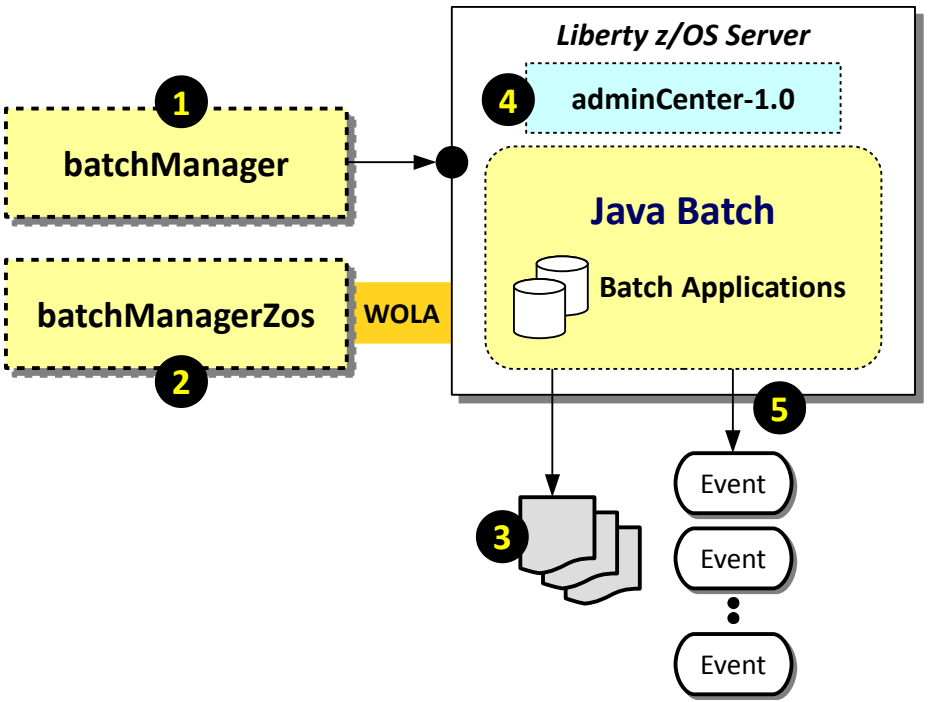
WebSphere Application Server

Unit 4

Job Submission and Control

Topics for this Unit



1. batchManager

Command line client that uses the REST interface of IBM Java Batch to submit, monitor and control batch jobs

Note: this can be used from anywhere Liberty Java Batch is installed; all it needs is a network connection to the server where the batch job runs

2. batchManagerZos

Command line client that uses WOLA as the interface to Liberty z/OS to submit, monitor and control batch jobs

Note: WOLA limits this client to the same LPAR where the server operates.

3. Jog Logging

As jobs execute, a job log is written out to the file system.

4. AdminCenter

The AdminCenter is a browser-based graphical management interface to Liberty. It has a Java Batch tool that lets you view jobs and job results

5. Batch Events

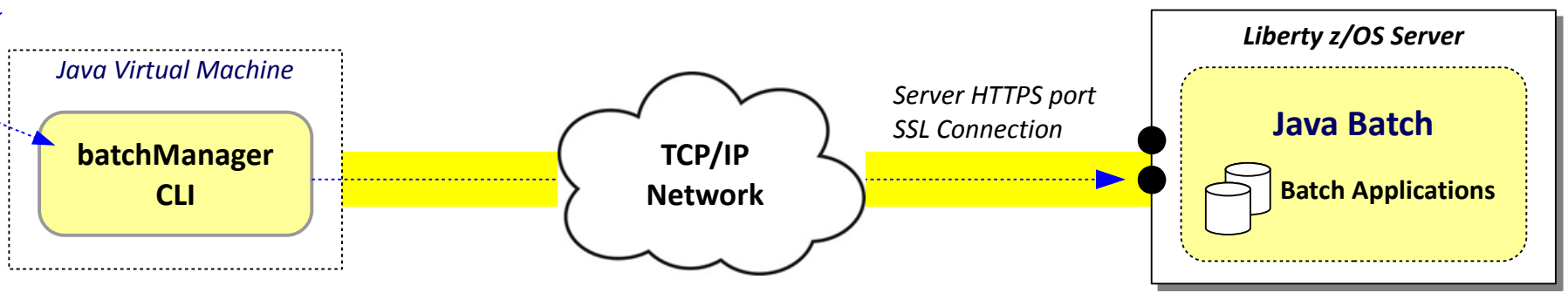
IBM Java Batch can publish "events" to a pub/sub topic so subscribers can monitor results

batchManager

Overview of the batchManager Command Line Interface Utility

/<install_path>/bin
- batchManager ← Any UNIX environment, including z/OS
- batchManager.bat ← Windows

- Command Action (verb)
- Action options and parameters



Key Points:

- Is a Java utility, so JAVA_HOME must be available to the environment
- Establishes network connection to the target server host:port specified
- Can be run from *any* WebSphere Liberty Java Batch platform OS

batchManager Command Line Utility

```
> export JAVA_HOME=/shared/java/J8.0_64
> cd /shared/zWebSphere/Liberty/V16004/bin
> ./batchManager help

Usage: batchManager {help|submit|stop|restart|status|getJobLog|listJobs|purge} [options]

Actions:
    help
        Print help information for the specified action.
    submit
        Submit a new batch job.
    stop
        Stop a batch job.
    restart
        Restart a batch job.
    status
        View the status of a job.
    getJobLog
        Download the joblog for a batch job.
    listJobs
        List job instances.
    purge
        Purge all records and logs for a job instance or purge a list of
        job instance records.
Options:
        Use help [action] for detailed option information of each action.
```

Must have JAVA_HOME in the environment. The utility lives under the /bin directory

We're going to focus on the 'submit' function, but note the other things that can be done with the utility.

