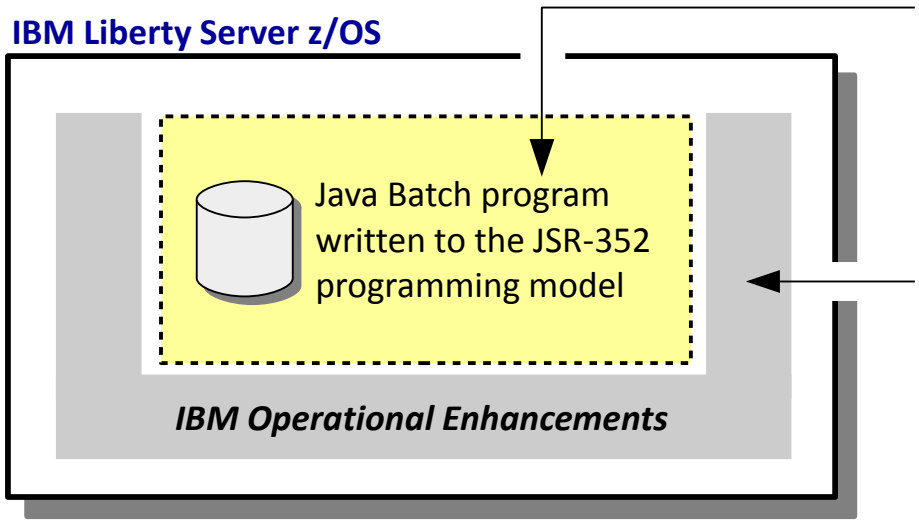
WebSphere Application Server

IBM Liberty
Java Batch Workshop

The Objective of the Workshop in One Chart



The JSR-352 Programming Model

We have a unit to cover the essentials of this, but we will *not* be going into a deep-dive on programming JSR-352 applications.

The Operational Enhancements

This is our focus.

The objective is to provide a good understanding of what these operational enhancements are and the value they provide.

Agenda ...

Workshop Agenda

Unit 1 – Introduction and Overview

Set context, establish terminology, level-set essential understanding

Unit 2 – Liberty, Server Creation and Setup + Lab

Cover the essentials of creating a Liberty z/OS server and enabling the basics of Java Batch

Unit 3 – JSR-352 Concepts + Lab

Review the JSR-352 specification and illustrate what it means for building a Java Batch program

Unit 4 – Job Submission and Control + Lab

Discuss batchManager, batchManagerZos, job logging, the AdminCenter, and batch events

Unit 5 – Multi-JVM Configuration + Lab

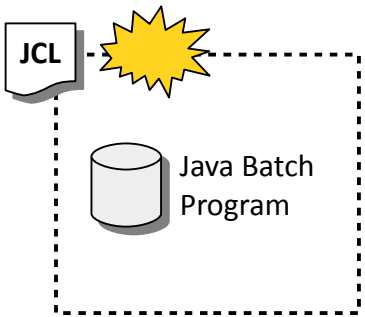
Review the setup and usage of the multi-JVM design for IBM Liberty Java Batch

Unit 6 – Java Batch Security + Lab

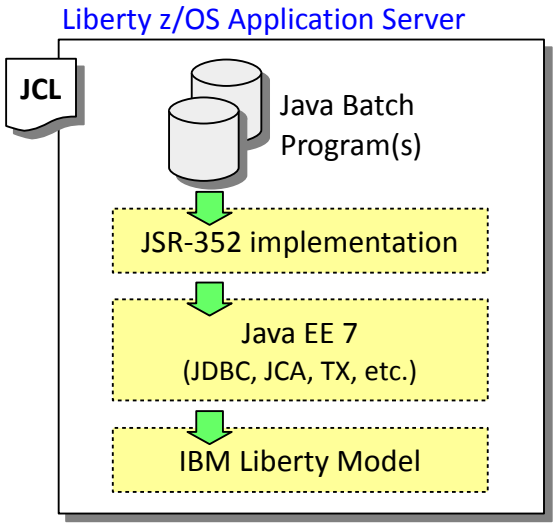
Review how to "harden" Java Batch security using SAF for authentication, authorization, and SSL

Launch JVM, or "persistent?" ...

