## IBM WebSphere Liberty Java Batch

# Lab Artifacts

***Important Disclaimer***

The examples provided in this document is intended to be *samples*, and is provided "as is" without any implied assurance or warranty of it working in your environment. You are responsible for inspecting the provided samples and making any modifications necessary to match your local environment and local security policies. IBM is *not* responsible for any use of these samples in any environment other than the specific IBM lab environment used by the ZJBATCH workshop.

*Lab Version Date: February 15, 2018*

# Table of Contents

# Lab 2 - Server Creation and Setup

## *RACF job*

The following JCL job created the RACF group and users, as well as the `STARTED` profiles. In addition, it created the `SURROGAT` profile so the `USER1` ID would have the authority to switch user to the server ID (`JSRSERV`) without requiring a password.

```
//LAB2    JOB (????,????),'LAB2 JOB',MSGCLASS=O,CLASS=A,
//         NOTIFY=????????,REGION=4000K TYPRUN=HOLD
//*-------------------------------------------------------------*
//RACF EXEC PGM=IKJEFT01,REGION=0M
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
ADDGROUP JSRGRP OMVS(AUTOGID) OWNER(SYS1)

ADDUSER JSRANGL DFLTGRP(JSRGRP) OMVS(AUTOUID HOME(/shared/jsrhome/) -
  PROGRAM(/bin/sh)) NAME('LIBERTY ANGEL')  NOPASSWORD NOOIDCARD

ADDUSER JSRSERV DFLTGRP(JSRGRP) OMVS(AUTOUID HOME(/shared/jsrhome) -
  PROGRAM(/bin/sh)) NAME('LIBERTY SERVER')

ALTUSER JSRSERV NOPASSWORD

RDEFINE STARTED JSRZANGL.* UACC(NONE) -
  STDATA(USER(JSRANGL) GROUP(JSRGRP) -
  PRIVILEGED(NO) TRUSTED(NO) TRACE(YES))

RDEFINE STARTED JSRDISP.* UACC(NONE) -
 STDATA(USER(JSRSERV) GROUP(JSRGRP) -
 PRIVILEGED(NO) TRUSTED(NO) TRACE(YES))

RDEFINE STARTED JSREXEC1.* UACC(NONE)-
 STDATA(USER(JSRSERV) GROUP(JSRGRP) -
 PRIVILEGED(NO) TRUSTED(NO) TRACE(YES))

RDEFINE STARTED JSREXEC2.* UACC(NONE)-
 STDATA(USER(JSRSERV) GROUP(JSRGRP) -
 PRIVILEGED(NO) TRUSTED(NO) TRACE(YES))

RDEFINE STARTED JSRMON.* UACC(NONE)-
 STDATA(USER(JSRSERV) GROUP(JSRGRP) -
 PRIVILEGED(NO) TRUSTED(NO) TRACE(YES))

SETROPTS RACLIST(STARTED) REFRESH

RDEFINE SURROGAT BPX.SRV.JSRSERV
RDEFINE SURROGAT JSRSERV.SUBMIT
PERMIT BPX.SRV.JSRSERV CLASS(SURROGAT) ID(USER1) ACC(READ)
PERMIT JSRSERV.SUBMIT  CLASS(SURROGAT) ID(USER1) ACC(READ)

SETROPTS RACLIST(SURROGAT) REFRESH
/*
//*
```

## *JCL start procedures*

The **JSRDISP** server JCL follows. This was modified from the supplied sample BBGZSRV. The elements that were modified are highlighted:

```
//JSRDISP PROC PARMS='JSRDISP'
//*--------------------------------------------------------------------
//* This proc may be overwritten by fixpacks or iFixes.
//* You must copy to another location before customizing.
//*--------------------------------------------------------------------
//* INSTDIR - the path to the WebSphere Liberty Profile install.
//*           This path is used to find the product code and is
//*           equivalent to the WLP_INSTALL_DIR environment variable
//*           in the Unix shell.
//* USERDIR - the path to the WebSphere Liberty Profile user area.
//*           This path is used to store shared and server specific
//*           configuration information and is equivalent to the
//*           WLP_USER_DIR environment variable in the Unix shell.
//*--------------------------------------------------------------------
//  SET INSTDIR='/shared/zWebSphere/Liberty/V17001'
//  SET USERDIR='/shared/jsrhome'
//*--------------------------------------------------------------------
//* Start the Liberty server
//*
//* WLPUDIR - PATH DD that points to the Liberty Profile's "user"
//*           directory. If the DD is not allocated, the user
//*           directory location defaults to the wlp/usr directory
//*           in the install tree.
//* STDOUT  - Destination for stdout (System.out)
//* STDERR  - Destination for stderr (System.err)
//* MSGLOG  - Destination for messages.log (optional)
//* STDENV  - Initial Unix environment - read by the system.  The
//*           installation default and server specific server
//*           environment files will be merged into this environment
//*           before the JVM is launched.
//*--------------------------------------------------------------------
//STEP1   EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT,
//  PARM='PGM &INSTDIR./lib/native/zos/s390x/bbgzsrv &PARMS'
//WLPUDIR  DD PATH='&USERDIR.'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//*MSGLOG   DD SYSOUT=*
//*STDENV   DD PATH='/etc/system.env',PATHOPTS=(ORDONLY)
//*STDOUT   DD PATH='&ROOT/std.out',
//*            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//*            PATHMODE=SIRWXU
//*STDERR   DD PATH='&ROOT/std.err',
//*            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//*            PATHMODE=SIRWXU
//* ================================================================ */
//* PROPRIETARY-STATEMENT:                                           */
//* Licensed Material - Property of IBM                              */
//*                                                                  */
//* (C) Copyright IBM Corp. 2011, 2012                               */
//* All Rights Reserved                                              */
//* US Government Users Restricted Rights - Use, duplication or      */
//* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.*/
//* ================================================================ */
```

The **JSREXEC1** and **JSREXEC2** start procedures follow. Elements modified from the BBGZSRV sample proc are highlighted:

```
//JSREXEC1 PROC PARMS='JSREXEC1'
//*--------------------------------------------------------------------
//* This proc may be overwritten by fixpacks or iFixes.
//* You must copy to another location before customizing.
//*--------------------------------------------------------------------
//* INSTDIR - the path to the WebSphere Liberty Profile install.
//*           This path is used to find the product code and is
//*           equivalent to the WLP_INSTALL_DIR environment variable
//*           in the Unix shell.
//* USERDIR - the path to the WebSphere Liberty Profile user area.
//*           This path is used to store shared and server specific
//*           configuration information and is equivalent to the
//*           WLP_USER_DIR environment variable in the Unix shell.
//*--------------------------------------------------------------------
//  SET INSTDIR='/shared/zWebSphere/Liberty/V17001'
//  SET USERDIR='/shared/jsrhome'
//*--------------------------------------------------------------------
//* Start the Liberty server
//*
//* WLPUDIR - PATH DD that points to the Liberty Profile's "user"
//*           directory. If the DD is not allocated, the user
//*           directory location defaults to the wlp/usr directory
//*           in the install tree.
//* STDOUT  - Destination for stdout (System.out)
//* STDERR  - Destination for stderr (System.err)
//* MSGLOG  - Destination for messages.log (optional)
//* STDENV  - Initial Unix environment - read by the system.  The
//*           installation default and server specific server
//*           environment files will be merged into this environment
//*           before the JVM is launched.
//*--------------------------------------------------------------------
//STEP1   EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT,
//  PARM='PGM &INSTDIR./lib/native/zos/s390x/bbgzsrv &PARMS'
//WLPUDIR  DD PATH='&USERDIR.'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//*MSGLOG   DD SYSOUT=*
//*STDENV   DD PATH='/etc/system.env',PATHOPTS=(ORDONLY)
//*STDOUT   DD PATH='&ROOT/std.out',
//*            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//*            PATHMODE=SIRWXU
//*STDERR   DD PATH='&ROOT/std.err',
//*            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//*            PATHMODE=SIRWXU
//* ================================================================ */
//* PROPRIETARY-STATEMENT:                                           */
//* Licensed Material - Property of IBM                              */
//*                                                                  */
//* (C) Copyright IBM Corp. 2011, 2012                               */
//* All Rights Reserved                                              */
//* US Government Users Restricted Rights - Use, duplication or      */
//* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.*/
//* ================================================================ */
```

```
//JSREXEC2 PROC PARMS='JSREXEC2'
//*--------------------------------------------------------------------
//* This proc may be overwritten by fixpacks or iFixes.
//* You must copy to another location before customizing.
//*--------------------------------------------------------------------
//* INSTDIR - the path to the WebSphere Liberty Profile install.
//*           This path is used to find the product code and is
//*           equivalent to the WLP_INSTALL_DIR environment variable
//*           in the Unix shell.
//* USERDIR - the path to the WebSphere Liberty Profile user area.
//*           This path is used to store shared and server specific
//*           configuration information and is equivalent to the
//*           WLP_USER_DIR environment variable in the Unix shell.
//*--------------------------------------------------------------------
//  SET INSTDIR='/shared/zWebSphere/Liberty/V17001'
//  SET USERDIR='/shared/jsrhome'
//*--------------------------------------------------------------------
//* Start the Liberty server
//*
//* WLPUDIR - PATH DD that points to the Liberty Profile's "user"
//*           directory. If the DD is not allocated, the user
//*           directory location defaults to the wlp/usr directory
//*           in the install tree.
//* STDOUT  - Destination for stdout (System.out)
//* STDERR  - Destination for stderr (System.err)
//* MSGLOG  - Destination for messages.log (optional)
//* STDENV  - Initial Unix environment - read by the system.  The
//*           installation default and server specific server
//*           environment files will be merged into this environment
//*           before the JVM is launched.
//*--------------------------------------------------------------------
//STEP1   EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT,
//  PARM='PGM &INSTDIR./lib/native/zos/s390x/bbgzsrv &PARMS'
//WLPUDIR  DD PATH='&USERDIR.'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//*MSGLOG   DD SYSOUT=*
//*STDENV   DD PATH='/etc/system.env',PATHOPTS=(ORDONLY)
//*STDOUT   DD PATH='&ROOT/std.out',
//*            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//*            PATHMODE=SIRWXU
//*STDERR   DD PATH='&ROOT/std.err',
//*            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//*            PATHMODE=SIRWXU
//* ================================================================ */
//* PROPRIETARY-STATEMENT:                                           */
//* Licensed Material - Property of IBM                              */
//*                                                                  */
//* (C) Copyright IBM Corp. 2011, 2012                               */
//* All Rights Reserved                                              */
//* US Government Users Restricted Rights - Use, duplication or      */
//* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.*/
//* ================================================================ */
```

The **JSRMON** start procedure follows. Elements modified from the BBGZSRV sample proc are highlighted:

```
//JSRMON  PROC PARMS='JSRMON'
//*--------------------------------------------------------------------
//* This proc may be overwritten by fixpacks or iFixes.
//* You must copy to another location before customizing.
//*--------------------------------------------------------------------
//* INSTDIR - the path to the WebSphere Liberty Profile install.
//*           This path is used to find the product code and is
//*           equivalent to the WLP_INSTALL_DIR environment variable
//*           in the Unix shell.
//* USERDIR - the path to the WebSphere Liberty Profile user area.
//*           This path is used to store shared and server specific
//*           configuration information and is equivalent to the
//*           WLP_USER_DIR environment variable in the Unix shell.
//*--------------------------------------------------------------------
//  SET INSTDIR='/shared/zWebSphere/Liberty/V17001'
//  SET USERDIR='/shared/jsrhome'
//*--------------------------------------------------------------------
//* Start the Liberty server
//*
//* WLPUDIR - PATH DD that points to the Liberty Profile's "user"
//*           directory. If the DD is not allocated, the user
//*           directory location defaults to the wlp/usr directory
//*           in the install tree.
//* STDOUT  - Destination for stdout (System.out)
//* STDERR  - Destination for stderr (System.err)
//* MSGLOG  - Destination for messages.log (optional)
//* STDENV  - Initial Unix environment - read by the system.  The
//*           installation default and server specific server
//*           environment files will be merged into this environment
//*           before the JVM is launched.
//*--------------------------------------------------------------------
//STEP1   EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT,
//  PARM='PGM &INSTDIR./lib/native/zos/s390x/bbgzsrv &PARMS'
//WLPUDIR  DD PATH='&USERDIR.'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//*MSGLOG   DD SYSOUT=*
//*STDENV   DD PATH='/etc/system.env',PATHOPTS=(ORDONLY)
//*STDOUT   DD PATH='&ROOT/std.out',
//*            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//*            PATHMODE=SIRWXU
//*STDERR   DD PATH='&ROOT/std.err',
//*            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//*            PATHMODE=SIRWXU
//* ================================================================ */
//* PROPRIETARY-STATEMENT:                                           */
//* Licensed Material - Property of IBM                              */
//*                                                                  */
//* (C) Copyright IBM Corp. 2011, 2012                               */
//* All Rights Reserved                                              */
//* US Government Users Restricted Rights - Use, duplication or      */
//* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.*/
//* ================================================================ */
```

