IBM Advanced Technical Skills

WBSR85

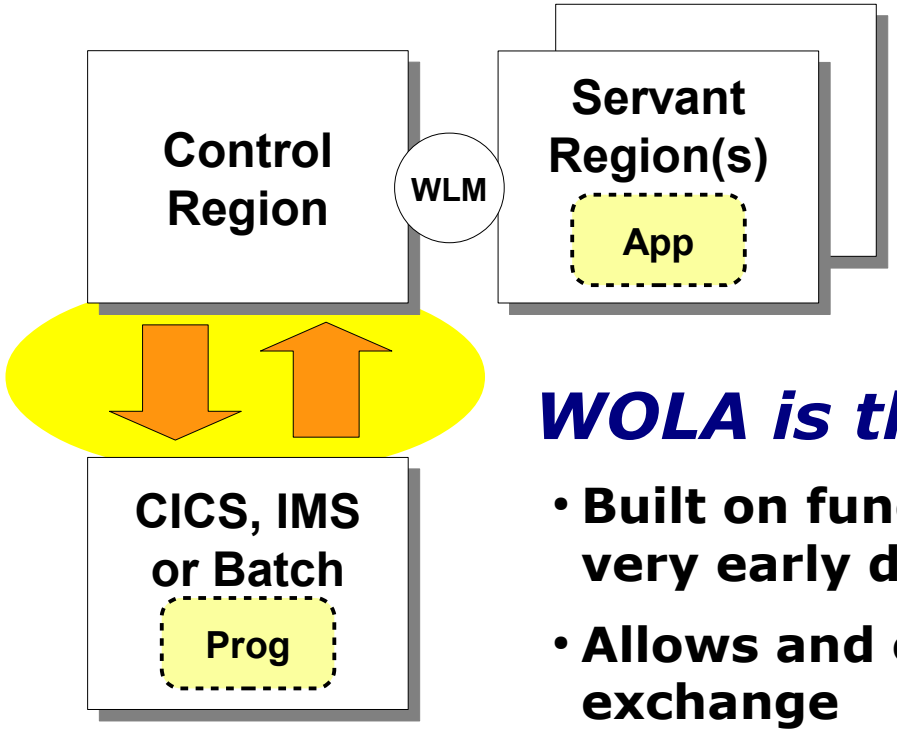
WebSphere Application Server z/OS V8.5

Unit 6 - WOLA

This page intentionally left blank

Overview of WebSphere Optimized Local Adapters

WOLA is a means of communicating between WAS z/OS and external address spaces by transferring message blocks between virtual memory locations:



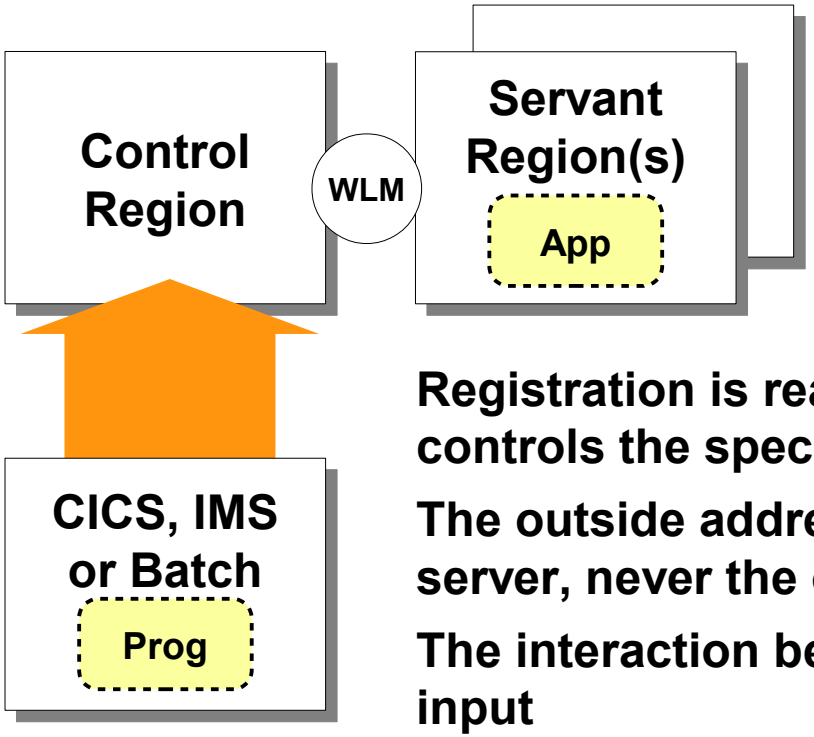
WOLA is this piece …

- **Built on function WAS z/OS has had since the very early days**
- **Allows and coordinates this cross-memory exchange**
- **Provides the higher-level interface to the lower-level exchange**
- **Provides the infrastructure code for use with CICS and IMS**

Registration …

Registration

An important key concept is "registration" ... the construction of the cross-memory linkage into the WAS z/OS application server:



Serves as the cross-memory "pipe" over which exchanges occur

Registration is really a set of control blocks that permits and controls the specific cross-memory exchanges

The outside address space always registers into the WAS z/OS server, never the other way around

The interaction between CR and SR is the same as for any form of input

Any given WAS z/OS server may have multiple registrations into it

Registration is accomplished in several ways:

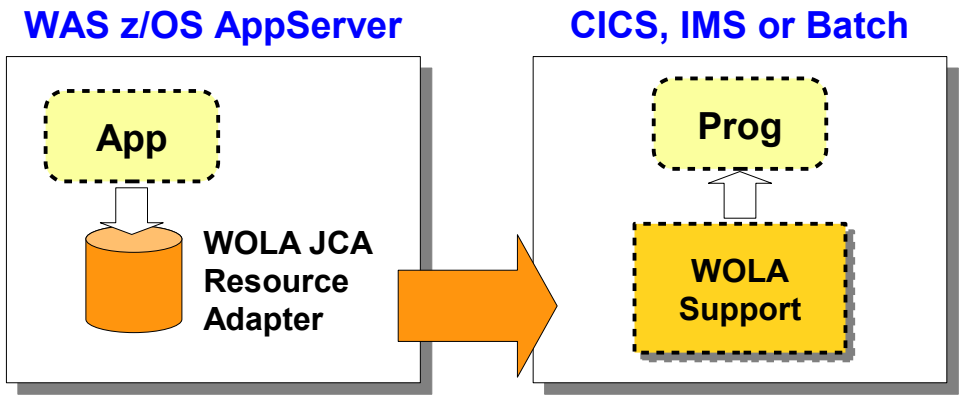
- **A supplied CICS control transaction**
- **The BBOA1REG API**

Outbound vs. Inbound …

"Outbound" and "Inbound"

WOLA is bi-directional. The key to "outbound" vs. "inbound" is thinking about who initiates the conversation ... or, what program invokes the other program.

Outbound

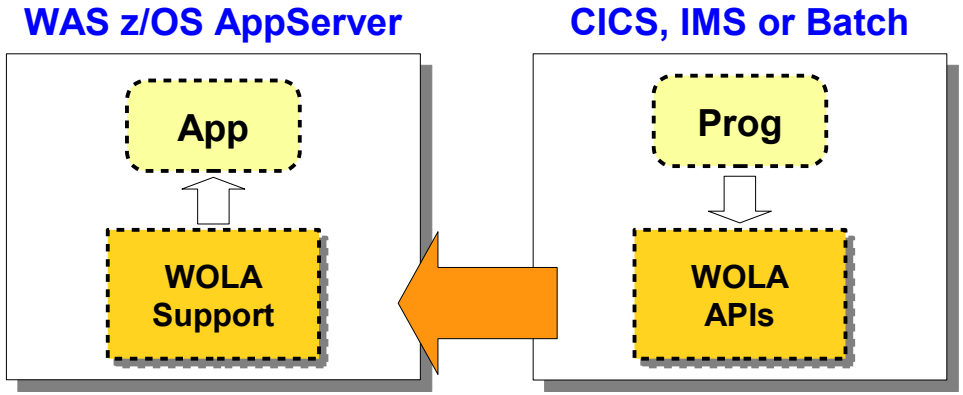


Java program invokes "outbound"

Uses supplied JCA resource adapter

Implementation in external A/S depends on system - CICS, IMS or Batch

Inbound



COBOL, C/C++, Assembler or PL/I

Uses WOLA APIs

Invokes "inbound" to WAS EJB

To target EJB it looks like IIOP

Source of Information on WOLA

In addition to the InfoCenter, which has many valuable reference articles, the WP101490 Techdoc is ATS's central location for WOLA-related documentation

