IBM

Simplified JMS Messaging Support for Liberty Profile

Lohitashwa Thyagaraj
WebSphere Messaging – Product Architect
IBM Master Inventor

Please Note

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

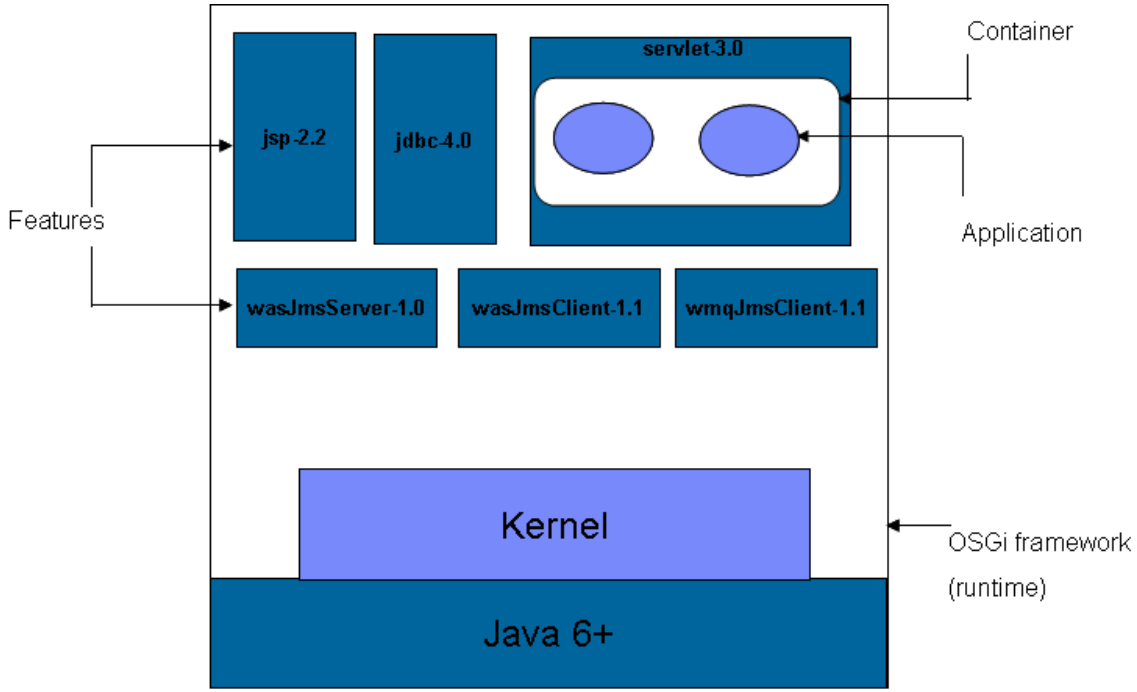
The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

Overview of Liberty messaging

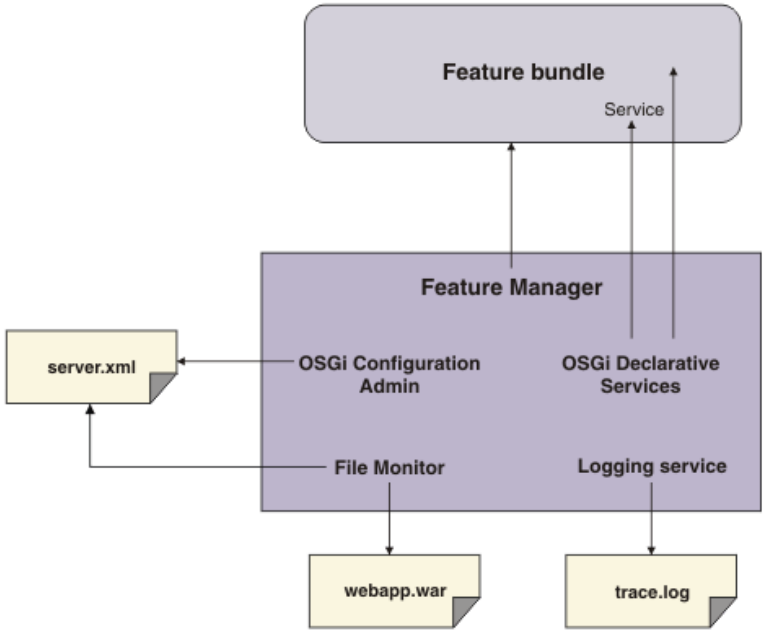
- Provides light weight JMS server
- JMS 1.1 compliant
- Composable, flexible and dynamic messaging service
- Supports WebSphere MQ as the messaging provider
- Supports Message Driven Bean (MDB) deployment
- Interoperability with full profile WebSphere Application Server
- Supports transactions, security and persistence

Liberty profile : Architecture



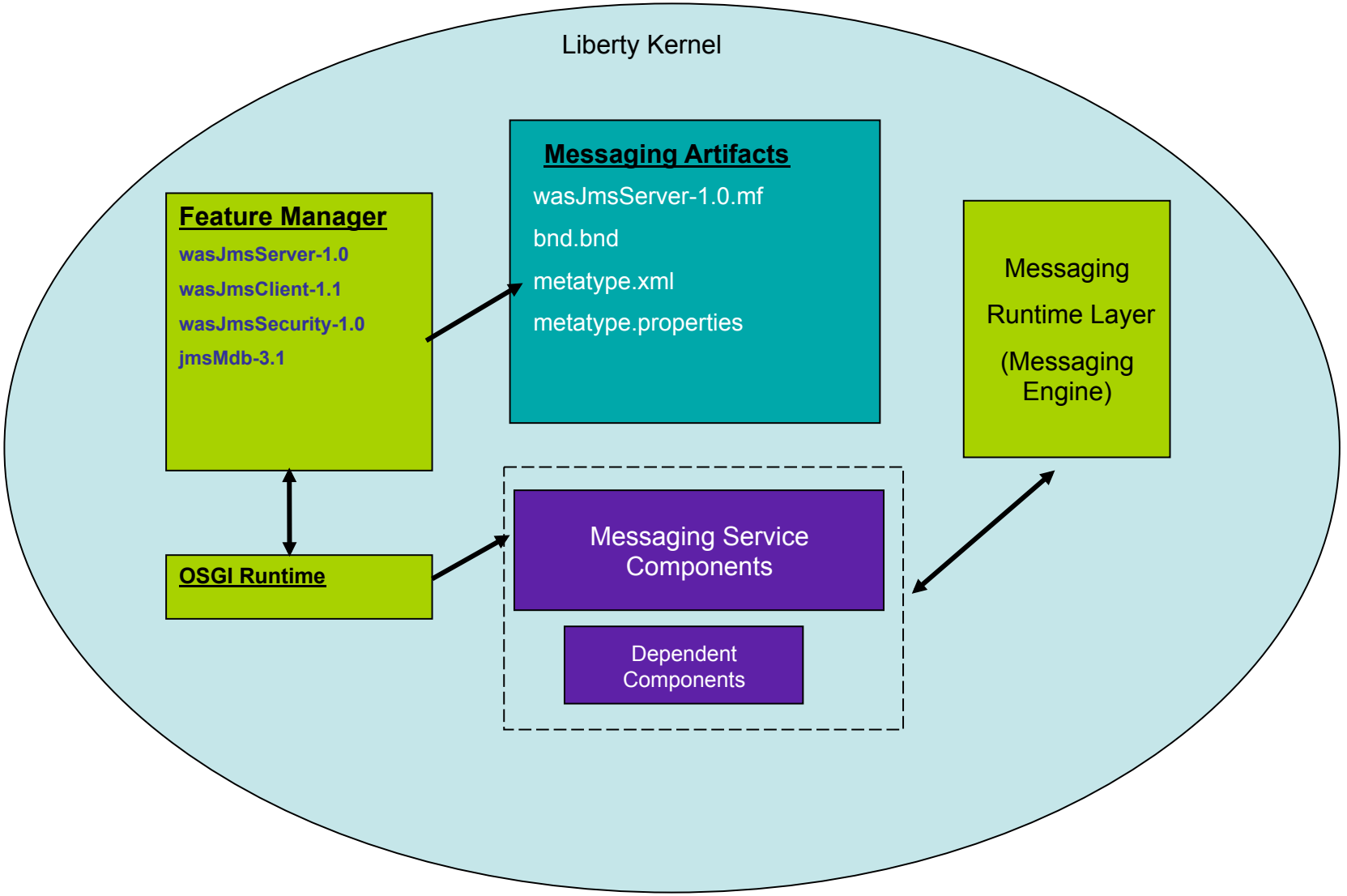
- The Liberty profile is a highly composable and dynamic runtime environment
- OSGi services are used to manage component lifecycles, and the injection of dependencies and configuration
- The server process comprises a single JVM, the Liberty kernel, and any number of optional features.
- The feature code and most of the kernel code runs as OSGi bundles within an OSGi framework.

