JUDCon

JBoss Users & Developers Conference

2010:Berlin

blacktie

Migrating to JBoss from Oracle Tuxedo using the BlackTie project

blacktie

JUDCon2010:Berlin JBoss Users & Developers Conference

Speaker overview

- Tom Jenkinson
- BlackTie project lead since 2008
- Previously a consultant specializing in integration projects 2005-2008
- Prior to that, an Arjuna JTS developer between 2000-2005

Agenda

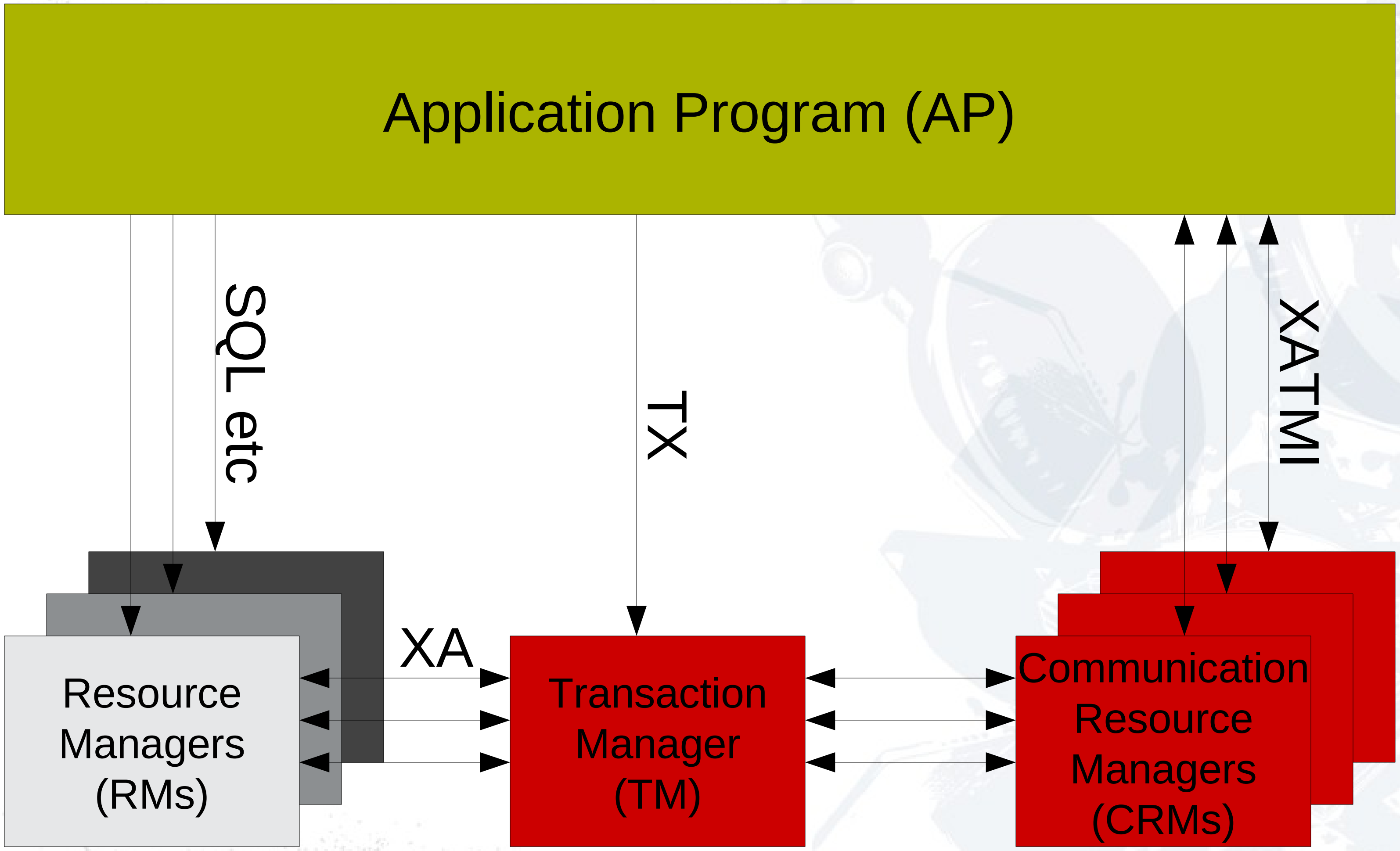
- Compare the features offered by Oracle Tuxedo and those provided by JBoss BlackTie
- Illustrate the steps to migrate applications from Oracle Tuxedo to JBoss BlackTie
- Discuss some of the difficulties involved and walk through examples of how these may be addressed using BlackTie today
- Highlight some of the features to assist users with migration that are being worked on for BlackTie 3