The "Angel" start procedure (`JSRZANGL`) isn't used until the Unit 4 lab, but we'll show it here so it's with the other JCL start procs. This was modified from the supplied sample BBGZANGL proc. Elements modified are highlighted:

```
//JSRZANGL PROC PARMS='',COLD=N,NAME=''
//*-------------------------------------------------------------------
//  SET ROOT='/shared/zWebSphere/Liberty/V17001'
//*-------------------------------------------------------------------
//* Start the Liberty angel process
//*-------------------------------------------------------------------
//* This proc may be overwritten by fixpacks or iFixes.
//* You must copy to another location before customizing.
//*-------------------------------------------------------------------
//STEP1   EXEC PGM=BPXBATA2,REGION=0M,TIME=NOLIMIT,
//      PARM='PGM &ROOT./lib/native/zos/s390x/bbgzangl COLD=&COLD NAME=X
//             &NAME &PARMS'
//STDOUT    DD SYSOUT=*
//STDERR    DD SYSOUT=*
//* ================================================================= */
//* PROPRIETARY-STATEMENT:                                            */
//* Licensed Material - Property of IBM                               */
//*                                                                   */
//* (C) Copyright IBM Corp. 2011, 2012                                */
//* All Rights Reserved                                               */
//* US Government Users Restricted Rights - Use, duplication or       */
//* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.*/
//* ================================================================= */
```

## *Shell script, server.xml and server.env files*

In the lab exercise we had you run a shell script called `copy.sh`, which copied over to all four Liberty servers a simple[1] `server.xml` and the `server.env` file. That shell script contained:

```
#!/bin/sh
echo 'Copying files for JSRDISP'
cp /u/user1/jbatch/lab2/disp.xml   /shared/jsrhome/servers/JSRDISP/server.xml
cp /u/user1/jbatch/lab2/server.env /shared/jsrhome/servers/JSRDISP/server.env

echo 'Copying files for JSREXEC1'
cp /u/user1/jbatch/lab2/exec1.xml  /shared/jsrhome/servers/JSREXEC1/server.xml
cp /u/user1/jbatch/lab2/server.env /shared/jsrhome/servers/JSREXEC1/server.env

echo 'Copying files for JSREXEC2'
cp /u/user1/jbatch/lab2/exec2.xml  /shared/jsrhome/servers/JSREXEC2/server.xml
cp /u/user1/jbatch/lab2/server.env /shared/jsrhome/servers/JSREXEC2/server.env

echo 'Copying files for JSRMON'
cp /u/user1/jbatch/lab2/mon.xml    /shared/jsrhome/servers/JSRMON/server.xml
cp /u/user1/jbatch/lab2/server.env /shared/jsrhome/servers/JSRMON/server.env
echo 'All done!'
```

The purpose of that shell script was to spare the students from issuing eight copy commands.

The same `server.env` file was copied to each server's directory. That file contained:

```
JAVA_HOME=/shared/java/J8.0_64
```

The four `server.xml` files were unique to each server. Initially they were very simple, with the only difference being the HTTP port values:

1 The default server.xml had Java EE and some security elements that would work, but were more complex than needed.

### *JSRDISP*

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>servlet-3.1</feature>
    </featureManager>

    <httpEndpoint id="defaultHttpEndpoint"
        host="*"
        httpPort="10080"
        httpsPort="10443" />

</server>
```

### *JSREXEC1*

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>servlet-3.1</feature>
    </featureManager>

    <httpEndpoint id="defaultHttpEndpoint"
        host="*"
        httpPort="11080"
        httpsPort="11443" />

</server>
```

### *JSREXEC2*

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>servlet-3.1</feature>
    </featureManager>

    <httpEndpoint id="defaultHttpEndpoint"
        host="*"
        httpPort="12080"
        httpsPort="12443" />

</server>
```

### *JSRMON*

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>servlet-3.1</feature>
    </featureManager>

    <httpEndpoint id="defaultHttpEndpoint"
```

```
        host="*"
        httpPort="13080"
        httpsPort="13443" />

</server>
```

## *DDL generation*

To create the DDL a new `server.xml` was copied in to the JSRDISP server. It contained the following, with the *new* XML elements highlighted:

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>servlet-3.1</feature>
        <feature>batch-1.0</feature>
        <feature>localConnector-1.0</feature>
    </featureManager>

    <batchPersistence jobStoreRef="BatchDatabaseStore" />

    <databaseStore id="BatchDatabaseStore"
      createTables="false"
      dataSourceRef="batchDB" schema="JBATCH" tablePrefix="" />
    <jdbcDriver id="DB2T4" libraryRef="DB2T4LibRef" />

    <library id="DB2T4LibRef">
      <fileset dir="/shared/db21210/jdbc/classes/"
      includes="db2jcc4.jar db2jcc_license_cisuz.jar sqlj4.zip" />
    </library>

    <authData id="batchAlias" user="SYSADM1" password="SYSADM1" />

    <dataSource id="batchDB"
      containerAuthDataRef="batchAlias"
      type="javax.sql.XADataSource"
      jdbcDriverRef="DB2T4">
     <properties.db2.jcc
       serverName="wg31.washington.ibm.com"
       portNumber="2446"
       databaseName="DSN2LOC"
       driverType="4" />
    </dataSource>

    <httpEndpoint id="defaultHttpEndpoint"
        host="*"
        httpPort="10080"
        httpsPort="10443" />

</server>
```

We then had you run a JCL job that contained the DDL we had generated ahead of the worskhop. The JCL to create the JobRepository tables looked like this:

```
//DB2JOB  JOB 1,'SEL DB2',CLASS=A,REGION=0M,NOTIFY=?,
//    USER=SYSADM1,PASSWORD=SYSADM1
//*-------------------------------------------------------
//* JOB TO RUN DB2 STATEMENTS                           -
```

```
//*-------------------------------------------------------
//DB2EXEC EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB  DD  DSN=DSN1210.DB2.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSIN  DD *
 DSN SYSTEM(DSN2)
 RUN PROGRAM(DSNTEP2) PLAN(DSNTEP12) LIB('DSN2.RUNLIB.LOAD')
 END
//*-------------------------------------------------------
//* PLACE DB2 STATEMENTS FOLLOWING HERE                 -
//*-------------------------------------------------------
```

**Important!** Do ***not*** simply copy this DDL and run for your JobRepository database creation. Be sure to generate the DDL, as the DDL may change from what you see here.

Also, this job uses *default* tables spaces and storage groups, and your DB2 administrator will very likely prefer to organize the tables according to your local standards.

```
CREATE TABLE JBATCH.JOBINSTANCE
 (JOBINSTANCEID BIGINT GENERATED ALWAYS AS IDENTITY NOT NULL,
  AMCNAME VARCHAR(512), BATCHSTATUS INTEGER NOT NULL, CREATETIME
  TIMESTAMP NOT NULL, EXITSTATUS VARCHAR(512),
  INSTANCESTATE INTEGER NOT NULL, JOBNAME VARCHAR(256),
  JOBXMLNAME VARCHAR(128), JOBXML BLOB(64000),
  NUMEXECS INTEGER NOT NULL, RESTARTON VARCHAR(128),
  SUBMITTER VARCHAR(256), UPDATETIME TIMESTAMP,
  PRIMARY KEY (JOBINSTANCEID))  CCSID UNICODE;
CREATE TABLE JBATCH.STEPTHREADINSTANCE
 (PARTNUM INTEGER NOT NULL, STEPNAME VARCHAR(128) NOT NULL,
  THREADTYPE VARCHAR(31), CHECKPOINTDATA BLOB(64000),
  FK_JOBINSTANCEID BIGINT NOT NULL, FK_LATEST_STEPEXECID
  BIGINT NOT NULL, PARTITIONED SMALLINT DEFAULT 0 NOT NULL,
  PARTITIONPLANSIZE INTEGER, STARTCOUNT INTEGER,
  PRIMARY KEY (PARTNUM, STEPNAME, FK_JOBINSTANCEID))
  CCSID UNICODE;
CREATE INDEX JBATCH.STI_FKINSTANCEID_IX ON
  JBATCH.STEPTHREADINSTANCE (FK_JOBINSTANCEID);
CREATE INDEX JBATCH.STI_FKLATEST_SEI_IX ON
  JBATCH.STEPTHREADINSTANCE (FK_LATEST_STEPEXECID);
CREATE TABLE JBATCH.JOBEXECUTION
 (JOBEXECID BIGINT GENERATED ALWAYS AS IDENTITY NOT NULL,
  BATCHSTATUS INTEGER NOT NULL, CREATETIME TIMESTAMP NOT NULL,
  ENDTIME TIMESTAMP, EXECNUM INTEGER NOT NULL,
  EXITSTATUS VARCHAR(512), JOBPARAMETERS BLOB(64000),
  UPDATETIME TIMESTAMP, LOGPATH VARCHAR(512),
  RESTURL VARCHAR(512), SERVERID VARCHAR(256),
  STARTTIME TIMESTAMP, FK_JOBINSTANCEID BIGINT
  NOT NULL, PRIMARY KEY (JOBEXECID))  CCSID UNICODE;
CREATE INDEX JBATCH.JE_FKINSTANCEID_IX ON
  JBATCH.JOBEXECUTION (FK_JOBINSTANCEID);
CREATE TABLE JBATCH.STEPTHREADEXECUTION
 (STEPEXECID BIGINT GENERATED ALWAYS AS IDENTITY NOT NULL,
  THREADTYPE VARCHAR(31), BATCHSTATUS INTEGER NOT NULL,
  M_COMMIT BIGINT NOT NULL, ENDTIME TIMESTAMP, EXITSTATUS
  VARCHAR(512), M_FILTER BIGINT NOT NULL, INTERNALSTATUS
  INTEGER NOT NULL, PARTNUM INTEGER NOT NULL,
  USERDATA BLOB(64000), M_PROCESSSKIP BIGINT NOT NULL,
  M_READ BIGINT NOT NULL, M_READSKIP BIGINT NOT NULL,
```

```
  M_ROLLBACK BIGINT NOT NULL, STARTTIME TIMESTAMP,
  STEPNAME VARCHAR(128) NOT NULL, M_WRITE BIGINT NOT NULL,
  M_WRITESKIP BIGINT NOT NULL, FK_JOBEXECID BIGINT NOT NULL,
  FK_TOPLVL_STEPEXECID BIGINT, ISPARTITIONEDSTEP SMALLINT
  DEFAULT 0, PRIMARY KEY (STEPEXECID))  CCSID UNICODE;
CREATE INDEX JBATCH.STE_FKJOBEXECID_IX ON
  JBATCH.STEPTHREADEXECUTION (FK_JOBEXECID);
CREATE INDEX JBATCH.STE_FKTLSTEPEID_IX ON
  JBATCH.STEPTHREADEXECUTION (FK_TOPLVL_STEPEXECID);
CREATE TABLE JBATCH.JOBPARAMETER (NAME VARCHAR(255),
  VALUE VARCHAR(255), FK_JOBEXECID BIGINT)  CCSID UNICODE;
CREATE INDEX JBATCH.JP_FKJOBEXECID_IX ON
  JBATCH.JOBPARAMETER (FK_JOBEXECID);
ALTER TABLE JBATCH.STEPTHREADEXECUTION ADD CONSTRAINT
  STPTHRADEXECUTION0 UNIQUE (FK_JOBEXECID, STEPNAME, PARTNUM);
ALTER TABLE JBATCH.STEPTHREADINSTANCE ADD CONSTRAINT
  STPTHRFKLTSTSTPXCD FOREIGN KEY (FK_LATEST_STEPEXECID)
  REFERENCES JBATCH.STEPTHREADEXECUTION (STEPEXECID);
ALTER TABLE JBATCH.STEPTHREADINSTANCE ADD CONSTRAINT
  STPTHRDNFKJBNSTNCD FOREIGN KEY (FK_JOBINSTANCEID)
  REFERENCES JBATCH.JOBINSTANCE (JOBINSTANCEID);
ALTER TABLE JBATCH.JOBEXECUTION ADD CONSTRAINT
  JBXCTNFKJBNSTNCEID FOREIGN KEY (FK_JOBINSTANCEID)
  REFERENCES JBATCH.JOBINSTANCE (JOBINSTANCEID);
ALTER TABLE JBATCH.STEPTHREADEXECUTION ADD CONSTRAINT
  STPTHFKTPLVLSTPXCD FOREIGN KEY (FK_TOPLVL_STEPEXECID)
  REFERENCES JBATCH.STEPTHREADEXECUTION (STEPEXECID)
  ON DELETE NO ACTION;   <- for our lab system we had to manually add this
ALTER TABLE JBATCH.STEPTHREADEXECUTION ADD CONSTRAINT
  STPTHRDXCTNFKJBXCD FOREIGN KEY (FK_JOBEXECID)
  REFERENCES JBATCH.JOBEXECUTION (JOBEXECID);
ALTER TABLE JBATCH.JOBPARAMETER ADD CONSTRAINT
  JBPRMETERFKJBXECID FOREIGN KEY (FK_JOBEXECID)
  REFERENCES JBATCH.JOBEXECUTION (JOBEXECID);
/*
//*
```

## *Java Batch validation server.xml*

To validate the essential Java Batch operations we copied in another `server.xml` to the JSRDISP server. It looked like the following. Elements were added and ~~deleted~~.

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>servlet-3.1</feature>
        <feature>batch-1.0</feature>
        <feature>localConnector-1.0</feature>
        <feature>batchManagement-1.0</feature>
        <feature>appSecurity-2.0</feature>
    </featureManager>

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

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

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

    <batchPersistence jobStoreRef="BatchDatabaseStore" />

    <databaseStore id="BatchDatabaseStore"
      createTables="false"
      dataSourceRef="batchDB" schema="JBATCH" tablePrefix="" />
    <jdbcDriver id="DB2T4" libraryRef="DB2T4LibRef" />

    <library id="DB2T4LibRef">
      <fileset dir="/shared/db21210/jdbc/classes/"
      includes="db2jcc4.jar db2jcc_license_cisuz.jar sqlj4.zip" />
    </library>

    <authData id="batchAlias" user="SYSADM1" password="SYSADM1" />

    <dataSource id="batchDB"
      containerAuthDataRef="batchAlias"
      type="javax.sql.XADataSource"
      jdbcDriverRef="DB2T4">
     <properties.db2.jcc
       serverName="wg31.washington.ibm.com"
       portNumber="2446"
       databaseName="DSN2LOC"
       driverType="4" />
    </dataSource>

    <httpEndpoint id="defaultHttpEndpoint"
        host="*"
        httpPort="10080"
        httpsPort="10443" />

</server>
```