---

`http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp`

InfoCenter `cdat_ola`

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

TechDocs `WP101490`

- Quick Start Guide
- Introduction to WOLA
- History of Updates to WOLA
- Native API Primer
- YouTube Video URL PDF

WOLA is a functionally rich feature of WAS z/OS

In this Unit we'll cover the essential framework

In the hands-on lab you'll use WOLA with CICS and Batch

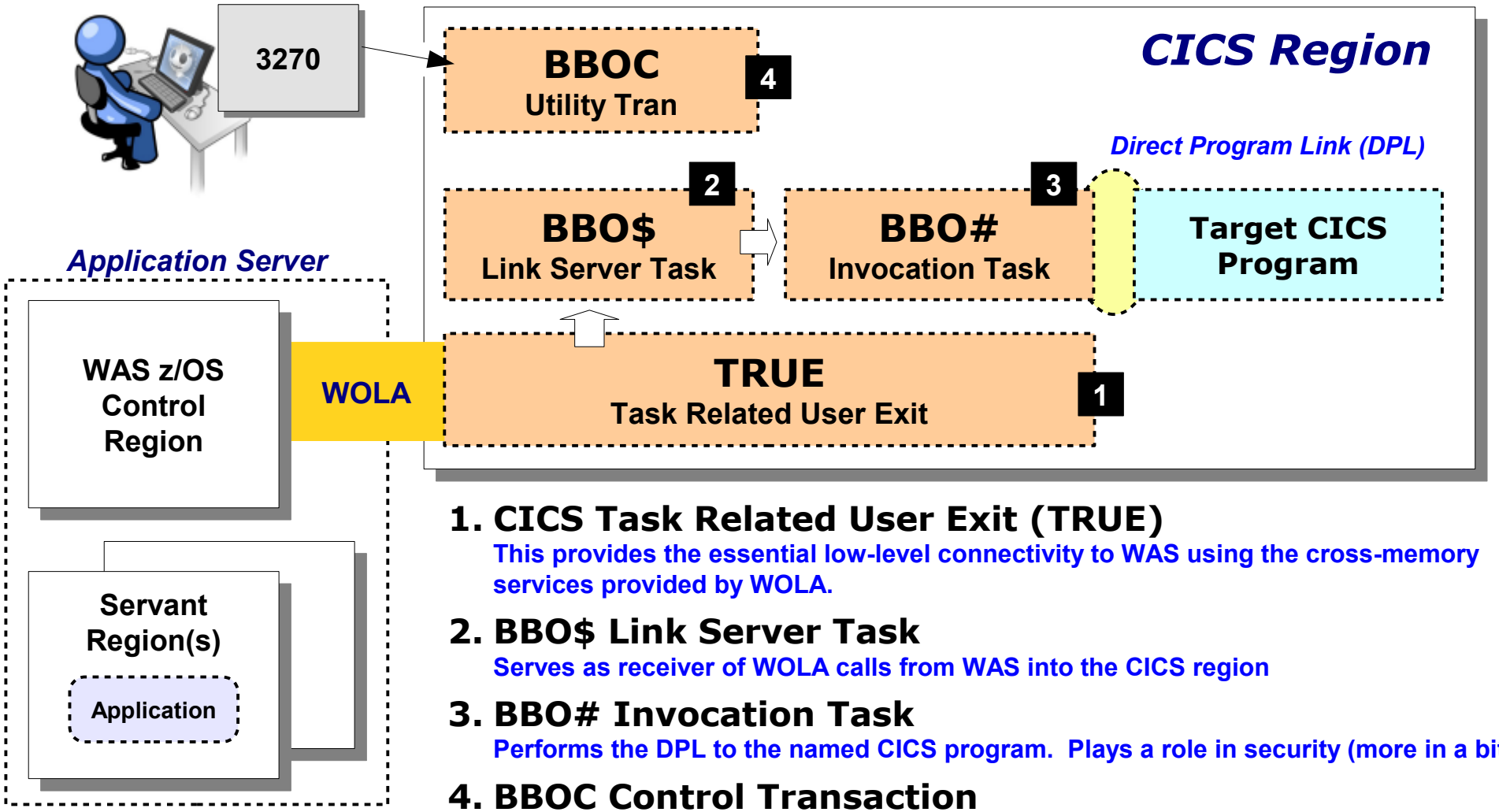
Outbound to CICS …

Outbound to CICS

Using the Supplied CICS Link Server Task

The WOLA Infrastructure Components for CICS

WAS z/OS supplies a few key components that install into a CICS region so it may use WOLA to communicate with WAS z/OS:



1. **CICS Task Related User Exit (TRUE)**
   This provides the essential low-level connectivity to WAS using the cross-memory services provided by WOLA.
2. **BBO$ Link Server Task**
   Serves as receiver of WOLA calls from WAS into the CICS region
3. **BBO# Invocation Task**
   Performs the DPL to the named CICS program. Plays a role in security (more in a bit)
4. **BBOC Control Transaction**
   A 3270 application useful for things such as starting the link server task

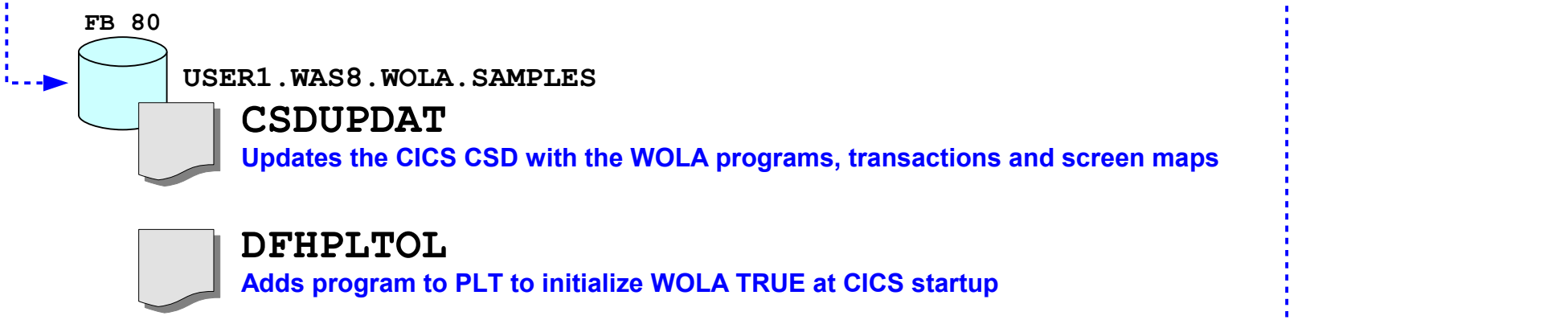
Enabling in CICS …

Enabling WOLA in CICS Region

The following diagram summarizes the steps. The InfoCenter article has details:

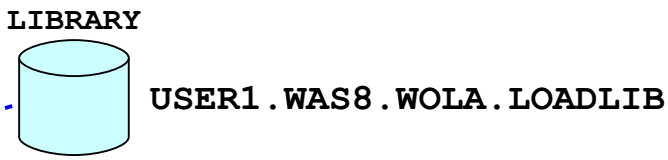
```
/wasv8config/z9cell/z9nodea/AppServer/profiles/default/bin/copyZOS.sh
```

```
copyZOS.sh OLASAMPS 'USER1.WAS8.WOLA.SAMPLES'
copyZOS.sh OLAMODS  'USER1.WAS8.WOLA.LOADLIB'
```



CICS start procedure

```
//DFHRPL   DD DSN=&CICSDS..SDFHLOAD,DISP=SHR
//         DD DSN=SYSS.CICS.LOADLIB,DISP=SHR
                :
//         DD DSN=USER1.WAS8.WOLA.LOADLIB
```



InfoCenter `tdat_enableconnectorcics`

Enable WOLA in WAS …

Enabling WOLA in WAS z/OS

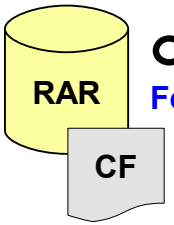
Just a few relatively easy steps to begin using WOLA from an application server:

Two scope=cell environment variables

```
WAS_DAEMON_ONLY_enable_adapter = 1
ola_cicsuser_identity_propagate = 1
```

InfoCenter `cdat_olacustprop`

Will require a restart of the entire WAS cell to pick up these changes



`ola.rar`

Found in the `/installableApps` directory

Simple connection factory ... no native library path, no custom properties to start with

