User Guide

Using BlackTie

by Tom Red Hat Jenkinson, Michael Red Hat Musgrove, and Amos Red Hat Feng

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts* [https://fedorahosted.org/liberation-fonts/] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keycaps and key combinations. For example:

> To see the contents of the file `my_next_bestselling_novel` in your current working directory, enter the `cat my_next_bestselling_novel` command at the shell prompt and press **Enter** to execute the command.

The above includes a file name, a shell command and a keycap, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from keycaps by the hyphen connecting each part of a key combination. For example:

> Press **Enter** to execute the command.
>
> Press **Ctrl**+**Alt**+**F2** to switch to the first virtual terminal. Press **Ctrl**+**Alt**+**F1** to return to your X-Windows session.

The first paragraph highlights the particular keycap to press. The second highlights two key combinations (each a set of three keycaps with each set pressed simultaneously).

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in `mono-spaced bold`. For example:

> File-related classes include `filesystem` for file systems, `file` for files, and `dir` for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).

To insert a special character into a **gedit** file, choose **Applications** → **Accessories** → **Character Map** from the main menu bar. Next, choose **Search** → **Find…** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

`Mono-spaced Bold Italic` or ***Proportional Bold Italic***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

To connect to a remote machine using ssh, type `ssh` *`username@domain.name`* at a shell prompt. If the remote machine is `example.com` and your username on that machine is john, type `ssh john@example.com`.

The `mount -o remount` *`file-system`* command remounts the named file system. For example, to remount the `/home` file system, the command is `mount -o remount /home`.

To see the version of a currently installed package, use the `rpm -q` *`package`* command. It will return a result as follows: *`package-version-release`*.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

Publican is a *DocBook* publishing system.

1.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in `mono-spaced roman` and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in `mono-spaced roman` but add syntax highlighting as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
  public static void main(String args[])
      throws Exception
  {
    InitialContext iniCtx = new InitialContext();
    Object         ref    = iniCtx.lookup("EchoBean");
    EchoHome       home   = (EchoHome) ref;
    Echo           echo   = home.create();

    System.out.println("Created Echo");

    System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
  }
}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update

will apply. Ignoring a box labeled 'Important' will not cause data loss but may cause irritation and frustration.

Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

2. We Need Feedback!

You should over ride this by creating your own local Feedback.xml file.

About This Guide

The Programmers Guide contains information on how to use BlackTie. This document provides a detailed look at the design and operation of BlackTie. It describes the architecture and the interaction of components within this architecture.

1.1. Audience

This guide is most relevant to engineers who are responsible for developing using BlackTie. Although this guide is specifically intended for service developers, it will be useful to anyone who would like to gain an understanding of transactions and how they function.

1.2. Prerequisites

This guide assumes a basic familiarity with Java service development and object-oriented programming. A fundamental level of understanding in the following areas will also be useful:

- General understanding of the APIs, components, and objects that are present in Java applications.
- A general understanding of the Windows and UNIX operating systems.

Standards

2.1. X/Open

BlackTie implements the following standards:

- XATMI

  http://www.opengroup.org/pubs/catalog/c506.htm

- TX

  http://www.opengroup.org/pubs/catalog/u011.htm

Using Buffers with BlackTie

BlackTie supports all three of the buffer types defined in the XATMI specification.

Namely these are:

1. X_OCTET
2. X_COMMON
3. X_C_TYPE

BlackTie also supports the Nested Buffer Format described below.

3.1. X_OCTET

The X_OCTET buffer type operates exactly as per is defined within Chapter 9 of the specification with the following points of interest:

- A non-zero size MUST be provided to tpalloc and tprealloc calls
- The entire buffer is initialized to \0 (NULL) prior to returning to the client

3.2. X_COMMON/X_C_TYPE

For our implementation of these buffer types we have required the user to configure the structure of the buffer in the btconfig.xml

3.3. How to use Nested Buffers in BlackTie

3.3.1. Defining a schema

Nested Buffer Format is defined in a xsd schema file, such as employee.xsd:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://www.jboss.org/blacktie" targetNamespace="http://www.jboss.org/blacktie" elementFormDefault="qualified">
   <xsd:element name="employee" type="employee_type"/>
   <xsd:complexType name="employees_type">
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
         <xsd:element name="employee" type="employee_type"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="employee_type">
```

```
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="name" minOccurs="0" maxOccurs="unbounded">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:maxLength value="8"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>

 <xsd:element name="id" minOccurs="0" maxOccurs="unbounded" type="xsd:long" default="0">
        </xsd:element>
      </xsd:choice>
   </xsd:complexType>
</xsd:schema>
```

All xsd schema files should been placed in buffers directory.

Get a nested buffer in blacktie, you should use

