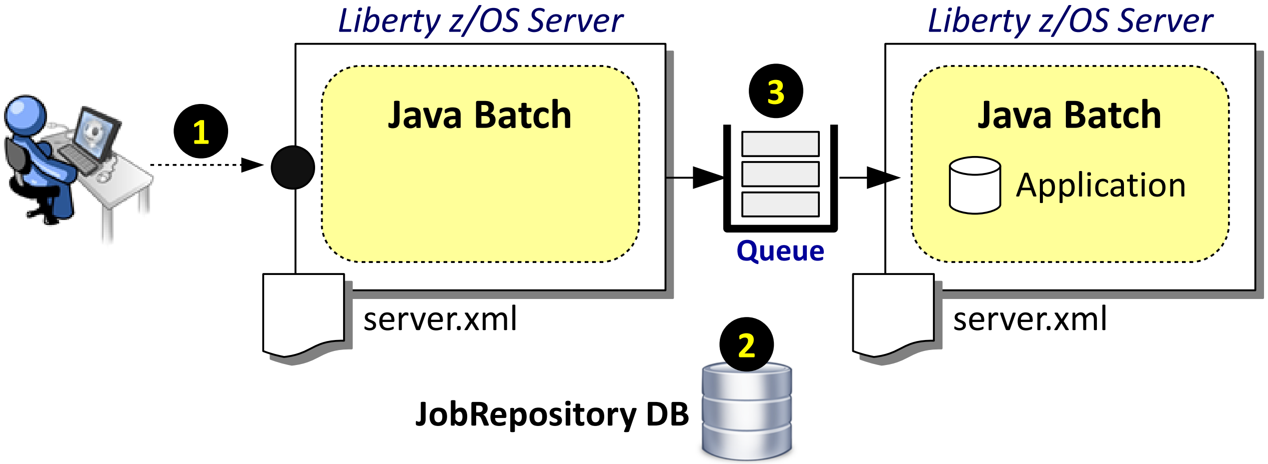
WebSphere Application Server

Unit 6

Java Batch Security

Up to This Point We've Kept Security as Simple as Possible ...



1. Client Access

We simplified this by using "basic" security in server.xml for SSL, user registry, and authorization roles.

2. DB2 Access

We simplified this by having the "authentication alias" on the JDBC definition be an ID that had access to the created batch tables.

3. MQ Access

We simplified this by having an MQ system that does not have security enabled.

We did this to keep initial focus on the other things. But the time has come to turn our attention to the "real world" implications of security with Java Batch.

Disclaimer ...

Important Disclaimer

An army of lawyers ☺

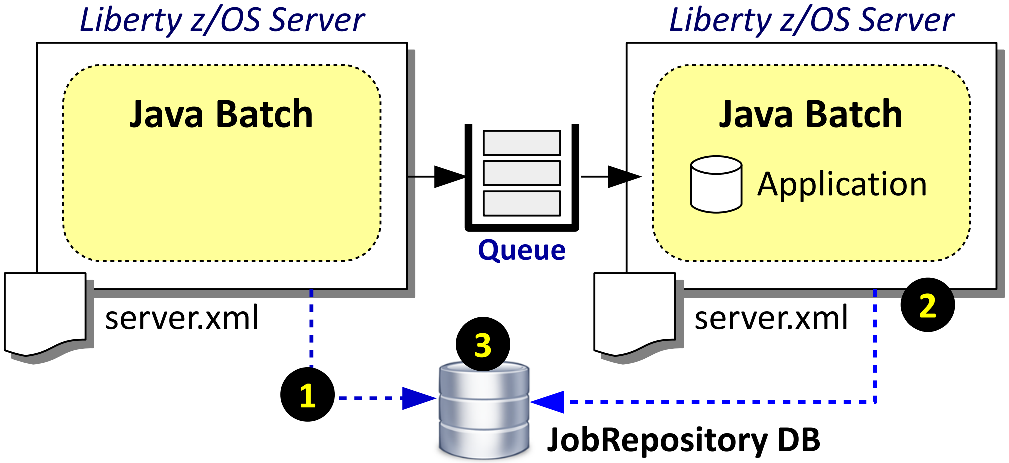
What we're providing in this unit are *examples*

Always review security examples with your security administrator

Your security policies take precedence over any examples shown here

Infrastructure

JobRepository (DB2) Access



Java Batch Tables

- JOBINSTANCE
- STEPTHREADINSTANCE
- STEPTHREADEXECUTION
- JOBEXECUTION
- JOBPARAMETER

As of 16.0.0.4. This may change if new features are added that require additional tables.

1. Type of Access

If JDBC Type 2, then the ID that asserted into DB2 will the STC ID, unless you have an auth alias on the JDBC datasource.

If JDBC Type 4, then the ID that is asserted into DB2 will be the authentication alias, or a client certificate ID.

2. Who is accessing

This is related to the type of access, but it's also related to how many servers you have accessing and whether they are the same ID or different IDs.

