



## Examining the New CORBA 3 Specifications

Jon Siegel, Ph.D.  
Director, Technology Transfer  
Object Management Group  
siegel@omg.org  
781-444-0404

4/11/01

Copyright ©2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

1



## CORBA 3.0 Will Add --

- Improved Java and Internet Integration
  - Java-to-IDL (reverse) Mapping
  - Firewall Specification
  - CORBA Object URLs
- Quality of Service Control
  - Asynchronous Invocation/Messaging
  - Invocation QoS Control
  - Realtime, Minimum, Fault Tolerant CORBA
- CORBA Component Model
  - Objects Pass-by-Value
  - Component container
    - Transactional, Persistent, Secure
  - Distribution Format
  - Scripting Language Specification

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

2



## The CORBA 2.0/2.1 Client...

- Can invoke and pass CORBA objects by reference only
- Can invoke synchronously using the SII or DII, or deferred-synchronous using DII only



## The CORBA 3 Client...

- Can invoke synchronously or asynchronously on any CORBA object, using either SII or DII
- Can make time-independent invocations
- Can negotiate a Quality of Service level with the server and network
- Can take advantage of a reliable network infrastructure
- Can invoke and pass CORBA objects by reference, and pass CORBA valuetypes by value
- Can find and invoke objects using URL-format object references
- Can invoke operations on RMI/IDL Java objects
- Can deal with both client-side and server-side firewalls
- Can use both factory and finder patterns to interact with CORBA Components
- Can navigate among the interfaces of a CORBA component



## The CORBA 2.0/2.1 Server

- May be able to control resource usage by activating object instances in a few different ways
- May provide a way to associate persistent state with an object



## The CORBA 3.0 Server

- Supports many flexible patterns of resource control for servants with differing qualities of transience or persistence
- Associates an ObjectID with the Object Reference for retrieval of Persistent State
- Can negotiate Quality of Service with the client and the network
- Can deal with Firewalls
- Can be specialized for large, scalable, fault-tolerant enterprise and Internet servers, smaller embedded systems, and real-time applications

And, a CORBA Components server

- Integrates Persistence, Transactionality, Security, and Event Handling
- Simplifies programming with new declarative languages for server-side functions and characteristics
- Provides class-like methods including instance creation and destruction, queries across the extent, and client-visible identity
- Allows the client to navigate among the multiple interfaces born by a component
- Integrates with Enterprise JavaBeans



## Java-to-IDL Reverse Mapping

- Not just the usual IDL to Java mapping
  - but also -
- a Java to IDL mapping:
  - Write a Java server object
  - automatically generate the interface in OMG IDL
  - and access its methods from a CORBA client, written in any language

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

7



## Java and CORBA

- Unify Java RMI & CORBA IIOP
- Java Programmers gain -
  - Distribution using Java RMI
    - no IDL to learn
    - No IDL advantages, either (asynch, QoS, etc)
  - Cross-Language Invocations
  - Standard Protocol (IIOP)
- Java Objects become CORBA-accessible
  - No longer a single-language distributed environment

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

8



## Mapping Details

- Most Java language features map
  - Special handling for non-CORBA features
    - overloading
    - Name capitalization collisions
    - etc
- CORBA Objects-by-Value a necessity
- But no *Helper* or *Holder* classes

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

9



## Specific Goals

- Strict subset of RMI
- Avoid creating an incompatible dialect
- Needed to add some RMI extensions
- As large a subset as practicable
- There are a few restrictions
- Support all "reasonable" RMI programs
- Well-integrated with OBV specification
- Support for Java serialization
- Support for narrowing

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

10



## Non-Goals

- **Not the inverse of the IDL to Java mapping**
  - Round trips do *not* restore
- **No round-trip mapping proposed**
  - IDL types recognized and marshaled using forward mapping rules
- **Not all of IDL supported**
  - e.g., no support for struct, enum, union
  - No Helper or Holder classes - no need!
- **No distributed garbage collection**
  - subject of possible future RFP

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

11



## Mapping for Primitive Types

<u>Java</u>	<u>IDL</u>
void	void
boolean	boolean
char	wchar
byte	octet
short	short
int	long
long	long long
float	float
double	double

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

12



## Mapping for Remote Interfaces

- These extend `java.rmi.Remote`
- Mapped to IDL interfaces
- Java inheritance preserved in IDL
- Methods mapped to operations
- JavaBeans properties mapped to attributes
- Compile-time constants mapped



