Client Portfolio API

Lursoft IT

18.10.2022

Table of contents

Version	Content	
31.08.2021	Translation in English	L. Jirgena
25.01.2022	Added parameters client_id, form	D. Reinmanis
18.10.2022	Updated documentation	S. Zeiļa

Description

The Client Portfolio API is a SOAP web service that provides the most frequently used and important Client Portfolio functionality.

WSDL service description is available at https://www.klientuportfelis.lv/api/wsdl

To use API, you need a Client Portfolio account with API usage rights and an SSL key provided by Lursoft. At the beginning of the work session, you must log in to the system by calling API. *Login* function This function returns a session key, which must be added to requests for all other functions in current session. In addition to security, requests must be signed with the SSL private key assigned to you, and the signature must be added to the requests as *signature* variable.

Generating requests signature

Client Portfolio API requests are signed by adding a specific string of characters to the SOAP request. To create a signature, you must create **a JSON format string** that contains the variables in the request. They must be listed in the order which they appear in the description of each function.

Important! Follow these instructions for formatting a variable string:

- ◦ attributes (keys) must be enclosed in double quotation marks (“)
- ◦ values – character strings – must be enclosed in double quotation marks (“)
- ◦ numbers and bool values must be given without quotes
- ◦ Latvian alphabet symbols must be converted according to JSON requirements
- ◦ whitespaces and newline symbols may not be included in a string. This statement does not apply to the values of request variables

The example shows the request XML variables and their corresponding JSON string:

XML variables:

```
<foo>Hello World!</foo>
<bar>true</bar>
<baz>3.14</baz>
<qux>Jānis Bērziņš</qux>
```

JSON string:

```
{“foo”:”Hello World!”,”bar”:true,”baz”:3.14,”qux”:”J\u0101nis B\u0113rzi\u0146\u0161”}
```

The resulting string must be signed with the **SSL private key** assigned to you using the **SHA-256** algorithm.

In order for the obtained signature to be transmitted in the HTTP protocol, it must be encoded in **Base64** encoding. The resulting character string must be added to the XML request as a *<signature>* element. This element must be the first in the request. The authorization procedure with examples is described below. The sequence of signing operations is identical to all other API functions. **A signature must be attached to each request.**

Authorization procedure

The initial XML request for *login* function looks like this:

```
<?xml version=”1.0”?>
<soap:Envelope xmlns:soap=”http://schemas.xmlsoap.org/soap/envelope/”
soap:encodingStyle=”http://www.w3.org/2001/12/soap-encoding”>
    <soap:Body>
        <login>
            <username>user_name</username>
            <password>password</password>
            <role_id>1234</role_id>
        </login>
    </soap:Body>
</soap:Envelope>
```

An SSL signature must be attached to this request. To do this, you need to create a JSON string with the request variables *username*, *password* and *role_id*. JSON string will look like this-

```
{“username”:”user_name”,”password”:”password”,”role_id”:1234}
```

The result string must be signed with the SSL private key assigned to you using the SHA-256 algorithm, and the resulting binary signature must be encrypted in Base64 encoding. String variable must be added to the XML request as a <signature> element. This element must be the first in the request. The result is the following XML request-

```
<?xml version=”1.0”?>
<soap:Envelope xmlns:soap=”http://schemas.xmlsoap.org/soap/envelope/”
soap:encodingStyle=”http://www.w3.org/2001/12/soap-encoding”>
    <soap:Body>
        <login>
            <signature>
PfjPgL9scoPLIA2niVMsQgJoCeqkIRtE6MISBekHuWAMVGR1kCUcsCY2KgrSfwpco6WCRUHBteLx4Lyp4vJpVaZ
iTvkQjCK21v1keidQqzRoDziMzjaghO0ZkCuxJ+zdINFwGWGxTVIl9/+WstaNDhx+OdYb5UUItYixUKEoCfRAHQ
YjTtw/zvkkFjjoGfOwIrx7BtfpTZwDsT5FVR6opvLo7iSMxtJtjhfHt5Lqgmcy3TG+MimlXV1Mlklia40Bluy8b
ND4B6C0fGLVoLSHhBmMgk5fI4CB42B2hWBT+dRxzJIunR/0b080Nr6tLbxlKdyDHvAAV5dxsMD5g4Osdw==</si
gnature>
            <username>user_name</username>
            <password>password</password>
            <role_id>1234</role_id>
        </login>
    </soap:Body>
</soap:Envelope>
```