Liberty profile : Kernel

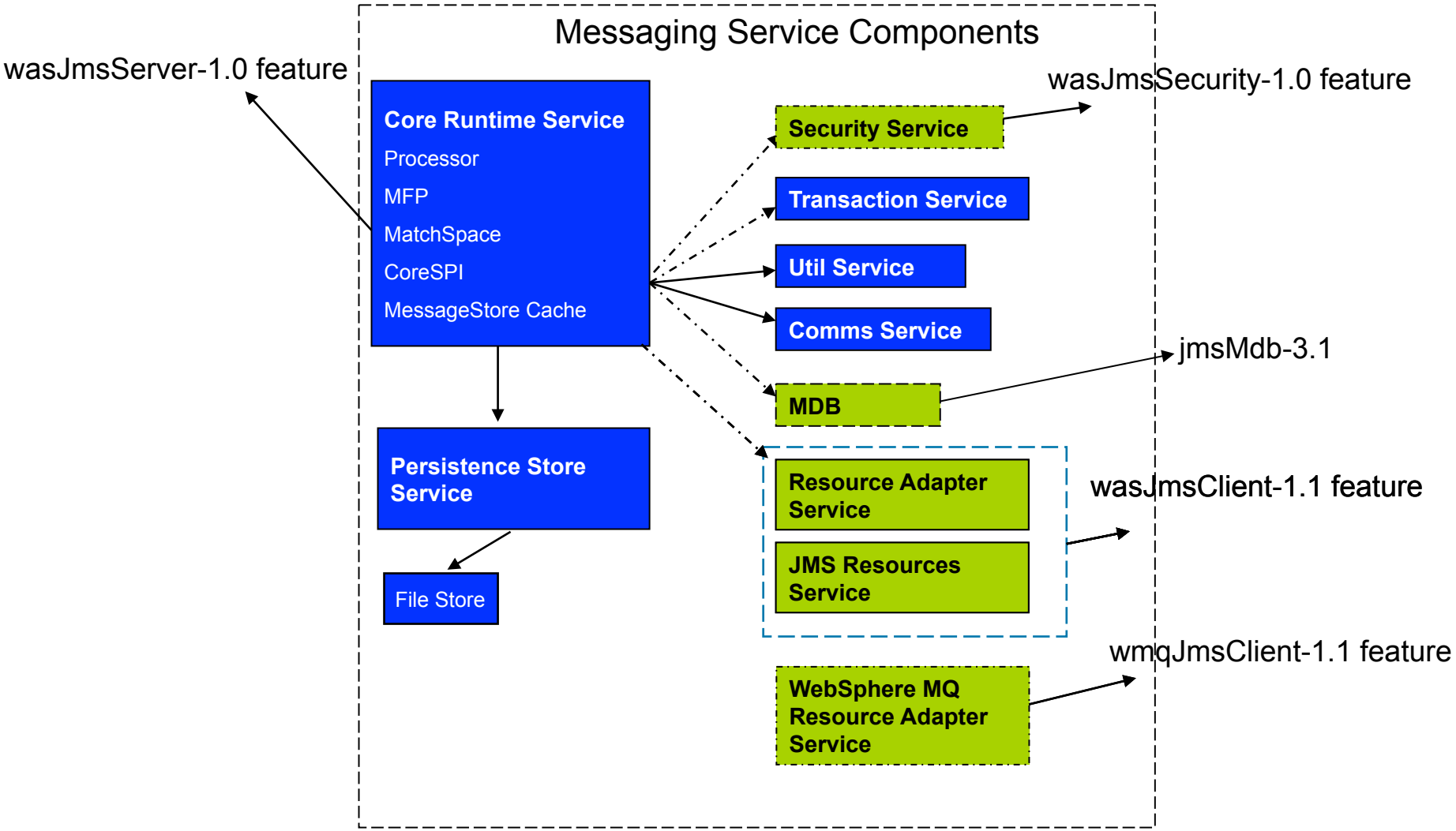


- Features are specified in the system configuration files that are the server.xml file and any other included files
- The server configuration files populate the OSGi Configuration Admin service, which injects the feature configuration into the feature manager service
- The feature manager maps each feature name to a list of bundles that provide the feature. The bundles are installed into the OSGi framework and started
- The feature manager responds to configuration changes by dynamically adding and removing features while the server is running

