SECURE TRANSACTION APPROVAL TO BE PSD2 COMPLIANT

Secure Transaction Approval to be PSD2 compliant

Signature Confirmation

1. Overview

1.1 The Problem

RCDevs' PSD2-READY solution has all the tools you need to stay compliant.

Online banking and most business processes require controls and approvals. It could be for a large financial transaction, a simple work expense reimbursement or procurement approval. In banking, these have been traditionally managed using One-Time Passwords (OTP) or PIN codes and in business applications with simple username+password authentication. These mechanisms are now insufficient to meet today's regulatory requirements, security and usability expectations.

1.2 The revised Payment Services Directive (PSD2)

PSD2 introduces new regulation to banks operating in Europe: Strong Customer Authentication (SCA) and Dynamic Linking. Strong customer authentication entails an authentication based on two or more elements categorized as:

They must be independent of each other, meaning that acquiring one factor does not compromise the other.

1.3 The Solution

RCDevs OpenOTP can help you meet the PSD2 requirements while at the same time making your business processes more efficient and user-friendly.

Key Features

- Compliant with PSD2 SCA and Dynamic linking requirements
- One solution for multi-factor authentication and secure transaction approval
- Online communication using end to end encryption
- Offline communication using encrypted QR code when the phone does not have a network connect

1.4 The Challenge

Dynamic Linking Integrations

Dynamic Linking requirement poses a greater challenge for currently deployed solutions. It requires that the credentials used to approve a payment can only be used to approve the specific payment, that the user is made aware of the amount and payee and that the credentials cannot be used to approve any other or modified payment. Traditional OTP tokens cannot meet this requirement, as the OTP is in no way linked to the transaction details being approved, leaving the user and your business exposed

to “man in the middle” attacks. RCDevs OpenOTP allows you to meet these requirements and prevent fraudulent transactions. When approving a transaction using OpenOTP, the user can review the transaction details on their smartphone, including attached documents, and approve or reject the transaction directly. All information is protected with end to end encryption and any change to the transaction will invalidate the approval.

Easy to use API

OpenOTP secure transaction approval solution can be easily integrated into existing applications with a flexible and easy to use API. OpenOTP provides easy to use API (REST+SOAP) for integrating into your existing business application, minimizing development effort, deployment time and disruption to existing processes.

1.5 Who is affected?

Any business that processes payments that are completed in the European Union, even if only one part of the transaction is in the European Union.

Transactions that are covered

Strong Customer Authentication is required in the European Union and European Economic Area for:

- Online access to payment accounts
- Initiating electronic transactions
- Anything did remotely which presents a risk of payment fraud
- Provisioning of information through a service provider

1.6 Technical Points

- The data sent by the server in the data parameter in transaction requests are signed and encrypted in AES128.
- If a document is attached to a transaction request, then the document is also signed.
- The generated signature is an OATH OCRA signature.
- If a form is attached to a transaction, then the form is signed.

2. Requirements

- First, you must have a WebADM/OpenOTP server installed and configured. Please refer to the installation documentation for instructions on this.
- You have a valid license for OpenOTP and the Secure Transaction Approval feature.
- For online approval transaction, you need to configure a Push login infrastructure. Refer to push documentation for more details.
- OpenOTP Token application usage is mandatory to sign a transaction and an OATH Token must be registered on each user accounts who need to sign transactions. Please see these documents for more information about OpenOTP Token and the Token Registration.

3. Transaction server configuration

Some Confirmation options can be configured under OpenOTP configuration. `WebADM Admin GUI` > `Applications` tab > `MFA Authentication Server` > `CONFIGURE` > `Secure Confirmation` section.

In this example all the options are checked, which means that all selected data will be sent back in the transaction response and stored on the server.

- Address : This option stores the location of the end user when a transaction is approved or refused.
- Localtime : This option stores the local time of the end user when a trasaction is approved or refused.
- The handwritten signature and paraph must be configured by the end-user himself on the OpenOTP token application. Open the `OpenOTP Token application` > `Settings` > `Handwritten signature` and add your signature and paraph. If not set on the phone and required by the server, then the end-user will be prompted on the first transaction to configure them. The signature and paraph are sent to the server only if the end user approve the transaction.
- Comment : Allow the end user to give a comment through the mobile application if he refuses the transaction.

