WebSphere Application Server for z/OS

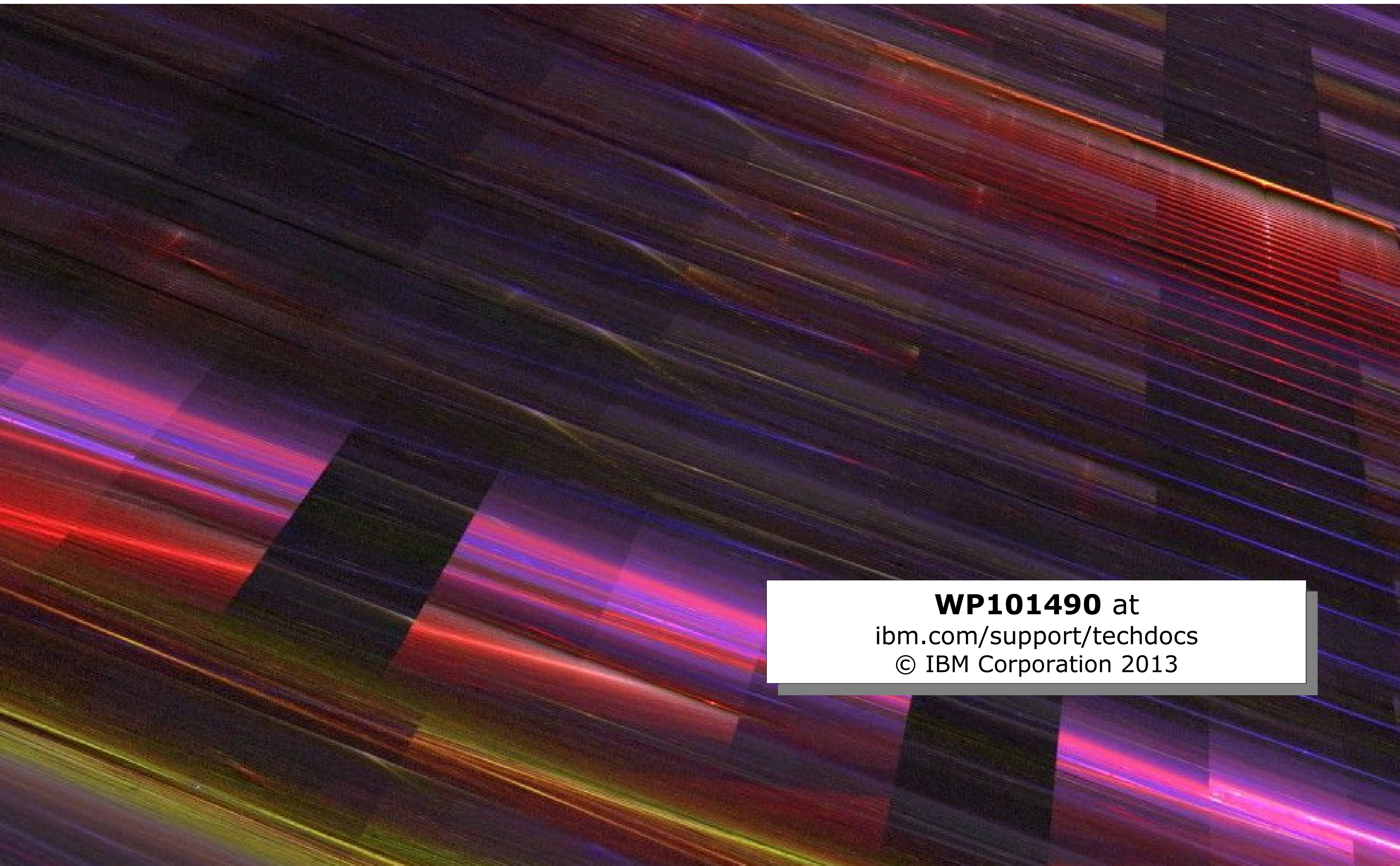
WOLA Quick Start Guide

A step-by-step guide to setting up and using WOLA

Version Date: August 16, 2013

See "Document Change History" on page 80 for a description of the changes in this version of the document

IBM Advanced Technical Skills
Gaithersburg, MD



WP101490 at
ibm.com/support/techdocs
© IBM Corporation 2013

The WAS z/OS team in ATS consists of John Hutchinson, Mike Kearney, Mike Loos, Louis Wilen, Lee-Win Tai and Don Bagwell. Glenn Fisher manages the team.

Mike Cox, Distinguished Engineer, serves as technical consultant to all our activities.

Special thanks to Jim Mulvey, Tim Kaczynski and David Follis of the WAS z/OS development organization.

And thanks for John Cowel of IBM Technical Sales for his help with the CICS sequential terminal section, as well as Leigh Compton of ATS CICS support for her overall assistance with CICS.

Table of Contents

Overview 5
Objective of this document 5
Assumptions and prerequisites to using this guide 5
WOLA and Version 7 5
Sources of information referred to by this guide 5
WebSphere Application Server for z/OS Information Center 5
IBM Advanced Technical Skills Techdocs Site 5
Preparing to Use WOLA - Preliminary Setup 6
FYI: Overview of the process 6
Capture the version and fixpack level of WAS installation 6
Copy WOLA samples to a sample data set 6
Copy WOLA modules to a load library data set 7
Check contents of sample and module data sets 7
Create cell-level WOLA "enable adapter" environment variable 8
Create cell-level WOLA "identity propagate" environment variable 8
Stop and restart the entire cell 9
Install the ola.rar resource adapter and create a connection factory 9
Install sample EAR file 11
Status checkpoint 13
Enabling WOLA in CICS Region and Initial Validation 14
FYI: Overview of the process 14
Customize sample CSDUPDAT member and submit 14
Customize sample DFHPLTOL member and submit 14
Customize sample OLAMAP and OLAUTIL members and submit 15
Customize sample OLACB01 member and submit 15
Update CICS region SIP member 16
Update the CICS region JCL and concatenate to the DFHRPL DD 16
Review the WAS z/OS server CBIND profile for CICS region ID access 16
Stop and restart the CICS region and verify WOLA components 17
Log into CICS region and use BBOC to start the Link Server Task and register into server 17
Other BBOC commands issued in CICS terminal session 18
Status checkpoint 19
WOLA and CICS Usage Scenarios 20
Issue BBOC START_SRVR to start the link server task and register into WAS z/OS 20
Outbound WAS into CICS with OLACB01 as target service 20
Inbound CICS into WAS using OLAUTIL 3270 application in CICS 21
Outbound WAS into CICS using ATS Java application 22
Optional - Propagation of Userid from WAS into CICS 24
WAS z/OS Version 7 and application with Basic Auth enabled 29
Optional - Use the CICS Channels and Containers sample OLACB02 32
Optional - Link Server start with BBOACPL2 PLT and INITPARM 35
Optional - Link Server start with BBOACPL3 PLT 35
Optional - Link Server start with Sequential Terminal 36
Optional - "Development Mode" ... Distributed WAS using Proxy EJB into CICS 36
Overview 36
Determine security settings for WAS z/OS 37
Determine security settings for distributed WAS, and enable if necessary 37
Assumed starting point for WAS z/OS environment 39
Install Proxy EJB in WAS z/OS 40
Download and install ola.rar in distributed WAS environment 41
Download and install the OLASample2 sample application 42
Configure custom properties on JCA connection factory 42
Configure security settings on both distributed and z/OS 44
Restart WAS on both distributed and z/OS 47

Validate proxy function up to WAS z/OS and into CICS....48
Alternative authentication - client certificate....49
Status checkpoint....57
WOLA and Batch Usage Scenarios 58
Pull COBOL samples from WP101490 Techdoc....58
Review the SAF CBIND profile for batch job ID access....58
Review contents of EXER2B job....59
Customize, Compile and run EXER2B....60
Review contents of EXER2F job....60
Customize, Compile and run EXER2F....61
Status checkpoint....61
Appendix A: Enabling WOLA in WAS z/OS Version 7....62
WebSphere Application Server for z/OS Version 7 Information Center....62
Determine the Fixpack level of WAS z/OS V7....62
Pre-allocate a LIBRARY and SAMPLES data sets....62
Run olaInstall.sh shell script with INIT....63
Copy the WOLA samples to your samples data set....64
Create two cell-level environment variables....64
Install the ola.rar JCA resource adapter....65
Review CBIND Profile....65
Stop and restart the entire cell environment....65
Appendix B: Starting Link Server Task at CICS Startup....66
Comparing the capabilities of the approaches (unfinished)....66
BBOACPL2 and INITPARM=()....66
BBOACPL3 and longer BBOC commands....67
If BBOACPL3 not part of your copy of samples....68
Sequential Terminal....71
Example of sequential terminal definitions in our test environment....72
Appendix C: Miscellaneous Information....73
Requirements to match module library with WAS z/OS level in use....73
Is there a way to tell what level the modules in the LIBRARY are?....73
What's the rule ... update modules with every fixpack?....73
What symptom would is seen if there's a mismatch?....73
Update of ola.rar file....73
Location of the olaRarUpdate.py script file....74
Invoking the script file to update the RAR file....74
Improved Channels and Containers using 8.0.0.5 or 8.5.0.2....75
8.0.0.5 / 8.5.0.2 (or higher) and OLACB02 sample....78
Document Change History 80

Overview

WebSphere Optimized Local Adapters, or WOLA for short, is a component of IBM's WebSphere Application Server for z/OS product. It provides a very efficient and very low-latency mechanism for exchanging information between WAS on z/OS and external address spaces such as CICS, IMS or batch programs. The WP101490 Techdoc has much more overview information.

WOLA is unique to the WAS z/OS product because only the z/OS operating system has the capability to exchange memory buffers between running programs.

Objective of this document

The objective of this guide is to provide you with a step-by-step process to enable and use some of the basics of WOLA.

Assumptions and prerequisites to using this guide

To use this guide the following is assumed:

WAS z/OS 8.0.0.1 or higher	This provides a baseline from which this document may then provided detailed instructions on how to enable and use.
WAS z/OS operational runtime	This guide assumes you have an operational WAS z/OS runtime available to you, and that you have the authority to make changes to the runtime.
Target server in 64-bit mode	For a server to use WOLA it must operate in 64-bit mode.
Knowledge and skills	Though this guide will provide relatively detailed step-by-step instructions, a certain level of familiarity is assumed: • General knowledge of the WAS Admin Console • General knowledge of Java application concepts • General knowledge of z/OS (ISPF, allocate, copy, etc.) • General knowledge of navigating within a UNIX environment • General knowledge of compiling COBOL programs • General knowledge of CICS system programming tasks

WOLA and Version 7

The process of enabling WOLA in WAS z/OS is slightly different for V7 than V8 or V8.5. If you are using V7, see "Appendix A: Enabling WOLA in WAS z/OS Version 7" starting on page 62.

Note: If you on WAS z/OS V7, make sure you are at fixpack 7.0.0.21 or above. This is particularly important if you plan to use WOLA with CICS TS 4.2, but the recommendation applies regardless of the level of CICS you're at.

Sources of information referred to by this guide

WebSphere Application Server for z/OS Information Center

The InfoCenter provides the official product documentation.

The URLs for the InfoCenter are:

V8	`http://pic.dhe.ibm.com/infocenter/wasinfo/v8r0/index.jsp`
V8.5	`http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/index.jsp`

Throughout this guide when an InfoCenter article is referenced, we will provide the unique search string to narrow to the specific article.

IBM Advanced Technical Skills Techdocs Site

The WP101490 document on Techdocs is where ATS stores its documents on WOLA:

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

Preparing to Use WOLA - Preliminary Setup

Before using WOLA there are a few housekeeping steps to be done ahead of time.

FYI: Overview of the process

Enabling WOLA in WAS z/OS is relatively simple with V8 and above[1]. It involves a few simple steps:

- Running a supplied shell script to copy the WOLA samples and the WOLA modules out to z/OS data sets
- Creating a few cell-level environment variables in the WAS z/OS runtime
- Installing a supplied JCA resource adapter (ola.rar) into the runtime and creating a JCA connection factory (all done through the Admin Console)

Success is achieved when the WAS z/OS runtime is started with messages indicating the WOLA adapter has been enabled.

Capture the version and fixpack level of WAS installation

Overview:

Having knowledge of the version and fixpack level of the WAS installation is a good thing to have. Here you will capture it and use it as part of the z/OS data set qualifier for WOLA samples and modules.

Do the following:

- ☐ Open a Telnet or OMVS session into UNIX Systems Services on the z/OS system
- ☐ Change directories to the `/profiles/default/bin` directory[2] of the cell environment you intend to use for this WOLA test.
- ☐ Issue the command:

```
./versionInfo.sh
```

- ☐ In the output, find the following:

```
Name          IBM WebSphere Application Server for z/OS
Version       x.x.x.x
```

Where *x.x.x.x* is the version and fixpack level for the WAS installation.

- ☐ Write that value here: ________________

Copy WOLA samples to a sample data set

Overview:

WOLA ships with a set of JCL samples that are needed to enable WOLA. They first need to be copied out from the file system to a FB 80 PDS.

Do the following:

- ☐ Issue command **`pwd`** and make sure you're still in the `/profiles/default/bin` directory.
- ☐ Issue the following command:

```
./copyZOS.sh OLASAMPS 'hlq.version.WOLA.SAMPLES'
```

Where:

hlq	Is your desired data set high-level qualifier
version	Is the version and fixpack level of the WAS installation with the dots removed, for example: 8001 or 8500[3].

1 With WAS z/OS Version 7 there was a bit more involved because WOLA first came available as part of the maintenance stream, which meant the new function had to be deliberately enabled by the administrator. See "Appendix A: Enabling WOLA in WAS z/OS Version 7" on page 62 for more on V7 enablement.

2 Either `/AppServer` or `/DeploymentManager` ... both will work equally well.

Note: The `copyZOS.sh` shell script will create the data set if it does not already exist.

However, *depending on the authority of the ID used to issue the shell script*, the allocate of the data set *may* fail. The symptom you will see for a failed allocate will be:

```
RC(12)
cp: FSUM8979 target "//'hlq.version.WOLA.SAMPLES'" must exist
```

If that happens, then pre-allocate the samples data set with the following attributes and re-issue the `copyZOS.sh` command.

```
Space units . . . . . TRACK
Primary quantity  . . 10
Secondary quantity    2
Directory blocks  . . 10
Record format . . . . FB
Record length . . . . 80
Block size  . . . . . 27920
Data set name type    PDS
```

Copy WOLA modules to a load library data set

Overview:

The WOLA native modules are shipped as part of the WAS z/OS file system. For external address spaces such as CICS to use them they need to be in a `LIBRARY` data set. This step simply copies the modules from the file system to the data set.

Do the following:

- ☐ Issue command **`pwd`** and make sure you're still in the `/profiles/default/bin` directory.
- ☐ Issue the following command:

```
./copyZOS.sh OLAMODS 'hlq.version.WOLA.MODULES'
```

Where:

`hlq`	Is your desired data set high-level qualifier
`version`	Is the version and fixpack level of the WAS installation with the dots removed, for example: 8001 or 8500[4].

Note: The `copyZOS.sh` shell script will create the data set if it does not already exist.

However, *depending on the authority of the ID used to issue the shell script*, the allocate of the data set *may* fail. The symptom you will see for a failed allocate will be:

```
RC(12)
cp: FSUM8979 target "//'hlq.version.WOLA.MODULES'" must exist
```

If that happens, then pre-allocate the samples data set with the following attributes and re-issue the `copyZOS.sh` command.

```
Space units . . . . . TRACK
Primary quantity  . . 35
Secondary quantity    2
Directory blocks  . . 10
Record format . . . . U
Record length . . . . 0
Block size  . . . . . 32760
Data set name type    LIBRARY
```

Check contents of sample and module data sets

Overview:

This is a simple visual check to insure files copied out properly.

3 Having the samples data set qualified with the version number is not as important as having the module library. But we have you do it for consistency.

4 There is a requirement that the WOLA modules used by external address spaces like CICS be at a level compatible with the level of WAS used by the target server. Having the version number as part of the qualifier will help if compatibility is suspected as a problem. See "Requirements to match module library with WAS z/OS level in use" on page 73 for more on this topic.

Do the following:

- ☐ Browse the *`hlq.version`*`.WOLA.SAMPLES` data set. You should see a member called `@@README` as well as a many more members in the data set.
- ☐ Browse the *`hlq.version`*`.WOLA.MODULES` data set. You should see a long list of members, some starting with `BBG*` and some with `BBO*`.

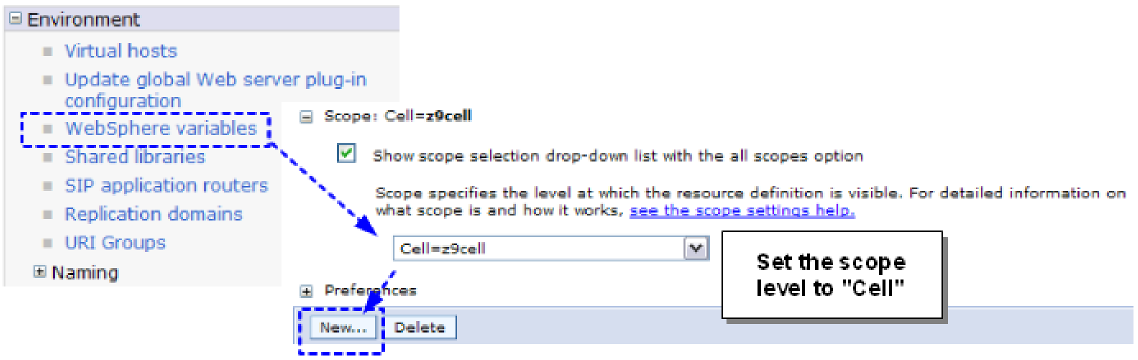
Create cell-level WOLA "enable adapter" environment variable

Overview:

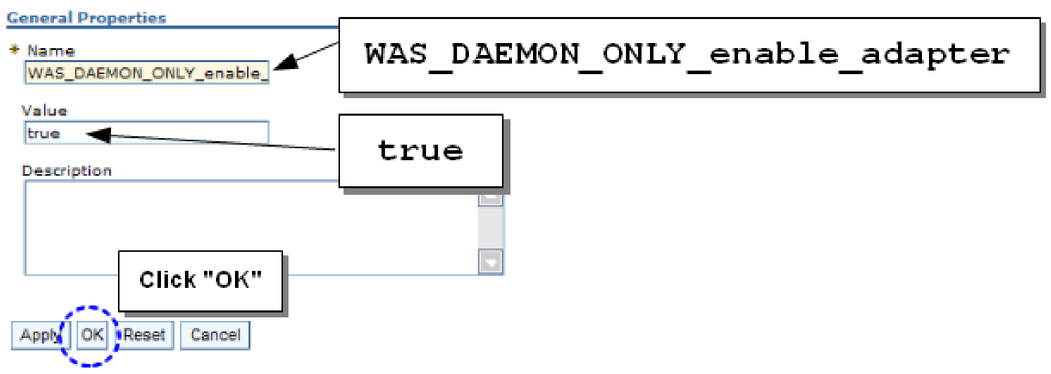
The WAS z/OS cell[5] requires a cell-level variable to enable WOLA. Think of this variable as a kind of "switch" you turn on for those cells you wish to use WOLA, and you leave off for those you do not wish to use WOLA.

Do the following:

- ☐ Log onto the Administrative Console for the cell.
- ☐ Begin the process of creating a cell-scoped environment variable:


Environment
Virtual hosts
Update global Web server plug-in configuration
WebSphere variables
Shared libraries
SIP application routers
Replication domains
URI Groups
Naming
Scope: Cell=z9cell
Show scope selection drop-down list with the all scopes option
Scope specifies the level at which the resource definition is visible. For detailed information on what scope is and how it works, see the scope settings help.
Cell=z9cell
Set the scope level to "Cell"
Preferences
New...
Delete

- ☐ Create a variable called **`WAS_DAEMON_ONLY_enable_adapter`** with a value of **`true`**:



- ☐ Save and synchronize the change.

Create cell-level WOLA "identity propagate" environment variable

Overview:

One more cell-level environment variable is needed. This allows the CICS task user ID to be propagated into the WAS environment. This is not needed if you do not plan to use WOLA with CICS[6] [7].

5 Either Network Deployment or Standalone Server.

6 If you think you *might* use CICS in the future, then add this. It does no harm. It is *not* a security exposure as access to the WAS z/OS server is controlled not by this environment variable but by the SAF `CBIND` profile you will create later.

7 Using WOLA with CICS will work without this variable only if the registration is done with `SEC=N` set. If `SEC=Y` set then a `RC=8/RSN=21` error is thrown. The InfoCenter RC/RSN article (search: **`olaapis`**) states: *"If you set the reg_flag_C2Wprop bit (bit 29) to 1* [equivalent to `BBOC START_SRVR` with `SEC=Y`]*, ensure that the WebSphere environment variable, ola_cicsuser_identity_propagate, is set to 1.*

Do the following:

- ☐ Using the same process you used for WAS_Daemon_ONLY_enable_adapter, create another cell-level environment variable. This one should be defined as:

Variable *Name*	**`ola_cicsuser_identity_propagate`**
Variable *Value*	**`1`**

- ☐ Save and synchronize your change.

Stop and restart the entire cell

Overview:

The cell-scoped environment variable you created earlier will not take effect until you stop and restart the **entire cell**, *including the Daemon server(s).*

Do the following:

- ☐ Stop all the application servers, Node Agents and Deployment Manager for the cell.
- ☐ Stop the Daemon servers (this is a very important step).
- ☐ Restart the servers in the cell.
- ☐ In the held output for the *Daemon*, check to make sure the following message is present:

```
enable_adapter: 1.
```

If you see that message, then the WOLA support is enabled in the cell.

Important! Do not proceed if you can't find this message. No further work can be accomplished until you successfully enable WOLA in the cell and see that "enable_adapter: 1" message.

If you do not see it, go back and carefully check the spelling of the variable:

`WAS_DAEMON_ONLY_enable_adapter`

Check also that it is scoped at the cell level, and that the changes have been successfully saved and synchronized out to all the nodes.

- ☐ If you set the `ola_cicsuser_identity_propagate` variable, then in the the held output for the target server controller region, check for the following message[8]:

```
BBOM0001I ola_cicsuser_identity_propagate: 1.
```

- ☐ Finally, if you are running at WAS z/OS Version 8.0.0.1 or above, check for the following messages in the target server controller region[9]:

```
BBOO0393I OLA API VECTOR ADDR/LEN: <two 16 byte hex strings>
BBOO0394I OLA API VECTOR VERSION/LEVEL: <two 16 byte hex strings>
BBOO0395I OLA API VECTOR CREATE DATE/TIME: <date and time stamp>
```

Install the `ola.rar` resource adapter and create a connection factory

Overview:

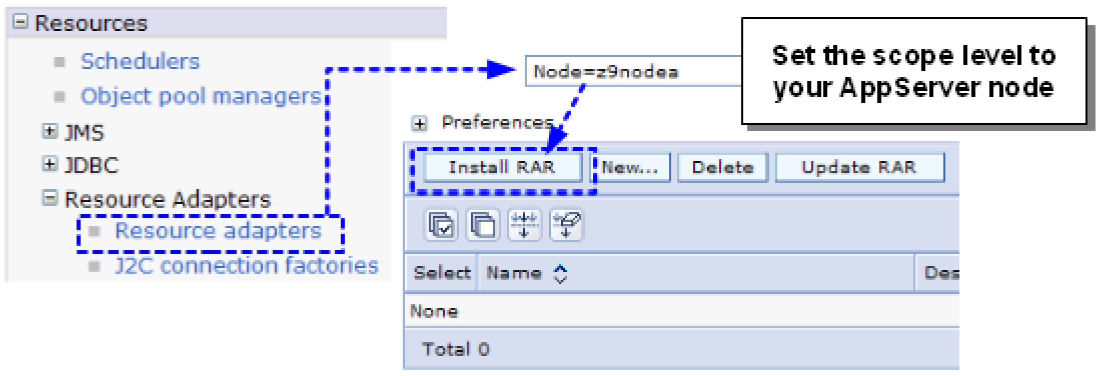
WOLA provides a JCA resource adapter that is used by Java applications seeking to use WOLA to communicate with external address spaces.

Do the following:

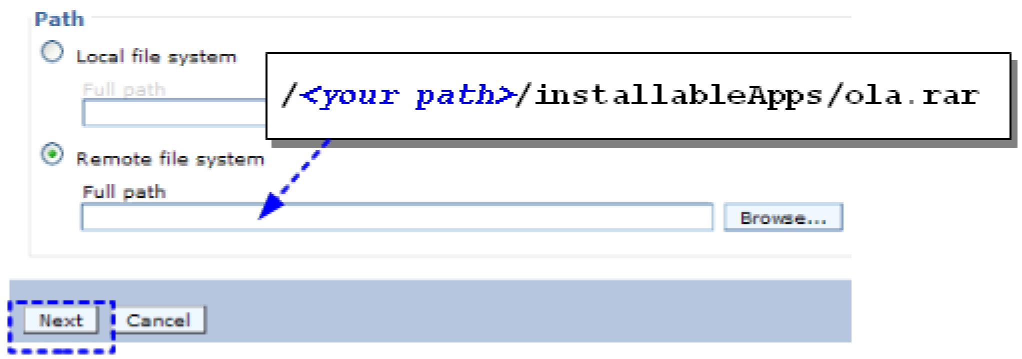
- ☐ Locate the `/installableApps` directory[10], either under `/DeploymentManager` or `/Appserver`. Note the presence of the file **`ola.rar`** in that directory.

8 This is a way of validating that synchronization of the earlier changes made their way out to the server. Failure to synchronize (for whatever reason) will prevent WOLA from working properly. You should make certain synchronization completed successfully.
9 This is a way of validating that WOLA is active in the application server you wish to use for WOLA testing. But this is only valid for 8.0.0.1 and above. Below that level check for proper synchronization, then continue with validation testing.
10 For V7, V8 or V8.5.

☐ In the Admin Console, do the following:

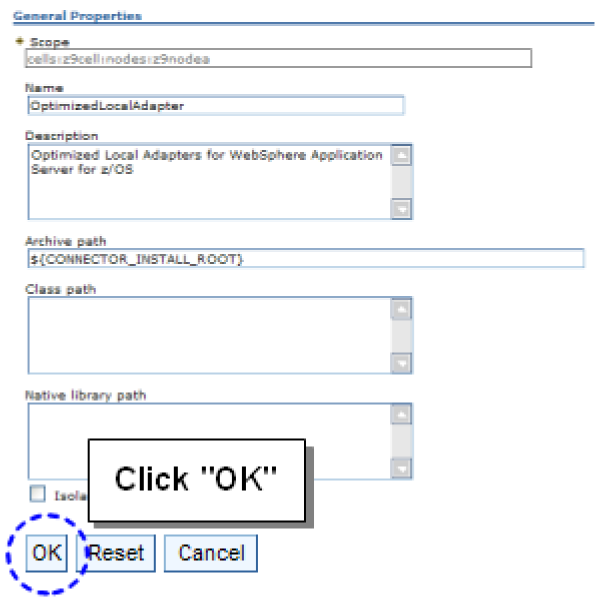


☐ Then:



/*<your path>*/installableApps/ola.rar

☐ Then:



Scope: cells:z9cell:nodes:z9nodea

Name: OptimizedLocalAdapter

Description: Optimized Local Adapters for WebSphere Application Server for z/OS

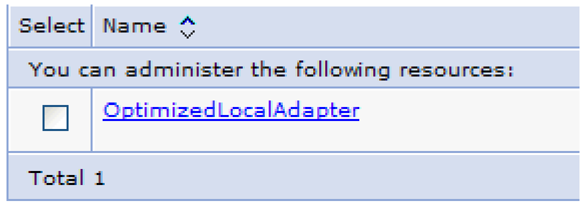
Archive path: ${CONNECTOR_INSTALL_ROOT}

Class path:

Native library path:

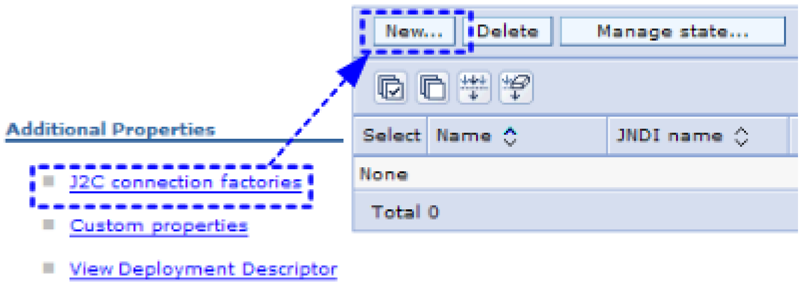
☐ Isola

☐ You should then see the Resource Adapter created:

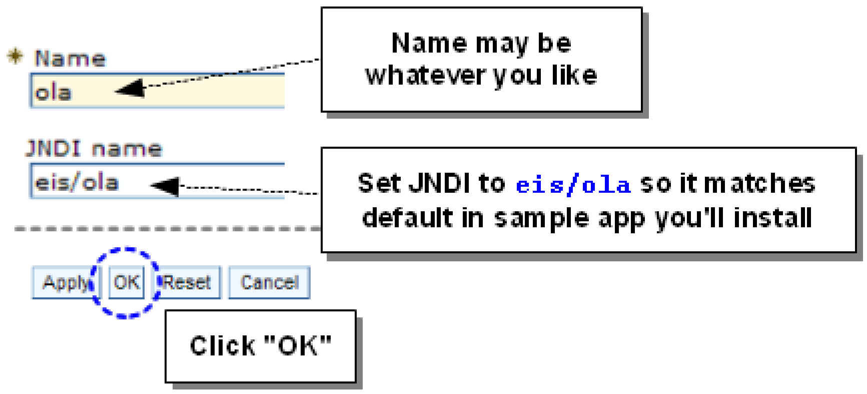


Select	Name
You can administer the following resources:	
☐	OptimizedLocalAdapter
Total 1	

- ☐ Click on that link and then do the following:



- ☐ And then:


Name
ola
Name may be whatever you like
JNDI name
eis/ola
Set JNDI to eis/ola so it matches default in sample app you'll install
Apply
OK
Reset
Cancel
Click "OK"

- ☐ Save and synchronize your changes.
- ☐ Go and review "Update of ola.rar file" on page 73. With new maintenance you *may* need to update the installed RAR file[11]. A useful WSADMIN script file is supplied to do that.

Install sample EAR file

Overview:

To use WOLA you will need an application in the WAS z/OS server. The WOLA function ships with a sample application that may be used for validation. Two EAR files are supplied; which you should use is a function of what level of WAS z/OS you're at:

V7.0.0.4 to V7.0.0.11[12]	`OLASample1.ear`
V7.0.0.12 or above	`OLASample2.ear`
V8 or V8.5 (any fixpack)	`OLASample2.ear`

Further, where those sample EAR files are located is a function of the WAS z/OS version level:

Version 7	`/<product_file_system_mount>/mso/OLA/samples`
Version 8 or 8.5	`/<product_file_system_mount>/util/zos/OLASamples/`

However, once located the act of deployment is relatively simple.

Do the following:

- ☐ From the information in the "Overview" section just above, determine the location and file name of the EAR file appropriate for your level of WAS z/OS.

