IMS Technical Symposium

W&W Informatik GmbH

Using WOLA and OTMA for Efficient Communication between IMS/COBOL and WebSphere z/OS at W&W

Session B10

Daniel Schoeman – Senior Middleware Specialist

daniel.schoeman@ww-informatik.de

W&W Informatik GmbH, Germany

wüstenrot württembergische

Agenda

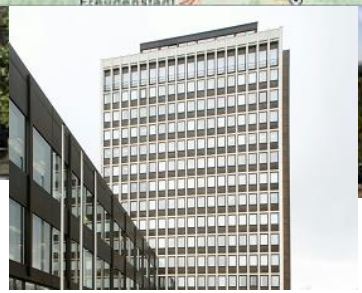
1. Wuestenrot & Wuerttembergische at a Glance
2. Starting Position
3. Solution: WOLA
4. Implementation: WOLA
5. W&W Use Cases: WOLA
6. Extra Infos
7. Summary and Outlook

Wuestenrot and Wuerttembergische

- Agenda
  1. Wuestenrot & Wuerttembergische at a Glance
  2. Starting Position
  3. Solution: WOLA
  4. Implementation: WOLA
  5. W&W Use Cases: WOLA
  6. Extra Infos
  7. Summary and Outlook

W&W at a Glance

- Pictures
  - Stuttgart
  - Karlsruhe
  - Ludwigsburg
- Short Description
  - WV – Insurance
    - Life-, Medical-, Car Insurance, etc
  - BSW – Home Loans and Banking
    - Home Loans
    - Building Society
    - Banking
    - Real Estate etc
  - Others
    - IT Services
    - Asset Management
    - etc



W&W in Numbers (2014)

Shareholders: **Wüstenrot Holding: 66,10%**
Landesbank Baden-Württemberg: 8,78%
UniCredito: 7,54%
L-Bank 4,99%
Schweizerische Rück: 4,67%
Freefloat: 7,92%

6 million customers

6.000 sales-partners
9.000 office staff

W&W AG
Strategic Management-Holding

Home loans, banking, investment funds

General/personal accident insurance, life assurance, health insurance

Home loan and savings bank division

- Home loans
- Banking
- Wüstenrot Bausparkasse AG
- Wüstenrot Bank AG Pfandbriefbank
- Real estate Wüstenrot Haus- und Städtebau GmbH

Insurance division

- Life and health insurance
- General/personal accident insur.
- Württembergische Lebens-versicherung AG
- Württembergische Krankenvers. AG
- Württembergische Versicherung AG

Foreign companies: Wüstenrot Vertrags-Holding Prag/Czek Republic

- Bausparkasse
- Hypothekenbank
- Lebens-versicherung
- Schaden- und Unfallversicherung

Service functions

- W&W Asset Mgmt. GmbH
- W&W Informatik GmbH
- W&W Produktion GmbH
- W&W Service GmbH

Key performance indicators of the W&W-group (2010, IFRS)

New home loan savings business (gross)	14,7 bn €
New home loans business	5,6 bn €
Earned premiums	3,7 bn €
Financial investments	
Net income	188,2 mn €
Equity	2,8 bn €

Württembergische: is among the TOP 10 life and general insurance companies

wüstenrot württembergische

W&W Mission (2014)

The W&W Group is *THE* expert in savings and investment,
home ownership, financial cover and risk protection –
throughout every stage of the customer's life.

Having united the forces of Wüstenrot and Württembergische, we've become a one-stop shop for comprehensive financial planning.

We leverage the combined power of our two equally strong pillars **Wüstenrot**, a home loan savings bank and **Württembergische**, an insurance provider, to respond to our clients needs like no other financial services provider in Germany.

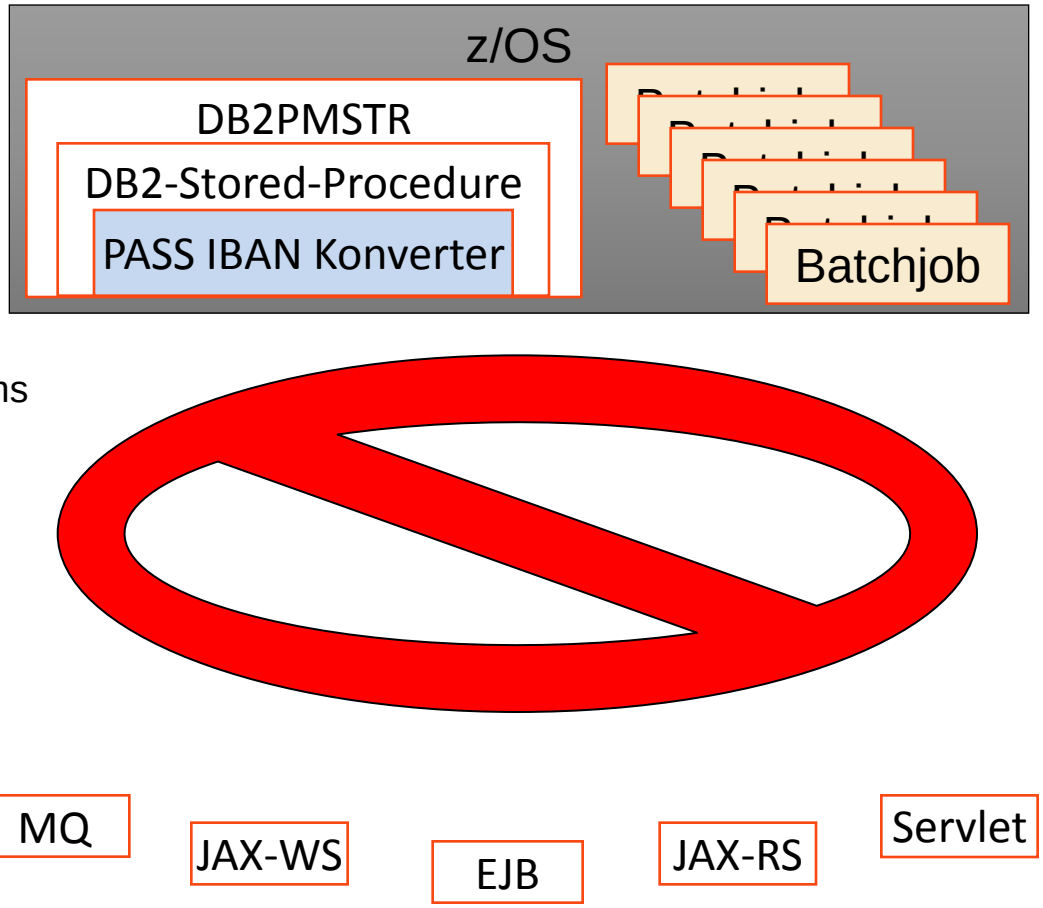
This integrated range of services enables Wüstenrot & Württembergische to offer all clients tailor-made solutions to meet their individual financial planning needs, encompassing savings and investment, home loan savings, financial cover and risk protection. Coupled with our sales channels comprising more than 6,000 W&W advisors, this makes us one of the strongest financial services providers in Germany.

Starting Position

- Agenda
  1. Wuestenrot & Wuerttembergische at a Glance
  2. Starting Positon
     1. Problem Description
        - Example: Java IBAN Converter using DB2 Stored Procedures
        - Problems: Runtime, Builds, Deployments, Management and Monitoring
     2. Problem Solution
        - WOLA with zWAS
  3. Solution: WOLA
  4. Implementation: WOLA
  5. W&W Use Cases: WOLA
  6. Extra infos
  7. Summary and Outlook

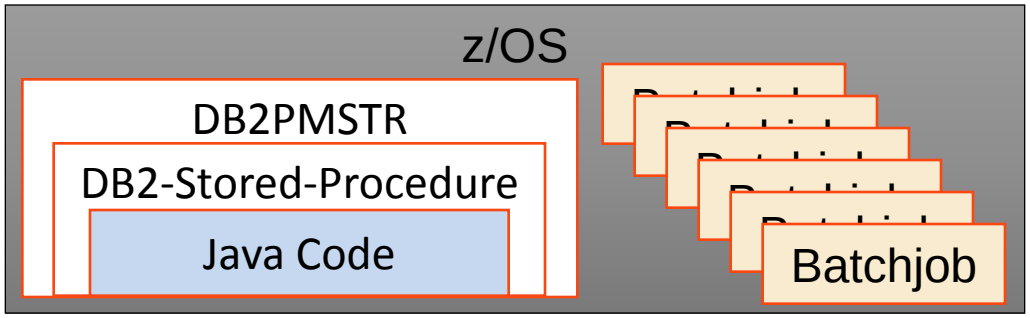
Problem Description

- Actual State: Access only via the DB2-Stored Procedure
  - DB2-Dependant
  - No Flexibility
  - Blackbox Runtime
  - No Automatic Builds
  - No Automatic Deployments
  - Inadequate Management
  - Limited Monitoring
  - No Access from Distributed Systems

