JUDCon

JBoss Users & Developers Conference

Boston:2010

Speakers qualifications

- Tom Jenkinson
- JBoss BlackTie project lead since 2008
  - Also project lead for JBoss MASS
- Previously an Arjuna JTS developer between 2000-2005
- Previously a consultant specializing in integration projects 2005-2008

Speakers qualifications

- Tom Jenkinson
- JBoss BlackTie project lead since 2008
  - Also project lead for JBoss MASS
- Previously an Arjuna JTS developer between 2000-2005
- Previously a consultant specializing in integration projects 2005-2008

Agenda

- What are XATMI, TX and XA?
- So, why should I be interested?
- What is BlackTie?
- How can I use BlackTie today?
- Where next?

What are XATMI, TX and XA?

What are XATMI, TX and XA?

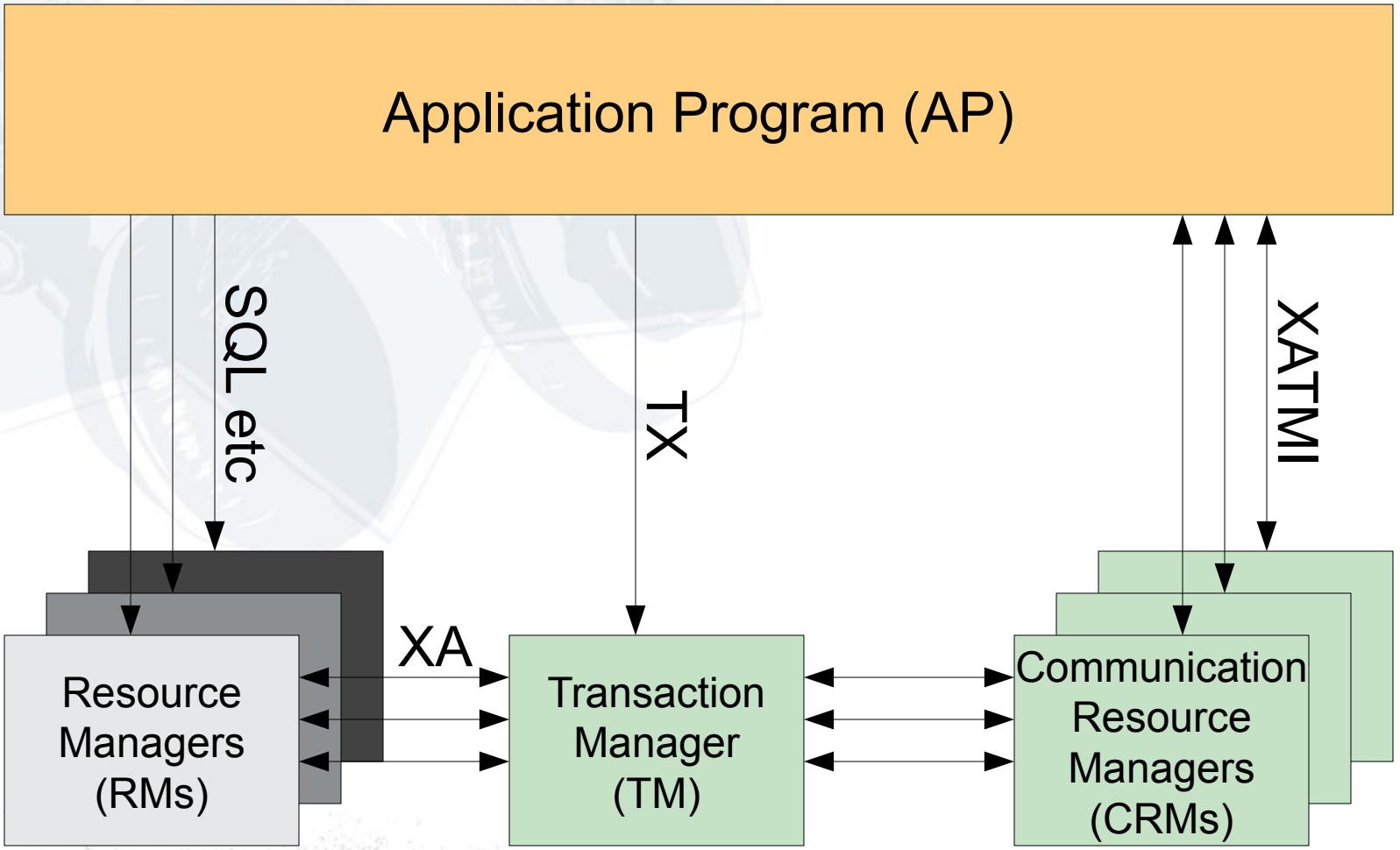
- Were published by the X/Open Group in the 90s
- X/Open is now called The Open Group
- Their remit is to define an open systems environment
- The story of The Open Group is very much intertwined with that of UNIX

What are XATMI, TX and XA?