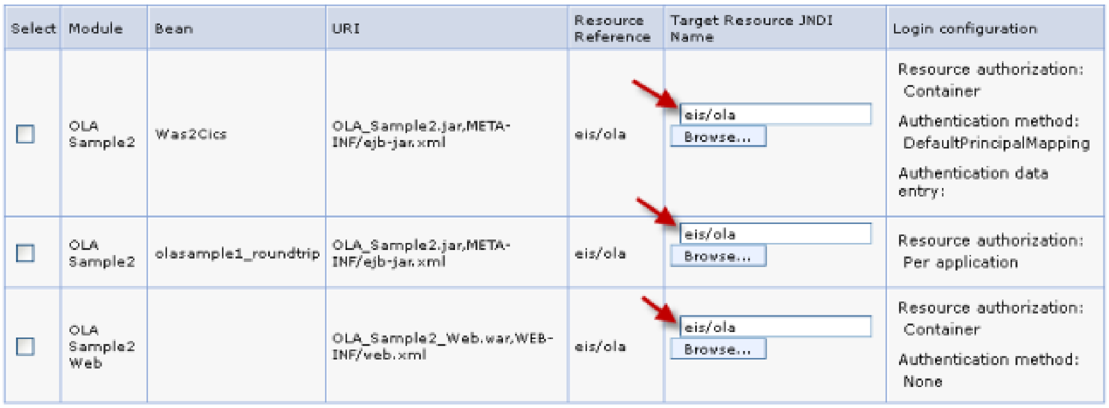
11 Or may not. The fixpack release notes will indicate this. It depends on whether the contents of the `ola.rar` file have been updated as part of the maintenance included in the fixpack.

12 If you're running WAS z/OS at a level between 7.0.0.4 and 7.0.0.11, you should give serious consideration to updating your level of WAS z/OS to the most recent fixpack for Version 7. That is particularly important if you're running with CICS TS 4.2 and wish to use WOLA. WOLA and CICS TS 4.2 requires a minimum of WAS z/OS V7.0.0.21.

☐ Using the WAS z/OS Administrative Console, deploy the application as you would any EAR file ... with the following two notes:

Resource References

The application has several "resource references" that will need to be mapped to the WOLA JCA resource adapter connection factory JNDI you created earlier:



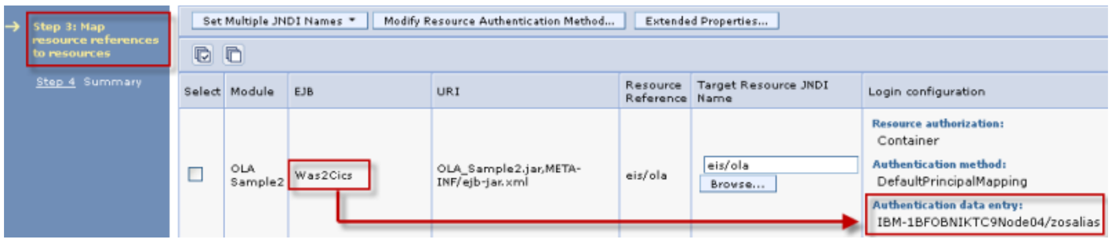
Select	Module	Bean	URI	Resource Reference	Target Resource JNDI Name	Login configuration
☐	OLA Sample2	Was2Cics	OLA_Sample2.jar,META-INF/ejb-jar.xml	eis/ola	eis/ola Browse...	Resource authorization: Container Authentication method: DefaultPrincipalMapping Authentication data entry:
☐	OLA Sample2	olasample1_roundtrip	OLA_Sample2.jar,META-INF/ejb-jar.xml	eis/ola	eis/ola Browse...	Resource authorization: Per application
☐	OLA Sample2 Web		OLA_Sample2_Web.war,WEB-INF/web.xml	eis/ola	eis/ola Browse...	Resource authorization: Container Authentication method: None

Problem in 8.5.0.0 OLASample2.ear

The `OLASample2.ear` file shipped with 8.5.0.0 contained a minor definition problem, but that problem has a relatively easy workaround.

Note: If you're at V7, V8.0 (any fixpack), or 8.5.0.1 *or above* you do ***not*** need to concern yourself with this step.

The "problem" shows itself like this:

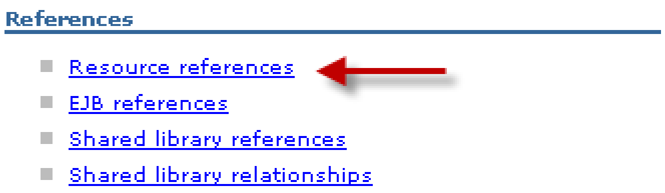


Step 3: Map resource references to resources
Step 4 Summary
Set Multiple JNDI Names
Modify Resource Authentication Method...
Extended Properties...

Select	Module	EJB	URI	Resource Reference	Target Resource JNDI Name	Login configuration
☐	OLA Sample2	Was2Cics	OLA_Sample2.jar,META-INF/ejb-jar.xml	eis/ola	eis/ola Browse...	Resource authorization: Container Authentication method: DefaultPrincipalMapping Authentication data entry: IBM-1BFOBNIKTC9Node04/zosalias

On "Step 3" of the application deployment, that "Authentication data entry" value of `IBM-1BFOBNIKTC9Node/zosalias` should *not* be there. If you see that with the `OLASample2.ear` you're deploying, then you should perform the next few steps to override that alias. If you *don't* see that alias, then you can skip the next few bullets and complete the application deployment as usual.

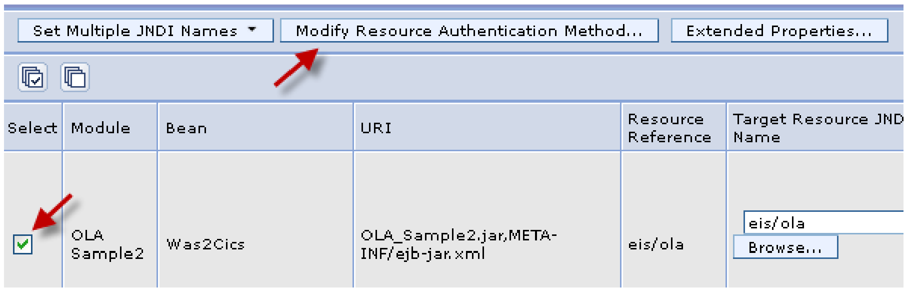
- Deploy the application. Save and synchronize the changes.
- Navigate to the list of deployed applications and click on the link for the `OLASample2` application.
- Click on the "Resource References" link:



References

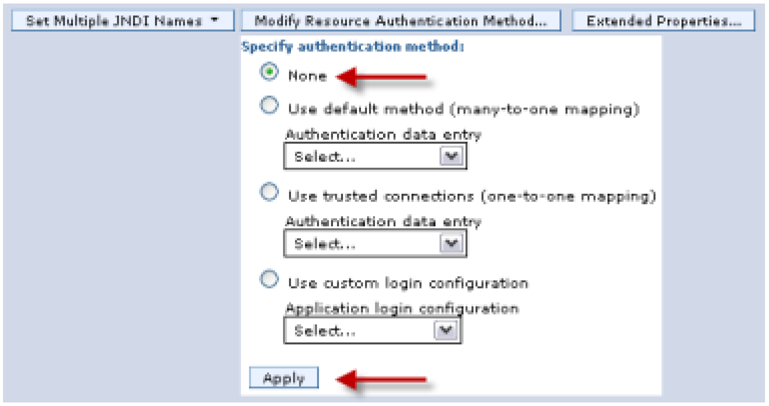
- Resource references
- EJB references
- Shared library references
- Shared library relationships

- Check the box next to the "Was2Cics" bean, then click on the "Modify Resource Authentication Method" button:



Select	Module	Bean	URI	Resource Reference	Target Resource JND Name
✓	OLA Sample2	Was2Cics	OLA_Sample2.jar,META-INF/ejb-jar.xml	eis/ola	eis/ola

- Select the "None" radio button and click the "Apply" button (important but easily overlooked step)



Specify authentication method:

[✓ None]

Use default method (many-to-one mapping)

Authentication data entry: Select...

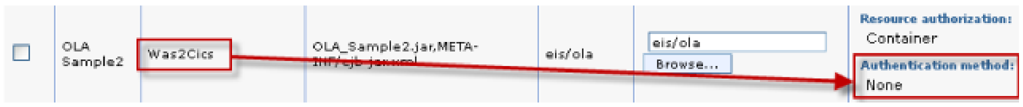
Use trusted connections (one-to-one mapping)

Authentication data entry: Select...

Use custom login configuration

Application login configuration: Select...

- Then check the "Login Configuration" column for the "Was2Cics" bean row in the Admin Console to make sure "None" shows for the "Authentication method":



☐	OLA Sample2	Was2Cics	OLA_Sample2.jar,META-INF/ejb-jar.xml	eis/ola	eis/ola	Resource authorization: Container Authentication method: None

- Click the "OK" button, then save and synchronize the changes.

☐ Save and synchronize any changes not yet saved and synchronized.

☐ From the Admin Console, start the application and insure it goes active.

Status checkpoint

At this point you've enabled WOLA in the WAS z/OS cell, and installed the WOLA JCA resource adapter with a defined connection factory (CF) and deployed the sample application.

Now you're ready to have an external address space register into the WAS server using WOLA and validate basic operations.

- If you wish to use CICS with WOLA, go to page 14.
- If you wish to use WOLA with batch programs, go to page 58.

Enabling WOLA in CICS Region and Initial Validation

There is a handful of steps needed to enable the WOLA support in a CICS region. This section will focus on those steps and include the initial validation up through starting the CICS Link Server Task and registering into the WAS server.

"WOLA and CICS Usage Scenarios" starting on page 20 will cover exercises that illustrate Java and CICS programs communicating with WOLA.

FYI: Overview of the process

At a very high level the process of enabling WOLA in the CICS region involves:

- Customizing the WOLA sample jobs to update the CSD and install the WOLA programs and tasks
- Adding the WOLA module library to the `DFHRPL DD` in the CICS start procedure
- Performing a few security tasks

Success with this process is achieved when the CICS region starts with WOLA and you are able to start the WOLA link server task and register into the WAS z/OS server.

Customize sample CSDUPDAT member and submit

Overview:

The `CSDUPDAT` job will update the CSD of the CICS region and install the WOLA programs, tasks and transactions.

Do the following:

- ☐ Locate the `CSDUPDAT` job within your *`hlq.version`*`.WOLA.SAMPLES` data set.
- ☐ Customize the `JOB` card to match your local requirements.
- ☐ Customize the `STEPLIB DD` and `DFHCSD DD` cards to match your CICS installation.
- ☐ Customize the two `ADD GROUP() LIST()` statements so `LIST` matches your CICS installation.
- ☐ Submit and look for `RC=0`.

Customize sample DFHPLTOL member and submit

Overview:

The `DFHPLTOL` job compiles a few programs that assist in the starting of the WOLA components in the CICS region. `BBOACPLT` starts the Task Related User Exit (TRUE); `BBOACPL2` provides a means of passing commands to the BBOC control program through `INITPARM`[13].

Do the following:

- ☐ Locate the `DFHPLTOL` job within your *`hlq.version`*`.WOLA.SAMPLES` data set.
- ☐ Customize the `JOB` card to match your local requirements.
- ☐ Customize the `C.SYSLIB DD` lines to match your CICS installation.
- ☐ Customize the `L.SYSLIB DD` lines to match your CICS installation.
- ☐ Customize the `L.SYSLMOD DD` line so it points to your *`hlq.version`*`.WOLA.MODULES` data set (the compiled `DFHPLTOL` module will go there).
- ☐ Submit and look for `RC=0`.

13 Depending on what level of WAS z/OS you have and when you're reading this, a third program (`BBOACPL3`) may be present in the sample. That provides a way to start pass commands and parameters to the BBOC control program without having to live within the length limitations of `INITPARM`.

Customize sample OLAMAP and OLAUTIL members and submit

Overview:

The OLAUTIL sample is a 3270 program. The `OLAMAP` job compiles the screen map for OLAUTIL. The `OLAUTIL` job compiles the COBOL used when functions keys of the 3270 screen are pressed.

Do the following:

- ☐ Locate the `OLAMAP` job within your *`hlq.version`*`.WOLA.SAMPLES` data set.
- ☐ Customize the `JOB` card to match your local requirements.
- ☐ Customize the `KIXPROC` line to match your CICS installation.
- ☐ Customize the `MAP1` statement:
  - ○ Update `INDEX=` to match your CICS installation
  - ○ Update `MAPLIB=` to point to your *`hlq.version`*`.WOLA.MODULES` data set
  - ○ Update `DSCTLIB=` to match your CICS installation. Or, if you wish, allocate a small FB 80 PDS and name that data set on `DSCTLIB=`.
- ☐ Submit and look for `RC=0`.
- ☐ Locate the `OLAUTIL` job within your *`hlq.version`*`.WOLA.SAMPLES` data set.
- ☐ Customize the `JOB` card to match your local requirements.
- ☐ Customize the `MYPROCS` line to match your CICS installation.
- ☐ Customize the `CMP EXEC DFHYITVL` section:
  - ○ Update `INDEX=` to match your CICS installation
  - ○ Update `PROGLIB=` to point to your *`hlq.version`*`.WOLA.MODULES` data set
  - ○ Update `DSCTLIB=` to point to the data set you used for the `OLAMAP` job
  - ○ Update `AD370HLQ=` to match your Enterprise COBOL installation.
  - ○ Update `LE370HLQ=` to match your Language Environment (LE) installation.
- ☐ Go to the bottom and update the `LKED.SYSLIB` section"
  - ○ Point to your CICS installation `SDFHLOAD` data set
  - ○ Point to your Language Environment `SCEELKED` data set
  - ○ Point to your *`hlq.version`*`.WOLA.MODULES` data set.
- ☐ Submit and look for `RC=0`.

Customize sample OLACB01 member and submit

Overview:

The `OLACB01` job compiles a simple COBOL echo program. It won't be used in this section of the document. We have you compile it here simply because you're customizing and compiling the other jobs. It seemed a good place to fit this in.

Do the following:

- ☐ Locate the `OLACB01` job within your *`hlq.version`*`.WOLA.SAMPLES` data set.
- ☐ Customize the `JOB` card to match your local requirements.
- ☐ Customize the `MYPROCS` line to match your CICS installation.
- ☐ Customize the `CMP EXEC DFHYITVL` section to match what you did for the `OLAUTIL` job (both jobs are compiling COBOL so they both point to the same things).
- ☐ Submit and look for `RC=0`.

Update CICS region SIP member

Overview:

This step involves updating the CICS region system initialization parameter (SIP) member to include a pointer to the PLT compiled by the `DFHPLTOL` job. That job provided two PLT programs: one to initialize the Task Related User Exit (TRUE), and the other to allow starting the Link Server Task with `INITPARM`. In this section we will have the TRUE started with this PLT, but the Link Server Task will be started manually with a BBOC command.

Do the following:

- ☐ Locate the CICS region's data set in which the SIP member is included.
- ☐ Update the member to include the following line:

```
PLTPI=OL
```

- ☐ Save the member.

Update the CICS region JCL and concatenate to the DFHRPL DD

Overview:

The WOLA modules in your *`hlq.version`*`.WOLA.MODULES` data set is made available to the CICS region by concatenating that data set to the `DFHRPL DD` statement.

Do the following:

- ☐ Locate the CICS region JCL start procedure.
- ☐ Concatenate the following to `DFHRPL DD`:

```
//         DD DSN=hlq.version.WOLA.MODULES,DISP=SHR
```

Where *`hlq.version`* matches what you used earlier in this process.

- ☐ Save the member.

Review the WAS z/OS server CBIND profile for CICS region ID access

Overview:

The SAF `CBIND` profile -- specifically, the `CB.BIND.`*`<your_environment_value>`* profile[14] -- will control the ability of an outside address space such as CICS being being able to register into the WAS z/OS server[15]. If the CBIND for your WAS z/OS server does not allow at least `READ` for the ID trying to register, the WOLA register from CICS will fail with a RC=12, RSN=14.

Therefore, you must insure the CICS region ID has `READ` to the profile.

Do the following:

- ☐ Check with your security administrator to see what format of `CBIND` was created for your WAS z/OS environment.

Note: The SAF class will be CBIND and the profile will be `CB.BIND.`*`<some_value>`*. What the value for *`<some_value>`* in your environment is what your security administrator must determine.

The format will one of the following, depending on how your cell was built:

(1) `CB.BIND.`*`<server_cluster_transition>`*`.**`
(2) `CB.BIND.`*`<saf_prefix>`*`.**`
(3) `CB.BIND.`*`<cell_identifier>`*`*`

The first is the format used if the generated "BRAK" job was used and no SAF prefix was specified; the second is the format if the generated BRAK job was used and a SAF prefix was provided; and the third is if you used a job that created "generic" profiles.

Determine what is used for your environment, then issue the command to update the access list for that CBIND profile as illustrated next.

14 There is also a `CB.`*`<your_environment>`* `CBIND` profile, but that is different and does not apply to this specific topic.
15 Register in either with the `BBOC START_SRVR` command or with the `BBOA1REG` API.

- ☐ Check the Universal Access (UACC) setting for that profile. If `UACC=`**READ**, then no further action is needed as the CICS region ID will have `READ` via `UACC=READ`.
- ☐ However, if `UACC=`**NONE**, then update the access list for the `CBIND` profile with a command[16] like what's shown here. Modify according to your specific `CBIND` profile:

```
PERMIT <your_CBIND_profile> CLASS(CBIND) ID(<CICS ID>) ACCESS(READ)
SETROPTS RACLIST(CBIND) REFRESH
```

Stop and restart the CICS region and verify WOLA components

Overview:

The changes made to the CICS definitions require a stop and restart of the CICS region. Validation of the WOLA updates can be seen in the CICS region output.

Do the following:

- ☐ Stop the CICS region.
- ☐ Restart the CICS region.
- ☐ Look in the `MSGUSR` output of the region for the following[17]:

```
Install for group BBOACSD has completed successfully.
  :
Install for group BBOASAMP has completed successfully.
```

- ☐ Then look in the `BBOOUT` output of the region for the following[18]:

```
Enabling BBOATRUE exit.
Return Code: 0 Reason Code: 0
```

Log into CICS region and use BBOC[19] to start the Link Server Task and register into server

Overview:

The BBOC control transaction is one installed with the CSDUPDAT job. One BBOC command is `START_SRVR`, which starts the Link Server Task and registers into the named WAS z/OS server.

Note: At this point we are illustrating the simplest means of starting the Link Server Task. However, it is possible to have the Link Server Task started at CICS startup. See "Appendix B: Starting Link Server Task at CICS Startup" starting on page 66 for details on the different ways this may be accomplished.

Do the following:

- ☐ Make sure the intended WAS z/OS target server is started
- ☐ Log into the CICS region
- ☐ Clear the screen, then enter the following command *as one command line*:

```
BBOC START_SRVR RGN=CICSREG DGN=aaaaaa NDN=bbbbbb SVN=cccccc SVC=*
                                   MNC=1 MXC=10 TXN=N SEC=N REU=Y
```

Where:

- `aaaaaa` is your *cell* SHORT name
- `bbbbbb` is the *node* SHORT name
- `cccccc` is the *server* SHORT name

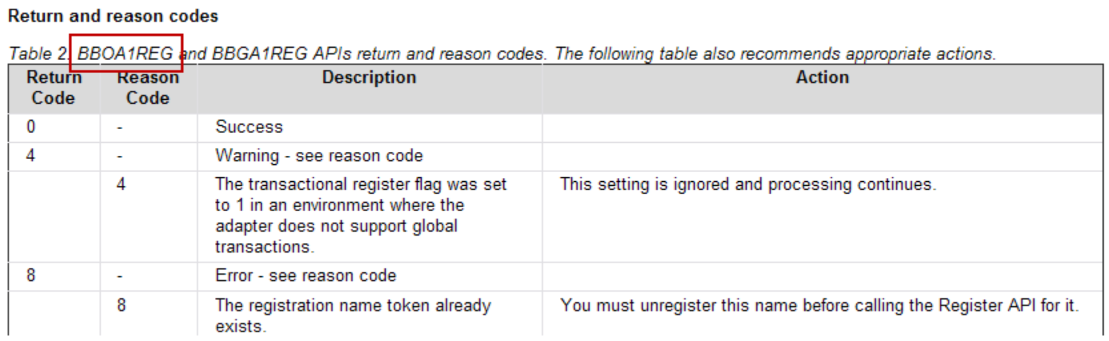
16 For a security product other than IBM RACF, modify the command to meet the equivalent command for the security product you use.
17 These messages indicate the `CSDUPDAT` job effectively updated the CICS CSD and the CICS region was able to read the program modules from the concatenation to `DFHRPL`.
18 These messages indicate the `DFHPLTOL` job effectively updated the PLT, the addition of `PLTPI=OL` to the region SIP has taken effect, and the WOLA Task Related User Exit (TRUE) has successfully started.
19 In the InfoCenter, search on the string **`rdat_cics`**. That article provides parameters and syntax for the BBOC command.

☐ The following message is positive confirmation the Link Server Task was started and a registration into the target WAS z/OS was successful:

```
BBOA8000I Start server completed successfully.
```

If you do not see that message, then the InfoCenter article located by searching for **cdat_olaapis** provides a reference for return code / reason code pairs.

Look for the BBOA1REG return code section:



Return and reason codes

Table 2. BBOA1REG and BBGA1REG APIs return and reason codes. The following table also recommends appropriate actions.

Return Code	Reason Code	Description	Action
0	-	Success	
4	-	Warning - see reason code	
	4	The transactional register flag was set to 1 in an environment where the adapter does not support global transactions.	This setting is ignored and processing continues.
8	-	Error - see reason code	
	8	The registration name token already exists.	You must unregister this name before calling the Register API for it.

Some common errors you might encounter at this point in your validation:

RC=**8**,RSN=**8**	You tried to issue the command again after it was successful the first time. The error is "registration name token already exists."
RC=**12**,RSN=**10**	The DGN= value of the command did not match the cell short name, or the Daemon for the cell is not started.
RC=**12**,RSN=**16**	The NDN= value and / or the SVN= value of the command did not properly match the node and server short names.
RC=**12**,RSN=**24**	The SVN= value of the command is correctly spelled, but the server itself is not started.

Make whatever corrections necessary and re-issue the command until you get the message indicating success.

☐ Now go to the z/OS operator console and issue the following z/OS MODIFY command to display the registrations into your server:

```
F aaaaaa,DISPLAY,ADAPTER,DAEMONRGES
```

Where ***aaaaaa*** is the JOBNAME for the target WAS z/OS application server.

You should see something like this:

```
BBOA0007I: SHOWING REGISTRATIONS FOR DAEMON GROUP:
BBOA0003I: Name          Jobname  SWT TL Min  Max  Act  State
BBOA0004I: CICSREG       CICS     000 00 0001 0010 0001 00
BBOA0004I: $WASDEFAULT$ CICS     000 00 0000 0001 0000 00
BBOO0188I END OF OUTPUT FOR COMMAND DISPLAY,ADAPTER,DAEMONRGES
```

Where the highlighted "CICSREG" is the registration you created with the BBOC command. This is another validation of the successful registration into WAS z/OS.

Other BBOC commands[20] issued in CICS terminal session

Overview:

The objective of this section is to show some other BBOC commands and how they operate

Do the following:

☐ From the CICS terminal session, clear the screen and then enter the following:

```
BBOC STOP_SRVR RGN=CICSREG
```

20 In the InfoCenter, search on the string **rdat_cics**. That article provides parameters and syntax for the BBOC command.

You should first see a message indicating an unregister command is underway, then a message indicating the server was stopped successfully.

- ☐ Clear the screen and enter the following command:

```
BBOC STOP_TRUE
```

You should see a message indicating "Exit disabled Successfully."

- ☐ Clear the screen and enter the following command[21]:

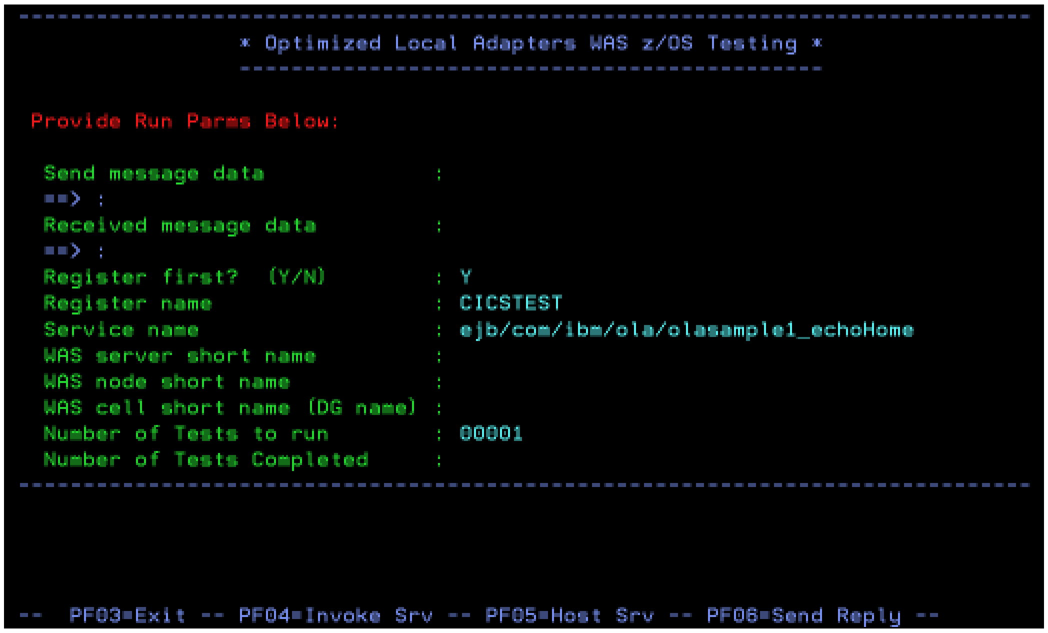
```
BBOC START_TRUE
```

You should see a message indicating "Exit enabled Successfully.."

- ☐ Clear the screen and enter the following command:

```
OLAUTIL
```

You should see a 3270 application screen[22] that looks like this:



```
--------------------------------------------------------------------------------
                  * Optimized Local Adapters WAS z/OS Testing *
                  ---------------------------------------------

 Provide Run Parms Below:

  Send message data                :
  ==> :
  Received message data            :
  ==> :
  Register first?  (Y/N)           : Y
  Register name                    : CICSTEST
  Service name                     : ejb/com/ibm/ola/olasample1_echoHome
  WAS server short name            :
  WAS node short name              :
  WAS cell short name (DG name)    :
  Number of Tests to run           : 00001
  Number of Tests Completed        :
--------------------------------------------------------------------------------



--  PF03=Exit -- PF04=Invoke Srv -- PF05=Host Srv -- PF06=Send Reply --
```

- ☐ Press F3 to exit that screen.

Status checkpoint

By making it this far you have validated the installation of the WOLA components into the CICS region and the successful start of the Link Server Task and registration into the WAS server.

The essential structure is now in place. In the next section you will begin to exercise the WOLA connection using the supplied sample programs.

21 This is the way the TRUE may be started manually. Recall that earlier you enabled the automatic start of the TRUE with a PLT.
22 The OLAUTIL application will be used in the next section to demonstrate WOLA operations.

WOLA and CICS Usage Scenarios

At this point in the document WOLA should be enabled in your WAS z/OS environment as well as the CICS region. You have validated the ability to register into the WAS z/OS server. Now you're ready to start using WOLA.

Issue BBOC START_SRVR to start the link server task and register into WAS z/OS

Overview:

At the conclusion of the previous section the Task Related User Exit (TRUE) was operating but the Link Server Task and registration into the WAS z/OS server was not. The first test will be outbound from WAS into CICS, and for that the Link Server Task will be used.

Do the following:

- ☐ Make sure the intended WAS z/OS target server is started
- ☐ Log into the CICS region
- ☐ Clear the screen, then enter the following command *as one command line*:

```
BBOC START_SRVR RGN=CICSREG DGN=aaaaaa NDN=bbbbbb SVN=cccccc SVC=*
                                 MNC=1 MXC=10 TXN=N SEC=N REU=Y
```

Where:

- aaaaaa is your *cell* SHORT name
- bbbbbb is the *node* SHORT name
- cccccc is the *server* SHORT name

- ☐ The following message is positive confirmation the Link Server Task was started and a registration into the target WAS z/OS was successful:

```
BBOA8000I Start server completed successfully.
```

Outbound WAS into CICS with OLACB01 as target service

Overview:

This test will involve a browser session into the deployed WAS sample application, which will drive a message over WOLA into CICS. The target CICS program will be the `OLACB01` echo program compiled earlier.

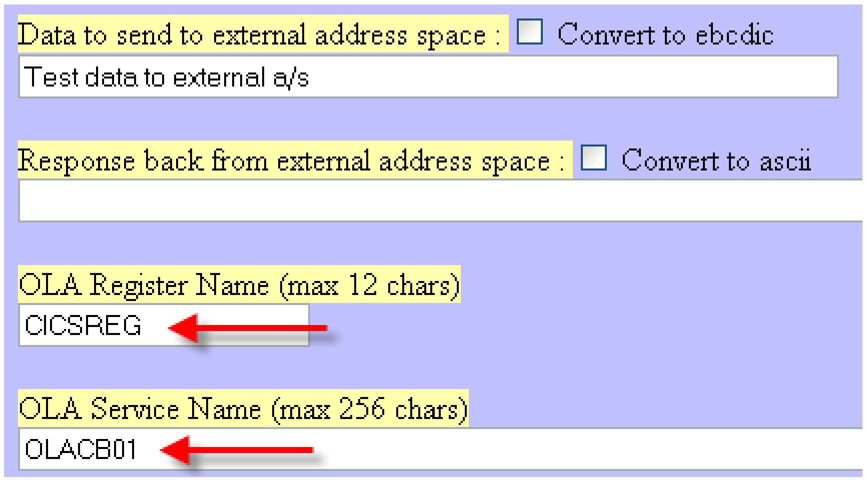
Do the following:

- ☐ From a browser, issue the following URL:

If `OLASample1.ear`	**`http://<host>:<port>/OLA_Sample1_Web/`**
If `OLASample2.ear`	**`http://<host>:<port>/OLA_Sample2_Web/`**

Where ***<host>*** and ***<port>*** are the what's appropriate for the WAS z/OS server in which the sample application is deployed.

- ☐ In the browser screen enter the following information[23]:



Data to send to external address space : ☐ Convert to ebcdic

Test data to external a/s

Response back from external address space : ☐ Convert to ascii

OLA Register Name (max 12 chars)

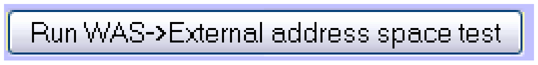
CICSREG

OLA Service Name (max 256 chars)

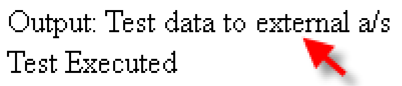
OLACB01

Leave all the other fields at their default values.

- ☐ Click on the "Run WAS → External address space test" button:



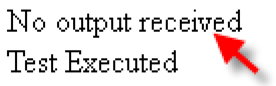
- ☐ The following in the browser screen is a sign of *success*:



Output: Test data to external a/s
Test Executed

`OLACB01` will simply echo back what it was sent.

However, if you see this:



No output received
Test Executed

Then there is a problem. Make sure the registration is in place (the MODIFY command shown earlier), and make sure you spelled the registration name and service name correctly.

Inbound CICS into WAS using OLAUTIL 3270 application in CICS

Overview:

The previous test was "outbound" ... that is, the Java program in WAS invoked the COBOL program in CICS. It is "outbound" from the perspective of WAS z/OS.

This test will show "inbound" ... that is, a program in CICS will invoke the Java program in WAS. It is "inbound" from the perspective of WAS z/OS.

The sample EAR file you deployed will work for this test. No additional EAR file deployments are required[24]. In CICS you will will the `OLAUTIL` 3270 program compiled earlier in this document.