## Mapping Exceptions

- Java exceptions are objects
- Map to IDL exception containing a value type
- Java throws clause maps to IDL raises
- RMI RemoteExceptions correspond to CORBA Standard Exceptions



## Also, look for Mappings of...

- Java Strings to boxed wstring
- Packages to Modules
- Value Classes to valuetypes
- Arrays to boxed sequences
- Names with mangling if necessary
- Handling of non-conforming types
- java.rmi.remote to CORBA::Object
- Handling of narrow



## Firewall Specification

- CORBA through Firewalls
- Two Major Types Treated:
  - Transport-Level Firewalls:
    - TCP Firewalls
    - SOCKS Firewalls
  - Application-Level Firewall:
    - GIOP Proxy Firewalls
- And an alternative mechanism
  - Bi-Directional GIOP
  - For Callbacks Only





## Two Goals for CORBA Firewalls

- Handle CORBA Communication better when it *is* supposed to get through, and
- Use Firewall Techniques to Control when it's *not* supposed to get through.
- Put Firewall Information into the IOR
- TCP Techniques so client can get out
- Socks Proxies (e.g.) to get to the Server
- or a GIOP (application level) Proxy

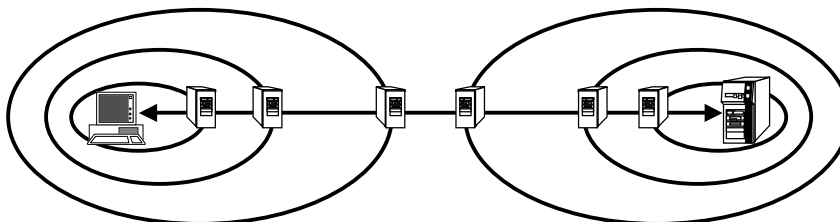
4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

17



## Firewall Configuration



- **Client Side:**
  - Restrict Outbound (Server) Access
  - Protect Clients

- **Server Side:**
  - Protect Servers
  - Restrict inbound (Client) Access

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

18



## Transport-Level Firewalls

- Transport only; no content or application dependence
- TCP Firewall would allow, e.g.,
  - FTP, HTTP, IIOP, Telnet
- Restricted by Host and Port
- Configure to forward traffic to a particular Host and Port
- OMG's IANA Well-Known Port Numbers:

IIOP: 683  
IIOP/SSL: 684

4/11/01

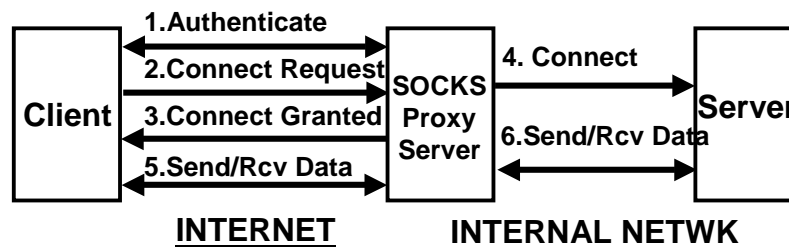
Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

19



## SOCKS Firewalls

- Another Transport Firewall
- Authentication step first; communication follows
- Link ORB with SOCKSified TCP Library



4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

20



## Application-Level Firewall

- **Works on a particular application level protocol:**
  - FTP, IIOP, etc
- **Allows/Restricts Access to Particular Resources**
  - even though they may be on the same host and port
- **GIOP Proxy is an Application Level Firewall**



## Bi-Directional GIOP & Callbacks

- **GIOP is a one-directional protocol**
  - Client must open the connection
- **What about Callbacks?**
- **Best option: Put Firewall Info in Callback Object's IOR**
  - but sometimes you can't
  - e.g. if you didn't write the application
- **So, Use Bi-Directional GIOP**



## Bi-Directional GIOP

- Governed by a new POA Policy
- Both Ends Must Enable Bi-Dir GIOP
- Callback Object's IOR Must be Sent to the Server over the Connection
- Then the Server can Call Back to the Object over the Connection in the Reverse Direction



## CORBA URLs

- Defined by the Interoperable Naming Service
- Two formats:
  - corbaloc
  - corbaname
- Resolved by `string_to_object`



