#SHAREorg

SHARE
Technology · Connections · Results

WebSphere Application Server for z/OS: Version 8 - New z/OS Exploitation Features

(Session 10561)

David Follis
IBM
z/OS Runtime Architect

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

CICS*
DB2*
GDPS*
Geographically Dispersed Parallel Sysplex
HiperSockets
IBM*
IBM eServer
IBM logo*
IMS
On Demand Business logo
Parallel Sysplex*
RACF*
System z9
WebSphere*
z/OS
zSeries*

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.
SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.

* All other products may be trademarks or registered trademarks of their respective companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Disclaimer

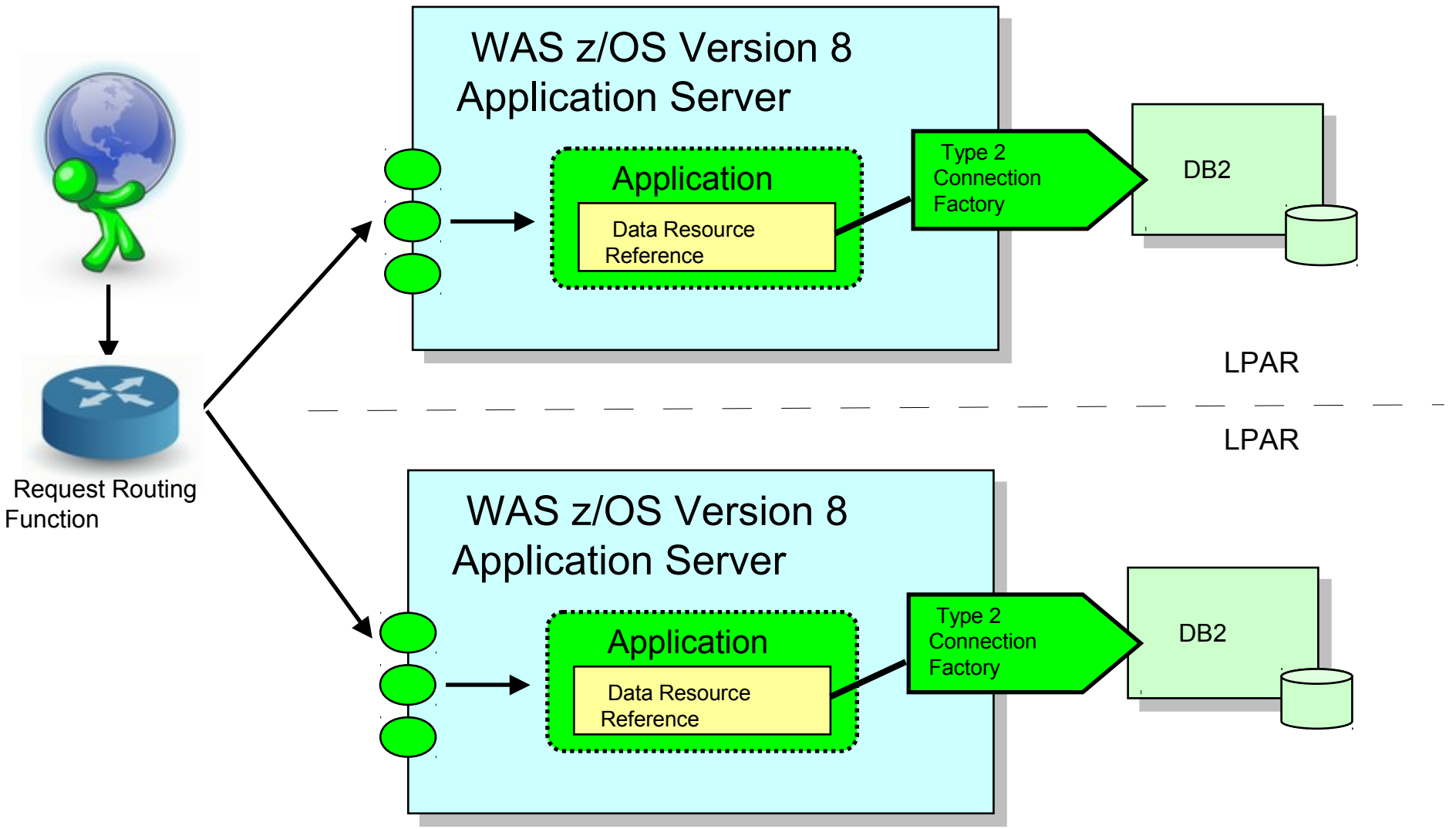
- The information contained in this documentation is provided for informational purposes only. While efforts were many to verify the completeness and accuracy of the information contained in this document, it is provided “as is” without warranty of any kind, express or implied.
- This information is based on IBM’s current product plans and strategy, which are subject to change without notice. IBM will not be responsible for any damages arising out of the use of, or otherwise related to, this documentation or any other documentation.
- Nothing contained in this documentation is intended to, nor shall have the effect of , creating any warranties or representations from IBM (or its suppliers or licensors), or altering the terms and conditions of the applicable license agreement governing the use of the IBM software.
- Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.
- All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

WebSphere Application Server on z/OS

Session	Day	Time	Room	Title	Speaker
10560	Monday	9:30	International Ballroom F	Version 8 – Overview and Update	David Follis
10580	Monday	11:00	Cottonwood A/B	Back to Basics	Mike Loos
10633	Wednesday	1:30	International Ballroom C	Installation Manager – The Cross Platform Installer for WAS	Mike Loos
10561	Wednesday	3:00	Cottonwood A/B	Version 8 – New z/OS Exploitation Features	David Follis
10562	Thursday	11:00	Cottonwood A/B	Batch Update	John Hutchinson
10581	Thursday	1:30	Cottonwood A/B	Getting Started with Version 8 – Part Zero!	Mike Loos
10518	Thursday	6:00	Cottonwood A/B	Potpourri	Anybody
10516	Friday	8:00	Dogwood B	Level 2 Update	Mike Stephen
10563	Friday	9:30	Pine	Hands on Lab	Mike Stephen, David Follis, Ken Irwin

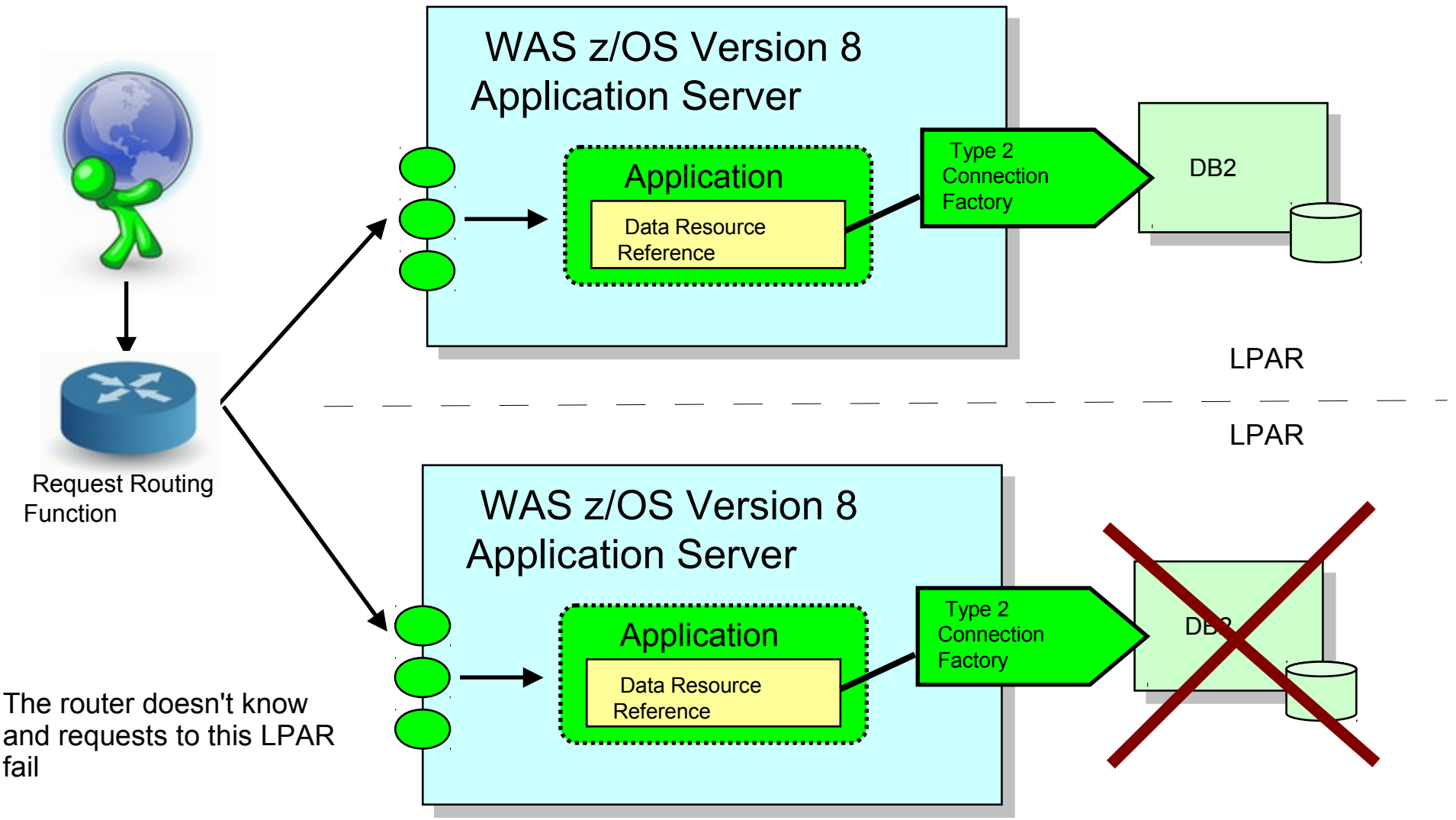
High Availability Connection Management

A Typical Clustered Environment with Type-2 Connectors



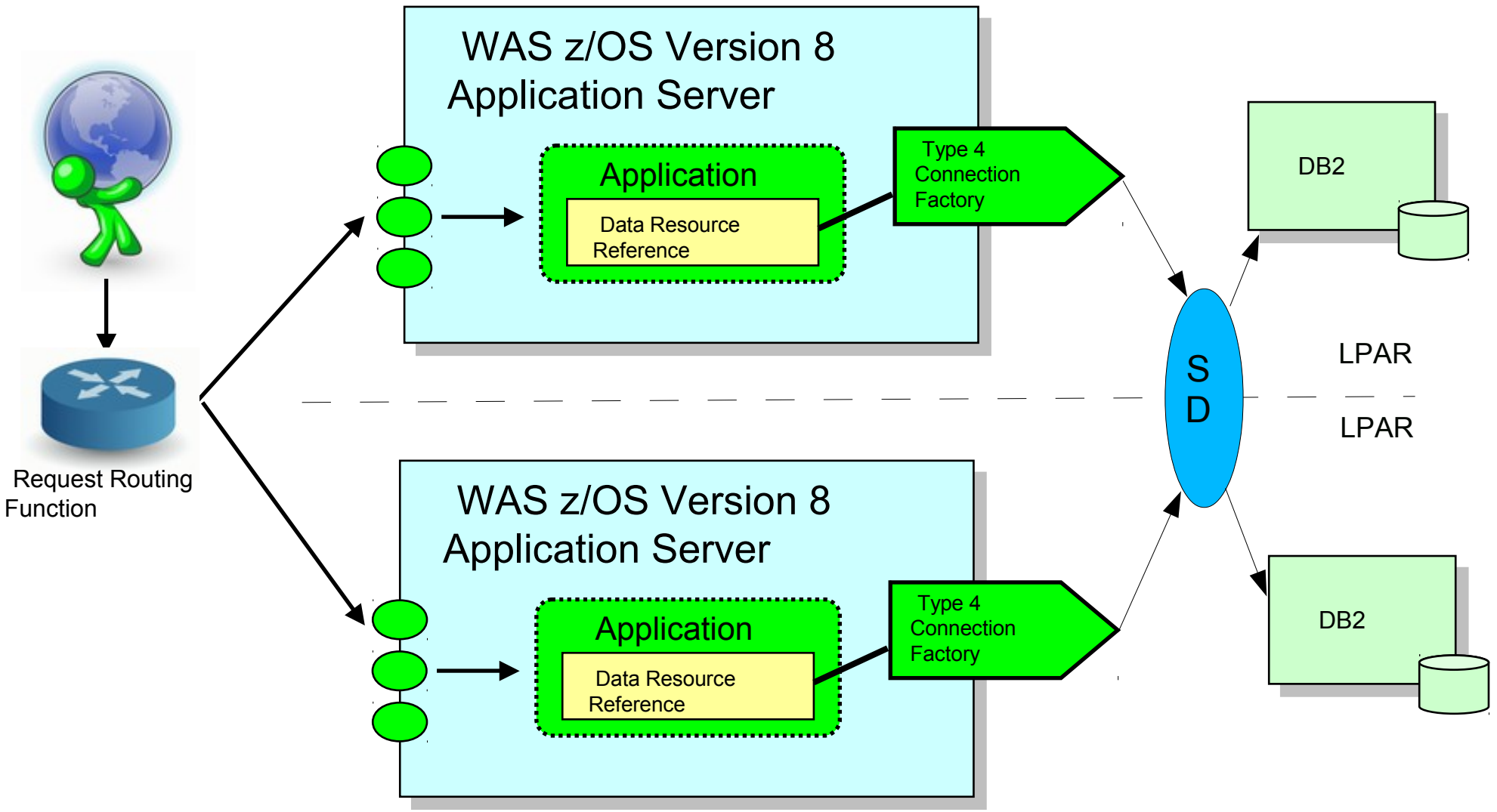
Just using DB2 as an example..

When something bad happens....