Do the following:

- ☐ Open a 3270 terminal session and log onto your CICS region.
- ☐ Issue the command **`OLAUTIL`**

23 The "Register Name" is what was specified on the `BBOC START_SRVR` command `RGN=` parameter. The "Service Name" is the CICS program that will be invoked. In this case, `OLACB01` ... the echo program compiled earlier.

24 Because `OLASample1.ear` (or `OLASample2.ear`) were written to include Java that called outbound over the JCA resource adapter *as well as* providing a stateless EJB that would serve as the inbound target. In other words, *this* sample application was written to support *both* inbound and outbound. Please do not assume all WOLA applications support both inbound and outbound. The programming model is different for inbound compared to outbound, and it all depends on how the developer designed and constructed the application.

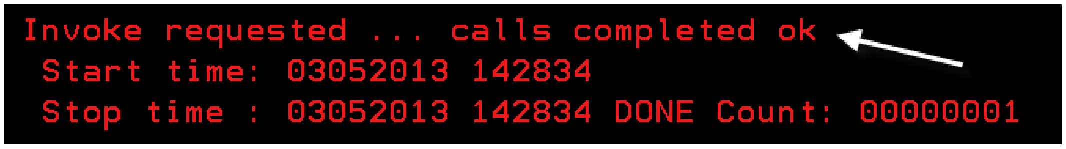
☐ In the resulting 3270 screen, follow the instructions as indicated by the numbered blocks:

```
Send message data             :
==> :  Message from CICS 1
Received message data         :
==> :
Register first?  (Y/N)        : N  2
Register name                 : CICSREG 3
Service name                  : ejb/com/ibm/ola/olasample1_echoHome   4
WAS server short name         :
WAS node short name           :
WAS cell short name (DG name) :
Number of Tests to run        : 00001  5
Number of Tests Completed     :
------------------------------------------------------------------
                6
  PF03=Exit -- PF04=Invoke Srv -- PF05=Host Srv -- PF06=Send Reply
```

Notes:

1. Provide whatever message string you wish to send
2. Specify "N" for register first. You created the registration with the `BBOC START_SRVR` command earlier.
3. Specify the registration name. You registered earlier with the `BBOC START_SRVR` command, and on that we suggested a value of `RGN=CICSREG`. Use that value here.
4. Leave the "Service Name" field unchanged. The default value is the correct value for `OLASample1.ear` or `OLASample2.ear`[25].
5. This field determines how many loops to execute. For the first test leave this at "1".
6. PF04 is what "invokes the service" in WAS by driving the request over WOLA to the target Java program[26]. Press the PF04 key.

☐ Look for the indication of success at the bottom of the CICS 3270 screen:



```
Invoke requested ... calls completed ok
 Start time: 03052013 142834
 Stop time : 03052013 142834 DONE Count: 00000001
```

☐ Change the "Number of tests to run" field to something like 100 and press PF04 again. You should see the "DONE Count" show 100 successfully done.

Outbound WAS into CICS using ATS Java application

Overview:

The OLASample1 and 2 applications are good validation samples, but they are limited by a single invocation per click of the invoke button. The "ATS Java application" is simply another EAR file that uses WOLA, but has the additional capability of invoking the target CICS service multiple times.

Do the following:

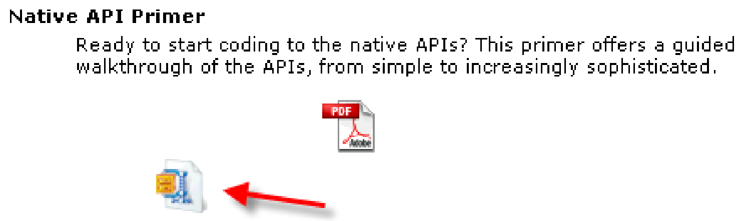
☐ Go to the WP101490 document

`http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101490`

25 When going inbound to WAS z/OS, the "Service Name" is the JNDI name assigned to the application module that serves as the target for the WOLA call. If you're curious, you can go into the Admin Console under the deployed application and click on "EJB JNDI Names." You'll find this "service name" value assigned to the `olasample1_echo` bean.

26 If you look in the `OLAUTIL` sample you'll see it uses the `BBOA1INV` API. You'll see that API in use in the "WOLA and Batch Usage Scenarios" section starting on page 58.

- ☐ Scroll down to the sub-section titled "Native API Primer" and locate the ZIP file that is part of that section:



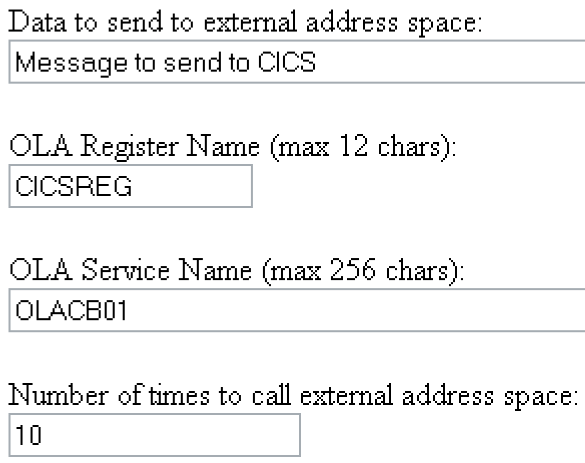
Native API Primer

Ready to start coding to the native APIs? This primer offers a guided walkthrough of the APIs, from simple to increasingly sophisticated.

- ☐ Download that ZIP file and extract from it the `ATSSample1-new.ear` file[27].
- ☐ Deploy that application into the same server as the `OLASample2.ear`[28]. Map the resource reference to the ola.rar connection factory JNDI ... `eis/ola` if you took the default.
- ☐ Save and synchronize the changes, then start the application.
- ☐ In a browser, issue the following URL:

  `http://<host>:<port>/ATSSample1Web/`

  Where ***`<host>`*** and ***`<port>`*** are the what's appropriate for the WAS z/OS server in which the sample application is deployed.
- ☐ Fill in the fields of the application:



Data to send to external address space:
Message to send to CICS

OLA Register Name (max 12 chars):
CICSREG

OLA Service Name (max 256 chars):
OLACB01

Number of times to call external address space:
10

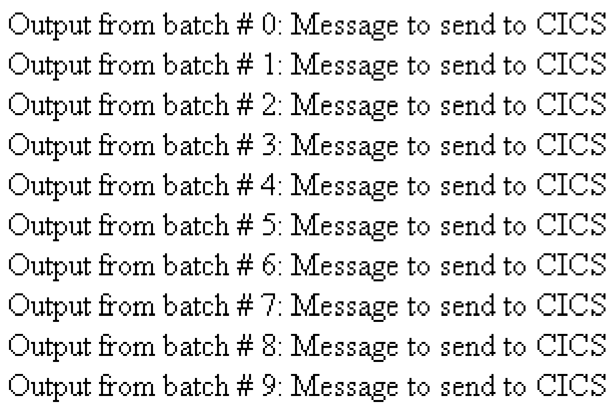
Where:

- "Data to send" may be whatever string you wish
- "OLA Register Name" is the registration created by the `BBOC START_SRVR` command earlier (`CICSREG` is what we illustrated)
- "OLA Service Name" is the target program in CICS. In this exercise that target is the `OLACB01` program.
- "Number of times to call" is loop count indicator.

- ☐ Click the "Run WAS → External Address space test" button:

You should see the results come back to the browser. It will look something like this:

27 Or `ATSSample1.ear` if you are using WAS z/OS V7.
28 Or `OLASample1.ear`, if that is what you deployed.



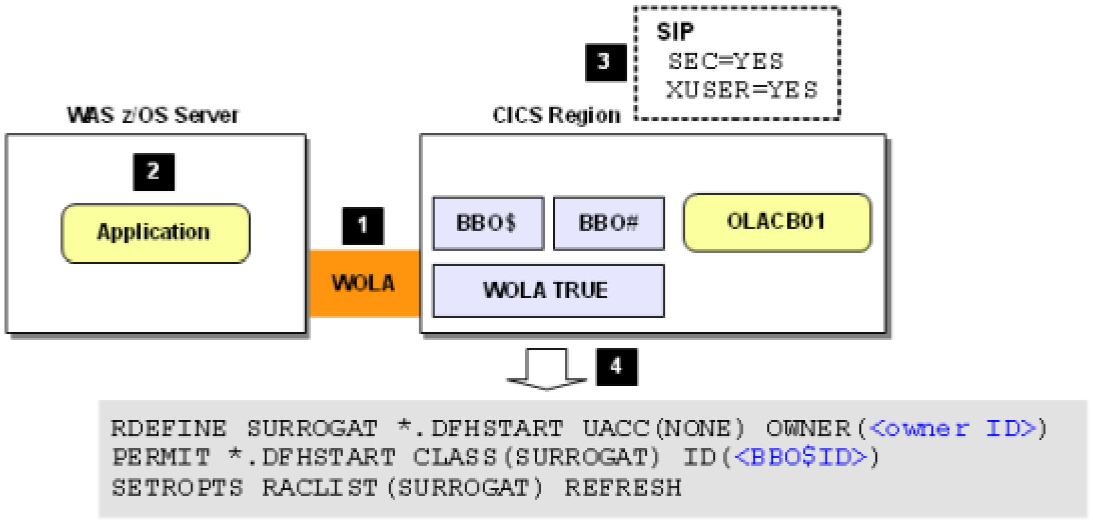
```
Output from batch # 0: Message to send to CICS
Output from batch # 1: Message to send to CICS
Output from batch # 2: Message to send to CICS
Output from batch # 3: Message to send to CICS
Output from batch # 4: Message to send to CICS
Output from batch # 5: Message to send to CICS
Output from batch # 6: Message to send to CICS
Output from batch # 7: Message to send to CICS
Output from batch # 8: Message to send to CICS
Output from batch # 9: Message to send to CICS
```

Optional - Propagation of Userid from WAS into CICS

Overview:

Up to this point the `BBOC START_SRVR` command was issued with **`SEC=N`**, which meant no identity was asserted from WAS over to CICS. The `BBO#` invocation task ran under the ID associated with the `BBO$` link server task. However, you may wish `BBO#` to run under the ID of the user thread in WAS. If so, that requires that ID be propagated across WOLA into CICS.

The following picture summarizes what's involved with making this happen. Step-by-step instructions will follow the picture.



```
RDEFINE SURROGAT *.DFHSTART UACC(NONE) OWNER(<owner ID>)
PERMIT *.DFHSTART CLASS(SURROGAT) ID(<BBO$ID>)
SETROPTS RACLIST(SURROGAT) REFRESH
```

Notes:

1. The `BBOC START_SRVR` command is started with `SEC=Y`. That allows security assertion. This requires the WAS cell to have the `ola_cicsuser_identity_propagate = 1` variable set as we spelled out under "Create cell-level WOLA "identity propagate" environment variable" starting on page 8.
2. The application authenticates the users so an ID can be asserted. The default OLASample application does not authenticate, but we can force an ID by coding an alias on the connection factory resource reference[29].
3. The CICS region has security enabled (`SEC=YES`) and `EXEC CICS START` checking enabled (`XUSER=YES`).
4. SAF profiles exist to allow the `BBO$` task ID to start `BBO#` with the asserted ID[30].

29 Unless you're running WAS z/OS V7, then you'll need an application that authenticates. See "WAS z/OS Version 7 and application with Basic Auth enabled" on page 29.

30 Then `BBO#`, operating under the asserted ID, will invoke the target service. The sample `OLACB01` has no security enabled against it, so any ID will be allowed. In a real-world setting with security enabled for the target program, the asserted ID's call of the program would also be checked to see if it had the authority.

Do the following:

- ☐ Take a moment and confirm where the `BBOC START_SRVR` command is being issued[31]. It may be manually (from CICS console), through PLT sample and `INITPARM=()`, through PLT sample 3, or from sequential terminal.

  Note: Up to this point we've shown issuing the `BBOC START_SRVR` manually from the CICS terminal session.

  However, if you read ahead you would have seen we demonstrate other ways to issue the `BBOC` command:

  - Using a PLT program and `INITPARM()` on page 35
  - Using a PLT program for longer `BBOC` commands on page 35
  - Using a sequential terminal input on page 36

  The point here is simply this: determine where the `BBOC` is being issued so you can modify that command and know you're changing the command that will take effect.

- ☐ Determine if the CICS region security is on (`SEC=YES` in SIP). If not, arrange for CICS security to be enabled.

- ☐ Determine if `XUSER=YES` is specified in the SIP for the region. If not, arrange for it to be added.

- ☐ Modify the `BBOC START_SRVR` command (in what method you're using to issue that command) so **`SEC=Y`**, **`REU=N`** and **`TRC=1`** is set[32] [33].

- ☐ Start the Link Server -- manually with `BBOC START_SRVR`, or using one of the other methods. If you get `RC=8/RSN=21` then check to make sure your cell has `ola_cicsuser_identity_propagate = 1` coded and in effect.

- ☐ Now issue the command:

  ```
  F aaaaaa,DISPLAY,ADAPTER,DAEMONRGES
  ```

  Where ***aaaaaa*** is the `JOBNAME` for the target WAS z/OS application server.

  You should see something like this:

  ```
  BBOA0007I: SHOWING REGISTRATIONS FOR DAEMON GROUP:
  BBOA0003I: Name          Jobname   SWT TL Min  Max  Act  State
  BBOA0004I: CICSREG       CICS      100 00 0001 0010 0001 00
  BBOA0004I: $WASDEFAULT$ CICS      000 00 0000 0001 0000 00
  BBOO0188I END OF OUTPUT FOR COMMAND DISPLAY,ADAPTER,DAEMONRGES
  ```

  Where the highlighted "CICSREG" is the `RGN=` value you specified on the `BBOC START_SRVR` command, and the highlighted "S" and "1" indicate that the `SEC=Y` value has taken effect.

- ☐ From a CICS region terminal, issue the command:

  ```
  CEMT INQ TAS(BBO$)
  ```

  You should see something like this:

  ```
  INQ TAS(BBO$)
  STATUS:  RESULTS - OVERTYPE TO MODIFY
   Tas(0000043) Tra(BBO$)           Run Tas Pri( 001 )
  ```

31 For this exercise two parameters will change. Knowing where the command is being issued will help make certain any changes are made in the right place.

32 Tracing (`TRC=1`) is *not* required to do security assertion. We're using it here because it's a great way to confirm the ID has been asserted. the trace will indicate what ID was asserted and what ID the `BBO#` transaction task was started under. Default tracing (nothing specified, which resolves to `TRC=0`, doesn't show that level of detail.

33 You may find the addition of `TRC=1` bumps you past limits of `INITPARM` if that's the method you're using. If so, remove `INITPARM` from your SIP and restart the CICS region. You'll get an error messaging indicating `INITPARM` not present, which causes no harm. Then issue `BBOC START_SRVR` from a CICS region terminal.

```
    Sta(SD) Use(uuuuuuuu) Uow(CB107F0340C3EBF2)
```

Where `uuuuuuuu` is the userid under which the `BBO$` link server task is running. You'll need that for the SAF `SURROGAT` profile step coming up.

☐ For the ID associated with the `BBO$` link server task to start `BBO#` under the asserted ID, a SAF `SURROGAT` profile update may be needed. Work with your security administrator to verify, create or update the following:

```
RDEFINE SURROGAT *.DFHSTART UACC(NONE) OWNER(<owner_ID>)
PERMIT *.DFHSTART CLASS(SURROGAT) ID(<BBO$_ID>)
SETROPTS RACLIST(SURROGAT) REFRESH
```

Where:

- `<owner_ID>` is whatever ID your security administrator chooses to use when creating the profile
- `<BBO$_ID>` is the ID under which the `BBO$` link server task runs

Note: The specific `SURROGAT` profile that will be sought is:

`<asserted_ID>.DFHSTART`

We show `*.DFHSTART` as a way of having one profile apply to many IDs that may come across. Use whatever security conventions are appropriate for your installation.

The important thing is to make sure no *other* `SURROGAT` profile matches before the one you intend.

Failure to have the `SURROGAT` set up properly will result in the following error in the `BBOOUT` held output:

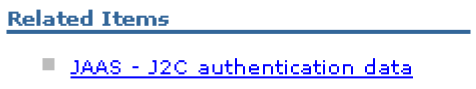
```
BBOA8600E An error occurred with CICS START TRANSID: BBO# RESP: 70 RESP2: 9.
```

The next step involves updates to get an ID to flow over to CICS[34]. How you accomplish that is dependent upon the level of WAS z/OS you're using:

If V7	You will need to use a copy of the `OLASample2.ear` application where Basic Authentication is enabled. See "WAS z/OS Version 7 and application with Basic Auth enabled" on page 29, then when installed, come back to this place in the document and proceed at the point *after* the alias is configured.
If V8 or V8.5	You may code an alias on the application module that performs the call, or you may use an application that authenticates. Coding an alias is fairly simple so that's what we'll illustrate here[35].

☐ Create an alias for the ID you intend to assert across WOLA into CICS:

- ○ In the Admin Console, go to *Resources* ⇨ *Resource Adapters* ⇨ *J2C connection factories*
- ○ Click on the link for the "ola" connection factory created earlier
- ○ Over to the right, click on the link for "JAAS - J2C authentication data":

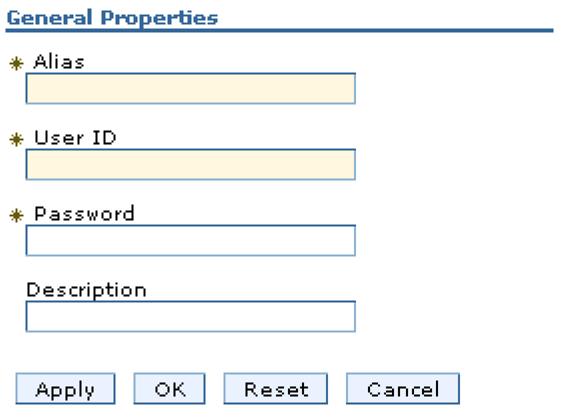


- ○ Click on the "New" button

34 If you use the sample applications as delivered, then no ID will flow. That will result in a `RESP=69, RESP2=8` error in CICS `BBOOUT`.

35 Although if you wish, you may use the copy of OLASample2.ear with Basic Auth we show being used for WAS z/OS V7 (page 29). It works with Version 8 or V8.5 as well. *However, please review the steps specified for V7 starting on page 29.* You will need to turn on application security and create an `EJBROLE` to make Basic Auth work, whether V7, V8 or V8.5.

- Provide a name for the Alias (whatever name you wish), and then provide the ID and password for the ID:



General Properties

* Alias

* User ID

* Password

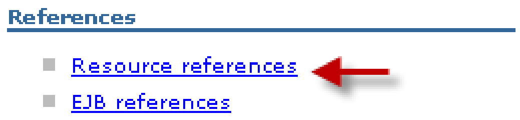
Description

When completed, click the "OK" button.

- Save and synchronize your changes.

☐ With the alias created, now you can go into the application and provide that alias to the Java module that makes the WOLA call. Do the following:

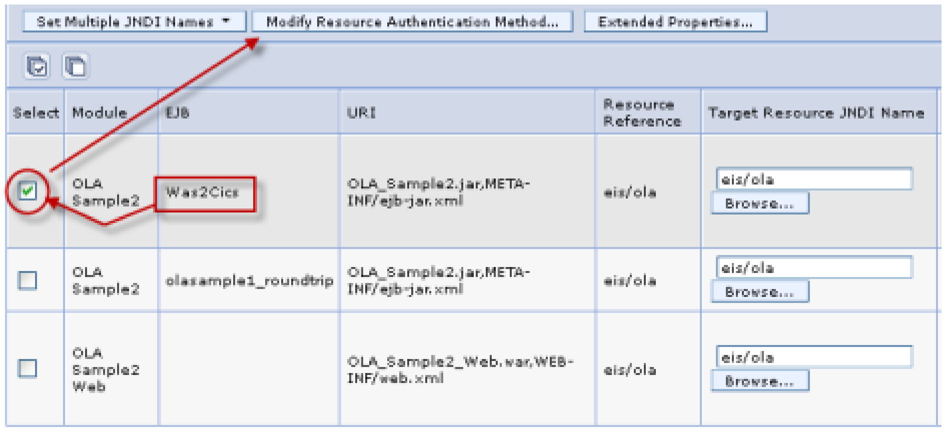
- In the Admin Console, go to *Applications* ⇨ *Application Types* ⇨ *WebSphere enterprise applications*
- Click on the link for the "OLASample2" application
- Click on the link for "Resource References":



References

- Resource references
- EJB references

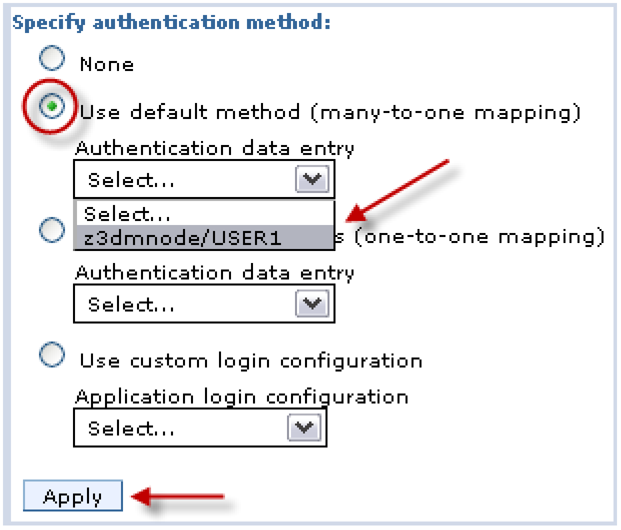
- Select the check box next to the "Was2Cics" EJB and then click on the "Modify Resource Authentication Method" button:



Set Multiple JNDI Names ▾
Modify Resource Authentication Method...
Extended Properties...

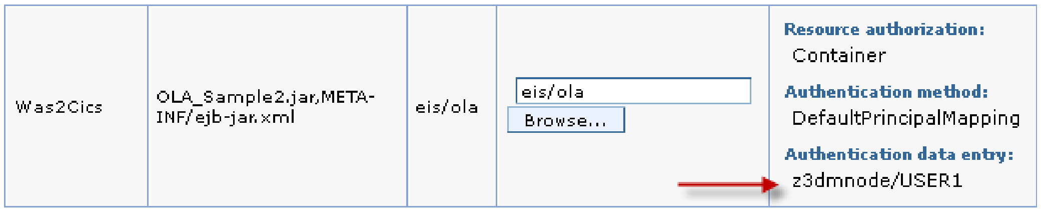
Select	Module	EJB	URI	Resource Reference	Target Resource JNDI Name
✓	OLA Sample2	Was2Cics	OLA_Sample2.jar,META-INF/ejb-jar.xml	eis/ola	eis/ola Browse...
☐	OLA Sample2	olasample1_roundtrip	OLA_Sample2.jar,META-INF/ejb-jar.xml	eis/ola	eis/ola Browse...
☐	OLA Sample2 Web		OLA_Sample2_Web.war,WEB-INF/web.xml	eis/ola	eis/ola Browse...

- Select the "Use default method" radio button, then select your alias:


Specify authentication method:
None
Use default method (many-to-one mapping)
Authentication data entry
Select...
Select...
z3dmnode/USER1
s (one-to-one mapping)
Authentication data entry
Select...
Use custom login configuration
Application login configuration
Select...
Apply

Then click the "Apply" button to place your alias on the EJB module resource reference.

- Make sure your alias appears on the module:



Was2Cics	OLA_Sample2.jar,META-INF/ejb-jar.xml	eis/ola	eis/ola Browse...	**Resource authorization:** Container **Authentication method:** DefaultPrincipalMapping **Authentication data entry:** z3dmnode/USER1

- Click "OK"
- Save and synchronize your changes

If you are using V7 and deployed the Basic Auth copy of the application, start back up from here

- ☐ Stop and restart WAS and CICS. This may not be strictly necessary at this point, but it's a good way to insure all the changes are properly enabled and any caching is cleared.
- ☐ Make sure the registration into WAS z/OS is in place with `SEC=Y` and `TRC=1`[36].
- ☐ Run application from browser just as you did before[37].
- ☐ Check the `BBOOUT` of your CICS region. You should see *something* like the following:

```
  :
BBOA8501I Calling Get Context API ... connhdl: 0000020009729bb000000001
BBOA8501I Get context req. completed. connhdl: 0000020009729bb000000001
BBOA8501I Userid propagated: <your asserted ID>
BBOA8501I Link transaction id propagated:
BBOA8501I Link request container id prop.: WAS_BBOA_REQ
BBOA8501I Link request container type: 0
BBOA8501I Link response container id prop.: WAS_BBOA_REQ
BBOA8501I Link response container type: 0
BBOA8501I Link tx/cmt/rb: 0/0/0
BBOA8501I Var len ctxt area: 163155A4
BBOA8501I Num var ctxts: 1
BBOA8501I Extended TX context: 0
BBOA8501I Correlator context: 163155B4
BBOA8501I WAS correlator id: ID=WebSphere Application Server for z/OS
 :
```

36 You can validate this with the `MODIFY` command shown on page 25.
37 And if using the V7 Basic Auth copy of the application, provide the ID and password you wish to assert across WOLA into CICS.

```
BBOA8501I WAS correlator data 2: INSTANCE=
BBOA8501I Instance data: 1478b863000000d0000000
BBOA8501I WAS correlator data 3:
BBOA8501I WAS application is NOT IN a Global Transaction.
BBOA8501I Security propagation ON. Userid requested: <your asserted ID>
BBOA8501I Container interface NOT requested. Use COMMAREA.
BBOA8501I Starting WAS adapters Link task ...
BBOA8501I Link transaction id: BBO#
BBOA8501I linkparms.aclnk_connhdl: 0000020009729bb000000001
BBOA8501I Starting with USERID <your asserted ID>
BBOA8501I Start server completed successfully.
BBOA8501I Calling Receive request Any API for '*'
BBOA8701I <++++++++++  ADAPTERS CICS LINK TASK START  +++  Thu Mar 14
18:41:03   ++>
BBOA8701I Initiated by server task number: 0000043
BBOA8701I Connected to WAS server Z9CELL/Z9NODEA/Z9SR01A
BBOA8701I Initiated with register name: CICSREG
BBOA8701I Initiated for service name: OLACB01
BBOA8701I Initiated with tsqname:
BBOA8701I Initiated with connection handle: 0000020009729bb000000001
BBOA8701I Initiated with trace level setting: 1
BBOA8701I Initiated with trace TDQ name: BBOQ
BBOA8701I Initiated under userid: <your asserted ID>
Data to be passed in a COMMAREA
Calling Get Data API with connhdl: 0000020009729bb000000001
Get data request completed
Copied linkparms to tsq: BBO0051C
EXEC CICS LINK to PROGRAM: OLACB01
EXEC CICS LINK to OLACB01  with Commarea Successful!
Calling Send Response API. connhdl: 0000020009729bb000000001
Calling Send Response, sending 25 bytes...
Send Response completed.
Releasing conn back to pool. connhdl: 0000020009729bb000000001
Release connection back to pool completed.
Return Code: 0 rsn Code: 0
Elapsed time: 0.000000 seconds
Elapsed CPU time: 0.000403 seconds
```

☐ Troubleshooting: if your test fails and in the `BBOOUT` you see:

```
BBOA8600E An error occurred with CICS START TRANSID: BBO# RESP: xx RESP2: y.
```

RESP	*RESP2*	*Cause and Correction*
69	8	Then the ID was not asserted across. If you look higher in the TRC=1 output you should see the "Security propagation ON. Userid requested:" message with no ID following. The probable cause is the alias was not present. Go back and check the alias is properly defined and all changes have been synchronized.
70	9	The `SURROGAT` update to `*.DFHSTART` was not in effect for the ID under which the `BBO$` link server task is running. Verify the ID for `BBO$`, verify it is provided `READ` to `*.DFHSTART`, and verify a `SETROPTS REFRESH` was issued.

WAS z/OS Version 7 and application with Basic Auth enabled

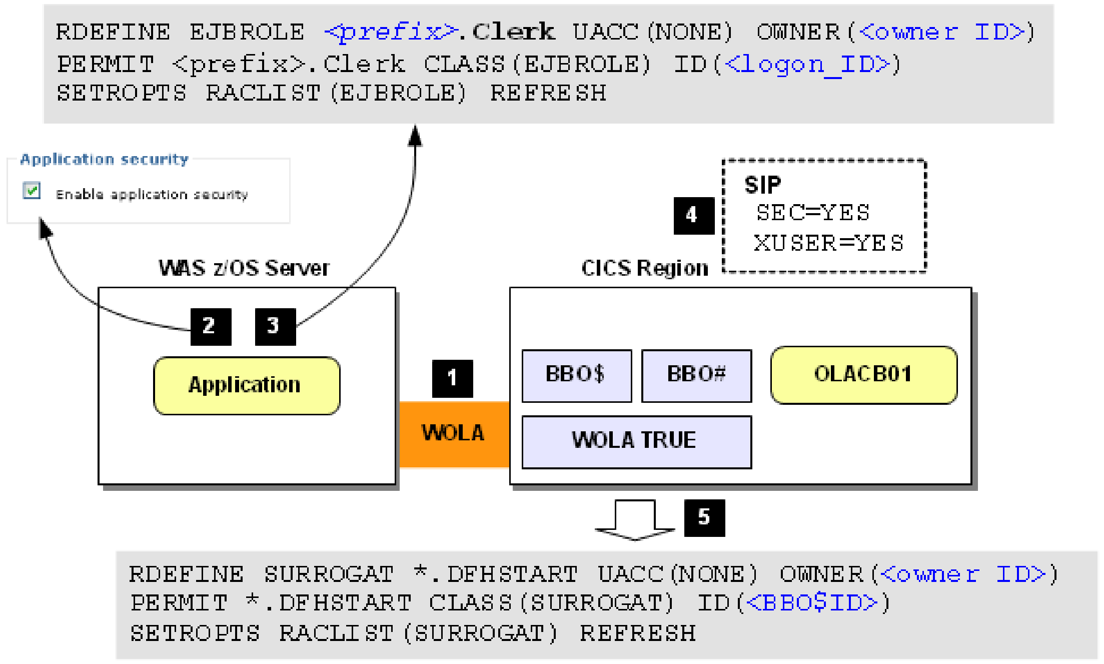
Overview:

During testing we discovered that WAS z/OS Version 7 did not assert the alias ID across the WOLA link to CICS. However, an application that performed authentication did. Demonstrating identity assertion with an authenticated ID is more "real world" than using an alias. We used an alias for the V8 and V8.5 exercise above simply because it's the easiest to set up.

To show identity assertion with WAS z/OS V7 we took the `OLASample2.ear` that comes with V7 and added Basic Authentication to it. Using that application you can validate identity assertion from WAS z/OS V7 over WOLA into CICS.

Note: If your level of WAS z/OS is Version 7.0.0.11 or below, then this `OLASample2.ear` application is *not applicable*. You really should update the level of maintenance for your copy of WAS z/OS to the most current, or at a minimum 7.0.0.21[38].