BlackTie

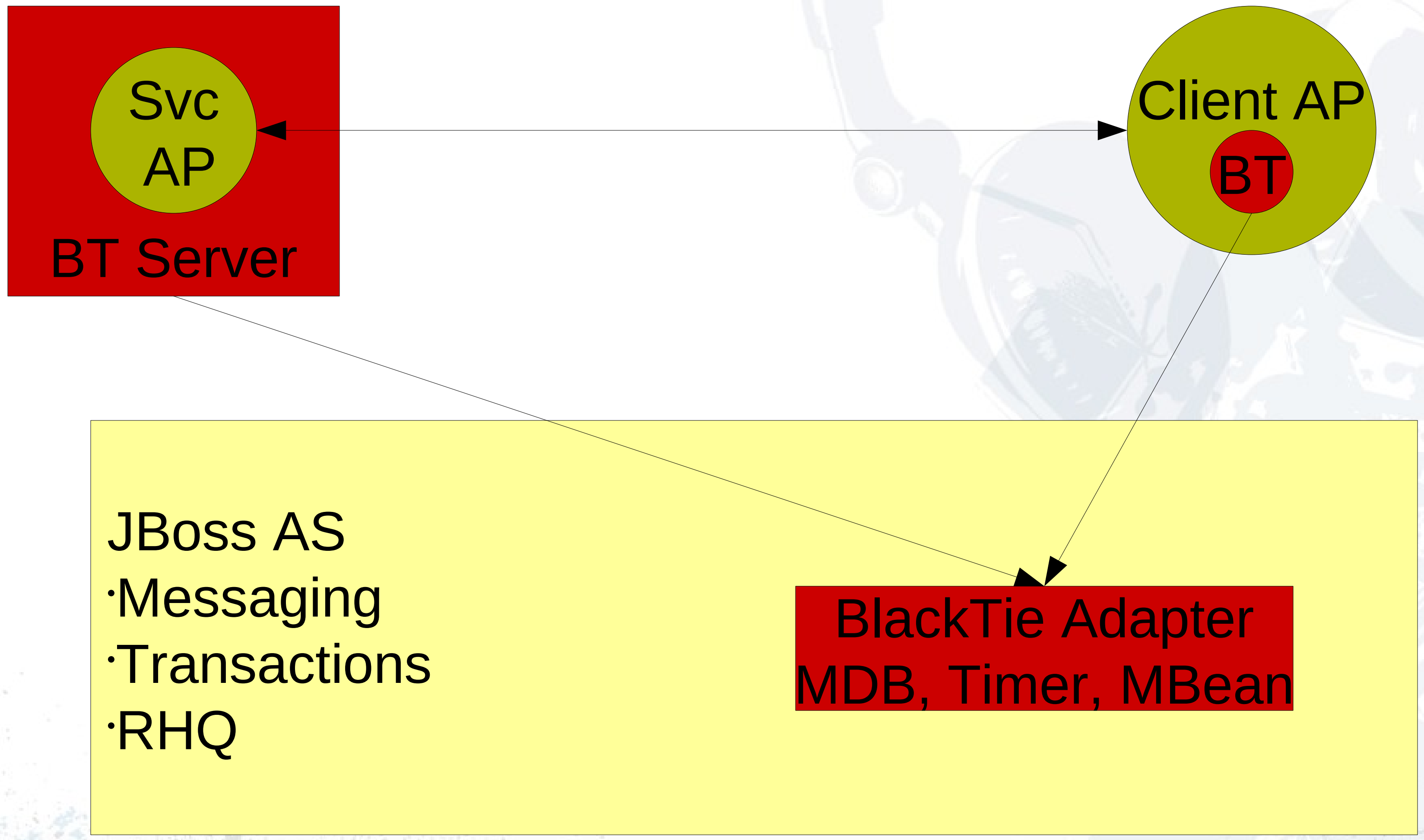
What is BlackTie?

- BlackTie is an implementation of X/Open XATMI and TX
- Provides bindings for Java and C
- Co-ordinates access to XA aware datasources such as Oracle and DB2
- Most of the services are provided by JBoss AS
  - Messaging, transactions, RHQ

What does BlackTie facilitate?



