SOAP Messaging API for txttools v6.4

Software Version - 6.4
Document Version - 1.5

John Hunsley / Andrew Alexander
aalexander@txttools.co.uk
Date – 02/06/2010

Contents

Document Change Log

1.4 Unicode section and element added. TTL element removed.

1.5 InboxMessageRequest example updated.

1.0 Introduction

1.1 Intended Audience

This document is aimed at txttools customers who wish to send and/or receive SMS text messages from/to their own application without using the txttools web user interface.

Before reading this document and implementing a software client the reader should have a solid understanding of the Hyper Text Transfer Protocol (HTTP) and Extensible Markup Language (XML). In particular, the reader should understand the HTTP request and response mechanisms and be able to create well formed XML strings of data.

For more information regarding these technologies please refer to the following guides

HTTP
http://www.w3.org/Protocols/
http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

XML
http://www.xml.com/pub/a/98/10/guide0.html
http://en.wikipedia.org/wiki/XML

1.2 Useful Tools

To leverage the usability of SOAP web services you will require a SOAP client capable of processing WSDL files and making SOAP calls to the API. Your aim is to integrate your application/database/MIS System with the txttools SOAP API, for which you will need to implement your own client. Java, .net, Oracle and many other platforms provide some sort of SOAP client support framework enabling you to create your own client as part of your application.

You will however require a 'stand alone' SOAP client which you can use to test out the API and formulate the required XML to make a SOAP call. I suggest using a Java based client called the Soap UI. It is platform independent and requires the Java Runtime Environment installed which you should already have as part of a standard Windows, MAC or Linux operating system distribution. The Soap UI is free and may be downloaded from the following url

http://www.soapui.org/

There are many other SOAP clients out there, but for the purposes of the tutorials and examples in this document I will assume you are using the Soap UI.

You may also require some XML editing schema validation tools to help you create and validate your XML.

http://www.xmlcooktop.com/

If you would like to closely inspect the HTTP traffic between your client and the SOAP API then I suggest using the following application.

http://www.parosproxy.org

1.3 What is SOAP?

Simple **O**bject **A**ccess **P**rotocol is an implementation of Service Oriented Architecture. It is a standard method for exchanging messages in XML format between computers over a network, usually HTTP. XML data may be passed between computers over HTTP on the request and response. SOAP is an extension of this and defines a standard XML format and port of call for schema discovery. A SOAP API is usually accompanied by a Web Service Description Language (WSDL) file and it is this which should be your first port of call for information about the services available to the client.

For a detailed discussion of SOAP and WSDL files see -

http://en.wikipedia.org/wiki/SOAP
http://en.wikipedia.org/wiki/Web_Services_Description_Language

The WSDL file is an XML document which describes the ports and end points of services which may be invoked by a call and the XML schema expected by each service. The schema definitions show what data is required, the format it must be in and the nature of the response the client should expect when invoking an operation at an end point. This document is not meant to be human readable, however if you provide it to a client, such as the Soap UI the file will be processed and a default request schema created for each service end point. The client requires only the WSDL file, the service end point name and the data in order to make a call to a service and invoke an operation.

1.4 Why use the SOAP API?

txttools provides basic XML over HTTP APIs for both messaging and address book functions. To use these APIs the client must provide an XML string in the expected schema on the HTTP request. The SOAP mechanism is exactly the same, its just some XML sent over HTTP on the request. So why bother using SOAP if we already have an XML/HTTP API?

The reason is simple! The SOAP client does not need to know anything about the expected XML schema prior to reading the WSDL, whereas an XML/HTTP API client must provide the XML in the schema outlined in the documentation for that API. The programmer creating the client must ensure the XML is hard coded with the known schema.

If you don't want to waste time hard coding String XML values into your client then use the SOAP API rather than the old basic XML/HTTP API.

1.5 A note on SOAP and .net

If you choose to use .net as your development platform, it is important to be aware of a discrepancy between SOAP and .net. Whilst the official specification for SOAP states that the nonce tag in the SOAP header is optional, Microsoft makes the nonce tag mandatory. There is unfortunately no way of removing the nonce tag from a SOAP header created programmatically by .net. As a result, it is necessary to hard code the SOAP header in this circumstance.

2.0 Authentication

A client must authenticate before invoking any operation from the SOAP API. The client must provide the username and password in plain text for the account they wish to use. If you are using the messaging API any messages you send will be sent from that account, and any inbox messages you retrieve will come from that account and will be designated as 'read' once the API has successfully returned them in the SOAP response.

If you are using the address book API you will be managing contacts and groups from that account address book and any account which shares that address book. You must have permission to alter the address book before invoking any Address Book API operation.

All txttools SOAP APIs employ the standard Web Services Security (WSSE) specification to define authentication credentials. The specification is defined by Oasis Open Standards Consortium. You can find more information at the following url -

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

Clients must provide their credentials in plain text in the SOAP header of the message. It is strongly recommended you use HTTPS for all your communication with all txttools SOAP APIs. This will secure both your authentication credentials in the header and any information you provide in the SOAP body.

Clients must provide both their account username and password in the format outlined in the example below.

```
<soapenv:Header>
      <wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://docs.oasis-
      open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <wsse:UsernameToken wsu:Id="UsernameToken-2750779"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
      utility-1.0.xsd">
      <wsse:Username>USERNAME</wsse:Username>
      <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-username-token-profile-1.0#PasswordText">PASSWORD</wsse:Password>
  </wsse:UsernameToken>
      </wsse:Security>
</soapenv:Header>
```

3.0 The SMS Messaging API

The txttools SMS Messaging WSDL file can be found at the following url -

https://www.txttools.co.uk/connectors/soap/messagingk/txttoolsMessaging.wsdl

Processing this WSDL file reveals one service –

- txttoolsMessagingServiceBinding

This service has three distinct operations -

- Message
- Status
- InboxMessage

3.1 MT SMS Message sending

Invoking the Message operation will send a Mobile Terminated (MT) SMS text message to the recipients phone number(s). The namespace prefix for all elements for this operation is - **`sch`**

The following table describes the elements of the XML schema for this operation.

