



IBM Software Group

WebSphere Application Server 6.0 for z/OS

GSE – 8 juni 2005



ON DEMAND BUSINESS™

© 2005 IBM Corporation

Agenda

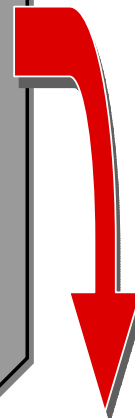
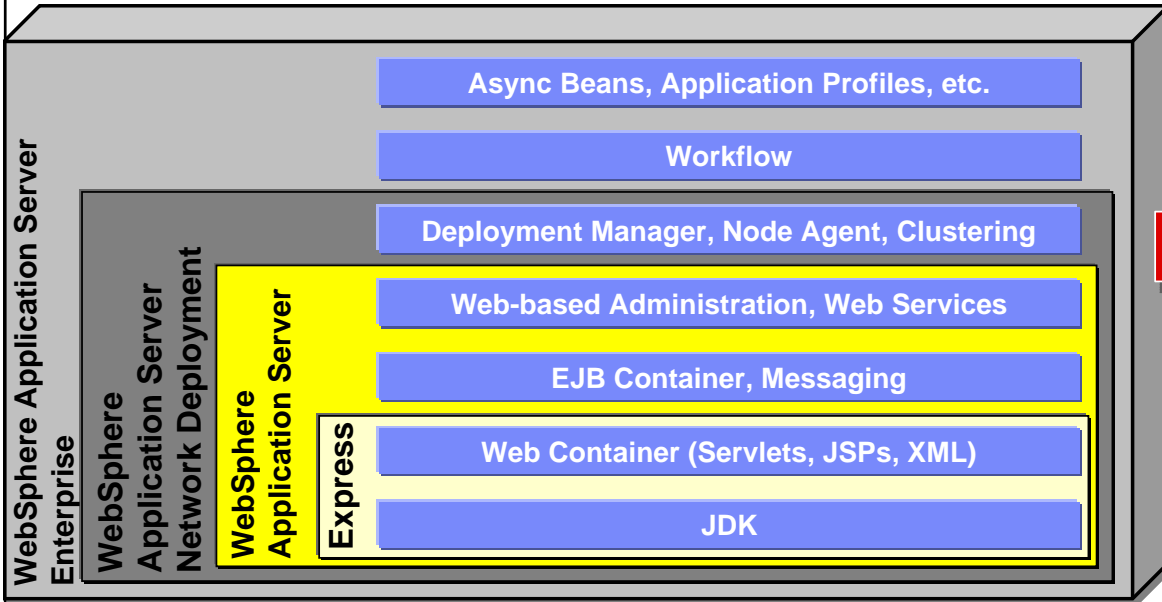
- Platform Enablement
- Standards Based Architecture
- Ease of Use
- Enterprise Class Deployment
- WAS for zOS Transaction Capabilities

Agenda

→ Platform Enablement → Packaging

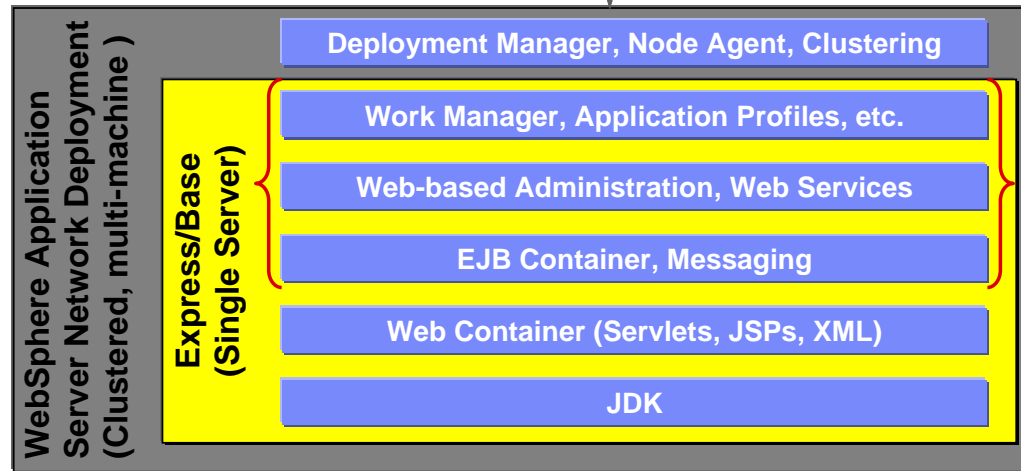
- Standards Based Architecture
- Ease of Use
- Enterprise Class Deployment





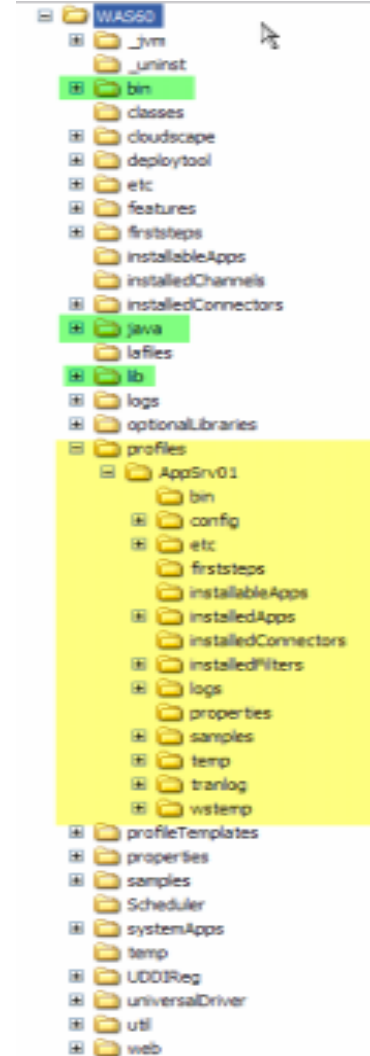
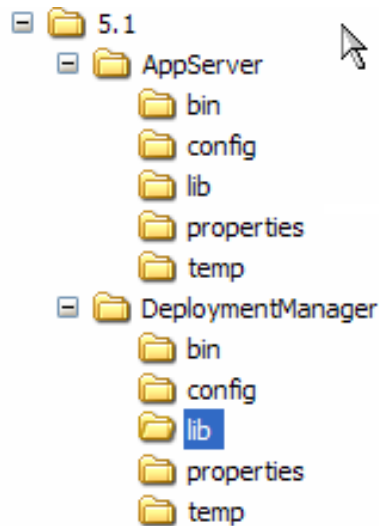
Packaging changes from WebSphere 5.x to 6.0

