IBM WebSphere Liberty Batch z/OS

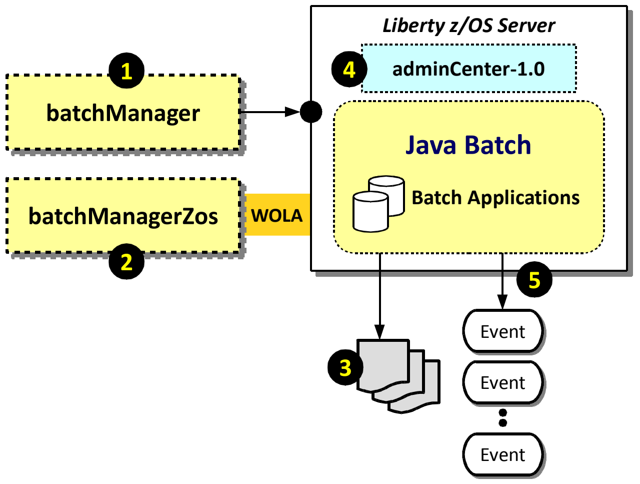
WebSphere Application Server

Unit 4

Job Submission and Control

IBM WebSphere Liberty Batch z/OS

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

© 2017 IBM Corporation 2 **batchManager ...**

In this unit we're going to turn our attention back to the IBM operational enhancements to the JSR-352 specification and show how they operate. To that end we're going to cover five topics:

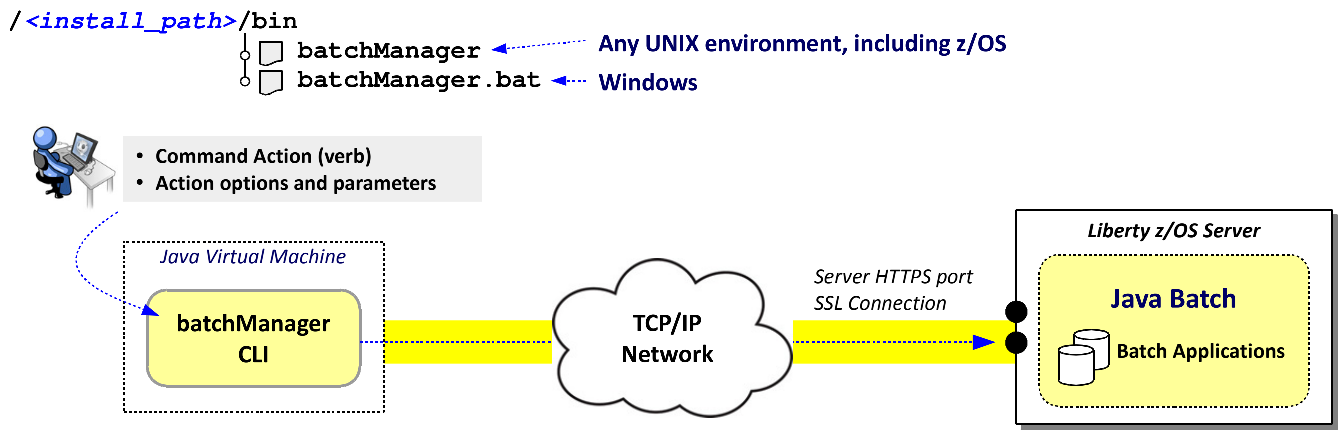
1. batchManager – this is a command line client useful for submitting jobs and managing job execution. It's a Java-based utility that behind-the-scenes uses the REST interface of IBM WebSphere Liberty Java Batch. Because it's a network-based utility, it can be run from anywhere. It does *not* need to be run on the same LPAR as the Liberty server running the batch program.
2. batchManagerZos – this is another command line client, very similar to batchManager, that is useful for submitting jobs and managing job execution. Unlike batchManager, this is a native program that uses WebSphere Optimized Local Adapers (WOLA) to get from the batchManagerZos client into the Liberty z/OS server. This limits batchManagerZos execution to the same LPAR as the Liberty z/OS server it communicates with. But in return you get (a) the avoidance of overhead instantiating a JVM to run batchManager, and (b) a more efficient mechanism (batch events rather than periodic polling) to determine when the Java batch job has completed.
3. Job logging – the JSR-352 specification mentions job logging, but does not spell out any details how that is to be done. The IBM implementation provides for a structured mechanism for logging job output, as well as a means of retrieving the job logs.
4. The Liberty AdminCenter – the AdminCenter is a web-based administrative interface to Liberty itself. It is designed to have "tools" (functional plugin-ins) to do different things. One of the tools is for Java batch in which you can view the status of jobs and retrieve job logs.
5. Batch events – this is an IBM enhancement that uses JMS (Java Messaging Service) publish/subscribe to emit "events" (messages) at various stages of job execution. These events are published to a "topic" (a queue) which can be "subscribed to" (monitored) by a process looking for key things in the events that have been published. This opens up range of interesting architectural things related to monitoring the batch environment. One of the exploiters of this is the batchManaagerZos client, which can be configured to monitor the topic watching for the job status indicator. That avoids having to periodically poll the server to determine the job status. For very long running jobs this savings can add up.

Ready? Here we go ...

batchManager

IBM WebSphere Liberty Batch z/OS

Overview of the batchManager Command Line Interface Utility



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 CLI ...

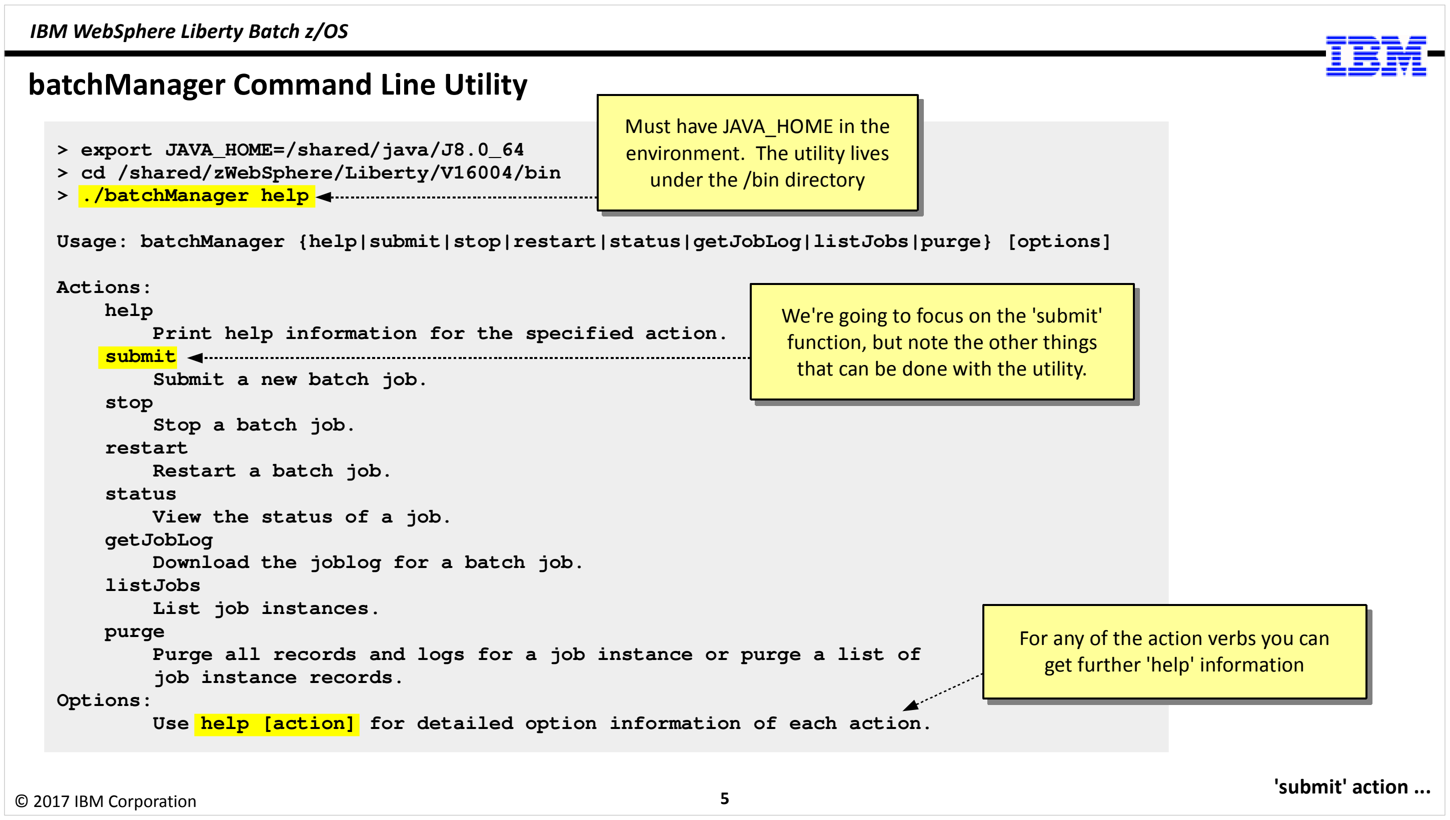
The batchManager Command Line Interface (CLI) utility comes with Liberty on all platforms, and can be found under the /bin directory of the install path. You will find two copies of it – one with no extension, which is used on any UNIX environment, including z/OS; and one with an extension of "bat" which is for Windows.

The command is executed in a UNIX shell or Windows command prompt environment. On z/OS it can be wrapped in JCL with BPXBATCH being the mechanism that provides the UNIX shell. It's a Java program, so it'll need access to Java via a JAVA_HOME variable that points to a valid 64-bit Java implementation.

The utility takes as input a command string consisting of an action (or a "verb") and a set of options. Over the next several charts we'll explore the actions and options available to batchManager.

This utility makes use of a network connection to the Liberty server where the Java Batch is running, so the client can be run on any platform and connect to another platform where the Liberty server is located. All you need is a network between the two.

Note: as mentioned before, what batchManager is doing is using the REST interface provided by the IBM WebSphere Liberty Java Batch function. The value of batchManager is that it eliminates the need for you to undestand the REST URI patterns and the JSON format ... batchManager handles all that for you.



