Narayana + WildFly

JBug Brno
Jan 2015

Mike Musgrove

redhat

JBoss
a division of Red Hat

Agenda

- Transaction Basics
- JTA
- Narayana
- Narayana in WildFly
- WildFly Transaction Sub System Configuration
- Optional WildFly Sub Systems
- Q & A

Narayana

Transaction Basics

redhat

JBoss
a division of Red Hat

Transaction Basics

- A transaction is a group of business logic statements with certain shared properties. One or more of:
  - Atomic, Consistent, Isolated, Durable
- In JEE, JTA transactions are ACID transactions
- Other models may relax some of these properties
- From the user perspective, a transaction is a set of changes that happens or does not happen

redhat

JBoss
a division of Red Hat

Transaction Management

- A transaction manager or transaction service
  - Coordinates transactions to ensure correct and complete execution
  - Provides user and container APIs
  - May be distributed
  - Drives resource managers

- A resource manager
  - Manages data that may be manipulated transactionally
  - Provides a data management API to the user (eg JDBC)
  - Provides a transaction management API to the transaction manager (XA)

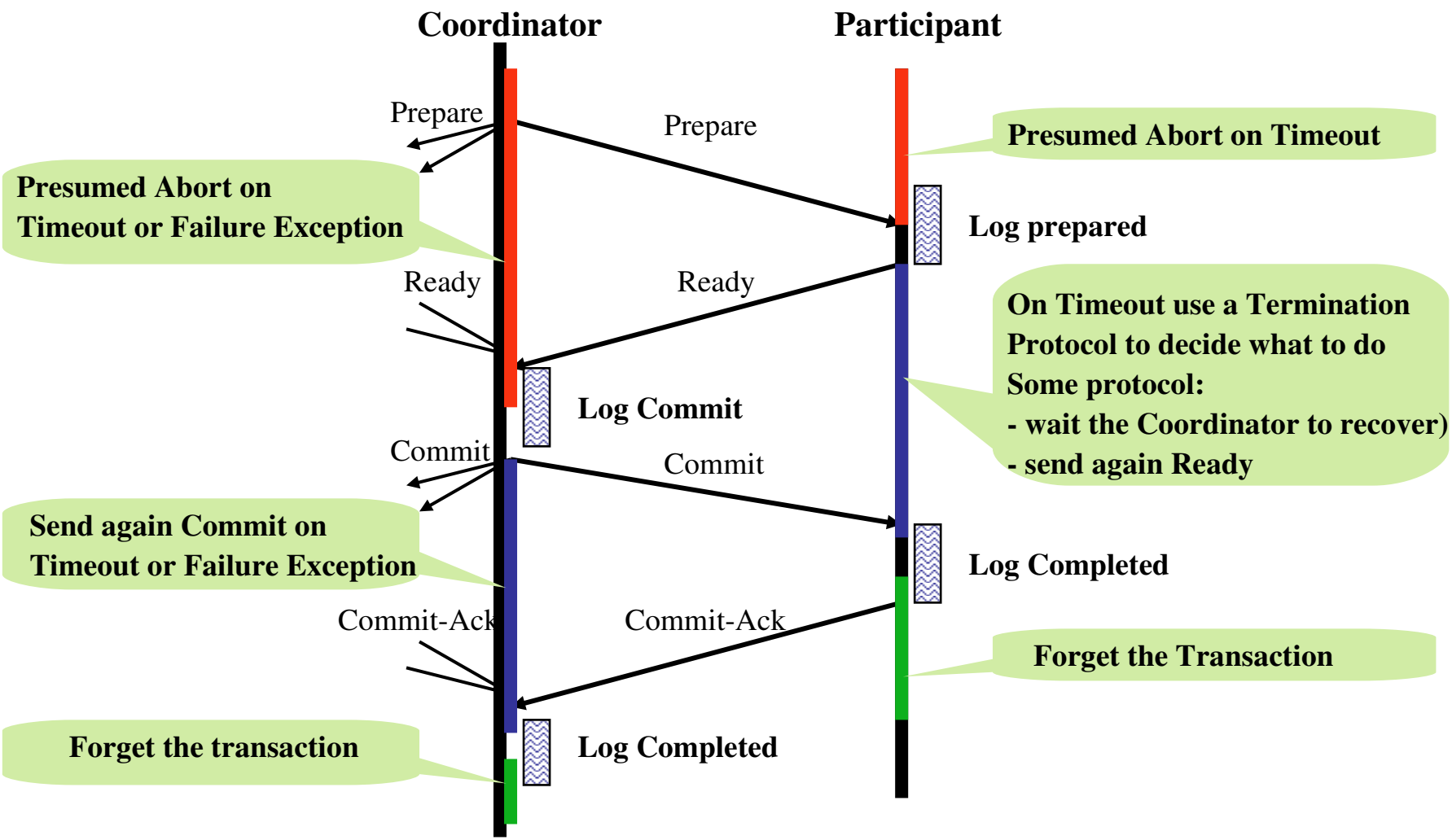
redhat

JBoss a division of Red Hat

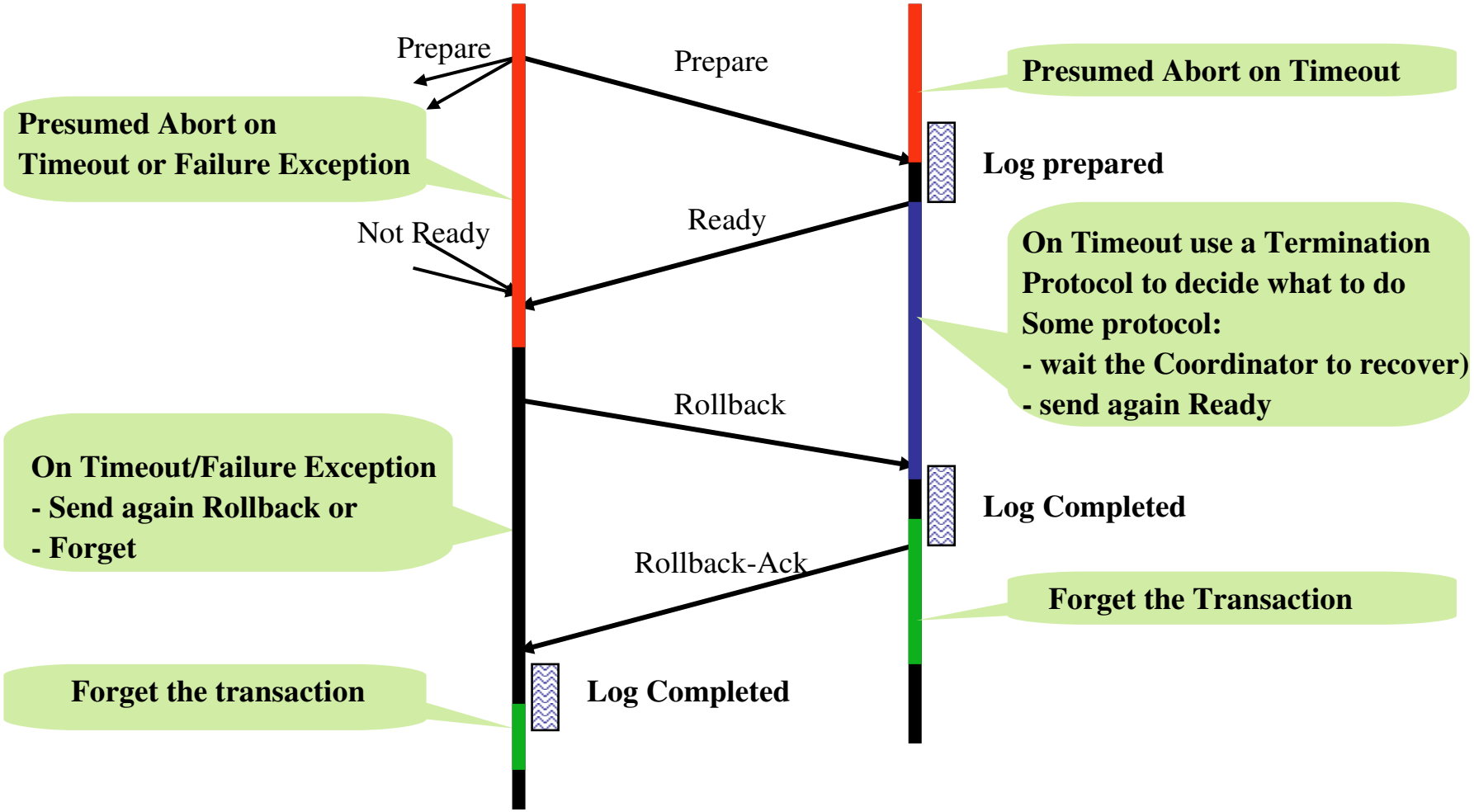
Failure Handling

- Presumed Abort Strategy
  - When in doubt, abort
  - any failure prior the commit phase lead to abort the transaction
- Type of failures:
  - Communication and processes (detected by timeout or exceptions)
- A coordinator or a participant can fail in two ways
  - it stops running (crashes)
  - it times out waiting for a message it was expecting
- A recovered coordinator or participant uses information on stable storage to guide its recovery

Failure Recovery in Commit Phase



Failure recovery during abort