The overview picture is similar to before, but with two other components:



```
RDEFINE EJBROLE <prefix>.Clerk UACC(NONE) OWNER(<owner ID>)
PERMIT <prefix>.Clerk CLASS(EJBROLE) ID(<logon_ID>)
SETROPTS RACLIST(EJBROLE) REFRESH
```

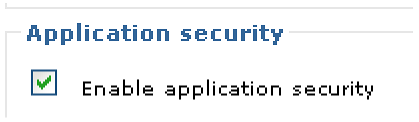
```
RDEFINE SURROGAT *.DFHSTART UACC(NONE) OWNER(<owner ID>)
PERMIT *.DFHSTART CLASS(SURROGAT) ID(<BBO$ID>)
SETROPTS RACLIST(SURROGAT) REFRESH
```

Notes:

1. The `BBOC START_SRVR` command is started with `SEC=Y`. That allows security assertion. This requires the WAS cell to have the `ola_cicsuser_identity_propagate = 1` variable set as we spelled out under "Create cell-level WOLA "identity propagate" environment variable" starting on page 8.
2. The cell has "Application security" enabled and that value synchronized out to the servers.
3. A SAF `EJBROLE` profile to check if the ID used to log on has access to the role.
4. The CICS region has security enabled (`SEC=YES`) and `EXEC CICS START` checking enabled (`XUSER=YES`).
5. SAF profiles exist to allow the `BBO$` task ID to start `BBO#` with the asserted ID[39].

Do the following:

- ☐ In the Admin Console, go to *Security* ⇨ *Global Security*
- ☐ Check the box "Enable application security":



Application security

✓ Enable application security

- ☐ Click "Apply," then save and synchronize the changes.

38 7.0.0.21 or higher is particularly important if you are using CICS TS 4.2. Fixpack 21 had significant fixes to allow the WOLA components to operate properly inside of CICS TS 4.2.

39 In this case the asserted ID is the logon ID.

- ☐ The application is going to check authorization based on an `EJBROLE` called "Clerk." Therefore, you need to create an `EJBROLE` profile in your SAF product. For RACF the commands would be:

```
RDEFINE EJBROLE <prefix>.Clerk UACC(NONE)
PERMIT <prefix>.Clerk  CLASS(EJBROLE) ID(aaaaa) ACCESS(READ)
SETROPTS RACLIST(EJBROLE) REFRESH
```

  Where:

  - ***`<prefix>`*** is the SAF profile prefix used for your cell[40]
  - ***`aaaaa`*** is the ID you will use when logging on to the application

  Note: EJBROLE definitions are *case sensitive*. The application is looking to match against `Clerk` with that exact mixed-case. Be sure to create the `EJBROLE` with a name that exactly matches the case for `Clerk`.

- ☐ Go to the WP101490 Techdocs website[41] and scroll down to the section titled "WOLA Quick Start Guide." There you will find a file named:

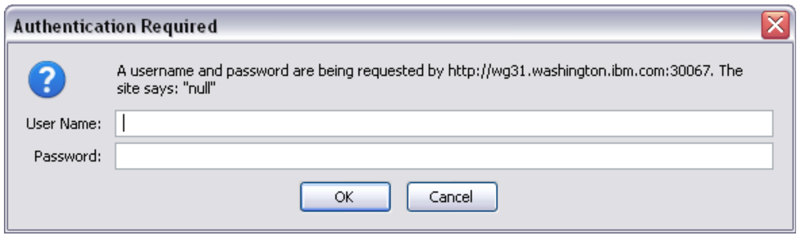
  `OLASample2.BasicAuth.v7.ear`

- ☐ Save that file to your workstation.
- ☐ In the Admin Console, uninstall any copy of `OLASample2.ear` you may have deployed.
- ☐ Deploy this "BasicAuth" copy of the EAR file into your server used for WOLA. It deploys in exactly the same way the non-authentication copies of the application deployed.
- ☐ Save and synchronize the changes.
- ☐ Start the application.
- ☐ Close all the browser instances and re-open fresh[42].
- ☐ From a browser, issue the following URL:

  `http://<host>:<port>/OLA_Sample2_Web/`

  Where ***`<host>`*** and ***`<port>`*** are the what's appropriate for the WAS z/OS server in which the sample application is deployed.

- ☐ You should see an authentication challenge:



Authentication Required

A username and password are being requested by http://wg31.washington.ibm.com:30067. The site says: "null"

User Name:

Password:

40 You can determine this in the Admin Console: go to *Security* ⇨ *Global Security*, then in the lower-right of that panel click on *Custom Properties*. Look for the custom property `com.ibm.security.SAF.profilePrefix`. The value assigned to that property is the prefix we are referring to here. If no prefix is specified, then your cell is operating without a prefix. In that case the `EJBROLE` would simply be `Clerk`.

41 URL: `http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101490`

42 The purpose of this is to clear any authentication cookies you may have; for example: the ID for your WAS Admin Console. The objective here is to have the application challenge you for authentication. If the application receives an authentication cookie from your browser, it will assume that's the ID to use and it won't challenge you.

- ☐ Provide the ID and password of the ID you wish to assert across WOLA into CICS.
- ☐ If the ID was accepted[43] then close the browser[44] and return to page 28 where you will pick back up where indicated on that page.

Optional - Use the CICS Channels and Containers sample OLACB02

Overview:

CICS "Channels and Containers" is a mechanism used to pass information to and from CICS. It first became available in CICS TS 3.1. Among other benefits, it overcomes the 32K size barrier of COMMAREA.

WOLA supports the use of Channels and Containers, but the degree of support is a function of the level of WAS z/OS you have:

Version 7.0.0.4 or above Version 8.0.0.0 through 8.0.0.4 Version 8.5.0.0 through 8.5.0.1	Limited to a single fixed channel name Limited to a single container of type BIT or CHAR	`OLACB02`

Version 8.0.0.5 or later
Version 8.5.0.2 or later
	Set channel name using supplied method
	Set channel type using supplied method
	One or more containers using mapped records
		`OLACB02`[45]
		`OLACB10`
		`OLACB11`
		`OLACB12`

In short, starting with 8.0.0.5 or 8.5.0.2 the support for channels and containers was improved considerably. The original support was effective but limited.

In this section we'll show the usage of channels and containers assuming the limited function and using the `OLASample2.ear` application and the `OLACB02` sample COBOL program for CICS[46].

Do the following:

- ☐ Locate the `OLACB02` sample source code:
  - ○ If V7, then if you followed the instructions under "Copy the WOLA samples to your samples data set" starting on page 64 then the `OLACB02` source deck should be in your samples data set.
  - ○ If V8 or V8.5 the `OLACB02` source deck should be in your sample data set. The `copyZOS.sh` shell script copied out all the supplied samples.
- ☐ Customize the `OLACB02` JCL to match your COBOL compiler environment and compile the `OLACB02` program into your `hlq.version.WOLA.MODULES` library data set[47].
- ☐ Start your CICS region, then log into the CICS region, clear the screen and then enter the following command *as one command line*:

```
BBOC START_SRVR RGN=CICSREG DGN=aaaaaa NDN=bbbbbb SVN=cccccc SVC=*
                                   MNC=1 MXC=10 TXN=N SEC=N REU=Y
```

Where:

- `aaaaaa` is your *cell* SHORT name
- `bbbbbb` is the *node* SHORT name
- `cccccc` is the *server* SHORT name

Note: That BBOC command is essentially the same as used before. In other words, the use of channel/containers does not require anything new or different on the command to start the Link Server or to register into the WAS server.

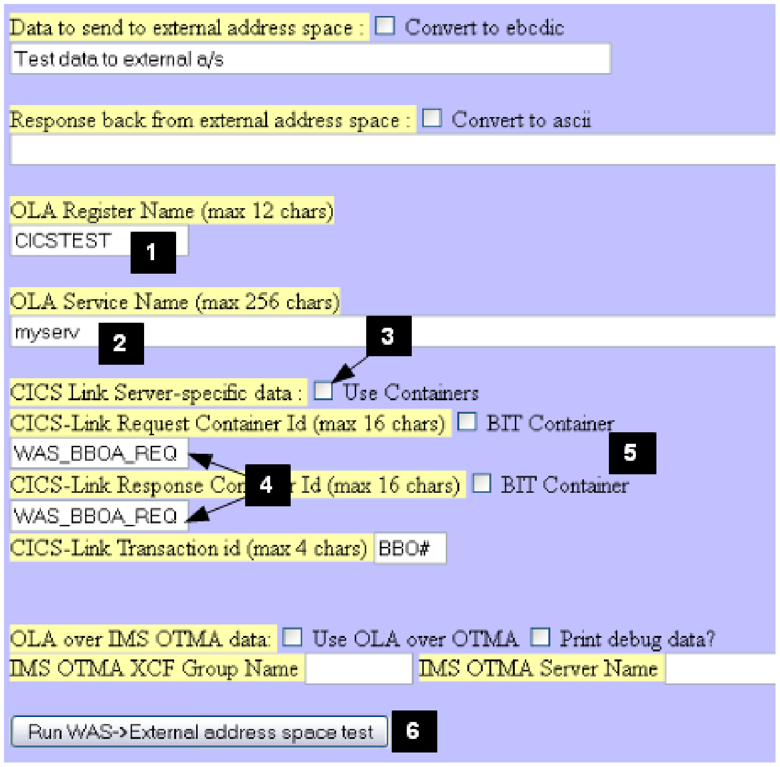
43 A browser "403" error would indicate the ID you provided did not have `READ` to the "Clerk" `EJBROLE`.
44 Don't actually drive the application into CICS at this point because a few other things are needed.
45 Using `OLACB02` with the newer level of support is possible. See "8.0.0.5 / 8.5.0.2 (or higher) and OLACB02 sample" on page 78.
46 If you have 8.0.0.5 or 8.5.0.2 then see "Improved Channels and Containers using 8.0.0.5 or 8.5.0.2" on page 75.
47 The `CSDUPDAT` job included a reference for `OLACB02`, so you should not need to update your CICS CSD.

- ☐ From a browser, issue the following URL:

  `http://<host>:<port>/OLA_Sample2_Web/`

  Where ***<host>*** and ***<port>*** are the what's appropriate for the WAS z/OS server in which the sample application is deployed[48].

- ☐ On the browser screen, see the notes that follow the diagram and correspond to the numbered blocks[49]:



Data to send to external address space : ☐ Convert to ebcdic
Test data to external a/s

Response back from external address space : ☐ Convert to ascii

OLA Register Name (max 12 chars)
CICSTEST

OLA Service Name (max 256 chars)
myserv

CICS Link Server-specific data : ☐ Use Containers
CICS-Link Request Container Id (max 16 chars) ☐ BIT Container
WAS_BBOA_REQ
CICS-Link Response Container Id (max 16 chars) ☐ BIT Container
WAS_BBOA_REQ
CICS-Link Transaction id (max 4 chars) BBO#

OLA over IMS OTMA data: ☐ Use OLA over OTMA ☐ Print debug data?
IMS OTMA XCF Group Name IMS OTMA Server Name

Notes:

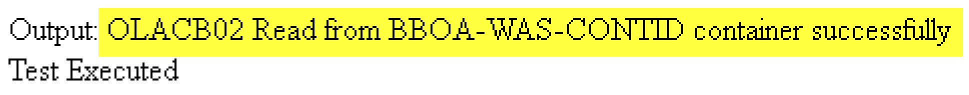
1. Supply the registration name used on the BBOC command (which was **`CICSREG`**)
2. The service name in this case will be **`OLACB02`**
3. ***Check*** the "Use Containers" check box[50].
4. For the request container and the response container, override the default seen on the browser screen and use the same value of **`BBOA-WAS-CONTID`** *for both*[51].
5. Do ***not*** check the "BIT Container" check boxes.
6. Click the button to send the message over to CICS using channels and containers.

48 If you're using the copy of OLASample2 that we updated to perform Basic Authentication, then supply the ID and password you used earlier for the identity assertion exercises.
49 This is showing the browser screen for OLASample2 **pre**-8.0.0.5/8.5.0.2. If you're using 8.0.0.5 / 8.5.0.2 or higher, see "Improved Channels and Containers using 8.0.0.5 or 8.5.0.2" starting on page 75.
50 This tells the Java code to use the `ConnectionSpecImpl` class `setUseCICSContainer()` method.
51 The OLACB02 program is hard-coded to expected that string for both the request and response container. Any other value will result in the message "OLACB02 sending string BBOA-WAS-CONTID container was not passed to the program" in the CICS region `CEEMSG DD` and with your input text echoed back to the browser, *which is not correct*. What you are expecting to see as success is "OLACB02 Read from BBOA-WAS-CONTID container successfully" on the browser screen.

☐ Success and troubleshooting:

***Successful** execution:*

Browser:

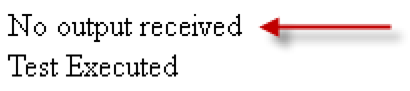

Output: OLACB02 Read from BBOA-WAS-CONTID container successfully
Test Executed

CEEMSG DD:

```
OLACB02 sending string OLACB02 Read from BBOA-WAS-CONTID container
successfully
```

***Unsuccessful** execution:*

Browser:


No output received
Test Executed

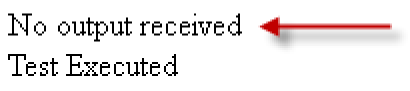
CEEMSG DD:

```
OLACB02 entered with channel <no channel name appears>
CEE3250C The system or user abend ERCH was issued.
  From compile unit OLACB02 at entry point OLACB02 at compile unit
  offset 16210800.
```

Cause: The "Use Containers" check box was not checked.

***Unsuccessful** execution:*

Browser:


No output received
Test Executed

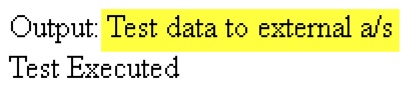
CEEMSG DD:

```
CEE3250C The system or user abend ECB1 was issued.
  From compile unit OLACB01 at entry point OLACB01 at compile unit
  offset 16212800.
CEE3DMP V1 R13.0: Condition processing resulted in the unhandled
  condition.
```

Cause: The "Use Containers" check box was checked, but the "Service Name" used was the `OLACB`**`01`** (not `OLACB02`) program.

Unsuccessful execution:

Browser:


Output: Test data to external a/s
Test Executed

(The text message you sent appears to be echoed back, just like with OLACB01)

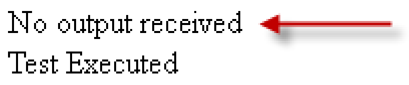
CEEMSG DD:

```
OLACB02 sending string BBOA-WAS-CONTID container was not passed
  to the program
```

Cause: Something *other* than `BBOA-WAS-CONTID` was used for the request container ID. Check to make sure that exact string is used for the *request* container ID.

***Unsuccessful** execution:*

Browser:



No output received
Test Executed

CEEMSG DD:

```
OLACB02 sending string OLACB02 Read from BBOA-WAS-CONTID container
successfully
```

Cause: The value `BBOA-WAS-CONTID` was used for the *request* container ID, but something *other than* `BBOA-WAS-CONTID` was used for the *response* value.

Optional - Link Server start with BBOACPL2 PLT and INITPARM

Overview:

Up to this point in the exercises you have started the Link Server Task using the BBOC transaction through a CICS terminal session. The manual method works, but a more common approach is to have the TRUE and the Link Server Task started at CICS initialization.

The section titled "Appendix B: Starting Link Server Task at CICS Startup" starting on page 66 covers the different means by which this may be accomplished.

In this section the first of three methods will be covered -- using `INITPARM=()` and the `BBOACPL2` sample program compiled with the `DFHPLTOL` job.

Do the following:

- ☐ Go to "BBOACPL2 and INITPARM=()" starting on page 66 and follow the instructions there. Return to this spot when you've completed the steps outlined there.
- ☐ Exercise any or all of the tests you've done to this point in the document:
  - ○ Outbound WAS ⇨ CICS using OLASample1 or OLASample2 application
  - ○ Inbound CICS ⇨ WAS using OLAUTIL 3270 application in CICS
  - ○ Outbound WAS ⇨ CICS using ATSSample1-new application

Optional - Link Server start with BBOACPL3 PLT

Overview:

One of the limitations of `BBOACPL2` and `INITPARM=()` is the length limitation CICS imposes on the use of `INITPARM`. There is only so much that can be done to reduce the length of the parameters. When the Link Server Task start string is simply too long, then another means of starting it at CICS initialization is needed.

Do the following:

- ☐ Go to "BBOACPL3 and longer BBOC commands" starting on page 67 and follow the instructions there. Return to this spot when you've completed the steps outlined there.
- ☐ Exercise any or all of the tests you've done to this point in the document:
  - ○ Outbound WAS ⇨ CICS using OLASample1 or OLASample2 application
  - ○ Inbound CICS ⇨ WAS using OLAUTIL 3270 application in CICS
  - ○ Outbound WAS ⇨ CICS using ATSSample1-new application

Optional - Link Server start with Sequential Terminal

Overview:

CICS has long supported the use of a "sequential terminal" to input commands. That sequential terminal may be a data set containing the `BBOC START_SRVR` command. This is another way to get around the length limitation on INITPARM.

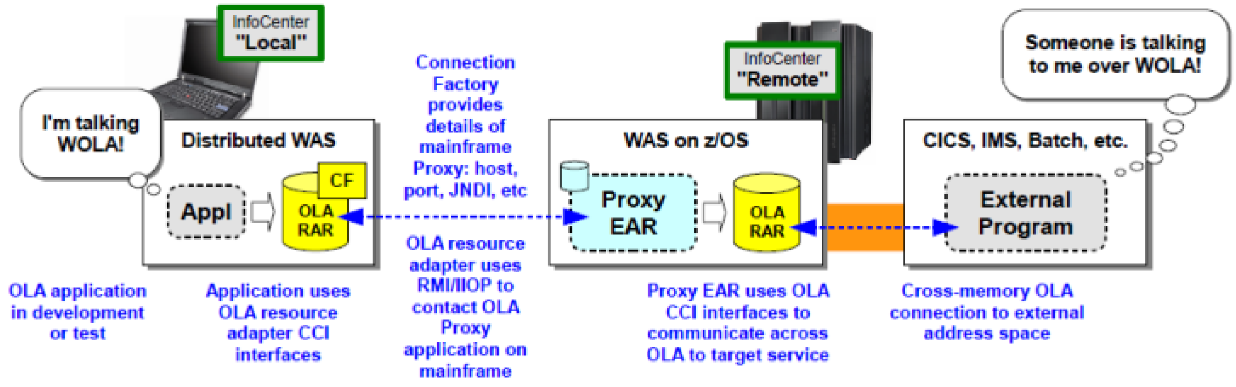
Do the following:

- ☐ Go to "Sequential Terminal" starting on page 71 and following the instructions there. Return to this spot when you've completed the steps outlined there.
- ☐ Exercise any or all of the tests you've done to this point in the document:
  - ○ Outbound WAS ⇨ CICS using OLASample1 or OLASample2 application
  - ○ Inbound CICS ⇨ WAS using OLAUTIL 3270 application in CICS
  - ○ Outbound WAS ⇨ CICS using ATSSample1-new application

Optional - "Development Mode" ... Distributed WAS using Proxy EJB into CICS[52]

This is function made available in WAS 8.0.0.1 or WAS 8.5.0.1[53]. It enables a Java developer using a distributed WAS server (for example, on their development Windows or Linux machine) to test their Java code without having to deploy the application on z/OS.

In the WP101490 Techdoc there is a PDF titled "WOLA History of Updates Timeline," and in that PDF is an overview picture of what this section is related to:


InfoCenter "Local"
I'm talking WOLA!
Distributed WAS
Appl
CF
OLA RAR
Connection Factory provides details of mainframe Proxy: host, port, JNDI, etc
OLA resource adapter uses RMI/IIOP to contact OLA Proxy application on mainframe
InfoCenter "Remote"
WAS on z/OS
Proxy EAR
OLA RAR
Someone is talking to me over WOLA!
CICS, IMS, Batch, etc.
External Program
OLA application in development or test
Application uses OLA resource adapter CCI interfaces
Proxy EAR uses OLA CCI interfaces to communicate across OLA to target service
Cross-memory OLA connection to external address space

Overview

Refer to the picture above when reading these bullets:

- The model we're exploring here is "outbound" from WAS to CICS (or IMS). It's just that the instance of WAS is on a distributed platform and not z/OS.
- True WOLA is limited to same-LPAR cross memory, so there's no way to do cross memory from the distributed WAS to CICS. Instead, a "proxy" function is placed between the distributed WAS instance and z/OS WAS. That proxy function involves communication over a standard IP network.
- The application on distributed WAS communicates with the WOLA JCA resource adapter. You can download that `ola.rar` file and install it into WAS on distributed in the exact same way you install it into WAS on z/OS.
- The connection factory on distributed is updated with a few custom properties to tell it what z/OS system to talk to. We'll cover those custom properties a bit later.
- On z/OS a supplied "Proxy EJB" is installed to serve as "catcher" for the WOLA proxy calls coming up from WAS distributed. The Proxy EJB then turns and drives across the ola.rar JCA resource adapter to do "true WOLA" into CICS (or IMS).

52 Special thanks to Mike Kearney for his help and support on configuring the CSIv2 security settings for this.
53 In the InfoCenter, search on the string `tdat_develop_config_localmode`.

This model still requires WAS on z/OS, and WAS on z/OS still needs to be configured to support WOLA. So all the earlier steps to enable and validate WOLA communications into CICS apply. But the Java application doing the WOLA calls runs on a distributed WAS instance. That allows the developer to start and stop the WAS instance as much as they want without having to get involved with z/OS operations.

Determine security settings for WAS z/OS

This "Development Mode" function involves RMI/IIOP calls from the distributed WAS server up to WAS on z/OS. When global security is enabled on the target WAS z/OS then it means global security *must* be enabled on the distributed WAS instance as well.

For this document we will assume global security is *enabled* for the WAS z/OS server[54].

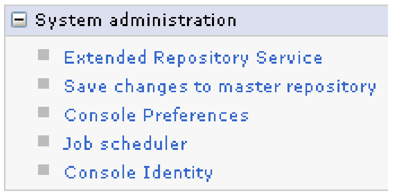
Assuming security is on for WAS z/OS implies making sure global security is on for the distributed WAS as well. That's next.

Determine security settings for distributed WAS, and enable if necessary

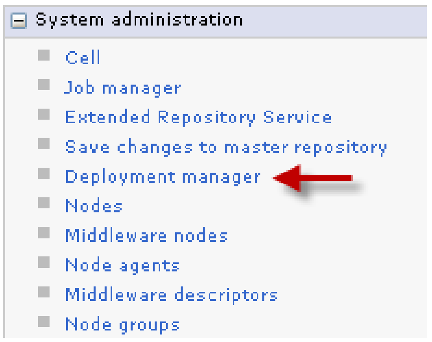
Do the following:

- ☐ Start your distributed WAS server and log onto the Admin Console. Does it prompt for a password?
  - ○ Yes -- then security is already enabled. Skip to "Assumed starting point for WAS z/OS environment" on page 39.
  - ○ No -- Security is not enabled. Proceed to the next step in this section.
- ☐ Determine if your distributed environment is a "Standalone" or "Network Deployment" configuration[55]. In the Admin Console, go to *System Administration*:

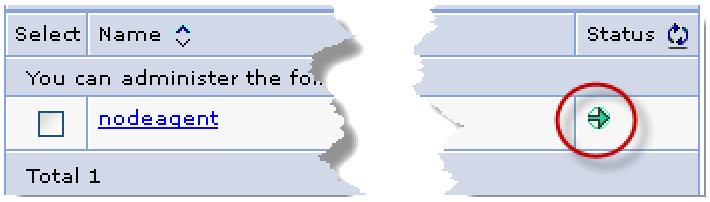
Standalone Server:



Network Deployment:



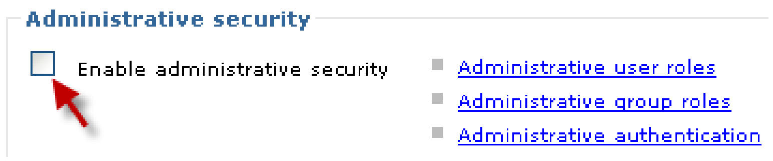
- ☐ If Network Deployment[56], then click on "Node Agents." Make sure *all* the Node Agents listed show a status of "started":



Start any Node Agents that are not already started. Verify "green arrow" for each.

54 That's easily determined by logging onto the Admin Console. If it prompts for an ID *and* password, then global security is enabled.
55 If Network Deployment, then turning on security requires synchronization to all the nodes. Failure to synchronize the security change could create problems later. Knowing the environment now means we can be extra careful about synchronizing.
56 If Standalone, then skip this step.

- ☐ Go to *Security* → *Global Security.* You should find the "Enable administrative security" checkbox unchecked:

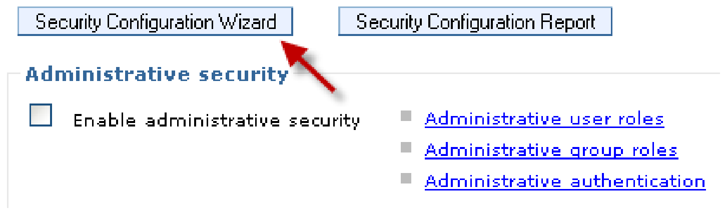


Administrative security

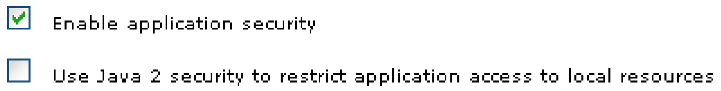
☐ Enable administrative security

Administrative user roles
Administrative group roles
Administrative authentication

- ☐ Click on the "Security Configuration Wizard" button:


Security Configuration Wizard
Security Configuration Report
Administrative security
Enable administrative security
Administrative user roles
Administrative group roles
Administrative authentication

- ☐ On the "Step 1: Specify extent of protection" panel it will ask you if you want application security and Java 2 security. The default will be:

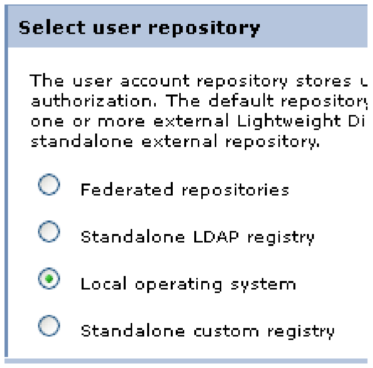


[✓] Enable application security

☐ Use Java 2 security to restrict application access to local resources

If you're not sure what to provide here, turn both off[57]. Since you had security off before, you were operating with no security. Go with minimal here.

- ☐ The next panel will ask you what user repository you wish to use for authentication. The easiest would be "Local operating system." That tells WAS to check against the local OS when a userid and password is provided:



Select user repository

The user account repository stores u
authorization. The default repositor
one or more external Lightweight Di
standalone external repository.

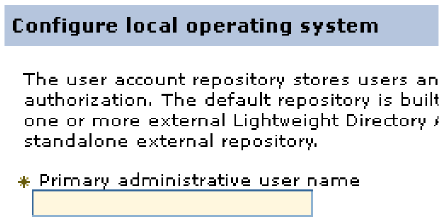
○ Federated repositories

○ Standalone LDAP registry

● Local operating system

○ Standalone custom registry

- ☐ Next, provide the "Primary adminstrative user name." This will be ID you use to log into the Admin Console. For example, on Windows you would supply the ID you use to log onto Windows. Then when you log into the Admin Console you would use your Windows ID and password.



Configure local operating system

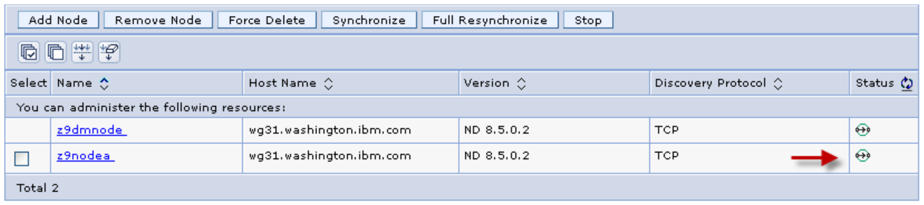
The user account repository stores users an
authorization. The default repository is built
one or more external Lightweight Directory
standalone external repository.

* Primary administrative user name

- ☐ Click next to the summary, then "Finish." Then Save *and synchronize the change to all the nodes*. (If Standalone, then a simple save is sufficient.)

57 The default with "Enable application security" box checked is acceptable for using the WOLA proxy function.

☐ If Network Deployment[58], now go to *System Administration* → *Nodes* and make sure the changes have been synchronized:



Select	Name	Host Name	Version	Discovery Protocol	Status
You can administer the following resources:					
	z9dmnode	wg31.washington.ibm.com	ND 8.5.0.2	TCP	●
☐	z9nodea	wg31.washington.ibm.com	ND 8.5.0.2	TCP	●
Total 2					

The little green circle indicates synchronized. A little red circle with a broken line through it means not synchronized. If not synchronized, then click the check box next to the node and then click the Synchronize button.

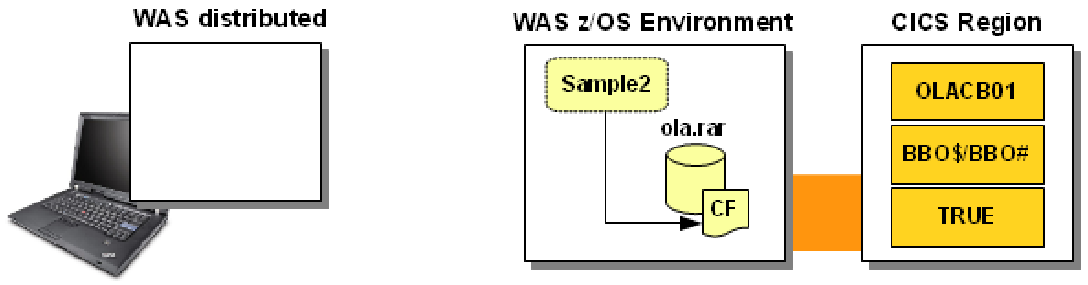
Important: Make sure all the nodes are showing synchronized before you move on. If a node is showing a question mark for the status that means the Node Agent is not started. Start the node agent and synchronize.

☐ Log off the Admin Console and stop *all the servers in the cell*. If Standalone, that means just the one server. If Network Deployment, then it means the Deployment Manager, Node Agents and Servers.

☐ Restart the servers and attempt to log into the Admin Console. You should be prompted for an ID and a password. Enter the password for the primary ID you entered earlier. You should then be taken into the Admin Console.

Assumed starting point for WAS z/OS environment

At this point in the step-by-step instructions we have to set a baseline on what we assume is configured at this point. The following picture illustrates the assumed starting point:



Notes:

- WOLA is enabled in the WAS cell
- The WOLA JCA resource adapter (`ola.rar`) is installed with a configured connection factory
- The OLASample2 application is installed
- WOLA is enabled in the CICS region, with the Task Related User Exit, the Link Server Task and the sample OLACB01 program installed.

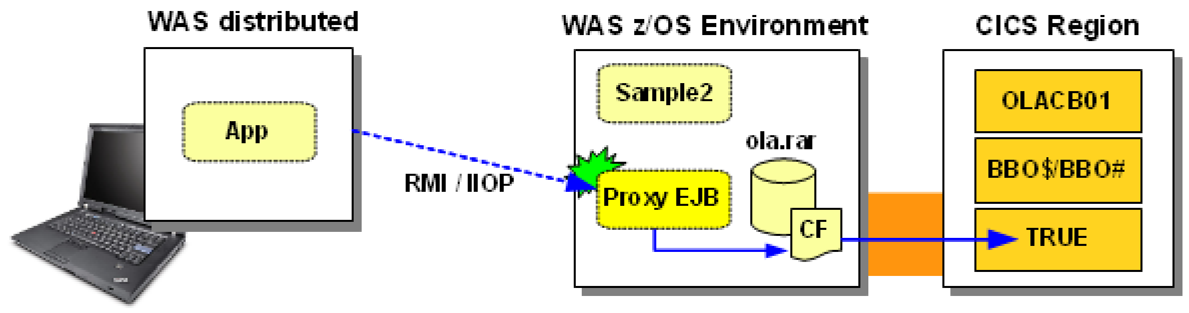
You would have all this if you worked through the earlier parts of this Quick Start guide up to *and including* "Outbound WAS into CICS with OLACB01 as target service" on page 20. Those steps took you though enabling WOLA in WAS and CICS and validating using the supplied samples.

