APS TCPIP Service Provider

User Guide

Version: 1.0.0

Author: Tommy Svensson

Table of Contents

1 APSTCPIPService

This service provides, in ways of communication, plain simple TCP/IP communication. Users of this service will however have very little contact with the java.net classes.

The following are the points of this service:

- Simple TCP/IP usage.
- Makes use of an URI to provide what I call a "connection point". tcp:, udp:, and multicast: are supported protocols.

Do note that you do need to have a basic understanding of TCP/IP to use this service!

1.1 Security

Makes use of 2 separate services if available for security: *APSTCPSecurityService* and *APSUDPSecurityService*. Neither these nor APSTCPIPService makes any assumptions nor demands on the what and how of the security services implementations. The APSTCPSecurityService must provide secure versions of Socket and ServerSocket, while APSUDPSecureService have 2 methods, one to encrypt the data and one to decrypt the data in a DatagramPacket.

APS currently does not provide any implementation of the APS(TCP/UDP)SecurityService.

1.2 Connection Point URIs

The service makes use of URIs to specify where to connect for sending or receiving.

The URI format is this:

protocol://host:port#fragment,fragment

Protocols:

tcp, **udp**, **multicast**, **named**

The *named* protocol just provides a name for the *host* part of the URI. This is not however a hostname! It is a name that has been entered in the service configuration and which has an URI value which will be used instead. So *named://mysvc* will lookup a config value having "mysvc" as destination name and use its destination URI as connection URI. So the valid URI protocols in the configuration is then tcp, udp, and multicast.

Fragments:

secure - If specified then one of the APS(TCP/UDP)SecurityService services will be used.

async (only valid on *tcp* protocol)

1.3 Examples

1.3.1 TCP

1.3.1.1 Write

```
    APSTCPIPService tcpipSvc;
    ...
    tcpipSvc.sendStreamedRequest(new URI("tcp://localhost:9999"), new StreamedRequest()
{
```

```
        void sendRequest(URI connectionPoint, OutputStream requestStream, InputStream
responseStream) throws IOException {
            // write to requestStream ...

            // read from response stream ...
        }
    })
```

1.3.1.2 Read

```
    APSTCPIPService tcpipSvc;
    ...
    tcpipSvc.setStreamedRequestListener(new URI("tcp://localhost:9999"), this);
    ...
    void requestReceived(URI receivePoint, InputStream requestStream, OutputStream
responseStream) {
        // Read request from reqStream ...

        // Write response to respStream ...
    }
```

Note that there can only be one listener per URI.

1.3.2 UDP / Multicast

Since Multicast uses UDP packets there is no difference between host and port connected UDP or Multicast. The only difference is in the URI where "udp://" is specified for UDP packets and "multicast://" is specified for multicast packets.

1.3.2.1 Write

```
    APSTCPIPService tcpipSvc;
    ...
    bytes[] bytes = "Some data".getBytes();
    tcpipSvc.sendDataPacket(new URI("udp://localhost:9999"),  bytes);
```

or

```
    tcpipSvc.sendDataPacket(new URI("multicast://all-systems.mcast.net:9999"), bytes);
```

1.3.2.2 READ

```
    APSTCPIPService tcpipSvc;
    ...
    tcpipSvc.addDataPacketListener(new URI("udp://localhost:9999"), this);
    ...
    void dataBlockReceived(URI receivePoint, DatagramPacket packet) {
        byte[] bytes = packet.getData();
        ...
    }
```