```
char* buffer = tpalloc("BT_NBF", "<buffer_name>", 0);  // in C++
BT_NBF buffer = (BT_NBF) connection.tpalloc("BT_NBF", "<buffer_name>", 0);   // in Java
```

After that, the following options are available:

- add a new attribute in buffer

  int btaddattribute(char** buf, char* attributeId, char* attributeValue, int len); // in C++

- retrieve a set attribute

  int btgetattribute(char* buf, char* attributeId, int attributeIndex, char* attributeValue, int* len); // in C++

- set an existing attribute

  int btsetattribute(char** buf, char* attributeId, int attributeIndex, char* attributeValue, int len); // in C++

- remove an existing attribute

  int btdelattribute(char* buf, char* attributeId, int attributeIndex); // in C++

e.g.

```
// add a attribute in buffer
```

```
char name[16];
char value[16];
int len = 16;

char* buf = tpalloc((char*) "BT_NBF", (char*) "employee", 0);
strcpy(name, "test");

rc = btaddattribute(&buf, (char*) "name", name, strlen(name));
rc = btgetattribute(buf, (char*) "name", 0, (char*) value, &len);

strcmp(value, "test");
```

more examples can be founded under <BLACKTIE_HOME>/example/nbf, and this example also runs as part of the run_all_samples script.

Services

4.1. Building XATMI services and clients

This article should explain how to work with the Blacktie framework in order to develop your own XATMI services and clients. We will discuss how to run the code generation tool along with how to provide the configuration files that are required on the client and server side.

4.1.1. Pre-requisites

The provided <BLACKTIE_BIN_DIR>/setenv.[sh|bat] should be executed to put the build tools in the environment.

4.1.1.1. WINDOWS ONLY

Install Visual V++ 2008 Express Edition (only used to provide the compiler and linker on Windows)

The PATH environment variable will need the runtime DLLs in it

```
C:\Program Files\Microsoft Visual Studio 9.0\VC\redist\Debug_NonRedist
\x86\Microsoft.VC90.DebugCRT
```

The PATH environment variable will also need the VC++ tools in it

```
C:\Program Files\Microsoft Visual Studio 9.0\VC\vcvarsall.bat
```

4.1.2. Configuring the server, services and clients

Each of the following configuration files must be present at runtime inorder for Blacktie servers or clients to operate correctly

- btconfig.xml (this file although of the same structure, the content will be different for windows and for linux)

The schemas that these files must validate against are in <BLACKTIE_BIN_DIR>/schemas/xsd, you can read more about providing configuration here.

4.1.3. Running the code generation tool

When building clients or servers using the provided ANT script you will need to make sure you have set the environment in the manner indicated in <BLACKTIE_BIN_DIR>/setenv.[sh|bat]

An example run configuration for the code generation tool would then look like:

```
<BLACKTIE_BIN_DIR>/bin/generate_server.<sh|bat>
```

```
  -Dserver.includes=BarService.c -Dservice.names=BAR
```

```
<BLACKTIE_BIN_DIR>/bin/generate_client.<sh|bat>
  -Dclient.includes=client.c
```

4.1.4. Running a generated server and client

The following are the requirements to run the generated server and client:

The LD_LIBRARY_PATH or PATH must be set to include the <BLACKTIE_BIN_DIR>/lib folder (done by setenv.[sh|bat]

The commands must be executed from the directory relative to where the configuration files exist

Assuming the pre-requisites are adhered to all that must be done is to execute

`server[.exe] -c [configuration_dir] -i [unique_id]` and then

`client[.exe]` (if the client is not in the location of the configuration files you need to set the environment variable `BLACKTIE_CONFIGURATION_DIR` .

4.2. XATMI Services and BlackTie

Here a service is defined as a unit of business logic that implements the tpservice(SVCINFO) method as defined in the XATMI specification section 3.3.

In BlackTie services are broadly categorised as either:

- Administrative
- User defined

4.2.1. Administrative Services

Each server exposes a single XATMI service that is built into the server executable and is exposed for the lifespan of the server. It is not intended to be accessed directly by the programmer although this is possible, instead however the results of the service are typically aggregated by a central Java domain-level XATMI administrative service.

4.2.2. C XATMI Per Server Service

The name of the service is not user defined although can be predicted:

`<SERVER_NAME>_ADMIN_<SERVER_ID>`

Note

User defined services are not allowed to be configured with a name with _ADMIN in them

4.2.3. Java XATMI Domain-wide Service

The name of the service again is not user defined although can be predicted from the domain name:

