

iDEAL SubID Management – SoapUI client API

Table of contents

1	Document Information	3
1	Document Overview	4
1.1	Version Control	4
2	API Client	5
2.1	General	5
2.1.1	Security	5
2.2	SoapUI	5
2.2.1	Download	5
2.2.2	Setup	5
2.2.3	Proxy setup (optional)	5
2.2.4	Sample project setup	6
2.2.5	Security setup	8
2.2.6	Upload a certificate to the iDEAL application	14
2.2.7	Sample request execution	15
2.2.8	Get fingerprint of the signing key from the response	18
2.2.9	Authentication errors	19

1 Document Information

Title	iDEAL SubID Management – SoapUI client API
Version	4.0
Date	24.01.2023
Author	Christiane Vahle
Document Type	Description of the iDEAL SubID Management API
Classification	public

1 Document Overview

1.1 Version Control

This document is updated continuously. Major modifications on content or size will lead to new release numbers, whereas textual revisions are reflected as new level numbers. The following list shows the document's history.

Version	Date	Author	Reason of modification
0.1	12.10.2016	Paquet Olivier	Initial version
0.2	17.10.2016	Christiane Vahle / Olivier Paquet	Review
1.0	19.12.2017	Christiane Vahle	Final document
2.0	22.05.2019	Christiane Vahle	Updated with new webservice.cer
3.0	07.12.2022	Christiane Vahle	Review
4.0	24.01.2023	Christiane Vahle	Adapted screenshots

2 API Client

2.1 General

This document describes how to test the iDEAL SubID Mgmt API. It covers the tooling, security configuration and sample requests in order to test the API.

Only merchants which are able to manage their SubIDs in iDEAL MAD can use this API.

2.1.1 Security

The iDEAL SubID Mgmt API is secured by XML signatures that must be uploaded in the MSP GUI web application (see 2.2.6).

2.2 SoapUI

This document describes how to call the API using the free webservice tool SoapUI.

It allows to send requests to a SOAP webservice, applying all security requirements needed by this API.

2.2.1 Download

SoapUI can be downloaded here: <https://www.soapui.org/downloads/soapui.html>
Please choose "SoapUI OpenSource".

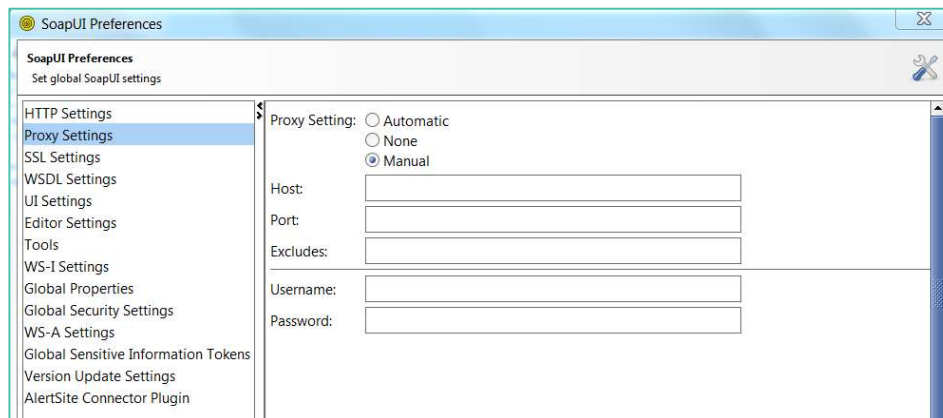
2.2.2 Setup

Install SoapUI and follow the instructions.

2.2.3 Proxy setup (optional)

In case you have to specify a proxy, due to company restrictions, the following steps show how to configure it.

Go to File -> Preferences, a new window is opened and you can choose on the left menu the Proxy Settings tab.



2.2.4 Sample project setup

- WSDL and XSD files

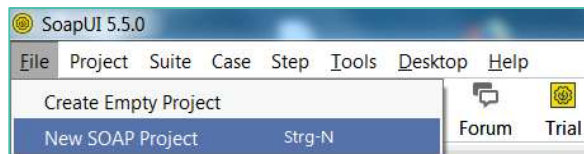


subid-mgmt-service-v1.wsdl

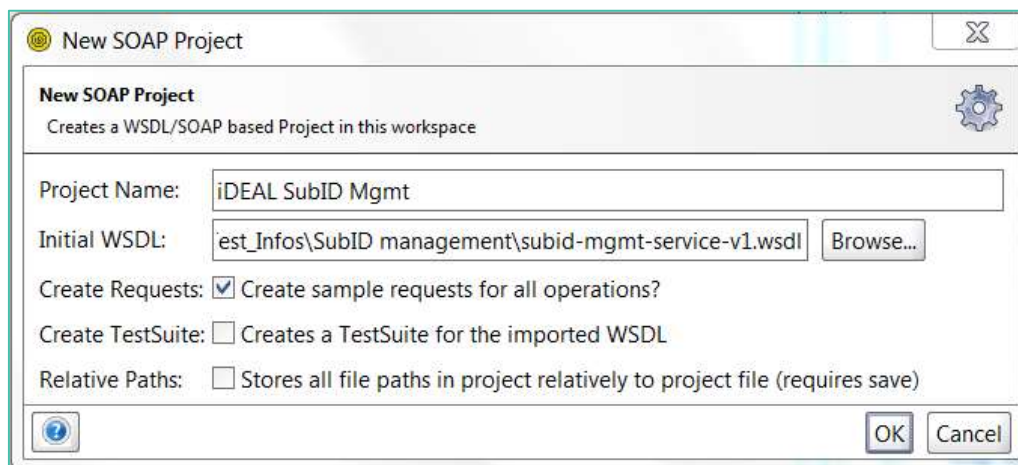


subid-mgmt-service-v1.xsd

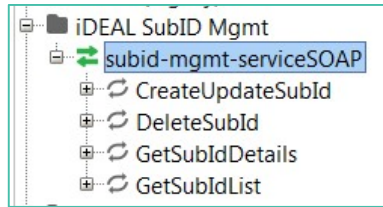
- Start SoapUI and create a new project:



