PRESENTED BY RED HAT

LEARN. NETWORK.
EXPERIENCE OPEN SOURCE.

Real-time Intelligent Messaging for the Enterprise with Red Hat Enterprise MRG Messaging

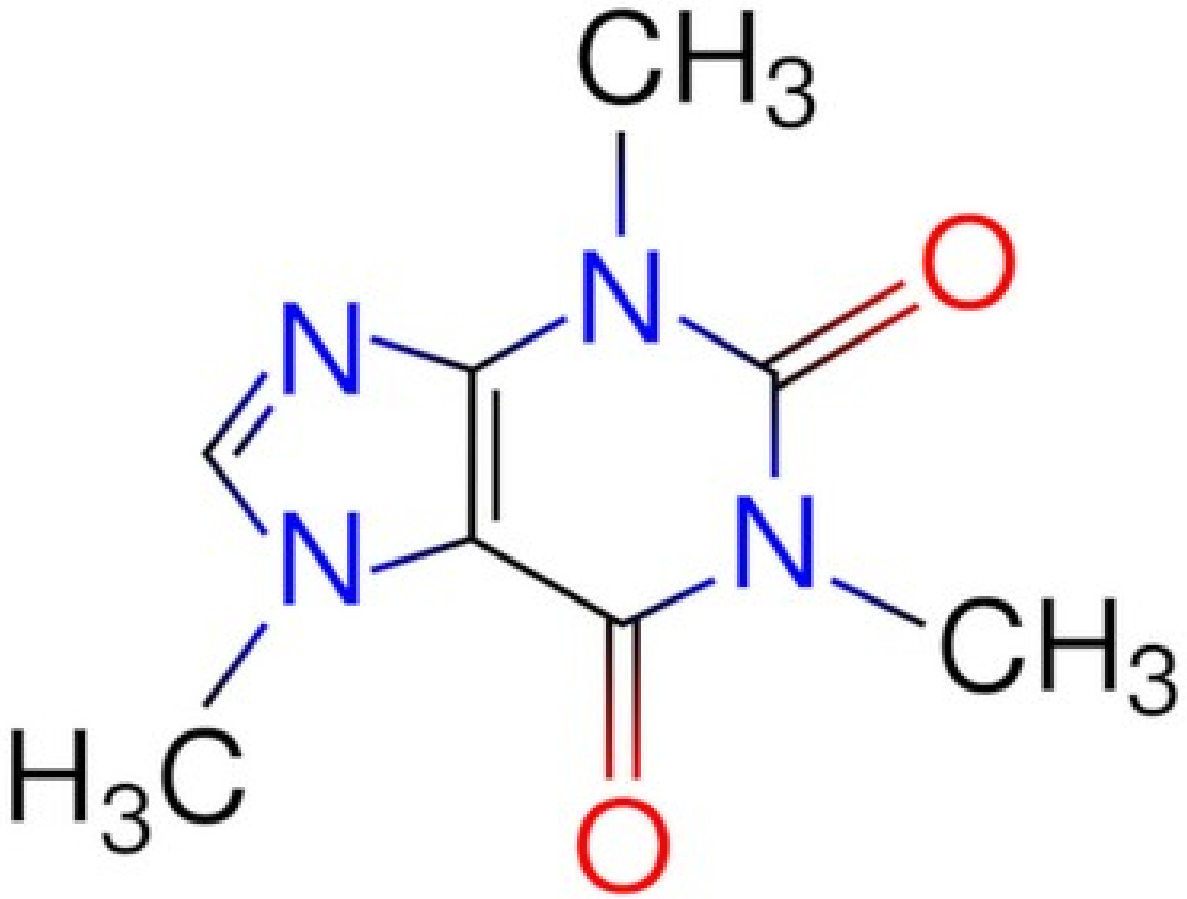
Vamsi Chemitiganti, Chief Solution Architect

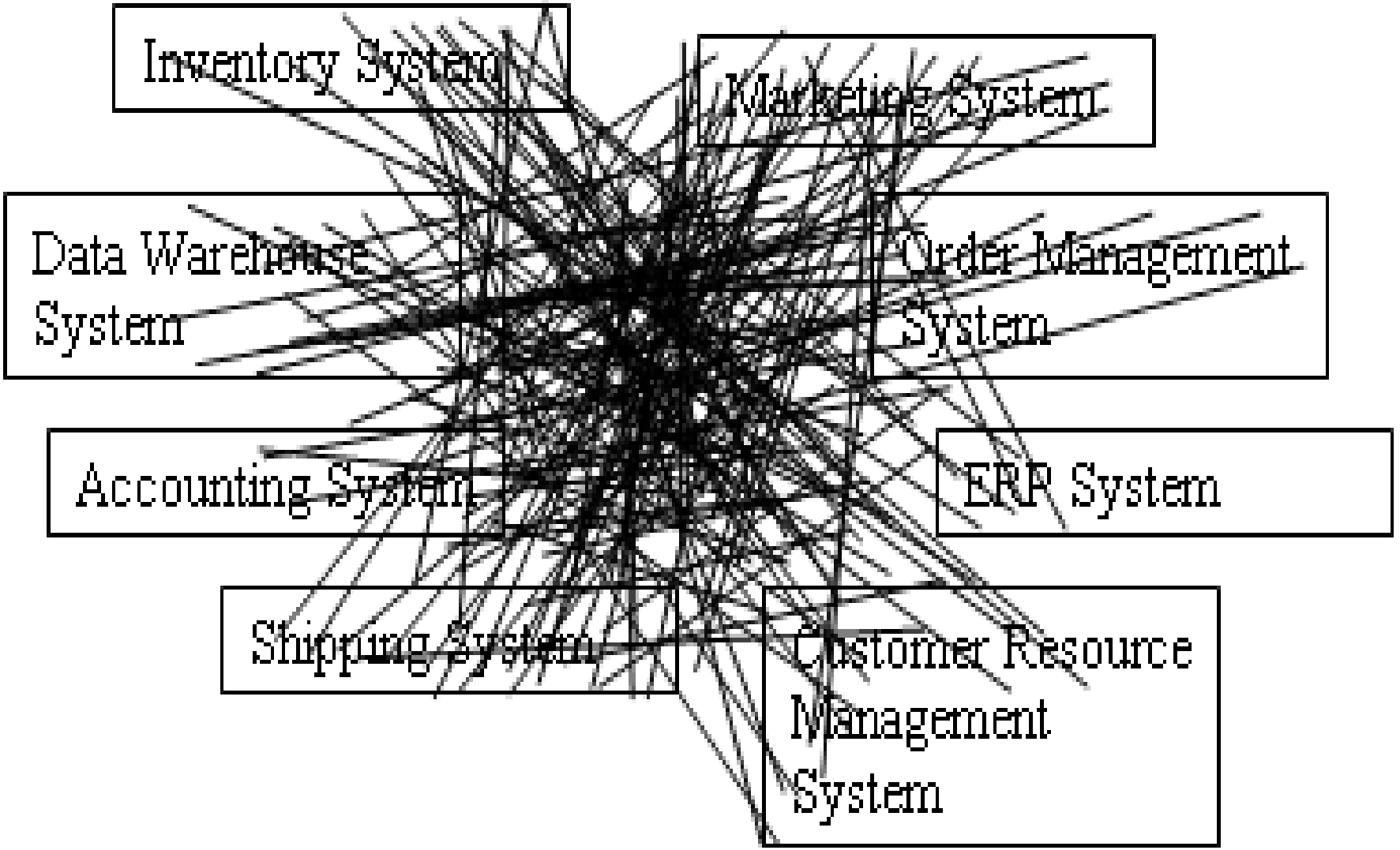
Derrick Kittler, Senior Solution Architect

