inWebo API Documentation

Contents

Introduction

This document describes the list of Web Services forming inWebo API. The SOAP API is described by 2 WSDL files:

- Authenticate.wsdl // used for authentication requests
- Provisioning.wsdl // used for user management tasks

Access to inWebo API is 2-factor secured. Configuration is done in inWebo Administration Console:

- Restrict authorized IP addresses allowed to submit SOAP requests
- Implement SSL client certificate authentication

Some API methods are also proposed in REST format.

Warning: depending on the offer you subscribed to (Identity Guard, Safe Transactions, Enterprise), some functions will not be available.

Authentication with SOAP API

There are 2 possible authentication methods: authenticate and authenticateWithIP. The first one will just validate a login and an OTP. The second one will allow, for authentication with inWebo Application for Mac & PC or inWebo Helium only, to verify the source IP Address of the end-user.

Authenticate

```
authenticate(String userId, String serviceId, String token)
```

Returns a string

```
authenticateResponse
```

This is an object with the following properties:

```
public String authenticateReturn;
```

This function allows validating a One-Time Password for a given login. The string returned is a code which can be “OK” (success) or “NOK:<cause>” (error).

AuthenticateWithIP

```
authenticateWithIP(String userId, String serviceId, String token, String ip)
```

Returns:

```
authenticateWithIpResponse
```

This is an object with the following properties:

```
public String authenticateReturn;
```

AuthenticateWithIP provides an additional MITM detection for OTP generated by inWebo. The behavior of this function is as follows:

- If the OTP is generated by inWebo Application for Mac & PC or inWebo Helium, inWebo server first compares the IP address with the one it knows, then verifies the OTP
- If the OTP is generated by nCode ou inWebo Authenticator, inWebo server ignores the IP address

The string returned is a code which can be “OK” (success) or “NOK:<cause>” (error).

Authentication with REST API

authenticateExtended

authenticateExtended

URL:

https://api.myinwebo.com/FS?action=authenticateExtended + parameters

Mandatory parameters are:

&serviceId= <id of the service> //integer
&userId=<login name> //string
&token=<otp generated> //string

Optional parameters are :

&format=json //allows to get the API response in json format instead of XML format

Response:

By default, the API response is in XML format

The response contains the following information:

- err : the authentication result ("OK" or "NOK:<cause>")
- name: name of the device that performed the authentication
- alias: alias of the device that performed the authentication
- version: version of the device that performed the authentication
- platform: platform of the device (can be helium / windows / mac / android…)
- type: type of the device - can be ma (mobile app) / ca (inWebo application or inWebo Helium / mac (an application using maccess)

If the service ID sent in the URL does not reference a valid service ID, the value of "err" will be "NOK:srv unknown".

If the login name sent in the URL does not match a valid service login name, the value of "err" will be "NOK:account unknown ".

Default response format in XML:

```
<authenticateExtended>
 <err> </err>
 <name> </name>
 <alias> </alias>
 <version> </version>
 <platform> </platform>
 <type> </type>
 <timestamp> </timestamp>
</authenticateExtended>
```

Alternative response format in json:

```
{"timestamp":"","platform":"","alias":"","name":"","err":"","type":"","version":""}
```

2-step Multi-Factor Authentication using a Push request to a Smartphone

This method is used to add a security layer to an existing login / password authentication. From the user end, it is seen as the steps:

1. Standard login / password authentication
2. Then, mult-factor authentication (MFA) on the mobile, woken up by a Push notification

To achieve this, inWebo provides 2 REST Web Services that enable the 2nd step. The 1st call requests inWebo platform to send a Push notification to an identified user's Smartphone. This notification will wake up inWebo Authenticator and prompt for Authorization (using a PIN or not). The 2nd call allows your server to verify whether MFA for this session was successful or not.

