Kerkenbos 13-27
6546 BG Nijmegen

T : (024) 345 5000
E : info@tunix.nl
K : Arnhem 09145594
B : 813915120B01

TUNIX/Authenticatie

Web Service
Developer Guide

(released)

Classificatie: Classified Customer Document
Doelgroep: Web Service Client Developers
Eigenaar: William Wanders
Afdeling: TUN/PNB
Project: TUN/MAR212

Dit document is het laatst geauditeerd door Ronald Pikkert op 11-09-2018.
De informatie in dit document is geldig tot 25-06-2021.

ABN-Amro Bank: 42.16.42.955
BIC: ABNANL2A
IBAN: NL98ABNA0421642955

Table of contents

1. Version History 2

2. Introduction 3

3. Security Context 4

4. AuthenticateService 5
4.1 doAuthenticate 5
4.1.1 Request / Response 7
4.2 lookupAuthenticateType 10
4.2.1 Request / Response 10
4.3 getActiveBlocks 11
4.3.1 Request / Response 12
4.4 unBlock 14
4.4.1 Request / Response 14
4.5 doPoll 15
4.5.1 Request / Response 19

5. Usage Examples 22
5.1 Curl SOAP CLI Script 22

1. Version History

Version	Change description
2.2	- moved to new naming scheme.
2.1	- added doPoll service.
2.0	- added getActiveBlocks service; - added unBlock service.
1.1	- added lookupAuthenticateType service; - first public release.
1.0	- initial version

Table 1 Document Revisions

2. Introduction

In its standard offering the TUNIX/Authenticatie service supports RADIUS as an authentication end-point. This guide describes how the TUNIX/Authenticatie 2FA-authentication service can be integrated in many other applications or services by means of a SOAP Web Service.

3. Security Context

The TUNIX/Authenticatie Web Service is implemented as a SOAP Web Service over HTTPS in conjunction strong client certificate based authentication. There are two different usage scenarios available within the Web Service:

1 Factor 1 and Factor 2 authentication;
Factor 1 authentication requires a related TUNIX/CloudProxy instance to be available.

2 Factor 2 only authentication.

Factor 1 consist of a principal/credential combination like the well-known username and password scheme. Factor 2 can take two forms:

1 App based authentication;

2 SMS based One-Time-Password (OTP) authentication.

Factor 2 OTP involves a second challenge/response step. A secure relation with the preceding Factor 1 is implemented by means of a secure token handed down as part of the Factor 1 phase and which is required to sent back with the OTP during the Factor 2 handling phase.

In short all communication, with exception of the OTP SMS message, will be encrypted and authenticated. Related authentication steps are securely linked by a token. The Web Service client must handle this token with care to ensure secure, safe and reliable completion of the authentication process.

4. AuthenticateService

The Web Service is accessible through the URL's:

Version	URL
2.1	https://api1.keyapp.nl:443/txauth-ws-v2/AuthenticateService

For additional information, like WSDL, please consult the following table:

Endpoint	Information
Service Name:	{http://txauth.ws.tunix.nl/}AuthenticateService
Port Name:	{http://txauth.ws.tunix.nl/}AuthenticatePort
Address (2.1):	https://api1.keyapp.nl:443/txauth-ws-v2/AuthenticateService
WSDL (2.1):	https://api1.keyapp.nl:443/txauth-ws-v2/AuthenticateService?wsdl
Implementation class:	nl.tunix.ws.txauth.Authenticate

When importing the WSDL into an IDE please note that access to this Web Service requires authentication with a client certificate issued by TUNIX for this web service.

Note, prior to 2018-11-27 a different URL (https://ws.keyapp.nl/...) was primarily used to provide wider backward compatibility for SSLv3/TLSv1 and TLSv1.x up to TLSv1.2. Version 2.1 is only supported in conjunction with enforced TLS-handshake and TLSv1.0, TLSv1.1 and TLSv1.2. Use of TLSv1.2 is highly recommended.

