

API Pārvaldnieka publicētāju un izstrādātāju portāls
Skaidrojumi un izmantošana

v2.7

API Publicētāju un API Izstrādātāju portāls - Instrukcija un skaidrojumi

Šī dokumenta mērķis ir sniegt API Publicētāju (Publisher) un API Izstrādātāju (Devportal) portālu sadaļu un lauku skaidrojumus API Pārvaldnieka lietotājiem, kas publicē vai abonē API programmas saskarnes, norādes par servisu testēšanu izmantojot Postman, kā arī piekļuves talona iegūšanu izmantojot sertifikātu.

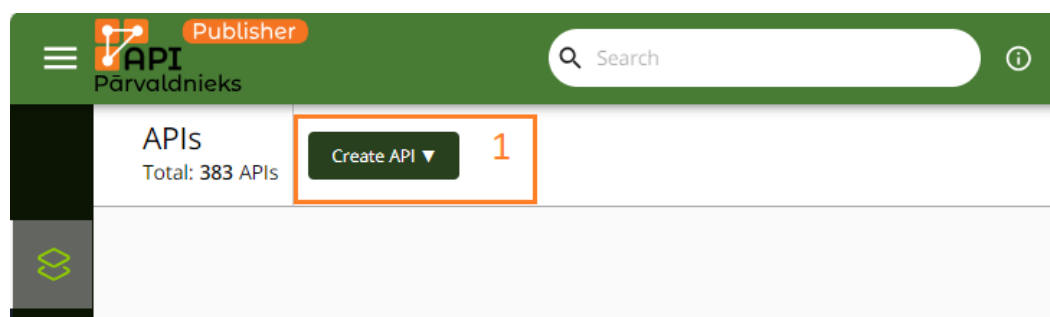
Kopā ar šo dokumentu aicinām izmantot Datu apmaiņas izveides vadlīniju dokumentu, kas ir pieejams <https://www.vdaa.gov.lv/lv/api-parvaldnieks> sadaļā "**Rokasgrāmatas un vadlīnijas**", kur ir aprakstītas datu apmaiņas iespējas, to veidošana, izsaukumu piemēri un citi ieteikumi, kā arī informācija, kura sniedz atbalstu datu apmaiņu izstrādātājiem.

1. API Publicētāju portāls

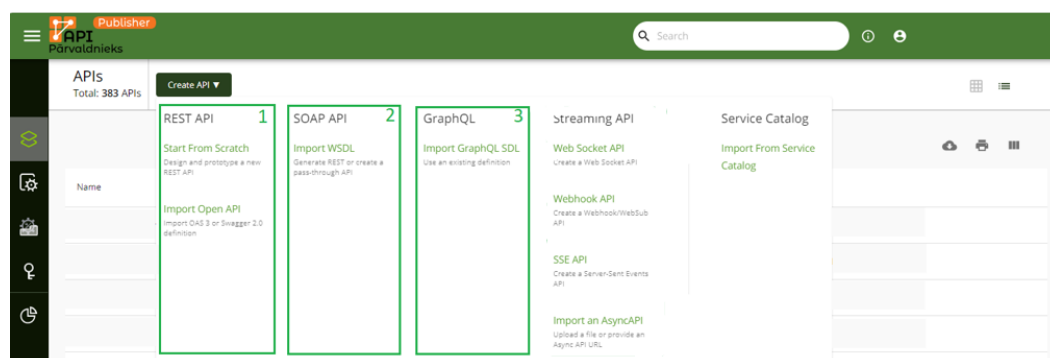
API Publicētāju (Publisher) portāls paredzēts API reģistrēšanai, publicēšanai un pārvaldībai. Portāls lietotājam tiek attēlots valodā, kāda ir iestatīta interneta pārlūkprogrammā.

1.1 Sadaļa "Create API"

Pievienot jaunu API Publicētāju portālā iespējams sadaļā "CREATE API":



Izvēlas veidu, kā pievienot API, atzīmējot vienu no trim opcijām:



REST API

- a. *Start From Scratch* – izvēlas gadījumā, ja publicētājam nav gatavas Swagger definīcijas un API apraksts tiks veidots no jauna, lauku nosaukumus un definīcijas jāveido atbilstoši aktuālākai [Datu apmaiņas izveides vadlīniju](#) dokumenta versijai;

Create an API
Create an API by providing a Name, a Version, a Context and Backend Endpoint (optional)

1 Authority*
VARAM.VRAA

2 Name*
MansApiServiss

3 Context*
API-VARAM_VRAA-MansApiServiss
API will be exposed in (context)/v1_0 context at the gateway

4 Version*
v1_0

5 Endpoint
https://google.com

* Mandatory fields

Create Create & Publish Cancel

Servisa reģistrācijas skatā attēlojas šādi lauki, kurus ir nepieciešams aizpildīt un daļēji tie aizpildās automātiski:

- Authority - Iestādes identifikators, kurš jānorāda tādā pat formātā kā ir klasifikatoru katalogā, piemēram, VARAM.VRAA
- Name – Servisa nosaukums, kurš nevar saturēt speciālos simbolus
- Context – Lauks aizpildās automātiski, ar informāciju no “Servisa veids”+”Authority”+”Name”. Šajā gadījumā API-VARAM_VRAA-MansApiServiss
- Version – Versijai jābūt norādītai noteiktā formāta notācijā – “v{Major}_{Minor}”, piemēram, v1_0
- Endpoint – Jāievada servisa galapunkts (Endpoint)

Servisa pilnais nosaukums būs tāds pats, kā šajā skatā redzamais ‘Context’

- b. Import Open API** – izvēlas gadījumā, ja API Swagger definīcija jau ir iepriekš izveidota un to ir nepieciešams importēt API Publisher portālā.

Lai izveidotu un publicētu REST API, jāpievieno Swagger, augšupielādējot API Swagger definīcijas .json failu vai norādot API galapunkta URL izvēloties attiecīgo punktu “Create an API using an OpenAPI definition” solī.

Swagger definīcija ir formāts, kādā apraksta REST API.

Create an API using an OpenAPI definition.

Create an API using an existing OpenAPI definition file or URL.

1 Provide OpenAPI 2 Create API

* Input Type

☒ OpenAPI URL

☐ OpenAPI File/Archive

OpenAPI URL

Enter OpenAPI URL

Click away to validate the URL

Cancel Next

Pēc swagger definīcijas augšupielādes vai API galapunkta URL ievades un pārejas uz nākamo soli, attēlosies servisa reģistrācijas skats, kura aizpildīšana aprakstīta iepriekš.

SOAP API – izvēlas gadījumā, ja ir iepriekš izveidots SOAP galapunkts.

Import WSDL

Expose a SOAP Service as a REST API

Expose an existing SOAP service as a REST API by importing the WSDL of the SOAP service.

1 Provide WSDL 2 Create API

* Implementation Type

☒ Pass Through

☐ Generate REST APIs

* Input Type

☒ WSDL URL

☐ WSDL File/Archive

WSDL URL

<https://www.crcind.com/csp/samples/SOAP.Demo.CLS?WSDL=1> ✓

Click away to validate the URL

Cancel Next

Lai izveidotu API ar SOAP galapunktu, ir jāizvēlas SOAP servisa Implementācijas tipu, kas varētu būt vai Pass Through (API Pārvaldnieks kā SOAP Servisa vārteja) vai Generate Rest APIs (API Pārvaldnieks eksponē SOAP servisu uz āru kā REST API).

- Izvēloties “Pass Through” – tiek izveidots proxy SOAP pieprasījumiem, kas iet uz API Gateway.

- Izvēloties “*Generate Rest APIs*” – tiek izveidota REST API definīcija no norādītās WSDL URL. API definīcijas metodes tiek ielasītas automātiski no WSDL saites, taču API definīciju iespējams rediģēt atbilstoši [Open API](#) specifikācijai. REST API metodes sasaistītas ar atbilstošajām SOAP operācijām, izmantojot Swagger lauku **x-wso2-soap**. Piemēram:

/Endpoint:

post:

operationId: EndpointOperation

parameters:

...

x-wso2-soap:

soap-action: " https://www.crcind.com/csp/samples/SOAP.Demo.CLS"

soap-operation: SoapEndpointOperation

namespace: " https://www.crcind.com/csp/samples "

x-soap-version: "1.2"

