JBoss Transactions

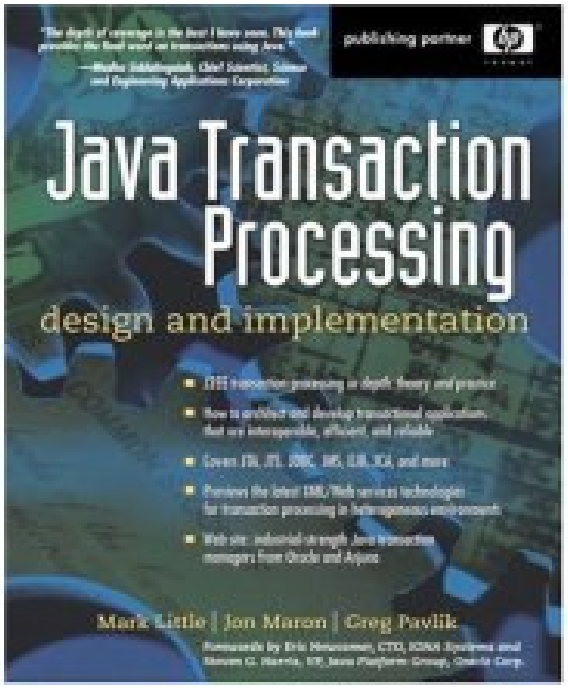
Transactions Everywhere!

What this talk will cover

- Background
- ArjunaCore
  - ✓ Transaction engine
- JTA
  - ✓ JDBC driver
- JTS
- WS-T
- Summary

What this talk won't cover

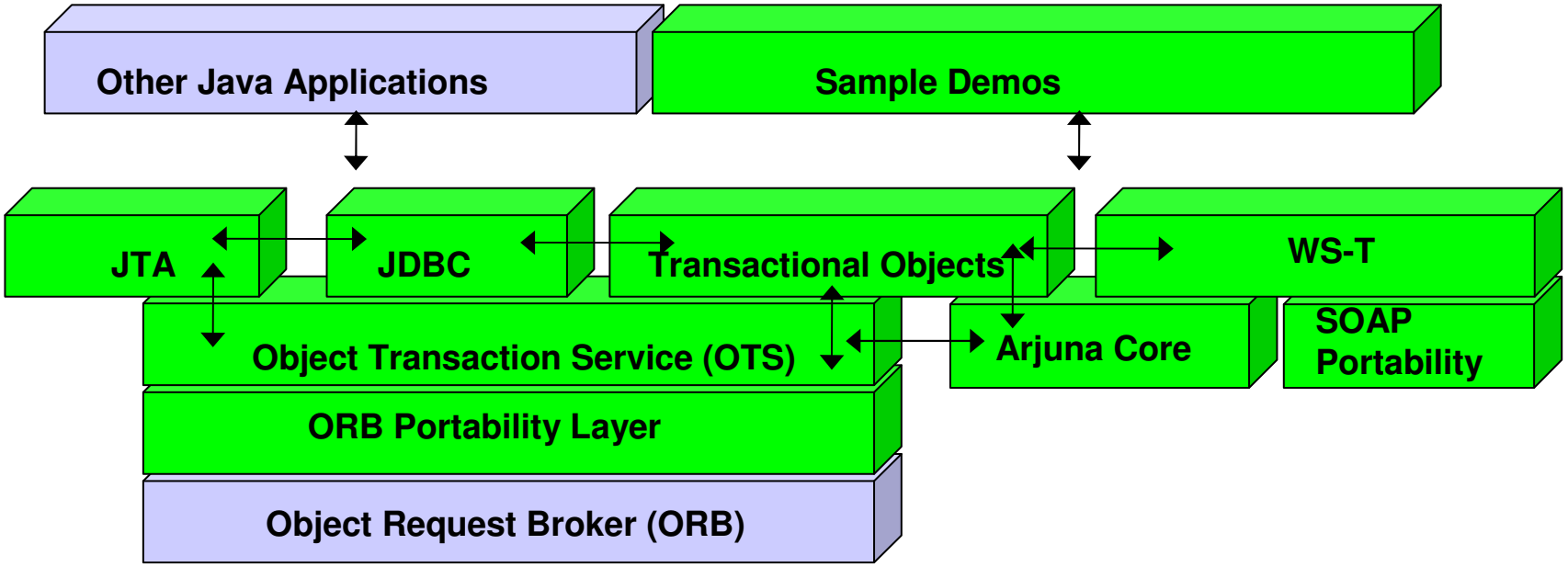
- Transaction processing basics
  - ✓ Maybe the subject of another webinar
  - ✓ There are enough good books out there to do the job