4.1 doAuthenticate

The *doAuthenticate* SOAP-Action is used to perform an authentication with the TUNIX/Authenticatie service. The action accepts the following parameters from the SOAP Message:

Parameter	Type	Description
principal	string	Login, Username, ...
credential	string	Password, OTP
telephoneNumber	string	Internationally formatted telephone number
state	string	secure token returned in AuthenticateResponse by preceding doAuthenticate action

Results of a doAuthenticate action are reported as parameters inside an *AuthenticateResponse* SOAP-response. Parameters are described in Following table:

Parameter	Type	Description
status	string	
	ACK	authentication succeeded
	CHALLENGE	challenge request, literal text in challenge parameter
	NACK	authentication rejected
	NACK/EXPIRED	authentication rejected or expired
	EXPIRED	unable to authenticate, challenge expired
	FAILED	authentication failed
		no telephone number available
	UNREGISTERED	no App is registered for the telephone number
	UNAVAILABLE	CloudProxy properties unavailable
		CloudProxy service unavailable
		CloudProxy unavailable
		SMS Gateway unavailable
		Sending OTP by SMS unavailable
		Service unavailable
	BLOCKED	authentication failed, temporarily blocked
		Principal temporarily blocked
		unable to authenticate, temporarily blocked
	DISABLED	Service disabled
	INVALIDDATA	illegal state argument
		Invalid state data
		Invalid telephone number
	EXCEPTION	
challenge	string	Challenge for Factor 2 OTP
state	string	Secure token for Phase 2 OTP
emsg	string	Error message
trace	string	Last phase transition trace

4.1.1 Request / Response

In the following section basic SOAP Request and SOAP Responses are discussed for an authentication process. The example describes the full Factor 1 and Factor 2 authentication process with uses OTP as the second factor. Deviations for the other authentication forms are noted where applicable.

To initiate an authentication process a *doAuthenticate* SOAP-request such as the following is POST-ed:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmln
s:txa="http://txauth.ws.tunix.nl/">
   <soapenv:Header/>
   <soapenv:Body>
      <txa:doAuthenticate>
         <principal>user</principal>
         <credential>secret</credential>
      </txa:doAuthenticate>
   </soapenv:Body>
</soapenv:Envelope>
```

In the above SOAP-request the *<principal>* tag contains the literal string that identifies the user. This parameter is mandatory. The *<credential>* contains the literal plain password. The *<principal>* and *<credential>* are sufficient to initiate an authentication process from Factor 1.

NOTE:

To initiate a Factor 2 Only authentication process leave out the *<credential>*-tag and insert a *<telephoneNumber>*-tag that contains the phonenumber related to the principal.

Upon successful passing Factor 1 of the authentication process, principal and credential were successfully verified, an OTP is sent to the principals telephone number and a *CHALLENGE* status SOAP-response like the following is returned:

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
<S:Body>
<ns2:doAuthenticateResponse xmlns:ns2="http://txauth.ws.tunix.nl/">
<state>NkFtOXltcmM4eXBqZG9zdGZja0wrdVZnd24yRGxuQ1QxOE16QkVGWXU</state>
<challenge>Please type your One Time Password</challenge>
<status>CHALLENGE</status>
</ns2:doAuthenticateResponse>
</S:Body>
</S:Envelope>
```

The *<challenge*-tag contains the challenge for the user. The *<state>*-tag contains the security token to be returned together with the OTP as received from the user in a subsequent *doAuthenticate* action to continue the authentication process.

NOTE:

If an App is registered with the user's telephone number then Factor 2 is completed *out-of-band*. The SOAP response is not sent until:

1 external completion of Factor 2 authentication by the user's registered App instance;

2 expiry of the Factor 2 timeslot;

3 an error occurs.