Java Batch: Launch a JVM each time? Or use a "persistent" JVM model?



- This is what early Java Batch projects used, typically with a Java main() program invoked.
- Downside is you incur the overhead of JVM instantiation and tear-down each time.
- The JSR-352 programming model *can* be used with this model, but not with IBM's implementation of JSR-352.



- This is how IBM implemented the JSR-352 programming model*.
- It is available on all platforms, not just z/OS. z/OS is the focus of this workshop.
- The Liberty server stays active as long as you leave it up. The batch application is deployed into Liberty like any other application. It sits ready to be "submitted," at which time it will run and perform batch processing.
- *How* batch jobs are submitted and controlled is a focus of this workshop.

* JSR-352 is part of the Java EE 7 specification, so any Java EE 7 compliant runtime provides the programming interfaces. Therefore, the recent WAS traditional implementation with Java EE 7 has the JSR-352 compliant programming interfaces, but **not** the operational enhancements we will cover in this workshop. Only the Liberty implementation has those enhancements.

Why Java Batch?

People have varying reasons. Here are a few:

- On z/OS, zIIP offload from General Processors
- Availability of programming skills
- Desire to modernize certain batch processing
  Example: externalizing business rules to a rules engine
- Others?

Modernization Approaches

Java for New Batch Processes

Existing batch preserved; Java used for new batch processes that are identified

Targeted Re-Engineering of Existing

Existing batch processes are evaluated for potential re-engineering

Identified processes are modernized; move to Java *and* identify improvements in process

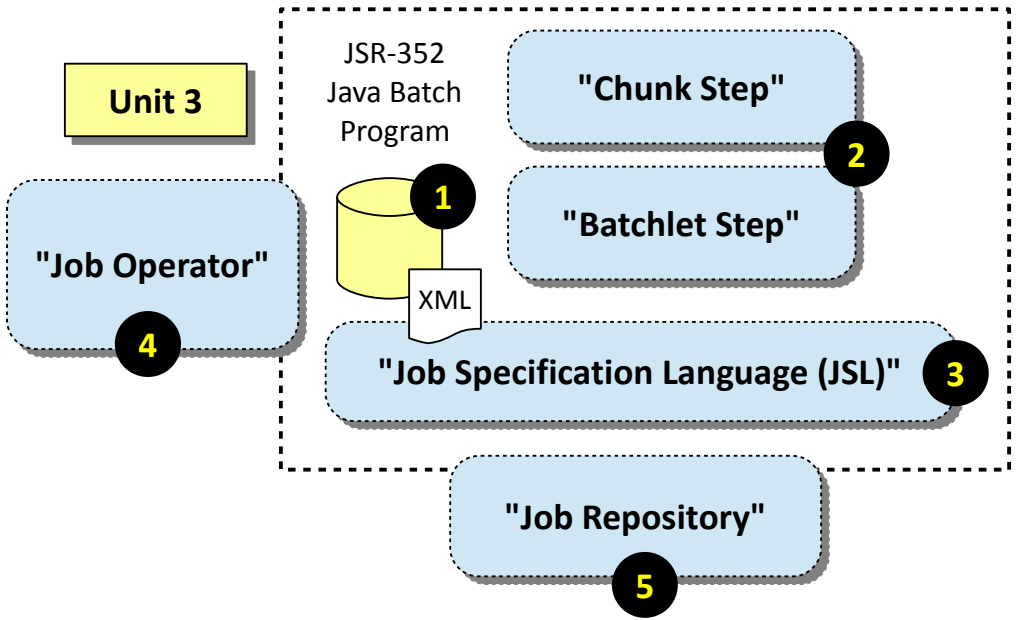
New workloads would be in Java as well.

Rip-and-Replace

Nobody seriously advocates this … the risk of disruption to the business is too great

High level of JSR-352 specification ...