For any of the action verbs you can get further 'help' information

'submit' action ...

batchManager 'submit' Action (derived from 'help submit')

```
batchManager submit [options]

   --user=[username]
   --password[=pwd]
   --batchManager=[host]:[port],[host2]:[port2],...
   --controlPropertiesFile=[control-properties-file]
   --trustSslCertificates
   --httpTimeout_s=[http timeout in seconds]
   --jobXMLFile=[jobXMLFile]
   --jobParametersFile=[job-parameters-file]
   --applicationName=[applicationName]
   --jobParameter=[key]=[value]
   --jobPropertiesFile=[job-properties-file]
   --componentName=[componentName]
   --restartTokenFile=[restart-token-file]
   --moduleName=[moduleName]
   --stopOnShutdown
   --jobXMLName=[jobXMLName]
   --pollingInterval_s=[polling interval in seconds]
   --returnExitStatus
   --wait
   --getJobLog
   --verbose
```

This flows over encrypted connection. If no password provided, you will be prompted. It is possible to use a client certificate for authentication when using z/OS SAF.

The 'help submit' function of the batchManager command provides a description for each.

We'll review some of these in a few charts, but first an example ...

Example syntax ...

Example Command Syntax for the BonusPayout Sample Application

Action

```
./batchManager submit
    --batchManager=wg31.washington.ibm.com:25443
    --user=Fred --password=fredpwd
    --applicationName=BonusPayout-1.0
    --jobXMLName=SimpleBonusPayoutJob.xml
    --jobParameter=dsJNDI=jdbc/bonus
    --jobParameter=tableName=BONUSDB.ACCOUNT
    --trustSslCertificates
    --wait
```

The host and HTTPS port of the Liberty z/OS server hosting batch application

ID and password for authentication. This ID must be in the effective registry, and have access to a role sufficient to allow it to submit

The application name

The name of the job JSL file packaged with the application. Alternative is to use jobXMLFile to provide path and name of file external to the job package

Command line utility waits for job to complete before returning control to the prompt

Allows easy acceptance of the self-generated Liberty SSL certificate

Two jobParameter name/value pairs: one to name the JNDI to access DB2 for the application table; the other to name the DB2 table to use

Command options ...

Revisit the Command Options ...

```
--controlPropertiesFile=[control-properties-file]
--trustSslCertificates
--httpTimeout_s=[http timeout in seconds]
--jobXMLFile=[jobXMLFile]
--jobParametersFile=[job-parameters-file]
--applicationName=[applicationName]
--jobParameter=[key]=[value]
--jobPropertiesFile=[job-properties-file]
--componentName=[componentName]
--restartTokenFile=[restart-token-file]
--moduleName=[moduleName]
--stopOnShutdown
--jobXMLName=[jobXMLName]
--pollingInterval_s=[polling interval in seconds]
--returnExitStatus
--wait
--getJobLog
--verbose
```

Provides a way to supply command parameters from a file rather than on the command. Can't use true/false parameters in this file, such as trustSslCertificates or verbose.

Point to a JSL file inline rather than naming one packaged with the application.

If the job has many parameter name/value pairs, you can specify them in a file and point to the file rather than supplying name/value pairs on the command.

An alias of jobParameters

The name of a file which holds the instance id of the job to be restarted. If the file contains an instance id, the job is restarted. If not, a new job is submitted and the resulting instance id is stored in the file.

This option can be used together with --wait. It registers a shutdown hook with the JVM that gets control when the batchManager program is abnormally terminated.

The interval of time at which to poll for job status. Default is 30 seconds.

If specified, the program will download the joblog and print it to STDOUT after the job finishes. This option must be combined with --wait.