The installation of this RAR file is like any JCA RAR file

WAS z/OS SAF Profile

```
CB.BIND.Z9*
```

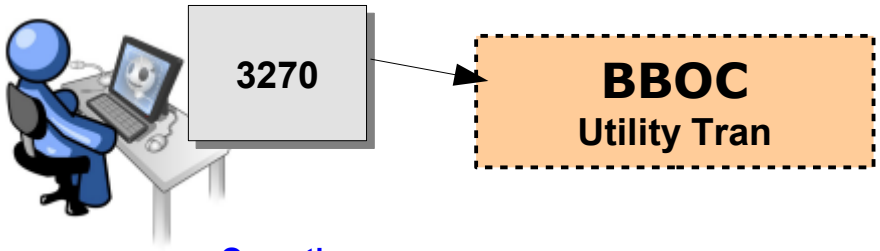
- **Grant CICS ID READ, or**
- **Make profile UACC READ**

InfoCenter `tdat_enableconnector`

Starting Link Server Task …

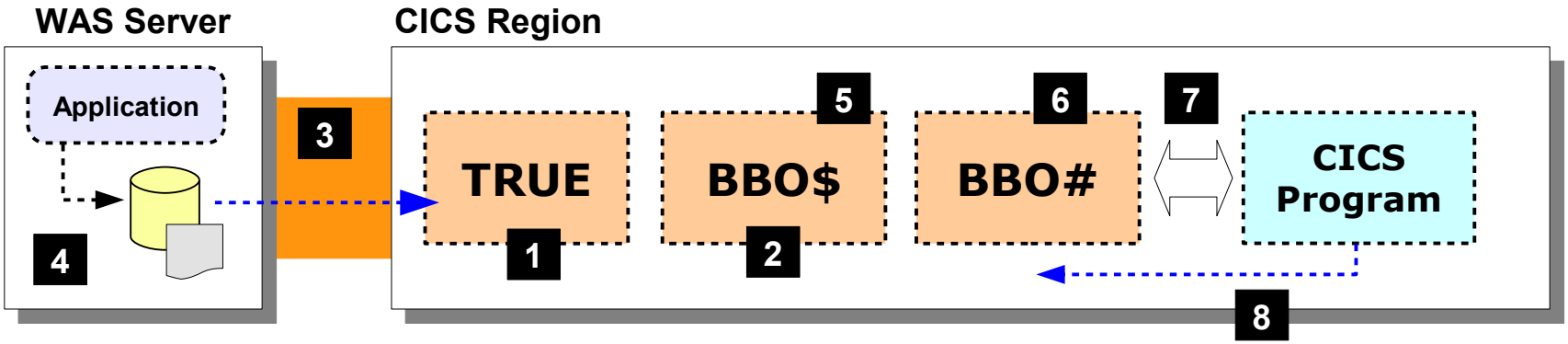
Starting the WOLA Link Server Task in CICS

This performs two roles -- it initates the registration into the WAS server, and it prepares the Link Server to accept requests from the application in WAS:



Operation	Register Name	Cell SHORT	Node SHORT	Server SHORT
BBOC	START_SRVR	RGN=CICSXREG	DGN=Z9CELL	NDN=Z9NODEA	SVN=Z9SR01

SVC=*	MNC=1	MXC=10	TXN=N	SEC=N	REU=Y
Accept any service name	Minimum Connections	Maximum Connections	Propagate Transaction?	Propagate Security?	Reuse BBO#?



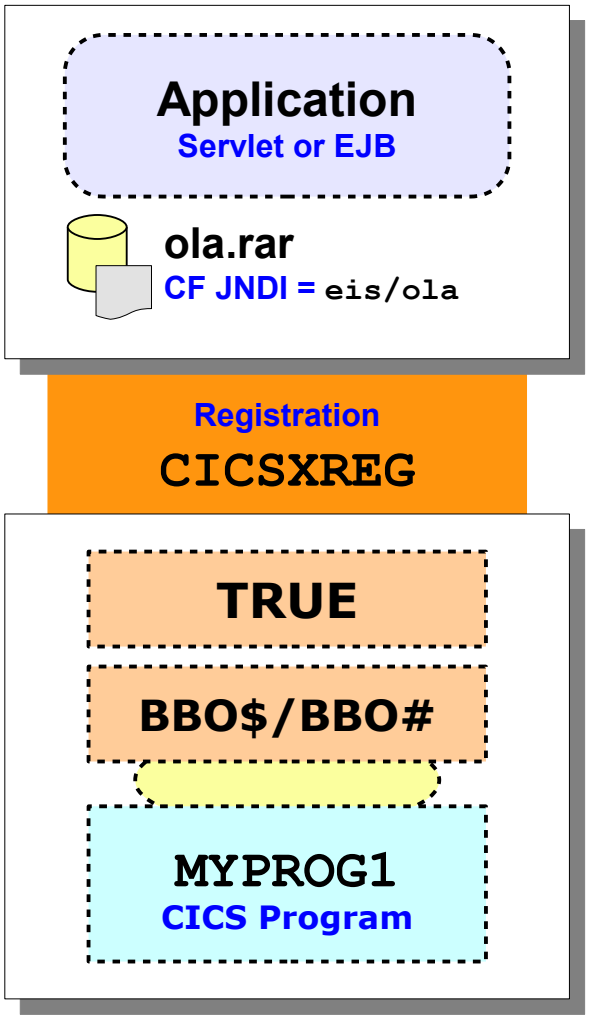
See notes for explanation of numbered blocks

InfoCenter `rdat_cics`

Java application considerations …

Java Application Considerations

For outbound use of WOLA to CICS using the Link Server Task the following considerations come into play:




```
Context ctx = new InitialContext();
ConnectionFactory cf
                1 = ctx.lookup("java:comp/env/eis/ola");
ConnectionSpecImpl csi = new ConnectionSpecImpl();
csi.setRegisterName ("CICSXREG"); 2
Connection con = cf.getConnection(csi); 3
```

```
Interaction int = con.createInteraction();
InteractionSpecImpl isi = new InteractionSpecImpl();
isi.setServiceName("MYPROG1"); 4
int.execute(isi, data);
```

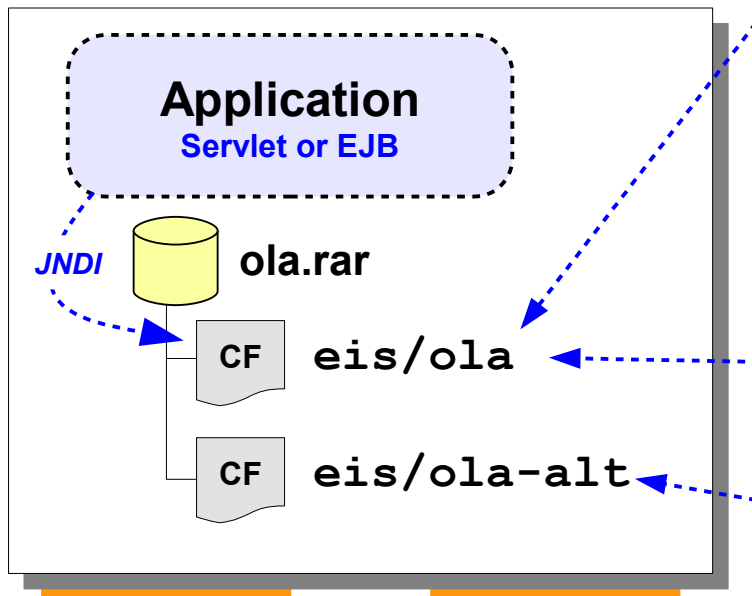
Either COMMAREA or CICS channel and container. If channel and container, see InfoCenter `rdat_cics`

InfoCenter `tdat_connect2wasapp, tdat_useoutboundconnection`

RA failover …

Using Resource Failover with WOLA Outbound to CICS

In many ways this is just like what we saw with resource failover earlier. But there's a few important things to note about making this work properly:



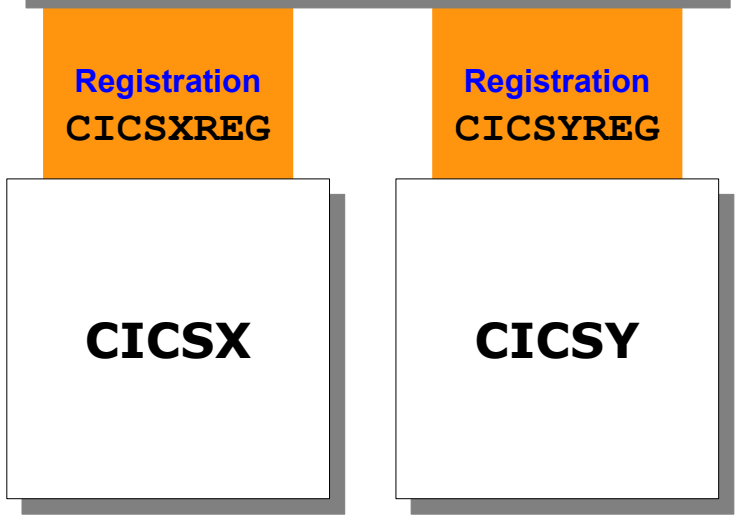
Connection Pool Custom Properties

```
alternateResourceJNDIName = eis/ola-alt
failureThreshold = n attempts
resourceAvailabilityTestRetryInterval = n secs
failureNotificationActionCode = 1, 2 or 3
```

