UDP Transport

UDP Transport

CXF provides a transport plugin to support transporting small (under about 60K) message payloads over UDP. It supports both unicast and multicast packet transfers.

Creating a server on a UDP port

To use the UDP transport, you just need to include the cxf-rt-transports-udp module on the classpath and use a "udp://host:port" style URL for the address.

```
        JaxWsServerFactoryBean factory = new JaxWsServerFactoryBean();
        factory.setBus(getStaticBus());
        factory.setAddress("udp://localhost:8888");
        factory.setServiceBean(new GreeterImpl());
        server = factory.create();
```

That will start the server on localhost UDP port 8888. You can also omit the hostname (udp://:8888) to bind to all the addresses or use one of the multicast addresses (example: udp://239.255.255.250:3702) to respond to the appropriate broadcasts.

Client configuration

Similar to the server side, you just need to use the appropriate UDP url in order for the client to use UDP.

If the hostname is specified in the URL, the Datagram will be sent directly to the host:port. If the hostname is not specified, the Datagram will be sent as a broadcast to the specific port. If the hostname is a multicast address, the Datagram will be sent Multicast to the given port.

Accepting multiple responses

UDP is different than the other CXF transports in that it allows multiple responses to be received for a single request. For example, if you send out a request via a multicast or broadcast, several servers could respond to that request. The basic JAX-WS generated interfaces only allow a single response to be returned to the application. However, if you use the JAX-WS Asynchronous methods, you can have CXF call the AsyncHandler for each response. To enable this, set the request property "udp.multi.response.timeout" to a timeout value greater than 0. CXF will wait that long for responses to come in before returning back to the application.

```
//wait 3 seconds for responses
((BindingProvider)proxy).getRequestContext().put("udp.multi.response.timeout", 3000);
proxy.greetMeAsync("World", new AsyncHandler<String>() {
    public void handleResponse(Response<Object> res) {
            System.out.println(res.get());
    }
});
```