mule and message-based integration - using an esb

dr gerald loeffler, mba
senior architect/developer
objectlab financial ltd

references and resources

- mule resources
  - [1] web site/wiki/guides http://www.mulesource.org
  - mailing list
  - examples (!)
  - source code (!)
- [2] "patterns of enterprise integration architecture", gregor hohpe et al.
  - [3] http://www.eaipatterns.com/
- [4] "Combining Service-Oriented Architecture and Event-Driven Architecture using an Enterprise Service Bus", jean-louis marechaux, ibm

contents

- mule by example
- message-based integration
- introduction to esbs and mule
- mule's core abstractions
- a closer look at the example
- gotchas / idioms /patterns

mule by example

starting/stopping, configuration, deployment options

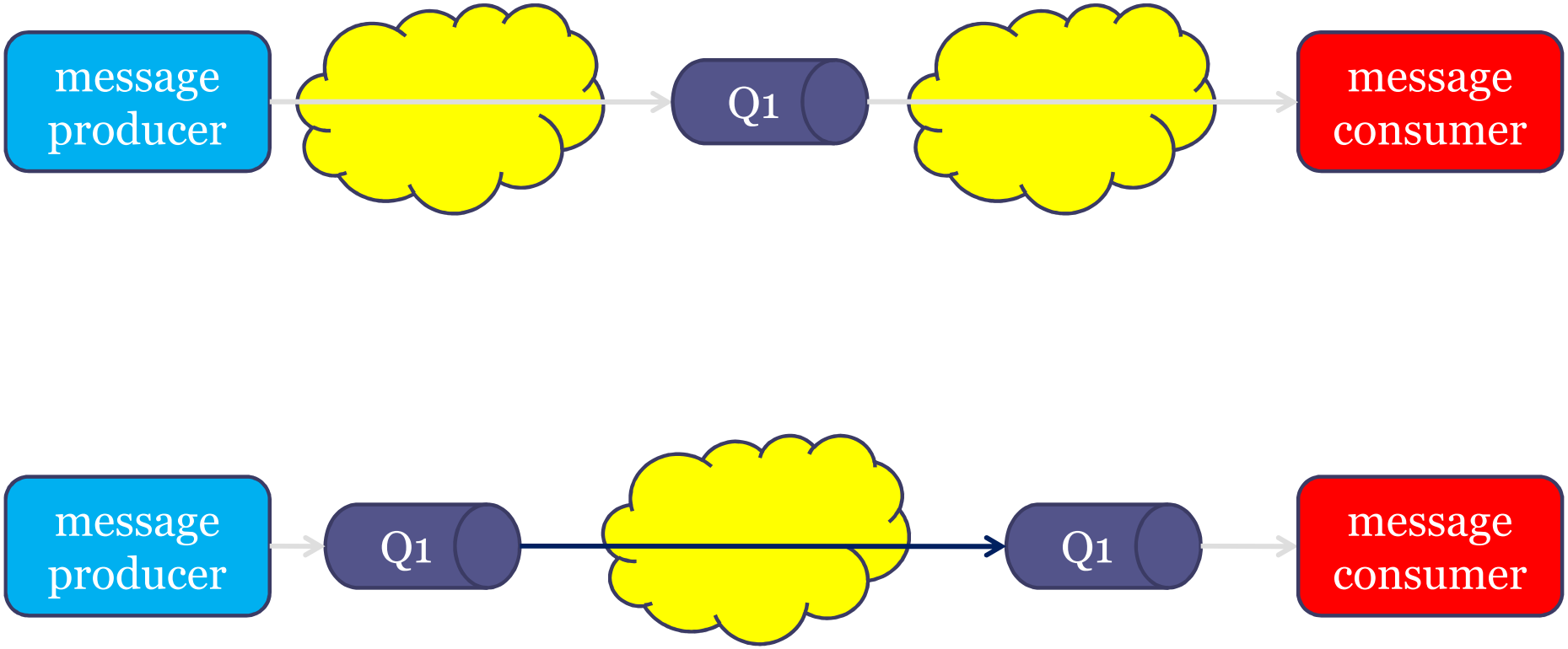
message-based integration

jms, store-and-forward messaging, pipes-and-filters architectures, enterprise integration patterns