Liberty embedded JMS Server



Liberty messaging - Architecture

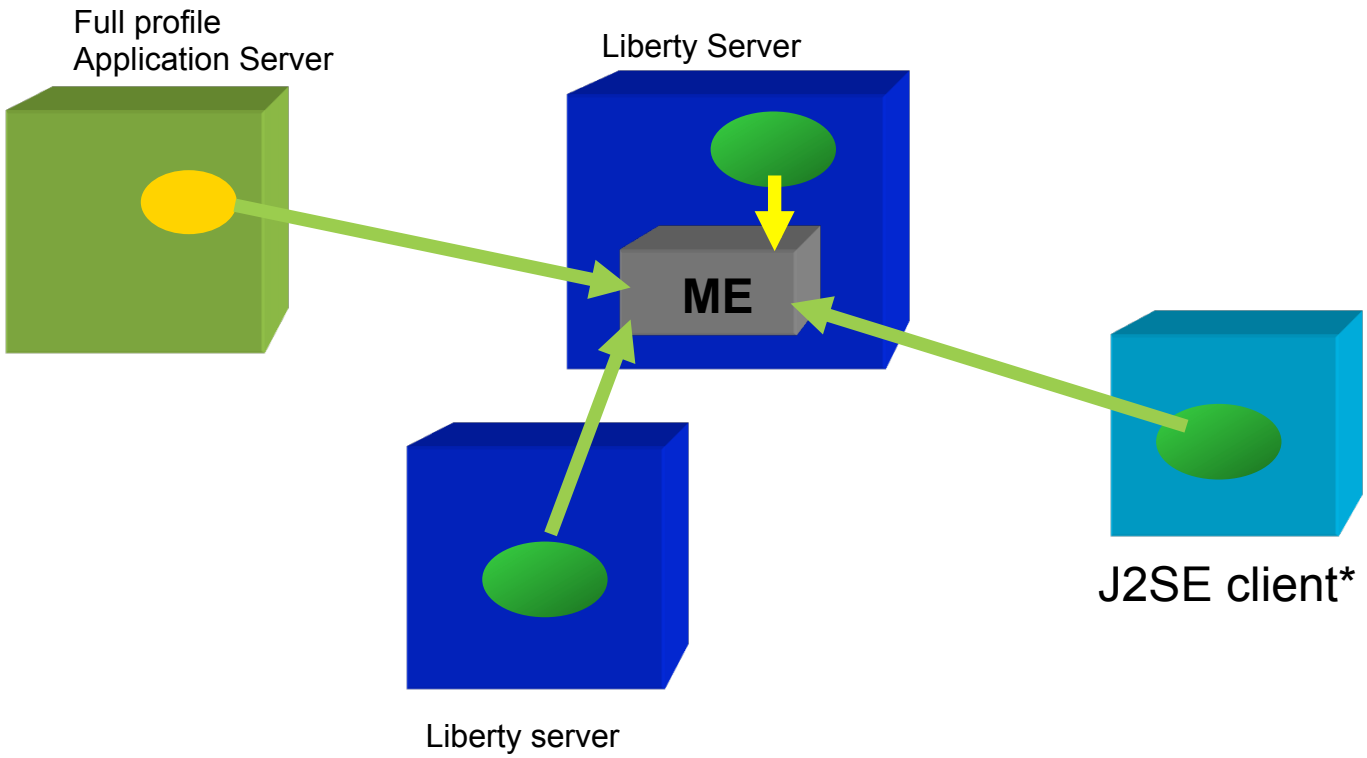


SIBus v/s Liberty Messaging footprint

SIBus	Liberty Messaging
Single Jar - com.ibm.ws.sib.server.jar	Multiple Bundles & Service
SIBus jar size – 12.8 MB	Core runtime size < **5MB**
Resource Adapter size sibc.jmsra.rar – 3.8 MB WebSphere MQ RA – 8.3 MB	Liberty Messaging Resource Adapter Bundle size – 1.5 MB
Tightly coupled	Loosely coupled

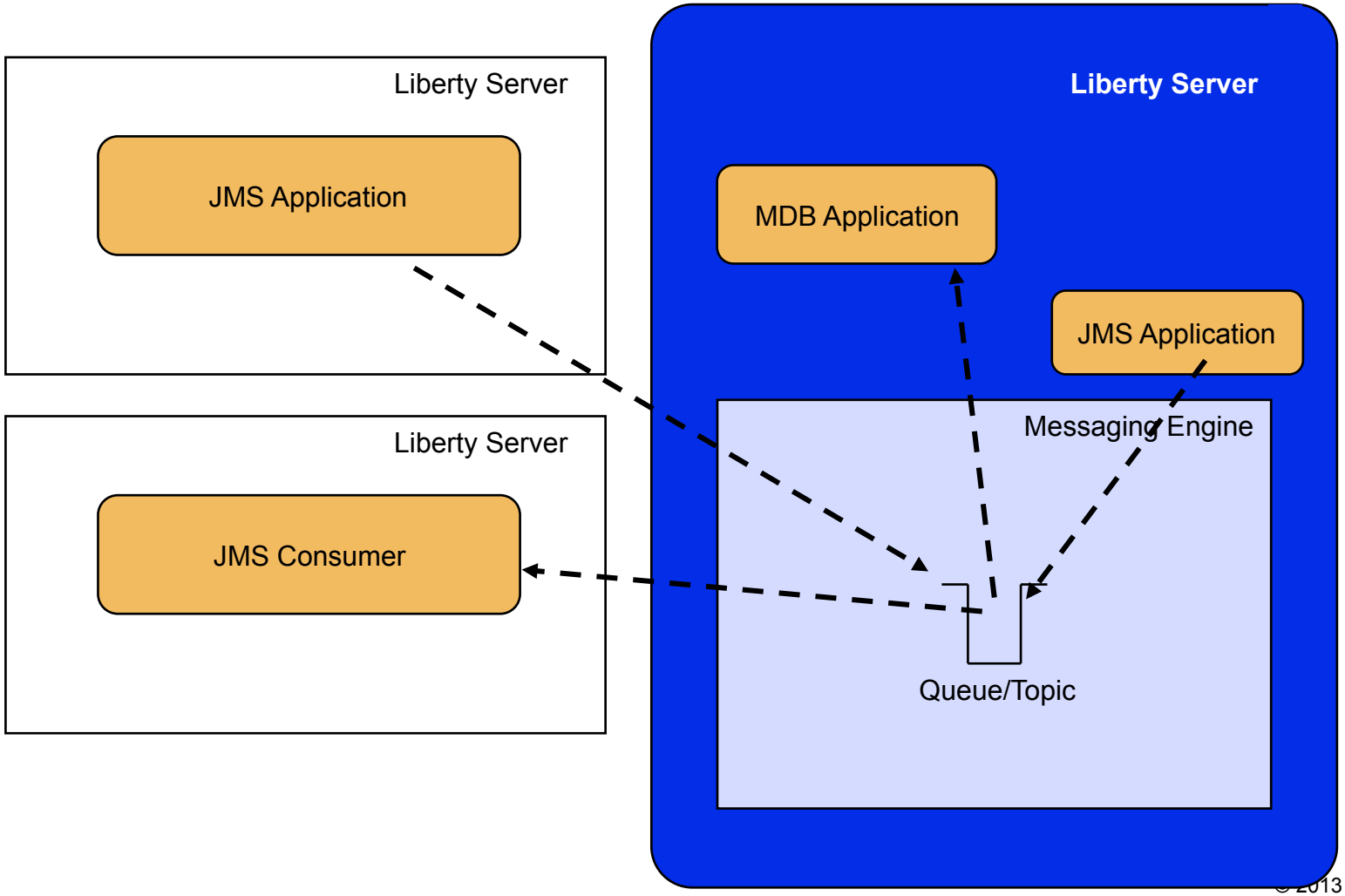
Application Deployment

- An application can run:
  - in a liberty profile that is hosting the messaging engine
  - in a liberty profile that is not hosting the messaging engine
  - in a full profile application server
  - in a J2SE client
- An application always connects directly to the messaging engine



