WebSphere Application Server for z/OS Version 8.5.5.2

Liberty Profile Optimized Local Adapters

Quick Start Guide

Version Date: September 19, 2014

See "Document Change History" on page 29 for a description of the changes in this version of the document

WP101490 at
ibm.com/support/techdocs
© IBM Corporation 2014

Many thanks to **Tim Kaczysnski** and **David Follis** of the IBM WebSphere z/OS Development Lab.

The IBM Gaithersburg WAS z/OS support team consists of **Mike Cox**, **Mike Kearney** and **Don Bagwell**

Table of Contents

Overview

WOLA – WebSphere Optimized Local Adapters – became available in Liberty Profile z/OS with the release of Version 8.5.5.2. This document will guide you through setting up and validating WOLA with Liberty Profile z/OS.

WOLA and Liberty Profile z/OS

WOLA has been around for several years now as part of the function offered with WebSphere Application Server z/OS. WOLA has shown itself to be an excellent solution for use-cases where very low-latency is desired between WAS z/OS and regions on z/OS that wish to communicate with WAS.

WOLA for Liberty Profile z/OS is very similar to WOLA for full-function WAS z/OS in concept, but has some differences. For a summary of that, see "Summary of Liberty WOLA compared to full-function WAS WOLA" on page 27.

Important prerequisites

Liberty Profile z/OS has a requirement for z/OS 1.11 or later.

If you are using z/OS 1.11, 1.12 or 1.13[1], there is an important APAR you must make sure you have on your system: APAR 0A3905. See the following website for more on that APAR:

`http://www.ibm.com/support/docview.wss?uid=isg1OA39035`

In addition, WebSphere Liberty Profile z/OS 8.5.5.2 or higher is required, along with two interim fixes (iFixes)[2]. See "Install 8.5.5.2 and required iFixes" on page 5 for more on those iFixes.

How this document is structured

This document will have a combination of information and 'to-do' activities. You can spot the to-do activities because they will have a checkbox next to them:

- ☐ Indicates some action you are to take
  - ○ A sub-action to take under the square checkbox activity

Informational text will not have checkbox or check-circles.

The WP101490 Techdoc at ibm.com/support/techdocs

That Techdoc is the repository for all things WOLA-related coming out of the Washington Systems Center. The URL is:

`http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101490`

With the addition of this document, that Techdoc now has two "Quick Start Guides" – one for full-function WAS z/OS, and one for Liberty Profile z/OS. The Techdoc is arranged by section, and the header for the section will indicate what documents are contained in the section.

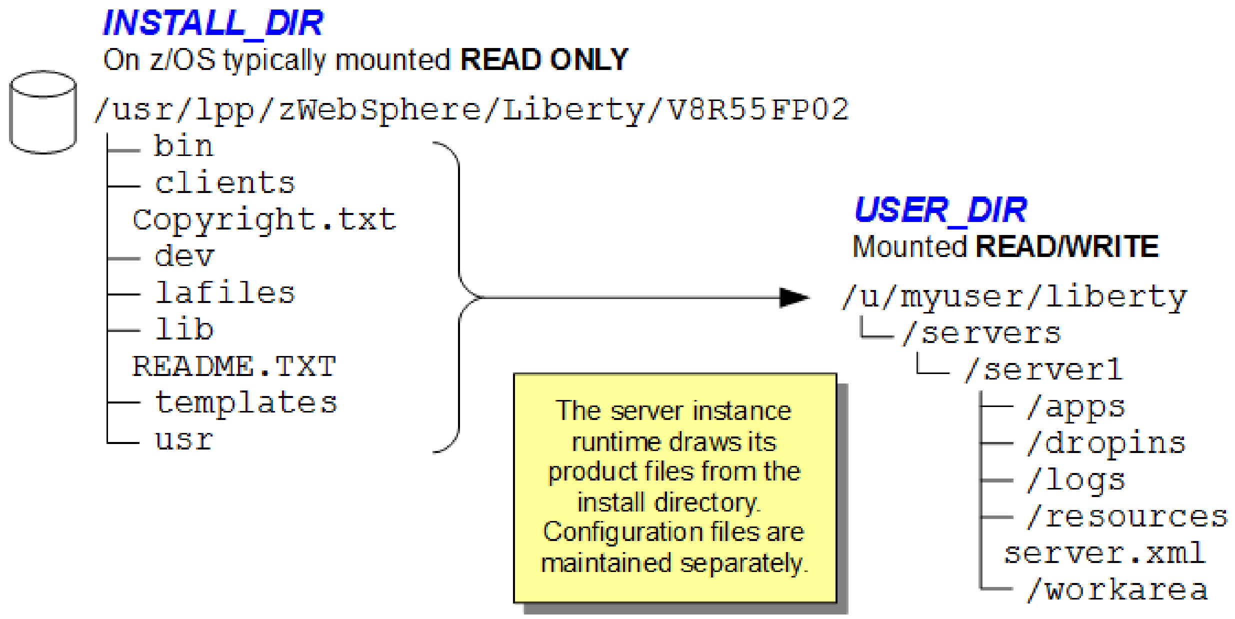
1 The APAR fix is included in z/OS 2.1 and higher.
2 If at the time of this reading 8.5.5.3 is available, the iFixes should be included in that level of WebSphere Liberty Profile z/OS.

Preliminary Setup and Feature Validation

The local adapter feature is not automatically available to WebSphere Liberty Profile z/OS – it needs to be enabled. It's a simple process which we'll cover in this section.

A little background in Liberty Profile for z/OS

When discussing Liberty Profile, it's necessary to keep separate the notion of the *install file system* from the *user directory*. The install file system is where the product files reside; the user directory is where a Liberty Profile server's[3] configuration and runtime files reside:


INSTALL_DIR
On z/OS typically mounted READ ONLY
/usr/lpp/zWebSphere/Liberty/V8R55FP02
bin
clients
Copyright.txt
dev
lafiles
lib
README.TXT
templates
usr
USER_DIR
Mounted READ/WRITE
/u/myuser/liberty
/servers
/server1
/apps
/dropins
/logs
/resources
server.xml
/workarea
The server instance runtime draws its product files from the install directory. Configuration files are maintained separately.

A few notes on this:

- The WP102110 Techdoc[4] covers Liberty Profile for z/OS, and provides a separate “Quick Start Guide” on Liberty Profile z/OS. If you're interested in digging deeper into Liberty Profile z/OS in general, consult that Techdoc.
- Liberty Profile provides a way to create an archive (for example, a PAX file) that contains *both* the configuration files *and* the install files in one portable file[5]. In this document we're not talking about that. In this document we're illustrating the install directory and user directory as separate.

The reason we highlight this is because as part of the initial setup work to enable WOLA we will have you work against the `INSTALL_DIR` files. Then later we will have you work against the `USER_DIR` files. Keeping them separate in your mind helps.

Install 8.5.5.2 and required iFixes[6]

WOLA is provided initially in WebSphere Application Server z/OS Liberty 8.5.5.2. To fully support using WOLA, two interim fixes (“iFixes”) are required – PI18279 and PI18379. From an IBM Installation Manager perspective, they are known as:

```
8.5.5.2-WS-WLP-OS390-IFPI18279_8.5.5002.20140522_1502
8.5.5.2-WS-WLP-OS390-IFPI18379_8.5.5002.20140522_1030
```

3 Or *servers*, plural … it's possible to have multiple server instances defined under a single user directory.
4 http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP102110
5 It's a handy way to distribute a complete Liberty Profile runtime to other servers, or in the case of z/OS, other LPARs.
6 Or, if at the time of reading 8.5.5.3 is available, then that will include the iFixes.

Do the following:

- ☐ Using IBM Installation Manager[7], install WebSphere Liberty Profile z/OS 8.5.5.2
- ☐ Using IBM Installation Manager, install the two iFixes listed earlier
- ☐ If your local conventions install first into a `/Service` location, copy the file system and mount READ/WRITE at the location that will be your `INSTALL_DIR`. If you install directly to your `INSTALL_DIR`, then make sure that's mounted READ/WRITE.

At this point you should have WebSphere Liberty z/OS 8.5.5.2 + the two iFixes.

Plan the Liberty Profile server environment

To make the configuration process easier, a little planning up-front will help. Come up with values for the following:

☐	ID under which Liberty Profile will run	*liberty_id =*
☐	ID under which the Angel process will run	*angel_id =*
☐	Group to which Liberty ID and Angel ID will connect	*lib_group =*
☐	UNIX path where Liberty configuration will reside	*WLP_USER_DIR=*
☐	UNIX path to valid 64-bit Java SDK	*JAVA_HOME=*

Notes:

- The IDs and Group may be new, or you may re-use existing.
- The UNIX path to the Liberty Profile configuration may reside anywhere you like. The ID you choose to run Liberty under must have WRITE to that directory.
- The 64-bit Java SDK may be IBM Java SDK6 or SDK7. It must be 64-bit.

Install features using featureManager

There's some work that needs to be done against the `INSTALL_DIR`, which is why we had you mount the file system READ/WRITE[8]. That work involves installing the WOLA feature into your installation of Liberty Profile.

Do the following:

- ☐ Take a look at the ownership of the directories and files in your `INSTALL_DIR`. You will need an ID that has WRITE permission within that directory structure. Determine which ID you will use.
- ☐ Log into OMVS, Telnet or SSH session using the ID determined in the previous step.
- ☐ Go to the `/`*`INSTALL_DIR`*`/bin` directory.
- ☐ Enter the command echo **`$JAVA_HOME`**. Do you get a response to a valid 64-bit SDK? If not, then enter the command **`export JAVA_HOME=`*`<SDK_path>`*** to point to a 64-bit SDK installation on your system.
- ☐ Invoke the `featureManager` script with this command:

```
./featureManager featureList /<path>/features.txt
```

- ☐ The file produced will be an ASCII XML file, and it will contain a listing of all the features for that installation instance of Liberty. Browse and look for the following string:

```
<feature name="zosLocalAdapters-1.0">
```

7 See Techdoc `http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP102014` for more on IM and z/OS.
8 Once the setup work is complete, the `INSTALL_DIR` may be mounted READ ONLY.

> **Note:** It should not be present. That indicates the features needs to be installed. That is done with the featureManager script as well.

☐ Install the z/OS Local Adapters feature with the following command:

```
featureManager install zosLocalAdapters-1.0 --when-file-exists=ignore
```

Then:

- ○ Enter **1** to accept the license
- ○ Look for the `CWWKF1017I` message indicating successful install.