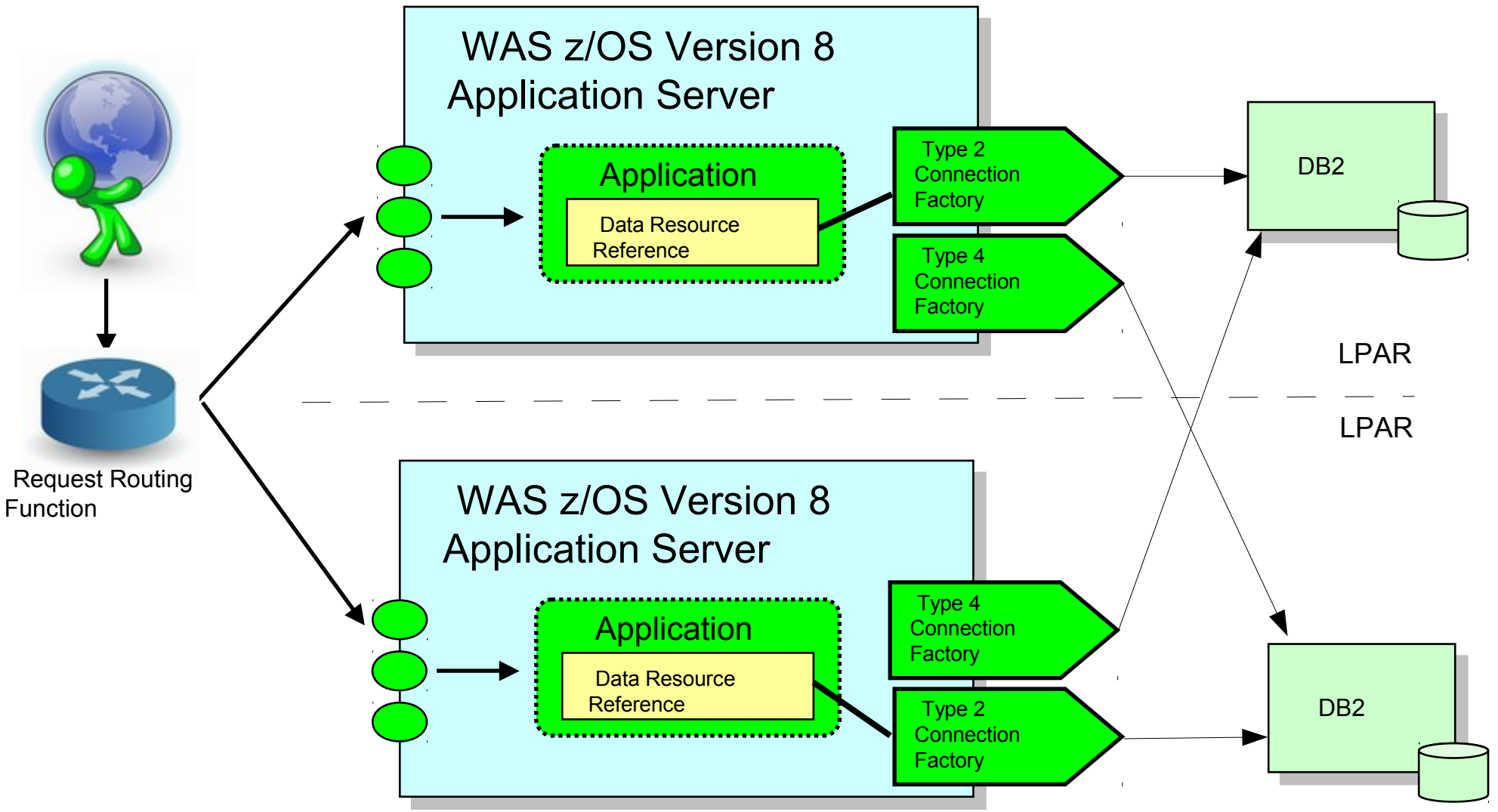
The router doesn't know and requests to this LPAR fail

A Common Solution...



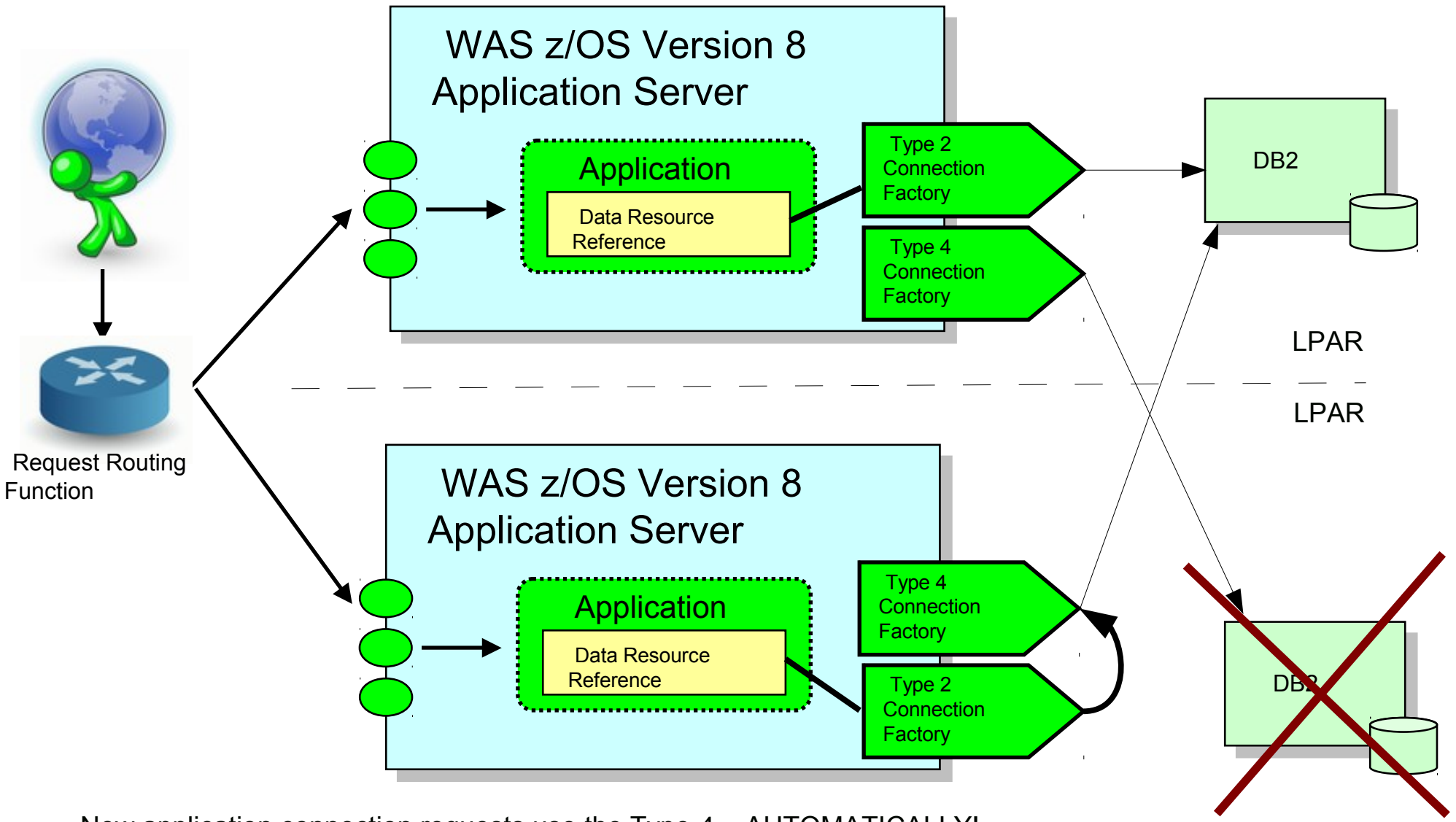
Use Type-4 Connectors and Sysplex Distributor to eliminate close coupling between WAS and DB2... But this surrenders the value of co-location!

Suppose we configure both connectors...



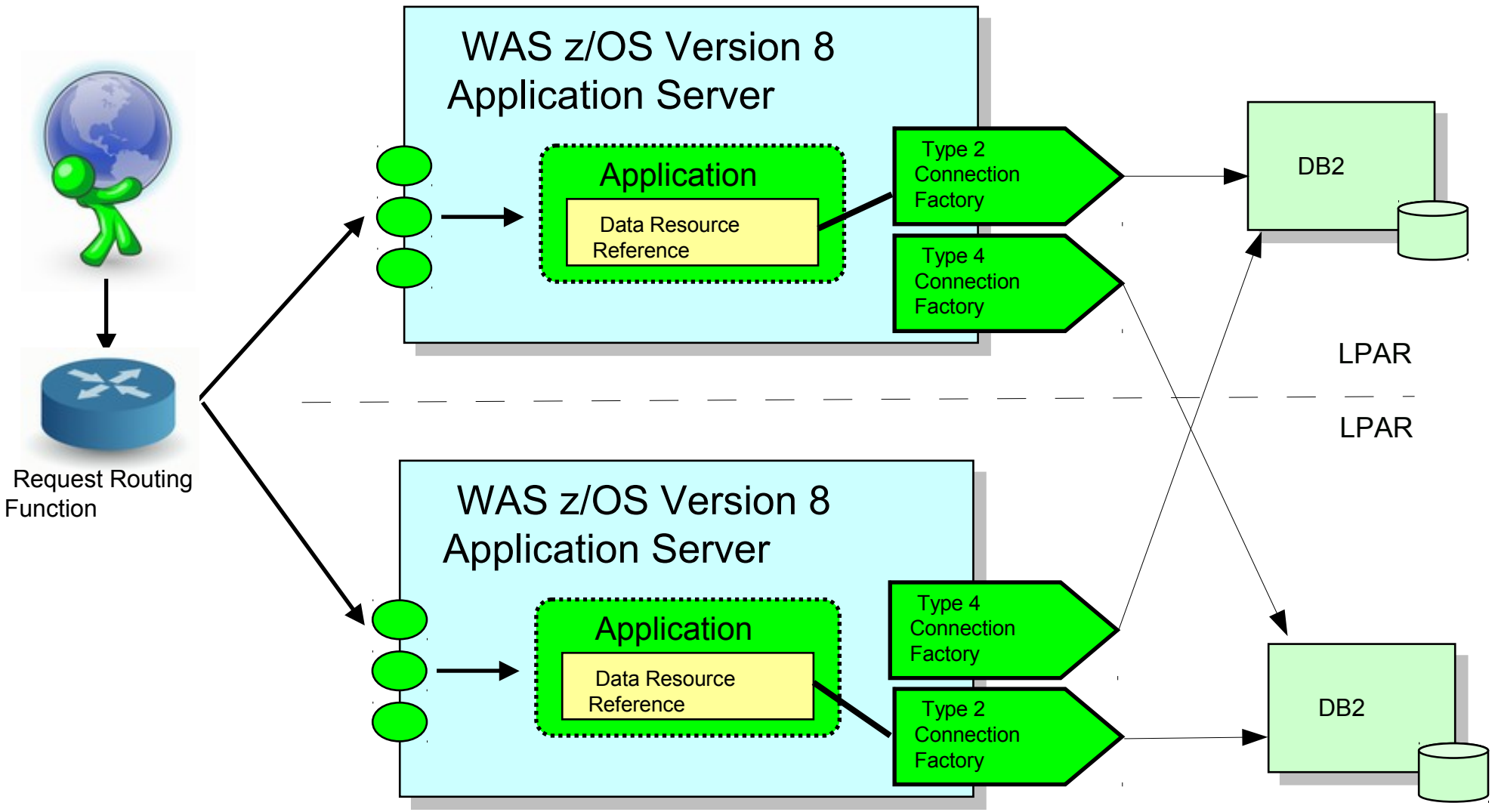
And the Type-4 connector is the 'alternate' for the Type-2

Then something bad happens....



New application connection requests use the Type-4 – AUTOMATICALLY!
WAS will also 'watch' for DB2 to come back....

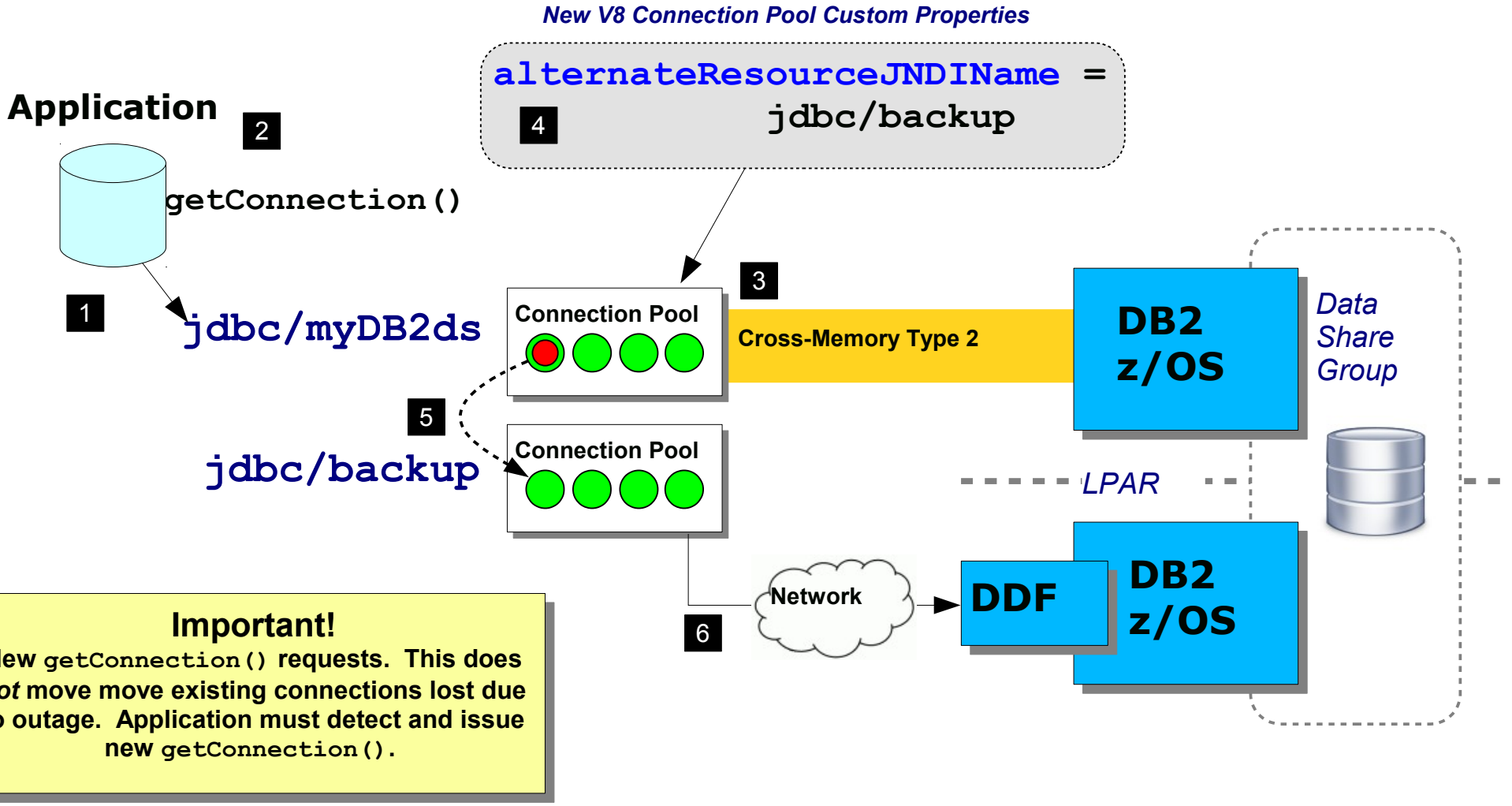
When the bottom DB2 is back...



Use of the Type-4 quiesces and we're back to normal

Essentials of Resource Failover

A new environment variable is used to define an "alternate JNDI" for use when the primary JNDI experiences `getConnection()` problems:



Important!
New `getConnection()` requests. This does *not* move move existing connections lost due to outage. Application must detect and issue new `getConnection()`.