How does BlackTie fit in the AS?



What isn't addressed by the specs

- Scalability
- Administration
- Performance
- Security
- These qualities and features help to distinguish vendors

Why should I be interested?

- BlackTie provides an open source alternative to a previously difficult domain to migrate applications from
- Consultancies specialising in JBoss technologies now have an easier migration path to recommend to their customers currently deployed to Tuxedo

Migration Guide

Migrating from Tuxedo to BlackTie

- Simplified checklist:
  - Functional comparison
  - Build tool support
  - Deployment considerations
    - Hardware
  - Runtime capabilities
    - Administration

First stop: Functional Comparison

- Identifying dependencies on proprietary Tuxedo APIs:

Tuxedo	BlackTie
Jolt	JAB
WTC	jatmibroker-xatmi
View	X_C_TYPE
FML	Not currently available
tmadmin	btadmin
/Q	XATMI
Eventing API	XATMI

What type of user are you?

- After this analysis, you will be in one of the following positions:
  1) You have developed purely against the XATMI API and are in good luck!
  2) You have made use of proprietary extensions which BlackTie already provides equivalents for
  3) You have used proprietary extensions where no equivalent is currently available

What type of user are you?

- To turn case 2 into case 1, we can usually do this by either:
  - Provide an adapter which maps Oracle API calls to the BlackTie equivalent – clearly not open source!
  - Search and replace where straight mappings are present, e.g. usage of the CARRAY buffer

What type of user are you?

- Given a case 1 or 2 (with the modifications for case 2 made), the showstoppers are already done!
- All that remains is to migrate the Tuxedo configuration files to BlackTie ones and start the actual phasing in of BlackTie servers.

Java Porting

Jolt vs JAB

- Simple Java API for clients
- Javadoc available:
  - http://docs.jboss.org/blacktie/docs/2.0.0.Final/javadoc/jatmibroker-xatmi/overview-summary.html

WTC vs jatmibroker-xatmi

- Pure Java implementations of the XATMI API
- Allow clients and services to be developed
- Example available:
  - example-mdb-xatmi-service

Buffers

Buffer Support

- Tuxedo
  - X_OCTET [CARRAY, STRING], X_C_TYPE, X_COMMON [VIEW(32)]
  - FML(32)
- BlackTie
  - X_OCTET, X_C_TYPE, X_COMMON
  - XML – coming soon!
    - https://jira.jboss.org/browse/BLACKTIE-327

VIEWS

- Described in a view descriptor file ending .v.
- Ran through the viewc compiler to create the .V and .h files
- Each view file is composed of the following:
  - VIEW <name>
  - #type cname fbname count flag size null

Purpose of VIEWS

- This information allows Tuxedo to perform the following major functions:
  1) Header creation
  2) NULL initialization
  3) Maximise platform support – endian etc
  4) Routing

BlackTie equivalent

- X_C_TYPE and X_COMMON can be used with the following caveats:
  - Routing is not yet available – coming soon!
  - Headers must be created by the user by hand
  - Buffers are described in the btconfig.xml file
  - Nulls are initialized to \0, no freedom here – If it is required we have Jira!

Example comparison

- BlackTie

```
<BUFFER name="dc_buf">
         <ATTRIBUTE id="input" type="char[]" arrayLength="100" />
         <ATTRIBUTE id="output" type="int" />
         <ATTRIBUTE id="failTest" type="int" />
</BUFFER>
```

- Tuxedo

```
VIEW dc_buf
char input - 100 - 0
int output - 1 - 0
int failTest - 1 - 0
```

FML

- Tuxedo provides the FML (Field Manipulation Language) format buffers.
- FML has the following benefits over and above those provided by VIEW:
  - Array lengths need not be known
  - Nested record sets are allowed

BlackTie equivalent of FML

- BlackTie does not support this format of buffer. It is a proprietary extension to XATMI provided by Tuxedo.
- We are working on providing a buffer format with similar characteristics.
  - https://jira.jboss.org/browse/BLACKTIE-330
    - Provide an API for a nested buffer format
  - https://jira.jboss.org/browse/BLACKTIE-334
    - Allow array lengths to be determined at runtime

Current work

- Available from: https://svn.jboss.org/repos/blacktie/trunk/blacktie-nbf
    - int btaddattribute(char* buf, char* id, char* value, int len)
    - int btgetattribute(char* buf, char* id, int index, char* value, int* len)
    - int btsetattribute(char* buf, char* id, int index, char* value, int len)
    - int btdelattribute(char* buf, char* id, int ind)