```
<DOMAIN_NAME>_ADMIN
```

4.2.4. User Defined Services

All user defined service names must be declared within the constraints defined in the XATMI specification plus several constraints imposed by BlackTie:

- Service names must be 128 characters or less
- Servers may not contain overlapping service names, e.g. server 1 and server 2 cannot define service A
- Service names may not use the reserved keyword _ADMIN_ in their name
- The service must be defined in the btconfig.xml
- The service may optionally be customized in the btconfig.xml
- The same function for the same service is expected at each server
- tpadvertise and tpunadvertise operate on the local service state, they are not used to (un)advertise the service at a domain wide level
- If a service is not thread safe, it is important to set the SIZE for the SERVICE in the btconfig.xml file to 1

4.3. Decoupling XATMI services and clients using a queuing pattern

4.3.1. Introduction

Starting from release blacktie-3_0_0 we have added two methods called btenqueue and btdequeue to simplify sending and receiving message. The M1 release supports transactional sends and the M2 release will support transactional receives.

This page will be updated soon to reflect the change.

XATMI (http://www.opengroup.org/pubs/catalog/c506.htm) is a standard which specifies how to connect clients to services. A client is a program that requests services to be performed. A service is a program that performs a specific application function on behalf of clients. In XATMI there are two types of service:

1. Request/response services receive a single request and produce at most a single response to the request. The request is the application data sent from the client to the service. The service processes the request and returns application data to the client by means of at most one response;
2. Conversational services are invoked by means of a connection request from the client. Once the connection is established and the service invoked, the client and the service can exchange data in an application-specific manner until the service returns, whereupon the connection is logically terminated.

However this model breaks down if there is no service currently running that can handle the client request. The latest release of BlackTie includes an enhancement (to XATMI) that facilitates the queuing of client requests such that the request can be processed at a later time when an appropriate service becomes available.

In XATMI message payloads are obtained using the following call:

```
char * tpalloc(char *type, char *subtype, long size)
```

To take advantage of the new BlackTie feature you will need to pass an extra parameter to the XATMI buffer allocation routine:

```
char* btalloc(msg_opts_t* ctrl, char* type, char* subtype, long size);
```

The resulting buffer can be used to make service requests with the following restrictions:

1. the client must use the asynchronous request/response model, i.e. it must use tpacall.
2. the client must not expect a response, i.e. it must set the TPNOREPLY flag.
3. it must not have any outstanding transaction.

[Currently we are using the extra parameter to control the priority of message delivery though we could, in the future, add other features to give the client more control over the mechanism].

And that's it, now when the requested service is brought on-line the queued requests will be made available to its' service routine. The latest BlackTie release (http://www.jboss.org/blacktie/

downloads.html) includes an example showing this feature in action. Unpack the release and look at the REAME in the examples/xatmi/queues directory.

There are two BlackTie specific config requirements to ensure that requests are queued even when the target server is unavailable;

- the SERVICE definition in btconfig.xml must set the "externally-managed-destination" attribute to true;
- you must deploy a JMS queue named BTR_[service name] to the application server (the XML file containing this queue definition must end with the suffix -service.xml)

Internally BlackTie uses JMS queues to connect services with clients. Normally when a server calls tpadvertise a queue is created and clients may then connect with the server via that queue. However, no such server is running in the scenario we are discussing and this is the reason the queue needs to be deployed before running the client.

So, for example, if the service name is TestOne then deploy a queue called BTR_TestOne to the JBoss deploy directory using an XML file containing the following:

```
<?xml version="1.0" encoding="UTF-8"?>

<server>
   <mbean code="org.jboss.jms.server.destination.QueueService"
      name="jboss.messaging.destination:service=Queue,name=BTR_TestOne"
      xmbean-dd="xmdesc/Queue-xmbean.xml">
      <depends optional-attribute-name="ServerPeer">jboss.messaging:service=ServerPeer</
depends>
      <depends>jboss.messaging:service=PostOffice</depends>
   </mbean>
</server>
```

4.4. How to use topics

Basically, topics are used to burst out messages to multiple XATMI services that do not necessarily expect a reponse.

If it helps, you might consider using topics for things like stock ticker updates where the receiver is not required to notify the sender that they have received the ticker.

To config a SERVIECE use topics, you could set type to 'topic', for example:

```
<SERVICES>
   <SERVICE name='foo' type='topic'/>
```

```
</SERVICES>
```

Note

clients do not need to configure the list of services in their btconfig.xml, they will need to try to connect to the queue first, then, the topic (unless they have the destination is configured in their btconfig.xml).

Note

tpacalls to topics MUST have TPNOREPLY set. If the client tpacalls a topic without TPNOREPLY we should fail the call. It also means tpcalls should fail too.

And there is a intention that externally-managed-destination is set to true as there are multiple receivers.

You can get more information about these in examples/xatmi/topics.

BlackTie Configuration

5.1. Environment variables

BLACKTIE_SCHEMA_DIR: the directory which contains all xsd schema files.

BLACKTIE_CONFIGURATION_DIR: the directory which contains all config files. If not assigned, current directory will be used.

5.2. Configuration Files

We need btconfig.xml for all servers and client, and individual <SERVICE>.xml for all services.

The latest schemas for all configuration can be found at: http://www.jboss.org/blacktie/docs/index.html

5.3. btconfig.xml

The btconfig.xml is the most significant runtime configuration file for BlackTie. The content of its structure is defined and validated against the btconfig.xsd file which ships with each version of BlackTie.

Important

ipv6 note

It is important to note that if you are using the sample btconfig.xml files and you are using ipv6 you will most likely need to change the word localhost to localhost4. Ways to check if this is neccesary are:

1. In your hosts file is the entry for localhost 127.0.0.1 or ::1
2. If you telnet to localhost does it start with "Trying ::1..."

The configuration includes:

- DOMAIN: A user defined name for the domain
- VERSION: The version of BlackTie software that is allowed to operate in this domain
- MACHINES: The list of machines and paths to executables
- SERVERS: The list of servers
  - SERVER: A server configuration element

- MACHINE-REF: A reference to the machne
- SERVICES: The list of services at a server
  - SERVICE: The key attributes of this element are: advertised - should the server automatically launch this service, function_name - the C function name (NOTE: Must be declspec(dllexport) exported on win32), java_class_name - the fully qualified class of the service for java Services
    - LIBRARY_NAME: The name of the .so or .dll library that exports the symbol, this is restricted by the mandatory "configuration" attribute which will restrict this XA_RESOURCE to the BLACKTIE_CONFIGURATION environment variable/-c parameter
    - SIZE: The number of dispatcher threads to handle requests
- XA_RESOURCES: The list of configured XA resources available at any client/server in this domain
  - XA_RESOURCE: A configuration of XA_RESOURCE, note that the "configuration" attribute will restrict this XA_RESOURCE to the BLACKTIE_CONFIGURATION environment variable/-c parameter
- ORB: The configuration to connect to the transaction service
- MQ: The configuration to connect to the message broker
- JMX: The url of the JMX agent
  - ENV_VARIABLES: Special key/value pairs can be specified here, known ENV_VARIABLEs are defined below, note that the "configuration" attribute will restrict this XA_RESOURCE to the BLACKTIE_CONFIGURATION environment variable/-c parameter
- BUFFERS: The buffer configuration.

5.4. SERVICES

More information on services is available *Chapter 4, Services* .

5.5. ENV_VARIABLES

The list of ENV_VARIABLES that are supported is:

1. LOG4CXXCONFIG is used for logging
2. RC_LOG_NAME The name of the recovery log file
3. JMXURL The URL to use to connect to the JMX kernel

4. QueueReaperInterval: The period of time in seconds to wait before assuming a service has disconnected and pruning its queue

5.6. BUFFERS

More information on buffer configuration is available *Chapter 3, Using Buffers with BlackTie* .

5.7. log4cxx.properties

This is a standard log4cxx configuration file, more information can be found here: http://logging.apache.org/log4cxx/index.html

Advanced run instructions for BlackTie

The wiki contains the latest instructions for running BlackTie, the relevant article can be found here: http://community.jboss.org/wiki/DeployingBlacktie

6.1. Running the TAO implementation repository with BlackTie

6.1.1. Introduction

The docs for the ImR are locted at

http://www.dre.vanderbilt.edu/Doxygen/Stable/tao/implrepo_service/usersguide.html

and

http://www.dre.vanderbilt.edu/~schmidt/DOC_ROOT/TAO/docs/implrepo/index.html

The build instructions on the WIKI includes a step on building the TAO Implementation repository (it is essentially cd $ACE_ROOT/ACEXML && make; cd $TAO_ROOT/orbsvcs/ImplRepo_Service; make)

Make sure the ImR executable is on your path:

```
export PATH=$PATH:$TAO_ROOT/orbsvcs/ImplRepo_Service
```

6.1.2. To start the ImR:

```
ImplRepo_Service -o locator.ior -d -x
```

locator.ior is the name of the file that the ImR uses to store server records.

Servers that create persistent IORs embed the location of the ImR into the IORs it creates. On each host where we need to run a server we need to start an 'activator' which in turn knows how to contact the ImR:

```
ImR_Activator ImplRepoService=file://locator.ior &
```

(Question: if the ImR_Activator and ImplRepo_Service are running on different hosts which startup option do we need to use to enable the activator to find the ImR?)

6.1.3. Adding a new server

The ImR can start servers on demand - if a client has an IOR that embeds the ImR location then the ImR knows how to start the required server. You can add servers using the tao_imr executable

by specifying the comand line to start server. (Note that there is also an option to start servers whenever the ImR is started).

For example, to start our examples:

```
tao_imr -ORBInitRef ImplRepoService=file://locator.ior add
ATMI_RM_202 -c "./server -c linux" -
w "/home/mmusgrov/blacktie/trunk/atmibroker-xatmi/src/example/txfooapp" -
e LD_LIBRARY_PATH="/home/mmusgrov/blacktie/releases/blacktie-1.0-MR6-SNAPSHOT/
lib:.:/usr/local/product/11.1.0/db_1/lib:/home/mmusgrov/blacktie/trunk/
atmibroker-tx/target/cxx/test/lib:/home/mmusgrov/blacktie/util/ACE+TAO-5.7.1/
ACE_wrappers/lib" -e BLACKTIE_SCHEMA_DIR="/home/mmusgrov/blacktie/releases/
blacktie-1.0-MR6-SNAPSHOT/schemas/xsd"
```

(where ATMI_RM_202 is a unique name for the server) or we could supply shell or batch files to simplify the command.

The unique name for the server must correspond to the name of a POA running in the server (this is a limitation of the TAO orb).

I use a single POA for recovery (created in `XAResourceManagerFactory::create_poa`) and use the environment variable BLACKTIE_SERVER_NAME for the name of this poa. We must use the same name when starting servers via the ImR.

The `BLACKTIE_SERVER_NAME` environment variable is defined by concatenating domain, server and serverid (see `AtmiBrokerServer::serverinit`).

To manually start this server use tao_imr command providing it with the name of the server we just added:

```
tao_imr -ORBInitRef ImplRepoService=file://locator.ior start ATMI_RM_202
```

The server needs to know that is should use an ImR. To do this we need to update the btconfig.xml file and add:

`-ORBUseImR 1` to the ORBOPT environment variable. For example:

```
<ENV_VARIABLE>
  <NAME>ORBOPT</NAME>
  <VALUE>-ORBUseImR 1 -ORBInitRef
    NameService=corbaloc::localhost:3528/NameService</VALUE>
</ENV_VARIABLE>
```

6.1.4. Is the ImR a single point of failure?

We can run multiple ImRs. I/We still need to investigate how this works. Here's a quote from the TAO docs:

"One or more Implementation Repositories will be stored in additional profiles in the IOR. Other Implementation Repositories can also be located by multicasting (on a default multicast group) the server name of the Persistent Object the client is interested in. The default multicast group and default port of the Implementation Repository can be overridden through command line options or environment variables.

In most cases, one Implementation Repository will be enough. For redundancy, several Implementation Repositories can be specified."

6.2. Running BlackTie with a different host IP address

If you want to run blacktie server and client on a different host, such as run blacktie server on linux and run blacktie client on windows.

1. update Name Service running in JBoss AS

   edit JBOSS_HOME/server/all/conf/jacorb.properties chanage ORBInitRef.NameService=corbaloc::localhost:3528/JBoss/Naming/root to ORBInitRef.NameService=corbaloc::YOUR_HOST_IPADDR:3528/JBoss/Naming/root

   uncomment #OAIAddr, and add OAIAddr=YOUR_HOST_IPADDR

   Note: I tried to use 0.0.0.0, but it does not work. JBoss runs OK, but both blacktie server and client are running errors with CORBA::TRANSIENT.

2. update btconfig.xml to use corbaloc::YOUR_HOST_IPADDR:3528/NameService

3. Make sure firewall doesn't block the packages

4. run.sh | bat -c all on linux or windows as usual.

5. run server on linux and run client on windows

BlackTie Administration

The *Section 7.1, “BlackTie Administration Functions”* are exposed through a both a JMX MBean and a Java XATMI service deployed into JBoss AS, configured the same btconfig.xml of the domain it is administering

- *Section 7.2, “BlackTieAdminService XATMI Service”*
- *Section 7.3, “BlacktieAdminService JMX Bean”*

Each BlackTie server exposes the same administration service, registered during AtmiBrokerServer startup

1. *Section 7.4, “AtmiBrokerAdmin XATMI Service”*

Queue administration for the AtmiBrokerAdmin service is performed through a separate Java XATMI service

1. stompconnectservice: BlacktieStompAdministrationService

Administration is facilitated graphically from a JBoss AS RHQ administration console plugin

1. *Section 7.5, “Monitoring and management of blacktie servers by blacktie-rhq-plugin ”*

Figure 7.1. Administration

7.1. BlackTie Administration Functions

7.1.1. The following operations all perform functionality at the domain level

- getSoftwareVersion:

  Retrieve the software version of the domain

- getServerList:

  Retrieve the full list of servers

- listRunningServers:

  Discover running servers

  To achieve this, the administration routings lists all the queues locally for the appearance of the string ADMIN

  Note servers connected to other JBoss AS instances will not be discoverable

- reloadDomain:

  Halt servers, update configuration, restart

  This causes reloadServer to be called on all running instances

- pauseDomain:

  This calls pauseServer for each server

  servers which connect to the domain which it is paused are paused automatically

- resumeDomain:

  This calls resumeDomain for each server

  Paused servers which reconnect after the domain is resumed will resume automatically

- getServersStatus:

  This method looks in the btconfig.xml for the list of servers and returns an HTML structure to describe the status of the servers in the domain, if there are no servers running, than instances would not be returned