Element	Description & format	Required	Cardinality
`MessageRequest`	Wraps the entire messages request body	Yes	1
`Message`	Warps the message request	Yes	1 - many
`SuppressUnicode`	Boolean which denotes whether to override the automatic configuration of a message to Unicode. Defaults to false if not supplied. See section **3.1.2**.	No	0-1
`MessageText`	The content of the SMS message. Not null. No limit on message length. If any special characters are used then escape the entire data string using the CDATA tag.	Yes	1
`Recipient`	The international prefixed mobile phone number of the intended recipient. 13 – 15 characters. First character may be '+' , all others must be numeric '0-9'. Should be unique if multiples supplied.	Yes	1 - many
`Source`	The source of the message. An	No	0 - 1

	alpha numeric value min length 1 max length 13 characters. If not supplied this will be the account default message source		
SendTime	Future send time in yyyy-MM-dd HH:mm format. Defaults to now if not included or in the past	No	0 - 1
Header	Binary or Hex header for UDH setting. If included, MessageText must also be in binary or hex.	No	0 - 1

3.1.1 Examples

Send a simple SMS message to a single recipient -

```
<soapenv:Body>
 <sch:MessageRequest>
   <sch:Message>
    <sch:Recipient>+447772211117</sch:Recipient>
    <sch:MessageText>I love SOAP</sch:MessageText>
   </sch:Message>
 </sch:MessageRequest>
</soapenv:Body>
```

Send an SMS message to multiple recipients and alter the source -

```
<soapenv:Body>
 <sch:MessageRequest>
   <sch:Message>
    <sch:Recipient>+447772211117</sch:Recipient>
    <sch:Recipient>+447971817195</sch:Recipient>
    <sch:Recipient>+447817577380</sch:Recipient>
    <sch:MessageText>I love SOAP</sch:MessageText>
    <sch:Source>txttools</sch:Source>
   </sch:Message>
 </sch:MessageRequest>
</soapenv:Body>
```

3.1.2 Unicode

If the message text contains a character not found in the GSM charset, the message will be sent as a Unicode message. Whilst this will ensure that all Unicode characters are interpreted correctly, there are also some disadvantages to consider. Firstly, not all mobile phones support full Unicode. Secondly, each Unicode message may contain a maximum of 70 characters only, rather than the usual 160 characters. The automatic configuration of a message to Unicode can therefore be overridden by setting the **SuppressUnicode** element to true. Bare in mind, however, that any Unicode characters contained within the message will not be properly displayed on the recipient's phone. If the **SuppressUnicode** element is not supplied, it will default to false.

3.1.3 Response

Upon successful invocation of the Message operation the API will respond to the client request with information regarding each message sent to each specified recipient. This information will contain a unique ticket number for each message sent to each recipient specified in the request. It is important that clients record this ticket value as it can be used at a later time to request further information about the status of the message.

No SOAP header values are specified in the response, only the SOAP body contains information. The name space prefix for all body elements for this type of response is - **`txttoolsMessaging`**
The following table describes the elements of the XML schema for the response.

Element	Description & format	Required	Cardinality
`StatusResponse`	Wraps the entire soap body response for all message recipient responses.	Yes	1
`StatusResponseType`	Wraps each individual message recipient response.	Yes	1 - many
`Recipient`	The recipient phone number in international standard format to which the MT SMS message was sent. The leading character will always be '+' followed by numeric characters.	Yes	1
`Ticket`	The unique ticket number for this message to the recipient phone number. Positive integer.	Yes	1
`Status`	The integer status code for this recipient message. Status values are as follows - -1 – Failed at txttools 0 – Queued at txttools 1 – Sent to network 2 – Failed at network 3 – Delivered to network 4 – Retrying 5 – Delivered to hand set 6 – Unknown failure The initial response will usually be 0 if sent, or -1 if you have no credit.	Yes	1
`StatusMessage`	The message for the given status as outlined above.	Yes	1

3.1.4 Message Response Example

A response to a single recipient Message operation invocation -

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
   <txttoolsMessaging:StatusResponse
xmlns:txttoolsMessaging="http://www.txttools.co.uk/connectors/soap/messaging/schemas">
     <txttoolsMessaging:StatusResponseType>
      <txttoolsMessaging:Recipient>+447772211117</txttoolsMessaging:Recipient>
      <txttoolsMessaging:Ticket>414469</txttoolsMessaging:Ticket>
      <txttoolsMessaging:Status>0</txttoolsMessaging:Status>
      <txttoolsMessaging:StatusMessage>Queued at
txttools</txttoolsMessaging:StatusMessage>
     </txttoolsMessaging:StatusResponseType>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.2 MT SMS Message Status Reports

As described in 3.1, when the client invokes the Message operation and sends a Mobile Terminated (MT) SMS message the service will return a ticket value for every recipient of that message. This ticket can be used to request status information for that message by invoking the Status operation.

The namespace prefix for all elements for this operation is - **`sch`**

The following table describes the elements of the XML schema for this operation.

