

Exercise 14

Using SOAPUI to interact with a sample SOAP Service

Prior Knowledge

XML

Objectives

Deploy a ready built service in Docker

Call the service using SOAP message

See sample SOAP messages

Software Requirements

These are already installed on the VM.

- Docker
- SOAPUI 5.0.0 or later



1. Firstly, you can make sure all previous servers (tomcat, purchase, node, mitmdump, etc) are closed down as we don't need them for this exercise.
2. We are going to use Docker to run a SOAP service. In this lab, we are going to take a WSDL/SOAP payment service that is loosely modeled on a real SOAP API.
(Barclaycard SmartPay
<https://www.barclaycard.co.uk/business/accepting-payments/website-payments/web-developer-resources/smartpay#tabbox1>).

3. Use docker to run the SOAP service

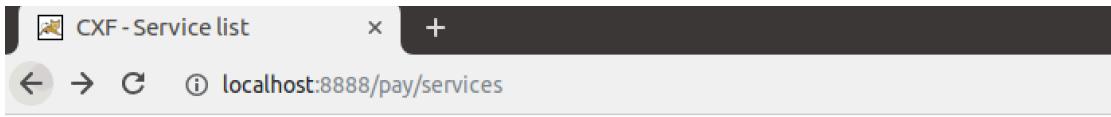
```
docker run -d -p 8888:8080 pizak/pay
```

This offers the docker based service (which is a WAR file running in Tomcat) at port 8888 (mapped from the original 8080 that the Docker image offers).

4. Check to see if its running:

Browse: <http://localhost:8888/pay/services>

5. You should see a SOAP Web Service listed with a link to the WSDL.

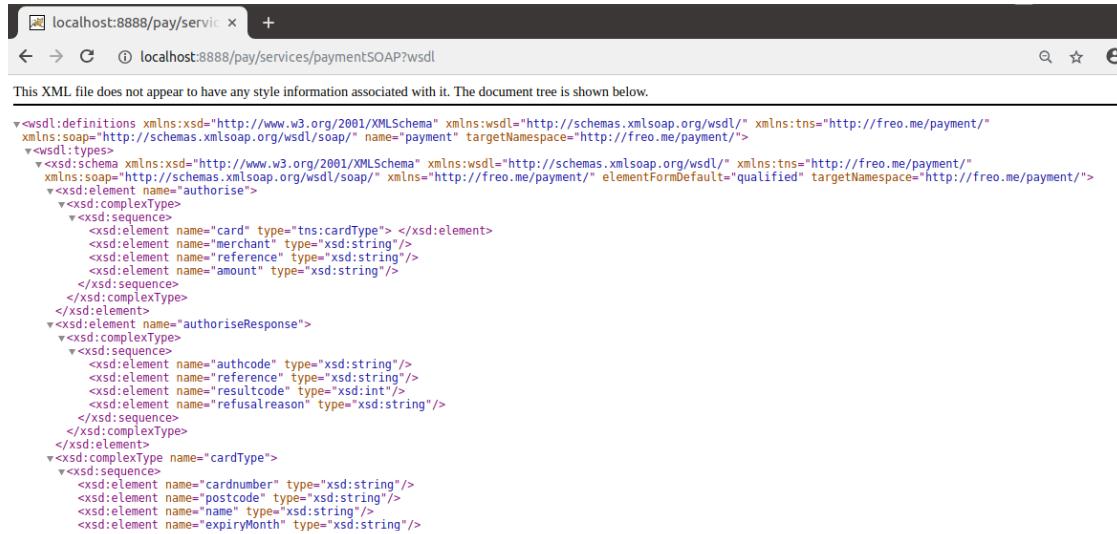


The screenshot shows a web-based interface for managing CXF services. At the top, there's a header bar with a logo, the text 'CXF - Service list', and a close button ('x'). Below the header is a toolbar with navigation icons: back, forward, and a refresh symbol. The main content area is titled 'Available SOAP services:' and contains a single table row. The table has two columns: the first column lists the service name 'payment' and its methods 'authorise' and 'ping'; the second column provides the 'Endpoint address' (http://localhost:8888/pay/services/paymentSOAP), the 'WSDL' location ({http://freo.me/payment/}payment), and the 'Target namespace' (http://freo.me/payment/).