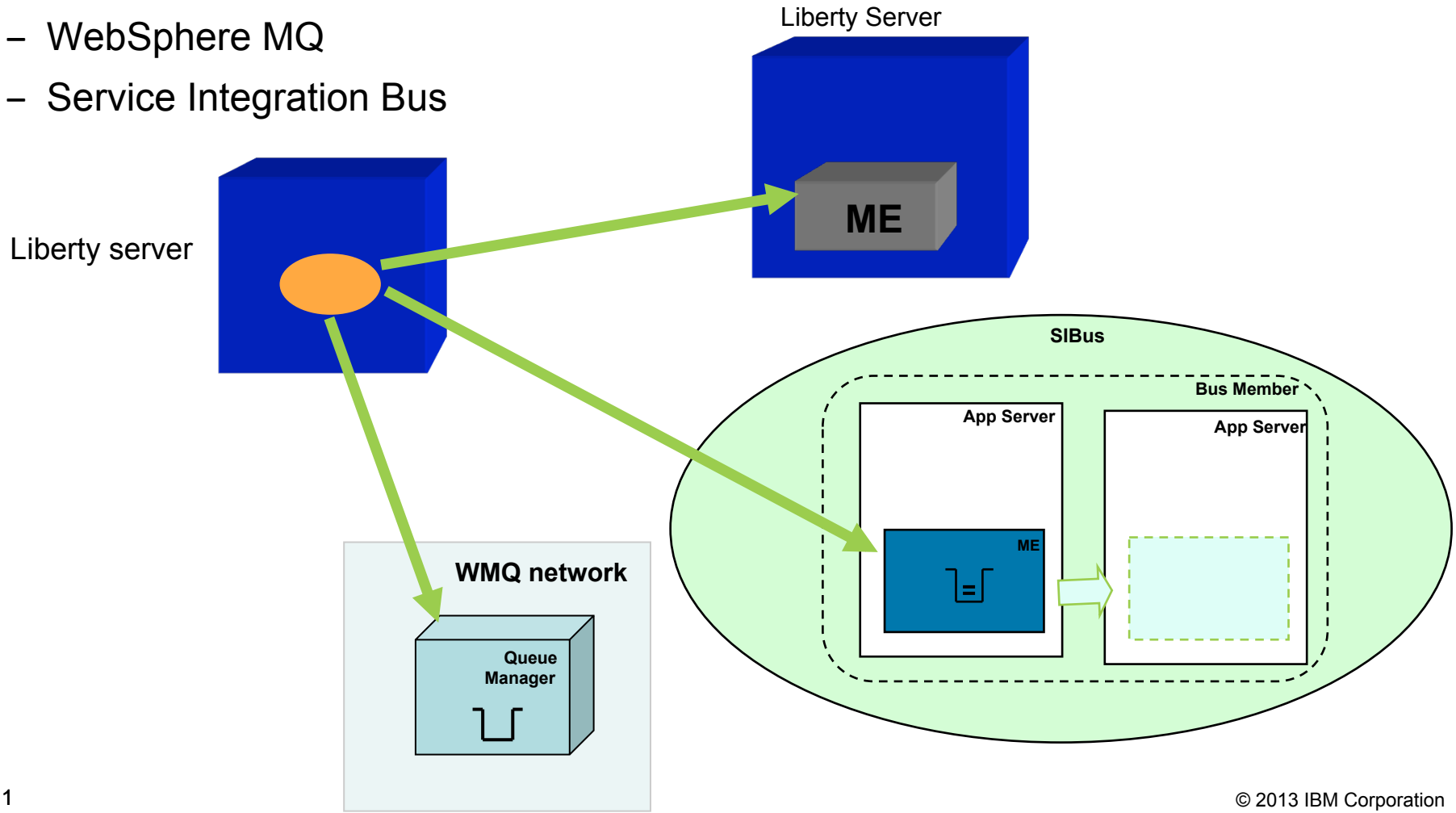
Message Handling

•In liberty messaging, all messages are sent and received from a messaging engine that localizes the destination



Liberty JMS provider support

- Liberty provides first class support for the following JMS providers
  - Liberty embedded JMS server
  - WebSphere MQ
  - Service Integration Bus



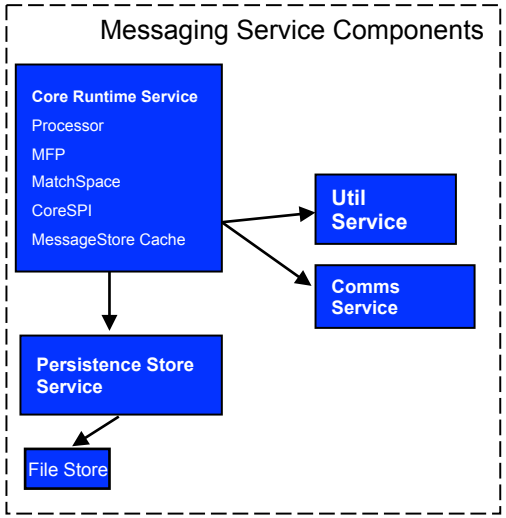
Liberty messaging features

- **wasJmsServer-1.0**
  - Instantiates the messaging engine runtime
- **wasJmsClient-1.1**
  - Provides client libraries and defining the JMS resources
- **wasSecurity-1.0**
  - Enables messaging engine to be secured
- **wmqJmsClient-1.1**
  - Enables connectivity to WebSphere MQ
- **jmsMdb-3.1**
  - Provides capabilities to configure and deploy MDB within Liberty

JMS Server (wasJmsServer-1.0)

- Provides the runtime capabilities for messaging server
  - Manages connections, persistence, transactions, security
- Flexible, Dynamic and Light weight
- Allows creation of destinations and managing authorization for messaging resources

```
<server description="Server with JMS">
    <featureManager>
        <feature>servlet-3.0</feature>
        <feature>wasJmsServer-1.0</feature>
    </featureManager>
</server>
```