RegisterName	CICSXREG

RegisterName	CICSYREG

Set the RegisterName as a custom property of the CF, not in the application program as we saw earlier



Failover custom properties same as we saw for JDBC and JCA resource connections

Same properties, settings and behavior

The registration into the WAS server must exist ahead of time

Registration is always performed from external space into WAS. For CICS and WOLA, start Link Server in each CICS region ahead of usage

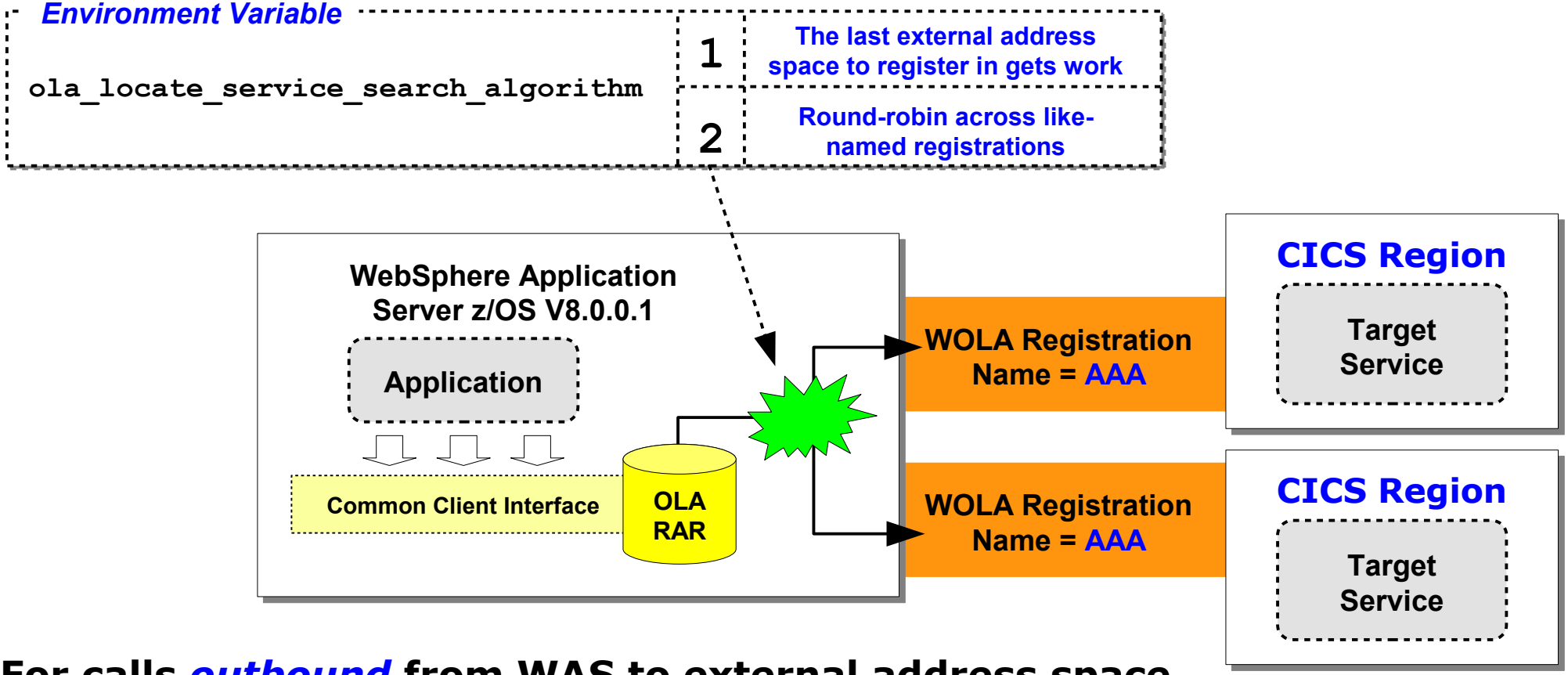
Set RegisterName custom property on each CF to name the registration to communicate over

Permits different registration names to be used transparent to application

Round-robin …

V8.0.0.1 and WOLA Round-Robin

The 8.0.0.1 fixpack brought new WOLA function, including ability to round-robin between multiple CICS regions registered into the server with the same name:



For calls *outbound* from WAS to external address space

Registration names *must be identical*

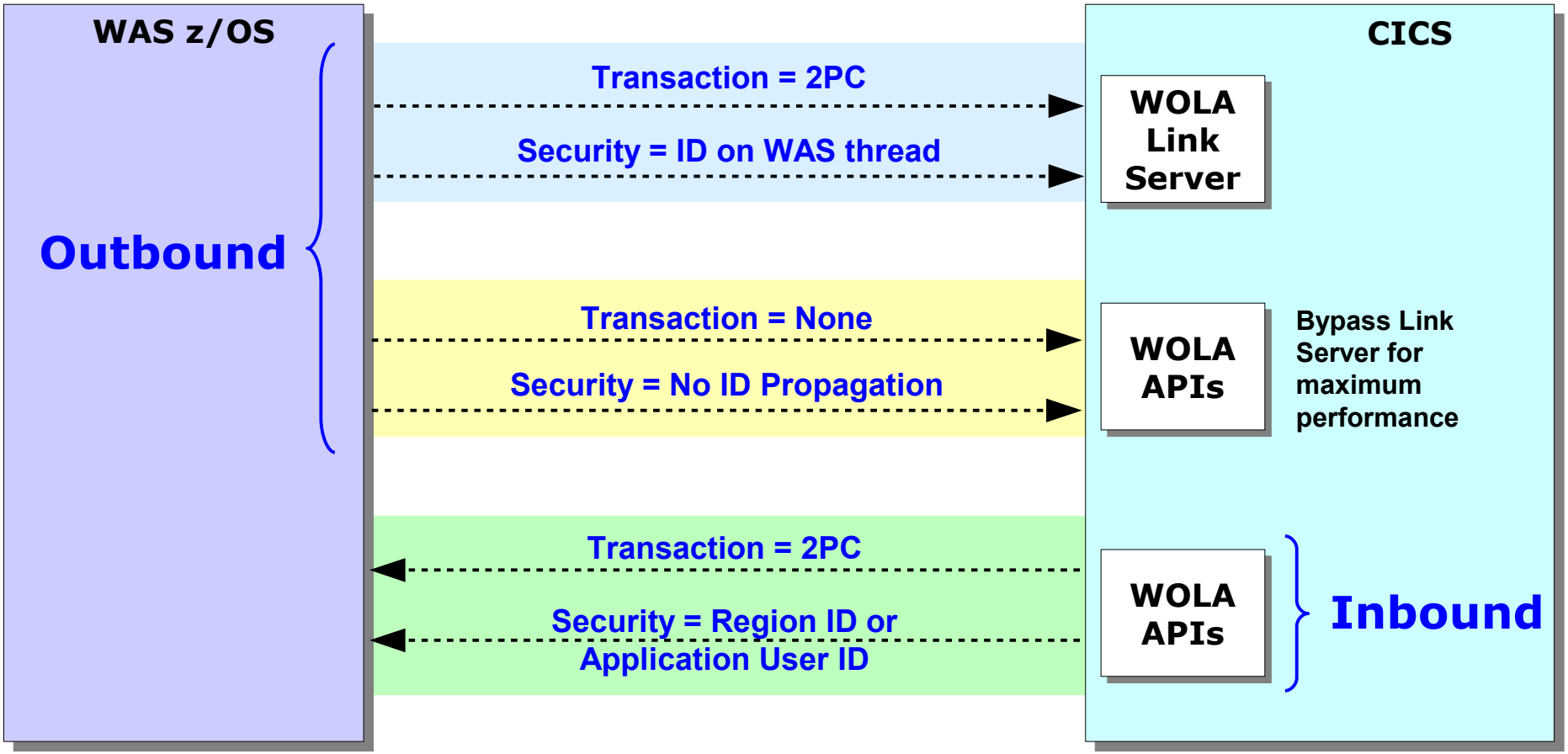
Targeted service must be present in address spaces participating in the work distribution