## *Commands*

The commands used in this lab were as follows. Some commands are longer than one line here; those were entered as one line into the UNIX shell environment.

```
TSO USER1

USER1.JBATCH.*

mkdir /shared/jsrhome

chown JSRSERV:JSRGRP /shared/jsrhome

chmod 750 /shared/jsrhome

su -s JSRSERV

whoami
```

```
cd /shared/zWebSphere/Liberty/V17001/bin

export JAVA_HOME=/shared/java/J8.0_64/

export WLP_USER_DIR=/shared/jsrhome

./server create JSRDISP

./server create JSREXEC1

./server create JSREXEC2

./server create JSRMON

ls -al /shared/jsrhome/servers

/u/user1/jbatch/lab2/copy.sh

SYS1.PROCLIB

=sdsf.da

PRE JSR*

/S JSRDISP

/S JSREXEC1

/S JSREXEC2

/S JSRMON

netstat | grep JSR

/P JSREXEC1

/P JSREXEC2

/P JSRMON

PRE DSN*

/-DSN2 START DB2

cp /u/user1/jbatch/lab2/disp_ddl.xml
/shared/jsrhome/servers/JSRDISP/server.xml

cd /shared/zWebSphere/Liberty/V17001/bin

export PATH=$PATH:/shared/java/J8.0_64/bin

./ddlGen generate JSRDISP
```

```
=3.17

/shared/jsrhome/servers/JSRDISP/ddl

USER1.JBATCH.JCL

cp /u/user1/jbatch/lab2/disp_app.xml
/shared/jsrhome/servers/JSRDISP/server.xml

/shared/jsrhome/servers/JSRDISP/logs

cp /u/user1/jbatch/apps/SleepyBatchletSample-1.0.war
/shared/jsrhome/servers/JSRDISP/dropins/SleepyBatchletSample-1.0.war

./batchManager submit --batchManager=localhost:10443 --user=Fred
--password=fredpwd --applicationName=SleepyBatchletSample-1.0
--jobXMLName=sleepy-batchlet.xml --trustSslCertificates --wait
```

# Lab 3 - JSR-352 Concepts

## *JCL jobs for BonusPayout sample application*

The following JCL job was used to create the `BONUSDB.ACCOUNT` table:

```
//BONUS   JOB 1,'SEL DB2',CLASS=A,REGION=0M,NOTIFY=?,
//    USER=SYSADM1,PASSWORD=SYSADM1
//*-------------------------------------------------------
//* JOB TO RUN DB2 STATEMENTS                            -
//*-------------------------------------------------------
//DB2EXEC EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB  DD  DSN=DSN1210.DB2.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSIN  DD *
 DSN SYSTEM(DSN2)
 RUN PROGRAM(DSNTEP2) PLAN(DSNTEP12) LIB('DSN2.RUNLIB.LOAD')
 END
//*-------------------------------------------------------
//* PLACE DB2 STATEMENTS FOLLOWING HERE                  -
//*-------------------------------------------------------
CREATE TABLE BONUSDB.ACCOUNT(
 ACCTNUM INTEGER NOT NULL,
 BALANCE INTEGER NOT NULL,
 INSTANCEID BIGINT NOT NULL,
 ACCTCODE VARCHAR(30),
 CONSTRAINT ACCOUNT_PK PRIMARY KEY (ACCTNUM,INSTANCEID));

CREATE UNIQUE INDEX ACCT_IDX ON BONUSDB.ACCOUNT(ACCTNUM,INSTANCEID);

COMMIT;
/*
//*
```

The following was used to select from that table to show what the Java Batch application produces in the table:

```
//SELECT  JOB 1,'SEL DB2',CLASS=A,REGION=0M,NOTIFY=?,
//    USER=SYSADM1,PASSWORD=SYSADM1
//*-------------------------------------------------------
//* JOB TO RUN DB2 STATEMENTS                            -
//*-------------------------------------------------------
//DB2EXEC EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB  DD  DSN=DSN1210.DB2.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSIN  DD *
 DSN SYSTEM(DSN2)
 RUN PROGRAM(DSNTEP2) PLAN(DSNTEP12) LIB('DSN2.RUNLIB.LOAD')
 END
//*-------------------------------------------------------
//* PLACE DB2 STATEMENTS FOLLOWING HERE                  -
//*-------------------------------------------------------
SELECT * FROM BONUSDB.ACCOUNT;
/*
//*
```

## *Configuration server.xml in support of BonusPayout sample*

Another `server.xml` was copied into the JSRDISP server to support the BonusPayout sample application. This added another JDBC Type 4 data source. The key was the JNDI for the data source -- `jdbc/bonus` -- which was used by the application to do the lookup:

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>servlet-3.1</feature>
        <feature>batch-1.0</feature>
        <feature>batchManagement-1.0</feature>
        <feature>appSecurity-2.0</feature>
    </featureManager>

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

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

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

    <batchPersistence jobStoreRef="BatchDatabaseStore" />

    <databaseStore id="BatchDatabaseStore"
      createTables="false"
      dataSourceRef="batchDB" schema="JBATCH" tablePrefix="" />
    <jdbcDriver id="DB2T4" libraryRef="DB2T4LibRef" />

    <library id="DB2T4LibRef">
      <fileset dir="/shared/db21210/jdbc/classes/"
      includes="db2jcc4.jar db2jcc_license_cisuz.jar sqlj4.zip" />
    </library>

    <authData id="batchAlias" user="SYSADM1" password="SYSADM1" />

    <dataSource id="batchDB"
      containerAuthDataRef="batchAlias"
      type="javax.sql.XADataSource"
      jdbcDriverRef="DB2T4">
     <properties.db2.jcc
       serverName="wg31.washington.ibm.com"
       portNumber="2446"
       databaseName="DSN2LOC"
       driverType="4" />
    </dataSource>

    <dataSource id="bonusDB" jndiName="jdbc/bonus"
     type="javax.sql.XADataSource"
     jdbcDriverRef="DB2T4">
```

```
     <properties.db2.jcc
       serverName="wg31.washington.ibm.com"
       portNumber="2446"
       databaseName="DSN2LOC"
       user="SYSADM1"
       password="SYSADM1"
       driverType="4" />
    </dataSource>

    <httpEndpoint id="defaultHttpEndpoint"
        host="*"
        httpPort="10080"
        httpsPort="10443" />

</server>
```

## *Commands*

The commands used in this lab were as follows. Some commands are longer than one line here; those were entered as one line into the UNIX shell environment.

```
cp /u/user1/jbatch/apps/JSR352Concepts2ProjectWAR.war
/shared/jsrhome/servers/JSRDISP/dropins/JSR352Concepts2ProjectWAR.war

cd /shared/zWebSphere/Liberty/V17001/bin

export JAVA_HOME=/shared/java/J8.0_64/

./batchManager submit --batchManager=localhost:10443 --user=Fred
--password=fredpwd --applicationName=JSR352Concepts2ProjectWAR
--jobXMLName=SplitExample.xml --pollingInterval_s=5
--trustSslCertificates --wait

/shared/jsrhome/servers/JSRDISP/logs

cp /u/user1/jbatch/apps/JSR352ConceptsProjectWAR.war
/shared/jsrhome/servers/JSRDISP/dropins/JSR352ConceptsProjectWAR.war

./batchManager submit --batchManager=localhost:10443 --user=Fred
--password=fredpwd --applicationName=JSR352ConceptsProjectWAR
--jobXMLName=JSR352ConceptsJob.xml --trustSslCertificates
--pollingInterval_s=5 --wait --jobParameter=points=50

./batchManager submit --batchManager=localhost:10443 --user=Fred
--password=fredpwd --applicationName=JSR352ConceptsProjectWAR
--jobXMLName=JSR352ConceptsJob.xml --trustSslCertificates
--pollingInterval_s=5 --wait --jobParameter=points=5000

./batchManager submit --batchManager=localhost:10443 --user=Fred
--password=fredpwd --applicationName=JSR352ConceptsProjectWAR
--jobXMLName=JSR352ConceptsJob.xml --trustSslCertificates
--pollingInterval_s=5 --wait --jobParameter=points=xyz

cp /u/user1/jbatch/apps/JSR352Concepts3WAR.war
/shared/jsrhome/servers/JSRDISP/dropins/JSR352Concepts3WAR.war
```

```
cp /u/user1/jbatch/lab3/jsrdisp.xml
/shared/jsrhome/servers/JSRDISP/server.xml

=3.4

USER1.JBATCH.JCL

./batchManager submit --batchManager=localhost:10443 --user=Fred
--password=fredpwd --applicationName=JSR352Concepts3WAR
--jobXMLName=LoggingSimpleBonusPayout.xml
--jobParameter=dsJNDI=jdbc/bonus
--jobParameter=tableName=BONUSDB.ACCOUNT --jobParameter=logging=true
--jobParameter=numRecords=10 --jobParameter=chunkSize=2
--pollingInterval_s=5 --trustSslCertificates --wait

/tmp/

/shared/jsrhome/servers/JSRDISP/logs

=3.4

USER1.JBATCH.JCL

=sdsf.st

PRE SEL*

./batchManager submit --batchManager=localhost:10443 --user=Fred
--password=fredpwd --applicationName=JSR352Concepts3WAR
--jobXMLName=LoggingSimpleBonusPayout.xml
--jobParameter=dsJNDI=jdbc/bonus
--jobParameter=tableName=BONUSDB.ACCOUNT --jobParameter=logging=false
--jobParameter=numRecords=1000 --jobParameter=chunkSize=100
--pollingInterval_s=5 --trustSslCertificates --wait

rm /shared/jsrhome/servers/JSRDISP/dropins/JSR352*

ls /shared/jsrhome/servers/JSRDISP/dropins
```

# Lab 4 - Job Submission and Control

## *JCL BPXBATCH and shell script to invoke batchManager*

The following is the JCL with BPXBATCH that calls the `subjob.sh` shell script to invoke batchManager and submit a job:

```
//BMGRJCL  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 +
 /u/user1/jbatch/lab4/subjob.sh
/*
//*
```

The shell script that's called looked like this:

```
#!/bin/sh
# ----------------------------------------------------------------------------
# NOTICE - THIS IS SAMPLE CODE
#
# The following is sample code created by IBM Corporation.  This sample code
# is not part of any standard IBM product.  This shell script is provided
# "AS IS", without warranty of any kind.  IBM shall not be liable for any
# damages arising out of your use, or any other party's use, of this sample.
# If you do not agree with these terms, do not use this sample shell script.
# ----------------------------------------------------------------------------
echo 'Setting Variables'
JAVA="/shared/java/J8.0_64/"
LOCATION="/shared/zWebSphere/Liberty/V17001/bin/batchManager"
ACTION="submit"
SERVER="--batchManager=wg31.washington.ibm.com:10443"
USER="--user=Fred"
PW="--password=fredpwd"
APP="--applicationName=SleepyBatchletSample-1.0"
JOBXML="--jobXMLName=sleepy-batchlet.xml"
PARM1="--jobParameter=sleep.time.seconds=5"
PARM2="--getJobLog"
PARM3="--trustSslCertificates"
PARM4="--wait"
OUTLOC="/u/user1/jbatch/lab4"

echo 'Set Java Home'
export JAVA_HOME=$JAVA

echo 'Submitting Command'
$LOCATION $ACTION $SERVER $USER $PW $APP $JOBXML $PARM1 $PARM2 $PARM3 $PARM4 >
$OUTLOC/out.txt 2> $OUTLOC/err.txt

rc=$?

if [ rc -eq 35 ]; then
  exitcode=0
  echo 'batchManager exit code=' $rc 'Status=Good: submitted and completed.'
  echo 'Will set shell script exit code to' $exitcode
```

```
  echo 'If BPXBATCH, then JES return code should be RC=0000'
  exit $exitcode

elif [ rc -eq 33 ]; then
  exitcode=4
  echo 'batchManager exit code=' $rc 'Status=Issue: someone stopped the job.'
  echo 'Will set shell script exit code to' $exitcode
  echo 'If BPXBATCH, then JES return code should be RC=1024'
  exit $exitcode

elif [ rc -eq 20 ] || [ rc -eq 21 ] || [ rc -eq 22 ]; then
  exitcode=8
  echo 'batchManager exit code=' $rc 'Status=Issue: argument problem --
required, unrecognized, invalid.'
  echo 'Will set shell script exit code to' $exitcode
  echo 'If BPXBATCH, then JES return code should be RC=2048'
  exit $exitcode

elif [ rc -eq 34 ] || [ rc -eq 36 ]; then
  exitcode=12
  echo 'batchManager exit code=' $rc 'Status=Problem: did not complete or was
abandoned.'
  echo 'Will set shell script exit code to' $exitcode
  echo 'If BPXBATCH, then JES return code should be RC=3072'
  exit $exitcode

else
  exitcode=13
  echo 'batchManager exit code=' $rc 'Status=Problem: some other problem
occurred.  See Knowledge Center.'
  echo 'Will set shell script exit code to' $exitcode
  echo 'If BPXBATCH, then JES return code should be RC=3328'
  exit $exitcode
fi

echo 'Done!'
```

### *Preparing to use WOLA*

The batchManagerZos command utility uses WOLA to access the Liberty z/OS server, and WOLA is an authorized service. That implies the Angel process and `SERVER` profiles to permit access. Here is the RACF job we had students run to set up the `SERVER` and `CBIND` profiles.