Red Hat

06.29.12


VOLVO
redhat
COVIDIEN
FENWAY PARK 100th ANNIVERSARY
newbalance
DUNKIN' DONUTS
Ford
xfinity
Budweiser


CH3
N
O
N
N
N
CH3
H3C
O


Inventory System
Marketing System
Data Warehouse System
Order Management System
Accounting System
ERP System
Shipping System
Customer Resource Management System

Goals

- Operational Efficiency
  - Better, faster, cheaper (possibilities with Red Hat integration technologies)
- Solve targeted pain points; compliance, mobility, etc...
  - Looking beyond today
- Explore new go-to-business models
  - Blue Oceans

Significance

- The Age of 'Big Data' and Cloud
- Volume of data
- Velocity of data
- Variety of sources
- Complexity

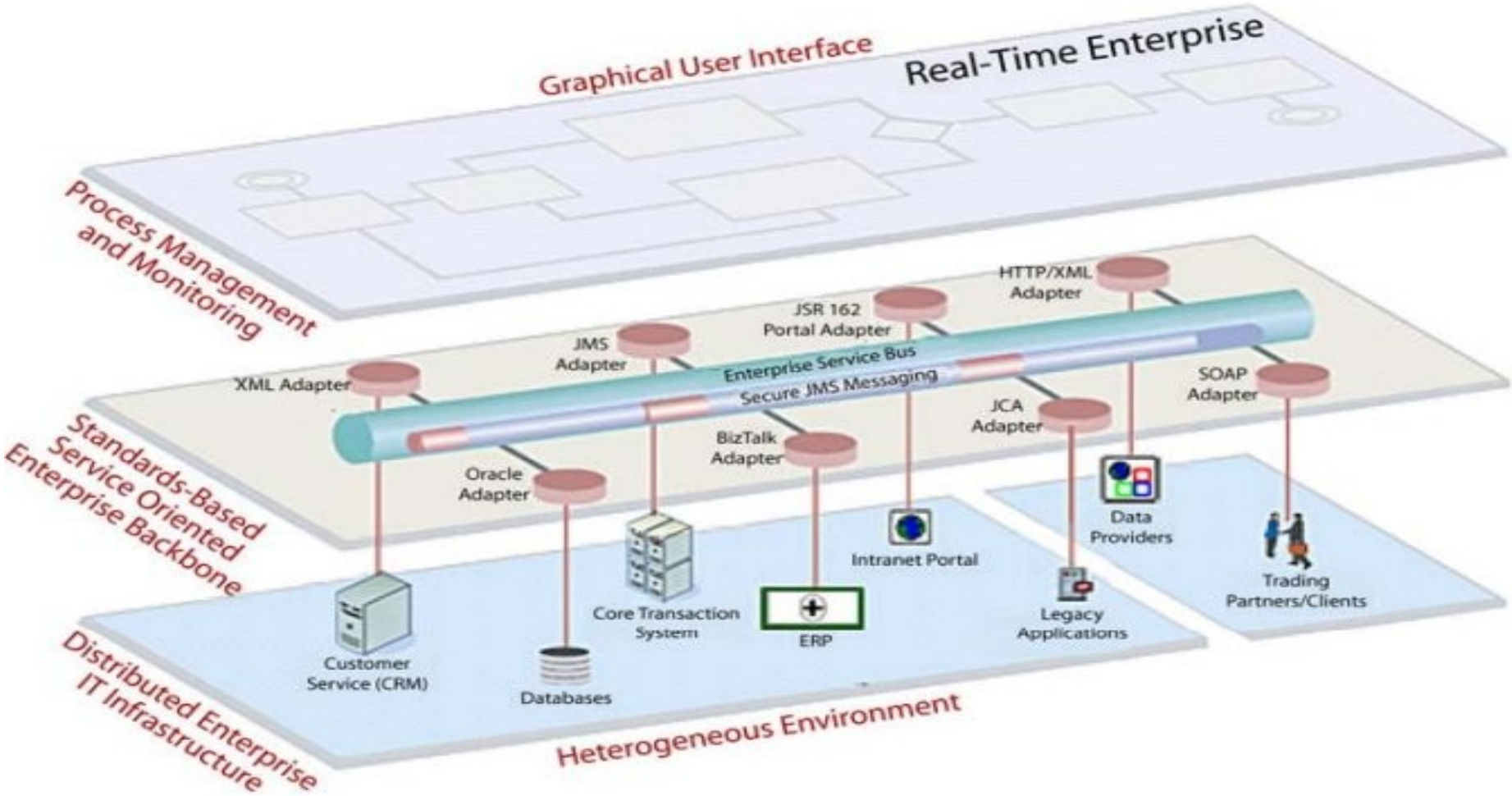
Business Drivers – vertical focus

- Operational Efficiency
  - Business Intelligence
- Blue Ocean Business Models
- React to Customer Trends
- Regulatory Compliance
- Detect Fraud

