IBM WebSphere Liberty Profile for z/OS

Java Batch with JSR 352

See the WP102544 Techdoc at ibm.com/support/techdocs for more

Batch Processing

In the early days of electronic computers, batch processing was the type of work that was most often performed. The mainframe computer would work through bulk data and perform whatever processing the program was designed to do.

Batch processing is *still* an important part of modern information technology. That is because some work is best suited to being performed in bulk. Batch processing was there in the early days; it's still here today; and it will still be here tomorrow.

What has changed is the *approach* to batch processing. Programming languages such as COBOL are still used, but increasingly Java is part of the discussion for batch processing as well.

Business Value of Java Batch

Some reasons cited for considering Java for batch include:

- *Modernization* – an effort to analyze and improve business process for the purpose of increasing productivity and decreasing cost. Batch processing with Java is often part of this analysis.
- *Skills* – Java skills are more readily available than are COBOL skills. The cost of developing and maintaining programs is a function of the availability of the programming skills required.
- *Mainframe offload* – on IBM z/OS Java workload is eligible for offload to specialty engines[1], which provides a financial benefit to running Java.

For these reasons and perhaps others, many are looking to Java as a programming language for batch processing.

Java Batch Before Open Standards

Java has been available for almost 20 years. In that time many approaches to processing batch with Java have emerged:

- *JVM launchers* – a Java Virtual Machine (JVM) is launched with a shell script or a mainframe tool such as BPXBATCH or JZOS and the Java program runs. All batch process handling is the responsibility of the batch program.
- *Vendor frameworks and runtimes* – these provide programming and functional services for batch programs to use. This allows batch programmers to focus on business logic while taking advantage of vendor-provided function.

Both approaches are in use today, and each serves a role for which it is particularly suited. However, as Java batch became more prevalent, the demand for an open standard emerged. That resulted in the open standard for Java batch, which is known as JSR 352.

Open Standard Java Batch – JSR 352

In 2011 a group was formed to study and design an open standard for Java batch processing. Representatives from many companies, including IBM[2], developed a draft standard. The initial release of the standard was released in 2013. The standard, known as JSR 352, is now included as part of the Java EE 7 open standard[3].

1 For example, the zAAP or zIIP processor.
2 Led by IBM.
3 IBM wrote the reference implementation for JSR 352 included with Java EE 7.

Going Beyond the Specification

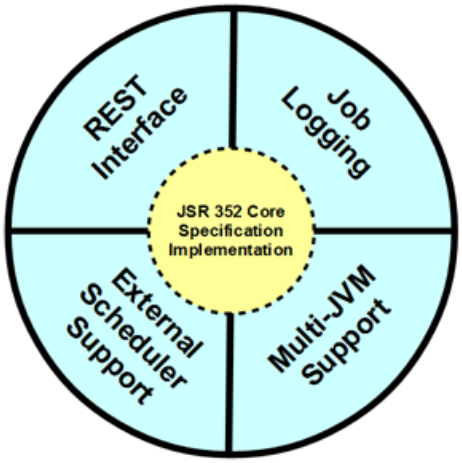
The JSR 352 standard is, in many ways, a *programming interface* standard. It makes reference to operational components, such as a data store for holding job information, but it does *not* mandate how those components are to be implemented. Vendors such as IBM are free to provide operational and runtime enhancements to the standard *as long as the specifications of the standard are met*.

IBM's JSR 352 implementation meets the JSR 352 specification requirements and provides operational enhancements IBM saw as important for JSR 352 to be ready for enterprise Java batch processing.

IBM's JSR 352 Implementation

IBM's initial implementation of JSR 352 was made available in June 2015 as part of Liberty Profile Version 8.5.5, Fixpack 6[4].

The IBM implementation of JSR 352 includes compliance to the specification requirements, as well as the funtional enhancements illustrated by the following diagram and described below:



Rest Interface

The JSR 352 specification calls for a "JobOperator," which is an interface to submit and manage batch jobs. The specification simply provides details on the *methods* of the interface, but not how it is to be implemented.

The IBM JSR 352 implementation running in a Liberty Profile server includes a RESTful interface to the JobOperator. This allows jobs to be submitted and managed by a client external to the server itself, as well as providing additional functions such as retrieving job logs or purging jobs. The REST interface also provides the ability to integrate the IBM JSR 352 runtime with external schedulers, as we describe below.

Job Logging

The IBM JSR 352 implementation provides a way to separate the Java batch job log from the server output. It also provides a mechanism to separate each job log from others, and organize by date and job execution. This makes easier the task of finding and archiving batch job logs.

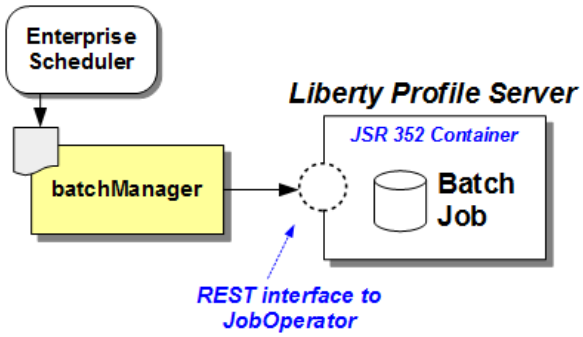
4 Often notated as simply 8.5.5.6. This applies to all operating system platforms supported by IBM Websphere Liberty Profile.