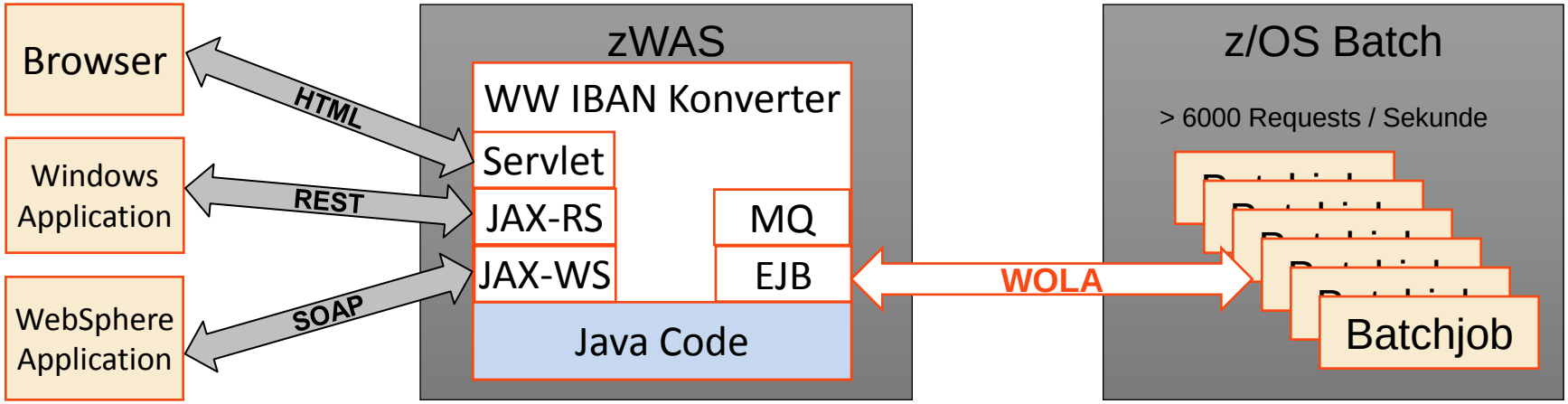


Problem Solution: zWAS and WOLA

- Actual State: Access only via the DB2-Stored Procedure
  - DB2-Dependant
  - No Flexibility
  - Blackbox Runtime
  - No Automatic Builds
  - No Automatic Deployments
  - Inadequate Management
  - Limited Monitoring
  - No Access from Distributed Systems



- Target State: Multiple Access Method (MainFrame and Distributed Systems)
  - http://te01.ww-intern.de:32247/IbanKonverter/IKGUI?kto=494949&blz=60050101&land

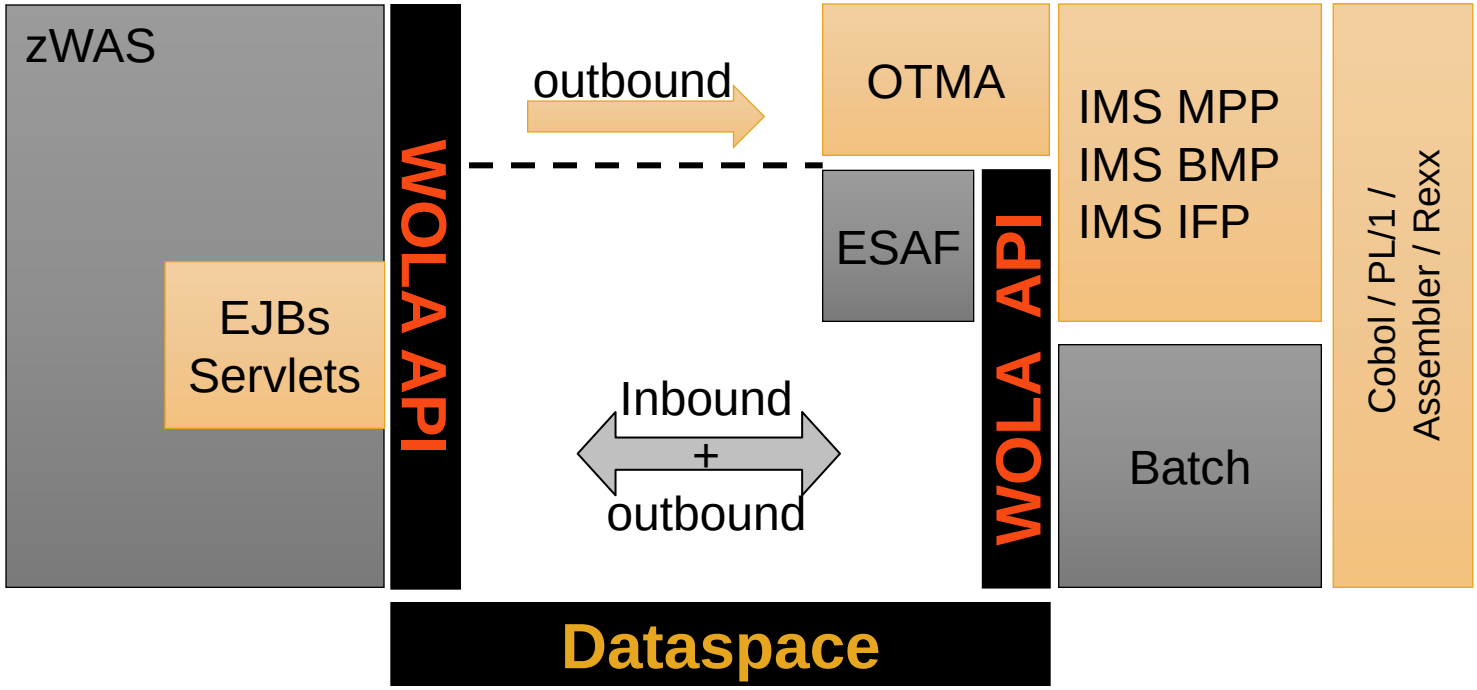


Solution: WOLA

- Agenda
  1. Wuestenrot & Wuerttembergische at a Glance
  2. Starting Positon
  3. Solution: WOLA
     1. WOLA Description
     2. WOLA Details
     3. WOLA Programming Model
     4. WOLA and IMS
     5. WOLA Use Cases
     6. zWAS Introduction
     7. WOLA Preparations on zOS
     8. WOLA Preparations in IMS
  4. Implementation: WOLA
  5. W&W Use Cases: WOLA
  6. Extra Infos
  7. Summary and Outlook

WOLA Description

- WebSphere Optimized Local Adapters
  - Allows very fast low-latency memory-to-memory data transfers between WAS z/OS and programs in other address spaces (CICS, IMS, Batch)
  - No TCP, SNA or other network latencies
  - Bi-directional byte array to and from WAS z/OS
  - Speed increased since it disregards data formatting, content type and code pages