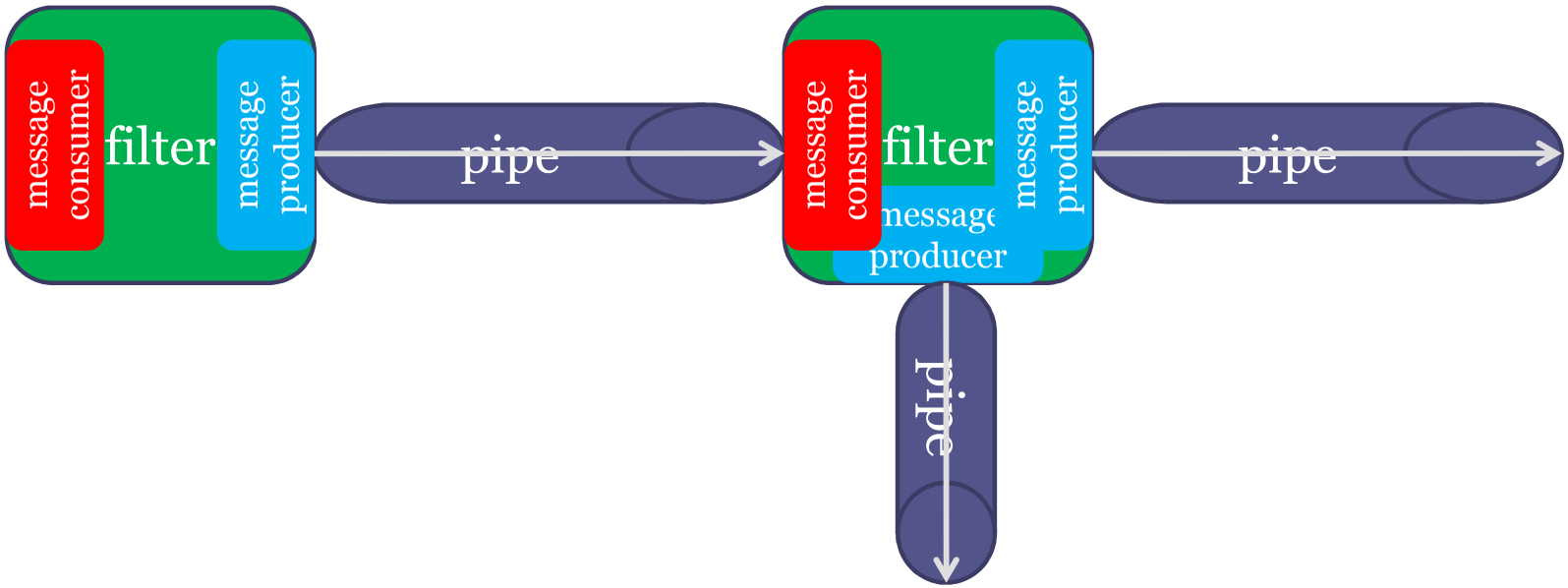
messaging-abstractions and jms terminology

- message
  - header
  - body
- message destination
  - queue, topic
- message producer
  - queue sender, topic publisher
- message consumer
  - queue receiver, topic subscriber
- de-coupling of producer and consumer
  - address space/jvm, time, language

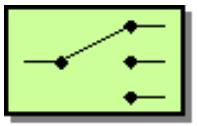
store-and-forward vs. client-server messaging



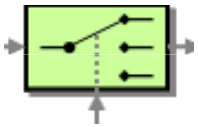
pipes-and-filters architecture

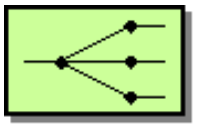


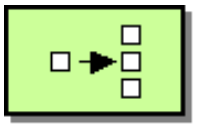
enterprise integration patterns

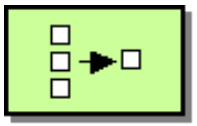

content-based router

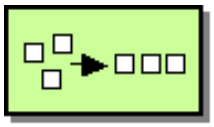

message filter

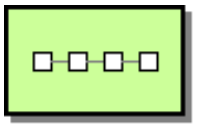

dynamic router

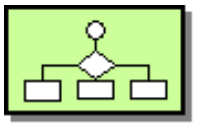

recipient list


splitter

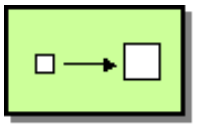

aggregator

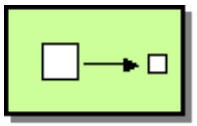

resequencer

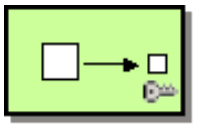

routing slip

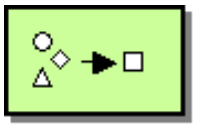

process manager


envelope wrapper


content enricher


content filter


claim check


normalizer

source: www.eaipatterns.com / G. Hohpe et al.

introduction to esbs and mule

what's an esb?, mule

esb - pattern or product?