Background – use case details

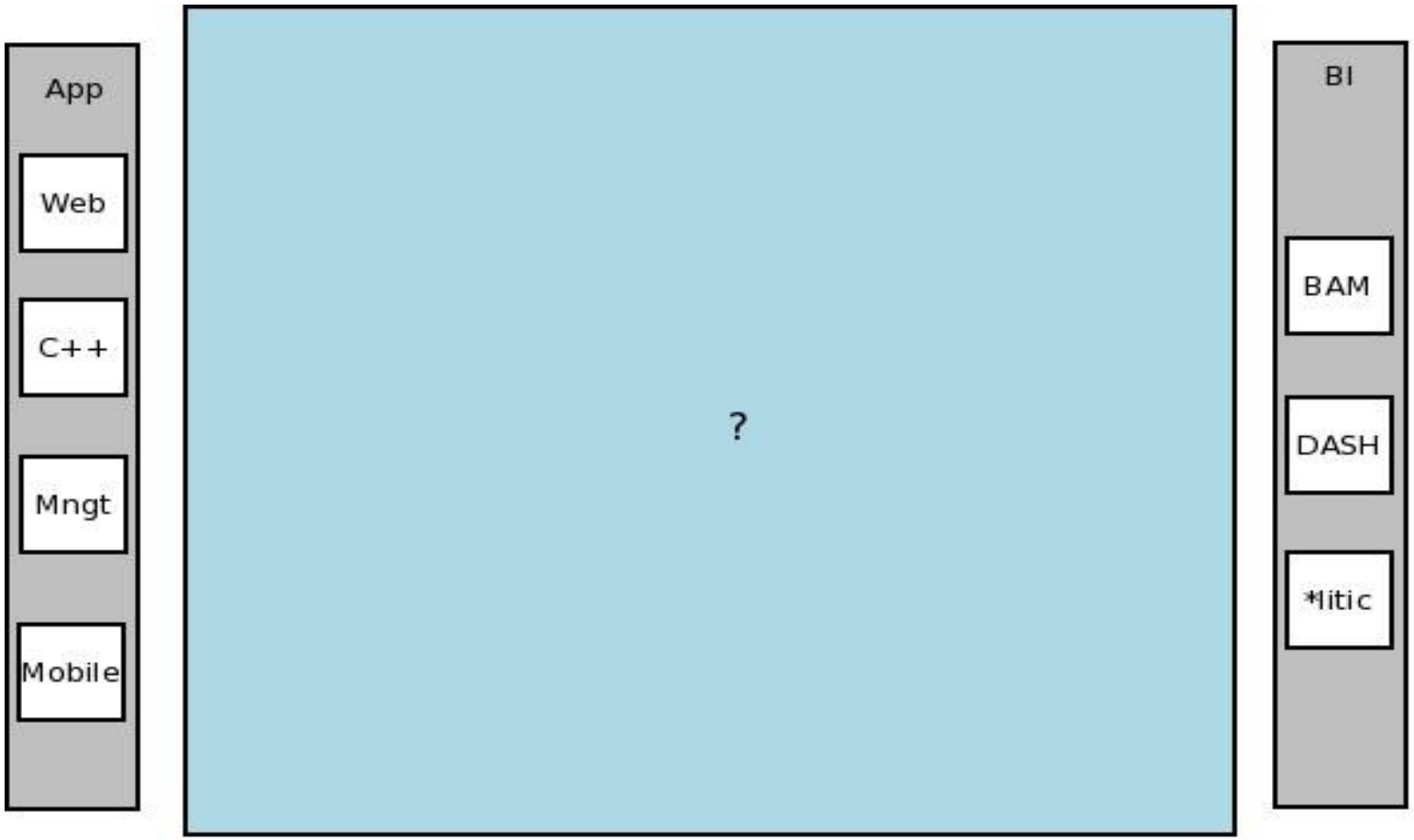
- Mine realtime data for Business Intelligence
- Support multiple types of clients (Web, Mobile, System, B2B, etc...)
- Customer intelligence for targeted marketing
- Application and Performance Management
- Include semi-structured and un-structured data in decision management

The problem...



Open Source SOA, Jeff Davis, Manning Press

In Simpler Terms...



Background – solution requirements...

- Reactive correlation – compare the incoming event data to reference configurations.
- Scan the data for patterns that known to trigger outages
- Automation
  - automate the handling of detected incidents (rather than just feeding the manual L1/L2 resolution path)
- Avoid incidents
  - reduce time to manage incidents leading to reduced operational support costs.

The Challenges

- Large and Complex Datasets
- Multi-channel
  - Midrange systems, web servers, web applications
- Require near realtime capabilities
- Need to correlate a lot of data
- Where do the rules come from? Need to define them!
- Business processes govern everything
- But what about my heterogeneous infrastructure?

