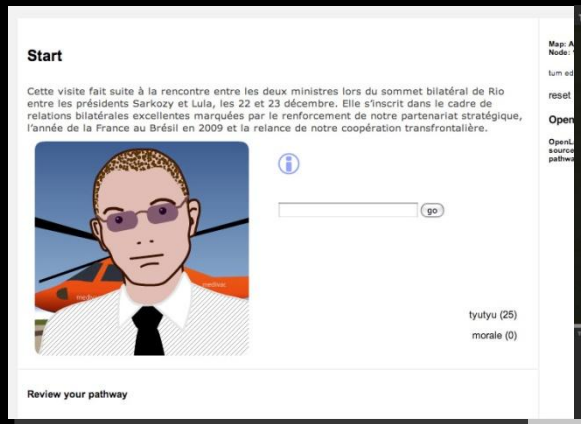
Introduction to HSVO/SAVOIR

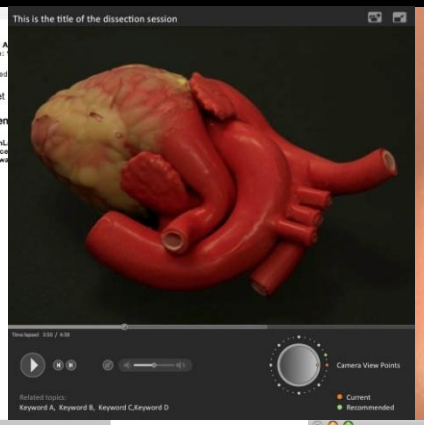
- Health Service Virtual Organization(HSVO) is a CANARIE funded project
- True virtual organization
- Based around a network-enabled platform
- Connect and control devices as services