As it is not possible to predict how long the user will take to authenticate, we implemented an asynchronous procedure:

1. Your server requests inWebo to notify the user's Smartphone, and gets back a session id
2. Your server verifies periodically with inWebo platform the authentication result of the session

PushAuthenticate

URL:

https://api.myinwebo.com/FS?action= pushAuthenticate + parameters

Mandatory parameters are:

&serviceId= <id of the service> //integer
&userId=<login of the previsously authenticated user> //string

Optional parameters are:

&format=json // allows to get the API response in json format instead of XML format

Response:

By default, API response is in XML. It contains:

- err : the notification result ("OK" or "NOK:<cause>")
- name: name of the device that performed the authentication
- alias: alias of the device that performed the authentication
- version: version of the device that performed the authentication
- platform: platform of the device (can be iphone, android, wp8 in case of Authenticator)
- type: will be "ma" (mobile app) or "mac" (your application developed with mAccess)
- sessionId: the session Id that will allow you to check Authentication result (to be used in checkPushResult)

Possible error codes:

- NOK:NoPush: user's mobile app does not support Push (inWebo nCode)
- NOK:NoMA: user does not have any inWebo mobile app
- NOK:NOLOGIN : user does not exist

- NOK:SN: syntax error in input parameters
- NOK:access forbidden: serviceId is wrong

JSON response format:

```
{"timestamp":"","platform":"","sessionId":"","alias":"","name":"","err":"","type":"","version":""}
```

CheckPushResult

This call is used to get the authentication result of a specific session

URL :

https://api.myinwebo.com/FS?action= checkPushResult + paramètres

Mandatory parameters are:

&serviceId= <id du service> //integer
&sessionId=<id de session> //string
&userId=<login> //string

Optional parameters are:

&format=json // allows to get the API response in json format instead of XML format

Response:

By default, API response is in XML. It contains:

- err : the authentication result (“OK” or “NOK:<cause>”)
- name: name of the device that performed the authentication
- alias: alias of the device that performed the authentication
- version: version of the device that performed the authentication
- platform: platform of the device (can be iphone, android, wp8 in case of Authenticator)
- type: will be “ma” (mobile app) or “mac” (your application developed with mAccess)

Possible error codes:

- NOK:WAITING: Request is pending, try again later (in 0.5 sec for instance)
- NOK:REFUSED: user refused authentication
- NOK:TIMEOUT: user did not authenticate in time (1 minute)
- NOK:SN: syntax error in input parameters
- NOK:access forbidden: serviceId is wrong
- NOK: sessionId does not exist or has expired

JSON response format:

```
{"timestamp":"","platform":"","alias":"","name":"","err":"","type":"","version":""}
```

Transaction sealing with REST API

sealVerify

URL:

https://api.myinwebo.com/FS?action=sealVerify + parameters

Mandatory parameters are:

&serviceId= <id of the service> //integer
&userId=<login> //string
&token=<OTP received from the client app> //string
&data=<sealed data> //string

Optional parameters are:

&format=json // allows to get the API response in json format instead of XML format

Response:

By default, response is in XML. It contains:

- err: operation result (« OK » or « NOK:<cause> »)
- name: name of the device that performed the sealing
- alias: alias of the device that performed the sealing
- version: version of the device that performed the sealing
- platform: platform of the device (the one you passed to mAccess)
- type: will be "mac" (your application developed with mAccess)

Si l'id du service passé dans l'appel ne représente pas un id de service valide, la valeur de « err » sera « NOK:srv unknown ».

Si le login passé dans l'appel ne représente pas un login valide du service, la valeur de « err » sera « NOK:account unknown ».

Possible error codes:

- NOK:srv unknown: serviced is wrong
- NOK:account unknown: login does not exist
- NOK:NoKey: user does not have a sealing key. This means that he has not activated your app
- NOK:BadData: The data sealing in the OTP is different from the one sent in as an input parameter. Potential MITM.
- NOK:FORBIDDEN : Sealing option is not authorized for your account. Contact inWebo Sales.