- Single Install Image for Application Server and Deployment Manager
- Product Binaries Separated from Configuration



Packaging and Product Install

- Single Install Image for Application Server and Deployment Manager
- Product Binaries Separated from Configuration
- Greatly Reduced Number of Product CDs
- Support for multiple product profiles from single binary installation





Server Profiles

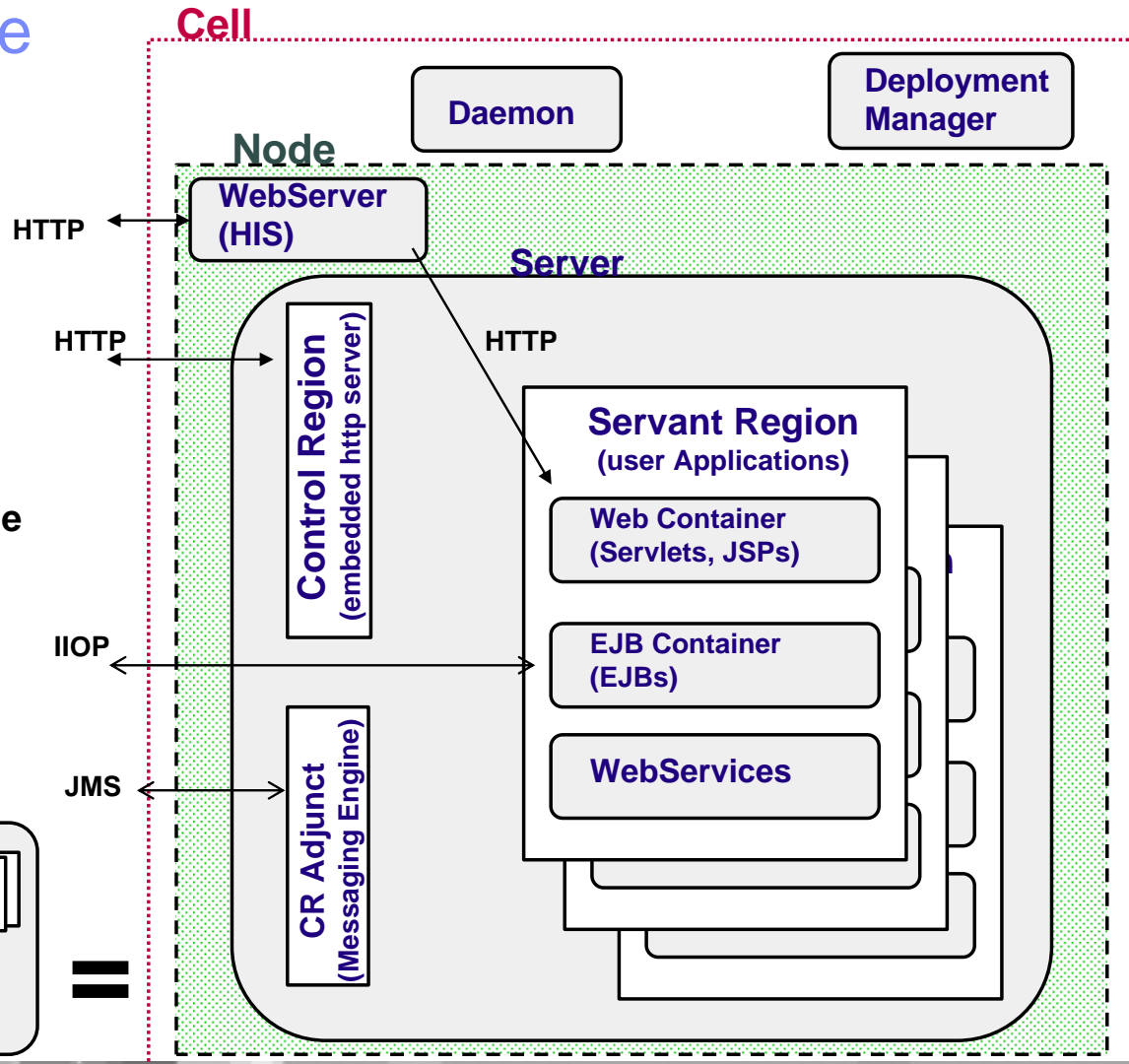
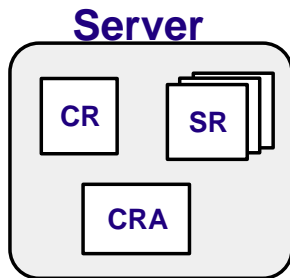
- Server Profiles is a tool to create independent configurations of WebSphere Application Server from a set of shared binaries on a machine
- This is an enhancement to the wsinstance support in WAS 5.x
- Changes for WebSphere Application Server 6.0
 - ▶ The default product configuration will itself be a profile
 - ▶ Support for profile templates that are used to create the profiles
 - ▶ Enhanced profile create/delete/list commands
 - ▶ Support for profile import/export allowing movement of an profile from one environment to another
- Profile Templates that ship with WAS 6.0
 - ▶ Standalone Server – similar to WAS 5.x default configuration
 - ▶ Managed Node – An empty node for the initial install of a production server
 - ▶ Deployment Manager – A profile that defines the Deployment Manager for a Cell



