Some topics we need to cover:

- Server proxies
- Server join & leases with lookup service
- Restarting services & Jini state information
- Managing leases with clients

Server & Proxies

Note the presentation of Server Proxies follows very closely to the Simple example in Jan Newmarch’s Jini Tutorial at: http://pandonia.canberra.edu.au/java/jini/tutorial/Jini.xml. The server provides a different service in this example. A few other details relating to RMI usage and registration are also changed. It is instructive to look at both examples.

In designing a Jini service there are several different strategies:

- Client-side server, client downloads the server
  
  When the client contacts the lookup service the code that provides the actual service is downloaded to the client. Think of this as an applet version without a browser. While this may not be a common solution, it might make sense in some situations.
Client downloads a proxy to the server

When the client contacts the lookup a proxy to the server is downloaded to the client. We will look at two different ways of doing this: using an RMI proxy or building our own proxy. We have seen using an RMI proxy in other examples. Building our own proxy will be covered in a later document.

### Client-side Server

Will use an encryption server as an example

The classes:

- **EncryptionInterface**
  
  Interface that DirectEncryptionServer implements and the client uses to interact with the server

- **DirectEncryptionServer**
  
  Does the encryption using xor. Not good encryption, but the interface allows for adding more methods of encrypting.

- **DirectEncryptionClient**
  
  Uses the server

- **RegisterEncryptionService**
  
  Registers the server using unicast join

### The Example Classes

#### EncryptionInterface

```java
package whitney.jini.examples.serverProxy;
public interface EncryptionInterface {
    public String[] encodeTypes();
    public String encode(String encodeType, String plainText);
    public String decode(String encodeType, String encodedText);
}
```

#### DirectEncryptionServer

```java
package whitney.jini.examples.serverProxy;
import java.io.Serializable;
import sdsu.io.XorOutputStream;
public class DirectEncryptionServer implements EncryptionInterface, Serializable {
    private static final String XOR = "xor";
    private byte mask;
    private String[] encodeTypes = { XOR };
    public DirectEncryptionServer(byte aMask) {
        mask = aMask;
    }
    public String[] encodeTypes() {
        return encodeTypes;
    }
    public String encode(String encodeType, String plainText) {
        if (encodeType.equals(XOR))
            return xorText(plainText);
        else // just an example. Save space by not using exception
            return "";
    }
    public String decode(String encodeType, String encodedText) {
        if (encodeType.equals(XOR))
            return xorText(encodedText);
        else // just an example. Save space by not using exception
            return "";
    }
    public String xorText(String text) {
        return text;
    }
    public String textXor(String text) {
        return xorText(text);
    }
}
```

private String xorText(String text) {
    byte inputBytes[] = text.getBytes();
    byte[] xorBytes = new byte[inputBytes.length];
    for (int k = 0; k < inputBytes.length; k++)
        xorBytes[k] = (byte) (inputBytes[k] ^ mask);
    return new String(xorBytes);
}

DirectEncryptionClient

package whitney.jini.examples.serverProxy;
import net.jini.core.entry.Entry;
import net.jini.core.lookup.ServiceTemplate;
import net.jini.core.lookup.ServiceRegistrar;
import net.jini.core.discovery.LookupLocator;
import java.rmi.RMISecurityManager;
public class DirectEncryptionClient{
    public static void main (String[] args) throws Exception {
        System.setSecurityManager (new RMISecurityManager ());
        LookupLocator lookup = new LookupLocator("jini://eli.sdsu.edu");
        ServiceRegistrar registrar = lookup.getRegistrar ();
        Entry[] serverAttributes = new Entry[1];
        serverAttributes[0] = new Name("EncryptionService");
        ServiceTemplate template =
            new ServiceTemplate(null, null, serverAttributes);
        EncryptionInterface encoder =
            (EncryptionInterface) registrar.lookup (template);
        String testText = "Hi Mom";
        String encodedText = encoder.encode( "xor", testText );
        String decodedText = encoder.encode( "xor", encodedText );
        System.out.println ("Original Text: "+ testText );
        System.out.println ("Encoded Text: "+ encodedText );
        System.out.println ("Decoded Text: "+ decodedText );
    }
}