Available SOAP services:	
payment <ul style="list-style-type: none">• authorise• ping	Endpoint address: http://localhost:8888/pay/services/paymentSOAP WSDL : {http://freo.me/payment/}payment Target namespace: http://freo.me/payment/

Available RESTful services:

6. Click on this link. You should see a WSDL



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<wsdl:definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://freo.me/payment/">
  <wsdl:types>
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://freo.me/payment/">
      <xsd:element name="payment" targetNamespace="http://freo.me/payment/">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="card" type="tns:cardType"/>
            <xsd:element name="merchant" type="xsd:string"/>
            <xsd:element name="reference" type="xsd:string"/>
            <xsd:element name="amount" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="authorise">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="card" type="tns:cardType"/>
            <xsd:element name="merchant" type="xsd:string"/>
            <xsd:element name="reference" type="xsd:string"/>
            <xsd:element name="amount" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="authoriseResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="authcode" type="xsd:string"/>
            <xsd:element name="reference" type="xsd:string"/>
            <xsd:element name="resultCode" type="xsd:int"/>
            <xsd:element name="refusalReason" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </wsdl:types>
  <wsdl:message name="PaymentMessage">
    <wsdl:part name="parameters" type="tns:payment"/>
  </wsdl:message>
  <wsdl:message name="AuthoriseMessage">
    <wsdl:part name="parameters" type="tns:payment"/>
  </wsdl:message>
  <wsdl:message name="AuthoriseResponseMessage">
    <wsdl:part name="parameters" type="tns:authoriseResponse"/>
  </wsdl:message>
  <wsdl:portType name="PaymentPortType">
    <wsdl:operation name="PaymentOperation">
      <wsdl:input message="PaymentMessage"/>
      <wsdl:output message="AuthoriseMessage"/>
    </wsdl:operation>
    <wsdl:operation name="AuthoriseOperation">
      <wsdl:input message="AuthoriseMessage"/>
      <wsdl:output message="AuthoriseResponseMessage"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="PaymentBinding" type="tns:PaymentPortType">
    <wsdl:operation name="PaymentOperation">
      <wsdl:input message="PaymentMessage"/>
      <wsdl:output message="AuthoriseMessage"/>
    </wsdl:operation>
    <wsdl:operation name="AuthoriseOperation">
      <wsdl:input message="AuthoriseMessage"/>
      <wsdl:output message="AuthoriseResponseMessage"/>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="AuthoriseBinding" type="tns:AuthorisePortType">
    <wsdl:operation name="AuthoriseOperation">
      <wsdl:input message="AuthoriseMessage"/>
      <wsdl:output message="AuthoriseResponseMessage"/>
    </wsdl:operation>