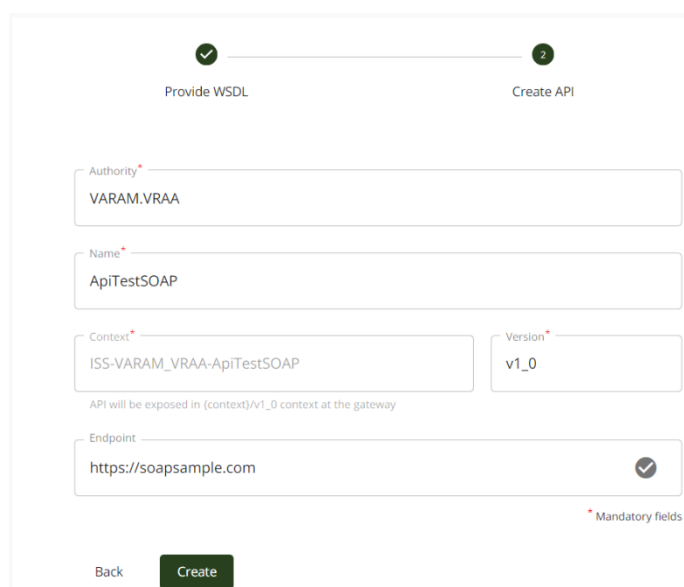
Apraksts SOAP API publicēšanai:

[Expose a SOAP service as a REST API](#)

[Generate REST API from SOAP Backend](#)

[Create and Publish a SOAP API](#)

Nākamajā solī tiks attēlots servisa reģistrācijas skats:



The screenshot shows a two-step registration process. Step 1, 'Provide WSDL', is completed. Step 2, 'Create API', is active. The form contains the following fields:

- Authority**: VARAM.VRAA
- Name**: ApiTestSOAP
- Context**: ISS-VARAM_VRAA-ApiTestSOAP
- Version**: v1_0
- Endpoint**: https://soapsample.com

A note states: 'API will be exposed in {context}/v1_0 context at the gateway'. A 'Back' button and a 'Create' button are at the bottom. A legend indicates that fields with an asterisk (*) are mandatory.

Servisa reģistrācijas skatā attēlojas šādi lauki, kurus ir nepieciešams aizpildīt un daļēji tie aizpildās automātiski:

- **Authority** - Iestādes identifikators, kurš jānorāda tādā pat formātā kā ir klasifikatoru katalogā, piemēram, VARAM.VRAA
- **Name** – Servisa nosaukums, kurš nevar saturēt speciālos simbolus
- **Context** – Lauks aizpildās automātiski, ar informāciju no “Servisa veids”+”Authority”+”Name”. Šajā gadījumā ISS-VARAM_VRAA-ApiTestSOAP
- **Version** – Versijai jābūt norādītai noteiktā formāta notācijā – “v{Major}_{Minor}”, piemēram, v1_0
- **Endpoint** – Jāievada servisa galapunkts (Endpoint)

Servisa pilnais nosaukums būs tāds pats, kā šajā skatā redzamais ‘Context’

GraphQL - API vaicājumu valoda, kas strādā ar jūsu esošajiem datiem. **GraphQL** nodrošina saprotamu API datu aprakstu un pieprasījumos sniedz klientiem iespēju pieprasīt un saņemt tieši tos datus no servisa, kas viņiem nepieciešami.

Import GraphQL SDL ļauj pievienot API Pārvaldniekā servisu aprakstus kas izmanto GraphQL vaicājumu valodu. SDL datnēm jābūt *.graphql* paplašinājumā un viņu saturam jāatbilst *text/plain* formātam.. SDL shēmas apraksts un cita vispārīga informācija ir pieejama noderīgo saišu apakšsadaļā.

Pirmajā solī nepieciešams augšupielādēt GraphQL SDL definīciju:

Nākamajā solī tiks attēlots servisa reģistrācijas skats:

Servisa reģistrācijas skatā attēlojas šādi lauki, kurus ir nepieciešams aizpildīt un daļēji tie aizpildās automātiski:

- Authority - Iestādes identifikators, kurš jānorāda tādā pat formātā kā ir klasifikatoru katalogā, piemēram, VARAM.VRAA
- Name – Servisa nosaukums, kurš nevar saturēt speciālos simbolus
- Context – Lauks aizpildās automātiski, ar informāciju no “Servisa veids”+”Authority”+”Name”. Šajā gadījumā GQL-VARAM_VRAA-ApiTestGQL
- Version – Versijai jābūt norādītai noteiktā formāta notācijā – “v{Major}_{Minor}”, piemēram, v1_0
- Endpoint – Jāievada servisa galapunkts (Endpoint)

Servisa pilnais nosaukums būs tāds pats, kā šajā skatā redzamais 'Context'

GraphQL servisu apraksts un noderīgas saites:

<https://graphql.org/learn/>

<https://graphql.org/learn/schema/>

1.2 Sadaļa "Overview"

Vispārīgā informācija

Standarta lauku vērtības tiek ielasītas no pievienotās Swagger definīcijas, taču līdz API publicēšanai tās ir iespējams rediģēt. Nosaukuma, Konteksta, Versijas laukos ievadītā informācija pēc API publicēšanas nav maināma, līdz ar to jau sākotnēji reģistrējot API, **nosaukuma veidošanai un versionēšanai ir jāizmanto principi, kas aprakstīti aktuālākajā [Datu apmaiņas izveides vadlīniju](#) dokumentā!**

API-VARAM_VRAA-Example :v1_0 CREATED State

Overview

Develop

Endpoint

Business Plan

Deploy

Test

Publish

Metadata

Description

Provider

Context: /API-VARAM_VRAA-Example

Version: v1_0

Type: HTTP

Created Time: A few seconds ago

Last Updated Time: A few seconds ago

Business Owner

Technical Owner

Configuration

Transports: HTTP, HTTPS

API Security: OAuth2

Access Control: None

Workflow Status

Visibility on Developer Portal: Public

Business Plans: Unlimited

Tags

Endpoints

Sandbox Endpoint

Sandbox Endpoint

Endpoint Security

Resources

/* GET PUT POST DELETE PATCH

Show More

API-VARAM.VRAA-TESTApiForTestsWithoutScope :... CREATED State

Overview

Develop

Endpoint

Business Plan

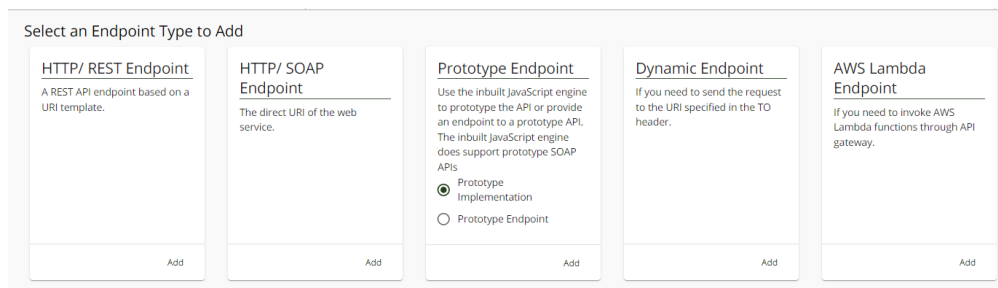
Deploy

Test

Publish

Metadata

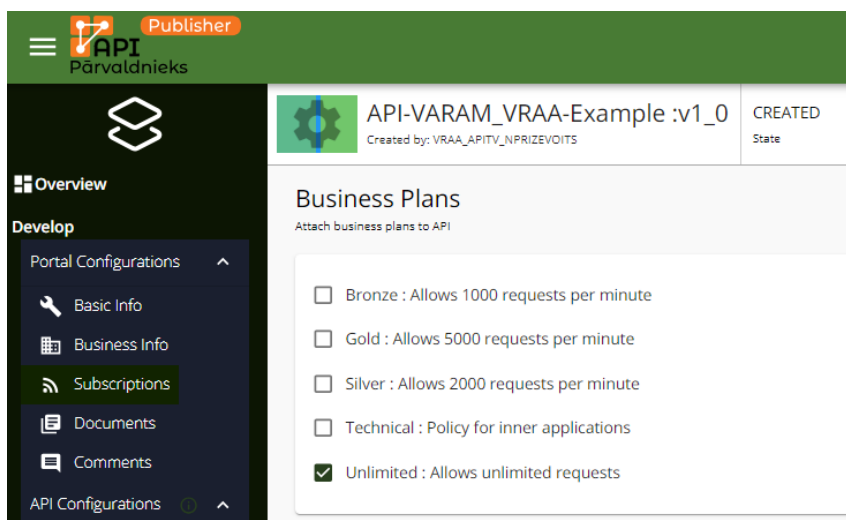
Izvēlnē Develop (a) nav pieejama rediģēšanai, jo tā tiek aizpildīta pirmajā solī. Pievienojot jau eksistējošu API, API metodes tiek norādītas automātiski, jo tie aprakstītas Swagger definīcijā. Papildus tam, galapunktu iestatījumus var rediģēt sadaļā *Endpoint*. Ja API tiek veidots bez Swagger definīcijas, rediģēšanai Develop solī obligāti jāizmanto *Endpoint(b)* izvēlnē, lai varētu veikt metožu un galapunktu konfigurāciju.



