We make sure

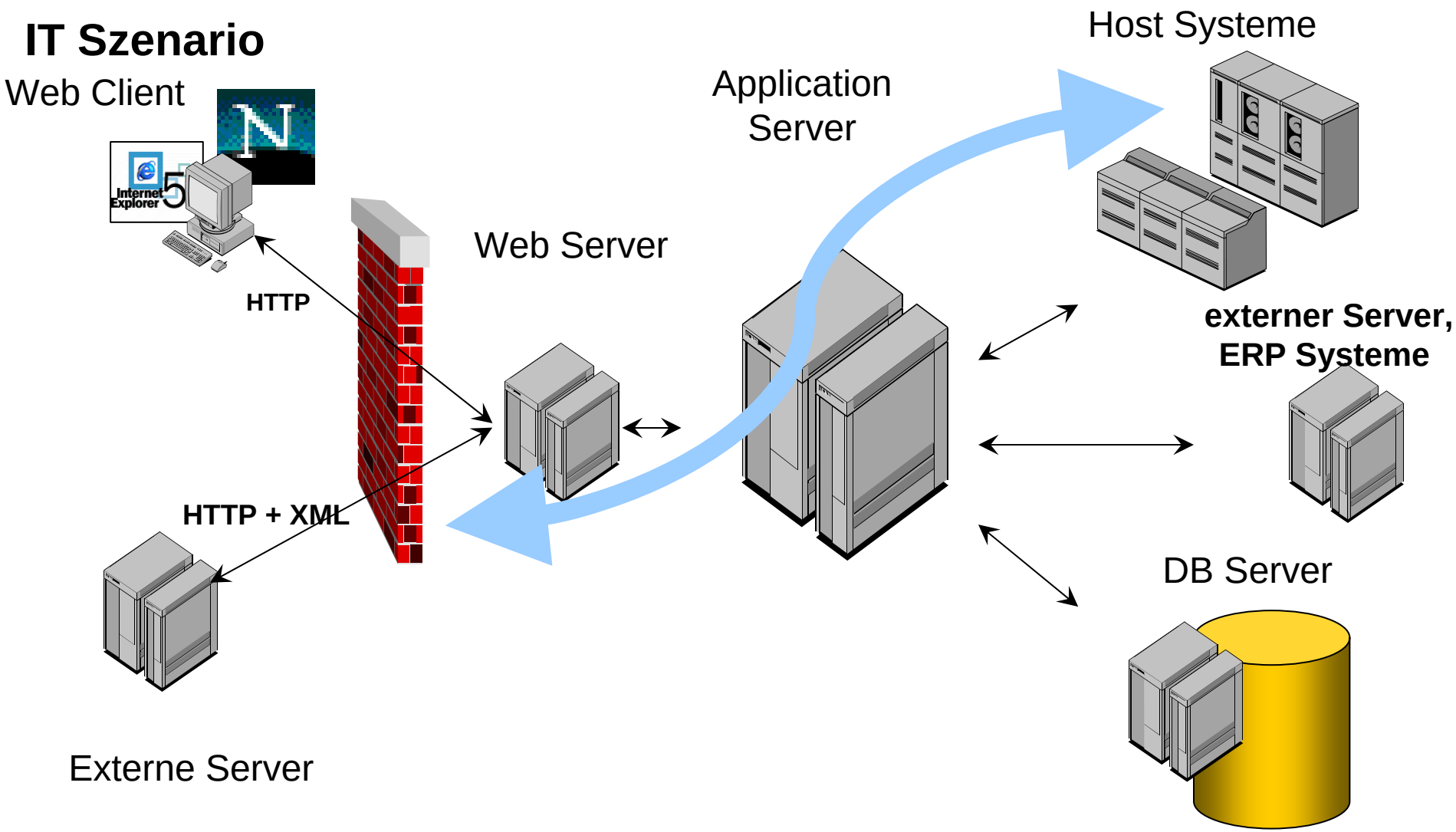
FUJITSU COMPUTERS SIEMENS

Mainframe und J2EE auf Augenhöhe?

JCA 1.5 am Beispiel von
Oracle 10g und BeanConnect 2.0

JAVAFORUM 2005 stuttgart

IT Szenario


Web Client
Internet Explorer 5
Application Server
Host Systeme
Web Server
HTTP
externer Server, ERP Systeme
HTTP + XML
DB Server
Externe Server

Host Anwendungen

- Host Anwendungen bieten relativ oft geeignete Services / Funktionen für Integration
- Host Anwendungen „sprechen“ i.d.R. nicht JAVA, JMS, OO, XML / SOAP
- „Übersetzung / Anpassung“ in einer vorgelagerten Schicht ist daher nötig
- .NET oder J2EE bieten sich an

Charakterisierung von Host Services (1)

- Bereits heute über das Netz aufrufbar, i.d.R. eingebettet in TP Monitor Umgebungen wie CICS, IMS, openUTM
- Paradigma = Servicename + Datenaustausch
- Datenaustausch in Form von strukturierten, abdruckbaren Datenstrukturen (Records) im jeweiligen Host Zeichensatz (EBCDIC)
- Service Paradigmen:
  - synchron (request/reply)
  - asynchron (send and forget)
- Sprechen selbst entfernte Services nach obigen Paradigmen an
- Sind in 3GL Sprachen, meist in COBOL implementiert, manchmal auch in C,PL/1 etc. oder gar noch 390/Assembler
- Beschreibung dieser Daten als COBOL, C, Assembler Datenstrukturen liegt i.d.R. vor

Charakterisierung von Host Services (2)

Wie können Host Services integriert werden?