IBM WebSphere Liberty Batch z/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

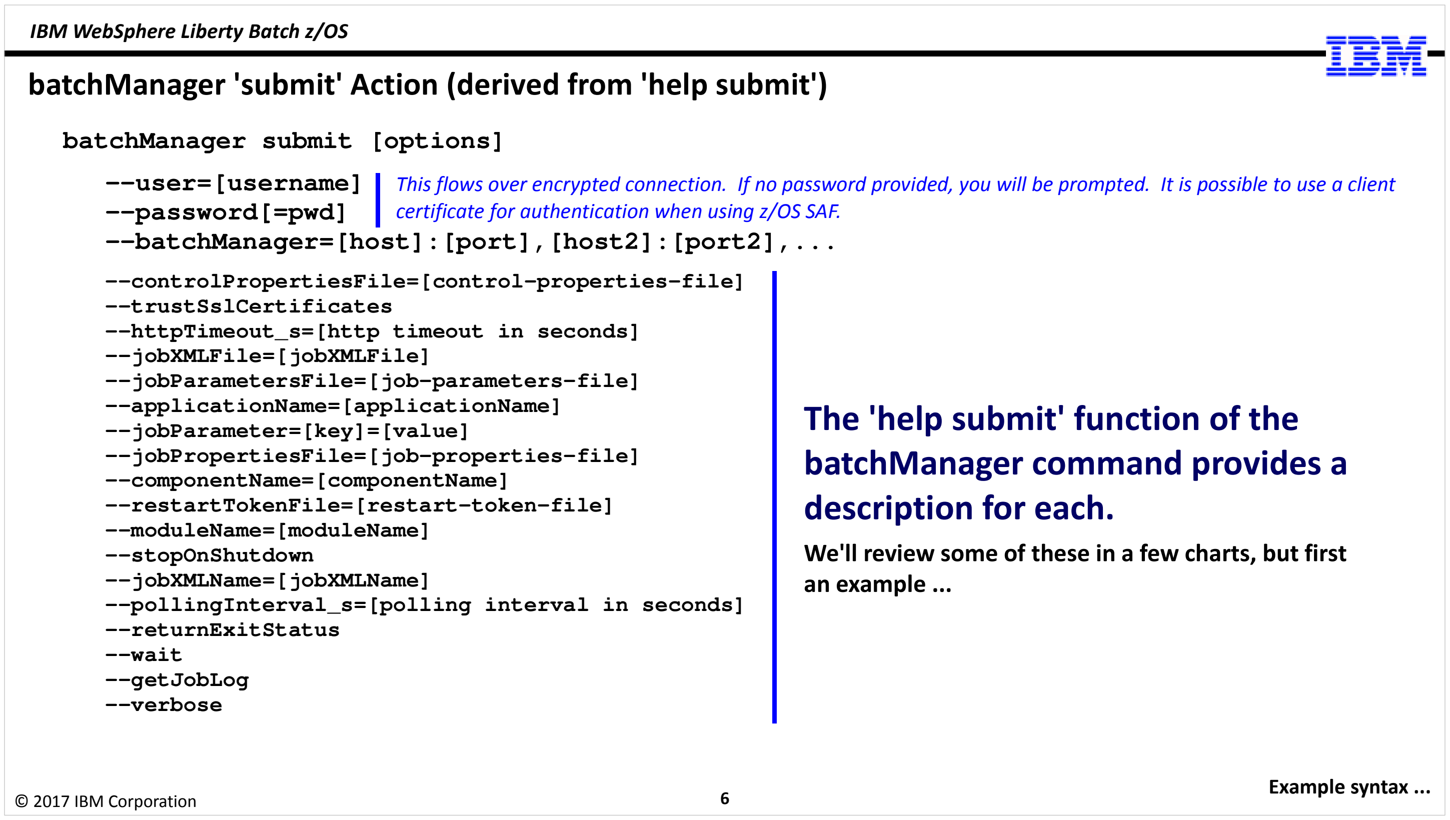
'submit' action ...

© 2017 IBM Corporation 5

The chart above shows a shell environment where we ran the batchManager utility with the 'help' action. Notice how we exported the JAVA_HOME variable, then we changed directories to the /bin directory of our install path.

The result was a listing of the actions that are provided by batchManager – submit, stop, restart, status, getJobLog, listJobs, and purge. Our focus will be on the 'submit' action, but we wanted you to see the other actions that are available.

Note: the 'help' function is particularly useful in getting a listing of details for any of the given actions. On the next chart we're going to show you the results of 'help submit'. You can issue the same for the other actions – 'help stop' or 'help restart,' etc. Whenever you're not sure of a syntax, remember this help facility.



IBM WebSphere Liberty Batch z/OS

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

```
batchManager submit [options]

    --user=[username]
    --password[=pwd]
    --batchManager=[host]:[port],[host2]:[port2],...
```

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.

```
    --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
```

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 ...

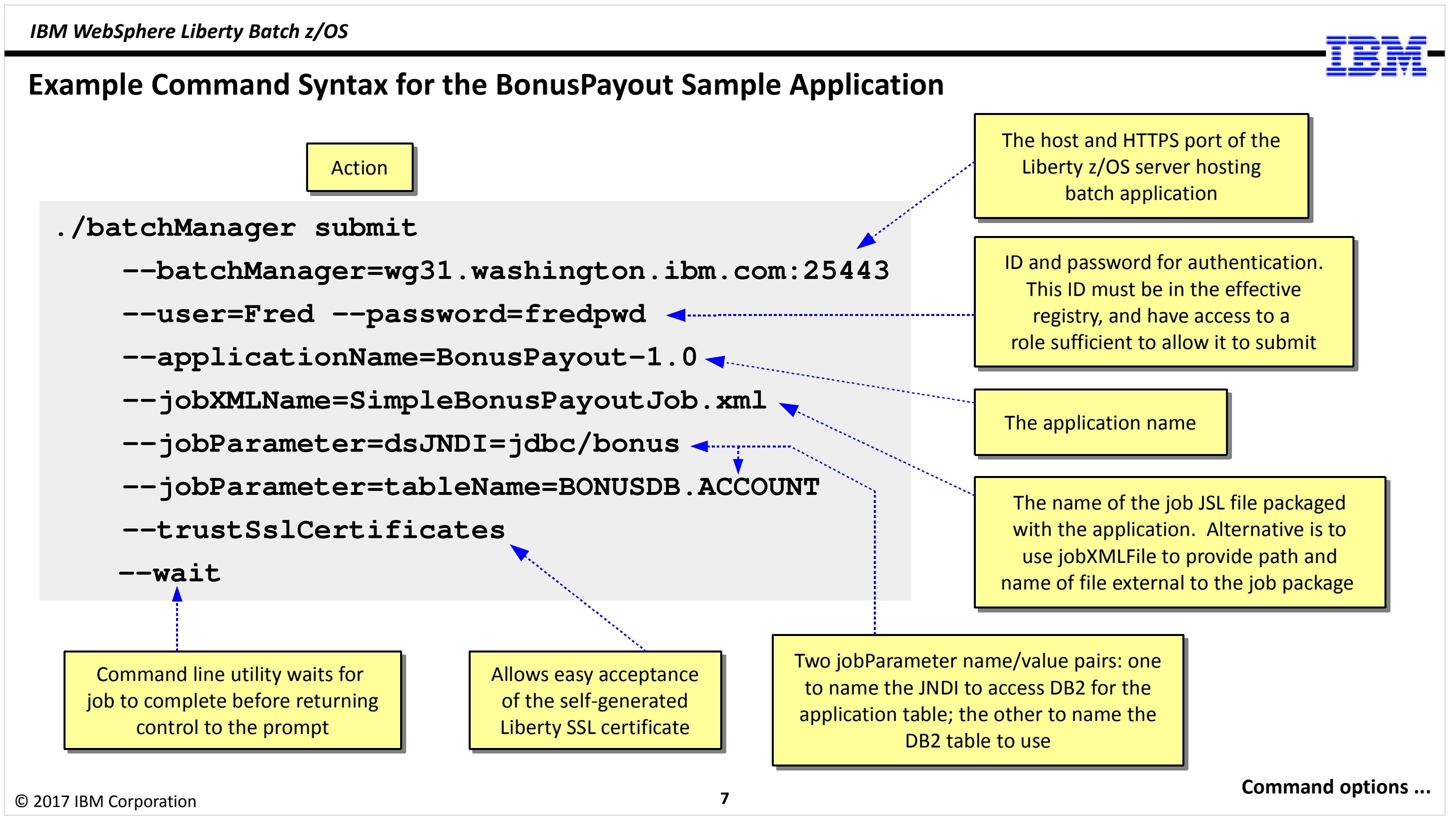
© 2017 IBM Corporation 6 **Example syntax ...**

Here is the result of the 'help submit' command. The actual output is more verbose with descriptive text for each of the options. We've trimmed that out to save space and highlight the options themselves.

Some of the options are required, some are truly optional. The three at the top are required – a user and password, and the server to which you wish to connect. Some interesting things to keep in mind:

- You can actually not supply user and password ... it'll then prompt you. That works okay when you're an interactive shell environment, but if you'd doing this using BPXBATCH and JCL it's not really an option.
- You can use a client certificate to authenticate, which would eliminate the need to code user and password on the command line. This requires the Liberty server be set up to request a client certificate, and making a client certificate available to the JVM that runs batchManager. Explaining all that here is too much for this unit.
- Notice the host:port, host:port of the command. You can specify multiple host:port pairs and have batchManager failover from one to the other if it can't make a connection. This would be useful when you're seeking a highly-available environment with two (or more) Liberty servers set up to receive batch submission requests.

As for the options ... we're review some of those in several charts, but first we'll show you an example to set context.



IBM WebSphere Liberty Batch z/OS

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

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

Allows easy acceptance of the self-generated Liberty SSL certificate

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

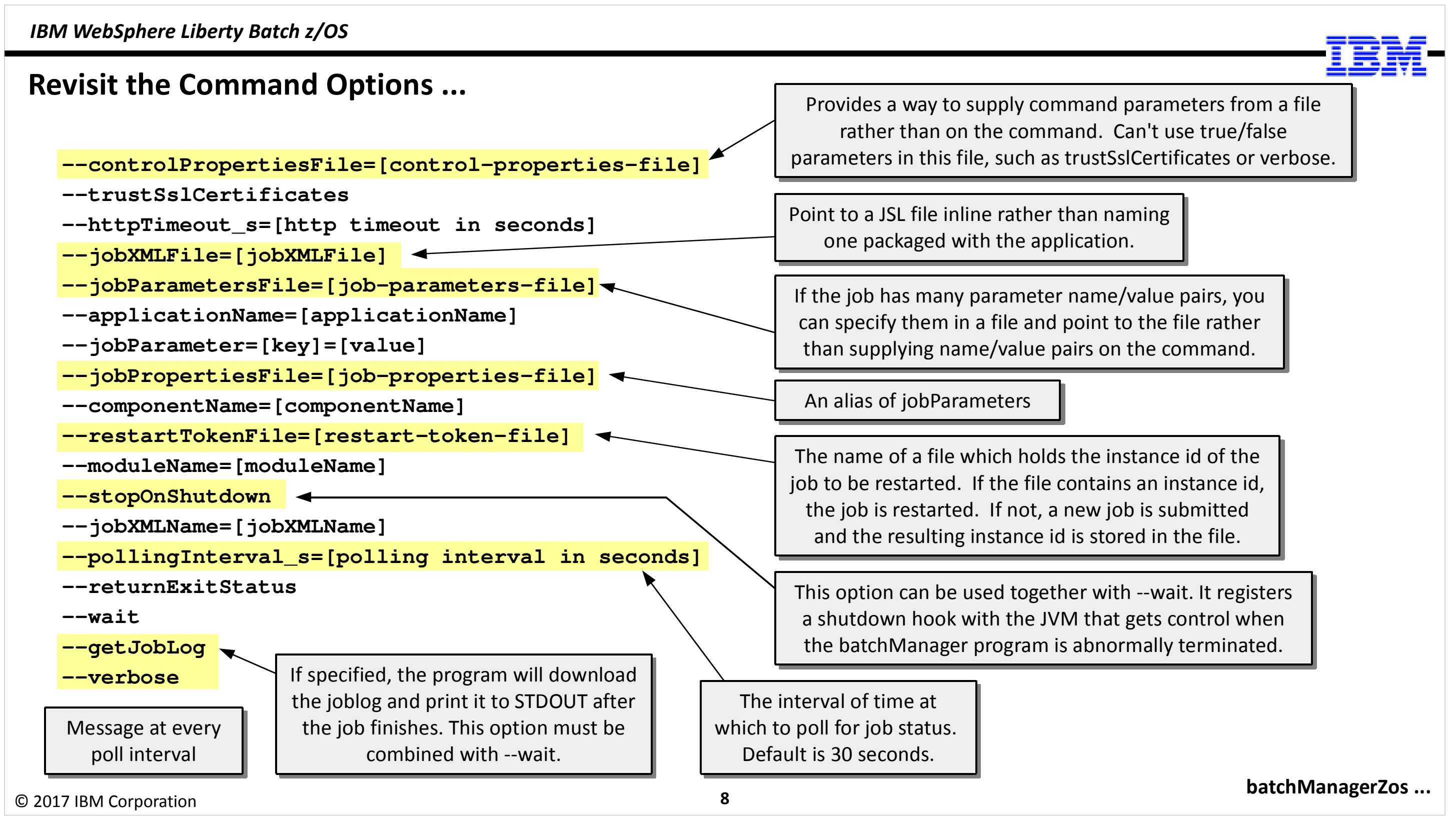
Command options ...