☐ Run the command to list out the installed features, this time providing a new file name:

```
./featureManager featureList /<path>/features2.txt
```

☐ Browse that file and look for the following string:

```
<feature name="zosLocalAdapters-1.0">
```

This time you should see them:

```
<feature name="zosLocalAdapters-1.0">
    <symbolicName>com.ibm.websphere.appserver.zosLocalAdapters-1.0</symbolicName>
    <displayName>z/OS WebSphere Optimized Local Adapter Channel</displayName>
```

> Note: That was simply a validation of the feature install. It wasn't strictly required.

☐ Re-mount the `INSTALL_DIR` file system so it is READ ONLY.

☐ Log out of the OMVS or Telnet session you were in.

Create server

In this section we will have you create and start a Liberty Profile server in which WOLA will be configured and operated.

Do the following:

☐ Open an OMVS, Telnet or SSH session to your system and log on with any ID of your choosing.

> Note: We'll start in the UNIX shell, and later go to z/OS started tasks. For this section any ID will work. When we move to using z/OS started tasks we'll have you `chown` the created directories and files to the ID you determined on page 6.

☐ Change directories to the `/INSTALL_DIR/bin` directory.

☐ Enter the following command:

```
export JAVA_HOME=<Path_to_Java>
```

where `<Path_to_Java>` is the path to the 64-bit installation on your system.

☐ Verify that by entering the command **`echo $JAVA_HOME`**. It should return the value you just entered.

☐ Enter the following command:

```
export WLP_USER_DIR=<User_Directory>
```

where `<User_Directory>` is the directory you chose as the location where the Liberty Profile server would be created.

☐ Verify that by entering the command echo **`$WLP_USER_DIR`**. It should return the value you just entered.

☐ Enter the following command to create the server:

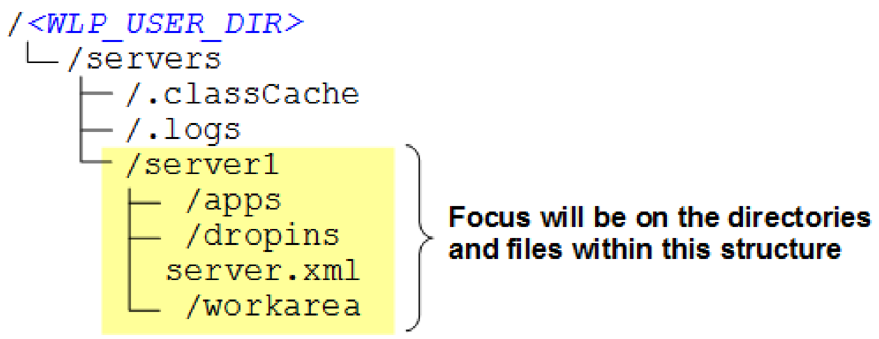
```
./server create server1
```

You should see the following response:

```
Server server1 created.
```

Note: The server configuration was created under the directory specified by the UNIX `WLP_USER_DIR` environment variable. That was the reason for setting that variable.

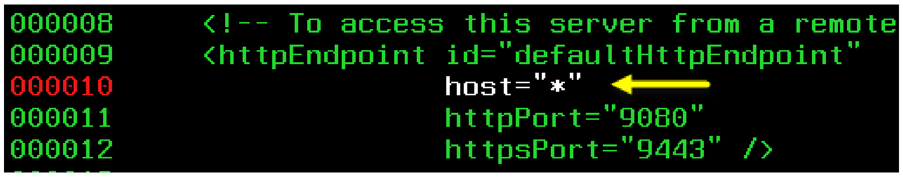
- ☐ Change directories to `WLP_USER_DIR` (whatever you set that value to be). You should see a structure something like this:



```
/<WLP_USER_DIR>
 └─/servers
     ├─/.classCache
     ├─/.logs
     └─/server1
         ├─ /apps
         ├─ /dropins
         │ server.xml
         └─ /workarea
```

Focus will be on the directories and files within this structure

- ☐ Edit the file `server.xml`, which is in ASCII[9]. Add `host="*"` to the file as shown here:



```
        <!-- To access this server from a remote
        <httpEndpoint id="defaultHttpEndpoint"
                      host="*"
                      httpPort="9080"
                      httpsPort="9443" />
```

Change the port values as well if you think the default values will conflict.

Save the file.

- ☐ From the /*INSTALL_DIR*/bin directory, enter the following command:

`./server start server1`

You should get an indication of the server start with process ID:

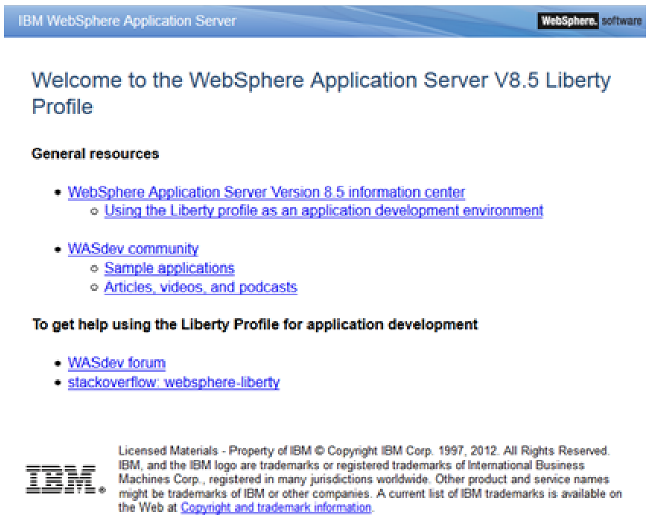
```
Server server1 started with process ID 50595976.
```

- ☐ From a browser, enter the following URL:

`http://`*`<host>`*`:9080/`

where *`<host>`* is the host on which the Liberty Profile server is running. You should get a page that looks like this:

9 However, it is "tagged" as ASCII and z/OS tools such as OEDIT should recognize it and auto-convert if your UNIX shell environment has `_BPXK_AUTOCVT=ON` set. See Liberty Profile Quick Start Guide for more – `ibm.com/support/techdocs`, search for WP102110.


IBM WebSphere Application Server

Welcome to the WebSphere Application Server V8.5 Liberty Profile

General resources

- WebSphere Application Server Version 8.5 information center
  - Using the Liberty profile as an application development environment
- WASdev community
  - Sample applications
  - Articles, videos, and podcasts

To get help using the Liberty Profile for application development

- WASdev forum
- stackoverflow: websphere-liberty

Licensed Materials - Property of IBM © Copyright IBM Corp. 1997, 2012. All Rights Reserved. IBM, and the IBM logo are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information.

☐ From the `/INSTALL_DIR/bin` directory, enter the following command:

```
./server stop server1
```

You should get an indication the server has been stopped.

The basics are now in place, but you haven't really *exercised* the z/OS Local Adapters feature. That's coming up.

Setup Liberty Profile server and Angel process to run as a z/OS started tasks

Note: You will require an Angel Process[10] to use the Local Adapters support[11], and that must run as a started task.

Create the IDs and Group

Perform these steps if you wish to create new IDs and a new group. If you are re-using existing IDs and group, then skip to “Copy JCL start procedures to proclib and modify” on page 10.

Do the following:

☐ Work with your security administrator and create the following group and IDs:

```
ADDGROUP lib_group OMVS(GID(gid)) OWNER(SYS1)

ADDUSER angel_id DFLTGRP(lib_group) OMVS(UID(uid)
      HOME(angel_home) PROGRAM(/bin/sh)) NAME('Liberty Angel')
                              OWNER(lib_group) NOPASSWORD NOOIDCARD

ADDUSER liberty_id DFLTGRP(lib_group) OMVS(UID(uid)
     HOME(server_home) PROGRAM(/bin/sh)) NAME('Liberty Server')
                              OWNER(lib_group) NOPASSWORD NOOIDCARD
```

Where the highlighted values are substituted with the proper values.

Change Liberty Profile configuration directory and file ownerships

Earlier you created a Liberty Profile server from the UNIX shell using some ID you logged on with. Now is the time to change the ownership of the created directories and files to the ID you wish the Liberty Profile server to run under when operating as a started task.

Do the following:

10 The Angel Process is a started task that provides an anchor point for access to z/OS authorized services, such as the local adapters support. It is *not* a server, per se: it has no TCP ports, it has no configuration file, and it uses virtually no CPU once started.

11 See “Angel process and the new WOLA support” on page 28 for more on the Angel and the new WOLA support.

- ☐ Log into the UNIX shell with an ID that has the authority to issue the `chown` command.
- ☐ Change directories to your *`WLP_USR_DIR`* directory.
- ☐ Issue the following command:

  `chown -R` ***`liberty_id`*****`:`*****`lib_group`*** **`servers`**

  where *`liberty_ID`* and *`lib_group`* are the ID and group values you decided upon back on page 6, and created in the previous section if you are going with new IDs and groups for this exercise.
- ☐ Verify the owner and group has been changed for the directory `servers` and the directories and files under it.

Copy JCL start procedures to proclib and modify

Do the following:

- ☐ Locate the `bbgzsrv.jcl` file, which is in the following directory:

  `/`*`INSTALL_DIR`*`/templates/zos/procs`
- ☐ Copy that file to your procedure library, for example from ISPF Option 6:

  `OGET '/`*`INSTALL_DIR`*`/templates/zos/procs/bbgzsrv.jcl' 'SYS1.PROCLIB(BBGZSRV)'`
- ☐ Edit the JCL and change the following two lines as instructed:

```
000014 //*-----------------------------------
000015 //   SET INSTDIR='/u/MSTONE1/wlp'      1
000016 //   SET USERDIR='/u/MSTONE1/wlp/usr'  2
000017 //*-----------------------------------
```

  Notes:

  1. Set `INSTDIR` to your *`INSTALL_DIR`* location. Be sure to enclose the path in single quotes as shown above.
  2. Set `USERDIR` to your *`WLP_USER_DIR`* location. Be sure to enclose the path in single quotes as shown above.
- ☐ Save the file
- ☐ Locate the `bbgzangl.jcl` file, which is in the following directory:

  `/`*`INSTALL_DIR`*`/templates/zos/procs`
- ☐ Copy that file to your procedure library, for example from ISPF Option 6:

  `OGET '/`*`INSTALL_DIR`*`/templates/zos/procs/bbgzangl.jcl' 'SYS1.PROCLIB(BBGZANGL)'`
- ☐ Edit the JCL and change the following one line as instructed:

```
000002 //*-----------------------------------
000003 //   SET ROOT='/u/MSTONE1/wlp'      1
000004 //*-----------------------------------
```

  Notes:

  1. Set `ROOT` to your *`INSTALL_DIR`* location. Be sure to enclose the path in single quotes as shown above.
- ☐ Save the file

Create `server.env` and provide `JAVA_HOME` variable to environment

When you started the server from the UNIX shell, it had knowledge of the `JAVA_HOME` because you set the UNIX environment variable. When run as a started task, you need to provide the `JAVA_HOME` value in a different way – with a `server.env` file.