To complete Factor 2 of the authentication process the user entered OTP has to be included with the security token in a new *doAuthenticate* action SOAP Message to the Web Service. The OTP is enclosed within the <*credential*>-tag as is the security token within the <*state*>-tag.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
         xmlns:txa="http://txauth.ws.tunix.nl/">
  <soapenv:Header/>
  <soapenv:Body>
    <txa:doAuthenticate>
      <principal>user</principal>
      <credential>OTP</credential>
      <state>NkFtOXltcmM4eXBqZG9zdGZja0wrdVZnd24yRGxuQ1QxOE16QkVGWXU</state>
    </txa:doAuthenticate>
  </soapenv:Body>
</soapenv:Envelope>
```

A successful completion of the authentication process is indicated by the value *ACK* in the <*status*>-tag in the response.

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:doAuthenticateResponse xmlns:ns2="http://txauth.ws.tunix.nl/">
      <trace>Phase 2 Completed</trace>
      <status>ACK</status>
    </ns2:doAuthenticateResponse>
  </S:Body>
</S:Envelope>
```

Any other <*status*>-tag value indicates the authentication process is *NOT* successfully completed.

NOTE:

If the OTP was wrongly entered and there does not validate along security token the process repeats the status *CHALLENGE* response until the token actually expires.

4.2 lookupAuthenticateType

The *lookupAuthenticateType* SOAP-Action queries the type authentication that will be used for a specific telephone number when using the doAuthenticate SOAP-Action. The action accepts the following parameters from the SOAP Message:

Parameter	Type	Description
telephoneNumber	string	Internationally formatted telephone number

Results of a lookupAuthenticateType action are reported as parameters inside an *AuthenticateResponse* SOAP-response. Parameters are described in Following table:

Parameter	Type	Description
atype	string	
	CHALLENGE	One-Time-Password (SMS) type
	OUTOFBAND	TUNIX/KeyApp type
status	string	
	OK	authentication succeeded
	FAILED	lookup failed
	UNREGISTERED	no App is registered for the telephone number
	DISABLED	Service disabled
	EXCEPTION	
emsg	string	Error message
trace	string	Last phase transition trace

4.2.1 Request / Response

In the following section basic SOAP Request and SOAP Responses are discussed for a lookup process.

To initiate a lookup process a *lookupAuthenticateType* SOAP-request such as the following is POST-ed:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"→
←line continued→ xmlns:txa="http://txauth.ws.tunix.nl/">
   <soapenv:Header/>
   <soapenv:Body>
      <txa:lookupAuthenticateType>
         <telephoneNumber>0031612345789</telephoneNumber>
      </txa:lookupAuthenticateType>
   </soapenv:Body>
</soapenv:Envelope>
```

In the above SOAP-request the *<telephoneNumber>* tag contains the literal telephone number string. This parameter is mandatory. Upon completion the following is returned:

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:lookupAuthenticateTypeResponse xmlns:ns2="http://txauth.ws.tunix.nl/">
      <atype>CHALLENGE</atype>
      <status>OK</status>
    </ns2:lookupAuthenticateTypeResponse>
  </S:Body>
</S:Envelope>
```

A successful completion of the lookup process is indicated by the value *OK* in the *<status>*-tag in the response. Any other *<status>*-tag value indicates the lookup process is *NOT* successfully completed.

The *<atype>*-tag contains the type of authentication that will be applicable for the reqeuested telephone number. A value of *CHALLENGE* indicates an One-Time-Password via SMS type of authentication for the given telephone number whereas *OUTOFBAND* indicates a TUNIX/KeyApp type authentication is expected.

4.3 getActiveBlocks

The *getActiveBlocks* SOAP-Action queries for any blocked principals. Returned blocked principal data can be used as a information source and to perform one or more unBlock SOAP-Actions to remove an existing block. This action does not require any parameter.

Results of a getActiveBlocks action are reported as parameters inside an *AuthenticateResponse* SOAP-response. Parameters are described in Following table:

Parameter	Type	Description
status	string	
	OK	authentication succeeded
	FAILED	lookup failed
	UNREGISTERED	no App is registered for the telephone number
	DISABLED	Service disabled
	EXCEPTION	
emsg	string	Error message
trace	string	Last phase transition trace

4.3.1 Request / Response

In the following section basic SOAP Request and SOAP Responses are discussed for a lookup process.