© 2017 IBM Corporation 7

Here's an example of the batchManager command in use to submit another sample job called BonusPayout, which is a sample job with a batchlet and a chunk step. It has the ability to pass in several job parameters, and it's useful as an illustration of a batchManager command that's something different from what we've already seen with SleepyBatchlet. Let's walk through the options from top to bottom:

- The --batchManager option points to the host where Liberty is running and the HTTPS port on which to connect.
- The --user and --password options provide what's necessary to authenticate. This flows on an encrypted SSL connection, so the password is not in the clear. With basic security we don't have the option of client certificate authentication, but with SAF security we do, so with client certificate authentication it would be possible to eliminate --user and --password entirely.
- The --applicationName option is used to specify which application to submit. A Liberty server may be hosting several different Java batch applications, so we need to indicate which one is being referenced. This option does that.
- The --jobXMLName option is used to specify the JSL file packaged with the application. It's possible to have multiple JSL files packaged with an application, so this lets you specify which one you want to use.

  Note: the --jobXMLFile option lets you point to a JSL file *outside* the application package.
- This sample application has two parameter name/value pairs: one to indicate the JNDI name of the JDBC data source to access the application table, and one for the table name itself. So we code two --jobParameter options, one for each job parameter we wish to pass in.
- The --trustSslCertificates option allows us to get around an issue we have with "basic" security; specifically, the fact that the self-signed certificate Liberty generated with the basic security settings in server.xml is not recognized because there is no Certificate Authority that has signed the certificate. --trustSslCertificates simply tells batchManager to ignore the "no CA has signed this" issue and keep going. Great for testing; never use this in production.
- Finally, the --wait option tells batchManager to hold off returning to the command prompt until the Java Batch job has completed. It does this by periodically polling the server to check the status of the job. When the Java Batch job completes, batchManager ends and returns to the command prompt.



IBM WebSphere Liberty Batch z/OS

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

© 2017 IBM Corporation 8 **batchManagerZos ...**

Let's go back to the command options and take another tour of what they do. The ones highlighted in yellow are the ones we're looking at on this chart, and the boxes that point to each offers a brief description of what each is for.

Next we're going to look at batchManagerZos, which is very similar to batchManager, but it uses a different connectivity mechanism.

batchManagerZos

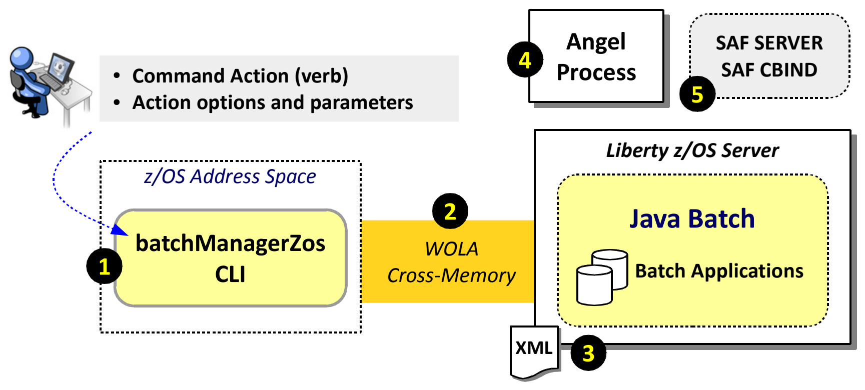
IBM WebSphere Liberty Batch z/OS

Overview of the batchManagerZos Command Line Interface Utility

```
/<install_path>/lib/native/zos/s390x
                                    batchManagerZos
```

Note this is down a different path from batchManager

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**

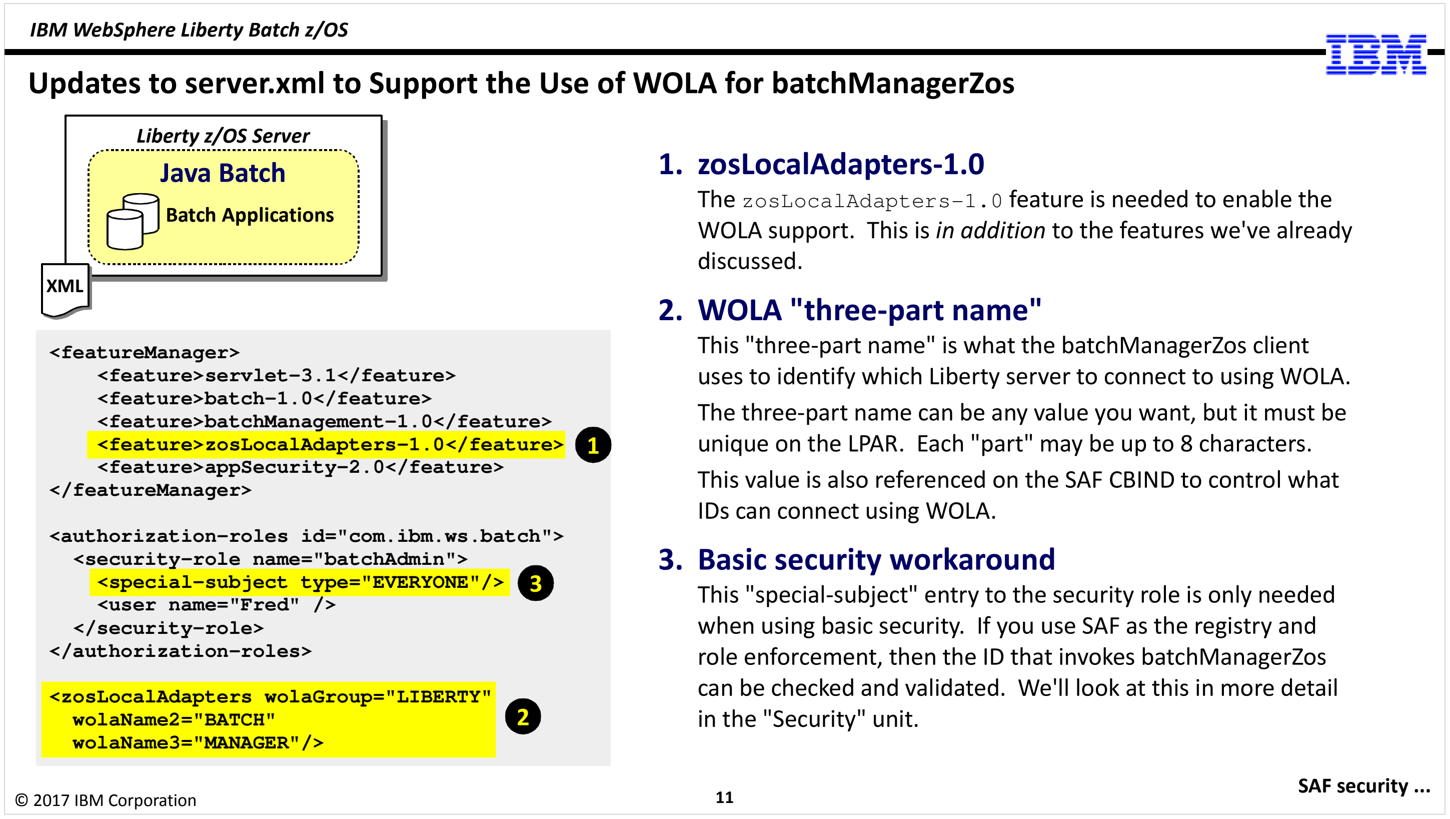
Updates to server.xml ...

© 2017 IBM Corporation 10

The batchManagerZos client is located in a different place under the install path ... not under /bin but rather under the path shown on the top of the chart. This is a z/OS-only thing, so there is no option for "bat" extensions for Windows. This can be invoked from a z/OS shell environment -- Telnet, SSH, OMVS, or BPXBATCH in JCL.

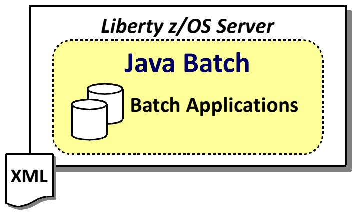
Let's take a high-level tour by following the numbered circles:

1. The batchManagerZos client is *not* Java-based, it is native code, so there's no need to have JAVA_HOME in the environment. But more to the point, this does *not* instantiate a JVM each time it is invoked like batchManager does, so the overhead associated with building a JVM each time is avoided.
2. batchManagerZos uses WebSphere Optimized Local Adapters (WOLA) to communicate with the Liberty z/OS server. WOLA is very fast, does not use the TCP/IP stack at all, but it limits you to having batchManagerZos on the same LPAR as the Liberty z/OS server you are talking to.
3. The use of WOLA by batchManagerZos means the Liberty z/OS server must also support WOLA. This is done with some updates to the server.xml file. We'll show you what that looks like in a few charts.
4. Because WOLA is a "z/OS authorized service," the Angel Process is required to provide the access to the authorized service for the server. The Angel Process is a started task that protects access to specified services. We'll see some details of that in a few charts, and go into more detail in Unit 6 on security.
5. Finally, the use of WOLA triggers the need for a few SAF profiles; specifically, SERVER profiles to grant the Liberty z/OS server access to the authorized services, and a CBIND profile to control who can use the WOLA register function to connect to the Liberty z/OS server.



IBM WebSphere Liberty Batch z/OS

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>   1
    <feature>appSecurity-2.0</feature>
</featureManager>

<authorization-roles id="com.ibm.ws.batch">
  <security-role name="batchAdmin">
    <special-subject type="EVERYONE"/>   3
    <user name="Fred" />
  </security-role>
</authorization-roles>