Technology Integration

- Devil is in the details...
  - Multiple types of clients – web, mobile, system, B2B etc
  - Scale requirements grow (50k nodes)
  - There is no point solution that will scale to process those kinds of volumes and support all those types of clients out of the box
  - Need specialized middleware to handle each component of the solution (messaging, caching, distribution, correlation, analytics, BI, etc)

Solution Architecture – Main Building Blocks

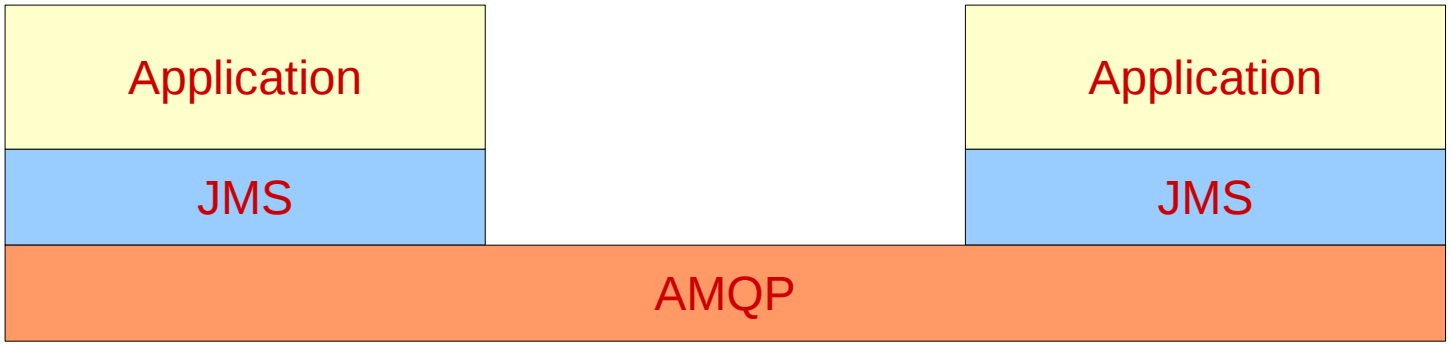
- Open Messaging
  - Advanced Messaging Queuing Protocol (AMQP)
- Realtime deterministic processing (MRG Realtime)
- Complex Event Processing
  - event detection and correlation (JBoss BRMS)
- Decision processing
  - JBoss Business Rules Management System
- Automation via JBoss BRMS (Rules, Business Process & Complex Event Processing)

AMQP

- Practical, comprehensive messaging specification
- Supports pervasive deployments
  - Open IP governs usage
  - Supports expanding ecosystem
- Designed for real world requirements
  - Developed by vendors and user organizations

Bank of America	INETCO Systems	Software AG
Barclays Bank	Informatica	Solace Systems
Cisco Systems	JPMorgan Chase Bank	Storm
Credit Suisse	Microsoft	Tervela
Deutsche Börse Systems	Novell	TWIST
Envoy Technologies	Progress Software	WSO2
Goldman Sachs	Rabbit	VMWare Inc
HCL Technologies	Red Hat	29 West

