CORBA Services
Which of these nodes could be a CORBA client or server?
What is CORBA?

- A specification for developing language and platform independent distributed object applications
- Uses remote proxy design pattern in conjunction with Object Request Brokers (ORBS)
  - uses IIOP for wire level protocol
  - precursor to RMI

A CORBA environment is based on client applications finding and using objects that provide a desired function. The objects typically represent something in the real world, for example shopping carts, and are hosted by servers (typically CORBA servers or EJB servers). The type of object is defined by its interface and the semantics defined for that interface. There can be many instances of an object (with the same interface and semantics), but representing different entities. CORBA provides the Interface Definition Language (IDL) to define object interfaces, and ORBs to provide access to objects through a distributed environment. The binding of an object's interface to a specific implementation is handled in the server environment.
Supporting CORBA

The purpose of providing support for CORBA is to enable customers with existing CORBA investments to access new business value on the J2EE model while continuing to leverage their existing systems.

- **Basic Function**
  - C++ client to WebSphere EJB
  - WebSphere EJB to C++ ORB

- **Strategy**
  - support popular (in-use) ORBs
    - Visibroker
    - Orbix
  - test and verify common interoperability scenarios
WebSphere CORBA Support

CORBA Support comes in several forms

- Interoperability with other ORBS
  - C++
  - Java
- CORBA C++ SDK
  - C++ Client
    - headers
    - samples
    - runtime
  - C++ Server
    - headers
    - samples
    - runtime
- Value Type Library
  - used for pass-by-value support
  - a library of C++ implementations of commonly used Java classes
    - Integer, Float, Vector, Exception, OutputStream, etc.

WebSphere Application Server provides CORBA support that enables the use of CORBA interfaces between a server object providing a service and a client using the service. In practice this means WebSphere C++ CORBA servers and WebSphere EJB services can be accessed by CORBA clients, and WebSphere CORBA clients can access CORBA servers.

As part of the WebSphere J2EE environment, the C++ CORBA support provides a basic CORBA environment that can bootstrap into the J2EE name space and can invoke J2EE transactions. However, it does not provide its own Naming and Transaction services, for which a C++ CORBA client or server relies on the J2EE environment as a service provider. The C++ CORBA technology is provided on Solaris (Forte C++), AIX (VisualAge for C++), and Windows NT and Windows 2000 (Microsoft Visual C++).
WebSphere CORBA Interoperability

- Support Statements
  - Results of Controlled Tests
  - Specifically Identified Environments
  - Focused around IIOP interoperability

- Supported Environments
  - Client Platform/OS
  - Server Platform/OS
  - Variety of Client/Server Scenarios
    - Vendor C++ CORBA Client to WAS EJB (RMI-IIOP)
    - WebSphere CORBA Client to Vendor C++ CORBA Server
    - WebSphere CORBA Client to Vendor Java CORBA Server
    - WebSphere CORBA Client coexistent with Vendor Java ORB to Vendor C++ CORBA Server

- Sample Code
  - For Supported Environments
  - Demonstrates Supported IIOP Interoperability

WebSphere CORBA Interoperability

- IIOP Interoperability
- simple data types
- variety of compound data types
- valuetypes
- message formats (fragmented not supported)
- bi-directional IIOP not supported.
Interoperability vs. Coexistence

- **Interoperability** (*preferred mode*)
  - Invoke distributed objects using the ORB included with WebSphere Application Server.
    - Enables the propagation of service contexts, such as in-progress transactions.

- **Coexistence**
  - Invoke distributed objects using a 3rd party ORB running in the WebSphere software platform environment.
    - Enables the use of 3rd party ORB bootstrapping protocol and vendor-specific APIs.
Interoperability

WebSphere 4.0 Application Server

- EJBEJB Container
- Web Container
- JSP
- Servlet

Some J2EE client
- IBM or vendor

Vendor C++ CORBA client

Value Types

IDL is generated from the EJB

WebSphere as an EJB server

WebSphere as a CORBA client

Java CORBA

Common Vendor ORBs
Visibroker and Orbix (Java and C++)

Vendor ORB

IBM ORB

C++ CORBA

Value Types

Value Types
When you really need more.