Two Phase Commit Optimizations

- One Phase Commit
  - Single resource manager does not require voting
- Read Only
  - Unmodified resources don't need phase two commit/abort
- Last Resource Commit Optimization (LRCO)
  - Prepare 2PC, commit LRCO, write log, commit 2PC
  - Crash after step 2 before writing the log (heuristic)
- Commit Markable Resource (CMR)
  - Fixes the LRCO failure window by committing the intentions list to the database

redhat

JBoss a division of Red Hat

Distributed Transactions

- Local transaction: within a single resource manager, no separate transaction manager.
- Global transaction: one potentially spanning several logically separate resource managers, coordinated by a transaction manager (XA/JTA)
- Distributed transaction: a global transaction where transaction context is propagated on business logic calls between nodes (JTS, XTS, RTS, BT)
- Interposition
  - A useful performance optimization
- Subordinate transactions
  - Used by JCA

XA

- XA is the interface between the transaction manager and resource managers
- JEE users usually don't interact with XA, except in server configuration
- Widely supported in e.g. JDBC drivers and JCA connectors
  - With varying degrees of standards compliance

Narayana

Java Transaction API (JTA)

redhat

JBoss a division of Red Hat

JTA

- Standard transaction management API for JEE
  - Standardizes interactions between components involved in a distributed transaction (Application, application server, transaction manager, resource adapter)
  - Influenced by X/Open DTP
  - Assumes ACID transactions
  - Interacts with JCA, JDBC, JMS, EJB specs
- Transaction Management
  - UserTransaction, TransactionManager, Transaction, Status, Xid, TransactionSynchronizationRegistry
- Resource Management
