JBoss Remoting

What is JBoss Remoting

JBoss Remoting is a framework with a single, simple API for making network based invocations and other network related services.

Features of JBoss Remoting:

- **Server identification** – a simple String identifier which allows for remoting servers to be identified and called upon

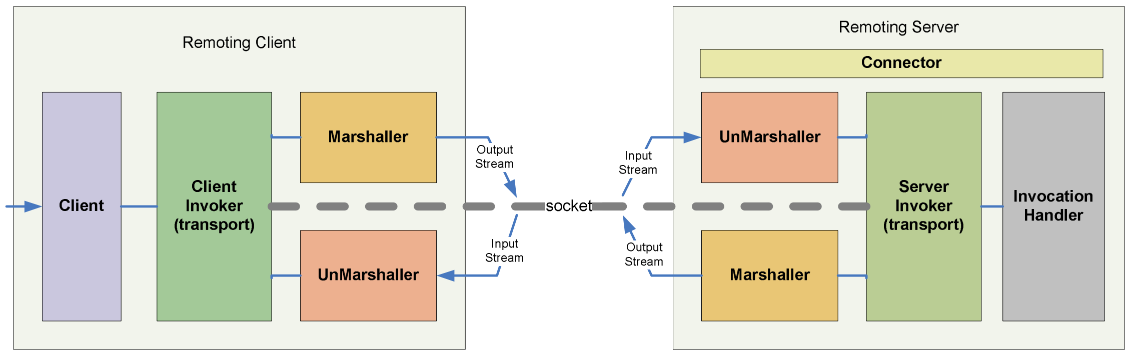
  For example: `socket://192.168.0.42:5400`
- **Pluggable transports** – can use different protocol transports, such as socket, rmi, http, etc., via the same remoting API.
- **Pluggable data marshallers** – can use different data marshallers and unmarshallers to convert the invocation payloads into desired data format for wire transfer.
- **Automatic discovery** – can detect remoting servers as they come on and off line.

What is JBoss Remoting

Features of JBoss Remoting (cont.):

– **Server grouping** – ability to group servers by logical domains, so only communicate with servers within specified domains.
– **Callbacks** – can receive server callbacks via push and pull models.
– **Asynchronous calls** – can make asynchronous, or one way, calls to server.
– **Local invocation** – if making an invocation on a remoting server that is within the same process space, remoting will automatically make this call by reference, to improve performance.
– **Dynamic classloading** – will load remote classes from server if not present within client VM. Is used for loading custom marshaller/unmarshaller.

Core Architecture



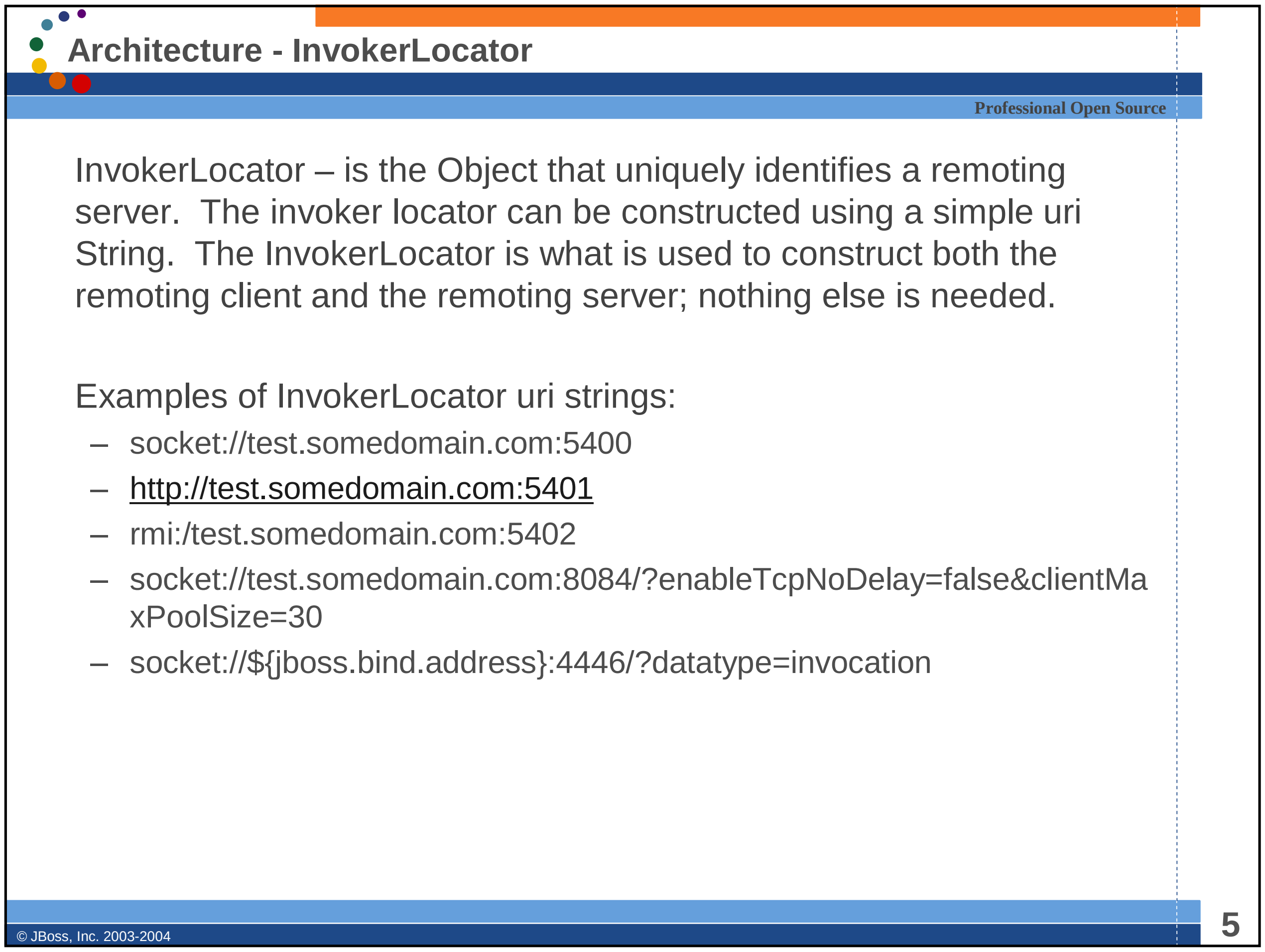
Client – the external API access point for client code.