Queues

- Tuxedo provides an extension API called /Q
- It provides the following:
    - tpqctl – A control structure which encodes priority etc
    - int tpenqueue(char *qspace, char *qname, TPQCTL *ctl, char *data, long len, long flags)
    - int tpdequeue(char *qspace, char *qname, TPQCTL *ctl, char **data, long *len, long flags)

BlackTie equivalent

- Due to the way that BlackTie is architected with lightweight services underpinned by JBoss queues, we are able to provide a very similar API which stays true to the original XATMI specification

BlackTie queues

- The API (btxatmi.h) is composed of:
  - char* btalloc(msg_opts_t* ctrl, char* type, char* subtype, long size)
  - msg_opts
    - Priority supported
  - int tpacall(char * svc, char* idata, long ilen, long flags)
    - TPNOREPLY should be set
  - tpadvertise/tpservice/tpunadvertise
- To enqueue a message, simply tpacall the service

Dequeuing a message

- To create a simple dequeue operation with definition:
    - int btdequeue(char *svc, msg_opts *ctl, char **data, long *len, long flags)

Defining a dequeue operation

```
// Declare some variables to store the dequeued message
long mlen; char* message; SynchronizableObject lock = new SynchronizableObject();
// Define a simple service to accept incoming data
void dqservice (TPSVCINFO* tpsvcinfo) {
   mlen = svcinfo->len;
   message = svcinfo->data;
   int err = tpunadvertise(SERVICE);
   lock.notify(); // Notify the dequeue call that a message has been received
}
// Define a method
int btdequeue(char *svc, msg_opts *ctl, char **data, long *len, long flags) {
   int err = tpadvertise(SERVICE, dqservice);
   lock.wait();
   *len = mlen;
   memcpy(*data, message, len);
   return 0;
}
```

Eventing

- Tuxedo provides an API to allow clients to set unsolicited message handlers
- Allows actors to broadcast events to these through the following methods:
    - int tpbroadcast(char *lmid, char *usrname, char *cltname, char *data, long len, long flags)
    - void (*tpsetunsol (void (*func) (char *data, long len, long flags)))

BlackTie equivalent

- BlackTie identified these capabilites are achieved through the standard XATMI API, with a minor semantic change.
- XATMI has an understanding that a single service will respond to a single invocation(p2p messaging)
- Our underlying JBoss JMS implementation also supports topics (pub/sub) messaging.

BlackTie equivalent

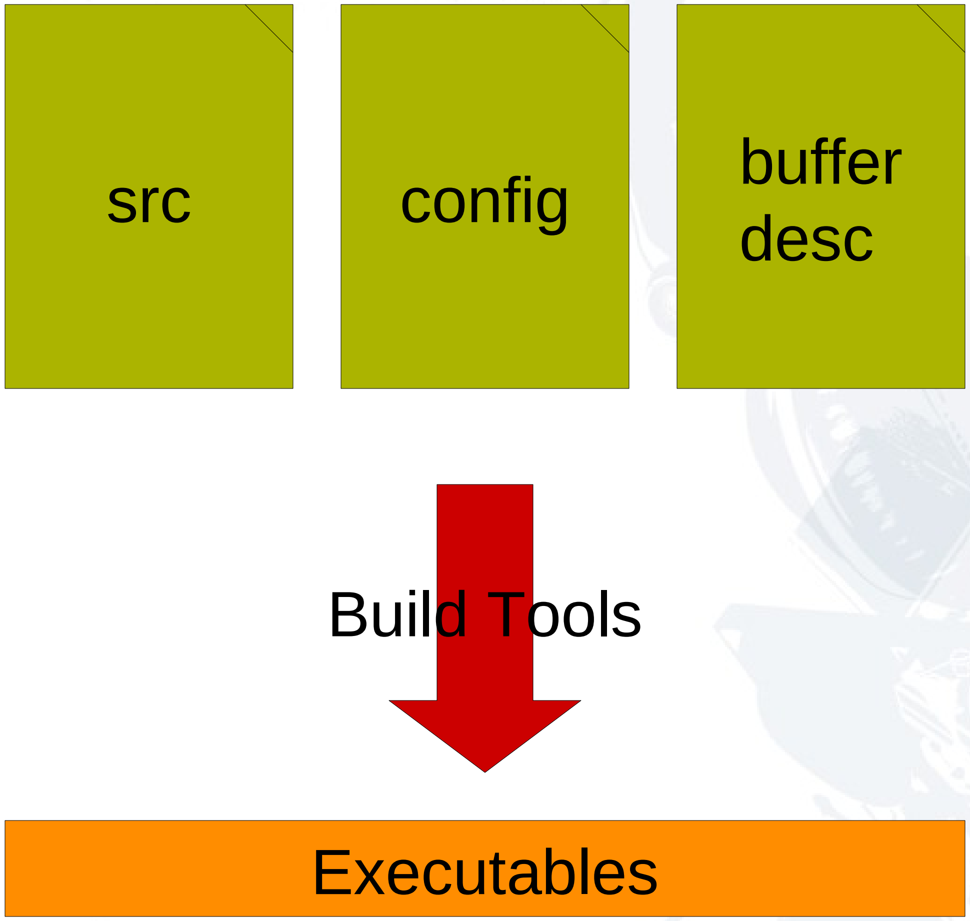
- We have an outstanding Jira pending to update BlackTie to support using topics for service invocation, which when coupled with the conventions discussed for queues means that we can achieve unsolicited messaging with the XATMI API:
  - https://jira.jboss.org/browse/BLACKTIE-303