Other Connection Pool Custom Properties

Four other connection pool custom properties are also made available:

`failureThreshold`

Determines the number of consecutive `getConnection()` failures are needed to trigger the failover processing
Integer, Default = 5

`resourceAvailabilityTestRetryInterval`

After failover has occurred, this determines the frequency of polling to see if the primary resource has recovered
Integer, Default = 10 seconds

`enablePartialResourceAdapterFailoverSupport`

Indicates that automatic failover is permitted but automatic failback is disabled
Boolean, Default = False

`disableResourceFailOver`
`disableResourceFailBack`

Disables automatic failover or failback. Used to allow configuration of failover values, but control using z/OS `MODIFY`
Boolean, Default = False

z/OS MODIFY Control of Failover and Failback

The following MODIFY commands will act upon a server where the connection pool custom property `alternateResourceJNDIName` has previously been configured:

Manual Failover to Alternate and Failback to Primary

```
F <server>,FAILOVER,'<JNDI Name>'
F <server>,FAILBACK,'<JNDI Name>'
```

The JNDI name is that of the primary data source. Never the defined alternate data source.

Note the *single quotes* enclosing the JDNI name

Manual Disable or Enable of Automatic Failover / Failback

```
F <server>,DISABLEFAILOVER,'<JNDI Name>'
F <server>,ENABLEFAILOVER,'<JNDI Name>'
```

The JNDI name is that of the primary data source. Never the defined alternate data source.

These MODIFY commands override connection pool custom properties you may have set of enable and disable of failover and failback

z/OS **failureNotificationActionCode**

These define actions to take when the primary is unreachable *and* any defined alternate JNDI resources are also unreachable:

```
failureNotificationActionCode = 1 | 2 | 3
```

1 **Issue a `BBOJ0130I` message, but take no other action**

```
BBOJ0130I: CONNECTION MANAGEMENT IN A SERVANT REGION DETECTED THAT THE
RESOURCE IDENTIFIED BY JNDI NAME jdbc/type2ds IS DISCONNECTED FROM SERVER
z9cell/z9nodea/Z9SR01/z9sr01a. ACTION TAKEN: NONE.
```

2 **Issue `PAUSELISTENERS` for the server; `RESUMELISTENERS` when resource is back**

```
ACTION TAKEN: PAUSING LISTENERS.
BBOO0222I: ZAIO0002I: z/OS asynchronous IO TCP Channel TCP_1 has stopped
listening on host * port 10065.
 :
BBOO0222I: ZAIO0002I: z/OS asynchronous IO TCP Channel TCP_4 has stopped
listening on host * port 10068.
```

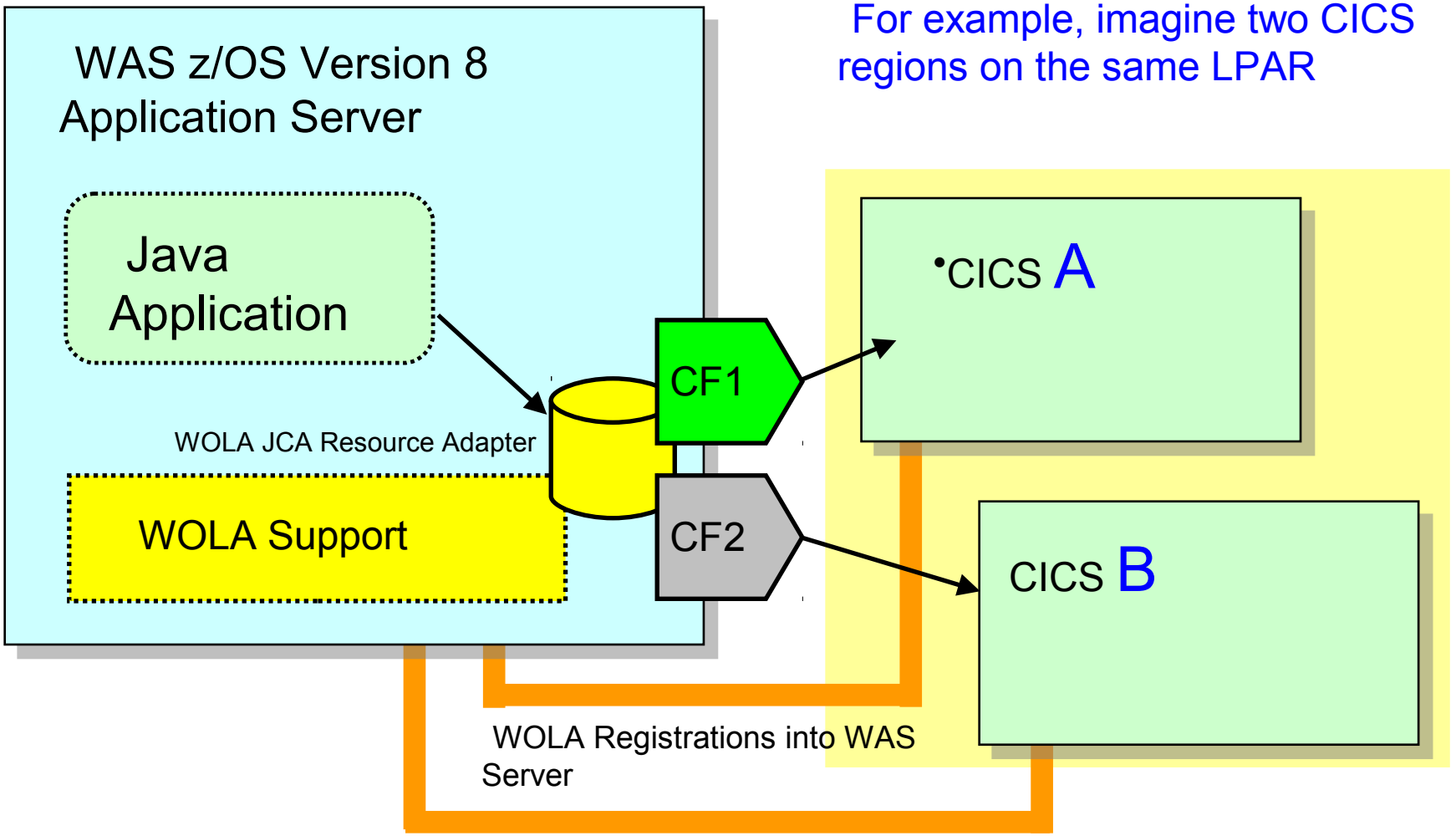
Front-end routing devices will detect loss of listener ports and route to other members of a cluster

3 **Stop applications using failed resource; restart applications when resource is back**

Makes affected application unavailable but leaves intact other applications in the server

WOLA Variation on This New Function

WOLA participates in this as well in that a backup registered external address space now be used in the event the primary is lost:



WOLA is by definition “same LPAR,” and this gives you a degree of availability by allowing routing to secondary registered external address space

This is tricky stuff... Remember

TEST IT BEFORE YOU NEED IT

Granular RAS Controls

(Agenda)

XML File Extended -- Control Driven to Request Level

The XML file identifies requests ... this new function then picks up and drives various WAS behavior controls from server level down to the request level:

```
<Classification schema_version="1.0">
   <InboundClassification type="http" schema_version="1.0"
      default_transaction_class="Z9DEFLT" >
      <http_classification_info
            uri="/SuperSnoopWeb/*" transaction_class="Z9TRANA"
            description="Snoop"                                  />
      <http_classification_info
            uri="/MyIVT/*" transaction_class="Z9TRANB"
            description="MyIVT"                                  />
   </InboundClassification>
</Classification>
```

Granular Control to Request Level

Granular Control to Request Level

Various Timeouts

Stalled Thread Dump Actions

CPU Time Used Limit

DPM Interval and Dump Action

SMF Recording

Tracing

Message Tagging

Timeout Recovery Actions

Topics to Cover in this Section:

- What those functions are and how they work
- How to dynamically reload a new or updated XML file
- How to dynamically revert to previous XML file

First Example - Dispatch Timeout

Work dispatched from queue to servant starts a timer to control timeout of that work. Before: environment variable, server level at best. Now: request level:

Environment Variable

```
IIOP     control_region_wlm_dispatch_timeout
HTTP     protocol_http_timeout_output
HTTPS    protocol_https_timeout_output
MDB      control_region_mdb_request_timeout
WOLA     control_region_wlm_dispatch_timeout
```

The current environment variables for HTTP dispatch timeouts. Granular down to server.

Other protocol dispatch timeouts

XML Classification File

```
<Classification schema_version="1.0">
   <InboundClassification type="http" schema_version="1.0"
      default_transaction_class="TRANCL" >
      <http_classification_info
            uri="/SuperSnoopWeb/*" transaction_class="TRANCL"
            description="Snoop" dispatch_timeout="60" />
      <http_classification_info
            uri="/MyIVT/*" transaction_class="TRANCL"
            description="MyIVT" dispatch_timeout="15" />
   </InboundClassification>
</Classification>
```

XML Classification file section for HTTP. File supports `http`, `iiop`, `mdb`, `ola`, `sip`, `internal`

Requests matching this get 60 second timeout

Requests matching this get 15 second timeout

If timeout in XML and it applicable then it takes precedence over configured environment variable

The Available Granular Control Options

Here's a complete list of the options available with this new function:

Previous chart

```
dispatch_timeout="______
queue_timeout_percent ="_____"
request_timeout="_____"
stalled_thread_dump_action="_____"
cputimeused_limit="_____"
cputimeused_dump_action="_____"
dpm_interval="_____"
dpm_dump_action="_____"
SMF_request_activity_enabled="__"
SMF_request_activity_timestamps="__"
SMF_request_activity_security="__"
SMF_request_activity_CPU_detail="__"
classification_only_trace="__"
message_tag="_____"
timeout_recovery="_____">
```

Timeout for time spent in queue prior to dispatching to servant
Expressed as a percent of the dispatch timeout
Example:

- Dispatch = 300 seconds
- Queue = 10 percent
- Request must be dispatched to servant within 30 seconds or request times out
- Set this too high and request sits in queue and if dispatched has very little time to complete

Multiple Keywords in XML Acceptable

At this point you may be wondering whether multiple keywords can be coded in the XML, and the answer is yes ...

```
<InboundClassification type="http"
   schema_version="1.0" default_transaction_class="TRANCL" >
   <http_classification_info transaction_class="TRANCL"
      host="host.company.com"
      dispatch_timeout="300"
      stalled_thread_dump_action="traceback" >
      <http_classification_info transaction_class="TRANA"
         uri="/SuperSnoop/*"
         dispatch_timeout="60"
         queue_timeout_percent="10"
         cputimeused_limit="500" />
      <http_classification_info transaction_class="TRANB"
         uri="/MyIVT/*" dispatch_timeout="15"/>
   </http_classification_info>
</InboundClassification>
```

These will apply to lower nodes in the nested XML unless overridden at lower level

Example of three keywords used for the SuperSnoop classification node on the XML tree

Request Timeout and CPU Time Used Limit

```
dispatch_timeout="_____"
queue_timeout_percent ="_____"
request_timeout="_____"
stalled_thread_dump_action="_____"
cputimeused_limit="_____"
cputimeused_dump_action="_____"
dpm_interval="_____"
dpm_dump_action="_____"
SMF_request_activity_enabled="__"
SMF_request_activity_timestamps="__"
SMF_request_activity_security="__"
SMF_request_activity_CPU_detail="__"
classification_only_trace="__"
message_tag="_____"
timeout_recovery="_____">
```

Timeout for *outbound* requests issued by Java programs in servant
It is a request from the perspective of the servlet or EJB
Expressed in seconds

Maximum CPU this request may consume before having the WLM enclave quiesced
Expressed in milliseconds

Dump Action When Timeout Occurs

```
dispatch_timeout="_____"
queue_timeout_percent ="_____"
request_timeout="_____"
stalled_thread_dump_action="_____"
cputimeused_limit="_____"
cputimeused_dump_action="_____"
dpm_interval="_____"
dpm_dump_action="_____"
SMF_request_activity_enabled="__"
SMF_request_activity_timestamps="__"
SMF_request_activity_security="__"
SMF_request_activity_CPU_detail="__"
classification_only_trace="__"
message_tag="_____"
timeout_recovery="_____">
```

This controls what happens when two other controls expire:

dispatch_timeout
cputimeused_limit

Options are:

svcdump
javacore
heapdump
traceback
javatdump
none

Dispatch Progress Monitor (DPM) Settings

```
dispatch_timeout="_____"
queue_timeout_percent ="_____"
request_timeout="_____"
stalled_thread_dump_action="_____"
cputimeused_limit="_____"
cputimeused_dump_action="_____"
dpm_interval="_____"
dpm_dump_action="_____"
SMF_request_activity_enabled="__"
SMF_request_activity_timestamps="__"
SMF_request_activity_security="__"
SMF_request_activity_CPU_detail="__"
classification_only_trace="__"
message_tag="_____"
timeout_recovery="_____">
```

DPM stands for Dispatch Progress Monitor. It is a function that will process a dump action every *n* seconds.
`dpm_interval` is the interval period expressed in seconds
`dpm_dump_action` is the same as we just saw for the other dump action: `svcdump`, `javacore`, `heapdump`, `traceback`, `javatdump` and `none`

This function has a set of `MODIFY` commands that may be used to clear DPM settings or reset to XML settings

DPM Modify Command Options

MODIFY server,DPM,CLEAR_ALL

Override XML, DPM Action=None, All DPM Intervals to zero

MODIFY server,DPM,RESET

Honor the XML content

MODIFY server,DMP,DUMP_ACTION=NONE (or other values)

Ignore DPM dump actions in the XML

MODIFY server,DPM,DUMP_ACTION=RESET

Honor DPM dump actions in the XML

MODIFY server,DPM,HTTP=500 (or other protocol, other values)

Ignore DPM intervals in the XML for HTTP and use 500 instead

MODIFY server,DPM,HTTP=RESET

Honor HTTP DPM intervals in the XML

MODIFY server,DPM,INTERVAL=0 (or other values)

Ignore DPM intervals in the XML and set all intervals to zero

MODIFY server,DPM,INTERVAL=RESET

Honor all DPM intervals in the XML

SMF 120.9 Recording

```
dispatch_timeout="_____"
queue_timeout_percent ="_____"
request_timeout="_____"
stalled_thread_dump_action="_____"
cputimeused_limit="_____"
cputimeused_dump_action="_____"
dpm_interval="_____"
dpm_dump_action="_____"
SMF_request_activity_enabled="__"
SMF_request_activity_timestamps="__"
SMF_request_activity_security="__"
SMF_request_activity_CPU_detail="__"
classification_only_trace="__"
message_tag="_____"
timeout_recovery="_____">
```

WAS z/OS Version 7 introduced a new SMF record format -- the SMF 120 subtype 9 records.
With WAS z/OS V8 the recording of SMF 120.9 records now down to identified requests
This includes the base records as well as the optional additional information records.