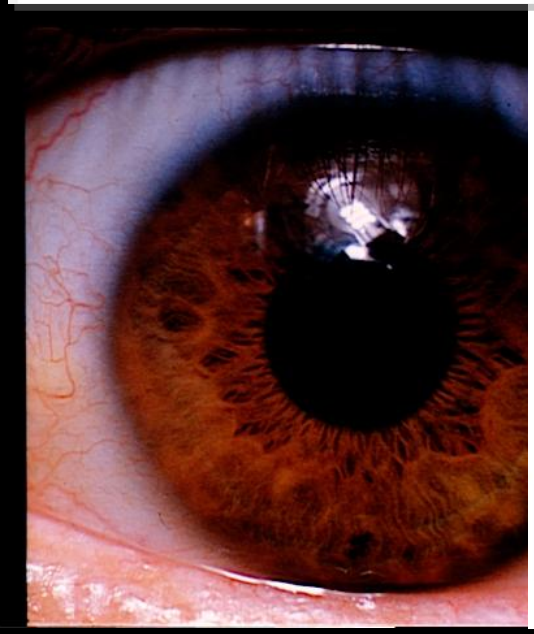
HSVO Connects

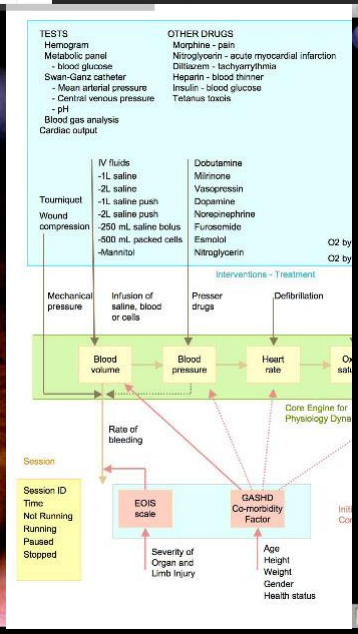


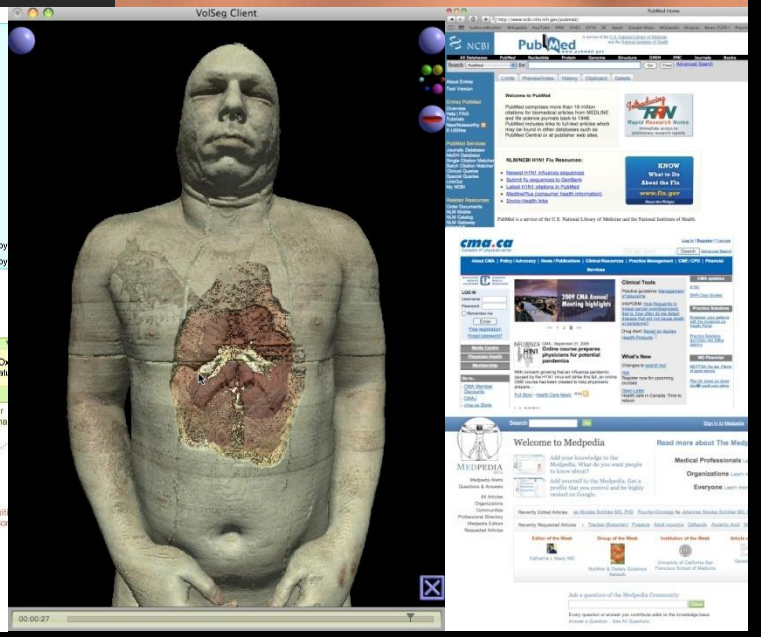
HSVO Services











HSVO

Health Services Virtual Organization

SAVOIR	• Collaboration as a Service
Argia	• Infrastructure as a Service
Tools	• Software as a Service

Northern Ontario School of Medicine

Lakehead University

National Research Council Canada / Conseil national de recherches Canada
Institute for Information Technology / Institut de technologie de l'information

Communications Research Centre Canada / Centre de recherches sur les communications Canada
An Agency of Industry Canada / Un organisme d'Industrie Canada