```
//LAB4    JOB (????,????),'LAB4 JOB',MSGCLASS=O,CLASS=A,
//         NOTIFY=????????,REGION=4000K TYPRUN=HOLD
//*-------------------------------------------------------------*
//RACF EXEC PGM=IKJEFT01,REGION=0M
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
RDEFINE SERVER BBG.ANGEL UACC(NONE) OWNER(SYS1)
PERMIT  BBG.ANGEL CLASS(SERVER) ACCESS(READ) ID(JSRSERV)

RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM UACC(NONE) OWNER(SYS1)
PERMIT  BBG.AUTHMOD.BBGZSAFM -
        CLASS(SERVER) ACCESS(READ) ID(JSRSERV)

RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.SAFCRED UACC(NONE)
PERMIT  BBG.AUTHMOD.BBGZSAFM.SAFCRED -
        CLASS(SERVER) ACCESS(READ) ID(JSRSERV)
```

```
RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.ZOSWLM UACC(NONE)
PERMIT  BBG.AUTHMOD.BBGZSAFM.ZOSWLM  -
        CLASS(SERVER) ACCESS(READ) ID(JSRSERV)

RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.TXRRS UACC(NONE)
PERMIT  BBG.AUTHMOD.BBGZSAFM.TXRRS   -
        CLASS(SERVER) ACCESS(READ) ID(JSRSERV)

RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.ZOSDUMP UACC(NONE)
PERMIT  BBG.AUTHMOD.BBGZSAFM.ZOSDUMP  -
        CLASS(SERVER) ACCESS(READ) ID(JSRSERV)

RDEFINE SERVER BBG.SECPFX.BBGZDFLT UACC(NONE)
PERMIT  BBG.SECPFX.BBGZDFLT  -
        CLASS(SERVER) ACCESS(READ) ID(JSRSERV)

RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.WOLA UACC(NONE) OWNER(SYS1)
PERMIT  BBG.AUTHMOD.BBGZSAFM.WOLA -
        CLASS(SERVER) ACCESS(READ) ID(JSRSERV)

RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.LOCALCOM UACC(NONE) OWNER(SYS1)
PERMIT  BBG.AUTHMOD.BBGZSAFM.LOCALCOM -
        CLASS(SERVER) ACCESS(READ) ID(JSRSERV)

RDEFINE SERVER BBG.AUTHMOD.BBGZSCFM UACC(NONE) OWNER(SYS1)
PERMIT  BBG.AUTHMOD.BBGZSCFM -
        CLASS(SERVER) ACCESS(READ) ID(JSRSERV)

RDEFINE SERVER BBG.AUTHMOD.BBGZSCFM.WOLA UACC(NONE) OWNER(SYS1)
PERMIT  BBG.AUTHMOD.BBGZSCFM.WOLA -
        CLASS(SERVER) ACCESS(READ) ID(JSRSERV)

SETROPTS RACLIST(SERVER) REFRESH

RDEFINE CBIND BBG.WOLA.LIBERTY.BATCH.MANAGER UACC(NONE) OWNER(SYS1)
PERMIT BBG.WOLA.LIBERTY.BATCH.MANAGER CLASS(CBIND) -
  ACCESS(READ) ID(USER1)
SETROPTS RACLIST(CBIND) REFRESH
/*
//*
```

Using WOLA requires a few additions to the `server.xml` file. Those are shown here:

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>servlet-3.1</feature>
        <feature>batch-1.0</feature>
        <feature>batchManagement-1.0</feature>
        <feature>appSecurity-2.0</feature>
        <feature>zosLocalAdapters-1.0</feature>
    </featureManager>

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

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

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

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

    <batchPersistence jobStoreRef="BatchDatabaseStore" />

    <databaseStore id="BatchDatabaseStore"
      createTables="false"
      dataSourceRef="batchDB" schema="JBATCH" tablePrefix="" />
    <jdbcDriver id="DB2T4" libraryRef="DB2T4LibRef" />

    <library id="DB2T4LibRef">
      <fileset dir="/shared/db21210/jdbc/classes/"
      includes="db2jcc4.jar db2jcc_license_cisuz.jar sqlj4.zip" />
    </library>

    <authData id="batchAlias" user="SYSADM1" password="SYSADM1" />

    <dataSource id="batchDB"
      containerAuthDataRef="batchAlias"
      type="javax.sql.XADataSource"
      jdbcDriverRef="DB2T4">
     <properties.db2.jcc
       serverName="wg31.washington.ibm.com"
       portNumber="2446"
       databaseName="DSN2LOC"
       driverType="4" />
    </dataSource>

    <httpEndpoint id="defaultHttpEndpoint"
        host="*"
        httpPort="10080"
        httpsPort="10443" />

</server>
```

### *JCL BPXBATCH and shell script to invoke batchManagerZos*

The JCL to call BPXBATCH and a shell script to invoke batchManagerZos looked like this:

```
//BMGRZJCL  JOB (0),'BMGRZ+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 +
```

```
 /u/user1/jbatch/lab4/subzjob.sh
/*
//*
```

The `subzjob.sh` shell script looked like this:

```
#!/bin/sh
# ------------------------------------------------------------------------
# NOTICE - THIS IS SAMPLE CODE
#
# The following is sample code created by IBM Corporation.  This sample code
# is not part of any standard IBM product.  This shell script is provided
# "AS IS", without warranty of any kind.  IBM shall not be liable for any
# damages arising out of your use, or any other party's use, of this sample.
# If you do not agree with these terms, do not use this sample shell script.
# ------------------------------------------------------------------------
echo 'Setting Variables'
LOCATION="/shared/zWebSphere/Liberty/V17001/lib/native/zos/s390x/batchManagerZo
s"
ACTION="submit"
SERVER="--batchManager=LIBERTY+BATCH+MANAGER"
APP="--applicationName=SleepyBatchletSample-1.0"
JOBXML="--jobXMLName=sleepy-batchlet.xml"
PARM1="--jobParameter=sleep.time.seconds=5"
PARM2="--wait"
OUTLOC="/u/user1/jbatch/lab4"

echo 'Submitting Command'
$LOCATION $ACTION $SERVER $APP $JOBXML $PARM1 $PARM2 > $OUTLOC/out.txt 2>
$OUTLOC/err.txt

rc=$?

if [ rc -eq 35 ]; then
  exitcode=0
  echo 'batchManagerZos exit code=' $rc 'Status=Good: submitted and completed.'
  echo 'Will set shell script exit code to' $exitcode
  echo 'If BPXBATCH, then JES return code should be RC=0000'
  exit $exitcode

elif [ rc -eq 33 ]; then
  exitcode=4
  echo 'batchManagerZos exit code=' $rc 'Status=Issue: someone stopped the
job.'
  echo 'Will set shell script exit code to' $exitcode
  echo 'If BPXBATCH, then JES return code should be RC=1024'
  exit $exitcode

elif [ rc -eq 20 ] || [ rc -eq 21 ] || [ rc -eq 22 ]; then
  exitcode=8
  echo 'batchManagerZos exit code=' $rc 'Status=Issue: argument problem --
required, unrecognized, invalid.'
  echo 'Will set shell script exit code to' $exitcode
  echo 'If BPXBATCH, then JES return code should be RC=2048'
  exit $exitcode

elif [ rc -eq 34 ] || [ rc -eq 36 ]; then
  exitcode=12
  echo 'batchManagerZos exit code=' $rc 'Status=Problem: did not complete or
```

```
was abandoned.'
  echo 'Will set shell script exit code to' $exitcode
  echo 'If BPXBATCH, then JES return code should be RC=3072'
  exit $exitcode

else
  exitcode=13
  echo 'batchManagerZos exit code=' $rc 'Status=Problem: some other problem
occurred.  See Knowledge Center.'
  echo 'Will set shell script exit code to' $exitcode
  echo 'If BPXBATCH, then JES return code should be RC=3328'
  exit $exitcode
fi

echo 'Done!'
```

### *Batch events*

To enable batch events in the JSRDISP server, the following `server.xml` was copied in. The added elements are highlighted:

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>servlet-3.1</feature>
        <feature>batch-1.0</feature>
        <feature>batchManagement-1.0</feature>
        <feature>appSecurity-2.0</feature>
        <feature>zosLocalAdapters-1.0</feature>
        <feature>wmqJmsClient-2.0</feature>
    </featureManager>

    <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>

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

    <wmqJmsClient startupRetryCount="999"
      startupRetryInterval="1000ms"
      reconnectionRetryCount="10"
      reconnectionRetryInterval="5m">
    </wmqJmsClient>

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

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

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

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

    <batchPersistence jobStoreRef="BatchDatabaseStore" />

    <databaseStore id="BatchDatabaseStore"
      createTables="false"
      dataSourceRef="batchDB" schema="JBATCH" tablePrefix="" />
    <jdbcDriver id="DB2T4" libraryRef="DB2T4LibRef" />

    <library id="DB2T4LibRef">
      <fileset dir="/shared/db21210/jdbc/classes/"
      includes="db2jcc4.jar db2jcc_license_cisuz.jar sqlj4.zip" />
    </library>

    <authData id="batchAlias" user="SYSADM1" password="SYSADM1" />

    <dataSource id="batchDB"
      containerAuthDataRef="batchAlias"
      type="javax.sql.XADataSource"
      jdbcDriverRef="DB2T4">
     <properties.db2.jcc
       serverName="wg31.washington.ibm.com"
       portNumber="2446"
       databaseName="DSN2LOC"
       driverType="4" />
    </dataSource>

    <httpEndpoint id="defaultHttpEndpoint"
        host="*"
        httpPort="10080"
        httpsPort="10443" />

</server>
```

The JCL to call BPXBATCH to invoke batchManagerZos and take advantage of batch events looked like this:

```
//BMGREJCL JOB (0),'BMGRZ+E+JCL',CLASS=A,REGION=0M,
// MSGCLASS=H,NOTIFY=USER1
//SUBMIT   EXEC  PGM=IKJEFT01
//STEPLIB  DD  DSN=MQ901.SCSQAUTH,DISP=SHR
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//STDOUT   DD  SYSOUT=*
//STDERR   DD  SYSOUT=*
//SYSTSIN  DD  *
 BPXBATCH  SH +
```

```
 /u/user1/jbatch/lab4/subzjob_events.sh
/*
//*
```

The `subzjob_events.sh` shell script looked like this:

```
#!/bin/sh
# ------------------------------------------------------------------------
# NOTICE - THIS IS SAMPLE CODE
#
# The following is sample code created by IBM Corporation.  This sample code
# is not part of any standard IBM product.  This shell script is provided
# "AS IS", without warranty of any kind.  IBM shall not be liable for any
# damages arising out of your use, or any other party's use, of this sample.
# If you do not agree with these terms, do not use this sample shell script.
# ------------------------------------------------------------------------
echo 'Setting Variables'
LOCATION="/shared/zWebSphere/Liberty/V17001/lib/native/zos/s390x/batchManagerZo
s"
ACTION="submit"
SERVER="--batchManager=LIBERTY+BATCH+MANAGER"
APP="--applicationName=SleepyBatchletSample-1.0"
JOBXML="--jobXMLName=sleepy-batchlet.xml"
PARM1="--jobParameter=sleep.time.seconds=5"
PARM2="--getJobLog"
PARM3="--wait"
PARM4="--queueManagerName=MQS1"

echo 'Submitting Command'
$LOCATION $ACTION $SERVER $APP $JOBXML $PARM1 $PARM2 $PARM3 $PARM4

rc=$?