- These standards form part of the X/Open “Common Applications Environment”
  - Distributed Transaction Processing
  - CAE extends much beyond the scope of middle-ware to everything over the hardware level
- The CAE shares some of the objectives of JEE, that respect the DTP is a competitor

"[CAE] provides for portability and interoperability of applications, and so protects investment in existing software while enabling additions and enhancements. It also allows users to move between systems with a minimum of retraining"

CAE DTP: From the DTP Spec



What does the CAE give you?

- Ability to write a standard AP which can be deployed into a container to get access to XA aware resources in a consistent and portable manner
- Single standard the developer can conform to when developing their application

What doesn't it address

- Doesn't address:
  - Scalability
  - Administration
  - Performance
  - Security
- These qualities and features help to distinguish the various vendors

Spec overview

- XATMI defines the communication centric operations
  - Think JMS
- TX defines the transaction demarcation operations
  - JTA and 2PC
- XA is the equivalent of a `javax.transaction.xa.XAResource` and allows the TM to control the RM

Categories of XATMI invocations

- RPC
  - tpcall
    - `int tpcall(char *svc, char *idata, long ilen, char **odata, long *olen, long flags)`
  - Similar to a javax.jms.QueueRequestor
    - `javax.jms.Message request(javax.jms.Message message)`

Categories of XATMI invocations

- Asynchronous
  - tpacall/tpgetrply
    - `int tpacall(char *svc, char *data, long len, long flags)`
    - `int tpgetrply(int *cd, char **data, long *len, long flags)`
  - In javax.jms like sending a message with a JMSReplyTo destination set to a temporary queue or topic:
    - `Session: TemporaryQueue createTemporaryQueue()`
    - `Message: void setJMSReplyTo(Destination destination)`
    - `MessageProducer: void send(Message message)`
    - `MessageConsumer: Message receive()`

Categories of XATMI invocations

- Conversation oriented
  - tpconnect
    - `int tpconnect(char *svc, char *data, long len, long flags)`
    - `int tpsend(int cd, char *data, long len, long flags, long *revent)`
    - `int tprecv(int cd, char **data, long *len, long flags, long *revent)`
  - In JMS this is just an extended variant of the previous MessageProducer and MessageConsumer scenario

Application Programs

- An AP can be:
  - client-side
  - server-side
  - both

What does a client AP look like?

```
#include "xatmi.h"
#include "tx.h"

int main(int argc, char **argv) {
        char* request;
        char* response;
        tx_begin();
        int result = tpcall(request, response, ...);
        if (response...) {
                tx_commit();
        } else {
                tx_rollback();
        }
        return 0;
}
```

What does a service AP look like?

Method must fit the following signature:

```
void (*func)(TPSVCINFO *)
```

For example:

```
#include "xatmi.h"

void foo(TPSVCINFO * svcinfo) {
        //optional
        tpreturn(TPSUCCESS, 0, svcinfo->data, svcinfo->len, 0);
}
```

Notice how similar this is to javax.jms.MessageListener:

```
public void onMessage(Message message) {
        //optional
        MessageProducer producer = session.createProducer
                (message.getJMSReplyTo());
        producer.send(message);
}
```

Why should I be interested?

Why should I be interested?