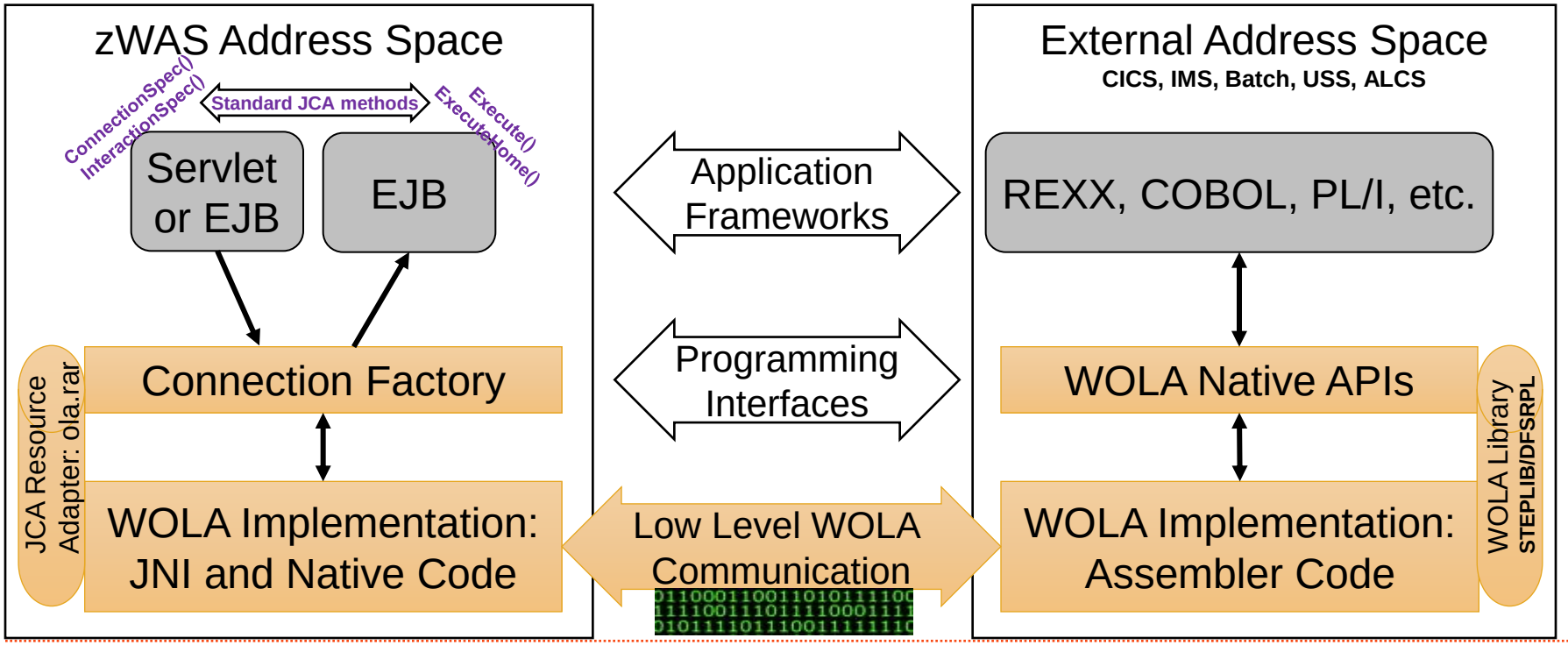


WOLA Details

- Bi-directional
  - Inbound: IMS, CICS and other batch programs call WAS z/OS
  - Outbound: WAS z/OS applications call IMS, CICS or other address spaces
- Transactional integrity using z/OS RRS
  - CICS: 2PC for CICS to WAS and WAS to CICS
  - IMS: 2PC Inbound IMS into WAS (ab WAS 8)
  - IMS: 2PC Outbound WAS into IMS over OTMA (ab WAS 8)
- WOLA is not a cross-LPAR technology or a cross-platform technology
  - Cross-platform can be handles by zWAS using TCP.
- IMS Transactions can be called directly when using OTMA
- WOLA Calls can be synchronous or asynchronous
- SMF-Record advantages

WOLA Programming Model 1/2

- Mainframe Software can communicate with Java Software

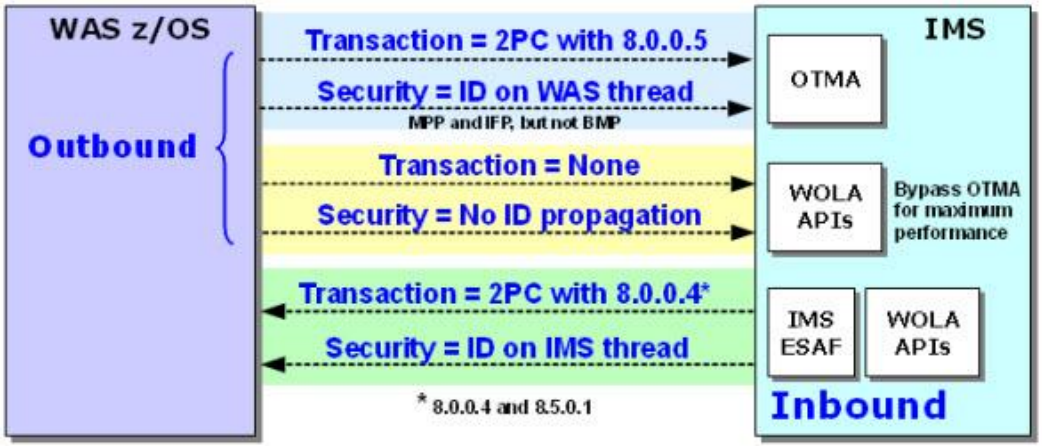


WOLA Programming Model 2/2

- Programming Model:
  - Inbound (External to WAS)
    - Sender: Programs using the WOLA APIs to invoke WAS
    - Receiver: Stateless EJB using supplied WOLA class libraries
  - Outbound (WAS to External)
    - Sender: Java programs using CCI (Common Client Interface) of the WOLA JCA resource adapter
    - Receiver: Programs using WOLA APIs to register for and receive data
      - CICS programs may use the WOLA Link Server to hide WOLA
      - IMS programs may use OTMA communication to hide WOLA

WOLA and IMS

- MPP, BMP, IFP, DL/I and non-IMS applications supported
- Inbound (zOS→zWAS) using WOLA APIs
  - IMS Programs use WOLA APIs to call zWAS applications
- Outbound (zWAS→IMS) using WOLA API‘S
  - IMS Modules can register as WOLA receivers
  - Faster than IMS over OTMA
- Outbound (zWAS→IMS) over OTMA:
  - No application changes required in IMS
  - No enablement of WOLA required in IMS
  - WAS z/OS uses the WOLA JCA resource adapter to speak to OTMA
  - 2PC (2 Phase Commit)
  - Security ID Propagation

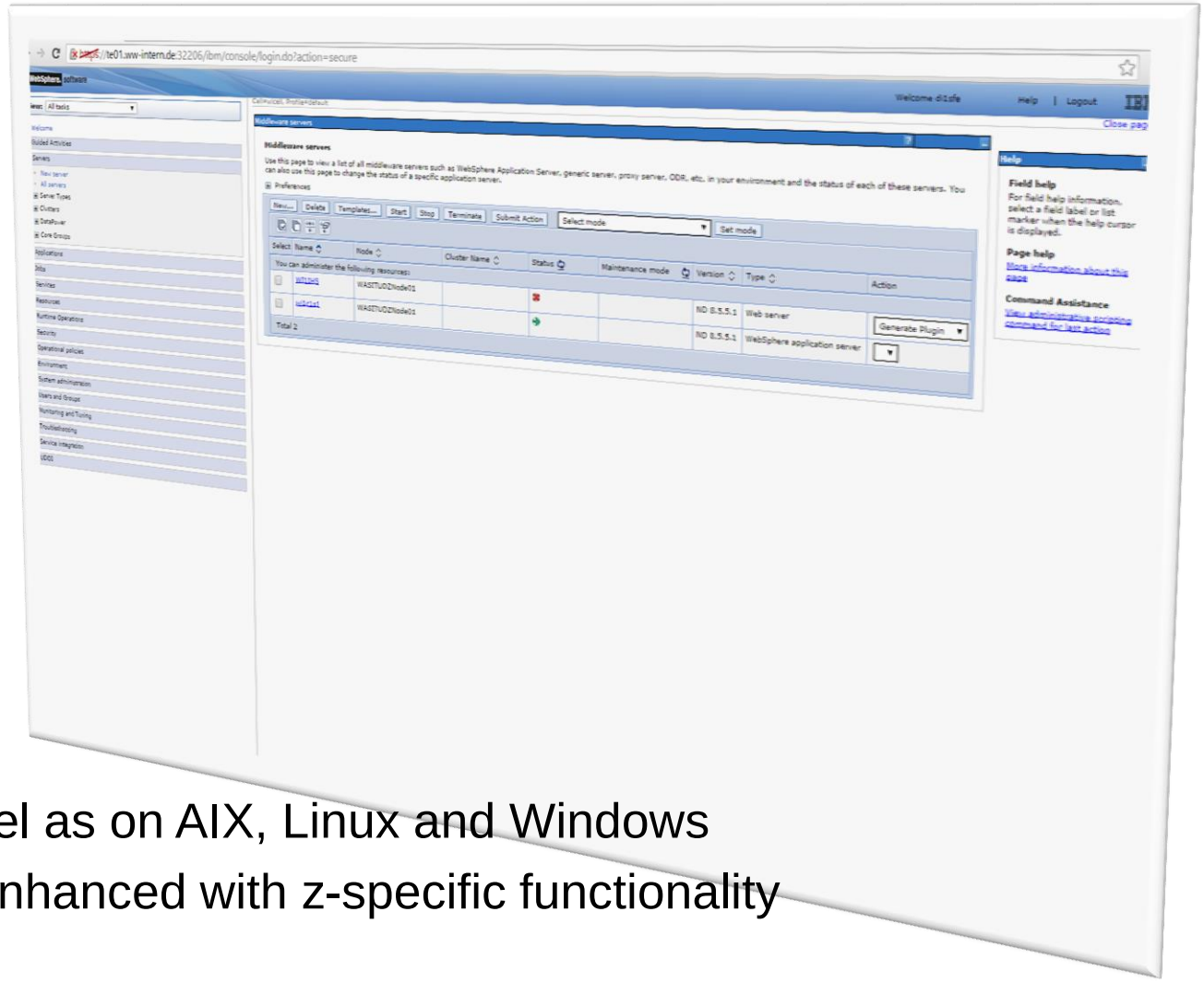


http://en.wikipedia.org/wiki/WebSphere_Optimized_Local_Adapters

WOLA Use Cases

- Centralized Java Applications
  - Central Java applications can be called from IMS, COBOL, EJBs, Servlets, WebServices etc
- Centralized Databases on zOS
  - Data access is not restricted to Mainframe only
  - Java applications on WAS z/OS open the Mainframe data to everyone
    - Either by direct DB2 calls via a WAS DataSource
    - Or by accessing IMS Transactions to access the data
- Workload optimization
  - CPU intensive batch processing can be offloaded to zAAP or zAAP-on-zIIP processors to relieve the normal processor
- COBOL / PL/I to Java
  - Shortage of COBOL and PL/I programmers can be compensated by programming new modules in Java
  - Vendors might also supply Java instead of COBOL Software