McGill

Stanford University

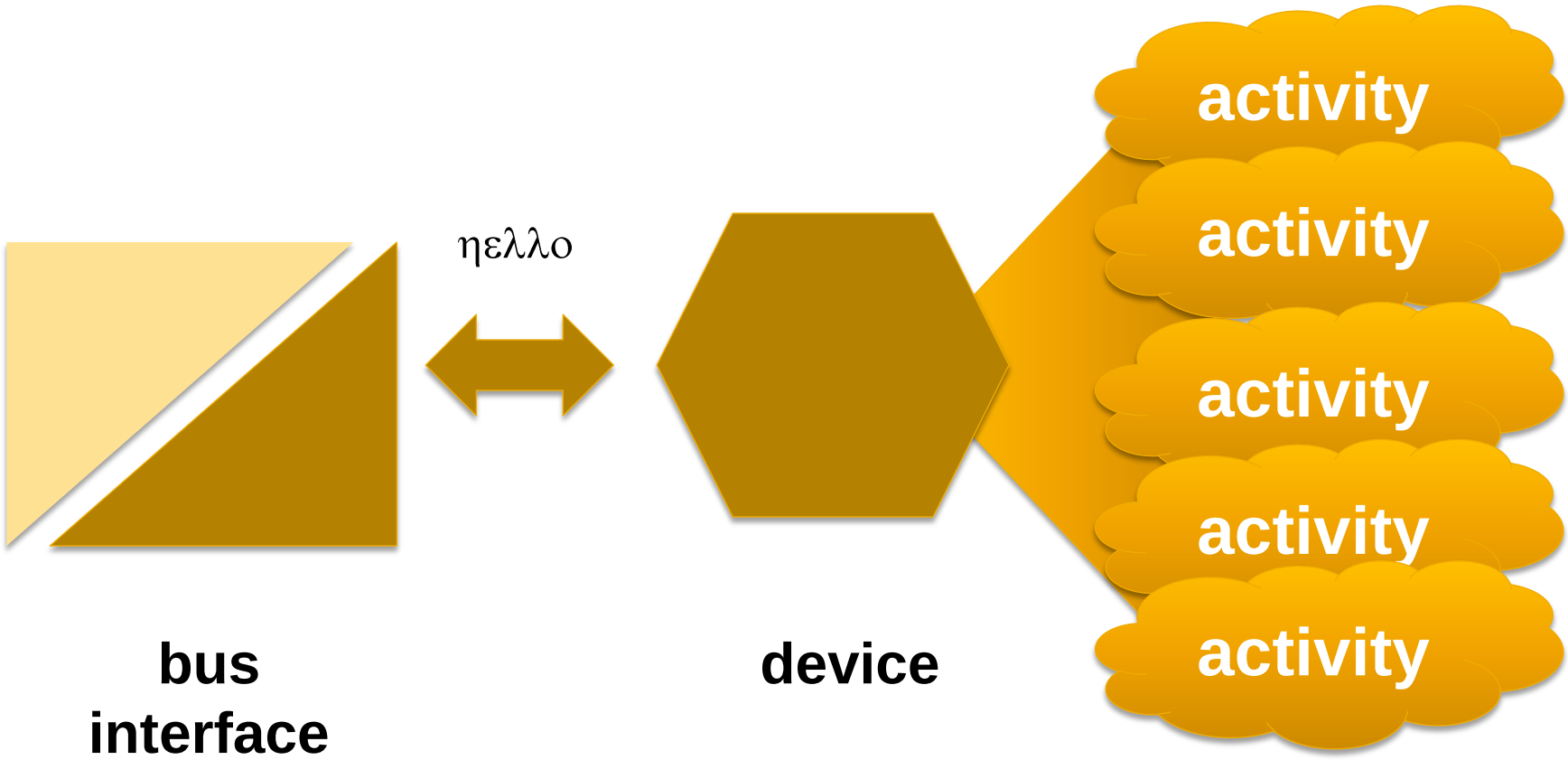
iDEAL

iil

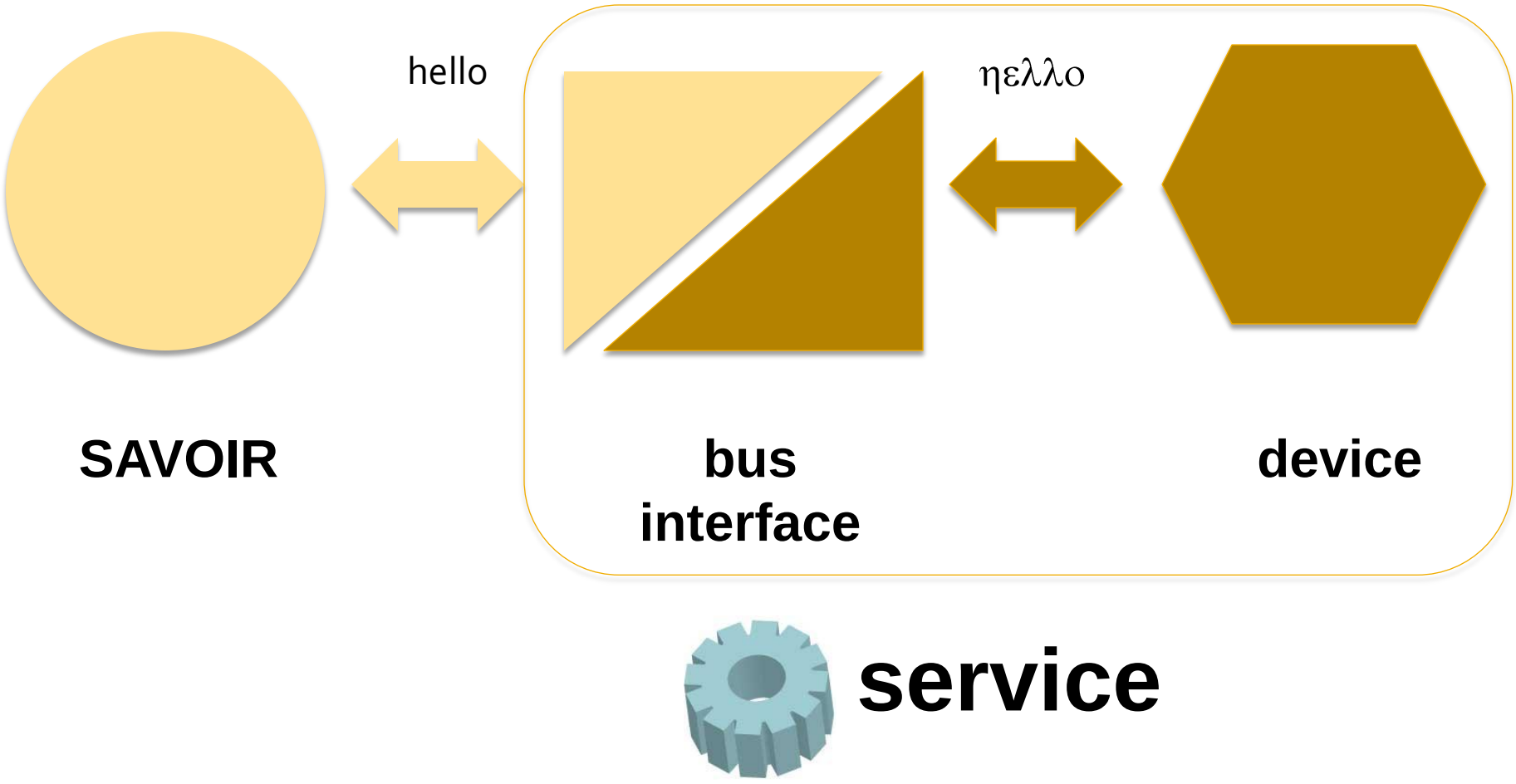
UW-L University of Wisconsin-La Crosse

canarie
Canada's Advanced Research and Innovation Network
Le réseau évolué de recherche et d'innovation du Canada