<zosLocalAdapters wolaGroup="LIBERTY"
  wolaName2="BATCH"
  wolaName3="MANAGER"/>   2
```

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 ...

© 2017 IBM Corporation 11

In order to use batchManagerZos against a Liberty z/OS server, the target server needs to support WOLA. There are three things that need to be added to server.xml to accomplish this:

1. The zosLocalAdapters-1.0 feature needs to be added to the feature list. This is *in addition* to the batch-1.0 and batchManagement-1.0 features we discussed earlier.
2. We need to add the <zosLocalAdapters> element and provide the "three-part name." This "three-part name" is used by the batchManagerZos client when it connects: it has to know which Liberty z/OS server to cross-memory connect to, and it does that with the three-part name. The three part name elements are really arbitrary, but it has to be unique on the LPAR. As part of planning for Liberty z/OS a review of naming conventions is recommended, and one way to achieve uniqueness of this three-part name is to make sure the server names are unique, and to employ the server name as the wolaName3= value.
3. Because we're using "basic security" we need to add an element to the <security-role> definition. The reason for this is because WOLA is coming cross-memory and the ID which is asserted across WOLA (the ID used to invoke batchManagerZos) can't be seen by the Java container that's running the basic security function. This is *only* because we're using basic security at this point in the workshop; this is *not* needed when we use SAF security as discussed in Unit 6.

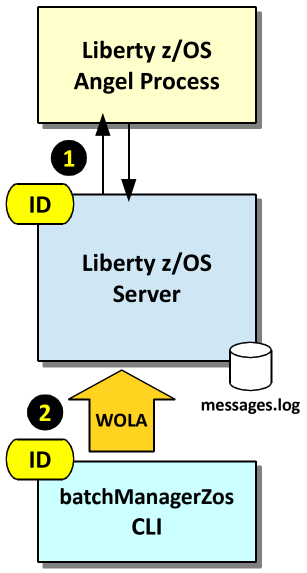
   The addition we need to add is <special-subject> and name EVERYONE. This allows the basic security to work with batchManagerZos.

   Note: for those who may be concerned this represents a security risk, we offer several points: (a) "basic" security is really intended for development and ad hoc testing, and not "real world". When using SAF things work as you'd expect. And (b) there's already quite a bit of security in play here: the ID that invokes batchManagerZos has already been authenticated because the utility can only be invoked on z/OS, and to get to the z/OS shell environment you have to authenticate, and the CBIND profile prevents just anyone from using batchManagerZos to connect to the Liberty server. So this EVERYONE special-subject is not much of an exposure for initial testing.

Let's look at the SAF security requirements for using WOLA ...

IBM WebSphere Liberty Batch z/OS

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 ...

The use of WOLA means the Angel process must be present, and it brings two key SAF profile requirements into play -- SERVER and CBIND.

The Angel Process is a started task that provides an "anchor point" for protecting access to z/OS authorized services. The sample JCL start procedure for the Angel is in the same location as the sample for the server itself. Copy that to your system proclib and customize by pointing to the install location for Liberty. Getting the Angel to start is easy; the more critical thing is creating the SAF SERVER profiles and granting READ to the Liberty z/OS Java Batch server ID. When that is done properly, you will see the highlighted messages come out in the messages.log file indicating the WOLA authorized services as "available." You will also see the "three part name" you defined show up in a message indicating the server has registered that name as a target for batchManagerZos connection. These arel all good signs.

Note: we are not showing the details of the SAF SERVER profiles or the CBIND profiles. We'll give you that detail back in Unit 6. For the upcoming lab we'll have you submit a job we created ahead of time that will create the profiles.

The next piece of this puzzle is a CBIND profile that protects which IDs running batchManagerZos can connect to the server. The CBIND profile is based on the three-part name the server is configured with. The ID that will run batchManagerZos is granted READ to that profile, and that's what allows them to "register in" to the Liberty z/OS server. If someone else tries to run batchManagerZos they will be denied because their ID does not have READ to the CBIND profile.

IBM WebSphere Liberty Batch z/OS

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

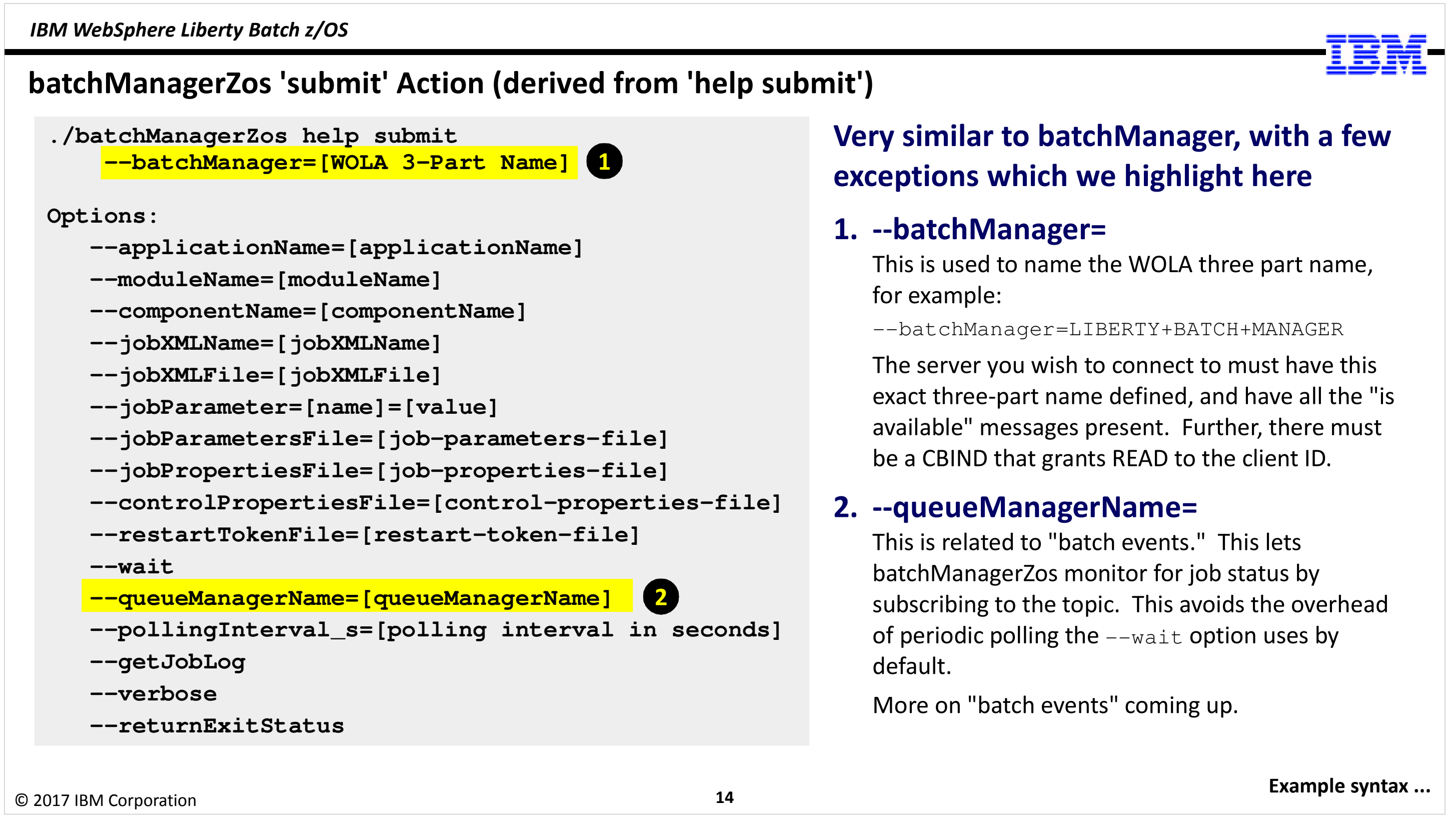
Actions are very similar to those provided with batchManager CLI

'submit' action ...

Here we're showing you the batchManagerZos command with the 'help' action. The actions available to batchManagerZos are very similar to what we saw with batchManager, but there are some differences. For example, note the 'ping' action ... this is a verification action to see whether the WOLA connectivity is there.

Note: 'ping' will return silence when it works; it'll return an error message if connectivity to the Liberty z/OS can't be achieved.

Let's look at the 'submit' action for batchManagerZos ...



IBM WebSphere Liberty Batch z/OS

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

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

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]  ❷
   --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.

© 2017 IBM Corporation 14

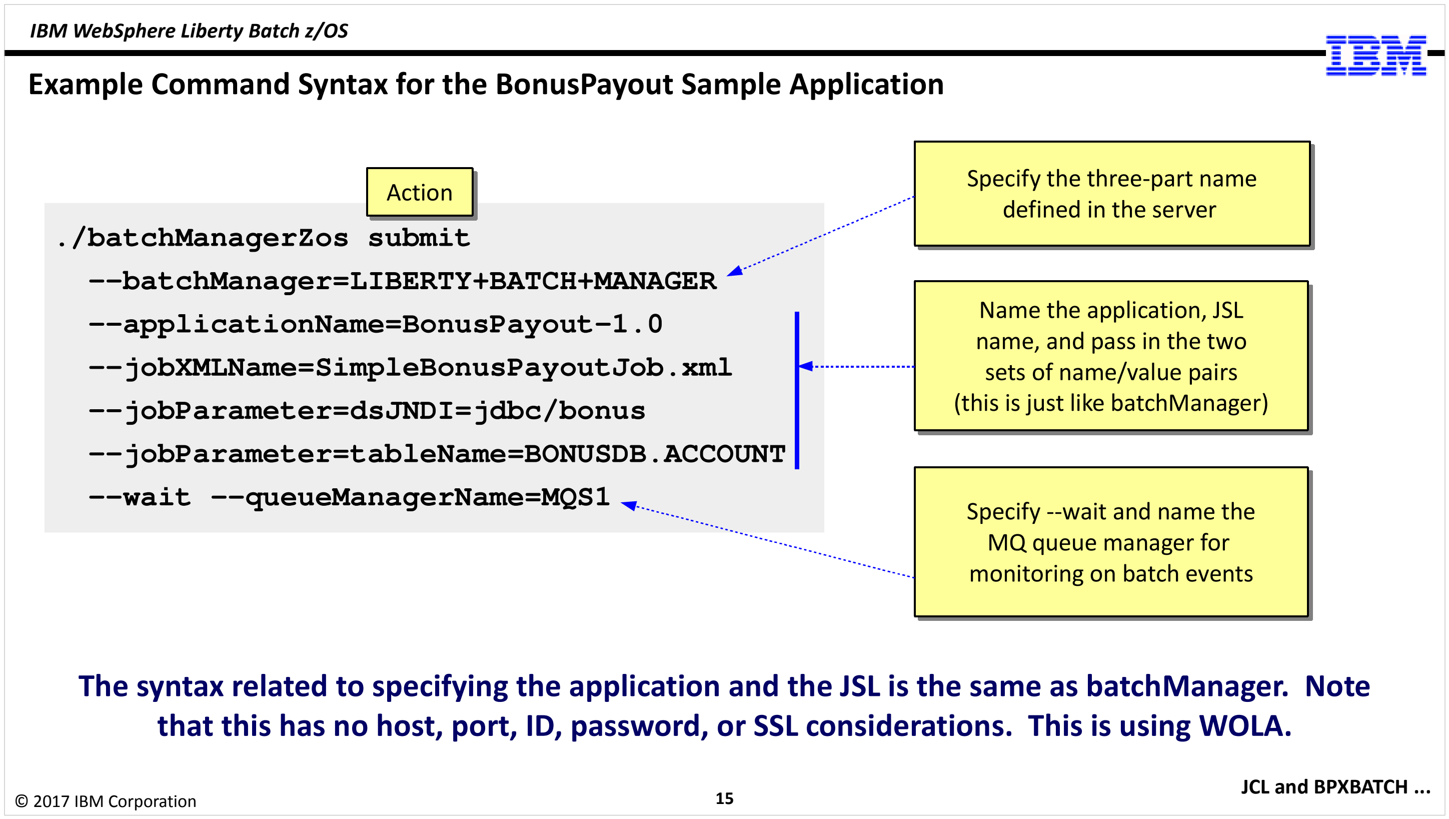
Example syntax ...

Here again we're showing you the results of a 'help submit' action, but this time for batchManagerZos. We have again removed the verbose descriptions to show just the command options. This is very similar to what we saw with batchManager, but there are a few interesting new things. We'll look at the new things and skim over the ones that are identical to what we saw with batchManager.

- --batchManager= is used to point to the three-part name being hosted by the Liberty z/OS server. You separate the names with a plus sign (+) with no spaces. So for example: LIBERTY+BATCH+MANAGER, or whatever your three part name is.

Note: there is no need for userid/password because the ID that invokes batchManagerZos will be asserted cross-memory into the server. We know that ID is a valid z/OS ID because the user would not be able to get to a z/OS shell environment without authenticating sometime prior to invoking batchManagerZos. Oh, and there's no need for SSL because there's no network.

- --queueManager= is optional, and is used to name the MQ queue manager which is hosting the batch events topic. This is used to monitor for job completion without having to poll the server like batchManager does. With this specified, the batchManagerZos client subscribes to the topic and watches for one of several different batch events to know when the job completes.



IBM WebSphere Liberty Batch z/OS

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

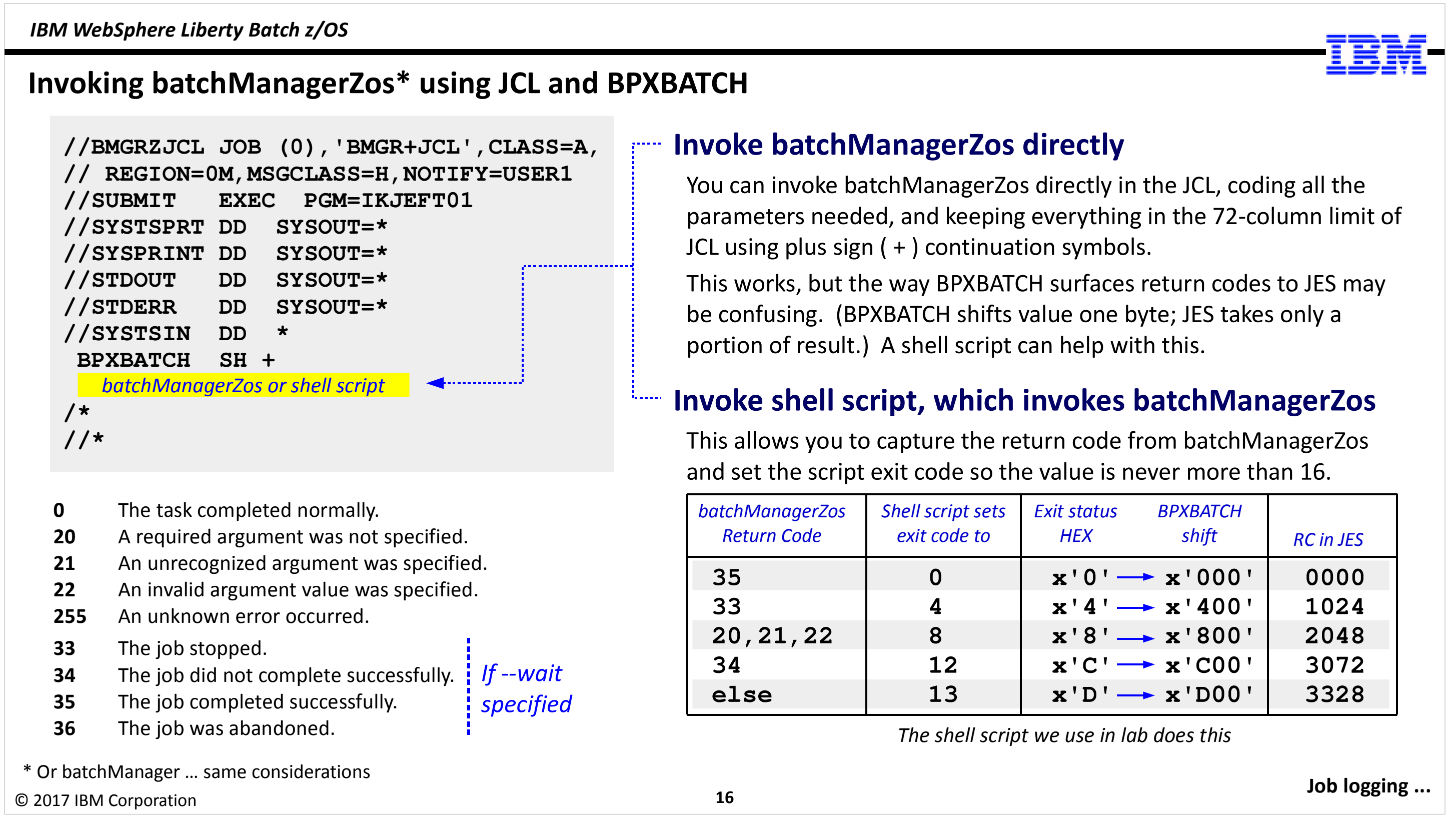
Name the application, JSL name, and pass in the two sets of name/value pairs (this is just like batchManager)

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.

© 2017 IBM Corporation 15 **JCL and BPXBATCH ...**

Here's an example of the command syntax for batchManagerZos to submit the same BonusPayout application we saw for the batchManager example. It's very similar. The difference is the three-part name rather than host:port, and here we're showing the use of the optional --queueManagerName function to monitor for the batch events. We have more on batch events later in this unit.



IBM WebSphere Liberty Batch z/OS

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
/*
//*
```