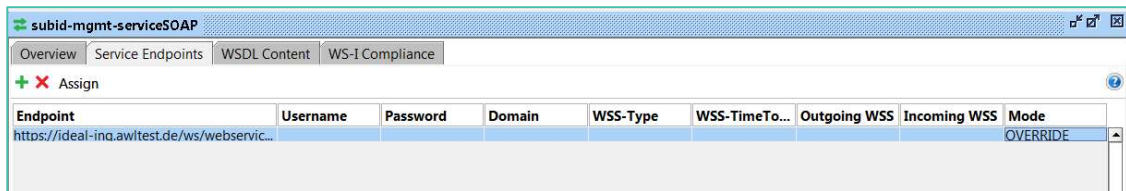
- A new window is opened.
- Please specify a project name, browse to the provided WSDL file (subid-mgmt-service-v1.wsdl), select "Create Requests" and click OK:



- A new project is generated:



- Double click the menu entry “subid-mgmt-serviceSOAP”
- Navigate to the Service Endpoints tab
- Specify in the Endpoint field the environment URL you want to test against(see table below) - Please note: In this example the quality environment URL is used.
- Change the Mode to OVERRIDE.



- SOAP addresses for iDEAL production system: per tenant:

<https://ecommerce.abnamro.nl/msp/services/LegacyiDEALSubscriptionInterfaceIntegration?wsdl>

<https://myideal.db.com/msp/services/LegacyiDEALSubscriptionInterfaceIntegration?wsdl>

<https://ideal.rabobank.nl/msp/services/LegacyiDEALSubscriptionInterfaceIntegration?wsdl>

- SOAP addresses for iDEAL test/integration system per tenant:

<https://ecommerce.abnamro.nl/msp/services/LegacyiDEALSubscriptionInterfaceIntegration?wsdl>

<https://myideal.db.com/msp/services/LegacyiDEALSubscriptionInterfaceIntegration?wsdl>

<https://ideal.rabobank.nl/msp/services/LegacyiDEALSubscriptionInterfaceIntegration?wsdl>

- Open one of the generated requests in order to check that the configured URL is used correctly



- If you see, after opening a request, the correct url on the top address bar, the project setup it done.
- Please continue now with the security setup (see 2.2.5).

2.2.5 Security setup

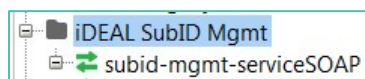
2.2.5.1 Signature configuration

In order to send requests to the API the security must be configured correctly.

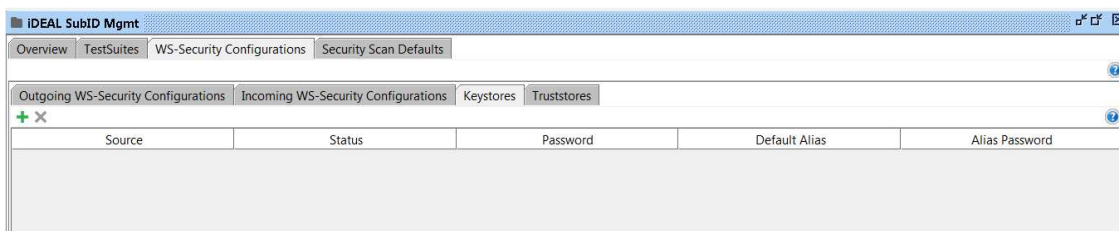
Merchants have to upload the corresponding public key in the MSP web application (see 2.2.6)

The private key should be located in a keystore.

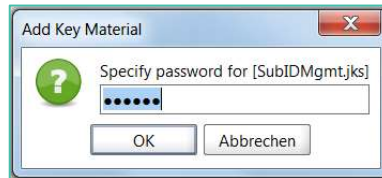
1. Double click the main menu entry



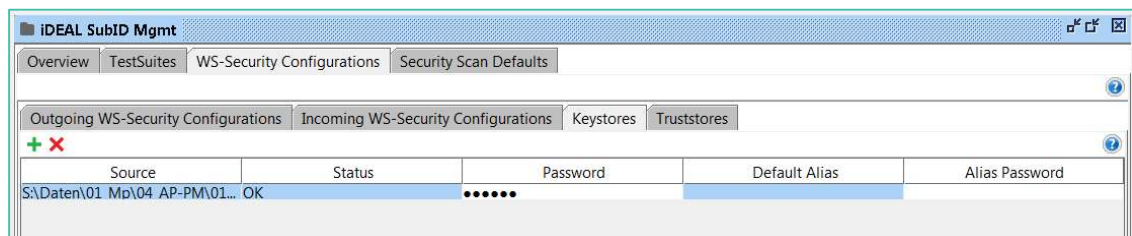
2. Navigate to the “WS-Security Configuration” tab -> Keystores