  - Xid, XAResource (hides Connection), Synchronization

redhat

JBoss a division of Red Hat

UserTransaction

- The application programmer's interface to the transaction manager
  - Provides transaction boundary demarcation
- Lookup through JNDI (or injected)
- begin() / commit() / rollback()
  - finally: Be careful not to leave tx uncompleted
- setTimeout(int seconds)
  - But no getTimeout()
  - Timeouts happen on a background thread
- setRollbackOnly()
- getStatus()

redhat

JBoss
a division of Red Hat

TransactionManager

- The container's interface to the transaction manager
- Same functions as UserTransaction, plus:
- Thread management
  - suspend()
  - resume()
- Access to the transaction...
  - getTransaction()

Transaction

- Represents an individual transaction instance
- Usually for container, not end-user use
- Lifecycle management
  - commit / rollback / setRollbackOnly / getStatus
- XAResource Management
  - enlistResource
  - delistResource
- Synchronization Management
  - registerSyncronization

XAResource

- The interface between the transaction manager and the resource manager
- Provided by the client library (driver) of most transaction capable resource managers
- The transaction manager deals only with XAResource, not Connection or other driver/API specific abstractions (cf JCA)
- Start/end, prepare/commit, recover/forget

Narayana

What is it?

Narayana is...

- A general purpose transaction manager
- Written in Java but usable from other languages: ceylon, Ruby (via TorqueBox), C/C++, REST, ...
- A JTA implementation
- A JTS implementation, with an optional JTA API
- A Web Services transaction manager
- Usable standalone or embedded
- Embedded in WildFly as a subsystem
- More that just an implementation of the standards
  - TxOJ, BA Framework, REST-AT, STM, ...