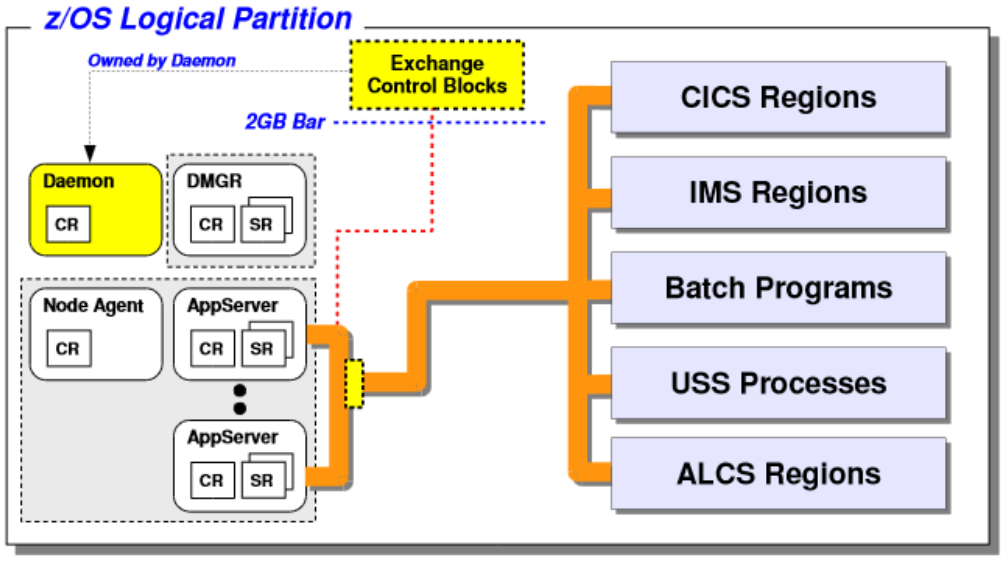
zWAS Introduction 1/2



- Same Look and Feel as on AIX, Linux and Windows
- Same Codebase, enhanced with z-specific functionality

zWAS Introduction 2/2

- The Control Region (CR) controls what work goes is and out
- The Servant Region (SR) does all the actuel work
  - it does not know about WOLA
- Using WOLA, the data is transferred between the external address space and the WAS CR
  - The CR passes the data addess to the Servant Region for processing



zWAS Installation

- SMP/E Installation (FMIDs and Datasets)
  - XXXX.WAS.V805.SMPE.CNTL
  - XXXX.WAS.V805.HBBO850.USER.CNTL (zWAS)
  - XXXX.WAS.V805.HBJA700.USER.CNTL (Java 7)
  - XXXX.WAS.V805.HGIN140.USER.CNTL (Installation Manager)
- Prepare Filesystem:
  - Create Filesystem: zCreateFileSystem.sh -name target.dsn -type HFS
  - Mount Filesystem:
    - MOUNT FILESYSTEM('..V805.SBBOHFS') MOUNTPOINT('/var/../zWAS/V8R5')
- Install the Binaries
  - Install zWAS, IHS, Plugins etc. using „imcl“ (InstallationManager CommandLine)
    - imcl install com.ibm.websphere.zOS.v85 <parameters>
- Prepare WebSphere Profiles
  - Use **zPMT Tool** to create responsefiles for the Profiles
  - Use **zpmt.sh** to create the Profile

WOLA Preparations on zWAS

- Prepare WOLA in zWebSphere
  - olaInstall.sh / copyzOS.sh
    - copies WOLA modules to a load library dataset
  - Create zWebSphere cell-level WOLA "enable adapter" environment variable
    - WAS_DAEMON_ONLY_enable_adapter=true
  - Create cell-level WOLA "identity propagate" environment variable
    - ola_cicsuser_identity_propagate=1
  - AppServer needs the WOLA-Modules (BBOA*)
  - WOLA JCA RAR file must be installed in zWAS
  - ConnectionFactories for zWebSphere connection to WOLA
    - Specify XCF Group Name for WOLA
    - Extra considerations for OTMA:
      - OTMAServerName() - XCF Server name for IMS control region
      - OTMAGroupID() - XCF Group name for IMS control region
      - OTMASyncLevel() - To set SyncOnReturn or SyncLevel1 (CM0 or CM1)

WOLA Preparations in IMS

- Enabling optimized local adapters support in IMS
  - Add a copy of the modules Data Set to:
    - APF list, IMS control region STEPLIB and DFSESL DDs
    - Message Processing Region (MPR), fast path Region (IFP) or Batch Message Processing (BMP) JCL DFSESL concatenation
  - Define the WOLA external subsystem to IMS
    - Include the following entry in the proclib member: WOLA,BBOA,BBOAIEMT
      - WOLA – Subsystem Name
      - BBOA – Language Interface Token
      - BBAOIEMT – External Subsystem Table Module
  - Pass the SSM parameter in your IMS startup data (SSM=WASZ)
    - IMS uses member <IMSID>WASZ in its PROCLIB to activate external subsystems
  - Define the CBIND class to RACF for the target WebSphere server

https://www-01.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.nd.multiplatform.doc/ae/tdat_enableconnectorims.html