## CORBALOC

- Gets an Object Reference for a Standard Service at a Remote Location
- Format:

`corbaloc:iiop:1.2@MyVendor.com:2089/Prod/NameService`

- Some fields may be defaulted:

`corbaloc::MyVendor.com/Prod/NameService`



## CORBANAME

- Invokes the Naming Service at a Remote Location
- Format:

`corbaname::MyVendor.com#Pub/Menswear/SaleCatalog.obj`

- Same Defaults as iioploc



## Quality of Service Control

- **Asynchronous Invocation and Messaging**
- **Invocation QoS Control**
- **Realtime, Minimum, Fault-Tolerant CORBA**



## Asynchronous/Messaging RFP

- **Four extensions to the Architecture.**  
**Clients will be able to:**
  - make requests which do not block;
  - make requests which do not complete during the lifetime of the client;
  - control QoS associated with a request;
  - control ordering of multiple requests.



## Two Distinct Models

- **A *Programming* Model:**
  - Choose Synchronous or Asynchronous
  - If Asynch, choose Callback or Polling
- **A *Communications* Model:**
  - Optionally, Install and Configure Routers and Messaging
  - If no Routers, all Comms go Direct
  - With Routers, Store-and-Forward and optionally Time-Independent Comms
  - And more QoS Options



## Programming I: Callback

- Operation name and signature become  
void sendc\_<opname> ( ... )
- Callback object reference, in and inout parameters follow – no out parameters
- Only return value, inout, out parameters returned via callback
- Exceptions are raised in the Callback (which is a object, remember)
- Callback object can be *anywhere!*



## Programming II: Polling

- Operation name and signature become  
Poller\_return\_type sendp\_<opName> ( . . . )
- Only in and inout parameters follow
- Only return value, inout, out parameters, exceptions return with Poller
- Poller is constrained to Calling Process, unless STORE\_AND\_FORWARD



## Communications Architecture

- Synch/Asynch is a *Client-Side* Issue
- Server-Side Invocation Model Unchanged
  - Server is invoked synchronously
  - over a maintained connection
- Something Changes from
  - Client-Side Asynchronous Invocation, to
  - Server-End Synchronous Invocation
- This is done by either
  - Client's ORB, if no Routers, or
  - A Router, if they're available
- Asynch Invocations of Existing Servers

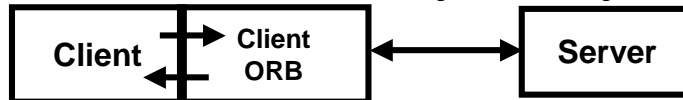




## Synchronous Invocation

1) Client makes Synchronous Invocation and waits while call blocks

2) Client's ORB makes synchronous Invocation over Maintained Connection using Serialized Messages



3) Client receives response from ORB when it returns

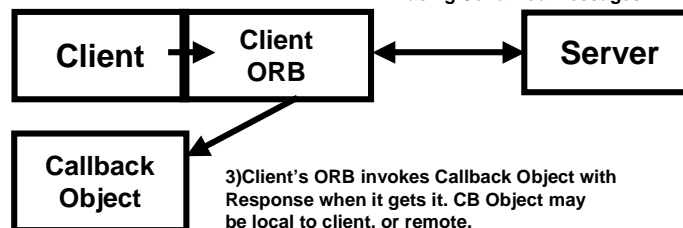
- Client makes Synchronous Call and Waits
- Client Call Blocks during processing
- Out and Back over TCP Connection



## Asynchronous No-Router Callback Communications

1) Client makes Asynchronous call and *doesn't wait* for response

2) Client's ORB makes Synchronous Invocation over Maintained Connection using Serialized Messages

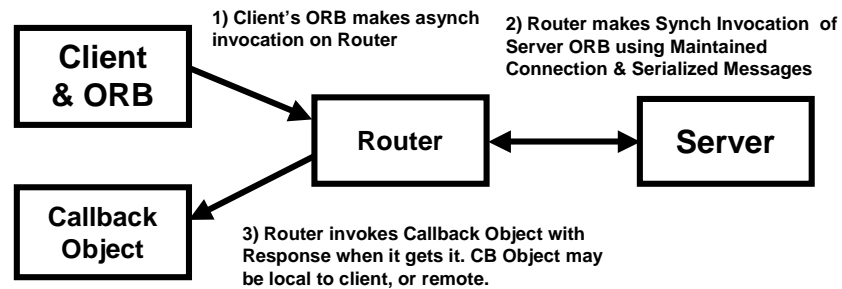


3) Client's ORB invokes Callback Object with Response when it gets it. CB Object may be local to client, or remote.