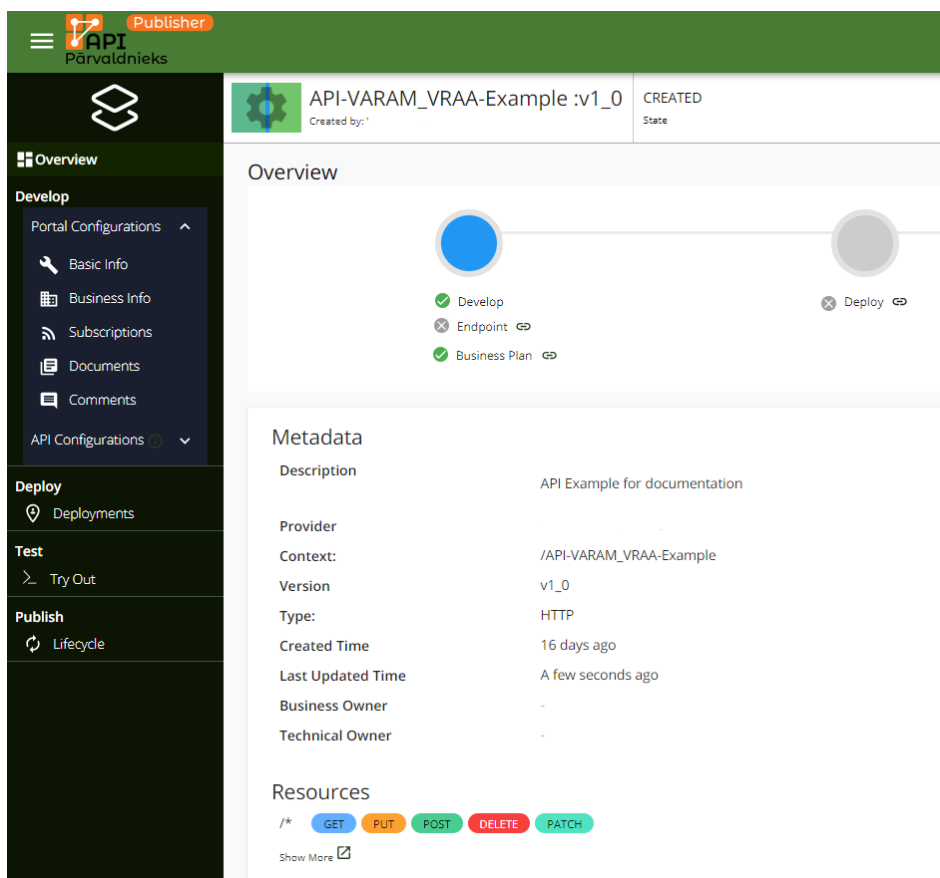
Izvēlnē *Endpoint(b)* jāizvēlas un jānokonfigurē atbilstošu galapunkta tipu un galapunkta saiti. Rediģējot un pievienojot galapunktu saites, obligāti kā pamata vidi ir jāizmanto *Production*. Pārējie tipi ir pieejami rediģēšanai, bet to izmantošana ir atkarīga no konkrēta klienta vajadzībām

Business Plan(c) izvēlnē var konfigurēt Servisa pieejamību klientiem. Pēc noklusējuma pieprasījumu skaits nav ierobežots, bet pēc publicētāja velmēm var ļaut klientiem izvēlēties sev piemērotus izsaukumu skaita ierobežojumus. API abonēšanas plāni var būt sekojoši:

- Bronze : Allows 1000 requests per minute
- Gold : Allows 5000 requests per minute
- Silver : Allows 2000 requests per minute
- **Unlimited : Allows unlimited requests** (pievienots pēc noklusējuma)

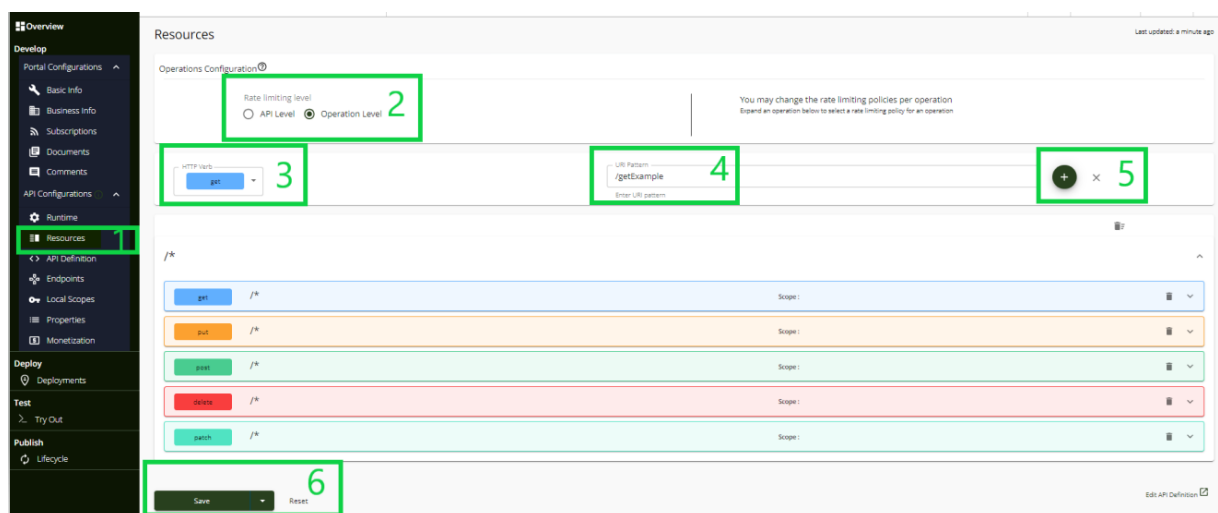


API definīciju, iespējams rediģēt vai tai pievienot jaunas metodes nospiežot pogu “Show More” pie Overview apakšpunkta *Resources*.



Sadaļā *Resources* izvērstajā skatā ir pieejama jauno metožu pievienošana un API darbināšanai nepieciešamo metožu pārvaldības funkcionalitāte. Lai varētu pievienot publicētam servisam jauno metodi ir nepieciešams izpildīt sekojošus soļus:

1. Atvērt sadaļu *Resources*;
2. Izvēlēties Operāciju limitēšanas līmeni (pēc noklusējuma ir *Operational Level*);
3. Izvēlēties nepieciešamo metode no saraksta;
4. Norādīt konkrēta galapunkta saites modeli (parasti *"/foo"* formātā);
5. Pievienot modeli servisam;
6. Saglabāt izmaiņas.



Turpmāk pievienoto modeli ir nepieciešams aprakstīt, pievienot nepieciešamos atribūtus un obligāti aizsargāt, piešķirot tai attiecīgu atļauju atbilstoši [Datu apmaiņas izveides vadlīniju](#) dokumentā aprakstītajai procedūrai (5.4.3.4. *Scope pievienošana*).

Darbības ir ieteicams izpildīt sekojošā secībā:

1. Izvērst nepieciešamo modeli;
2. *Summary & Description* sadaļā sniegt Metodes aprakstu;
3. *Summary & Description* sadaļā aizpildīt vispārīgu informāciju, ja tāda ir;
4. *Operation Governance* sadaļā ir nepieciešams ieslēgt Security parametru atļauju funkcionalitātes ieslēgšanai, turpat pēc nepieciešamības iestatīt pieejamības politiku (nav obligāti aizpildāms parametrs);
5. *Operation Governance* sadaļā jāpievieno jau izveidotu atļauju vai jāizveido no jauna (*Create New Scope* izvēlne) atļauju konkrētai metodei;
6. Jāveic servisa izsaukumiem nepieciešamo parametru konfigurāciju, atbalstāmie parametri atgriežamiem datiem ir ierobežoti atbilstoši [RFC 6838](#) specifikācijai:
 - application/json
 - application/xml
 - application/x-www-form-urlencoded
 - multipart/form-data
 - text/plain; charset=utf-8
 - text/html
 - application/pdf
 - image/png
7. Pēc nepieciešamības var veikt izmaiņas servisa Swagger datnē (Swagger parāda servisa iespējas bez piekļuves tā pirmkodam) izvēloties *Edit API Definition* izvēlni, piemēru var apskatīt 1. pielikumā;
8. Saglabāt konfigurācijā veiktās izmaiņas.

Pievienotās metodes ir iespējams rediģēt, piemēram, pievienojot tās aprakstu vai papildu parametrus, kā arī dzēst (nospiežot ikonu).