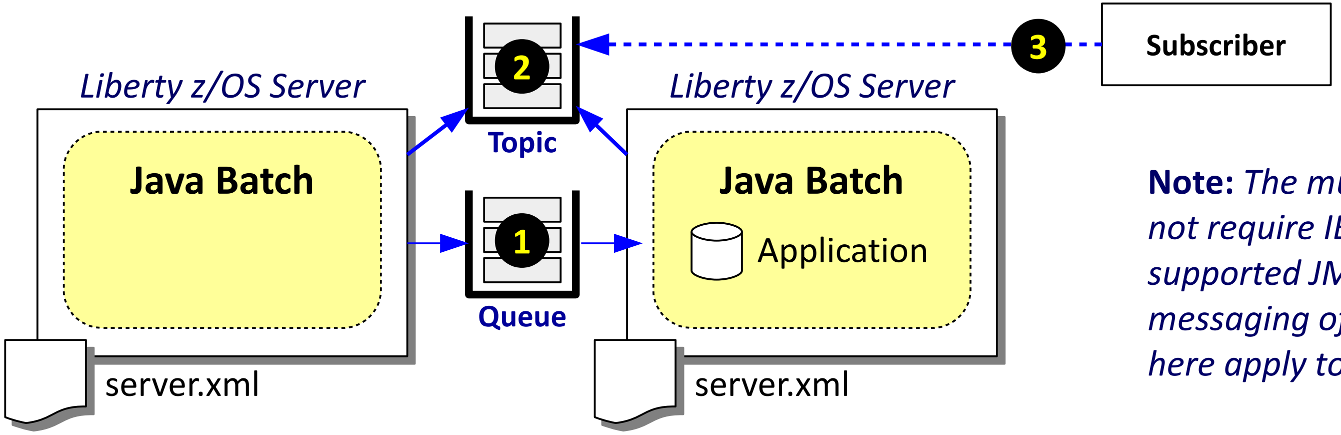
3. GRANT on what resource?

The GRANT can be against the tables, table spaces, database, STOGROUP, buffer pools. This becomes a DB Admin question how this is arranged for your environment.

Key point: this is standard DB Admin security practices. Work with your DB Admin to set up and secure the JobRepository

Multi-JVM and queue ...

Multi-JVM Queueing (MQ) Access; Batch Events Topic (MQ) Access



Note: *The multi-JVM model and batch events to not require IBM MQ. They can be used with any supported JMS provider, including the default messaging of IBM Liberty. The principles outlined here apply to all. The specifics may differ.*

1. Job Submission Queue

Only in the picture if multi-JVM. This queue must allow PUT from the dispatcher ID(s), and GET from the executor ID(s). The ID is a function of the type of access: BINDINGS (cross-memory) or CLIENT (network).

2. Topic Publishers

This may be used with either a single server model or a multi-JVM model. The servers that are configured to publish events must have the authority to do so against the target QMGR.

3. Topic Subsribers

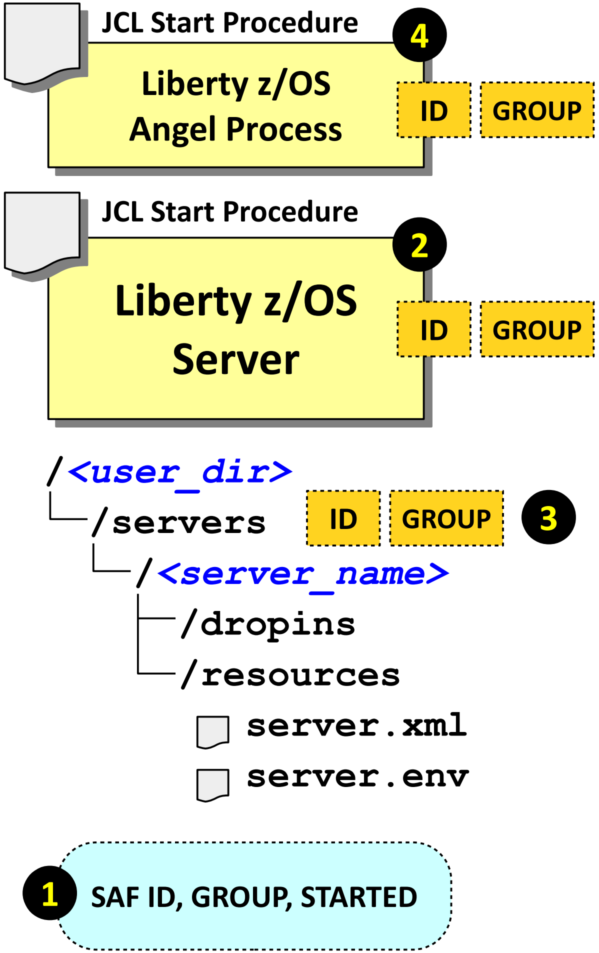
The ID of any subscriber to the topic will need to be authorized to subscribe.

batchManagerZos uses BINDINGS to access the local QMGR, so that ID is the ID that invokes the batchManagerZos client.

Key point: this is standard MQ Admin security practices. Work with your MQ Admin to set up and secure the queue and topic.

Liberty infrastructure ...

Liberty z/OS Server Infrastructure Security



1. Essential SAF Profiles

These include the ID definitions, group definitions, and the STARTED profiles to assign the IDs to the groups for started tasks.

2. Server STC ID and GROUP

The server is going to be assigned an ID, and it will have a group. There's a relationship between this ID/group and the configuration file ID/group because the server ID will need to **read** configuration files, and **write** output files.

3. Configuration file owner and group

By default this will be set to the ID/group of the userid that creates the server. You may have that ID be the same as the server STC ID/group. That is the easiest approach. But it may not be the "best," depending on your security policies. So you may require the file ownership to be different from the STC ID. Another consideration is how others will manage (delete, trim, update) the configuration files. Do you grant 'write' to the 'other' permission bits?

It's a big topic that's covered more thoroughly under the WP102687 Techdoc "Security" section. For this workshop we went with the easy approach: STC ID = file system owner ID, and your TSO ID has sufficient authority to read/write the files.