What is JBoss Transactions

- JBoss Transactions 4.2
  - ✓ Next generation of JBoss transaction service
  - ✓ Based on
    - JTA 1.0.1
    - JTS 1.0 (OTS 1.4)
    - WS-Coordination, WS-Atomic Transaction, WS-Business Activity
      - ✓ Demonstrated interoperability with IBM and MSFT
  - ✓ Used at HP World for Web Services seminars
  - ✓ Licensed to Tibco, webMethods and others
    - Does not require an application server to run
  - ✓ I18N and L10N

JBossTS Components



A-TS Package

Interact with each other

ArjunaCore

- Stand-alone transaction engine
- Full failure recovery
- ACID properties can be relaxed
  - ✓ Gray's matrix of transaction models
  - ✓ Does not restrict to XA
- Designed to be used stand-alone
  - ✓ Own set of APIs
- Similar to what MSFT are doing with Indigo

ArjunaCore

- Knows nothing about distribution
  - ✓ Does not require an ORB or an application server
  - ✓ Has hooks to allow distribution to occur in a system specific manner, e.g., IIOP or XML/SOAP
- 100% pure Java
- Small footprint (290 KB jar)
  - ✓ E.g., runs on an HP Jornada 720!
- Highly adaptable, e.g., used in Arjuna's messaging product as well as our Web Services component

Failure recovery

- Automatic failure recovery daemon
  - ✓Runs periodically
  - ✓Can be driven directly
- Recover inflight transactions
- Recover resources
  - ✓different recovery mechanisms required for each resource type
  - ✓different mechanisms can be easily added

Further features

- Many configuration options
  - ✓Transaction nesting
- Checked transactions
  - ✓Per transaction basis
- Last resource commit optimization
- Asynchronous commit protocol
  - ✓Prepare and commit
- Transaction management tools
  - ✓Heuristic resolution

JEE Component

- Compliant with OMG OTS specification
  - ✓Supports all optional features
  - ✓Superset with flexible implementations
- JTA 1.0.1 compliant
- Supports JDBC 2.0
  - ✓E.g., Oracle 8/9, Sequelink, Cloudscape
- In use for nearly a decade
  - ✓Longer if you consider C++

JEE support

- Local and remote JTA implementations
- World's first JTS implementation
  - ✓Used to push the OTS specification
  - ✓Completely multi-thread aware
- Portable to a number of ORBs
  - ✓E.g., Orbix 2k, JacORB, JDK ORB, …
- Distributed failure recovery
- Sub-transaction aware resources