Any supported external address space, not just CICS

TX, Security summary …

Summary of Transaction and Security Support

The following picture summarizes the support for TX and security:

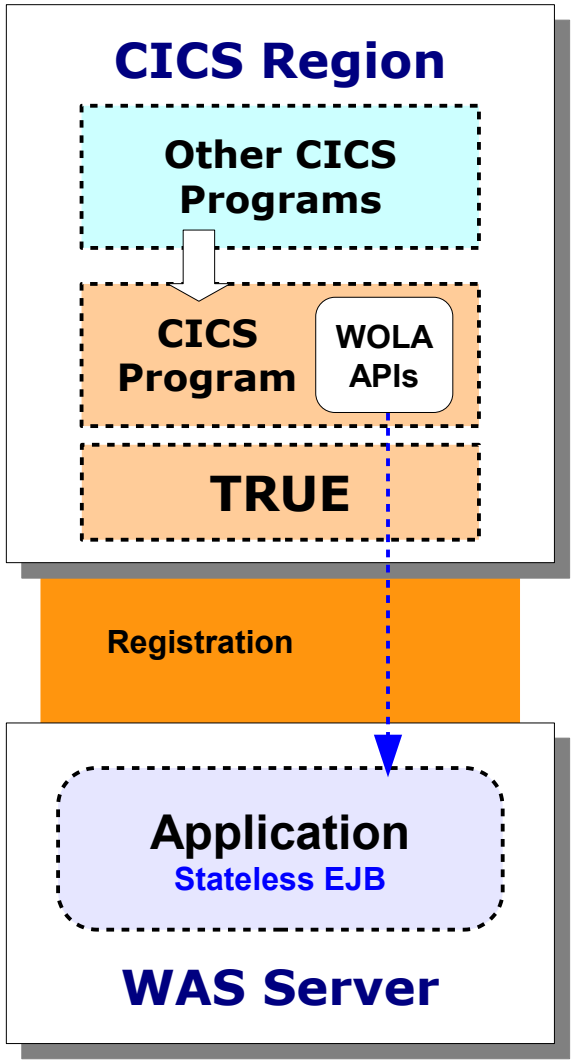


The registration into WAS must have the appropriate TXN and SEC settings to support propagation of global transaction and propagation of security identity

Inbound? …

Inbound to WAS from CICS?

It is possible to have a program in CICS invoke a Java service in WAS z/OS using WOLA. It implies the use of the WOLA native APIs:



The TRUE is still needed

Always needed in CICS because it provides the fundamental WOLA function

Link Server Task not used

Link Server task is for outbound WAS-to-CICS, not inbound to WAS

Registration into WAS server must be present

Accomplish with BBOC REGISTER or BBOA1REG native API

CICS program must use WOLA APIs

Note the concept of a "bridge" program that shields other CICS programs from having to understand the APIs. We'll explore those APIs next

The ola.rar adapter not used

That's for outbound calls ... general WOLA support used for inbound calls

Target must be stateless EJB

And it must implement using the supplied WOLA class files

This is just like what an external batch program would use. We'll explore inbound from batch next ... keep in mind same lessons apply to inbound from CICS

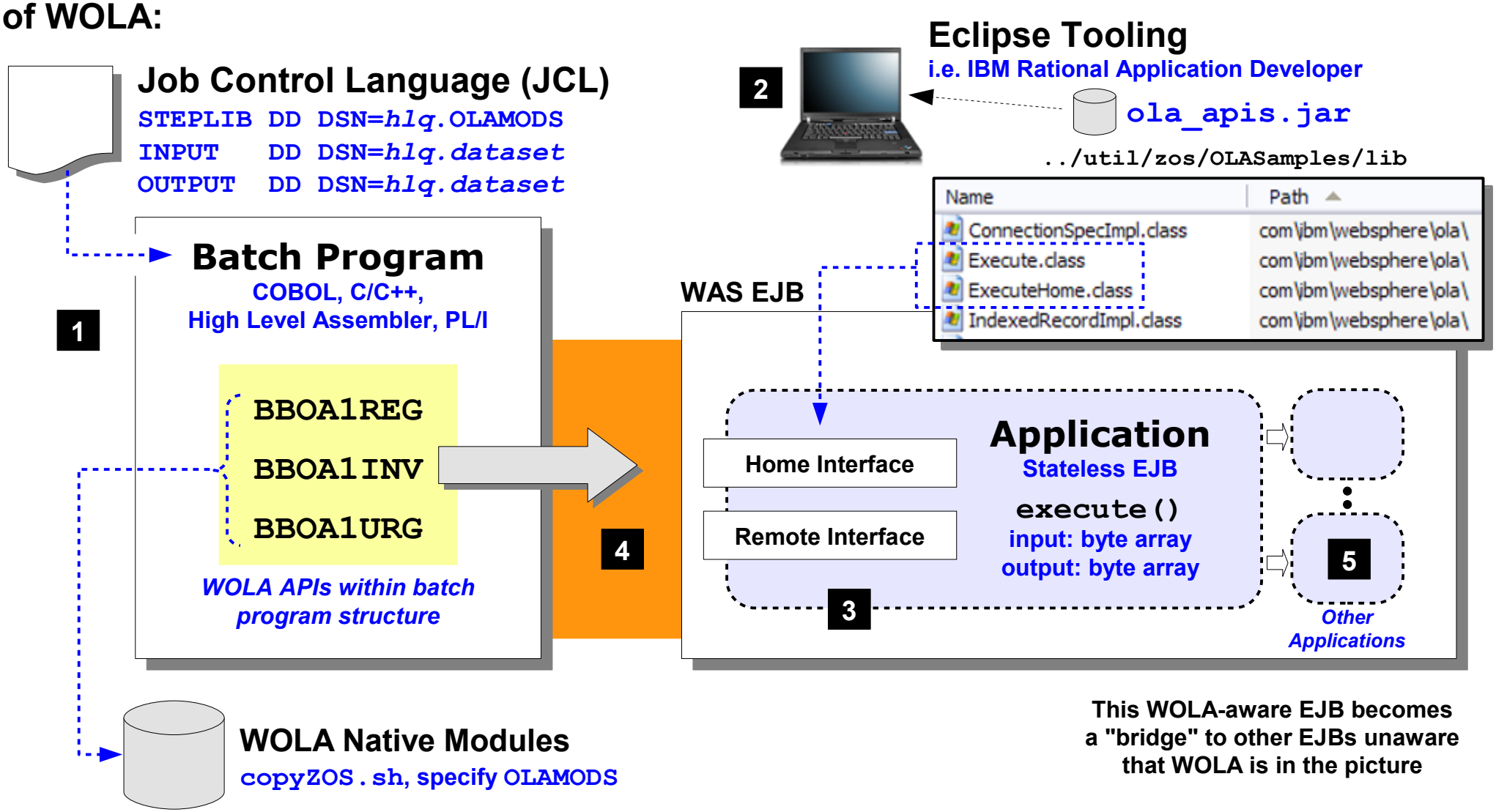
Batch ...

Inbound from Batch

Using the native APIs of WOLA

Essentials of Batch Program Use of WOLA

Relatively simple setup, but there is a bit more exposure to the programming interfaces of WOLA:



This WOLA-aware EJB becomes a "bridge" to other EJBs unaware that WOLA is in the picture

InfoCenter `cdat_olaapis, tdat_useola_in_step2`

API InfoCenter …

The WOLA Native APIs InfoCenter Article

An incredibly useful InfoCenter article that details all 13 of the native APIs, including parameters and return code / reason code descriptions

13 APIs plus an internal link to JCA adapter APIs

- Register - BBOA1REG/BBGA1REG
- Unregister - BBOA1URG/BBGA1URG
- Connection Get - BBOA1CNG/BBGA1CNG
- Connection Release - BBOA1CNR/BBGA1CNR
- Send Request - BBOA1SRQ/BBGA1SRQ
- Send Response - BBOA1SRP/BBGA1SRP
- Send Response Exception - BBOA1SRX/BBGA1
- Receive Request Any - BBOA1RCA/BBGA1RCA
- Receive Request Specific - BBOA1RCS/BBGA1
- Receive Response Length - BBOA1RCL/BBGA1
- Get Message Data - BBOA1GET/BBGA1GET
- Invoke - BBOA1INV/BBGA1INV
- Host Service - BBOA1SRV/BBGA1SRV
- JCA Adapter APIs

APIs that start with BBO* are 31-bit callable; BBG* are 64-bit callable

Parameter map (with full descriptions following)

API	Syntax

BBOA1INV or BBGA1INV	`BBOA1INV ( registername, requesttype, requestservicename, requestservicenamel, requestdata, requestdatalen, responsedata, responsedatalen, waittime, rc, rsn, rv )`
	
	`BBGA1INV ( registername, requesttype, requestservicename, requestservicenamel, requestdata, requestdatalen, responsedata, responsedatalen, waittime, rc, rsn, rv )`

Return Code / Reason Code descriptions for each API

Return Code	Reason Code	Description	Action
0	-	Success	
4	-	Warning - see reason code	
8	-	Error - see reason code	
	8	Register name token already exists.	Ensure that the register name passed is valid.
	10	The connection is unavailable. The wait time expired before the connection request is obtained.	The application behavior varies. Wait and retry, or accept this failed Invoke API call. Another option is to increase the maximum connections setting on the Register API call.