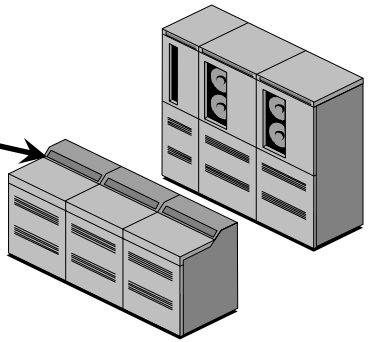
- Über entsprechende Client Bibliotheken der Hersteller
- Gibt es für die gängigen Plattformen
  Windows, Unix, Java, ...
- Vertreter sind ECI (IBM), UPIC (FSC), JCO (SAP) usw.
- Direkt über Host Protokolle LU6, OSI-TP
- Beide Wege können als als J2EE konforme Resource Adapter (JCA Connector) angebotenwerden
- Host Protokollanbindung bietet:
  - 2PC Transaktionen
  - asynchrone Services
  - Bidirektionale Kommunikation (Inbound)

Host Connectivity

J2EE Application Server

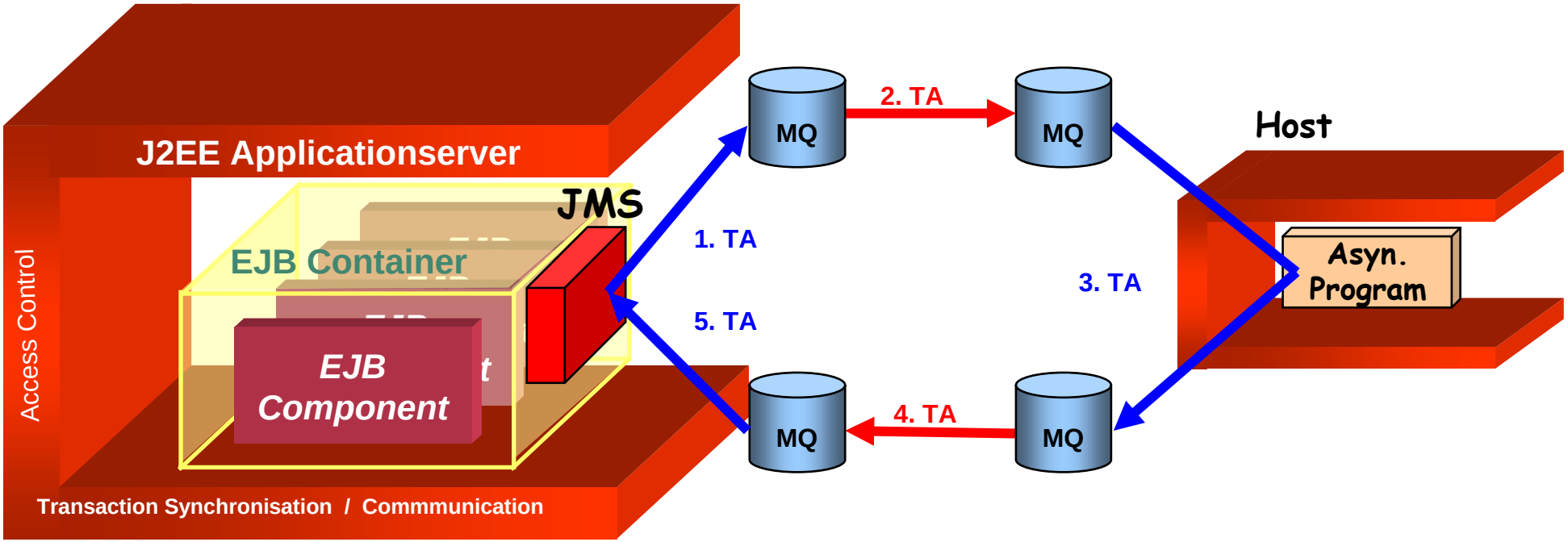
FSC / BS2000/OSD

IBM / z/OS



andere
Unisys über OSI-TP

Host Integration via Message Queues

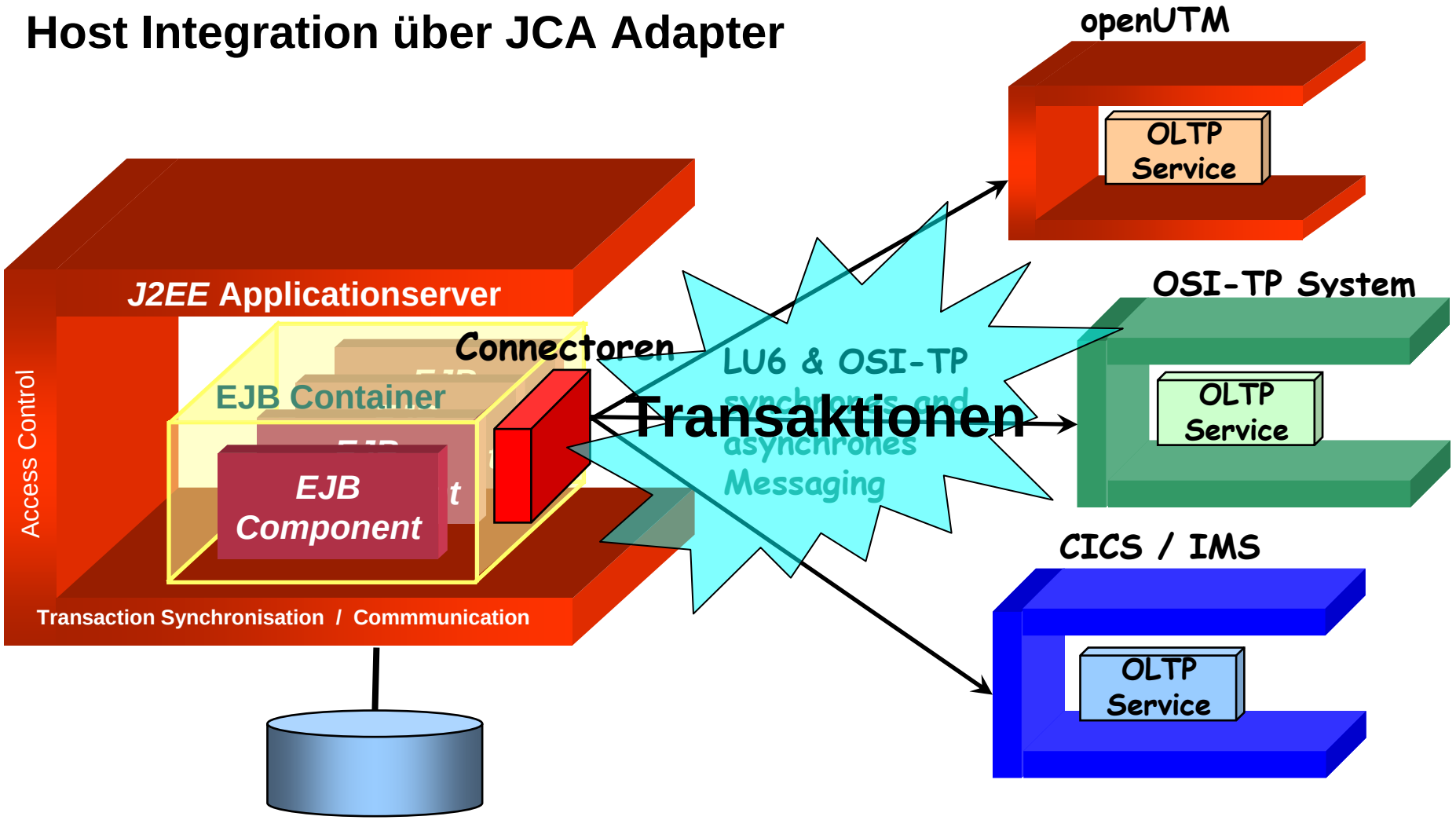


MQ kann sein:

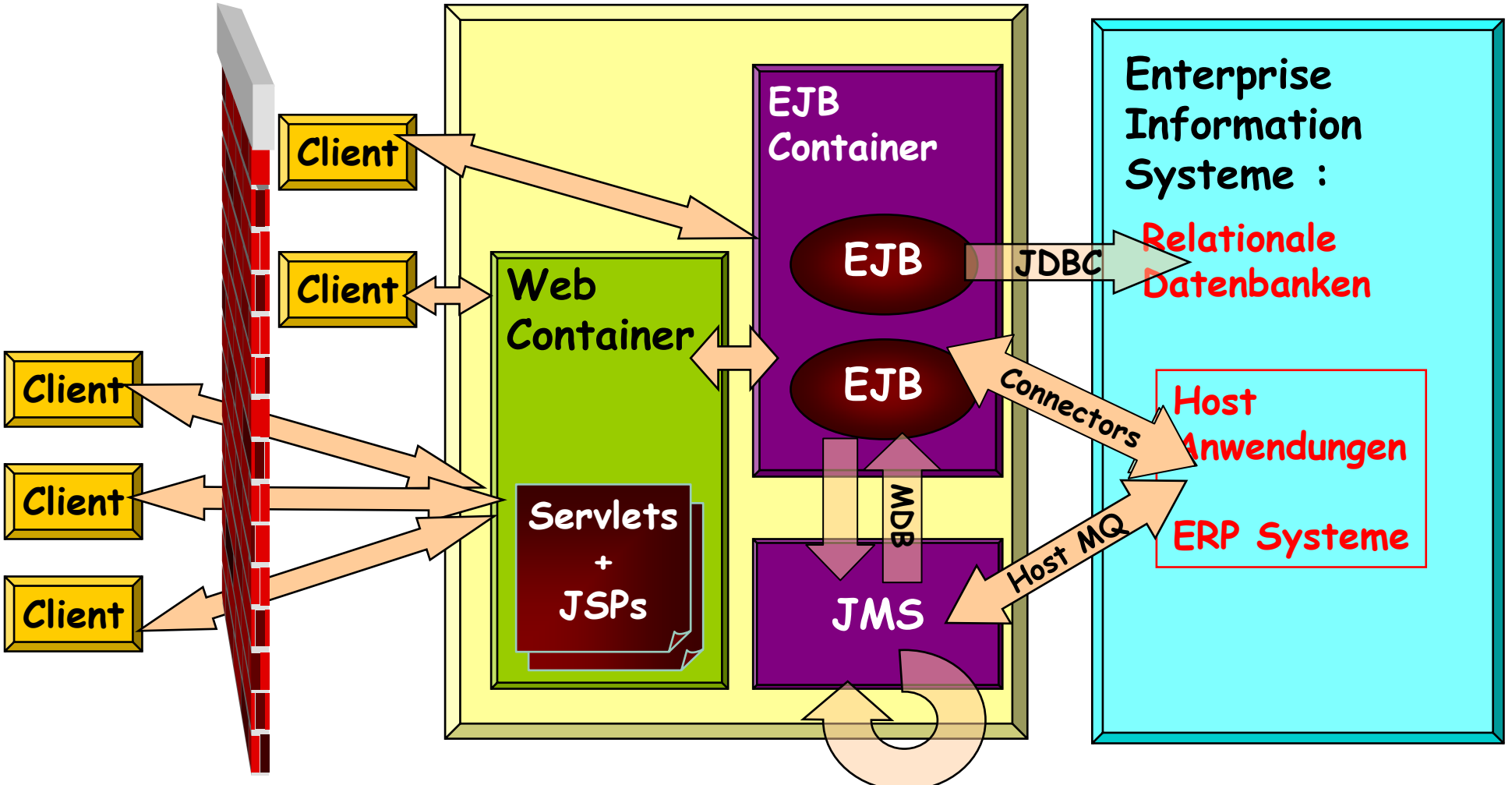
- **JMS**
- **MQSeries**
- **UTM Message Queues (FPUT/DPUT)**
- **CICS Queues**

- **Round-Trip besteht aus 5 Transaktionen, davon 3 in der Anwendung selbst**
- **undefinierte Dauer/Wartezeit für den Round-Trip**
- **Fehlerbehandlung nur über logische Stornierungen (aufwändige Programmierung)**