Very High-Level Overview of the JSR-352 Programming Model



1. Java Batch Program

You write this based on the defined JSR-352 requirements. This is packaged as a servlet (WAR) and deployed into Liberty just like any other application would be deployed.

2. Job Step Programming Types

Two job step types defined:

Chunk: the looping model we most often associate with batch processing. This includes functions such as checkpointing, commits and rollbacks, and job restarts.

Batchlet: a simple "invoke and it runs" model. This is useful for non-looping functions such as file FTP steps.

3. Job Specification Language (JSL)

An XML specification to describe the batch job: the steps, the Java programs that implement the steps, and the flow of steps within the job.

4. Job Operator

This interface defines how to submit and control jobs. This workshop focuses on the IBM enhancements around this job operator definition.

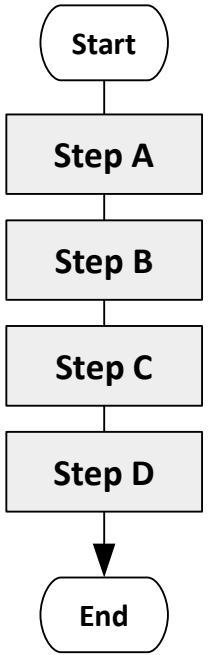
5. Job Repository

The specification calls for a repository to track job submissions and results, but leaves it to the vendor to implement. We'll use IBM DB2 z/OS in workshop.

Job step flow ...

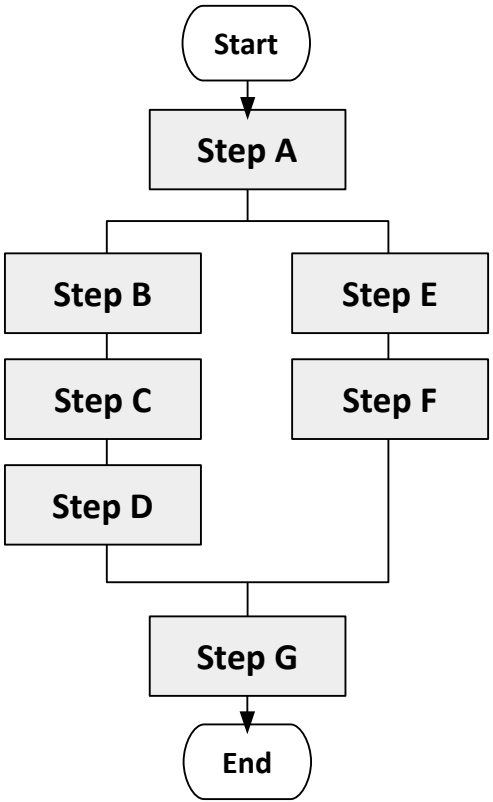
Job Step Flow Control

A simple sequential processing of steps:



This is what we commonly think of as "batch," and this may be what you do

You can also accomplish more sophisticated flows

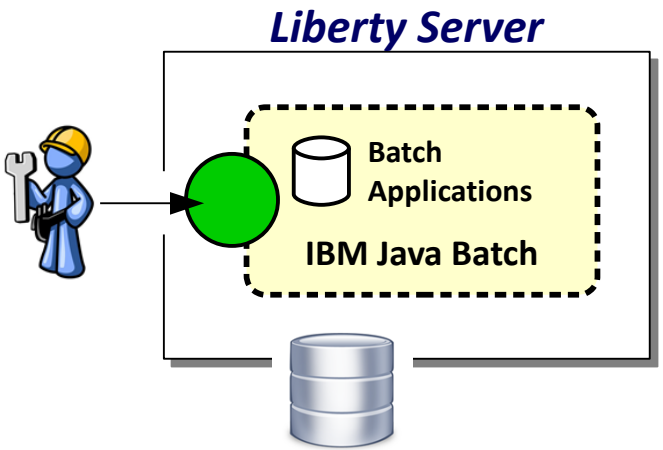


The JSR-352 specification has the ability to define simple or sophisticated step processing flows within a job.

The Job Specification Language (JSL) file is what spells all this out.