```
<servers>
   <server>
      <name>[SERVER_NAME]</name>
      <instances>
         <instance>
            <id>[ID]</id>
            <status>[STATUS]</status>
         </instance>
      </instances>
   </server>
</servers>
```

- [SERVER_NAME] is a arbitrary length character string
- [ID] is a server id
- [STATUS] is equal to 0 (Idle) or 1 (Running)

7.1.2. The following operations perform functionality at the server level

- getServiceCounter (service name):

Retrieves the counter for a service from all servers

- listRunningInstanceIds (server name):

  Get the list of ids of currently running servers

- serverdone:

  Calls server_sigint_handler_callback

  Returns 1

- reloadServer:

  requires an external shell script to repeatedly start server processes dependent upon exit status of server

  If the server exits with the reloadServer code, then an svn update on the configuration is issued and the server restarts

- advertise (service name):

  Does not allow advertise of <SERVER>_ADMIN_<ID>

  Calls advertiseByAdmin (don't need the function pointer)

  Returns 1

- unadvertise (service name):

  Suppresses unadvertise of <SERVER>_ADMIN_<ID>

  Calls tpunadvertise

  Returns 1

- pauseServer:

  Stop all currently running services

  Record which services were paused

  A call to tpadvertise will now place a service in the paused status

  A call to tpunadvertise will allow unadvertisement of the service

- resumeServer:

  Start the list of paused services

- counter (serviceName):

  Calls getServiceMessageCounter

Returns a 16 length buffer

First byte is the number 1

Remaining 15 bytes contains a long

- status [optional service name]:

  Calls getServiceStatus

  Returns an HTML structure ServiceStatus

  First byte of the response is the number 1

  Remaining bytes contains the ServiceStatus

```
<server>
   <name>[SERVER_NAME]</name>
   <services>
      <service>
         <name>[SERVICE_NAME]</name>
         <status>[STATUS]</status>
      </service>
   </services>
</server>
```

- [SERVICE_NAME] is a 15 character length name
- [STATUS] is equal to 0 (unadvertised) or 1 (advertised)

7.1.3. Viewing Transaction Statistics

In a typical BlackTie installation the transaction service will be running in the JBoss Application server where you installed the stomp connect service. In such a setup basic transaction management information is exposed by the native transaction service running in the AS. For example to enable statistics open the JBossTS property file located at:

- $JBOSS_HOME/server/<server>/conf/jbossts-properties.xml

and set the enableStatistics property to YES:

```
<property name="com.arjuna.ats.arjuna.coordinator.enableStatistics" value="YES"/>
```

The statistics are then available via any standard JMX browser or programatically via JMX. The object name of the MBean that maintains the stats is:

```
jboss.management.local:J2EEServer=Local,j2eeType=JTAResource,name=TransactionManager
```

If you are using the web based JBossAS jmx console then look for the name

```
J2EEServer=Local,j2eeType=JTAResource,name=TransactionManager
```

in the section called jboss.management.local

In the jmx console some extra transaction information is also exposed as a service with the following name:

```
service=TransactionManager
```

The statistics are global to the application server. We would like to provide similar information on a per BlackTie domain and per BlackTie server basis in a future release.

7.2. BlackTieAdminService XATMI Service

Defined in stompconnectservice:BlacktieAdminServiceXATMI

Contains the same functionality as the JMX bean

The XATMI service is accessed by:

- User defined XATMI administration clients

7.3. BlacktieAdminService JMX Bean

Defined in the module: stompconnectservice:BlacktieAdminServiceMBean

The JMX bean is accessed by either:

- blacktie-rhq-plugin
- Custom JMX administration programs

7.4. AtmiBrokerAdmin XATMI Service

The BlackTie administration services marshal calls to this raw administration service deployed in each server

The service is bound with a well known name:

- <SERVER_NAME>_ADMIN_>SERVER_ID>

Each server exposes the administration service for the duration of its lifespan

Clients (typically the BlackTie administration services) are able to invoke a standard XATMI service with a structured command

Command structure is (note the trailing comma's):

```
<command>,[<arg1>,<arg2>,....]
```

7.5. Monitoring and management of blacktie servers by blacktie-rhq-plugin

7.5.1. Install blacktie-rhq-plugin

1. copy <BLACKTIE_BIN_DIR>/blacktie-admin-services/blacktie-rhq-plugin-<VERSION>.jar to <JBOSS_HOME>/server/all/deploy/admin-console.war/plugins
2. restart jboss as
3. open http://localhost:8080/admin-console, now you can find 'Blacktie Domains'

7.5.2. How to manage a blacktie server

1. start a server in <BLACKTIE_BIN_DIR>/example/xatmi/foo following with README
2. navigate to "Blacktie Domains" -> "fooapp" -> "Servers" -> "default", you can click "Control"
3. there is a operation named 'shutdown', and you can click the button. You should type in server id which used to start example server, e.g if you start example server with command 'server -c linux -i 1' , and you should type in '1'
4. other operations such like 'listServiceStatus', 'getServiceCounter', 'advertise' and 'unadvertise'. you could type in a SERVICE NAME, e.g if you want to unadvertise service BAR in server 1, you should click operation 'unadvertise' and type '1' in id also type 'BAR' in service.

7.5.3. How to monitor a blacktie server

1. If there is no server instance running, a red cross will be in front of the server name
2. navigate to 'Metric' tab, you should see the number of running instances of server

7.6. BlackTie Command Line Administration

7.6.1. Introduction

The following commands are supported by BlackTie:

Table 7.1. Commands

Command	Description
Section 7.6.2, “btadmin”	The btadmin command is used to administer the BlackTie serve process from the command line
Section 7.6.3, “generate_server”	The generate_server command is used to build a BlackTie server
Section 7.6.4, “generate_client”	The generate_client command is used to build a client application linked against the BlackTie libraries

7.6.2. btadmin

7.6.2.1. Overview

The btadmin command is used to perform command line administration of the BlackTie server processes. In a well configured environment, the btadmin tool can be launched in either interactive mode (by ommitting a command) or in non-interactive mode by issues a command, for example: "btadmin startup". For the list of commands type: "help"

7.6.2.2. Expected Environment

You must have called

```
<BLACKTIE_HOME>/setenv.[sh|bat]
```

You must, either set `BLACKTIE_CONFIGURATION_DIR` to the directory containing a btconfig.xml for the domain that you wish to administer or ensure a valid btconfig.xml is in the working directory

7.6.2.3. Usage

```
btadmin [<command>]
```

7.6.2.4. Commands

Table 7.2. btadmin commands

Command	Description
version	Print out the version of BlackTie that is linked against

Command	Description
startup [<serverName>]	Start the server, optionally constrained by the serverName
pauseDomain	Pause the domain
advertise <serverName> <serviceName>	Advertise the service indicated by the name at the server
help [command]	Get help on a command
getServersStatus	The status of the running servers
unadvertise <serverName> <serviceName>	Unadvertise the service at the specified server
resumeDomain	Resume the domain
listRunningServers	List all running servers in the domain
quit	Shutdown the btadmin tool, a no-op for a non-interactive btadmin shell
listRunningInstanceIds <serverName>	List the running instance ids of a specific server
listServiceStatus <serverName> <serviceName>	List the status of the service running on a particular server group
shutdown [<serverName> [<serverId>]]	Shutdown items in the domain, optionally constrained by a server name and (again, optionally) a serverId

7.6.2.5. Credentials

The administration connection is assumed to be made within a secure network

7.6.3. generate_server

7.6.3.1. Overview

The generate_server command is used to create a standalone BlackTie server application linking the user code against the BlackTie runtime libraries.

7.6.3.2. Expected Environment

<BLACKTIE_HOME>/setenv.[sh|bat] must be executed prior to execution

7.6.3.3. Usage