- Client makes Asynchronous Call & *Doesn't Wait*
- Client's ORB invokes Server Synchronously
- Result Returns, Client's ORB calls Callback Object



## Asynchronous Router Callback Communications



- Roles in Callback Invocation



## Routers

- Store-and-Forward Software Agents
- Some will be co-located with CORBA processes
- Others will be separate, out on the enterprise network
- Optionally, Routers may have Persistent Storage



## Router Functions

- **Routers Serve Three Basic Functions:**
  - Store and Forward Invocations and Replies
  - Routers co-located with Clients permit Disconnected Operation
  - Routers co-located with Servers Queue and Order Invocations for delivery according to QoS Settings in Effect

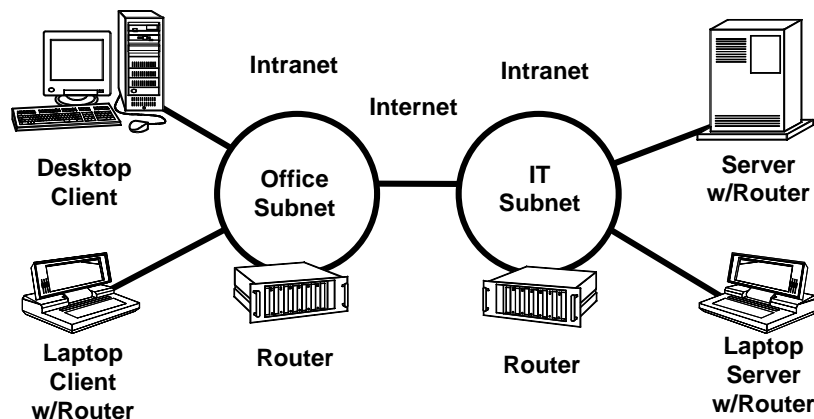
4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

37



## Network with Routers



4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

38



## Quality-of-Service Control

- Optimize Resource Usage when your System is Stressed
- QoS Settings are CORBA *Policies*
- May apply to Synchronous as well as Asynchronous Invocations
- New Concept: *Binding*
  - Rebind Policy:
    - TRANSPARENT, NO\_REBIND, NO\_RECONNECT
- Without Routers, Timing Control Only
- With Routers, Much More QoS Control

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

39



## Timing QoS Settings

- Start Time, End Time
  - For Request and Reply Separately
- Round Trip Timeout
  - Max time for the invocation
  - Request Only, or Round-Trip Time
  - Time out, and Request is Cancelled, or Response is Dropped
  - Doesn't clutter your network with Obsolete Packets

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

40



## Routing QoS Settings

- **Three Possible Values:**
  - **Routing=ROUTE\_NONE**
    - Must allow this if you don't have Routers
  - **Routing=ROUTE\_FORWARD**
    - The highest possible Routing QoS if your Routers do not have persistent storage
  - **Routing=ROUTE\_STORE\_AND\_FORWARD**
    - The Best - Use this if your Routers have Persistent Storage
    - Transaction-Quality Network Transmission
    - Time-Independent Invocation
    - Pollers function Out of Client Process
- **You can set Routing Policy to a Range**



## Ordering Invocations

- **Determines the Order that Invocations are Forwarded by a Router**
- **You can allow one, several, or all of these values**
- **Values except ANY require Routers, and**
- **AMI-aware Clients send Invocations to Routers for Queuing**
- **ANY** - the client doesn't specify in what order its requests are forwarded.
- **TEMPORAL** - requests that invocations be forwarded in the order in which they were issued. **TEMPORAL** is the default.
- **PRIORITY** - requests that invocations be forwarded based on the priority assigned in the QoS structure
- **DEADLINE** - requests that invocations whose time\_to\_live is about to expire are moved to the front of the queue.