4. Online Signature with Push

4.1 Synchronous Confirmation

4.1.1 Workflow

4.1.2 Python Example

```
#!/usr/bin/python3

import requests

url = 'http://my_webadm_server:8080/openotp/json/' # In Production, HTTPS on 8443 port
must be used.
s = requests.Session()

response = s.post(url+'openotpNormalConfirm', data = {
 "username": "John",
 "client": "My_Web_Banking", # Can be used to match a client policy
 "source": "1.2.3.4", # User IP
 "async": False,
 "data": "<html><b>Sample Transaction</b><br><br>Account: <font
color=red>Testing</font><br>Amount: <font color=red>x Euros</font><br></html>",
}).json()

print(response['message'])
```

4.2 Asynchronous Confirmation

4.2.1 Workflow

4.2.2 Python Example

```
#!/usr/bin/python3
import requests, base64, time, tempfile, PIL, multiprocessing
from PIL import Image

url = 'http://my_webadm_server:8080/openotp/json/' # In Production, HTTPS on 8443 port
must be used.

s = requests.Session()

r = s.post(url+'openotpConfirmQRCode', data = {
 "username": "John",
 "client": "My_Web_Banking", # Can be used to match a client policy
 "source": "1.2.3.4", # User IP
 "async": True,
 "data": "<html><b>Sample Transaction</b><br><br>Account: <font
color=red>Testing</font><br>Amount: <font color=red>x Euros</font><br></html>",
}).json()

f = tempfile.TemporaryFile ()
f.write(base64.b64decode(r['qrImage']))
```

```
PIL.image = Image.open(f)
PIL.image.show()

def push_response ():
 while True:
  time.sleep(1)
  response = (s.post(url+'openotpCheckConfirm', data = {
   "session": r['session'],
  }).json())
  if response['code'] != 2:
   break
 return response

def ask_otp ():
 otp = input("Enter the otp: ")
 return  s.post(url+'openotpOfflineConfirm', data = {
  "session": r['session'],
  "response": otp,
 }).json()

pool1 = multiprocessing.pool.ThreadPool(processes=1)
pool2 = multiprocessing.pool.ThreadPool(processes=1)

push = pool1.apply_async(push_response)
otp  = pool2.apply_async(ask_otp)

while True:
 try:
  response = push.get(timeout=1)
  if response:
   break;
 except :
  pass
 try:
  response = otp.get(timeout=1)
  if response:
   break;
 except :
  pass

print(response['message'])
```

5. Offline Signature with QRcode

5.1 Synchronous Confirmation

5.1.1 Workflow

5.1.2 Python Example

```
#!/usr/bin/python3
import requests, base64, tempfile, PIL

url = 'http://my_webadm_server:8080/openotp/json/' # In Production, HTTPS on 8443 port
must be used.

s = requests.Session()

r = s.post(url+'openotpConfirmQRCode', data = {
 "username": "John",
 "client": "My_Web_Banking", # Can be used to match a client policy
 "source": "1.2.3.4", # User IP
 "async": False,
 "data": "<html><b>Sample Transaction</b><br><br>Account: <font
color=red>Testing</font><br>Amount: <font color=red>x Euros</font><br></html>",
}).json()

f = tempfile.TemporaryFile ()
f.write(base64.b64decode(r['qrImage']))
PIL.image = Image.open(f)
PIL.image.show()

otp = input("Enter the otp (empty string will check the confirmation): ")
if len(otp) >0:
 response = s.post(url+'openotpOfflineConfirm', data = {
  "session": r['session'],
  "response": otp,
 }).json()
else:
 response = (s.post(url+'openotpCheckConfirm', data = {
  "session": r['session'],
 }).json())

print(response['message'])
```