HSVO Bus Interface

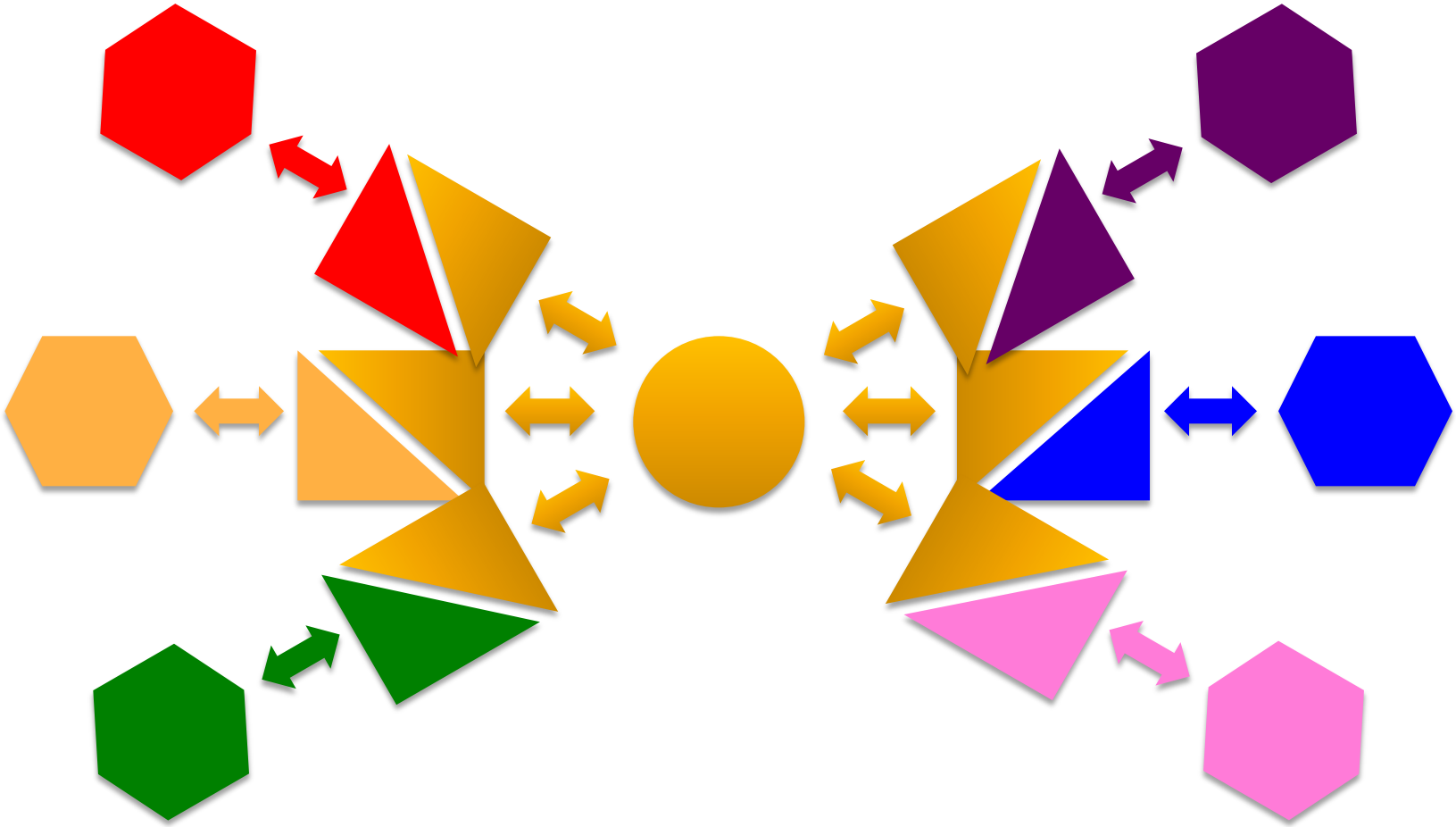


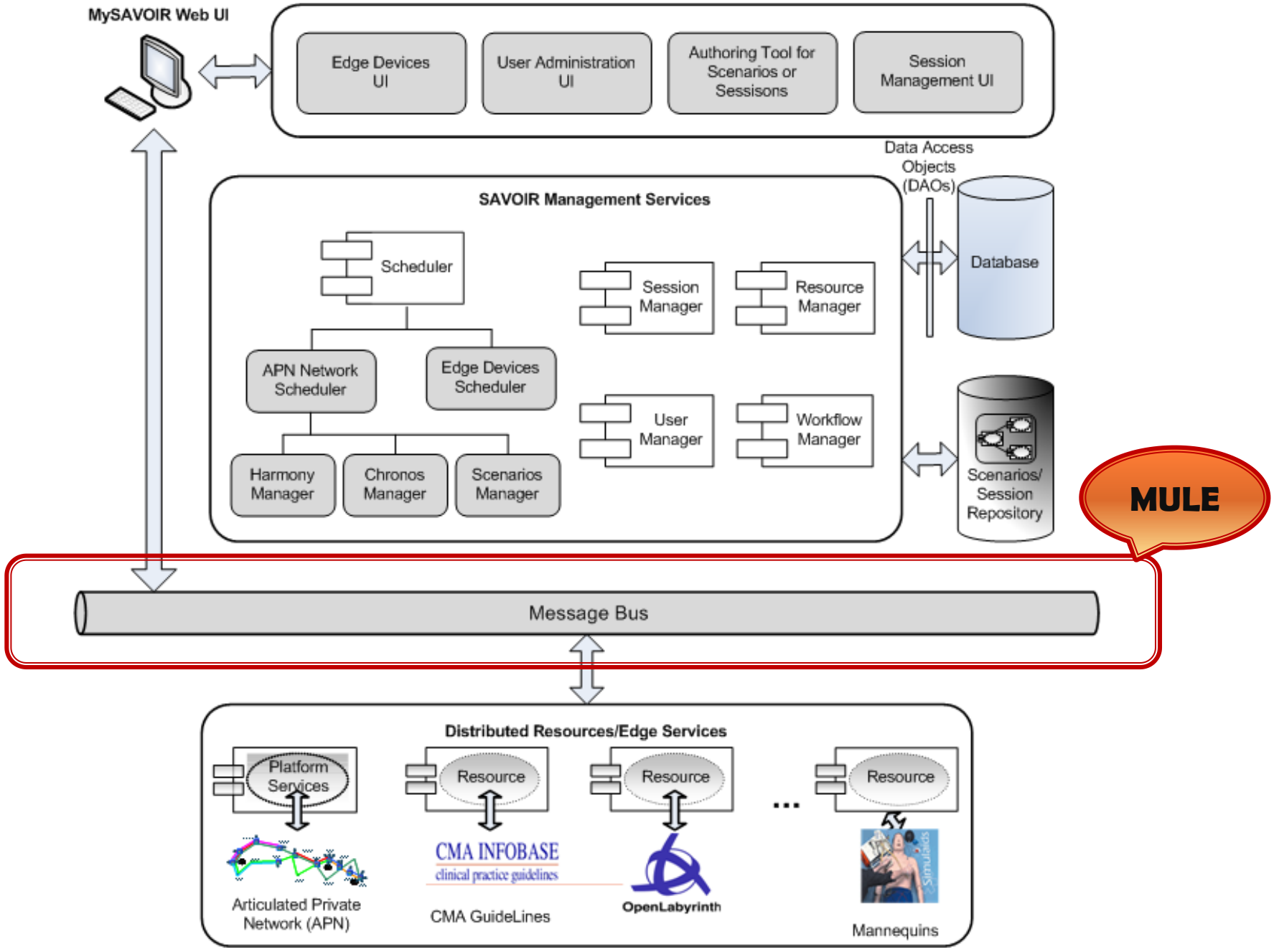
HSVO Bus Interface



service

HSVO SAVOIR Bus




MySAVOIR Web UI
Edge Devices UI
User Administration UI
Authoring Tool for Scenarios or Sessisons
Session Management UI
Data Access Objects (DAOs)
SAVOIR Management Services
Scheduler
Session Manager
Resource Manager
Database
APN Network Scheduler
Edge Devices Scheduler
User Manager
Workflow Manager
Harmony Manager
Chronos Manager
Scenarios Manager
Scenarios/ Session Repository
MULE
Message Bus
Distributed Resources/Edge Services
Platform Services
Resource
Resource
Resource
Articulated Private Network (APN)
CMA INFOBASE
clinical practice guidelines
CMA GuideLines
OpenLabyrinth
Mannequins