A wonderful reference article, but it doesn't highlight how easy using the APIs can be …

InfoCenter `cdat_olaapis`

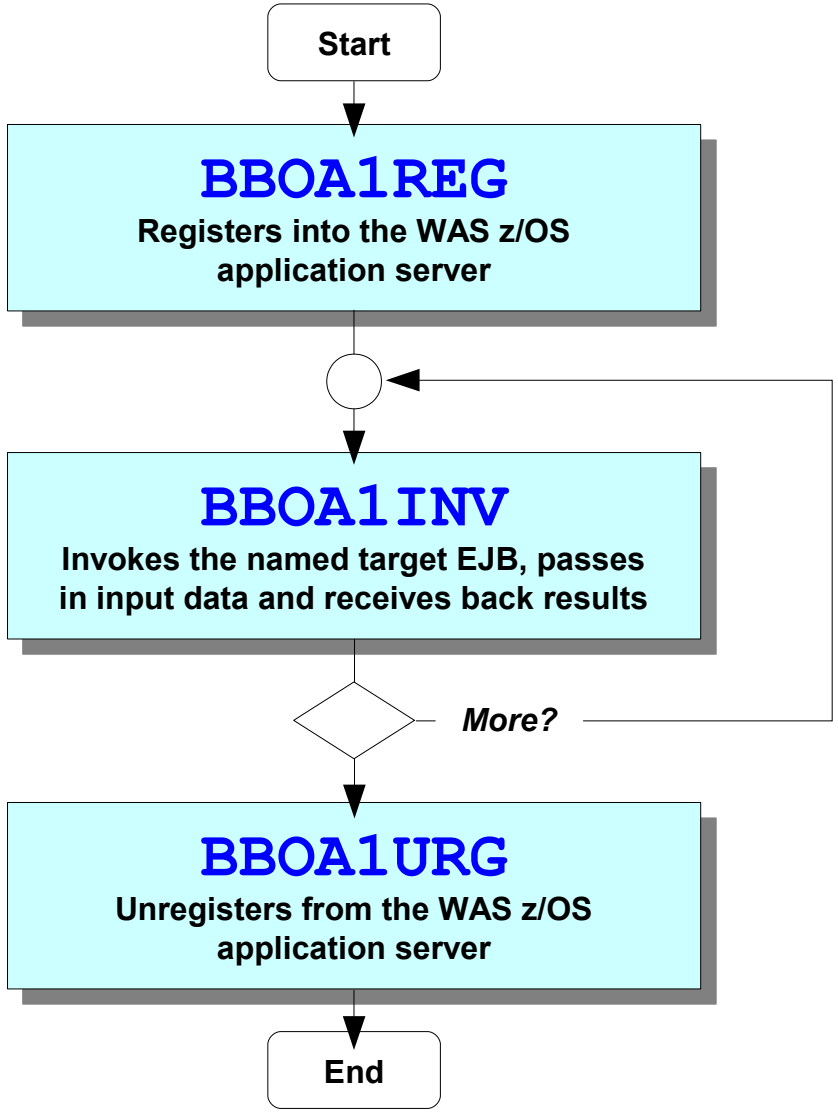
Simplest use …

The Simplest Inbound Use of Native APIs

There are 13 APIs, but that doesn't mean you have to use all 13 ...

13 APIs as listed in the InfoCenter article

- **BBOA1REG**
- **BBOA1URG**
- BBOA1CNG
- BBOA1CNR
- BBOA1SRQ
- BBOA1SRP
- BBOA1SRX
- BBOA1RCA
- BBOA1RCS
- BBOA1RCL
- BBOA1GET
- **BBOA1INV**
- BBOA1SRV



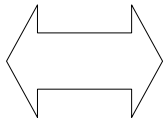
What are other APIs used for?

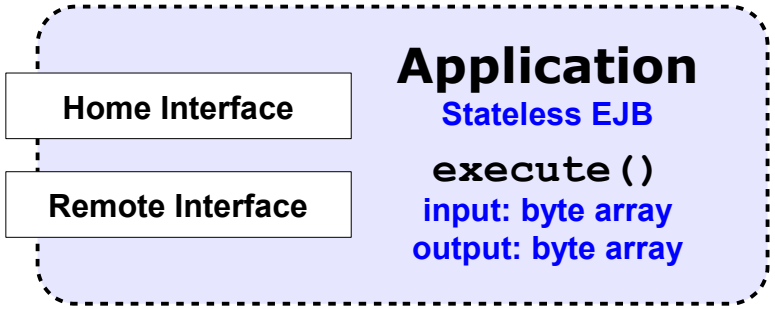
Assumptions ...

BBOA1INV Makes Some Assumptions

To keep the BBOA1INV API simple to understand and simple to use, it makes some assumptions. Explaining this will begin to surface why the other APIs exist ...

BBOA1INV
Invokes the named target EJB, passes in input data and receives back results





Assumptions Made ...

- **Program control held until WAS reponds**
  In other words, it operates *synchronously* ... invoke, wait for response, process response
- **Connections returned to pool each time**
  Which implies a little bit of extra overhead to get the connection each time
- **The maximum response length is predictable**
  You set the maximum response length as an input parameter on the API
  If response back is unpredictable it means you'll need more granular control

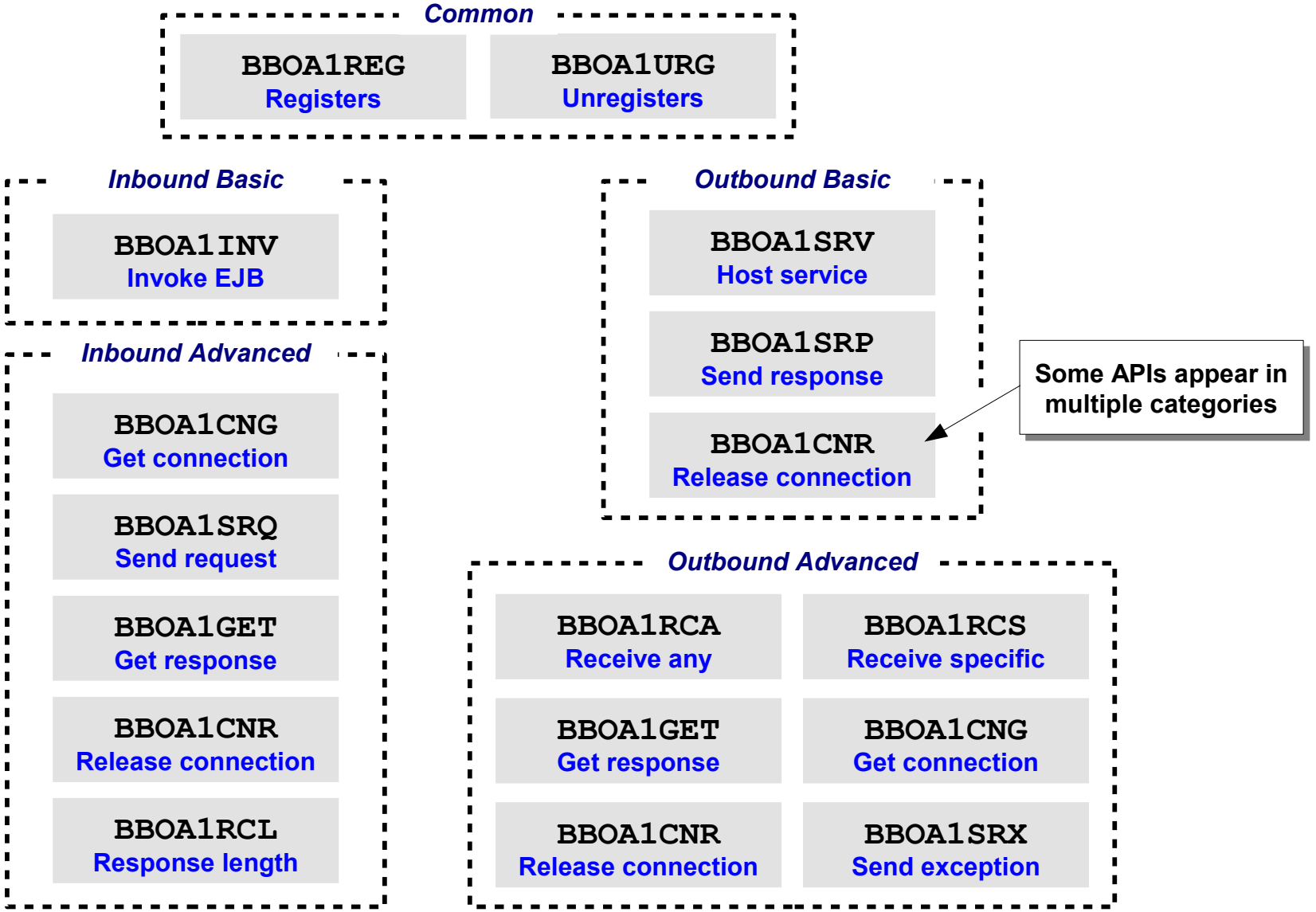
This suggests WOLA provides "basic" APIs and "advanced" APIs