4. Angel Process

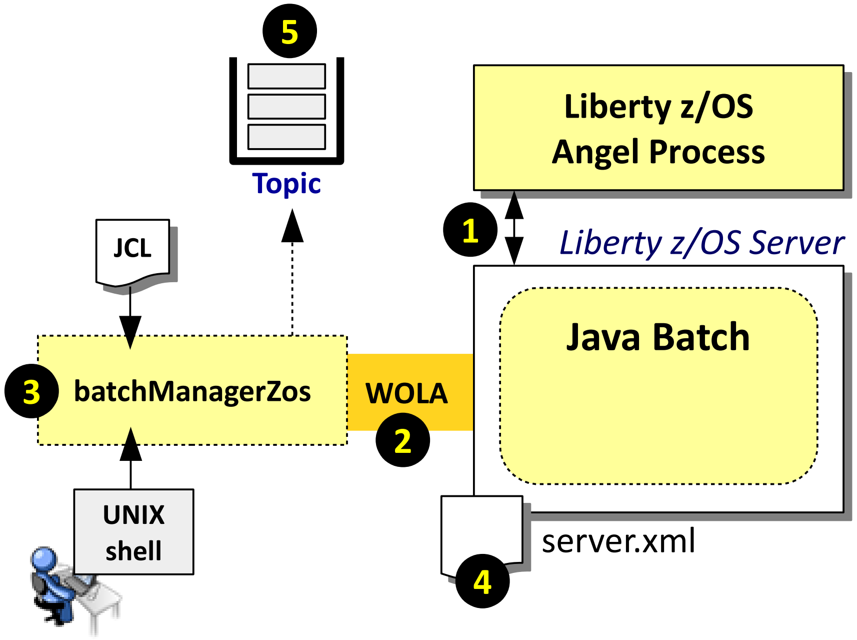
The Angel is needed when z/OS authorized services are employed. For Java Batch, that implies the batchManagerZos client. We'll defer this to that section of the unit.

batchManagerZos ...

Client Access

batchManagerZos

batchManagerZos, WOLA, the Angel Process, and Topic Subscription



1. Server access to Angel

This is controlled by SAF SERVER profiles. Details on that coming up.

2. WOLA registration into server

This is controlled by a SAF CBIND profile, which is based on the WOLA three part name configured in the server.xml. The ID under which batchManagerZos runs needs READ to that CBIND.

3. The ID under which batchManagerZos runs

This is determined by *how* the utility is invoked. If from a UNIX shell environment, then the ID in the shell environment is used. If by submitted JCL (using BPXBATCH), then it's the effective ID of the job: either the submitter's ID, or USER= on the JOB card.

4. Basic vs. SAF authentication

If you have "basic" in effect, then the batchManagerZos ID can't be determined, which is why we used SPECIAL-SUBJECT=EVERYONE earlier.

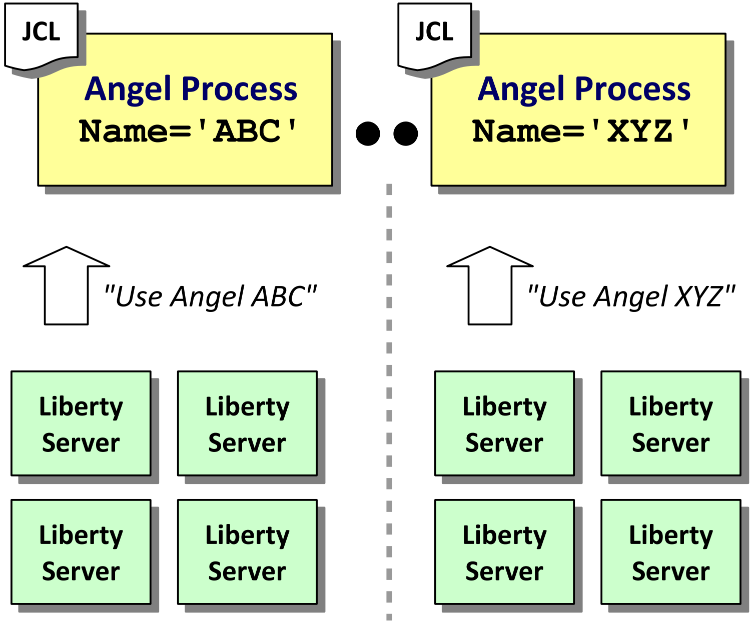
If you have SAF authentication, then it can validate the ID and use that for further role checking.

5. batchManagerZos --queueManagerName

This is optional, but if used then ID under which batchManagerZos runs will be asserted over BINDINGS mode to the named QMGR. That ID must be given authority to connect *and* to subscribe to the topic.

Detour: named angels ...

Slight Detour: "Named Angels" introduced in 16.0.0.4



bootstrap.properties

```
com.ibm.ws.zos.core.angelRequired=true
com.ibm.ws.zos.core.angelName=<name>
```

Before: one Angel per LPAR

Liberty z/OS prior to 16.0.0.4 was limited to a single Angel per LPAR. If that Angel needed to be stopped, all servers on the LPAR using that Angel were affected. The ability to isolate was needed.

Now: ability to start multiple "named" Angels

With 16.0.0.4 the Angel code was updated to take a name=' ' parameter on the PROC statement. The Angel starts with a name. Servers can be told to use a specific "named" Angel.

bootstrap.properties file

We've not mentioned this file before now. This file names properties set when the Liberty server starts initially. Two new properties can be set: use named Angel, and the name of the Angel to use. If these properties are absent, the server will seek the one "unnamed" Angel.