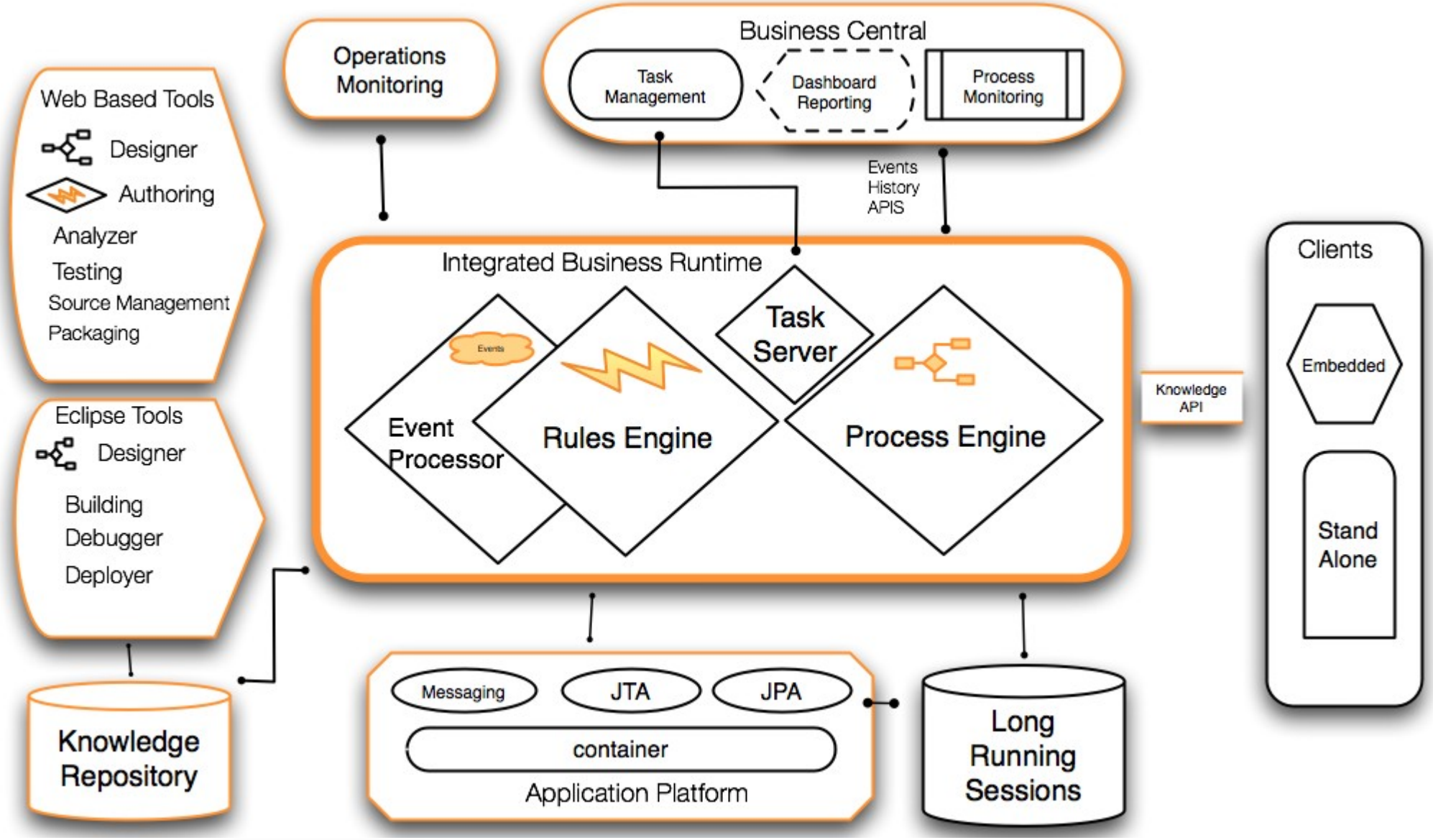
APIs and Protocols



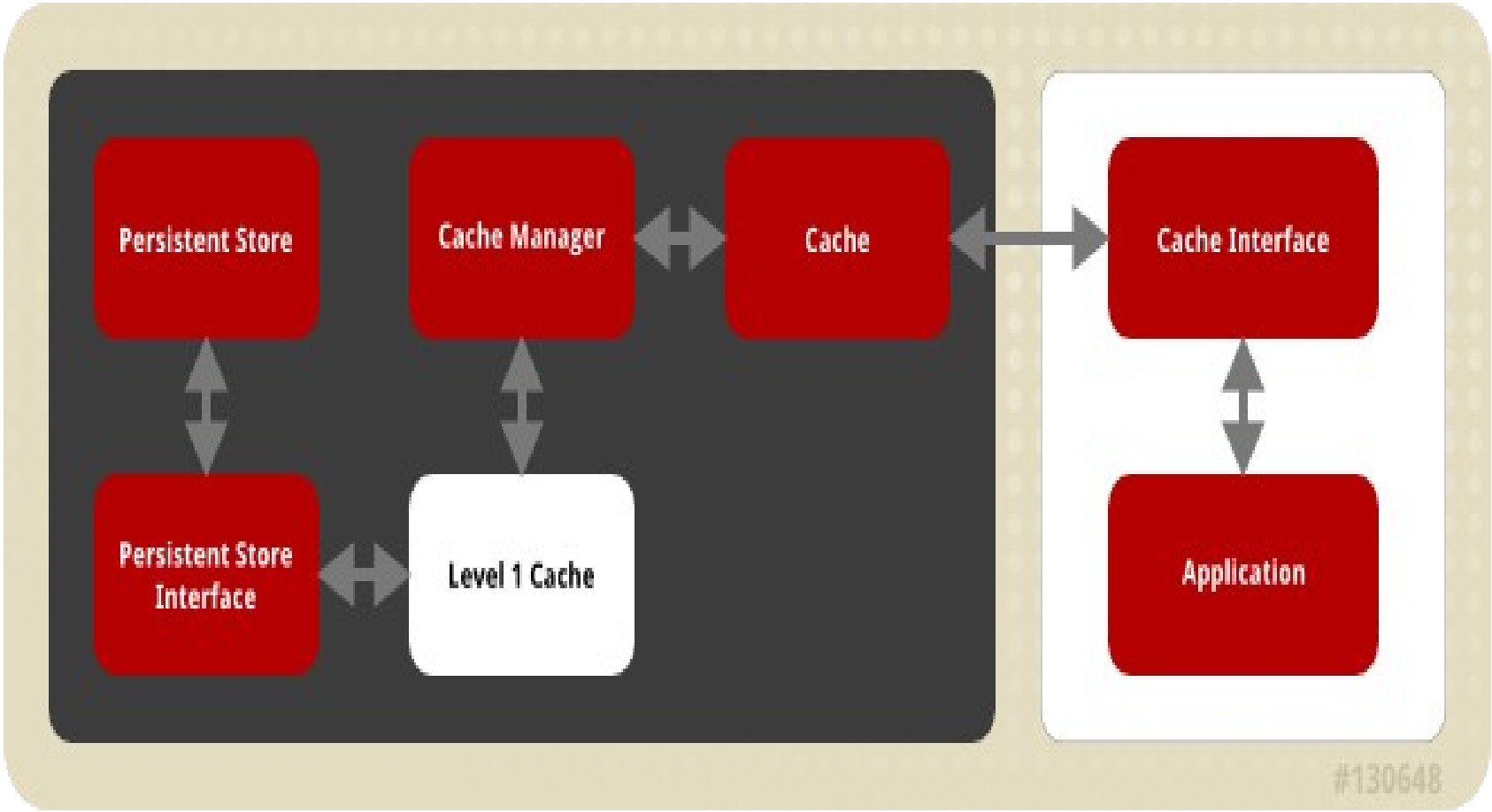
MRG Messaging Feature Highlights

- Core Messaging
  - P2P, fanout, pub-sub, async
  - Transactions -local to dtx
  - Multiple clients (C++, .NET, Java, JMS, Python, etc.)
- High Performance
  - C++ broker, optimized for RHEL
  - AIO for high-speed durable messaging
  - Infiniband RDMA support for ultra low latency messaging
- Management tools
  - Command line tools to Web-based GUI
  - AMQP-based framework & APIs
- Advanced Features
  - Queue Semantics: Ring Queue, Last Value Queue, TTL, Initial Value Exchange, etc
  - Routing patterns, including XML XQuery
  - Federation with dynamic routes
- High Availability
  - Active-Active Broker Clustering
  - Federated disaster recovery
- Security
  - SASL auth
  - SSL encryption
  - role-based access control