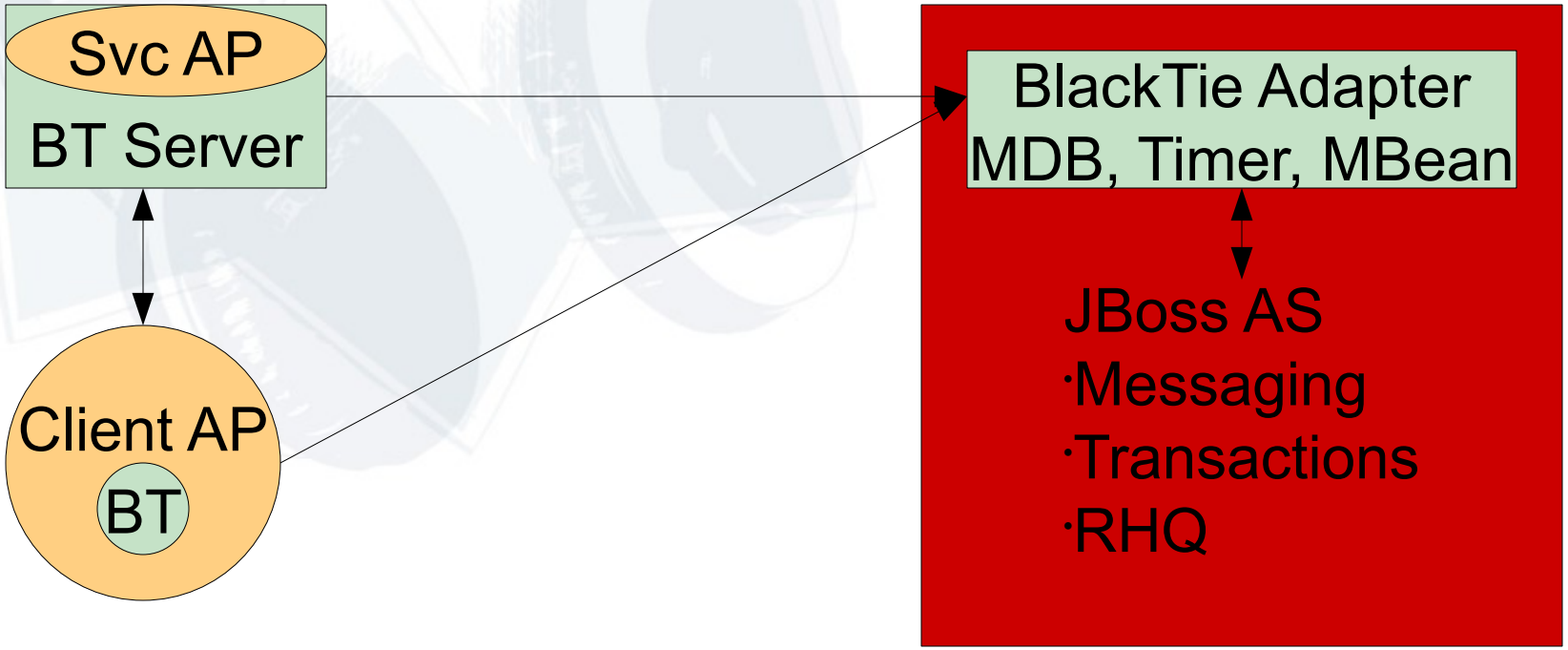
- For end users of XATMI and TX:
  - BlackTie provides an open source alternative to a previously difficult domain to migrate applications from
- Also allows Java consultancies an opportunity to assist their customers with migration from incumbent players in this area

What is BlackTie?

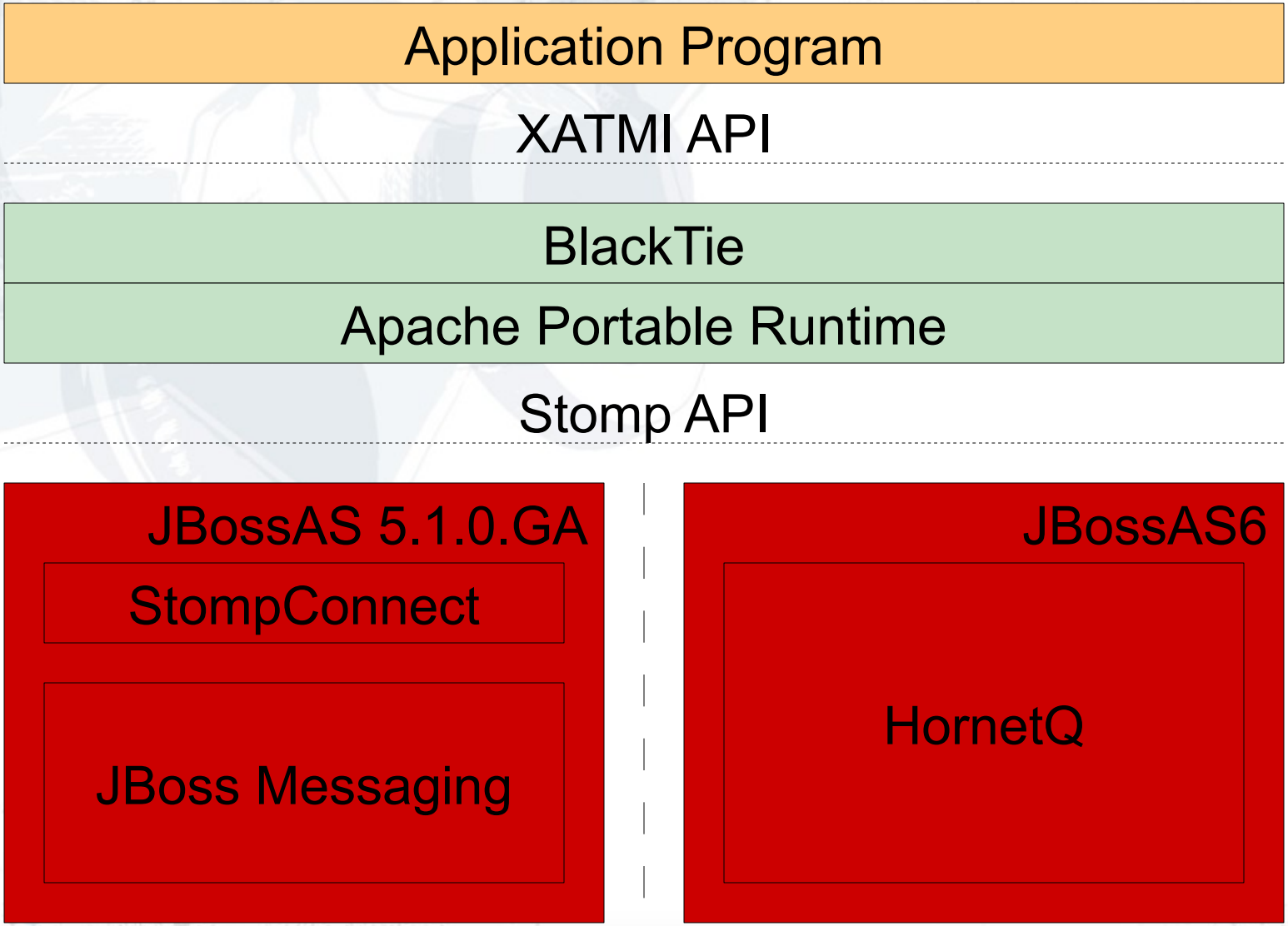
What is BlackTie?