What happens if you need closer integration with the vendor ORB for such things as their naming service or security?

Sometimes the only way to talk to a given vendor ORB is to use their client ORB.

If this is the case, then we can help here too
The client ORB for the vendor ORB is instantiated on the fly by the EJB or the Servlet.

Some J2EE client
- IBM or vendor

Some HTTP client

WebSphere as a server hosting vendor ORBs

WebSphere 4.0 Application Server

EJB

Servlet

Vendor ORB

Java CORBA

Vendor ORB

C++ CORBA

Value Types

Common Vendor ORBs
Visibroker and Orbix
( Java and C++ )
WebSphere CORBA Interop Testing

R3.5 ptf 3
- AE 3.5.3 and Visibroker Java 3.4 interoperability and coexistence on Windows NT and Solaris 7
- AE 3.5.3 and Visibroker C++ 3.3 interoperability on Windows NT
- AE 3.5.3 and Orbix Web 3.2 interoperability and coexistence on Windows NT and Solaris 7
- AE 3.5.3 and Orbix C++ 3.0.1 interoperability on Windows NT
- AE 3.5.3 and Orbix C++ 3.0.2 interoperability on Solaris 7

R4.0
- WS 4.0 and Visibroker Java 3.4 interoperability and coexistence on Windows NT, Windows 2000, Solaris 7, AIX 4.3
- WS4.0 and Visibroker C++ 3.3 interoperability on Windows NT, Windows 2000, AIX 4.3 (no Solaris)
- WS 4.0 and Visibroker Java 4.0 interoperability and coexistence on Windows NT, Windows 2000, Solaris 7, AIX 4.3
- WS 4.0 and Visibroker C++ 4.0 interoperability on Windows NT, Windows 2000, Solaris 7, AIX 4.3
- WS4.0 and Visibroker C++ 4.1 interoperability on Windows NT, Windows 2000, Solaris 7, AIX 4.3
- WS 4.0 and Orbix Web 3.2 interoperability and coexistence on Windows NT, Windows 2000, Solaris 7, AIX 4.3
- WS 4.0 and Orbix C++ 3.0.1 interoperability on Windows NT, Windows 2000, AIX 4.3
- WS 4.0 and Orbix C++ 3.0.2 interoperability on Solaris 7
- WS 4.0 and Orbix 2000 C++ interoperability on Solaris 7
- CORBA C++ SDK 4.0 client and WS 4.0 EJB interoperability on Windows NT, Windows 2000, Solaris 7, AIX 4.3
CORBA
C++ SDK
WebSphere CORBA C++ SDK

- **Basic Function description**
  - CORBA C++ 2.1 basic function - no extended qualities of service
  - used to integrate C and C++ applications with J2EE applications

- **Lightweight C++ CORBA server**
  - replace existing C++ CORBA servers
  - a place to do heavy duty C++ functions

- **Build C++ CORBA clients**
  - headers and libraries
  - idl compiler (idlc)
You can use the C++ CORBA SDK to build a lightweight WebSphere CORBA server for use with new or existing C and C++ programs. You can also use the SDK to build a WebSphere C++ CORBA client for use with a WebSphere C++ CORBA server, WebSphere EJB server, or with a 3rd-party C++ CORBA server (as part of a CORBA interoperation scenario). For example, you could use the SDK to build a C++ CORBA client to connect a C++ desktop application to a WebSphere EJB server.
As part of the WebSphere J2EE environment, the C++ CORBA support provides a basic CORBA environment that can bootstrap into the J2EE name space and can invoke J2EE transactions. However, it does not provide its own Naming and Transaction services, for which a C++ CORBA client or server relies on the J2EE environment as a service provider. The C++ CORBA technology is provided on Solaris (Forte C++), AIX (VisualAge for C++), and Windows NT and Windows 2000 (Microsoft Visual C++).
Value Type Library
The Java Language to IDL specification maps Java serializables to CORBA value types. Therefore every Java serializable to be passed by a CORBA client as a parameter or return value for an Enterprise JavaBean must be re-implemented in the language of the client.