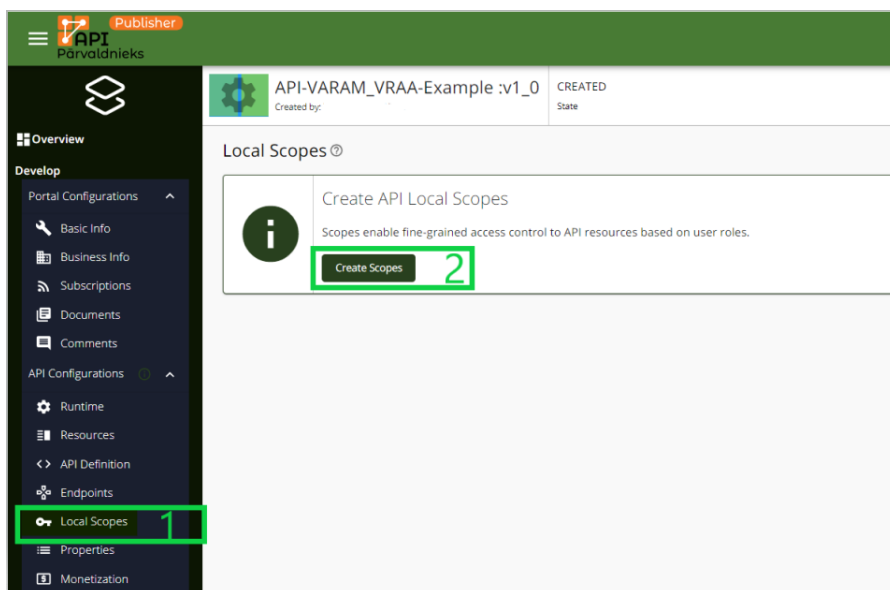
- **Consumes** – datu formāts, kādā dati tiek padoti. Attiecas tikai uz metodēm POST, PUT un PATCH. GET metodes gadījumā vērtība nav jānorāda. [Plašāks apraksts](#).
- **Parameter Name** – parametra, kuru izmanto konkrētā metode, apraksts
- **Parameter Type** – jānorāda viens no parametra veidiem. Vienai metodei var būt dažādi parametri.
 - a. query – vaicājums jeb atlases kritēriji. Parametru sāk ar “?”, piemēram,
 - b. header – tiek iekļauti pieprasījuma galvenē, parasti saistīti ar autorizāciju, piemēram, kad jānorāda lietotājvārds.
 - c. formData – izmanto augšupielādes apmēra noteikšanai
 - application/x-www-form-urlencoded – izmanto POST metodes gadījumā, lai padotu vienkāršas vērtības. Nav piemērots bināru datu padošanai.
 - multipart/form-data – ļauj augšupielādēt binārus datus un dažāda tipa failus. Katram laukam ir noteikts apjoms.
- **Data Type**– manuāli norāda padoto datu tipu, piemēram, integer (skaitli) / string (simbolu virkne) / file / u.c.

- **Required** – norāda, vai parametrs ir obligāts

Atļauju (scope) pievienošana

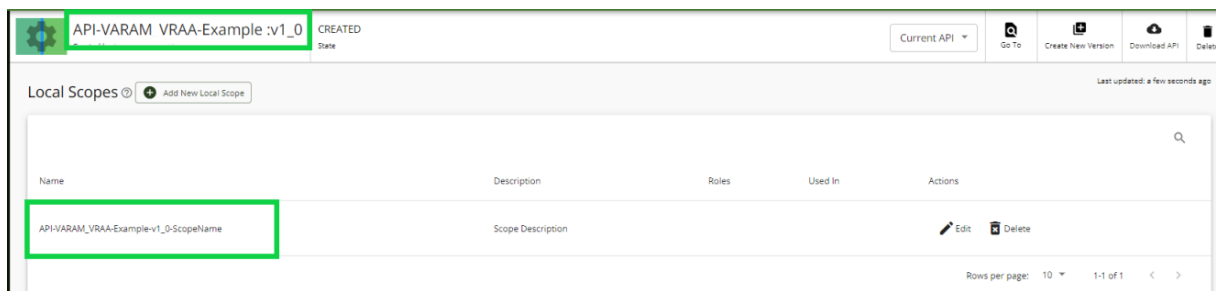
5. solī aprakstītajai servisa aizsargāšanai, jeb konkrētai metodes atļaujas (scope) piešķiršanai ir nepieciešams izveidot jaunu atļauju sadaļā *Local Scopes*, vai nospiežot pogu *Create New Scope* pie metodes modeļa izveides.

Laukā “Name” nepieciešams ievadīt atļaujas nosaukumu. Atļaujas nosaukums var sastāvēt no brīvi izvēlētas burtu vai ciparu virknes bez speciālajiem simboliem. Pēc atļaujas nosaukuma un apraksta definēšanas nepieciešams saglabāt jaunizveidoto atļauju



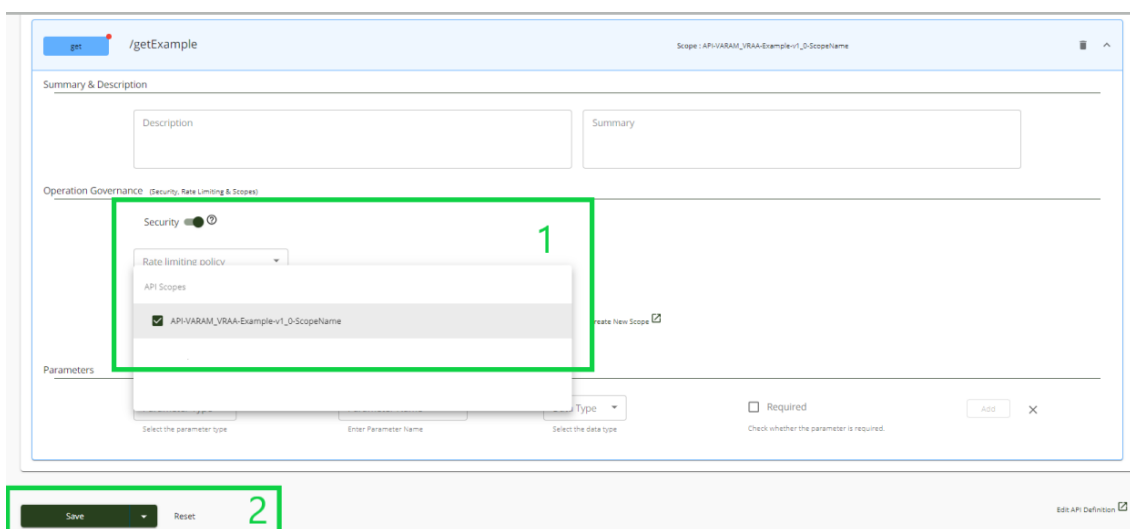
Atļaujas pilnais nosaukums, kurš būs arī jāizmanto atļauju piešķiršanai PFAS Uzticampo pušu sadaļā un API izsaukumos veidojas automātiski no “API nosaukums + versija” + “Name”.

The screenshot shows the 'Scopes > Create New Scope' form. The top header shows 'API-VARAM_VRAA-Example :v1_0' with a 'CREATED' status. The form contains several input fields: 'Name' (labeled 'ScopeName' and highlighted with a green box and red '1'), 'Display Name' (labeled 'Scope Display Name'), 'Description' (labeled 'Scope Description'), and 'Roles' (labeled 'Enter roles and press Enter'). At the bottom, there is a 'Save' button highlighted with a green box and a red '2'.



Pēc atļaujas saglabāšanas, to iespējams sasaistīt ar konkrēto metodi (vai vairākām metodēm) *Resources* sadaļā, kā tas parādīts zemāk attēlā.

1. Jāizvēlas ar metodi sasaistāmo atļauju;
2. Jāsaglabā izmaiņas konfigurācijā.



1.3 Servisa atribūtu aizpilde

Pēc *Develop* soļa un pirms servisa izmitināšanas ir obligāti jāveic Servisa parametru aizpilde.

1.3.1 Basic Info sadaļa

Design Configuration lauki (ar **bold** atzīmēti obligāti aizpildāmie lauki):

- API bilde;
- **Edit Description**, jeb servisa apraksts;
- **Publisher Access Control**
 - **All** – servisu varēs rediģēt visi lietotāji ar administratora lomu (API Publisher);
 - **Restricted by role(s)** – servisu publicētāju portālā varēs rediģēt lietotāji ar speciāli izveidotu lomu (API Publisher);
- **Developer Portal Visibility**
 - **Public** – servisu varēs redzēt visi lietotāji ar datu ņēmēja lomu (API Devportal);
 - **Restricted by role(s)** – servisu abonēšanas portālā varēs redzēt lietotāji ar speciāli izveidotu lomu (API Devportal);
- **Tags** – tagi un atzīmes atvieglotai API meklēšanai un atlasei no saraksta;
- **API Categories** – API piederības definēšana konkrētai kategorijai, jāizvēlas iestāde, kuras pārziņā ir šis API;
- GitHub URL – saite uz API GitHub portālā;
- Slack URL – saite uz API Slack;
- **Make this the default version** –

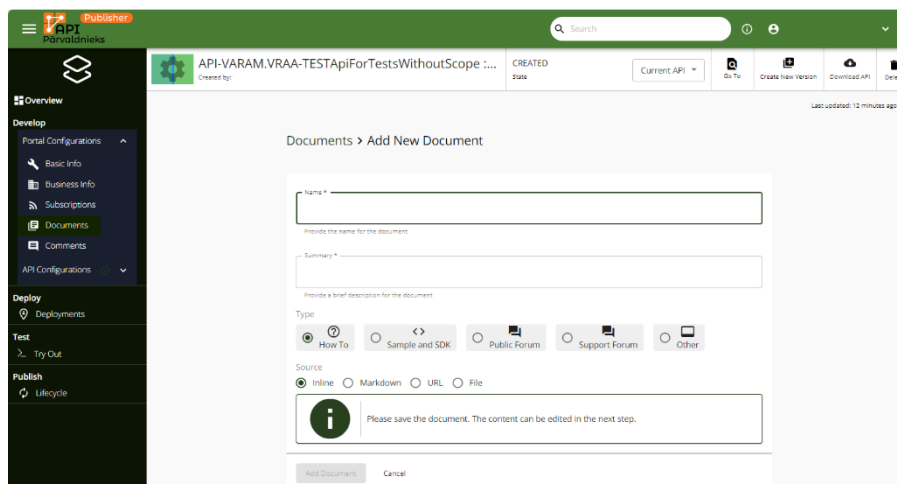
- **YES** – Servisa versija (ja ir vairākas) ar šo iezīmi tiks uzskatīta par primāro un pie vairākām servisa iterācijām var tikt izsaukta bez versijas apzīmējuma;
- **NO** – Servisa izsaukumā vienmēr būs jānorāda versiju.