In Unit 3 we will explore the JSL definitions and the tooling that generated the file

The "JobOperator" Interface



The JobOperator interface is used to submit, monitor, and control Java batch jobs in the JSR-352 container.

The JSR-352 specification spells out a programming interface for JobOperator. The operational specifics beyond the programming interface is left up to the vendors implementing JSR-352.

With IBM WebSphere Liberty Batch you have two main options:

1. REST Interface

A REST interface is implemented that will process REST requests to the Liberty server to submit, monitor, and control Java batch jobs in the server. Any REST client will work provided the REST request is formatted properly.

2. Command Line Interface (CLI) clients

Two different command line clients are provided:

a) batchManager

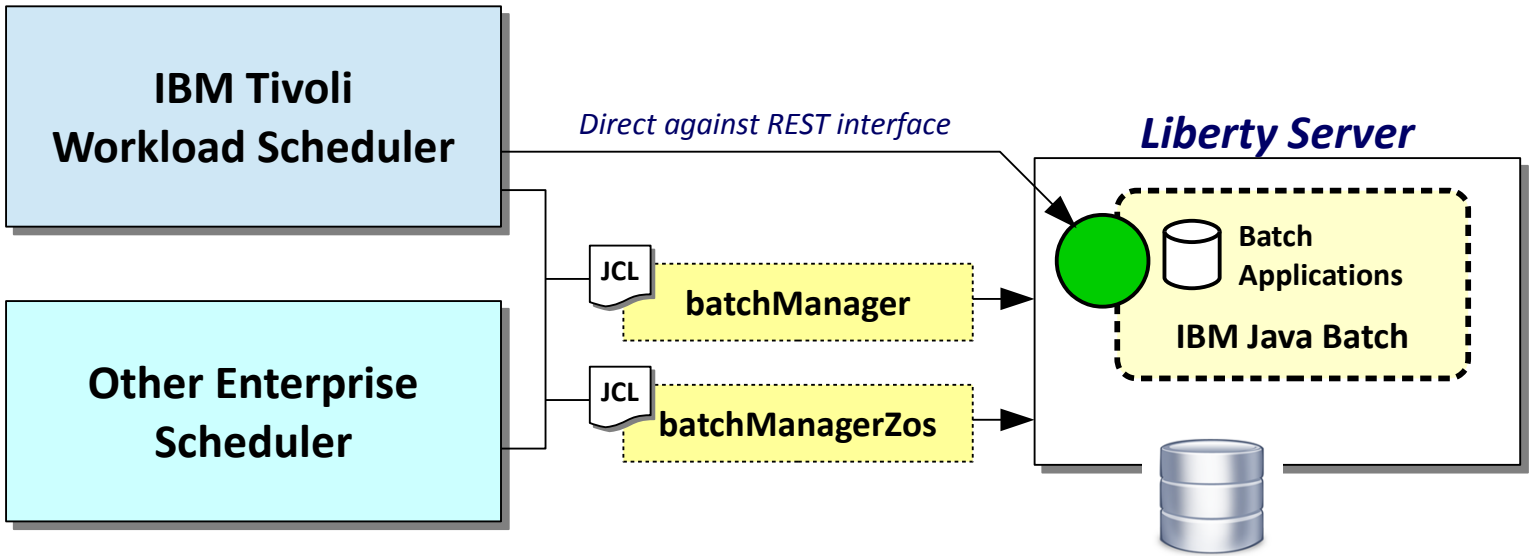
This is a Java-based client and uses the REST capabilities to manage jobs. This CLI may be run on the same OS as the Liberty server, or on any server with a network connection to the Liberty server running IBM Java Batch.

b) batchManagerZos z/OS Exclusive

This uses WOLA (a cross-memory mechanism) to communicate with the server to manage Java batch jobs.