JBoss Business Rules Management System



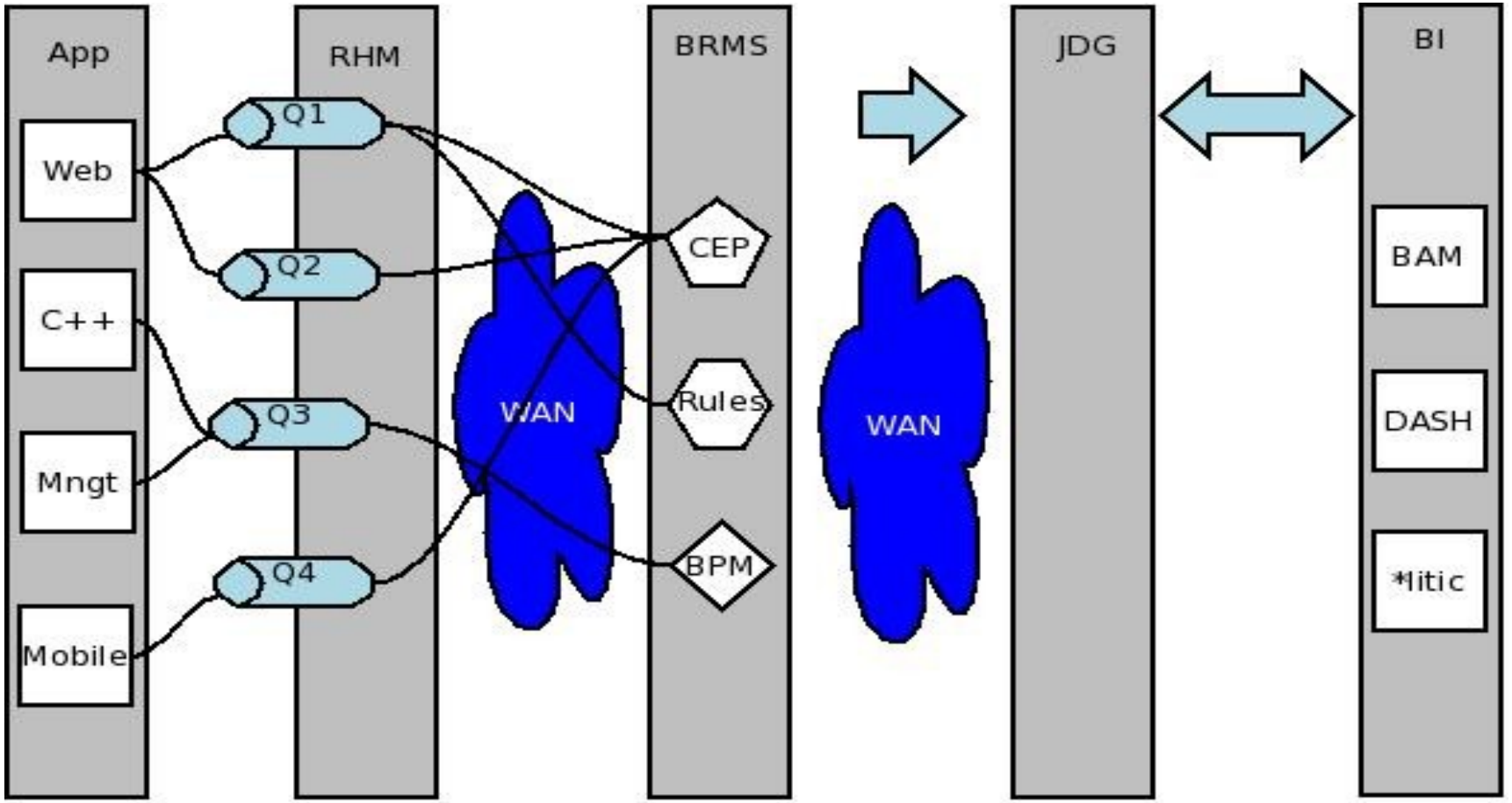
JBoss Data Grid – Core Architecture



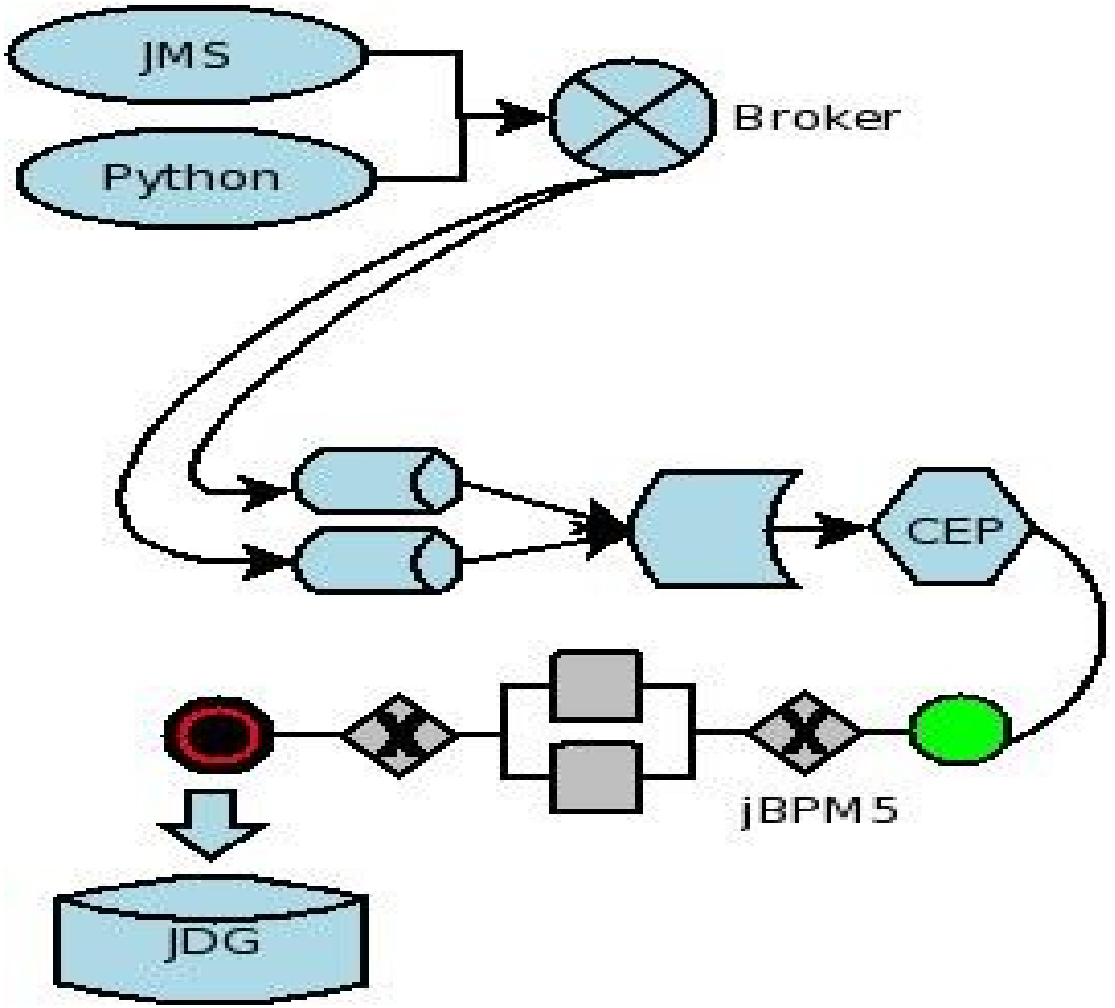
Solution Architecture - three distinct logical tiers