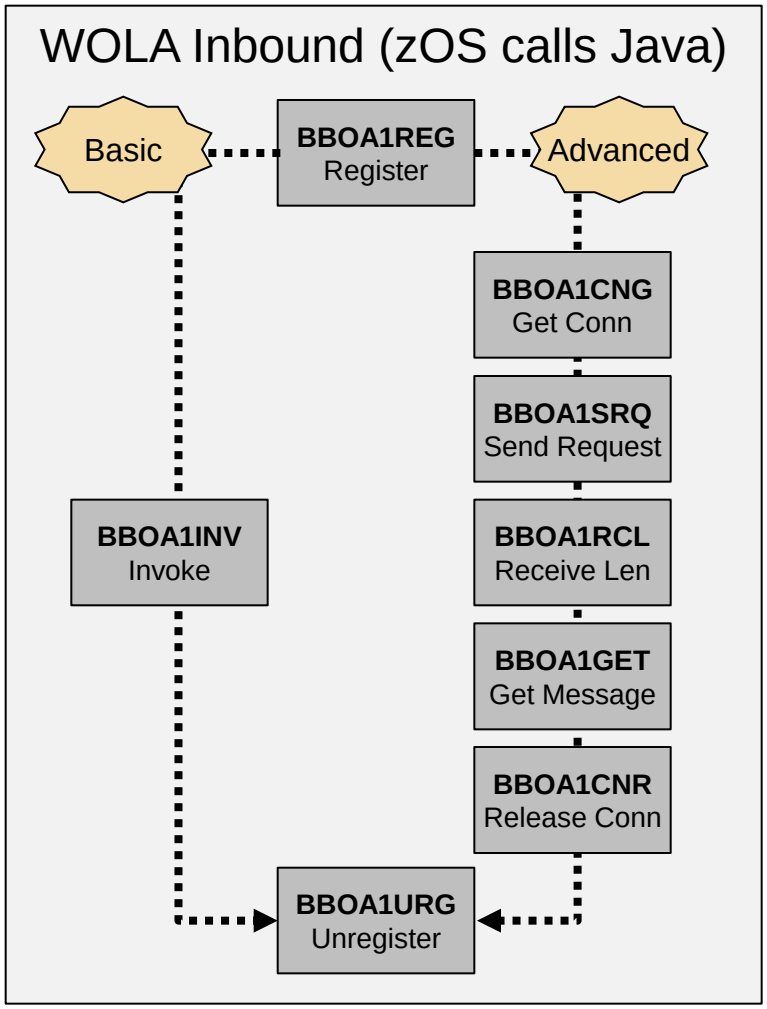
Implementation: WOLA

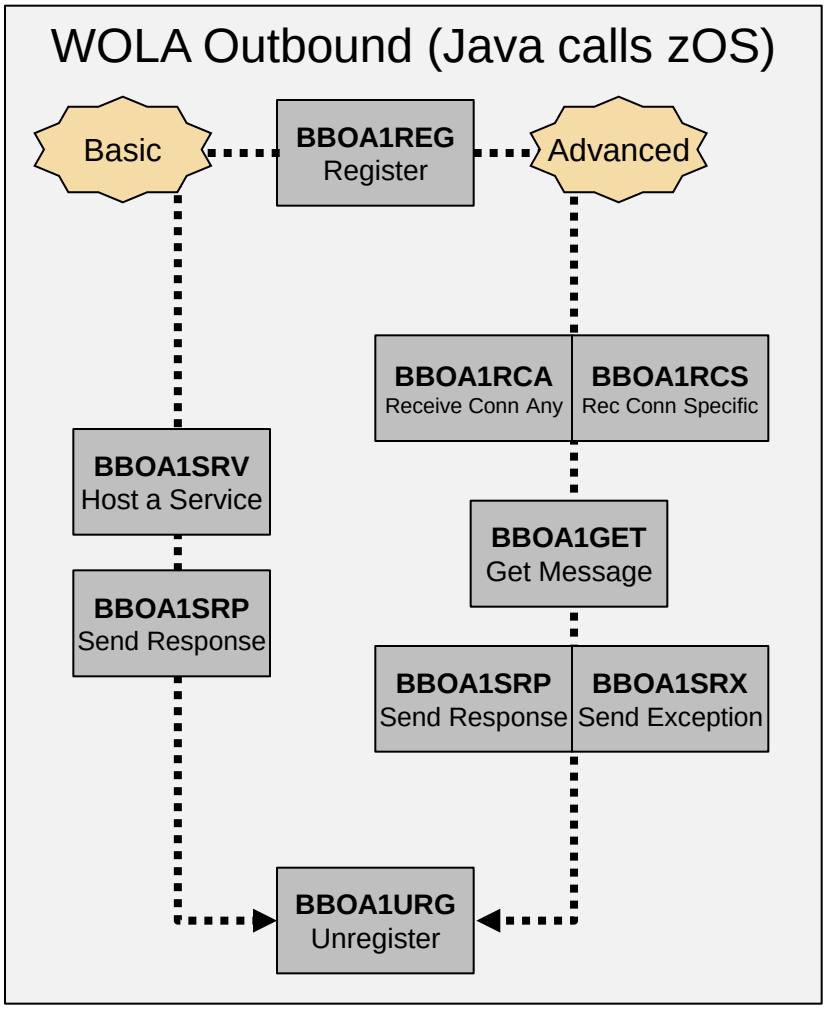
- Agenda
  1. Wuestenrot & Wuerttembergische at a Glance
  2. Starting Positon
  3. Solution: WOLA
  4. Implementation: WOLA
     1. WWI z-Environment
     2. WOLA Inbound and Outboud APIs summarized
     3. WOLA Example 1 – COBOL calls an EJB
     4. WOLA Example 2 – Java calls COBOL
     5. WOLA Example 3 – Java calls IMS via OTMA
  5. W&W Use Cases: WOLA
  6. Challenges
  7. Extra infos
  8. Summary and Outlook

WWI z-Environment

- WWI z-Envorinment
  - zWebSphere 8.5.5.3 Network Deployment
  - zEC12 - 5 CPs, 4 zIIPs
  - IMS 13
  - z/OS 1.13
  - Unix System Services ZFS Filesystem Cluster
  - WOLA
    - J2C-Adapter - ola.rar
    - Java APIs – ola-apis.jar

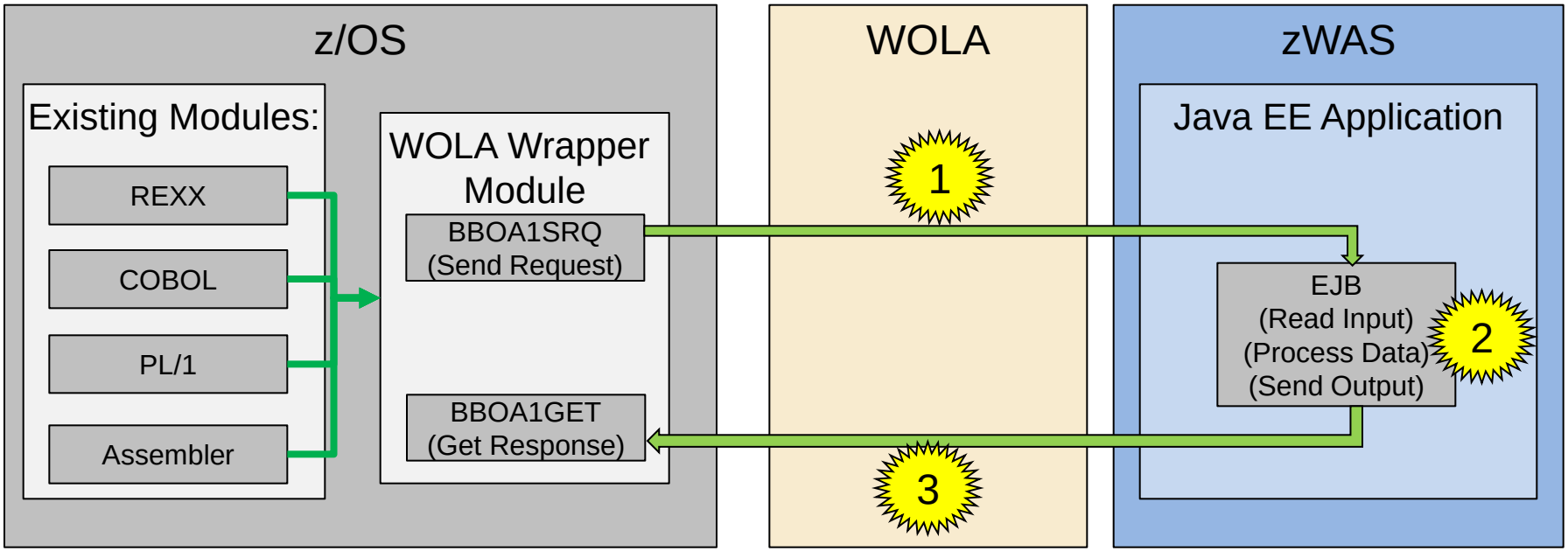
WOLA Inbound and Outbound APIs summarized for zOS





Source: WP101490

WOLA Example 1 – COBOL calls an EJB 1/3



WOLA Example 1 – COBOL calls an EJB 2/3

- Logical flow for COBOL calling EJB
  - Prepare Connection
    - BBOA1REG – Register Service
    - BBOA1CNG – Get Conection
  - Application Logic (Send and Receive Code)
    - BBOA1SRQ → Call WOLA Enabled Java Program
    - BBOA1RCL ← Receive length
    - BBOA1GET ← Get Data from WOLA Enabled Java Program
  - Close Connection
    - BBOA1CNR – Connection Reset
    - BBOA1URG – Unregister Service

```
* COBOL CopyBook for WOLA
 01 daemongroup                 PIC X(8) VALUE LOW-VALUES.
 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.
 01 maxconn                     PIC 9(8) COMP VALUE 10.
 01 regopts                     PIC 9(8) COMP VALUE 0.
 01 urgopts                     PIC 9(8) COMP VALUE 0.
 01 service-name                PIC X(255).
 01 service-name-length         PIC 9(8) COMP.
 01 rqst-area.
    02 rqst-area                PIC X(100) VALUE SPACES.
 01 rqst-area-addr              USAGE POINTER.
 01 rqst-area-length            PIC 9(8) COMP VALUE 100.
 01 resp-area.
    02 resp-area                PIC X(100) VALUE SPACES.
 01 resp-area-addr              USAGE POINTER.
 01 resp-area-length            PIC 9(8) COMP VALUE 100.
 01 connect-handle              PIC X(12) VALUE LOW-VALUES.
 01 rqst-type                   PIC 9(8) COMP VALUE 1.
 01 SRQasync                    PIC 9(8) COMP VALUE 0.
 01 RCLasync                    PIC 9(8) COMP VALUE 0.
 01 rc                          PIC 9(8) COMP VALUE 0.
 01 rsn                         PIC 9(8) COMP VALUE 0.
 01 rv                          PIC 9(8) COMP VALUE 0.
```

Connection Code

```
* Initialize Variables
MOVE 'WASITUDZ    ' TO registername.
MOVE 'WICELL      '    TO daemongroup.
MOVE 'WI1         '    TO node-name.
MOVE 'WI1C1S1     '    TO server-name.

* Register Service
CALL 'BBOA1REG' USING
      daemongroup,
      node-name,
      server-name,
      registername,
      ........., rc, rsn.

* Get Connection
 CALL 'BBOA1CNG' USING
       register-name,
       connect-handle,
       ........., rc, rsn.
```

```
* Connection Release
CALL 'BBOA1CNR' USING
      connect-handle,
      rc,
      rsn.
* Service Unregister
CALL 'BBOA1URG' USING
    registername,
    urgopts,
    rc,
    rsn.
```

Send/Receive Code

```
* Init WOLA Service to Call
MOVE 'ejb/WOLA' TO service-name.
MOVE 'Wola Msg' TO rqst-area
* Send Request
CALL 'BBOA1SRQ' USING
      connect-handle,
      rqst-type,
      service-name,
      service-name-length,
      rqst-area-addr,
      rqst-area-length,
      SRQasync,
      resp-area-length,
      rc, rsn.
```

```
* Receive length
CALL 'BBOA1RCL' USING
      connect-handle,
      RCLasync,
      resp-area-length,
      rc, rsn.
```

```
* Get Message
CALL 'BBOA1GET' USING
      connect-handle,
      resp-area-addr,
      resp-area-length,
      rc, rsn, rv
```