Do the following:

- ☐ Create[12] the following file:

  `/WLP_USER_DIR/servers/server1/server.env`

  Note: this is the same directory where the `server.xml` file is located.

- ☐ Edit that file[13] and enter one line:

  `JAVA_HOME=<Path_to_Java>`

  Note: this is the same location you captured earlier on page 6.

- ☐ Save the file.
- ☐ Change the ownership of the file to the Liberty Profile ID and group.

Create SAF STARTED profiles for Angel process and Server

To get the IDs you planned earlier (page 6) to be assigned to these started tasks, you will need to create two SAF `STARTED` profiles[14].

Do the following:

- ☐ Work with your security administrator and issue the following commands[15]:

```
RDEF STARTED BBGZANGL.* UACC(NONE) STDATA(USER(angel_id)
              GROUP(lib_group) PRIVILEGED(NO) TRUSTED(NO) TRACE(YES))
RDEF STARTED BBGZSRV.* UACC(NONE) STDATA(USER(liberty_id)
              GROUP(lib_group) PRIVILEGED(NO) TRUSTED(NO) TRACE(YES))
SETROPTS RACLIST(STARTED) REFRESH
```

Start the Angel process as a started task

The Angle process is very simple and lightweight – it has no configuration, no TCP ports, and it consumes almost no CPU once started. Do the following:

- ☐ From the z/OS operator console, issue the following command:

  `S BBGZANGL`

  You should see the Angel come active very quickly:

```
S BBGZANGL
IRR812I PROFILE BBGZANGL.* (G) IN THE STARTED CLASS WAS USED
                       TO START BBGZANGL WITH JOBNAME BBGZANGL.
$HASP100 BBGZANGL ON STCINRDR
IEF695I START BBGZANGL WITH JOBNAME BBGZANGL IS ASSIGNED TO
                                 USER angel_id, GROUP lib_group
$HASP373 BBGZANGL STARTED
CWWKB0056I INITIALIZATION COMPLETE FOR ANGEL
```

Start server as a started task

You're ready to run the server as a started task – you've created the JCL proc, created a `STARTED` profile, and created a `server.env` with `JAVA_HOME`.

Do the following:

- ☐ Open a 'System Command Extension' window[16].

12 You can do this with the UNIX `touch server.env` command, or using ISHELL, or by copying the `server.xml` file to `server.env`.
13 For a started task, the content should be in EBCDIC.
14 Early in this document we have you create the SAF profiles as you need them. If you would prefer to see the complete set of profiles needed and create them ahead of time, see “SAF definitions” on page 24.
15 Here we are showing IBM RACF commands. If you have another vendor's security product, use the commands appropriate for that vendor's product.
16 A single slash will open the command extension. To specify your server, you provide `PARMS=` on the `START` command. However, your server name is *lowercase*, and you must preserve the case on the `START` command. The command extension window will preserve the case. A simple `/START` from the operator console will not.

- ☐ Enter the following command:

  `START BBGZSRV,PARMS='server1'`

  You should see the server come active:

  ```
  JOBNAME  StepName ProcStep JobID
  BBGZSRV  BBGZSRV  STEP1    STC04122
  ```

- ☐ Go back and edit the `server.xml` file and *add* the following line:

  `<feature>zosLocalAdapters-1.0</feature>`

  In the `<featureManager>` section near the top of the file. Save the file.

- ☐ Because Liberty Profile is dynamic[17], the changes you made to the `server.xml` are automatically detected and implemented. Look at the log file[18]:

  `/WLP_USER_DIR/servers/server1/logs/messages.log`

  In particular, look for the following messages which will be near the bottom:

  ```
   :
  J2CA7018I: Installing resource adapter ola.
  J2CA7001I: Resource adapter ola installed in 0.496 seconds.
   :
  ```

  The first two messages (`J2CA*`) indicate the Optimized Local Adapter JCA resource adapter has been installed. That's a preliminary validation of the local adapters feature being operative. We'll validate it more fully under "Enabling and Validating WOLA" starting on page 13.

- ☐ Enter the MVS command:

  `P BBGZSRV`

  The started task should stop.

Status at this point

You have constructed the essential framework to begin testing and validating WOLA.

17 Not all changes are dynamic, but many are.
18 The `messages.log` file is in ASCII.

Enabling and Validating WOLA

In this section we will take you through the steps to set up and validate WOLA working with first a COBOL batch program, then a CICS region.

Enable WOLA

You did some of this earlier when you used the `featureManager` shell script to install the local adapters function (starting on page 6). You have the Angel process running (page 11). What's left to do is update the `server.xml` with a few things, add some SAF profiles and perform some validation.

Configuration updates to support WOLA

Do the following:

☐ Verify you have the `zosLocalAdapters-1.0` feature configured in your `server.xml`. Your copy of `server.xml` should look like this right now:

```
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>jsp-2.2</feature>
        <feature>zosLocalAdapters-1.0</feature>
    </featureManager>

    <!-- To access this server from a remote client
    <httpEndpoint id="defaultHttpEndpoint"
                  host="*"
                  httpPort="9080"
                  httpsPort="9443" />

</server>
```

☐ Add the following (see notes that follow):

```
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>jsp-2.2</feature>
        <feature>zosLocalAdapters-1.0</feature>
    </featureManager>

    <zosLocalAdapters
        wolaGroup="GROUP"
        wolaName2="NAME2"
        wolaName3="NAME3" />

    <connectionFactory id="wolaCF" jndiName="eis/ola">
       <properties.ola />
    </connectionFactory>

    <!-- To access this server from a remote client
    <httpEndpoint id="defaultHttpEndpoint"
                  host="*"
                  httpPort="9080"
                  httpsPort="9443" />

</server>
```

Notes:

- WOLA requires a three-part name to be used when external address spaces (CICS regions, batch programs) register into the Liberty Profile server. In full-function WAS z/OS that three part name is the cell, node and server short names. Liberty Profile has no cell or node constructs. To keep the WOLA programming model consistent between full-function WAS z/OS and Liberty Profile z/OS the three-part name model was carried into Liberty Profile. The three-part name is accomplished with the XML you see in the diagram.

> **Note:** The *values* are arbitrary. Each name component is limited to 8 characters alpha-numeric with no blank spaces. The names are case sensitive, so if you have mixed case in the `server.xml` then the outside program seeking to register into the Liberty Server would need to match the case.
>
> ***Recommendation***: make the values upper-case. It makes things easier when you create the `CBIND` profile (see page 18 for that).
>
> The three-part name used by a Liberty Profile z/OS server instance *must be unique* on the z/OS LPAR.

- The `connectionFactory` XML defines the WOLA JCA resource adapter that is used for communication outbound from Liberty Profile to the external address space.

☐ Save the file.

SAF SERVER updates to support WOLA

The WOLA support requires the Angel process. But having the Angel running is not enough. You must also create a set of SAF `SERVER` profiles to grant your Liberty Profile server authority to use the services of the Angel[19].

Do the following:

☐ Work with your security administrator to see if the following SAF `SERVER` profiles already exist, and if not create them. Use the following commands:

- ○ **`RDEF SERVER BBG.ANGEL UACC(NONE) OWNER(SYS1)`**
- ○ **`RDEF SERVER BBG.AUTHMOD.BBGZSAFM UACC(NONE) OWNER(SYS1)`**
- ○ **`RDEF SERVER BBG.AUTHMOD.BBGZSAFM.WOLA UACC(NONE) OWNER(SYS1)`**
- ○ **`RDEF SERVER BBG.AUTHMOD.BBGZSAFM.LOCALCOM UACC(NONE) OWNER(SYS1)`**
- ○ **`RDEF SERVER BBG.AUTHMOD.BBGZSCFM UACC(NONE) OWNER(SYS1)`**
- ○ **`RDEF SERVER BBG.AUTHMOD.BBGZSCFM.WOLA UACC(NONE) OWNER(SYS1)`**
- ○ **`SETROPTS RACLIST(SERVER) REFRESH`**

> **Note:** If you've worked with Liberty Profile z/OS in the past, the first two may already exist[20]. You do not need to recreate them. The last four are new to the WOLA support introduced in 8.5.5.2. Those will very likely need to be created.

☐ When you have all six of `SERVER` profiles verified or created, then grant your Liberty Profile server ID[21] `READ` access to each. Issue the following commands:

- ○ **PERMIT BBG.ANGEL CLASS(SERVER) ACCESS(READ) ID(*liberty_id*)**
- ○ **PERMIT BBG.AUTHMOD.BBGZSAFM CLASS(SERVER) ACCESS(READ) ID(*liberty_id*)**
- ○ **PERMIT BBG.AUTHMOD.BBGZSAFM.WOLA CLASS(SERVER) ACCESS(READ) ID(*liberty_id*)**
- ○ **PERMIT BBG.AUTHMOD.BBGZSAFM.LOCALCOM CLASS(SERVER) ACCESS(READ) ID(*liberty_id*)**
- ○ **PERMIT BBG.AUTHMOD.BBGZSCFM CLASS(SERVER) ACCESS(READ) ID(*liberty_id*)**
- ○ **PERMIT BBG.AUTHMOD.BBGZSCFM.WOLA CLASS(SERVER) ACCESS(READ) ID(*liberty_id*)**
- ○ **SETROPTS RACLIST(SERVER) REFRESH**

19 See "SAF definitions" on page 24 for a complete list of all the SAF profiles.
20 And in fact you may have four other `AUTHMOD` profiles: `SAFCRED`, `TXRRS`, `ZOSDUMP`, `ZOSWLM`. Those were part of the original Liberty Profile z/OS support introduced with 8.5.0.0.
21 See page 6 for the value you planned; see page 11 for the SAF STARTED profile that assigns the ID to the started task.