More detail is provided in Unit 4 of this workshop

Integrating Enterprise Scheduler Functions

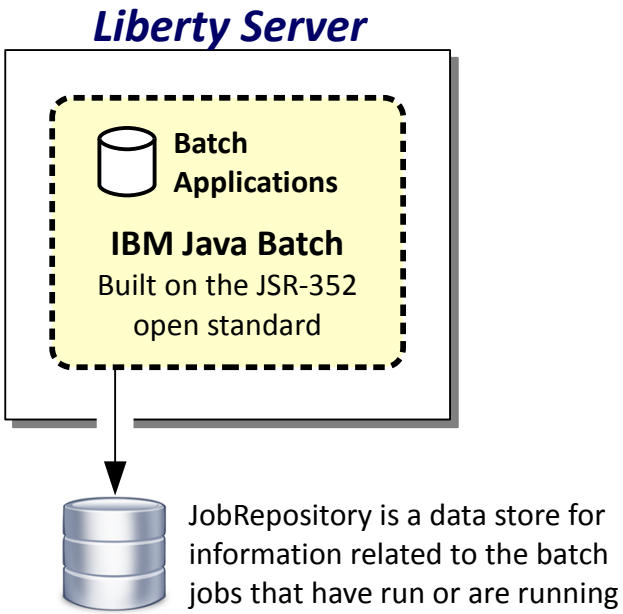


Key things to this:

- Be able to submit the Java Batch job inside the IBM Liberty Java Batch environment
- Hold completion of CLI invocation until Java Batch inside container completes
- Be able to retrieve the job log output if desired

We explore batchManager and batchManagerZos in Unit 4

The JSR-352 "JobRepository"



The JSR-352 specification only says a "Job Repository" is needed, but not how that is implemented or what layout the data takes. That is left up to the implementers of the specification.

With IBM WebSphere Liberty Batch you have two main options:

1. In-memory

The JobRepository is maintained in a *memory* data structure. Very easy to use as it requires no setup at all. Downside is the data goes away when the server stops. Great for development and ad-hoc testing; not good for anything more robust.

2. Persisted

The JobRepository is held in a persistent data store, such as:

a) File-based Derby database

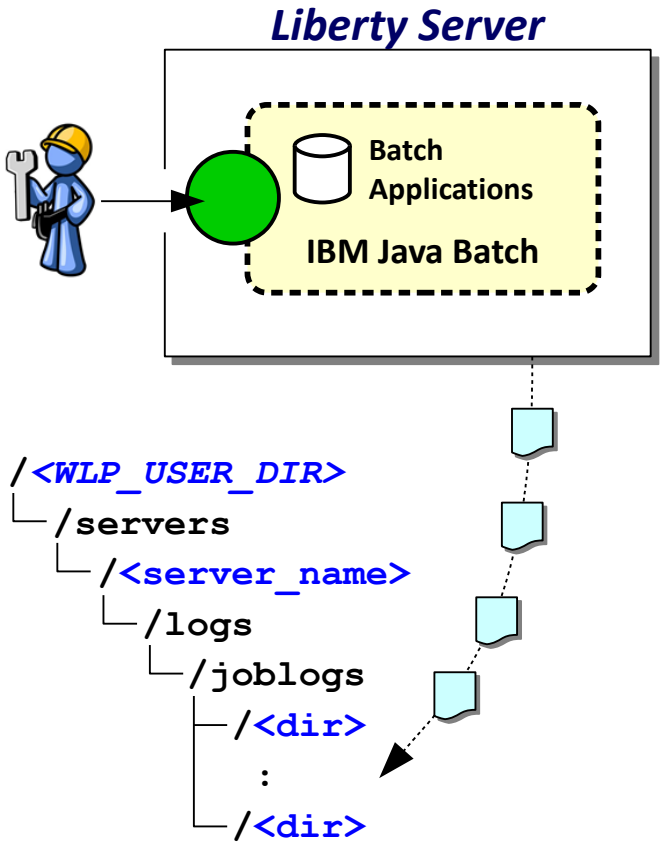
Job data survives a server stop and restart. Relatively simple to set up and use; you can do this without involving a DB admin. Good for development and testing, but not robust enough for production usage.

b) Vendor relational, such as IBM DB2

This is the approach we will focus on in this workshop. This provides the most robust persistence model. Provides for data sharing between Liberty servers, which is key to the "multi-JVM" topology design.