Element	Description & format	Required	Cardinality
`StatusRequest`	Wraps all Ticket status values for this request	Yes	1
`Ticket`	Wraps the Ticket values tag for all ticket request	Yes	1
`TicketNumber`	The ticket number as provided by the Ticket element in the response to the Message operation invocation. Must be a positive integer.	Yes	1 - many

3.2.1 Examples

Get the status of the message with ticket number 12345 -

```
 <soapenv:Body>
   <sch:StatusRequest>
     <sch:Ticket>
      <sch:TicketNumber>12345</sch:TicketNumber>
     </sch:Ticket>
   </sch:StatusRequest>
  </soapenv:Body>
```

Get the status of multiple messages -

```
    <soapenv:Body>
```

```
  <sch:StatusRequest>
    <sch:Ticket>
     <sch:TicketNumber>12345</sch:TicketNumber>
           <sch:TicketNumber>12346</sch:TicketNumber>
    </sch:Ticket>
  </sch:StatusRequest>
</soapenv:Body>
```

3.2.2 Response

If the client successfully invokes the operation with a valid ticket number(s) the service will respond with the latest information for the messages with those tickets.

No SOAP header values are specified in the response, only the SOAP body contains information. The name space prefix for all body elements for this type of response is - **`txttoolsMessaging`**

The response schema for this operation is exactly the same as the response for the Message operation as outlined in 3.1.2 The following table describes the elements of the XML schema for the response.

Element	Description & format	Required	Cardinality
`StatusResponse`	Wraps the entire soap body response for all message recipient responses	Yes	1
`StatusResponseType`	Wraps each individual message recipient response	Yes	1 - many
`Recipient`	The recipient phone number, in international standard format to which the MT SMS message was sent. The leading character will always be '+' followed by numeric characters	Yes	1
`Ticket`	The unique ticket number for this message to the recipient phone number. Positive integer	Yes	1
`Status`	The integer status code for this recipient message. Status values are as follows - -1 – Failed at txttools 0 – Queued at txttools 1 – Sent to network 2 – Failed at network 3 – Delivered to network 4 – Retrying 5 – Delivered to hand set 6 – Unknown failure	Yes	1
`StatusMessage`	The message for the given status as outlined above.	Yes	1

3.2.3 Status Response Example

A response to a single recipient Status operation invocation with ticket number 12345 -

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
   <txttoolsMessaging:StatusResponse
xmlns:txttoolsMessaging="http://www.txttools.co.uk/connectors/soap/messaging/schemas">
     <txttoolsMessaging:StatusResponseType>
      <txttoolsMessaging:Recipient>+447772211117</txttoolsMessaging:Recipient>
      <txttoolsMessaging:Ticket>12345</txttoolsMessaging:Ticket>
      <txttoolsMessaging:Status>0</txttoolsMessaging:Status>
      <txttoolsMessaging:StatusMessage>Queued at
txttools</txttoolsMessaging:StatusMessage>
     </txttoolsMessaging:StatusResponseType>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.3 MO SMS Message receiving

Mobile Originating (MO) SMS messages may be directed to a txttools account inbox and can be read online. It is also possible to retrieve unread messages from the inbox through the SOAP API. Once retrieved, MO messages will remain in the inbox but will be marked as 'read' and will not be eligible for retrieval through the API.

The InboxMessage operation may be invoked by the client. The only parameter required is the number of unread MO messages the client wishes to retrieve. If zero all unread messages are returned in the SOAP response.

The namespace prefix for all elements for this operation is - **`sch`**

The following table describes the elements of the XML schema for this operation.

Element	Description & format	Required	Cardinality
`InboxMessageRequest`	Wraps the entire soap body response for inbox message requests	Yes	1
`MaxReturn`	The number of unread MO messages to retrieve. A positive integer or zero returns all unread MO messages.	Yes	1

3.3.1 Example

Retrieve just one unread MO message from the inbox -

```
<soapenv:Body>
  <sch:InboxMessageRequest>
    <sch:MaxReturn>1</sch:MaxReturn>
  </sch:InboxMessageRequest>
</soapenv:Body>
```

3.3.2 Response

If the client has successfully invoked the InboxMessage operation the details of those MO messages

are returned in the response. The details of the response schema is outlined below.

No SOAP header values are specified in the response, only the SOAP body contains information. The name space prefix for all body elements for this type of response is - **txttoolsMessaging**

Element	Description & format	Required	Cardinality
InboxMessageResponse	Wraps all MO Message details for this response.	Yes	1
InboxMessageResponseType	Wraps all details for a single MO Message.	Yes	1 - many
Source	The originating phone number of this message. Could be alphanumeric but will usually be an internationalized phone number beginning with '+'.	Yes	1
Content	The content of the MO message, max 160 characters.	Yes	1
Destination	The destination the message was sent to.	Yes	1
DeliveryTime	The date / time the MO message was delivered to the inbox. YYYY-MM-DD HH:MM	Yes	1

3.3.3 InboxMessage Response Example

A single MO message -

```
 <SOAP-ENV:Body>
   <txttoolsMessaging:InboxMessageResponse
xmlns:txttoolsMessaging="http://www.txttools.co.uk/connectors/soap/messaging/schemas">
     <txttoolsMessaging:InboxMessageResponseType>
      <txttoolsMessaging:Source>+447772211117</txttoolsMessaging:Source>
      <txttoolsMessaging:Content>Why do bananas all bend the same way?
</txttoolsMessaging:Content>
      <txttoolsMessaging:Destination>88020</txttoolsMessaging:Destination>
      <txttoolsMessaging:DeliveryTime>2008-08-12 23:03</txttoolsMessaging:DeliveryTime>
     </txttoolsMessaging:InboxMessageResponseType>
   </txttoolsMessaging:InboxMessageResponse>
  </SOAP-ENV:Body>
```

3.4 Reports Request

The Messaging Service provides an operation which may be invoked to request an MT message statistical report for a given period. The client may request a report for their authenticated account or sub account of which the authenticated account is an administrator. The client must provide a start and end date for the period over which the report will be generated. The format for both start and end dates is day, month and year. The periods are inclusive of those dates.