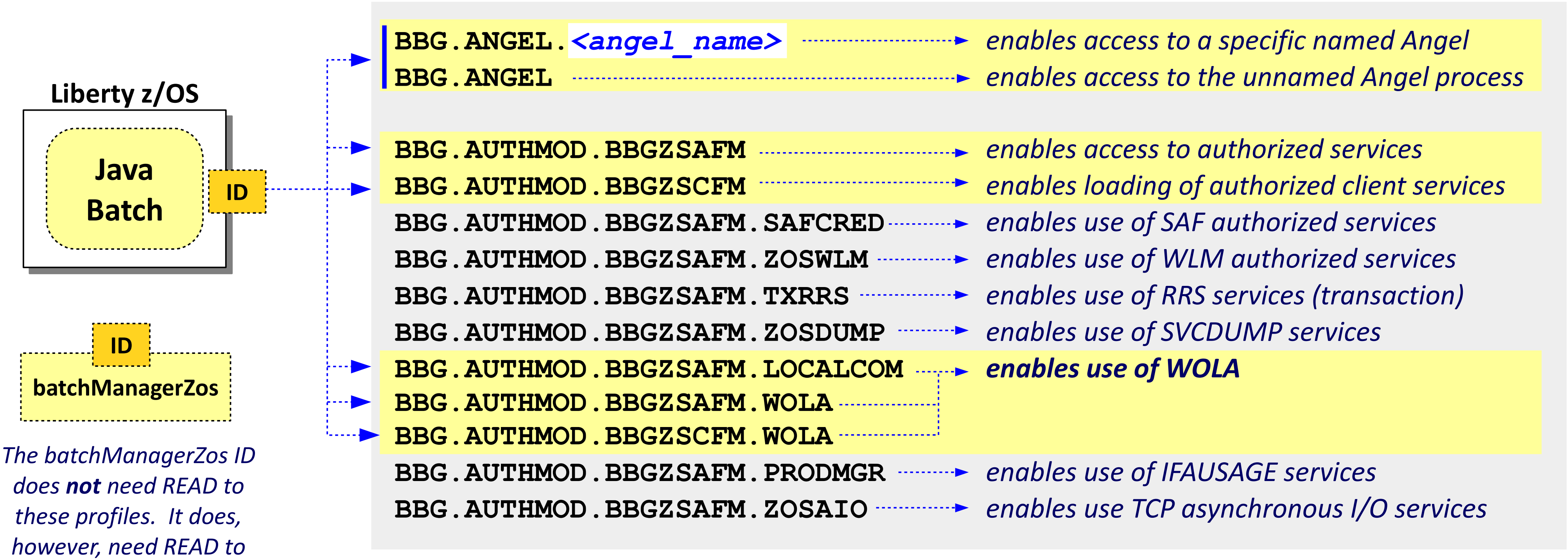
Why we mention this

Because the SAF SERVER profiles you create and grant the server ID READ to is affected by this. On the next chart we'll show you those SERVER profiles, and you'll see the "named Angel" effect.

SAF SERVER ...

SAF SERVER Profiles to Allow the Liberty z/OS Server to Use WOLA

If you're using a named Angel, then make sure your server ID has read to the BBG.ANGEL.*<angel_name>* profile. Otherwise, READ to the BBG.ANGEL profile.



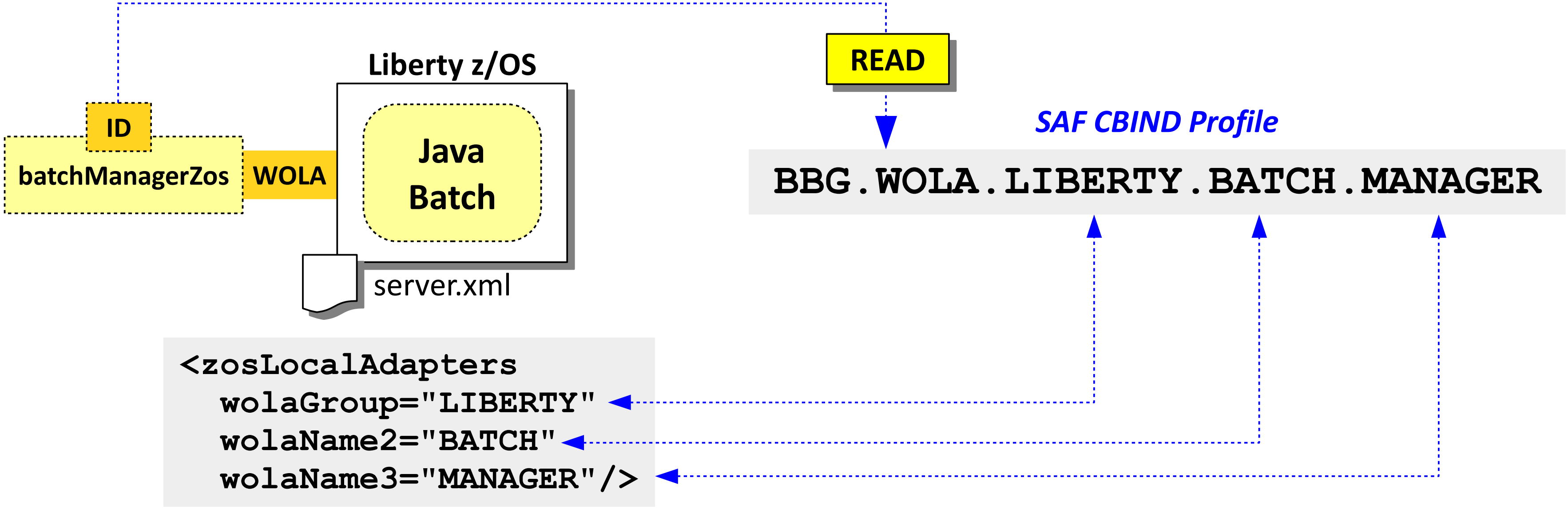
BBG.ANGEL.*<angel_name>*	*enables access to a specific named Angel*
BBG.ANGEL	*enables access to the unnamed Angel process*
BBG.AUTHMOD.BBGZSAFM	*enables access to authorized services*
BBG.AUTHMOD.BBGZSCFM	*enables loading of authorized client services*
BBG.AUTHMOD.BBGZSAFM.SAFCRED	*enables use of SAF authorized services*
BBG.AUTHMOD.BBGZSAFM.ZOSWLM	*enables use of WLM authorized services*
BBG.AUTHMOD.BBGZSAFM.TXRRS	*enables use of RRS services (transaction)*
BBG.AUTHMOD.BBGZSAFM.ZOSDUMP	*enables use of SVCDUMP services*
BBG.AUTHMOD.BBGZSAFM.LOCALCOM	***enables use of WOLA***
BBG.AUTHMOD.BBGZSAFM.WOLA	
BBG.AUTHMOD.BBGZSCFM.WOLA	
BBG.AUTHMOD.BBGZSAFM.PRODMGR	*enables use of IFAUSAGE services*
BBG.AUTHMOD.BBGZSAFM.ZOSAIO	*enables use TCP asynchronous I/O services*