If you don't have this as part of your WAS z/OS environment, then return to the earlier parts of this document and step through the instructions until this is constructed.

58 If Standalone, then skip this step.

Install Proxy EJB in WAS z/OS

The Proxy EJB is a supplied application that serves as the "catcher" for RMI/IIOP calls coming up from the distributed WAS. You install it into the application server you intend to use for WOLA:



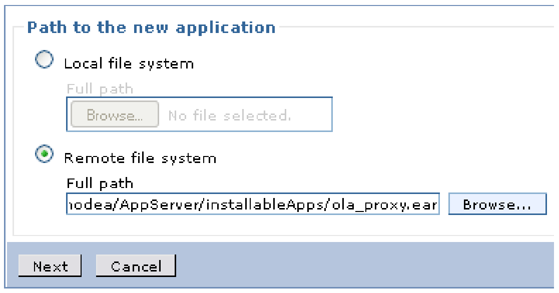
The Proxy EJB application is supplied[59] at the following location:

/*<node_mount_point>*/AppServer/installableApps/ola_proxy.ear

It's a relatively simple installation, very much like any other application.

Do the following:

- ☐ In your WAS z/OS configuration node directory, locate the `ola_proxy.ear` file.
- ☐ Begin a new application installation as you would any EAR installation.
- ☐ For the "Path to the new application," use "Remote file system" and point to the path and file name for the EAR file:



Path to the new application

( ) Local file system

Full path

Browse... No file selected.

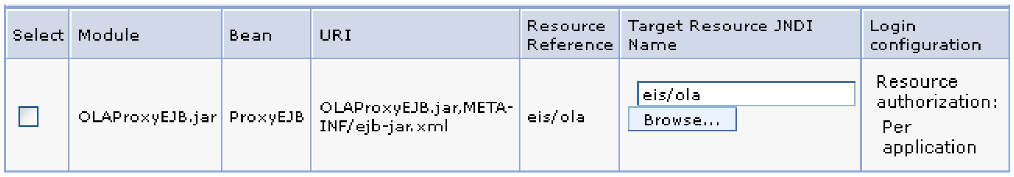
(●) Remote file system

Full path

nodea/AppServer/installableApps/ola_proxy.ear Browse...

Next Cancel

- ☐ For "Step 3," map the "Target Resource JNDI Name" field to the JNDI of the WOLA JCA connection factory[60].



Select	Module	Bean	URI	Resource Reference	Target Resource JNDI Name	Login configuration
☐	OLAProxyEJB.jar	ProxyEJB	OLAProxyEJB.jar,META-INF/ejb-jar.xml	eis/ola	eis/ola Browse...	Resource authorization: Per application

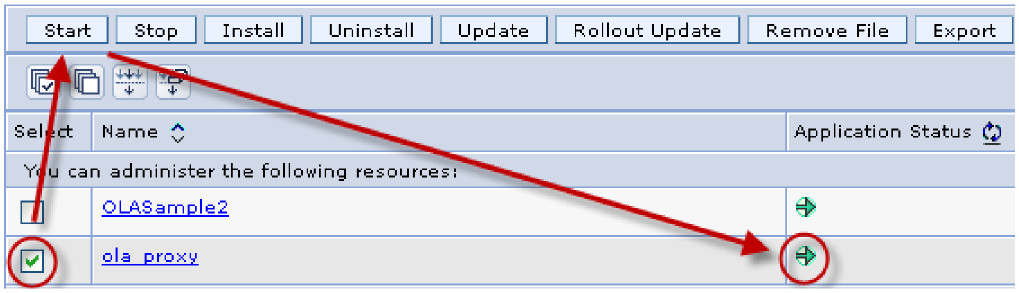
Note: Pause a moment and think about that. The WOLA JCA resource adapter connection factory is how Java applications gain access to outbound WOLA. We saw that earlier with the OLASample2 application.

Here we have another application -- the "proxy" application -- doing the exact same thing. The proxy serves as the "catcher" for RMI/IIOP calls coming up from the distributed WAS. It then turns and drives the WOLA JCA just like the OLASample2 application did. The difference is the proxy EJB is driving WOLA on behalf of the Java application on the distributed WAS.

- ☐ Complete the application installation, then save and synchronize.

59 At levels 8.0.0.1 or 8.5.0.0 or later. Earlier levels of WAS z/OS did not have this development mode support and thus no proxy EJB.
60 We had you set this to `eis/ola` back on page 11.

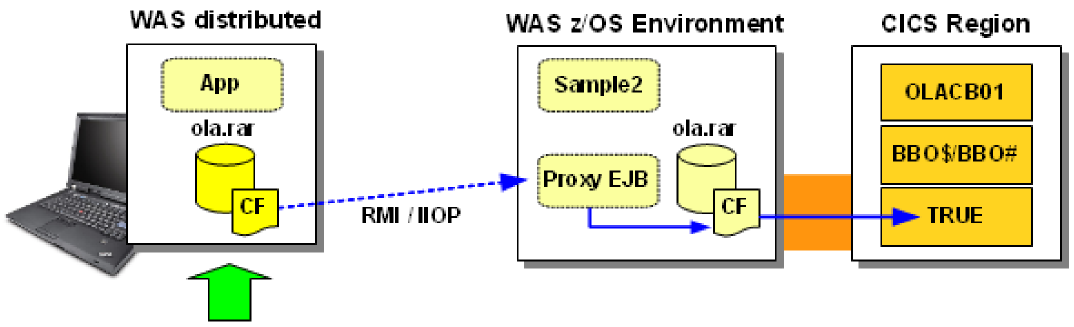
- ☐ Go to *Applications* → *Application types* → *WebSphere enterprise applications* and start the proxy application:



The proxy is installed and ready to accept calls from the distributed WAS instance. The next step is to configure the distributed WAS instance to be able to make the calls.

Download and install `ola.rar` in distributed WAS environment

Back under "Install the ola.rar resource adapter and create a connection factory" on page 9 we had you install `ola.rar` into your WAS z/OS node. This is essentially the same process, except the installation target is your *distributed* WAS node:



The application in the distributed WAS instance will use the ola.rar JCA resource adapter just like the application on WAS z/OS did. The difference will be in the Connection Factory ("CF"), which will be configured with custom properties to indicate where the Proxy EJB is located.

Note: During testing it was discovered the 8.5.0.2 copy of `ola.rar` had problems. The symptom was a FFDC event emitted on the distributed WAS instance that started with:

```
Exception:com.ibm.ISecurityLocalObjectCSIv2UtilityImpl.GSSEncodeDecodeException
```

This document was written with the 8.5.5.0 copy of `ola.rar` installed in an instance of WAS distributed running at 8.5.5.0 as well. That worked well.

If you're running at the 8.5.0.2 level of WAS z/OS, consider updating the distributed WAS server to 8.5.5.0 and pulling the 8.5.5.0 WAS z/OS copy of `ola.rar`.

Do the following:

- ☐ Locate the `ola.rar` file in your WAS z/OS configuration. It should be found at:

  `/<node_mount_point>/AppServer/installableApps/ola.rar`

- ☐ Download that file *in binary* to your distributed workstation or server.
- ☐ Follow the procedures[61] documented under "Install the ola.rar resource adapter and create a connection factory" starting on page 9. The steps are identical on a distributed WAS instance as they were on WAS z/OS.
- ☐ Once the RAR has been installed and the connection factory created, make sure to save (and if necessary, synchronize) the changes.

61 Taking into account the RAR file will be located on your *local* file system in this case rather than *remote* as was done earlier.

Download and install the OLASample2 sample application

The same `OLASample2.ear` file deployed on WAS z/OS earlier may be used to test and validate this "development mode" function using the Proxy EJB.

Do the following:

- ☐ Locate the OLASample2.ear file on your WAS z/OS system. It should be located at:

  `/<node_mount_point>/AppServer/util/zos/OLASamples/OLASample2.ear`

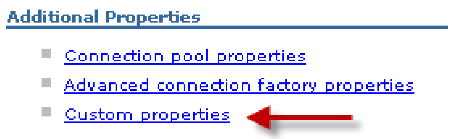
- ☐ Download that file *in binary* to your distributed workstation or server.
- ☐ Follow the procedures[62] documented under "Install sample EAR file" starting on page 11. The steps are identical on a distributed WAS instance as they were on WAS z/OS.
- ☐ Once the RAR has been installed and the connection factory created, make sure to save (and if necessary, synchronize) the changes.

Configure custom properties on JCA connection factory

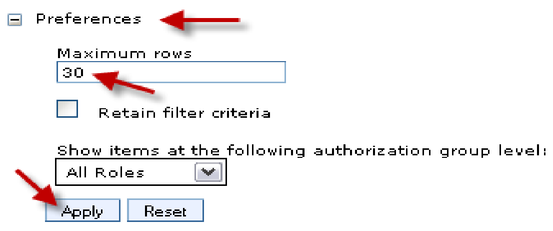
This is where things change from what was done earlier on z/OS. Custom properties on the connection factory on distributed WAS provide a way to define where the Proxy EJB resides.

Do the following:

- ☐ In your **distributed** WAS instance Admin Console, go to *Resources → Resource Adapters → J2C connection factories*
- ☐ Set the scope to "All scopes" and then click on the link that represents the WOLA connection factory you created in the previous section.
- ☐ On the right side of the resulting panel, click on "Custom properties:"


Additional Properties
Connection pool properties
Advanced connection factory properties
Custom properties

- ☐ Expand "Preferences," then set the row value to 30[63] and click apply:



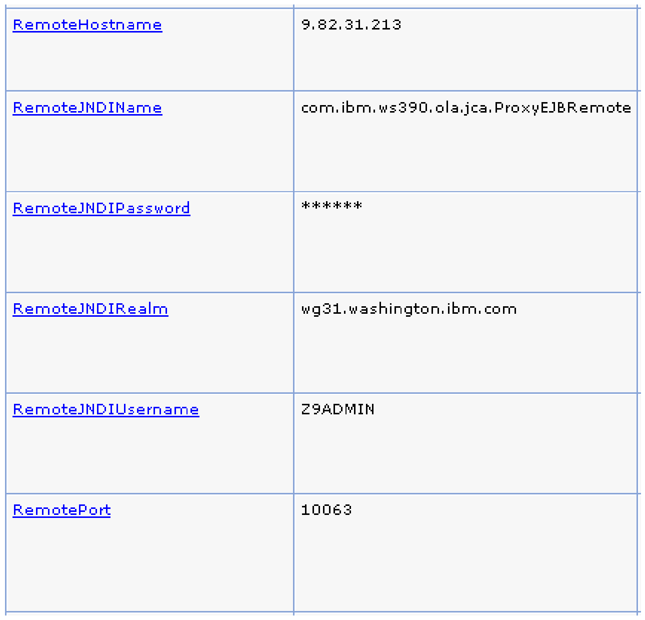
- ☐ Scroll down and locate "RemoteHostname". That's the first of six custom properties to be set.
- ☐ Set the following custom properties according to the this table:

Custom Property	***Description of value to provide***
RemoteHostname	Set this to the host name where the WAS z/OS server running the Proxy EJB can be located. A DNS host name or dotted-decimal IP address may be used.

62 Taking into account the EAR file will be located on your *local* file system in this case rather than *remote* as was done earlier.
63 Why? Because there are 24 WOLA custom properties and setting the rows to 30 allows all to be displayed at once.

Custom Property	***Description of value to provide***
RemoteJNDIName	This is the JNDI name of the deployed Proxy EJB. By default it will be: **`com.ibm.ws390.ola.jca.ProxyEJBRemote`** **Note:** this assumes the port you specify for the `RemotePort` property is hosted by the application server where the Proxy EJB is deployed. If you specify the Node Agent ORB port then the syntax of the JNDI name changes: `cell/nodes/`*`aaaa`*`/servers/`*`bbbb`*`/`*`cccc`* Where *`aaaa`* is the node long name, *`bbbb`* is the server short name, and *`cccc`* is the JNDI name of the Proxy EJB.
RemoteJNDIPassword	The password for the ID specified for the RemoteJNDIUsername property, which appears two rows lower in this table. There we recommend coding the WAS Admin ID. So provide the WAS Admin ID password here. The value you supply with be encoded when saved into the XML. See note at end of table.
RemoteJNDIRealm	In most cases this will be the configured host name for the Daemon associated with the Deployment Manager on z/OS. If the target server on z/OS is a standalone server then it's the hostname for the Dameon associated with the standalone server.
RemoteJNDIUsername	The ID to be used to log in and authenticate as part of the JNDI lookup of the Proxy EJB. Provide the WAS Admin ID[64]. See note at end of table.
RemotePort	The `ORB_LISTENER_ADDRESS` port configured for the WAS z/OS server where the Proxy EJB is installed.
Note: another way to authenticate into the WAS z/OS system is to use client certificates. That requires no coding of ID or password. See "Alternative authentication - client certificate" on page 49 for more on that process.	

For *example*, during testing used to write this section of the document, our custom properties looked like this:



RemoteHostname	9.82.31.213
RemoteJNDIName	com.ibm.ws390.ola.jca.ProxyEJBRemote
RemoteJNDIPassword	******
RemoteJNDIRealm	wg31.washington.ibm.com
RemoteJNDIUsername	Z9ADMIN
RemotePort	10063

Example of J2C connection factory custom properties filled out. Your values may differ.

☐ Save and synchronize the changes.

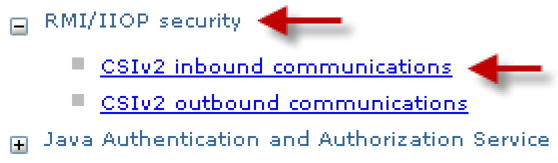
64 Other IDs can be made to work. The WAS Admin ID is recommended here simply because it's most likely to already exist, and during testing we validated that ID's authority is sufficient.

Configure security settings on both distributed and z/OS

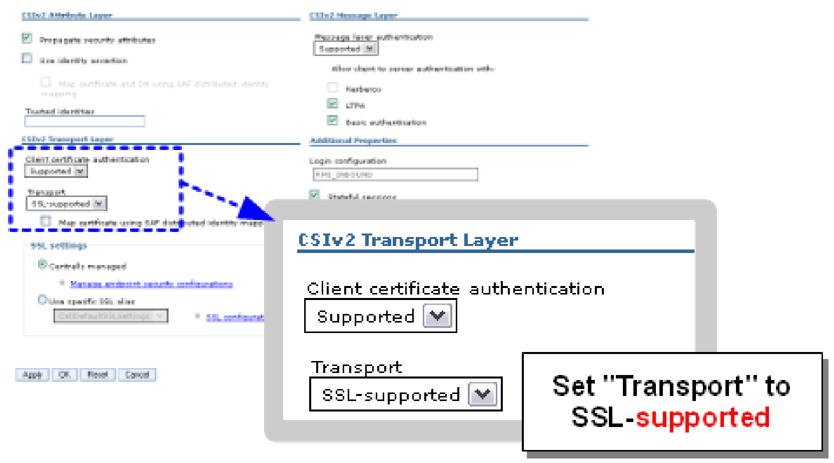
This will involve changes on *both* the distributed WAS instance *and* the WAS z/OS instance.

Do the following on the ***WAS z/OS*** side:

- ☐ In the Admin Console, go to *Security* → *Global security*
- ☐ On the right side of the screen, expand "RMI/IIOP security," then click on "CSIv2 *inbound* communications":


RMI/IIOP security
CSIv2 inbound communications
CSIv2 outbound communications
Java Authentication and Authorization Service

- ☐ Then set the "CSIv2 Transport Layer" to "SSL-supported":

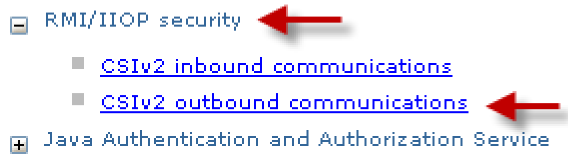

CSIv2 Transport Layer

Client certificate authentication: Supported

Transport: SSL-supported

Set "Transport" to SSL-supported

- ☐ Click "OK" to commit that change. You'll return to the Global Security panel.
- ☐ Now click on the "CSIv2 outbound communications" link:

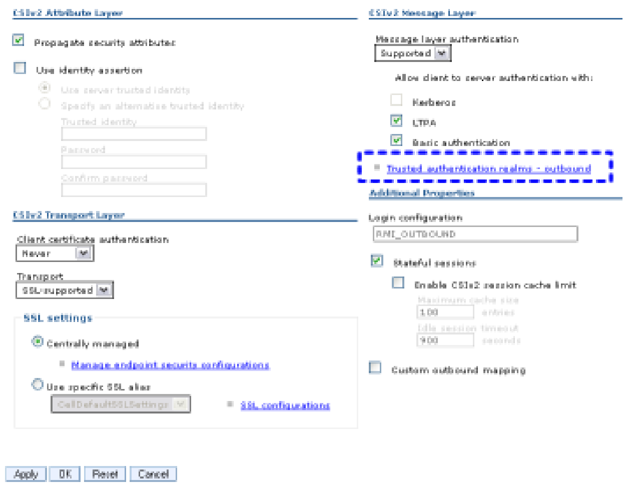

RMI/IIOP security
CSIv2 inbound communications
CSIv2 outbound communications
Java Authentication and Authorization Service

- ☐ Set the "CSIv2 Transport Layer" to "SSL-supported" just as you did for inbound.
- ☐ Click "OK" and then save and synchronize the changes.

Do the following on the ***distributed WAS*** side:

- ☐ In the Admin Console, go to *Security* → *Global security*
- ☐ Set the "CSIv2 Transport Layer" to "SSL-supported" for *both* inbound and outbound, just like you did for WAS z/OS in the previous section.
- ☐ Save and synchronize changes.
- ☐ From the main Global Security panel, click the "CSIv2 outbound communications" link.

- ☐ Click on the "Trusted authentication realms - outbound" link:



CSIv2 Attribute Layer

- [✓] Propagate security attributes
- [ ] Use identity assertion
  - ● Use server trusted identity
  - ○ Specify an alternative trusted identity
  - Trusted identity
  - Password
  - Confirm password

CSIv2 Transport Layer

Client certificate authentication: Never

Transport: SSL-supported

SSL settings

- ● Centrally managed
  - Manage endpoint security configurations
- ○ Use specific SSL alias: CellDefaultSSLSettings
  - SSL configurations

CSIv2 Message Layer

Message layer authentication: Supported

Allow client to server authentication with:

- [ ] Kerberos
- [✓] LTPA
- [✓] Basic authentication

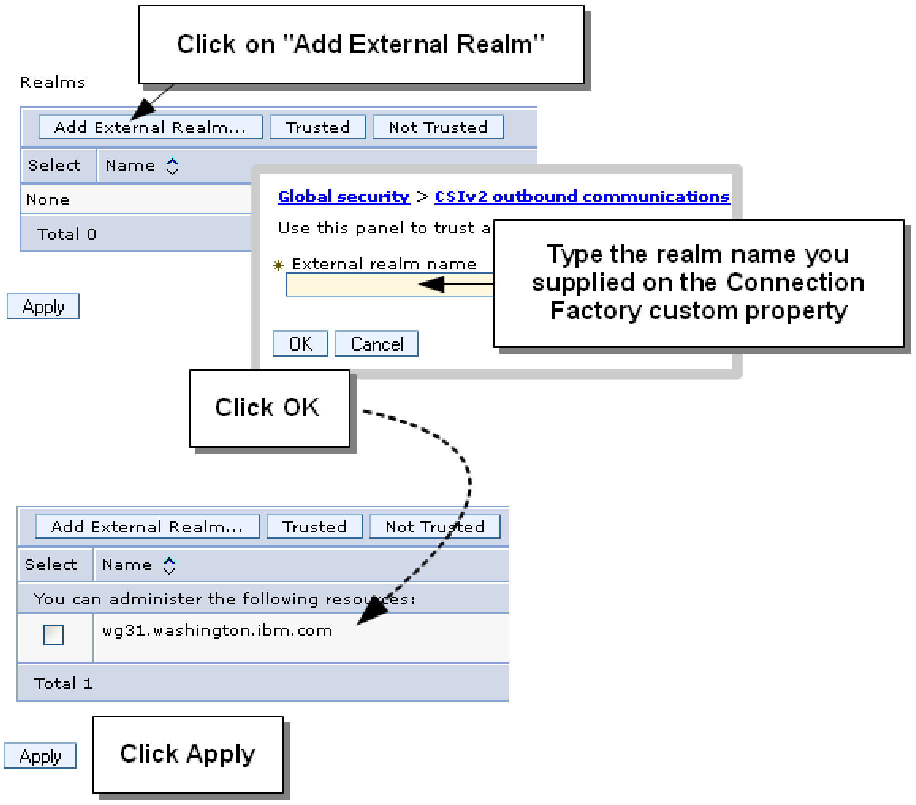
- Trusted authentication realms - outbound

Additional Properties

Login configuration: RMI_OUTBOUND

- [✓] Stateful sessions
  - [ ] Enable CSIv2 session cache limit
    - Maximum cache size: 100 entries
    - Idle session timeout: 900 seconds
- [ ] Custom outbound mapping

- ☐ Now supply the trusted realm name, which will be the same value you provided on the connection factory custom property back on page 43. Do the following:



Realms

Select	Name
None	
Total 0	

Global security > CSIv2 outbound communications

Use this panel to trust a

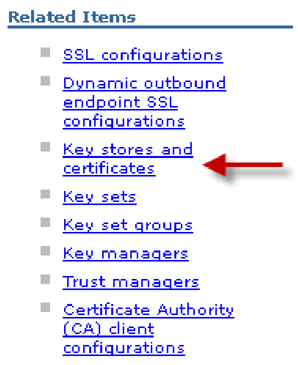
* External realm name

Select	Name
You can administer the following resources:	
☐	wg31.washington.ibm.com
Total 1	

Click on "Add External Realm"
Add External Realm...
Trusted
Not Trusted
Apply
Type the realm name you supplied on the Connection Factory custom property
OK
Cancel
Click OK
Click Apply

- ☐ Save and synchronize all changes.
- ☐ The next step involves importing the Certificate Authority (CA) certificate from your z/OS system into your distributed WAS trust store. This can be done through the Admin Console. Go to *Security* → *SSL certificate and key management*.

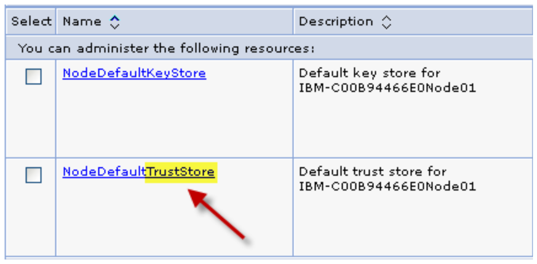
- ☐ Click on "Key stores and certificates":



Related Items

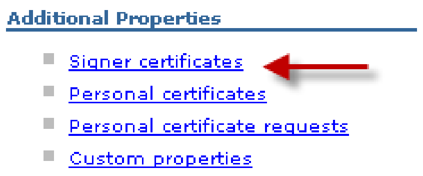
- SSL configurations
- Dynamic outbound endpoint SSL configurations
- Key stores and certificates
- Key sets
- Key set groups
- Key managers
- Trust managers
- Certificate Authority (CA) client configurations

- ☐ Then click on the TrustStore link:



Select	Name	Description
You can administer the following resources:		
☐	NodeDefaultKeyStore	Default key store for IBM-C00B94466E0Node01
☐	NodeDefaultTrustStore	Default trust store for IBM-C00B94466E0Node01

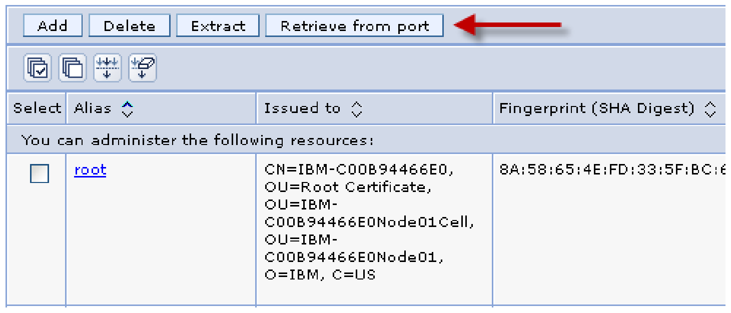
- ☐ Click on the "Signer certificates" link:



Additional Properties

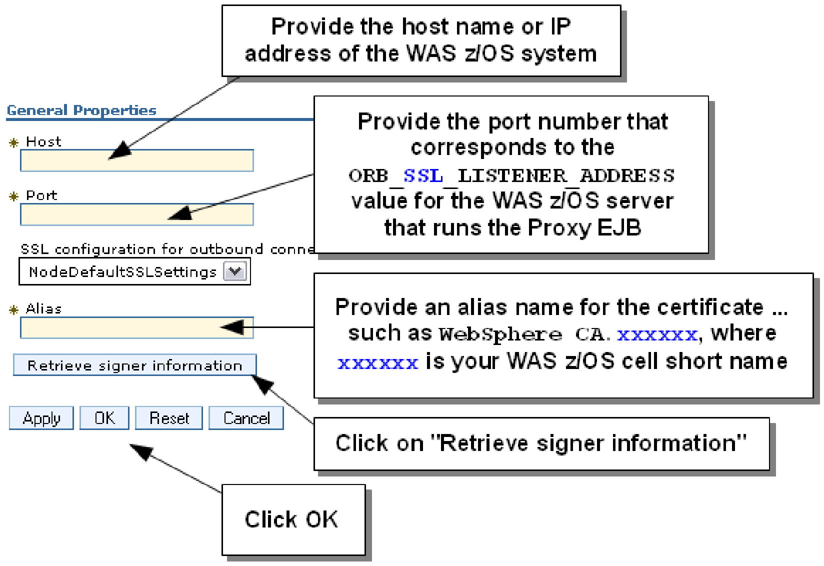
- Signer certificates
- Personal certificates
- Personal certificate requests
- Custom properties

- ☐ Then click on the "Retrieve from port" button:



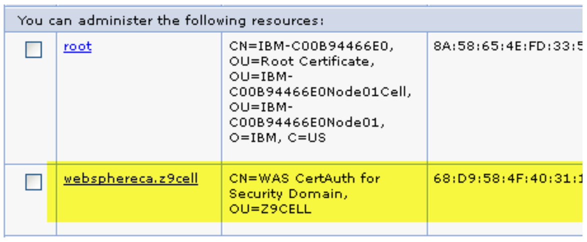
Select	Alias	Issued to	Fingerprint (SHA Digest)
You can administer the following resources:			
☐	root	CN=IBM-C00B94466E0, OU=Root Certificate, OU=IBM-C00B94466E0Node01Cell, OU=IBM-C00B94466E0Node01, O=IBM, C=US	8A:58:65:4E:FD:33:5F:BC:6

- ☐ Now do the following:


General Properties
Host
Port
SSL configuration for outbound conne
NodeDefaultSSLSettings
Alias
Retrieve signer information
Apply
OK
Reset
Cancel
Provide the host name or IP address of the WAS z/OS system
Provide the port number that corresponds to the ORB_SSL_LISTENER_ADDRESS value for the WAS z/OS server that runs the Proxy EJB
Provide an alias name for the certificate ... such as WebSphere CA.xxxxxx, where xxxxxx is your WAS z/OS cell short name
Click on "Retrieve signer information"
Click OK

Important: You must specify the ORB SSL port, not the other ORB port. And you must click the "Retrieve signer information" before clicking OK.

- ☐ You should then see your WAS z/OS CA certificate[65] in the list:



You can administer the following resources:

☐	root	CN=IBM-C00B94466E0, OU=Root Certificate, OU=IBM-C00B94466E0Node01Cell, OU=IBM-C00B94466E0Node01, O=IBM, C=US	8A:58:65:4E:FD:33:5
☐	webspherecaz9cell	CN=WAS CertAuth for Security Domain, OU=Z9CELL	68:D9:58:4F:40:31:1

- ☐ Save and synchronize your changes

Restart WAS on both distributed and z/OS

You've made a lot of security changes on both sides, so it's best to start fresh and restart everything.

Do the following on the z/OS side:

- ☐ Check to see if you have any WOLA registrations into your server. Issue the command:

```
F <server_JOBNAME>,DISPLAY,ADAPTER,DAEMONRGES
```

If you see something like this:

```
F Z9SR01A,DISPLAY,ADAPTER,DAEMONRGES
BBOA0007I: SHOWING REGISTRATIONS FOR DAEMON GROUP:
BBOA0003I: Name          Jobname  SWT TL Min  Max  Act  State
BBOA0004I: CICSXREG      CICSX    000 00 0001 0010 0003 00
BBOA0004I: $WASDEFAULT$ CICSX    000 00 0000 0001 0000 00
BBOO0188I END OF OUTPUT FOR COMMAND DISPLAY,ADAPTER,DAEMONRGES
```

Then your server has a registration from CICS from earlier. Close the registration[66] from the CICS region with:

```
BBOC STOP_SRVR RGN=CICSXREG
```

65 It will lower-case the alias value you provide.

66 Strictly speaking this is not necessary. You could simply stop the WAS server. The registration will break (because the WAS server goes away) and when it comes back the TRUE in CICS will re-establish the registration. That reconnect function has been improved since the early days of WOLA, and with V8.5 it works very well. Still, a clean shutdown and a clean restart is good practice.

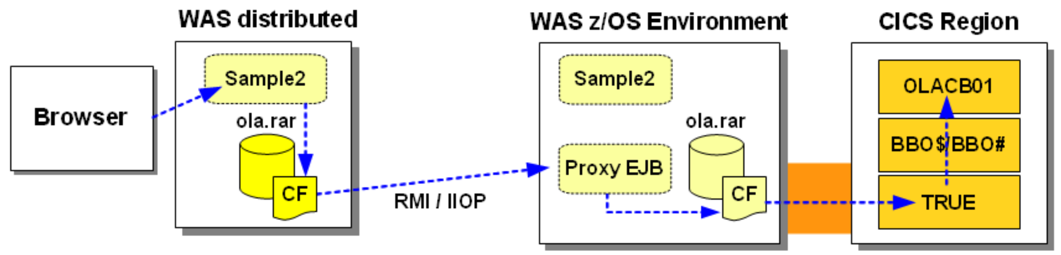
- ☐ Stop all the servers in the cell, including the Daemon, and then restart them[67].
- ☐ From the CICS region re-establish the WOLA registration into the server. For an example of the `BBOC` command to do that, see page 17.

Do the following on the distributed WAS side:

- ☐ Stop your servers and restart them.

Validate proxy function up to WAS z/OS and into CICS

You're ready to test the proxy function. Here's what will take place:



- The browser is pointed at the OLASample2 application deployed on the distributed WAS. The WOLA registration name and service name are specified.
- The Sample2 application in distributed WAS does a JNDI lookup of the JCA connection factory
- The JCA connection factory has custom properties that specify a remote call to the Proxy EJB deployed on WAS z/OS. The lookup and call is made, and the WOLA registration and service names are passed over.
- The Proxy EJB on WAS z/OS performs a lookup of the local WOLA JCA connection factory.
- The call is made over the WOLA registration passed over from the distributed WAS instance
- The service name that was passed over from the distributed WAS instance is used to invoke the target program in CICS

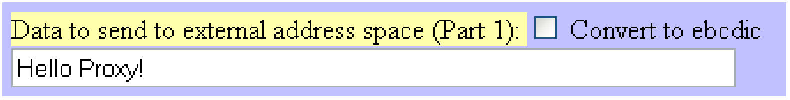
Do the following:

- ☐ Open a browser and issue the following URL:

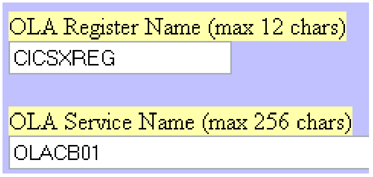
  `http://<host>:<port>/OLA_Sample2_Web/`

  Where `<host>` and `<port>` correspond to the WAS instance on your distributed workstation or server.