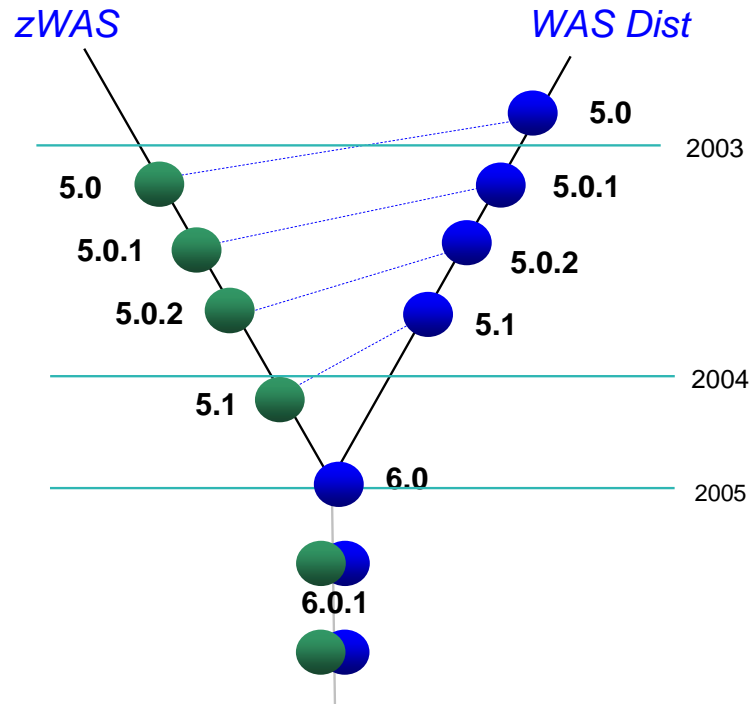
System Structure

- Cell
 - ▶ Node
 - Server
 - Controller (CA)
 - Servants (SA)
 - Containers
 - Adjunct (CRA)
 - Msging engine for SIB
 - WebServer (HIS)

Key:



Common code



Agenda

- Platform Enablement

- ➔ **Standards Based Architecture**

 - ➔ Standards

 - ➔ Web Services

 - ➔ Programming Model Extensions

 - ➔ Service Data Objects

- Ease of Use

- Enterprise Class Deployment





J2EE 1.4 Highlights

Web Services and XML support

- **Standards / Portability** - XML Schema definitions for all deployment descriptors
- **JAX-P 1.2** - New properties for XML parsers
- **JAX-R** - XML registry API
- **JAX-RPC** - APIs for representing WSDL-based services as RPCs in Java (and vice-versa)
- **JSR 109** - Web services programming and deployment model
- **SAAJ 1.1** - SOAP Attachments API for Java

Messaging

- **EJB 2.1**
 - Typed message beans (used for any inbound JCA including pluggable JMS provider)
 - Timer service Web service end-point support
- **JMS 1.1**
 - Unification of point-to-point and pub-sub interfaces

ISV Enablement

- **JMX 1.2 / JSR-077 (J2EE Management)**
 - Notification emitters, and standard patterns
 - Information model representing J2EE application server concepts
- **JSR-088 (J2EE Deployment)**
 - XML-based deployment interfaces for J2EE
- **JACC 1.0**
 - Java Authorization Contract with Containers
 - APIs for registering J2EE component authorization policies

Other

- **Servlet 2.4**
 - Extensible deployment descriptors
 - Request/response listeners
- **JSP 2.0**
 - Expression Language
 - Simple Tag Extension
- **EJB 2.1**
 - Timer Service
- **JDBC 3.0**
 - Meta data and cursor support
- **JavaMail 1.3** updates
- **J2CA 1.5**
 - In-bound connections
 - RA lifecycle support
 - Work manager (threads for resource adapters)



Web Services Standard Update

- **JAX-RPC (JSR-101) 1.1**
 - ▶ Additional type support
 - ▶ xsd:list
 - ▶ Fault support
 - ▶ Name collision rules
 - ▶ New APIs for creating Services
 - ▶ isUserInRole()
- **JSR-109 – WSEE 1.1**
 - ▶ Moved to J2EE 1.4 schema types
 - ▶ Migration of web services client DD moving to appropriate container DDs
 - ▶ Handlers support for EJBs
 - ▶ Service endpoint interface (SEI) is a peer to LI/RI
- **SAAJ 1.2**
 - ▶ APIs for manipulating SOAP XML messages
 - ▶ SAAJ infrastructure now extends DOM (easy to cast to DOM and use)
- **WS-Security**
 - ▶ WSS 1.0
- **WS-I Basic Profile 1.1**
 - ▶ Attachments support
- **WS-TX AT (Atomic Transactions)**
- **UDDI v3 support**
 - ▶ Includes both the registry implementation and the client API library
 - ▶ Client UDDI v3 API different than JAXR (exposes more native UDDI v3 functionality not available in JAXR)



Programming Model Extensions

- Most of the Programming Model Extensions moving into
 - ▶ WebSphere Application Server-Express ...or...
 - ▶ WebSphere Application Server - Network Deployment

- Moving to WebSphere Application Server-Express
 - ▶ Last Participant Support
 - ▶ Internationalization Service
 - ▶ WorkArea Service
 - ▶ ActivitySession Service
 - ▶ Extended JTA Support
 - ▶ Startup Beans
 - ▶ Asynchronous Beans (now called WorkManager)
 - ▶ Scheduler Service (now called Timer Service)
 - ▶ Object Pools
 - ▶ Dynamic Query
 - ▶ WSGW Filter Programming Model (with migration support)
 - ▶ Distributed Map
 - ▶ Application Profiling
 - ▶ CScope Service

- Moving to WebSphere Application Server - Network Deployment
 - ▶ Back-up Cluster Support