*The batchManagerZos ID does **not** need READ to these profiles. It does, however, need READ to the CBIND to allow it to "register" into the server.*

To make things easy you could create a GROUP that has READ to these profiles, then simply connect server IDs to the group.

SSL and authentication? ...

SAF CBIND Profile and the WOLA Three Part Name



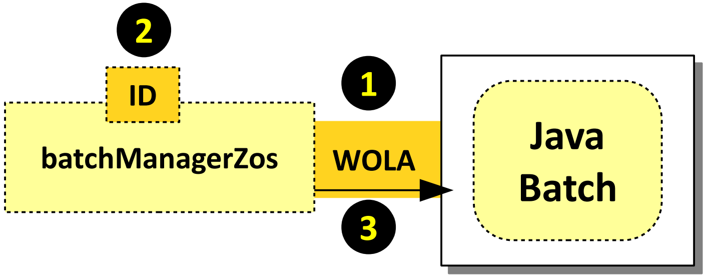
```
<zosLocalAdapters
  wolaGroup="LIBERTY"
  wolaName2="BATCH"
  wolaName3="MANAGER"/>
```

If the ID does not have READ to the CBIND, you will get this error:

```
ERROR: [jnu_wolaGetConnection(_CHAR12 *, int, _CHAR12 *)] rc=12  BBOA1CNG RC:12, RSN:14,
  registrationName:13b681ab batchManager, waitTime:30, connHandle:13b681b7
ERROR: [jnu_initAndOpenWolaConn(WolaConn *, char *)] rc=12  after jnu_openConn
ERROR: [jnu_newBatchWolaClient(cJSON *)] rc=12  after jnu_initAndOpenWolaConn
ERROR: [jnu_main(int, char **)] rc=255  exit: jnu_newBatchWolaClient returned NULL
```

SSL and authentication? ...

What About WOLA Data Encryption (SSL) and User Authentication?



1. **Cross-memory; no SSL needed**

   WOLA is a cross-memory technology; there is no network involved. The cross-memory communication between batchMangagerZos and the Liberty z/OS server can't be "sniffed." There is no need to encrypt.

2. **Already authenticated to z/OS**

   batchManagerZos is a z/OS-native utility that can only be invoked on z/OS. To invoke it you have already either logged into OMVS, or you opened a Telnet or SSH session to the mainframe. The z/OS system knows who you are. If it is invoked through submitted JCL, then the effective ID for that job submission applies, and that ID is either good or not good.

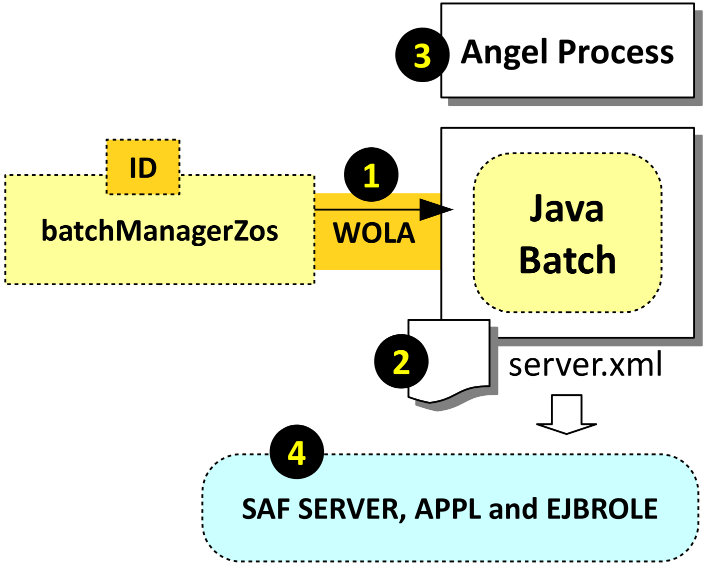
3. **WOLA connection requires READ to CBIND**

   The batchManagerZos ID will only be able to connect using WOLA if that ID has READ to the appropriate CBIND profile. That's further validation that the ID is good and permitted.