- BlackTie provides an implementation of the XATMI and TX specifications
- BlackTie provides a coordinator for XA aware datasources
  - Appears like a transaction manager to the resource

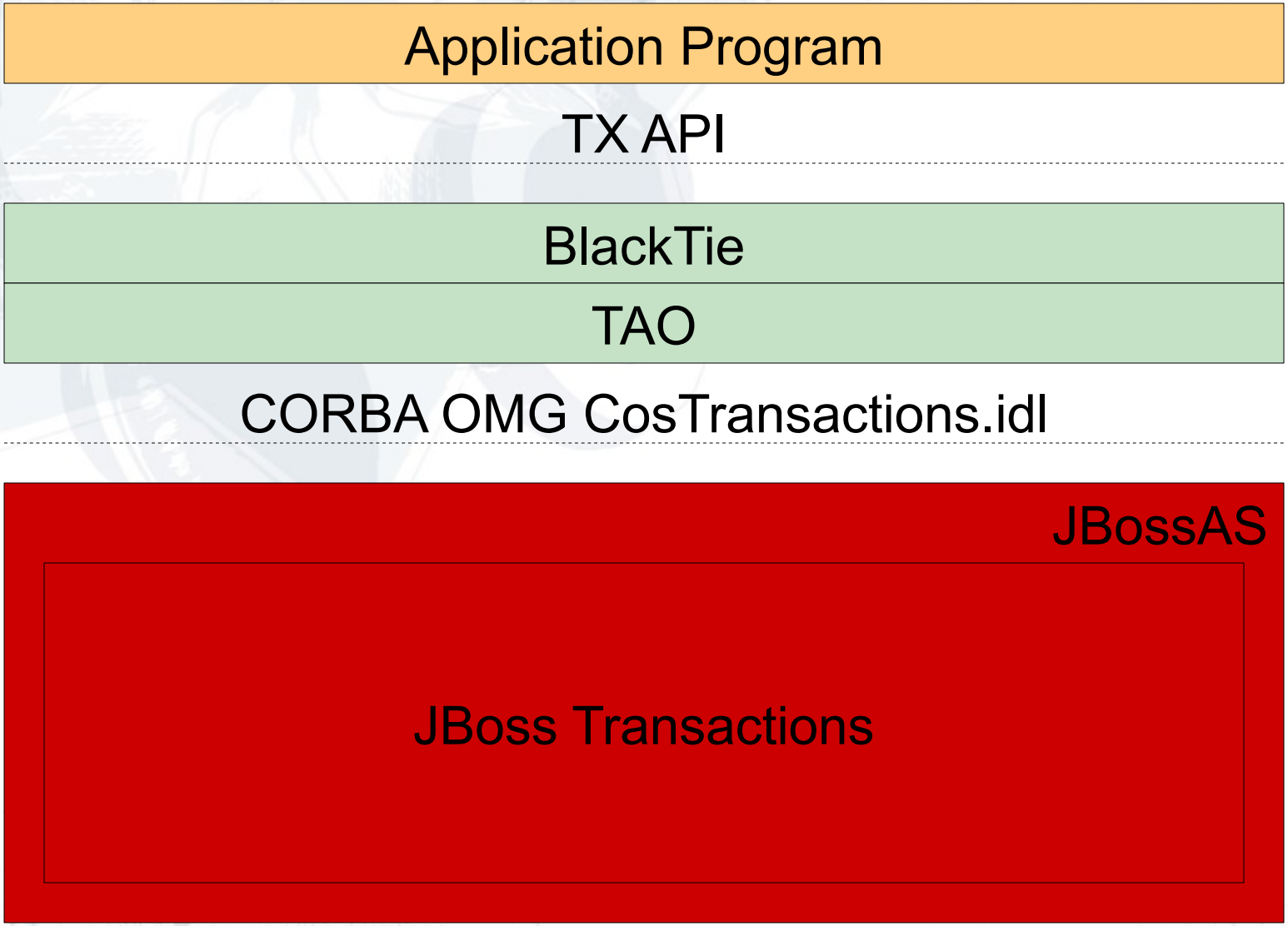
How does BlackTie fit in the AS?