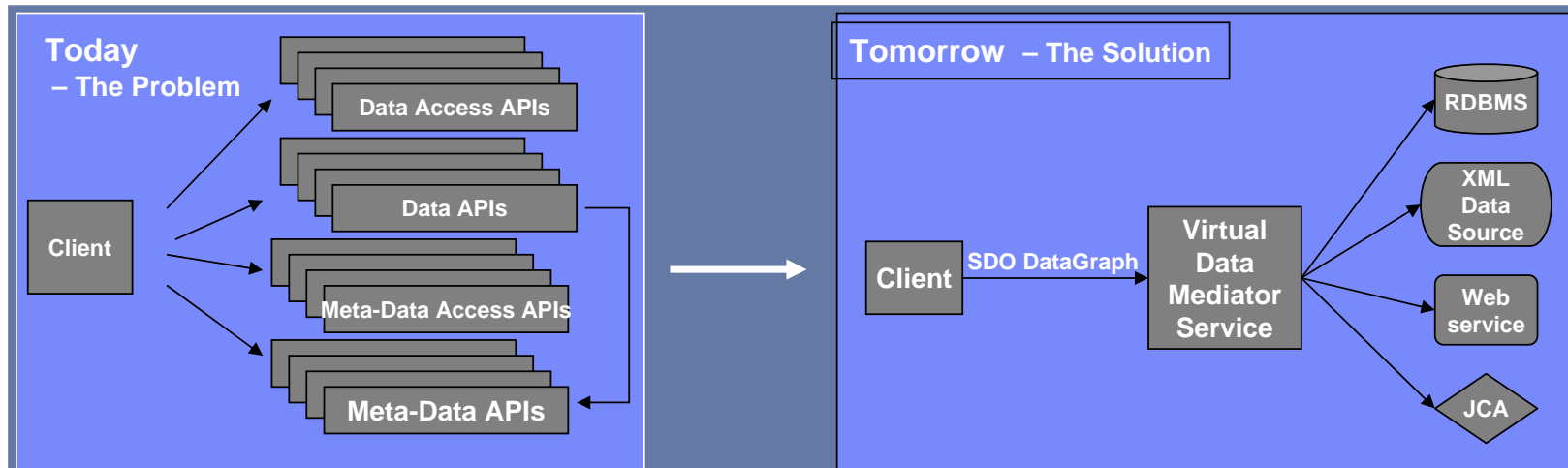
Service Data Objects

The Problem...

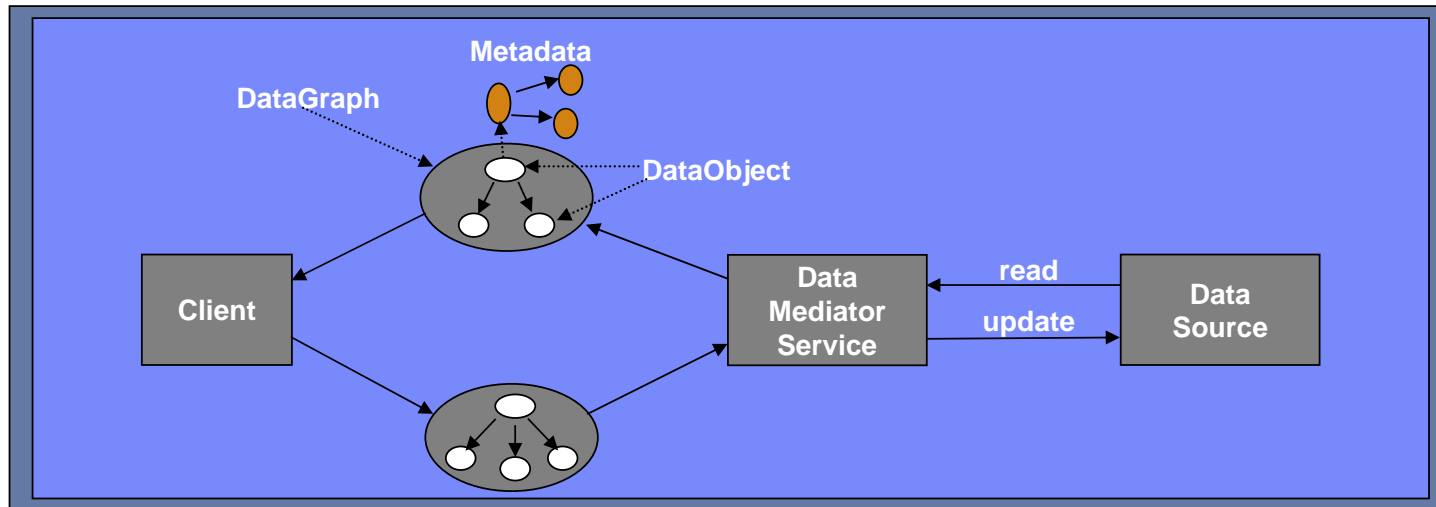
- Many different models and APIs for Data retrieval, Data representations, Meta-data retrieval, Meta-data representations, logic components
- No reasonable API available for “typed” XML data
- Lack of support for standard application patterns
 - ▶ Optimistic concurrency, pagination of large data-sets

The Solution...

- Service Data Objects (SDO) are a data programming architecture and API for the Java platform that unifies data programming across data source types, provides robust support for common application patterns, and enable applications, tools, and frameworks to more easily query, view, bind, update, and introspect data.



Service Data Objects



- Data is stored in a disconnected, source-independent format defined by the DataObject
 - ▶ Stored in a graph
 - ▶ Provides both dynamic loosely-typed and static strongly-typed interfaces to the data
 - getAddress()
 - getString("Address")
- DataGraph holds the root data object
 - ▶ Remembers change history
 - ▶ Provides Access to metadata about the DataObjects
- Data Mediator Service is responsible for filling graph of DataObjects from data source, updating data source from DataObject changes
- WAS 6.0 provides XML, JDBC, and EJB mediators





Java Server Faces

- Java Server Faces (JSF) Support
 - ▶ User interface framework for Java Web Applications
 - ▶ “Swing/AWT” Widgets for the Web
 - ▶ Makes it easy to construct a UI from a set of reusable UI components
 - ▶ Simplifies binding of application data to the UI
 - ▶ Helps manage UI state across server requests
 - ▶ Provides a simple model for wiring client-generated events to server-side application code
 - ▶ Allows custom UI components to be easily built and re-used
 - ▶ Can be easily tooled to allow true drag-and-drop Web UI creation
 - ▶ Supports multiple rendering kits for different client devices

Agenda

- Platform Enablement
- Standards Based Architecture
- ➔ **Ease of Use**
 - ➔ Rapid Deployment
 - ➔ Administration
- Enterprise Class Deployment