RegisterEncryptionService

package whitney.jini.examples.serverProxy;
import java.rmi.RMISecurityManager;
import java.util.Date;
import net.jini.core.discovery.LookupLocator;
import net.jini.core.entry.Entry;
import net.jini.core.lease.Lease;
import net.jini.core.lookup.ServiceID;
import net.jini.core.lookup.ServiceItem;
import net.jini.core.lookup.ServiceRegistrar;
import net.jini.core.lookup.ServiceRegistration;
import net.jini.core.lease.LeaseRenewalManager;
public class RegisterEncryptionService {
    public static void main (String[] args) throws Exception {
        System.setSecurityManager (new RMISecurityManager ());
        Entry[] serverAttributes = new Entry[1];
        serverAttributes[0] = new Name("EncryptionService");
        ServiceID serverID = null;
        DirectEncryptionServer theServer =
            new DirectEncryptionServer((byte) 11);
        ServiceItem encryptServer =
            new ServiceItem(serverID, theServer, serverAttributes);
        LookupLocator lookup = new LookupLocator("jini://eli.sdsu.edu");
        ServiceRegistrar registrar = lookup.getRegistrar ();
        ServiceRegistration serverReg =
            registrar.register(encryptServer, 1000 * 60 );
        System.out.println ("Server registered");
    }
}
RegisterEncryptionService Continued

```java
Lease serverLease = serverReg.getLease();
LeaseRenewalManager manageLease =
    new LeaseRenewalManager( serverLease, Lease.FOREVER, null );
serverID = serverReg.getServiceID();
long now = (new Date()).getTime();
long leaseDuration = (serverLease.getExpiration() - now)/1000;
System.out.println ( "Server id: " + serverID );
System.out.println ( "Server lease: " + serverLease );
System.out.println ( "Lease duration in seconds " + leaseDuration );
Thread.sleep( 1000 * 60 * 15 );
System.out.println ( "Server going down: ");
manageLease.cancel(serverLease);
}
```

RegisterEncryptionService Explained

Set the security manager to allow downloading of reggie code

```java
System.setSecurityManager (new RMISecurityManager ());
```

Prepare to register server with lookup service using unicast join

```java
// Entry is used described the attributes of the service
Entry[] serverAttributes = new Entry[1];
serverAttributes[0] = new Name ("EncryptionService");
ServiceID serverID = null;
// Create the server
DirectEncryptionServer theServer =
    new DirectEncryptionServer((byte) 11);
// Service items contain the data needed to register the service
ServiceItem encryptServer =
    new ServiceItem( serverID, theServer, serverAttributes);
```

RegisterEncryptionService Explained Continued

Perform the unicast join

```java
// Find the lookup service
LookupLocator lookup = new LookupLocator ("jini://eli.sdsu.edu");
ServiceRegistrar registrar = lookup.getRegistrar ();

// Register the server, the second arugment of register is the requested lease time
ServiceRegistration serverReg =
    registrar.register(encryptServer, 1000 * 60 );
```

ServiceRegistration contains information about the registration.

Handle the lease for the server

The LeaseRenewalManager handles renewing the lease for the server with the lookup service. The LeaseRenewalManager creates a thread that handles renewal requests. The thread sends renewal requests before the lease expires. The null in the constructor takes place of a LeaseListener, which would be called when the lease can not be renewed. The Lease.FOREVER really means give the longest lease period. With a lightly loaded lookup service the longest lease is about 5 minutes.

```java
Lease serverLease = serverReg.getLease();
LeaseRenewalManager manageLease =
    new LeaseRenewalManager( serverLease, Lease.FOREVER, null );
```

Print out some information about the lease & server registration

```java
serverID = serverReg.getServiceID();
long now = (new Date()).getTime();
```
long leaseDuration = (serverLease.getExpiration() - now)/1000;
System.out.println ( "Server id: " + serverID);
System.out.println ( "Server lease: " + serverLease);
System.out.println ( "Lease duration in seconds " + leaseDuration);

RegisterEncryptionService Explained Continued

Keep the server object & LeaseRenewalManager alive

Since the server object has been uploaded to the lookup service, it is not really needed. If the LeaseRenewalManager is not kept alive, the server will be purged from the lookup service when the lease expires. This is just a test, so only keep the server alive for 15 minutes.