To initiate a lookup process a *lookupAuthenticateType* SOAP-request such as the following is POST-ed:

```
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"→
←line continued→ xmlns:txa="http://txauth.ws.tunix.nl/">
   <soapenv:Header/>
   <soapenv:Body>
      <txa:getActiveBlocks>
      </txa:getActiveBlocks>
   </soapenv:Body>
</soapenv:Envelope>
```

The above SOAP-request queries for any blocked principals. Please note that this action parameterless.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:getActiveBlocksResponse xmlns:ns2="http://txauth.ws.tunix.nl/">
      <block>
        <id>0afe6a72-b0f6-11e8-8c42-ba69df25180b</id>
        <principal>mies</principal>
        <fromDate>2018-09-05T10:25:42Z</fromDate>
        <untilDate>2018-09-05T10:30:42Z</untilDate>
      </block>
      <block>
        <id>07e75a1f-b0f6-11e8-8c42-ba69df25180b</id>
        <principal>noot</principal>
        <fromDate>2018-09-05T10:25:37Z</fromDate>
        <untilDate>2018-09-05T10:30:37Z</untilDate>
      </block>
      <block>
        <id>0560de39-b0f6-11e8-8c42-ba69df25180b</id>
        <principal>aap</principal>
        <fromDate>2018-09-05T10:25:33Z</fromDate>
        <untilDate>2018-09-05T10:30:33Z</untilDate>
      </block>
      <trace>transduceActiveBlocks</trace>
      <status>OK</status>
    </ns2:getActiveBlocksResponse>
  </S:Body>
</S:Envelope>
```

A successful completion of the active block lookup process is indicated by the value *OK* in the *<status>*-tag in the response. Any other *<status>*-tag value indicates the block lookup process is *NOT* successfully completed.

Multiple *<block>* tags refer to multple active blocks. A *<block>*-tag contains sub-tags as described in table:

Tag	Description
id	0560de39-b0f6-11e8-8c42-ba69df25180b</id>
principal	name of principal/user that is blocked
fromDate	date and time the block was activated
untilDate	date and time untill which the block will be activate

Table 2 Block parameters

Please not that date en time references used in the *fromDate* and *untillDate* tags are based on UTC. The *id* tag value can be used to perform an *unBlock* operation.

4.4 unBlock

The *unBlock* SOAP-Action removes a specific (active) block. The action accepts a single parameters from the SOAP Message:

Parameter	Type	Description
id	string	id of block to be removed

Results of an unBlock action are reported as parameters inside an *AuthenticateResponse* SOAP-response. Parameters are described in Following table:

Parameter	Type	Description
status	string	
	OK	authentication succeeded
	FAILED	lookup failed
	DISABLED	Service disabled
	EXCEPTION	
emsg	string	Error message
trace	string	Last phase transition trace

Table 3 AuthenticateResponse

4.4.1 Request / Response

In the following section basic SOAP Request and SOAP Responses are discussed for an unBlock process.

To initiate an unBlock process an *unBlock* SOAP-request such as the following is POST-ed:

```
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"→
←line continued→ xmlns:txa="http://txauth.ws.tunix.nl/">
   <soapenv:Header/>
   <soapenv:Body>
      <txa:unBlock>
        <blockId>21c48332-b102-11e8-8c42-ba69df25180b</blockId>
      </txa:unBlock>
   </soapenv:Body>
</soapenv:Envelope>
```

In the above SOAP-request the *<blockId>* tag contains the id string corresponding to an active block that should be removed. This parameter is mandatory. Upon completion the following is returned:

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:unBlockResponse xmlns:ns2="http://txauth.ws.tunix.nl/">
      <emsg>kid=462,name=TUNIX/Authenticatie Demo,dn=CN=demo1.dmo.tunixx,→
←line continued→ OU=DMO, O=TUNIX Digital Security, L=Nijmegen, ST=Gelderland, C=NL</emsg>
      <trace>Lookup</trace>
      <status>OK</status>
    </ns2:unBlockResponse>
  </S:Body>
</S:Envelope>
```

A successful completion of the unblock process is indicated by the value *OK* in the *<status>*-tag in the response. Any other *<status>*-tag value indicates the unblock process is *NOT* successfully completed.