Build Tool Support

Build Tools

- The main tools that are used in Tuxedo to build the executables are:
  - buildserver, buildclient, viewc, tmloadcf
- BlackTie provides the following equivalents:
  - generate_server, generate_client

Familiar development processes



UBBConfig vs btconfig.xml

- A ubbconfig is composed of:
  - RESOURCES
    - System wide limits – e.g. maxservers
  - MACHINES
    - The machines that are to be used in the domain are represented here
  - GROUPS
    - Allows several machines to be administered as a single entity. Specify transaction manager and XA open configuration
  - NETGROUPS
    - Group two+ machines and prioritise traffic

UBBConfig vs btconfig.xml

- A ubbconfig is composed of:
  - NETWORK
    - Configure the network ports and encryption
  - SERVERS
    - Specifies the command line options to the server process, a conversational server only supports conversation services. Also a file containing the environment to provide to the server is specified here
  - SERVICES
    - Provide optional configuration to the services – the full list of services is not required, although all the routines must be provided to buildserver

UBBConfig vs btconfig.xml

- A ubbconfig is composed of:
  - INTERFACES
    - CORBA interfaces
  - ROUTING
    - A routing table can be configured, e.g. for buffer type FML with certain field of various ranges to a particular group.

UBBConfig vs btconfig.xml

- A btconfig file has the following sections:
  - MACHINES
    - The paths to the machines working directory and executable
  - SERVERS
    - The group of machines that should be considered as a single server for administration and the complete list of services for this server (either conversational or rpc)

UBBConfig vs btconfig.xml

- A btconfig file has the following sections:
  - XA_RESOURCES
    - The list of XA resources for this domain
  - ORB
    - Network configuration
  - MQ
    - Network configuration
  - JMX
    - Network configuration

UBBConfig vs btconfig.xml

- A btconfig file has the following sections:
  - ENV_VARIABLES
    - The list of environment variables to make available to the process (optionally constrained by a configuration attribute
  - BUFFERS
    - The format of the X_COMMON and X_C_TYPE buffers is provided here.

Differences

- Tuxedo needs the full list of methods that are to be exposed as services
- Tuxedo does NOT need the full list of names that the services are to be exposed as
- BlackTie needs both, it needs the list of names to be configured in a servers btconfig.xml and needs the methods providing to it during generate_server

Differences

- BlackTie uses the list of services in btconfig.xml to validate that the same server can only export the same set of services:

```
<SERVERS>
         <SERVER name='example'>
                  <SERVICE_NAMES>
                           <SERVICE name='BAR' />
                           <SERVICE name='FOO' />
                  </SERVICE_NAMES>
         </SERVER>
</SERVERS>
```