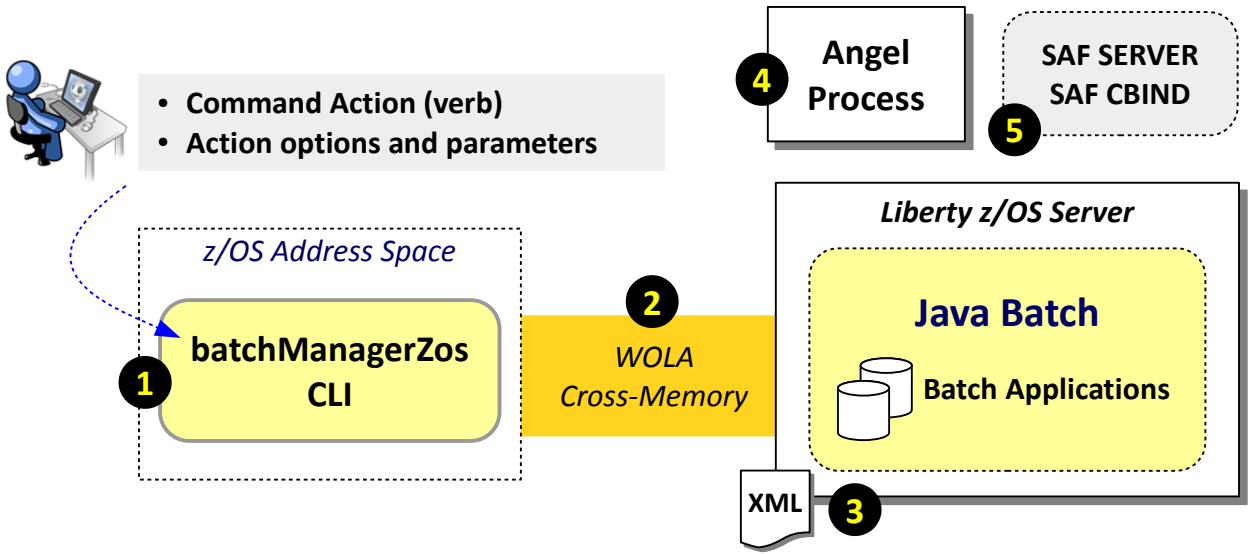
Message at every poll interval

batchManagerZos

Overview of the batchManagerZos Command Line Interface Utility

`/<install_path>/lib/native/zos/s390x` *Note this is down a different path from batchManager*

`batchManagerZos` **z/OS only; not supported on other platforms**

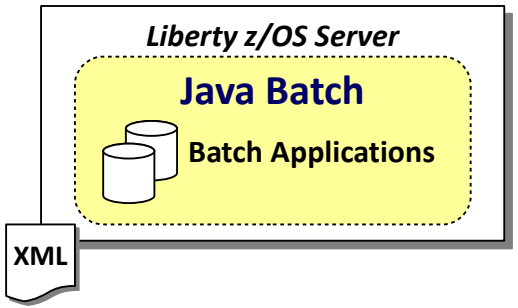


1. **Native z/OS, not Java**
   Which means no JVM is instantiated for each invocation.
2. **WOLA, not network**
   Uses WOLA cross-memory, which is very fast and very secure, but limits to same-LPAR.
3. **Server must support WOLA**
   There's a handful of XML required for a server to be able to use WOLA.
4. **Angel Process required**
   WOLA is a z/OS authorized service, which means the Angel comes into play.
5. **SAF SERVER and CBIND**
   WOLA requires a few SERVER profiles and a CBIND profile.

- **It's a command line interface very similar to batchManager**
- **Pro: can monitor for batch events rather than poll**
- **Con: same LPAR, discussions of high-availability more involved**

Updates to server.xml ...

Updates to server.xml to Support the Use of WOLA for batchManagerZos




```
<featureManager>
    <feature>servlet-3.1</feature>
    <feature>batch-1.0</feature>
    <feature>batchManagement-1.0</feature>
    <feature>zosLocalAdapters-1.0</feature>
    <feature>appSecurity-2.0</feature>
</featureManager>

<authorization-roles id="com.ibm.ws.batch">
  <security-role name="batchAdmin">
    <special-subject type="EVERYONE"/>
    <user name="Fred" />
  </security-role>
</authorization-roles>

<zosLocalAdapters wolaGroup="LIBERTY"
  wolaName2="BATCH"
  wolaName3="MANAGER"/>
```

1

2

3

1. zosLocalAdapters-1.0

The `zosLocalAdapters-1.0` feature is needed to enable the WOLA support. This is *in addition* to the features we've already discussed.

2. WOLA "three-part name"

This "three-part name" is what the batchManagerZos client uses to identify which Liberty server to connect to using WOLA.

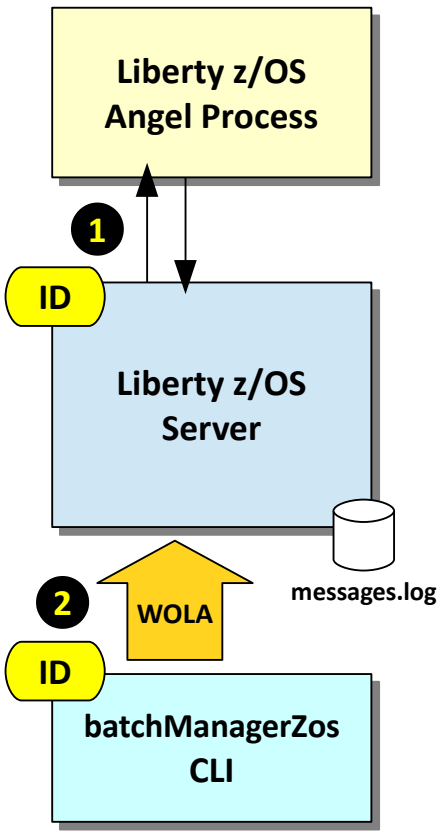
The three-part name can be any value you want, but it must be unique on the LPAR. Each "part" may be up to 8 characters.

This value is also referenced on the SAF CBIND to control what IDs can connect using WOLA.

3. Basic security workaround

This "special-subject" entry to the security role is only needed when using basic security. If you use SAF as the registry and role enforcement, then the ID that invokes batchManagerZos can be checked and validated. We'll look at this in more detail in the "Security" unit.

SAF Security That Must be in Place for WOLA to Work (Simplified Explanation*)



1. Angel Present; Server ID access via SERVER profiles

Two key things going on here – (1) access to the Angel process, and (2) authority for server to load the authorized code. These are provided by SAF SERVER profiles.
The process is:

- Create the SAF SERVER profiles (details in Security unit)
- Grant server STC ID (or group) READ to the SERVER profiles
- Start the server and check messages.log for key indicator of access:

```
CWWKB0103I: Authorized service group LOCALCOM is available.
CWWKB0103I: Authorized service group WOLA is available.
CWWKB0103I: Authorized service group CLIENT.WOLA is available.
```

and:

```
CWWKB0501I: The WebSphere Optimized Local Adapter channel registered with
the Liberty profile server using the following name: LIBERTY BATCH MANAGER
```

2. batchManagerZos ID access via CBIND profile

The purpose of the CBIND is to control what client IDs are allowed to create the WOLA "registration" (cross-memory connection) into the Liberty server. The SAF CBIND profile is based on the three-part name in the server.xml (wildcard characters allowed). Grant READ to the ID running batchManagerZos and it will be allowed to create the registration.