4.5 doPoll

The *doPoll* SOAP-Action is a heartbeat function that allows the Web Service to actively monitor peer liveliness and pass on environmental information usefull for support.

The action accepts a single parameters from the SOAP Message:

Parameter	Type	Description
hia	string	an extendable JSON string consisting of key,value pairs as described in the *HIA Hearbeat JSON* table.

The following table descrtibes all mandatory key/value-pairs for a doPoll heartbeat. Although all keys are required their values need only to comply to the JSON-value-string format. It is highly recommended to supply meaningfull data to enable and ease future communications, assistence and/or support.

All values must be given with respect to the actual start of the main application as mentioned in the *started* key.

Key	Value description / Examples
started	Startup timestamp (date+time) the application was started - use of UTC time-reference is highly recommended; - native timestamp should contain or be augmented with a specific timezone reference (Eg. CET, CEST, .). - 08/11/2018 17:51:22 - Dec 08 12:21:57 EST 2018
hardwarePlatform	Hardware Platform - AMD/Intel x86/x64 - VMware hardware version 4 - Amazon EC2 micro.t2 - ARMv7

Key	Value description / Examples
processorCount	Number of CPU's - 2 - 1 socket 2 cores
os	Type of Operating System - Microsoft Windows 10 Pro - 64 bits - Ubuntu Certified 16.04 - Oracle Solaris 11.2
commonLanguageRuntimeVersion	Common Language Runtime Version - 4.0.30319.42000 - OpenJDK 1.8.181 - NodeJS 6.14.4 - Go 1.9 - PHP 5.7 - cURL 7.43.0
targetFrameworkAttribute	Target Framework Attribute - .NET4.5.2 - Java EE 7 - Spring - WordPress - ExpressJS

Key	Value description / Examples
version	Application Version - 3.7b159
hostName	Reference name to host or system sending the doPoll heartbeat - WIN10-EA683AF0 - udfs.acme-bv.nl
localIP	Local/Internal IP ddress, refering to the system sending the heartbeat information - 192.168.1.7
externalIP	Public/External IP address or range, refering to the external orginating address the heartbeat ought to have been sent from - 192.0.2.7
customerCode	Tenant or Company reference code - C20160227-2102 - ACM
customerName	Tenant or Company name - C321 T754 - ACME - W.E. Coyote Industries

Key	Value description / Examples
ackOnConnectFailureCount	Number of internally performed auto acknowlegdements on external connection failure. zero or postive integer number as string
authSkipCount	Number of skipped authentications zero or postive integer number as string
authDoneCount	Number of performed authentications zero or postive integer number as string
failedPollCount	Number of failed doPoll heatbeat requests zero or postive integer number as string

Table 4 Heartbeat JSON key/value-data

Results of an doPoll action are reported as parameters inside an *AuthenticateResponse* SOAP-response. Parameters are described in Following table:

Parameter	Type	Description
status	string	
	OK	authentication succeeded
	FAILED	lookup failed
	DISABLED	Service disabled
	EXCEPTION	
emsg	string	Error message
trace	string	Last phase transition trace

Table 5 doPoll reponses

4.5.1 Request / Response

In the following section basic SOAP Request and SOAP Responses are discussed for a doPoll heartbeat.

To initiate a doPoll heartbeat, a *doPoll* SOAP-request such as the following needs to be POST-ed:

```
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"→
←line continued→ xmlns:txa="http://txauth.ws.tunix.nl/">
   <soapenv:Header/>
   <soapenv:Body>
      <txa:doPoll>
        <hia>{}</hia>
      </txa:doPoll>
   </soapenv:Body>
</soapenv:Envelope>
```

In the above SOAP-request the *<hia>* tag contains the JSON string with heartbeat information This parameter is mandatory.