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.

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

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

© 2017 IBM Corporation 16

Job logging ...

You can invoke batchManagerZos (or batchManager) from JCL using BPXBATCH. This is what you would do to integrate an enterprise scheduler on z/OS with the Java Batch runtime in Liberty z/OS. The scheduler would submit a JCL job which uses batchManagerZos to submit the job. You would typically code --wait on the batchManagerZos command so the JCL job would stay active as long as the Java batch job is active in Liberty.

You can invoke batchManagerZos (or batchManager) using BPXBATCH by either invoking the command directly, or by invoking a shell script which invokes the command. The benefit of invoking a shell script which then invokes the utility is it allows you greater control over setting the exit code that surfaces to JES. This requires a bit of explanation.

BPXBATCH is going to shift up one byte the return code it sees from batchManagerZos. That is what BPXBATCH does. So take, for example, the case where you run batchManagerZos with --wait, and everything runs as expected. The return code out of batchManagerZos will be 35, which is 0x23 (hex). BPXBATCH will take that 0x23 value and shift it to 0x2300. JES, on the other hand, will take only the last three digits from that -- 0x300 -- and surface a return code of 768. That's confusing enough; the real confusion comes in if some *other* return code gets shifted and the last three is the same 0x300. Then JES returns the same 768 for a *different* condition.

With a shell script we can eliminate a fair amount of this confusing by taking control of the exit code that's issued by the shell script and making sure no exit code is greater than 16. By doing that, we insure what JES receives is the same two-byte hex that comes out of the shell script.

The chart shows the return codes that batchManagerZos (or batchManager; they use the same return codes) will issue under different circumstances. When invoking batchManagerZos from JCL, we would code --wait because we want the JCL job to stay active while the Java batch is running. So in our shell script we would want to set rc=35 to shell script exit=0. 0 is the universal "things are good" code. BPXBATCH shifts that, JES strips the last three, and the result is a JES return code of 0. Yea! :-)

batchManagerZos return code of 33 means someone stopped the job. That's not necessarily a failure. So in our shell script we can set that value to an exit of 4. BPXBATCH shifts that, JES strips the last three, and the result is a JES return code of 1024. In this case you (and the enterprise scheduler) need to understand that JES RC=1024 means the shell script issued exit 4.

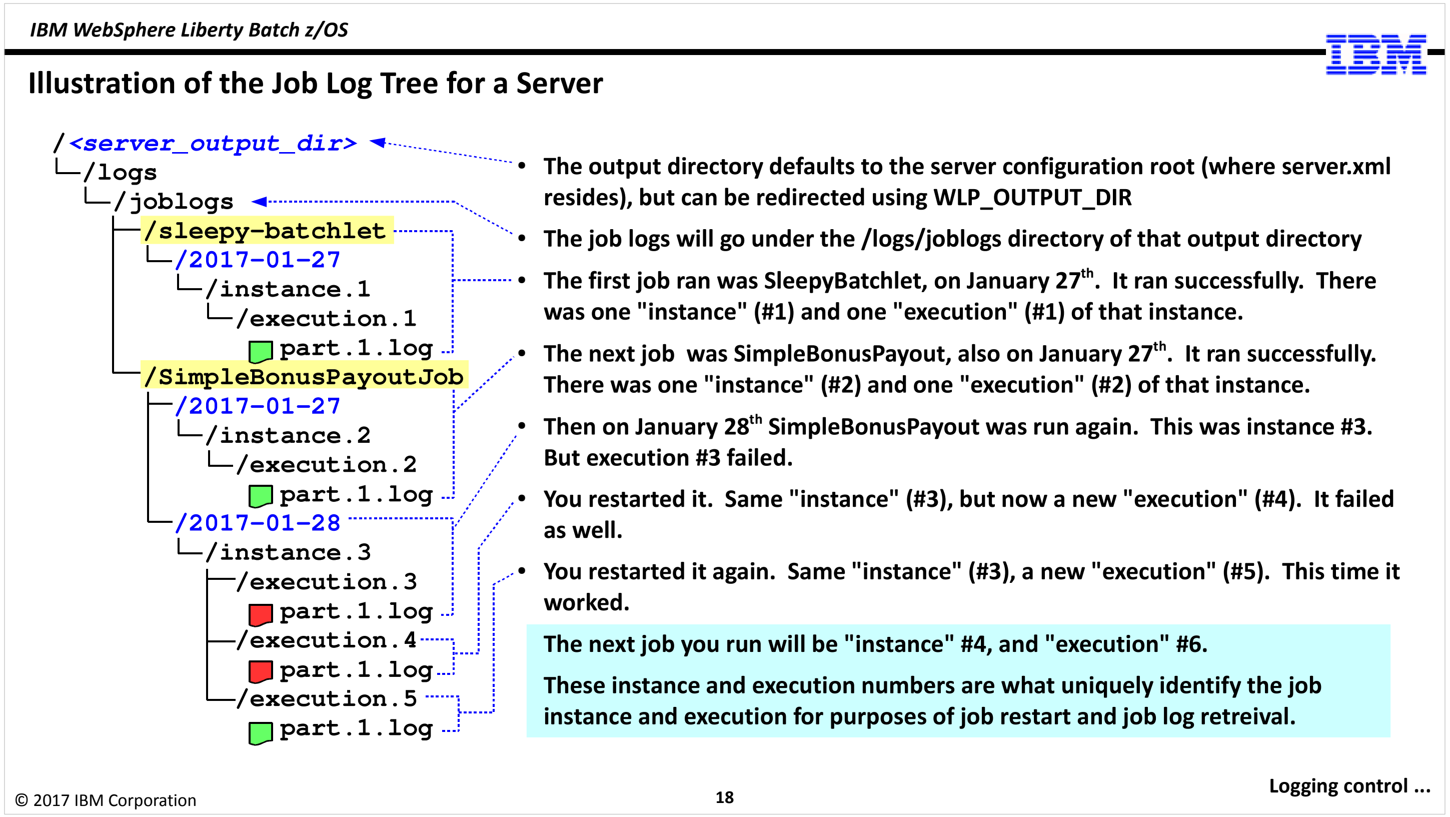
batchManagerZos return codes 20, 21, and 22 are used when there's something wrong with the command -- a required argument was missing, an unrecognized argument (a typo, for example) was specified, or an invalid value for an argument was used. That's not a problem with the Java batch code itself; those are problems with the job submission. So we'll set a return code of 8 for those. BPXBATCH shifts, JES strips the last three, and we get a JES return code 2048.

batchManagerZos return code 34 is the case where it was successfully submitted, but it did not complete. Something is wrong inside the code. For that we set an exit of 12, which is hex 0xC. BPXBATCH shifts, JES strips the last three, and the JES return code is 3072.

Any other problem results in an exit code of 13. If you see a JES return code of 3328, you know it's something not otherwise captured in the other exit codes.

That's the value of the shell script. We're going to use a shell script like that in the lab that's coming up.

Job Logging



IBM WebSphere Liberty Batch z/OS

Illustration of the Job Log Tree for a Server

```
/<server_output_dir>
└─/logs
  └─/joblogs
    ├─/sleepy-batchlet
    │ └─/2017-01-27
    │   └─/instance.1
    │     └─/execution.1
    │         part.1.log
    └─/SimpleBonusPayoutJob
      ├─/2017-01-27
      │ └─/instance.2
      │   └─/execution.2
      │       part.1.log
      └─/2017-01-28
        └─/instance.3
          ├─/execution.3
          │   part.1.log
          ├─/execution.4
          │   part.1.log
          └─/execution.5
              part.1.log
```

- 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.

© 2017 IBM Corporation 18 Logging control ...

