Service Oriented Mainframe Integration with CORBA, J2EE and Web Services

Dr. Arne Koschel
Technical Product Manager, Product Specialist
IONA Orbix Application Server Platform

Content

• Background
  – CORBA, J2EE, Web Services

• Interoperability examples
  – Code: EJB <-> CORBA, Web Services & …

• Service Oriented Architecture

• Technology Example:
  Orbix E2A Application Server platform

• Application Examples

• Q & A
CORBA: Common Object Request Broker Architecture
OMA: Object Management Architecture

IDL
for Services
and application objects

Java
C++
Naming

ORB: CORBA Software Bus

VB
COBOL
Smalltalk

CORBA services
Naming
Events
Transactions
Security
Trader
Notification
Persistence
Management
(plus other less important ones)

CORBA facilities
Logging, Finance, Healthcare, ...
(most: work in progress)

UML & MDA
Model Driven Architecture

J2EE Platform Specification

Applet Container
J2SE

Application Client Container
J2SE

JSP
Servlet

EJB
J2SE

Existing Systems
(CORBA, Mainframe, ...)

Database
Web Services “Core”

Interoperability examples
Deep CORBA/J2EE Interop

- Full sharing of security and transaction context
- Java-to-IDL mapping using OBV
- Wrap a CORBA server with EJB, or have EJB make direct IIOP calls
- Pre-CORBA 2.3 clients via bridge

EJB -> CORBA: How to?

- Generate CORBA stubs from CORBA IDL
- use e.g. CORBA Naming Service to get the IOR
- just do usual CORBA calls
- With an ORB based J2EE implementation often transactional interoperability “very easy”, if needed 2-PC
  - just: “Get same ORB as EJB’s”
  - this allows for transparent TX-context passing.
EJB to CORBA

// IDL: EJB_to_CORBA.IDL

module corbaserver {
    struct addressStruct {
        string name;
        string number;
        string street;
        string town;
        string state;
        string zip;
    };

    struct nameStruct {
        string name;
        string number;
    };

    interface ejb_to_corba_int {
        void getNumber(in nameStruct Name, out addressStruct Number);
    };
};

EJB to CORBA (1)

- Within an EJB, e.g. a method in a Session Bean:
  start with getting the CORBA ORB (here: IIOP is native remote protocol, thus ORB call is efficient “in process”)

// Declare some CORBA variables
org.omg.CORBA.ORB orb = null;
org.omg.CosNaming.NamingContextExt context = null;
javax.naming.Context jndi_context = null; org.omg.CORBA.Object objRef = null;
Properties myProps;

// Get the CORBA ORB from "your" EJB
// The ORB named "Orbix.J2EE" is preconfigured with certain properties such as the
// ability to propagate security and transaction context.
Properties orbProperties = new Properties();
orbProperties.put("org.omg.CORBA.ORBClass", "com.iona.corba.art.artimpl.ORBImpl");
orbProperties.put("org.omg.CORBA.ORBName", "Orbix.J2EE");
org.omg.CORBA.ORB init(orbArgs, orbProperties);
EJB to CORBA (2)

corbaserver.ejb_to_corba_int.directoryEnquiries = null;
NameComponent[] tmpName = new NameComponent[1];
try
{
    // Ask the ORB for a reference to the Naming Service
    objRef = orb.resolve_initial_references("NameService");
    context = NamingContextExtHelper.narrow(objRef);

    // Create a NameComponent from the object name
    tmpName[0] = new NameComponent("DirectoryEnquiries", ";");

    // Resolve the NameComponent in the Naming Service
    objRef = context.resolve(tmpName);

    // Narrow the returned object reference to an object of type "directory"
    directoryEnquiries=(corbaserver.ejb_to_corba_int)corbaserver.ejb_to_corba_intHelper.narrow
        (org.omg.CORBA.Object)objRef;
}

catch ( org.omg.CORBA.ORBPackage.InvalidName ex )
{
    ...
}

EJB to CORBA (3)

// Populate the input address struct with strings, using the generated constructor
nameStruct myName = new nameStruct(in_name, "0");

// Create a Holder to store the output nameStruct
addressStructHolder ns = new addressStructHolder();

// Call the CORBA method getName on the CORBA object reference directory
try
{
    System.out.println("BEAN> Calling getName() method on DirectoryEnquiries CORBA server object");

directoryEnquiries.getNumber(myName, ns);