Upon completion the following is returned:

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:doPollResponse xmlns:ns2="http://txauth.ws.tunix.nl/">
      <emsg>kid=462,name=TUNIX/Authenticatie Demo,dn=CN=demo1.dmo.tunixx,→
←line continued→ OU=DMO, O=TUNIX Digital Security, L=Nijmegen, ST=Gelderland, C=NL</emsg>
      <trace>Lookup</trace>
      <status>OK</status>
    </ns2:doPollResponse>
  </S:Body>
</S:Envelope>
```

A successful completion of the doPoll heartbeat is indicated by the value *OK* in the *<status>*-tag in the response. Any other *<status>*-tag value indicates the doPoll heartbeat is *NOT* successfully completed. The *<tya>*-tag returns a basic JSON string with the following information variables:

Variable	Comment
version	version id
received_on	an UTC timestamp indicating when the doPoll heartbeat was received


TUNIX
digital security

5. Usage Examples

In this section some simple code-examples are given to present a starting point from where to integrate the TUNIX/Authenticatie Web Service in Web Service Client developments of your own.

To use these examples you will need a valid TUNIX/Authenticatie Client Certificate.

5.1 Curl SOAP CLI Script

The following script illustrates the basic use of SOAP Messages to use the TUNIX/Authenticatie Web Service. The script needs the client certificate in PEM format.

NOTE
Not all curl versions are alike and you may need to tweak the code a little to make it work.

Script txauth-test.sh

```
#!/bin/sh
# txauth-test.sh - test script
#
# Sample code for TUNIX/Authenticatie Web Service
#
DOA_SRV="ws.keyapp.nl"
DOA_REQ="/tmp/doa.req1.$$.xml"
DOA_RES="/tmp/doa.res1.$$.xml"

doSoapRequest()
{
    echo "Generate doAuthenticate from request.xml.in"
    sed -e "s|@PARAMS@|${params}|" <request.xml.in >${DOA_REQ}
curl \
        --insecure \
        --cert certs/host.pem:changeit \
        --cert-type PEM \
        --header "Content-Type: text/xml;charset=UTF-8" \
        --header "SOAPAction: doAuthenticate" \
        --data @${DOA_REQ} \
        "https://${DOA_SRV}/txauth-ws/AuthenticateService" 2>/dev/null |
                grep "<" >${DOA_RES}
    xmllint --format ${DOA_RES}
    rm -f ${DOA_REQ} ${DOA_RES}
}

params=""
principal=""
while getopts p:c:t: name
do
    case ${name} in
        p)   params="${params}<principal>${OPTARG}</principal>"; principal=${OPTARG};;
        c)   params="${params}<credential>${OPTARG}</credential>";;
        t)   params="${params}<telephoneNumber>${OPTARG}</telephoneNumber>";;
        h|?) printf "Usage: %s: -p principal ( -c credential | -t telno)\n"  $0
                exit 2;;
    esac
done

result=$(doSoapRequest | sed -e 's|<|:|g' -e 's|>|:|g' | nawk -F: '
$2 == "state" { state=$3 }
$2 == "status" { status=$3 }
$2 == "challenge" { challenge=$3 }
END { print status ":" challenge ":" state }
\&')

while /bin/true
```

```
do
    case "$(echo ${result}|cut -d : -f 1)" in
        CHALLENGE)
            printf "$(echo ${result}|cut -d : -f 2): "
            read credential
            state="$(echo ${result}|cut -d : -f 3)"
            params="<principal>${principal}</principal>
                    <credential>${credential}</credential>
                    <state>${state}</state>"
            result=$(doSoapRequest | sed -e 's|<|:|g' -e 's|>|:|g' | nawk -F: '
$2 == "state" { state=$3 }
$2 == "status" { status=$3 }
$2 == "challenge" { challenge=$3 }
END { print status ":" challenge ":" state }
\&')
            ;;
        *) echo "${result}"; exit
    esac
done
```

The script uses a template SOAP-request file *request.xml.in* with the following contents:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:txa="http://txauth.ws.tunix.nl/">
   <soapenv:Header/>
   <soapenv:Body>
      <txa:doAuthenticate>
         @PARAMS@
      </txa:doAuthenticate>
   </soapenv:Body>
</soapenv:Envelope>
```