- "An Enterprise Service Bus is an *architectural pattern* and can be implemented by many *different products* within the organization, and assembled together to act as a *federated* bus. More and more vendors are now offering a *complete product* to fulfill *enterprise integration* needs." [4]
- "combines *event-driven* and *service oriented* approaches to simplify integration of business units, bridging heterogeneous platforms and environments. The ESB acts as an intermediary layer to *enable communication* between different application processes." [4]

esb - pattern or product?

- "facilitates and simplifies business integration through *transport, event and mediation services*. It connects and mediates all communications and interactions between heterogeneous nodes, both in a Service-Oriented Architecture (synchronous one-to-one approach) and an Event-Driven Architecture (asynchronous many-to-many approach)." [4]
- esb characteristics/features:
  - supports synchronous and asynchronous interaction
  - message routing and transformation
  - transport mediation
  - distributed /federated?

mule 1/2

- event-based open-source esb
  - event ≈ message
- mature, proprietary (non-jbi) architecture
- active community
- integrates nicely with spring
- supports many transports/protocols
  - ejb invocation (jrmp, iiop), email (smtp, pop3, imap), file, ftp, http/https, jdbc, jms, ip multicast, tcp/ip incl. ssl, udp, soap/wsdl (via axis1, xfire, glue), xmpp, internal in-memory ("vm") [1]

mule 2/2

- deployment stand-alone or in app server (also as rar)
- framework for and implementations of
  - transformers
  - routers
  - splitters/aggregators
- transaction demarcation (local and xa, pluggable)
- thread, queue and pool management
- currently 1.4.3
  - 2.0 (late 2007) will be tightly integrated with spring 2.x

mule's core abstractions

event/message, connector, endpoint, component, router, transformer, interceptor, spring integration

event and message

- UMOEvent
  - all data sent or received within the mule environment will be passed between components as an UMOEvent
  - has a UMOMessage
- UMOMessage / UMOMessageAdapter
  - abstracts message implementations (jms, ftp, file, soap, ...)
  - has a (typed) payload and properties

connector: file

```
<connector name="file_connector"
      className="org.mule.providers.file.FileConnector">
  <properties>
    <property name="moveToDirectory" value="${submitted_dir}" />
    <property name="moveToPattern"
      value="${ORIGINALNAME}.${DATE:yyyy-MM-dd_HH-mm-ss-SSS}" />
    <property name="binary" value="false" />
    <property name="autoDelete" value="false" />
  </properties>
</connector>
```

- was unable to configure it to not move/delete files when outbound communication fails
  - make outbound communication impossible to fail

connector: ejb (slsb)

```
<connector name="ejb_connector"
      className="org.mule.providers.ejb.EjbConnector">
  <properties>
    <property name="jndiInitialFactory"
              value="${jndi_initial_factory}" />
    <property name="jndiProviderUrl" value="${jndi_provider_url}" />
    <property name="securityPolicy" value="security.policy" />
  </properties>
</connector>
```

- for calling slsb method through mule endpoints
  - from outbound or nested routers (but prefer spring to latter)
- ejb container context is an "early binding" alternative
- undocumented lookup/caching behaviour fails on reconnect
  - prefer spring `jee:remote-slsb` whenever possible

jms connector

```
<connector name="jms_connector"
      className="org.mule.providers.jms.JmsConnector">
  <properties>
    <property name="jndiInitialFactory"
              value="${jndi_initial_factory}" />
    <property name="jndiProviderUrl" value="${jndi_provider_url}" />
    <property name="connectionFactoryJndiName"
              value="ConnectionFactory" />
    <property name="jndiDestinations" value="true" />
    <property name="forceJndiDestinations" value="true" />
    <property name="persistentDelivery" value="true" />
    <property name="specification" value="1.1" />
    <property name="maxRedelivery" value="5" />
  </properties>
</connector>
```

endpoints

```
<global-endpoints>
  <endpoint name="to_submit_dir" address="file://${to_submit_dir}"
    connector="file_connector" transformers="f2s" />
  <endpoint name="z_reviews_csv" address="vm://z_reviews_csv" />
  <endpoint name="review"
    address="ejb:/ReviewSLSBean?method=addReview"
    connector="ejb_connector" remoteSync="true" />
  <endpoint name="invalid_msg"
    address="jms://queue/INT.INVALID_MESSAGE"
    connector="jms_connector" transformers="o2jms" />
</global-endpoints>
```