As a result of a successful login, the Client Portfolio API server will return the following response:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope/”
xmlns:ns1=”urn:ApiControllerwsdl” xmlns:xsd=”http://www.w3.org/2001/XMLSchema”
xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance” xmlns:SOAP-
ENC=”http://schemas.xmlsoap.org/soap/encoding/” SOAP-
ENV:encodingStyle=”http://schemas.xmlsoap.org/soap/encoding/”>
    <SOAP-ENV:Body>
        <ns1:loginResponse>
            <return xsi:type=”ns1:LoginResponse”>
                <sid xsi:type=”xsd:string”>7dde835d8f06bb20e37b286279c7c21c</sid>
                <success xsi:type=”xsd:boolean”>true</success>
                <message xsi:type=”xsd:string”>Autorizācija veiksmīga</message>
            </return>
        </ns1:loginResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The response includes the session key ***sid***, which is required to perform other API functions in current session. In all API functions except login, the session key must also be included in the signed data.

Description of functions

API functions are described with their name, description, parameters and returned data. Mandatory parameters are highlighted in red. When creating a JSON array for signing data, the sequence of variables in the function description must be followed.

User authorization

Function: login

Description: the function performs user authorization and returns the session key (required to perform other API functions). The session key expiration time renews after each request to the API and expires if the session has not been renewed for 30 minutes. Note that the session is associated with an IP address, so any change to the IP address means re-authorization.

Function parameters:

Parameter	Type	Description
signature	string	SSL signature in base64 encoding
username	string	Username used for Client Portfolio login
password	string	Password used for Client Portfolio login
role_id	integer	User role identifier that you will receive from Lursoft

Response data:

Parameter	Type	Description
sid	string	Session key
success	boolean	Notice that request was successfully executed
message	string	Request execution notice
code	int	Error code or null if the request is successful

Termination of a working session

Function: logout

Description: the function terminates the user's work session.

Function parameters:

Parameter	Type	Description
signature	string	SSL signature in base64 encoding
sid	string	Session key

Response data:

Parameter	Type	Description
success	boolean	Notice that request was successfully executed
message	string	Request execution notice
code	int	Error code or null if the request is successful

Obtaining data about user's lists

Function: get_lists

Description: function returns all user Client Portfolio lists.

Function parameters:

Parameter	Type	Description
signature	string	SSL signature in base64 encoding
sid	string	Session key

Response data:

Parameter	Type	Description
list_count	integer	Count of user lists
lists	ResponseList[]	User lists
success	boolean	Notice that request was successfully executed
message	string	Request execution notice
code	int	Error code or null if the request is successful

Obtaining the list and objects in it

Function: get_objects

Description: the function returns the selected list and the objects in it.

Function parameters:

Parameter	Type	Description
signature	string	SSL signature in base64 encoding
sid	string	Session key
list_id	integer	List identifier
start	integer	Number of records to skip (offset)
rows	integer	Number of records to return (limit); maximum value: 25 000

Response data:

Parameter	Type	Description
list	ResponseList	List
success	boolean	Notice that request was successfully executed
message	string	Request execution notice
code	int	Error code or null if the request is successful

Obtaining object

Function: get_object

Description: function returns selected Client Portfolio object.