The ID that runs batchManagerZos is either the ID logged into the UNIX shell, or the effective ID of JCL job that invokes batchManagerZos.

* More detail is provided in the "Security" unit

batchManagerZos help ...

batchManagerZos 'help'

```
./batchManagerZos help

Usage: batchManagerZos {help|ping|submit|stop|restart|status|purge|listJobs} [options]

Actions:
   help
       Use help [action] for detailed option information.
   ping
       'Ping' the batch manager to test connectivity.
   submit
       Submit a new batch job.
   stop
       Stop a batch job.
   restart
       Restart a batch job.
   status
       View the status of a job.
   purge
       Purge all records and logs for a job instance or purge a list of job instance records.
   listJobs
       List job instances

Options:
       Use help [action] for detailed option information.
```

'help' on action provides details for any of the actions

Action 'ping' is unique to batchManagerZos. Use this to test for basic WOLA connectivity

Actions are very similar to those provided with batchManager CLI

batchManagerZos 'submit' Action (derived from 'help submit')

```
./batchManagerZos help submit
    --batchManager=[WOLA 3-Part Name]  1

Options:
   --applicationName=[applicationName]
   --moduleName=[moduleName]
   --componentName=[componentName]
   --jobXMLName=[jobXMLName]
   --jobXMLFile=[jobXMLFile]
   --jobParameter=[name]=[value]
   --jobParametersFile=[job-parameters-file]
   --jobPropertiesFile=[job-properties-file]
   --controlPropertiesFile=[control-properties-file]
   --restartTokenFile=[restart-token-file]
   --wait
   --queueManagerName=[queueManagerName]  2
   --pollingInterval_s=[polling interval in seconds]
   --getJobLog
   --verbose
   --returnExitStatus
```

Very similar to batchManager, with a few exceptions which we highlight here

1. --batchManager=

This is used to name the WOLA three part name, for example:

`--batchManager=LIBERTY+BATCH+MANAGER`

The server you wish to connect to must have this exact three-part name defined, and have all the "is available" messages present. Further, there must be a CBIND that grants READ to the client ID.

2. --queueManagerName=

This is related to "batch events." This lets batchManagerZos monitor for job status by subscribing to the topic. This avoids the overhead of periodic polling the `--wait` option uses by default.

More on "batch events" coming up.

Example Command Syntax for the BonusPayout Sample Application

Action

```
./batchManagerZos submit
  --batchManager=LIBERTY+BATCH+MANAGER
  --applicationName=BonusPayout-1.0
  --jobXMLName=SimpleBonusPayoutJob.xml
  --jobParameter=dsJNDI=jdbc/bonus
  --jobParameter=tableName=BONUSDB.ACCOUNT
  --wait --queueManagerName=MQS1
```

Specify the three-part name defined in the server

Name the application, JSL name, and pass in the two sets of name/value pairs (this is just like batchManager)

Specify --wait and name the MQ queue manager for monitoring on batch events

The syntax related to specifying the application and the JSL is the same as batchManager. Note that this has no host, port, ID, password, or SSL considerations. This is using WOLA.

Invoking batchManagerZos* using JCL and BPXBATCH

```
//BMGRZJCL JOB (0),'BMGR+JCL',CLASS=A,
// REGION=0M,MSGCLASS=H,NOTIFY=USER1
//SUBMIT   EXEC  PGM=IKJEFT01
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//STDOUT   DD  SYSOUT=*
//STDERR   DD  SYSOUT=*
//SYSTSIN  DD  *
 BPXBATCH  SH +
   batchManagerZos or shell script
/*
//*
```

0	The task completed normally.
20	A required argument was not specified.
21	An unrecognized argument was specified.
22	An invalid argument value was specified.
255	An unknown error occurred.
33	The job stopped.
34	The job did not complete successfully.
35	The job completed successfully.
36	The job was abandoned.

If --wait specified

Invoke batchManagerZos directly

You can invoke batchManagerZos directly in the JCL, coding all the parameters needed, and keeping everything in the 72-column limit of JCL using plus sign (+) continuation symbols.

This works, but the way BPXBATCH surfaces return codes to JES may be confusing. (BPXBATCH shifts value one byte; JES takes only a portion of result.) A shell script can help with this.

Invoke shell script, which invokes batchManagerZos

This allows you to capture the return code from batchManagerZos and set the script exit code so the value is never more than 16.