Format of XML response:

```
<authenticateExtended>
 <err> </err>
 <name> </name>
 <alias> </alias>
 <version> </version>
 <platform> </platform>
 <type> </type>
 <timestamp> </ timestamp >
</authenticateExtended>
```

Format of JSON response:

```
{ "timestamp":"","platform":"","alias":"","name":"","err":"","type":"","version":""}
```

User Management with SOAP API

loginsQuery

```
loginsQuery(long userid, long serviceid, long offset, long nmax, long sort)
```

Returns:

```
LoginsQueryResult
```

This is an object with the following properties:

```
public String err;
public int n;
public long[] id;
public long[] count;
public String[] login;
public String[] code;
public long[] status;
public long[] role;
public String[] firstname;
public String[] name;
public String[] mail;
public String[] phone;
public String[] extrafields;
public long[] createdby;
```

This function returns the list of logins for the requested Service.

Parameter “n” returns the number of listed entries.

Possible values for “code”:

- “ok”: The code has been used. This login should be considered as active.
- “expired”: The code was never used and has expired. There is no more valid activation code for this login.
- “in:<9-digit code>”: The code has not been used yet, and is inactive. Use loginActivateCode to switch the code to “active” state (see loginCreate with codetype=1)
- “link”: The code has not been used yet, an activation link (valid for 3 weeks) was sent to the email address given at loginCreate time (see loginCreate with codetype=2)
- “<9-digit code>”: The code is valid at present time and has not been used yet

Parameters “offset” and “nmax” are used to page requests: “offset” is the first line of the page to request (can start at 0), “nmax” is the number of lines per page.

Parameter “sort” can be set to:

- 0: no sorting
- 1: sort by login (ascending)
- 2: sort by login (descending)
- 3: sort by name (ascending)
- 4: sort by name (descending)
- 5: sort by mail (ascending)
- 6: sort by mail (descending)

Parameter “count” returns the total number of logins, whereas parameter “n” returns the number of logins of the request.

Parameter “createdby” is used to distinguish logins created by inWebo Aministration console from logins created by the API. Possible values:

- 0: login created by WebConsole
- 1: login created by API

Parameter “extrafields” is reserved for future use.

loginQuery

```
loginQuery(long userid, long loginid)
```

Returns:

```
LoginQueryResult
```

This is an object with the following properties:

```
public String err;
public String login;
public String code;
public long status;
public long role;
public String firstname;
public String name;
public String mail;
public long createdby;
public long lastauthdate;     // timestamp of the last successful authentication (0 if none)
public long nca;              //no. of inWebo applications or inWebo Helium
public long caid[];           //id of the tool
public long castate[];        //state of the tool (0:active, 1:locked; 2:pin locked)
public long caname[];         //name of the tool
public long caalias[];        //alias of the tool (can be correlated with authenticateExtended)
public long cault[];          //type of tool (0 : application; 1:Helium)
public long nma;              //no. mobile applications
public long maid[];           //id of the mobile application
public long mastate[];        //state of the mobile application
public long maname[];         //name of the mobile application
public long maalias[];        //alias of the mobile application
public long nmac;             //no. maccess
public long macid[];          //id of the maccess
public long macstate[];       //state of the maccess
public long macname[];        //name of the maccess
public long macalias[];       //alias of the maccess
```

This function returns attributes of a particular login.

Parameter “role” defines the rights for this login:

- 0: user

- 1: manager of the service (can create, modify and delete users)
- 2: administrator of the service (can also modify parameters of the service in the Administration Console)

Parameter "status":

- 0: login is active
- 1: login is blocked (authentication requests will be rejected)

loginSearch

loginSearch(**long** userid, **long** serviceid, **String** loginname, **long** exactmatch, **long** offset, **long** nmax, **long** sort)

Returns:

LoginSerachResult

This is an object with the following properties:

```
public String err;
public int n;
public long[] id;
public String[] login;
public String[] code;
public long[] status;
public long[] role;
public String[] firstname;
public String[] name;
public String[] mail;
public String[] phone;
public long[] activation_status;
```

This function allows looking for a user based on its login (loginname).