Value is `0` (off) or `1` (on)
`F <server>,SMF,REQUEST,OFF` will override XML
`F <server>,SMF,REQUEST,RESET` will go back to XML settings

Tracing for Identified Requests Only

```
dispatch_timeout="_____"
queue_timeout_percent ="_____"
request_timeout="_____"
stalled_thread_dump_action="_____"
cputimeused_limit="_____"
cputimeused_dump_action="_____"
dpm_interval="_____"
dpm_dump_action="_____"
SMF_request_activity_enabled="__"
SMF_request_activity_timestamps="__"
SMF_request_activity_security="__"
SMF_request_activity_CPU_detail="__"
classification_only_trace="__"
message_tag="_____"
timeout_recovery="_____">
```

Prior to V8 tracing was granular to server only. All activity in the server traced. That often resulted in a great deal of trace output.

This allows you to set a trace level for the server, but trace only identified requests.

Value is `0` (off) or `1` (on)

If WAS z/OS sees this value set to 1 in the XML file, then tracing is done only for matching records.

MODIFY server,TRACERECORD,OFF
- turns all tracing off

MODIFY server,TRACERECORD,ON
- ignores the XML, trace on

MODIFY server,TRACERECORD,RESET
- honor the XML

MODIFY server,DISPLAY,TRACERECORD
- shows current setting

Custom Message Tagging

```
dispatch_timeout="_____"
queue_timeout_percent ="_____"
request_timeout="_____"
stalled_thread_dump_action="_____"
cputimeused_limit="_____"
cputimeused_dump_action="_____"
dpm_interval="_____"
dpm_dump_action="_____"
SMF_request_activity_enabled="__"
SMF_request_activity_timestamps="__"
SMF_request_activity_security="__"
SMF_request_activity_CPU_detail="__"
classification_only_trace="__"
message_tag="_____"
timeout_recovery="_____">
```

Message tagging goes to JES but not to HPEL

This allows you to place a custom string on all log, trace and system messages output for requests that match the classification.

Up to 8 characters
Output shows up as:
`tag=MYTAG`
within the log, trace or message.

This may affect system automation. Either correct system automation, or not use in XML, or specify environment variable:
`ras_tag_wto_messages = 0`

That tells WAS to ignore XML settings for message tags written to the operator console.

Message Tagging Examples

```
<InboundClassification type="http"    . . .>
   <http_classification_info uri=”/gcs/*” message_tag=”GCS”. . ./>
</InboundClassification>
```

The tag 'GCS' will show up in messages, traces, and printlns issued by a thread dispatching any request that matches /gcs/*

For example:

```
Trace: 2011/03/21 22:15:48.298 02 t=6BEE88 c=0.6 key=S2 tag=GCS (0401D00A)

BossLog: { 0233} 2011/03/24 14:05:52.951 03 SYSTEM=SY1 CELL=WAS00 NODE=NDN1
CLUSTER=BBOC001 SERVER=BBOS001  PID=0X010063 TID=0X3156630000000043 t=6C6938 c=UNK
./bbgrjtr.cpp+717  tag=GCS ... BBOO0220E: SECJ6237E: Authorization failed. The SAF user
MSTONE1 does not have READ access to any of the following SAF profiles in the EJBROLE
class:  [All#Role]. com.ibm.ws.security.zOS.authz.SAFAuthorizationTableImpl
```

Timeout Recovery Option

```
dispatch_timeout="_____"
queue_timeout_percent ="_____"
request_timeout="_____"
stalled_thread_dump_action="_____"
cputimeused_limit="_____"
cputimeused_dump_action="_____"
dpm_interval="_____"
dpm_dump_action="_____"
SMF_request_activity_enabled="__"
SMF_request_activity_timestamps="__"
SMF_request_activity_security="__"
SMF_request_activity_CPU_detail="__"
classification_only_trace="__"
message_tag="_____"
timeout_recovery="_____">
```

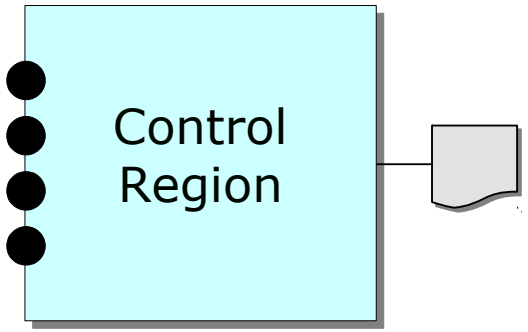
We are accustomed to a timeout resulting in an EC3 abend of the servant region.
The V7 feature to delay timeout abends, particularly with the hung thread threshhold setting, could delay loss of the servant.
This new function in V8 allows you to set the recovery action:

`SERVANT` - normal EC3 abend (or delay if hung thread threshhold in play)

`SESSION` - sends error message to client, then closes the TCP socket and the HTTP session. Servant stays up. Thread either completes or ends up hung.

How XML File Can Be Read and Made Active

There's a few ways to bring an XML file or changes to an XML file into the server:



Environment Variable

```
wlm_classification_file = /<path>/<file>
```

Then start or restart the server

MODIFY to load initial or replace existing

```
F <server>,RECLASSIFY,FILE='/<path>/<file>'
```

WAS will load the specified file.

MODIFY to reread the current file

```
F <server>,RECLASSIFY
```

WAS will reread the current file picking up whatever changes you have made

Checking The State of the Classification File

Here's a quick summary of what to check for to make certain what file was loaded and whether any XML parsing errors occurred:

In the Control Region output -- Positive Sign

```
BBOJ0129I: The /wasetc/was8lab/other/classification.xml workload
classification file was loaded at 2011/11/25 12:22:22.710 (EST)
```

In the Control Region output -- Sign of Problems

```
BBOJ0085E: PROBLEMS ENCOUNTERED PARSING WLM CLASSIFICATION XML FILE
```

It then offers fairly good details on what the problem is

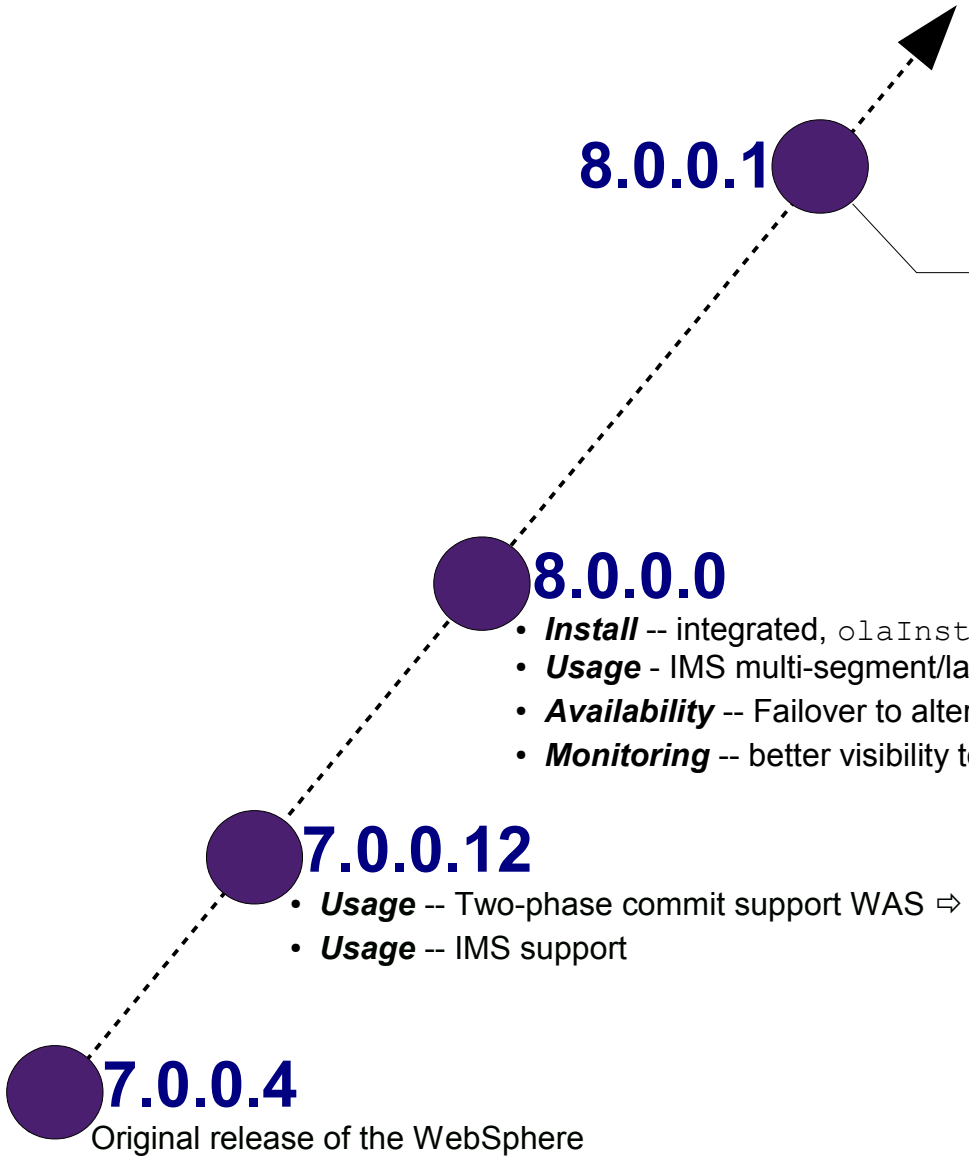
MODIFY to see the state of the XML file

```
F Z9SR01A,DISPLAY,WORK,CLINFO
  :
BBOJ0129I: The /wasetc/was8lab/other/classification.xml
workload classification file was loaded at 2011/11/26
14:58:28.586 (EST)
```

SHARE
Technology · Connections · Results

WebSphere Optimized Local Adapters (WOLA)

History of Functional Updates



8.0.0.1

- ***Development/Test***
  - Test code on distributed development machine and have WOLA call "proxied" to z/OS where WAS z/OS server and backend data systems are located
  - Support for inbound calls for EJB that resides in another WAS server from that connected to with WOLA registration
- ***Work Distribution***
  - Ability to round-robin between multiple external address spaces registered into server with the same register name
- ***Monitoring***
  - New SMF 120.10 record for outbound WOLA calls
  - Correlator tokens for WAS to CICS TS 4.2 requests
- ***APIs*** -- support for 64-bit calls from C/C++ and Assembler

8.0.0.0

- ***Install*** -- integrated, `olaInstall.sh` no longer needed
- ***Usage*** - IMS multi-segment/large message (>32K) support
- ***Availability*** -- Failover to alternate connection factory JNDI and failback
- ***Monitoring*** -- better visibility to WOLA *inbound* workload in SMF 120.9

7.0.0.12

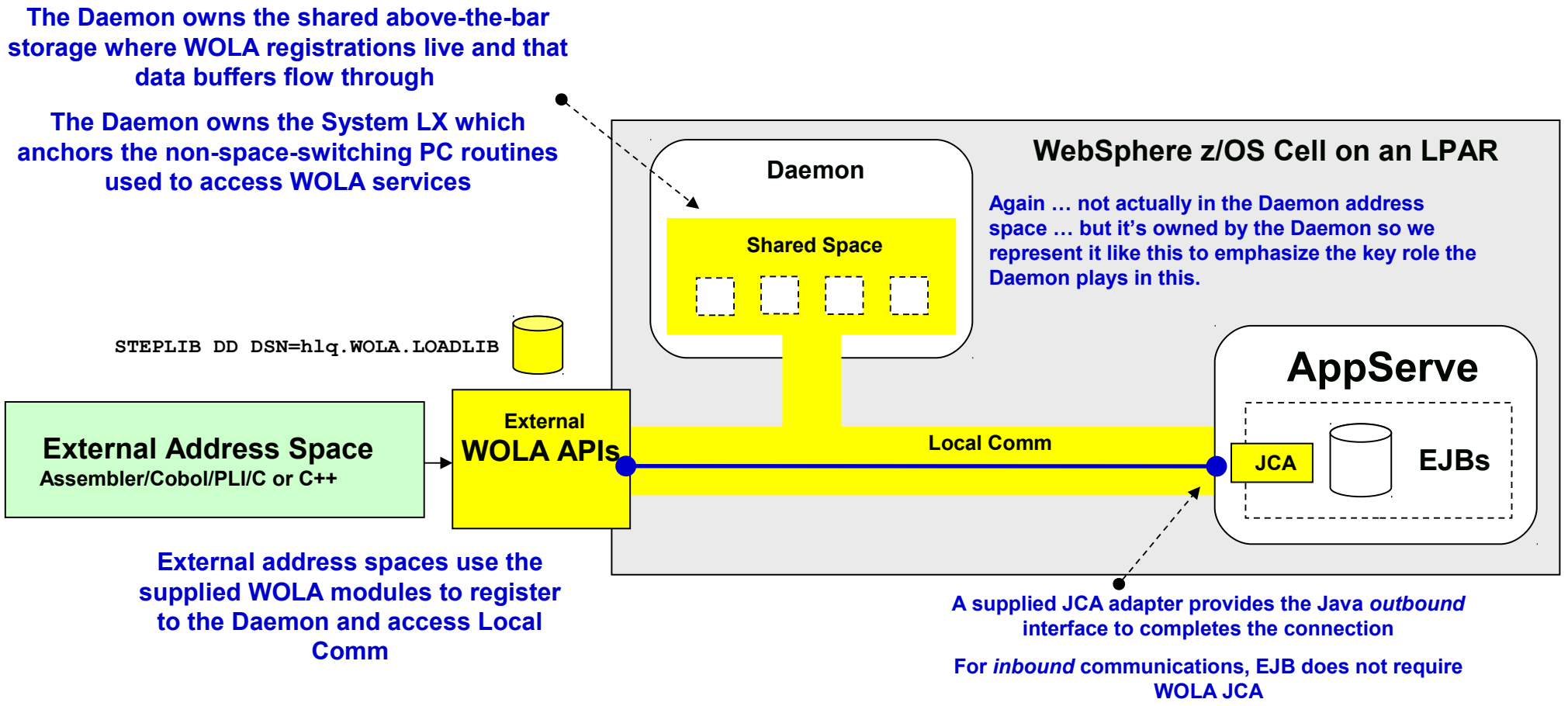
- ***Usage*** -- Two-phase commit support WAS ⇨ CICS 4.1
- ***Usage*** -- IMS support