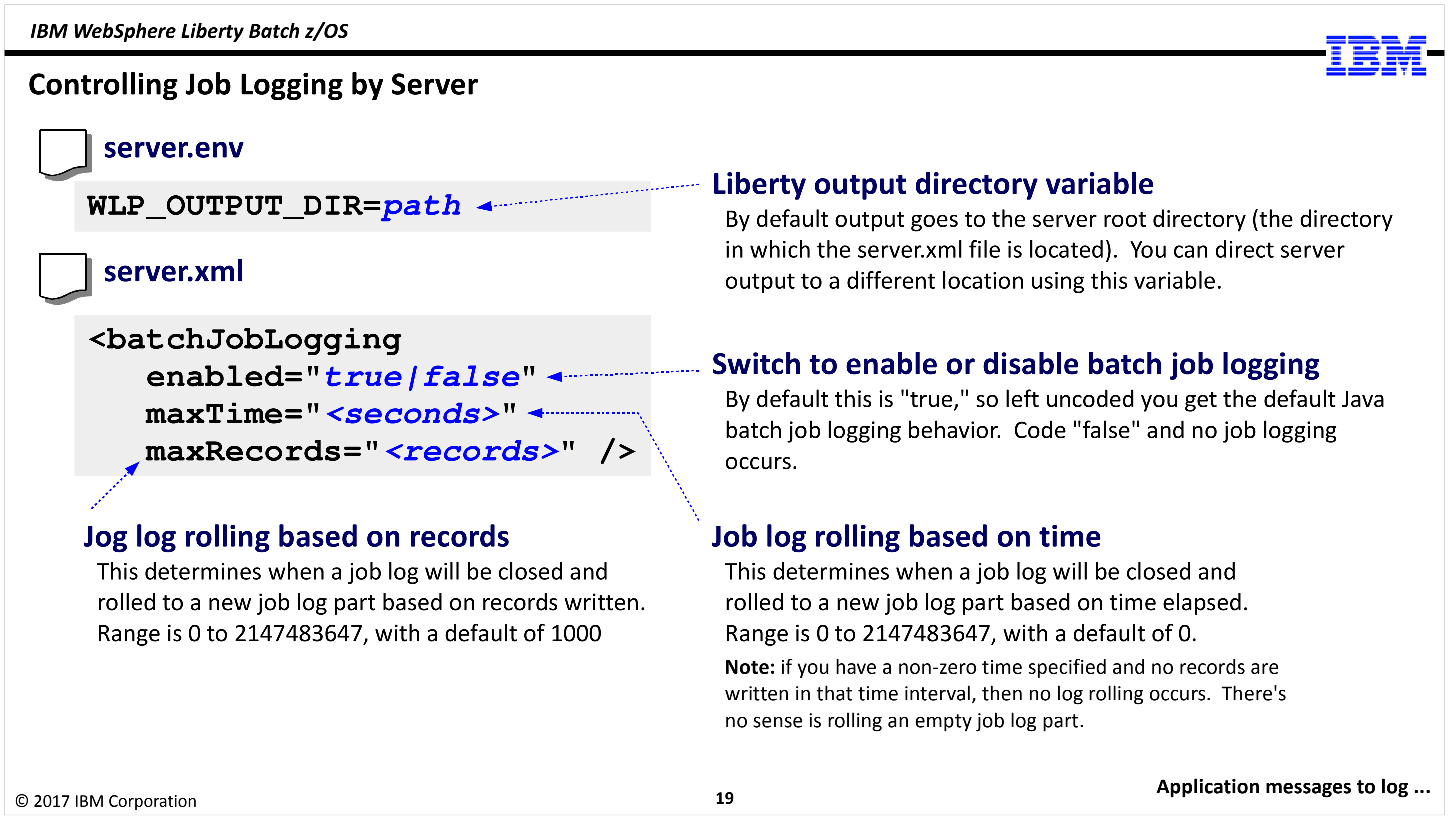
We mentioned earlier that the JSR-352 specification mentions the importance of job logging, but it does not specify how it should be implemented. It leaves that up to the vendors to implement as they see best.

Also earlier we described the JSR-352 terminology of "job instance ID" and "job execution ID" mentioned how those things are important because they tie to job logging. Well, here's an example of a job log tree created by the IBM WebSphere Liberty Java Batch function for the running of two jobs -- both SleepyBatchlet and SimpleBonusPayoutJob -- across two days. There's a lot of words on this chart, but let's take a few moments and go through it:

- First, by default the logs are written under the server's /logs directory under the server configuration directory. But the WLP_OUTPUT_DIR variable can be used to redirect the server's output -- *all* the output, not just batch job logs -- to any location you wish.
- The job logs will appear under /joglogs
- The next layer down is the application name. We have two applications illustrated here. Let's focus first on the SleepyBatchlet job logs. On January 27th, 2017 the job was submitted. The runtime assigned a job instance number of "1" and then it create a job execution number of "1" as well. (That's because the JobRepository tables was fresh and new when this test was run.) The job ran to completion, and that was the last time this job was run.
- The next job is the SimpleBonusPayout job. It was run on both January 27th and January 28th, but with different results. On January 27th it was submitted and it ran to successful completion. It was assigned a job instance number of "2", which was the next job *instance* number based on the information in the JobRepository. The job execution number was also "2" because it was the next job *execution* number based on the JobRepository.
- Then on January 28th the SimpleBonusPayout job was run again. It was assigned a job instance number of "3" and a job execution number of "3". But this execution failed for some reason. You investigated, discovered the problem, corrected it and *restarted* the job. This *restart* of the job results in a new execution number under the same instance number, but it gets a job execution number of "4".

  Note: it's tempting to think instance and execution numbers march up in lock-step, but that's not necessarily the case. It would if every job submission went to successful completion and you never had to restart a job. But that's not real-world ... sometimes things happen and you have restart. That's when the numbers cease being the same value. The two numbers are in fact not related to each other at all. They both start at "1" but that's the only relationship between them.

  But it too fails. So you investigate, correct and *restart again*. This creates yet another *execution* number ("5") but under the same *instance* number ("3"). This illustrates how a given job instance can have multiple executions under it.



IBM WebSphere Liberty Batch z/OS

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.

© 2017 IBM Corporation 19

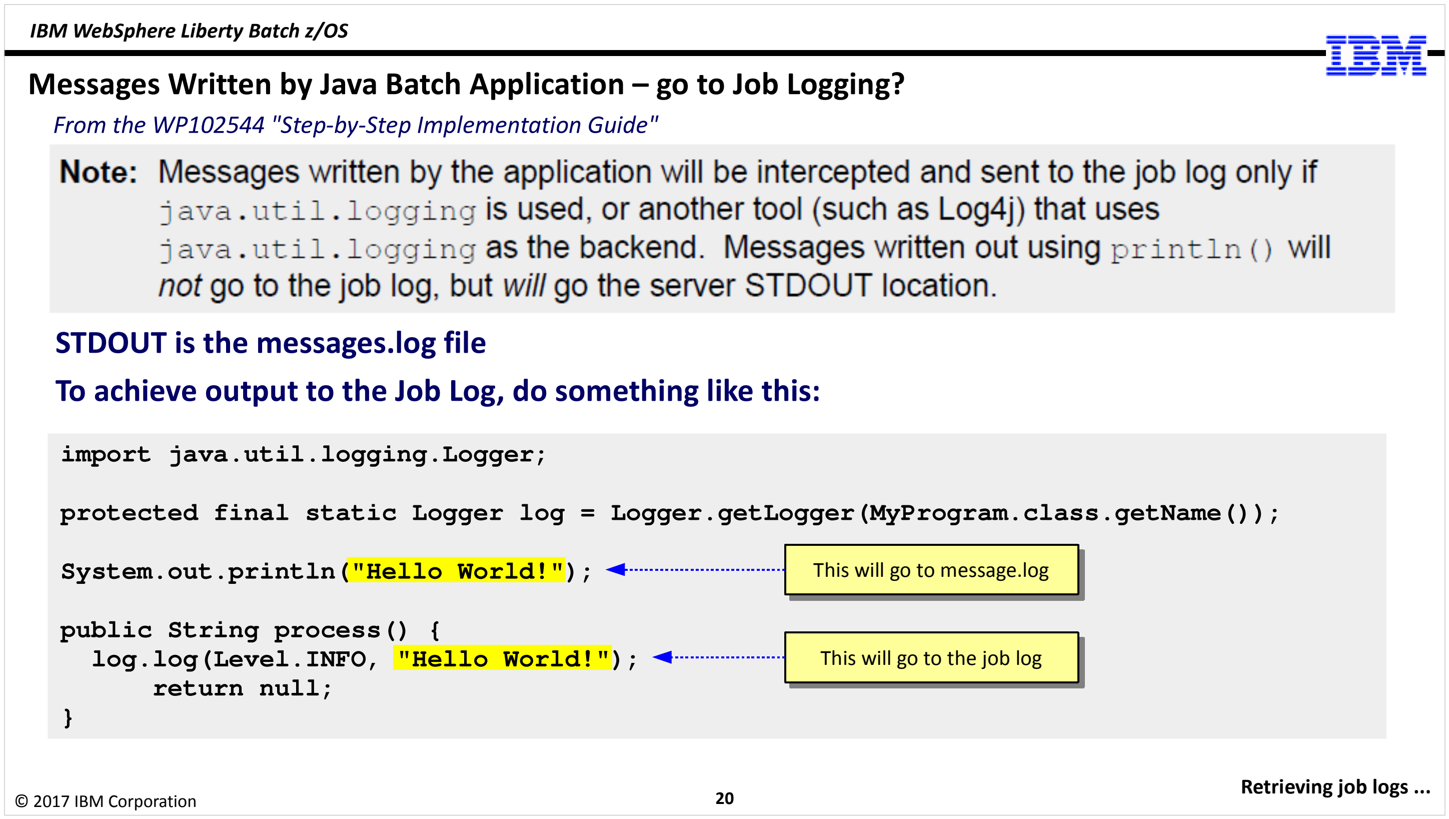
Application messages to log ...

You have the ability to control the behavior of this job logging.

The first thing is what we mentioned on the previous chart -- we can control where the server output goes with the WLP_OUTPUT_DIR variable coded in the server.env file. This will redirect all the server's output to whatever location you specify, not just the jog logging directory. This is useful function when you have a policy that says the server STC ID can't be the same as the configuration file owner ID. In that case we need to redirect the server output (a "write" operation) to a different location.

Then in the server.xml file we have the <batchJogLogging> element, which controls the behavior of Java Batch job logging:

- enabled="truth|false" ... this is a switch that turns logging off. By default this is "true." But if for some reason you have a batch server you don't want to produce job logs, you can disable logging with a "false" setting here.
- maxRecords= ... this determines when the job log file is closed and a new job log part file is opened. The default is "1000" which means every 1000 records written to the job log will result in a close of the file and a new job log part created. The SleepyBatchlet writes about 20 lines of output, so we only saw one job log part.
- maxTime= ... this provides another way to determine when a job log part is closed and another. This is based on a time interval. This defaults to "0" which means no time interval is in effect.



IBM WebSphere Liberty Batch z/OS

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

© 2017 IBM Corporation 20 **Retrieving job logs ...**

We have to be careful in our understanding of what application messages go to the job log. A simple println() will not go to the job log, but it will go to the STDOUT, which is the messages.log file.

To get application messages to go to the job log you have to use java.util.logging. The example in the bottom half the chart illustrates this in action: java.util.logging.Logger is imported, a log object is instantiated, then that's used to log the message. The println() will go to messages.log, the lower one goes to the job log file.