Job logging ...

Job Logging

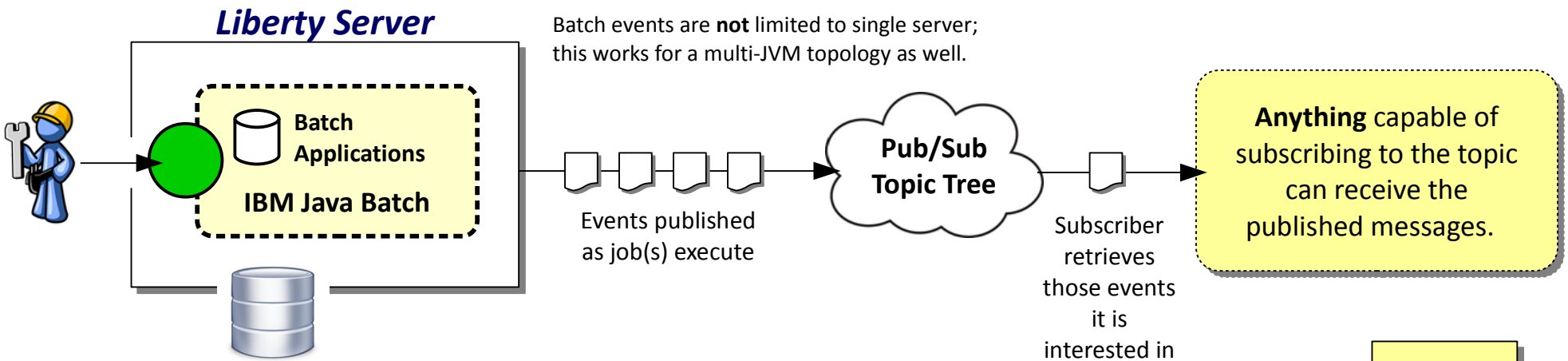


- **For each job that executes the server will write job log files to separate directories under /logs**
  The files are not all jumbled together; the job log tree is very orderly based on application name, date, and execution numbers
- **This is controllable by server**
  You can turn logging on or off on a server-by-server basis; you may also control log file size (if maximum exceeded it closes file and opens another job log part file).
- **Can be directed to a separate location using the Liberty WLP_OUTPUT_DIR variable**
  This gives you a way to direct output to a location where you can mount a large ZFS to hold verbose output
- **batchManagerZos: capture "as it happens"**
  This is a way to route to JES spool
- **Can retrieve job logs using REST interface**
  Job log parts are returned as JSON objects.
- **Can retrieve job logs using batch events**
  The jobLogPart topic contains a job's log parts
- **Can retrieve directly from the file system**
  They're just files in a directory, so this avenue is always available

Unit 4

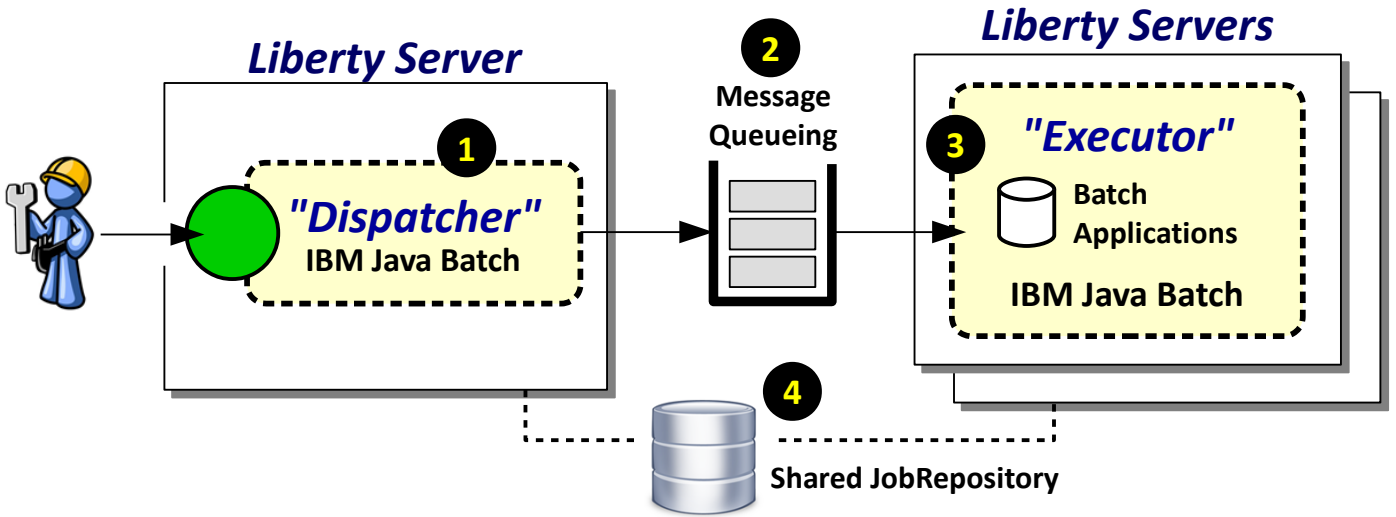
Batch events ...