7.0.0.4

Original release of the WebSphere Optimized Local Adapters function

Very High Level Picture of WOLA

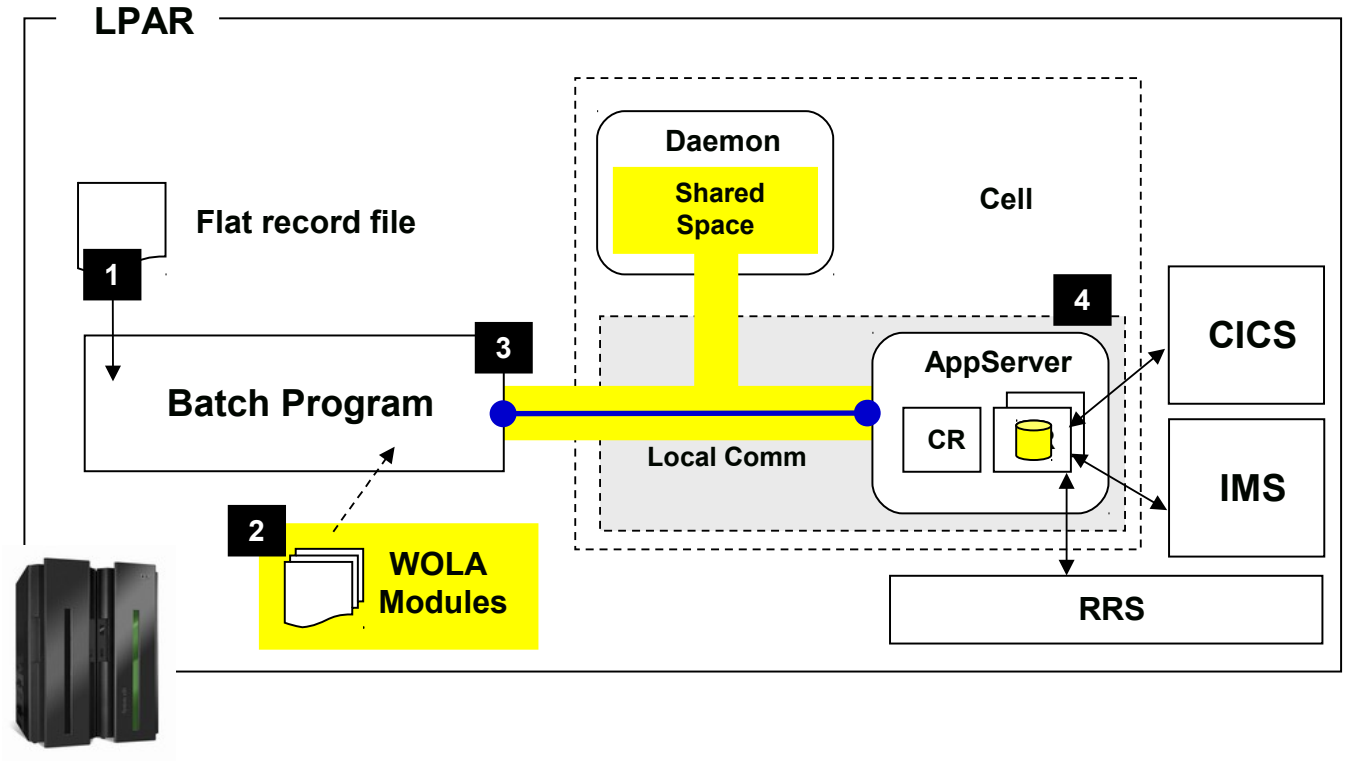
This is just a schematic, but it helps position some key concepts:



Essentially this is a set of APIs that externalizes the Local Comm function that is already in use within a WebSphere cell on an LPAR

A Simple Use-Case Scenario

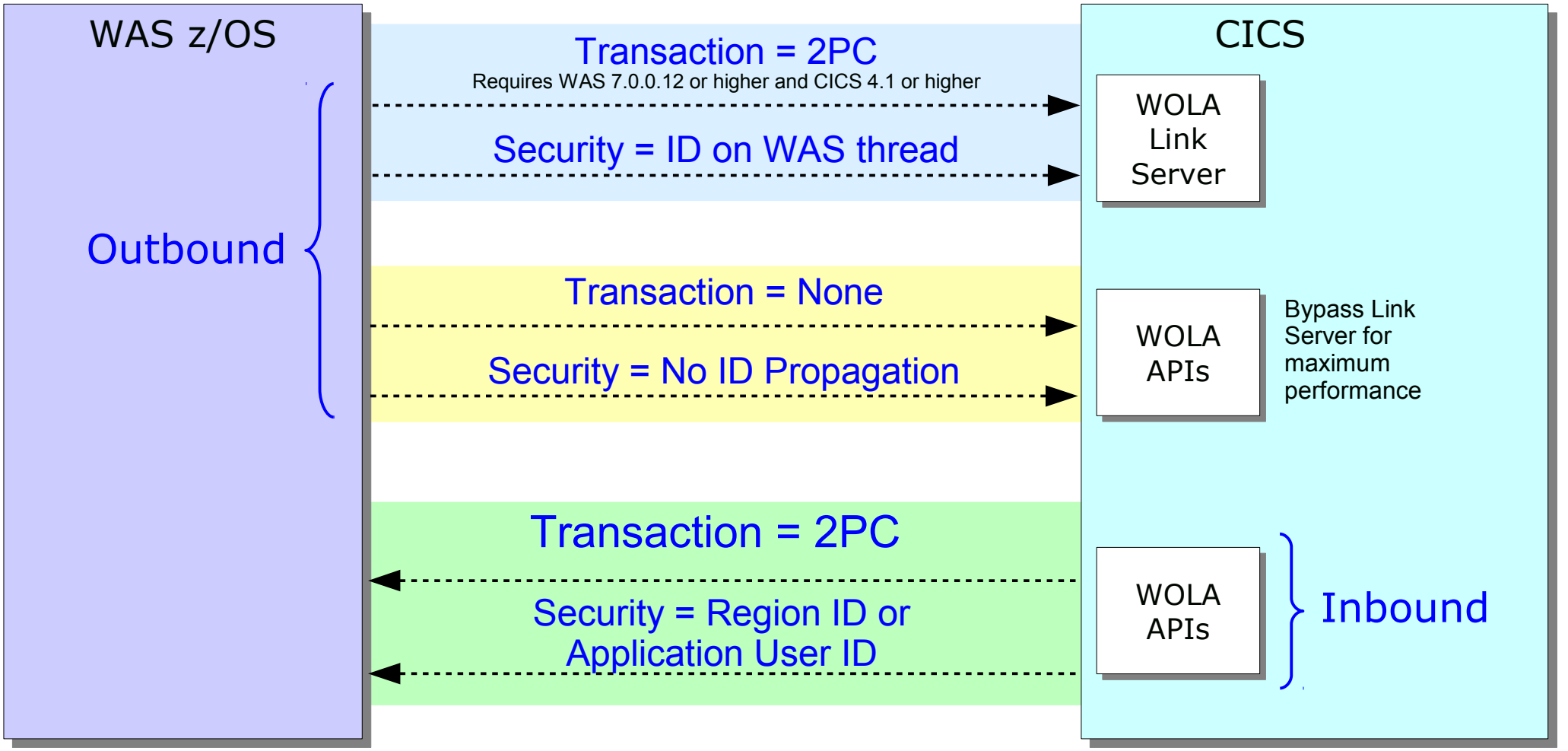
Here's a picture … just to help us get our minds around this thing:



1. Flat record file serves as input to batch program
2. WOLA modules STEPLIBed to from batch JCL
3. Batch program uses WOLA APIs to access WAS and invoke EJB
4. EJB initiates transaction and updates CICS, IMS with two-phase commit using standard WAS data connector archictures

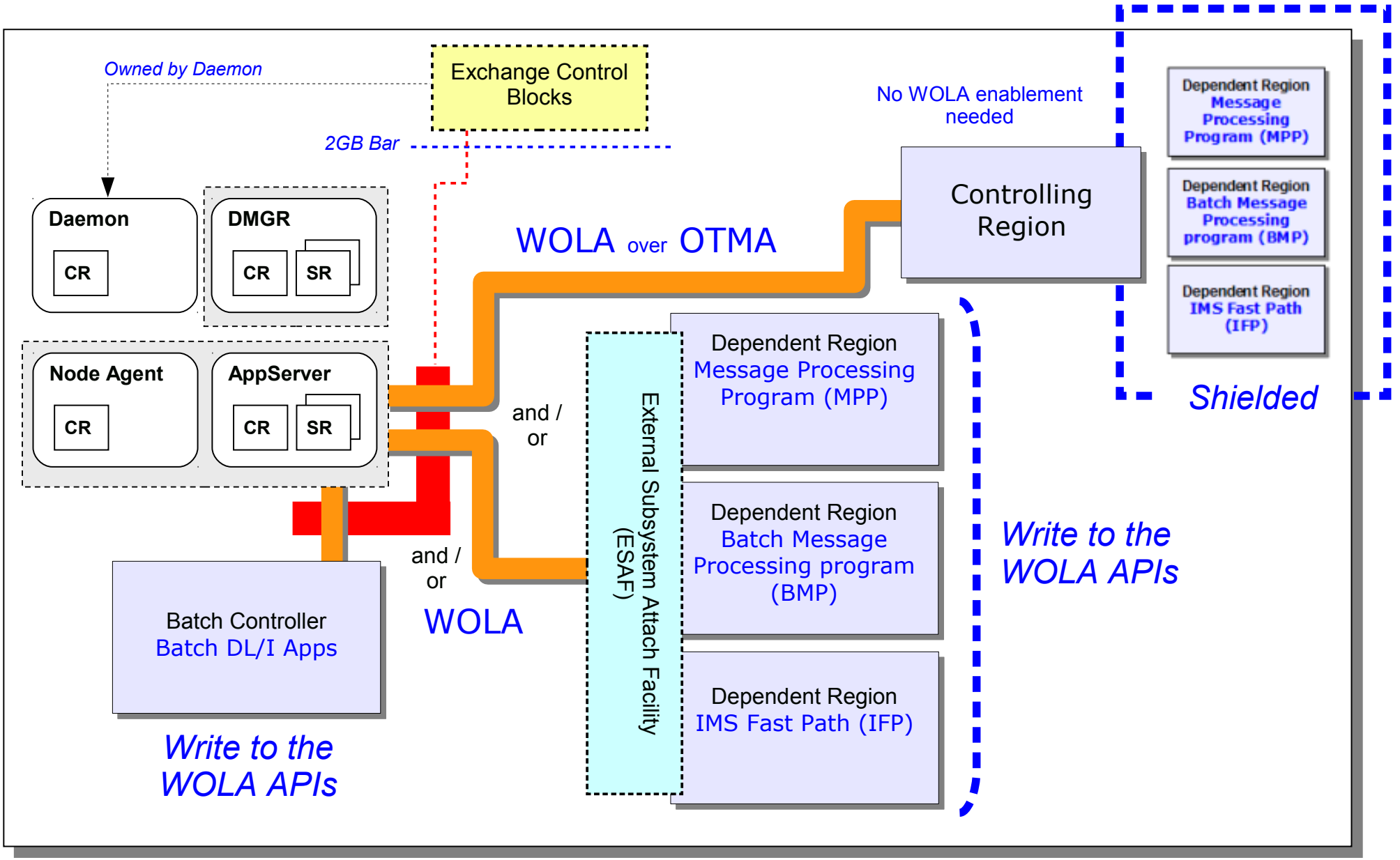
This illustrates a relatively simple -- but likely common -- usage: batch file using an existing transactional EJB to update data.

WOLA and CICS, Transaction and Security



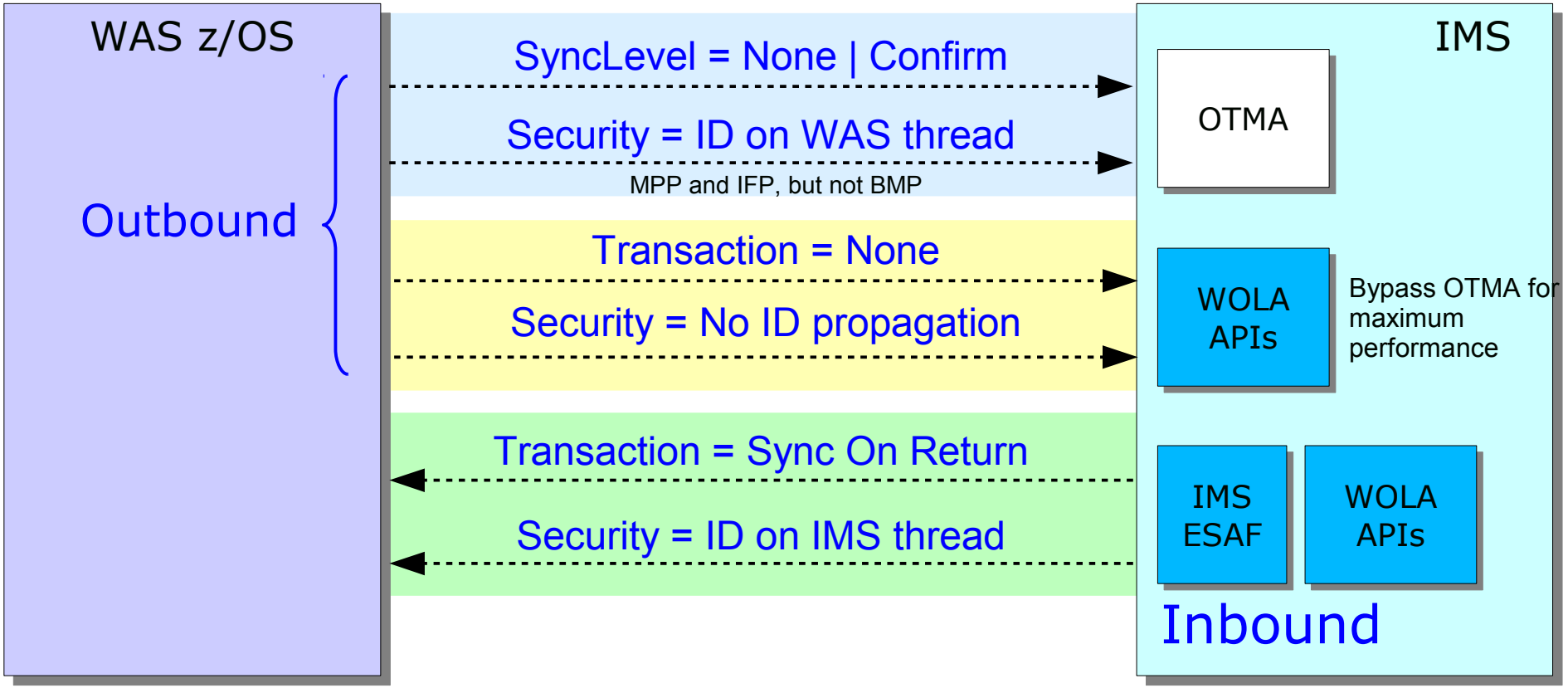
See the WP101490 "Design and Planning Guide or the InfoCenter for the specific details of this

High Level Overview of IMS Support – (requires 7.0.0.12)