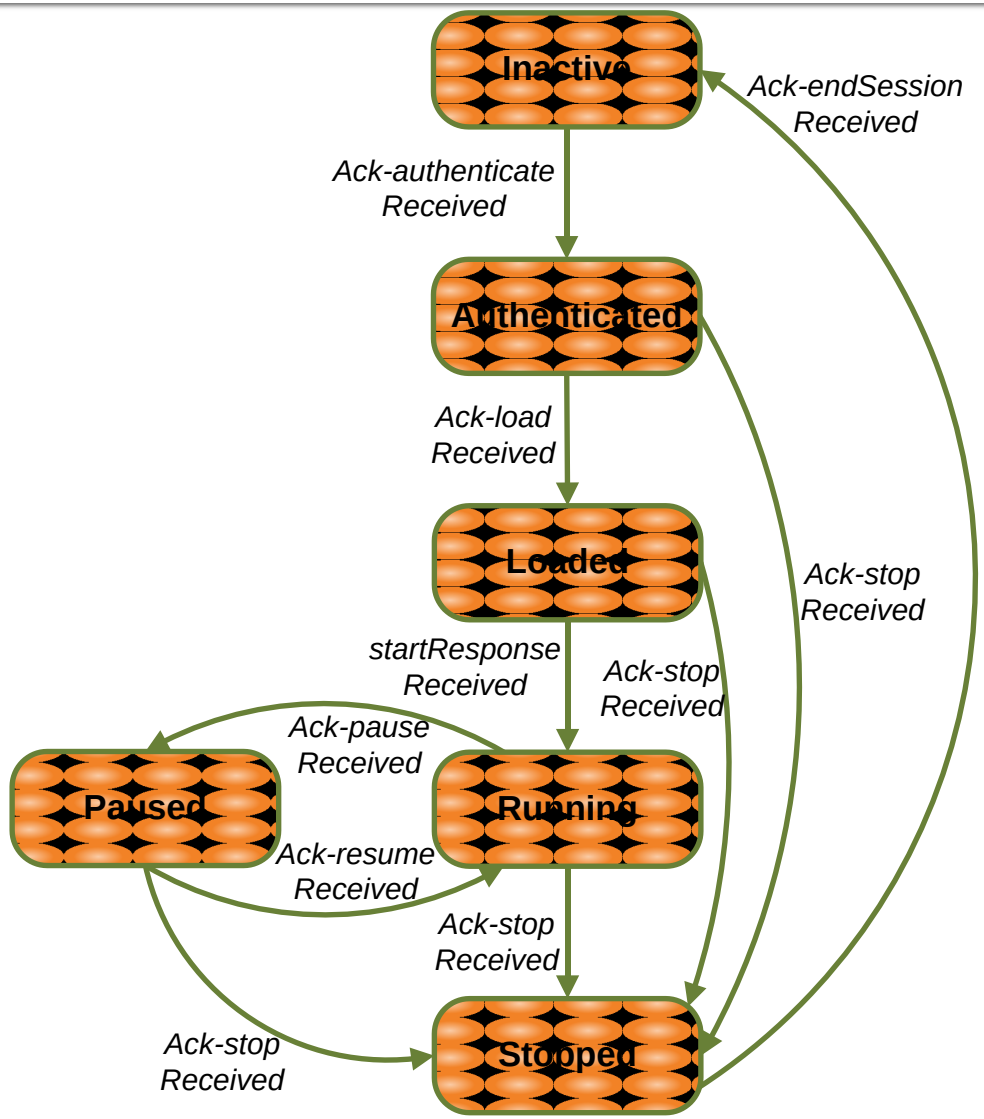
HSVO Messaging Spec

- Defines the communication syntax & semantics between SAVOIR and Edge Services.
- Uses Asyn. Communication model
- SAVOIR gets an ack. msg for every outgoing msg.
- The flow of interaction between SAVOIR & EDs are defined by Drools Rules
- SAVOIR maintains a state machine for each ED