if [ rc -eq 35 ]; then
  exitcode=0
  echo 'batchManagerZos exit code=' $rc 'Status=Good: submitted and completed.'
  echo 'Will set shell script exit code to' $exitcode
  echo 'If BPXBATCH, then JES return code should be RC=0000'
  exit $exitcode

elif [ rc -eq 33 ]; then
  exitcode=4
  echo 'batchManagerZos exit code=' $rc 'Status=Issue: someone stopped the
job.'
  echo 'Will set shell script exit code to' $exitcode
  echo 'If BPXBATCH, then JES return code should be RC=1024'
  exit $exitcode

elif [ rc -eq 20 ] || [ rc -eq 21 ] || [ rc -eq 22 ]; then
  exitcode=8
  echo 'batchManagerZos exit code=' $rc 'Status=Issue: argument problem --
required, unrecognized, invalid.'
  echo 'Will set shell script exit code to' $exitcode
  echo 'If BPXBATCH, then JES return code should be RC=2048'
  exit $exitcode

elif [ rc -eq 34 ] || [ rc -eq 36 ]; then
  exitcode=12
  echo 'batchManagerZos exit code=' $rc 'Status=Problem: did not complete or
```

```
was abandoned.'
  echo 'Will set shell script exit code to' $exitcode
  echo 'If BPXBATCH, then JES return code should be RC=3072'
  exit $exitcode

else
  exitcode=13
  echo 'batchManagerZos exit code=' $rc 'Status=Problem: some other problem
occurred.  See Knowledge Center.'
  echo 'Will set shell script exit code to' $exitcode
  echo 'If BPXBATCH, then JES return code should be RC=3328'
  exit $exitcode
fi

echo 'Done!'
```

The `server.xml` that was copied into the JSRMON server to act as a monitor of the batch events topic was this:

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>mdb-3.2</feature>
        <feature>jndi-1.0</feature>
        <feature>jca-1.7</feature>
        <feature>servlet-3.1</feature>
        <feature>jsonp-1.0</feature>
        <feature>wmqJmsClient-2.0</feature>
    </featureManager>

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

    <wmqJmsClient startupRetryCount="999"
      startupRetryInterval="1000ms"
      reconnectionRetryCount="10"
      reconnectionRetryInterval="5m">
    </wmqJmsClient>

    <jmsTopic id="JobLogEventTopic"
      jndiName="jms/batch/batchJobTopic">
      <properties.wmqJms
      baseTopicName="batch/jobs/execution/jobLogPart" />
    </jmsTopic>

    <jmsActivationSpec id="JobLogEventsDirCreator-1.0/JobLogEventsSubscriber">
      <properties.wmqJms
       destinationRef="JobLogEventTopic"
       destinationType="javax.jms.Topic"
       transportType="CLIENT"
       channel="SYSTEM.DEF.SVRCONN"
       queueManager="MQS1"
       hostName="wg31.washington.ibm.com"
       port="1414" />
    </jmsActivationSpec>

    <httpEndpoint id="defaultHttpEndpoint"
```

```
        host="*"
        httpPort="13080"
        httpsPort="13443" />

</server>
```

The key thing to note there was this server had *no* IBM Java Batch definitions at all. The point we made in the workshop was this illustrates how any process capable of subscribing to a topic can monitor batch events.

### *AdminCenter*

Finally, we copied in the following XML to enable the AdminCenter:

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>servlet-3.1</feature>
        <feature>batch-1.0</feature>
        <feature>batchManagement-1.0</feature>
        <feature>appSecurity-2.0</feature>
        <feature>zosLocalAdapters-1.0</feature>
        <feature>wmqJmsClient-2.0</feature>
        <feature>adminCenter-1.0</feature>
    </featureManager>

    <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>

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

    <wmqJmsClient startupRetryCount="999"
      startupRetryInterval="1000ms"
      reconnectionRetryCount="10"
      reconnectionRetryInterval="5m">
    </wmqJmsClient>

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

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

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

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

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

    <batchPersistence jobStoreRef="BatchDatabaseStore" />

    <databaseStore id="BatchDatabaseStore"
      createTables="false"
      dataSourceRef="batchDB" schema="JBATCH" tablePrefix="" />
    <jdbcDriver id="DB2T4" libraryRef="DB2T4LibRef" />

    <library id="DB2T4LibRef">
      <fileset dir="/shared/db21210/jdbc/classes/"
      includes="db2jcc4.jar db2jcc_license_cisuz.jar sqlj4.zip" />
    </library>

    <authData id="batchAlias" user="SYSADM1" password="SYSADM1" />

    <dataSource id="batchDB"
      containerAuthDataRef="batchAlias"
      type="javax.sql.XADataSource"
      jdbcDriverRef="DB2T4">
     <properties.db2.jcc
       serverName="wg31.washington.ibm.com"
       portNumber="2446"
       databaseName="DSN2LOC"
       driverType="4" />
    </dataSource>

    <httpEndpoint id="defaultHttpEndpoint"
        host="*"
        httpPort="10080"
        httpsPort="10443" />

</server>
```

## *Commands*

The commands used in this lab were as follows. Some commands are longer than one line here; those were entered as one line into the UNIX shell environment.

```
whoami

su -s JSRSERV

pwd

cd /shared/zWebSphere/Liberty/V17001/bin

echo $WLP_USER_DIR
```

```
export WLP_USER_DIR=/shared/jsrhome

echo $JAVA_HOME

export JAVA_HOME=/shared/java/J8.0_64/

./batchManager submit --batchManager=localhost:10443 --user=Fred
--password=fredpwd --applicationName=SleepyBatchletSample-1.0
--jobXMLName=sleepy-batchlet.xml --trustSslCertificates --wait

./batchManager submit --batchManager=localhost:10443 --user=Fred
--password=fredpwd --applicationName=SleepyBatchletSample-1.0
--jobXMLName=sleepy-batchlet.xml --jobParameter=sleep.time.seconds=5
--trustSslCertificates --wait

/shared/jsrhome/servers/JSRDISP/logs

./batchManager submit --batchManager=localhost:10443 --user=Fred
--password=fredpwd --applicationName=SleepyBatchletSample-1.0
--jobXMLName=sleepy-batchlet.xml --jobParameter=sleep.time.seconds=5
--getJobLog --trustSslCertificates --wait

cd /u/user1/jbatch/lab4

cat subjob.sh

./subjob.sh

cat out.txt

cat err.txt

USER1.JBATCH.JCL

pre bmgr*

=sdsf.da

pre JSR*

/P JSRDISP

USER1.JBATCH.JCL

cp /u/user1/jbatch/lab4/wola.xml
/shared/jsrhome/servers/JSRDISP/server.xml

=sdsf.da

/S JSRZANGL

/S JSRDISP
```

```
/shared/jsrhome/servers/JSRDISP/logs

whoami

cd /shared/zWebSphere/Liberty/V17001/lib/native/zos/s390x

./batchManagerZos ping --batchManager=LIBERTY+BATCH+MANAGER

./batchManagerZos submit --batchManager=LIBERTY+BATCH+MANAGER
--applicationName=SleepyBatchletSample-1.0 --jobXMLName=sleepy-
batchlet.xml --wait

=sdsf.st

PRE BMGR*

=sdsf.da

PRE MQS1*

/-MQS1 START QMGR

cp /shared/mqm/V9R0M1/java/lib/jca/wmq.jmsra.rar
/shared/jsrhome/servers/JSRDISP/wmq.jmsra.rar

export STEPLIB='MQ901.SCSQAUTH'

cp /u/user1/jbatch/lab4/events.xml
/shared/jsrhome/servers/JSRDISP/server.xml

cd /shared/zWebSphere/Liberty/V17001/lib/native/zos/s390x

./batchManagerZos submit --batchManager=LIBERTY+BATCH+MANAGER
--applicationName=SleepyBatchletSample-1.0 --jobXMLName=sleepy-
batchlet.xml --wait --queueManagerName=MQS1

./batchManagerZos submit --batchManager=LIBERTY+BATCH+MANAGER
--applicationName=SleepyBatchletSample-1.0 --jobXMLName=sleepy-
batchlet.xml --getJobLog --wait --queueManagerName=MQS1

cp /u/user1/jbatch/apps/JobLogEventsDirCreator-1.0.war
/shared/jsrhome/servers/JSRMON/dropins/JobLogEventsDirCreator-1.0.war

cp /shared/mqm/V9R0M1/java/lib/jca/wmq.jmsra.rar
/shared/jsrhome/servers/JSRMON/wmq.jmsra.rar

cp /u/user1/jbatch/lab4/monitor.xml
/shared/jsrhome/servers/JSRMON/server.xml

PRE JSR*

/S JSRMON
```

```
/shared/jsrhome/servers/JSRMON/JobLogEvents

/P JSRMON

cp /u/user1/jbatch/lab4/adminCenter.xml
/shared/jsrhome/servers/JSRDISP/server.xml

http://wg31.washington.ibm.com:10080/adminCenter

Fred

fredpwd
```

# Lab 5 - Multi-JVM

## *JCL Jobs (RACF and other)*

No JCL jobs are run in this lab.

## *Configuration server.xml files*

Updates to enable the **JSRDISP** server to be a "dispatcher" server:

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>servlet-3.1</feature>
        <feature>batch-1.0</feature>
        <feature>batchManagement-1.0</feature>
        <feature>appSecurity-2.0</feature>
        <feature>zosLocalAdapters-1.0</feature>
        <feature>wmqJmsClient-2.0</feature>
        <feature>adminCenter-1.0</feature>
    </featureManager>

    <batchJmsDispatcher
      connectionFactoryRef="batchConnectionFactory"
      queueRef="batchJobSubmissionQueue" />

    <jmsQueue id="batchJobSubmissionQueue"
      jndiName="jms/batch/jobSubmissionQueue">
      <properties.wmqJms baseQueueName="JAVA.BATCH.QUEUE"
        priority="QDEF"
        baseQueueManagerName="MQS1">
      </properties.wmqJms>
    </jmsQueue>

    <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>

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

    <wmqJmsClient startupRetryCount="999"
      startupRetryInterval="1000ms"
      reconnectionRetryCount="10"
      reconnectionRetryInterval="5m">
    </wmqJmsClient>

    <zosLocalAdapters wolaGroup="LIBERTY"
        wolaName2="BATCH"
```

```
        wolaName3="MANAGER"/>

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

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

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

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

    <batchPersistence jobStoreRef="BatchDatabaseStore" />

    <databaseStore id="BatchDatabaseStore"
      createTables="false"
      dataSourceRef="batchDB" schema="JBATCH" tablePrefix="" />
    <jdbcDriver id="DB2T4" libraryRef="DB2T4LibRef" />

    <library id="DB2T4LibRef">
      <fileset dir="/shared/db21210/jdbc/classes/"
      includes="db2jcc4.jar db2jcc_license_cisuz.jar sqlj4.zip" />
    </library>

    <authData id="batchAlias" user="SYSADM1" password="SYSADM1" />

    <dataSource id="batchDB"
      containerAuthDataRef="batchAlias"
      type="javax.sql.XADataSource"
      jdbcDriverRef="DB2T4">
     <properties.db2.jcc
       serverName="wg31.washington.ibm.com"
       portNumber="2446"
       databaseName="DSN2LOC"
       driverType="4" />
    </dataSource>

    <httpEndpoint id="defaultHttpEndpoint"
        host="*"
        httpPort="10080"
        httpsPort="10443" />

</server>
```

Key updates to enable the **JSREXEC1** server to be an executor. The copy of the new `server.xml` file brought in all the JMS definitions, not just what's highlighted here. What's highlighted here are the key things associated with being an executor server and the queue to listen on.