The namespace prefix for all elements for this operation is - **sch**

The following table describes the elements of the XML schema for this operation.

Element	Description & format	Required	Cardinality
ReportsRequest	Wraps the entire request for a MT.	Yes	1
SubUser	Denotes the user to generate the report for. The ID must be a positive integer and that user must be a sub user of the authenticated account the client is using to make the request. If omitted, the report will be generated for the authenticated user.	No	1
StartDate	Encloses the information detailing the start date from which to generate the report. Must contain dd, mm and YYYY sub elements.	Yes	1
EndDate	Encloses the information detailing the end date to which the report will be generated. Must contain dd, mm and YYY sub elements.	Yes	1
dd	The day of month. Positive integer 01 – 31.	Yes	1 in StartDate or EndDate context
mm	The month of year. Positive integer 01 – 12.	Yes	1 in StartDate or EndDate context
YYYY	The year. Positive integer 2000 – 2100.	Yes	1 in StartDate or EndDate context

3.4.1 Example

The following reports request will generate a report for the authenticated user between the 1st November 2008 and the 8th November 2008 inclusive.

```
<SOAP-ENV:Body>
<sch:ReportsRequest>
```

```
      <sch:StartDate>
       <sch:dd>01</sch:dd>
       <sch:mm>11</sch:mm>
       <sch:YYYY>2008</sch:YYYY>
      </sch:StartDate>
      <sch:EndDate>
       <sch:dd>07</sch:dd>
       <sch:mm>11</sch:mm>
       <sch:YYYY>2008</sch:YYYY>
      </sch:EndDate>
    </sch:ReportsRequest>
  </SOAP-ENV:Body>
```

3.4.2 Response

Following a successful invocation of the Reports operation the server will return a SOAP response with figures for Failed (Red), Unconfirmed (Amber), Confirmed Received (Green), and the total MT messages sent from the requested account for the given period inclusive of the given dates.

The name space prefix for all body elements for this type of response is - **`txttoolsMessaging`**

Element	Description & format	Required	Cardinality
`Totals`	Encloses the report response containing the sub elements detailing the totals for each status and the total of MT messages.	Yes	1
`Red`	Details the number of failed MT messages for the requested period.	Yes	1
`Amber`	Details the number of MT messages which are unconfirmed as either failed or delivered.	Yes	1
`Green`	Details the number of MT messages which are confirmed received by the hand set.	Yes	1
`Total`	Details the total number of MT messages of all three statuses sent over the given period.	Yes	1

3.4.3 Reports Response Example

```
<SOAP-ENV:Body>
   <txttoolsMessaging:Totals
xmlns:txttoolsMessaging="http://www.txttools.co.uk/connectors/soap/messaging/schemas">
     <txttoolsMessaging:Red>3</txttoolsMessaging:Red>
     <txttoolsMessaging:Amber>29</txttoolsMessaging:Amber>
     <txttoolsMessaging:Green>5</txttoolsMessaging:Green>
     <txttoolsMessaging:Total>37</txttoolsMessaging:Total>
```

```
  </txttoolsMessaging:Totals>
 </SOAP-ENV:Body>
```

3.5 Error Handling

Should the SOAP client attempt to invoke an operation with insufficient or invalid data, a standard SOAP error message will be generated and returned in the response. Failure to authenticate, due to invalid credentials or an inactive account, will result in a WSSE authentication error.

If the client is attempting to invoke the Message operation, the MT SMS message(s) will not be sent to the mobile networks if a fault is reported. Only a status > -1 returned in the response denotes that the message has been successfully received and processed by the API. If the client receives a fault or a status of < 0 the message should be retried by the client at a later time.

4.0 Messaging Tutorial

This tutorial will take you through the process of sending an MT SMS message through the txttools SOAP Messaging API using the Message operation. We will then use the response data to request the status of that message and retrieve MO SMS Message data. This tutorial will use the Soap UI, see section 1.2 for more details.

4.1 Download and install the Soap UI client.

Open up a browser and go to http://www.soapui.org. Download the latest free version of the Soap UI. Ensure your computer has the Java Runtime Environment installed.

Open the Soap UI application. If you are using a Unix based operating system issue the following command – *SOAP_UI_HOME~/bin/soapui.sh.* If you are using Windows double click the soapui.exe file. You will see the splash screen as the application starts up followed by the Soap UI graphical user interface.

4.2 Create a new project

Select *File* from the top menu bar and select *New WSDL Project.* Give the project a name and enter the txttools Messaging API WSDL url -

http://www.txttools.co.uk/connectors/soap/messaging/txttoolsMessaging.wsdl

Select to create default requests for all operations and hit OK. The Soap UI will now process the WSDL file and create default requests for all operations.