5.2 Asynchronous Confirmation

5.2.1 Workflow

5.2.2 Python Example

```
#!/usr/bin/python3
import requests, base64, time, tempfile, PIL, multiprocessing

url = 'http://my_webadm_server:8080/openotp/json/' # In Production, HTTPS on 8443 port
must be used.

s = requests.Session()

r = s.post(url+'openotpConfirmQRCode', data = {
 "username": "John",
 "client": "My_Web_Banking", # Can be used to match a client policy
 "source": "1.2.3.4", # User IP
 "async": True,
 "data": "<html><b>Sample Transaction</b><br><br>Account: <font
color=red>Testing</font><br>Amount: <font color=red>x Euros</font><br></html>",
}).json()

f = tempfile.TemporaryFile ()
f.write(base64.b64decode(r['qrImage']))
PIL.image = Image.open(f)
PIL.image.show()

def push_response ():
 while True:
  time.sleep(1)
  response = (s.post(url+'openotpCheckConfirm', data = {
   "session": r['session'],
  }).json())
  if response['code'] != 2:
   break
 return response

def ask_otp ():
 otp = input("Enter the otp: ")
 return  s.post(url+'openotpOfflineConfirm', data = {
  "session": r['session'],
  "response": otp,
 }).json()

pool1 = multiprocessing.pool.ThreadPool(processes=1)
pool2 = multiprocessing.pool.ThreadPool(processes=1)

push = pool1.apply_async(push_response)
otp  = pool2.apply_async(ask_otp)

while True:
 try:
  response = push.get(timeout=1)
  if response:
   break;
 except :
  pass
```

```
 try:
  response = otp.get(timeout=1)
  if response:
   break;
 except :
  pass

print(response['message'])
```

6. Transactions Management

The confirmation feature include 3 methods to help you to manage your trasactions :

6.1 openotpCheckConfirm

This method allow you to check the status of a transaction based on the session number.

```
#!/usr/bin/python3
import requests

url = 'http://my_webadm_server:8080/openotp/json/' # In Production, HTTPS on 8443 port
must be used.

s = requests.Session()

r = s.post(url+'openotpNormalConfirm', data = {
 "username": "john",
 "client": "my_web_banking_application",
 "source": "1.2.3.4", # customer IP addresse
 "async": True,
 "data": "<html><b>Sample Transaction</b><br><br>Account: <font
color=red>Testing</font><br>Amount: <font color=red>x Euros</font><br></html>",
}).json()

#check the confirmation:
print(s.post(url+'openotpCheckConfirm', data = {
 "session": r['session'],
}).json())
```

6.2 openotpListConfirm

This method will return all pending transactions.

```
#!/usr/bin/python3

import requests

url = 'http://my_webadm_server:8080/openotp/json/' # In Production, HTTPS on 8443 port
must be used.

s = requests.Session()

print(s.post(url+'openotpListConfirm'}).json())
```

6.3 openotpCancelConfirm

This method allows you to cancel a transaction based on a session number.

```
#!/usr/bin/python3
import requests

url = 'http://my_webadm_server:8080/openotp/json/' # In Production, HTTPS on 8443 port
must be used.

s = requests.Session()

r = s.post(url+'openotpNormalConfirm', data = {
 "username": "john",
 "client": "my_web_banking_application",
 "source": "1.2.3.4", # customer IP addresse
 "async": True,
 "data": "<html><b>Sample Transaction</b><br><br>Account: <font
color=red>Testing</font><br>Amount: <font color=red>x Euros</font><br></html>",
}).json()

#stop the confirmation:
print(s.post(url+'openotpCancelConfirm', data = {
 "session": r['session'],
}).json())
```

7. Test through WebADM web interface

You can test online methods through the WebADM Administrator Portal. The account must be activated and a push Token must be registered on the user account who will sign the transaction. The test page is found under the users page in `WebADM Admin GUI`. First, find the user you want to test with on the LDAP browser in the left side of the window, or using the search function. Once you have the users page open, go to `Application Actions` box > click `MFA Authentication Server` > `Test User Confirmation`.

You are now on the WebADM Confirmation Tester.