1.3.2 Business info sadaļa

Sadaļa ir radīta, lai servisa īpašnieks vai publicētājs varētu norādīt kontaktinformāciju. Visi sadaļā esošie lauki ir obligāti aizpildāmi.

- **Business Owner** – servisa īpašnieks;
- **Business Owner Email** – servisa īpašnieka e-pasts;
- **Technical Owner** – servisa uzturētājs;
- **Technical Owner Email** – servisa uzturētāja e-pasts.

1.3.3 Documents sadaļa



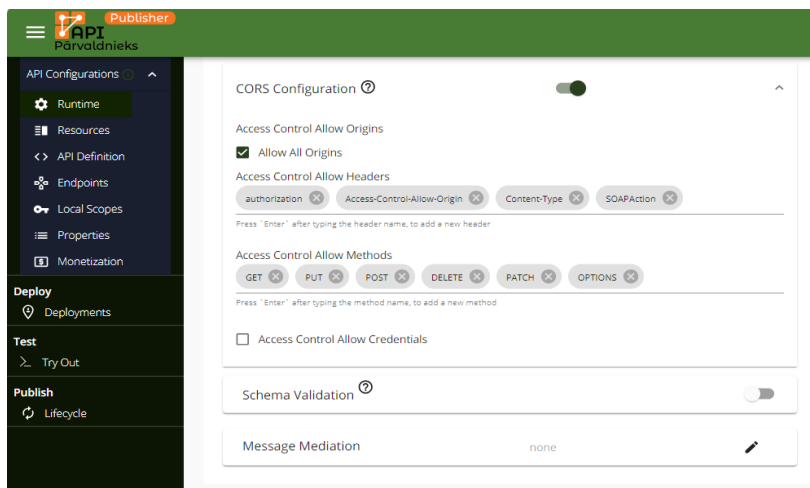
Sadaļā radīta, lai publicētāji vai servisa uzturētāji varētu pievienot saistīto dokumentāciju, piemērus, SDK vai citu lietotājiem noderīgu informāciju. Turpmāk tā būs pieejama lejupielādei Izstrādātāju portālā.

Izvēloties apakšsadaļu *Runtime* pie *API Configurations* sadaļas, iespējams konfigurēt norādītā galapunkta atgrieztās kļūdas aiztures un noildzes gadījumā. Iespējams izvēlēties vairākus kļūdu kodus.

1. *Endpoint Suspend State* – “aizturēts” galapunkts nesaņem pieprasījumus un neatgriež atbildes.
 - a. **Initial Duration** – norāda, pēc cik ilga laika tiek sūtīts atkārtots ziņojums. Pēc noklusējuma pēc 30 sekundēm;
 - b. **Max Duration** – norāda milisekundēs, cik ilgi galapunkts tiks aizturēts;
2. *Error Codes* – Jānorāda, kādas saņemtās kļūdas gadījumā, galapunkts tiks aizturēts. Iespējams norādīt vairākus atgrieztos kļūdas kodus;
3. *Connection Timeout* - norāda atbildes darbības gadījumā, kad tiek saņemta savienojuma noilgums:

1.4 CORS konfigurācija

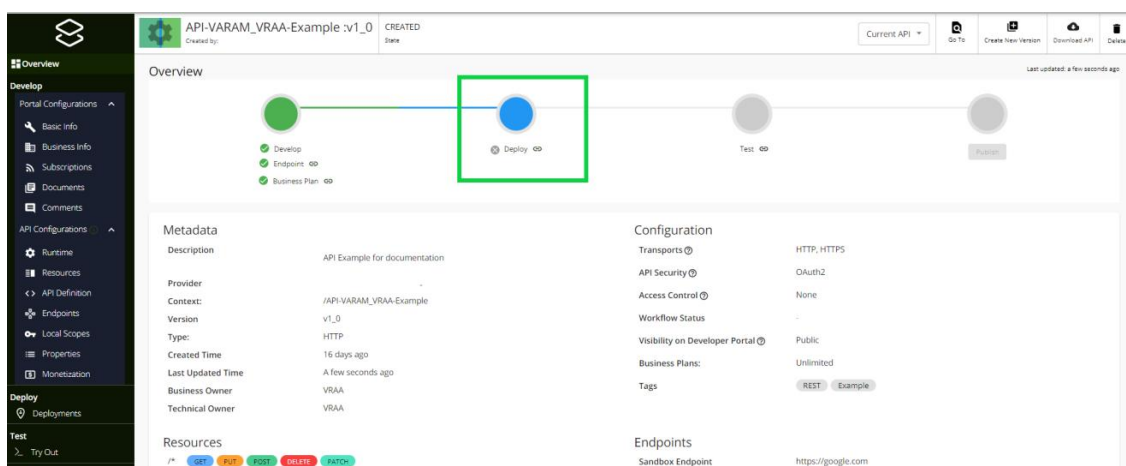
CORS (*Cross-Origin Resource Sharing*) ir mehānisms, kas ļauj piekļūt aizsargātiem resursiem no ārēja domēna (kas nav resursu domēns). CORS ir iespējams norādīt globāli visiem publicētajiem API. Parametra definēšanas pieejama sadaļā *Runtime*.



CORS apraksts un noderīgas saites
[Enabling CORS for APIs](#)

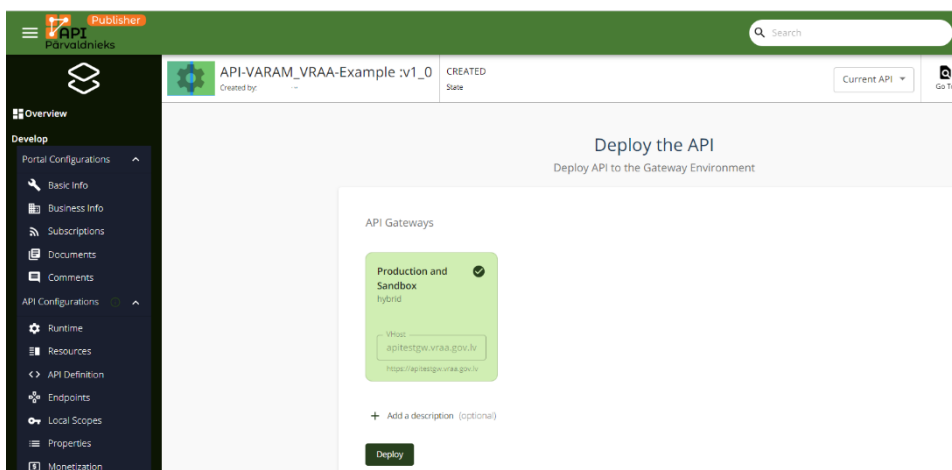
1.5 Sadaļa “Deploy”

Pēc visu Develop soļu izpildes un aprakstu pievienošanas Portal Configurations apakšsadaļās, kļūst iespējams veikt servisa izmitināšanu. Lai iniciētu šo procesu, publicētājam ir nepieciešams pāriet uz Overview sadaļu un servisa saskarnē izvēlēties punktu Deploy.