- central vehicle for mules protocol/transport abstraction
- best defined as global endpoints
- only define fixed transformers => can amend later

transformers

```
<transformers>
  <transformer name="f2s"
    className="org.mule.providers.file.transformers.FileToString" />
  <transformer name="o2jms" className=
    "org.mule.providers.jms.transformers.ObjectToJMSMessage" />
</transformers>
```

- infrastructural, technical transformers provided by mule
- more business-related transformers written in projects
  - extend org.mule.transformers.AbstractTransformer
  - not a spring bean if configured via mule xml
    - simple transformation => transformer, complex => component

component and routers 1

```
<mule-descriptor name="z_filedrop"
    implementation="org.mule.components.simple.PassThroughComponent">
  <inbound-router>
    <global-endpoint name="to_submit_dir" />
  </inbound-router>
  <outbound-router>
    <router className=
        "org.mule.routing.outbound.OutboundPassThroughRouter">
      <global-endpoint name="z_reviews_csv" />
    </router>
  </outbound-router>
  <interceptor name="default" />
</mule-descriptor>
```

component and routers 2

```
<mule-descriptor name="splitter" implementation=
    "org.mule.components.simple.PassThroughComponent">
  <inbound-router>
    <global-endpoint name="z_reviews_csv" />
  </inbound-router>
  <outbound-router>
    <router className="util.esb.mule.LineSplitter">
      <global-endpoint name="z_review_csv" />
      <properties>
        <property name="preserveHeader" value="true" />
      </properties>
    </router>
  </outbound-router>
  <interceptor name="default" />
</mule-descriptor>
```

component and routers 3

```
<mule-descriptor name="transformer"
    implementation="zReviewCsvToReviewTransformer">
  <inbound-router>
    <global-endpoint name="z_review_csv" />
  </inbound-router>
  <outbound-router>
    <router className=
        "org.mule.routing.outbound.OutboundPassThroughRouter">
      <global-endpoint name="review" />
    </router>
  </outbound-router>
  <interceptor name="default" />
</mule-descriptor>
```

container context and spring integration

```
<container-context
    className='org.mule.extras.spring.SpringContainerContext'>
  <properties>
    <property name="configFile"
              value="springContext-monolith.xml" />
  </properties>
</container-context>
```

- mule containers are external component (!) factories
  - object/beans referenced with `implementation` attribute
- other container context implementations
  - jndi tree, session ejbs, pico, plexus, hivemind [1]
- mule can also be configured and launched entirely from a spring context (without a mule config xml)!

component exception strategy

```
<exception-strategy
    className="util.esb.mule.ExceptionAwareExceptionStrategy">
  <global-endpoint name="invalid_msg" />
  <global-endpoint name="manual_intervention" />
</exception-strategy>
```

```
public class ExceptionAwareExceptionStrategy implements
    ExceptionListener {
  public void exceptionThrown(Exception e) {...}
}
```

- handles exceptions thrown by components
  - usually business exceptions, i.e. project-dependent
  - a special form of synchronous content-based routing for exceptions

interceptors

```
<interceptor-stack name="default">
  <interceptor className="util.esb.mule.LoggingInterceptor" />
</interceptor-stack>
```

```
public class LoggingInterceptor extends EnvelopeInterceptor {

  @Override public void before(Invocation i) {...}
  @Override public void after(Invocation i) {...}
}
```

- cf. aop around advice
- applies to component invocations
  - not invoked for BridgeComponent
    - use PassThroughComponent instead
- for debugging, but also for message store/history/audit

a closer look at the example

real-live mule configuration

gotchas/idioms/patterns

gotchas

- always consult javadocs or source
- mule-internal config of threads/queues/timeouts
- tx handling across transports (avoid)
- jms connector number of sessions
- component exception handling and tx rollbacks

esb/mule design idioms/patterns

- use global endpoints
- keep it simple and use spring where possible
- database access from a mule node? what db? how?
- leave remote operations to messaging backbone
- isolate synchronous operations
- tie in to cep/esp (complex event processing/event stream processing) library like esper for calculating statistics over and analyzing/querying message flows
  - statistics over number/size/type of messages per time unit/sender
  - real-time outlier detection in terms of message size/type
  - QoS monitoring

thank you

dr gerald loeffler, mba
senior architect/developer
objectlab financial ltd

http://www.gerald-loeffler.net/download/mule_javaabend_2007-10-22.zip