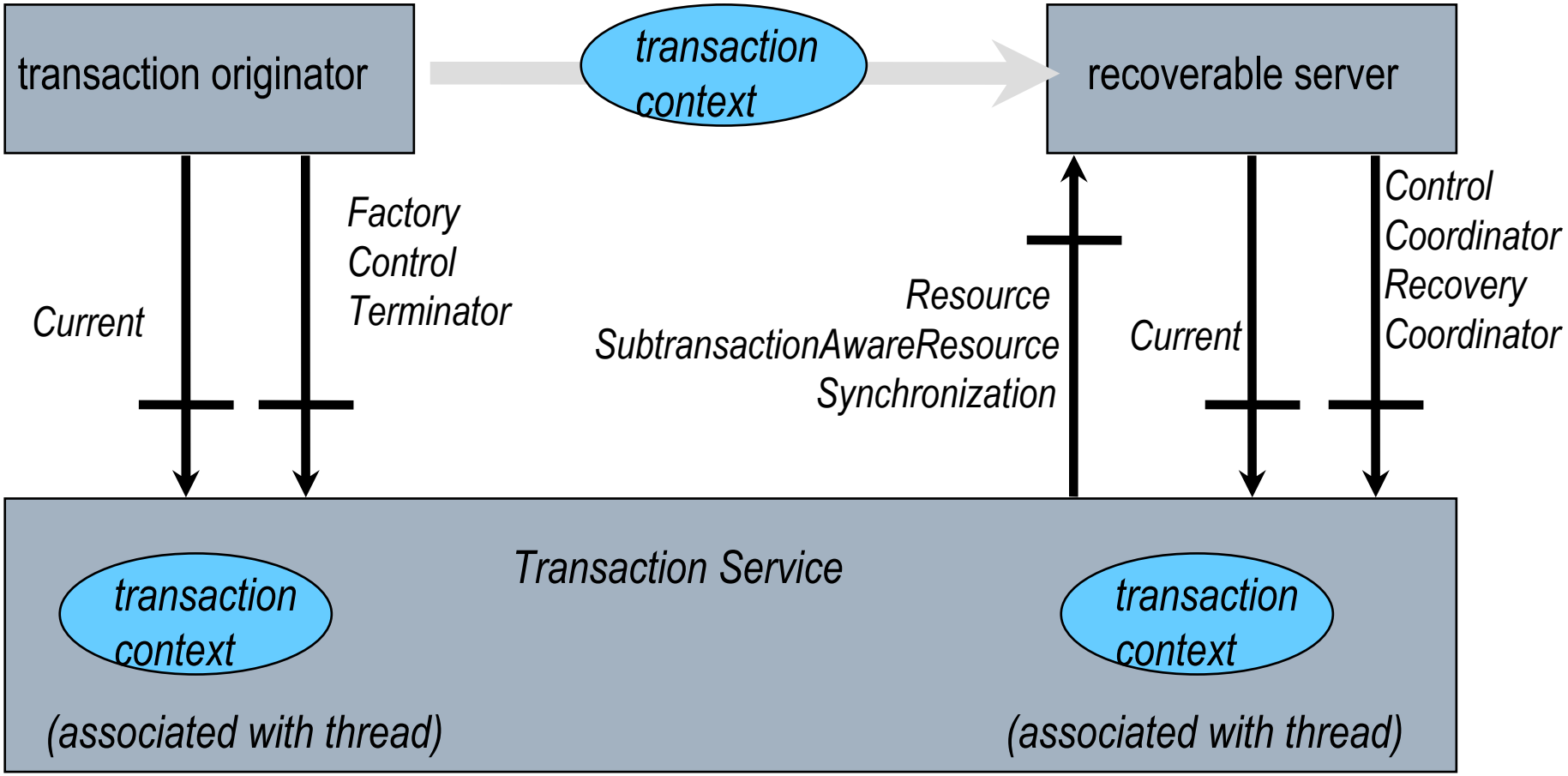
JTA component

- Two variants of JTA
  - ✓ Distributed version
    - Layered on JTS
      - ✓ Requires ORB
  - ✓ Non-distributed version
    - Does not require ORB
      - ✓ Faster
- Transactional JDBC driver
  - ✓ Automatically enlists resources with JTA
    - Works with Oracle 8/9, Cloudscape/Derby, …

JTS component

- Supports distributed two-phase commit
  - ✓ One-phase commit optimisation
- Failure recovery automatically completes transactions
  - ✓ Driven from resource side as well as from transaction manager
- Fast, in-process transaction management
  - ✓ Separate transaction server possible

OTS architecture



Nested transactions

- Optional part of JTS specification
  - ✓ few sub-transaction-aware resources
- Registered resources are only informed of transaction termination
- No two-phase commit for sub-transactions
  - ✓ can result in heuristic-like outcomes
  - ✓ Implementation specific extensions

Why use subtransactions?