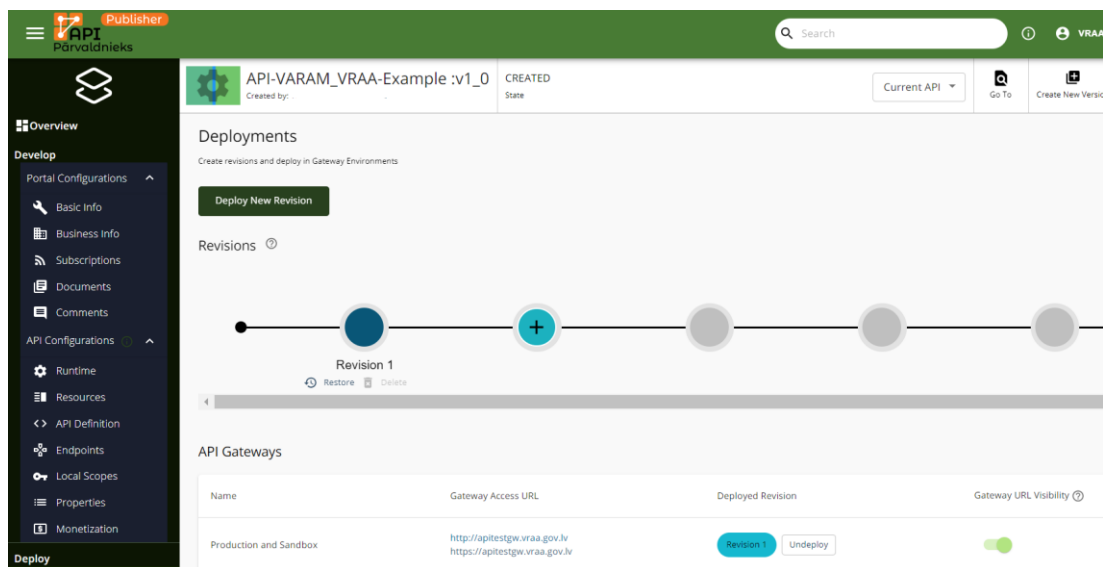


Turpmāk izvērsies Deploy sadaļas izvēlnē, kur ir sekojoši lauki:

1. API publicēšanas vide, parasti paliek bez izmaiņām;
2. API galapunkta apraksts konkrētai servisa revīzijai;
3. Veikt izmitināšanu.

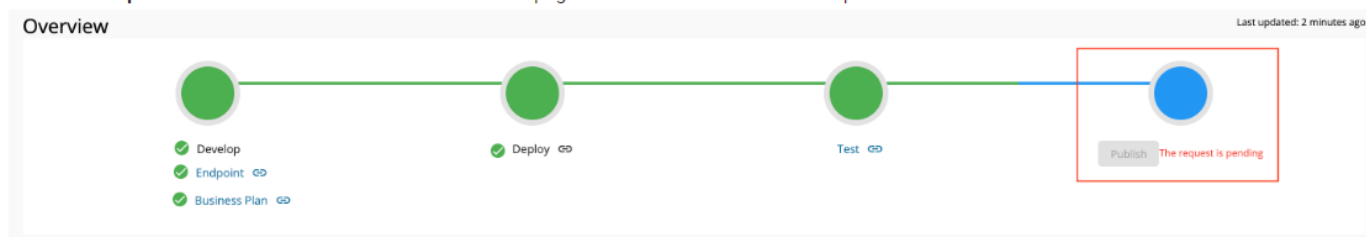


Pēc servisa izmitināšanas, kļūst pieejami revīzijai piesķirtie atribūti, kuri satur vārtejas saiti, publicētas servisa revīzijas statusu un vārtejas saites redzamības konfigurēšanas izvēlni.



Pēc soļu izpildes nepieciešams pāriet uz Overview sadaļu. Produkcijas vidē, nospiežot 'Publish', VDAA tiks nosūtīts servisa publicēšanas pieprasījums, un tiks pārbaudīta servisa atbilstība vadlīnijām. Kamēr servisa publicēšanas pieprasījums nebūs apstiprināts, būs redzams statuss “The request is pending” (skatīt zemāk attēlā):

Note that the **publish** button will be disabled in the overview page until the workflow task is completed or deleted.



Lai serviss varētu tikt apstiprināts publicēšanai, tam jābūt izpildītiem šādiem kritērijiem:

- Servisa nosaukums atbilst notācijai: API/ISS-{Autoritātes identifikators no iestāžu un struktūrvienību klasifikatora (Authority)}-{datu ņēmēja servisa nosaukums bez speciālajiem simboliem}-{versija notācijā – “v{Major}_{Minor}”, piemēram, “v1_0”};
- Servisa versijas numurs atbilst notācijai – “v{Major}_{Minor}”, piemēram, “v1_0”;
- Servisam ir izveidots un aktualizēts TAG saraksts, kas minimāli satur - iestādes nosaukumu, iestādes nosaukuma saīsinājumu, servisa nosaukumu, servisa tipu;
- Servisam ir pievienots apraksts pie “Edit description”;
- Servisam ir ievadīta informācija par biznesa un tehnisko īpašnieku;

Līdz servisa apstiprināšanai ir iespējams atcelt nosūtīto publicēšanas pieprasījumu un veikt izmaiņas servisa aprakstā. Lai to varētu izdarīt, sadaļā “Lifecycle” pie aktuālā publicēšanas pieprasījuma jānospiež poga “Delete task” (skatīt zemāk attēlā):

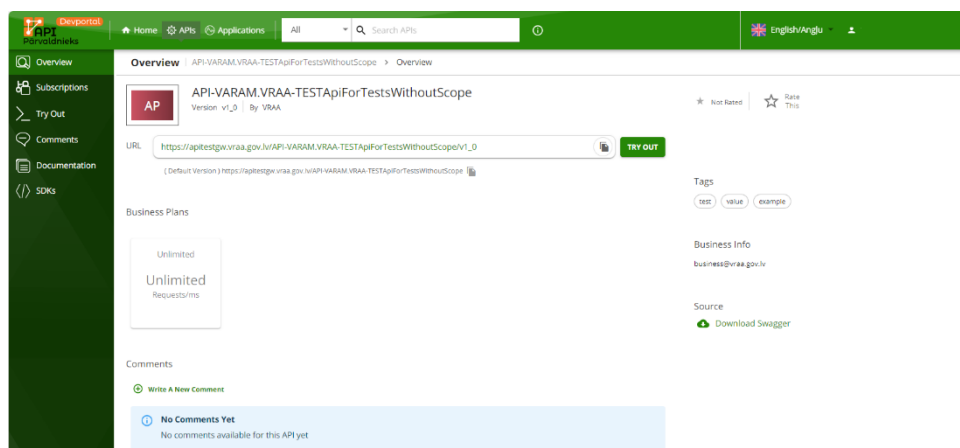
Lifecycle

Pending lifecycle state change.

Current state is Created

Delete Task

Ja serviss atbildīs visām vadlīnijām un tiks apstiprināts publicēšanai, tas automātiski tiks publicēts un to būs iespējams apskatīt API Izstrādātāju portālā. Par servisa publicēšanas apstiprinājumu un publicēšanu, lietotājam tiks nosūtīts informatīvs e-pasts.



Gadījumā, ja servisa publicēšana tiks noraidīta, serviss tiks atgriezts Created stāvoklī un servisa publicēšanas pieprasījuma veicējs saņems e-pastu ar ziņu, ka publicēšanas pieprasījums ir noraidīts ar uzskaitītiem trūkumiem, kurus nepieciešams izlabot pirms veikt atkārtotu servisa publicēšanu.

Testa vidē, pēc pogas 'Publish' nospiešanas, serviss automātiski tiks publicēts un būs pieejams izmantošanai API Izstrādātāju portālā.

1.6 Uzticamo pušu pārvaldība

Atļauju piešķiršanas kārtība

Ja serviss ir aizsargāts ar atļauju (scope), tad to ir nepieciešams pievienot izveidotajam klienta lietojumam jeb aplikācijai, izmantojot <https://visstv.vraa.gov.lv/> (testa vidē) vai <https://viss.gov.lv/> (produkcijas vidē) portāla PFAS uzticamo pušu pārvaldības sadaļas. Tās atrodas pēc autorizācijas portālā, nospiežot uz sava vārda, uzvārda, un ir pieejamas tikai lietotājiem ar attiecīgo PFAS lomu. Piekļuves PFAS Uzticamo pušu sadaļai tiek izveidotas reizē ar API Pārvaldnieka "Publisher" portāla lietotāju. Papildus tās ir iespējams iegūt aizpildot [veidlapu](#) un norādot lomu *PfasRelyingParty_Authority_Administrator*.

Uzticamās puses

- ▼ **PFAS uzticamo pušu pārvaldība**
- ▼ VPM uzticamo pušu pārvaldība
- ▼ IDS uzticamo pušu pārvaldība
- ▼ Atļaujas

Nākošajā solī jāatrod API Izstrādātāju (Devportal) portālā izveidoto klienta lietojumu, ar kuru tika abonēts konkrēts serviss. To var atrast divos veidos: 1) Nosaukumā ievadot izveidotā klienta lietojuma nosaukumu. 2) Identifikatora laukā ievadot klienta lietojuma Consumer Key (Tā iegūšana aprakstīta 3.1. nodaļā). Piešķirt atļaujas Datu ņēmēju aplikācijām var tikai konkrēta servisa īpašnieks (Datu devējs) ar attiecīgo PFAS lomu.