Architecture: XATMI Call Path



Architecture: TX Call Path

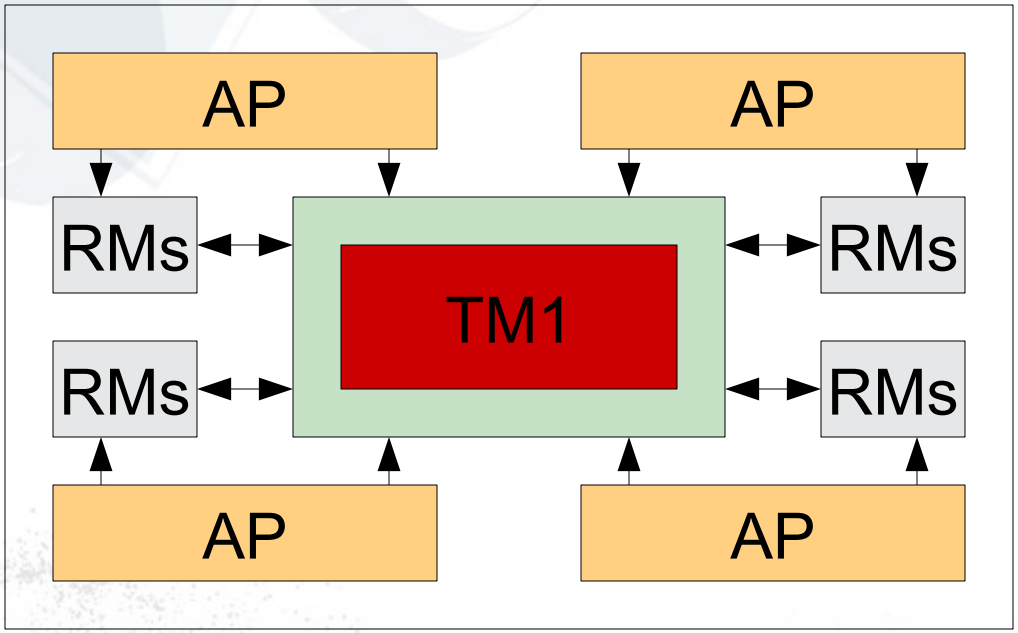


Commitment Protocol used by BlackTie

- Two-phase commit with presumed rollback
  - RM rollbacks any transaction not P1P2C
- Usual transactional attributes you would expect from JBoss:
  - Atomicity - all-or-nothing
  - Consistency – e.g. referential integrity
  - Isolation – effects invisible till tx completes
  - Durability – changes survive failures