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
```

```
<!-- Enable features -->
<featureManager>
    <feature>servlet-3.1</feature>
    <feature>batch-1.0</feature>
    <feature>batchManagement-1.0</feature>
    <feature>appSecurity-2.0</feature>
    <feature>wmqJmsClient-2.0</feature>
</featureManager>

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

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

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

<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>

<batchPersistence jobStoreRef="BatchDatabaseStore" />

<databaseStore id="BatchDatabaseStore"
 dataSourceRef="batchDB" schema="JBATCH" tablePrefix="" />

<jdbcDriver id="DB2T4" libraryRef="DB2T4LibRef" />
 <library id="DB2T4LibRef">
  <fileset dir="/shared/db21210/jdbc/classes/"
  includes="db2jcc4.jar db2jcc_license_cisuz.jar sqlj4.zip" />
 </library>

<authData id="batchAlias" user="SYSADM1" password="SYSADM1" />

<dataSource id="batchDB"
 containerAuthDataRef="batchAlias"
 type="javax.sql.XADataSource"
 jdbcDriverRef="DB2T4">
 <properties.db2.jcc
  serverName="wg31.washington.ibm.com"
  portNumber="2446"
  databaseName="DSN2LOC"
  driverType="4" />
</dataSource>
```

```
    <batchJmsExecutor activationSpecRef="batchActivationSpec1"
      queueRef="batchJobSubmissionQueue"/>

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

    <wmqJmsClient startupRetryCount="999"
      startupRetryInterval="1000ms"
      reconnectionRetryCount="10"
      reconnectionRetryInterval="5m">
    </wmqJmsClient>

    <jmsActivationSpec id="batchActivationSpec1" >
      <properties.wmqJms
        destinationRef="batchJobSubmissionQueue"
   messageSelector="com_ibm_ws_batch_applicationName = 'SleepyBatchletSample-1.0'"
        maxPoolDepth="1"
        transportType="CLIENT"
        channel="SYSTEM.DEF.SVRCONN"
        destinationType="javax.jms.Queue"
        queueManager="MQS1"
        hostName="wg31.washington.ibm.com"
        port="1414">
      </properties.wmqJms>
    </jmsActivationSpec>

    <jmsQueue id="batchJobSubmissionQueue"
      jndiName="jms/batch/jobSubmissionQueue">
      <properties.wmqJms baseQueueName="JAVA.BATCH.QUEUE"
         baseQueueManagerName="MQS1">
      </properties.wmqJms>
    </jmsQueue>

    <httpEndpoint id="defaultHttpEndpoint"
                  host="*"
                  httpPort="11080"
                  httpsPort="11443" />

</server>
```

Key updates to enable the **JSREXEC2** server to be an executor. The copy of the new `server.xml` file brought in all the JMS definitions, not just what's highlighted here. What's highlighted here are the key things associated with being an executor server and the queue to listen on.

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>servlet-3.1</feature>
        <feature>batch-1.0</feature>
        <feature>batchManagement-1.0</feature>
        <feature>wmqJmsClient-2.0</feature>
        <feature>appSecurity-1.0</feature>
    </featureManager>

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

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

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

<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>

<batchPersistence jobStoreRef="BatchDatabaseStore" />

<databaseStore id="BatchDatabaseStore"
 dataSourceRef="batchDB" schema="JBATCH" tablePrefix="" />

<jdbcDriver id="DB2T4" libraryRef="DB2T4LibRef" />

<library id="DB2T4LibRef">
 <fileset dir="/shared/db21210/jdbc/classes/"
 includes="db2jcc4.jar db2jcc_license_cisuz.jar sqlj4.zip" />
</library>

<authData id="batchAlias" user="SYSADM1" password="SYSADM1" />

<dataSource id="batchDB"
 containerAuthDataRef="batchAlias"
 type="javax.sql.XADataSource"
 jdbcDriverRef="DB2T4">
 <properties.db2.jcc
  serverName="wg31.washington.ibm.com"
  portNumber="2446"
  databaseName="DSN2LOC"
  driverType="4" />
</dataSource>

<dataSource id="bonusDB" jndiName="jdbc/bonus"
 type="javax.sql.XADataSource"
 jdbcDriverRef="DB2T4">
 <properties.db2.jcc
  serverName="wg31.washington.ibm.com"
  portNumber="2446"
  databaseName="DSN2LOC"
  user="SYSADM1"
```

```
      password="SYSADM1"
      driverType="4" />
    </dataSource>

    <batchJmsExecutor activationSpecRef="batchActivationSpec2"
      queueRef="batchJobSubmissionQueue"/>

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

    <wmqJmsClient startupRetryCount="999"
      startupRetryInterval="1000ms"
      reconnectionRetryCount="10"
      reconnectionRetryInterval="5m">
    </wmqJmsClient>

    <jmsActivationSpec id="batchActivationSpec2" >
      <properties.wmqJms
        destinationRef="batchJobSubmissionQueue"
  messageSelector="com_ibm_ws_batch_applicationName = 'BonusPayout-1.0'"
        transportType="CLIENT"
        channel="SYSTEM.DEF.SVRCONN"
        destinationType="javax.jms.Queue"
        queueManager="MQS1"
        hostName="wg31.washington.ibm.com"
        port="1414">
      </properties.wmqJms>
    </jmsActivationSpec>

    <jmsQueue id="batchJobSubmissionQueue"
      jndiName="jms/batch/jobSubmissionQueue">
      <properties.wmqJms baseQueueName="JAVA.BATCH.QUEUE"
         baseQueueManagerName="MQS1">
      </properties.wmqJms>
    </jmsQueue>

    <httpEndpoint id="defaultHttpEndpoint"
                  host="*"
                  httpPort="12080"
                  httpsPort="12443" />

</server>
```

The following update to the server.xml of JSREXEC1 introduced a conditional message selector. Here the selector was coded to select SleepyBatchlet *and* a specific job property value:

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>servlet-3.1</feature>
        <feature>batch-1.0</feature>
        <feature>batchManagement-1.0</feature>
        <feature>appSecurity-2.0</feature>
        <feature>wmqJmsClient-2.0</feature>
    </featureManager>

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

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

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

<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>

<batchPersistence jobStoreRef="BatchDatabaseStore" />

<databaseStore id="BatchDatabaseStore"
 dataSourceRef="batchDB" schema="JBATCH" tablePrefix="" />

<jdbcDriver id="DB2T4" libraryRef="DB2T4LibRef" />
 <library id="DB2T4LibRef">
  <fileset dir="/shared/db21210/jdbc/classes/"
  includes="db2jcc4.jar db2jcc_license_cisuz.jar sqlj4.zip" />
 </library>

<authData id="batchAlias" user="SYSADM1" password="SYSADM1" />

<dataSource id="batchDB"
 containerAuthDataRef="batchAlias"
 type="javax.sql.XADataSource"
 jdbcDriverRef="DB2T4">
 <properties.db2.jcc
  serverName="wg31.washington.ibm.com"
  portNumber="2446"
  databaseName="DSN2LOC"
  driverType="4" />
</dataSource>

<batchJmsExecutor activationSpecRef="batchActivationSpec1"
  queueRef="batchJobSubmissionQueue"/>

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

<wmqJmsClient startupRetryCount="999"
  startupRetryInterval="1000ms"
  reconnectionRetryCount="10"
```

```
      reconnectionRetryInterval="5m">
    </wmqJmsClient>

    <jmsActivationSpec id="batchActivationSpec1" >
      <properties.wmqJms
        destinationRef="batchJobSubmissionQueue"
        messageSelector="com_ibm_ws_batch_applicationName =
          'SleepyBatchletSample-1.0' AND jobPriority = '1'"
        maxPoolDepth="1"
        transportType="CLIENT"
        channel="SYSTEM.DEF.SVRCONN"
        destinationType="javax.jms.Queue"
        queueManager="MQS1"
        hostName="wg31.washington.ibm.com"
        port="1414">
      </properties.wmqJms>
    </jmsActivationSpec>

    <jmsQueue id="batchJobSubmissionQueue"
      jndiName="jms/batch/jobSubmissionQueue">
      <properties.wmqJms baseQueueName="JAVA.BATCH.QUEUE"
         baseQueueManagerName="MQS1">
      </properties.wmqJms>
    </jmsQueue>

    <httpEndpoint id="defaultHttpEndpoint"
                  host="*"
                  httpPort="11080"
                  httpsPort="11443" />

</server>
```

Finally, the following update to the server.xml of JSREXEC2 introduced a conditional message selector for that server. Here the selector was coded to select SleepyBatchlet and **not** the job property coded for the other server, *or* BonusPayout.:

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>servlet-3.1</feature>
        <feature>batch-1.0</feature>
        <feature>batchManagement-1.0</feature>
        <feature>wmqJmsClient-2.0</feature>
        <feature>appSecurity-1.0</feature>
    </featureManager>

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

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

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

```
</authorization-roles>

<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>

<batchPersistence jobStoreRef="BatchDatabaseStore" />

<databaseStore id="BatchDatabaseStore"
 dataSourceRef="batchDB" schema="JBATCH" tablePrefix="" />

<jdbcDriver id="DB2T4" libraryRef="DB2T4LibRef" />

<library id="DB2T4LibRef">
 <fileset dir="/shared/db21210/jdbc/classes/"
 includes="db2jcc4.jar db2jcc_license_cisuz.jar sqlj4.zip" />
</library>

<authData id="batchAlias" user="SYSADM1" password="SYSADM1" />

<dataSource id="batchDB"
 containerAuthDataRef="batchAlias"
 type="javax.sql.XADataSource"
 jdbcDriverRef="DB2T4">
 <properties.db2.jcc
  serverName="wg31.washington.ibm.com"
  portNumber="2446"
  databaseName="DSN2LOC"
  driverType="4" />
</dataSource>

<dataSource id="bonusDB" jndiName="jdbc/bonus"
 type="javax.sql.XADataSource"
 jdbcDriverRef="DB2T4">
 <properties.db2.jcc
  serverName="wg31.washington.ibm.com"
  portNumber="2446"
  user="SYSADM1"
  password="SYSADM1"
  databaseName="DSN2LOC"
  driverType="4" />
</dataSource>

<batchJmsExecutor activationSpecRef="batchActivationSpec2"
  queueRef="batchJobSubmissionQueue"/>

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

```
    <wmqJmsClient startupRetryCount="999"
      startupRetryInterval="1000ms"
      reconnectionRetryCount="10"
      reconnectionRetryInterval="5m">
    </wmqJmsClient>

    <jmsActivationSpec id="batchActivationSpec2" >
      <properties.wmqJms
        destinationRef="batchJobSubmissionQueue"
        messageSelector="(com_ibm_ws_batch_applicationName =
         'SleepyBatchletSample-1.0' AND NOT jobPriority = '1')
          OR com_ibm_ws_batch_applicationName = 'BonusPayout-1.0'"
        transportType="CLIENT"
        channel="SYSTEM.DEF.SVRCONN"
        destinationType="javax.jms.Queue"
        queueManager="MQS1"
        hostName="wg31.washington.ibm.com"
        port="1414">
      </properties.wmqJms>
    </jmsActivationSpec>

    <jmsQueue id="batchJobSubmissionQueue"
      jndiName="jms/batch/jobSubmissionQueue">
      <properties.wmqJms baseQueueName="JAVA.BATCH.QUEUE"
         baseQueueManagerName="MQS1">
      </properties.wmqJms>
    </jmsQueue>

    <httpEndpoint id="defaultHttpEndpoint"
                  host="*"
                  httpPort="12080"
                  httpsPort="12443" />

</server>
```

## *Commands*

The commands used in this lab were as follows. Some commands are longer than one line here; those were entered as one line into the UNIX shell environment.

```
=sdsf.da

PRE JSR*

/P JSRDISP

cd /shared/jsrhome/servers/JSRDISP/dropins

rm SleepyBatchletSample-1.0.war

cp /u/user1/jbatch/lab5/jsrdisp.xml /shared/jsrhome/servers/JSRDISP/server.xml

=sdsf.da

/S JSRDISP

cp /u/user1/jbatch/apps/SleepyBatchletSample-1.0.war
/shared/jsrhome/servers/JSREXEC1/dropins/SleepyBatchletSample-1.0.war
```

```
cp /shared/mqm/V9R0M1/java/lib/jca/wmq.jmsra.rar
/shared/jsrhome/servers/JSREXEC1/wmq.jmsra.rar

cp /u/user1/jbatch/lab5/jsrexec1.xml
/shared/jsrhome/servers/JSREXEC1/server.xml

=sdsf.da

/S JSREXEC1

cd /shared/zWebSphere/Liberty/V17001/bin

./batchManager submit --batchManager=localhost:10443 --user=Fred
--password=fredpwd --applicationName=SleepyBatchletSample-1.0
--jobXMLName=sleepy-batchlet.xml --trustSslCertificates --wait

/shared/jsrhome/servers/JSREXEC1/logs

cd /shared/zWebSphere/Liberty/V17001/lib/native/zos/s390x

export STEPLIB='MQ901.SCSQAUTH'

./batchManagerZos submit --batchManager=LIBERTY+BATCH+MANAGER
--applicationName=SleepyBatchletSample-1.0 --jobXMLName=sleepy-batchlet.xml
--queueManagerName=MQS1 --wait

cp /u/user1/jbatch/apps/BonusPayout-1.0.war
/shared/jsrhome/servers/JSREXEC2/dropins/BonusPayout-1.0.war

cp /shared/mqm/V9R0M1/java/lib/jca/wmq.jmsra.rar
/shared/jsrhome/servers/JSREXEC2/wmq.jmsra.rar

cp /u/user1/jbatch/lab5/jsrexec2.xml
/shared/jsrhome/servers/JSREXEC2/server.xml

/S JSREXEC2

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

cp /u/user1/jbatch/apps/SleepyBatchletSample-1.0.war
/shared/jsrhome/servers/JSREXEC2/dropins/SleepyBatchletSample-1.0.war

cp /u/user1/jbatch/lab5/jsrexec1_2.xml
/shared/jsrhome/servers/JSREXEC1/server.xml

cp /u/user1/jbatch/lab5/jsrexec2_2.xml
/shared/jsrhome/servers/JSREXEC2/server.xml

./batchManagerZos submit --batchManager=LIBERTY+BATCH+MANAGER
--applicationName=SleepyBatchletSample-1.0 --jobXMLName=sleepy-batchlet.xml
--queueManagerName=MQS1 --jobParameter=jobPriority=1 --wait

./batchManagerZos submit --batchManager=LIBERTY+BATCH+MANAGER
--applicationName=SleepyBatchletSample-1.0 --jobXMLName=sleepy-batchlet.xml
--queueManagerName=MQS1 --jobParameter=jobPriority=2 --wait
```