Function parameters:

Parameter	Type	Description
signature	string	SSL signature in base64 encoding
sid	string	Session key
id	integer	Object identifier

Response data:

Parameter	Type	Description
object	ResponseObjectInfo	Object
success	boolean	Notice that request was successfully executed
message	string	Request execution notice
code	int	Error code or null if the request is successful

Creating a list

Function: add_list

Description: function creates a new Client Portfolio list.

Function parameters:

Parameter	Type	Description
signature	string	SSL signature in base64 encoding
sid	string	Session key
name	string	List name, the maximum lengthis 100 characters.
Type	string	List type. Possible values: • PERSON – list of legal and natural persons • PLEDGES – list of commercial pledges • ADDRESS – list of addresses
description	string	List description

form	string	List form. Possible values: • SIMPLE- the default list format • OUTERID- list of external identifiers

Response data:

Parameter	Type	Description
list	ResponseList	List
success	boolean	Notice that request was successfully executed
message	string	Request execution notice
code	int	Error code or null if the request is successful

Editing a list

Function: update_list

Description: function edits the data of the specified Client Portfolio list.

Function parameters:

Parameter	Type	Description
signature	string	SSL signature in base64 encoding
sid	string	Session key
id	integer	List identifier
name	string	List name, the maximum lengthis 100 characters
description	string	List description

Response data:

Parameter	Type	Description
list	ResponseList	List
success	boolean	Notice that request was successfully executed
message	string	Request execution notice
code	int	Error code or null if the request is successful

Deleting a list

Function: delete_list

Description: function deletes the selected list and the items in it.

Function parameters:

Parameter	Type	Description
signature	string	SSL signature in base64 encoding

sid	string	Session key
id	integer	List identifier

Response data:

Parameter	Type	Description
list_count	integer	Count of user lists
lists	ResponseList[]	User lists
success	boolean	Notice that request was successfully executed
message	string	Request execution notice
code	int	Error code or null if the request is successful

Adding an object to the list

Function: add_object

Description: function inserts a new object in the specified Client Portfolio list.

In cases when a person is added to the lists of legal and natural persons, it is checked in the database of the Register of Enterprises (UR). If the person is in the UR database, an object with the name, registration number and address found in the UR database is added to the Client Portfolio. If it is necessary to add a person to the list who is not in the UR database, the boolean value *FALSE* must be set in the parameter "*check_ur*".

Function parameters:

Parameter	Type	Description
signature	string	SSL signature in base64 encoding
sid	string	Session key
list_id	integer	List identifier
code	string	11-digit registration number of a legal entity, or the personal identification code of a natural person, or address identifier, or commercial pledge number. The maximum length is 20 characters.
Type	string	Object type. Possible values: • COMPANY – legal entity • PERSON – a natural person • ADDRESS – address • PLEDGE – commercial pledge • AUTHORITY – public person or institution

name	string	Object name. The maximum length is 255 characters
check_ur	boolean	Indication of whether to check object in the Register of Enterprises (UR).

Response data:

Parameter	Type	Description
object	ResponseObjectInfo	Object
success	boolean	Notice that request was successfully executed
message	string	Request execution notice
code	int	Error code or null if the request is successful

Editing an object

Function: update_object

Description: function edits the data of the specified object.

Function parameters:

Parameter	Type	Description
signature	string	SSL signature in base64 encoding
sid	string	Session key
id	integer	Object identifier
address	string	Object address. The maximum length is 200 characters
phone	string	Phone number of the object. The maximum length is 200 characters
fax	string	Fax number of the object. The maximum length is 200 characters
email	string	Email of the object. The minimum length is 4 characters and maximum length is 254 characters. Email address is being validated
www	string	Website of the object. The maximum length is 200 characters
contact_person	string	Contact person of the object. The maximum length is 200 characters
client_id	string	Customer identifier. The maximum length is 255 characters

Response data:

Parameter	Type	Description
object	ResponseObjectInfo	List object

success	boolean	Notice that request was successfully executed
message	string	Request execution notice
code	int	Error code or null if the request is successful

Deleting an object

Function: delete_object

Description: function deletes the specified object from all lists containing the selected object or from one list if a list identifier is provided.