Start server and check for key validation messages in log

Do the following:

- ☐ From *the command extension*, issue the START command for the server:

  `S BBGZSRV,PARMS='server1'`

- ☐ Go the `/`*`WLP_USER_DIR`*`/servers/server1/logs` directory and look at the contents of the `messages.log` file. Specifically, look for the following WOLA success indicators:

```
TRAS0018I: The trace state has been changed. The new trace state is *=info.
CWWKE0001I: The server server1 has been launched.
CWWKB0103I: Authorized service group LOCALCOM is available.  1
CWWKB0103I: Authorized service group WOLA is available.  2
CWWKB0104I: Authorized service group PRODMGR is not available.  3
CWWKB0104I: Authorized service group SAFCRED is not available.
CWWKB0104I: Authorized service group TXRRS is not available.
CWWKB0104I: Authorized service group ZOSDUMP is not available.
CWWKB0104I: Authorized service group ZOSWLM is not available.
 :
CWWKB0501I: The WebSphere Optimized Local Adapter channel registered with
    the Liberty profile server using the following name: GROUP NAME2 NAME3 4
 :
J2CA7001I: Resource adapter ola installed in 0.327 seconds.  5
```

Notes:

1. The `LOCALCOM` service group being available is a key indicator that the Angel is running *and* your Liberty Server ID has `READ` to that `SERVER` profile.
2. The `WOLA` service group being available is another key indicator that the setup work was successfully completed.
3. The next file services groups being *not available* is okay for now[22].
4. This message is showing the successful use of the WOLA three-part name values you provided in the `server.xml` file. This message indicates that an outside address space (we'll show using COBOL batch in a little bit) may register into the Liberty Profile server using this three-part name on the `BBOA1REG` API.
5. The WOLA JCA resource adapter is available.

The key messages we wanted to see here were 1, 2 and 4.

Sample Java application to validate WOLA

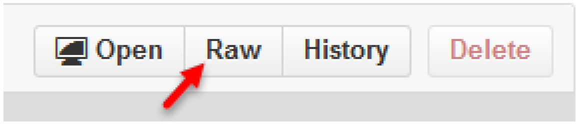
For validation of WOLA we're going to use a sample Java application provided with the Liberty Profile z/OS WOLA support.

Do the following:

- ☐ Go to the following URL:

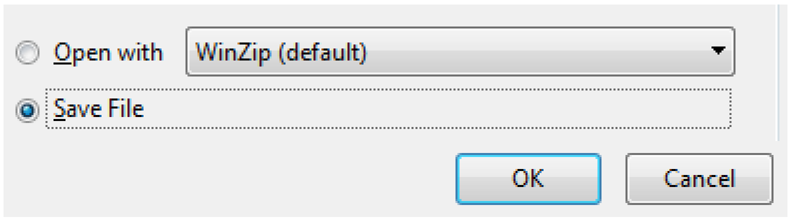
  `https://github.com/WASdev/sample.wola/blob/master/OLASampleLiberty.ear`

- ☐ Click on the “raw” tab:



22 They're not available because the Liberty Profile server ID has not been granted `READ` to the respective `SERVER` profiles. If you see some of these being available, that's likely due to previous Liberty Profile z/OS work on this LPAR being done with those SAF profiles being set up with your server ID being granted `READ`.

- ☐ When prompted, choose to save the file:



- ☐ FTP the file *in binary* up to your system and place it in the `/dropins` directory[23] with permissions that provides the Liberty ID at least READ to the file[24]:

```
/<WLP_USER_DIR>
 └─/servers
    ├─/.classCache
    ├─/.logs
    └─/server1
       ├─ /apps
       ├─ /dropins
       server.xml
       └─ /workarea
```

The `/dropins` directory is located just under `/server1` directory

- ☐ Now look in the `messages.log` file, specifically for the messages indicating the application is being started, and it has completed starting:

```
CWWKZ0018I: Starting application OLASampleLiberty.
 :
CWWKZ0001I: Application OLASampleLiberty started in 0.104 seconds.
```

Use sample COBOL batch to register into Liberty using WOLA

The Liberty Profile WOLA support is up and ready to go. Now all you need is something to talk to over WOLA. In this section we'll illustrating using a sample COBOL batch program. We start with that because it's relatively simple to set up and use. Later, on page 20, we'll show how to set up and use WOLA with CICS.

Copy WOLA modules from file system to load library data set

The WOLA modules come with Liberty Profile z/OS 8.5.5.2. But for use by external address spaces such as batch programs or CICS regions those modules must be copied out to a load library. The load library may then be pointed to with `STEPLIB` or `DFHRPL`.

Do the following:

- ☐ Allocate a PDSE[25] with 30 tracks:

```
Average record unit
Primary quantity  . . 30
Secondary quantity    2
Directory blocks  . . 15
Record format . . . . U
Record length . . . . 0
Block size  . . . . . 32760
Data set name type    LIBRARY
```

- ☐ Open a Telnet session to your system and change directories to the following location:

23 The `/dropins` directory is created when the server is started for the first time. Liberty Profile monitors the `/dropins` directory and will dynamically load any application files placed in the directory.
24 Liberty Profile will throw an FFDC event if it doesn't have at least READ to the application file.
25 A good practice is to provide the fixpack level of the modules in the name qualifier, for example: `LIBERTY.V855FP02.LOADLIB`. That helps reduce questions about what level of WOLA code is resident in the PDSE.

/*INSTALL_DIR*/clients/zos

There you will find the WOLA modules:

```
USER:/INSTALL_DIR/clients/zos-> ls
bboa1cng  bboa1inv  bboa1rcs  bboa1srq  bboa1urg  bboacsrv
bboa1cnr  bboa1rca  bboa1reg  bboa1srv  bboaclnk  bboatrue
bboa1get  bboa1rcl  bboa1srp  bboa1srx  bboacntl
```

☐ From the Telnet session, issue the following command to copy the modules from the file system to the PDSE you just allocated:

cp -Xv ./* "//'*<data.set>*'"

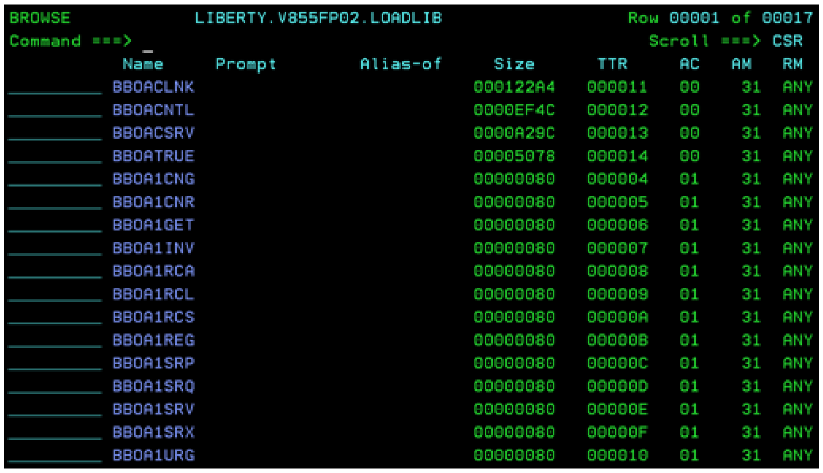
Where *<data.set>* is the target PDSE. For example:

```
cp -Xv ./* "//'LIBERTY.V855FP02.LOADLIB'"
```

You should see something like this:

```
USER:/INSTALL_DIR/clients/zos-> cp -Xv ./* "//'LIBERTY.V855FP02.LOADLIB'"
./bboa1cng -> //'LIBERTY.V855FP02.LOADLIB(bboa1cng)': executable
./bboa1cnr -> //'LIBERTY.V855FP02.LOADLIB(bboa1cnr)': executable
./bboa1get -> //'LIBERTY.V855FP02.LOADLIB(bboa1get)': executable
./bboa1inv -> //'LIBERTY.V855FP02.LOADLIB(bboa1inv)': executable
./bboa1rca -> //'LIBERTY.V855FP02.LOADLIB(bboa1rca)': executable
./bboa1rcl -> //'LIBERTY.V855FP02.LOADLIB(bboa1rcl)': executable
./bboa1rcs -> //'LIBERTY.V855FP02.LOADLIB(bboa1rcs)': executable
./bboa1reg -> //'LIBERTY.V855FP02.LOADLIB(bboa1reg)': executable
./bboa1srp -> //'LIBERTY.V855FP02.LOADLIB(bboa1srp)': executable
./bboa1srq -> //'LIBERTY.V855FP02.LOADLIB(bboa1srq)': executable
./bboa1srv -> //'LIBERTY.V855FP02.LOADLIB(bboa1srv)': executable
./bboa1srx -> //'LIBERTY.V855FP02.LOADLIB(bboa1srx)': executable
./bboa1urg -> //'LIBERTY.V855FP02.LOADLIB(bboa1urg)': executable
./bboaclnk -> //'LIBERTY.V855FP02.LOADLIB(bboaclnk)': executable
./bboacntl -> //'LIBERTY.V855FP02.LOADLIB(bboacntl)': executable
./bboacsrv -> //'LIBERTY.V855FP02.LOADLIB(bboacsrv)': executable
./bboatrue -> //'LIBERTY.V855FP02.LOADLIB(bboatrue)': executable
```

And if you look in the PDSE, you should see:



```
BROWSE            LIBERTY.V855FP02.LOADLIB                    Row 00001 of 00017
Command ===>                                                  Scroll ===> CSR
           Name     Prompt        Alias-of     Size      TTR     AC   AM  RM
_________ BBOACLNK                           000122A4  000011   00   31  ANY
_________ BBOACNTL                           0000EF4C  000012   00   31  ANY
_________ BBOACSRV                           0000A29C  000013   00   31  ANY
_________ BBOATRUE                           00005078  000014   00   31  ANY
_________ BBOA1CNG                           00000080  000004   01   31  ANY
_________ BBOA1CNR                           00000080  000005   01   31  ANY
_________ BBOA1GET                           00000080  000006   01   31  ANY
_________ BBOA1INV                           00000080  000007   01   31  ANY
_________ BBOA1RCA                           00000080  000008   01   31  ANY
_________ BBOA1RCL                           00000080  000009   01   31  ANY
_________ BBOA1RCS                           00000080  00000A   01   31  ANY
_________ BBOA1REG                           00000080  00000B   01   31  ANY
_________ BBOA1SRP                           00000080  00000C   01   31  ANY
_________ BBOA1SRQ                           00000080  00000D   01   31  ANY
_________ BBOA1SRV                           00000080  00000E   01   31  ANY
_________ BBOA1SRX                           00000080  00000F   01   31  ANY
_________ BBOA1URG                           00000080  000010   01   31  ANY
```

Compile sample COBOL batch

Do the following:

☐ Go to the WP101490 Techdoc page[26] and download the ZIP file that accompanies this Liberty Profile z/OS WOLA Quick Start Guide PDF. Extract the `LEXER2B.txt`

26 http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101490

file. That contains a COBOL source deck[27] for a WOLA program that registers into the server, drives the sample target 10 times, then unregisters and shuts down.