Administration

Command-line administration

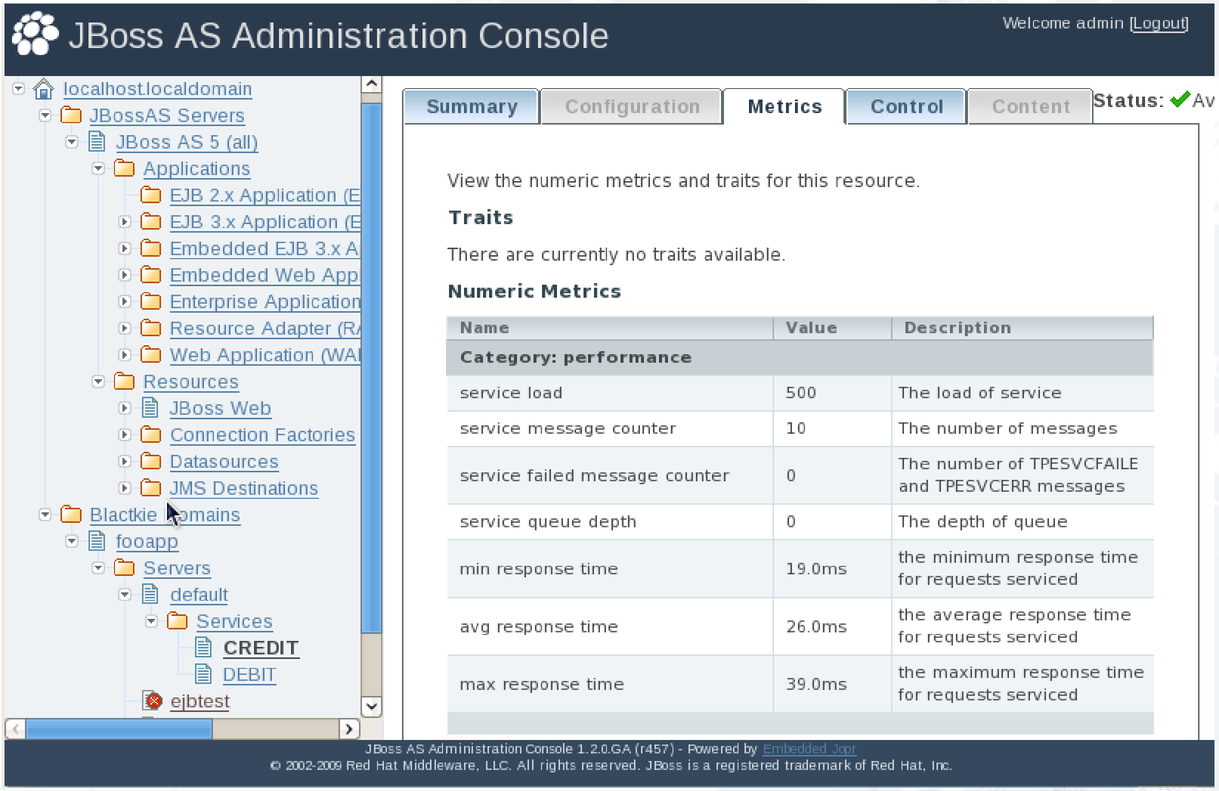
- Oracle Tuxedo uses a process called tmadmin to start and stop the servers at runtime.
  - It inspects the binary tuxconfig file to determine various defaults such as the local server name.
- BlackTie uses a process called btadmin to similar effect

Graphical Administration

- Tuxedo WebGUI Hasn't been updated since 7.1 according to Oracle's website
- Provides a graphical view of the domain using Java Applets
- BlackTie provides a state of the art RHQ plugin to JBossAS

BlackTie RHQ GUI

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



View the numeric metrics and traits for this resource.

Traits

There are currently no traits available.

Numeric Metrics

Name	Value	Description
Category: performance		
service load	500	The load of service
service message counter	10	The number of messages
service failed message counter	0	The number of TPESVCFAILE and TPESVCERR messages
service queue depth	0	The depth of queue
min response time	19.0ms	the minimum response time for requests serviced
avg response time	26.0ms	the average response time for requests serviced
max response time	39.0ms	the maximum response time for requests serviced

JBoss AS Administration Console 1.2.0.GA (r457) - Powered by Embedded Jopr
© 2002-2009 Red Hat Middleware, LLC. All rights reserved. JBoss is a registered trademark of Red Hat, Inc.

Using Security

- BlackTie supports user group role authentication
- Authentication is done at the queue level
- This is similar to Tuxedo but will require some effort to migrate these to BlackTie as they are both proprietary mechanisms

Current limitations

Interoperability

- Currently applications cannot interoperate between BlackTie and Tuxedo
- Version 3 of BlackTie focuses on this
  - All XATMI implementations are required to implement a common wire protocol
  - OSI TP
  - https://jira.jboss.org/browse/BLACKTIE-64

Routing

- Tuxedo allows the routing of FML and VIEW buffers based on the table declared in the users UBBCONFIG file.
- BlackTie does not currently support routing, although we are working on delivering this via integration with JBoss ESB and our XML buffer format:
  - https://jira.jboss.org/browse/BLACKTIE-125

COBOL

- Tuxedo supports COBOL clients and services, BlackTie does not..
  - Any help will be gratefully received ;)

Examples!