Function parameters:

Parameter	Type	Description
signature	string	SSL signature in base64 encoding
sid	string	Session key
id	integer	Object identifier
list_id	integer	List identifier if the item needs to be deleted from only one list

Response data:

Parameter	Type	Description
list_count	integer	Count of user lists
lists	ResponseList[]	User lists
success	boolean	Notice that request was successfully executed
message	string	Request execution notice
code	int	Error code or null if the request is successful

Obtaining monitoring information

Function: get_list_monitoring

Description: function returns list monitoring information.

Function parameters:

Parameter	Type	Description
signature	string	SSL signature in base64 encoding
sid	string	Session key
list_id	integer	List identifier

Response data:

Parameter	Type	Description
monitoring	ResponseMonitoring	List monitoring
success	boolean	Notice that request was successfully executed
message	string	Request execution notice
code	int	Error code or null if the request is successful

Obtaining list monitoring sessions

Function: get_monitoring_sessions

Description function returns information about the list monitoring sessions. You can specify the start and / or end date of the desired period

Function parameters:

Parameter	Type	Description
signature	string	SSL signature in base64 encoding
sid	string	Session key
list_id	integer	List identifier
from	string	Start date of the selected period in ISO 8601 format (YYYY-MM-DD)
to	string	End date of the selected period in ISO 8601 format (YYYY-MM-DD)

Response data:

Parameter	Type	Description
sessions	ResponseSession[]	Monitoring sessions
session_count	int	Count of selected sessions
monitoring_id	int	List monitoring identifier
success	boolean	Notice that request was successfully executed
message	string	Request execution notice
code	int	Error code or null if the request is successful

Obtaining monitoring session events

Function: get_session_events

Description: function returns information about the events selected in one monitoring session.

Function parameters:

Parameter	Type	Description
signature	string	SSL signature in base64 encoding
sid	string	Session key
list_id	integer	List identifier
session_id	integer	Monitoring session identifier

Response data:

Parameter	Type	Description
events	ResponseEvent[]	Monitoring events
success	boolean	Notice that request was successfully executed
message	string	Request execution notice
code	int	Error code or null if the request is successful

Obtaining available services for the list

Function: get_available_services

Description: function returns the codes and names of the subscribed services to the list (only events that have not yet been added to monitoring are returned).

Function parameters:

Parameter	Type	Description
signature	string	SSL signature in base64 encoding
sid	string	Session key
list_id	integer	List identifier

Response data:

Parameter	Type	Description
services	ResponseService[]	Monitoring services
success	boolean	Notice that request was successfully executed
message	string	Request execution notice
code	int	Error code or null if the request is successful

Creating a monitoring

Function: add_list_monitoring

Description: function creates a monitoring for the selected Client Portfolio list. Email addresses and monitoring service codifiers must be specified as array elements. Example:

```
…
<email>
    <item>user1@domain.lv</item>
     <item>user2@domain.lv</item>
</email>
<services>
    <item>SV_BIRTH</item>
    <item>SV_UBIRTH</item>
</services>
…
```

Function parameters:

Parameter	Type	Description
signature	string	SSL signature in base64 encoding
sid	string	Session key
list_id	integer	Identifier of the selected list
email	string[]	Email addresses to send monitoring emails to. The minimum length is 4 and the maximum is 254 characters. Email address is being validated.
Empty_mail	boolean	Indication of whether to send an e-mail if no event is selected in the monitoring session
period	string	Monitoring period. Possible values: • D – day • W – week • M – month. Only one of the allowed values is required.
Start_date	string	Start date of the monitoring period in ISO 8601 format (YYYY-MM-DD)
services	string[]	Codes of selected monitoring services
end_date	string	End date of the monitoring period in ISO 8601 format (YYYY-MM-DD)

Response data:

Parameter	Type	Description
monitoring	ResponseMonitoring	Monitoring
success	boolean	Notice that request was successfully executed
message	string	Request execution notice
code	int	Error code or null if the request is successful

Editing a monitoring

Function: update_list_monitoring

Description: function edits the specified monitoring. Email addresses and monitoring service codes must be specified as array elements. Example:

```
…
<email>
    <item>user1@domain.lv</item>
     <item>user2@domain.lv</item>
</email>
<services>
    <item>SV_BIRTH</item>
    <item>SV_UBIRTH</item>
</services>
…
```

When editing monitoring, the previously selected codes for monitoring services must also be specified.