***Authorization* to use Java Batch is another issue … we look at that next**

Authorization? ...

What About WOLA Authorization?



1. **The ID of batchManagerZos will be asserted over WOLA**

   There is no ID/password specified on the batchManagerZos command line. The ID that will be asserted in will be the ID under which the batchManagerZos client is operating.

2. **If WOLA, then use SAF registry and authorization**

   Long story short: "basic" security does not have visibility to the ID that's asserted over WOLA. Since it can't know the ID, then it can't authorize the ID. That's why we used SPECIAL-SUBJECT=EVERYONE earlier.

   If we wish to get visibility to the asserted ID, we need to tell Liberty z/OS to use SAF for the registry. Then we get visibility, and from there we can check against SAF roles for authorization. Enabling SAF involves XML updates to the server.xml file.

3. **The Angel is needed for SAF use**

   The Angel is needed for WOLA, and it's also needed for SAF. The server ID needs READ to the **SAFCRED** profile as well as the WOLA profiles.

4. **SAF profiles in support of SAF registry and authorization**

   There's a handful of SAF updates needed to support role authorization.

XML updates and SAF ...

XML updates and SAF Updates for SAF Registry and SAF Authorization

```
<feature>zosSecurity-1.0</feature>

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

<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>~~

Add to the feature list

Add these three lines

The "profilePrefix" value may be any string you want. Showing default here. Key is be consistent with the EJBROLE prefix used (see below).

Leave the "basic" keystore (for now)

WOLA doesn't care, but others may be using it.

Remove basic registry and basic roles

Define APPL for profile prefix

Define EJBROLE (case sensitive)

RACLIST REFRESH as needed

```
RDEF APPL BBGZDFLT OWNER(SYS1) UACC(READ)

RDEF EJBROLE BBGZDFLT.com.ibm.ws.batch.batchAdmin OWNER(SYS1) UACC(NONE)
RDEF EJBROLE BBGZDFLT.com.ibm.ws.batch.batchSubmitter OWNER(SYS1) UACC(NONE)
RDEF EJBROLE BBGZDFLT.com.ibm.ws.batch.batchMonitor OWNER(SYS1) UACC(NONE)
```

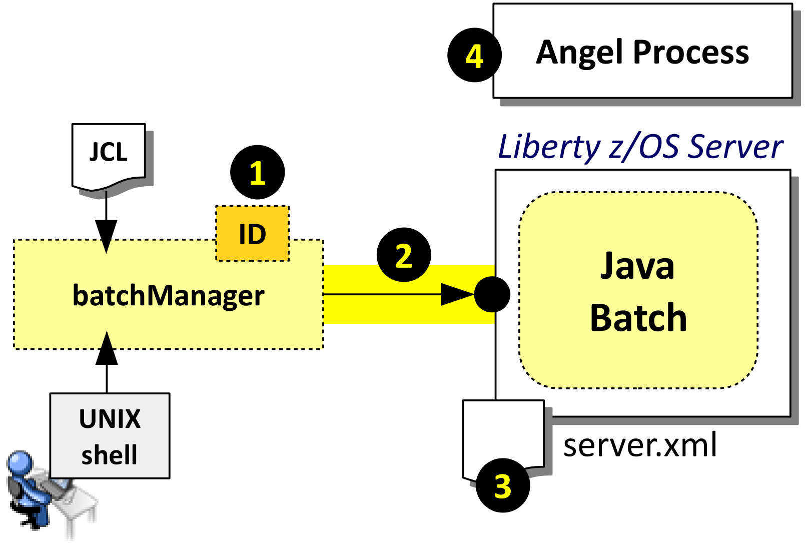
Grant ID of your batchManagerZos user to the profile for access desired

batchManager ...

Client Access

batchManager, including SAF SSL

batchManager and Things to Consider when Changing from "Basic" to SAF



What follows in this section is what would be required for the REST interface as well.

The batchManager client uses the REST interface of Liberty when it operates.

1. batchManager ID that flows

This is *not* based on the ID that invokes the client, it is based on either the user/password on the command line, or the client certificate that flows.

Note: if using SAF registry, the ID that flows must be defined in SAF.

2. Network connection and SSL

batchManager is a network-based client, and the connection will be redirected to SSL, which implies certificates. The "basic" security setup used a Liberty-generated certificate, but a more "real-world" scenario would use a generated server certificate signed by a well-known CA. Further, that certificate would be held in SAF keyrings.

3. Changes to server.xml

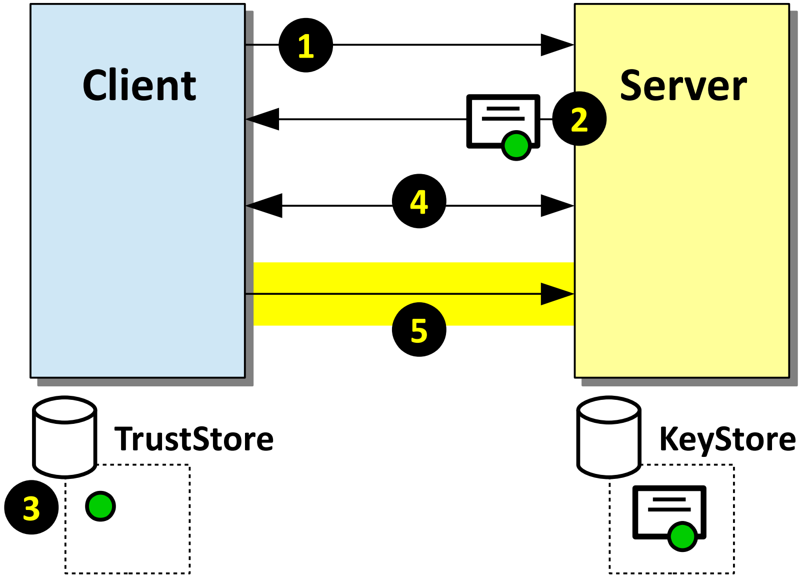
To move from "basic" security to SAF implies a few new features, the removal of the "basic" elements and the addition of elements for SAF registry, authorization, and SSL. It's similar to what we saw for batchManagerZos, but with some additional things for SSL and for the "unauthenticated userid."

4. Angel Process

The Angel is not needed for batchManager per se, or for SAF keyring SSL, but it *is* required if we use SAF for role authorization.

Detour: how SSL works ...

Brief Detour: How TLS (aka, "SSL") Works with Certificates and CA Signers



The Key and Trust stores may be file-based or SAF keyrings. We're getting ready to illustrate SAF keyrings.

1. Client attempts connection with Server

The client (batchManager in our scenario) initiates the conversation with a connection request to the server. The Java Batch interface in Liberty is protected, so it redirects to SSL

2. Server sends its certificate, signed by CA

The server retrieves its "server certificate" from its KeyStore and sends it to the client. The certificate has been signed by a "Certificate Authority" (CA). This is the process by which the client comes to "trust" the server is who the server is supposed to be.

3. Client compares server with CAs it knows about*

The client receives the server certificate and inspects it. It sees the certificate is signed by a CA, so it looks in its TrustStore to see if it has a matching "public key" for that CA. If it has a match, then it "trusts" the certificate and therefore the server. If no match, then it throws a security challenge. This is the browser challenge thing we've all seen.

4. They negotiate encryption keys

The seek to find the strongest encryption both support.

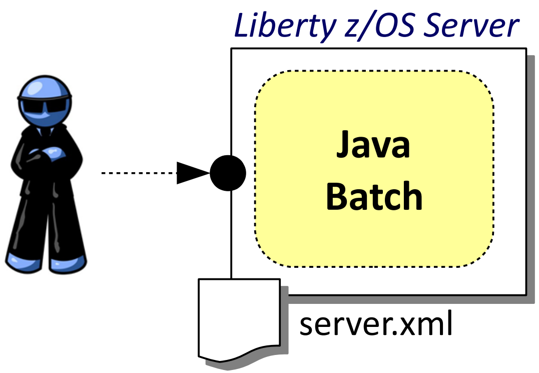
5. SSL connection established

When they agree, the SSL connection is established and the data is encrypted using the agreed-to keys.

* The `-trustSslCertificates` parameter on the batchManager command tells batchManager to trust the certificate regardless.

Detour: unauthenticated user ...

Another Brief Detour: The Unauthenticated User



When coming in on a network connection using SAF registry, there's a *brief* period of time between the user's entry and that user being properly authenticated.

During that brief period they are *unauthenticated*. They're on a thread in the server … and we don't yet know who they are.

We need to make sure they are *powerless* during that brief time.