WOLA and IMS, Transaction and Security

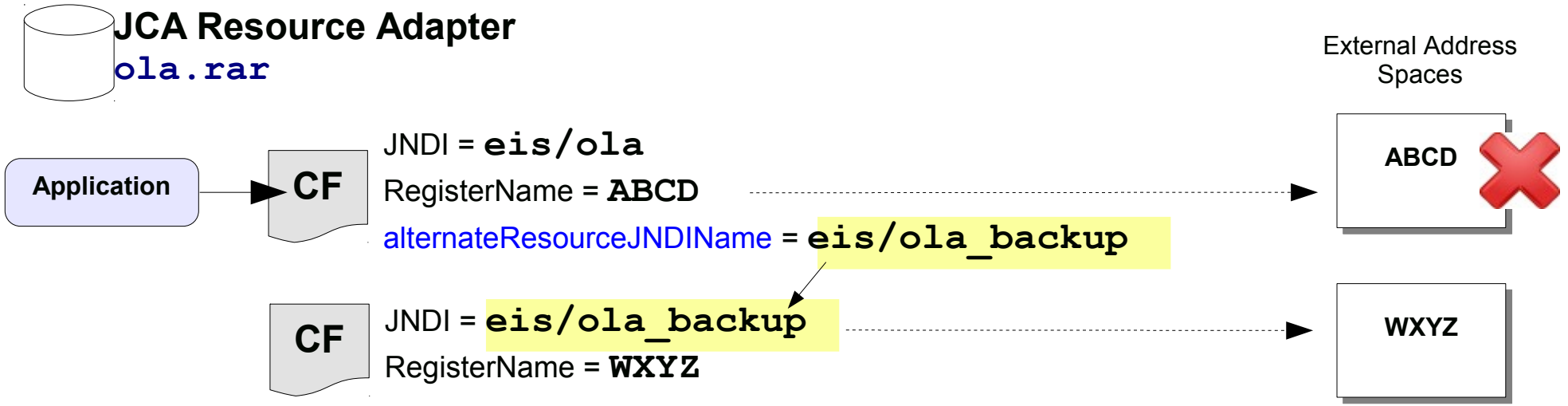
A summary picture:



8.0.0.0 Release

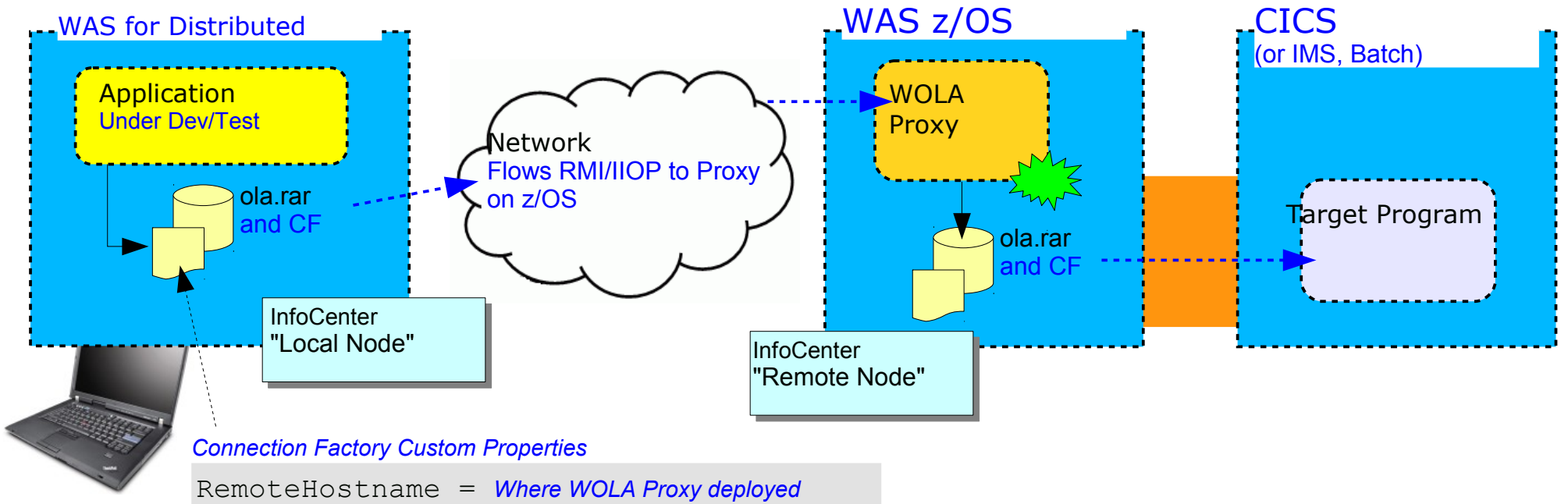
Functional Updates:

- **Integration script `olaInstall.sh` no longer needed to enable function in a node**
  New script `copyZOS.sh` is what copies WOLA modules to pre-allocated load library. The samples are now located in the directory `/util/zos/OLASamples` under the node's install root.
- **Support for IMS large multi-segment messages (> 32K) added**
- **Support for inbound transaction classification separate from IIOP, and identification in SMF 120.9 as OLA inbound call. See:**
- **Availability -- ability to fail over to alternate JNDI for outbound calls and then fail back upon recovery of the original external address space**