Function parameters:

Parameter	Type	Description
signature	string	SSL signature in base64 encoding
sid	string	Session key
list_id	integer	Identifier of the selected list
email	string[]	Email addresses to send monitoring emails to
empty_mail	boolean	Indication of whether to send an e-mail if no event is selected in the monitoring session
period	string	Monitoring period. Possible values: • D – day • W – week • M – month. Only one of the allowed values is required.
Start_date	string	Start date of the monitoring period in ISO 8601 format (YYYY-MM-DD)
services	string[]	Codes of selected monitoring services
end_date	string	End date of the monitoring period in ISO 8601 format (YYYY-MM-DD)

Response data:

Parameter	Type	Description
monitoring	ResponseMonitoring	Monitoring
success	boolean	Notice that request was successfully executed
message	string	Request execution notice
code	int	Error code or null if the request is successful

Answer objects

ResponseRole

Parameter	Type	Description
id	int	Role identifier
name	string	Role name
rights	string	Role rights for the list. Possible values: • OWNER – list owner • DELETE – the right to read, edit and delete the list • WRITE – read and edit list • READ – the right to read the list

ResponseObject

Parameter	Type	Description
id	int	Object identifier
code	string	11-digit registration number of a legal entity, or the personal identification code of a natural person, or address identifier, or commercial pledge number.
virtual_id	string	9-digit registration number of a legal entity, or the personal identification code of a natural person, or address identifier, or commercial pledge number.
client_id	string	Customer identifier
Name	string	Object name
status	string	Object status. Possible values: • ACTIVE • DISABLED
created	string	Object creation date in the format “YYYY-MM-DD hh: mm: ss”
updated	string	Object edited in “YYYY-MM-DD hh: mm: ss” format
address	string	Object address

ResponseList

Parameter	Type	Description
id	int	List identifier
name	string	List name
type	string	List type. Possible values: • PERSON – list of legal and natural persons • PLEDGES – list of commercial pledges • ADDRESS – list of addresses
form	string	List form. Possible values: • SIMPLE- the default list format • OUTERID- list of external identifiers
description	string	List description
monitoring_id	int	Identifier of the monitoring attached to the list
object_count	int	Count of objects in the list
objects	ResponseObject[]	Objects in the list
monitored_object_count	int	Count of monitored objects in the list
users	ResponseRole[]	Users who have access to this list

ResponseObjectInfo

Parameter	Type	Description
id	int	Object identifier
code	string	11-digit registration number of a legal entity, or the personal identification code of a natural person, or address identifier, or commercial pledge number.
Virtual_id	string	9-digit registration number of a legal entity, or the personal identification code of a natural person, or address identifier, or commercial pledge number.
Name	string	Object name
status	string	Object status. Possible values: • ACTIVE • DISABLED
created	string	Object creation date in the format “YYYY-MM-DD hh: mm: ss”
updated	string	Object edited in “YYYY-MM-DD hh: mm: ss” format
address	string	Object address
contact_person	string	Contact person
user_address	string	The address of the object entered by the user
phone	string	Phone number
email	string	Email address
fax	string	Fax number
www	string	Website
client_id	string	Customer identifier