Parameter "exactmatch" can be:

- 0: The search request will return all the logins containing the string 'loginname'
- 1: The search request will return the login exactly matching 'loginname'

The return parameter "activation_status" is a mask:

- 1: The user has at least one nCode active (and not blocked)
- 2: The user has at least one Toolbar active (and not blocked)

Usage of parameters "offset", "nmax" and "sort" are identical to loginsQuery.

loginCreate

loginCreate(**long** userid, **long** serviceid, **String** login, **String** firstname, **String** name, **String** mail, **String** phone, **long** status, **long** role, **long** access, **long** codetype, **String** lang, **String** extrafields)

Returns:

LoginCreateResult

This is an object with the following properties:

```
public String err;
public String code;
public long id;
```

This function creates a login for the requested service and generates an activation code, which can be either retrieved in the "code" return parameter, or sent by Email using loginSendByMail.

This activation code should be entered by the end-user in one of the inWebo authentication tools. The tool will then become activated for this service, and available to generate OTPs for this service.

The behavior depends on the "codetype" parameter:

- 0: An activation code is generated, valid immediately for 15 minutes
- 1: An "inactive" activation code, valid for 3 weeks, is generated (it will become active later on, thanks to loginActivateCode)
- 2: An activation link, valid for 3 weeks, is generated. LoginSendByMail must be used immediately after

Parameter "lang" can be "fr" or "en".

- With inWebo Identity Guard, you can only use codetype=0
- With inWebo Safe Transactions, you can use codetype=0 or 1
- With inWebo Enterprise, you can use codetype=0, 1 or 2

Parameter "access" can be:

- 0: service bookmarks are not associated to this user
- 1: all service bookmarks are associated to this user

loginCreate will return "NOK:full" if the maximum number of users for the service has been reached, and "NOK:loginexists" if the login already exists.

loginUpdate

```
loginUpdate(long userid, long serviceid, long loginid, String firstname, String name, String mail, String phone, long status, long role, String extrafields)
```

Returns:

```
loginUpdateResponse
```

This is an object with the following properties:

```
public String loginUpdateReturn
```

This function allows to update a service user (a login).

The string returned is a result code which can be "OK' (success) or "NOK:<cause>" (error). The function will return "NOK:login already used" if you tried to update the login name and that login name already exists.

loginSendByMail

```
loginSendByMail(long userid, long serviceid, long loginid)
```

Returns:

```
loginSendByMailResponse
```

This is an object with the following properties:

```
public String loginSendByMailReturn
```

This function sends the activation code to the requested login per email. The login must have a valid email address previously configured with loginCreate or loginUpdate.

The string returned is a result code which can be “OK”(success) or “NOK” (error).

Available only with In-Webo Enterprise.

loginActivateCode

```
loginActivateCode(long userid, long serviceid, long loginid)
```

Returns:

```
loginActivateCodeResponse
```

This is an object with the following properties:

```
public String loginActivateCodeReturn
```

The string returned can be “NOK” (error) or the inWebo activation code of the login.

To be used with logins created with codetype=1.

Available only with inWebo Safe Transactions or Enterprise.

loginDelete

```
loginDelete(long userid, long serviceid, long loginid)
```

Returns:

```
loginDeleteResponse
```

This is an object with the following properties:

```
public String loginDeleteReturn
```

This function allows to delete a service user (a login).

The string returned is a result code which can be “OK” (success) or “NOK” (error).

loginResetPwd

```
loginResetPwd(long userid, long serviceid, long loginid)
```

Returns:

```
loginResetPwdResponse
```

This is an object with the following properties:

```
public String loginResetPwdReturn
```

This function allows to generate a recovery code a user will be able to use to reset its password from an inWebo device that has been blocked by 3 erroneous PIN codes or passwords.