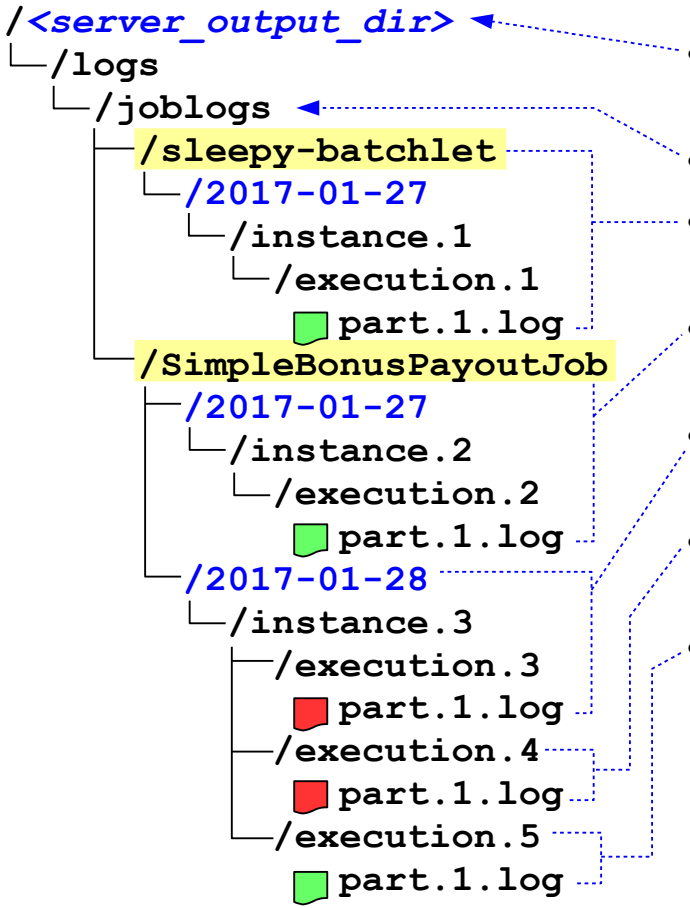
batchManagerZos Return Code	*Shell script sets exit code to*	*Exit status HEX*	*BPXBATCH shift*	*RC in JES*
35	0	x'0' →	x'000'	0000
33	4	x'4' →	x'400'	1024
20,21,22	8	x'8' →	x'800'	2048
34	12	x'C' →	x'C00'	3072
else	13	x'D' →	x'D00'	3328

The shell script we use in lab does this

* Or batchManager … same considerations

Job Logging

Illustration of the Job Log Tree for a Server



- The output directory defaults to the server configuration root (where server.xml resides), but can be redirected using WLP_OUTPUT_DIR
- The job logs will go under the /logs/joblogs directory of that output directory
- The first job ran was SleepyBatchlet, on January 27th. It ran successfully. There was one "instance" (#1) and one "execution" (#1) of that instance.
- The next job was SimpleBonusPayout, also on January 27th. It ran successfully. There was one "instance" (#2) and one "execution" (#2) of that instance.
- Then on January 28th SimpleBonusPayout was run again. This was instance #3. But execution #3 failed.
- You restarted it. Same "instance" (#3), but now a new "execution" (#4). It failed as well.
- You restarted it again. Same "instance" (#3), a new "execution" (#5). This time it worked.

The next job you run will be "instance" #4, and "execution" #6.

These instance and execution numbers are what uniquely identify the job instance and execution for purposes of job restart and job log retreival.

Controlling Job Logging by Server

server.env

```
WLP_OUTPUT_DIR=path
```

Liberty output directory variable

By default output goes to the server root directory (the directory in which the server.xml file is located). You can direct server output to a different location using this variable.

server.xml

```
<batchJobLogging
    enabled="true|false"
    maxTime="<seconds>"
    maxRecords="<records>" />
```

Switch to enable or disable batch job logging

By default this is "true," so left uncoded you get the default Java batch job logging behavior. Code "false" and no job logging occurs.

Jog log rolling based on records

This determines when a job log will be closed and rolled to a new job log part based on records written. Range is 0 to 2147483647, with a default of 1000

Job log rolling based on time

This determines when a job log will be closed and rolled to a new job log part based on time elapsed. Range is 0 to 2147483647, with a default of 0.

Note: if you have a non-zero time specified and no records are written in that time interval, then no log rolling occurs. There's no sense is rolling an empty job log part.

Messages Written by Java Batch Application – go to Job Logging?

From the WP102544 "Step-by-Step Implementation Guide"

Note: Messages written by the application will be intercepted and sent to the job log only if `java.util.logging` is used, or another tool (such as Log4j) that uses `java.util.logging` as the backend. Messages written out using `println()` will *not* go to the job log, but *will* go the server STDOUT location.

STDOUT is the messages.log file

To achieve output to the Job Log, do something like this:

```
import java.util.logging.Logger;

protected final static Logger log = Logger.getLogger(MyProgram.class.getName());

System.out.println("Hello World!");

public String process() {
  log.log(Level.INFO, "Hello World!");
      return null;
}
```

This will go to message.log

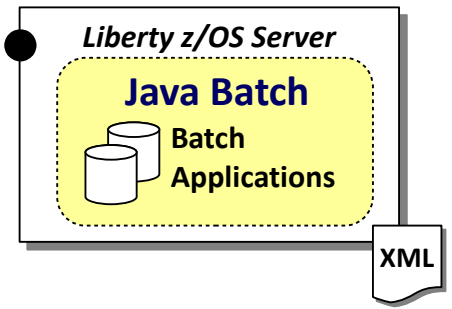
This will go to the job log

Ways to Retrieve the Job Logs

1. **Directly**
   The job log parts are written to the directory location we showed earlier. If you have access, you are free to go into that directory and retrieve the job log parts. If multiple job log parts, you must piece them back together yourself.
2. **batchManager 'getJobLog' action**
   This action (or 'verb') takes as input the job instance number and job execution number, along with where you want the output file written, and the type of file to write: text or zip. This will concatenate multiple job log parts.
3. **REST interface (which is what batchManager ultimately uses)**
   A set of GET verbs to retrieve a listing of the job log parts and then retrieve the files. If you want more details, go to the Knowledge Center and search: `rwlp_batch_rest_api`.
4. **--getJobLog on batchManager or batchManagerZos command line**
   This parameter on the 'submit' action, along with --wait, will return the job log to STDOUT. Depending on how you invoke either CLI client, that would be your shell (telnet, SSH, or OMVS), or if BPXBATCH then you can redirect to a file or to JES spool.
5. **From the AdminCenter Java Batch tool GUI** (upcoming section in this unit)
   The little file icon to on the GUI will retrieve the job parts and display them in the browser session. If multiple job log parts then you must scroll through each part individually.
6. **Using batch events and monitoring on jobLogPart topic** (upcoming section in this unit)
   A monitoring application can subscribe to the jobLogPart topic and retrieve the job log parts as they're published. We show an example of that in the "batch events" section later in this unit.