- ☐ Upload that `LEXER2B.txt` file to your system. Whether you put that in a PDS or a sequential data set is a matter of your standard COBOL compilation procedures.
- ☐ Look in the `LEXER2B` source and note the following lines:

```
       MAINLINE SECTION.
           MOVE 'LEXER2B'                    TO register-name.
           MOVE 'GROUP'                      TO daemongroup.
           MOVE 'NAME2'                      TO node-name.
           MOVE 'NAME3'                      TO server-name.
```

  That must match[28] the updates you made to the `server.xml` back on page 13. Your Liberty Profile server is known to WOLA by its three-part name; this COBOL program will now register into the server using the *same* three-part name.

  Modify either `server.xml` or this COBOL source to make the two sets of values match one another. The key is they must match.

- ☐ Compile that COBOL source.

Create SAF CBIND profile

Another SAF profile is needed, this one to control what external IDs may use WOLA to register into the Liberty Profile server.

Do the following:

- ☐ Work with your security administrator to create the following profile:

  `RDEF CBIND BBG.WOLA.`*`group`*`.`*`name2`*`.`*`name3`*` UACC(NONE) OWNER(SYS1)`

  where *group*, *name2* and *name2* match what was used in the `server.xml` (page 13) to create the three-part name for WOLA.

  Notes: You can wildcard the profile to make it more generic. For example, the `CBIND` profile `BBG.WOLA.*` would apply to any three-part name you use in Liberty.

  `CBIND` profiles are stored in *upper-case*. If your three-part name contains lowercase, there will be a mismatch when checking CBIND authority. You can avoid problems by using upper-case values in your `server.xml`. When the three part name is all upper-case you can be sure it'll match an uppercase CBIND. Otherwise, you can work around lower-case values by using wildcard values in the CBIND profile.

- ☐ Then grant the external address space ID (the batch program in this example) `READ` access to the `CBIND` profile:

  `PERMIT BBG.WOLA.`*`group`*`.`*`name2`*`.`*`name3`*` CLASS(CBIND) ACCESS(READ) ID(`*`accessing_id`*`)`

- ☐ And then refresh:

  `SETROPTS RACLIST(CBIND) REFRESH`

Run COBOL batch and validating using sample Java program

Earlier you compiled the `LEXER2B` program. Now it is time to run that program so it will register into the Liberty Profile z/OS server.

27 That sample COBOL is based on a program from WP101490 Techdoc at `ibm.com/support/techdocs`. A “COBOL Primer” document shows how to use the WOLA APIs, and a set of sample COBOL files is provided. `LEXER2B` is based on `EXER2B` from that Techdoc.

28 Including case. We recommend your WOLA three-part name be in uppercase in the `server.xml`, and matched with uppercase in the calling program. This will also help match the CBIND profile.

Do the following:

- ☐ Using whatever JCL you normally use to run a job, run the `LEXER2B` program. It should run for a few seconds then end with RC=0. Look in the output:

  Good sign:

  The batch program output shows:

```
Successfully registered into GROUP
Message sent: 0001 This is a test message
Message back: 0001 This is a test message
Message sent: 0002 This is a test message
Message back: 0002 This is a test message
Message sent: 0003 This is a test message
Message back: 0003 This is a test message
Message sent: 0004 This is a test message
Message back: 0004 This is a test message
Message sent: 0005 This is a test message
Message back: 0005 This is a test message
Message sent: 0006 This is a test message
Message back: 0006 This is a test message
Message sent: 0007 This is a test message
Message back: 0007 This is a test message
Message sent: 0008 This is a test message
Message back: 0008 This is a test message
Message sent: 0009 This is a test message
Message back: 0009 This is a test message
Message sent: 0010 This is a test message
Message back: 0010 This is a test message
Successfully unregistered from GROUP
```

  That means it registered in, looped 10 times and called the `OLASampleLiberty` target program each time, then it unregistered.

  Bad signs:

Batch program output	*Cause*
`SEC3 abend with reason code=20000100`	A Liberty Profile Angel process is in effect operating with code lower than 8.5.5.2 level. See note below.

`BBOA1`**`REG`**` problem, rc/rsn: 00000012/00000016`
`EXITING program due to non-RC=0.`
The very first API call – BBOA1REG – failed.
	The Liberty Profile server is not up and running, or there's a mismatch in the three-part name used by the calling program compared to what the Liberty Profile server is hosting.

`Successfully registered into GROUP`
The very first API call – BBOA1REG – worked, but ...
`OLA - BBOA1INV problem, rc/rsn: 00000012/00000014`
The BBOA1INV call had a problem.
`EXITING program due to non-RC=0.`
	This is a problem with the CBIND profile and the authority of the calling ID to access the profile. See page 18.

Note: If you experience the SEC3 abend with RC=20000100, the likely cause is you have z/OS 2.1 with zOSMF running. That is based on Liberty Profile and has its own Angel process. To get around this, stop zOSMF and its Angel. Start your copy of the Angel at 8.5.5.2 or higher. Restart zOSMF, which will use your Angel. Rerun the batch job to validate WOLA.

Important! zOSMF is now dependent on your Angel process. Do not cancel or stop the Angel.

If you receive the error conditions, then correct as indicated and re-run until you see the success indicated above. When you see it register, loop and unregister, then you will have validated WOLA functionality.

Enable WOLA support in a CICS region

The WOLA support with Liberty Profile works with CICS regions as well[29].

Note: The instructions given here are the *basics* to enable WOLA support in CICS and establish communications between the Liberty Profile server and the CICS region.

Beyond the basics there's a range of CICS system programmer activities for things such as enabling the WOLA Task Related User Exit (TRUE) to start at region startup, and configuring the Link Server Task to start using either INITPARM or sequential terminal. Those tasks are really CICS system programmer tasks, and the steps for Liberty Profile z/OS WOLA are pretty much the same as for full-function WAS z/OS WOLA.

For more on how to perform those CICS system programmer tasks, see the WP101490 Techdoc at `ibm.com/support/techdocs`. The "WOLA Quick Start Guide[30]" provides those step-by-step instructions.

Do the following:

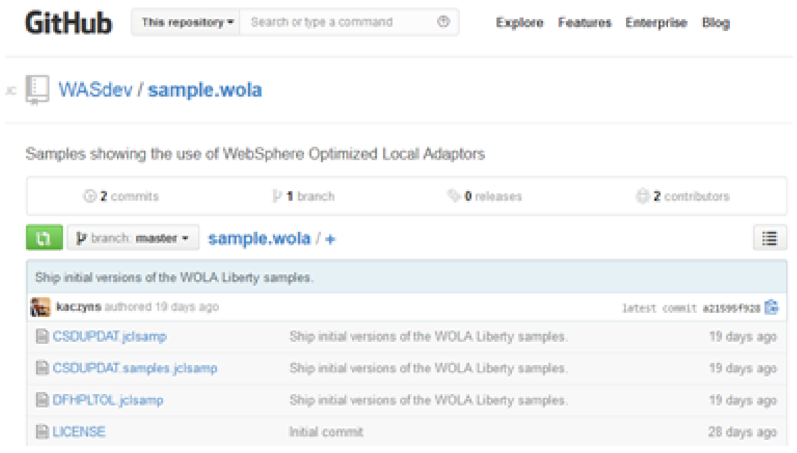
- ☐ Identify the CICS region you wish to use for this exercise. Capture the following:

Region Name:	
ID of started task[31]	

- ☐ Go to the following website

  `https://github.com/WASdev/sample.wola`

  That is where the sample JCL jobs are provided that enable the WOLA support into the CICS region, along with sample WOLA programs. You should see something like this:



WASdev / sample.wola

Samples showing the use of WebSphere Optimized Local Adaptors

2 commits 1 branch 0 releases 2 contributors

branch: master sample.wola / +

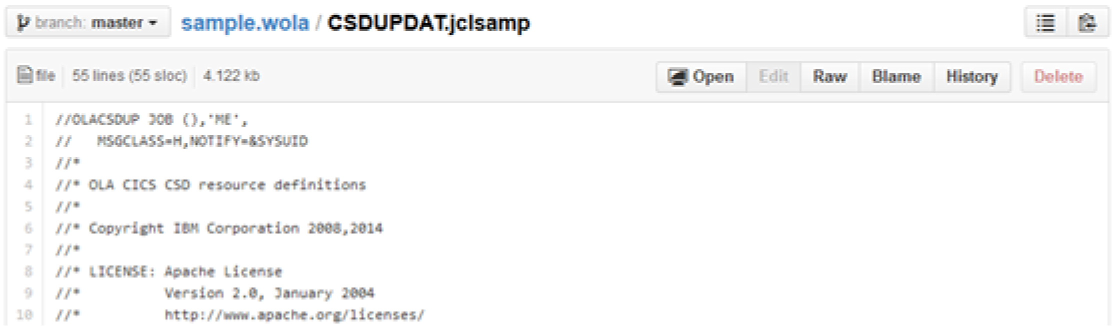
Ship initial versions of the WOLA Liberty samples.

kaczyns authored 19 days ago latest commit a21595f928

CSDUPDAT.jclsamp	Ship initial versions of the WOLA Liberty samples.	19 days ago
CSDUPDAT.samples.jclsamp	Ship initial versions of the WOLA Liberty samples.	19 days ago
DFHPLTOL.jclsamp	Ship initial versions of the WOLA Liberty samples.	19 days ago
LICENSE	Initial commit	28 days ago

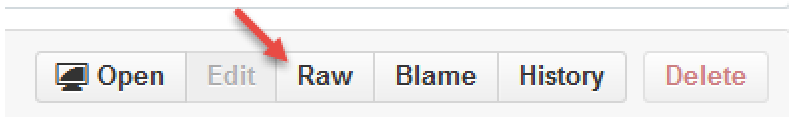
29 See "WOLA and CICS with both Liberty and full-function WAS z/OS" on page 26 or a discussion of using both flavors of WOLA.
30 As opposed to this document, which is the ***Liberty Profile*** Optimized Local Adapters Quick Start Guide.
31 This ID will be granted `READ` to the WOLA `CBIND` profile, which will allow the region to register into the Liberty Profile server using WOLA.