- Fault isolation
  - ✓subtransaction work can be rolled back independently of enclosing transaction
    - can try alternate work
- Modularity
  - ✓objects can be responsible for their own transactionality irrespective of client

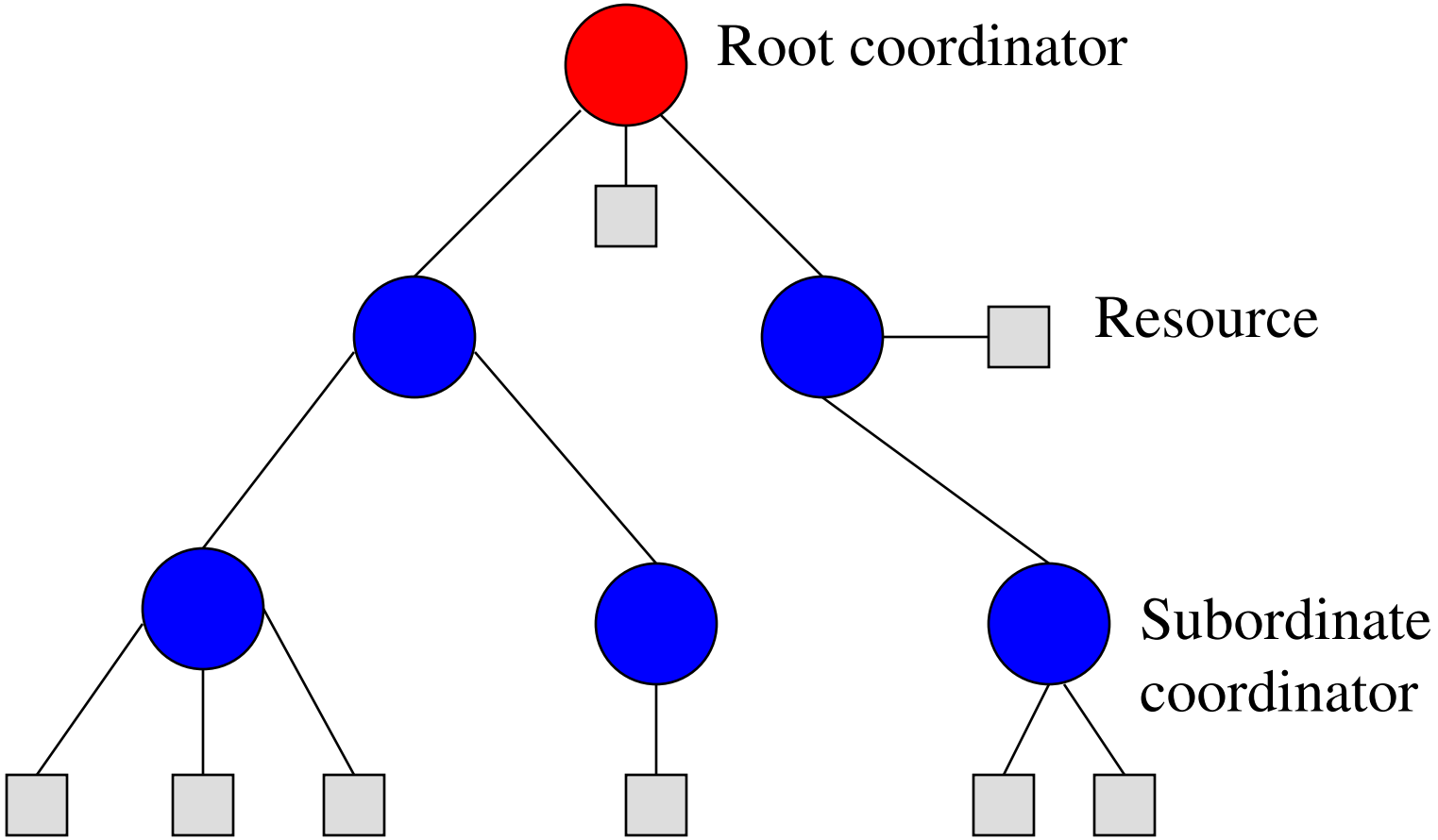
Transaction propagation

- Explicit propagation
  - ✓Context passed as parameter
  - ✓Object implementation responsible for using it when required
- Implicit propagation
  - ✓Transaction context is implicitly passed from client to object
  - ✓All operations are assumed transactional

Interposition

- Allows a subordinate coordinator to be created
- Interposed coordinator registers with transaction originator
  - ✓Form tree with parent coordinator
  - ✓Application resources register locally
- JBossTS supports interposition for implicit and explicit propagation

Interposition



Web Services transactions

- Traditional ACID transactions are not appropriate for Web Services
  - ✓ No longer strongly coupled and trusted environments
  - ✓ Potentially long duration processes
- WS-AtomicTransaction/WS-BusinessActivity
- OASIS WS-Transaction Management