- ☐ Provide some text to send over:


Data to send to external address space (Part 1): ☐ Convert to ebcdic
Hello Proxy!

- ☐ Provide the WOLA registration and service name:


OLA Register Name (max 12 chars)
CICSXREG

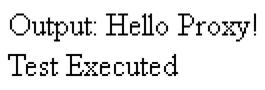
OLA Service Name (max 256 chars)
OLACB01

- ☐ Click the button to execute the WOLA call:

Note: The first invocation may take a moment as the bootstrap and JNDI lookup takes place. Once that's established, subsequent calls go much more quickly.

67 You could restart just the application server. But you've made a change to global security (the CSIv2 changes) and it's best to have that apply to everything so there's no confusion.

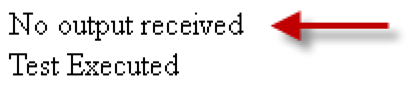
- ☐ Look for success:



```
Output: Hello Proxy!
Test Executed
```

Look in the CICS region `CEEMSG` held output for proof the call made it over to CICS. It may look garbled since by default the text was sent in ASCII, and CICS is going to display ASCII as EBCDIC.

- ☐ If you see this:



```
No output received
Test Executed
```

Then you have a problem. Check the distributed WAS `SystemOut.log` and `SystemErr.log` first to see what it says about the processing. Problems are likely to fall into three categories:

SSL Errors	When global security is on, SSL will be required for the connections. That means attempting to do a lookup of the Proxy EJB will require SSL. That requires the WAS z/OS CA certificate be part of the distributed WAS Trust file. Earlier we showed how that is done. Make sure the CA certificate is present. Look for "SSL Handshake" or other references to "SSL" in the distributed WAS `SystemOut.log` file.
Realm Errors	Look for `JSAS1479W: The target realm [value] does not match the current realm [value]` in the distributed WAS `SystemOut.log` file. There are two places where the realm name is specified: as a custom property on the JCA resource adapter connection factory, and as a "trusted realm" under the "CSIv2 outbound communications" link. The key is the realm you specify must match the realm for the target WAS z/OS system. By default that will be the host name for the Daemon, but it *may* be something different. If you think you have the value properly coded but you continue to get the `JSAS1479W` message, then check with your WAS security administrator to find out what the target WAS z/OS CSIv2 security realm name really is.
Typo Errors	These are simple errors in values such as host names, ports, userids or passwords, the JNDI name of the proxy EJB, or WOLA registration name and service name passed in. Rule out SSL and Realm problems first. Then review carefully the settings on the connection factory custom properties to be sure you have everything typed correctly.

Alternative authentication - client certificate

The authentication mechanism we illustrated earlier involved the coding of an ID and password in the distributed WAS system. That information becomes encoded in the configuration XML, and is passed up to WAS z/OS over the SSL link.

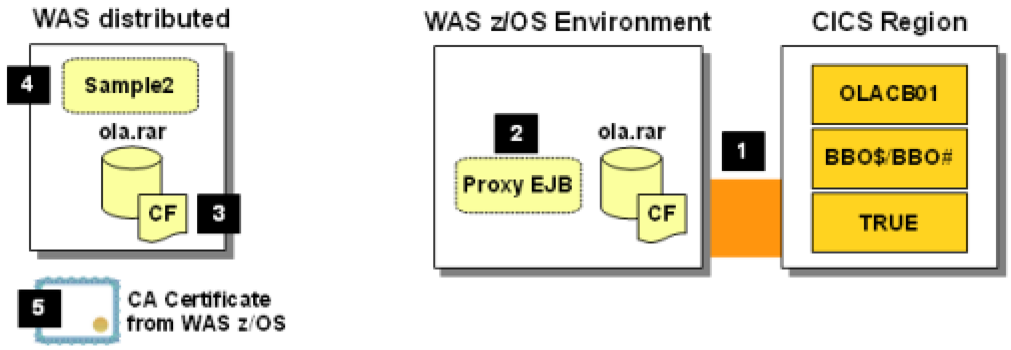
Client certificate authentication avoids the coding of ID and password. The client server certificate is asserted up to WAS z/OS, where the certificate is associated with an ID. That provides the authentication. This involves some SAF work on the mainframe.

Most everything else is the same as just discussed. In this section we will outline the tasks needed, and where a task is the same as shown above we will simply point to that location. Where a task is different we will provide details.

Do the following:

- ☐ Check to see if security is enabled for WAS z/OS (it likely is), and if so, then enable security on the distributed WAS environment. See page 37.
- ☐ Install the `ola_proxy.ear` file into the WAS z/OS environment. See page 40.
- ☐ Install `ola.rar` into the distributed WAS environment and create a JCA connection factory. See page 41.
- ☐ Install `OLASample2.ear` into the distributed WAS environment. See page 42.
- ☐ Configure the custom properties on the `ola.rar` connection factory (see page 42), except *leave blank* the `RemoteJNDIUserName` and `RemoteJNDIPassword` properties[68]. Code the other properties as indicated starting on page 42.
- ☐ See page 45 and look for the checkbox to-do item that starts with "The next step involves importing the Certificate Authority (CA) certificate from your z/OS system into your distributed WAS trust store." Follow the steps indicated there.

Your environment up to this point should look something like this:

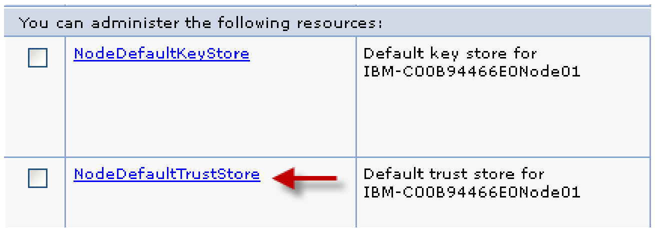


Where 1 represents all the WOLA setup work in WAS z/OS server and CICS as provided earlier in this document; 2 represents the installation of the Proxy EJB into the WAS z/OS server; 3 represents the installation of the ola.rar resource adapter into your distributed WAS environment, along with a JCA connection factory with the custom properties updated to specify the location of the Proxy EJB; 4 represents the installation of the sample application that will drive CICS across the Proxy connection; and 5 represents the import of the CA certificate from WAS z/OS into the distributed trust store.

The next step in this process is to get the CA certificate and the root certificate from the WAS distributed environment up into z/OS SAF. That involves some mainframe SAF work, so you may wish to work with your security administrator.

Continue by doing the following:

- ☐ Log into your distributed WAS Admin Console and go to *Security → SSL certificate and key management*, then click on the link for *Key stores and certificates*
- ☐ Click on the link for "NodeDefaultTrustStore":

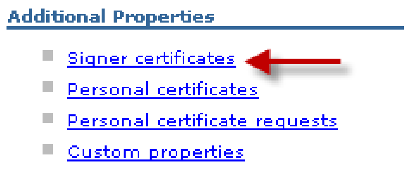


You can administer the following resources:

☐	NodeDefaultKeyStore	Default key store for IBM-C00B94466E0Node01
☐	NodeDefaultTrustStore	Default trust store for IBM-C00B94466E0Node01

68 The point of this client certificate authentication is to avoid coding a userid and password on the distributed WAS server.

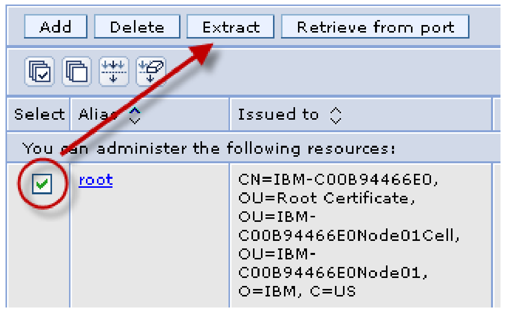
- ☐ Click on the link for "Signer certificates":



Additional Properties

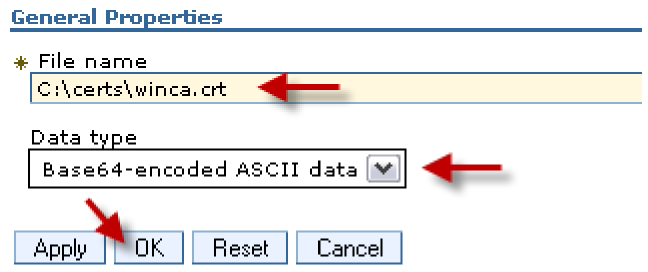
- Signer certificates
- Personal certificates
- Personal certificate requests
- Custom properties

- ☐ Check the box next to the "root" certificate and then click on the "Extact" button:



Select	Alias	Issued to
You can administer the following resources:		
✓	root	CN=IBM-C00B94466E0, OU=Root Certificate, OU=IBM-C00B94466E0Node01Cell, OU=IBM-C00B94466E0Node01, O=IBM, C=US

- ☐ Next, specify a path and file name for the certificate, set the "Data type" value and click "OK":



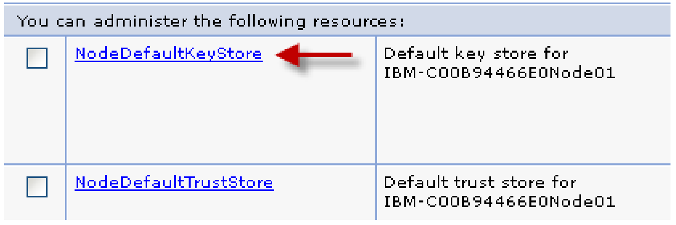
General Properties

* File name
C:\certs\winca.crt

Data type
Base64-encoded ASCII data

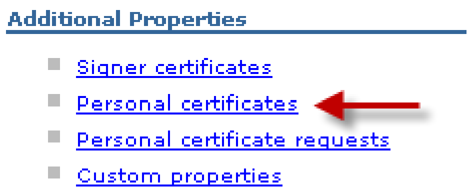
Note: the file name may be whatever you wish, and on Windows if the path does not exist it will create it. The "Data type" should be "Base64-encoded ASCII data" as shown.

- ☐ Go back to *Security* → *SSL certificate and key management*, then click on the link for *Key stores and certificates.* Then click on the link for "NodeDefaultKeyStore":



You can administer the following resources:		
☐	NodeDefaultKeyStore	Default key store for IBM-C00B94466E0Node01
☐	NodeDefaultTrustStore	Default trust store for IBM-C00B94466E0Node01

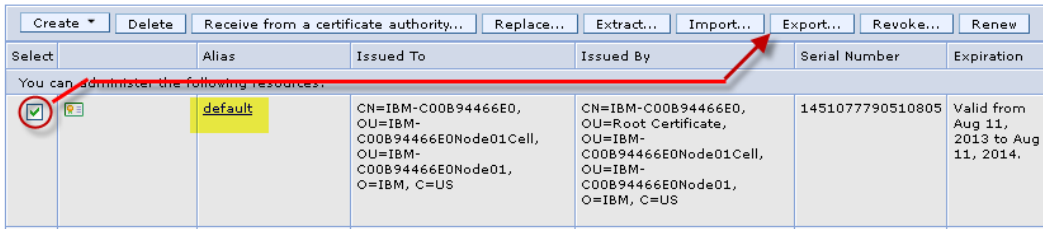
- ☐ Click on the link for "Personal certificates":



Additional Properties

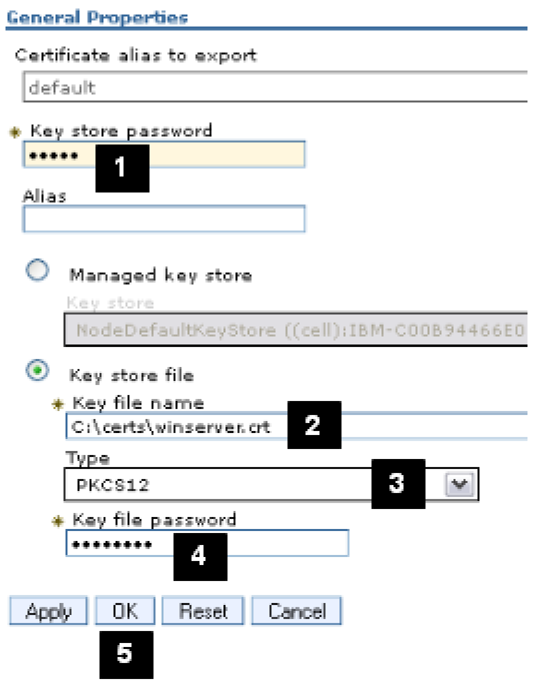
- Signer certificates
- Personal certificates
- Personal certificate requests
- Custom properties

☐ Check the box next to the "default" certificate and then click on the "Export" button:



Select		Alias	Issued To	Issued By	Serial Number	Expiration
You can administer the following resources:						
✓		default	CN=IBM-C00B94466E0, OU=IBM-C00B94466E0Node01Cell, OU=IBM-C00B94466E0Node01, O=IBM, C=US	CN=IBM-C00B94466E0, OU=Root Certificate, OU=IBM-C00B94466E0Node01Cell, OU=IBM-C00B94466E0Node01, O=IBM, C=US	1451077790510805	Valid from Aug 11, 2013 to Aug 11, 2014.

☐ Then do the following and click OK:



General Properties

Certificate alias to export: default

* Key store password: ●●●●● 1

Alias:

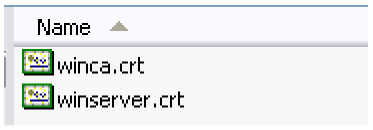
○ Managed key store
Key store: NodeDefaultKeyStore ((cell):IBM-C00B94466E0

● Key store file
* Key file name: C:\certs\winserver.crt 2
Type: PKCS12 3
* Key file password: ●●●●●●●●● 4

Notes:

1. The "Key store password" refers to the password for your distributed WAS server key store. By default that is "WebAS" but it may be different on your workstation.
2. Provide a path and file name for the exported certificate. We're showing it being exported to the same folder as the CA cert (`C:\certs`) and with a file name of `winserver.crt`.
3. The "Type" must be "PKCS12"
4. The "Key file password" refers to the password that will be used for the file that is created when you click "OK". You will need that password when processing the SAF commands in z/OS, so type carefully and note the value of the password.
5. The OK button processes the request.

☐ Go to your output location and verify the certificate files are present:



Name
winca.crt
winserver.crt

- ☐ The next step is to create the data sets on z/OS that will receive the certificate files FTP'd up from your workstation. Both are very small and have the same allocation values. Using ISPF Option 3.2, allocate two data sets:
  - ○ Name: **`WIN.CA.CRT`**

    Allocation:

```
Data class . . . . . .
 Space units . . . . . TRACK

 Average record unit   U
 Primary quantity  . . 2
 Secondary quantity    2
 Directory blocks  . . 0
 Record format . . . . VB
 Record length . . . . 84
 Block size  . . . . . 27998
 Data set name type
```

  - ○ Name: **`WIN.SERVER.CRT`**

    Allocation:

```
Data class . . . . . .
 Space units . . . . . TRACK

 Average record unit   U
 Primary quantity  . . 2
 Secondary quantity    2
 Directory blocks  . . 0
 Record format . . . . VB
 Record length . . . . 84
 Block size  . . . . . 27998
 Data set name type
```

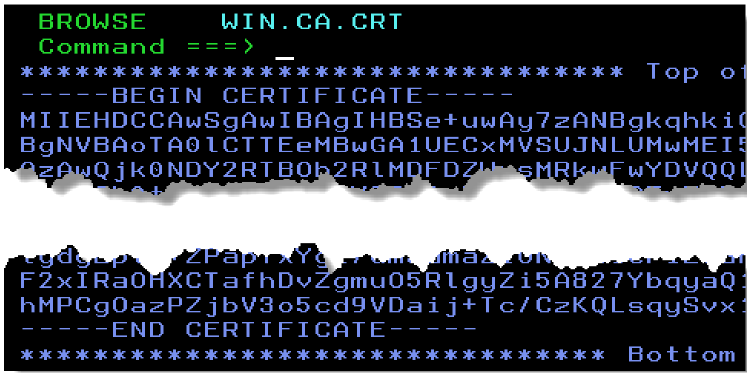
- ☐ FTP the CA certificate in *ASCII mode* up to the `WIN.CA.CRT` data set:

```
ftp> ascii
200 Representation type is Ascii NonPrint
ftp> put winca.crt 'WIN.CA.CRT'
200 Port request OK.
125 Storing data set WIN.CA.CRT
250 Transfer completed successfully.
ftp: 1500 bytes sent in 0.00Seconds 1500000.00Kbytes/sec.
ftp>
```

- ☐ FTP the server certificate in *binary mode* up to the `WIN.SERVER.CRT` data set:

```
ftp> binary
200 Representation type is Image
ftp> put winserver.crt 'WIN.SERVER.CRT'
200 Port request OK.
125 Storing data set WIN.SERVER.CRT
250 Transfer completed successfully.
ftp: 3770 bytes sent in 0.00Seconds 3770000.00Kbytes/sec.
ftp>
```

- ☐ In TSO, *browse* the `WIN.CA.CRT` data set. It should be semi-readable, and look something like this:

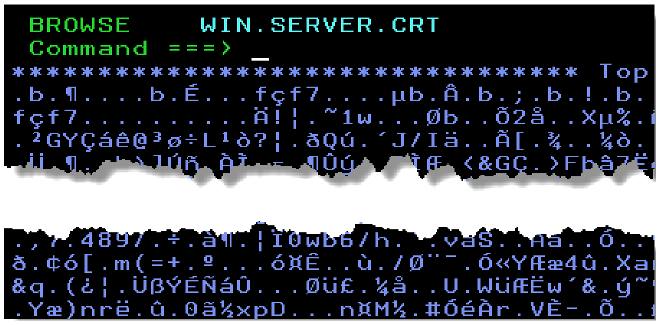


```
 BROWSE     WIN.CA.CRT
 Command ===> _
********************************* Top of
-----BEGIN CERTIFICATE-----
MIIEHDCCAwSgAwIBAgIHBSe+uwAy7zANBgkqhki
BgNVBAoTA0lCTTEeMBwGA1UECxMVSUJNLUMwMEI5
QzAwQjk0NDY2RTBOb2RlMDFDZW [illegible] sMRkwFwYDVQQ
```

```
[illegible]
F2xIRaOHXCTafhDvZgmu05RlgyZi5A827YbqyaQ
hMPCg0azPZjbV3o5cd9VDaij+Tc/CzKQLsqySvx
-----END CERTIFICATE-----
******************************** Bottom
```

Note the `BEGIN CERTIFICATE` and `END CERTIFICATE` lines. Those are important and should be in the data set.

- ☐ Now *browse* the `WIN.SERVER.CRT` data set. This is less readable:



```
 BROWSE     WIN.SERVER.CRT
 Command ===> _
******************************** Top
.b.¶....b.É...fçf7....µb.Â.b.;.b.!.b.
fçf7..........Ä!¦.˜1w...Øb..Õ2å..Xµ%.
.²GYÇáê@³ø÷L¹ò?¦.ðQú.´J/Iä..Ã[.¾..¼ò.
[illegible]
```

```
[illegible]
ð.¢ó[.m(=+.º...ó¤Ê..ù./Ø¨¯.Ó«YÆæ4û.Xa
&q.(¿¦.ÜßÝÉÑáÛ...Øü£.¼å..U.WüÆËw´&.ý˜
.Yæ)nrë.û.0ã½xpD...n¤M½.#ÓéÀr.VÈ-.Õ..
```

This is a password protected binary file.

- ☐ Go to TSO Option 6 and issue the following RACF command[69] to verify the validity of the CA certificate you uploaded:

```
RACDCERT CHECKCERT('WIN.CA.CRT')
```

You should see *something* like this:

```
Digital certificate information for CERTAUTH:
  Label: WindowsWASCA
  Certificate ID: 2QiJmZmDhZmjgeaJlYSWpqLmweLDwUBA
  Status: TRUST
  Start Date: 2013/08/11 15:19:08
  End Date:   2028/08/07 15:19:08
  Serial Number:
       >0527BEBB0032EF<
  Issuer's Name:
       >CN=IBM-C00B94466E0.OU=Root Certificate.OU=IBM-C00B94466E0Node01Cell.O<
       >U=IBM-C00B94466E0Node01.O=IBM.C=US<
  Subject's Name:
       >CN=IBM-C00B94466E0.OU=Root Certificate.OU=IBM-C00B94466E0Node01Cell.O<
       >U=IBM-C00B94466E0Node01.O=IBM.C=US<
  Subject's AltNames:
    EMail: ProfileUUID:myServer-BASE-6a40bba0-461e-415f-b7b7-2e0b754ea7a3
  Key Type: RSA
  Key Size: 2048
  Private Key: NO
```

That is RACF telling you it was able to read the certificate. The export and upload was successful. *If you don't get this result, go back through the steps carefully until you achieve this indication from RACF that the certificate is good.*

69 For other SAF vendor products, consult their documentation for the comparable command.

- ☐ Do the same thing for the server certificate. This will require an additional parameter since the certificate is password protected:

```
RACDCERT CHECKCERT('WIN.SERVER.CRT') PASSWORD('aaaaa')
```

Where *aaaaa* is the password you used when you exported the server certificate from your distributed WAS environment (see page 52). You should see something like this:

```
Digital certificate information for user Z9ASRU:

  Label: windowscrt
  Certificate ID: 2Qbp+cHi2eSmiZWElqaig5mj
  Status: TRUST
  Start Date: 2013/08/11 15:19:10
  End Date:   2014/08/11 15:19:10
  Serial Number:
       >0527BF603EAED5<
  Issuer's Name:
       >CN=IBM-C00B94466E0.OU=Root Certificate.OU=IBM-C00B94466E0Node01Cell.O<
       >U=IBM-C00B94466E0Node01.O=IBM.C=US<
  Subject's Name:
       >CN=IBM-C00B94466E0.OU=IBM-C00B94466E0Node01Cell.OU=IBM-C00B94466E0Nod<
       >e01.O=IBM.C=US<
  Subject's AltNames:
    EMail: ProfileUUID:myServer-BASE-6a40bba0-461e-415f-b7b7-2e0b754ea7a3
  Key Type: RSA
  Key Size: 2048
  Private Key: YES
```

Again, that is RACF telling you it was able to read the certificate. The export and upload was successful. *If you don't get this result, go back through the steps carefully until you achieve this indication from RACF that the certificate is good.*

- ☐ Now issue the following RACF command to add the uploaded CA certificate to the RACF database:

```
RACDCERT ADD('WIN.CA.CRT') CERTAUTH TRUST WITHLABEL('WindowsWASCA')
```

- ☐ Next, connect that CA certificate to the Controller Region ID keyring:

```
RACDCERT CONNECT(CERTAUTH LABEL('WindowsWASCA') RING(aaaa)) ID(bbbb)
```

Where:

  - ***aaaa*** is the name of the WAS z/OS keyring that is connected to the Controller Region ID where the Proxy EJB is deployed
  - ***bbbb*** is the Controller Region ID
  - The `LABEL()` value matches what you used on the `RACDCERT ADD` command. Case matters for these label values.

- ☐ The final RACF step is to add the distributed WAS server certificate to the RACF database. This is issued as one command[70]:

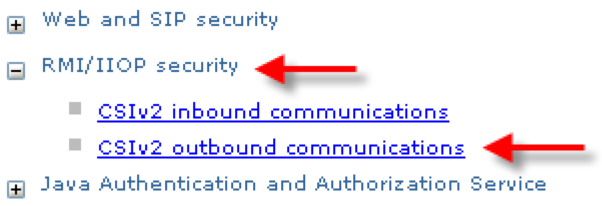
```
RACDCERT ADD('WIN.SERVER.CRT') ID(cccc)
                                    WITHLABEL('windowscrt') PASSWORD('dddd')
```

Where:

  - ***cccc*** is the ID of the Servant Region where the Proxy EJB runs
  - ***dddd*** is the password file used when you exported the server certificate on your distributed WAS server (see page 52)

70 It's broken across two lines here simply to keep it readable. One line required a font size of 7 points and that's getting a bit too small.

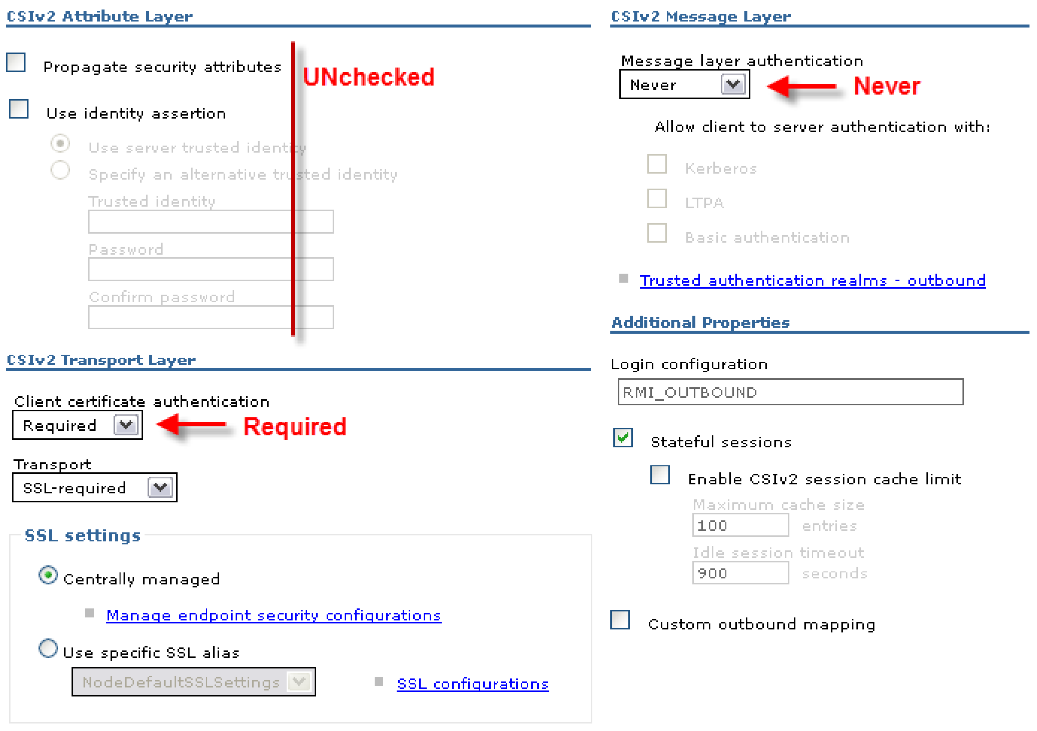
- ☐ In your *distributed* WAS server Admin Console, go to *Security* → *Global security*, then expand the "RMI/IIOP security" section and click on "CSIv2 outbound communications":



⊞ Web and SIP security
⊟ RMI/IIOP security
- CSIv2 inbound communications
- CSIv2 outbound communications

⊞ Java Authentication and Authorization Service

- ☐ Then set the outbound values like this:



CSIv2 Attribute Layer

☐ Propagate security attributes
☐ Use identity assertion
(● Use server trusted identity
○ Specify an alternative trusted identity
Trusted identity
Password
Confirm password)

UNchecked

CSIv2 Transport Layer

Client certificate authentication: Required — Required

Transport: SSL-required

SSL settings

● Centrally managed
- Manage endpoint security configurations

○ Use specific SSL alias: NodeDefaultSSLSettings
- SSL configurations

CSIv2 Message Layer

Message layer authentication: Never — Never

Allow client to server authentication with:
☐ Kerberos
☐ LTPA
☐ Basic authentication

- Trusted authentication realms - outbound

Additional Properties

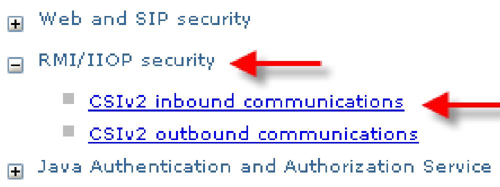
Login configuration: RMI_OUTBOUND

☑ Stateful sessions

☐ Enable CSIv2 session cache limit
Maximum cache size: 100 entries
Idle session timeout: 900 seconds

☐ Custom outbound mapping

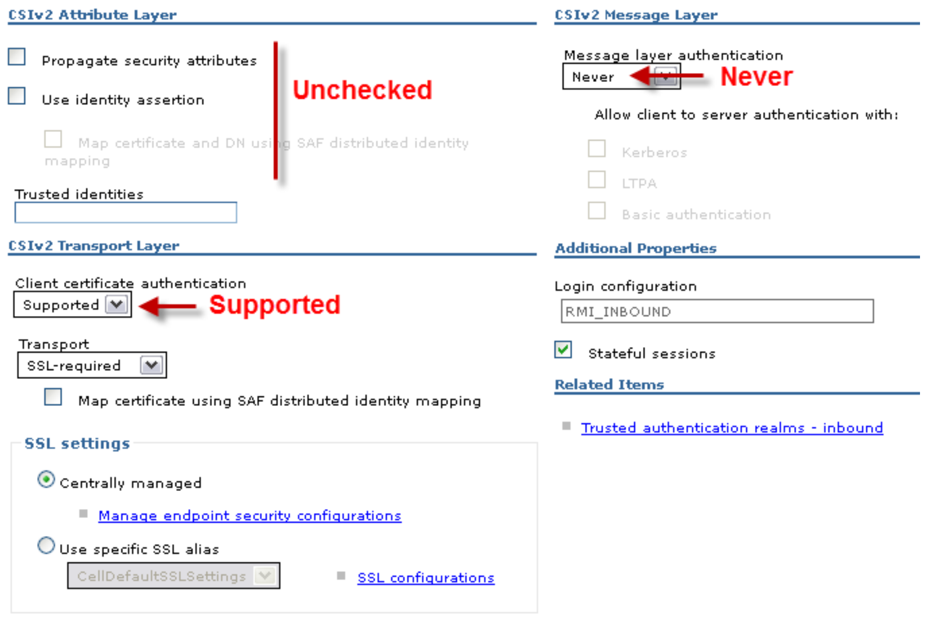
- ☐ Click "OK" and save and synchronize the changes.
- ☐ In your *z/OS* WAS server Admin Console, go to *Security* → *Global security*, then
- ☐ expand the "RMI/IIOP security" section and click on "CSIv2 inbound communications":



⊞ Web and SIP security
⊟ RMI/IIOP security
- CSIv2 inbound communications
- CSIv2 outbound communications

⊞ Java Authentication and Authorization Service

☐ Then set the inbound values like this:


CSIv2 Attribute Layer

☐ Propagate security attributes

☐ Use identity assertion

☐ Map certificate and DN using SAF distributed identity mapping

Trusted identities

CSIv2 Message Layer

Message layer authentication: Never

Allow client to server authentication with:

☐ Kerberos

☐ LTPA

☐ Basic authentication

CSIv2 Transport Layer

Client certificate authentication: Supported

Transport: SSL-required

☐ Map certificate using SAF distributed identity mapping

SSL settings

● Centrally managed

Manage endpoint security configurations

○ Use specific SSL alias: CellDefaultSSLSettings

SSL configurations

Additional Properties

Login configuration: RMI_INBOUND

✓ Stateful sessions

Related Items

Trusted authentication realms - inbound

Note: If "Client certificate authentication" is set to "Required" the Proxy function will work, but "real" WOLA inbound from, say, CICS or batch will start throwing messages in the servant region about security mismatch. But with "Supported" the Proxy function works and no security mismatch messages appear.