WebSphere Rapid Deployment

- **Annotation-based Programming**
 - ▶ Allow the developer to create and maintain a single artifact
 - ▶ Allow the developer to insert metadata into the source code of the application
 - ▶ Use the metadata to generate the additional artifacts required by the runtime that the developer does not need to be confronted with
 - ▶ Annotations are @javadoc tags (JSR-175 tags in the future)
 - ▶ Where appropriate, tags are compatible with XDoclet
 - ▶ Supports all of the WebSphere Programming Model
 - EJBs, Servlets, Web Services, SDOs, JMX, etc
- **Deployment Automation**
 - ▶ Enable automatic installation of applications and modules onto a running WebSphere Server
 - ▶ Supports both local and remote servers
 - ▶ Supports fine-grained application changes
 - ▶ Supports the concept of minimal application impact (affect the application in the minimum way possible to reflect the detected change)
- **Change Triggered Processing**
 - ▶ Drive processing operations based on the detection of change in artifacts of the application
 - ▶ Used to generate new application artifacts from existing artifacts
 - ▶ Used to drive deployment operations
 - ▶ Enables a “Hot Directory” concept for “file copy” and “Notepad” development and deployment
- **Support free-form projects**
- **Supports Studio tools integration and headless “Notepad” mode**
- **Ships with Application Server, AST, and Rational Tools**





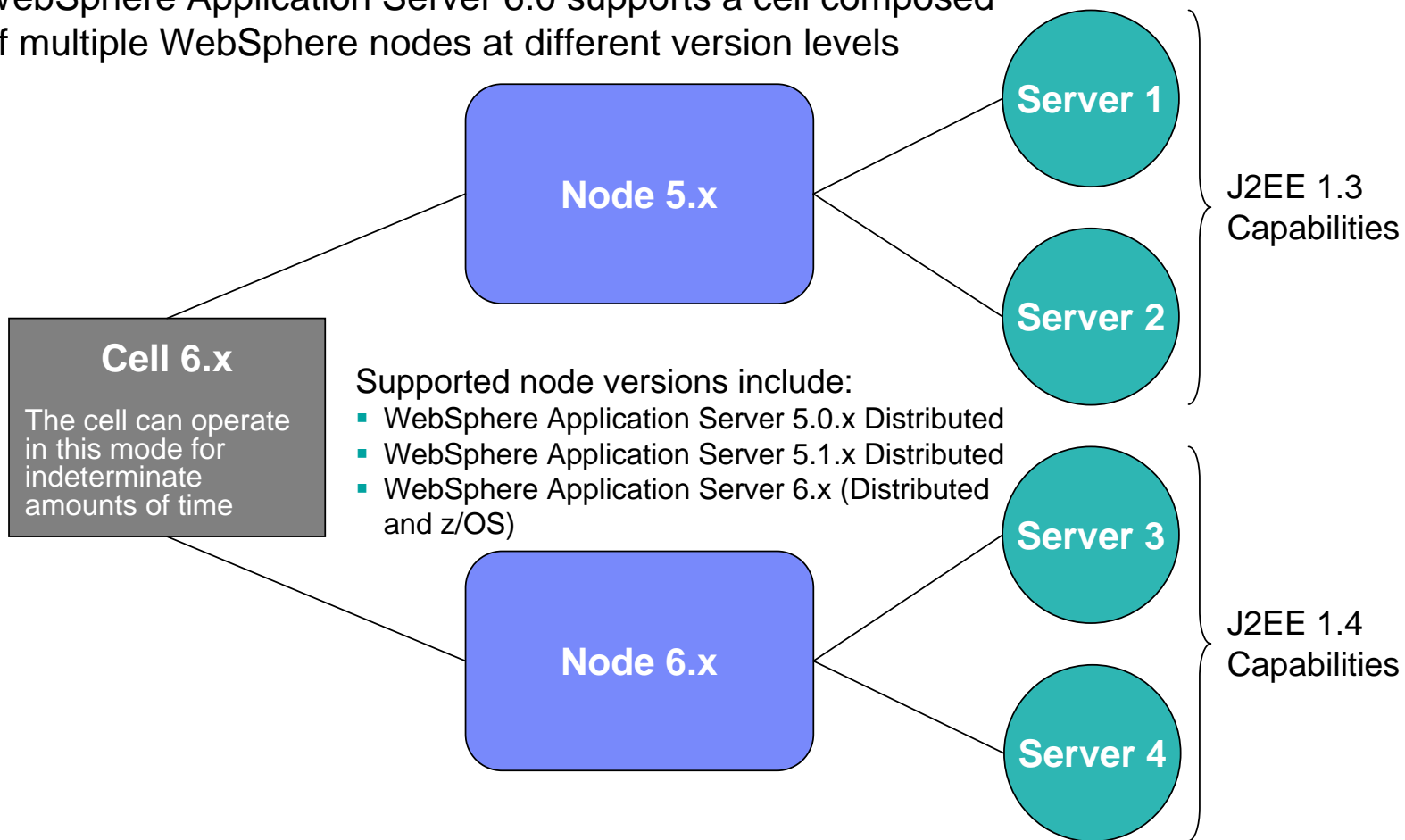
General Administration Changes

- **Enhanced EAR File**
 - ▶ Support an application installation process that includes ALL of the information needed to run the application on a server
 - J2EE EAR/Module
 - Bindings Information
 - Deployment.xml settings
 - Resource definitions
 - Per application security settings
 - ▶ Integrated with the WebSphere Test Environment in Studio and ATK
- **Fine-grained Application Update**
 - ▶ Ability to introduce small delta-changes to installed applications
 - ▶ Ability to add/remove(*) modules to installed applications
- **Support for extensible server types**
 - ▶ Web Server
 - ▶ Generic Server
- **System Applications**
 - ▶ Binaries are stored in the product binary folder
 - ▶ Simplifies PTF/Service Pack updates
 - ▶ Includes Admin Console, File Transfer App, etc.