Ways to Retrieve the Job Logs

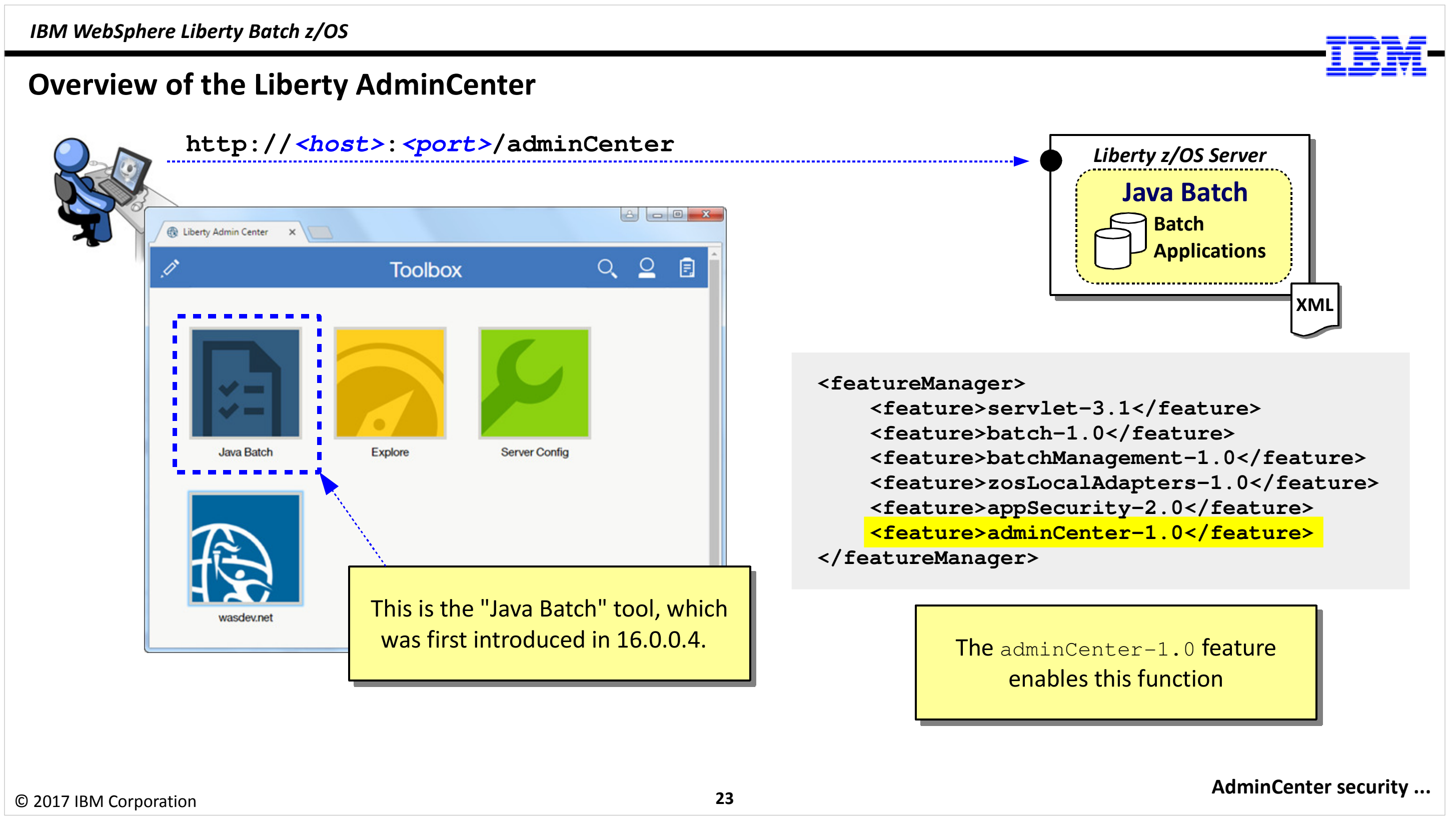
1. **Directly**
   The job log parts are written to the directory location we showed earlier. If you have access, you are free to go into that directory and retrieve the job log parts. If multiple job log parts, you must piece them back together yourself.
2. **batchManager 'getJobLog' action**
   This 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 ...

How do you retrieve the job logs once they've been written? There are six ways that can be done, and the chart spells out those approaches. They range from the most direct ("go into the file system and get the log") to the more sophisticated ("turn on batch events and subscribe to the topic and pull the jobLogPart events").

Note: if you're interested in getting the job log output "as it happens," then use batchManagerZos with batch events enabled and the --queueManagerName= value specified on the job submission. That tells batchManagerZos to subscribe to the topic and watch for the job log parts as they're published.

