What is mSoap?

mSoap is a SOAP client library for Java ME (J2ME) designed for use with the now aged GLUE SOAP server library. However it is the authors hope and goal that it will work well with most SOAP servers out there. mSoap is released by Machina Networks AS under the GPL version 2 or later.

Using mSOAP

SOAP calls are performed in msoap by asking a transporter (currently only HTTPTransport exists) to create a soap call based on an SOAPEnvelope object. The transport then makes the soap call and returs a new SOAPEnvelope which contains the results. This documents descibes the most common steps that are needed to make simple and complex SOAP requests

Requirements

mSoap requires CLDC 1.1 and MIDP 1.0. The reason for the rather harsh CLDC 1.1 requirement is that mSoap's support for double primitive type depends on the existence of java.lang.Double. This is likely to be remidied in a future relese. In the mean time it should be a relatively simple task to remove all references to double and recompile mSoap.

Simple SOAP requests

A simple soap request is defined as a SOAP request which contains only primitive values. There is a complete example of a simple request in no.machina.msoapapi.examples.WhoIsMidlet

The firs step is as allways to get a new SOAPEnvelope object and set the method and namespace. The Method will be the name of the first tag after the body tag when the enveope is serialized and should indicate to the webservice exatcly which service you are trying to reach. The namespace is the namespace that the method can be found in.

```
   //First a SOAP envelope must be created.  Every SOAP call needs
one without exception
SOAPEnvelope se = new SOAPEnvelope();

   //Every SOAPenvelope needs a namespace.  This is the target namespace
found in the WSDL file or elsewhere
se.setNameSpace("http://somenamespace.example.com");

   //The soap method to be called
se.setMethod("ExampleMethod");
```

When the envelope is created it will useally have to be filled with parameters. In this simple example we will add one string and one integer, se the section on supported primitives for a list of supported primitives. The name part of the methods to add the primitive determines wath the name of the value will be once it's serialized.

```
se.addString("arg0", "example parameter");
se.addInt("arg1", new Integer(5)
```

Then it is time to send the request. For this we need a transport. mSoap comes with one such transport, HTTPTransport. It's constructor takes two arguments, a boolean value indicating wheter or not it should try to keep sessions alive by returning cookies to the site in subsequent requests and the URL to the web service the transport will send all requests.

```
//A transport is nessesary to do the actual SOAP method call.  HTTPTransport
is the only one currently available.
HTTPTransport tansport = new HTTPTransport(false, "http://example.com/service");

//Makes the actual transport call wich returna a new envelope
SOAPEnvelope resultEnvelope = tansport.request(se);

//Any well behaved resultEnvelope has a response object
SOAPObject response = resultEnvelope.getResult();
```

And that's it. "response" can be used to get all information that is found in the body of the soap envelope, any other information cvan be reached through the envelope itself.

Complex soap requests

The only difference between a simple and complex soap request is that a complex request contains a parameter that is not a primitive. These object myst implement the interface SOAPObject so that it can be serialzed and created by the mSoap serializers and parsers. Some simple exaples of such classes can befound in the package no.machina.msoapapi.examples.AmazoneClasses. These classes are used in no.machina.msoapapi.examples.BookInfo. In order for the parser to return custom object the object must first be registered with the requesting envelope (the register will be copied to the result enveope by the transport). This is done like so:

```
se.registerClass(new customSOAPObject().getClass(), "nameSpace",
"type");
```

If you are connecting to a webservice that returns in the document/literal style/encoding, the namespace must be a empty String.

Supported primitives

- int
- long
- double
- decimal
- string

Credits

The mSoap distribution includes a binary version of kXML. For more information on kXML see http://kxml.sourceforge.net/.

Future enhacements

- Add support for compressed streams.
- Add support for binary xml formats.
- Create a tool to make SOAPObjects out of objects in a WSDL.
- Fix all the bugs that are left and make the api simpler and make the code better and faster and make better documentation. In general, everything will hopefully be improved.