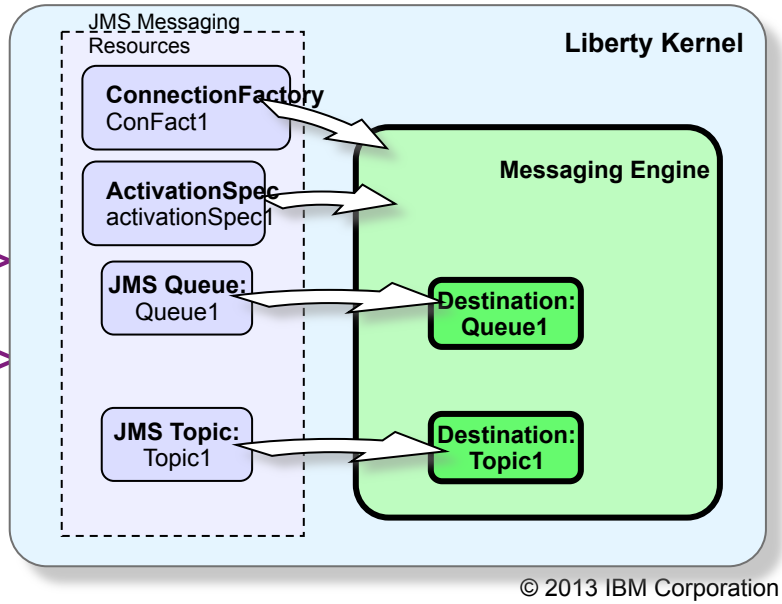
JMS Server (wasJmsServer-1.0) – Contd..


```
<server description="Server with JMS">
    <featureManager>
        <feature>servlet-3.0</feature>
        <feature>wasJmsServer-1.0</feature>
    </featureManager>
    <messagingEngine id="defaultME">
        <queue id="CreditQ" receivedAllowed="false"
                maxMessageDepth="1000"/>
        <topicSpace id="Sports.Topic.Space"/>
        <fileStore id="fileStore" logFileSize="20" fileStoreSize="100"/>
    </messagingEngine>
</server>
```

JMS Client (wasJmsClient-1.1)

- Provides the definition of the JMS Resources and registers the objects in the repository
  - Connection Factory (jmsConnectionFactory)
  - Queue Connection Factory (jmsQueueConnectionFactory)
  - Topic Connection Factory (jmsTopicConnectionFactory)
  - Activation Spec (jmsActivationSpec)
  - JMS Queue (jmsQueue)
  - JMS Topic (jmsTopic)

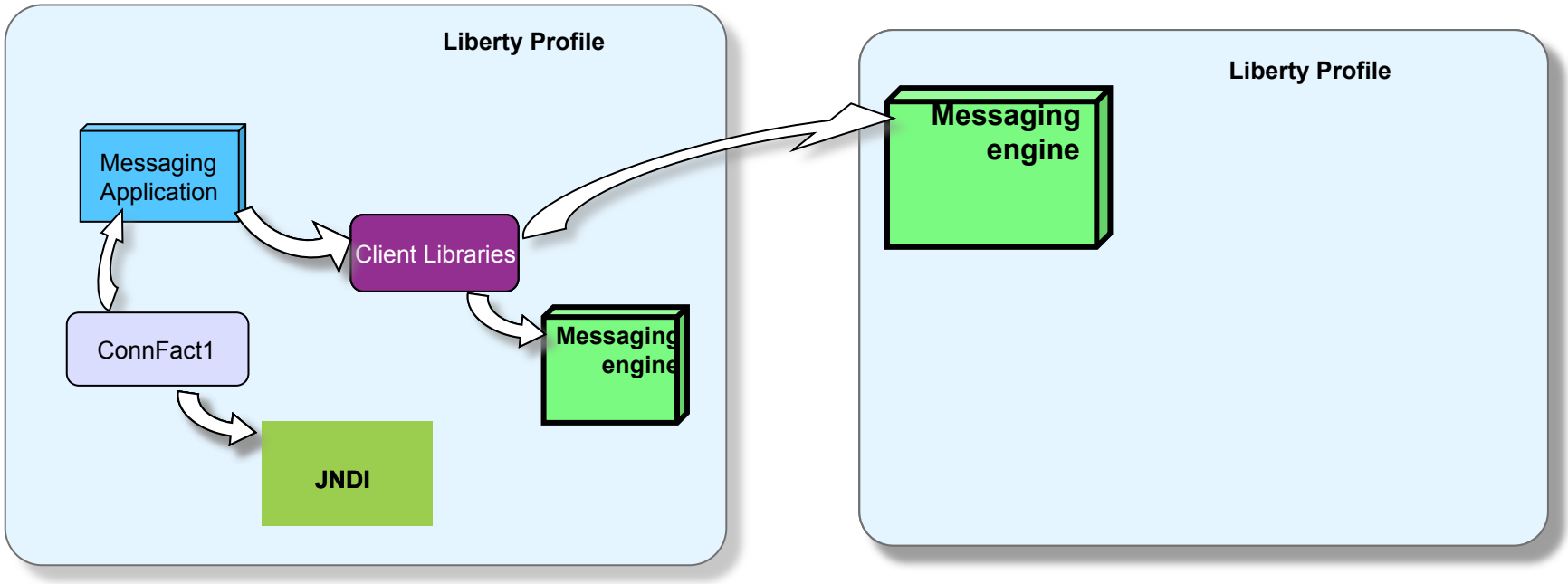
```
<featureManager>
        <feature>servlet-3.0</feature>
        <feature>wasJmsServer-1.0</feature>
        <feature>wasJmsClient-1.1</feature>
        <feature>jndi-1.0</feature>
</featureManager>
```



JMS Client (wasJmsClient-1.1)

- Provides the client libraries required for the JMS application to interact with the messaging engine

- Liberty supports two ways of application connectivity
  - IN-MEMORY
  - TCP/IP



JMS Client remote connectivity

•JMS applications can either connect on a secured or non-secured port

• By default JMS server listens for incoming data on 7276 (unsecured) and 7286 (secured). Default ports can be over-ridden by using the ***wasJmsEndPoint*** option in server.xml

```
<featureManager>
        <feature>wasJmsServer-1.0</feature>
</featureManager>
<wasJmsEndpoint id="InboundJmsEndpoint" host="*"
                wasJmsPort="9011"
                wasJmsSSLPort="9100" />
<messagingEngine id="defaultME">
        <queue id="libertyQ"/>
</messagingEngine>
```

JMS Client remote connectivity

- By default a JMS client application tries to connect to port 7276 (unsecured) and 7286 (secured)
- Use **remoteServerAddress** option in the connection factory to make JMS application connect to a remote messaging engine

```
<jmsQueueConnectionFactory jndiName="myQCF"
connectionManagerRef="ConMgr4">

      <properties.wasJms

       remoteServerAddress="lserver.ibm.com:9011: BootstrapBasicMessaging"/
>

</jmsQueueConnectionFactory>
```

Securing the JMS connections

- JMS client connections can be made secured by using custom chains and using SSL
- Use the **wasJmsOutbound** option to specify custom outgoing connection properties

```
<wasJmsOutbound id="jmsSecureCustomChain" useSSL="true">
        <sslOptions sslRef="CustomSslOptions" />
 </wasJmsOutbound>
<jmsQueueConnectionFactory jndiName="jms/QCF"
connectionManagerRef="ConMgr6">
<properties.wasJms remoteServerAddress="localhost:
7286:jmsSecureCustomChain" />
</jmsQueueConnectionFactory>
```

Securing the JMS connections (Contd..)