APIs categorized ...

13 APIs Categorized

The organize around inbound, outbound, basic and advanced:

BBOA1REG
BBOA1URG
BBOA1CNG
BBOA1CNR
BBOA1SRQ
BBOA1SRP
BBOA1SRX
BBOA1RCA
BBOA1RCS
BBOA1RCL
BBOA1GET
BBOA1INV
BBOA1SRV

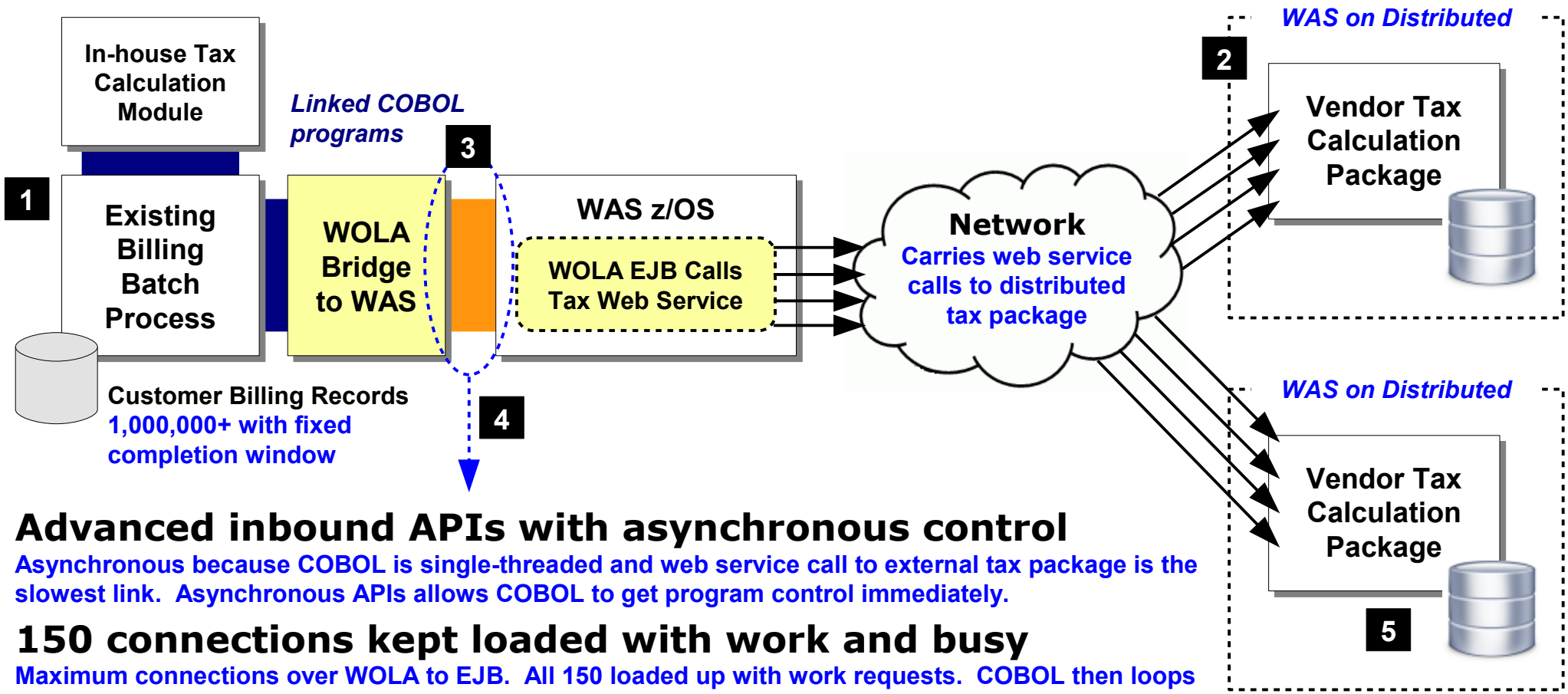


InfoCenter `cdat_olaapis`

Real example …

A Real-Life Example of Inbound Batch Processing

This involves a COBOL batch program that invokes a vendor tax calculation application running on distributed WAS and accessed with web services:



Advanced inbound APIs with asynchronous control

Asynchronous because COBOL is single-threaded and web service call to external tax package is the slowest link. Asynchronous APIs allows COBOL to get program control immediately.

150 connections kept loaded with work and busy

Maximum connections over WOLA to EJB. All 150 loaded up with work requests. COBOL then loops through array to see if response received. If so, then process back results and load that connection with another request. Connections kept fully busy in this manner.

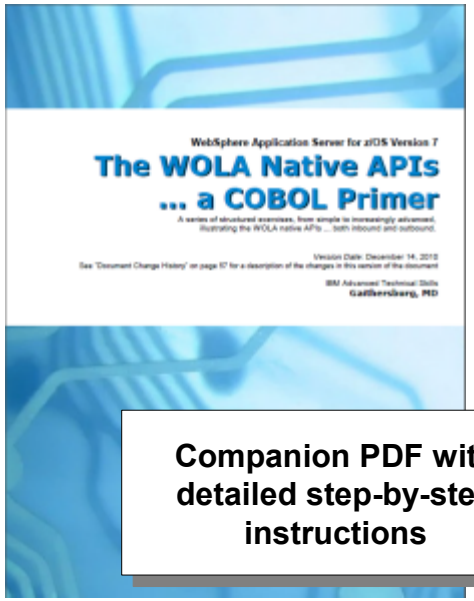
Multi-threaded Java then parallelized web service calls

WAS z/OS and WAS distributed are multi-threaded. Given sufficient processing capacity, the work requests from COBOL may then be handled in a parallel execution fashion.

Primer …

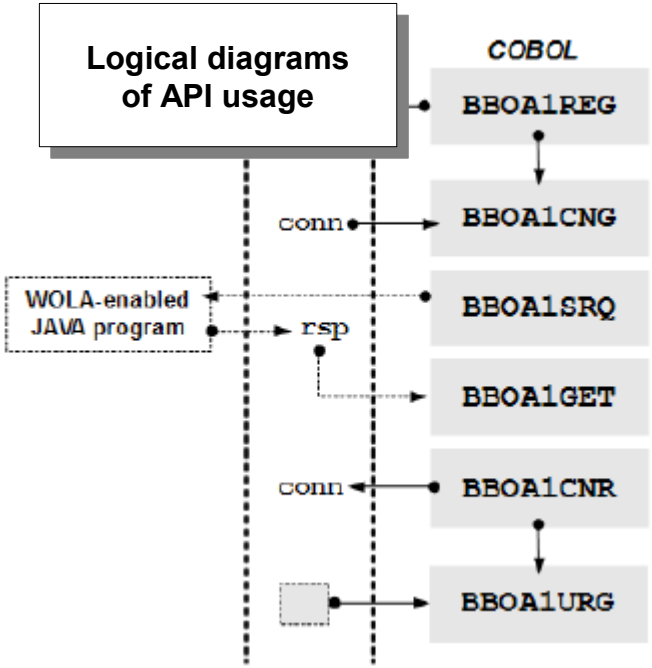
WP101490 Native API "Primer"

