

Using the NetworkModule

This document acts as a brief guide to using the JXTA NetworkModule. A simple client/server example that uses the network module is also downloadable.

The network module aims to remove the need for the application developer to spend time in writing code to connect to the JXTA network and to send a receive messages.

It should be noted that currently the JXTA NetworkModule is designed for semi-centralised systems, where one or more nodes within the network act as index nodes (similar to ICQ). Currently the NetworkModule does not provide any support for general node discovery, only service discovery.

Setting up

In order to use the network module you must import:

```
import net.jxta.pipe.*;
```

```
import net.jxta.endpoint.Message;
```

and implement **MessageInterface** within your application.

For example,

```
// Example application
```

```
import net.jxta.pipe.*;
```

```
import net.jxta.endpoint.Message;
```

```
import java.awt.*;
```

```
public class MyApp extends Frame implements MessageInterface
```

```
{
```

```
...
```

These commands import aspects of JXTA that your application will need to know about, and also provides a way in which the network module can communicate back to the application (via the MessageInterface). Because MessageInterface it is also necessary to at least define the method **public void MessageArrived(Message mess)**

To initialise the module:

```
NetworkModule module = new NetworkModule(this);
```

You pass a reference of the application (this), so the networkmodule can in turn inform the application when a message arrives

To connect to JXTA

```
String pipead = module.ConnectToJXTA(String pipeadvert, String pipename);
```

---- to initialise the node and connect it to the JXTA network

This method returns the pipe advert for the node as a String. The method will create a pipe advert if one is not passed as a parameter. If this is the case *pipename* can be used to give the to be created pipe a name.

PipeService pipeSvc = module.pipeSvc; ---- you need this if you are going to create messages

To obtain info about your node

String UniqueID = module.GetUniqueID(); ---- to get the JXTA uniqueID of the node

String NodeAdvertisement = module.GetNodeAdvertisement(); ----- to get the advert for the node

String NodePipeAd = module.GetNodePipeAd(); ----- to get the advert for the nodes input pipe

To find a service on the JXTA network

String ServicePipeAd = module.FindService(String ServiceName); ----- returns the pipe advert for that service

To send a message

module.SendMessage(Message mess, String TargetsInputPipeAdvert); ---- this method sends a Message object to the pipe designated by the target input pipe advert. You must construct the Message object before hand. This method can also return a string success report. NB, this method is generic and can be used to communicate to any other peer. For example, it can be used to send a text message to another peer, or be used to contact the server to make a request for another nodes pipe advert.

To load an advert

String ad = module.LoadAdvert(InputStream is); ---- this method loads in an advert from the specified input stream. Such input streams could be files or URLs. The advert is returned as a string.

To setup a service on the node

SetupService(String ServiceName, String ServiceDescription, String ServiceVersion, String ServiceCreator, String ServiceURL); ---- this method sets up and advertises a service on the node. The parameters that are passed make up the details of the advert

To receive a message

To receive a message your application **must** have defined the following method

public void MessageArrived(Message mess)

when the networkmodule receives a message it passes it up to the application by calling this method. It is up to the application to decode the Message object. Note: depending on how your application operates, you might want to add the **synchronized** keyword to this method

General comments

Adverts are converted into proper adverts by the network layer. To reduce an applications dependence on JXTA, all adverts are converted to strings when passed to the application, and converted back when passed the other way. Therefore the application should only ever have to deal with String based adverts.

All pipes are secureunicast pipes. This is the default and it what is created on initiation of the NetworkModule.

Further development.

This version of the networkmodule takes care of the more general JXTA methods. Possible additions to the module include network monitoring methods (what peers are out there, etc), and broadcast messaging (maybe useful for a server). Although the JXTA Network Module focuses on semi-centralised systems, it is possible that at a later date a decentralised version will be produced. It is likely that other additions will be made over time.