Client/Server Invoker – protocol specific implementation. For example, SocketClientInvoker and SocketServerInvoker.

Marshaller/UnMarshaller – receives the streams from the invoker converting wire data to Object form.

Invocation Handler – end target interface implemented by user that receives the invocation from the client.

Connector – the service that binds the invoker, marshaller/unmarshaller, and the invocation handler on the server side.



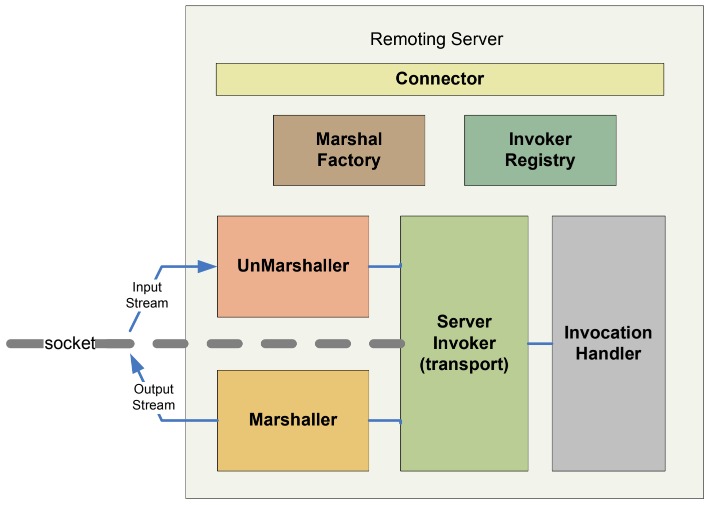
Architecture - InvokerLocator

InvokerLocator – is the Object that uniquely identifies a remoting server. The invoker locator can be constructed using a simple uri String. The InvokerLocator is what is used to construct both the remoting client and the remoting server; nothing else is needed.

Examples of InvokerLocator uri strings:

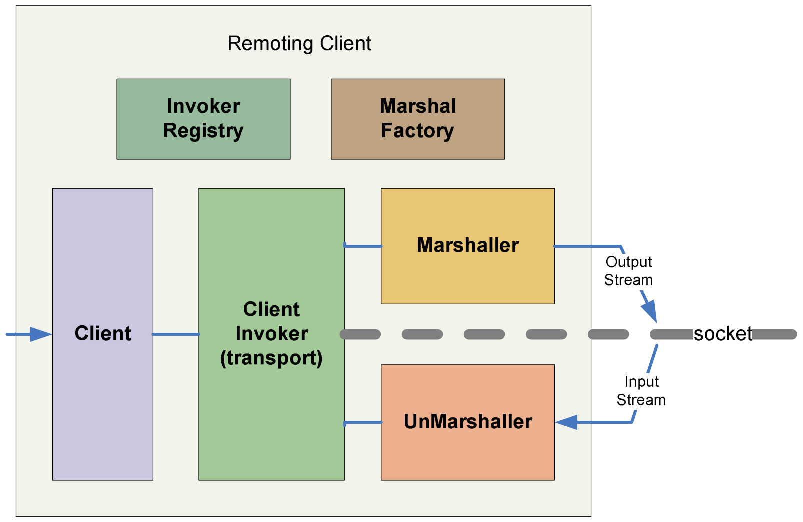
– socket://test.somedomain.com:5400
– http://test.somedomain.com:5401
– rmi:/test.somedomain.com:5402
– socket://test.somedomain.com:8084/?enableTcpNoDelay=false&clientMaxPoolSize=30
– socket://${jboss.bind.address}:4446/?datatype=invocation