```
ADDGROUP WSGUESTG OMVS(AUTOGID) OWNER(SYS1)
ADDUSER WSGUEST RESTRICTED DFLTGRP(WSGUESTG) OMVS(AUTOUID -
HOME(/u/wsguest) PROGRAM(/bin/sh)) -
NAME('UNAUTHENTICATED USER') NOPASSWORD NOOIDCARD

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

Separate group for the unauthenticated user

RESTRICTED means the user ID cannot be used to access protected resources they are not specifically authorized to access.

NOPASSWORD, NOOIDCARD means the ID can't be used to log onto the system.

Unauthenticated must have READ to the APPL that defines the profile prefix.

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

This is the server.xml we saw for batchManagerZos, but with the unauthenticated user specified. The default is "WSGUEST."

SAF work ...

SAF Work: Certificate Creation and Keyrings (batchManager client on z/OS)

```
RACDCERT CERTAUTH GENCERT SUBJECTSDN(CN('CA for Liberty') -
OU('LIBERTY')) WITHLABEL('LibertyCA.LIBERTY') TRUST -
SIZE(2048) NOTAFTER(DATE(2031/12/31))

RACDCERT ID(LIBSERV) 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(SERVERID) ADDRING(Keyring.SERVER)

RACDCERT ID(CLIENTID) ADDRING(Keyring.CLIENT)

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

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

PERMIT IRR.DIGTCERT.LISTRING -
CLASS(FACILITY) ID(<IDs>) ACCESS(READ)
PERMIT IRR.DIGTCERT.LIST -
CLASS(FACILITY) ID(<IDs>) ACCESS(READ)
```

Generate a Certificate Authority

You would not do this if you use a real, well-known Certificate Authority. This is a RACF-generated CA.

Generate the Server Cert and sign with CA

In the real-world you'd generate without a signer, then you'd have a real CA sign it and return along with their public key.

Create a keyring for the server ID

This will be used to hold the server certificate.

Create a keyring for the batchManager ID

This will be used to hold the CA public key.

Connect the server cert to the server keyring

The server will send this certificate to the client.

Connect the CA to the batchManager keyring

The client will use this to verify the server certificate.