XA Recovery

- Top down recovery (Transaction manager initiated)
  - tx manager drives recovery from its tx logs
- Bottom up recovery ('Resource manager initiated')
  - External resources managers have their own logs
  - But still driven by Narayana recovery manager
    - Xid 'ownership' becomes important
    - Can require extra permissions in the RM
- Recovery does not always work
  - autonomous/manual forcing of outcome to remove locks
  - Can lead to Heuristics

Narayana

Narayana in WildFly

redhat

JBoss
a division of Red Hat

Topics

- Transaction Subsystem Configuration
- Optional Subsystems
- Bridging
- Debugging
- Demonstration
- Q & A

Narayana

Transaction
Subsystem
Configuration

Sub-System Configuration

- The default profile is good enough for JTA
- JTS needs the full profile to get an ORB
- Statistics:
  - /subsystem=transactions/:write-attribute(name=enable-statistics,value=true)
  - /subsystem=transactions/:read-attribute(name=number-of-<statistic>,include-defaults=true)
    - transactions, inflight-transactions,
    - heuristics, committed-transactions,
    - aborted-transactions, timed-out-transactions,
    - application-rollbacks, resource-rollbacks

Sub-System Configuration

- Transaction timeout value:
  - /subsystem=transactions/:write-attribute(name=default-timeout,value=300)
- Transaction log location and recovery:
  - /subsystem=transactions/:write-attribute(name=object-store-path,value=tx-object-store)

Sub-System Configuration (node id)

- Node-identifier: uniqueness required when 2 TM's:
  - access the same RM
  - share an object store
- /subsystem=transactions/:write-attribute(name=node-identifier,value=1)

Sub-System Configuration (JTS)

- If you need to propagate a transaction to another instance of WildFly you will need to enable JTS.
  - /subsystem=transactions/:write-attribute(name=jts,value=true)
- JTS requires an ORB which needs to be configured to use transactions. For this you will need the full profile:
  - Enable security interceptors:
    - /subsystem=jacorb/:write-attribute(name=security,value=on)
  - Enable transactions:
    - /subsystem=jacorb/:write-attribute(name=transactions,value=on)
  - Remark: WildFly 9 replaces JacORB with the JDK ORB

Object Store Configuration

- The default is file based (1 file per log record)
- Journaling Store:
  - /subsystem=transactions/:write-attribute(name=use-hornetq-store,value=false)
  - /subsystem=transactions/:write-attribute(name=hornetq-store-enable-async-io,value=false)
- Database Store:
  - enable and set JNDI name of which db to use:
    - /subsystem=transactions:write-attribute(name=use-jdbc-store, value=true)
    - /subsystem=transactions:write-attribute(name=jdbc-store-datasource, value=java:jboss/datasources/TransDS)
  - Plus options to set the action, communication and state table names)

Viewing Transaction Logs

- Refresh the management view of the transaction logs:
  - /subsystem=transactions/log-store=log-store/:probe
- List all transaction logs:
  - ls /subsystem=transactions/log-store=log-store/transactions
- View a single log:
  - /subsystem=transactions/log-store=log-store/transactions=0\:ffff7f000001\:-b66efc2\:4f9e6f8f\:9:read-resource
- View a transaction participant log:
  - /subsystem=transactions/log-store=log-store/transactions=0\:ffff7f000001\:-b66efc2\:4f9e6f8f\:9/participants=java\:\/JmsXA:read-resource

Viewing Transaction Logs

- Delete a log:
  - /subsystem=transactions/log-store=log-store/transactions=0\:ffff7f000001\:-b66efc2\:4f9e6f8f\:9:delete
- Try to recover a heuristic:
  - /subsystem=transactions/log-store=log-store/transactions=0\:ffff7f000001\:-b66efc2\:4f9e6f8f\:9/participants=java\:\/JmsXA::recover
- Re-activate a log:
  - /subsystem=transactions/log-store=log-store/transactions=0\:ffff7f000001\:-b66efc2\:4f9e6f8f\:9/participants=java\:\/JmsXA::refresh