    System.out.println("BEAN> After calling getNumber()");
}

catch(org.omg.CORBA.SystemException e)
{
    ...
    return ns.value.number;
}
CORBA to EJB (1)

- Pure CORBA clients can call EJB’s
- Standard RMI/IIOP / reverse Java way:
  - use “JDK rmic” to generate IDL from EJB’s, e.g. take an EJB “Sample”
    rmic -idl SampleHome Sample
  - use pure CORBA client to call the generated interface to “Sample”

- Problem:
  - standard mapping exposes “value types” an awful lot, which results in very ugly IDL

CORBA to EJB (2)

- Proposal: CORBA friendly EJB’s
- restrict datatypes in order to limit value types; use automatic wrapper for calls
- note: with both approaches Orbix E2A CORBA & J2EE allow TX/Sec-context passing
EJB & CORBA: Security - cont’d

- Encryption:
  - J2EE: HTTPS, RMI-IIOP over TLS
  - CORBA: TLS/SSL, CSIV2

- Credentials / Interceptors
  - HTTPS & Pluggable Authentication “get X.509 certificate in Servlet” (alternative: direct CORBA client call)
  - J2EE “hook” to set own EJB-Principal for “Servlet --> EJB call”
  - Principal passed as “CORBA Service Context”
  - EJB --> EJB and EJB --> CORBA calls pass Service Context “under the hood”
  - final ART-Plugin: “Principle propagator”: “set CORBA User Principal” based for IONA IMS-Adapter to provide OS-390 RACF authentication

IONA Security Services
Web Service -> EJB / CORBA: WSDL

```xml
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="ConverterService" targetNamespace="urn:target-converter-service"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
  ... more "includes"

  <message name="inchToMM">
    <part name="param0" type="xsd:float"/>
  </message>
  <message name="inchToMMResponse">
    <part name="return" type="xsd:float"/>
  </message>
  ... more messages

  <portType name="ConverterPortType">
    <operation name="inchToMM">
      <input message="tns:inchToMM" name="inchToMM"/>
      <output message="tns:inchToMMResponse" name="inchToMMResponse"/>
    </operation>
  </portType>

  <binding name="ConverterPortBinding" type="tns:ConverterPortType">
    <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="inchToMM">
      <input name="inchToMM">
        <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="urn:target-converter-service" use="encoded"/>
      </input>
      <output name="inchToMMResponse">
        <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="urn:target-converter-service" use="encoded"/>
      </output>
    </operation>
  </binding>

  <service name="ConverterService">
    <port binding="tns:ConverterPortBinding" name="ConverterPort">
      <soap:address location="http://localhost:8080/xmlbus/container/Converter/ConverterService/ConverterPort"/>
    </port>
  </service>
</definitions>
```
Sample Java Client for Web Service

principle is similar for other languages

```java
String soapAction = "";
StringBuffer envelope = new StringBuffer("<?xml version="1.0" encoding="UTF-8"?>");
// prepare a SOAP message (a SOAP envelope)
envelope.append("<soap:Envelope 
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:m="urn:target-converter-service">
 <m:inchToMM xmlns:m="urn:target-converter-service" SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <param0 xsi:type="xsd:float">"");
envelope.append("</soap:Envelope>
// Define SOAP message handler
SOAPResponse_Handler handler = new SOAPResponse_Handler("urn:target-converter-service",
 "inchToMMResponse", "return");
// do SOAP call
URL url = new URL(soapEndPointURL);
HttpURLConnection connect = (HttpURLConnection)url.openConnection();
connect.setRequestProperty("SOAPAction","" + soapAction + "");
connect.setRequestProperty("content-type","text/xml");
connect.setRequestProperty("content-length","" + bytes.length);
OutputStream out = connect.getOutputStream();
out.write(bytes); out.flush();
```

J2EE 1.3 - Interoperability

• Within EJB 2.0 specification mandated
  – Interoperable requests using RMI/IIOP & Java IDL
  – Interoperable EJB calls across vendors
  – CSiV2 L0 Secure interoperability (“Security Context”)

• optional
  – Naming interoperability: now CORBA Interoperable Naming Service (INS) / JNDI over INS
  – Transactional interoperability: now CORBA Object Transaction Service (“Transaction Context”)

• Future J2EE 1.4 (H1 2003) - Web Services via JSR 109
  – SOAP
  – standard: expose stateless session beans as Web Services
  – ...
Service Oriented Architectures

Service Oriented Architecture

- Bridges multiple technology domains
- Multiprotocol support
- Independent services
Service oriented Design