AdminCenter

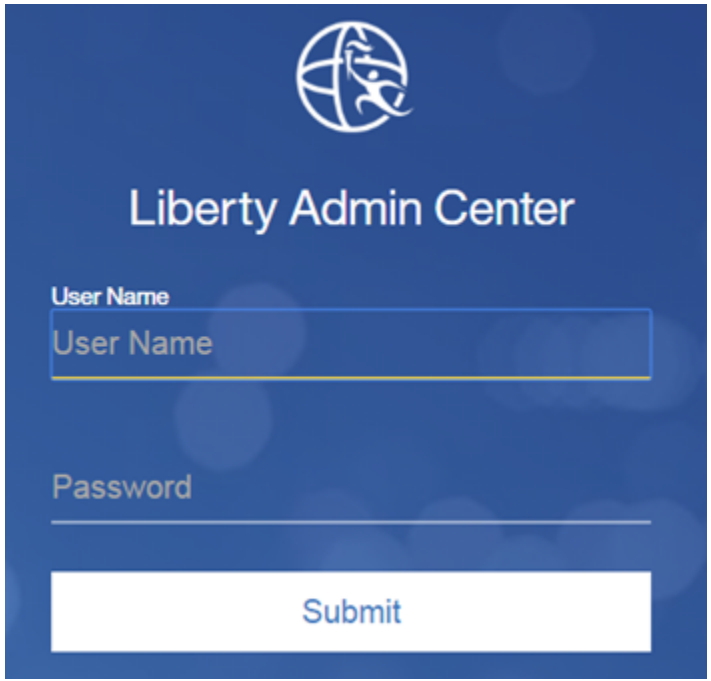
Overview of the Liberty AdminCenter




```
<featureManager>
    <feature>servlet-3.1</feature>
    <feature>batch-1.0</feature>
    <feature>batchManagement-1.0</feature>
    <feature>zosLocalAdapters-1.0</feature>
    <feature>appSecurity-2.0</feature>
    <feature>adminCenter-1.0</feature>
</featureManager>
```

The `adminCenter-1.0` feature enables this function

AdminCenter Security ("Basic" for now)



Basic Keystore (SSL)

```
<keyStore id="defaultKeyStore" password="Liberty"/>
```

```
<basicRegistry id="basic1" realm="jbatch">
  <user name="Fred" password="fredpwd" />
</basicRegistry>
```

Basic User Registry

```
<authorization-roles id="com.ibm.ws.batch">
  <security-role name="batchAdmin">
    <special-subject type="EVERYONE"/>
    <user name="Fred" />
  </security-role>
</authorization-roles>
```

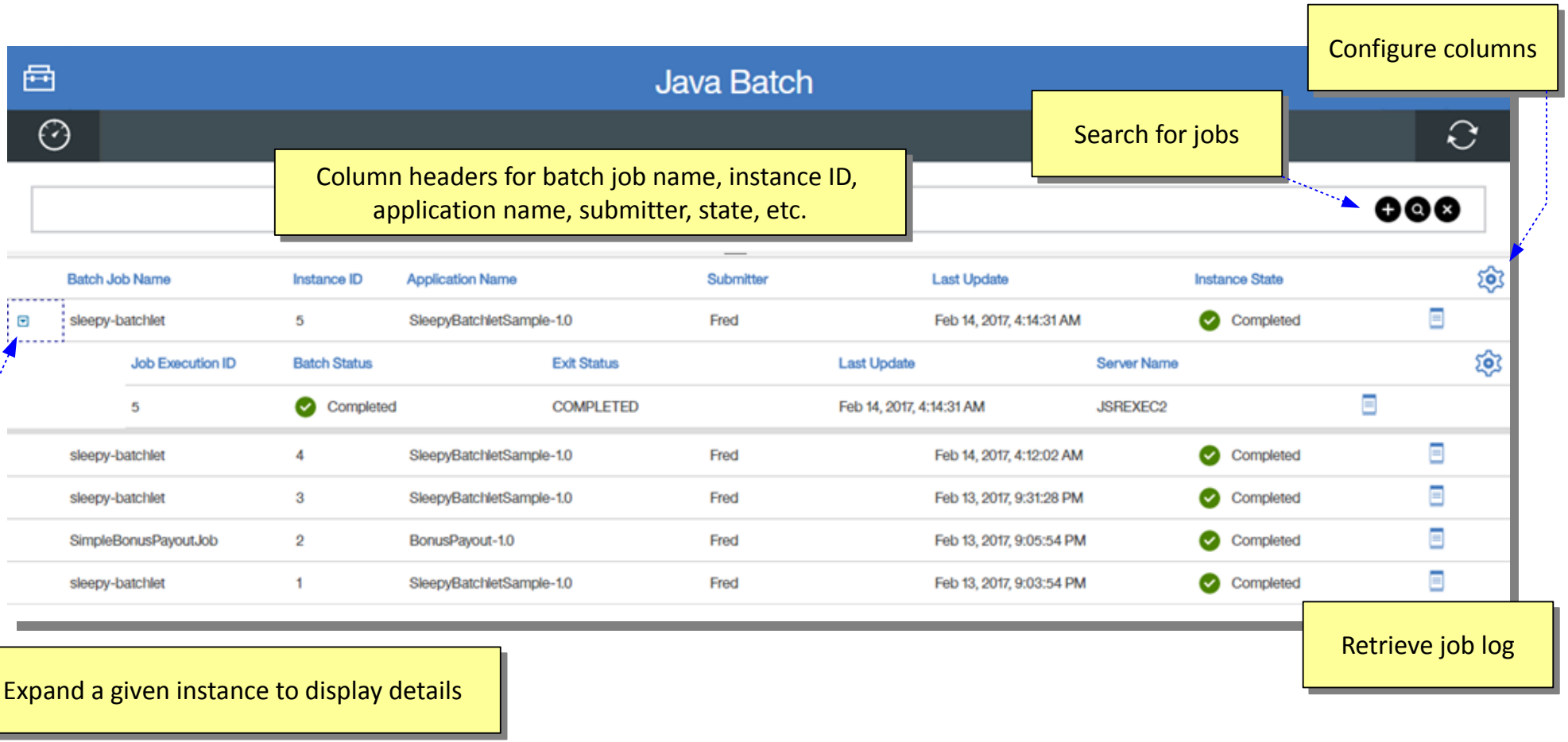
Authorization roles for Java Batch

```
<administrator-role>
  <user>Fred</user>
</administrator-role>
```