## Unshared Transactions

- **Transactions Usable over TII**
- **Not your usual Distributed Transaction Model - Examples:**
  - Planning a vacation: Plane Tix, Car Rental, Hotel
  - Bank Teller: Memo Posting
- **Every individual step is Transactional**
  - Even the Network Transport is Assured
- **But if a step fails, the Application has to Undo - there's no Rollback**



## Real-Time CORBA

- **Just adopted; Part of CORBA 3**
- **Two Main Goals Areas:**
  - End-to-End Predictable Behavior
  - Resource Management
- **Both Hard and Soft (Statistical) Real-Time**
- **Relies on Underlying Real-Time OS -**
  - Posix or otherwise
  - But does not standardize the layer from CORBA to the RT OS



## RT CORBA Mechanisms

- **Two Priority Models:**
  - Client Propagated or Server Declared
- **Mutexes and Priority Inheritance**
- **Threadpools**
- **Priority Banded Connections**
- **Non-Multiplexed Connections**
- **Client and Server Protocol Configuration**
- **Invocation Timeouts**
- **Scheduling Service**

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

45



## Minimum CORBA

- **Targeted at Embedded Systems**
- **Fixed, Predictable Systems**
  - Do not need dynamic aspects of CORBA
- **Eliminates -**
  - DII, DSI, Dynamic Any (DynAny)
  - Interface Repository
  - Some POA Features and Policies
  - Protocols except GIOP/IOP
- **Supports *ALL* OMG IDL Types**

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

46



## Fault-Tolerant CORBA

- Completed last year
- Fault Tolerance based on
  - Entity Redundancy – Replication of Objects
  - Fault Detection
  - Recovery
- Fault Tolerance Strategies:
  - Request Re-try
  - Redirection to Alternative Server
  - Passive and Active Replication

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

47



## Objects-by-Value (OBV)

- Architecture enhancement allowing Passing of CORBA objects *by value*
- Specification defines a new IDL type, valuetype
- A “struct with functionality”
- Copy semantics; no synchronization
- Actually issued in CORBA 2.3

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

48





## Objects-by-Value Goals

- Expose complex state in a language-independent manner
- Consistent semantics across languages
- Provide natural support for Java and C++
- Minimize impact on IDL and GIOP
- Guarantee interoperability and portability
- Enable/Facilitate RMI over IIOP



## Adds to Power of IDL Structs

- Arbitrary recursive value type definitions with sharing semantics
- Define lists, trees, lattices, arbitrary graphs (including cyclic)
- Single inheritance from other value types
- Multiple inheritance from interfaces and abstract values



## Parameter Passing - Sharing

- **Values types support sharing and null semantics**
  - can describe arbitrary graphs
  - sharable across and within other instances
  - can be null
- **Sharing is preserved across method invocations**
  - Graph reconstructed in receiving context is structurally isomorphic
- **In the Receiving context, the object is a new, separate identity**
  - local identity, in the implementation's programming language



## When you Pass a Value Type in a CORBA invocation

- The “Sending Context” marshals the state (data) and sends it over the wire
- A new copy must be created (by a factory) at the receiving end - no magic!
- The state is unmarshaled into the copy
- The factory may be local, or download an implementation over the wire



## When you Invoke an Operation on a Value Type

- It's a local, programming-language object
- Invocations are local, not mediated by an ORB
- Because you're not passing it as a parameter of a CORBA invocation



## Value Box

- Package a single data member as a *valuetype*
  - No methods or inheritance
- Simple or complex data
- Great for strings and sequences -
  - Only pass one copy when you use the same data as two parameters of a call
- Transmission of Nulls



## Abstract Interface Type

- Lets you choose between Interface and Value Types at Runtime

```
interface Example {  
    void foo(in MyType mydata);  
};
```

- Suppose some MyTypes are CORBA objects, and some are values
- Then declare

```
abstract interface MyType {  
    void bar(in long avalue);  
};
```



## CORBA Component Model (CCM)

- CORBA is great for building Enterprise and Internet applications
- But, of the thousands of CORBA usage patterns, a few stand out
- CCM packages up these successful patterns, including
  - POA servant management
  - Transactions and Persistence
  - Security
  - Event Handling
  - Configuration
  - Interface Connection and Assembly
- This speeds and simplifies application building, and ensures success



## What this Means to You:

- **CCM Applications are very compact:**
  - Much less code required
  - Developer code devotes to the business problem
- **CCM Applications are easier to code:**
  - Only successful patterns included in CCM
  - Much code is generated automatically
  - In declarative languages derived from IDL
- **CCM Applications are Modular:**
  - Components assemble into applications
  - Combine commercial, in-house, and custom components
  - Standardized Assembly, Packaging, Distribution, Deployment
- **CCM Applications scale to Internet and Enterprise**
  - Patterns known to scale well
  - Vendor experience used to build and tune products
  - Resource handling coded automatically in the best way