Mixed Version Cells

WebSphere Application Server 6.0 supports a cell composed of multiple WebSphere nodes at different version levels



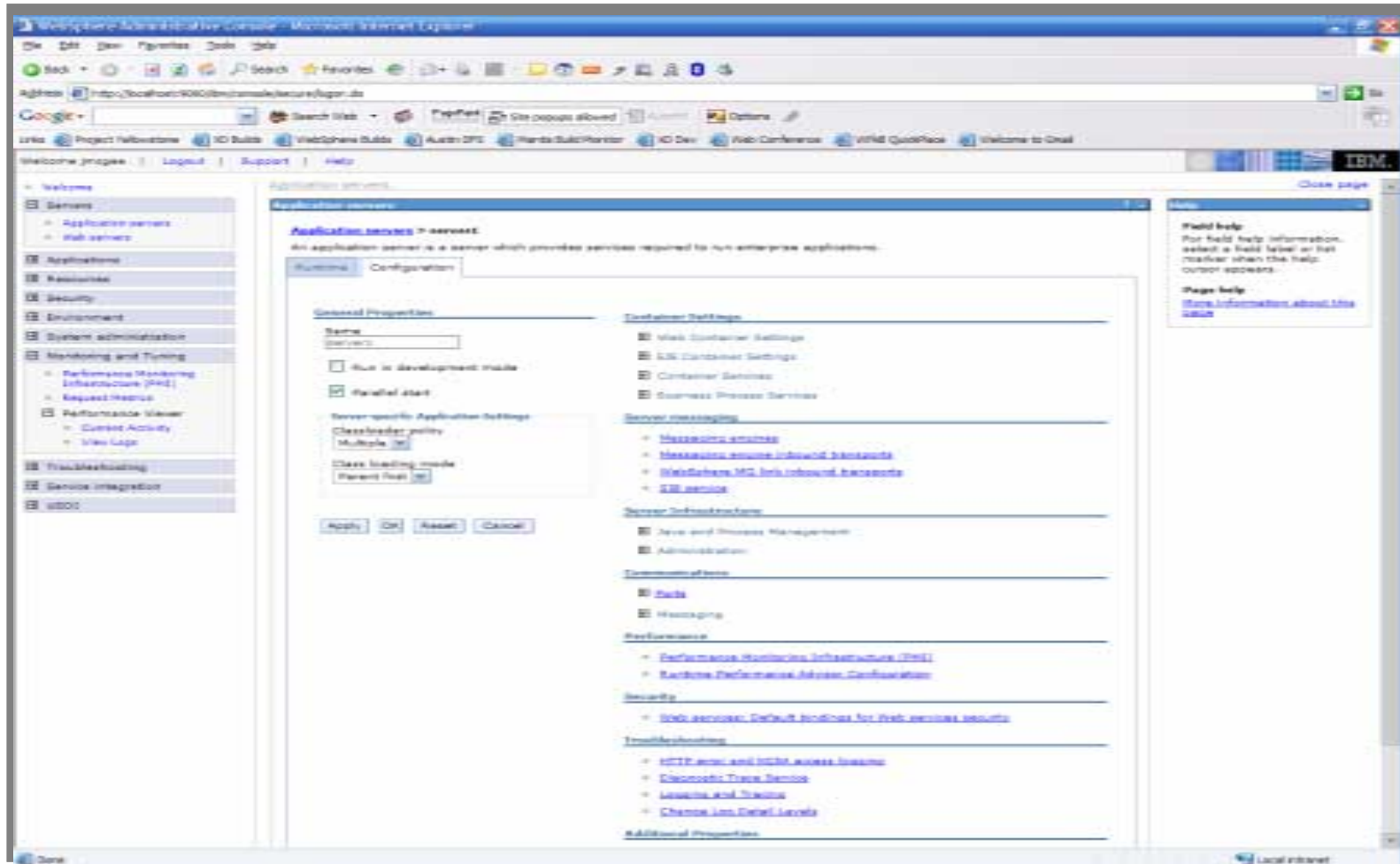


Administration Console Changes

- New style applied to existing WebSphere Application Server console structure
- Provides consistent cross-IBM product look and feel based on Integrated Solutions Console
- Adapt-a-View Support
 - ▶ Change console views based on context
 - Version
 - Platform
 - Installed Capabilities
- New panels for WebSphere 6.0 features have been added
- Addition of integrated browser-based performance viewer and advisor