Provides a step-by-step introduction to the use of the native APIs with COBOL:



Companion PDF with detailed step-by-step instructions

Logical diagrams of API usage

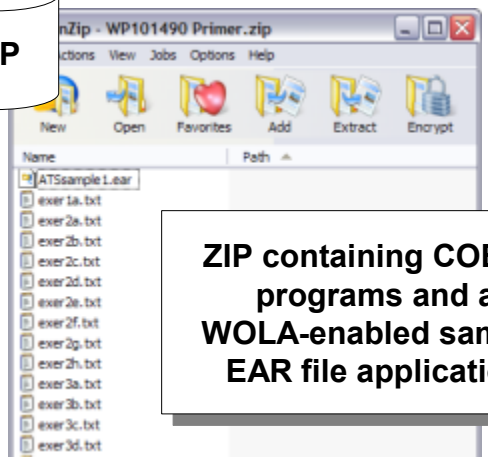


Here's an *example* of a COBOL snippet that would perform the registration:

```
WORKING-STORAGE SECTION.
01 daemongroup                PIC X(8) VALUE LOW-VALUES.  1
01 node-name                  PIC X(8).
01 server-name                PIC X(8).
01 register-name              PIC X(12) VALUE SPACES.
01 minconn                    PIC 9(8) COMP VALUE 1.      2
01 maxconn                    PIC 9(8) COMP VALUE 10.
01 regopts                    PIC 9(8) COMP VALUE 0.      3
01 rc                         PIC 9(8) COMP VALUE 0.
01 rsn                        PIC 9(8) COMP VALUE 0.
  :
MOVE 'OLAINBND'                 TO register-name.
MOVE 'S1CELL'                   TO daemongroup.
MOVE 'S1NODEC'                  TO node-name.           4
MOVE 'S1SR01C'                  TO server-name.
  :
INSPECT daemongroup CONVERTING ' ' to LOW-VALUES.       5
  :
CALL 'BBOA1REG' USING
     daemongroup,
     node-name,
     server-name,
     register-name,                                     6
     minconn,
     maxconn,
     regopts,
     rc,
     rsn.
  :
IF rc > 0 THEN
  DISPLAY "OLA - BBOA1REG problem -- rc/rsn : " rc "/" rsn
  GO TO Bad-RC
END-IF.
```

Working code illustrations

ZIP



ZIP containing COBOL programs and a WOLA-enabled sample EAR file application

When you're ready to begin using the native APIs, this "Primer" will assist you in understanding how the APIs are used

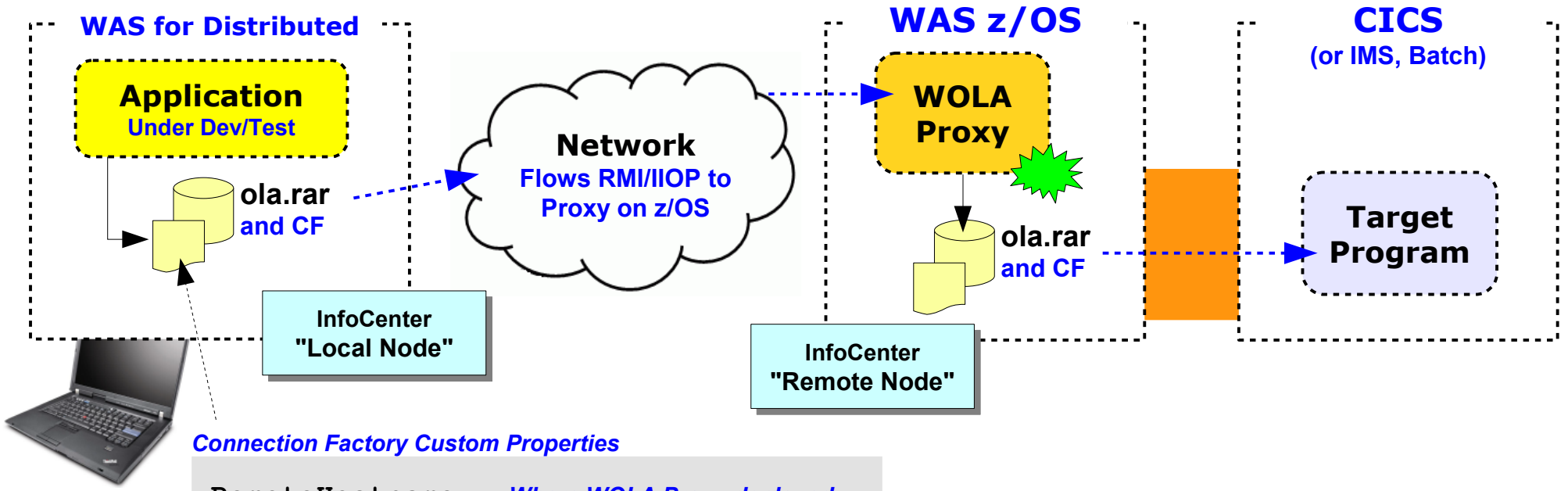
8.0.0.1 ...

New in V8.0.0.1

"Development Mode" using the Proxy Application

Development Mode - *Outbound* Applications

The focus here is on developing and testing WOLA outbound applications without the developer needing direct access to a z/OS system



Connection Factory Custom Properties

```
RemoteHostname = Where WOLA Proxy deployed
RemotePort = ORB port for bootstrap process
RemoteJNDIName = Of WOLA Proxy EJB
```

Limitations:

- Can not participate in global transaction 2PC
- Can not assert distributed WAS thread ID up to z/OS.

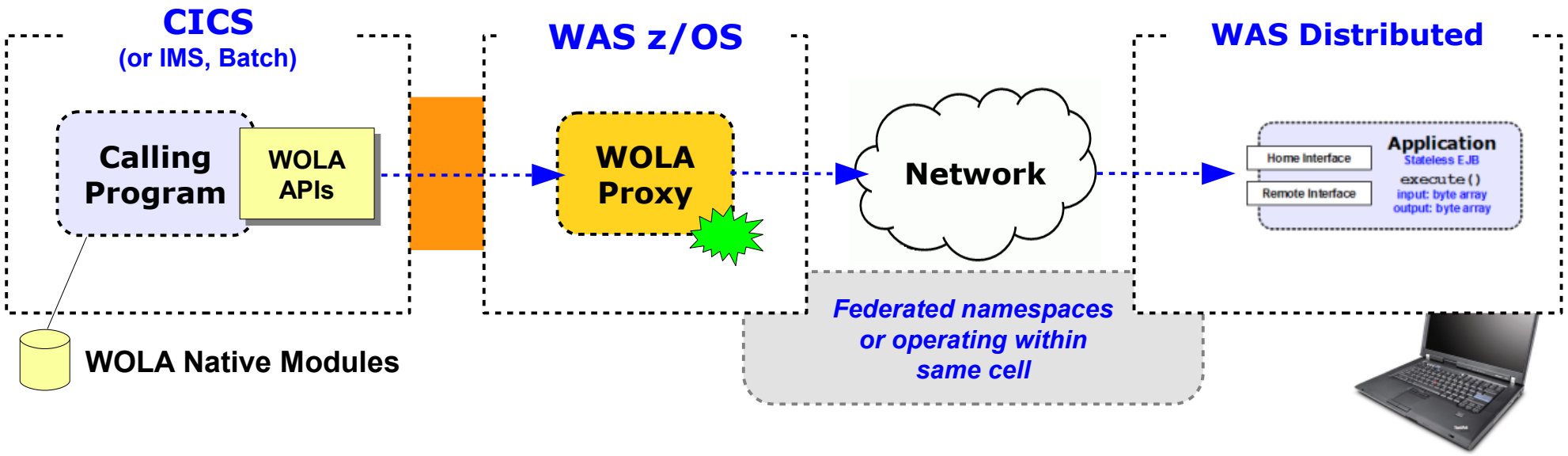
Java developer writes application to CCI in the WOLA JCA resource adapter just as if the application was deployed on WAS z/OS

InfoCenter `cdat_devmode_overview`

Inbound ...

Development Mode - *Inbound* Applications

Let's take the reverse ... the case where you wish a native z/OS program to make an inbound call to a target EJB running in WAS. Can EJB be on WAS distributed? Yes ...



WOLA API developer writes as if target EJB is in the WOLA-attached WAS z/OS server

One parameter difference - `requesttype` on `BBOA1INV` or `BBOA1SRQ` set to "2" (for remote EJB request) rather than "1"

EJB Developer develops stateless EJB with WOLA class libraries as if deployed on z/OS

InfoCenter `cdat_ola_remotequest`