The string returned is a result code which can be “OK” (success) or “NOK:<cause>” (error).

NB: If the user has more than one service, only the current service will be unlocked. Other services, including My inWebo, will not be available.

Available only with inWebo Safe Transactions or Enterprise.

loginResetPwdExtended

```
loginResetPwdExtended(long userid, long serviceid, long loginid, long codetype)
```

Returns:

```
loginResetPwdExtendedResponse
```

This is an object with the following properties:

```
public String loginResetPwdExtendedReturn
```

This function allows to generate a recovery code a user will be able to use to reset its password from an inWebo device that has been blocked by 3 erroneous PIN codes or passwords.

Compared to loginResetPwd, this function adds a ‘codetype’ parameter with 3 possible values:

- 0: An unlock code is generated, valid immediately for 15 minutes
- 1: An “inactive” unlock code, valid for 3 weeks, is generated (it will become active later on, thanks to loginActivateCode)
- 2: An unlock link, valid for 3 weeks, is generated. LoginSendByMail must be used immediately after

The string returned is a result code which can be “OK” (success) or “NOK:<cause>” (error).

NB: If the user has more than one service, only the current service will be unlocked. Other services, including My inWebo, will not be available.

Available only with inWebo Safe Transactions or Enterprise.

loginResetPINErrorCounter

```
loginResetPINErrorCounter(long userid, long serviceid, long loginid)
```

Returns:

```
loginResetPINErrorCounterResponse
```

This is an object with the following properties:

```
public String loginResetPINErrorCounterReturn
```

This function allows to reset the user PIN code or password error counter. It can be used only once for a given user, until he successfully authenticates again.

The string returned is a result code which can be “OK” (success) or “NOK:<cause>” (error).

If this code means an error occurs and is equal to “NOK:MULTI”, it means that this user (login) is associated to other services that do not allow to use this function.

Available only with inWebo Safe Transactions or Enterprise.

loginRestore

```
loginRestore(long userid, long serviceid, long loginid)
```

Returns:

```
loginRestoreResponse
```

This is an object with the following properties:

```
public String LoginRestoreReturn
```

This function allows to generate a code for a user to restore its access to the service, for example if he has blocked or lost all his inWebo authentication devices.

When the restore code is used, all other existing inWebo devices will be deleted.

Only the access to YOUR service will be restored. Accesses to other services will have to be restored one by one by contacting each service provider.

The string returned is a result code which can be “OK” (success) or “NOK” (error).

Available only with inWebo Safe Transactions or Enterprise.

loginAddDevice

```
loginAddDevice(long userid, long serviceid, long loginid, long codetype)
```

Returns:

```
loginAddDeviceResponse
```

This is an object with the following properties:

```
public String loginAddDeviceReturn
```

This function allows to generate a code a user will use to activate the device of its choice.

The string returned is a result code which can be “OK” (success) or “NOK:<cause>” (error).

If this code means an error occurs and is equal to:

- “NOK:NoPassword”: it means that this user’s password is empty

NB: If the user has more than one service, only the current service will be activated. Other services, including My inWebo, will not be available.

Available only with inWebo Safe Transactions or Enterprise.

loginDeleteTool

login*DeleteTool*(**long** userid, **long** serviceid, **long** *tool*id, ***String*** *tooltype*)

Returns:

loginAddDeviceResponse

This is an object with the following properties:

public String loginAddDeviceReturn

This function allows to delete a tool for a given user. Tool ID and Tool Type can be obtained by using loginQuery. Type can be ‘ma’ (nCode / Authenticator), ‘ca’ (inWebo for PC / Helium) or ‘mac’ (mAccess).

The string returned is a result code which can be “OK” (success) or “NOK:<cause>” (error).

Available only with inWebo Safe Transactions or Enterprise.