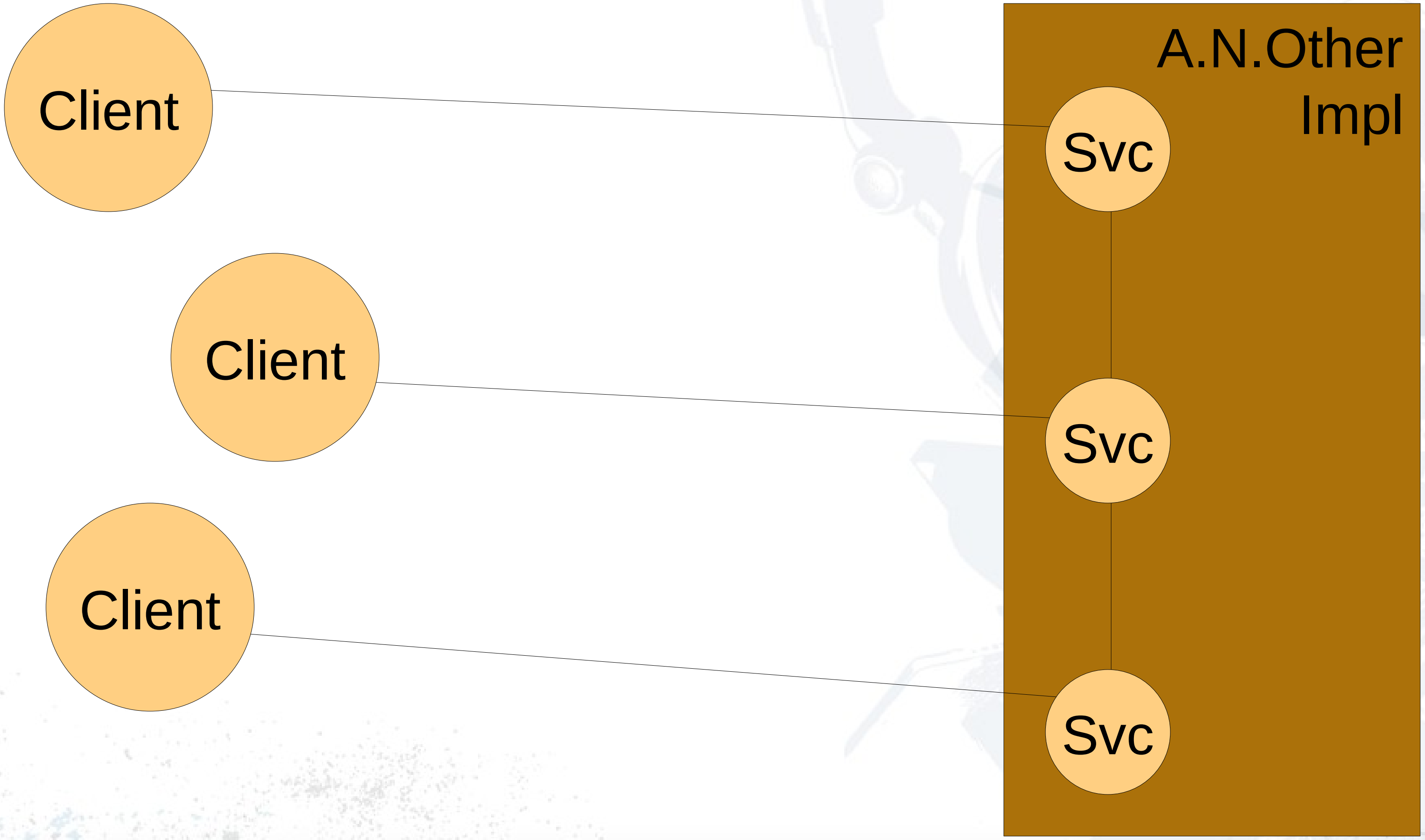
Walkthrough

- We will now take a look at some more examples from BlackTie:
  - 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