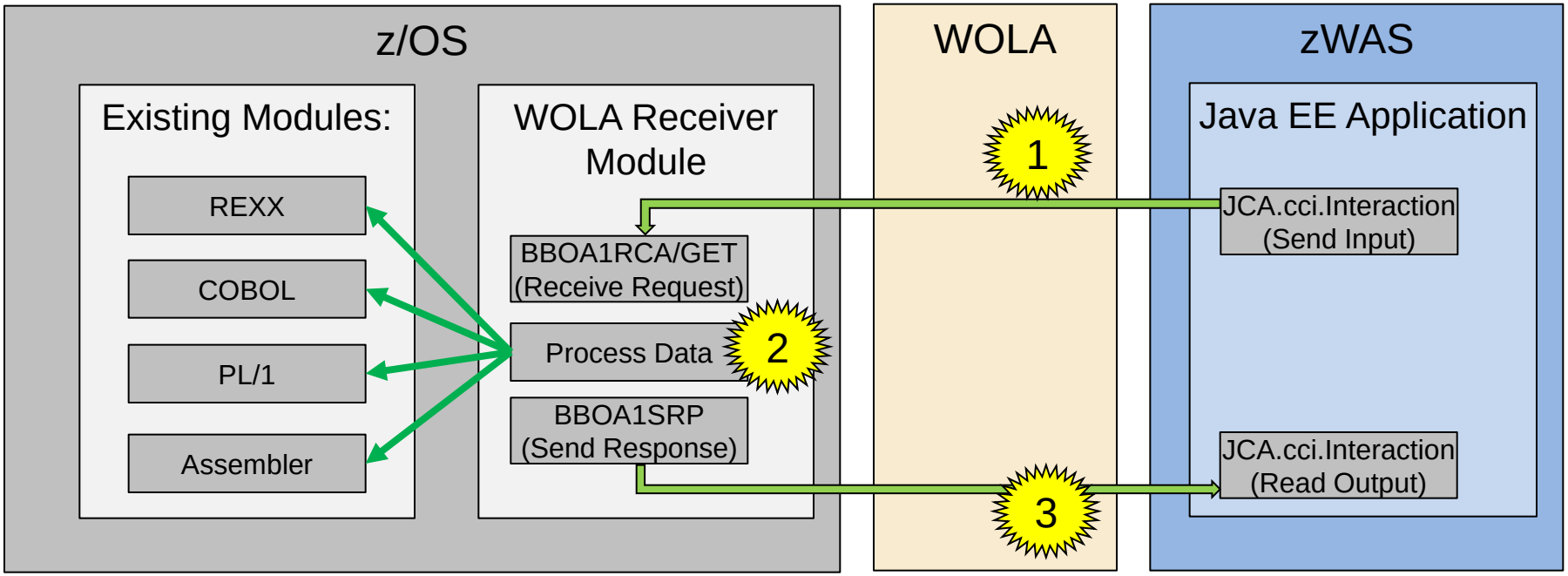
WOLA Example 1 – COBOL calls an EJB 3/3

- Logical flow for JAVA EJB receiving Data
  - Connection Code:
    - None: Handled by JEE
    - Only a simple Annotation needed
  - Application Code
    - Do Codepage Conversion
    - Process Data
    - Do Codepage Conversion
    - Return Data

Service Name in COBOL Module

```
@Stateless(mappedName = "ejb/WOLA")
@RemoteHome(com.ibm.websphere.ola.ExecuteHome.class)
public class WOLATEST2 {

  public WOLATEST2() {
  }

  public byte[] execute(byte[] inbytes)
  {
    try {
      String input = new String(inbytes, "cp1047");
      String output = processInput(input);

      return output.getBytes("cp1047");

    } catch (Exception E) {
      return "ERROR".getBytes();
    }
  }

   private String processInput(String inputdate) {
      // New Java Code to process data here
      // - JDBC
      // - SOAP
      // - Custom Java Code
      // - 3rd Party Java Software
   }

}
```

WOLA Example 2 – Java calls COBOL 1/3



WOLA Example 2 – Java calls COBOL 2/3

- Logical flow for JAVA Code sending request data and receiving response data
  - Connect to WOLA
    - Lookup for OLA-Connection from the connection Pool
    - Create Interaction Spec as per J2C-CCI (ToDo)
    - Create Interaction as per J2C-CCI (ToDo)
  - Send and Receive to and from WOLA
    - Create Record for input data
    - Execute request
    - Process Record with output data

```
protected void service(HttpServletRequest request, HttpServletResponse response) {

      /* Get Initial Context */
      InitialContext ctx = new InitialContext();

      /* Obtain connection factory by doing an indirect JNDI lookup */
      ConnectionFactory connectionFactory = (ConnectionFactory)ctx.lookup("eis/ola2");
      Connection connection = connectionFactory.getConnection();

      /* Create interactionSepcImpl and set the OLA service name */
      InteractionSpecImpl interactionSpecImpl = new InteractionSpecImpl();
      interactionSpecImpl.setServiceName("WOLA003S");

      /* Create an interaction */
      Interaction interaction = connection.createInteraction();

      /* Create an IndexRecordImpl instance and add the OLA message */
      IndexedRecordImpl indexRecordImpl = new IndexedRecordImpl();
      indexRecordImpl.add(„Hello World“.getBytes("Cp1047"));

      /* Invoke the OLA interface */
      Record outputRecord = interaction.execute(interactionSpecImpl ,indexRecordImpl);

      /* Read Output from OLA interface */
      byte[] output = (byte[])(((IndexedRecordImpl)outputRecord).get(0));
      String outputstr = new String(output,"Cp1047");

}
```

Service Name in COBOL Module

Connect the outside world to z/OS applications very easily

WOLA Example 2 – Java calls COBOL 3/3

- Logical flow for COBOL receiving Data from Java
  - Prepare Connection
    - BBOA1REG – Register Service
  - Application Logic
    - BBOA1RCA ← Request Any
    - BBOA1GET ← Get Data from Java
    - Process Data
    - BBOA1SRP → Send Data back to Java
  - Close Connection
    - BBOA1CNR – Connection Release
    - BBOA1URG – Unregister Service

```
* COBOL CopyBook for WOLA
 01 daemongroup                 PIC X(8) VALUE LOW-VALUES.
 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.
 01 maxconn                     PIC 9(8) COMP VALUE 10.
 01 regopts                     PIC 9(8) COMP VALUE 0.
 01 urgopts                     PIC 9(8) COMP VALUE 0.
 01 service-name                PIC X(255).
 01 service-name-length         PIC 9(8) COMP.
 01 rqst-area.
    02 rqst-area                PIC X(100) VALUE SPACES.
 01 rqst-area-addr              USAGE POINTER.
 01 rqst-area-length            PIC 9(8) COMP VALUE 100.
 01 resp-area.
    02 resp-area                PIC X(100) VALUE SPACES.
 01 resp-area-addr              USAGE POINTER.
 01 resp-area-length            PIC 9(8) COMP VALUE 100.
 01 connect-handle              PIC X(12) VALUE LOW-VALUES.
 01 rqst-type                   PIC 9(8) COMP VALUE 1.
 01 SRQasync                    PIC 9(8) COMP VALUE 0.
 01 RCLasync                    PIC 9(8) COMP VALUE 0.
 01 rc                          PIC 9(8) COMP VALUE 0.
 01 rsn                         PIC 9(8) COMP VALUE 0.
 01 rv                          PIC 9(8) COMP VALUE 0.
```

Connection Code

```
* Initialize Variables
MOVE 'WASITUDZ    '     TO registername.
MOVE 'WICELL  '         TO daemongroup.
MOVE 'WI1     '         TO node-name.
MOVE 'WI1C1S1 '         TO server-name.

* Register Service
CALL 'BBOA1REG' USING
      daemongroup,
      node-name,
      server-name,
      registername,
      ........., rc, rsn.
```

Request/Response Code

```
* Init WOLA Service to Call
MOVE 'WOLA003S' TO service-name.
* Receive Any
CALL 'BBOA1RCA' USING
      register-name,
      connect-handle,
      service-name,
      service-name-length,
      rqst-area-length,
      wait-time,
      rc, rsn.
```