Configure DataSources

- Should be configured with JTA enabled (tells JCA to enlist them as XAResources) by setting the jta="true" attribute.
- Credentials used by apps and the TM recovery system are generally different (see recover-credential parameter on the DataSource).
- Remark: the next EAP release (6.4) we support a reliable alternative to LRCO referred to as CMR:
  - The DS config must have the connectable parameter to "true". In the txn subsystem definine which resources can be used via the commit-markable-resources complex attributed (list of non XA aware datasources that can reliably participate in an XA transaction.). For each such DS specify the JNDI name and the table name containing pending XIDs.
- NB: Local DataSources are enlisted using LRCO (there can be only one)

Optional WildFly Subsystems

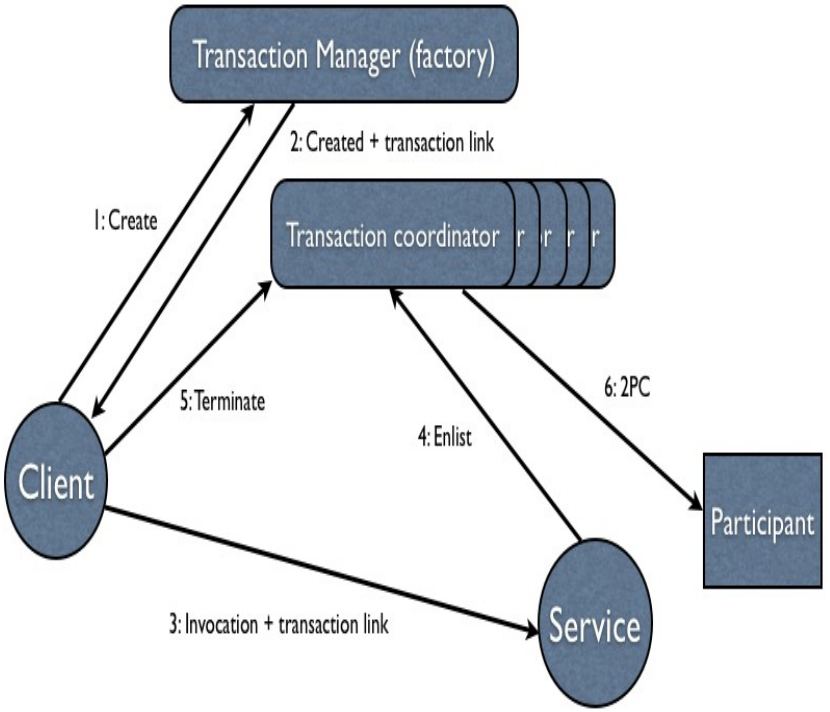
WS-AT, REST-AT and Bridging

XTS: XML Transaction Service

- Provides transaction support for Web Service applications. Implements the WS-Atomic Transaction (WS-AT) and WS-BusinessActivity (WS-BA) protocols, it allows for transactions to span multiple Web Services.
- WSAT is available as an optional subsystem (WSBA is not available yet). It can be enabled using the standalone-xts.xml configuration:
  - cp docs/examples/configs/standalone-xts.xml standalone/configuration
  - ./bin/standalone.sh --server-config=standalone-xts.xml&
  - ./bin/jboss-cli.sh --connect "/subsystem=xts:read-resource-description(recursive=true)"

REST-AT: Atomic Transactions

RESTful interface to Narayana



1. POST /transaction-manager HTTP/1.1
2. HTTP 1.1 201 Created Location: /transaction-coordinator/1234 Link:</transaction-coordinator/1234/terminator>; rel=”terminator”, </transaction-coordinator/1234/participant>; rel=”durable- participant”, </transaction-coordinator/1234/vparticipant>; rel=”volatile -participant”
3. REST service request
4. POST /transaction-coordinator/1234/participant

   Link:</res-url>; rel=participant,</res-url/prepare>; rel=prepare,</res-url/commit>; rel=commit,</res-url/rollback>; rel=rollback,</res-url/commit-one-phase>; rel=commit-one-phase
5. PUT /transaction-coordinator/1234/terminator HTTP/1.1 Content-Type: application/txstatus txstatus=TransactionCommitted
6. PUT /participant-resource/terminator HTTP/1.1 txstatus=TransactionPrepared