- ☐ Locate and click on the first file in the list – `CSDUPDAT.jclsamp`. That will open up the file and it will look like this:


branch: master
sample.wola / CSDUPDAT.jclsamp
file | 55 lines (55 sloc) | 4.122 kb
Open
Edit
Raw
Blame
History
Delete

```
//OLACSDUP JOB (),'ME',
//   MSGCLASS=H,NOTIFY=&SYSUID
//*
//* OLA CICS CSD resource definitions
//*
//* Copyright IBM Corporation 2008,2014
//*
//* LICENSE: Apache License
//*          Version 2.0, January 2004
//*          http://www.apache.org/licenses/
```

- ☐ Click on the "Raw" button at the top:



That removes the line numbers on the left and makes a "select all / copy" much easier.

- ☐ Enter `Ctrl-A` to select all the contents, then `Ctrl-C` to copy to the clipboard (or whatever mechanism on your workstation performs that function).
- ☐ Create a file on your workstation with name `CSDUPDAT.jclsamp` and paste the contents. Save the file.
- ☐ Do the same for the following three files:

○ `CSDUPDAT.jclsamp`	JCL to update the CICS CSD with the WOLA infrastructure components such as the Task Related User Exit and the Link Server Task.
○ `CSDUPDAT.samples.jclsamp`	JCL to update the CICS CSD with the WOLA samples. The only sample we'll be using is `OLACB01`.[32]
○ `OLACB01.jclsamp`	JCL and source for the OLACB01 source program, which is a simple COMMAREA echo program with a single field.

- ☐ Upload those files *in ASCII* to a FB 80 PDS on your system[33].
- ☐ Modify the `CSDUPDAT` member so the `STEPLIB` and `DFHCSD DD` cards accurately reflect your system's CICS installation.
- ☐ Review and modify as necessary the `ADD` statement at the bottom of the file so it conforms with your CICS installation standards.
- ☐ Run that job and look for RC=0.
- ☐ Modify the CSD update member for the samples and do the same[34].
- ☐ Modify the `OLACB01.jclsamp` member so the `OLACB01` program can be compiled into a load library that will be recognized by the CICS region[35].
- ☐ Edit the JCL start procedure for the CICS region and add the WOLA module PSDE (the one you created on page 17) to the `DFHRPL DD` concatenation.
- ☐ Grant the CICS region ID `READ` access to the `CBIND` profile created earlier (page 18):

`PERMIT BBG.WOLA.`*`group`*`.`*`name2`*`.`*`name3`*` CLASS(CBIND) ACCESS(READ) ID(`*`cics_id`*`)`

32 The WOLA support for full-function WAS z/OS provides the CSD updates in one sample file. For Liberty Profile z/OS WOLA the CSD updates are broken out into two files: one for the key WOLA infrastructure elements; another for the sample programs.
33 You'll need to make the PDS member name for the `CSDUPDAT.samples.jclsamp` something other than `CSDUPDAT` since the `CSDUPDAT.jclsamp` file will take that member name.
34 We will only make use of the `OLACB01` sample, so if you wish you may trim the file so only that update is made to the CSD. Otherwise, leave as is and have the CSD updates in place for the future if you choose to use the other samples.
35 For example, the PDSE into which the WOLA modules were copied as shown on page 17.

- ☐ Then refresh:

  SETROPTS RACLIST(CBIND) REFRESH

- ☐ Start the CICS region. In the `MSGUSR` held output for the region you should see a series of messages indicating the WOLA support is added:

```
Resource definition for BBOACALL has been added.
 :
TRANSACTION definition entry for BBOC has been added.
TDQUEUE entry for BBOQ has been added.
```

- ☐ Make sure the Liberty Profile server is started. The next step involves issuing a CICS command to start the link server task and register into the Liberty Profile server. Therefore, the server must be active for this to succeed.
- ☐ Open a terminal session with the CICS region.
- ☐ Issue the following command[36]:

  BBOC START_TRUE

  You should see the following message indicating success:

```
WOLA TRACE 1: Exit enabled successfully.
```

- ☐ Issue the following command (*as one line*) to start the link server task and register into your server[37] [38]:

```
BBOC START_SRVR RGN=CICSREG DGN=GROUP NDN=NAME2
                       SVN=NAME3 SVC=* MNC=1 MXC=10 TXN=N SEC=N REU=Y
```

  Note: Where `GROUP`, `NAME2` and `NAME3` matches the WOLA three-part name you set for your Liberty Profile server back on page 13.

  You should see the following message indicating success:

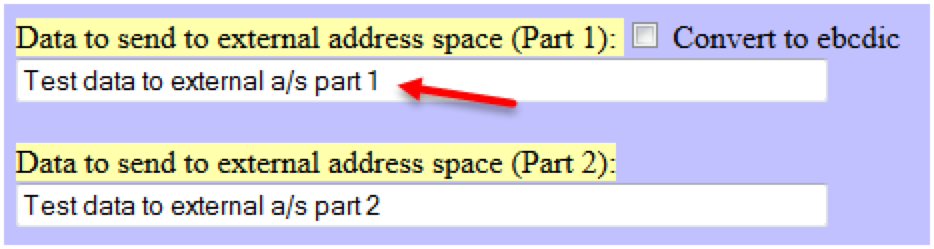
```
WOLA TRACE 0: Start server completed successfully.
```

  If you get a non-zero return code and reason code, then go to the WebSphere Application Server z/OS Knowledge Center[39] and search on the string **cdat_olaapis**. Open the link and go to the “Register” section (also known as the `BBOA1REG` section) and look for the RC and RSN code provided there[40].

- ☐ With a browser, go to the WOLA sample program you deployed in Liberty Profile earlier:

  http://*<host>*:9080/OLASampleLibertyWeb/

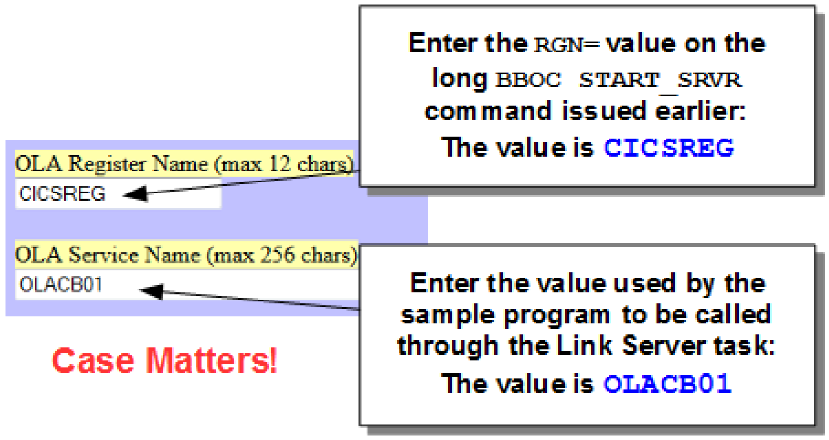
- ☐ Note the field at the top … that's the data that will be sent over to CICS:


Data to send to external address space (Part 1):
Convert to ebcdic
Test data to external a/s part 1
Data to send to external address space (Part 2):
Test data to external a/s part 2

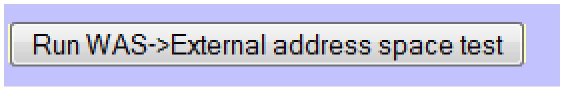
  If you want, change that value. Or let the default flow over and back.

36 This can be automated using samples `DFHPLTOL` and `BBOACPLT`. See WP101490 Techdoc Quick Start Guide.
37 `RGN=` may be any value you wish up to 12 characters. `CICSREG` is just an example of a registration name.
38 This can be automated using INITPARM or sequential terminal. See WP101490 Techdoc Quick Start Guide.
39 http://www.ibm.com/support/knowledgecenter
40 The most common problems are the ones we saw with the batch program on page 19.

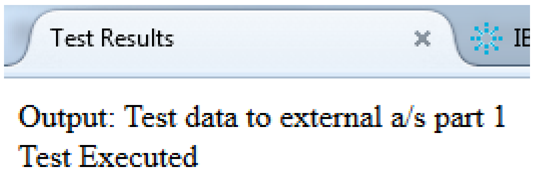
- ☐ Enter the registration name and the service name:


OLA Register Name (max 12 chars)
CICSREG
OLA Service Name (max 256 chars)
OLACB01
Case Matters!
Enter the RGN= value on the long BBOC START_SRVR command issued earlier: The value is CICSREG
Enter the value used by the sample program to be called through the Link Server task: The value is OLACB01

- ☐ Scroll to the bottom and click the "Run" button:



- ☐ This will simply echo back the text string sent over:



Output: Test data to external a/s part 1
Test Executed

That validates the ability to communicate over WOLA to CICS using the WOLA Link Server Task[41].

- ☐ From the CICS terminal, enter the command:

  BBOC STOP_SRVR RGN=CICSREG

  Where CICSREG matches the RGN= value supplied on the BBOC command you entered on page 22.

- ☐ Then issue the command:

  BBOC STOP_TRUE

At this point the WOLA Link Server Task is stopped, the registration into the Liberty Profile server taken down, and the TRUE stopped. If you wished to use WOLA into this CICS region you would restart the TRUE and the Link Server Task[42].

41 If you look at the source for OLACB01 you'll find it does not use any of the WOLA native APIs. OLACB01 is completely WOLA-unaware. The Link Server Task shields it from having to know about WOLA.

42 The TRUE is commonly started at CICS region initialization using the supplied PLT program. Starting the Link Server Task implies the creation of a registration, and that requires the Liberty Profile server be up ready to accept the registration. After the initial successful start of the Link Server with registration, the Link Server will retry registration if the Liberty Profile server goes down and comes back up.