Server side

```
<ssl id="CustomSslOptions" keyStoreRef="defaultKeyStore"/>

<keyStore location="serverKey.jks" id="defaultKeyStore"
password="key123" />

<wasJmsEndpoint id="InboundJmsEndpoint" host="*"
        wasJmsPort="7276"
        wasJmsSSLPort="7286" >
        <sslOptions sslRef="CustomSslOptions"/>
</wasJmsEndpoint>
```

Securing the JMS connections (Contd..)

Client side

```
<ssl id="CustomSslOptions" keyStoreRef="defaultKeystore"/>

<keyStore id="defaultKeystore" location="clientKey.jks" type="JKS"
password="key123"/>

<wasJmsOutbound id="jmsSecureCustomChain" useSSL="true">

        <sslOptions sslRef="CustomSslOptions" />

 </wasJmsOutbound>


 <jmsQueueConnectionFactory jndiName="jms/QCF"
connectionManagerRef="ConMgr6">

    <properties.wasJms userName="persona99" password="ppersona99"
remoteServerAddress="localhost:7286:jmsSecureCustomChain" />

   </jmsQueueConnectionFactory>
```

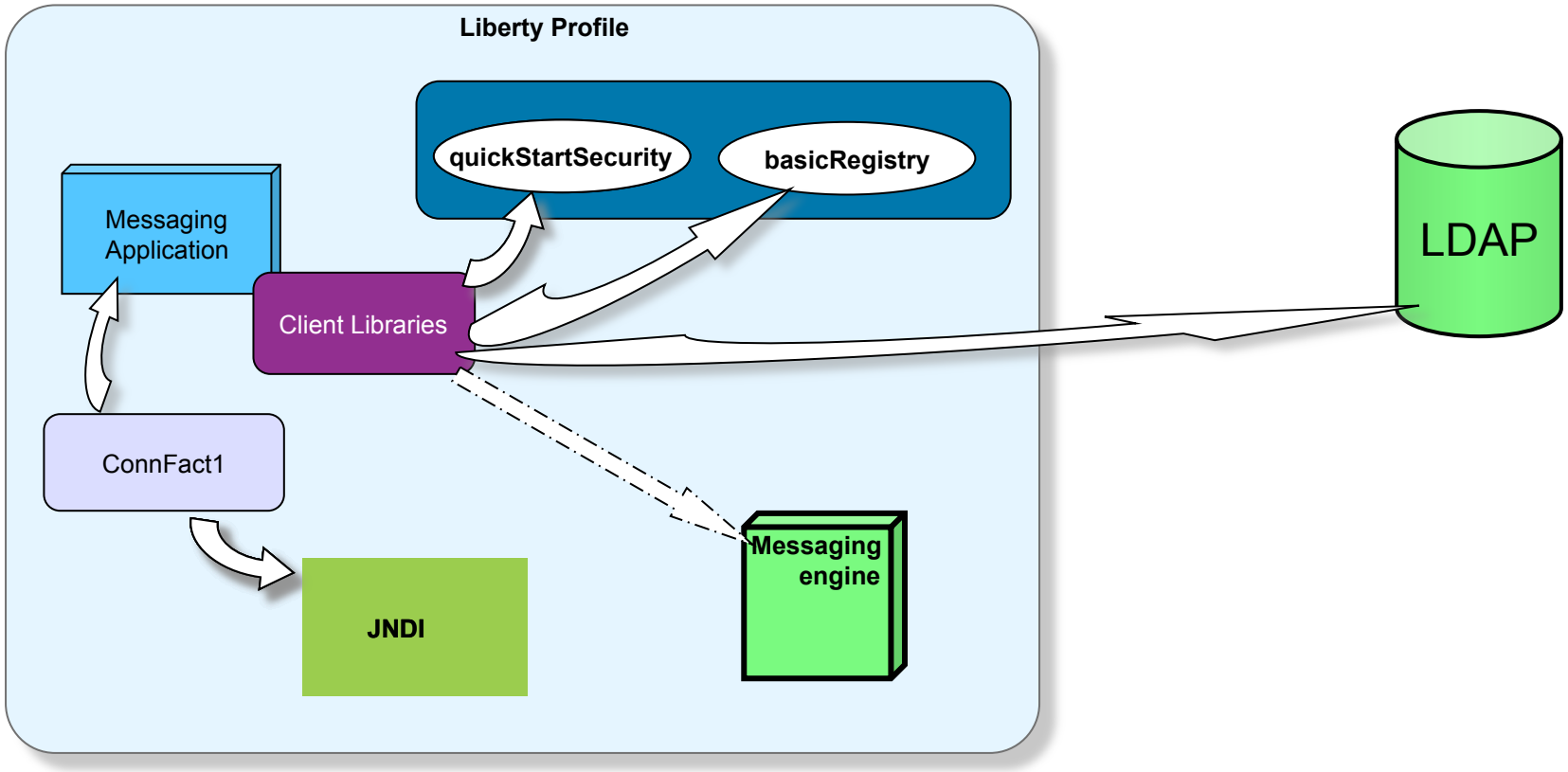
Securing Liberty Messaging (wasJmsSecurity-1.0)

- Enables the messaging engine to be secured
- Authentication
  - Validates if user has permission to *connect* to the messaging engine
- Authorization
  - Validates if user has sufficient *roles/permissions* to access the messaging resources (queues/topics)

```
<featureManager>
        <feature>servlet-3.0</feature>
        <feature>wasJmsServer-1.0</feature>
        <feature>wasJmsClient-1.1</feature>
        <feature>wasJmsSecurity-1.0</feature>
    </featureManager>
```

JMS Security - Authentication

- Liberty supports 3 types of user registries
  - quickStartSecurity : Simplest and easiest to configure
  - basicRegistry : Provides groups and member creations
  - ldapRegistry : Connectivity to external ldap registries



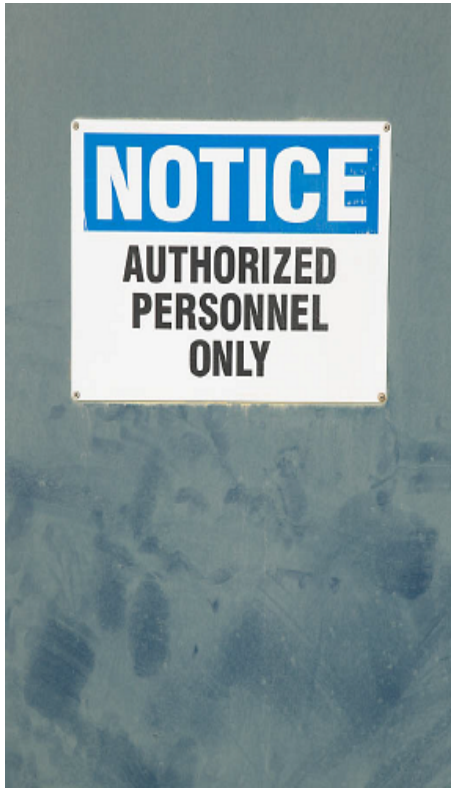
JMS Security - Authentication

- Different way of defining authentication information
  - Application
  - Connection Factory (Component)
  - AuthDat