### The Value Type Library

- Contains C++ value type implementations for the commonly used Java primitive classes
- Enables CORBA programmers to use the WebSphere C++ classes in the same way they would use their Java counterparts.
- Java Class hierarchy is preserved.
- Constructors become init() methods on the factory classes.

The Java Language to IDL specification maps Java serializables to CORBA value types. Therefore every Java serializable to be passed by a CORBA client as a parameter or return value for an Enterprise JavaBean must be re-implemented in the language of the client. Implementation of Java serializables as value types in C++ or another language can be a significant development effort.

To aid application development, WebSphere Application Server provides a valuetype library that contains C++ valuetype implementations for some commonly used Java classes in the java.lang, java.io, java.util, and javax.ejb packages. For example, Integer, Float, Vector, Exception, OutputStream, and so on. The valuetype library supports the WebSphere C++ ORB.
In a Nutshell

- Java objects serialized

- Serialized data sent across ORB
  - Deserialized by C++ implementation class

- C++ implementation classes reproduce methods, inheritance hierarchy of corresponding Java class
  - C++ ::java::lang::Integer has same methods as Java's java.lang.Integer class, such as intValue() method (inherited from ::java::lang::Number)

  - Some method names change (get/set removed on attribute accessors, overloaded methods mangled)

IBM
Value Types

- java_io
  - java_io_OutputStream
  - java_io_PrintStream
  - java_io.Writer
  - java_io_PrintWriter

- java_ejb
  - java_ejb_EJBMetaData

- java_util
  - java_util_Vector_Impl

- java_lang
  - java_lang_Throwable
  - java_lang_Number
  - java_lang_Boolean
  - java_lang(Byte
  - java_lang_Character
  - java_lang_Double
  - java_lang_Float
  - java_lang_Integer
  - java_lang_long
  - java_lang_short

- Hashtable
Using the Value Type Library

- Valuetype library is called "Vtlib" (HH, LIB, DLL)
  - Basic flow (NT/Win2K):
    - Run `rmic -idl` against your bean and home
    - Run `idlc -shh:uc` against generated IDL
    - Run MSVC++ CL compiler against generated `_C.cpp`
    - Run MSVC++ LINK to build client binding DLL
    - Run MSVC++ CL compiler against client program
    - Run MSVC++ LINK to build client executable

- Use `-DSOMINCLUDENVTLIB` with MSVC++ CL
  - Causes Vtlib.hh to be included

- Link against Vtlib.lib

- Set `SOMCBPROPS` to configure C++ ORB
  - Points to properties file (specifies name server host, etc.)

IBM
try {
    // get the home via normal CORBA lookup
    ... 
    // create an instance of the bean via the home
    bean = home->create();

    // example of method not using valuetype
    bean->processTheClaim(5);

    // call bean's "java.lang.Integer getPolicyNo()" method
    ::java::lang::Integer policyNumber = bean->policyNo();
    cout << "Policy Number = " << policyNumber.intValue() << endl;

    // call bean's "java.lang.String getPolicyHolder()" method
    // Note Java String maps to CORBA WStringValue
    ::CORBA::WStringValue_ptr policyHolder = bean->policyHolder();
    // convert to string turns double byte string to single byte
    cout << "Policy Holder = " << convert_to_string(policyHolder) << endl;
} catch (::javax::ejb::CreateException &ce) {
    cout << "CreateException: " << ce.message() << endl;
} catch(...) {
    cout << "Unknown exception" << endl;
}
Summary

- Bi-directional CORBA connectivity
  - J2EE EJB interoperability
  - coexistence with third party CORBA ORBs
    - Visibroker
    - Orbix
- Foreign ORBs as clients to EJB 1.1 server
  - Visibroker and Orbix
    - C++ via IIOP
    - Java via RMI-IIOP
- WebSphere Advanced Edition server as a CORBA client to
  - Visibroker
  - Orbix
  - using the embedded WebSphere or non-IBM ORB
- IBM CORBA (C++) client to EJB 1.1 server
  - (IBM C++ ORB stack running on a client)
- CORBA 2.1 C++ Software Development Kit (SDK) for
  - building C++ CORBA clients and servers
  - allow companies to integrate C and C++ assets with the J2EE

IBM