Host Integration über JCA Adapter



J2EE Architektur



We make sure

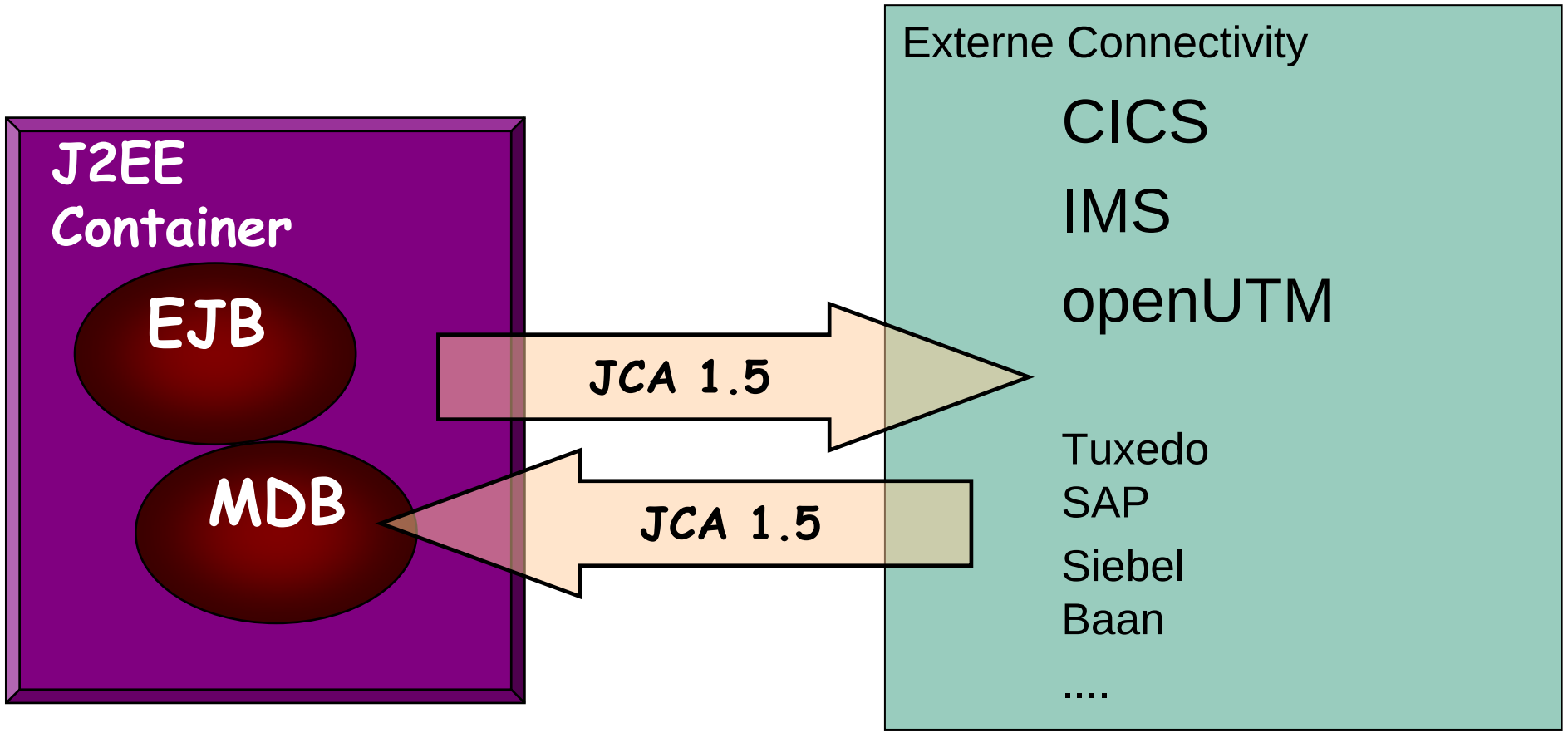
FUJITSU COMPUTERS SIEMENS

J2EE Connector Architektur

JCA 1.5 Resource Adapter

JAVAFORUM 2005 stuttgart

J2EE 1.4 Adapter



JCA Resource Adapter System Schnittstellen (Outbound)

- **Connection Management**

  Pooling etc.

- **Transaction Management**

  Propagierung derJ2EE Transaktion zum EIS (Host) System

- **Security Management**

  Abbildung der J2EE Security auf die EIS (Host) Security

JCA Resource Adapter Anwender Schnittstellen (Outbound)

- **ConnectionFactory**

  Im JNDI sichtbar, entspricht der JDBC Datasource bzw. den JMS Connectionfactories

  `getConnection();`

  `getConnection(...);`

- **Connection**

  Die wichtigste Schnittstelle für den J2EE Programmierer

  Stellt die Abstraktion einer Host Connection bzw. eines Host Service dar

- **Weitere Klassen**

  ConnectionSpec, z.B. zur Übergabe von Authentisierungsdaten

  InteractionSpec, z.B. Parametrisierung der Kommunikation

  Records, z.B. Definition der Übergabedaten

Nutzung von JCA (ConnectionFactory) - im Java Code

```
public class SimpleConnectBean implements SessionBean
{
 private EISConnectionFactory   oltpserver;
 private EISConnection oltp;
 ...
 public void ejbCreate()
 {
  naming = new InitialContext();
  oltpserver = naming.lookup("java:comp/env/eis/myOltp");
 }
 ...
 public String callHost( String input )
 {
  oltp = oltpserver.getConnection();
  oltp.setServiceName ("<EIS_SPECIFIC_FUNCTION_NAME>");
  oltp.snd(input);
  String output = oltp.rcv();
  oltp.close();
  ...
  return output;
 }
}
```

Nutzung von JCA (ConnectionFactory) –Deployment Descriptor

```
<enterprise-beans>
 <session>
  ...
  <resource-ref>
   <res-ref-name>eis/myOltp</res-ref-name>
   <res-type>
    net.fsc.jca.communication EISConnectionFactory
   </res-type>
   <res-auth>Application</res-auth>
   <res-sharing-scope>Unhareable</res-sharing-scope>

  </resource-ref>
  </session>
...
```

Nutzung von JCA (ConnectionFactory) – Deployment im Applicationserver

Connector Deployment BeanConnect

```
java -jar admin.jar ...
 -deployconnector
 -name BeanConnect
 -file beanconnect.rar
```

ConnectionFactory Definition in **oc4j-ra.xml**

```
<oc4j-connector-factories>
 <connector-factory location="eis/BC_OLTP ">
   <connectionfactory-interface>
    net.fsc.jca.communication.EISUpicConnectionFactory
   </connectionfactory-interface>
   <config-property
    name="ConnectionURL"
    value="oltp://UTMBS2"/>
   ...
```

Nutzung von JCA (ConnectionFactory) – Verknüpfung im Applicationserver

```
<orion-ejb-jar>
 <enterprise-beans>
  <session-deployment
    name="SimpleConnect"
    location="SimpleConnect" >
   <resource-ref-mapping

    name=" eis/myOltp "

    location=" eis/BC_OLTP " />

  ...

  </session-deployment>
```