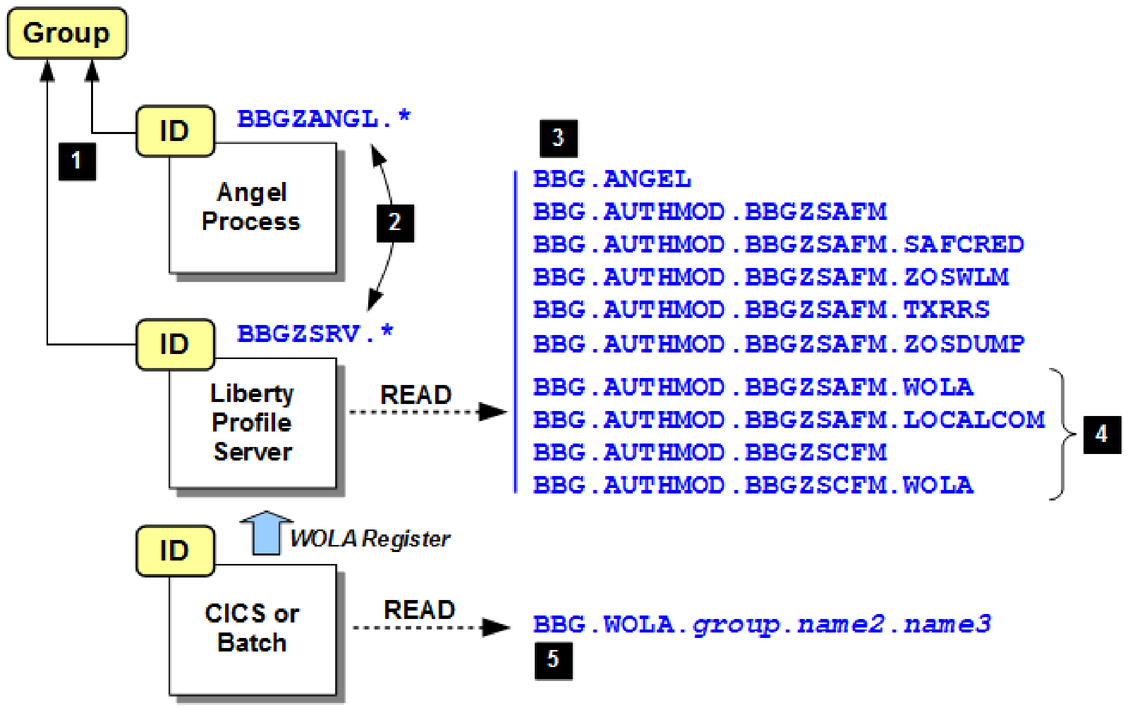
Miscellaneous Information

This section will serve as a gathering point for miscellaneous information we found useful, but chose not to include inline with the step-by-step discussion earlier.

SAF definitions

Overview of the SAF security model

What will follow is a set of SAF commands to create users, groups, and profiles. We used these in the earlier step-by-step instructions. To make sense of those details, here's an overview picture of the SAF security model:


Group
ID
BBGZANGL.*
Angel Process
1
2
3
ID
BBGZSRV.*
Liberty Profile Server
READ
BBG.ANGEL
BBG.AUTHMOD.BBGZSAFM
BBG.AUTHMOD.BBGZSAFM.SAFCRED
BBG.AUTHMOD.BBGZSAFM.ZOSWLM
BBG.AUTHMOD.BBGZSAFM.TXRRS
BBG.AUTHMOD.BBGZSAFM.ZOSDUMP
BBG.AUTHMOD.BBGZSAFM.WOLA
BBG.AUTHMOD.BBGZSAFM.LOCALCOM
BBG.AUTHMOD.BBGZSCFM
BBG.AUTHMOD.BBGZSCFM.WOLA
4
WOLA Register
ID
CICS or Batch
READ
BBG.WOLA.group.name2.name3
5

The following numbers correspond to the blocks in the diagram:

1. The Angel process and the Liberty Profile server will each have its own ID, separate from one another[43]. Both may connect to a common group.
2. SAF `STARTED` profiles are used to assign the IDs to the respective started tasks: the Angel process ID to the Angel started task; and the Liberty Profile server ID to the server started task.
3. A collection of SAF `SERVER` profiles are used to control access to various z/OS related services. The first two are required for WOLA support; the final four are used for other Liberty Profile functions that take advantage of the z/OS platform. A brief description of what each profile is provided under "SERVER profiles" on page 25.
4. The four `SERVER` profiles highlighted here are new in 8.5.5.2[44] and relate to the WebSphere Optimized Local Adapter support.
5. A `CBIND` profile is used to control who may "register in" to a Liberty Profile server using WOLA. Each Liberty Profile server will carry a unique *`group.name2.name3`* sequence. `READ` to the `CBIND` that specifies the *`group.name2.name3`* sequence provides access. This `CBIND` may be wild-carded so one `CBIND` permits access to many different server *`group.name2.name3`* sequences.

43 Having the Angel process ID separate from the server ID is not a technical requirement. It is however a good practice.
44 The first six in the picture are original to Liberty Profile z/OS, which became available with 8.5.0.0.

ID and Group

Creates a Liberty Profile group ID

```
ADDGROUP lib_group OMVS(GID(gid)) OWNER(SYS1)
```

Creates the Angel ID and connects it to the Liberty Profile group

```
ADDUSER angel_id DFLTGRP(lib_group) OMVS(UID(uid)
                   HOME(angel_home) PROGRAM(/bin/sh)) NAME('Liberty Angel')
                                       OWNER(lib_group) NOPASSWORD NOOIDCARD
```

Creates the Liberty Profile server ID and connects it to the Liberty Profile group

```
ADDUSER liberty_id DFLTGRP(lib_group) OMVS(UID(uid)
                 HOME(server_home) PROGRAM(/bin/sh)) NAME('Liberty Server')
                                       OWNER(lib_group) NOPASSWORD NOOIDCARD
```

STARTED for Liberty Profile Angel process and server

Creates the STARTED profile for the Angel Process

```
RDEF STARTED BBGZANGL.* UACC(NONE) STDATA(USER(angel_id)
                    GROUP(lib_group) PRIVILEGED(NO) TRUSTED(NO) TRACE(YES))
```

Creates the STARTED profile for the Liberty Profile server

```
RDEF STARTED BBGZSRV.* UACC(NONE) STDATA(USER(liberty_id)
                    GROUP(lib_group) PRIVILEGED(NO) TRUSTED(NO) TRACE(YES))
```

Refreshes the STARTED class profiles

```
SETROPTS RACLIST(STARTED) REFRESH
```

SERVER profiles

Grants an ID general access to the Angel process for authorized services

```
RDEF SERVER BBG.ANGEL UACC(NONE) OWNER(SYS1)
PERMIT BBG.ANGEL CLASS(SERVER) ACCESS(READ) ID(liberty_id)
```

Controls which server processes can use the BBGZSAFM authorized module in the Angel process

```
RDEF SERVER BBG.AUTHMOD.BBGZSAFM UACC(NONE) OWNER(SYS1)
PERMIT BBG.AUTHMOD.BBGZSAFM CLASS(SERVER) ACCESS(READ) ID(liberty_id)
```

Controls which server processes can use BBGZSAFM for SAF services

```
RDEF SERVER BBG.AUTHMOD.BBGZSAFM.SAFCRED UACC(NONE) OWNER(SYS1)
PERMIT BBG.AUTHMOD.BBGZSAFM.SAFCRED CLASS(SERVER) ACCESS(READ) ID(liberty_id)
```

Controls which server processes can use BBGZSAFM for WLM services

```
RDEF SERVER BBG.AUTHMOD.BBGZSAFM.ZOSWLM UACC(NONE) OWNER(SYS1)
PERMIT BBG.AUTHMOD.BBGZSAFM.ZOSWLM CLASS(SERVER) ACCESS(READ) ID(liberty_id)
```

Controls which server processes can use BBGZSAFM for RRS services

```
RDEF SERVER BBG.AUTHMOD.BBGZSAFM.TXRRS UACC(NONE) OWNER(SYS1)
PERMIT BBG.AUTHMOD.BBGZSAFM.TXRRS CLASS(SERVER) ACCESS(READ) ID(liberty_id)
```

Controls which server processes can use BBGZSAFM for z/OS Dump services

```
RDEF SERVER BBG.AUTHMOD.BBGZSAFM.ZOSDUMP UACC(NONE) OWNER(SYS1)
PERMIT BBG.AUTHMOD.BBGZSAFM.ZOSDUMP CLASS(SERVER) ACCESS(READ) ID(liberty_id)
```

Controls which server processes can use BBGZSAFM for WOLA services

```
RDEF SERVER BBG.AUTHMOD.BBGZSAFM.WOLA UACC(NONE) OWNER(SYS1)
PERMIT BBG.AUTHMOD.BBGZSAFM.WOLA CLASS(SERVER) ACCESS(READ) ID(liberty_id)
```

Controls which server processes can use BBGZSAFM for LOCALCOM services

```
RDEF SERVER BBG.AUTHMOD.BBGZSAFM.LOCALCOM UACC(NONE) OWNER(SYS1)
PERMIT BBG.AUTHMOD.BBGZSAFM.LOCALCOM CLASS(SERVER) ACCESS(READ) ID(liberty_id)
```

Controls which server processes can use the BBGZSCFM authorized module in the Angel process

```
RDEF SERVER BBG.AUTHMOD.BBGZSCFM UACC(NONE) OWNER(SYS1)
PERMIT BBG.AUTHMOD.BBGZSCFM CLASS(SERVER) ACCESS(READ) ID(liberty_id)
```

Controls which server processes can use BBGZSCFM for WOLA

```
RDEF SERVER BBG.AUTHMOD.BBGZSCFM.WOLA UACC(NONE) OWNER(SYS1)
PERMIT BBG.AUTHMOD.BBGZSCFM.WOLA CLASS(SERVER) ACCESS(READ) ID(liberty_id)
```

Refreshes the SERVER class profiles

```
SETROPTS RACLIST(SERVER) REFRESH
```

CBIND for registration into server

Creates the CBIND profile to control who may register into the Liberty Profile server using WOLA

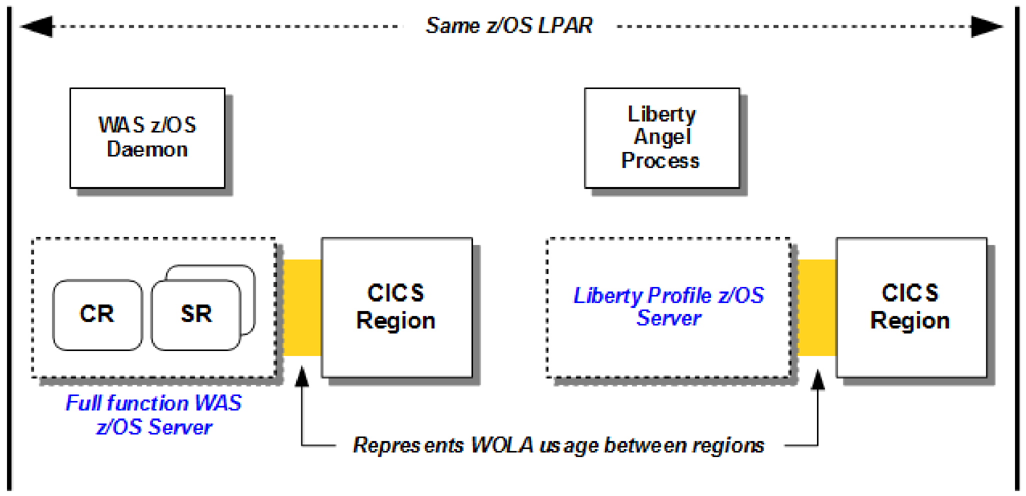
```
RDEF CBIND BBG.WOLA.group.name2.name3 UACC(NONE) OWNER(SYS1)
PERMIT BBG.WOLA.group.name2.name3 CLASS(CBIND) ACCESS(READ) ID(accessing_id)
```