AdminCenter



IBM WebSphere Liberty Batch z/OS

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>
```

This is the "Java Batch" tool, which was first introduced in 16.0.0.4.

The `adminCenter-1.0` feature enables this function

© 2017 IBM Corporation 23

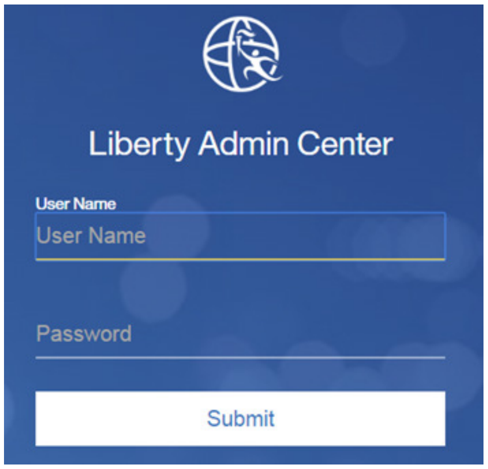
AdminCenter security ...

The "AdminCenter" is a graphical interface to the Liberty server. It is designed as an extensible framework into which "tools" can be installed to do different management functions. The AdminCenter is enabled with the adminCenter-1.0 feature.

One of the tools is a "Java Batch" tool. This provides the ability to view jobs and retrieve job logs. Before showing you that, however, we're going to once again dip into the topic of security.

IBM WebSphere Liberty Batch z/OS

AdminCenter Security ("Basic" for now)



Basic Keystore (SSL)

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

Basic User Registry

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

Authorization roles for Java Batch

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

Authorization role for AdminCenter

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

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.

Java Batch tool ...

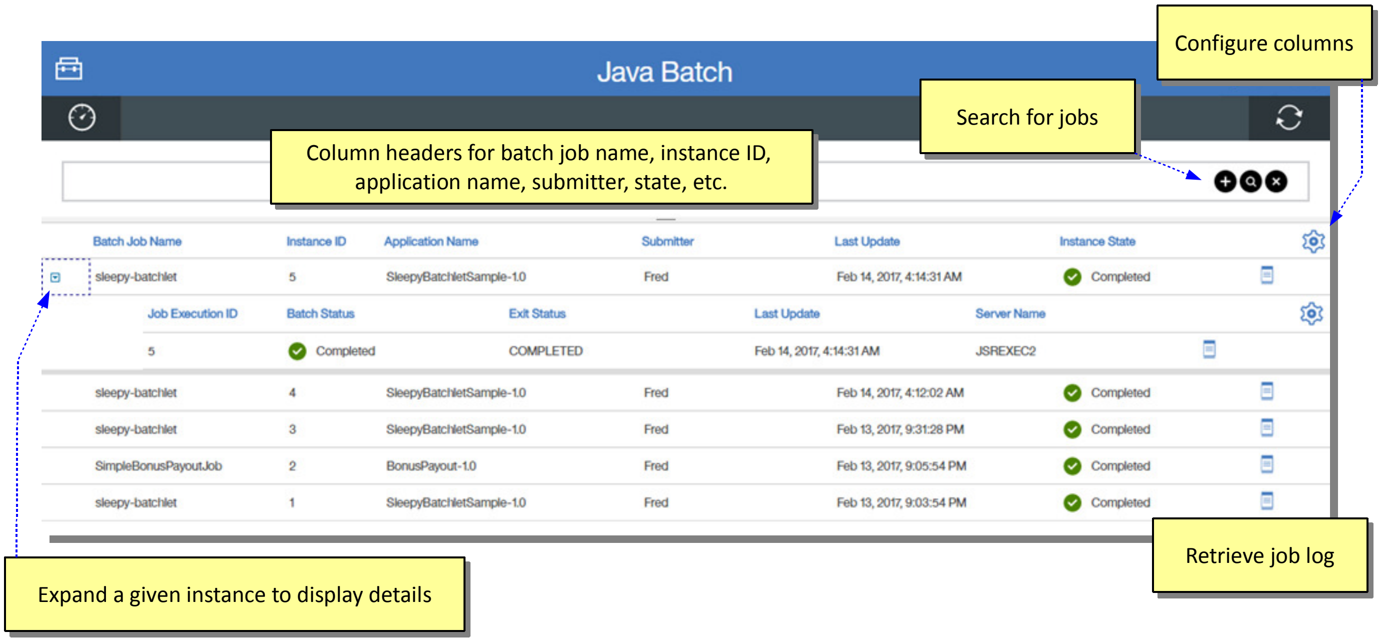
The AdminCenter imposes a security requirement -- it is a "protected" application, and requires encryption, authentication, and role authority to access it.

Earlier we discussed the "basic" security we put in place for batchManager-1.0. Thankfully that takes care of most of the security requirements of the AdminCenter as well. The only new thing we have to add is the <administrator-role> element with Fred given access to it.

In Unit 6 we'll dig into how to push all this down into SAF security.

IBM WebSphere Liberty Batch z/OS

The Java Batch Tool Display (in 16.0.0.4)


Java Batch
Configure columns
Search for jobs
Column headers for batch job name, instance ID, application name, submitter, state, etc.
Retrieve job log
Expand a given instance to display details

Batch Job Name	Instance ID	Application Name	Submitter	Last Update	Instance State
sleepy-batchlet	5	SleepyBatchletSample-1.0	Fred	Feb 14, 2017, 4:14:31 AM	✓ Completed
sleepy-batchlet	4	SleepyBatchletSample-1.0	Fred	Feb 14, 2017, 4:12:02 AM	✓ Completed
sleepy-batchlet	3	SleepyBatchletSample-1.0	Fred	Feb 13, 2017, 9:31:28 PM	✓ Completed
SimpleBonusPayoutJob	2	BonusPayout-1.0	Fred	Feb 13, 2017, 9:05:54 PM	✓ Completed
sleepy-batchlet	1	SleepyBatchletSample-1.0	Fred	Feb 13, 2017, 9:03:54 PM	✓ Completed

Job Execution ID	Batch Status	Exit Status	Last Update	Server Name
5	✓ Completed	COMPLETED	Feb 14, 2017, 4:14:31 AM	JSREXEC2

© 2017 IBM Corporation 25 **Batch events ...**

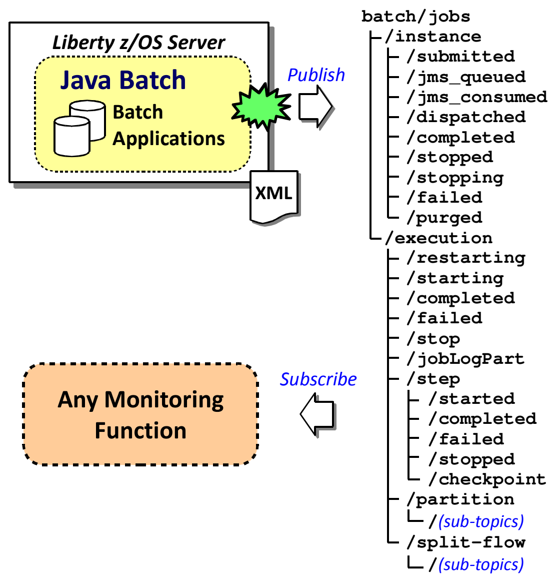
Here's a bitmap captures of the AdminCenter Java Batch tool showing the display of jobs. It's arranged in a column and row format, with each major row representing a different job instance, and the columns representing information about the job -- the Instance ID, the application name, the submitter ID, the last update, the state, and a little icon that can be used to retrieve the job log. Each major row can be expanded to show the executions under the instance. The search bar lets you search for a specific job.

It's a handy addition to the other tools available to you for managing Java Batch jobs.

Batch Events

IBM WebSphere Liberty Batch z/OS

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).

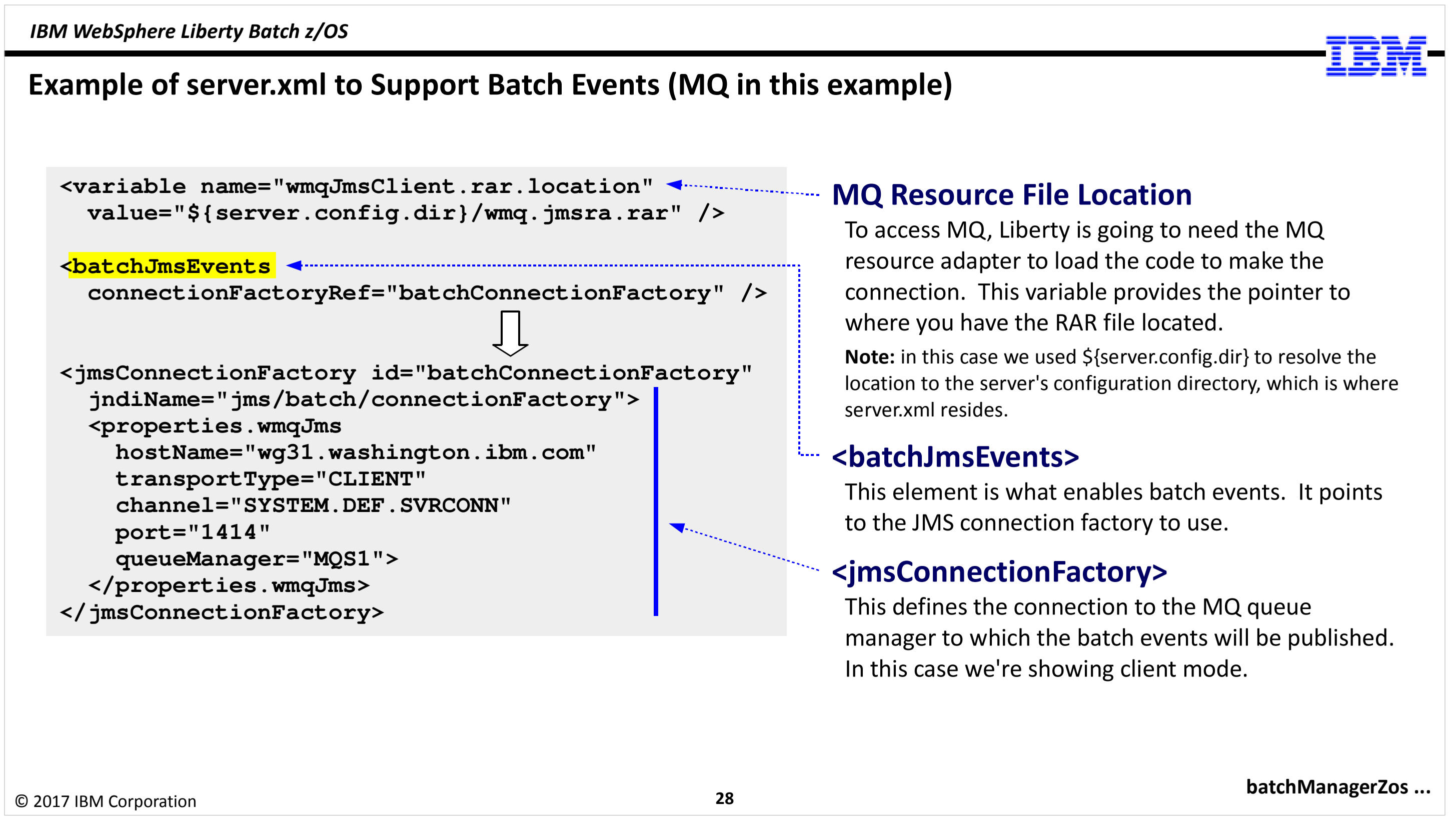
KC Search `twlp_batch_monitoring`

server.xml ...

This is the final topic in the unit. File this under "useful tools to monitor what's going on with my batch jobs."

Batch "events" are messages that are published to a JMS "topic" by the Liberty server as it cycles through the various states of a job, from submission, through execution, to completion. This allows any function capable of subscribing to the topic to monitor activity by watching for events in whatever part of the topic tree they are interested.

This requires server.xml updates to provide the JMS definitions so the server can reach the messaging engine used -- which can be either the default messaging of Liberty, or IBM MQ. We'll look at a sample of the server.xml updates next.



IBM WebSphere Liberty Batch z/OS

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>

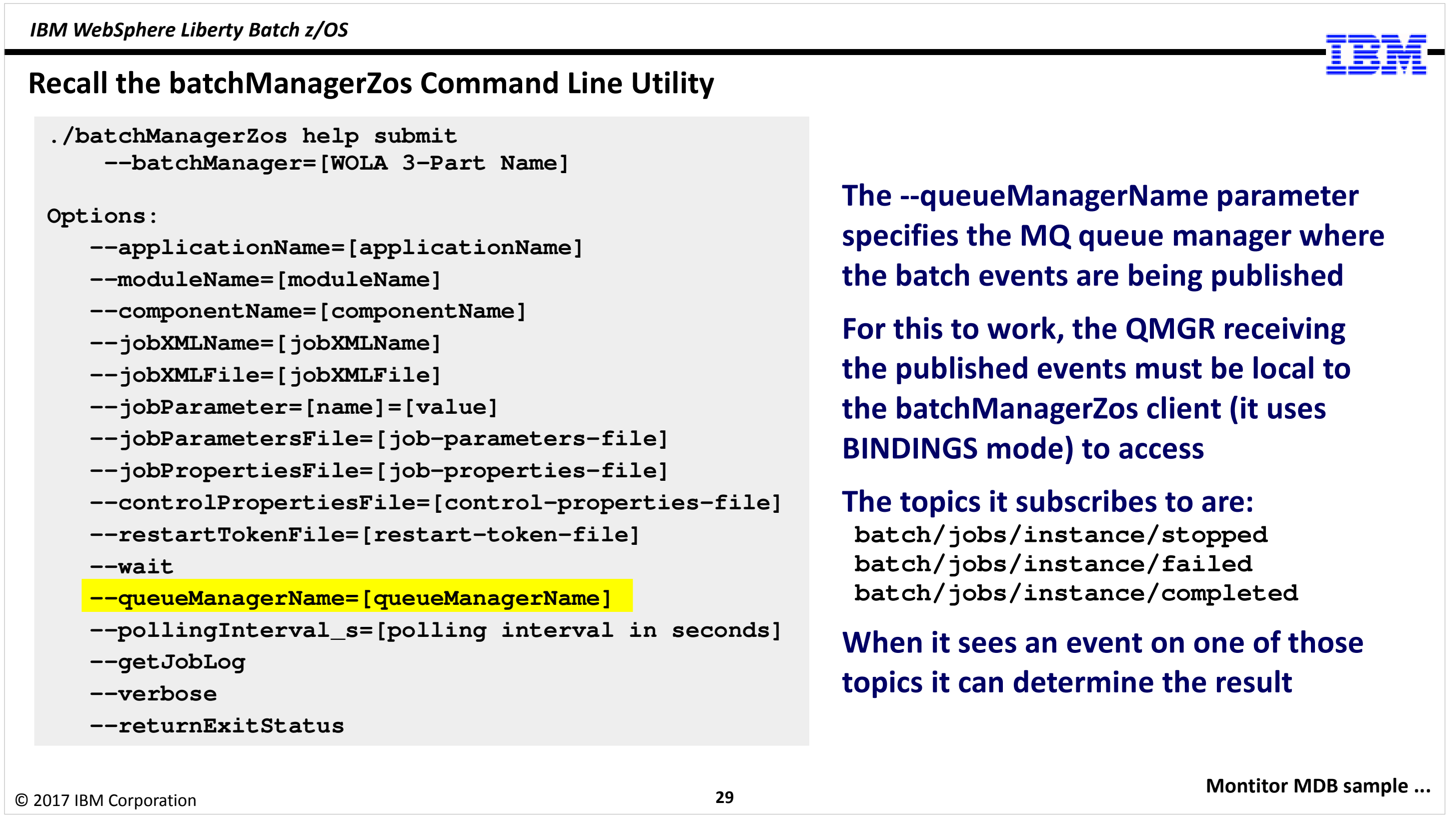
This defines the connection to the MQ queue manager to which the batch events will be published. In this case we're showing client mode.

© 2017 IBM Corporation 28 **batchManagerZos ...**

Here's an example of the server.xml updates to support batch events to an IBM MQ queue manager. At the topic is a <variable> element that defines the location of the MQ JMS resource adapter archive file. Liberty needs that to get the Java classes it needs to make the connection to MQ.

The <batchJmsEvents> element enables the batch events function. It has a pointer to the JMS connection factory for the definitions of the MQ queue manager it will publish to.

The <jmsConnectionFactory> element is pointed to from the <batchJmsEvents> element, and this defines how the server will get to the MQ queue manager. In this example we're showing CLIENT mode connection, which uses the network to communicate with MQ.



IBM WebSphere Liberty Batch z/OS

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

© 2017 IBM Corporation 29 **Montitor MDB sample ...**

We saw this earlier when we were exploring the batchManagerZos subject. That command line interface client has the ability to monitor the topic tree to determine the status of a submitted job. That's done with the --queueManagerName = option where you name the QMGR subsystem name. This must be done along with the --wait option.

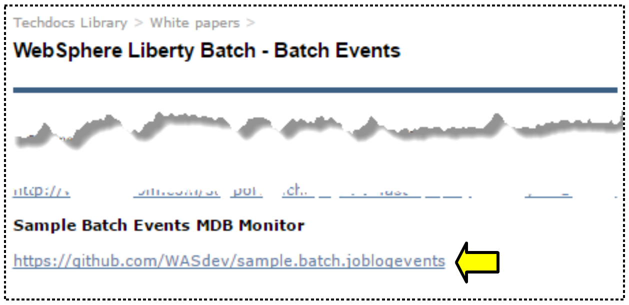
Doing this eliminates the need to poll the Liberty z/OS server periodically to determine the state of the job. It simply monitors the topic tree and when it sees the event showing either stopped, failed, or completed it knows the job is done.

This uses BINDINGS mode to connect to the MQ queue manager, so that means the batchManagerZos instance must be on the same LPAR as the queue manager that has the batch events topic.

IBM WebSphere Liberty Batch z/OS

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

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


Techdocs Library > White papers >

WebSphere Liberty Batch - Batch Events

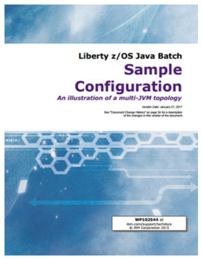
Sample Batch Events MDB Monitor

https://github.com/WASdev/sample.batch.joblogevents

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

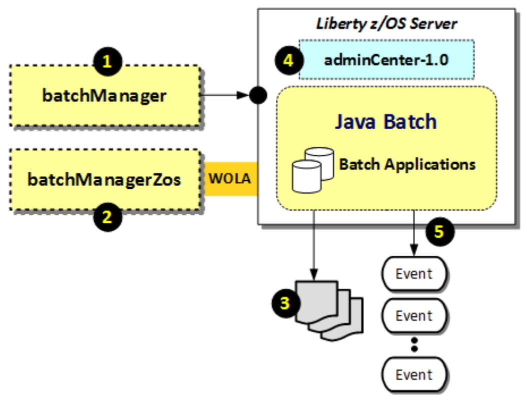
Summary ...

30

The WP102603 Techdoc has a link to a sample MDB application that monitors the jogLogPart topic and retrieves job log parts as they're published. It then writes the job logs into the /logs directory of the Liberty server that's hosting the MDB application. The use of this sample is illustrated in the WP102544 "Sample Configuration" guide.

IBM WebSphere Liberty Batch z/OS

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?

31

This unit covered a lot of topics ... five, as a matter of fact. The focus of this unit was on job submission, jog logging, and batch events.

The upcoming lab focuses on those same topics.

End of Unit