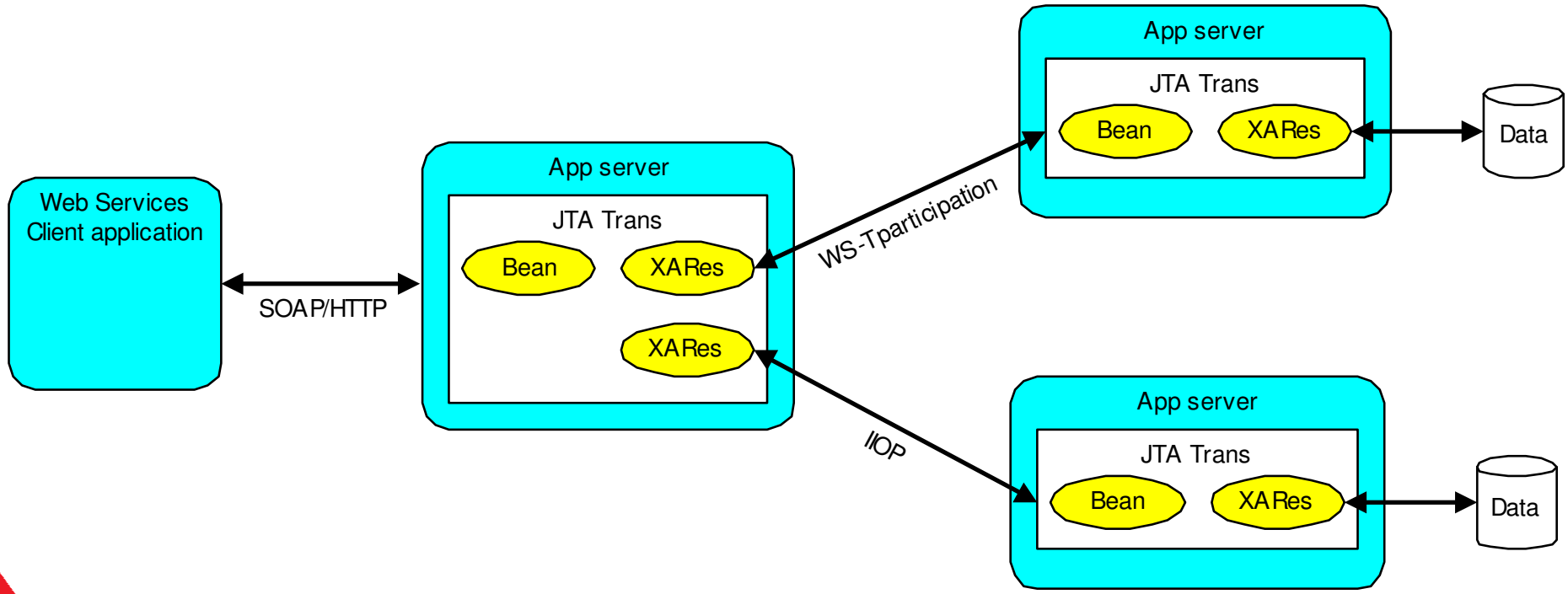
End-to-end transactions

- Most Web Services will use existing transaction aware resources
  - ✓ Relational databases (Oracle; DB2; SQLServer)
  - ✓ Message queuing systems
  - ✓ Interoperability with XA specification is required
- Many Web Services will use JEE
  - ✓ JTA/JTS transactions are similar but not identical to Web Services transactions
  - ✓ Bi-directional interoperability between JTA/S and WS-T
  - ✓ Seamless flowing of transaction context from Web Services client, through EJB to backend database

JBossTS provides the glue

- Therefore, WS transactions are not sufficient by themselves
- Require integration with back-end (e.g., JTS) solutions
  - ✓Provide end-to-end solution for customers
    - It is critical to the take-up of transactions in Web Services

Multi-modal transactions



JBoss WS-T component

- Support for general coordination framework
  - ✓ WS-Coordination
    - Similar to JSR 95/OMG Activity Service
- Support for Web services transactions
  - ✓ Atomic Transaction
    - Traditional two-phase commit
  - ✓ Business Activity
    - Similar to OASIS WS-TXM Long Running Action

Implementation overview

- Separate transaction and coordination core from messaging
  - ✓ Provide SOAP portability infrastructure
- Transaction services plugged in
  - ✓ Allows us to leverage existing implementations
    - E.g., ArjunaCore, Tuxedo, CICS, Encina
  - ✓ Customers trust pedigree
    - Especially where their money is concerned

JBoss 3/WebLogic/JBoss 4*