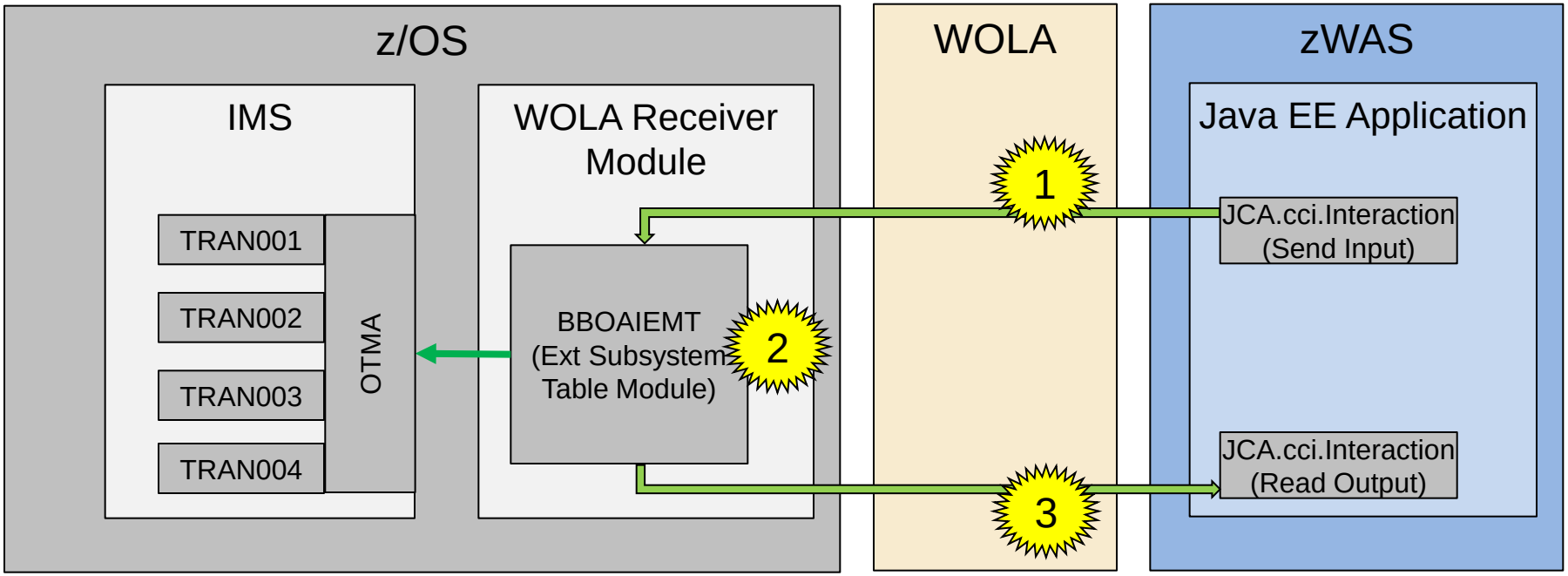
```
* Get Request Message
CALL 'BBOA1GET' USING
      connect-handle,
      resp-area-addr,
      resp-area-length,
      rc, rsn, rv.
```

```
Processing
```

```
* Send Response Message
CALL 'BBOA1SRP' USING
      connect-handle,
      resp-area-addr,
      resp-area-length,
      rc, rsn.
```

```
* Connection Release
CALL 'BBOA1CNR' USING
      connect-handle,
      rc,
      rsn.
* Service Unregister
CALL 'BBOA1URG' USING
    registername,
    urgopts,
    rc,
    rsn.
```

WOLA Example 3 – Java calls IMS over OTMA 1/3



WOLA Example 3 – Java calls IMS over OTMA 2/3

- Logical flow for JAVA Code sending request data and receiving response data
  - Connect to WOLA
    - Lookup for OLA-Connection from the connection Pool
    - Create Interaction Spec as per J2C-CCI (ToDo)
    - Create Interaction as per J2C-CCI (ToDo)
  - Send and Receive to and from WOLA
    - Create Record for input data
    - Execute request
    - Process Record with output data
  - Extra considerations for OTMA:
    - OTMAServerName
    - OTMAGroupID
    - OTMASyncLevel

```
protected void service(HttpServletRequest request, HttpServletResponse response) {

      /* Get Initial Context */
      InitialContext ctx = new InitialContext();

      /* Obtain connection factory by doing an indirect JNDI lookup */
      ConnectionFactory connectionFactory = (ConnectionFactory)ctx.lookup("eis/olaOTMA");
      Connection connection = connectionFactory.getConnection();

      /* Create interactionSepcImpl and set the OLA service name */
      InteractionSpecImpl interactionSpecImpl = new InteractionSpecImpl();
      //interactionSpecImpl.setServiceName("WOLA003S");

      /* Create an interaction */
      Interaction interaction = connection.createInteraction();

      /* Create an IndexRecordImpl instance and add the OLA message */
      IndexedRecordImpl indexRecordImpl = new IndexedRecordImpl();
      indexRecordImpl.add(„LLZZTRAN1001Hello World“.getBytes("Cp1047"));

      /* Invoke the OLA interface */
      Record outputRecord = interaction.execute(interactionSpecImpl ,indexRecordImpl);

      /* Read Output from OLA interface */
      byte[] output = (byte[])(((IndexedRecordImpl)outputRecord).get(0));
      String outputstr = new String(output,"Cp1047");

}
```

No need for Service Name

Connect the outside world to z/OS applications very easily

WOLA Example 3 – Java calls IMS over OTMA 3/3

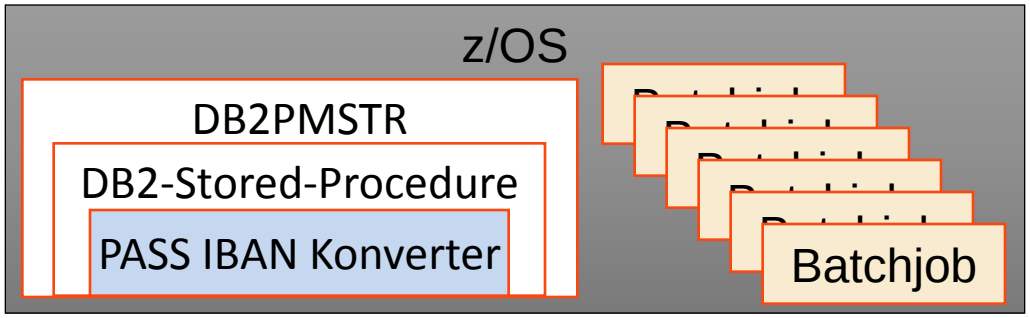
- Logical flow for COBOL receiving Data from Java over OTMA
  - No Changes to IMS Transactions

W&W Use Cases: WOLA

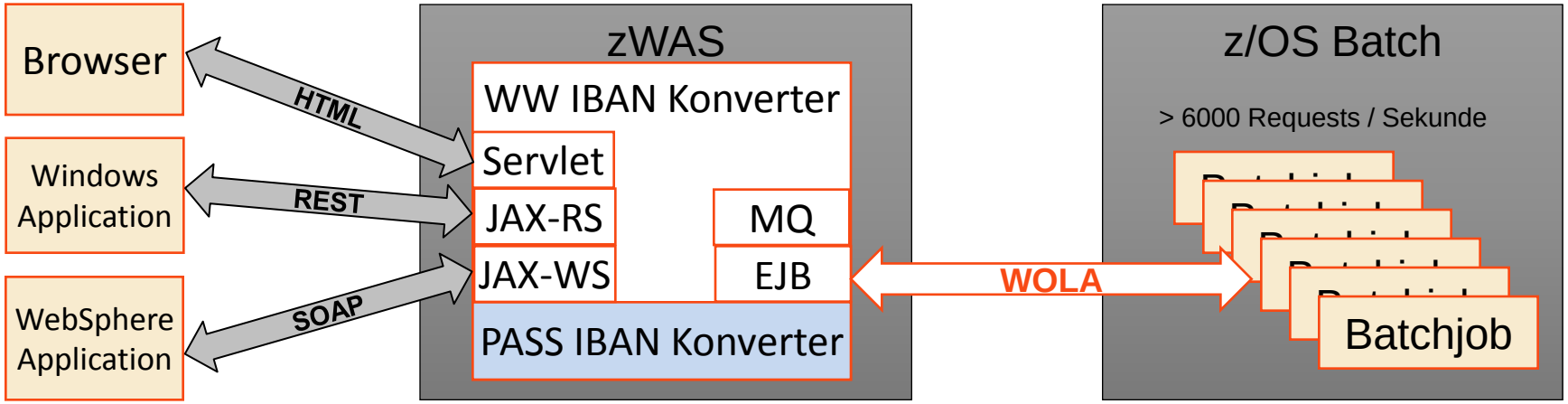
- Agenda
  1. Wuestenrot & Wuerttembergische at a Glance
  2. Starting Position
  3. Solution: WOLA
  4. Implementation: WOLA
  5. W&W Use Cases: WOLA
     1. Scenario 1 – IBAN Converter using WOLA (zOS → Java)
     2. Scenario 2 – WAA-Adapter using WOLA (Java → zOS → Java)
     3. Scenario 3 – KAP Client Lookup using WOLA (Java → zOS)
  6. Extra Infos
  7. Summary and Outlook

Scenario 1 – IBAN Converter using WOLA (Java Software)