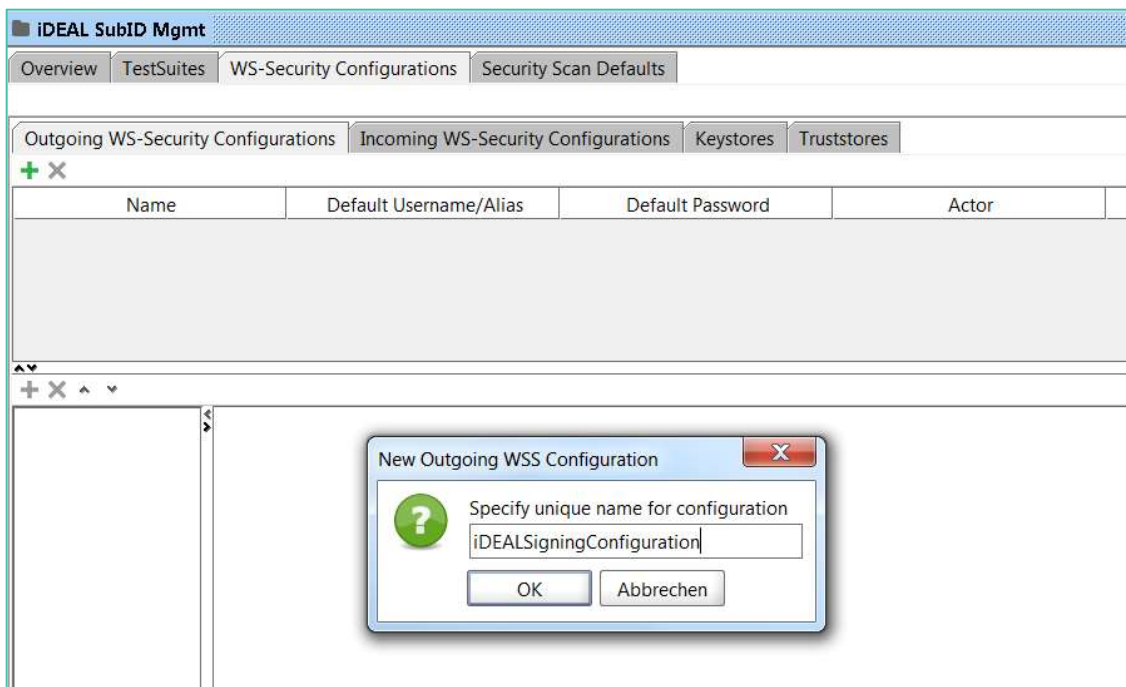
3. Add your keystore clicking on the green plus sign
4. After selecting the keystore file you are asked to enter the password



5. If the keystore was successfully imported, the status should be OK.

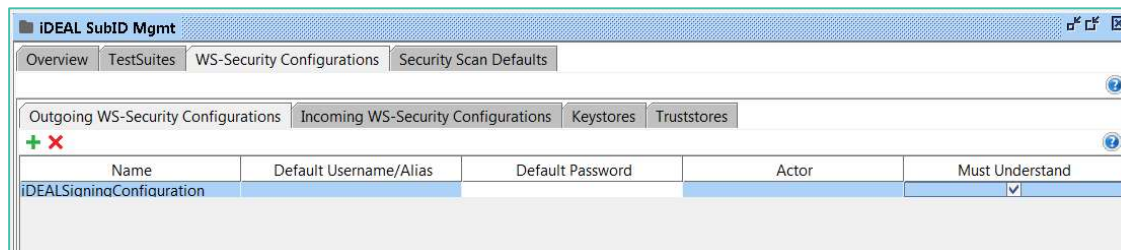


6. Now navigate to the “**Outgoing WS-Security Configurations**” tab and add a new configuration by clicking on the green plus sign