External Scheduler Support

Enterprise scheduler functions, such as IBM Tivoli Workload Scheduler, or CA-7 or Control-M[5] are often used to coordinate and control batch job submissions. Incorporating Java batch submission as part of enterprise batch submission is critical.

The JSR 352 specification does not speak to enterprise scheduler integration beyond the specification of the JobOperator. IBM's extension of the JSR 352 specification includes a mechanism to integrate with enterprise schedulers. It comes in two forms:

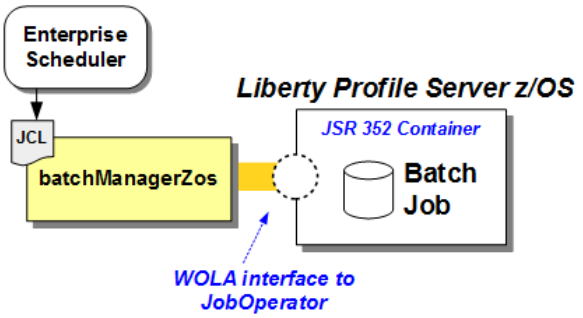
batchManager Command Line Utility



The batchManager command line utility is a Java-based tool that uses the REST interface provided by the IBM JSR 352 runtime. Enterprise schedulers may invoke the batchManager command line utility using shell script or .bat file. A "wait" parameter will keep batchManager active until the Java batch job ends. batchManager may be used to submit jobs on the same server platform or a remote server platform.

batchManagerZos Command Line Utility

On z/OS another command line utility is provided called batchManagerZos. This uses a cross-memory[6] call into the Liberty Profile server to submit and monitor jobs:



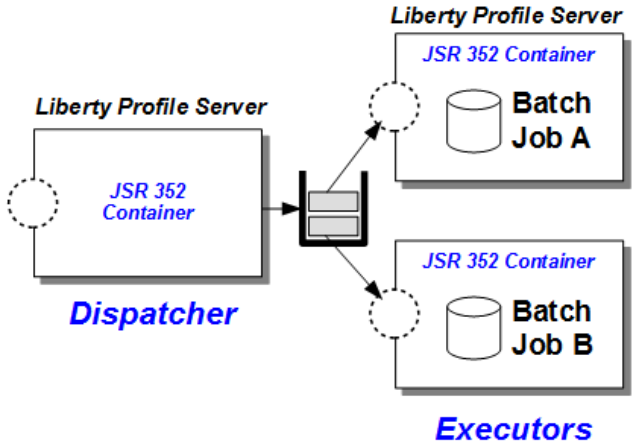
This also has a "wait" function so the submitted job that runs batchManagerZos remains active while the Java batch job runs in the Liberty Profile JSR 352 container.

batchManagerZos is a native program and requires no Java to operate. That means there is no JVM instantiation overhead associated with using batchManagerZos to integrate z/OS enterprise schedulers with IBM JSR 352 in Liberty Profile z/OS.

Multi-JVM Support

The objective of this feature is to separate job submission from job execution. That provides additional flexibility to manage the Java batch workload in your environment.

This feature uses JMS queuing (either MQ or the service integration messaging engine of Liberty Profile) between the job dispatcher and the job executors:



The executor servers look for job submissions messages that match what they are configured to execute. Considerable flexiblity exists for what attributes an executor server uses when picking up and running a job.

The benefits of this design include:

- Java batch job execution may occur on a Liberty Profile server separate from that used for job submission. The Liberty Profile server may be on the same operating system platform or a different platform.
- You can tailor where jobs run based on criteria of your choosing, such as Job A running on a Windows platform, and Job B running on z/OS.
- Jobs submission can take place even though the executor server is not running. This allows you to submit jobs during the day and later, during a batch window, start the executor servers. They will read the job submission messages and run the Java batch jobs.

Getting Started with IBM JSR 352

This may be as simple as creating a Liberty Profile server, making a few configuration changes, and using the supplied samples to see a JSR 352 Java batch job run. From there you may explore more of the features supplied by IBM JSR 352.

See the WP102544 Techdoc at `ibm.com/support/techdocs` for step-by-step guides on implementing and using IBM JSR 352 with Liberty Profile.

Summary

Batch processing has evolved over time to include different languages, including Java. The development of the JSR 352 open standard for Java batch means application developers may write batch jobs to the open standard specification. Vendors such as IBM implement the JSR 352 standard and provide additional function based on what they see as important to the companies and industries they serve. IBM's JSR 352 in Liberty Profile 8.5.5.6 and higher provides compliance with JSR 352 as well as the additional value features discussed.

More Details

See WP102544 at `ibm.com/suppor/techdocs` for more on the IBM JSR 352 implementation in Liberty Profile.

To discuss IBM JSR 352 or other WebSphere Application Server topics, contact Dave Sudlik, Product Manager - WebSphere Application Foundation, at `dsudlik@us.ibm.com`

End of Document

5 CA-7 and Control-M are trademarks CA Technologies and BMC Software respectively.

6 It uses the WebSphere Optimized Local Adapters (WOLA) function.