```
generate_server <-Dservice.names=service[,service2,...]> <-Dserver.includes="source1[ source2 ..."]> <-Dx.inc.dir="directory"> <-Dx.inc.dir2="directory2"> <-Dx.lib.dir="directory" -Dx.libs="library[ library2]"> <-Dx.lib.dir2="directory2"> <-Dx.libs2="library[ library2]"> <-Dx.define="symbol1[,symbol2]">
```

7.6.3.4. Parameters

Table 7.3. generate_server parameter

Parameter	Description
-Dservice.names=service[,service2,...]	This is the comma seperated list of services that should be advertised at boottime that were not coded in a DLL export compatible manner and as such cannot be registered for advertised in the btconfig.xml
-Dserver.includes="source1[source2 ..."]	The list of files that should be compiled into this server
-Dx.inc.dir="directory"	A single additional directory for including
-Dx.lib.dir="directory"	A single additional directory for linking against
-Dx.libs="library[library2]"	A space seperated list of libraries to link against in the lib dir
-Dx.inc.dir2="directory"	A second additional directory to include headers from
-Dx.lib.dir2="directory"	A second additional directory to include libraries from
-Dx.libs2="library[library2]"	The list of libraries to include from the second library directory
-Dx.define="symbol1[,symbol2]"	The comma-seperated list of addtional pre-processor symbols to define

7.6.3.5. Example

To link against the Oracle database

On Linux:

```
generate_server -Dservice.names=BAR -Dserver.includes="request.c ora.c DbService.c"
  -Dx.inc.dir="$ORACLE_HOME/rdbms/public"
  -Dx.lib.dir="$ORACLE_HOME/lib" -Dx.libs="occi clntsh"
  -Dx.define="ORACLE"
```

On Windows:

```
generate_server -Dservice.names=BAR -Dserver.includes="request.c ora.c DbService.c"
  -Dx.inc.dir="%ORACLE_HOME%\OCI\include"
  -Dx.lib.dir="%ORACLE_HOME%\OCI\lib\MSVC" -Dx.libs="oci"
```

```
-Dx.define="ORACLE"
```

7.6.4. generate_client

7.6.4.1. Overview

The generate_client command is used to create a standalone client application linking the user code against the BlackTie runtime libraries.

7.6.4.2. Expected Environment

<BLACKTIE_HOME>/setenv.[sh|bat] must be executed prior to execution

7.6.4.3. Usage

```
generate_client <-Dclient.includes="source1[ source2 ..."]> <-Dx.inc.dir="directory" > <-Dx.inc.dir2="directory2"> <-Dx.lib.dir="directory" -Dx.libs="library[ library2]"> <-Dx.lib.dir2="directory2"> <-Dx.libs2="library[ library2]"> <-Dx.define="symbol1[,symbol2]">
```

7.6.4.4. Parameters

Table 7.4. generate_client parameter

Parameter	Description
-Dclient.includes="source1[source2 ..."]	The list of files that should be compiled into this client
-Dx.inc.dir="directory"	A single additional directory for including
-Dx.lib.dir="directory"	A single additional directory for linking against
-Dx.libs="library[library2]"	A space seperated list of libraries to link against in the lib dir
-Dx.inc.dir2="directory"	A second additional directory to include headers from
-Dx.lib.dir2="directory"	A second additional directory to include libraries from
-Dx.libs2="library[library2]"	The list of libraries to include from the second library directory
-Dx.define="symbol1[,symbol2]"	The comma-seperated list of addtional pre-processor symbols to define

7.6.4.5. Example

To create a client which links against the Oracle database.

On Linux:

```
generate_client -Dclient.includes="client.c request.c ora.c cutil.c"
  -Dx.inc.dir="$ORACLE_HOME/rdbms/public"
  -Dx.lib.dir="$ORACLE_HOME/lib" -Dx.libs="occi clntsh"
  -Dx.define="ORACLE"
```

On Windows:

```
generate_client -Dclient.includes="client.c request.c ora.c cutil.c"
  -Dx.inc.dir="%ORACLE_HOME%\OCI\include"
  -Dx.lib.dir="%ORACLE_HOME%\OCI\lib\MSVC" -Dx.libs="oci"
  -Dx.define="ORACLE"
```

Appendix A. Revision History

Revision History

Revision 1 Fri May 13 2011 TomJenkinson<tom.jenkinson@redhat.com>

Initial conversion of wiki into Docbook