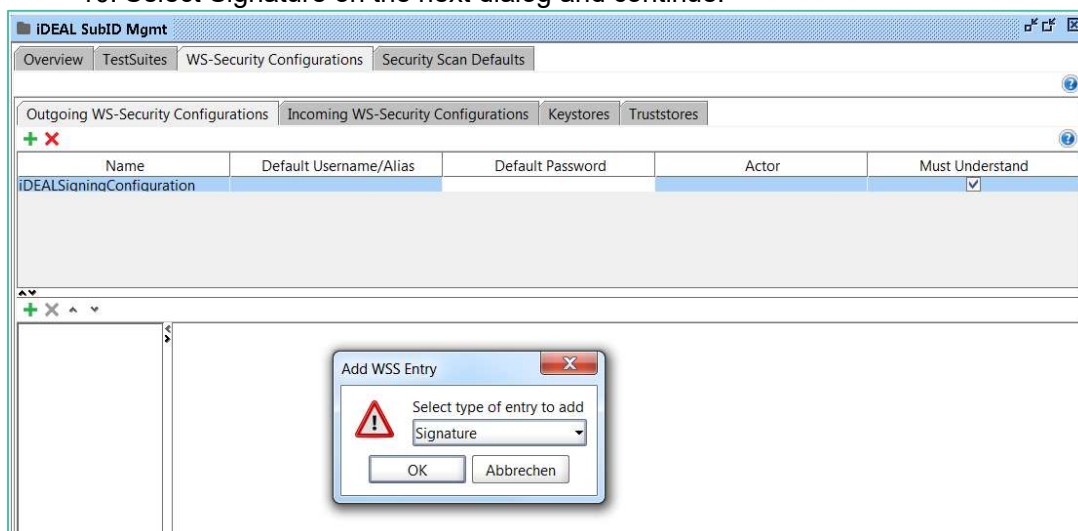


7. Specify a name. (e.g. iDEALSigningConfiguration)

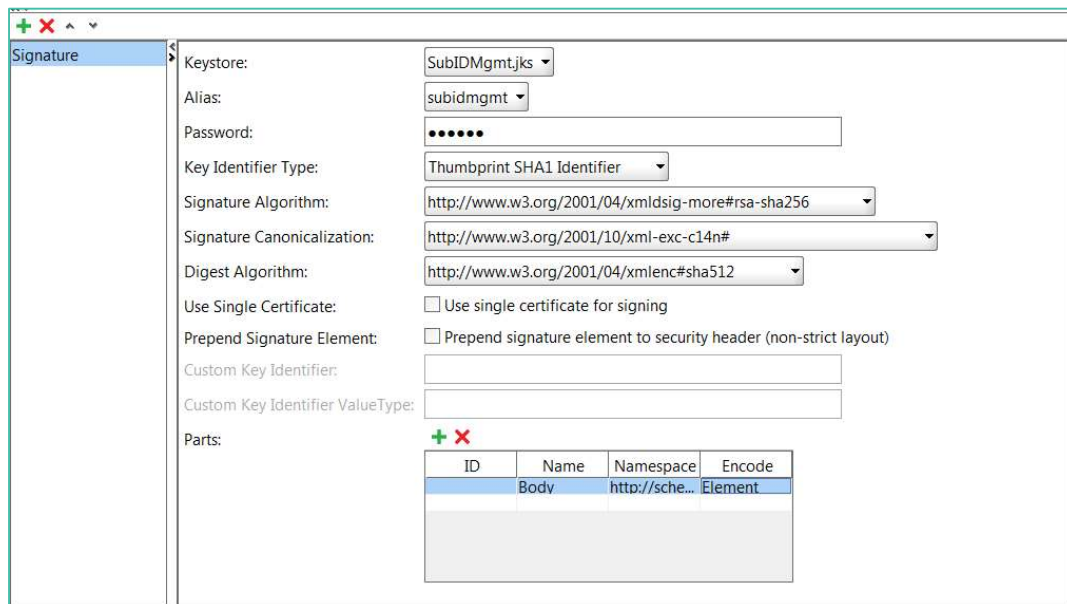
8. Afterwards the configuration is added, please check the checkbox on the column “Must Understand”.



9. Add a new WSS Entry by clicking on the green plus sign at the bottom
10. Select Signature on the next dialog and continue.



11. Fill all required fields as shown in the screenshot



Signature

Keystore: SubIDMgmt.jks

Alias: subidmgmt

Password:

Key Identifier Type: Thumbprint SHA1 Identifier

Signature Algorithm: http://www.w3.org/2001/04/xmldsig-more#rsa-sha256

Signature Canonicalization: http://www.w3.org/2001/10/xml-exc-c14n#

Digest Algorithm: http://www.w3.org/2001/04/xmlenc#sha512

Use Single Certificate: ☐ Use single certificate for signing

Prepend Signature Element: ☐ Prepend signature element to security header (non-strict layout)

Custom Key Identifier:

Custom Key Identifier ValueType:

Parts:

ID	Name	Namespace	Encode
	Body	http://sche...	Element

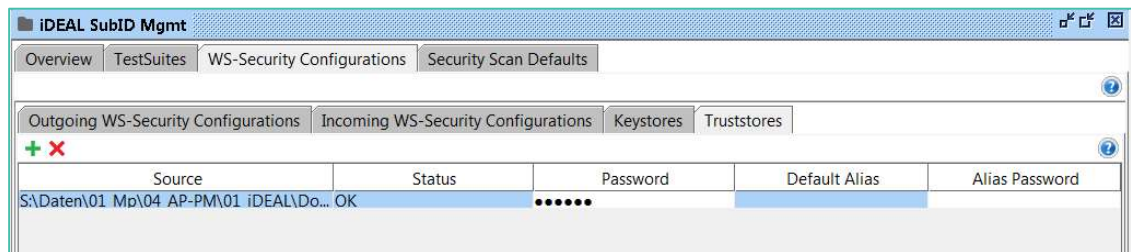
- Select the keystore you imported at step 3
- Select the alias corresponding to the private key entry in the keystore
- Specify the password if the entry is also password protected
- Key Identifier Type: *Thumbprint SHA1 Identifier*
- Signature Algorithm: <http://www.w3.org/2001/04/xmldsig-more#rsa-sha256>
- Signature Canonicalization: <http://www.w3.org/2001/10/xml-exc-c14n#>
- Digest Algorithm: <http://www.w3.org/2001/04/xmlenc#sha256>
- Add a new part in the table at the end
 - Name: Body
 - Namespace: <http://schemas.xmlsoap.org/soap/envelope/>
 - Encode: Element

2.2.5.2 Signature validation

In order to validate the iDEAL signature an incoming webservice configuration has to be done. The certificate I “webservice.cer” and can be downloaded from MSP GUI - Documents.