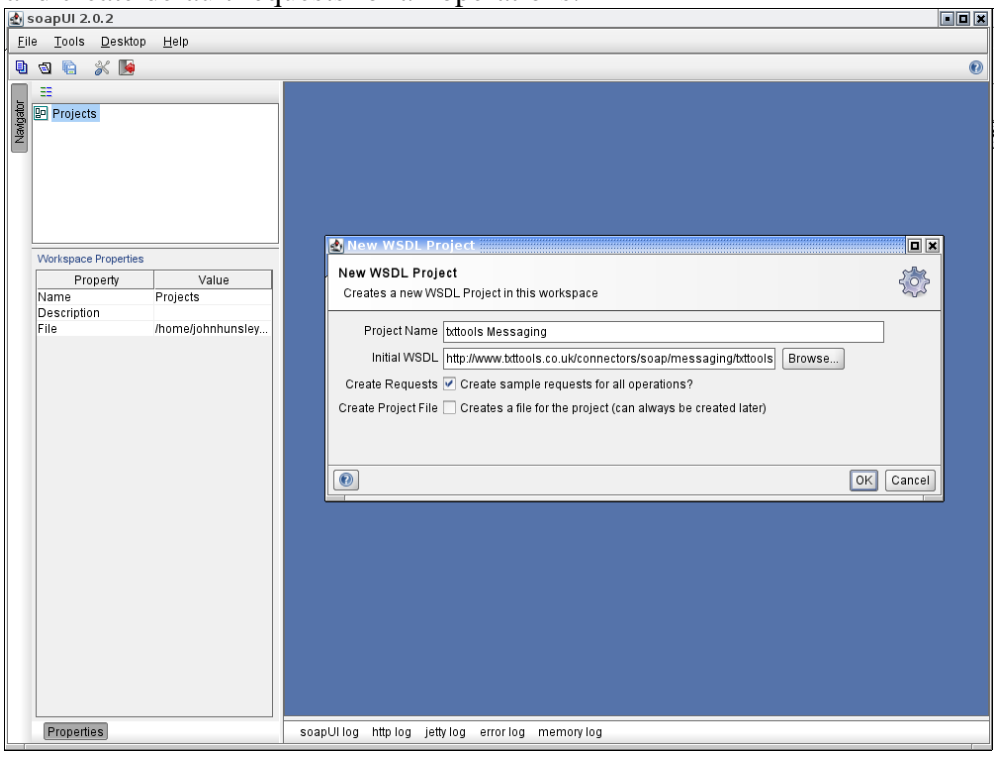

Fig1 Creating a new WSDL project

4.3 Add the WSSE authentication header

Open up the Message operation and double click on the default request which has been created for you. This will be called 'Request 1'. The Request Editor window will open and you will see the entire SOAP request XML ready for you to add some data - but first you need to add the authentication header.

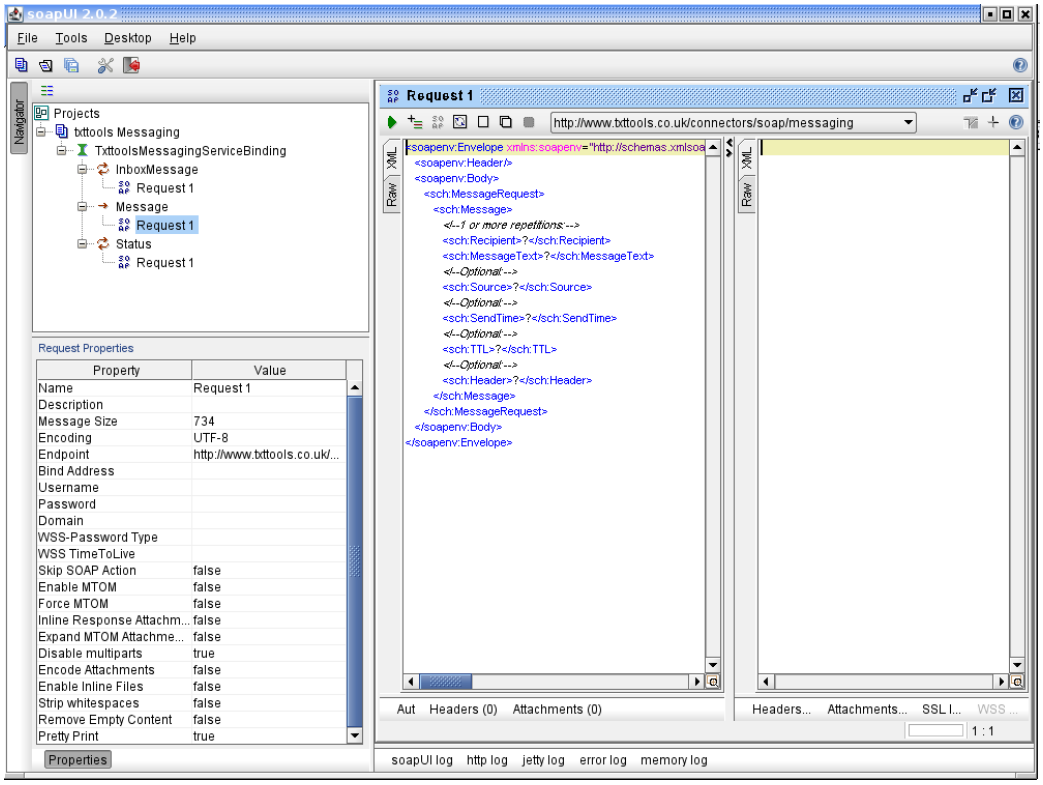

Fig 2 The Message Request

In the properties window in bottom left you will see the various properties and values for the request. Click in the username value field and enter your txttools account username. Now do the same for the password field. Right click in the request window again and click *'Add WSS Username Token'*. Select '*PasswordText'* and hit OK. The Soap UI will add the standard WSSE security header. I believe there is a bug here! Although you selected *PasswordText* it may have added the credentials as a digest hash. You only want them as plain text as you should be using Https to secure your request. If the header looks different to those in section 2.0 copy the example from 2.0 cut and paste them in to the request, and don't forget to edit the username and password values!