ResponseMonitoring

Parameter	Type	Description
monitoring_id	int	Monitoring identifier
list_id	int	List identifier
email	string	Email addresses to send monitoring emails to
period	string	Monitoring period. Possible values: • D – day • W – week • M – month.
Start_date	string	Start date of the monitoring period in ISO 8601 format (YYYY-MM-DD)”
end_date	string	End date of the monitoring period in ISO 8601 format (YYYY-MM-DD)
empty_email	boolean	Indication of whether to send an e-mail if no event is selected in the monitoring session
object_count	int	Object count in list
services	ResponseService[]	Selected monitoring services

ResponseService

Parameter	Type	Description
type	string	Service code
name	string	Service name

ResponseSession

Parameter	Type	Description
id	int	Monitoring session identifier
sent	string	Date and time the e-mail was sent in the format “YYYY-MM-DD hh: mm: ss”
sent_events_count	int	Count of monitoring events

ResponseEvent

Parameter	Type	Description
event_type	string	Event codifier (list of all monitoring event codes available https://www.klientuportfelis.lv/resources/Visi_monitoringa_notikumi_ar_kodiem_LV_EN.xlsx)
title	string	Event type
object_name	string	Object name
object_code	string	11-digit registration number of a legal entity, or the personal identification code of a natural person, or address identifier, or commercial pledge number.
Date	string	Event date and time in the format “YYYY-MM-DD hh:mm:ss”
sent_date	string	Date and time the e-mail was sent in the format “YYYY-MM-DD hh: mm: ss”
notes	string	Event notes

Error and HTTP status codes

HTTP status codes

The client portfolio API attempts to return the appropriate HTTP status code for each request.

Code	Text	Description
200	OK	The request has been successfully processed.
400	Bad Request	The server was unable to process the request because incorrect data was submitted. The error code will explain the cause of the problem.
401	Unauthorized	The request was not processed because the user could not be authorized. The error code will explain the reason.
403	Forbidden	The request is correct, but access is denied. The error code will explain the reason.
500	Internal Server Error	The server could not complete the request because of a server error.

Error codes

All API responses contain the attribute “code”, which returns an error code in the event of an error. The table below describes the error codes you may encounter when working with the Client Portfolio API.

Code	Text in Latvian	Description
General errors		
1000	Notika servera iekšēja kļūda	An internal server error occurred. The request could not be processed because of an internal server error. If the error persists, please contact the administration!
1001	Šai lomai nav tiesību izmantot API	The user role is not authorized to use the API. Please contact the administration to solve this problem!
1002	Nav tiesību piekļūt šim sarakstam	You do not have permission to view or make changes to the list provided in your request.
1003	Nav tiesību piekļūt šim objektam *vai* Objektu nevar dzēst no visa portfeļa, jo objekts atrodas sarakstā, uz kuru Jums nav tiesības.	There is no right to view or make changes to the requested object or the object is in several lists and the user does not have the right to delete in all these lists
1004	Sasniegts objektu skaita ierobežojums. Lai palielinātu Klientu portfeļa objektu limitu, lūdzu sazināties ar administrāciju!	The maximum number of objects allowed for the role of the Client Portfolio has been reached. To increase the object limit, please contact the administration!
1005	Lomai nav tiesību veikt konkrēto darbību.	The role does not have permission to perform that action.
Validation errors		
2000	Nav norādīts sesijas ID	Request is missing the required function *'sid'* parameter
2001	Nav norādīti visi parametri	Request is missing one of the required parameters
2002	Nav norādīti autorizācijas rekvizīti	Request is missing one of the required authorization parameters
2003	Autorizācija neveiksmīga	Invalid API username, password, or role identifier provided. For your security, the API does not report exactly which of the values might be incorrect.
2004	Padots nekorekts lomas identifikators	Incorect role identifier. The function expects an int type parameter, but a value is received that does not match the int type definition.
2005	Sesija neeksistē	The session key passed to the function does not exist. You need to re-authorize or enter the correct session key.
2006	Neatbilstoša IP adrese	The passed session key is valid but does not match the IP address from which the authorization was made. Re-authorization