Atverot atrasto, izveidoto klienta lietojumu, sadaļā Atļaujas jāpievieno nepieciešamā atļauja (scope)

Vis > Administrēšana > PFAS > Uzticamās puses > PFAS uzticamo pušu pārvaldība >

Detalizēta informācija par PFAS uzticamo pusi

Izmantojiet šo lapu uzticamās puses informācijas detalizētai apskatei.

Uzticamās puses īpašības Galapunkti Sertifikāti Pielaiides **Atļaujas**

Atļaujas

Nosaukums

Apraksts

[Pievienot atļauju](#)

Vis > Administrēšana > PFAS > Uzticamās puses > PFAS uzticamo pušu pārvaldība > Pievienot atļauju

Pievienot PFAS atļauju

Izmantojiet šo lapu PFAS atļauju pievienošanai.

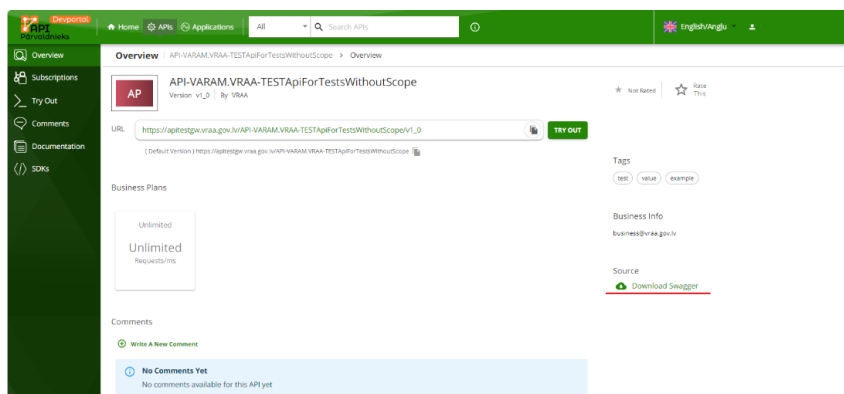
Nosaukums *

API-RAPLM_VRAA-CalculatationApiForTestsWithScope-v1_0-Divide

Saglabāt

Atcelt

Piemērs ar atļaujas pievienošanu



Nepieciešamās atļaujas iespējams atrast lejupielādējot swagger(.JSON) datni izvēlētajam API. Atļaujas ir saistītas ar konkrētām metodēm. Viena SOAP servisa ietvaros var būt tikai viena atļauja visām metodēm. REST un GraphQL servisu gadījumā katrai atsevišķai servisa metodei var būt sava atļauja, tāpēc Datu devējiem ir iespēja sīkāk un precīzāk pārvaldīt piekļuves iespējas pie saviem servisiem.

Apzinoties nepieciešamās metodes, klients, jeb Datu ņēmējs Json failā jāmeklē "scopes", un izvēlas nepieciešamās, kuras atbilst izmantošanai paredzētajām metodēm. Pēc tam klients vienojas ar Datu devēju par atļaujas pievienošanu savai aplikācijai PFAS uzticamo pušu pārvaldības sadaļā. **Pievienošanu nodrošina Datu devējs.**

```
"flow" : "implicit",
"scopes" : {
  "API-RAPLM_VRAA-CalculationApiForTestsWithScope-v1_0-Divide" : "",
  "API-RAPLM_VRAA-CalculationApiForTestsWithScope-v1_0-Multiply" : ""
},
```

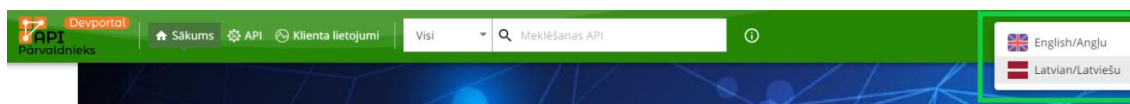
2. API Izstrādātāju portāls

Lai piekļūtu servisu abonēšanas funkcionalitātei ir jāautenticējas ar sev piešķirto lietotāju Izstrādātāju portālā (*API Devportal*):

- Testa vide: <https://apitest.vraa.gov.lv/devportal>
- Produkcijas vide: <https://api.viss.gov.lv/devportal>

Portāls ir pieejams servisu apskatīšanai arī neautenticētiem lietotājiem.

API Izstrādātāju portāls lietotājam attēlosies tādā valodā, kāda būs izvēlēta lapas galvenē



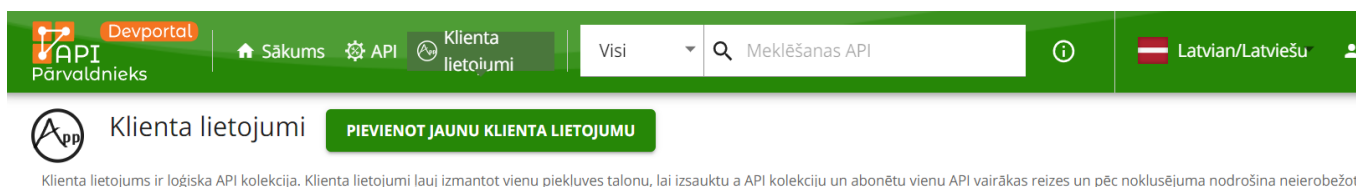
2.1. Klienta lietojuma izveide

Pirmais solis, pirms varētu sākt abonēt servisu Izstrādātāju portālā, ir Klienta lietojuma, jeb aplikācijas izveide. Klienta lietojumu izveides sadaļa ir atrodama lapas galvenē, izvēloties *Klienta lietojumi*.



Klienta lietojums ir loģiskais vairāku API apvienojums. *Klienta lietojums* pieļauj viena piekļuves talona izmantošanu, vairāku dažādu API servisu izsaukumu veikšanai. Talons tiek pieprasīts vienas konkrētas aplikācijas ietvaros, servisu ārpus aplikācijas ar šo konkrēto talonu nevarēs izsaukt. Pēc noklusējuma izveidots *Klienta lietojums DefaultApplication* nav pieļauts izmantošanai, jo veidotajām aplikācijām jāatbilst *Datu apmaiņas izveides vadlīniju* dokumentā aprakstītai sintaksei (5.3.1.3 *Klienta lietojuma reģistrācija*) (Skatīt 2.4. sadaļu šajā dokumentā).

1. Lai pievienotu jaunu *Klienta lietojumu*, sadaļā *Klienta lietojumi* ir jāizvēlas *Pievienot jaunu klienta lietojumu*.



2. Nākošajā solī Klienta lietojuma nosaukums jāveido vadoties pēc Klienta lietojuma nosaukuma formāta notācības: “APP- {Autoritātes identifikators (visscore:authorityIdent) no iestāžu un struktūrvienību klasifikatora (Authority), izmanto apakš svītru “_” punktu vietā, un divas apakš svītras “__” vienas apakš svītras vietā, piemēram, “SIA_Iestade1”} - {datu ņēmēja klienta lietojuma nosaukums bez speciālajiem simboliem, kurā konkrētais serviss (API) tiks izmantots, piemēram, “TapisUI” u.tml.) Piemērs: “APP-SIA_Iestade1-TapisUI”.

Obligāti aizpildāmie lauki ir atzīmēti ar **bold**

- **Application Name** – aplikācijas nosaukums ar atbilstošu sintaksi;
- **Shared Quota for Application Tokens** – aplikācijas izsaukumu ierobežojums;
- Application Description – aplikācijas apraksts.

The screenshot shows the 'Create an application' form in the API Pārvaldnieks portal. The form has a green header with the portal logo and navigation links. The main content area is white with a green border. It contains three input fields: 'Application Name' (required, with a hint 'APP-Identifier-Name-v*_.*'), 'Shared Quota for Application Tokens' (a dropdown menu set to '10PerMin'), and 'Application Description' (a text area with a character count '(512) characters remaining'). At the bottom are 'SAVE' and 'CANCEL' buttons.

Klienta lietojuma īpašnieka maiņa

Gadījumā, ja nepieciešams veikt klienta lietojuma īpašnieka maiņu jeb aplikācijas pārvietošanu no viena API Pārvaldnieka lietotāja profila uz citu starp vienas un tās pašas iestādes darbiniekiem, nepieciešams nosūtīt maiņas pieprasījumu uz atbalsts@vdaa.gov.lv

- Pieprasījumā jānorāda šāda informācija:
- Klienta lietojuma nosaukums un identifikators (ClientId)
- Esošā īpašnieka API Pārvaldnieka lietotājvārds
- Jaunā īpašnieka API Pārvaldnieka lietotājvārds
- Iestāde, kuras ietvaros tiks veikta aplikācijas īpašnieka maiņa

2.2. Klienta lietojuma atslēgu ģenerēšana

Lai ar izveidoto klienta lietojumu varētu veikt servisa izsaukumus, nepieciešams ģenerēt klienta lietojuma identifikatoru un atslēgu (Client ID un ClientSecret). Atslēgu ģenerēšana jāveic izvēloties nepieciešamo aplikāciju, apakšsadaļā *OAuth2 Taloni* nospiežot “Generate keys”.