Steps to follow:

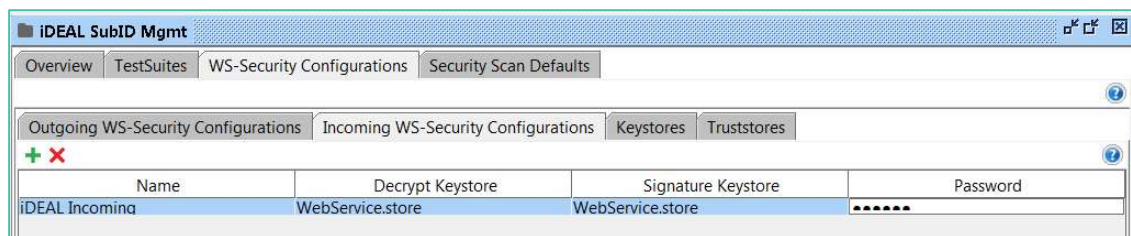
- In SoapUI double click the project → WS-SecurityConfigurations → Truststores



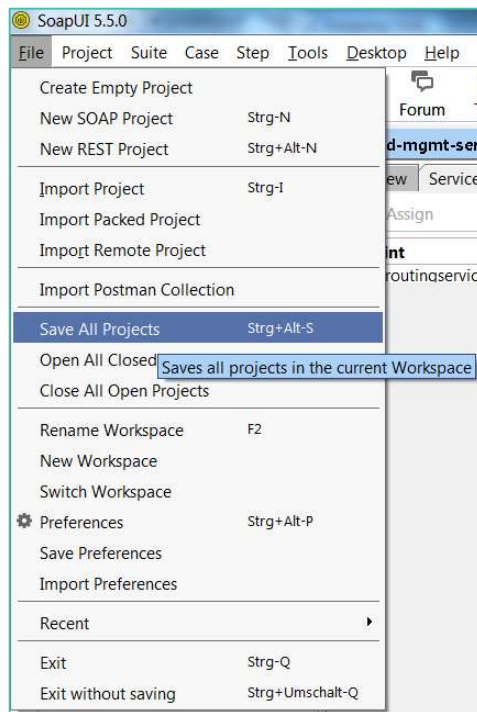
SoapUI expects a truststore, you can simply create one with [Keystore Explorer](#) for example.

When clicking the green plus sign, browse to the truststore which contains the iDEAL web service certificate. A new entry will be displayed with Status OK.

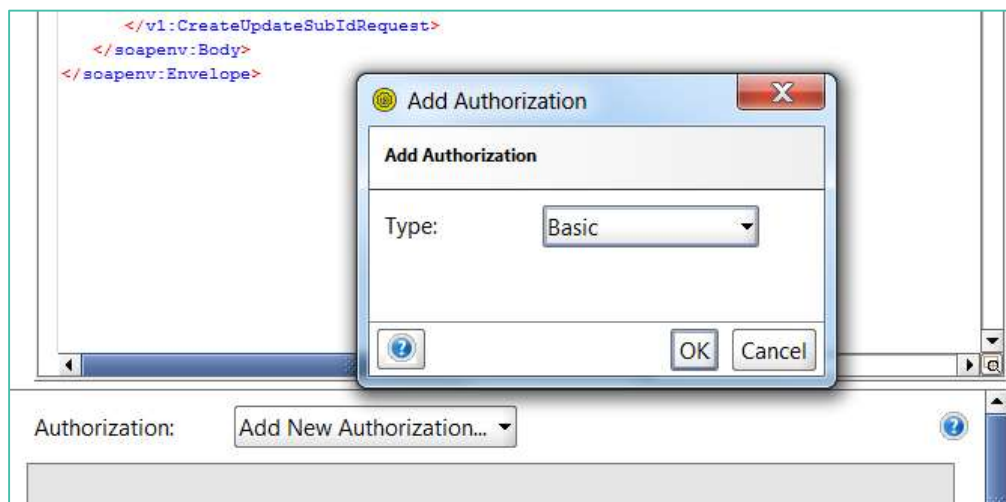
2. Navigate to the “Incoming WS-Security Configurations” tab



- Click on the green plus sign, enter a name for the new configuration eg. iDEAL Incoming and select the truststore for the both columns Decrypt Keystore and Signature Keystore.
- Now save all projects



5. Open the request you want to use the signature validation
6. Open the "Auth tab", add a new Authorization "Basic"



7. Select the "Outgoing WSS" and "Incoming WSS"

Authorization: Basic

Username:

Password:

Domain:

Pre-emptive auth: ☒ Use global preference
☐ Authenticate pre-emptively

Outgoing WSS: iDEALSigningConfiguration

Incoming WSS: iDEAL Incoming

Auth (Basic) Headers (0) Attachments (0) WS-A WS-RM JMS Headers JMS Property (0)

8. Check that the signature validation was successful

The screenshot shows a web browser window with the URL `https://routing-service-rabo.awttest.de/msp/services/LeacyiDEALSubscriptionInterface?WSDL`. The browser displays two panes of raw XML data.

Left Pane (Request):

```
<?xml version='1.0' encoding='UTF-8'>
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/' xmlns:vl='http://www.v3.org/2001/04/complextype'>
  <soap:Header>
    <wsse:Security soap:mustUnderstand='1' xmlns:wsse='http://docs.oasis-open.org/wss/2002/01/wss-schema'>
      <ds:Signature Id='SIG-353a60be-21ea-4bfb-baeb-0b8ab94b092c'>
        <ds:CanonicalizationMethod Algorithm='http://www.w3.org/2001/10/xml-exc16#canonicalization'>
          <ec:InclusiveNamespaces PrefixList='soap' xmlns:ec='http://www.w3.org/2001/10/xml-exc16#canonicalization'>
            <ds:SignatureMethod Algorithm='http://www.w3.org/2001/04/xmldsig-core#rsa-sha1'>
              <ds:Reference URI='#id-86f2a61f-4ecf-46ac-a41f-5801a8f3142'>
                <ds:Transforms>
                  <ds:Transform Algorithm='http://www.w3.org/2001/10/xml-exc16#canonicalization'>
                    <ds:DigestMethod Algorithm='http://www.w3.org/2001/04/xmldsig-core#sha1'>
                      <ds:DigestValue>gHfGK3dXlc+z6wIVqXoatXrlx2pn/K+2VZ</ds:DigestValue>
                    </ds:DigestMethod>
                  </ds:Transform>
                </ds:Transforms>
              </ds:Reference>
            </ds:SignatureMethod>
          </ds:Signature>
        </ds:SignatureMethod>
      </ds:Signature>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <vl:CreateUpdateSubIdRequest>
      <vl:merchantId>002047576</vl:merchantId>
      <vl:subId>111111</vl:subId>
      <vl:tradeName>subid9999</vl:tradeName>
      <vl:URL>www.111111.com</vl:URL>
      <vl:status>ACTIVE</vl:status>
    </vl:CreateUpdateSubIdRequest>
  </soap:Body>
</soap:Envelope>
```