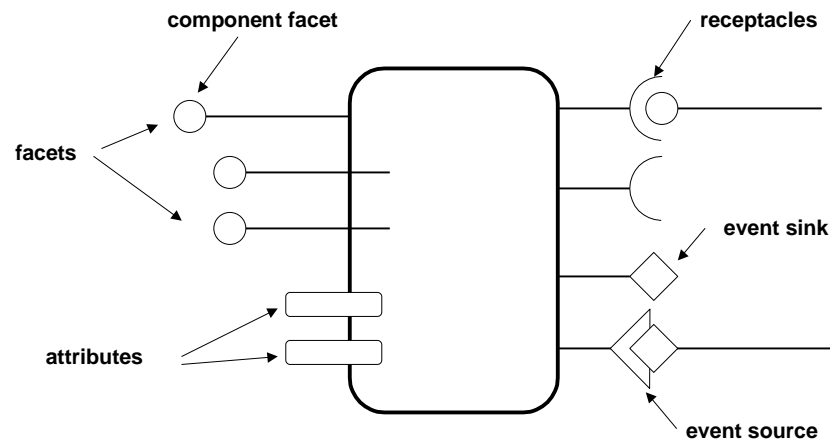


## What is CORBAcomponents?

- **A server-side component model for building and deploying CORBA applications**
  - An architecture for defining server components and their interactions
  - A packaging technology for deploying binary, multi-lingual, executables
  - A container framework for injecting security, transactions, events, and persistence
  - A CORBA container for Enterprise JavaBeans



## Component Model



4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

59



## For Business Programmers

- **A Development Environment Designed for their Strengths**
- **Finally, a packaged CORBA Computing Infrastructure:**
  - Persistence, Transactionality, Security, Event Handling, and more, abstracted to a higher level than the CORBA services
- **Business Programmers deal with Business Level Functionality**
- **Even the Component Archive and Distribution Format is Standardized**

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

60



## Component Basics

- ***Extensions to the CORBA Core Object Model*** add to the architecture:
  - Components, bearing more than one Interface
  - Component Assemblies - of more than one Component
- ***Extensions to OMG IDL***
  - define what components can do
- ***A new language, CIDL - Component Implementation Definition Language***
  - couples servant implementations to the container, especially for Persistence

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

61



## CCM Container Model

- **Introduces System Services into the CORBA Programming and Runtime Environment**
  - CCM Container is a specialized POA
  - plus pre-packaged CORBA services:
    - Persistence
    - Transactions
    - Security
    - Notification (i.e. Events) Channels

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

62



## Container Services

- *Navigation* by client among interfaces supported by all of its components
- Container connects provided and required interfaces of its components
- *Event Channels* - Connects and transmits supplied and consumed events
- Services provided through *locality-constrained interfaces*

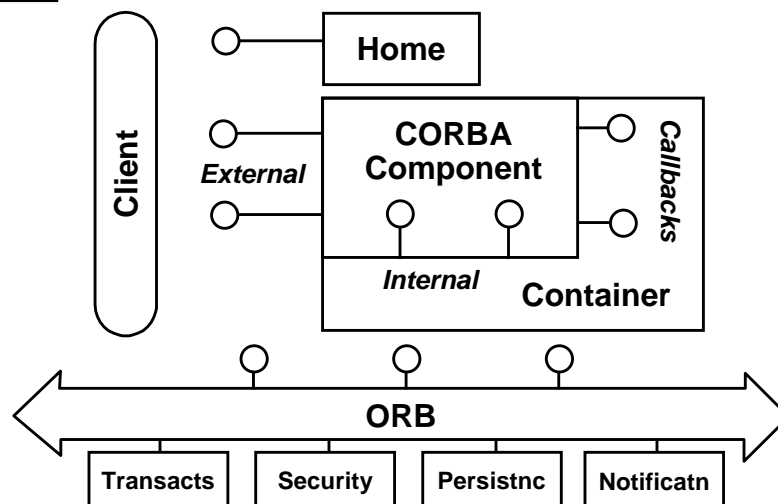
4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