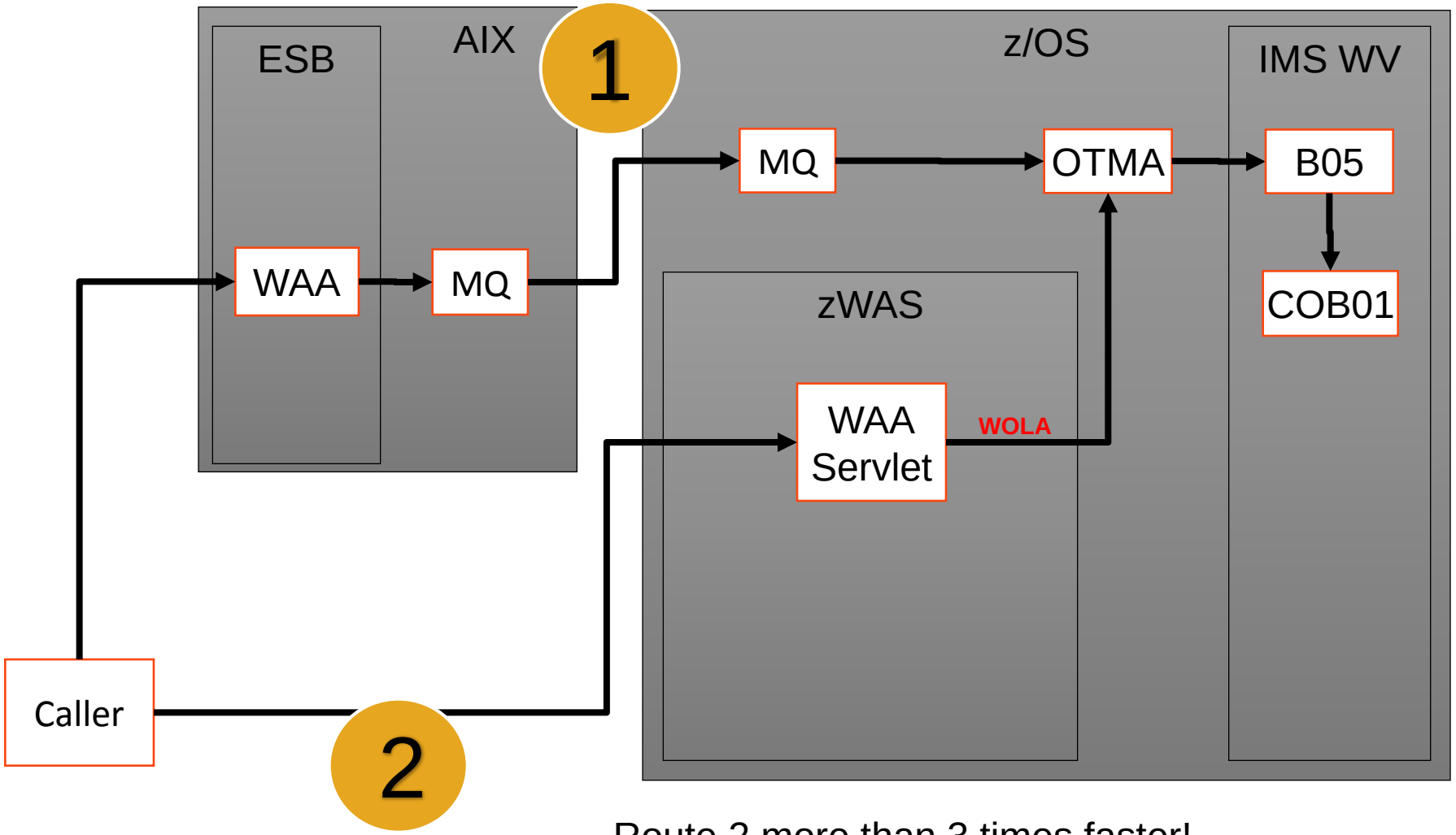
- Actual State: Access only via the DB2-Stored Procedure
  - DB2-Dependant
  - No Flexibility
  - No Access from Distributed Systems



- Target State: Multiple Access Method (MainFrame and Distributed Systems)
  - http://te01.ww-intern.de:32247/IbanKonverter/IKGUI?kto=494949&blz=60050101&land

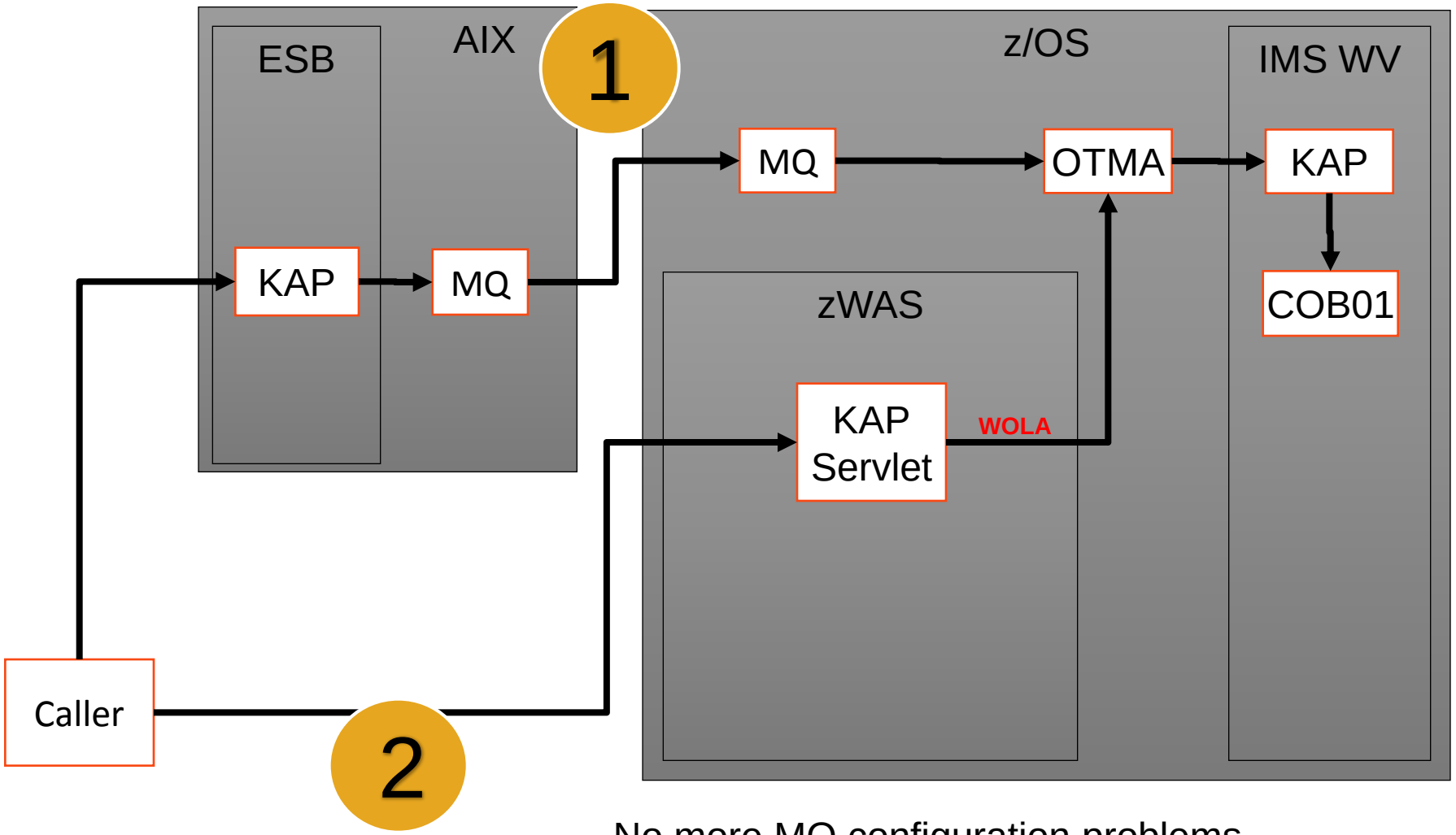


Scenario 2 – WAA-Adapter using WOLA



Route 2 more than 3 times faster!

Scenario 3 – KAP Client Lookup using WOLA



No more MQ configuration problems

WOLA Extra Infos

- Agenda
  1. Wuestenrot & Wuerttembergische at a Glance
  2. Starting Position
  3. Solution: WOLA
  4. Implementation: WOLA
  5. W&W Use Cases: WOLA
  6. Extra Infos:
     - WOLA Architectural Considerations 2010 (DL)
     - Redbook: The WOLA Native APIs – a COBOL Primer
     - http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101490
     - Redbook: New Ways of Running Batch Applications on z/OS: Volume 4 IBM IMS, Volume 4
  7. Summary and Outlook

Summary and Outlook

- Agenda
  1. Wuestenrot & Wuerttembergische at a Glance
  2. Starting Position
  3. Solution: WOLA
  4. Implementation: WOLA
  5. W&W Use Cases: WOLA
  6. Extra Infos
  7. Summary and Outlook
     - Planned Production Dates:
       - IBAN Converter: June 2015, with 100.000 daily COBOL to EJB Requests
       - KAP Client Lookup: October 2015, with 50.000+ daily Java to IMS Requests
       - Tax Software (Cortax): 2015, with 10.000 daily COBOL to EJB Requests

Et Wola

Thanks for listening ……..

……Any Questions

Or contact me on:
daniel.schoeman@ww-informatik.de