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

# Lab 6 - Security

## *RACF jobs*

The following RACF job created the FRED identity in SAF, then created the three EJBROLEs for Java batch. It granted FRED access to the administrator role. Finally, it allowed FRED access to the CBIND so that ID could use batchManagerZos and WOLA to access the server.

```
//LAB6A   JOB (????,????),'LAB6 JOB',MSGCLASS=O,CLASS=A,
//         NOTIFY=????????,REGION=4000K TYPRUN=HOLD
//*-------------------------------------------------------------*
//RACF EXEC PGM=IKJEFT01,REGION=0M
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
ADDUSER FRED DFLTGRP(SYS1) OMVS(AUTOUID HOME(/u/fred/) -
  PROGRAM(/bin/sh)) NAME('USER FRED')

ALTUSER FRED PASSWORD(FREDSAF) NOEXPIRED

RDEFINE EJBROLE BBGZDFLT.com.ibm.ws.batch.batchAdmin -
  OWNER(SYS1) UACC(NONE)

RDEFINE EJBROLE BBGZDFLT.com.ibm.ws.batch.batchSubmitter -
  OWNER(SYS1) UACC(NONE)

RDEFINE EJBROLE BBGZDFLT.com.ibm.ws.batch.batchMonitor -
  OWNER(SYS1) UACC(NONE)

PERMIT BBGZDFLT.com.ibm.ws.batch.batchAdmin -
  CLASS(EJBROLE) ID(FRED) ACCESS(READ)

SETROPTS RACLIST(EJBROLE) REFRESH

PERMIT BBG.WOLA.LIBERTY.BATCH.MANAGER CLASS(CBIND) -
  ACCESS(READ) ID(FRED)

SETROPTS RACLIST(CBIND) REFRESH
/*
//*
```

The following RACF job set up the SAF profiles to allow batchManager to be used with SAF. This involved setting up the unauthenticated guest ID, the APPL profile, and EJBROLE profiles for FRED to access the AdminCenter.

```
//LAB6B   JOB (????,????),'LAB6 JOB',MSGCLASS=O,CLASS=A,
//         NOTIFY=????????,REGION=4000K TYPRUN=HOLD
//*-------------------------------------------------------------*
//RACF EXEC PGM=IKJEFT01,REGION=0M
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
ADDGROUP WSGUESTG OMVS(AUTOGID) OWNER(SYS1)

ADDUSER WSGUEST RESTRICTED DFLTGRP(WSGUESTG) OMVS(AUTOUID -
  HOME(/u/wsguest) PROGRAM(/bin/sh)) -
  NAME('UNAUTHENTICATED USER') NOPASSWORD NOOIDCARD

RDEFINE APPL BBGZDFLT UACC(NONE) OWNER(SYS1)

PERMIT BBGZDFLT CLASS(APPL) RESET
```

```
PERMIT BBGZDFLT CLASS(APPL) ACCESS(READ) ID(WSGUEST)

RALT APPL BBGZDFLT UACC(READ)

SETROPTS RACLIST(APPL) REFRESH

RDEFINE EJBROLE -
BBGZDFLT.com.ibm.ws.management.security.resource.allAuthenticatedUsers -
  OWNER(SYS1) UACC(NONE)

RDEFINE EJBROLE -
BBGZDFLT.com.ibm.ws.management.security.resource.Administrator -
  OWNER(SYS1) UACC(NONE)

PERMIT -
BBGZDFLT.com.ibm.ws.management.security.resource.allAuthenticatedUsers -
  CLASS(EJBROLE) ID(FRED) ACCESS(READ)

PERMIT BBGZDFLT.com.ibm.ws.management.security.resource.Administrator -
  CLASS(EJBROLE) ID(FRED) ACCESS(READ)

SETROPTS RACLIST(EJBROLE) REFRESH
/*
//*
```

Finally, the following RACF job set up the certificates and keyrings so the encryption artifacts could be managed by SAF:

```
//LAB6C   JOB (????,????),'LAB6 JOB',MSGCLASS=O,CLASS=A,
//         NOTIFY=????????,REGION=4000K TYPRUN=HOLD
//*-------------------------------------------------------------*
//RACF EXEC PGM=IKJEFT01,REGION=0M
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
RACDCERT CERTAUTH GENCERT SUBJECTSDN(CN('CA for Liberty') -
  OU('LIBERTY')) WITHLABEL('LibertyCA.LIBERTY') TRUST -
  SIZE(2048) NOTAFTER(DATE(2031/12/31))

RACDCERT ID(JSRSERV) GENCERT SUBJECTSDN(CN('wg31.washington.ibm.com') -
  O('IBM') OU('LIBERTY')) WITHLABEL('DefaultCert.LIBERTY') -
  SIGNWITH(CERTAUTH LABEL('LibertyCA.LIBERTY')) SIZE(2048) -
  NOTAFTER(DATE(2031/12/31))

RACDCERT ID(JSRSERV) ADDRING(Keyring.SERVER)
RACDCERT ID(USER1) ADDRING(Keyring.CLIENT)

RACDCERT CONNECT(ID(JSRSERV) -
  LABEL('DefaultCert.LIBERTY') RING(Keyring.SERVER)) -
  ID(JSRSERV)

RACDCERT CONNECT(CERTAUTH LABEL('LibertyCA.LIBERTY') -
  RING(Keyring.SERVER)) ID(JSRSERV)

RACDCERT CONNECT(CERTAUTH LABEL('LibertyCA.LIBERTY') -
  RING(Keyring.CLIENT)) ID(USER1)

PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(JSRSERV) ACCESS(READ)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(USER1) ACCESS(READ)
```

```
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(JSRSERV) ACCESS(READ)
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(USER1) ACCESS(READ)

SETROPTS RACLIST(DIGTCERT DIGTRING) REFRESH
SETROPTS RACLIST(FACILITY) REFRESH
/*
//*
```

### *Configuration server.xml file: reset environment to known state*

The previous lab (Lab 5) updated the XML with various message selector variations. At the start of Lab 6 we reduced the servers to just JSRDISP and JSREXEC1, and we reset the environment to a known *non-SAF* state.

The JSRDISP `server.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>servlet-3.1</feature>
        <feature>batch-1.0</feature>
        <feature>batchManagement-1.0</feature>
        <feature>appSecurity-2.0</feature>
        <feature>zosLocalAdapters-1.0</feature>
        <feature>wmqJmsClient-2.0</feature>
        <feature>adminCenter-1.0</feature>
    </featureManager>

    <batchJmsDispatcher
      connectionFactoryRef="batchConnectionFactory"
      queueRef="batchJobSubmissionQueue" />

    <jmsQueue id="batchJobSubmissionQueue"
      jndiName="jms/batch/jobSubmissionQueue">
      <properties.wmqJms baseQueueName="JAVA.BATCH.QUEUE"
        priority="QDEF"
        baseQueueManagerName="MQS1">
      </properties.wmqJms>
    </jmsQueue>

    <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>

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

    <wmqJmsClient startupRetryCount="999"
      startupRetryInterval="1000ms"
```

```
      reconnectionRetryCount="10"
      reconnectionRetryInterval="5m">
    </wmqJmsClient>

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

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

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

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

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

    <batchPersistence jobStoreRef="BatchDatabaseStore" />

    <databaseStore id="BatchDatabaseStore"
      createTables="false"
      dataSourceRef="batchDB" schema="JBATCH" tablePrefix="" />
    <jdbcDriver id="DB2T4" libraryRef="DB2T4LibRef" />

    <library id="DB2T4LibRef">
      <fileset dir="/shared/db21210/jdbc/classes/"
      includes="db2jcc4.jar db2jcc_license_cisuz.jar sqlj4.zip" />
    </library>

    <authData id="batchAlias" user="SYSADM1" password="SYSADM1" />

    <dataSource id="batchDB"
      containerAuthDataRef="batchAlias"
      type="javax.sql.XADataSource"
      jdbcDriverRef="DB2T4">
     <properties.db2.jcc
       serverName="wg31.washington.ibm.com"
       portNumber="2446"
       databaseName="DSN2LOC"
       driverType="4" />
    </dataSource>

    <httpEndpoint id="defaultHttpEndpoint"
        host="*"
        httpPort="10080"
        httpsPort="10443" />

</server>
```

The JSREXEC1 `server.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>servlet-3.1</feature>
        <feature>batch-1.0</feature>
        <feature>batchManagement-1.0</feature>
        <feature>appSecurity-2.0</feature>
        <feature>wmqJmsClient-2.0</feature>
    </featureManager>

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

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

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

    <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>

    <batchPersistence jobStoreRef="BatchDatabaseStore" />

    <databaseStore id="BatchDatabaseStore"
     dataSourceRef="batchDB" schema="JBATCH" tablePrefix="" />

    <jdbcDriver id="DB2T4" libraryRef="DB2T4LibRef" />
     <library id="DB2T4LibRef">
      <fileset dir="/shared/db21210/jdbc/classes/"
      includes="db2jcc4.jar db2jcc_license_cisuz.jar sqlj4.zip" />
     </library>

    <authData id="batchAlias" user="SYSADM1" password="SYSADM1" />

    <dataSource id="batchDB"
     containerAuthDataRef="batchAlias"
     type="javax.sql.XADataSource"
     jdbcDriverRef="DB2T4">
     <properties.db2.jcc
```

```
      serverName="wg31.washington.ibm.com"
      portNumber="2446"
      databaseName="DSN2LOC"
      driverType="4" />
    </dataSource>

    <batchJmsExecutor activationSpecRef="batchActivationSpec1"
      queueRef="batchJobSubmissionQueue"/>

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

    <wmqJmsClient startupRetryCount="999"
      startupRetryInterval="1000ms"
      reconnectionRetryCount="10"
      reconnectionRetryInterval="5m">
    </wmqJmsClient>

    <jmsActivationSpec id="batchActivationSpec1" >
      <properties.wmqJms
        destinationRef="batchJobSubmissionQueue"
   messageSelector="com_ibm_ws_batch_applicationName = 'SleepyBatchletSample-1.0'"
        maxPoolDepth="1"
        transportType="CLIENT"
        channel="SYSTEM.DEF.SVRCONN"
        destinationType="javax.jms.Queue"
        queueManager="MQS1"
        hostName="wg31.washington.ibm.com"
        port="1414">
      </properties.wmqJms>
    </jmsActivationSpec>

    <jmsQueue id="batchJobSubmissionQueue"
      jndiName="jms/batch/jobSubmissionQueue">
      <properties.wmqJms baseQueueName="JAVA.BATCH.QUEUE"
         baseQueueManagerName="MQS1">
      </properties.wmqJms>
    </jmsQueue>

    <httpEndpoint id="defaultHttpEndpoint"
                  host="*"
                  httpPort="11080"
                  httpsPort="11443" />

</server>
```

### *batchManagerZos: SAF authentication and authorization*

The batchManagerZos client uses WOLA, not HTTP/REST, so there is no need to set up the encryption, unauthenticated user, and APPL profiles.  But it was necessary to tell the servers to go to SAF for userid information.

First, the XML copied over to the JSRDISP server:

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>servlet-3.1</feature>
```

```
        <feature>batch-1.0</feature>
        <feature>batchManagement-1.0</feature>
        <feature>appSecurity-2.0</feature>
        <feature>zosLocalAdapters-1.0</feature>
        <feature>wmqJmsClient-2.0</feature>
        <feature>adminCenter-1.0</feature>
        <feature>zosSecurity-1.0</feature>
    </featureManager>

    <batchJmsDispatcher
      connectionFactoryRef="batchConnectionFactory"
      queueRef="batchJobSubmissionQueue" />

    <jmsQueue id="batchJobSubmissionQueue"
      jndiName="jms/batch/jobSubmissionQueue">
      <properties.wmqJms baseQueueName="JAVA.BATCH.QUEUE"
        priority="QDEF"
        baseQueueManagerName="MQS1">
      </properties.wmqJms>
    </jmsQueue>

    <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>

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

    <wmqJmsClient startupRetryCount="999"
      startupRetryInterval="1000ms"
      reconnectionRetryCount="10"
      reconnectionRetryInterval="5m">
    </wmqJmsClient>

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

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

    <safRegistry id="saf" />
    <safAuthorization racRouteLog="ASIS" />
    <safCredentials profilePrefix="BBGZDFLT" />

    <batchPersistence jobStoreRef="BatchDatabaseStore" />

    <databaseStore id="BatchDatabaseStore"
      createTables="false"
      dataSourceRef="batchDB" schema="JBATCH" tablePrefix="" />
```

```
    <jdbcDriver id="DB2T4" libraryRef="DB2T4LibRef" />

    <library id="DB2T4LibRef">
      <fileset dir="/shared/db21210/jdbc/classes/"
      includes="db2jcc4.jar db2jcc_license_cisuz.jar sqlj4.zip" />
    </library>

    <authData id="batchAlias" user="SYSADM1" password="SYSADM1" />

    <dataSource id="batchDB"
      containerAuthDataRef="batchAlias"
      type="javax.sql.XADataSource"
      jdbcDriverRef="DB2T4">
     <properties.db2.jcc
       serverName="wg31.washington.ibm.com"
       portNumber="2446"
       databaseName="DSN2LOC"
       driverType="4" />
    </dataSource>

    <httpEndpoint id="defaultHttpEndpoint"
        host="*"
        httpPort="10080"
        httpsPort="10443" />