  </wsdl:binding>
</wsdl:definitions>
```

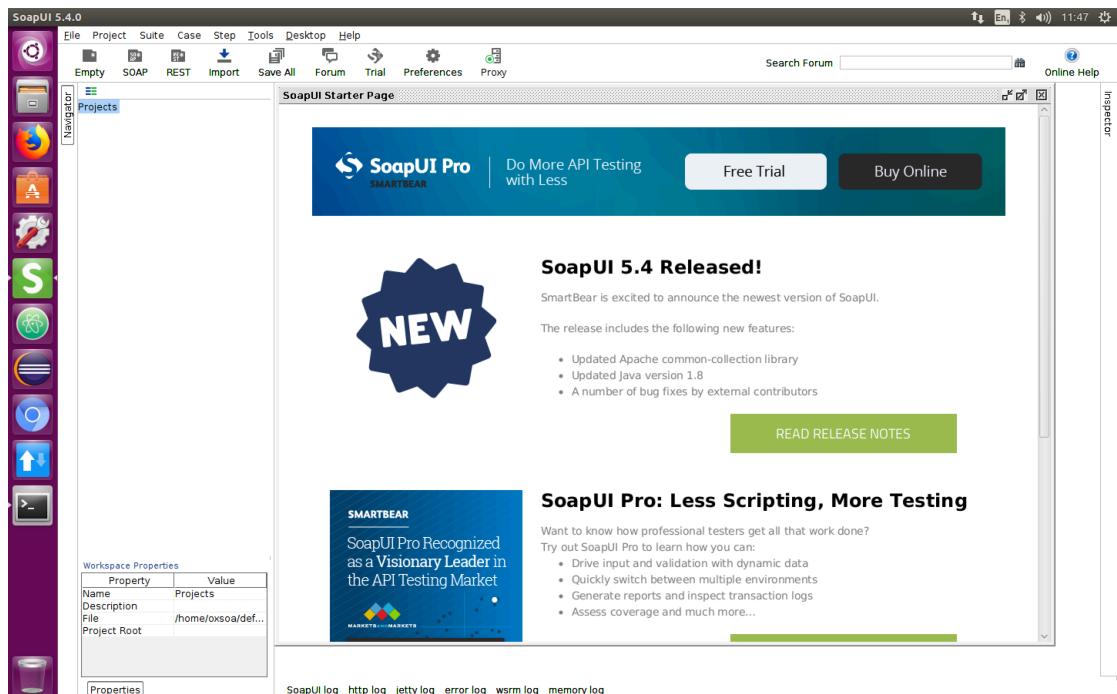
7. Copy the WSDL link

(<http://localhost:8888/pay/services/paymentSOAP?wsdl>) into the clipboard.

8. Now start up SOAPUI from the launcher:

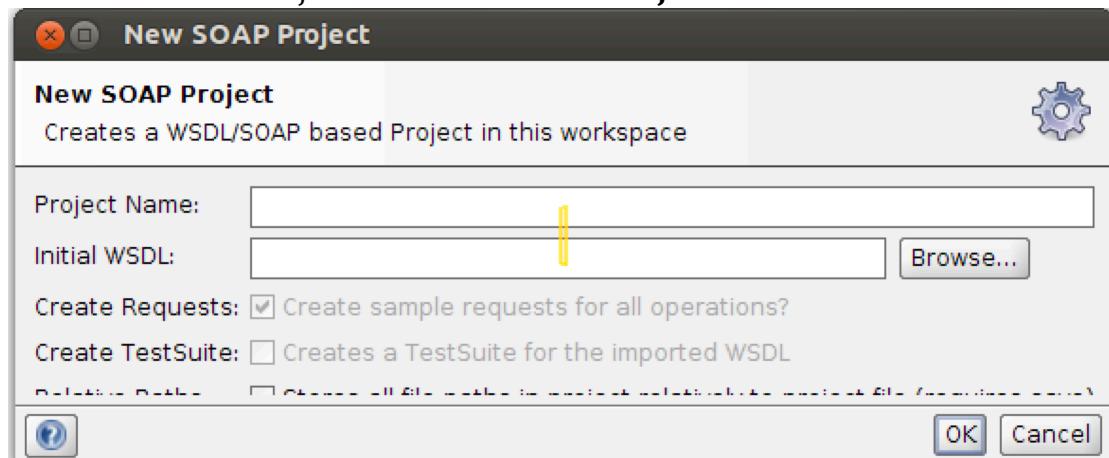


You should see a screen like this:



The screenshot shows the SoapUI 5.4.0 interface. The top menu bar includes File, Project, Suite, Case, Step, Tools, Desktop, Help, and a toolbar with icons for Empty, SOAP, REST, Import, Save All, Forum, Trial, Preferences, and Proxy. A search bar for 'Search Forum' is also present. The left sidebar has a 'Projects' section with a green 'S' icon. The main window displays the 'SoapUI Starter Page' with a banner for 'SoapUI Pro SMARTBEAR' and 'Do More API Testing with Less'. It features a large 'NEW' badge. Below the banner, a section for 'SoapUI 5.4 Released!' announces the new version and lists features: Updated Apache common-collection library, Updated Java version 1.8, and numerous bug fixes by external contributors. A 'READ RELEASE NOTES' button is available. At the bottom, there's a section for 'SoapUI Pro: Less Scripting, More Testing' with a brief description and a list of benefits: Drive input and validation with dynamic data, Quickly switch between multiple environments, Generate reports and inspect transaction logs, and Assess coverage and much more... Log tabs at the bottom include SoapUI log, http log, jetty log, error log, wsrm log, and memory log.

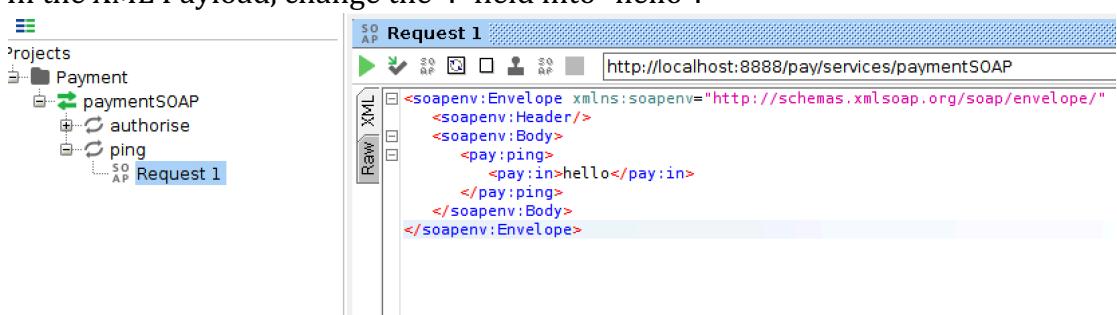
9. Start a new SOAP Project: File->New SOAP Project



10. Type in a name for the project (e.g. Payment)
Paste the WSDL URI into the **Initial WSDL** field
Hit **OK**

11. Now open up the Request editor for one of the operations. You can do this by navigating the service tree in the left window until you see a Request object and click on that.

12. Choose the **ping** request
In the XML Payload, change the '?' field into "hello".



13. Now hit the little green arrow (Run) button.

14. You should see a response from the service.



15. Now do the same for the Authorise method. Fill in some data. e.g.

The screenshot shows the SoapUI interface with a request named "Request 1". The URL is <http://localhost:8888/pay/services/paymentSOAP>. The request XML is as follows:

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:pay="http://fremantle.com/payment/v1.0">
  <soapenv:Header/>
  <soapenv:Body>
    <pay:authorise>
      <pay:card>
        <pay:cardnumber>4555 0000 1111 2222</pay:cardnumber>
        <pay:postcode>PO18 0PG</pay:postcode>
        <pay:name>FREMANTLE</pay:name>
        <pay:expiryMonth>12</pay:expiryMonth>
        <pay:expiryYear>18</pay:expiryYear>
        <pay:cvc>1111</pay:cvc>
      </pay:card>
      <pay:merchant>FREM</pay:merchant>
      <pay:reference>XCBJHJH</pay:reference>
      <pay:amount>10.20</pay:amount>
    </pay:authorise>
  </soapenv:Body>
</soapenv:Envelope>