JMS Security - Authorization

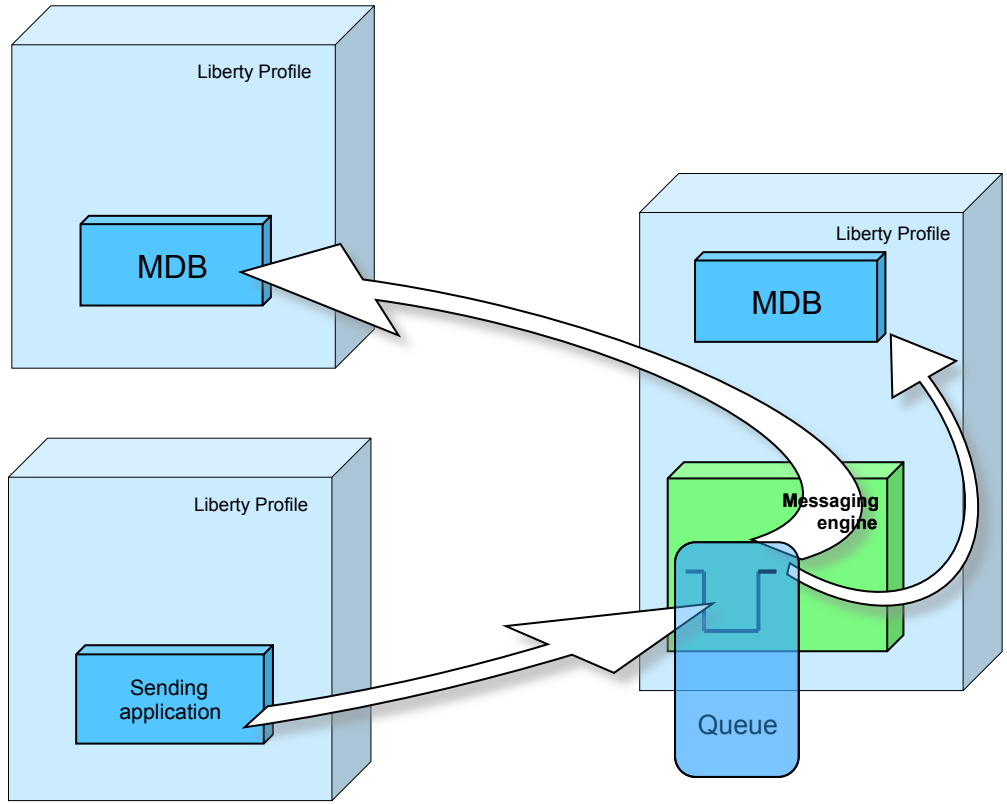
- Access to messaging resources is limited based on roles and credentials
- Only authenticated users will be authorized to access the resources
- Unless specified default access can be “allow all” or “deny all”

```
<messagingSecurity>
    <role name="developer">
        <queuePermission name="Q1">
            <action>ALL</action>
        </queuePermission>
        <user name="libertyUser"/>
    </role>
</messagingSecurity>
```



Message Driven Beans (jmsMdb-3.1)

- Liberty supports deployment of MDB's to asynchronously consume messages from messaging engine
- Supports both *Point to Point* and *Publish/Subscribe*
- Does not require EJB container to deploy MDB



Message Driven Beans (jmsMdb-3.1)

- MDB can be configured within the web project itself
- MDB utilizes Activation specifications to communicate with the messaging engine

```
<server description="Server with JMS">
    <featureManager>
        <feature>wasJmsServer-1.0</feature>
        <feature>wasJmsClient-1.1</feature>
        <feature>jmsMdb-3.1</feature>
    </featureManager>
</server>

<jmsActivationSpec id="jms/ActSpec" jndiName="jms/ActSpec" >
        <properties.wasJms destinationRef="jms/Q"/>
 </jmsActivationSpec>

<jmsQueue id="jms/Q" jndiName="jms/Q" >
        <properties.wasJms queueName="Q1"/>
 </jmsQueue>
```

Message Driven Beans (jmsMdb-3.1)

Liberty provides flexibility in the way MDB's can be configured and deployed

- Direct activation specification usage
- Annotation
-

server.xml

```
<jmsActivationSpec id="JMSSample/JMSSampleMDB">
```

```
@MessageDriven(
        name = "JMSSampleMDB", activationConfig = {
```

server.xml

```
<jmsActivationSpec id="PriceChangeAS">
      <properties.wasJms destinationRef="jms/TriggerQ" /> </jmsActivationSpec>
<jmsQueue id="jms/TriggerQ" jndiName="jms/TriggerQ">
      <properties.wasJms queueName="Q1"/>
</jmsQueue>
```

ibm-ejb-jar-bnd.xml

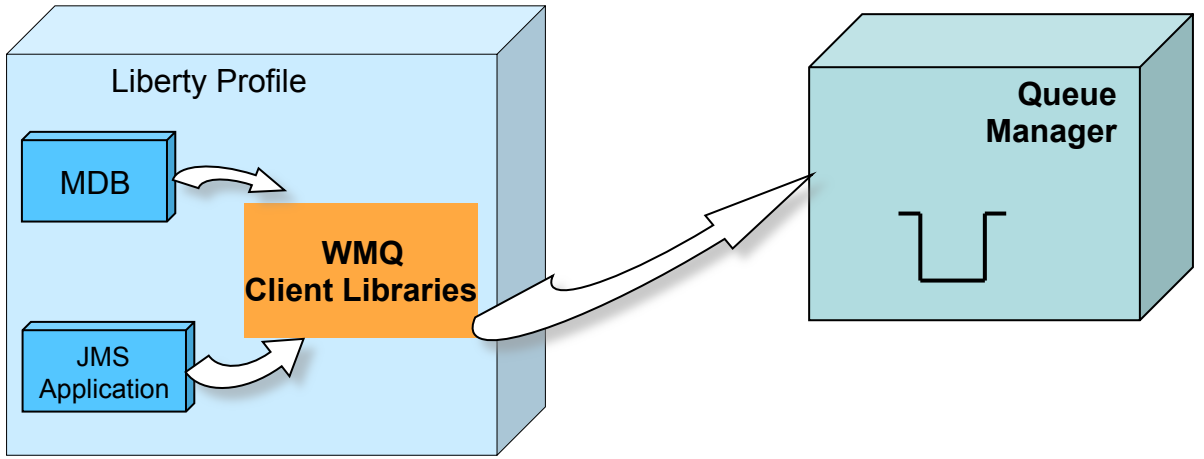
```
<message-driven name="PriceChangeMDBBean">
       <jca-adapter activation-spec-binding-name="PriceChangeAS" destination-
                    binding-name="jms/TriggerQ" />
</message-driven>
```