Batch "Events" and Monitoring Job Activity



- **Can be turned on or off on a server by server basis**
  It's a relatively simple `server.xml` update that tells a server to publish events. It does require a properly configured JMS connection to a message queueing mechanism such as IBM MQ
- **Topic tree has events for many state changes in the life cycle of a job**
  Too many to list here, but what you would expect exists: job is queued, starting, executing, stopped, failed, etc.

The "Multi-Server" Topology Model



The role of dispatcher is separated from executor, and a queueing "pull" model placed between the two

Unit 5

1. Dispatcher Server

This server has no batch applications deployed. It is configured to be a "dispatcher" and will put a job submission message on the configured queue.

2. Message Queueing

This serves as the connection point between the servers. This can be any message queueing function supported by Liberty, including MQ.

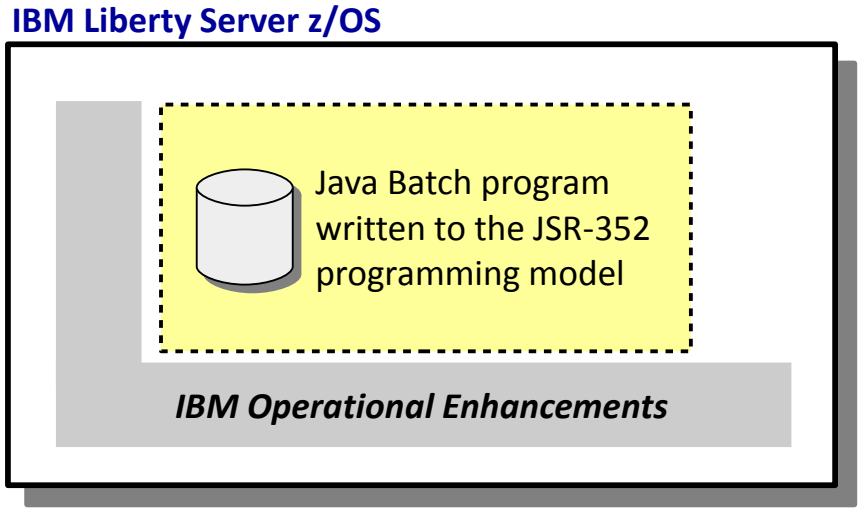
3. Executor Server(s)

These servers host the Java batch application. They use JMS activation specs to listen for and pick messages off the queue. They can employ "message selector" strings to pick certain job submission messages but not others.

4. Shared JobRepository

Key requirement: the JobRepository implementation must be shared between servers. DB2 z/OS can do this easily.

Summary



- Different motivations exist for considering Java for batch processing
- Different approaches exist for adopting Java for batch processing
- JSR-352 is the industry standard programming model for Java Batch
- JSR-352 is included with any Java EE 7 compliant runtime
- IBM added operational enhancements to JSR-352 in its Liberty runtime platform