Right Pane (Response):

```
<?xml version='1.0' encoding='UTF-8'>
<wsse:Security soap:mustUnderstand='1' xmlns:wsse='http://docs.oasis-open.org/wss/2002/01/wss-schema'>
  <ds:Signature Id='SIG-353a60be-21ea-4bfb-baeb-0b8ab94b092c'>
    <ds:CanonicalizationMethod Algorithm='http://www.w3.org/2001/10/xml-exc16#canonicalization'>
      <ec:InclusiveNamespaces PrefixList='soap' xmlns:ec='http://www.w3.org/2001/10/xml-exc16#canonicalization'>
        <ds:SignatureMethod Algorithm='http://www.w3.org/2001/04/xmldsig-core#rsa-sha1'>
          <ds:Reference URI='#id-86f2a61f-4ecf-46ac-a41f-5801a8f3142'>
            <ds:Transforms>
              <ds:Transform Algorithm='http://www.w3.org/2001/10/xml-exc16#canonicalization'>
                <ds:DigestMethod Algorithm='http://www.w3.org/2001/04/xmldsig-core#sha1'>
                  <ds:DigestValue>gHfGK3dXlc+z6wIVqXoatXrlx2pn/K+2VZ</ds:DigestValue>
                </ds:DigestMethod>
              </ds:Transform>
            </ds:Transforms>
          </ds:Reference>
        </ds:SignatureMethod>
      </ds:Signature>
    </ds:SignatureMethod>
  </ds:Signature>
</wsse:Security>
</soap:Header>
<soap:Body>
  <vl:CreateUpdateSubIdResponse xmlns='http://api.worldline.com/subid'>
    <result>
      <resultCode>SUCCESS</resultCode>
    </result>
    <resultOperation>UPDATE</resultOperation>
    <createdSubId>
      <merchantId>002047576</merchantId>
      <subId>111111</subId>
      <tradeName>subid9999</tradeName>
      <URL>www.111111.com</URL>
      <status>ACTIVE</status>
    </createdSubId>
  </vl:CreateUpdateSubIdResponse>
</soap:Body>
</wsse:Security>
```

After getting a response, open the WSS tab on the bottom. Now you should see the validation processing results.

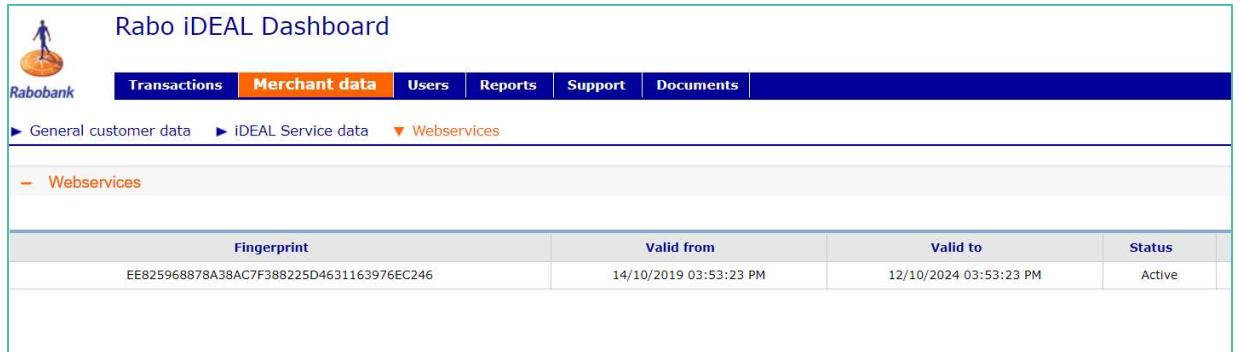
The validation failed if you see the following text:
org.apache.ws.security.WSSecurityException: The signature or decryption was invalid

Now you can continue with the sample request execution (see 2.2.7).

2.2.6 Upload a certificate to the iDEAL application

There are two ways to manage the certificates to be used by web services.