Refreshes the CBIND class profiles

```
SETROPTS RACLIST(CBIND) REFRESH
```

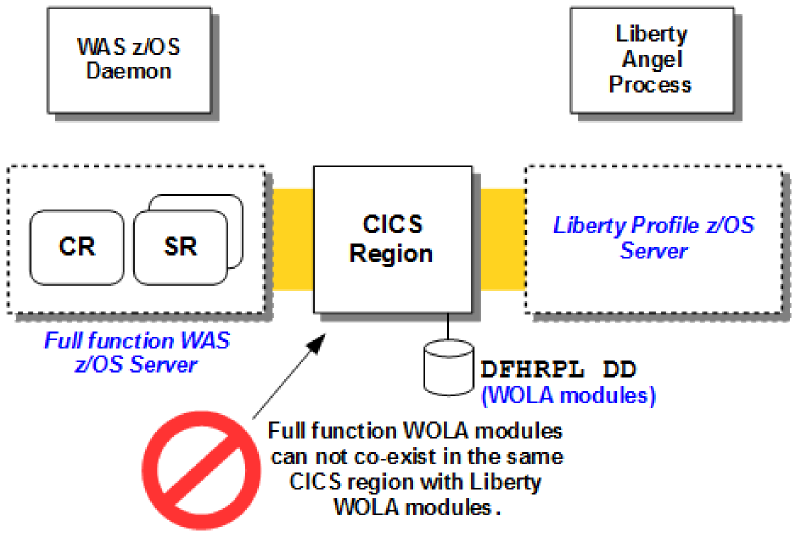
WOLA and CICS with both Liberty and full-function WAS z/OS

It is quite possible to have full-function WAS z/OS WOLA operating concurrently with Liberty Profile z/OS WOLA *on the same LPAR*. The following picture illustrates this:

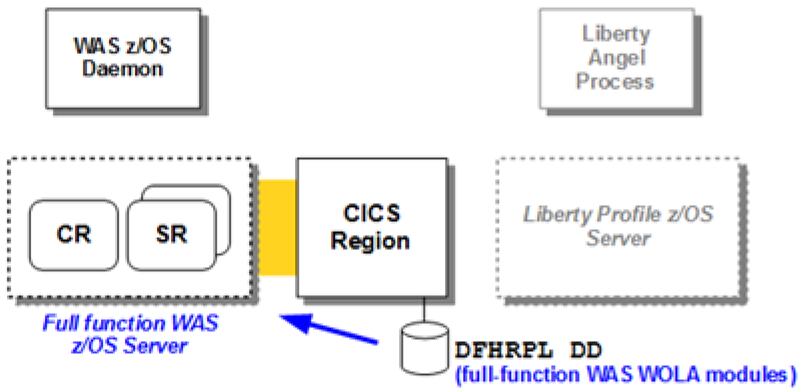

Same z/OS LPAR
WAS z/OS Daemon
Liberty Angel Process
CR
SR
CICS Region
Liberty Profile z/OS Server
CICS Region
Full function WAS z/OS Server
Represents WOLA usage between regions

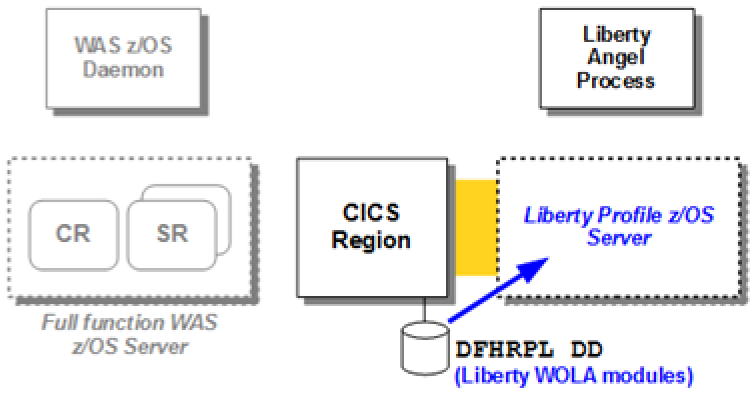
In this case there's no requirement full-function WAS z/OS and Liberty Profile z/OS be at the same verion and maintenance level. They can be different because the two environments are quite distinct and separate.

But there are limitations. You can *not*, for example, operate both flavors of WOLA *concurrently* in the same CICS region:


WAS z/OS Daemon
Liberty Angel Process
CR
SR
CICS Region
Liberty Profile z/OS Server
Full function WAS z/OS Server
DFHRPL DD
(WOLA modules)
Full function WOLA modules can not co-exist in the same CICS region with Liberty WOLA modules.

You *can* toggle a CICS region between flavors fairly easily. It would be a matter of stopping the region, changing the `DFHRPL` so it points to the PDSE with the proper modules in it, and restarting the region:


WAS z/OS Daemon
Liberty Angel Process
CR
SR
CICS Region
Liberty Profile z/OS Server
Full function WAS z/OS Server
DFHRPL DD
(full-function WAS WOLA modules)



Otherwise, the functions are very similar:

- The WOLA TRUE in CICS can be started either with the `BBOC START_TRUE` command or using a PLT initialization program.
- The WOLA Link Server task can be started either with the `BBOC START_SRVR` command, using INITPARM, or using sequential terminal.
- The syntax of the `BBOC START_SRVR` command is the same for both environments[45].
- The native APIs have the same interfaces, so the programming model is the same for coding to the APIs.
- The Java-side JCA resource adapter has the same interface.

Summary of Liberty WOLA compared to full-function WAS WOLA

There are many similarities between WOLA for Liberty and WOLA for full-function WAS z/OS. But there are some differences:

- Support for IMS is not included. The current support for Liberty Profile z/OS WOLA includes batch and CICS.
- No global transaction support, so registration into Liberty will always be `TXN=N`.
- Security assertion *is* supported, so `SEC=Y` works as before when outbound/inbound to CICS.
- JCA interface for outbound calls is the same. However, outbound WOLA Java apps may or may not carry to Liberty unchanged because of (a) global TX, which is not supported in Liberty WOLA, or (b) other things app is doing with full-function WAS z/OS that is not part of Liberty Profile z/OS[46].
- The CICS Link Server operates with same principles. However, the BBOC command for the Liberty WOLA support in CICS does not support the LIST_SRVR operand that became available with full-function WAS z/OS 8.5.5.2 WOLA support for CICS.

45 But some functionality is limited for Liberty WOLA. See "Summary of Liberty WOLA compared to full-function WAS WOLA" on page 27.
46 For example, full function WAS z/OS is Java EE compliant; Liberty Profile z/OS is not.

- Liberty Profile WOLA inbound EJB has different structure from full-function WAS z/OS target EJB. Liberty Profile requires the target EJB to be 3.x compliant. If full function WAS program was EJB 3.x then it would be relatively easy to modify. Liberty does not support EJB 2.x, so if the full-function WAS target EJB was EJB 2.x that implies a re-write.
- Native APIs the same, but the JNDI name of target EJB will be based on what Liberty generates. You have to start the Liberty Profile server and look in the log for the message that carries the JNDI of the target EJB. That is then used in the native program as the service name to be called over WOLA.
- Development Mode (also known as the "Proxy EJB support" is not in Liberty Profile WOLA. But that is not really much of a limitation since Liberty z/OS is essentially the same as Liberty on any other platform. Deploying an application is as simple as FTP'ing EAR up to `/dropins` directory.
- There is no `MODIFY` command for Liberty WOLA to display the state of the registrations, as there is for full-function WAS z/OS.

Angel process and the new WOLA support

Liberty Profile z/OS and the Angel process was first made available with WAS z/OS V8.5.0.0.

WOLA support was first made available with WAS z/OS 8.5.5.2.

The Angel process was designed to be started and left running essentially forever. Only one Angel process is needed per z/OS LPAR[47] regardless of how many Liberty Profile server instances are on the LPAR. Even then the Angel is only *required* for certain things[48]. It was designed to rarely, if ever, require a restart.

However, the new WOLA support of Liberty Profile z/OS requires the 8.5.5.2 level of code be loaded with the instance of the Angel process. That implies a stop and restart of Angel process so it can pick up and load the 8.5.5.2 code with the WOLA support.

Back on page 10 we had you copy the Angel JCL start procedure and modify the `SET ROOT=` value to point to your *`INSTALL_DIR`*, which would contain the 8.5.5.2 code. Then on page 11 we had you start the Angel process. We never again had you stop the Angel. There was no need – once started with the 8.5.5.2 code, the Angel can remain up.

Key Point: If you have a z/OS LPAR environment where an Angel process is running with some level of code prior to that needed by the WOLA support, you will need to restart the Angel at the new level. You can check to see what level of code an instance of the Angel process is running by issuing the following command:

```
F <jobname>,VERSION
```

where *`<jobname>`* is the job name of the Angel started task. The WOLA support is indicated with the following message:

```
CWWKB0053I ANGEL VERSION 2
```

If you see this:

```
CWWKB0053I ANGEL VERSION 1
```

then you're back level. You would need to stop your Angel, change the `SET ROOT=` pointer in the JCL start procedure to point to where 8.5.5.2 is installed, then restart the Angel process. Then issue the `MODIFY VERSION` command again to verify you're at the proper level for WOLA.

47 If you try to start a second, the second instance will fail to come up.
48 The WOLA support being one such thing that requires the Angel. The JDBC Type 2 support using RRS is another. The z/OS MODIFY command to process dump actions is a third.

Document Change History

Check the date in the footer of the document for the version of the document.

July 24, 2014	Original document at time of Techdoc creation
September 19, 2014	Updated to include information on a problem that surfaces running WOLA for the first time. The symptom is an SEC3 abend of the batch program with RC=20000100. That is likely caused by zOSMF 2.1 running with its Angel operating at some level of code lower than 8.5.5.2.

End of WP101490