</server>
```

And the `server.xml` copied to the JSREXEC1 server:

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>servlet-3.1</feature>
        <feature>batch-1.0</feature>
        <feature>batchManagement-1.0</feature>
        <feature>appSecurity-2.0</feature>
        <feature>wmqJmsClient-2.0</feature>
        <feature>zosSecurity-1.0</feature>
    </featureManager>

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

    <safRegistry id="saf" />
    <safAuthorization racRouteLog="ASIS" />
    <safCredentials profilePrefix="BBGZDFLT" />

    <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>

   <batchPersistence jobStoreRef="BatchDatabaseStore" />

   <databaseStore id="BatchDatabaseStore"
    dataSourceRef="batchDB" schema="JBATCH" tablePrefix="" />

   <jdbcDriver id="DB2T4" libraryRef="DB2T4LibRef" />
    <library id="DB2T4LibRef">
     <fileset dir="/shared/db21210/jdbc/classes/"
     includes="db2jcc4.jar db2jcc_license_cisuz.jar sqlj4.zip" />
    </library>

   <authData id="batchAlias" user="SYSADM1" password="SYSADM1" />

   <dataSource id="batchDB"
    containerAuthDataRef="batchAlias"
    type="javax.sql.XADataSource"
    jdbcDriverRef="DB2T4">
    <properties.db2.jcc
     serverName="wg31.washington.ibm.com"
     portNumber="2446"
     databaseName="DSN2LOC"
     driverType="4" />
   </dataSource>

   <batchJmsExecutor activationSpecRef="batchActivationSpec1"
     queueRef="batchJobSubmissionQueue"/>

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

   <wmqJmsClient startupRetryCount="999"
     startupRetryInterval="1000ms"
     reconnectionRetryCount="10"
     reconnectionRetryInterval="5m">
   </wmqJmsClient>

   <jmsActivationSpec id="batchActivationSpec1" >
     <properties.wmqJms
       destinationRef="batchJobSubmissionQueue"
  messageSelector="com_ibm_ws_batch_applicationName = 'SleepyBatchletSample-1.0'"
       maxPoolDepth="1"
       transportType="CLIENT"
       channel="SYSTEM.DEF.SVRCONN"
       destinationType="javax.jms.Queue"
       queueManager="MQS1"
       hostName="wg31.washington.ibm.com"
       port="1414">
     </properties.wmqJms>
   </jmsActivationSpec>

   <jmsQueue id="batchJobSubmissionQueue"
     jndiName="jms/batch/jobSubmissionQueue">
     <properties.wmqJms baseQueueName="JAVA.BATCH.QUEUE"
        baseQueueManagerName="MQS1">
     </properties.wmqJms>
   </jmsQueue>
```

```
    <httpEndpoint id="defaultHttpEndpoint"
                  host="*"
                  httpPort="11080"
                  httpsPort="11443" />

</server>
```

### *AdminCenter + batchManager: SAF Keyrings and SSL*

The previous update to `server.xml` pushed authentication and authorization down to SAF, but left encryption handled with the file-based key stores. Here we push encryption down to SAF as well.

This is the update to the JSRDISP server:

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>servlet-3.1</feature>
        <feature>batch-1.0</feature>
        <feature>batchManagement-1.0</feature>
        <feature>appSecurity-2.0</feature>
        <feature>zosLocalAdapters-1.0</feature>
        <feature>wmqJmsClient-2.0</feature>
        <feature>adminCenter-1.0</feature>
        <feature>zosSecurity-1.0</feature>
        <feature>ssl-1.0</feature>
    </featureManager>

    <batchJmsDispatcher
      connectionFactoryRef="batchConnectionFactory"
      queueRef="batchJobSubmissionQueue" />

    <jmsQueue id="batchJobSubmissionQueue"
      jndiName="jms/batch/jobSubmissionQueue">
      <properties.wmqJms baseQueueName="JAVA.BATCH.QUEUE"
        priority="QDEF"
        baseQueueManagerName="MQS1">
      </properties.wmqJms>
    </jmsQueue>

    <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>

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

```
<wmqJmsClient startupRetryCount="999"
  startupRetryInterval="1000ms"
  reconnectionRetryCount="10"
  reconnectionRetryInterval="5m">
</wmqJmsClient>

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

<sslDefault sslRef="DefaultSSLSettings" />
<ssl id="DefaultSSLSettings"
  keyStoreRef="CellDefaultKeyStore"
  trustStoreRef="CellDefaultTrustStore" />
<keyStore id="CellDefaultKeyStore"
  location="safkeyring:///Keyring.SERVER"
  password="password" type="JCERACFKS"
  fileBased="false" readOnly="true" />
<keyStore id="CellDefaultTrustStore"
  location="safkeyring:///Keyring.SERVER"
  password="password" type="JCERACFKS"
  fileBased="false" readOnly="true" />

<safRegistry id="saf" />
<safAuthorization racRouteLog="ASIS" />
<safCredentials profilePrefix="BBGZDFLT" />

<batchPersistence jobStoreRef="BatchDatabaseStore" />

<databaseStore id="BatchDatabaseStore"
  createTables="false"
  dataSourceRef="batchDB" schema="JBATCH" tablePrefix="" />
<jdbcDriver id="DB2T4" libraryRef="DB2T4LibRef" />

<library id="DB2T4LibRef">
  <fileset dir="/shared/db21210/jdbc/classes/"
  includes="db2jcc4.jar db2jcc_license_cisuz.jar sqlj4.zip" />
</library>

<authData id="batchAlias" user="SYSADM1" password="SYSADM1" />

<dataSource id="batchDB"
  containerAuthDataRef="batchAlias"
  type="javax.sql.XADataSource"
  jdbcDriverRef="DB2T4">
 <properties.db2.jcc
   serverName="wg31.washington.ibm.com"
   portNumber="2446"
   databaseName="DSN2LOC"
   driverType="4" />
</dataSource>

<httpEndpoint id="defaultHttpEndpoint"
    host="*"
    httpPort="10080"
    httpsPort="10443" />
```

```
</server>
```

And this is the update to the JSREXEC1 server:

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>servlet-3.1</feature>
        <feature>batch-1.0</feature>
        <feature>batchManagement-1.0</feature>
        <feature>appSecurity-2.0</feature>
        <feature>wmqJmsClient-2.0</feature>
        <feature>zosSecurity-1.0</feature>
        <feature>ssl-1.0</feature>
    </featureManager>

    <sslDefault sslRef="DefaultSSLSettings" />
    <ssl id="DefaultSSLSettings"
      keyStoreRef="CellDefaultKeyStore"
      trustStoreRef="CellDefaultTrustStore" />
    <keyStore id="CellDefaultKeyStore"
      location="safkeyring:///Keyring.SERVER"
      password="password" type="JCERACFKS"
      fileBased="false" readOnly="true" />
    <keyStore id="CellDefaultTrustStore"
      location="safkeyring:///Keyring.SERVER"
      password="password" type="JCERACFKS"
      fileBased="false" readOnly="true" />

    <safRegistry id="saf" />
    <safAuthorization racRouteLog="ASIS" />
    <safCredentials profilePrefix="BBGZDFLT" />

    <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>

    <batchPersistence jobStoreRef="BatchDatabaseStore" />

    <databaseStore id="BatchDatabaseStore"
     dataSourceRef="batchDB" schema="JBATCH" tablePrefix="" />

    <jdbcDriver id="DB2T4" libraryRef="DB2T4LibRef" />
     <library id="DB2T4LibRef">
      <fileset dir="/shared/db21210/jdbc/classes/"
      includes="db2jcc4.jar db2jcc_license_cisuz.jar sqlj4.zip" />
     </library>
```

```
    <authData id="batchAlias" user="SYSADM1" password="SYSADM1" />

    <dataSource id="batchDB"
     containerAuthDataRef="batchAlias"
     type="javax.sql.XADataSource"
     jdbcDriverRef="DB2T4">
     <properties.db2.jcc
      serverName="wg31.washington.ibm.com"
      portNumber="2446"
      databaseName="DSN2LOC"
      driverType="4" />
    </dataSource>

    <batchJmsExecutor activationSpecRef="batchActivationSpec1"
      queueRef="batchJobSubmissionQueue"/>

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

    <wmqJmsClient startupRetryCount="999"
      startupRetryInterval="1000ms"
      reconnectionRetryCount="10"
      reconnectionRetryInterval="5m">
    </wmqJmsClient>

    <jmsActivationSpec id="batchActivationSpec1" >
      <properties.wmqJms
        destinationRef="batchJobSubmissionQueue"
   messageSelector="com_ibm_ws_batch_applicationName = 'SleepyBatchletSample-
1.0'"
        maxPoolDepth="1"
        transportType="CLIENT"
        channel="SYSTEM.DEF.SVRCONN"
        destinationType="javax.jms.Queue"
        queueManager="MQS1"
        hostName="wg31.washington.ibm.com"
        port="1414">
      </properties.wmqJms>
    </jmsActivationSpec>

    <jmsQueue id="batchJobSubmissionQueue"
      jndiName="jms/batch/jobSubmissionQueue">
      <properties.wmqJms baseQueueName="JAVA.BATCH.QUEUE"
         baseQueueManagerName="MQS1">
      </properties.wmqJms>
    </jmsQueue>

    <httpEndpoint id="defaultHttpEndpoint"
                  host="*"
                  httpPort="11080"
                  httpsPort="11443" />

</server>
```

## *Commands*

The commands used in this lab were as follows. Some commands are longer than one line here; those were entered as one line into the UNIX shell environment.

```
=sdsf.da

PRE JSR*

/P JSRDISP

/P JSREXEC1

/P JSREXEC2

cp /u/user1/jbatch/lab6/jsrdisp.xml /shared/jsrhome/servers/JSRDISP/server.xml

cp /u/user1/jbatch/lab6/jsrexec1.xml
/shared/jsrhome/servers/JSREXEC1/server.xml

/S JSRDISP

/S JSREXEC1

http://wg31.washington.ibm.com:10080/adminCenter/

Fred

fredpwd

cd /shared/zWebSphere/Liberty/V17001/bin

export JAVA_HOME=/shared/java/J8.0_64/

./batchManager submit --batchManager=localhost:10443 --user=Fred
--password=fredpwd --applicationName=SleepyBatchletSample-1.0
--jobXMLName=sleepy-batchlet.xml --trustSslCertificates --wait

export STEPLIB='MQ901.SCSQAUTH'

cd /shared/zWebSphere/Liberty/V17001/lib/native/zos/s390x

./batchManagerZos submit --batchManager=LIBERTY+BATCH+MANAGER
--applicationName=SleepyBatchletSample-1.0 --jobXMLName=sleepy-batchlet.xml
--wait --queueManagerName=MQS1

USER1.JBATCH.JCL

cp /u/user1/jbatch/lab6/jsrdisp_2.xml
/shared/jsrhome/servers/JSRDISP/server.xml

cp /u/user1/jbatch/lab6/jsrexec1_2.xml
/shared/jsrhome/servers/JSREXEC1/server.xml

FRED

FREDSAF

cd /shared/zWebSphere/Liberty/V17001/lib/native/zos/s390x
```

```
export STEPLIB='MQ901.SCSQAUTH'

./batchManagerZos ping --batchManager=LIBERTY+BATCH+MANAGER

./batchManagerZos submit --batchManager=LIBERTY+BATCH+MANAGER
--applicationName=SleepyBatchletSample-1.0 --jobXMLName=sleepy-batchlet.xml
--wait --queueManagerName=MQS1

exit

USER1.JBATCH.JCL

http://wg31.washington.ibm.com:10080/adminCenter/

Fred  fredpwd

FRED  FREDSAF

whoami

cd /shared/zWebSphere/Liberty/V17001/bin

./batchManager submit --batchManager=localhost:10443 --user=FRED
--password=FREDSAF --applicationName=SleepyBatchletSample-1.0
--jobXMLName=sleepy-batchlet.xml --trustSslCertificates --wait

USER1.JBATCH.JCL

cp /u/user1/jbatch/lab6/jsrdisp_3.xml
/shared/jsrhome/servers/JSRDISP/server.xml

cp /u/user1/jbatch/lab6/jsrexec1_3.xml
/shared/jsrhome/servers/JSREXEC1/server.xml

http://wg31.washington.ibm.com:10080/adminCenter/

FRED  FREDSAF

whoami

export JVM_ARGS="-Djavax.net.ssl.trustStore=safkeyring://USER1/Keyring.CLIENT
-Djavax.net.ssl.trustStoreType=JCERACFKS
-Djavax.net.ssl.keyStore=safkeyring://USER1/Keyring.CLIENT
-Djavax.net.ssl.keyStoreType=JCERACFKS -Dcom.ibm.ssl.keyStoreFileBased=false
-Dcom.ibm.ssl.trustStoreFileBased=false
-Djava.protocol.handler.pkgs=com.ibm.crypto.provider
-Djavax.net.ssl.keyStorePassword=password"

./batchManager submit --batchManager=wg31.washington.ibm.com:10443 --user=FRED
--password=FREDSAF --applicationName=SleepyBatchletSample-1.0
--jobXMLName=sleepy-batchlet.xml --wait
```

**End of Document**