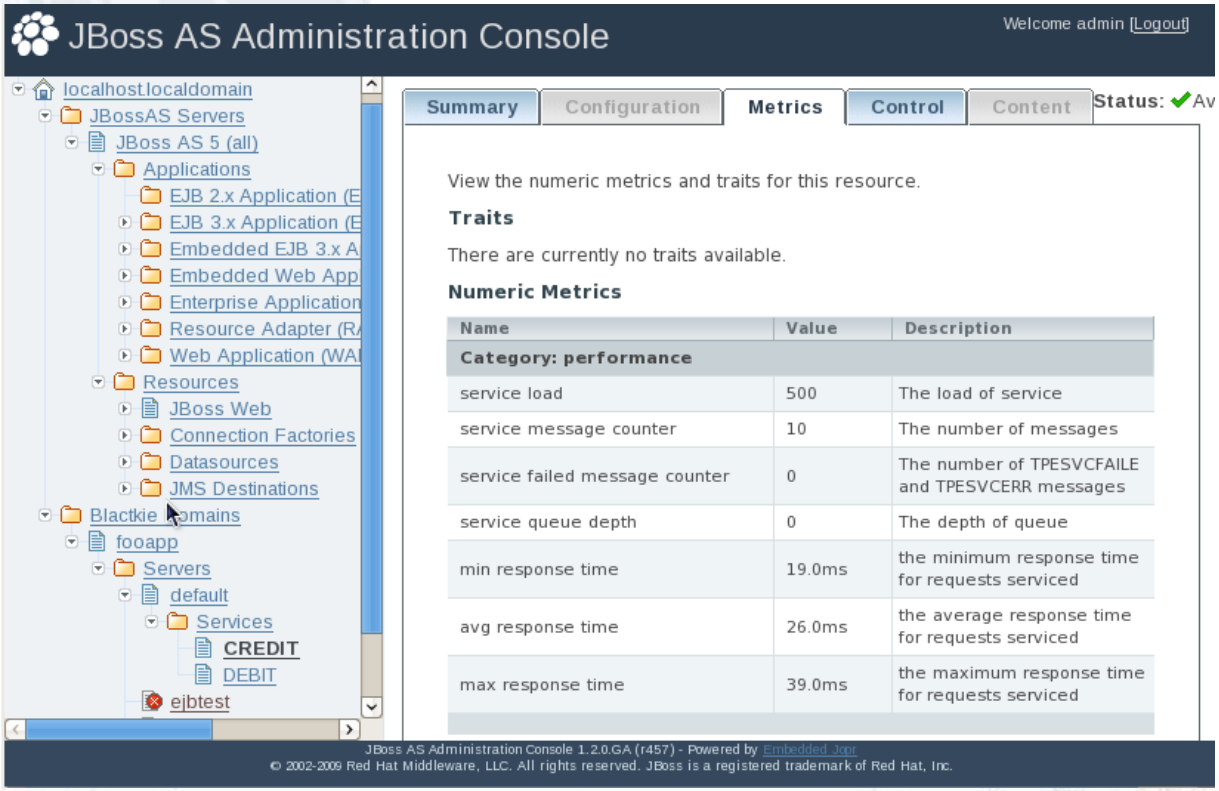
TM Domain

- With BlackTie all domains can be configured to share the same transaction manager



BlackTie Administration

- We have provided an RHQ plugin for JBoss
- Allows control of the external BlackTie servers
  - Start/stop
  - Advertise
  - Unadvertis
  - Metrics
    - TPS/FPS