7.1 Synchronous Confirmation with Push

Once you are on the test page, configure the `Confirmation Method` setting to `Push Notification` and each option you want to use. Click on the `Start` button to start the transaction.

The user in question will received a notification from OpenOTP and can select it to open the OpenOTP Token application. User will then see the transaction information send by the server.

In this case a file was attached to the request, so the user must select the `Next` button.

Once the user has reviewed the document and they can select the `Done` button to close the document and continue the transaction. This will bringe them back to the Transaction details screen and where they are able to approve the transaction.

The user selects “approve”.

On the WebADM test test page, the following information is returned by the phone to the server:

To review information about that transaction, from WebADM Admin GUI, click on `Databases` tab `SQL Data Tales` > `Recorded Sessions & Transactions` I put a filter on the User DN for the SQL result to easily find the transaction. The first entry is the one I performed.

Select `View` button to review the transaction details:

7.2 Synchronous Confirmation with QRCode

To test a synchronous confirmation with a QRCode through the WebaDM, go to the Confirmation tester and select

`QRCode scan` for `Confirmation Method` setting instead of `Push Notification`.

After configuring your desired settings, click the `Start` button to start and generate a QRCode containing transaction information.

Only the specific end-user with its the correct token can scan the QRCode and read the information contained in the transaction. When you scan it you see:

After the scan, the user is prompted for the transaction on the phone. A document as been attached, so the user has to click the `Next` button to review the document before they are able to sign it.

After reviewing the document, the user clicks the `Done` button to close the document and proceed to the signature. At the transaction details screen, they can click the approve button and the response is sent to the server.

When the response arrives at the server, you can see the result on the WebADM test page. The handwritten signature and paraph have been attached to the transaction.

To review information about that transaction from WebADM GUI, go to `Databases` tab `SQL Data Tales` > `Recorded Sessions & Transactions`.

8. Including a Web Form in a Transaction

You can add a custom form to a transaction request. The form and the response of that form will be signed.

```
<b>My Form</b><br>
<br>
<form name="myform">
  <table>
  <tr>
    <td>Your Name: </td>
 <td><input type="text" name="setting1"></td>
  </tr>
  <tr>
    <td>Are you Ok: </td>
 <td>
    <input type="radio" name="setting2" value="1" checked> Yes
    <input type="radio" name="setting2" value="0"> No
 </td>
  </tr>
  <tr>
    <td>Option: </td>
 <td>
    <select name="setting3">
       <option value="1">I'm Ok</option>
    <option value="2">I'm not Ok</option>
    </select>
 </td>
  </tr>
  </table>
</form>
```

This form can be passed in the transaction requests in the `form` parameter of `openotpNormalConfirmRequest` or `openotpConfirmQRCodeRequest` methods. Example from the WebADM test page:

User receives the notification on the phone, presses the `Next` button and on the next page, the form is displayed:

Once the form is completed, the user selects `Submit` and the transaction response is sent to the server.

On the server-side, we can see the completed form:

And here is the result in the logs database:

9. OpenOTP WSDL

```
<?xml version="1.0" encoding="UTF-8"?>

<definitions targetNamespace="http://www.rcdevs.com/wsdl/openotp/"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:tns="http://www.rcdevs.com/wsdl/openotp/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<!-- Mobile Confirmation Methods -->

<message name="openotpNormalConfirmRequest">
    <part name="username" type="xsd:string"/>
    <part name="domain" type="xsd:string"/>
    <part name="data" type="xsd:string"/>
    <part name="file" type="xsd:base64Binary"/>
    <part name="form" type="xsd:string"/>
    <part name="async" type="xsd:boolean"/>
    <part name="timeout" type="xsd:integer"/>
    <part name="issuer" type="xsd:string"/>
    <part name="client" type="xsd:string"/>
    <part name="source" type="xsd:string"/>
    <part name="settings" type="xsd:string"/>
</message>

<message name="openotpCheckConfirmRequest">
    <part name="session" type="xsd:string"/>
</message>

<message name="openotpOfflineConfirmRequest">
    <part name="session" type="xsd:string"/>
    <part name="response" type="xsd:string"/>
</message>

<message name="openotpConfirmResponse">
    <part name="code" type="xsd:integer"/>
    <part name="error" type="xsd:string"/>
    <part name="message" type="xsd:string"/>
    <part name="session" type="xsd:string"/>
    <part name="timeout" type="xsd:integer"/>
    <part name="comment" type="xsd:string"/>
```

```
    <part name="address" type="xsd:string"/>
    <part name="localTime" type="xsd:string"/>
    <part name="signature" type="xsd:string"/>
    <part name="responses" type="xsd:string"/>
    <part name="paraph" type="xsd:string"/>
</message>

<message name="openotpConfirmQRCodeRequest">
    <part name="username" type="xsd:string"/>
    <part name="domain" type="xsd:string"/>
    <part name="data" type="xsd:string"/>
    <part name="file" type="xsd:base64Binary"/>
    <part name="form" type="xsd:string"/>
    <part name="async" type="xsd:boolean"/>
    <part name="timeout" type="xsd:integer"/>
    <part name="issuer" type="xsd:string"/>
    <part name="client" type="xsd:string"/>
    <part name="source" type="xsd:string"/>
    <part name="settings" type="xsd:string"/>
    <part name="qrFormat" type="xsd:string"/>
    <part name="qrSizing" type="xsd:integer"/>
    <part name="qrMargin" type="xsd:integer"/>
</message>

<message name="openotpConfirmQRCodeResponse">
    <part name="code" type="xsd:integer"/>
    <part name="error" type="xsd:string"/>
    <part name="message" type="xsd:string"/>
    <part name="session" type="xsd:string"/>
    <part name="timeout" type="xsd:integer"/>
    <part name="qrImage" type="xsd:base64Binary"/>
</message>

<message name="openotpListRequest"/>

<message name="openotpListResponse">
    <part name="code" type="xsd:string"/>
    <part name="error" type="xsd:string"/>
    <part name="message" type="xsd:string"/>
    <part name="jsonData" type="tns:string"/>
</message>

<message name="openotpStatusRequest"/>

<message name="openotpStatusResponse">
   <part name="status" type="xsd:boolean"/>
   <part name="message" type="xsd:string"/>
</message>

<portType name="openotpPortType">
   <operation name="openotpSimpleLogin">
      <input name="openotpSimpleLoginRequest" message="tns:openotpSimpleLoginRequest"/>
```

```
      <output name="openotpSimpleLoginResponse" message="tns:openotpLoginResponse"/>
   </operation>
   <operation name="openotpNormalLogin">
      <input name="openotpNormalLoginRequest" message="tns:openotpNormalLoginRequest"/>
      <output name="openotpNormalLoginResponse" message="tns:openotpLoginResponse"/>
   </operation>
   <operation name="openotpChallenge">
      <input name="openotpChallengeRequest" message="tns:openotpChallengeRequest"/>
      <output name="openotpChallengeResponse" message="tns:openotpLoginResponse"/>
   </operation>
   <operation name="openotpListLogin">
      <input name="openotpListLoginRequest" message="tns:openotpListRequest"/>
      <output name="openotpListLoginResponse" message="tns:openotpListResponse"/>
   </operation>
   <operation name="openotpNormalConfirm">
      <input name="openotpNormalConfirmRequest"
message="tns:openotpNormalConfirmRequest"/>
      <output name="openotpNormalConfirmResponse"
message="tns:openotpConfirmResponse"/>
   </operation>
   <operation name="openotpOfflineConfirm">
      <input name="openotpOfflineConfirmRequest"
message="tns:openotpOfflineConfirmRequest"/>
      <output name="openotpOfflineConfirmResponse"
message="tns:openotpConfirmResponse"/>
   </operation>
   <operation name="openotpConfirmQRCode">
      <input name="openotpConfirmQRCodeRequest"
message="tns:openotpConfirmQRCodeRequest"/>
      <output name="openotpConfirmQRCodeResponse"
message="tns:openotpConfirmQRCodeResponse"/>
   </operation>
   <operation name="openotpCheckConfirm">
      <input name="openotpCheckConfirmRequest"
message="tns:openotpCheckConfirmRequest"/>