JCA Resource Adapter System Schnittstellen (Inbound)

- **Work Management**

  Möglichkeit des „Listening“ auf eingehende Verbindungen

- **Message Inflow**

  Behandlung eingehender Nachrichten

- **Transaction Inflow**

  Propagierung der EIS (Host) Transaktion zur J2EE Plattform

- **EJB Invocation** (Message Driven Bean)

JCA Resource Adapter Anwender Schnittstellen (Inbound)

Behandlung der Daten über Message Driven Beans

MessageListener Schnitstelle(n)

```
void onMessage(OltpMessage in)
   OltpMessage onMessage(OltpMessage  in)
  ...
```

Datenformate der MessageListener Schnitstelle

```
  OltpMessage msg

  msg.getText()
  msg.getBytes()
  ...
```

Nutzung von JCA (MessageListener) - im Java Code

```
public class SimpleMessageDrivenBean
    implements   MessageDrivenBean
               , OltpMessageListener
{
 MessageDrivenContext ctx;
 ...
 public void setMessageDrivenContext (MessageDrivenContext  ctx)
 {
  this.ctx = ctx;
 }
 ...
 public OltpMessage onMessage(OltpMessage msg)
 {
  ...
    msg.getText();
  ...
 }
}
```

Nutzung von JCA (Inbound) – Beschreibung im Deployment Descriptor

```
<enterprise-beans>
 <message-driven>
  <ejb-name> OltpMsgReader </ejb-name>
   ...
  <messaging-type>
   net.fsc.jca.communication.OltpMessageListener
  </messaging-type>
   ...
  <activation-config>
   <activation-config-property>
    <activation-config-property-name>
     messageEndpoint
    </activation-config-property-name>
    <activation-config-property-value>
     forOltpMsgReader
    </activation-config-property-value>
   </activation-config-property>
  </activation-config>
 </message-driven>
...
```

```
java -jar admin.jar ...
 -deployconnector
 -name BeanConnect
 -file beanconnect.rar
```

Inbound Port Definition in **ra.xml**

```
...
 <resourceadapter>
  <config-property>
   <config-property-name>inboundListenerPort</config-property-name>
   <config-property-type>java.lang.String</config-property-type>
   <config-property-value> 31099 </config-property-value>
...
```

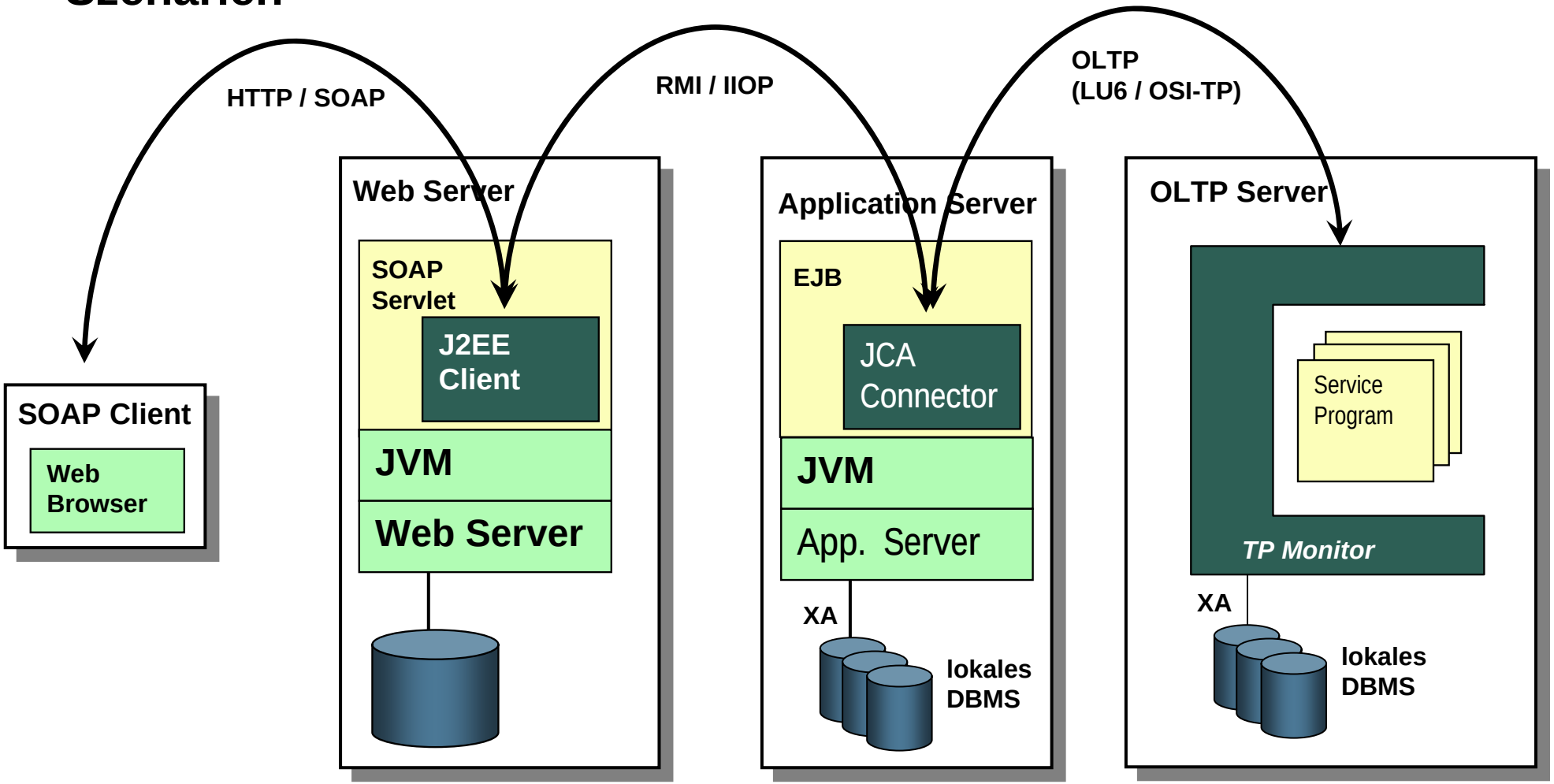
Message Endpoint
in Resource Adapter Konfiguration definiert