REST-AT: RESTful API to a TM

- Provides a REST based interface to JTA
- Configuration:
  - cp docs/examples/configs/standalone-rts.xml standalone/configuration
  - ./bin/standalone.sh --server-config=standalone-rts.xml&
- This entry point to the system is via the resource located at the endpoint: http://<host>:<port>/rest-at-coordinator/tx/transaction-manager
  - ./bin/jboss-cli.sh --connect "/subsystem=rts:read-resource-description(recursive=true)"

BlackTie (X/Open DTP model)

- Implements the XATMI and TX X/Open DTP standards:
  - Concurrent execution of applications on shared resources
  - Coordination of transactions across applications
  - Components, interfaces, and protocols that define the architecture and provide portability of applications
- Supports XA-compliant databases for data storage and retention
- C/C++/Java client library adapter framework for exposing familiar JEE components (Narayana and Hornetq) via the familiar ATMI APIs. Transactional queues (further decoupling clients and servers)
- Allows the creation of EJB-like software modules known as Application Programs
- Works with EJBs (for transaction inflow/outflow)

Optional WildFly Subsystems

Bridging Transaction Models

Transaction Bridging

- Existing JEE code understands JTA transactions
- Web Services understand WS-AT transactions
- REST services understand REST-AT
- Interoperability and reuse requires joining these models together to seamlessly flow a transaction between them and convert API calls to match the txn APIs provided in each environment.
- txbridge provides a way to do this
  - Interposition plus protocol adapter
  - Bi-directional for WS_AT (no WS-BA bridging)
  - Inflow only for REST (REST-AT -> JTA)
- Now JPA, JMS etc can be used with JAX-RS and WS-AT endpoints by enabling the relevant bridges

WildFly Subsystems

Debugging support

redhat

JBoss a division of Red Hat

Debugging

- Connect a debug tool such as Intellij by editing bin/standalone.conf
- Increase log level by editing the the logging subsystem config in the wildfly config file. The relevant logger category for transactions is <logger category="com.arjuna">

Debugging

- Targeted Debugging using Byteman (byteman.jboss.org)
  - simplifies tracing and testing of Java programs:
    - "The simplest use of Byteman is to install code which traces what your application is doing. This can be used for monitoring or debugging live deployments as well as for instrumenting code under test so that you can be sure it has operated correctly. By injecting code at very specific locations you can avoid the overheads which often arise when you switch on debug or product trace."
  - Enable byteman by using bminstall/bmsubmit or set JAVA_OPTS in bin/standalone.conf:
    - -javaagent:/<BYTEMAN_HOME>/lib/byteman.jar=script:<script location)

NTA (Narayana Transaction Analyser)

- NTA (https://github.com/jbosstm/transaction-analyser) analyses a wildfly log and presents a history of transaction related activity in a easily digestible form. The goal is to simply diagnosis of transaction-related problems.
- The tool can also be loaded with a suite of plugins that diagnose common issues:
  - Many of your transactions are rolling back, and you don't know why.
  - You have a distributed transaction crossing many servers, and you're finding it difficult to correlate the many log files.
  - You have a heuristic transaction, but you don't know which resource misbehaved.
  - A transaction appears to have 'hung', but you don't know why.

WildFly Transaction Subsystem

Demonstration

Demonstration: How to configure JTS

- Walkthrough of one of the JTS quickstarts
- Shows how to configure two servers for propagating transactions during EJB invocations
  - https://github.com/wildfly/quickstart/tree/master/jts
- Demonstrate recovery
  - https://github.com/wildfly/quickstart/tree/master/jts-distributed-crash-rec

Links

- Narayana:
  - http://narayana.io/docs
  - https://github.com/jbosstm/narayana
- BlackTie: http://narayana.io/docs/project/index.html#d0e16296
- RTS: http://narayana.io/docs/project/index.html#d0e15500
- XTS: http://narayana.io/docs/project/index.html#d0e3692
- Bridging:
  - http://narayana.io/docs/product/index.html#d0e6287
  - http://narayana.io/docs/project/index.html#_interoperating_with_other_transaction_models
- XATMI and TX X/Open DTP standards:
  - http://www.opengroup.org/pubs/catalog/c506.htm
  - http://www.opengroup.org/pubs/catalog/u011.htm


Questions?
http://narayana.io/community/