WebSphere MQ Support (wmqJmsClient-1.1)

- Liberty supports configuring WebSphere MQ as the external JMS messaging provider

- JMS Applications running within Liberty profile can communicate with WMQ to perform both
  - Point to Point
  - Publish/Subscribe

- Supports both type of connectivity
  - BINDING mode
  - CLIENT mode



WebSphere MQ Support (wmqJmsClient-1.1)

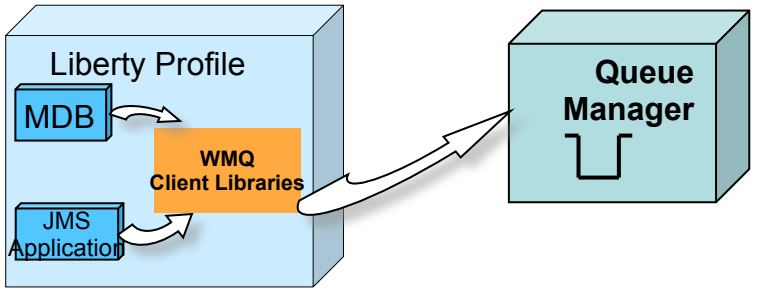
- WMQ client libraries are available to applications using the wmqJmsClient-1.1 feature

```
<featureManager>
   <feature>wmqJmsClient-1.1</feature>
<featureManager>
```

- Configuration very similar to liberty messaging configuration

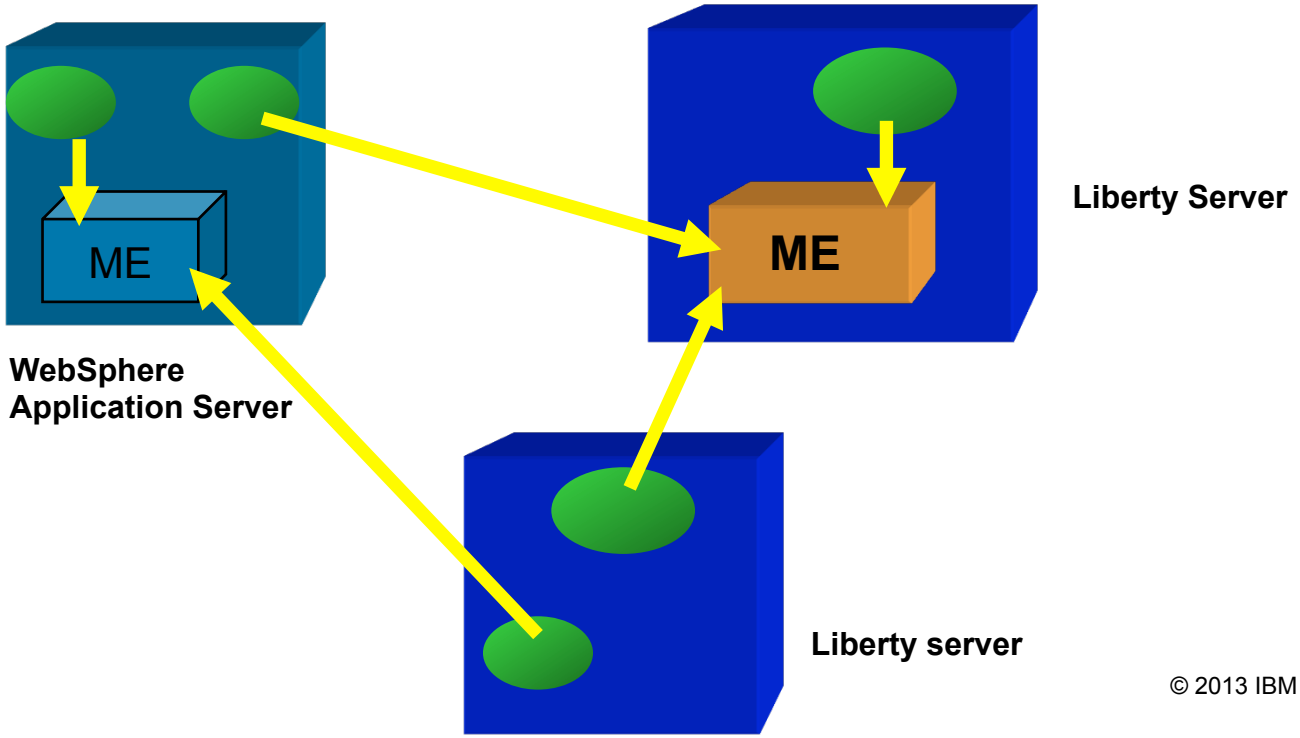
```
<jmsConnectionFactory id="jms/conFac connectionManagerRef="ConMgr1">
  <properties.wmqJms host="wmqserver.in.ibm.com port="1414"/>
</jmsConnectionFactory>

<jmsQueue id="eis/queue1" jndiName="eis/queue1">
   <properties.wmqJms   baseQueueName="Q1"
      baseQueueManagerName="qm1"/>
</jmsQueue>
```

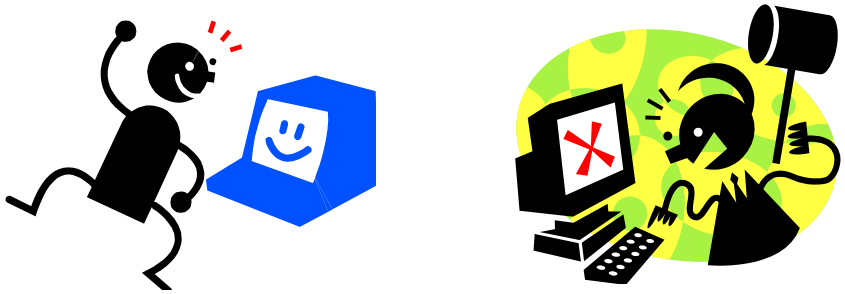


Interoperability between Liberty and Full Profile Application Server

- Application deployed in Liberty profile can connect to SIBus
- Application deployed in full profile WAS can connect to Liberty messaging
- Does not support ME to ME communication between Liberty and SIBus



And now the demo….



Summary

- Provides light weight single stand alone JMS server
- Composable, flexible and dynamic messaging service
- Supports WebSphere MQ as the messaging provider
- Supports Message Driven Bean (MDB) deployment
- Interoperability with full profile WebSphere Application Server

We love your Feedback!

Don't forget to submit your Impact session and speaker feedback!

- Your feedback is very important to us – we use it to improve next year's conference
- Go to the Impact 2013 SmartSite (http://impactsmartsite/com):
  - Use the session ID number to locate the session
  - Click the "Take Survey" link
  - Submit your feedback

Legal Disclaimer