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
```

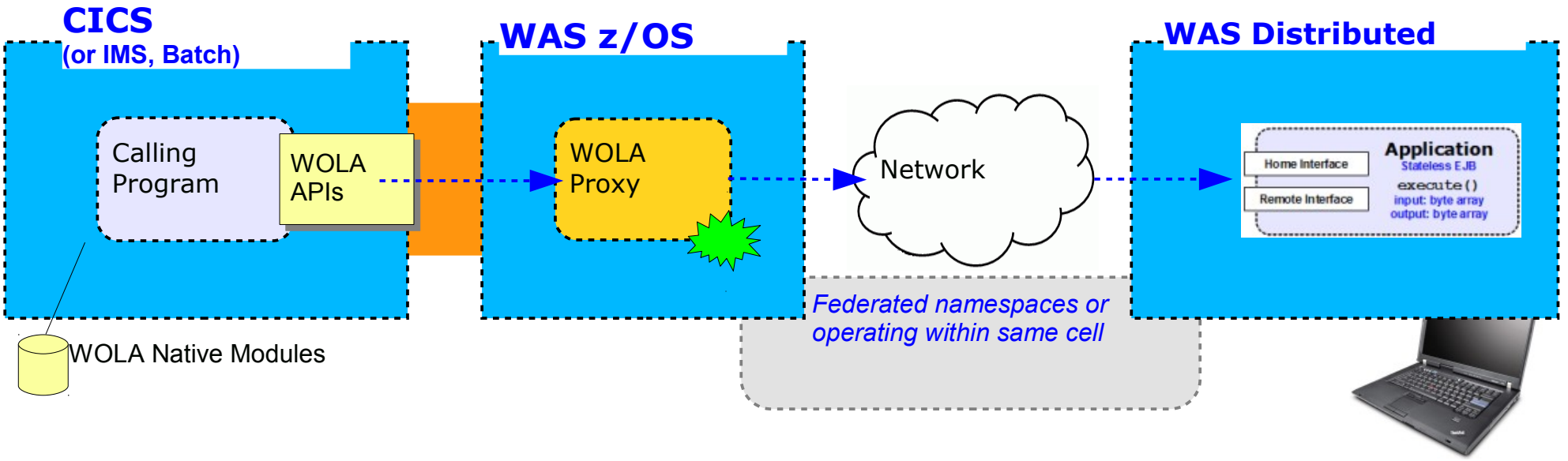
Java developer writes application to CCI in the WOLA JCA resource adapter just as if the application was deployed on WAS z/OS

Limitations:

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

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

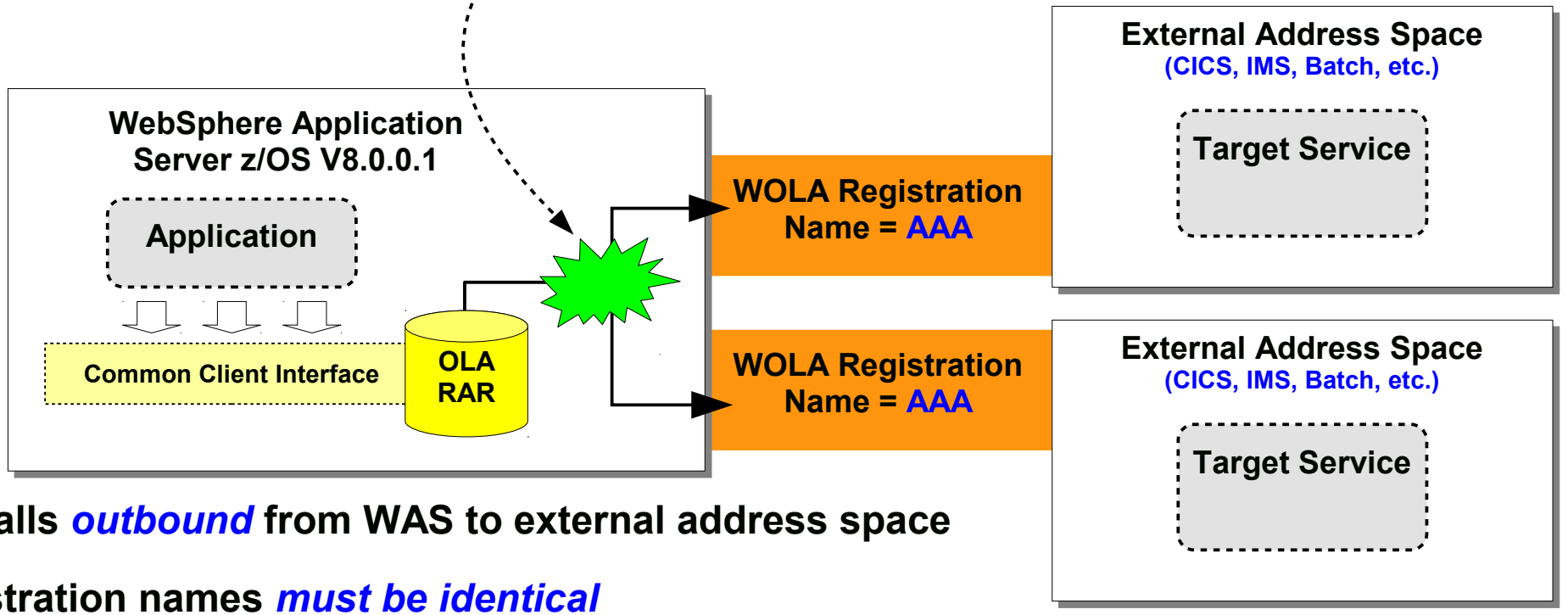
8.0.0.1 Release, Part 2

Work Distribution Enhancement

Environment Variable

`ola_locate_service_search_algorithm`

1	The last external address space to register in gets work
2	Round-robin across like-named registrations



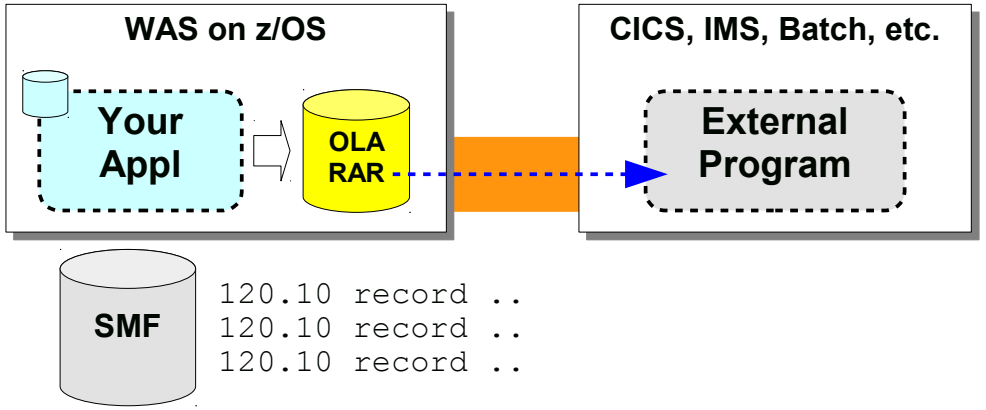
For calls *outbound* from WAS to external address space

Registration names *must be identical*

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

8.0.0.1 Release, Part 3

New SMF 120.10 Record, CICS Correlators and 64-bit APIs

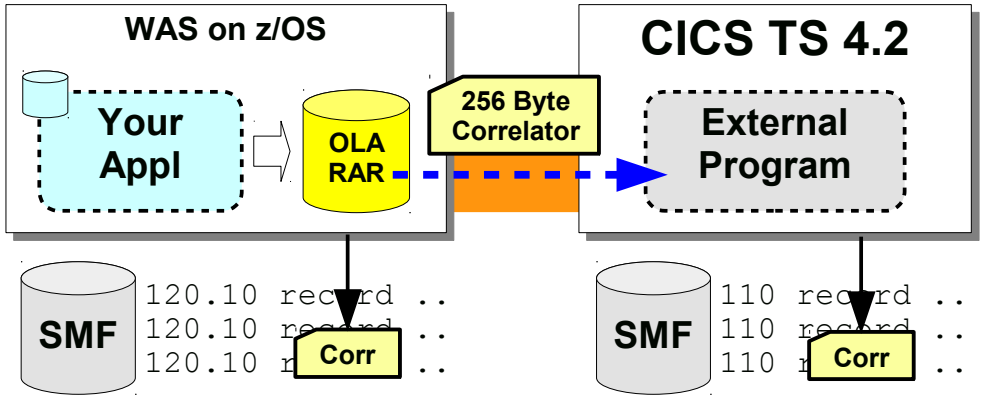


Similar to WAS z/OS 120.9 records

120.9 records inbound calls, the new 120.10 is used to record *outbound* calls

Good information about content and performance of outbound calls

InfoCenter: rtrb_SMFsubtype10

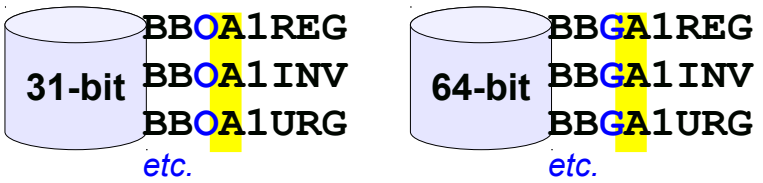


Part of the SMF 120.10 record function

256 bytes of specific information about the outbound request

With CICS 4.2 the correlator ends up in the CICS 110 records as well

InfoCenter: rtrb_SMFsubtype10



Callable by 64-bit address spaces running C or Assembler (Batch, USS)

API pointers 64 bits

InfoCenter: cdat_olaapis

#SHAREorg

SHARE
Technology · Connections · Results

Version 8 Hidden Gems

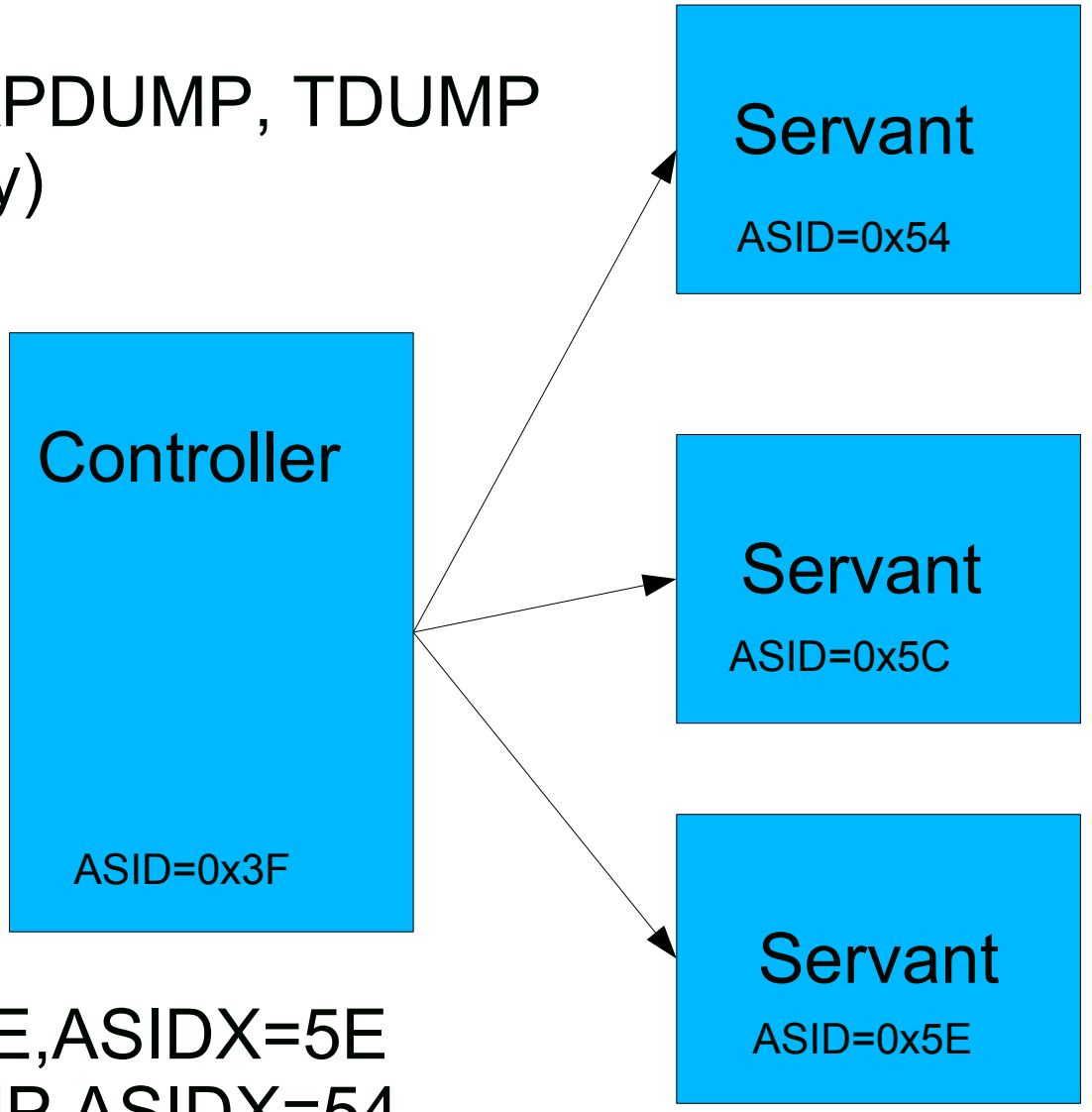
Routing for Modify Javacore etc.

Modify JAVACORE, HEAPDUMP, TDUMP
Dumps all regions (mostly)

Until V8!

Specify target ASID:

For example:

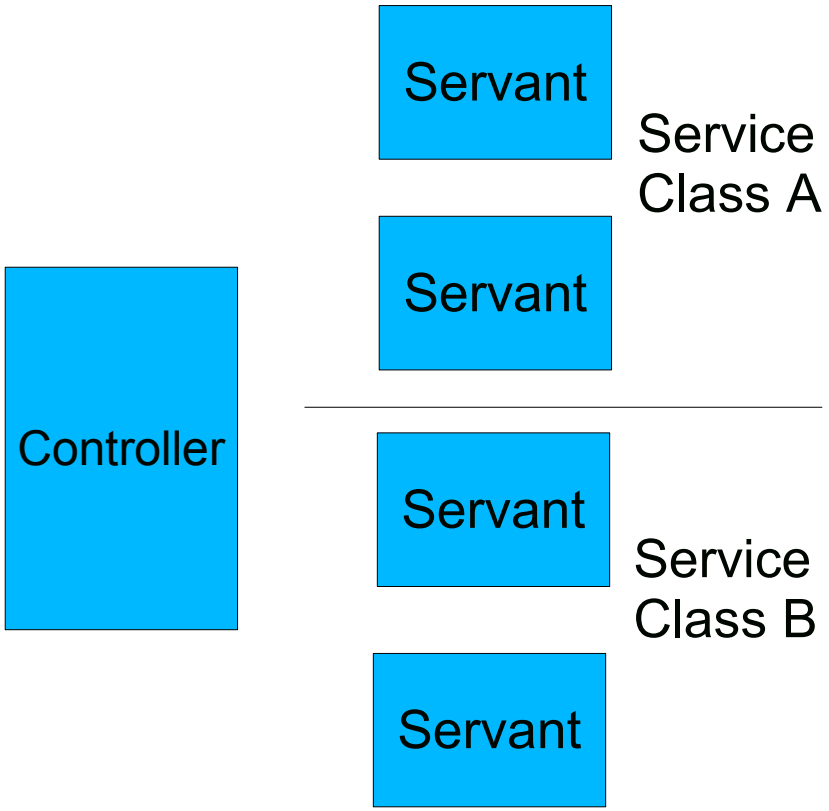


Modify server,JAVACORE,ASIDX=5E
Modify server,HEAPDUMP,ASIDX=54
Modify server,TDUMP,ASIDX=**3F**

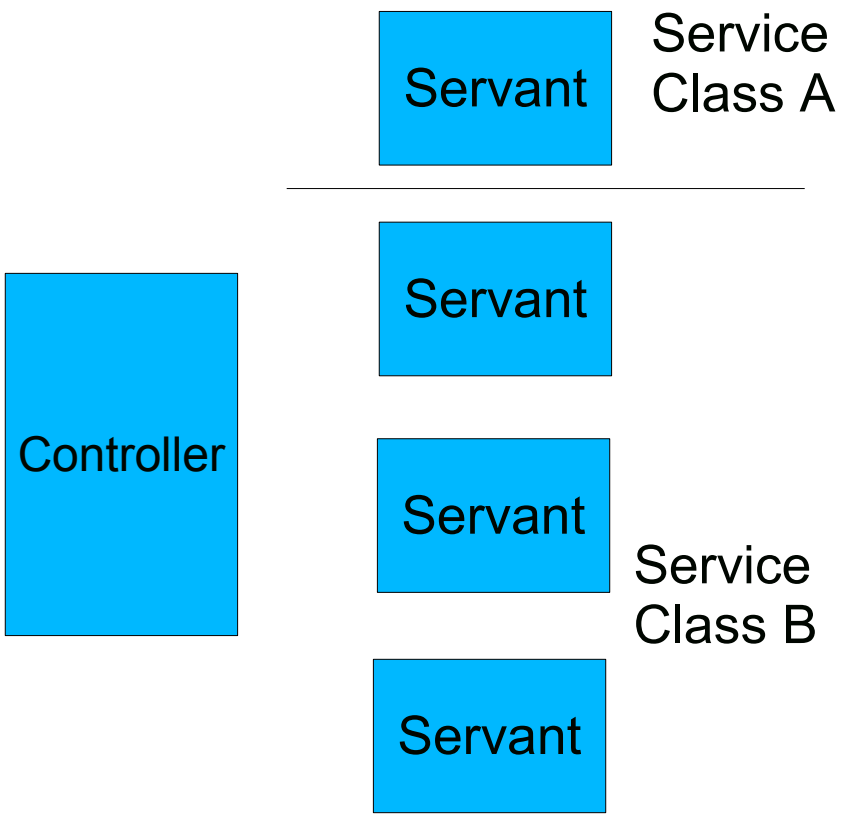
Configure WLM AE_SPREADMIN

Configure wlm_ae_spreadmin=0 or 1

- Sets AE_SPREADMIN(NO|YES) on IWM4SLI API call

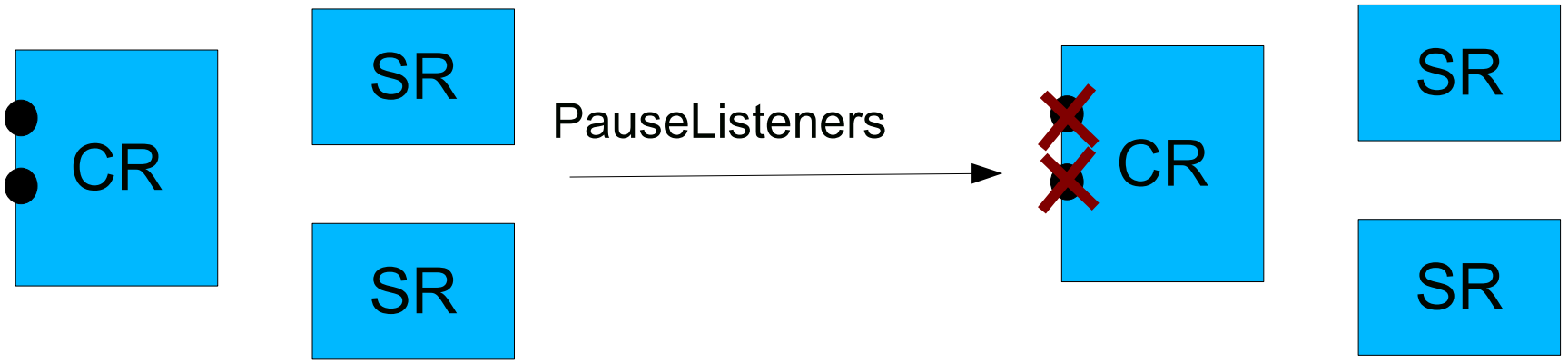


Spread 'Yes' balances equally



Spread 'No' WLM decides

Displaying 'Paused' state

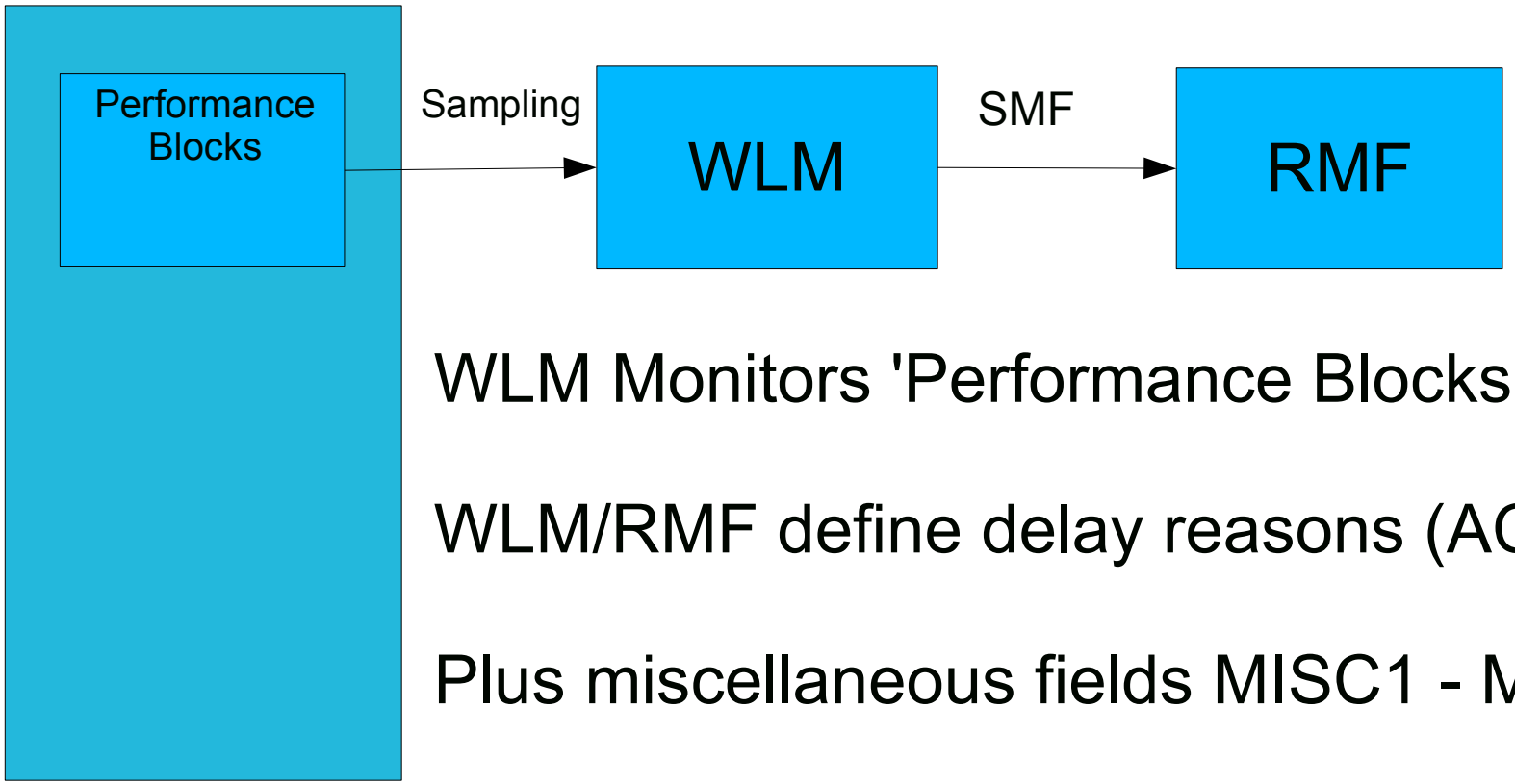


Modify PAUSELISTENERS closes ports etc.
But how do you know? In **Version 8!**

Modify server,DISPLAY,SERVERS output updated:

```
BBOO0182I SERVER            ASID SYSTEM LEVEL                STATE
BBOO0183I WAS00   /ZWASAXXX 6Fx  SY1    8.0.0.0 (ff1106.32) ACTIVE
BBOO0183I BBON001 /BBON001  58x  SY1    8.0.0.0 (ff1106.32) ACTIVE
BBOO0183I BBOC001 /BBOS001  5Bx  SY1    8.0.0.0 (ff1106.32) PAUSED/STOPPING
BBOO0183I BBODMGR /BBODMGR  57x  SY1    8.0.0.0 (ff1106.32) ACTIVE
```

Delay Monitoring Misc. States



WLM Monitors 'Performance Blocks' inside WAS

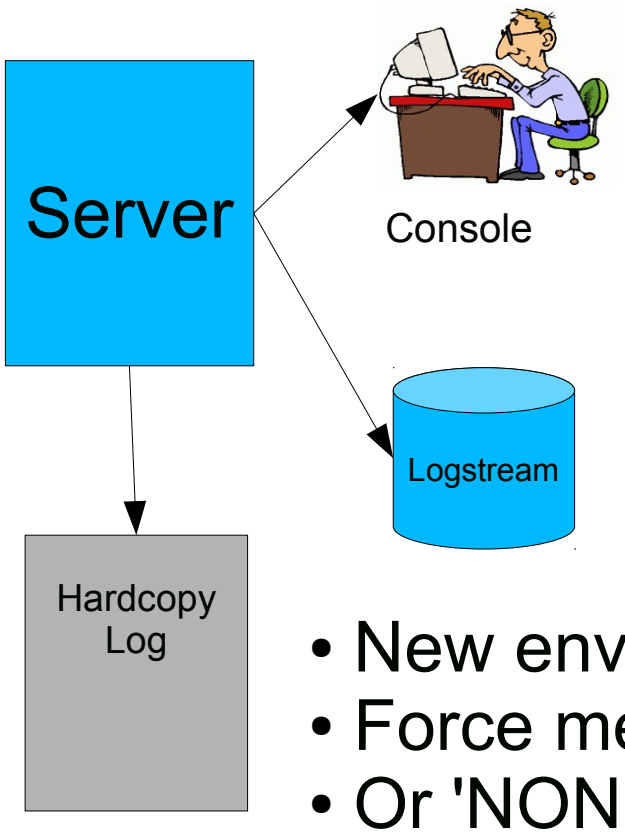
WLM/RMF define delay reasons (ACTIV_APPL)

Plus miscellaneous fields MISC1 - MISC15

WLM API IWM5MGDD allows WAS to 'explain' MISC fields

Reasons like 'RRS' will show up in RMF

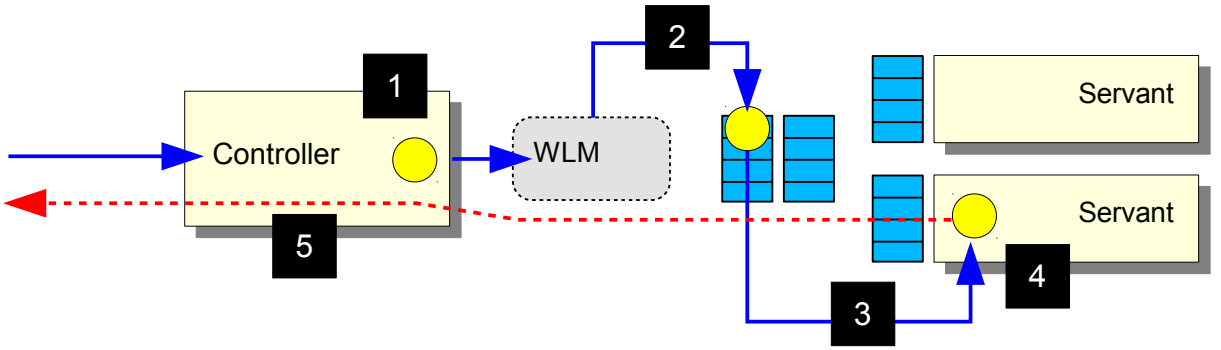
Message routing



- Messages are written as:
  - WTOs to the console
  - WTOs to the log
  - Writes to SYSOUT or Logstream
- The destination for a message is determined by the code that issues it

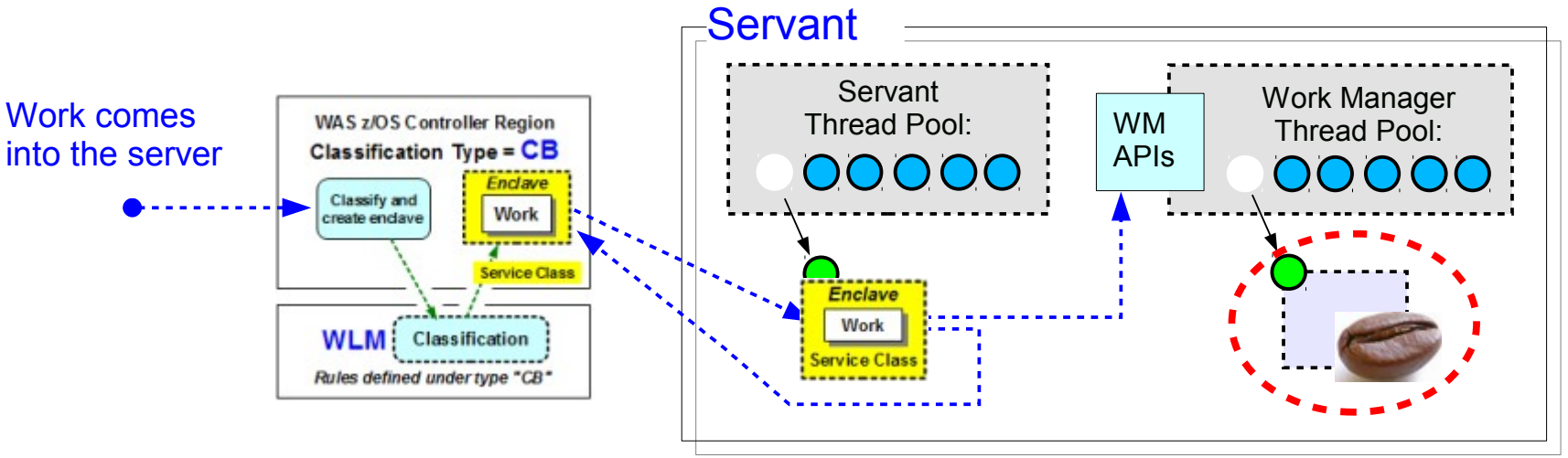
- New environment variables override the code
- Force messages (by ID) to a chosen target
- Or 'NONE' to suppress entirely
- Update dynamically with MODIFY
- Use DISPLAY to see current configuration

SMF 120-9 Updates for affinity routing



- Some requests establish an affinity to a servant region
- Later requests use that affinity and must run in the same servant
  - HTTPSession and Stateful Session Beans are examples
- The SMF 120.9 record already indicates if a request ran in a particular servant because of an affinity
- In **Version 8** we added an affinity token to the SMF record
- Find the request that created the affinity and all the later requests that used it

Async Work SMF Support in **8.0.0.1**



- Applications can use a Work Manager to schedule work to run asynchronously

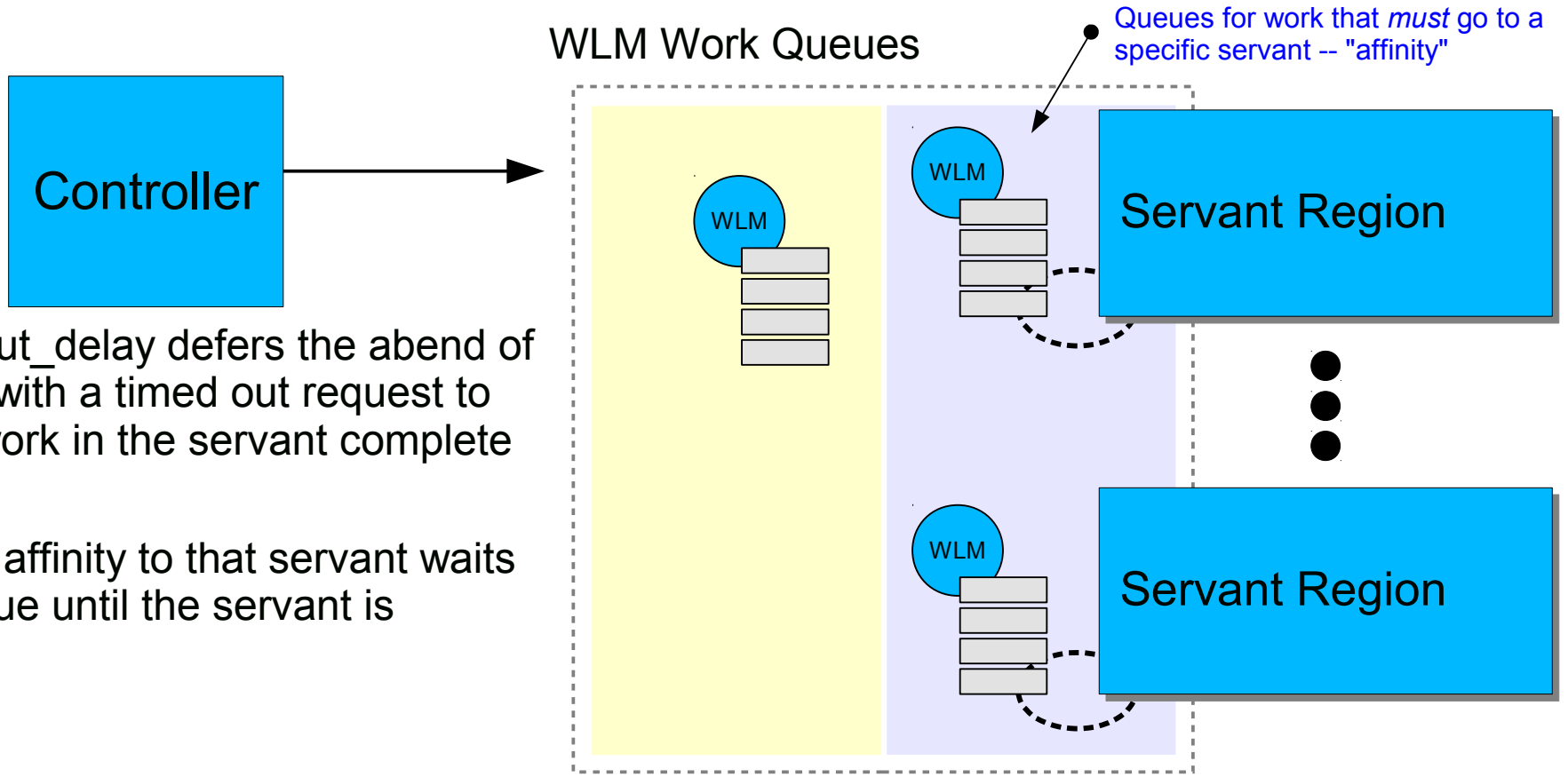
- If the same WLM enclave is used (depends) CPU time is captured in the SMF 120.9 (Type 72s and 30s will capture all enclave CPU time of course)

- No other information about the async work is captured at all

- In **Version 8** we modified the 120.9 record to also report async work

- Each piece of async work will produce its own 120.9
  - Information provided to correlate to the 'base' request'

Timeouts and Affinity Routing



The timeout_delay defers the abend of a servant with a timed out request to let other work in the servant complete

Work with affinity to that servant waits in the queue until the servant is abended

In **Version 8** we changed the behavior so work with affinity to a dying servant is rejected.
This frees up the client thread (maybe in IHS) to try again and run in another servant after the bad one dies

Thread Hang Recovery – improved diagnostics

Timeout processing in Version 7

1) Dispatch begins
2) Dispatch timer expires
3) WAS issues message BBOJ0113I
4) Try to interrupt the request
5) Repeat until we give up
6) Collect configured documentation (e.g. callstack)
7) Notify the controller
8) Controller begins process of abending the servant
9) Controller issues message BBOO0327I

Thread Hang Recovery – improved diagnostics

Timeout processing in **Version 8**

1) Dispatch begins
2) Dispatch timer expires
3) WAS issues message BBOJ0113I
4) **WAS issues message BBOJ0123I**
5) **If configured, gather pre-interrupt documentation**
6) **If configured issue message BBOJ0122I with ODI info**
7) Try to interrupt the request
8) Repeat until we give up
9) **Issue message BBOJ0124I to indicate we gave up**
10) Collect configured documentation (e.g. callstack)
11) Notify the controller
12) Controller begins process of abending the servant
13) Controller issues message BBOO0327I

What are all these new messages?

BBOJ0113I: The Interruptible Thread Infrastructure is attempting to advance work running under request ffff18b2

BBOJ0123I: The Interruptible Thread Infrastructure is attempting to advance work running under request ffff18b2,
request details: ThreadDetails: ASID = 0129, TCB = 0X006C62D8, Request = ffff18b2, Is JVM Blocked = false, Tried to interrupt = false, Given up = false, Internal Work Thread = false, Hung Reason = Not Hung, SR Dispatch Time = 2011/02/25 20:36:56.474373, CTL Receive Time = 2011/02/25 20:36:56.352540, CTL Queued to WLM Time = 2011/02/25 20:36:56.471058, Request Timeout limit = 63, Elapsed Execution Time = 65, CPU Time Used Limit = 3500000, Outbound Request Timeout Limit = 30, ODI Details = [JVM INTERRUPTIBLE THREAD, Monitor ACTIVE]

BBOJ0122I: The Interruptible Thread Infrastructure about to drive a ODI to advance work running under request ffff18b2, ODI details: Monitor ACTIVE

BBOJ0124I: The Interruptible Thread Infrastructure timed out a request and it has become unresponsive, request ffff18b2, request details: ThreadDetails: ASID = 0129, TCB = 0X006C62D8, Request = ffff18b2, Is JVM Blocked = false, Tried to interrupt = true, Given up = true, Internal Work Thread = false, Hung Reason = Dispatch Timer Popped, SR Dispatch Time = 2011/02/25 20:36:56.474373, CTL Receive Time = 2011/02/25 20:36:56.352540, CTL Queued to WLM Time = 2011/02/25 20:36:56.471058, Request Timeout limit = 63, Elapsed Execution Time = 65, CPU Time Used Limit = 3500000, Outbound Request Timeout Limit = 30, ODI Details = [JVM INTERRUPTIBLE THREAD, Monitor ACTIVE]

What are all these new messages?

```
BBOJ0117I: JAVA THREAD STACK TRACEBACK FOR THREAD WebSphere WLM Dispatch Thread t=006c62d8:
   Hung Thread Recovery--pre-interrupt
   Traceback for thread WebSphere WLM Dispatch Thread t=006c62d8:
   com.ibm.ejs.ras.CB390TraceEventListener.writeTrace(Native Method)
   com.ibm.ejs.ras.CB390TraceEventListener.processEvent(CB390TraceEventListener.java:390)
     .
     .
     .

BBOJ0117I: JAVA THREAD STACK TRACEBACK FOR THREAD WebSphere WLM Dispatch Thread t=006c62d8:
   Thread Hang Recovery--thread could not be encouraged to complete
   Traceback for thread WebSphere WLM Dispatch Thread t=006c62d8:
   com.ibm.ejs.ras.CB390TraceEventListener.writeTrace(Native Method)
   com.ibm.ejs.ras.CB390TraceEventListener.processEvent(CB390TraceEventListener.java:390)
     .
     .
     .
```

USS process Tag Data

USS has an API to specify 'tag' data for a process

WAS Version 8 calls this API in all regions

Visible with the 'ps' command or 'DISPLAY OMVS,PID='

Example strings:

```
WAS: WAS00/NDN1/BBOC001/BBOS001 - CTL-AS – HH110309
WAS: WAS00/WAS00/BBODMGR/BBODMGR - CTL-DM – HH110309
WAS: WAS00/NDN1/BBON001/BBON001 - CTL-NA – HH110309
WAS: WAS00/WAS00/BBODMGR/BBODMGR - SR – HH110309
WAS: WAS00/NDN1/WAS00/ZWASAXXX - DAEMON - HH110309
```

Form Feed Controls

- JES2 spins output based on SEGMENT
- SEGMENT defined by page count
- Pages delimited by Form Feeds

- WebSphere issues Form Feeds automatically
- By time
  - ras_stdout_ff_interval
  - ras_stderr_ff_interval
- Or by volume
  - ras_stdout_ff_line_interval
  - ras_stderr_ff_line_interval

- Or, in **Version 8**, on command:
  - MODIFY server,FORMFEED