Logon to MSP GUI -> Merchant -> Webservices



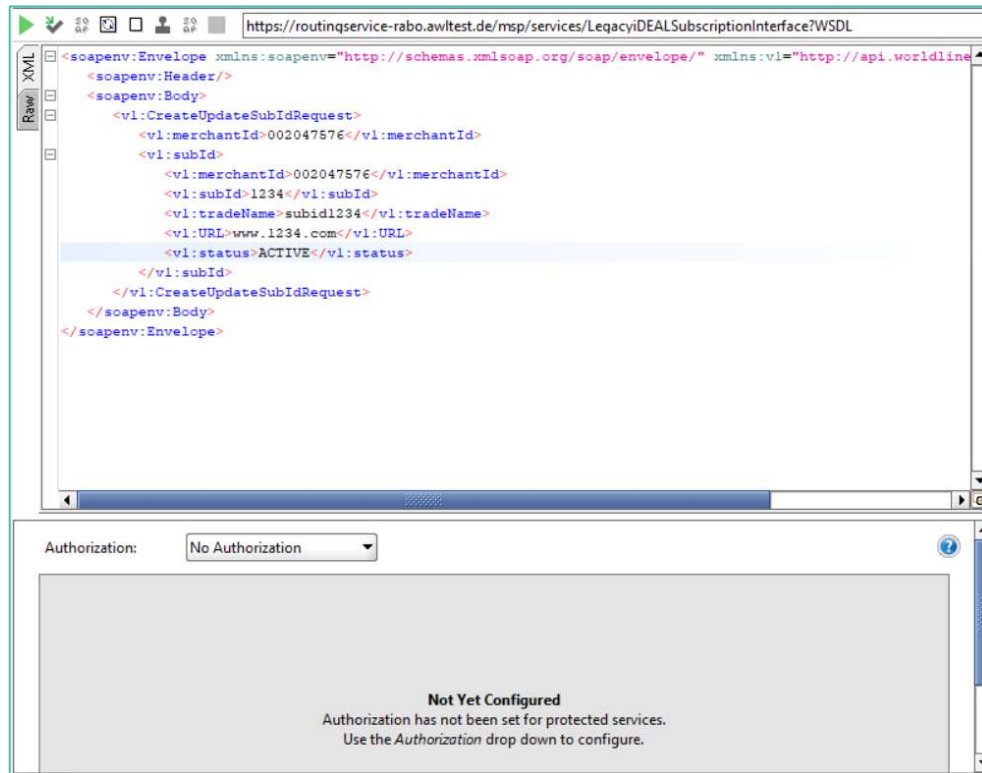
The screenshot shows the 'Rabo iDEAL Dashboard' with a navigation bar containing 'Transactions', 'Merchant data', 'Users', 'Reports', 'Support', and 'Documents'. Below the navigation bar, there are tabs for 'General customer data', 'IDEAL Service data', and 'Webservices'. The 'Webservices' tab is active, showing a table with the following data:

Fingerprint	Valid from	Valid to	Status
EE825968878A38AC7F388225D4631163976EC246	14/10/2019 03:53:23 PM	12/10/2024 03:53:23 PM	Active

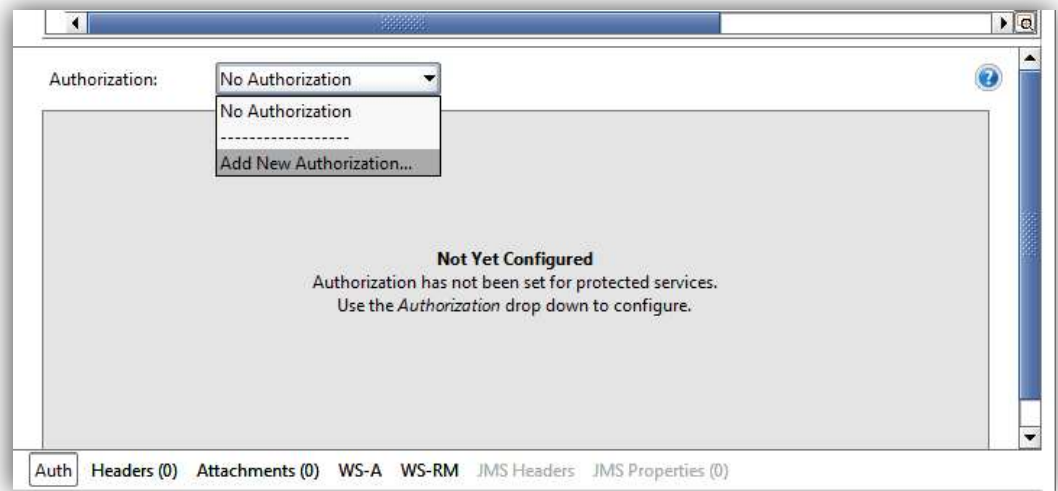
Upload your signing certificate there.

2.2.7 Sample request execution

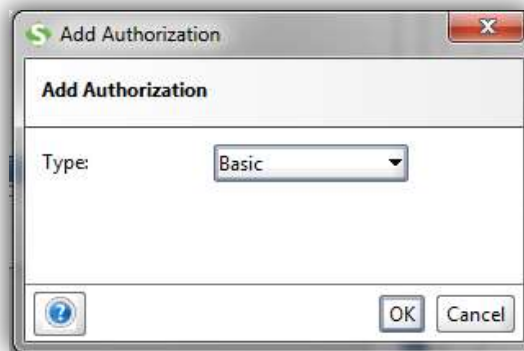
In order to send a signed request please open a generated request.