- Data Handling
  - Process incoming data streams and hand-off to downstream modules / applications
  - MRG M/QMF and RealTime are good fits
- Data Processing
  - Rules, Workflow and Event correlation
- Analytics and Automation
  - JBoss Data Grid and NoSQL

High Level Conceptual Architecture



Data Flow



Conceptual Architecture

- Red Hat Messaging and Realtime Kernel
  - handle incoming real time streams and
  - provide deterministic response times
- Normalized message format
- BRMS for rule authoring, event correlation, decision management & automation
- JBoss Data Grid exposes data for fast querying and ease of data consumption

Code ...

Twitter Stream – Messaging (py)

```
broker =  "localhost:5672" if len(sys.argv)<2 else sys.argv[1]
address = "amq.topic" if len(sys.argv)<3 else sys.argv[2]
connection = Connection(broker)

def on_results(results):
    """
    A callback to handle passed results. Wheeee.
    """
    try:
        connection.open()
        session = connection.session()
        sender = session.sender(address+'; {create:always}' )
        msg = json.dumps(results)
        sender.send(Message(msg))
    except MessagingError,m:
        print m
    finally:
        connection.close()


Twython.stream({
    'username': '<twitter_user>',
    'password': '<twitter_pass>',
    'track': 'redhatnews,vmware,oracle,microsoft,google'
}, on_results)
```


Twitter Stream – Messaging (java)

```
  public void connect() {
          if (producer != null)
                  return;

          try {
                  Properties properties = new Properties();
                  properties.load(this.getClass().getResourceAsStream(QPID_PROPS));
                  context = new InitialContext(properties);

                  connectionFactory = (ConnectionFactory) context.lookup(CONN_FACTORY);
                  connection = connectionFactory.createConnection();
                  connection.start();

                  session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
                  destination = (Destination) context.lookup(DESTINATION);

                  producer = session.createProducer(destination);


StatusListener listener = new StatusListener() {
        public void onStatus(Status status) {
                //String tweet = status.getUser().getName() + ":" + status.getText();
                String tweet = statusToJSON(status);
                sampler.enqueueTweet(tweet);
                System.out.println(tweet);
        }

        public void onException(Exception ex) {
                ex.printStackTrace();
                System.exit(1);
        }

        public void onDeletionNotice(StatusDeletionNotice statusDeletionNotice) { }
        public void onTrackLimitationNotice(int numberOfLimitedStatuses) { }
        public void onScrubGeo(long userId, long upToStatusId) {}
```


Stock Stream

```
broker =  "localhost:5672" if len(sys.argv)<2 else sys.argv[1]
address = "amq.topic" if len(sys.argv)<3 else sys.argv[2]
connection = Connection(broker)
tickers = ['vmw','goog','orcl','rht','msft']

try:
    connection.open()
    session = connection.session()
    while True:
        for sym in tickers:
            sender = session.sender(address+'; {create:always}' )
            msg = str(json.dumps(['Stock', {sym: (stockquote.get_quote(sym))}]))
            sender.send(Message(msg))
            time.sleep(2)
except MessagingError,m:
    print m
finally:
    connection.close()
```


Messaging JCA Integration

```
<mbean code="org.jboss.resource.deployment.AdminObject"
       name="qpid.jca:name=TwitterFeed">
   <attribute name="JNDIName">TwitterFeed</attribute>
   <depends optional-attribute-name="RARName">jboss.jca:service=RARDeployment,name='qpid-ra-0.14.rar'</depends>
   <attribute name="Type">javax.jms.Destination</attribute>
   <attribute name="Properties">
      destinationType=QUEUE
      destinationAddress=demo/tweets;{create:always, node:{type:queue, x-declare:{auto-delete:true}}}
   </attribute>
</mbean>

<mbean code="org.jboss.resource.deployment.AdminObject"
       name="qpid.jca:name=StockFeed">
   <attribute name="JNDIName">StockFeed</attribute>
   <depends optional-attribute-name="RARName">jboss.jca:service=RARDeployment,name='qpid-ra-0.14.rar'</depends>
   <attribute name="Type">javax.jms.Destination</attribute>
   <attribute name="Properties">
      destinationType=QUEUE
      destinationAddress=demo/stocks;{create:always, node:{type:queue, x-declare:{auto-delete:true}}}
   </attribute>
</mbean>
```

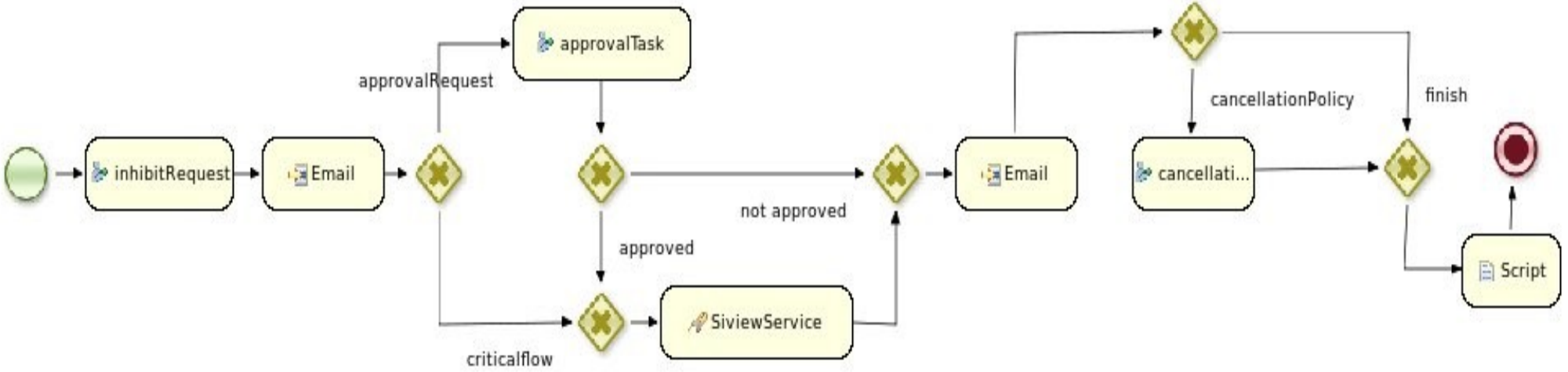

BRMS Integration – properties and changeset

```
name=retim
url=http://localhost:8080/jboss-brms/org.drools.guvnor.Guvnor/package/retim/LATEST
enableBasicAuthentication=true
username=admin
password=admin
drools.agent.newInstance=false
drools.agent.scanDirectories=false
```