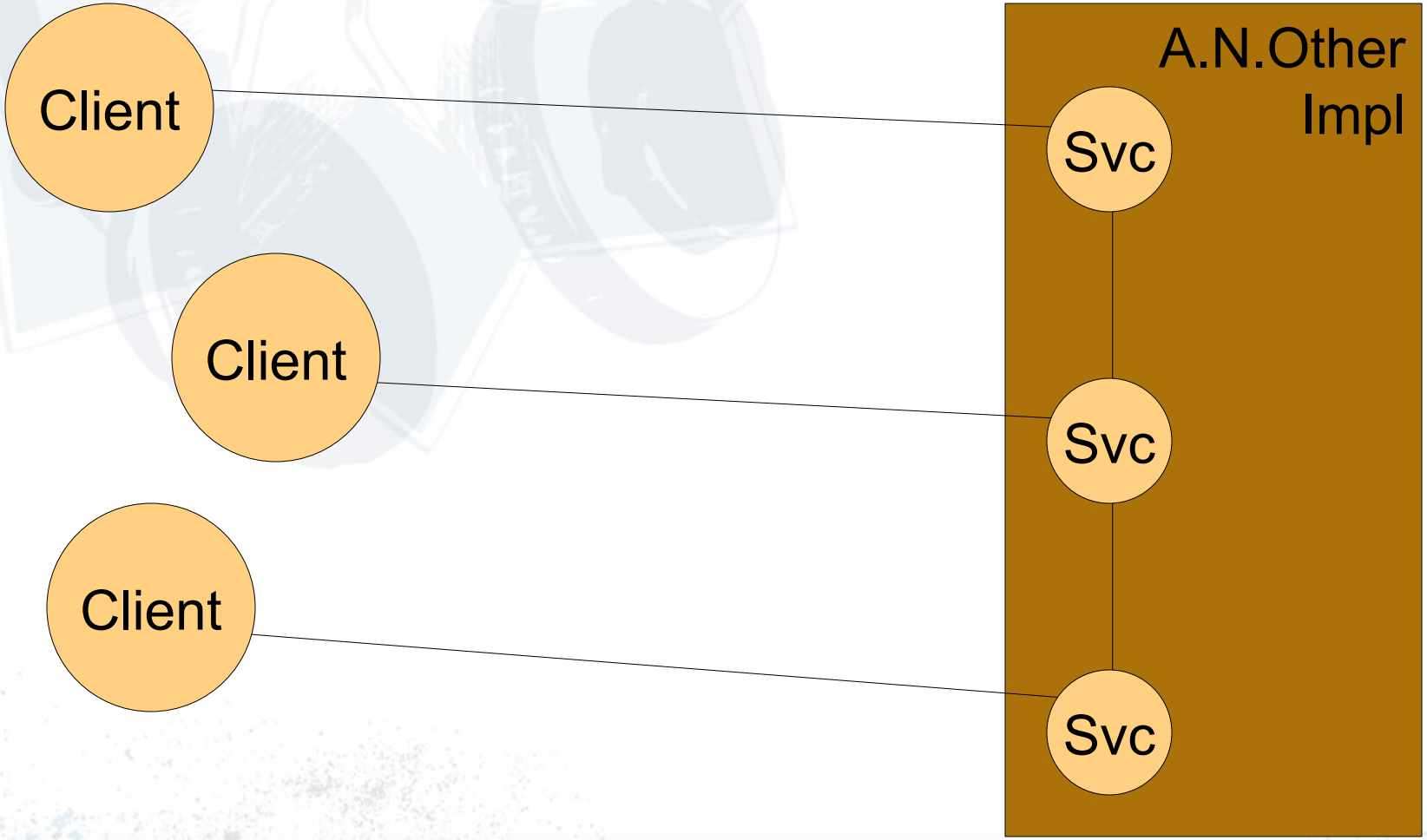


How can I use BlackTie today?

How can I use BlackTie today?

- There are two primary use cases we will now work through:
  - Migration of a total XATMI/TX application to JBoss AS
  - Migration of the client-side code and an upgrade of the services to native AS objects

HOWTO: Migrate your XATMI Application in its entirety



Simple!

- BlackTie is designed to be a drop-in replacement
- Code that has been written against the XATMI and TX APIs can be deployed directly onto JBoss without alteration!

Example Walkthrough

- We are going to walkthrough the migration of an ATM application

Deploying BlackTie

- Make sure you have JBoss running with BlackTie admin services deployed!
  - Unzip JBoss and BlackTie
  - Call ant jts in <JBOSS_HOME>/docs/examples/transactions
  - Edit <JBOSS_HOME>/server/all/conf/jbossts-properties.xml to change the CONFIGURATION_FILE to NAME_SERVICE
  - Configure setenv.[sh|bat]
  - Deploy blacktie-admin-services-[VERSION].ear
  - Copy blacktie-admin-services/btconfig.xml to the server
  - Deploy blacktie-rhq-plugin-[VERSION].jar to the admin-console
  - Start the AS: ./run.[sh|bat] -c all

Deploying the Example

- Follow these steps:
  - `. <BLACKTIE_HOME>/setenv.[sh|bat]`
  - `generate_server -Dservice.names=CREDIT,DEBIT -Dserver.includes=CreditService.c,DebitService.c`
  - `btadmin startup`
  - `generate_client -Dclient.includes=client.c`
  - `./client`

A look at the code: Client

```
#include "debit.h"
#include "credit.h"
#include "xatmi.h"
#include "tx.h"
int main(int argc, char **argv) {
        int status;
        DEBIT_T* debitBuf; // ALLOCATE & INITIALIZE
        CREDIT_T* creditBuf; // ALLOCATE & INITIALIZE
        char *retbuf; // ALLOCATE & INITIALIZE
        status = tx_open(); // Open connection to transaction manager
        if (status == 0) {
                status = tx_begin(); // Start a transaction
                if (status == 0) {
                        // Debit the account
                        status = tpcall("DEBIT", debitBuf, &retbuf, ...);
                        if (status == 0) {
                                // Credit the account
                                status = tpcall("CREDIT", ....);
                                if (status == 0) {
                                        // Commit the transaction
                                        status = tx_commit();
                                }
                        }
                }
        }
        sprintf("Status tperrno: %d status: %d", tperrno, status);
        // tpfree the buffers and return
}
```

A look at the code: Service

```
#include "xatmi.h"
#include "credit.h"

void CREDIT(TPSVCINFO * svcinfo) {
        char* buffer;
        int sendlen;
        CREDIT_T* creditBuf;

        creditBuf = (CREDIT_T*) svcinfo->data;
        sprintf((char*) "Credit called: acct_no: %d amount: %d",
                        creditBuf->acct_no, creditBuf->amount);

        sendlen = 10;
        buffer = tpalloc("X_OCTET", 0, sendlen);
        strcpy(buffer, "CREDITTED");
        tpreturn(TPSUCCESS, 0, buffer, sendlen, 0);
}
```

HOWTO: Upgrade your XATMI services but retain the client

Slightly more complex!