☐ Click "OK" and save and synchronize the changes.

☐ Stop and restart the WAS z/OS application server that hosts the Proxy EJB.

☐ Use the following command to make sure the WOLA registration from CICS is in place:

```
F <server>,DISPLAY,ADAPTER,DAEMONRGES
```

If not[71], then issue BBOC command from the CICS terminal to re-establish the registration. See page 17 for our earlier illustration of that command.

☐ Stop and restart the distributed WAS server.

☐ At this point everything should be in place to drive the OLASample2 application on your distributed WAS workstation and have it Proxy up to WAS z/OS and into CICS. The authentication should be with the assertion of the client certificate. No ID or PW was needed.

The steps to drive the request over the proxy connection is the same as we showed under "Validate proxy function up to WAS z/OS and into CICS" on page 48.

Status checkpoint

You have exercised the fundamentals of WOLA and CICS:

- Outbound from WAS into CICS using the Link Server Task
- Inbound from CICS into WAS using the OLAUTIL sample
- Several ways of starting the Link Server Task at CICS region initialization
- Allowing a userid from WAS to propagate over to CICS for `BBO#` to run under
- Used the "Development Mode" proxy support.

The environment is ready for you to begin testing the applications with WOLA.

71 If the WOLA Link Server Task in the CICS region is active, it should recognize the WAS server as having returned and automatically rebuild the registration into WAS z/OS. Earlier in V7 there were issues with this. In V8 and V8.5 it works very well.

WOLA and Batch Usage Scenarios

WOLA can be used to provide batch jobs[72] low-latency[73] access to EJB assets deployed in WAS z/OS[74]. More and more solutions are being developed and provided as EJB assets, and WOLA provides an excellent mechanism to access to those assets from batch jobs.

Pull COBOL samples from WP101490 Techdoc

Overview:

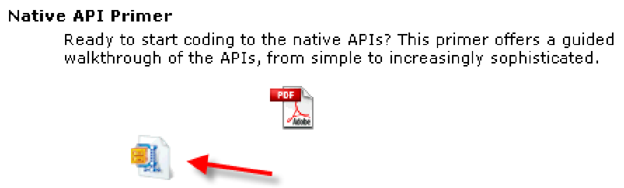
The WP101490 Techdoc has a "Native API Primer" guide with a supplied set of sample COBOL programs. That will serve as the basis for this section's exercises.

Do the following:

- ☐ Go to the WP101490 Techdoc:

  `http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101490`

- ☐ Scroll down to the sub-section titled "Native API Primer" and locate the ZIP file that is part of that section:



Native API Primer

Ready to start coding to the native APIs? This primer offers a guided walkthrough of the APIs, from simple to increasingly sophisticated.

- ☐ Download that ZIP file and extract from it **`exer2b.txt`**, **`exer2f.txt`** and **`runprog.txt`**.
- ☐ Upload those files to your *`hlq.version`*`.WOLA.SAMPLES` data set as members `EXER2B`, `EXER2F` and `RUNPROG` respectively. Transfer in such a way that the files are converted and stored as EBCDIC in the data set.

Review the SAF CBIND profile for batch job ID access

Overview:

The SAF `CBIND` profile -- specifically, the `CB.BIND.`*`<your_environment_value>`* profile[75] -- will control the ability of an outside address space such as your batch job from being being able to register into the WAS z/OS server. If the CBIND for your WAS z/OS server does not allow at least `READ` for the ID of the region trying to register, the register will fail with a RC=12, RSN=14.

Do the following:

- ☐ Check with your security administrator to see what format of `CBIND` was created for your WAS z/OS environment.

Note: The SAF class will be CBIND and the profile will be `CB.BIND.`*`<some_value>`*. What the value for *`<some_value>`* in your environment is what your security administrator must determine.

The format will one of the following, depending on how your cell was built:

(1) `CB.BIND.`*`<server_cluster_transition>`*`.**`
(2) `CB.BIND.`*`<saf_prefix>`*`.**`
(3) `CB.BIND.`*`<cell_identifier>`*`*`

72 Supported languages: COBOL, C/C++, High Level Assembler and PL/I.
73 For repetitive batch jobs the issue of per-record latency plays an important role in job completion times. Per-record latency is *additive*, which means the objective should be to reduce per-record latency as much as possible. That is what WOLA does exceptionally well.
74 That's the "inbound" model -- Batch inbound to WAS z/OS. It is also possible to do "outbound" WAS z/OS into a batch program, but that implies the batch program is started and waiting for the outbound call from WAS z/OS. That is possible, but less common. The more common use case would be outbound from WAS z/OS to a program running in CICS. If you are interested in outbound to a batch program, see WP101490 at `ibm.com/support/techdocs` and consult the WOLA "Primer" document, which provides examples as well as sample COBOL programs that "host a service" for outbound calls from WAS z/OS.
75 There is also a `CB.`*`<your_environment>`* `CBIND` profile, but that is different and does not apply to this specific topic.

The first is the format used if the generated "BRAK" job was used and no SAF prefix was specified; the second is the format if the generated BRAK job was used and a SAF prefix was provided; and the third is if you used a job that created "generic" profiles.

Determine what is used for your environment, then issue the command to update the access list for that CBIND profile as illustrated next.

- ☐ Check the Universal Access (UACC) setting for that profile. If `UACC=`**`READ`**, then no further action is needed as the CICS region ID will have `READ` via `UACC=READ`.
- ☐ However, if `UACC=`**`NONE`**, then update the access list for the `CBIND` profile with a command[76] like what's shown here. Modify according to your specific `CBIND` profile:

```
PERMIT <your_CBIND_profile> CLASS(CBIND) ID(<batch ID>) ACCESS(READ)
SETROPTS RACLIST(CBIND) REFRESH
```

Review contents of EXER2B job

Overview:

The `EXER2B` job is a relatively simple batch job that uses just three WOLA APIs -- `BBOA1REG` to register; `BBOA1INV` to invoke the target EJB; and `BBOA1URG` to unregister. A loop is built around the use of `BBOA1INV` to loop and invoke as many times as you indicate.

Do the following:

- ☐ Browse the `EXER2B` member and locate the following section (line 58):

```
        MAINLINE SECTION.
            MOVE 'EXER2B'                        TO register-name.
            MOVE 'cccccc'                        TO daemongroup.
            MOVE 'nnnnnnn'                       TO node-name.
            MOVE 'sssssss'                       TO server-name.
       *
            MOVE 10                              TO loop-limit.
       *
            MOVE 'This is a test message'        TO text-msg.
            MOVE 'ejb/com/ibm/ola/olasample1_echoHome'
                 TO service-name.
```

Note the following aspects of that portion of code:

- Placeholders are in the sample code that will be changed to match your cell, node and server ***short*** names (lines 60 - 62).

  These values are used on the `BBOA1REG` API to process the registration into the WAS z/OS server. You can see the `BBOA1REG` API in use starting on line 74 of the supplied sample.

  If you recall the `BBOC START_SRVR` also asked for the cell, node and server short names. The `BBOC` transaction was starting the Link Server Task *and* registering into the server; the `BBOA1REG` API performs *just* the registration[77].

- The `loop-limit` value is set to 10 (line 64). If you wished to loop more you would simply change this value and recompile.
- The `service-name` value (line 67) is set to the same JNDI name as used in the CICS exercises for inbound calls to the `OLASample2.ear` file.

  As explained earlier, when calling inbound to a WAS application over WOLA, the service name used by the external application is the JNDI name for the EJB module in the deployed application.

76 For a security product other than IBM RACF, modify the command to meet the equivalent command for the security product you use.
77 In fact the Link Server Task is completely out of the picture for this set of exercises. CICS is completely out of the picture as well. This is COBOL batch into WAS z/OS over WOLA. There is no CICS involved at all.

If you have `OLASample1.ear` or `OLASample2.ear` deployed then you do not need to change this value in the COBOL sample.

- ☐ Scroll down to line 74 and you'll see `CALL BBOA1REG`, which shows that API being called with the various API parameters supplied as variables[78]. No changes are needed to any of this code.
- ☐ On line 94 you will see `PERFORM UNTIL`, which is the start of the loop.
- ☐ On line 114 you will see `CALL BBOA1INV`, which shows the invoke of the target EJB with the parameters passed.
- ☐ Line 136 shows the end of the `PERFORM` loop.
- ☐ Line 140 shows `CALL BBOA1URG`, which will unregister from the WAS server.

Customize, Compile and run EXER2B

Overview:

The `EXER2B` sample has JCL to compile, but it requires customizing to your environment. Plus, the cell, node and server short names need to be supplied to the code so the `BBOA1REG` API succeeds in registering into your environment.

Do the following:

- ☐ Determine what load library data set you will compile the COBOL into.
- ☐ Customize the sample JCL's JOB card to match your environment's requirements.
- ☐ Customize the JCL to match your environment's COBOL compiler environment.
- ☐ Update the three placeholders -- **`cccccc`**, **`nnnnnnn`** and **`sssssss`** -- to match your WAS z/OS environment's cell, node and server SHORT names.
- ☐ Compile the code and look for `RC=0`.
- ☐ Customize the `RUNPROG` JCL so that the newly-compiled `EXER2B` program can be run.
- ☐ Make sure the WAS z/OS target server is up and running, along with the `OLASample1.ear` (or `OLASample2.ear`, depending on which you installed) is started.
- ☐ Submit `RUNPROG` and review the output when it completes.

Review contents of EXER2F job

Overview:

The `EXER2F` sample is like `EXER2B`, except it uses some of the "advanced" APIs. Specifically, it illustrates how to overcome one of the assumptions made by `BBOA1INV`, which is its synchronous nature -- it calls *and waits* for the Java program in WAS z/OS to return.

The "advanced" APIs allow you to call and set an "async flag," returning program control to your batch job so it can do other work. But that implies your program going back and seeing if a response has returned. More work for the COBOL programmer, but more efficient operations.

Do the following:

- ☐ Browse the `EXER2D` sample and note the following:
  - ○ It has the same placeholder values for cell, node and server short name as `EXER2B` had (starting on line 62)
  - ○ It uses `BBOA1REG` in the same way that EXER2B did (line 74)
  - ○ It sets a variable `SRQasync=1` (line 94) so the `BBOA1SRQ` (send request) API may operate in asynchronous mode. That means program control will return immediately after issuing `BBOA1SRQ` to send the request over to Java in WAS z/OS.

78 The InfoCenter has an excellent article outlining all the APIs and their parameters. Search on **`olaapis`**.

- It then sets another variable -- `RCLasync=0` (line 95) that indicates the `BBOA1RCL` API operates *synchronously*. `BBOA1RCL` is used to determine if a response has been received. It may operate asynchronously as well, but in this example it operates synchronously for the sake of simplicity *in this sample*. Sample `EXER2G` shows asynchronous for both `SRQ` *and* `RCL`, but `EXER2F` is asynchronous for *just* `BBOA1SRQ`.
- The `BBOA1CNG` (get connection) API is called (line 97). This is another "advanced" feature: the reuse of the same connection over and over. `BBOA1INV` (`EXER2B` sample) returned the connection to the pool after each invocation.
- Line 109 is the start of the `PERFORM` loop up to the limit set by `loop-limit` of 10.
- Line 129 shows `BBOA1SRQ` being called. Remember, the async flag is on for that call.
- Line 146 shows a call to "other work" to illustrate the asynchronous nature of the `SRQ` call. The "other work" is really nothing more than `DISPLAY` statement, but it shows how program control returns immediately after issuing SRQ.
- Line 151 shows `BBOA1RCL` being called. This will return the length of the response. In this example the call is *synchronous*, which means program control is held under the response is returned. But asynchronous *is possible* and is illustrated in the `EXER2G` sample.
- Line 163 shows the `BBOA1GET` API being called to get the response.
- Line 179 is the end of the `PERFORM` loop.
- When the loop is completed the `BBOA1CNR` API is called to return the connection to the pool (line 181).
- Line 193 shows `BBOA1URG` called to unregister and the program ends.

Customize, Compile and run EXER2F

Overview:

Just like `EXER2B`, the `EXER2F` sample has JCL to compile, but it requires customizing to your environment. Plus, the cell, node and server short names need to be supplied to the code so the `BBOA1REG` API succeeds in registering into your environment.

Do the following:

- ☐ Determine what load library data set you will compile the COBOL into.
- ☐ Customize the sample JCL's JOB card to match your environment's requirements.
- ☐ Customize the JCL to match your environment's COBOL compiler environment.
- ☐ Update the three placeholders -- **cccccc**, **nnnnnnn** and **sssssss** -- to match your WAS z/OS environment's cell, node and server SHORT names.
- ☐ Compile the code and look for `RC=0`.
- ☐ Customize the `RUNPROG` JCL so that the newly-compiled `EXER2F` program can be run.
- ☐ Make sure the WAS z/OS target server is up and running, along with the `OLASample1.ear` (or `OLASample2.ear`, depending on which you installed) is started.
- ☐ Submit `RUNPROG` and review the output when it completes.

Status checkpoint

You have exercised the fundamentals of WOLA and inbound work from batch. The samples illustrated some essential points about the WOLA APIs. The WP101490 "Primer" document covers the APIs in much greater detail.

Appendix A: Enabling WOLA in WAS z/OS Version 7

The mechanism used to enable WOLA in a Version 7 runtime environment is different from that for Version 8 and Version 8.5. The *concepts* are similar; the shell script and syntax is different.

WebSphere Application Server for z/OS Version 7 Information Center

The URL for the V7 InfoCenter is:

`http://pic.dhe.ibm.com/infocenter/wasinfo/v7r0/index.jsp`

Determine the Fixpack level of WAS z/OS V7

Overview:

Knowing the fixpack level of the WAS z/OS runtime you have is always a good thing, but with WOLA and WAS z/OS V7 it's particularly important. In this section you will simply capture that information.

Do the following:

- ☐ Open a Telnet or OMVS session into UNIX Systems Services on the z/OS system
- ☐ Change directories to the `/profiles/default/bin` directory[79] of the cell environment you intend to use for this WOLA test.
- ☐ Issue the command:

```
./versionInfo.sh
```

- ☐ In the output, find the following:

```
Name         IBM WebSphere Application Server for z/OS
Version      x.x.x.x
```

Where *x.x.x.x* is the version and fixpack level for the WAS installation.

- ☐ Write that value here[80]: ________________

Note: Make sure you are at fixpack 7.0.0.21 or above. This is particularly important if you plan to use WOLA with CICS TS 4.2, but the recommendation applies regardless of the level of CICS you're at.

Pre-allocate a LIBRARY and SAMPLES data sets

Overview:

In V8 and 8.5 the `copyzOS.sh` shell script had the ability to allocate the target data sets, but the V7 `olaInstall.sh` shell script does not. Therefore, you must pre-allocate it.

Do the following:

- ☐ Allocate a MODULES data set[81]:

Name:	*hlq.version*.WOLA.MODULES Where *hlq* is a qualifier of your choosing, and *version* is the fixpack version level with the dots removed; for example: 70027.

Allocation:	`Space units . . . . . TRACK`
	`Primary quantity  . . 40`
	`Secondary quantity    2`
	`Directory blocks  . . 10`
	`Record format . . . . U`
	`Record length . . . . 0`
	`Block size  . . . . . 32760`
	`Data set name type    LIBRARY`

79 Either `/AppServer` or `/DeploymentManager` ... both will work equally well.
80 If your fixpack level is 7.0.0.3 or lower, then the WOLA support will not be present. WOLA came into being with 7.0.0.4. 7.0.0.12 or higher would be a better level to be at. The latest fixpack for V7 would be better still.
81 Version 7 InfoCenter search: `tdat_enableconnector`

☐ Allocate a SAMPLES data set[82]:

Name:	`hlq.version.WOLA.SAMPLES` Where `hlq` is a qualifier of your choosing, and `version` is the fixpack version level with the dots removed; for example: 70027.

Allocation:	`Space units . . . . . TRACK`
	`Primary quantity  . . 40`
	`Secondary quantity    2`
	`Directory blocks  . . 10`
	`Record format . . . . FB`
	`Record length . . . . 80`
	`Block size  . . . . . 9040`
	`Data set name type    PDS`

Run olaInstall.sh shell script with INIT

Overview:

The V7 shell script `olaInstall.sh` will do two things: one, it will build symlinks from the node's configuration file system to the product file system where the WOLA files and modules exist; and second, it will copy the modules to your `hlq.version.WOLA.MODULES` data set.

Do the following:

☐ In a Telnet or OMVS session issue the following command from the *application server* node's `/profiles/default/bin` directory:

```
./olaInstall.sh INIT 'hlq.version.WOLA.MODULES'
```

Where `hlq.version` is the value you provided when you allocated the data set earlier.

You should see many messages come out indicating symlinks being built, then you should be returned to your UNIX command prompt.

Note: If you see the following message:

```
olaModules already exists
```

Then the node has already been initialized. If you would like to copy out *just* the modules, then the command would be:

```
./olaInstall.sh OLAMODS 'hlq.version.WOLA.MODULES'
```

☐ Browse the `hlq.version.WOLA.MODULES` data set and verify it has been populated with `BBO*` modules.

☐ Change directories to the Deployment Manager's `/profiles/default/bin` directory and run the same command again[83]:

```
./olaInstall.sh INIT 'hlq.version.WOLA.MODULES'
```

Why? We have you do this *just in case* you deploy an application that uses WOLA classes that requires the imbedded tooling workbench of the Admin function to compile the project. In that case it'll need access to the WOLA files. Running `INIT` against the Deployment Manager gives it access.

82 Version 7 InfoCenter search: `cdat_olasamples`

83 This will result in the modules in the LIBRARY data set being overwritten. That's okay. They'll be the same modules.

Copy the WOLA samples to your samples data set

Overview:

The V7 shell script `olaInstall.sh` has no facility for copying out the samples[84]. Copying the samples from the file system to your FB 80 PDS is a manual process.

The samples are located in the *product file system* (not the node's file system) at the following location:

`/<mount>/mso/OLA/samples`

Do the following:

- ☐ Locate the `/mso/OLA/samples` directory and copy the path information.
- ☐ From TSO Option 6, issue the following command:

```
OGET '/<path>/CSDUPDAT.jclsamp' 'hlq.version.WOLA.SAMPLES(CSDUPDAT)'
```

  Where `<path>` is the full path to the directory in which the WOLA samples are located, and `hlq.version` is the qualifier on the SAMPLES data set allocated before.

- ☐ Browse the **`hlq.version.WOLA.SAMPLES(CSDUPDAT)`** member and make sure it looks readable[85].
- ☐ Repeat the same `OGET` command, modifying the command appropriately, for the following files:

☑	CSDUPDAT.jclsamp
☐	DFHPLTOL.jclsamp
☐	OLAMAP.jclsamp
☐	OLAUTIL.jclsamp
☐	OLACB01.jclsamp
☐	OLACB02.jclsamp

Note: There are other samples beyond those shown here. These are the samples we illustrated in this document. Read the `@@README` file to get a sense for what each sample provides. If others are of interest to you, copy those as well.

Create two cell-level environment variables

Overview:

Two cell-level environment variables must be created to allow WOLA to work. The variables and the process are very much like what's done for V8 and V8.5[86].

The variables **must** be created at the cell level.

Do the following:

- ☐ Create a cell-level variable with the following name and value:

Name:	WAS_DAEMON_ONLY_enable_adapter
Value:	true

- ☐ Save your changes.
- ☐ Create a second cell-level variable with the following name and value:

Name:	ola_cicsuser_identity_propagate
Value:	1

84 The V8/V8.5 shell script `copyZOS.sh` does, however.
85 Just a quick test to insure there were no code page conversion issues.
86 See "Create cell-level WOLA "enable adapter" environment variable" starting on page 8 for an illustration of how to create the variables.

- ☐ Save and synchronize all changes.

Install the ola.rar JCA resource adapter

Overview:

For WOLA *outbound* communications from Java in WAS to an external address space a supplied JCA resource adapter is used. Enabling this function involves installing the JCA adapter (very simple) and configuring a Connection Factory (also very simple).

The process is just like it is with WAS z/OS V8 or V8.5.

Do the following:

- ☐ Follow the instructions as provided under "Install the ola.rar resource adapter and create a connection factory" starting on page 9.

Review CBIND Profile

Overview:

The CBIND profile is what allows or rejects outside access to the WAS z/OS server controller region. This is how you control what external address spaces are allowed to register into the server.

The process for V7 is very much like the process for V8 and V8.5.

Do the following:

- ☐ See "Review the WAS z/OS server CBIND profile for CICS region ID access" starting on page 16.
- ☐ Perform the update to your CBIND profile based on the known external address space IDs you intend to use when registering into the WAS z/OS server.

Stop and restart the entire cell environment

Overview:

The two cell-level environment variables you created earlier require a restart of the entire cell (*including the Daemon*) to take effect. Once restarted a simple check of the Daemon output will validate the WOLA enablement.

Do the following:

- ☐ See "Stop and restart the entire cell" starting on page 9. The messages you look for to validate WOLA enablement are the same in V7 as they are in V8 and V8.5.

Appendix B: Starting Link Server Task at CICS Startup

Earlier we showed the starting of the Link Server Task using the BBOC control transaction entered through a CICS terminal session. That was done because it was simple and achieved the objective of validating the basic setup.

In this section we will illustrate three other means of initializing the Link Server task at CICS initialization:

1. Using the `BBOACPL2` sample PLT program and `INITPARM=()` in the SIP
2. Using the `BBOACPL3` sample PLT program
3. Using a Sequential Terminal to input the `BBOC START_SRVR` command

In this section we will describe in detail how each is accomplished.

Comparing the capabilities of the approaches (unfinished)

	3270 BBOC	PLT BBOACPLT	PLT BBOACPL2	PLT BBOACPL3	Sequential Terminal
Start the TRUE	Yes	Yes	No	Yes[4,5]	Yes[5]
Stop the TRUE	Yes	No[1]	No[1]	No[1]	Yes[6]
Start Link Server Task	Yes	No	Yes[2]	Yes[4]	Yes
Start Link Server Long String	Yes	No	No[3]	Yes[4]	Yes
Stop the Link Server Task	Yes	No	No[1]	No[1]	Yes[6]

Notes:

1 - A `TYPE=FINAL` PLT program can be written, but it is not a supplied sample.

2 - `BBOACPL2` is used with `INITPARM`, which has length limitations. If the `INITPARM` string can be contained within the limits imposed by CICS, then `BBOACPL2` may be used to start the Link Server Task.

3 - Again, if the limits imposed by `INITPARM` are an issue, then `BBOACPL2` is not the solution.

4 - The `BBOACPL3` sample PLT may or may not be included with your level of WAS z/OS, depending on the version and fixpack. See "BBOACPL3 and longer BBOC commands" on page 67 for more.

5 - If you use `BBPACPL3` or Sequential Terminal to start the TRUE, then you would *not* use `BBOACPLT`. Said another way, if you use `BBOACPLT` to start the TRUE, then do not use `BBOACPL3` or Sequential Terminal to do the same thing of you'll encounter an error because the TRUE is already started.

6 - Using sequential terminal to stop the TRUE and link server is possible, but in this document we do not show an example of it. Many WOLA users use system automation tasks to stop the link server prior to terminating the CICS region.

BBOACPL2 and INITPARM=()

Overview:

The `BBOACPL2` program accepts commands from a CICS `INITPARM=()` string and passes them to the BBOC control program to start the Link Server Task.

However, the limitation of this approach is that CICS imposes a 60-character limit on the `INITPARM=()` string, which may or may not accommodate your `BBOC START_SRVR` command, depending on the parameters you include[87].

Do the following:

- ☐ Verify that you ran the `DFHPLTOL` sample job
- ☐ Verify that `PLTPI=OL` has been added to your CICS region Session Initialization Parameters (SIP) member
- ☐ Provide an `INITPARM=()` line in the SIP member with the following syntax:

```
INITPARM=(BBOACPL2='STA RGN=aaaa DGN=bbbb NDN=cccc SVN=dddd SVC=*')
```

87 There are two approaches to fitting the command into `INITPARM=()` ... (1) use the command and parameter abbreviations rather than the full values; and (2) omit parameters where the default value is acceptable. But that may not prove sufficient and you may need to use something other than `INITPARM` to start the Link Server Task, such as `BBOACPL3` or the Sequential Terminal.

Where:

aaaa = the WOLA registration name you wish to use

bbbb = the cell SHORT name

cccc = the node SHORT name

dddd = the server SHORT name

Additional parameters may be added as needed, but keep in mind the 60 character limit imposed by CICS for the contents of `INITPARM=()`.

- ☐ Stop and restart your CICS region.
- ☐ Look in the CICS region `BBOOUT` held output for messages indicating both the TRUE and the Link Server Task having started.
- ☐ From the z/OS operator console issue the following z/OS MODIFY command to display the registrations into your server:

```
F aaaaaa,DISPLAY,ADAPTER,DAEMONRGES
```

Where ***aaaaaa*** is the JOBNAME for the target WAS z/OS application server.

BBOACPL3 and longer BBOC commands

Overview:

The `BBOACPL3` program provides the ability to code complete BBOC commands and have them executed at CICS initialization as part of PLT processing.

Note: The `BBOACPL3` program is one you may or may not have as part of the supplied samples, depending on the level of WAS z/OS you have. The instructions below will be structured for both cases -- where your copy of WAS z/OS has `BBOACPL3` supplied, and where it does not.

Do the following:

- ☐ Look in your ***hlq.version*.WOLA.SAMPLES** data set and see if a copy of `BBOACPL3` exists. If yes, then continue; if not, then go to "If BBOACPL3 not part of your copy of samples" on page 68.
- ☐ If `BBPACPL3` is present in the samples, then locate the section of the code labeled, "Constants/Literals/Variables/Data definitions." This is where you provide your customized command to start the Link Server task[88]. Customize according to your requirements.
- ☐ Customize the JCL so the the assemble and link edit operates properly in your environment, with the `LKED.SYSLMOD` link edit statement pointing to your *hlq.version*`.WOLA.MODULES` data set.
- ☐ Submit the job and look for RC=0.
- ☐ Look in the `DFHPLTOL` job and add or uncomment the following line under `C.SYSIN`:

```
DFHPLT TYPE=ENTRY,PROGRAM=BBOACPL3
```

- ☐ Submit `DFHPLTOL` again and look for RC=0.
- ☐ Look in the `CSDUPDAT` job for the following:

```
DEFINE PROGRAM(BBOACPL3) GROUP(BBOACSD) LANGUAGE(ASSEMBLER)
       DATALOCATION(ANY) EXECKEY(CICS)
       CONCURRENCY(THREADSAFE) API(OPENAPI)
```

If it's *not* present, then add it and run the job again. If it's there and you've already run `CSDUPDAT` earlier, then there's nothing more to do.

88 The sample contains code for *two commands*. If you only wish to issue one command, then either (a) code the same STA command twice, which will result in a RC=8/RSN=8 (registration token already exists) for the second issuance; or (b) modify the sample code so only one command is issued.

If you run the job again you'll get `RC=8` because many of the `DEFINE` statements will reference programs already added to the CSD. If you look at the output carefully you'll see the add of `BBOACPL3` completed successfully.

- ☐ Stop and restart the CICS region.
- ☐ Check the CICS region `BBOOUT` held output for messages indicating the Link Server Task has been started.
- ☐ From the z/OS operator console issue the following z/OS MODIFY command to display the registrations into your server:

```
F aaaaaa,DISPLAY,ADAPTER,DAEMONRGES
```

Where ***aaaaaa*** is the `JOBNAME` for the target WAS z/OS application server.