63



## CCM Container Architecture



4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

64





## CORBA gets some Class!

- ***Component Home*** adds Class-like operations to Components
- For *all* components:
  - Lifecycle operations: Create, Remove
  - Access to the Extent
  - For queries or application-level operations
- For *entity* components, add:
  - Find by Primary Key
  - Operation and Key visible to Client

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

65



## Assembling a CCM Application

- Components *Assemble* into a CCM Application
- Connect up:
  - Provided and Consumed Events
  - Provided and Used Interfaces
- Do this in Layer after Layer
- Then Configure as an Application

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

66



## Deploying Components

- ***Distribution Format*** for component executables and configuration files
- **Configuration of components:**
  - Defaults with Distribution, and
  - Using a GUI Tool at Deployment
- **Configuration files are XML-Based**
- **Enables a CORBA Component Marketplace**

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

67



## Component Run-Time

- **Container Interfaces for Managing Activation and Passivation**
  - Based on POA Policies
- **Policies for Managing Servant Lifetimes**
  - Four Categories of Components
  - Optimize Resource Usage

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

68



## CORBAcomponents Container

- A Container includes one or more specialized POAs
- Implemented by a Component-Aware ORB
- Provides Framework for Server Applications
  - Provides External interfaces to clients (factories and method invocation)
  - Provides Locality-constrained interfaces (valuetypes) for Systems Services (transactions, security, events, persistence)
  - Provides Callbacks for instance management (activation and passivation)

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

69



## Client View

- Component-aware and Component-unaware:
  - AWARE: Navigate among component interfaces using new container services
  - UNAWARE: Invoke operations on individual component CORBA interfaces
- Three categories of components seen as two design patterns
  - Factory - Client finds factory to create new instance
  - Lookup - Client uses naming (or trader) to find existing component instance
- Optional demarcation of transactions
- Establishes initial security credentials
- Invokes business methods

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

70



## Server View

- **Seven categories of containers:**
  - Service (no memory from call to call)
  - Session (like EJB stateless Session Bean)
  - Process (server side application persistence)
  - Entity (like EJB Entity Bean with PSS-based automatic persistence)
  - plus Two EJB containers, and an Empty container
- **Container APIs for system services**
  - Classified into Two Container API Types
- **Callback APIs for instance management**

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

71



## Component Categories

<u>Component Category</u>	<u>Container Impl Type</u>	<u>Container Type</u>	<u>External Type</u>
Service	Stateless	Transient	Keyless
Session	Convrsatnl	Transient	Keyless
Process	Durable	Persistent	Keyless
Entity	Durable	Persistent	Keyful

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

72



## Service Components

- **Service component characteristics:**
  - Behavior, single invocation per instance
  - No state, ever!
  - No identity
- **Suggested Uses:**
  - Single operation per call
  - Single Transactions, one-time operations
  - Equivalent to a stateless Session EJB
- **Servant Lifetime Policy:**
  - Method (only)
  - Factory design pattern (only)



## Session Components

- **Session component characteristics:**
  - Behavior
  - Transient (conversational) state
  - Non-persistent identity
- **Suggested Uses:**
  - Repeated calls over short period
  - Iterators, for example
- **Servant Lifetime Policies:**
  - Method, Transaction, Component, Container
  - Factory design pattern (only)



## Process Components

- **Process component characteristics:**
  - behavior which may be transactional
  - explicitly declared state managed by container
  - Identity managed by client and implementation
- **Suggested Uses:**
  - Model Business Process w Beginning & End
  - Applying for a Loan, Shopping Cart
- **Servant Lifetime Policies:**
  - Method, Transaction, Component, Container
  - Factory design pattern (only)



## Entity Components

- **Entity component characteristics:**
  - Behavior which may be transactional
  - explicitly declared state managed by container
  - identity visible to client, managed by container
- **Suggested Uses:**
  - Non-Transient Business Entities
  - Customers, Accounts, etc.
- **Servant Lifetime Policies:**
  - Method, Transaction, Component, Container
  - Factory or Finder Design Patterns



## Container Services

- Reference creation
- Instance creation
- Transactions
- Security
- Events
- Persistence
- Servant lifetimes



## Component Assemblies

- A component assembly is a pattern or a template for a deployed set of interconnected components.
- Described by a component descriptor in terms of component files, partitioning, and connections.
- May be deployed as is or used in another assembly.