      <output name="openotpCheckConfirmResponse" message="tns:openotpConfirmResponse"/>
   </operation>
   <operation name="openotpCancelConfirm">
      <input name="openotpCancelConfirmRequest"
message="tns:openotpCheckConfirmRequest"/>
      <output name="openotpCancelConfirmResponse"
message="tns:openotpConfirmResponse"/>
   </operation>
   <operation name="openotpListConfirm">
      <input name="openotpListConfirmRequest" message="tns:openotpListRequest"/>
      <output name="openotpListConfirmResponse" message="tns:openotpListResponse"/>
   </operation>
   <operation name="openotpStatus">
      <input name="openotpStatusRequest" message="tns:openotpStatusRequest"/>
      <output name="openotpStatusResponse" message="tns:openotpStatusResponse"/>
   </operation>
</portType>
```

```
<binding name="openotpBinding" type="tns:openotpPortType">
   <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
   <operation name="openotpSimpleLogin">
      <soap:operation soapAction="openotpSimpleLogin"/>
      <input><soap:body use="literal" namespace="urn:openotp"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
      <output><soap:body use="literal" namespace="urn:openotp"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
   </operation>
   <operation name="openotpNormalLogin">
      <soap:operation soapAction="openotpNormalLogin"/>
      <input><soap:body use="literal" namespace="urn:openotp"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
      <output><soap:body use="literal" namespace="urn:openotp"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
   </operation>
   <operation name="openotpChallenge">
      <soap:operation soapAction="openotpChallenge"/>
      <input><soap:body use="literal" namespace="urn:openotp"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
      <output><soap:body use="literal" namespace="urn:openotp"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
   </operation>
   <operation name="openotpListLogin">
      <soap:operation soapAction="openotpListLogin"/>
      <input><soap:body use="literal" namespace="urn:openotp"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
      <output><soap:body use="literal" namespace="urn:openotp"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
   </operation>
   <operation name="openotpNormalConfirm">
      <soap:operation soapAction="openotpNormalConfirm"/>
      <input><soap:body use="literal" namespace="urn:openotp"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
      <output><soap:body use="literal" namespace="urn:openotp"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
   </operation>
   <operation name="openotpOfflineConfirm">
      <soap:operation soapAction="openotpOfflineConfirm"/>
      <input><soap:body use="literal" namespace="urn:openotp"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
      <output><soap:body use="literal" namespace="urn:openotp"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
   </operation>
   <operation name="openotpConfirmQRCode">
      <soap:operation soapAction="openotpConfirmQRCode"/>
      <input><soap:body use="literal" namespace="urn:openotp"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
      <output><soap:body use="literal" namespace="urn:openotp"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
   </operation>
```

```
   <operation name="openotpCheckConfirm">
      <soap:operation soapAction="openotpCheckConfirm"/>
      <input><soap:body use="literal" namespace="urn:openotp"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
      <output><soap:body use="literal" namespace="urn:openotp"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
   </operation>
   <operation name="openotpCancelConfirm">
      <soap:operation soapAction="openotpCancelConfirm"/>
      <input><soap:body use="literal" namespace="urn:openotp"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
      <output><soap:body use="literal" namespace="urn:openotp"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
   </operation>
   <operation name="openotpListConfirm">
      <soap:operation soapAction="openotpListConfirm"/>
      <input><soap:body use="literal" namespace="urn:openotp"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
      <output><soap:body use="literal" namespace="urn:openotp"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
   </operation>
   <operation name="openotpStatus">
      <soap:operation soapAction="openotpStatus"/>
      <input><soap:body use="literal" namespace="urn:openotp"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
      <output><soap:body use="literal" namespace="urn:openotp"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
   </operation>
</binding>

<service name="openotpService">
   <port name="openotpPort" binding="tns:openotpBinding">
      <soap:address location="%ADDRESS%"/>
   </port>
</service>

</definitions>
```