Architecture – server side



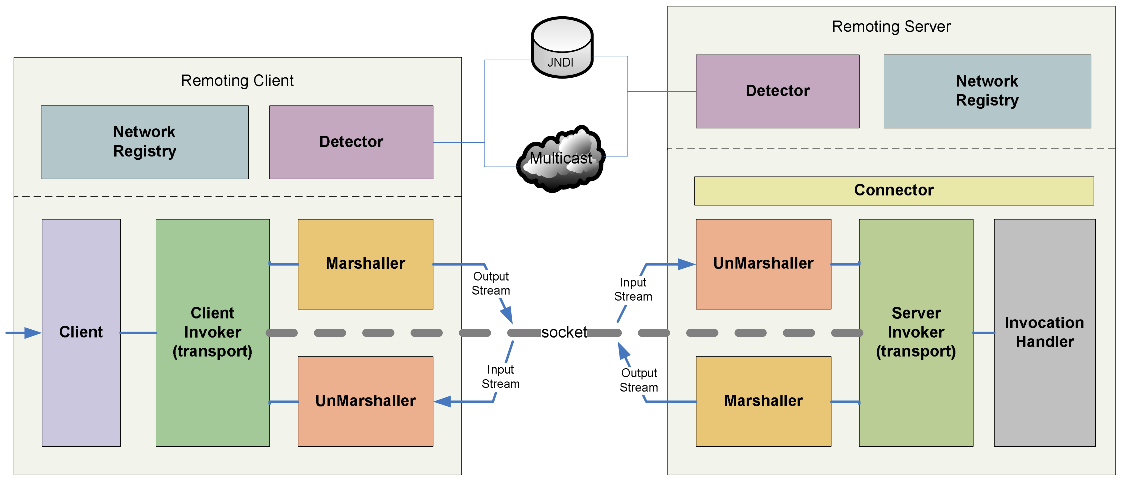
- The Connector is given the InvokerLocator (either via xml config or programmatically) and then uses the InvokerRegistry to create the ServerInvoker. The Connector also sets the handler on the invoker.
- The server invoker, will then use the marshal factory to create the appropriate Marshaller and UnMarshaller for the invoker.

Architecture – client side



- The Client is given the InvokerLocator and then in turn uses the InvokerRegistry to create the client invoker.
- The client invoker, will then use the marshal factory to create the appropriate Marshaller and UnMarshaller for the invoker.