You request window should now look something like the image in fig 3 below -

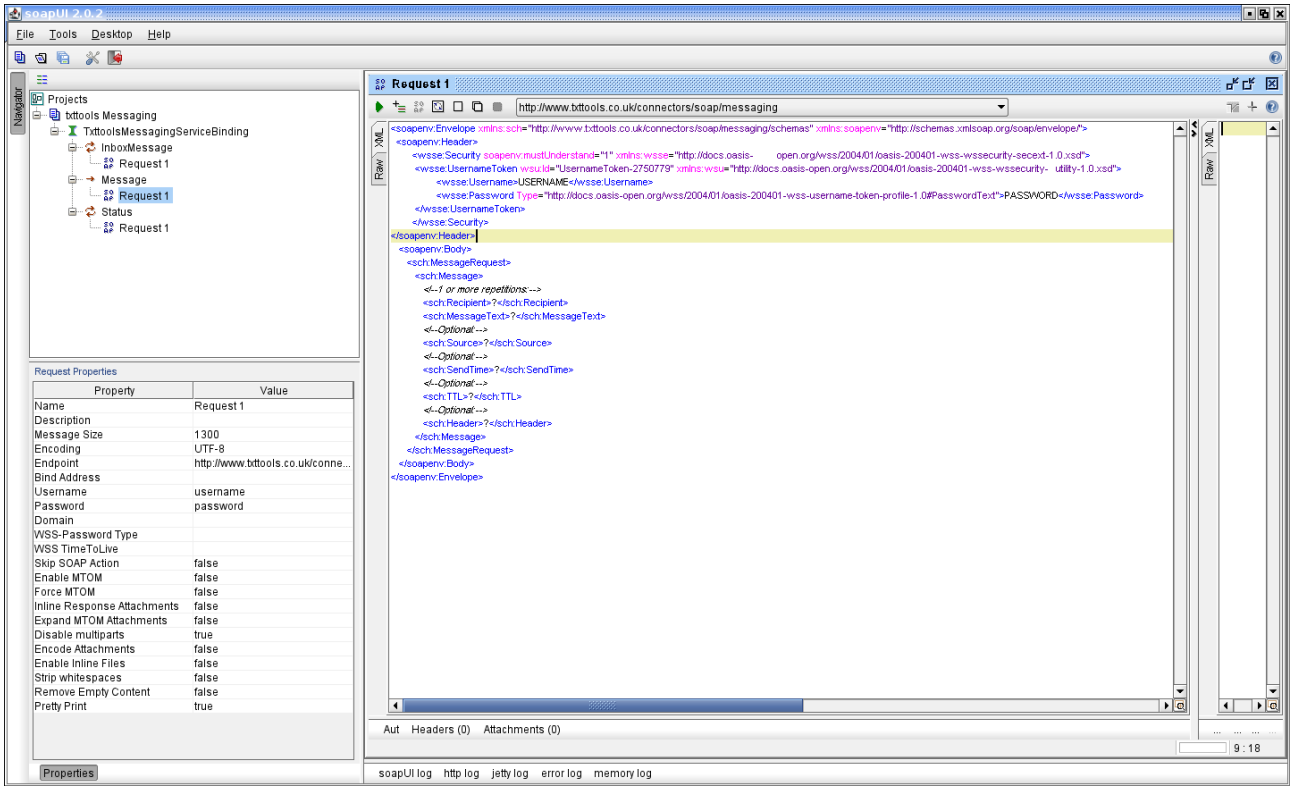

Fig 3 Adding the security header

4.4 Add some data to the Message request and invoke the operation

Now we have the header sorted out its time to add some data to the request body and make the call to the operation. Helpfully, the Soap UI has told us which elements are optional, we just want to send a simple message so we only really need the MessageText and one Recipient element. Enter your mobile phone number in international format over the '?' character in the Recipient element tag. Enter a message in the MessageText element, you can make the message as long as you like, i.e. more than the industry standard 160 characters. Remove all the other optional body elements and hit the green 'play' button in the top left corner of the request window. This will send the request and make the call.

The Response window should open on the right hand side of the screen showing the SOAP response to your request. Look closely for the Ticket element and note down the integer value. We will use this later to request the status of the message. If you sent a message to multiple recipients, or you sent more than 160 characters in the MessageText element, you will receive more than one StatusResponseType Element containing the initial status of each message. When you come to implement your own SOAP client you will need to parse this response and store each ticket value.