New Admin Console Look and Feel



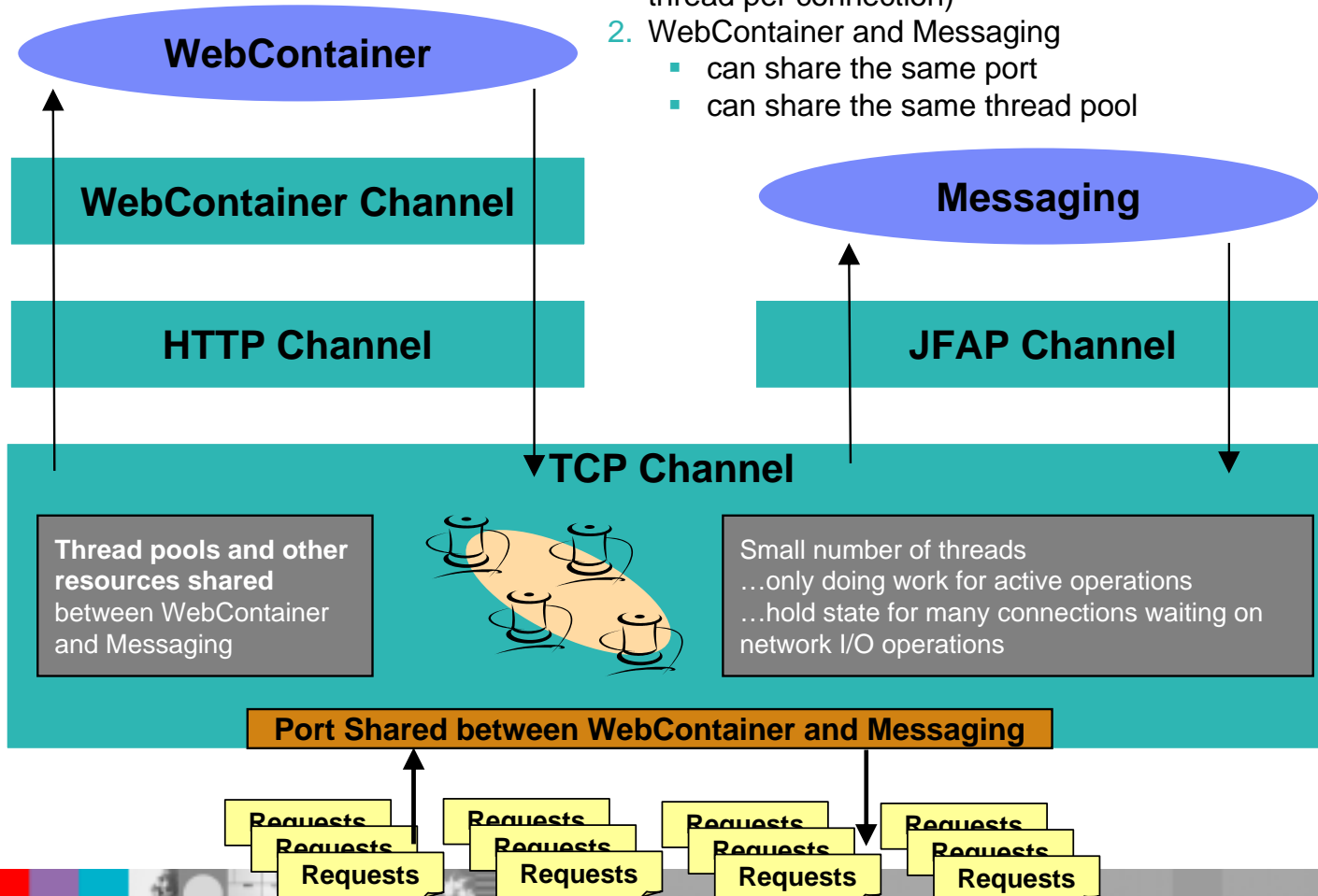
Agenda

- Platform Enablement
- Standards Based Architecture
- Ease of Use
- ➔ **Enterprise Class Deployment**
 - ➔ TransportChannel Service
 - ➔ JMS Support
 - ➔ High Availability Services
 - ➔ Enhanced Data Replication Service



TransportChannel Service

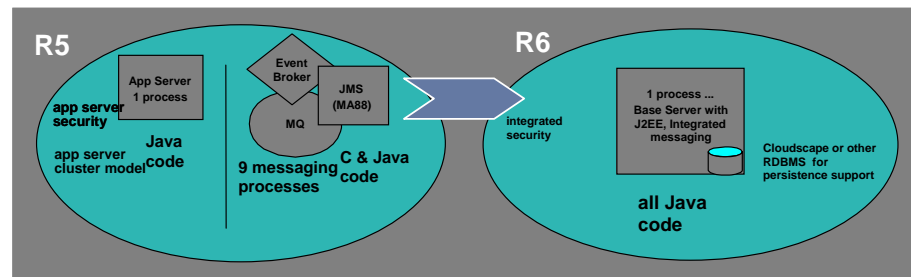
1. Non-blocking IO improves scalability (Does not require 1 thread per connection)
2. WebContainer and Messaging
 - can share the same port
 - can share the same thread pool





JMS Support

- WebSphere V6 provides a pure Java JMS 1.1 provider that is installed as part of the base server installation
 - ▶ not a silent install of another product with its own prerequisites and
 - ▶ runs completely inside the application server JVM
- Persistent messages are stored either in an embedded Cloudscape database or an external database of customer choice (DB2, Oracle, etc) via JDBC driver
- Each application server can host a messaging engine. Messaging engines can be interconnected to form a messaging bus
- Fully integrated with application server management including high availability. Messaging engines will failover along with application servers
- Interoperable with WebSphere MQ





WebSphere 6.0 High Availability Services

- WebSphere 6.0 has significant improvements in its high availability capabilities
 - ▶ Goal – WebSphere 6.0 be used as part of an overall 99.999% availability solution
- A **built-in high availability manager**
 - ▶ Provides Key Services needed to manage the server clusters
 - Reliable, high-speed, low-latency interconnect (Reliable Multicast UDP or Multicast over TCP)
 - Distributed Computing Services
 - Elections, Quorums, Heartbeats (Active or TCP Keepalive), Virtual Synchrony
 - Singleton Management
 - 1 of N
 - ▶ Runs key services on available servers rather than on a dedicated one (such as the deployment manager)
- Offers **hot standby** and **peer failover** for critical singleton services
 - ▶ Transaction Logs
 - ▶ Messaging Engines
- Takes advantage of fault tolerant **storage technologies** such as **Network Attached Storage (NAS)** to significantly **lower** the **cost** and **complexity** of High Availability configurations
- The configuration of high availability systems is significantly **simplified**





Enhanced Data Replication Service

- Rebasing on top of / Integrating with HA Manager and Transport Channel services
 - ▶ Improves performance and scale:
 - Improvements in the range of 4x to 8x
 - ▶ Improves high availability and failure recovery:
 - Leverages the failure detection provided by high availability services
 - Allows for "active failure recovery"
 - For example, with HttpSession replication, if the affinity server for a HttpSession goes down, WLM can route to another server that has a backup copy ready to use
 - ▶ Improves usability:
 - Leverages group services to simplify partitioning
 - Now have "n-replica", where the customer simply defines the number of backup copies they want for data
- Stateful Session Beans state now replicated





Miscellaneous

- Java Web Start Support for the client container

- 64-bit Support (6.0.1)
 - ▶ Itanium 2, AMD64, EM64T, PPC
 - ▶ Windows, Linux, HPUX

- Better documented and richer API and SPI support

- Hung Thread Detection

- IPv6 Support



Agenda

- Platform Enablement
- Standards Based Architecture
- Ease of Use
- Enterprise Class Deployment

➔ WAS for z/OS Transaction Capabilities



z/OS Recoverable Resource Service

- WebSphere Application Server is a transaction manager that supports the coordination of resource managers through their XAResource interface and participates in distributed global transactions with other OTS 1.2 compliant transaction managers (for example J2EE 1.3 application servers).
- WebSphere Application Server for z/OS also supports the coordination of local resource managers through RRS:
 - ▶ DB2
 - ▶ MQSeries
 - ▶ IMS
 - ▶ CICS
- IBM WebSphere Application Server for z/OS is capable of coordinating a mix of RRSTransactional resource managers and XA capable resource managers under the same global transaction