## Assembly Package

- Like a component package, an assembly package is a ZIP archive file.
- It contains a component assembly descriptor, component archive files, and property files.



## Assembly Descriptor

- The assembly descriptor describes an initial deployable configuration of homes, components, and connections.





## A Day in the Life of a Component

- A component is specified.
- A component is implemented.
- A component is packaged.
- A component may be assembled with other components (usually in a design tool).
- Components and assemblies are deployed.



## Component Builder Roles

Role (Human)	Uses (Tools)	Produces (Artifacts)
Component Creator	Text Editor IDL Compiler IDE Language Compiler Component Packaging Tool	Component Archive File - Implementation Files - Component Descriptor - Property State File
Application Assembler	Visual Assembly Tool  (or text editor and packaging tool)	Component Assembly Descriptor - or - Component Assembly Archive - Assembly Descriptor - Component Archives
System Deployer	Deployment Tool	Deployed Assembly and Components



## EJB Integration

- An EJB can look like a CCM Component to another CCM Component
- A CCM Component can look like an EJB to another EJB
- This Allows an Application to use a Combination of EJBs and Components

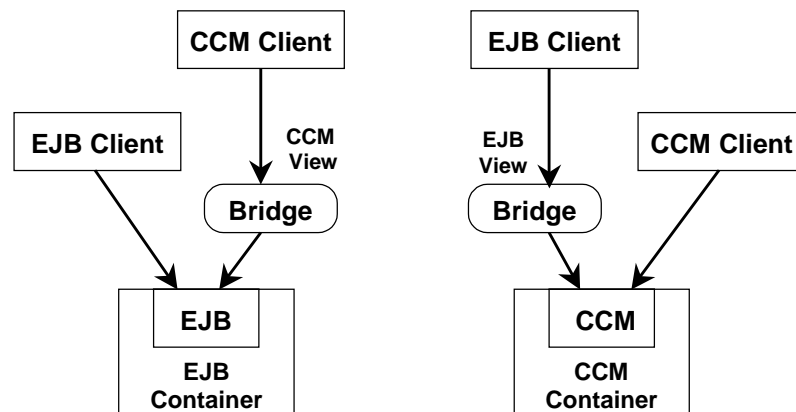
4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

83



## Bridges Integrate EJBs



### Interoperation in a Mixed Environment

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

84



## Two Levels of CCMs

- **Enable EJB Interoperability**
- **Basic Components:**
  - Features nearly identical to EJB 1.1
  - Transactions, Security, Simple Persistence, Single Thread
- **Extended Components:**
  - Additional Features
  - Multiple Interfaces & Navigation, Multiple Segments & Segmented Persistence, Event Handling, Multi-Threading, Component-selected Key in Object Reference

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

85



## CIDL

- **Component Implementation Definition Language**
- **Defines Component Behavior and State**
- **Behavior Linkage Not Defined Yet**
- **State Definition via Container Coupling to Persistence Services**
- **CIDL Generates some Implementation Code**

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

86



## CORBA Scripting Language

- A Full Mapping of IDL to Scripting Languages
- Interpreted, Higher-Level Access to CORBA and CCM
- Accessible to Business Users
- Assemble CORBA and CCM Applications Dynamically:
  - Generic Functionality in Purchased Components
  - Specific Business Rules in Site-Tailored Scripts

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

87



## Why CORBA Scripting?

**“A typical component-based application consists of a number of components constructed with traditional programming tools, embodying engines, transformations and services. These components are wired together with scripts which contain the application decision logic. This permits less skilled programmers, and sometimes end-users to modify to logic and decision making of the program to suit their needs.”**

4/11/01

Copyright © 2001 Object Management Group  
NOT LICENSED FOR PRESENTATION

88



## How Scripting Will Work

- Individual Specifications Map CORBA to Particular Scripting Languages
- Already Done:
  - IDLscript
  - Python
- Possible Future Candidates:
  - Tcl
  - JavaScript
  - Perl
  - Others



## CORBA 3 Summary

- The “Ease of Use” Release?
  - More Features and Flexibility
- The Enterprise Release?
  - Components, Scripting
  - Asynchronous Invocations
  - Quality of Service Control
- The Scope Release?
  - Minimum, Realtime, Fault Tolerant CORBA
- All of the above!