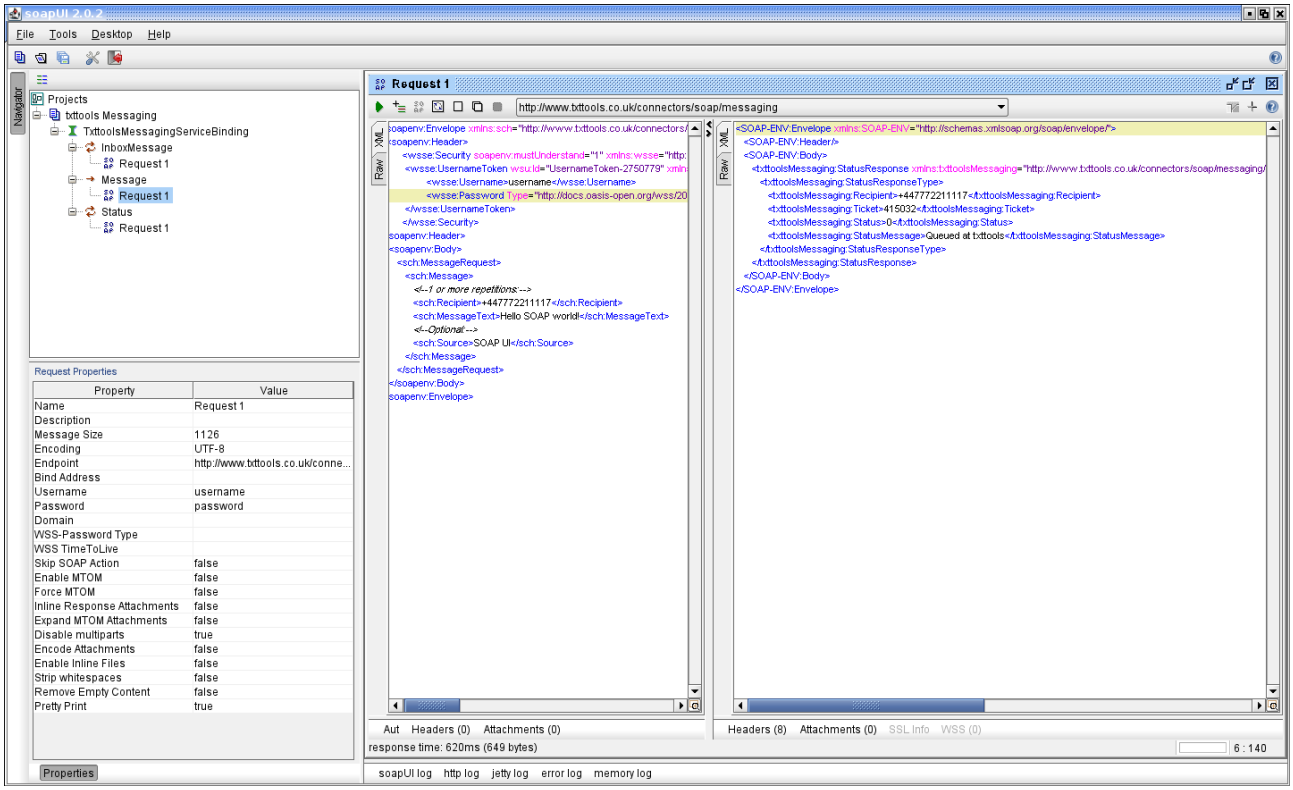

Fig 4 Response to a Message request

At this point you should receive the message on the phone :-)

4.5 Add the ticket number to the Status operation request

Now that we have successfully sent a message we can make a request to the Status operation and get information back concerning the statuses of the messages we have sent. You will need the ticket number(s) for each message you have sent.

Open up the default Status request and view the SOAP request in the request editor window. Add the WSSE security header the same way you did for the Message operation. There is only one body request value to enter, the TicketNumber element. If you want to make multiple requests for different message recipients or different messages add multiple TicketNumber elements.

Once you have formulated the request hit the 'play' button in the top left hand corner of the request window to make the call. You should see the response in the right hand window.