Architecture - detection

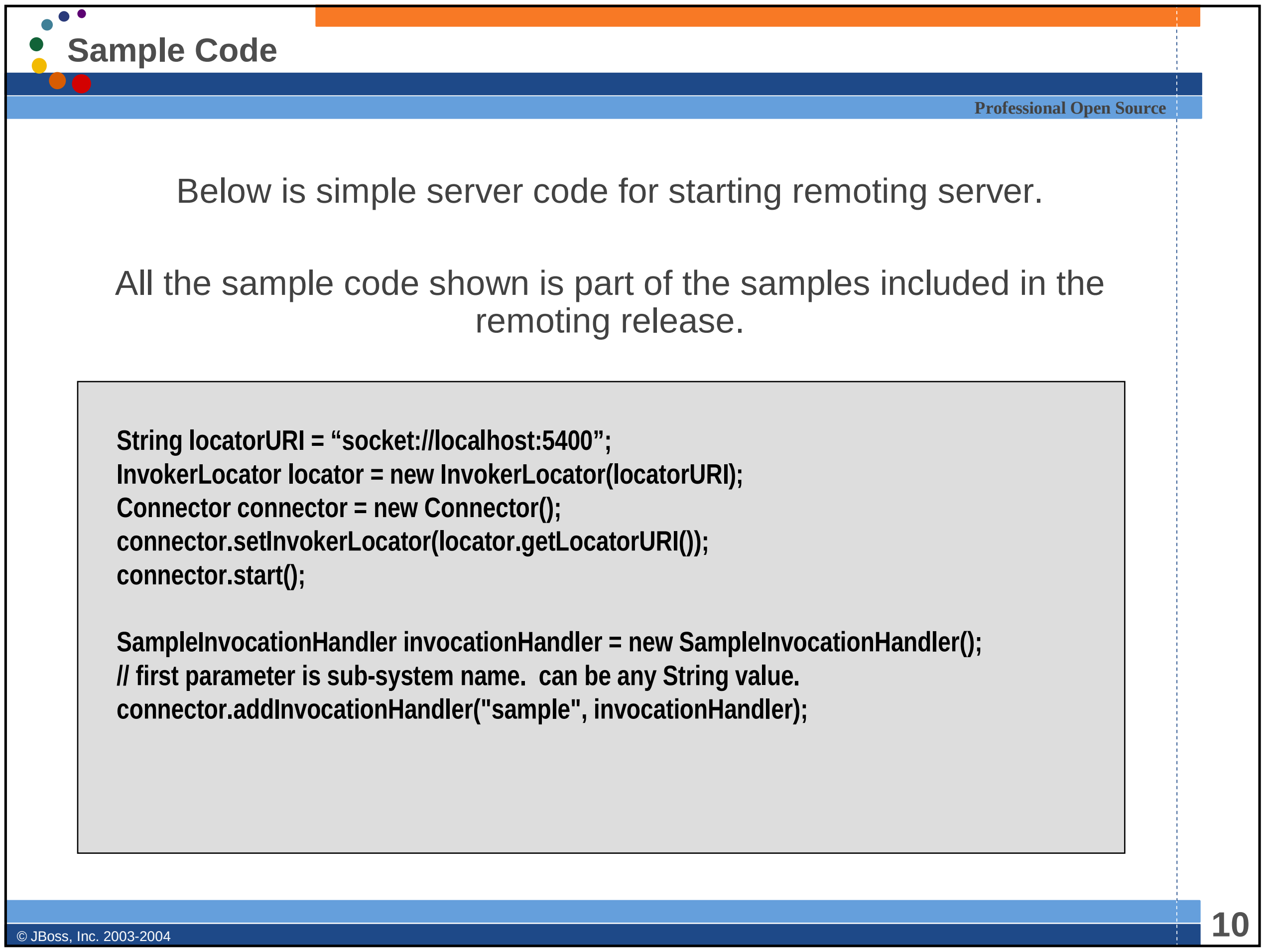


- Network registry and detectors run as a separate service.
- Detectors will populate the network registry as remoting servers discovered.
- Connector will register itself with local network registry upon startup.

Configuration

Configuration for remoting can be handled either programmatically or via JBoss service xml (when deployed within JBoss AS container).

The full documentation for configuration can be found within the remoting User Guide or on the Wiki (see links on http://www.jboss.org/products/remoting).



Sample Code

Below is simple server code for starting remoting server.

All the sample code shown is part of the samples included in the remoting release.

```
String locatorURI = “socket://localhost:5400”;
InvokerLocator locator = new InvokerLocator(locatorURI);
Connector connector = new Connector();
connector.setInvokerLocator(locator.getLocatorURI());
connector.start();

SampleInvocationHandler invocationHandler = new SampleInvocationHandler();
// first parameter is sub-system name.  can be any String value.
connector.addInvocationHandler("sample", invocationHandler);
```

Sample Code

Implementation for the ServerInvocationHandler.

```
public static class SampleInvocationHandler implements ServerInvocationHandler
{
    public Object invoke(InvocationRequest invocation) throws Throwable
    {
       // Print out the invocation request
       System.out.println("Invocation request is: " + invocation.getParameter());
       // Just going to return static string as this is just simple example code.
       return “This is the response”;
     }

     public void addListener(InvokerCallbackHandler callbackHandler)  { … }
     public void removeListener(InvokerCallbackHandler callbackHandler) { … }
     public void setMBeanServer(MBeanServer server) { … }
     public void setInvoker(ServerInvoker invoker)  { … }

}
```

Sample Code

Remoting client code.

```
String locatorURI = “socket://localhost:5400”;
InvokerLocator locator = new InvokerLocator(locatorURI);
System.out.println("Calling remoting server with locator uri of: " +
                locatorURI);

Client remotingClient = new Client(locator);
Object response = remotingClient.invoke("Do something");

System.out.println("Invocation response: " + response);
```

JBoss Remoting information

Project page - http://www.jboss.org/products/remoting

- User Guide
- Wiki
- Demo
- Forum

Introducing JBoss Remoting by John Mazzitelli published at OnJava.com