forOltpMsgReader

Nutzung von JCA (Inbound) – Verknüpfung im Applicationserver

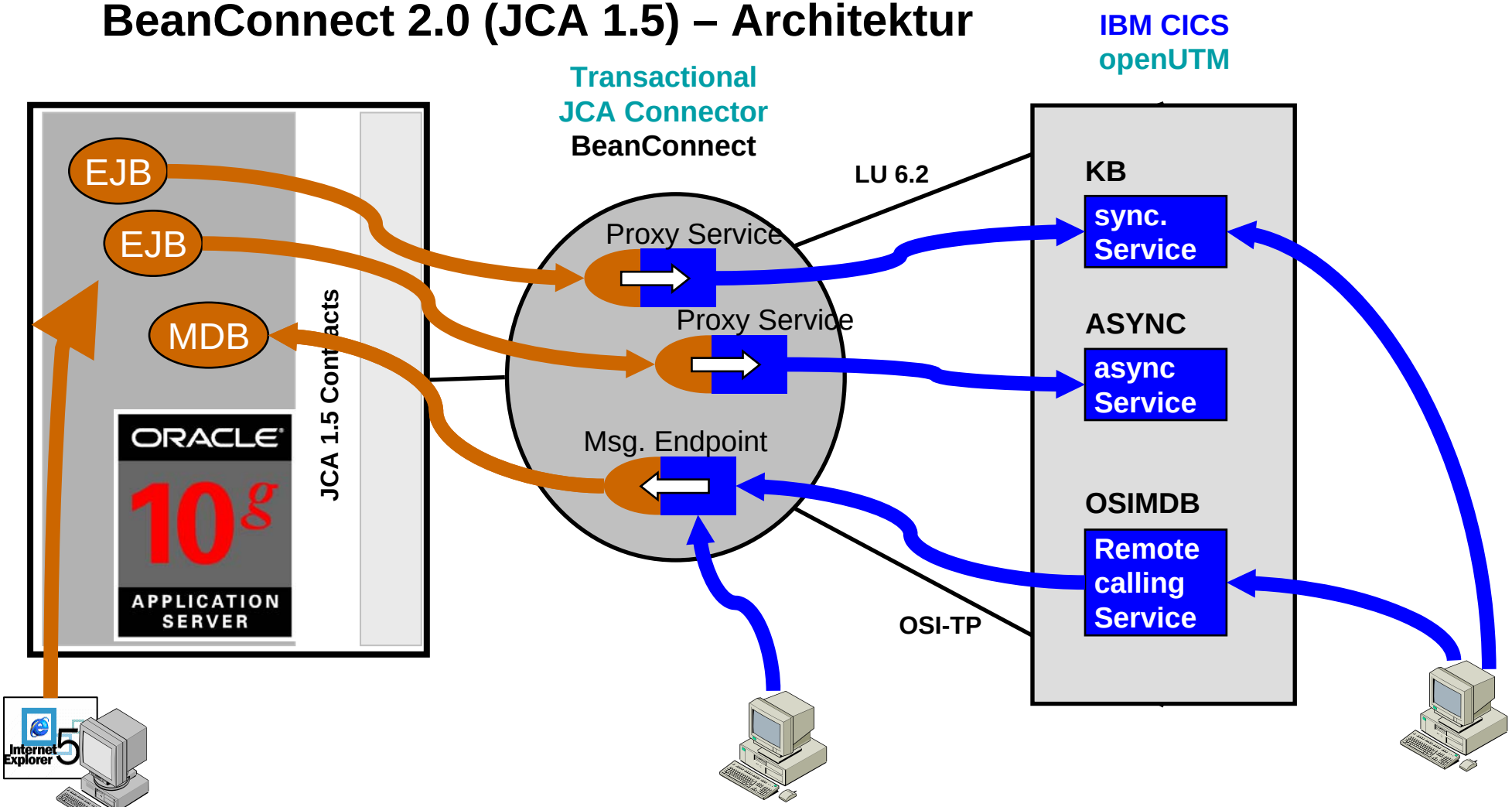
```
<orion-ejb-jar>
 <enterprise-beans>
  <message-driven-deployment
    name=" OltpMsgReader "

    resource-adapter=" BeanConnect ">

...
  </message-driven-deployment>
```