Note: if batchManager is off-platform, you would export the CA public key and import it to the client's keyring or TrustStore there

For both IDs, permit keyring access

Access to keyrings may not be UACC(READ)

server.xml ...

Updates to server.xml to Remove Basic Security and Add SAF for SSL, Registry, Auth

```
<feature>zosSecurity-1.0</feature>
<feature>ssl-1.0</feature>

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

<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" />
```

(all the "basic" security elements removed)

Add to the feature list

The ssl-1.0 feature is needed for SAF SSL support.

Add these three lines

The "profilePrefix" value may be any string you want. Showing default here. Key is be consistent with the EJBROLE prefix used (see below).

Note we've specified the unauthenticated user.

Add the SSL Information

Note the keyring. That must match what's been defined for the ID under which the server runs. The password is literally "password" ... it's just a dummy string ... SAF keyrings don't have passwords.

Remove all the "basic" security elements

SERVER and EJBROLE ...

What About SERVER and EJBROLE?

`BBG.ANGEL.<angel_name>`	*enables access to a specific named Angel*
`BBG.ANGEL`	*enables access to the unnamed Angel process*
`BBG.AUTHMOD.BBGZSAFM`	*enables access to authorized services*
`BBG.AUTHMOD.BBGZSCFM`	*enables loading of authorized client services*
`BBG.AUTHMOD.BBGZSAFM.SAFCRED`	***enables use of SAF authorized services***
`BBG.AUTHMOD.BBGZSAFM.ZOSWLM`	*enables use of WLM authorized services*
`BBG.AUTHMOD.BBGZSAFM.TXRRS`	*enables use of RRS services (transaction)*
`BBG.AUTHMOD.BBGZSAFM.ZOSDUMP`	*enables use of SVCDUMP services*

`BBG.AUTHMOD.BBGZSAFM.LOCALCOM`
`BBG.AUTHMOD.BBGZSAFM.WOLA`
`BBG.AUTHMOD.BBGZSCFM.WOLA`
	enables use of WOLA

`BBG.AUTHMOD.BBGZSAFM.PRODMGR`	*enables use of IFAUSAGE services*
`BBG.AUTHMOD.BBGZSAFM.ZOSAIO`	*enables use TCP asynchronous I/O services*

SAF registry and EJBROLE authorization, must have Angel up, and server ID READ to the highlighted SERVER profiles at a minimum

Grant ID of your batchManager to the "allAuthenticatedUsers" EJBROLE

Grant ID of your batchManager user to the profile for access desired

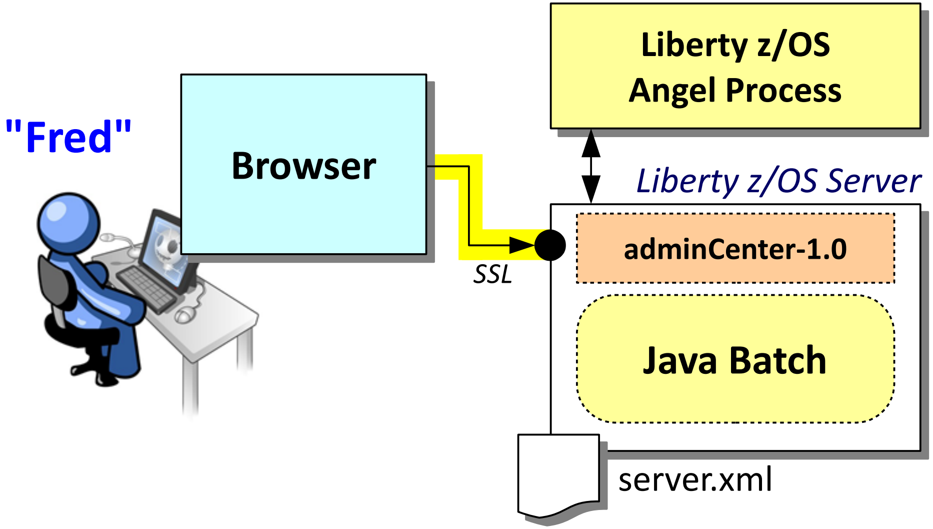
```
RDEF APPL BBGZDFLT OWNER(SYS1) UACC(READ)
RDEF EJBROLE BBGZDFLT.com.ibm.ws.management.security.resource.allAuthenticatedUsers OWNER(SYS1) UACC(NONE)
RDEF EJBROLE BBGZDFLT.com.ibm.ws.batch.batchAdmin OWNER(SYS1) UACC(NONE)
RDEF EJBROLE BBGZDFLT.com.ibm.ws.batch.batchSubmitter OWNER(SYS1) UACC(NONE)
RDEF EJBROLE BBGZDFLT.com.ibm.ws.batch.batchMonitor OWNER(SYS1) UACC(NONE)
```

AdminCenter ...

Client Access

adminCenter (once SAF authentication/authorization/SSL is in place)

Assuming the Setup From the Previous Section … This is Relatively Easy



All the infrastructure to support SAF registry, SAF authorization, and SAF SSL is in place

The difference is that the AdminCenter uses a different EJBROLE from the Java Batch support, so that AdminCenter EJBROLE needs to be created and the ID of the AdminCenter user needs to be given READ to the profile.

```
RDEFINE EJBROLE BBGZDFLT.com.ibm.ws.management.security.resource.Administrator UACC(NONE)

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

Summary …

Summary

Summary

Security is a big topic

The Java Batch function is mostly just relying on the security support provided by Liberty

Liberty z/OS has SAF as an additional security component

Key things: file security, z/OS authorized service access (Angel), authentication, authorization, and data link encryption