XA Partner log

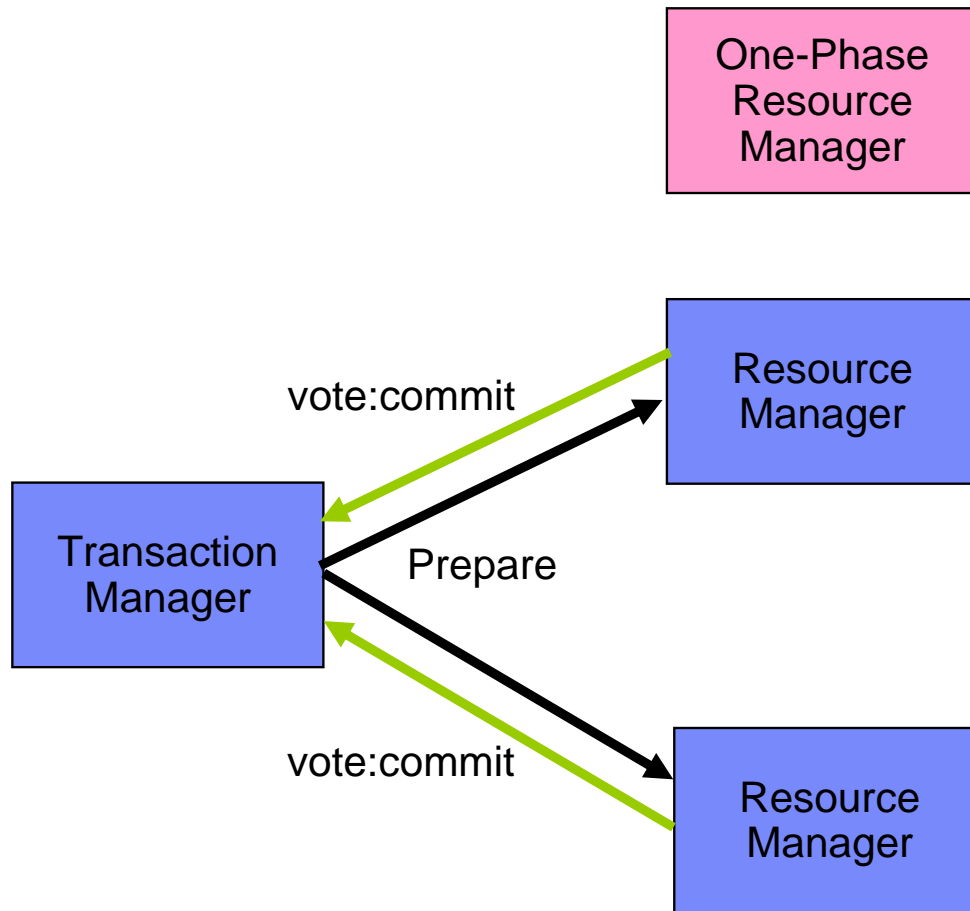
- When an XA resource manager enlists in a global transaction its interest in the RRS unit of recovery (UR) cannot be stored in RRS.
- WebSphere Application Server will enlist on its behalf with RRS and record this in an XA transaction log either on the HFS or in a LOGSTREAM.



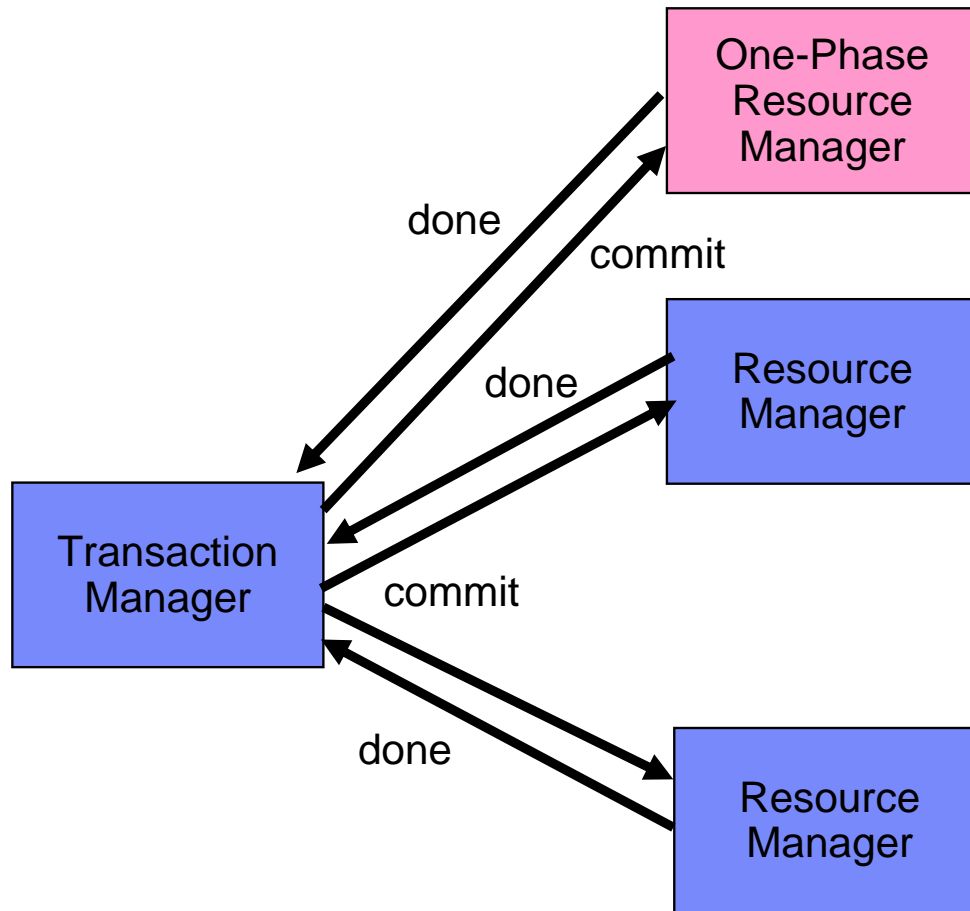
Mixing One and Two Phase Resources

- A single one-phase capable resource manager can participate in a global transaction in the following situations:
 - ▶ It is the only resource manager participating in the global transaction.
 - ▶ All the two-phase commit resource-providers that participate in the transaction are used in a read-only fashion. In this case, the two-phase commit resources all vote read-only during the prepare phase of two-phase commit. Because the one-phase commit resource provider is the only provider to actually perform any updates, the one-phase commit resource does not need to be prepared.
 - ▶ All other resources in the global transaction are RRS resource providers.
 - ▶ All other resources are two-phase commit resource providers and **last-participant** support is available.

Last Participant Scenario – Phase One



Last Participant Scenario – Phase One





IBM Software Group

Q & A



ON DEMAND BUSINESS™