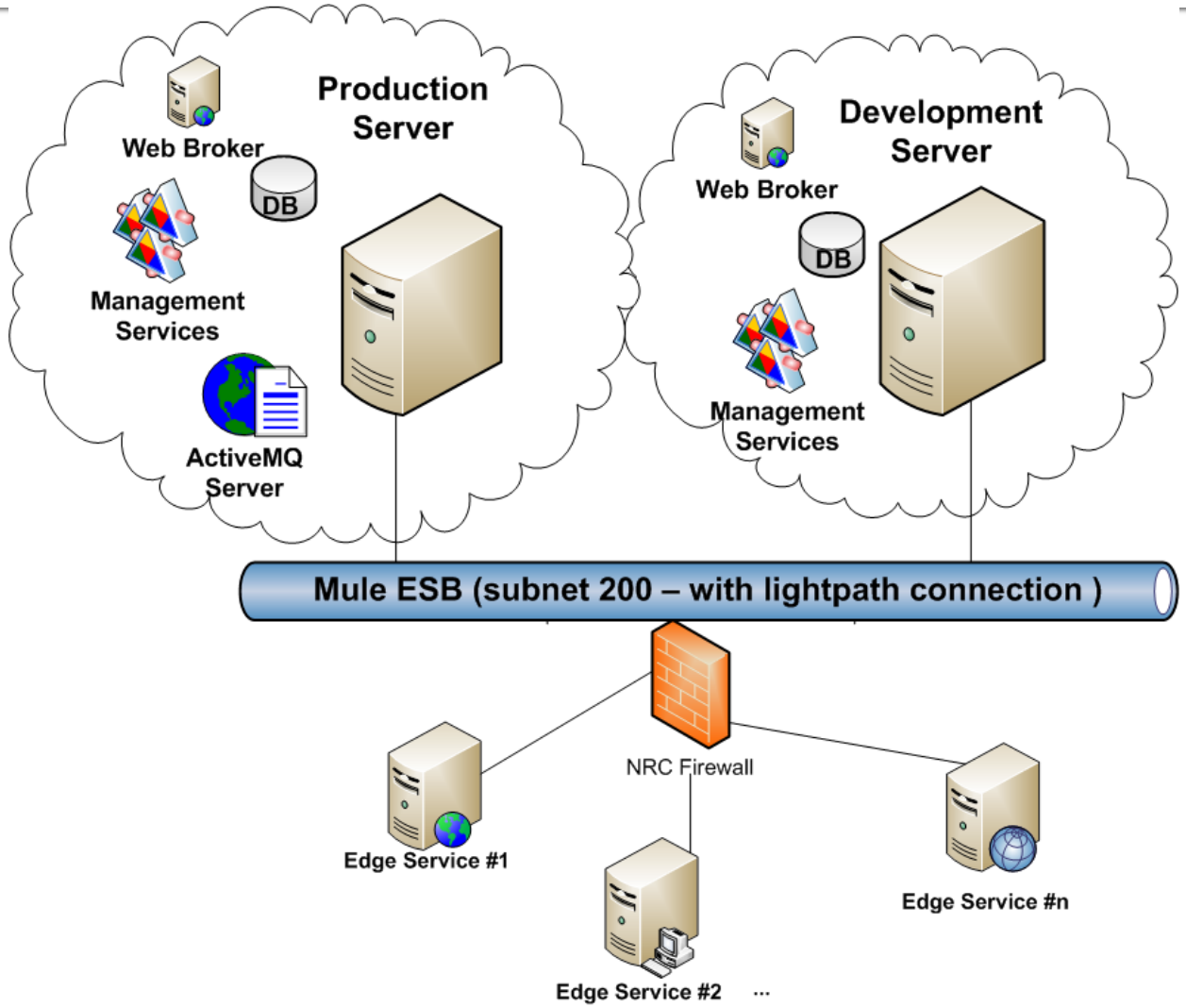
Edge Devices States

State	Meaning
Inactive	There is no active communication between SAVOIR and that ED for the given session (Note that this does NOT mean the device itself is inactive).
Authenticated	SAVOIR has authenticated a user to the edge service, but not yet loaded with the corresponding parameters for the session.
Loaded	The edge service activity is loaded, but not currently running.
Running	The edge service activity is currently loaded, started, and not paused.
Paused	The edge service activity is currently loaded, started, and paused.
Stopped	The edge service activity is stopped, but the session it is part of is not yet ended