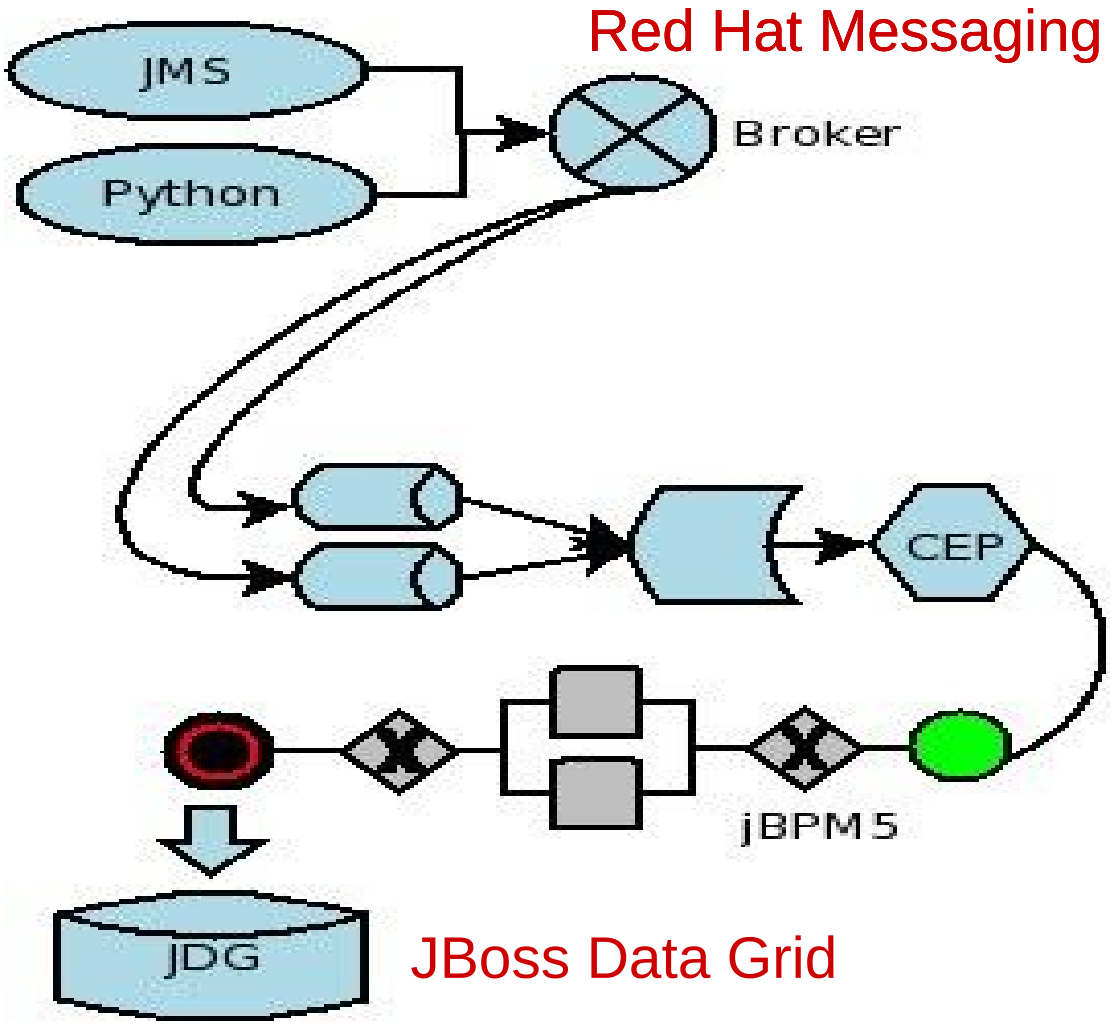
```
<?xml version="1.0" encoding="UTF-8"?>
<change-set xmlns='http://drools.org/drools-5.0/change-set'>
        <add>
                <resource
                        source='http://localhost:8080/jboss-brms/org.drools.guvnor.Guvnor/package/retim/LATEST'
                        type='PKG' basicAuthentication="enabled" username="admin" password="admin" />
        </add>
</change-set>
```


BRMS Integration – CEP / jBPM

```
<action class="org.jboss.soa.esb.actions.BusinessRulesProcessor" name="Fire Until Halt">
        <property name="ruleAgentProperties" value="/resources/brms.properties" />
        <property name="ruleReload" value="true" />
        <property name="stateful" value="true" />
        <property name="ruleEventProcessingType" value="STREAM" />
        <property name="ruleFireMethod" value="FIRE_UNTIL_HALT" />
        <property name="object-paths">
                <object-path esb="body.tweet" />
        </property>
        <property name="defaultContinue" value="true" />
</action>
```



Data Flow – to review ...



Lessons Learned

- Open platform
  - AMQP (common messaging patterns, multi-language bindings etc...)
- Comprehensive ecosystem
  - Rules, events, processes in one platform
- Design for scalability
- Pluggable and modular architecture

QUESTIONS?

LIKE US ON FACEBOOK

www.facebook.com/redhatinc

FOLLOW US ON TWITTER

www.twitter.com/redhatsummit

TWEET ABOUT IT

#redhat

READ THE BLOG

summitblog.redhat.com

GIVE US FEEDBACK

www.redhat.com/summit/survey