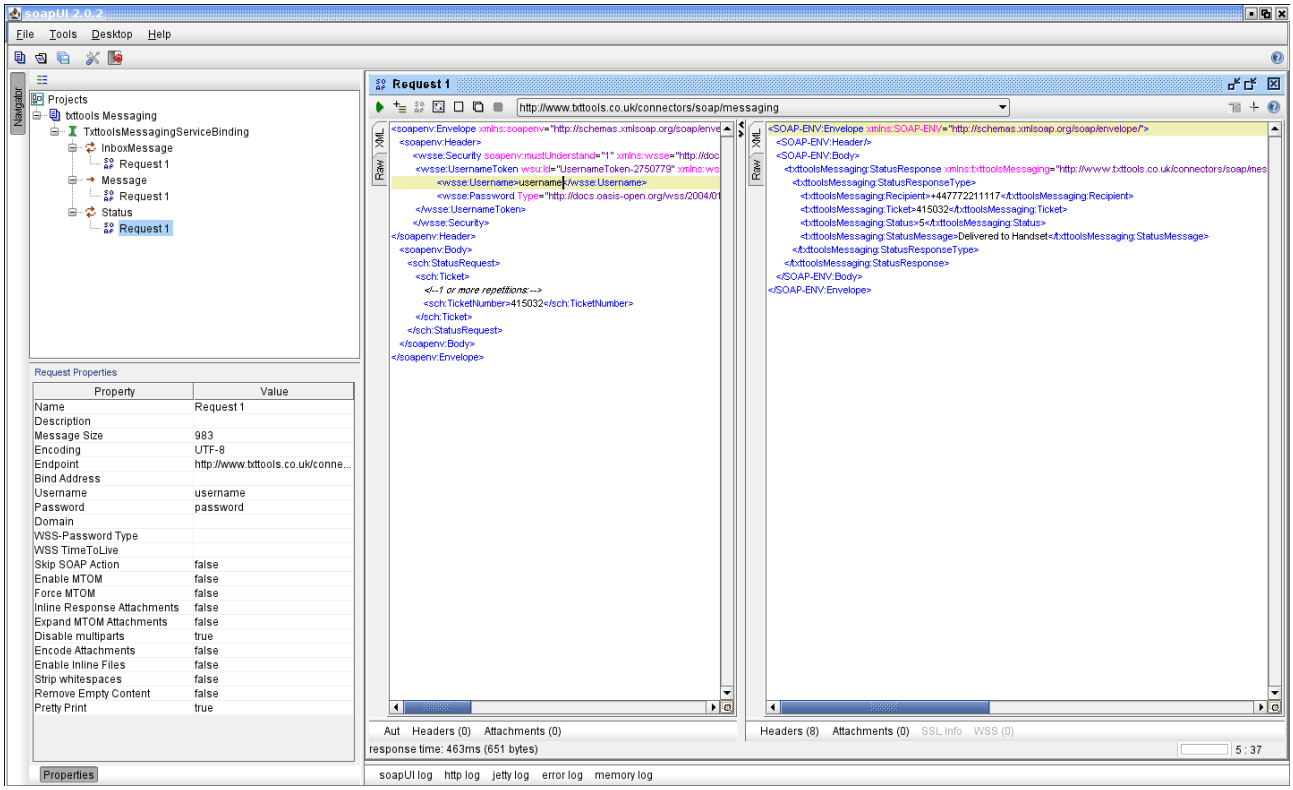

Fig 5 The Status request and response

You may wish to parse the response and store the status value and description against the message so it can be presented to the users on a report.

4.6 Make an InboxMessage operation request

The InboxMessage operation will allow you to retrieve MO messages from your txttools inbox. To test this effectively you will first need to send some messages from your phone into your txttools account inbox. You will need to set up inbound messaging on your account and ensure messages are routed directly, or by keyword, to your account inbox. If you do not have this functionality already, contact your administrator or call the txttools support team on 0113 2342111.

Once you have the inbox functionality set up on your account, send a message from your phone to your inbox. You may need to include a keyword depending upon how your account has been set up.

Open up the InboxMessage request editor and add the WSSE security header as we did with the Message and Status operation requests. There is only only field in the SOAP body you must populate, the InboxMessageRequest element. Enter a zero here to request all messages from your inbox.

Hit the 'play' button in the top left hand corner of the request window and view the response in the right hand pane.

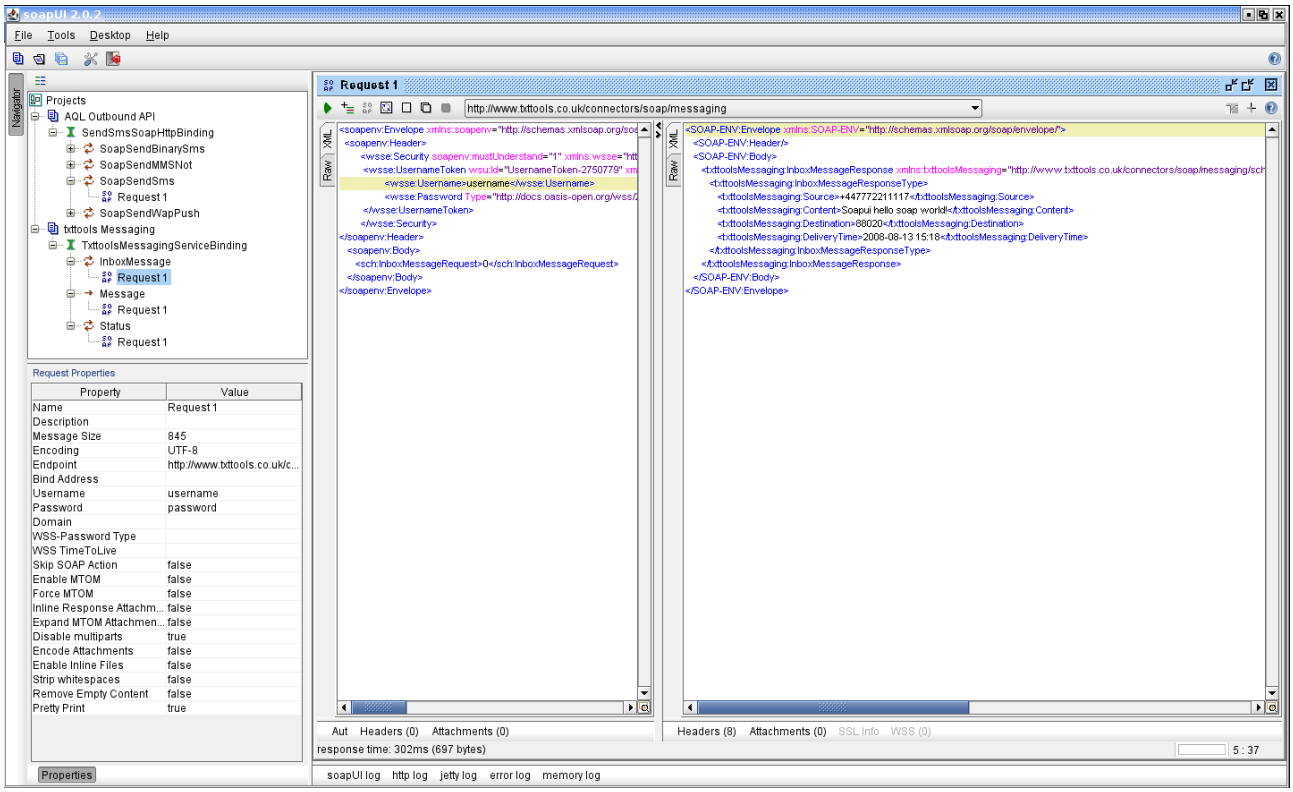

Fig 6 InboxMessage request and response

Once you have retrieved an inbox message it will be marked as 'read' and will not be returned in further invocations of the InboxMessage operation responses. Therefore, you may wish to parse and store the response data.

4.7 Help!

If you require further assistance with this tutorial or any of the information covered in this document please contact the txttools team – 0113 2342111 or email techies@txttools.co.uk