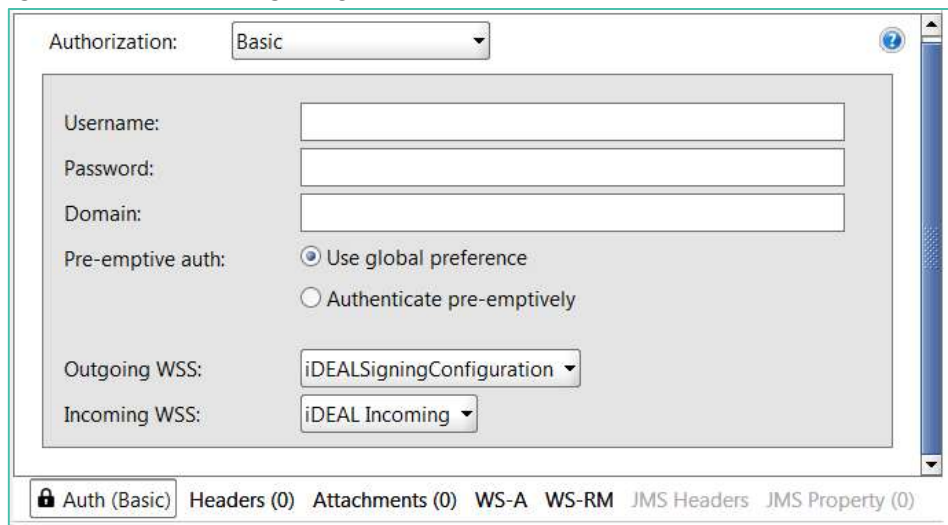
Select the "Auth" menu, as shown in the screenshot.



Select now “Add New Authorization”



Choose “Basic” and Click OK.



Go to “Outgoing WSS” and select the Outgoing configuration you created 2.2.5.2.

Go to “Incoming WSS” and select the Incoming configuration you created in 2.2.5.2.

Now you can adapt the request and set your merchant id:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:v1="http://api.worldline.com/subid/mgmt/types/v1">
  <soapenv:Header/>
  <soapenv:Body>
    <v1:GetSubIdListRequest>
      <v1:merchantId>YOUR_MERCHANT_ID</v1:merchantId>
    </v1:GetSubIdListRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

In case of success you get a response like:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <wsse:Security soap:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
      <ds:Signature Id="SIG-4a5af412-326e-4896-a8e1-127f031a4803" xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
            <ec:InclusiveNamespaces PrefixList="soap" xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" />
          </ds:CanonicalizationMethod>
          <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
          <ds:Reference URI="#id-1862ca7-380c-44dc-8b12-dbb3e8a3ab19">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                <ec:InclusiveNamespaces PrefixList="" xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" />
              </ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#sha256"/>
            <ds:DigestValue>vtHmVuKpWQ2YNYDX5enaS3znkFbXlzo9tYxgYIBC1vs=</ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>dzVqUeuzEqTyI3wAcIC+xJ9ty1i67xzlCbERXdl6ycAaK9w2MC9gawJbGipX+2UHWWhASN8SSxTQSZDDoQA6i1B8pnoWzFI6THJOls
n+22hMg0PAZOTv8Ka1h+W+88BLKChsNSBz4l/8PHfz56ychWvPkilvWVY71fBUcM8miYT+uTMXvj3m+K/77xqorcUt8r347UPoHuYAqjBYssNIAF5R/gEE
7vAMOCiRWuJ2r9+Zr4m/tkoZSGDurrHfyykzZU0tiM4n9Vb3FAVgDoB0VuzV2VVUxZx8nH9Kfa8BHgXmv+T42fkd2xq3PulRUTq1RpCF7ZX55wjl9Oj34
Pg==</ds:SignatureValue>
        <ds:KeyInfo Id="KI-fbee421e-5510-4cd9-860d-99d8a85b78d9">
          <wsse:SecurityTokenReference wsu:Id="STR-3d56eafc-576c-4fef-89aa-a5fbc07a3f67">
            <wsse:KeyIdentifier EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
Value="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.1#ThumbprintSHA1">sNMXCK35URl5heVI3ChlzbM6GAE=</wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </SOAP-ENV:Header>
  <soap:Body wsu:Id="id-1862ca7-380c-44dc-8b12-dbb3e8a3ab19" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd">
    <GetSubIdListResponse xmlns="http://api.worldline.com/subid/mgmt/types/v1">
```

```

<result>
  <resultCode>SUCCESS</resultCode>
</result>
<subIdListSize>1</subIdListSize>
<subIdList>
  <subIdRecord>
    <merchantId>YOUR_MERCHANTID</merchantId>
    <subId>99</subId>
    <tradeName>Signup1_subId</tradeName>
    <URL>www.Signup1.com</URL>
    <status>ACTIVE</status>
  </subIdRecord>
</subIdList>
</GetSubIdListResponse>
</soap:Body>
</soap:Envelope>

```

2.2.8 Get fingerprint of the signing key from the response

If you want to get the fingerprint of the key that has been used to sign the request or the response, you have to search the following section in the SOAP request / response:

```

<ds:KeyInfo Id="KI-6C60F19537204AFFA514737679313352">
  <wsse:SecurityTokenReference wsu:Id="STR-6C60F19537204AFFA514737679313363">
    <wsse:KeyIdentifier EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-
security-1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-
1.1#ThumbprintSHA1">YBSYFYI7boIQo0DeYor292MDEYM</wsse:KeyIdentifier>
  </wsse:SecurityTokenReference>
</ds:KeyInfo>

```

You can take the value defined for the “wsse:KeyIdentifier” tag.

This is the base64 encoded fingerprint.

Decode it and format this data to hexadecimal.

You can also use the tool “FingerprintDecoder.jar”, it requires to have java installed on your pc.

To start, just double click it, a windows opens, and paste the value of the “wsse:KeyIdentifier” tag into the displayed window.

2.2.9 Authentication errors

In case of authentication errors, you should get an error id back, so that the support can search for the reason.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:Client</faultcode>
      <faultstring>Authentication failed[referenceId:ID]</faultstring>
      <detail/>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

This could have multiple reasons, no certificate found, ...