Authorization role for AdminCenter

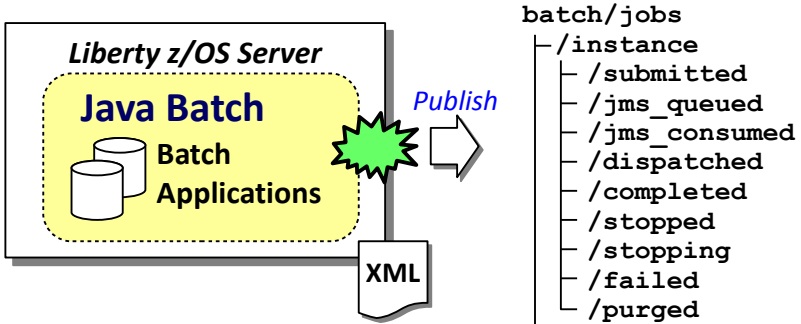
The AdminCenter URL will re-direct to SSL and prompt for userid and password. This "basic" security is what allows it to work. More security details in "Security" unit.

The Java Batch Tool Display (in 16.0.0.4)



Batch Events

Overview of Batch Events




```
batch/jobs
 ├/instance
 │  ├ /submitted
 │  ├ /jms_queued
 │  ├ /jms_consumed
 │  ├ /dispatched
 │  ├ /completed
 │  ├ /stopped
 │  ├ /stopping
 │  ├ /failed
 │  └ /purged
 └/execution
    ├ /restarting
    ├ /starting
    ├ /completed
    ├ /failed
    ├ /stop
    ├ /jobLogPart
    ├ /step
    │  ├ /started
    │  ├ /completed
    │  ├ /failed
    │  ├ /stopped
    │  └ /checkpoint
    ├ /partition
    │  └ /(sub-topics)
    └ /split-flow
       └ /(sub-topics)
```

Enabled with <batchJmsEvents> element in XML

As well as a valid <jmsConnectionFactory> that provides connectivity to a messaging engine capable of hosting a pub/sub topic.

Examples: IBM MQ, or the default messaging of Liberty.

While batch jobs execute, events emitted to topic tree

The topics are shown in the chart to the left. The topics represent different "states" of jobs, steps, partitions and split-flows.

For example, the "jobLogPart" event is published with the job log part each time a part is created.

Any monitoring function can subscribe and monitor

This pub/sub design allows any monitoring function to subscribe to the topic and receive the events as they occur.

For example, batchManagerZos has the ability to subscribe to the topic tree and monitor for job completion status (rather than periodic polling).

KC Search `twlp_batch_monitoring`

server.xml ...

Example of server.xml to Support Batch Events (MQ in this example)

```
<variable name="wmqJmsClient.rar.location"
  value="${server.config.dir}/wmq.jmsra.rar" />

<batchJmsEvents
  connectionFactoryRef="batchConnectionFactory" />

<jmsConnectionFactory id="batchConnectionFactory"
  jndiName="jms/batch/connectionFactory">
  <properties.wmqJms
    hostName="wg31.washington.ibm.com"
    transportType="CLIENT"
    channel="SYSTEM.DEF.SVRCONN"
    port="1414"
    queueManager="MQS1">
  </properties.wmqJms>
</jmsConnectionFactory>
```

MQ Resource File Location

To access MQ, Liberty is going to need the MQ resource adapter to load the code to make the connection. This variable provides the pointer to where you have the RAR file located.

Note: in this case we used ${server.config.dir} to resolve the location to the server's configuration directory, which is where server.xml resides.

<batchJmsEvents>

This element is what enables batch events. It points to the JMS connection factory to use.

<jmsConnectionFactory>

This defines the connection to the MQ queue manager to which the batch events will be published. In this case we're showing client mode.