Code	Text in Latvian	Description
		required.
2007	Sesija beigusies	The submitted session key is no longer valid because it has not been renewed for more than 30 minutes. Re-authorization required.
2008	Saraksts nav atrasts	Llist identifier does not exist.
2009	Padots nekorekts saraksta identifikators	Incorect list identifier. The function expects an int type parameter, but a value is received that does not match the int type definition.
2010	Padots nekorekts objekta identifikators	Incorect object identifier. The function expects an int type parameter, but a value is received that does not match the int type definition.
2011	Objekts nav atrasts	Object identifier does not exist.
2012	Sarakstam nevar pievienot šāda tipa objektus	You cannot add this type of item to the list. Only objects of the appropriate type can be added to the list. For example, you cannot add "COMPANY" type objects to the "ADDRESS" type list.
2013	Objekts jau ir sarakstā	The object with the code submitted in the request is already in the selected list.
2014	Uzņēmums ir likvidēts	The object with the requested code has been removed and cannot be added to the selected list. However, if you want to add a liquidated company, you must change the value of "check_ur" to "false".
2015	Uzņēmums ir reorganizēts	The object with the requested code has been reorganized and cannot be added to the selected list. However, if you want to add a reorganized company, you must change the value of "check_ur" to "false".
2016	Uzņēmums nav atrasts	The object with the code submitted in the request does not exist in the Register of Enterprises and cannot be added to the selected list. However, if you want to add this object, you must change the value of "check_ur" to "false".
2017	Sarakstā nav neviena objekta, kuru monitorēt	No objects have been added to the list provided in the request, so it is not possible to create / edit monitoring.
2018	Šim sarakstam nav izveidots	No monitoring has been created for the list

Code	Text in Latvian	Description
	monitorings	submitted in the request, so it is not possible to correct the monitoring.
2019	Objekts neatrodas izvēlētajā sarakstā	The requested object is not in the selected list. It is necessary to check whether the submitted object and list identifiers are correct.
2020	Dažādi validācijas kļūdu paziņojumi	The parameters provided in the request do not comply with the length limit or the allowed values of the parameter. Information on the parameter values and the maximum permitted length is available in the parameter description of each function.
2021	Sarakstā ir vairāk objektu kā norādītajā limitā	Requested list objects with the get_objects method and the limit specifies a larger number of returned objects than the maximum possible (25,000) or the limit (parameter rows) is not specified and the list contains more than 25,000 objects.
Signature errors		
3000	Šai lomai nav uzģenerēts SSL sertifikāts	No private SSL key generated for user role. Please contact the administration to solve this problem!
3001	Nav norādīts SSL paraksts	Request did not contains SSL signature, which is a required parameter for the function.
3002	Dati neatbilst parakstam	The request does not match the subordinate signature. Read more about creating a signature in the section "Generating requests signature ".

Example of API signature creation in PHP

```php
<?php

// the variable that will store the signature used by the API
$signatureString = '';

// an empty variable that will contain the binary signature
$signature = null;

// function "login" request data
$loginData = array(
    'username' => 'user_name',
    'password' => 'password',
    'role_id' => 123
```

```
);

// Creating a JSON string
// documentation: http://php.net/manual/en/function.json-encode.php
$loginString = json_encode($loginData);

// Obtaining private key from key file
// documentation: http://php.net/manual/en/function.openssl-pkey-get-
private.php
$privateKey = openssl_pkey_get_private('file://path_to_file/private_key.pem');

// Signing request data
// documentation: http://php.net/manual/en/function.openssl-sign.php
openssl_sign($loginString, $signature, $privateKey, 'SHA256');

// Deleting private key from memory
// documentation: http://php.net/manual/en/function.openssl-free-key.php
openssl_free_key($privateKey);

// Encoding in  Base64
// this must be done so that the binary signature can be sent as text
// documentation: http://php.net/manual/en/function.base64-encode.php
$signatureString = base64_encode($signature);

// the signature is usable in API
echo $signatureString;
```