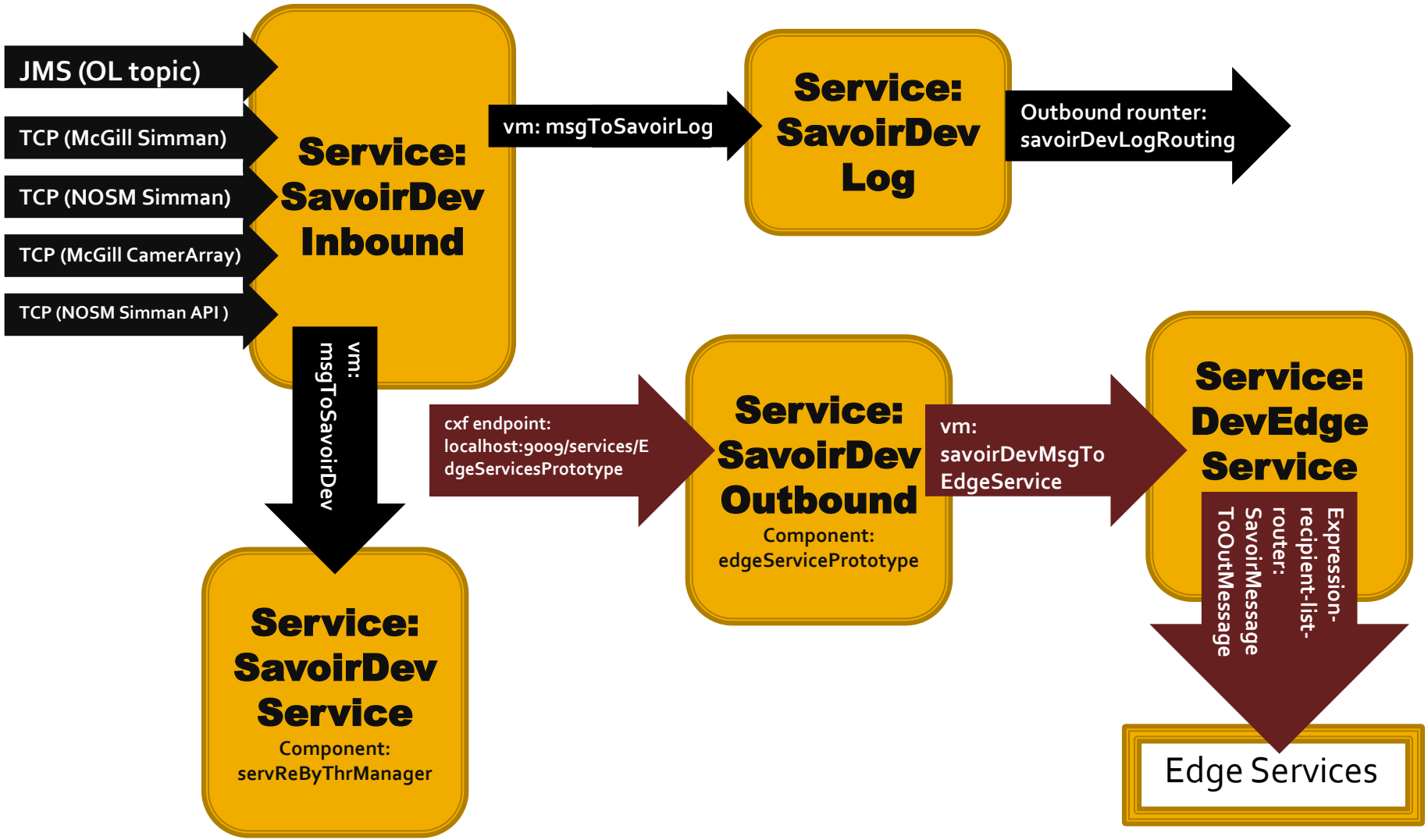
Message	States Sent In	States Discarded In
Authenticate	Inactive	Authenticated, Loaded, Running, Paused, Stopped
endSession	Stopped	Inactive
getStatus	Running, Paused	Inactive, Stopped
Load	Authenticated	Running, Paused, Stopped
Pause	Running	Paused, Stopped
Resume	Paused	Running, Stopped
setParameter	Running, Paused	Stopped, Inactive
Start	Loaded	Running, Paused, Stopped
Stop	Loaded, Running, Paused	Stopped, Inactive



Current Deployment



Mule Config for SAVOIR



Lessons Learned

- Mule is very flexible and configurable
- It can be stubborn at times
  - TCP connection/reconnection
  - CXF call dispatcher (threads are not being disposed)
  - Separate the bus and the mgmt services on two machines
  - Hot deployment option not available for community version
  - The cost of building a highly distributed system
    - Too many components to manage & maintain: consistency, debugging

Questions & Comments