Recall the batchManagerZos Command Line Utility

```
./batchManagerZos help submit
    --batchManager=[WOLA 3-Part Name]

Options:
   --applicationName=[applicationName]
   --moduleName=[moduleName]
   --componentName=[componentName]
   --jobXMLName=[jobXMLName]
   --jobXMLFile=[jobXMLFile]
   --jobParameter=[name]=[value]
   --jobParametersFile=[job-parameters-file]
   --jobPropertiesFile=[job-properties-file]
   --controlPropertiesFile=[control-properties-file]
   --restartTokenFile=[restart-token-file]
   --wait
   --queueManagerName=[queueManagerName]
   --pollingInterval_s=[polling interval in seconds]
   --getJobLog
   --verbose
   --returnExitStatus
```

The --queueManagerName parameter specifies the MQ queue manager where the batch events are being published

For this to work, the QMGR receiving the published events must be local to the batchManagerZos client (it uses BINDINGS mode) to access

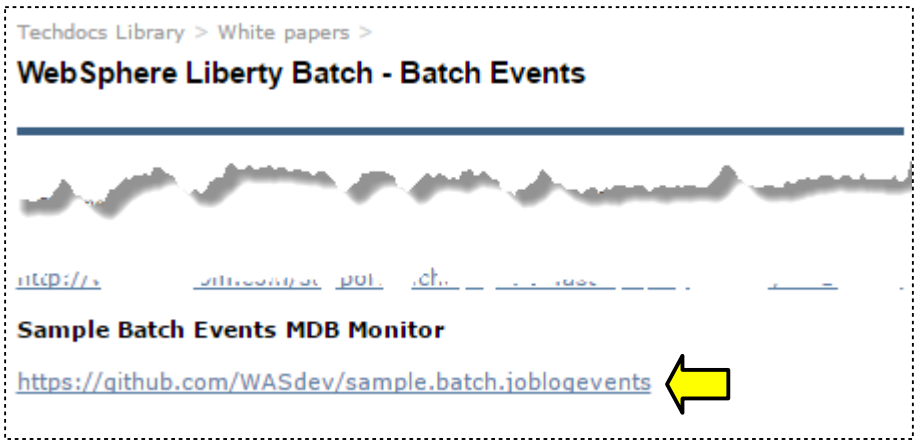
The topics it subscribes to are:

```
batch/jobs/instance/stopped
batch/jobs/instance/failed
batch/jobs/instance/completed
```

When it sees an event on one of those topics it can determine the result

Another Example: Batch Events MDB Monitor Linked to From WP102603 Techdoc

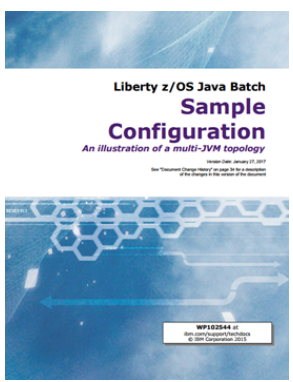
http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP102603



This is an MDB application that subscribes to the batch/jobs/execution/jobLogPart topic

When a Job Log Part is seen, it retrieves the log part and stores it in a directory under the server running the MDB

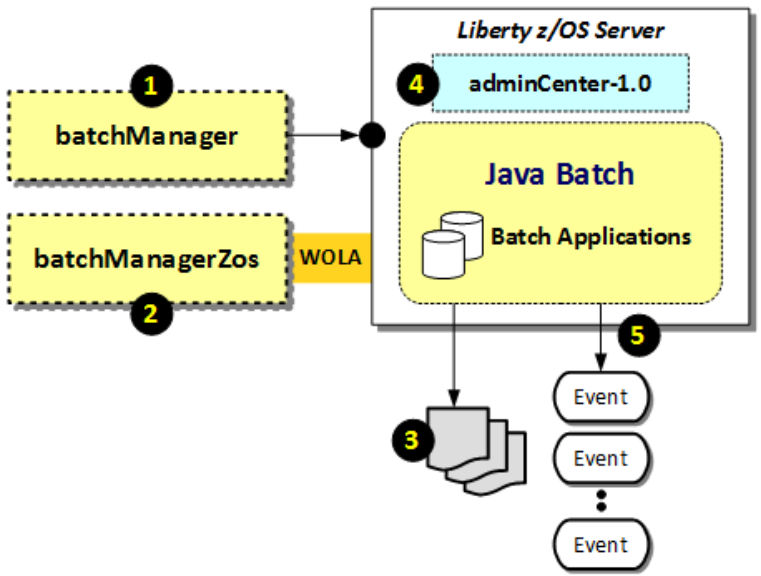
Source for this sample provided at Git location linked to from the Techdoc



The WP102544 WebSphere Liberty Batch anchor Techdoc has a PDF that provides a sample configuration which includes this MDB monitoring application running in a separate Liberty server

http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP102544

Summary of Topics We Covered In This Unit



1. batchManager

Command line client that uses the REST interface of IBM Java Batch to submit, monitor and control batch jobs

Note: this can be used from anywhere Liberty Java Batch is installed; all it needs is a network connection to the server where the batch job runs

2. batchManagerZos

Command line client that uses WOLA as the interface to Liberty z/OS to submit, monitor and control batch jobs

Note: WOLA limits this client to the same LPAR where the server operates.

3. Jog Logging

As jobs execute, a job log is written out to the file system.

4. AdminCenter

The AdminCenter is a browser-based graphical management interface to Liberty. It has a Java Batch tool that lets you view jobs and job results

5. Batch Events

IBM Java Batch can publish "events" to a pub/sub topic so subscribers can monitor results

Questions?