- You have a lot of options when upgrading your service tier from C to Java
- It is beyond the scope of this presentation to recommend the best approach to use
  - I am sure my colleagues will have shown several best-practices today!

What can be shown

- How you can keep the existing client, as it is
  - Will require migrating its runtime libraries to the BlackTie ones
- How you can deploy code into JBoss AS that understands the XATMI/TX invocations thus allowing you to invoke an EJB or whatever you like

Deploying the code

- Make sure you have JBoss running with BlackTie admin services deployed!
- Deploy some EJB business logic
  - `cd ejb`
  - `mvn install`
  - `cd ear`
  - `mvn install jboss:deploy`
- Deploy an XATMI adapter
  - `cd xatmi_adapter`
  - `mvn install`
  - `cd ear`
  - `mvn install jboss:deploy`
- Run the client as before

A look at the code: Client

- This is the same as before :)

The XATMI “Adapter”

- This is deployed as a special MDB which can marshal/unmarshal the XATMI parameters to/from the JMS message

A look at the code: Service

```
import org.jboss.blacktie.jatmibroker.xatmi.*;
import org.jboss.blacktie.jatmibroker.xatmi.mdb.MDBBlacktieService;
import org.jboss.ejb3.annotation.Depends;

@javax.ejb.TransactionAttribute(javax.ejb.TransactionAttributeType.NOT_SUPPORTED)
@javax.ejb.MessageDriven(activationConfig = {
                    @ActivationConfigProperty(propertyName = "destinationType",
propertyValue = "javax.jms.Queue"),
                    @ActivationConfigProperty(propertyName = "destination",
propertyValue = "queue/DEBIT") })
@Depends("jboss.messaging.destination:service=Queue,name=DEBIT")

public class DebitAdapterService extends MDBBlacktieService implements
                    javax.jms.MessageListener {
          public Response tpservice(TPSVCINFO svcinfo) throws ConnectionException {
                    X_COMMON rcv = (X_COMMON) svcinfo.getBuffer();
                    long acct_no = rcv.getLong("acct_no");
                    short amount = rcv.getShort("amount");
                    Context ctx = new InitialContext();
                    DebitRemote bean = (DebitRemote)ctx.lookup("DebitBean/remote");
                    String resp = bean.debit(acct_no, amount);
                    Connection connection = svcinfo.getConnection();
X_OCTET buffer = (X_OCTET)connection.tpalloc("X_OCTET", null, resp.length() + 1);
                    buffer.setByteArray(resp.getBytes());
                    return new Response(Connection.TPSUCCESS, 0, buffer, 0);
          }
}
```

A look at the code: EJB

```
import javax.ejb.Remote;
@Remote
public interface DebitRemote {
        public String debit(long acct_no, short amount);
}
```

```
import javax.ejb.Stateless;
import javax.ejb.TransactionAttribute;
import javax.ejb.TransactionAttributeType;

import org.jboss.ejb3.annotation.RemoteBinding;

@Stateless
@RemoteBinding(jndiBinding = "DebitBean/remote")
public class DebitBean implements DebitRemote {
        @TransactionAttribute(TransactionAttributeType.MANDATORY)
        public String debit(long acct_no, short amount) {
                return "DEBITTED";
        }
}
```

Points of note

- The transaction started in C is seamlessly placed onto the thread in Java by BlackTie
- Ideally we would have a TransactionAttributeType of NEVER but this is not possible
- Users are expected to extend our base class and provide the proper annotations

And finally: What next?

Interoperability

- Currently applications cannot interoperate between BlackTie and other XATMI providers
- Version 3 of BlackTie will focus on this
  - OSI TP
  - The Application Service Element

Routing

- Currently we are limited in our options for routing requests as the Stomp protocol does not support numeric headers
- We are somewhat dependent on when the Stomp protocol adds this, vote now!
  - http://groups.google.co.uk/group/stomp-spec

So, where can I get more information?

- http://jboss-blacktie.blogspot.com/
- http://www.jboss.org/blacktie
- http://community.jboss.org/en/blacktie
- irc://freenode.net/#blacktie
  - If you would like to become a contributor this is the place to start

Any Questions?