Application server versus transaction capabilities	Standards compliant	Industry proven	2PC	Failure recovery	Flexible deployment	Distributed transactions	Mgmt tools	Interop	Flexible partici p-ants	Web Services transactions	WS-tx to J2EE tx bridge
JBoss 3	√ (JTA)	X	√	X	X (tied to application server)	X	X	X	X (XA specific)	X	X
WebLogic	√ (JTA)	√	√	√	√ (can run out of application server)	√	√	X	X (XA specific)	X	X
JBoss 4*	√ (JTA and JTS)	√	√	√	√ (can run out of application server)	√	√	√ (via JTS)	√ (not just XA partici pa-nts)	√ (via Arjuna, IBM, MSFT, Oracle specs.)	√

Roadmap

- Rebrand and open source ArjunaTS as JBoss Transactions 4.2
  - ✓ Java Transaction Service
  - ✓ WS Transaction Service
  - ✓ This product appeals to Financials, Telco, and Insurance verticals as well as SOA ISVs.
- JBoss AS 4.x
  - ✓ Current JTA will remain default
  - ✓ Open source ArjunaJTA will be made available as separate download
  - ✓ Support for open source ArjunaJTA included in JBoss AS subscription
- JBoss AS 5
  - ✓ Open source ArjunaJTA will be default JTA
  - ✓ Current JBoss JTA will be phased out
- When will this be available?
  - ✓ Targeting Q1 2006 for JBoss AS 4 support

Summary

- Product features
  - ✓ High performance and reliability
  - ✓ Manageability and configurability
  - ✓ Standards compliance
  - ✓ Modular architecture to optimise footprint
  - ✓ Pure Java implementation
- Deployment options
  - ✓ Application server agnostic
  - ✓ Deployable in or outside a J2EE application server