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

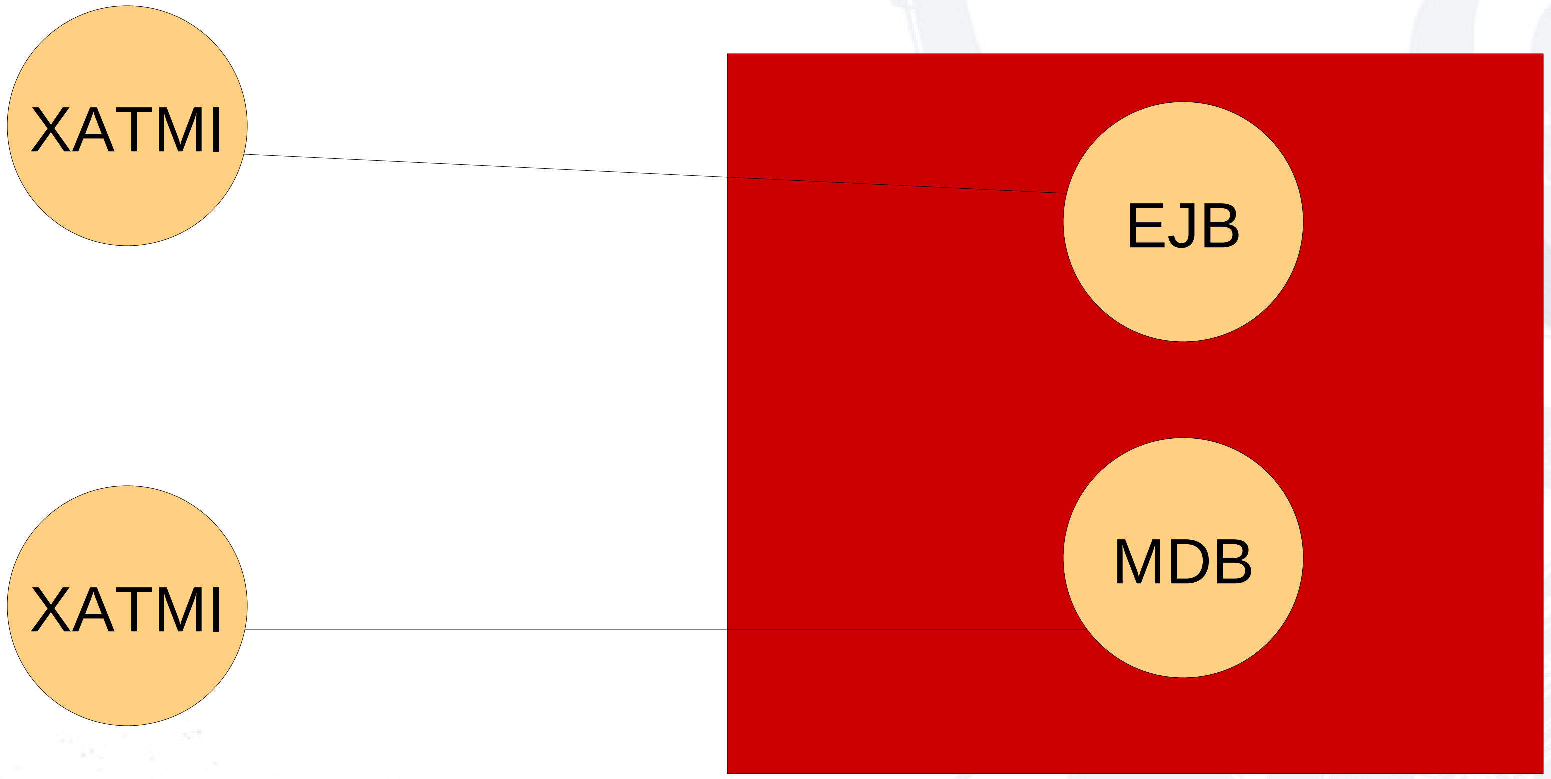
HOWTO: Migrate your XATMI Application in its entirety



Deploying the Example

- Follow these steps:
  - `. <BLACKTIE_HOME>/setenv.[sh|bat]`
  - `cd <BLACKTIE_HOME>/examples/integration1/xatmi_service`
  - `generate_server -Dservice.names=CREDIT,DEBIT -Dserver.includes=CreditService.c,DebitService.c`
  - `btadmin startup`
  - `cd <BLACKTIE_HOME>/examples/integration1/client`
  - `generate_client -Dclient.includes=client.c`
  - `./client`

HOWTO: Upgrade your XATMI services but retain the client



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?

blacktie