• Rough design notes:
  – large scale enterprise applications need service oriented architectures with true “peer to peer” views from EJB <-> CORBA and - with suitable granularity - Web Services
  
  – this needs clearly defined, semantically rich “rich enough” interfaces like EJB home/remote, IDL, WSDL
  
  – For example
    • within departments use EJB / CORBA, as proven scalable solutions
    • between departments / enterprises expose Web Services interfaces as well
Orbix Application Server Platform

IONA’s core for interoperability of CORBA, J2EE, Web Services & “the Mainframe”

Orbix E2A Application Server Platform

• Three integration technologies
  – The world’s leading provider of CORBA solutions
  – Technically superior J2EE app server, ranked 4th by Gartner
  – Award-winning Web services products

• One Integration Platform
  – Best-of-breed products, wholly integrated, with common enterprise qualities of service
### Adaptive Runtime Technology

- Distributed computing engine (C++ and Java versions available)
- Supports multiple “interface personalities”
- Easily tailored via plug-in architecture, letting us add new integration approaches easily
- Supplies essential enterprise app features (load balancing/fault tolerance, transactions, managed persistence, security, etc.)

### J2EE Tech Edition Architecture

![Diagram of J2EE Tech Edition Architecture]

**Components:**
- **ART (Adaptive Runtime Technology)**
- **EJB Container**
- **Stubs/Skeletons**
- **DII/DSI (Dynamic Interface/DSI)**
- **Core Managers**
- **Plug-ins**
- **JMS (Java Message Service)**
- **Java Mail**
- **T3 (Transaction Service)**
- **JNDI (Java Naming and Directory Interface)**
- **POA (Portable Object Adapter)**
- **GIOP/IOP (General Inter-ORB Protocol)**
- **SSL (Secure Sockets Layer)**
- **OTS (Object Transaction Service)**
- **DynAny**

---

© Copyright IONA Technologies 2003
Orbix E2A - CORBA

- Diverse heterogeneous distributed computing systems are here to stay
- CORBA is all about integration and flexibility
  - widely-applicable interface typing system
  - flexible remote invocation semantics
  - multiple language mappings
  - extensive platform coverage
  - suitable for wrapping old programmes and for writing new ones
  - many enterprise CORBA services

Mainframe Edt.: IMS/CICS Connectors and Adapters

- Connector: “raw” use of existing IMS/CICS transaction
  - Connector: “raw” use of existing IMS/CICS transaction
  - Adapter: full COBOL/PL-I support; peer-to-peer
  - both: “just” CORBA client/servers (2-PC Tx, secure if needed)
Architecture - ASP Web Services on XMLBus Technology
“Expose CORBA-Servers (incl. IMS/CICS) and EJB’s as Web Services

- Messaging Bridge
- Interface Aggregator
- CORBA, J2EE, Java Serializer
- Transformation
- Process Engine
- WS Router

Service Registry (UDDI, JAXR)
Contains WSDL generated for service
Allows access to component config/mgmt

MQSeries, JMS

J2EE, CORBA, and Java Apps

Web Services (developed with any tool(s))

Application examples
First Northern Bank (FNB) Demo

- Demonstrates CORBA, EJB and Web Services parts of Orbix E2A ASP all working together

- The basis of IONA’s Orbix E2A Standard & Enterprise Edition “Out Of The Box” demo (full source code)

- Have a look at
  - www.iona.com - to get an eval. licence => FNB source
  - www.iona.com/docs -> ASP 5.1 -> Tutorials incl. FNB

Structure of the FNB Demo
Winterthur e-Platform
X509-based security from HTML/Java -> …. -> OS/390

Large German bank computing centre
architectural goal: true service oriented transactional/secure peer-to-peer view of participating systems
Large American gifts, clothes, ... re-seller

• Secured Web-Services for gift cards
• XML-documents
• Rough technical architecture
  – .NET-based clients “Web Store”
  – XMLBus Web-Services access
  – Orbix E2A J2EE Technology Edition
    Session Beans, which call
  – the OS/390 CICS/COBOL “Gift Card Application”

Conclusion

• Technology Background
  – CORBA, J2EE, Web Services
• Interoperability examples: today
• Service Oriented Architecture
• It’s real:
  Orbix E2A Application Server Platform &
  Real world example architectures
Q & A

Further information:
www.iona.com

e-mail:
pm-asp@iona.com
arn.koschel@{iona.com | gmx.de}