```

16. You should see a response (either success or failure) like:

The screenshot shows the SoapUI interface with a response named "Request 1". The URL is <http://localhost:8888/pay/services/paymentSOAP>. The response XML is as follows:

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <authoriseResponse xmlns="http://fremantle.com/payment/v1.0">
      <authcode>FAILED</authcode>
      <reference>08a27b8b-4641-414b-e25ef61fa347</reference>
      <resultcode>100</resultcode>
      <refusalReason>INSUFFICIENT FUNDS</refusalReason>
    </authoriseResponse>
  </soap:Body>
</soap:Envelope>

```

17. Take a look at the HTTP Headers and the other data SOAPUI gives you.

Header	Value
Transfer-Encoding	chunked
#status#	HTTP/1.1 200
Server	Apache-Coyote/1.1
Date	Thu, 06 Dec 2018 15:19:04 GMT
Content-Type	text/xml;charset=UTF-8

Headers (5) Attachments (0) SSL Info WSS (0) JMS (0)

response time: 19ms (343 bytes) 2 : 3

```

Thu Dec 06 15:19:04 GMT 2018:DEBUG:><"<soapenv:body>[\n]"
Thu Dec 06 15:19:04 GMT 2018:DEBUG:><"</soapenv:Envelope>">
Thu Dec 06 15:19:04 GMT 2018:DEBUG:<< "HTTP/1.1 200 [\r][\n]"
Thu Dec 06 15:19:04 GMT 2018:DEBUG:<< "Server: Apache-Coyote/1.1[\r][\n]"
Thu Dec 06 15:19:04 GMT 2018:DEBUG:<< "Content-Type: text/xml;charset=UTF-8[\r][\n]"
Thu Dec 06 15:19:04 GMT 2018:DEBUG:<< "Transfer-Encoding: chunked[\r][\n]"
Thu Dec 06 15:19:04 GMT 2018:DEBUG:<< "Date: Thu, 06 Dec 2018 15:19:04 GMT[\r][\n]"
Thu Dec 06 15:19:04 GMT 2018:DEBUG:<< "[\r][\n]"
Thu Dec 06 15:19:04 GMT 2018:DEBUG:<< "157[\r][\n]"
Thu Dec 06 15:19:04 GMT 2018:DEBUG:<< "<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"><soap:Body><authoris"
Thu Dec 06 15:19:04 GMT 2018:DEBUG:<< "[\r][\n]"
Thu Dec 06 15:19:04 GMT 2018:DEBUG:<< "0[\r][\n]"
Thu Dec 06 15:19:04 GMT 2018:DEBUG:<< "[\r][\n]"

```

SoapUI log http log jetty log error log wsrm log memory log

18. That's all.

19. Extension: get the interaction to go via MITMDUMP.