If BBOACPL3 not part of your copy of samples

The following is the source for the `BBOACPL3` sample. Place this in your *`hlq.version`*`.WOLA.SAMPLES` data set, then go back and follow the steps provided above for the case where `BBOACPL3` was supplied as part of the samples.

```
//OLAPL3  JOB (),'ME',
//   MSGCLASS=H,NOTIFY=&SYSUID
//*
//* BUILD SAMPLE PLTPI MODULE FOR TESTING OLA
//*
//KIXPROC JCLLIB ORDER=MVSBUILD.CICSTS32.CICS.SDFHPROC
//*
//CMP1 EXEC DFHEITAL,INDEX='MVSBUILD.CICSTS32.CICS',
//         DSCTLIB='BOSS.OLA.JCL'
//TRN.SYSIN  DD *
*ASM XOPTS(CICS,SP,NOPROLOG,NOEPILOG)
         TITLE 'BBOACPL3 - WAS Optimized Local Adapters CICS PLTPI 3'
*/*START OF SPECIFICATIONS *******************************************
*
* DESCRIPTIVE-NAME: WAS z/OS Optimized Adapters CICS PLT Init program
*                   Support Start Server and Register requests
*
* CSECT NAME: BBOACPL3
*
* COMPONENT:  boss_adaptor
*
* EYE-CATCHER: none
*
* PROPRIETARY STATEMENT=
* IBM Confidential
* OCO Source Materials
* 5655-I35 (C) Copyright IBM Corp. 2008
* The source code for this program is not published or otherwise
* divested of its trade secrets, irrespective of what has been
* deposited with the U.S. Copyright Office.
* Status = H28W700
*
*
* FUNCTION:  Run as CICS PLTPI program and complete BBOC requests -
*            to do Start Server or Register requests
*
* METHOD-OF-ACCESS:
*
* SIZE:
*
* SERIALIZATION: None
*
* DEPENDENCIES: None
*
* EXTERNAL CLASSIFICATION: NONE
* END OF EXTERNAL CLASSIFICATION:
*
* CHANGE-ACTIVITY:
*
*   Flag Reason      Release  YYYYMMDD Origin  Description
*   #XXXXXX          HBBO800  20120502  JTM    Created for sample to
*   #                                          issue BBOC calls at
*   #                                          startup of CICS
*
*END OF SPECIFICATIONS *********************************************/
*
         DFHREGS  ,
BBOACPL3 AMODE 31
BBOACPL3 RMODE ANY
BBOACPL3 CSECT
*
         DFHEIENT CODEREG=11,EIBREG=10
*
         WTO   'BBOA9940I WAS z/OS OLA CICS PLT init 3 start.'
*
*-------------------------------------------------------------------
* Address the EIB.
* This is needed, because EIBREG above is giving us the USER EIB
* but we are compiled to use the SYSTEM EIB.
*-------------------------------------------------------------------

         EXEC CICS ADDRESS EIB(DFHEIBR)
*
         EXEC  CICS INQUIRE SYSTEM RELEASE(CICS_Release)
*
         CLC   CICS_Release,=CL4'0640'
         BE    CICS_REL_OK
         CLC   CICS_Release,=CL4'0650'
         BE    CICS_REL_OK
         CLC   CICS_Release,=CL4'0660'
         BE    CICS_REL_OK
         CLC   CICS_Release,=CL4'0670'
         BE    CICS_REL_OK
*
         WTO   'BBOA9930W Invalid CICS release. Requires CICS TS 3.1+'
*
CICS_REL_OK DS 0H
*
* Move Command 1 into BBOC_Parms
*
         MVC   BBOC_TX,=CL5'BBOC '
         MVC   BBOC_PCMD,BBOC_Cmd1
         LA    R2,250
         STH   R2,BBOC_Parms_L
*
* Issue command 1
*
*
         EXEC  CICS LINK PROGRAM('BBOACNTL')                           X
               COMMAREA(BBOC_Parms)                                    X
               LENGTH(BBOC_Parms_L) NOHANDLE
*
         CLC   EIBRESP,=F'0'
         BE    BBOC_Req_CkNext
         WTO   'BBOA9951E Error in OLA BBOC request. Check BBOQ.'
         B     Return
*
BBOC_Req_CkNext DS 0H
*
* Move Command 2 into BBOC_Parms
*
```

```
         MVC   BBOC_TX,=CL5'BBOC '
         MVC   BBOC_PCMD,BBOC_Cmd2
         LA    R2,250
         STH   R2,BBOC_Parms_L
*
* Issue command 2
*
*
         EXEC  CICS LINK PROGRAM('BBOACNTL')                          X
               COMMAREA(BBOC_Parms)                                   X
               LENGTH(BBOC_Parms_L) NOHANDLE
*
         CLC   EIBRESP,=F'0'
         BE    BBOC_Req_OK
         WTO   'BBOA9951E Error in OLA BBOC request. Check BBOQ.'
         B     Return
*
BBOC_Req_CkNext2 DS 0H
         WTO   'BBOA9941I WAS z/OS OLA CICS Request Successful.'
*
* Look for a BBOAxxxxE error message back from BBOC. If yes there was
* an error. Indicate this and return.
*
         CLI   BBOC_Parms+8,C'E'
         BNE   BBOC_Req_Ok
         WTO   'BBOA9951E Error in OLA BBOC request. Check BBOQ.'
         B     Return
*
BBOC_Req_Ok DS 0H
         WTO   'BBOA9941I WAS z/OS OLA CICS Request Successful.'
*
* ---------------------------------------------------------------------
*
* Return:  Exit program
*
* ---------------------------------------------------------------------
Return_OK DS  0H
         WTO   'BBOA9945I WAS z/OS OLA CICS PLT init 3 ending.'
*
Return   DS  0H
         DFHEIRET
*
* ----------------------------------------------------------------------
*
* Constants/Literals/Variables/Data definitions
*
* ----------------------------------------------------------------------
REG_ON   EQU  X'80'
SRVR_ON  EQU  X'40'
         DS     0F
Clear50  DC CL50'                                                  '
BBOC_Cmd1 DS 0CL250
Cmd1a    DC CL50'STA RGN=OUTTRAN1 DGN=WAS00 NDN=NDN1 SVN=BBOS001   '
Cmd1b    DC CL50'TXN=Y SEC=N SVC=* TRC=2 MXC=10 MNC=1              '
Cmd1c    DC CL50'                                                  '
Cmd1d    DC CL50'                                                  '
Cmd1e    DC CL50'                                                  '
*
BBOC_Cmd2 DS 0CL250
Cmd2a    DC CL50'STA RGN=OUTTRAN2 DGN=WAS00 NDN=NDN1 SVN=BBOS001   '
Cmd2b    DC CL50'TXN=Y SEC=N SVC=* TRC=2 MXC=10 MNC=1              '
Cmd2c    DC CL50'                                                  '
Cmd2d    DC CL50'                                                  '
Cmd2e    DC CL50'                                                  '
         DS     0F
*
```

```
Movedata  MVC   0(*-*,R6),0(R7)
*
DFHEISTG DSECT
         DS     0F
         DFHEISTG                      EXTENDED SAVE AREA FOR CICS
BBOC_Parms DS 0D
BBOC_TX    DS CL5
BBOC_PCMD  DS 0CL250
BBOC_P1    DS CL50
BBOC_P2    DS CL50
BBOC_P3    DS CL50
BBOC_P4    DS CL50
BBOC_P5    DS CL50
           DS 0F
BBOC_Parms_L DS H
CICS_Release      DS CL4
           DS     0F
BBOC_data      DS CL60
           DS     0F
BBOC_dataLen   DS H
           DS     0F
Request_flags  DS X
           DS     0F
*
EISTGLEN EQU *-DFHEISTG
*
         DFHEIEND
*
         END   BBOACPL3

//LKED.SYSLMOD DD DSN=MSTONE1.LATEST.SBBOLOAD,DISP=SHR
//LKED.SYSIN DD *
   MODE AMODE(31),RMODE(ANY)
   NAME BBOACPL3(R)
```

Sequential Terminal

CICS has for some time now supported the use of sequential terminals to input commands and parameters at time of region initialization.

That support can be taken advantage of for WOLA by starting the TRUE and Link Server Task using sequential terminal input from a data set. That provides a way to avoid manual Link Server starts as well as get around the length limitations of `INITPARM=` when used with the `BBOACPL2` PLT program sample.

There's nothing special about using the sequential terminal support of CICS for WOLA ... in other words, the setup of the sequential terminal for the CICS region is no different for WOLA than it would be for any other use.

Once the sequential terminal is properly defined to CICS, then you may use it to enter the BBOC commands to start the Task Related User Exit (TRUE) as well as the Link Server Task.

Note: A common pattern is to use the supplied PLT sample `BBOACPLT` to start the TRUE, and use sequential terminal to start the Link Server Task. That sample PLT is part of the `DFHPLTOL` job discussed under "Customize sample DFHPLTOL member and submit" on page 14.

Do the following:

- ☐ Work with your CICS systems programmer to see if a sequential terminal has already been defined for the region. If yes, then go to the next step. Otherwise, enable sequential terminal support for the region.
- ☐ In the sequential terminal input data set, code the `BBOC START_SRVR` command as you would if entering it from the CICS terminal session.

- ☐ When the region completes initialization, validate the registration into the WAS z/OS server with the `F aaaaaa,DISPLAY,ADAPTER,DAEMONRGES` command, where *aaaaaa* is the JOBNAME of the z/OS server into which the registration was made.

Example of sequential terminal definitions in our test environment

The following examples are offered only as a guide. Your definitions should be tailored to your environment.

DFHTCTSQ

```
TCTSQ    TITLE 'DFHTCTSQ - CICS TERMINAL CONTROL TABLE - CRLP'
*   <comment section removed to save space in this document>
         DFHTCT TYPE=INITIAL,                                          X
               ACCMETH=(NONVTAM,VTAM),                                 X
               SUFFIX=SQ
         COPY  DFH$TCTS             - SEQUENTIAL (CRLP) DEFS.
         DFHTCT TYPE=FINAL
         END   ,
```

DFH$TCTS

```
         TITLE 'DFH$TCTS - COPYBOOK OF TCT ENTRIES FOR SEQUENTIAL (CRLPX
               ) TERMINAL'
*   <comment section removed to save space in this document>
         DFHTCT TYPE=SDSCI,                                            X
               DEVICE=DASD,                                            X
               DSCNAME=CARDIN,                                         X
               MACRF=R

         DFHTCT TYPE=SDSCI,                                            X
               DEVICE=DASD,                                            X
               DSCNAME=PRINTER,                                        X
               MACRF=W
*
         DFHTCT TYPE=LINE,                                             X
               ACCMETH=BSAM,                                           X
               TRMTYPE=DASD,        U/R, CRLP, DASD, TAPE              X
               INAREAL=132,                                            X
               ISADSCN=CARDIN,                                         X
               OSADSCN=PRINTER
*
         DFHTCT TYPE=TERMINAL,                                         X
               TRMIDNT=SAMA,                                           X
               TRMPRTY=255,                                            X
               TRMTYPE=DASD,                                           X
               TRMSTAT=TRANSCEIVE,                                     X
               TIOAL=132,                                              X
               TCTUAL=255
```

Part of the CICS region JCL start procedure

```
//*
//CARDIN   DD DISP=SHR,
//             DSN=USER1.WAS8.WOLA.TCTIN132,
//             LRECL=132,BLKSIZE=132,RECFM=F
//*
//PRINTER  DD SYSOUT=*,DCB=BLKSIZE=121
```

Input data set

```
Organization  . . . : PS
Record format . . . : F
Record length . . . : 132
Block size  . . . . : 132
```

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0----+----1
********************************* Top of Data **************************************************************
CWTO BBOC START_TRUE IS DONE IN PLTPI\
CWTO ISSUING BBOC START_SRVR COMMAND\
BBOC START_SRVR RGN=CICSREG DGN=Z9CELL NDN=Z9NODEA SVN=Z9SR01A SVC=* MNC=1 MXC=10 TXN=N SEC=N REU=Y\
CESF GOODNIGHT\
******************************** Bottom of Data ************************************************************
```

Appendix C: Miscellaneous Information

Requirements to match module library with WAS z/OS level in use

As you have seen, part of using WOLA from an external address space such as CICS or batch requires the WOLA modules be copied out to a `LIBRARY` data set. That data set if then concatenated to the CICS `DFHRPL` or added to a batch job's `STEPLIB`[89].

When that external address space seeks to register into the WAS z/OS server, there's some interaction that takes place between the WOLA modules in that `LIBRARY` data set and the WOLA support that's included with the WAS z/OS server.

Key Point: In an ideal world the level of the modules in the `LIBRARY` data set would exactly match the level of support in the WAS z/OS server. That's one of the reasons why we had you qualify the data set with a WAS z/OS version and fixpack value ... to help you insure that match.

But the requirement is not that strict. Once again, the general answer "It depends" comes up, with our full agreement that "It depends" is never very satisfying.

What follows is a high-level explanation of a few key things to help set a framework of understanding of how the WAS z/OS level and the external module `LIBRARY` relate.

Is there a way to tell what level the modules in the LIBRARY are?

At this time not easily. The ability to do so is a known requirement.

What's the rule ... update modules with every fixpack?

Ideally, yes. Every time a fixpack is applied to your WAS z/OS environment and WOLA is in use, a recopy[90] of the modules out to a `LIBRARY` data set is the best policy. If a new data set name is used, then an update to the `DFHRPL` and `STEPLIB` references.

But that is *not* a hard-and-fast requirement. Just a good practice to reduce potential issues.

What symptom would is seen if there's a mismatch?

If you see RC=12, RSN=88 or 90 then a mismatch should be suspected.

The big change occurred with the 8.0.0.1 fixpack, when a "stubs vector" was added. Trying to use a newer (that is, 8.0.0.1 or later) level of the modules with an older (that is, 8.0.0.0 or V7) level of WAS z/OS will result in those RC/RSN errors.

This is why we started the doc with "Capture the version and fixpack level of WAS installation" on page 6. It was a way to draw your attention to the precise version and fixpack you were using as we next took you into copying the modules out to the `LIBRARY` data set. It is also why we encouraged you to include the fixpack level as part of the data set qualifier.

If in doubt, copy the modules out again[91] and make those modules available to your CICS region or batch job, then retest.

Update of ola.rar file

The `ola.rar` file you installed earlier is what implements the Java Connector Architecture (JCA) interfaces for WOLA. For example, if the Java program is using the `ConnectionSpecImpl` class or the `InteractionSpecImpl` class[92], those class are part of the `ola.rar` file.

The classes in `ola.rar` are sometimes part of maintenance released for WAS z/OS. That means the ola.rar resource adapter needs to be updated in those nodes that use `ola.rar`.

89 Or added to Link List.
90 If V7, then `olaInstall.sh OLAMODS '<data-set>'`. If V8 or V8.5 then `copyZOS.sh OLAMODS '<data-set>'`.
91 If V7, then `olaInstall.sh OLAMODS '<data-set>'`. If V8 or V8.5 then `copyZOS.sh OLAMODS '<data-set>'`.
92 If your Java program using WOLA is going *outbound* -- that is, the call is WAS outbound to CICS or IMS -- then it is using those classes.

The release information for the fixpack will indicate whether the resource adapter needs updating[93].

There are two ways you can accomplish the update of the `ola.rar` resource adapter:

1. By manually uninstalling and re-installing the `ola.rar` resource adapter
2. By using the supplied `olaRarUpdate.py` WSADMIN script

The following table summarizes the pros and cons of each approach:

	Pros	Cons
Manual update	Relatively simple process as illustrated under "Install the ola.rar resource adapter and create a connection factory" starting on page 9.	Deleting the `ola.rar` resource adapter in preparation for re-installing will result in all the Connection Factories defined under the adapter being lost. You will need to recreate them[94].
`olaRarUpdate.py`	Updates the adapter and preserves all the defined Connection Factories.	Involves using WSADMIN, which while not difficult may be unfamiliar to some.

We'll discuss the use of the `olaRarUpdate.py` script here. The manual update process involves deleting the resource adapter, then re-installing using the same instructions provided on page 9.

Location of the `olaRarUpdate.py` script file

The Jython script file is located in the *installation* file system, not the runtime configuration file system. The location is different depending on V7 or V8.x:

Version 7	`/<install mount point>/mso/OLA/bin/olaRarUpdate.py`
Version 8 Version 8.5	`/<install mount point>/util/zos/OLASamples/olaRarUpdate.py`

Invoking the script file to update the RAR file

The syntax required for the script file is relatively simple:

```
-f <path>/olaRarUpdate.py <cell_long_name> <node_long_name>
```

The script will take inventory of the adapter and connection factories and update the RAR file to the level found in the install file system from which the script was run. It preserves the defined connection factories.

Here is an example of the commands to update the RAR in a Network Deployment cell. Notes related to the numbered blocks follow:

Note: The WSADMIN command is shown as three lines here (because the page is not wide enough to show it all on one line), but in actual practice that command is entered as one long command.

```
> cd /wasv7config/z3cell/z3dmnode/DeploymentManager/profiles/default/bin  1
                         2                3                 4
> wsadmin.sh -lang jython -conntype SOAP -host wg31.washington.ibm.com
                                              5             6
                                         -port 30002 -user z3admin -password ******
                                                            7            8      9
              -f /shared/zWebSphere/V7R0/mso/OLA/bin/olaRarUpdate.py z3cell z3nodea
```

93 If you're moving from one *version* to another then it's a different matter. Moving from version to version implies *migration* of the cell environment. The migration utility is designed to make the necessary updates, including the installed resource adapter. An alternative approach is to build a new cell at the newer version parallel to the existing cell, and manually create all the customizations needed. This document won't go into the pros and cons of each. Rather, we'll simply state that the update of the `ola.rar` we're speaking of in this section of the document relates to fixpacks within the same version.

94 If you only have the one or a small few CFs then this is a relatively simple thing to do.

Notes:

1. In a network deployment configuration it's best to run WSADMIN from the DMGR's `/bin` directory and run against the DMGR.
2. The `olaRarUpdate.py` script is written in Jython, so it's necessary to tell WSADMIN the language to expect.
3. This example is showing using SOAP to connect to the running DMGR.
4. The host name of this DMGR. You would specify the host appropriate to your environment.
5. The configured SOAP port for this DMGR. You would specify the port appropriate to your environment.
6. If security is enabled for the cell, then you must pass in a userid and password with Admin authority. This example is showing the use of the Admin ID.
7. The `-f` switch points to the full path and file name of the `olaRarUpdate.py` file
8. The cell *long* name is specified
9. The node *long* name where the RAR file is to be updated is specified

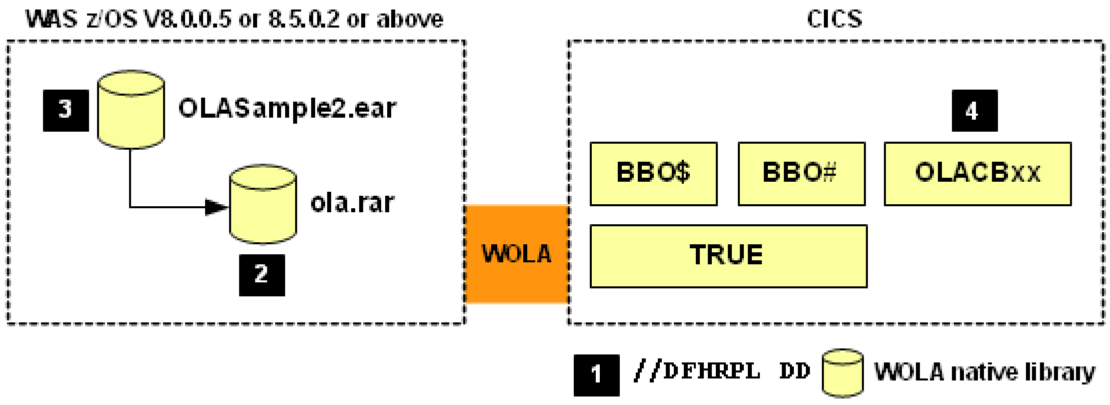
The script will save the configuration changes but not synchronize them to the nodes. You may wish to manually synchronize the node.

This script is run against *each node* that has a RAR installed you wish to update. Here we showed one node being updated. If you had more you would re-run the script for each node.

Improved Channels and Containers using 8.0.0.5 or 8.5.0.2

As mentioned back under "Optional - Use the CICS Channels and Containers sample OLACB02" starting on page 32, the channels and containers support was improved considerably starting with 8.0.0.5 and 8.5.0.2.

To use the new channels and containers support several pieces of the puzzle need to be in place. The following picture illustrates this, with notes following that correspond to the numbered blocks:



Notes:

1. The level of the WOLA modules concatenated to the CICS region's DFHRPL must be at a level that contains the updated channels and containers support. This is what will allow the `BBO$` Link Server Task and `BBO#` invocation task to support the enhanced function. So if you're recently updated your level of WAS z/OS then don't forget to update the WOLA module library as well.
2. The `ola.rar` resource adapter was updated as part of this improved support of channels and containers. To use multiple containers and `MappedRecordImpl()` you'll need the updated RAR. Again, if you've recently updated your level of WAS z/OS then see "Update of ola.rar file" on page 73 for information on updating your copy of the RAR file.

3. The `OLASample2.ear` file supplied with 8.0.0.5/8.5.0.2 was updated to illustrate the use of named channels and multiple containers. If you have an earlier copy of the application in your WAS server you'll need to re-install using the newer copy.
4. The sample target COBOL program you use makes a difference. The `OLACB02` program used was *not* written to recognize a named channel nor multiple containers. However, the samples supplied with 8.0.0.5 / 8.5.0.2 were updated. From the `@@README.jclsamp` file:

```
OLACB10   - JCL/source for CICS sample Cobol program that uses multiple
            containers to pass data to CICS from an EJB.  This is a sample
            target program when using the OLA CICS Link Server.
OLACB11   - JCL/source for CICS sample Cobol program that uses multiple
            containers to pass data to CICS from an EJB.  This is a sample
            target program when using the OLA CICS Link Server.  The data
            is modified inside the target program.
OLACB12   - JCL/source for CICS sample Cobol program that uses multiple
            containers to pass data to CICS from an EJB.  This is a sample
            target program when using the OLA CICS Link Server.  The data
            is deleted inside the target program.
```

If you wish to use the improved channels and container support, then do the following:

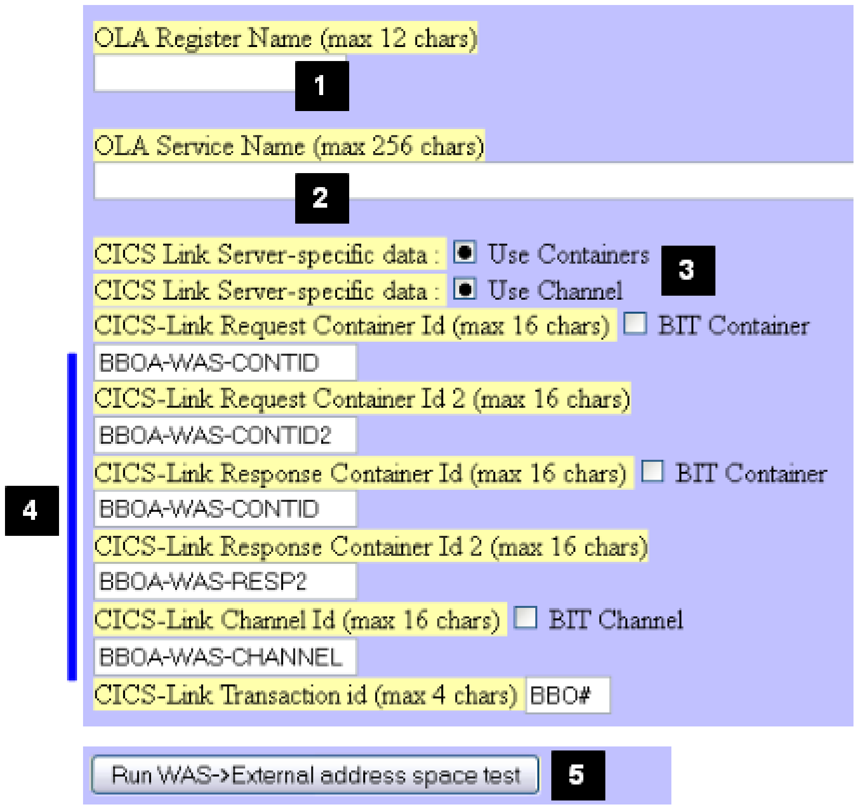
- ☐ Make sure your level of WAS z/OS is at 8.0.0.5 or higher, or 8.5.0.2 or higher.
- ☐ If you're recently updated maintenance to one of those levels[95], then:
  - ○ Copy the newer samples to a FB 80 data set so you can access the new COBOL programs that use the enhanced channels and container support. See "Copy WOLA samples to a sample data set" on page 6.
  - ○ Copy the newer modules to a LIBRARY data set so you may supply the newer modules to your CICS region. See "Copy WOLA modules to a load library data set" on page 7.
  - ○ Update your `ola.rar` file. See "Update of ola.rar file" on page 73.
  - ○ Delete and re-install the `OLASample2.ear` file using the newest copy of the EAR.
- ☐ Customize and compile the `OLACB10`, `OLACB11` and `OLACB12` samples.
- ☐ Modify the `CSDUPDAT` sample job to include references for `OLACB10`, `OLACB11` and `OLACB12` and submit.
- ☐ Start your CICS region, and start the Link Server Task so the region is registered into the WAS z/OS server.
- ☐ From a browser, issue the following URL:

  `http://`***`<host>`***`:`***`<port>`***`/OLA_Sample2_Web/`

  Where ***`<host>`*** and ***`<port>`*** are the what's appropriate for the WAS z/OS server in which the sample application is deployed.

95 As opposed to a fresh new install and cell creation. Then you'll have everything in place from the start and updates won't be needed.

☐ Access the OLASample2 program with the same URL as before. You will notice that application page has fields for the channel name and multiple containers:



OLA Register Name (max 12 chars)

OLA Service Name (max 256 chars)

CICS Link Server-specific data : ● Use Containers

CICS Link Server-specific data : ● Use Channel

CICS-Link Request Container Id (max 16 chars) ☐ BIT Container

BBOA-WAS-CONTID

CICS-Link Request Container Id 2 (max 16 chars)

BBOA-WAS-CONTID2

CICS-Link Response Container Id (max 16 chars) ☐ BIT Container

BBOA-WAS-CONTID

CICS-Link Response Container Id 2 (max 16 chars)

BBOA-WAS-RESP2

CICS-Link Channel Id (max 16 chars) ☐ BIT Channel

BBOA-WAS-CHANNEL

CICS-Link Transaction id (max 4 chars) BBO#

Do the following:

1. Provide a "Register Name" to match what you used when starting the Link Server Task into WAS z/OS.
2. Provide a service name of **`OLACB10`**
3. Check both "Use Containers" and "Use Channel"
4. Leave the container ID and channel name default. You'll change those in a bit.
5. Click the "Run" button

☐ In response, you should see the following in your browser screen:

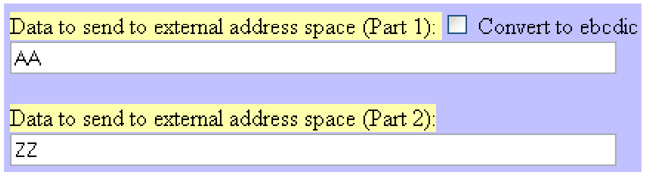
Received output for container: BBOA-WAS-CONTID2 with Container Data: Test data to external a/s part 2
Received output for container: BBOA-WAS-CONTID with Container Data: Test data to external a/s part 1
Test Executed

☐ In the CICS region `CEEMSG` output you should see:

```
OLACB10 entered with channel BBOA-WAS-CHANNEL
Data Length 00000032
Data in BBOA-WAS-CONTID2:
Test data to external a/s part 2
Data Length 00000032
Data in BBOA-WAS-CONTID :
Test data to external a/s part 1
```

Pause and think about what took place. On the application input panel a channel was named (`BBOA-WAS-CHANNEL`). This was not fixed like it was prior to 8.0.0.5 or 8.5.0.2. This was a channel name passed in by the application. Two containers were used, and their names were passed in as well. The data passed in was echoed back.

- ☐ Go back to the application input panel and change the input data to something short and trivial, such as:

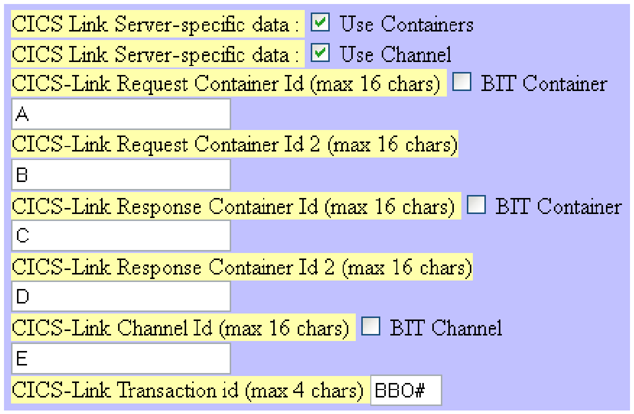

Data to send to external address space (Part 1): ☐ Convert to ebcdic

AA

Data to send to external address space (Part 2):

ZZ

Then change the container and channel names to different values, such as the letters **A** through **E** as shown here[96]:


CICS Link Server-specific data : ☑ Use Containers

CICS Link Server-specific data : ☑ Use Channel

CICS-Link Request Container Id (max 16 chars) ☐ BIT Container

A

CICS-Link Request Container Id 2 (max 16 chars)

B

CICS-Link Response Container Id (max 16 chars) ☐ BIT Container

C

CICS-Link Response Container Id 2 (max 16 chars)

D

CICS-Link Channel Id (max 16 chars) ☐ BIT Channel

E

CICS-Link Transaction id (max 4 chars) BBO#

Then click the button to send. You should see the following on the browser:

```
Received output for container: A with Container Data: AA
Received output for container: B with Container Data: ZZ
Test Executed
```

And the following in `CEEMSG`:

```
Test data to external a/s part 1
OLACB10 entered with channel E
Data Length 00000002
Data in B                :
ZZ
Data Length 00000002
Data in A                :
AA
```

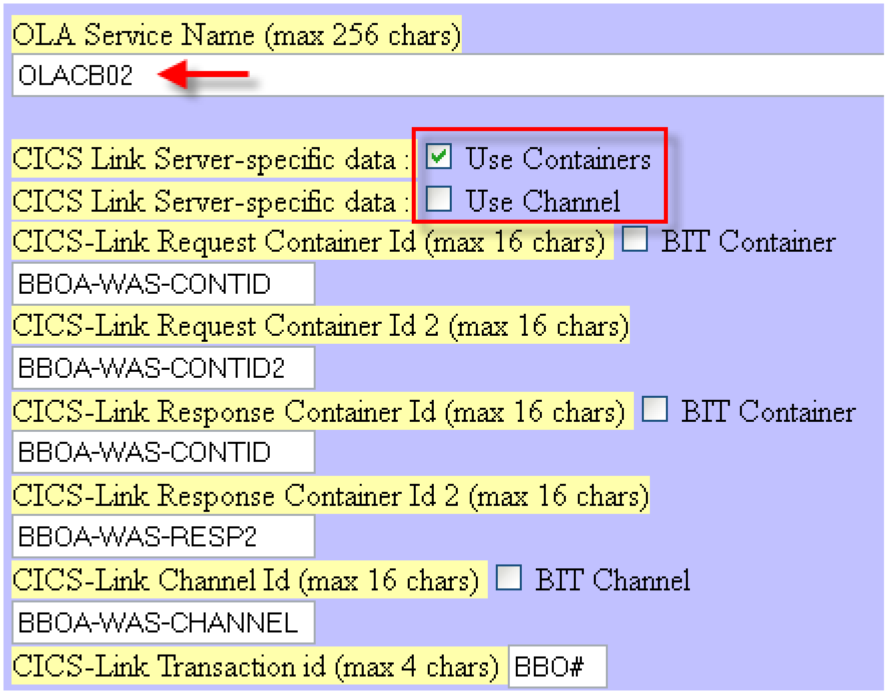
- ☐ Read the comment section of `OLACB10`, `OLACB11` and `OLACB12` to see what each does. Browse the COBOL and see how the processing works. Experiment with samples `OLACB11` and `OLACB12` and see what results you get.

8.0.0.5 / 8.5.0.2 (or higher) and OLACB02 sample

The newer level of channels and container support will still work with the OLACB02 sample. The OLACB02 sample isn't as useful as OLACB10, 11 and 12 because OLACB02 assumes a single container, and it doesn't do much with the data sent in. Still, it *is* possible to use it with the newer support.

96 There's nothing special about those values ... we're just trying to provide names that clearly aren't default values to show the interaction between the Java program and the CICS program is passing your values back and forth.

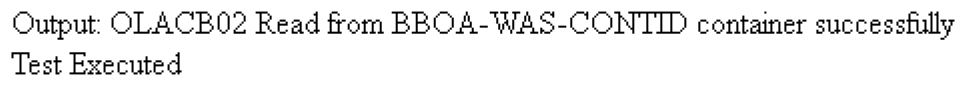
The key is to *not* select "Use Channel" checkbox, and to select "Use Containers." That emulates the pre-8.0.0.5 / 8.5.0.2 behavior:



OLA Service Name (max 256 chars)
OLACB02

CICS Link Server-specific data : [✓] Use Containers
CICS Link Server-specific data : ☐ Use Channel
CICS-Link Request Container Id (max 16 chars) ☐ BIT Container
BBOA-WAS-CONTID
CICS-Link Request Container Id 2 (max 16 chars)
BBOA-WAS-CONTID2
CICS-Link Response Container Id (max 16 chars) ☐ BIT Container
BBOA-WAS-CONTID
CICS-Link Response Container Id 2 (max 16 chars)
BBOA-WAS-RESP2
CICS-Link Channel Id (max 16 chars) ☐ BIT Channel
BBOA-WAS-CHANNEL
CICS-Link Transaction id (max 4 chars) BBO#

The request container ID name matches what's hard-coded in the OLACB02 sample program and with that the result you get back is as expected:



```
Output: OLACB02 Read from BBOA-WAS-CONTID container successfully
Test Executed
```

While that works, the OLACB10 - 12 samples are *much* better at showing the effective use of the enhanced channels and containers function provided with 8.0.0.5 / 8.5.0.2 or higher.

Document Change History

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

March 6, 2013	Original document
March 26, 2013	Miscellaneous updates: check for proper synchronization after setting cell-level variables; additional detail for sequential terminal; use of `SEC=Y` and assertion of thread identity from WAS into CICS.
April 17, 2013	Added section on use of `OLACB02` sample for CICS channels/containers; added section on use of `OLACB10`, `11` and `12` samples for enhanced channels and containers support; added section on using `olaRarUpdate.py` to update WOLA RAR file when maintenance calls for update.
August 13, 2013	Added information on using the "Development Mode" function using the Proxy EJB. This allows Java developers on their distributed WAS workstations to test their Java code against a local copy of the `ola.rar` JCA resource adapter and have the request sent to the mainframe over a network connection. The Proxy EJB "catches" the request and then drives "real WOLA" into CICS.
August 16, 2013	Updated the "Development Mode" section on Client Certificate Authentication to show the value "Supported" set in the WAS z/OS CSIv2 security panel rather than "Required." A value of "Required" resulted in "Security Mismatch" messages when using "real WOLA" from CICS into WAS z/OS. "Supported" allowed the development mode proxy to work as well as "real" WOLA with no security mismatch messages.

End of Document