Szenarien

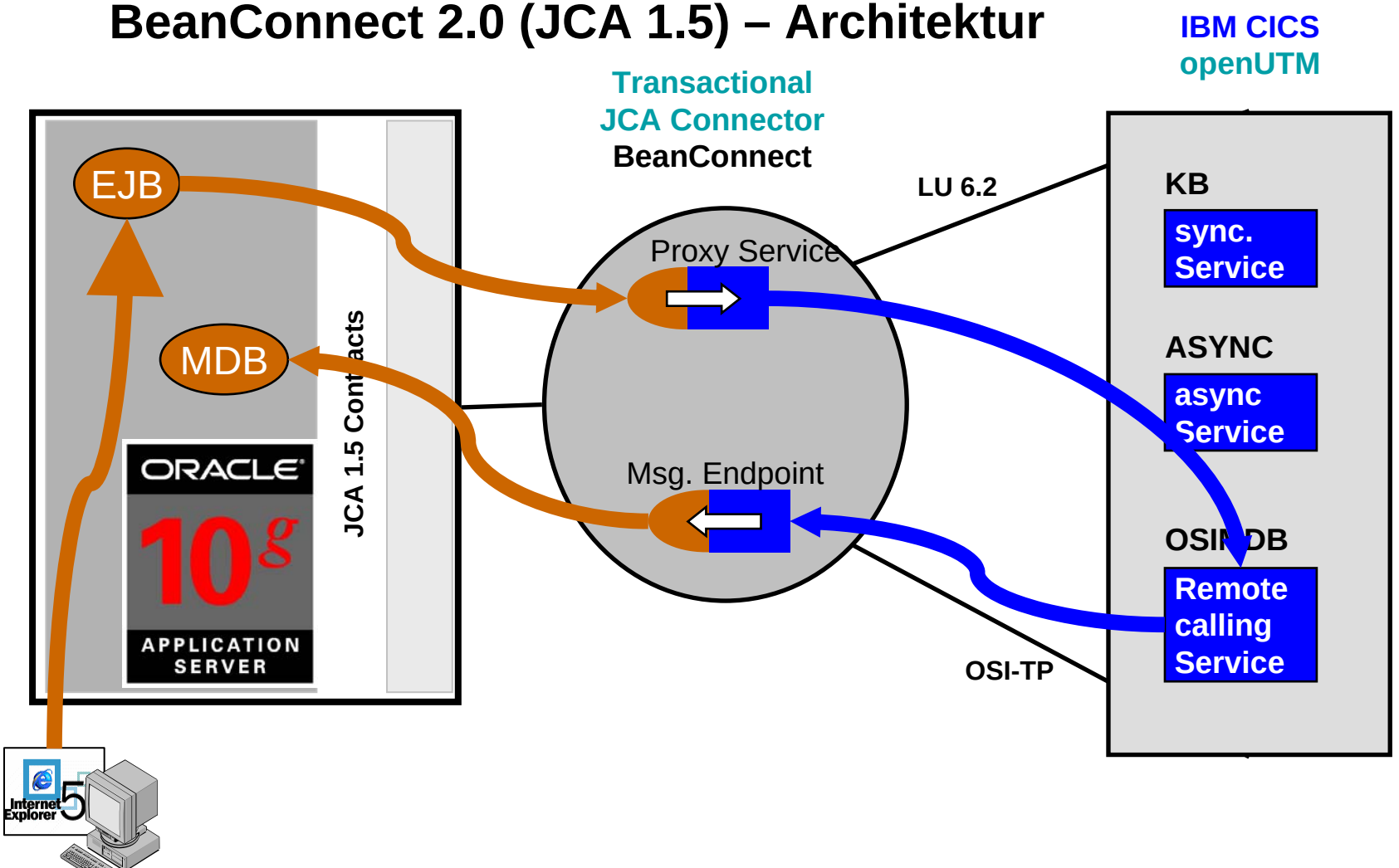

HTTP / SOAP
RMI / IIOP
OLTP
(LU6 / OSI-TP)
SOAP Client
Web Browser
Web Server
SOAP Servlet
J2EE Client
JVM
Web Server
Application Server
EJB
JCA Connector
JVM
App. Server
XA
lokales DBMS
OLTP Server
Service Program
TP Monitor
XA
lokales DBMS

Demo

BeanConnect 2.0 (JCA 1.5) – Architektur



BeanConnect 2.0 (JCA 1.5) – Architektur



... vielen Dank