Thread.sleep( 1000 * 60 * 15 );

Be polite and cancel the lease.

We don’t need to cancel the lease as the lookup service will cancel it when it ends and the lookup service has the server object. Clients do not contact this process. If we did not cancel the lease, a client could still download the server from the lookup service.

System.out.println ( "Server going down: ");
manageLease.cancel(serverLease);

DirectEncryptionClient Explained

Contact the lookup service

LookupLocator lookup = new LookupLocator ("jini://eli.sdsu.edu");
ServiceRegistrar registrar = lookup.getRegistrar ();

Request a DirectEncryptionServer object

The lookup service downloads a serialized DirectEncryptionServer object. The client needs access to the DirectEncryptionServer class to deserialize the object. The class can be downloaded dynamically at runtime via http. See the section on running the example. The client runs the DirectEncryptionServer object locally.

Entry[] serverAttributes = new Entry[1];
serverAttributes[0] = new Name ("EncryptionService");
ServiceTemplate template =
new ServiceTemplate (null, null, serverAttributes);
EncryptionInterface encoder =
(EncryptionInterface) registrar.lookup (template);

Interact with the DirectEncryptionServer object

String testText = "Hi Mom";
String encodedText = encoder.encode ("xor", testText);
String decodedText = encoder.encode ("xor", encodedText);

Running the Example

Class Location

Client Machine needs:

DirectEncryptionClient
EncryptionInterface

Server Machine needs:

EncryptionInterface
DirectEncryptionServer
RegisterEncryptionService

Lookup Service machine needs:

Does not need access to any of the example classes

HTTP server:

DirectEncryptionServer

This class must be accessible by a web server. Since the class in the package whitney.jini.examples.serverProxy the class must be a directory structure whitney/jini/examples/serverProxy/DirectEncryptionServer.class. In this example I used the web server that comes with Jini. The following command was used to start the web server on the machine eli.sdsu.edu.

```
java -jar /opt/jini1_0/lib/tools.jar -port 8880 -dir ~whitney/jini_data /jini_http_root
```

I placed the directory whitney/jini/examples/serverProxy in the root directory for the web server (export/home/whitney/jini_data /jini_http_root).

---

**Starting the programs**

**Jini components**

Start rmid

I used the command on a UNIX machine:

```
rmid &
```

Start reggie

I used the command on a UNIX machine:

```
java -jar /opt/jini1_0/lib/reggie.jar http://eli.sdsu.edu:8880/reggie-dl.jar ~whitney/jini_data/policy
~whitney/jini_data/jini_logs/reggie whitney.eli.sdsu.edu &
```

Start the web server

I used the command on a UNIX machine:

```
java -jar /opt/jini1_0/lib/tools.jar -port 8880 -dir ~whitney/jini_data /jini_http_root
```

**The Example Code**

The codebase property must be set properly to insure that the DirectEncryptionServer class is downloaded to the client at runtime. The property can be set either on the client side or the server side. For more details see Doc 16 Dynamically Downloading Classes at: http://www.eli.sdsu.edu/courses/spring99/cs696/notes/ddc/ddc.html. In this example I will set the property on the server side.

```
java -Djava.rmi.server.codebase=http://eli.sdsu.edu:8880 whitney.jini.examples.serverProxy.RegisterEncryptionService
```

Start the client

```
java whitney.jini.examples.serverProxy.DirectEncryptionClient
```

---

Copyright ©, All rights reserved.
1999 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA.

OpenContent license defines the copyright on this document.

---