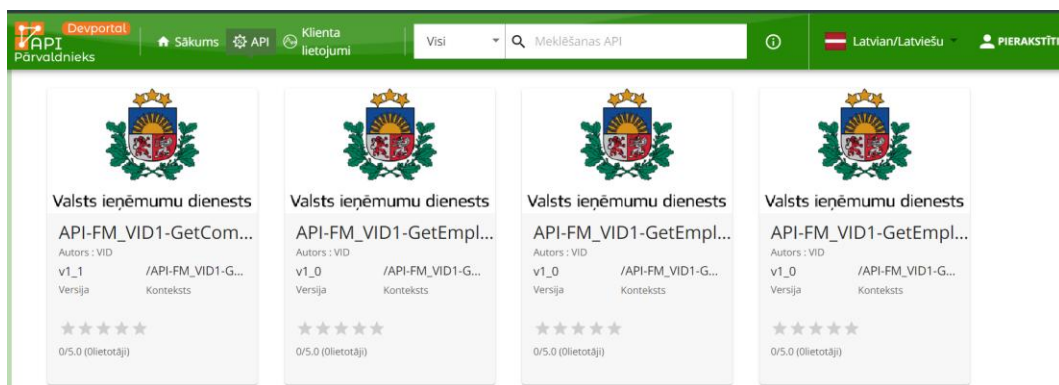
The screenshot shows the configuration page for 'APP-VARAM_VRAA-TestAPP' in the API Pārvaldnieks portal. The page has a green header with the portal logo and navigation links. The main content area is white with a green border. It contains a section for 'Production OAuth2 atslēgas' with a button 'IZMANTOT ESOŠAS OAUTH ATSLĒGAS'. Below this is a section for 'Atslēgas konfigurācija' with fields for 'talona galapunkts' (https://apitestgw.vraa.gov.lv/token), 'Atsaukt galapunktu' (Hidden), 'Prezentāciju veids' (Klienta lietojumam var izmantot šādus piešķiršanas veidus, lai ģenerētu piekļuves tokenus. Pamatojoties uz Klienta lietojum), 'Atzīmēšanas URL' (Atzīmēšanas URL), and 'Token type' (Token type, DEFAULT). At the bottom is a 'GENERATE KEYS' button.

Ja klienta lietojuma nosaukums ir izveidots atbilstoši vadlīnijām, tiks uzģenerēts un parādīts Consumer Key un Consumer Secret, savukārt gadījumā, ja tiek attēlota kļūda par atslēgu ģenerēšanu – ir jāveic izveidotās aplikācijas nosaukuma precizēšana. (Skatīt 2.1. sadaļu)

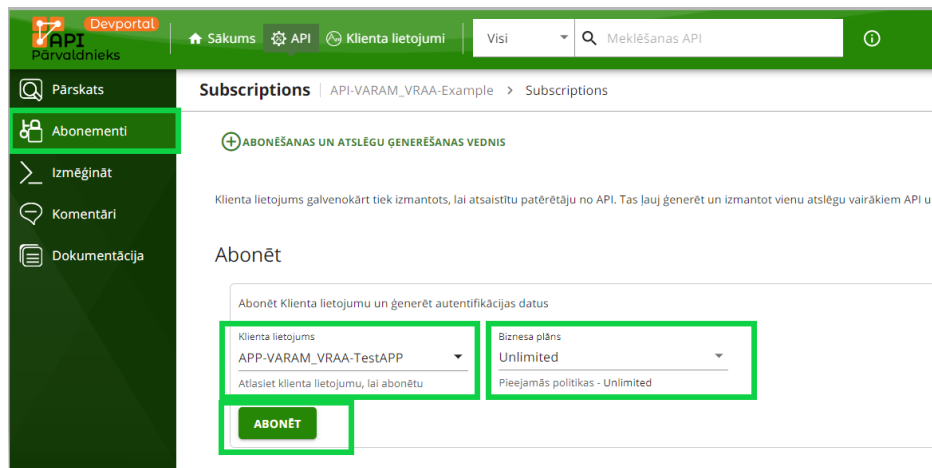
!Svarīgi: Piekļuves talona (access_token) iegūšanai **neizmanto** piedāvāto iespēju *CURL To Generate Access Token/CURL Piekļuves Talona Ģenerēšana*. Tā vietā skatīt nodaļu 3. “*Servisu testēšana izmantojot Postman*”, par piekļuves talona iegūšanu izmantojot sertifikāta autentifikāciju.

2.3. API abonēšana un izmantošana

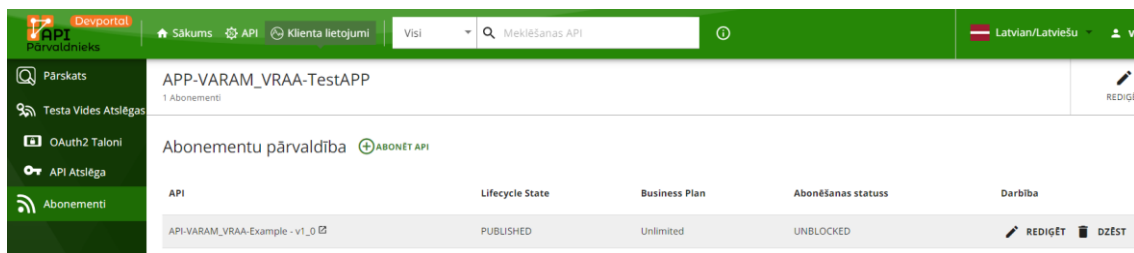
Pēc sekmīgas klienta lietojuma izveides ir iespējams izvēlēties un abonēt nepieciešamos servissus no API kataloga. Nepieciešamo servisu iespējams meklēt vai nu kopējā API katalogā, vai arī izmantojot meklētāja funkcionalitāti.



Izvēloties vajadzīgo servisu, servisa lapā jāizvēlas *Abonementi* apakšsadaļu (1), kur pierakstoties ir jānorāda attiecīgo Klienta lietojumu (2), kura ietvaros ir nepieciešams izmantot izvēlēto API, jāizvēlas vienu no pieejamiem pieprasījumu limitēšanas veidiem (3) un jānospiež poga *Abonēt, lai veiktu servisa abonēšanu* (4).



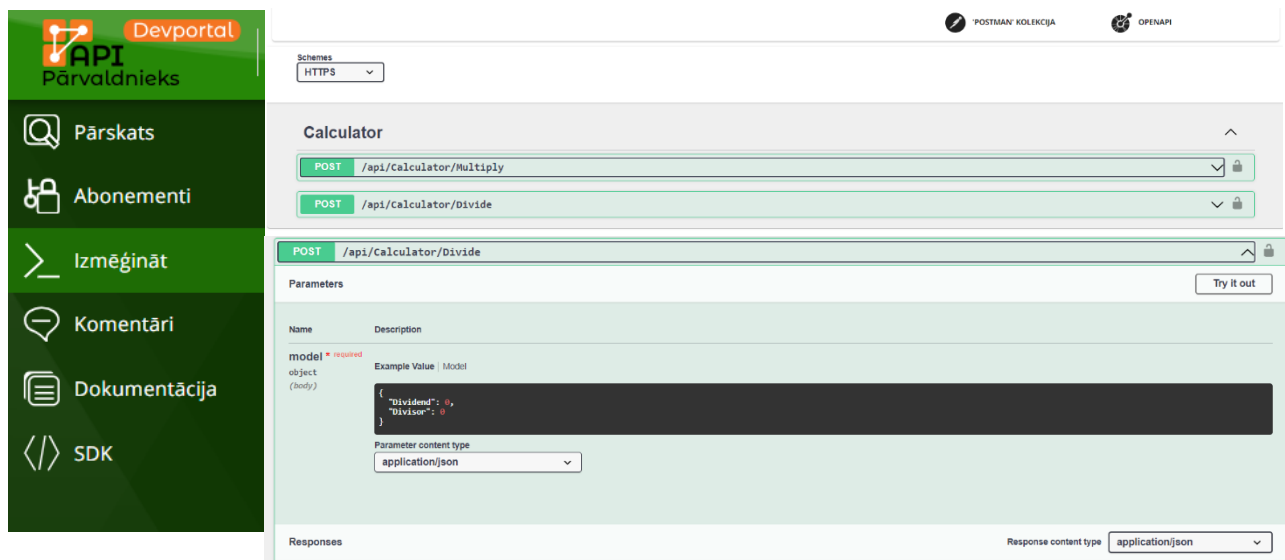
Savus abonētos API servissus iespējams redzēt *Klienta lietojumi* sadaļas *Abonementi* apakšsadaļā. Ja serviss ir pieejams šajā sadaļā, tas nozīmē, ka servisa abonēšana noritēja veiksmīgi.



Turpmāka servisu izmantošanas un izsaukšanas procedūra no tehniskā skatupunkta ir aprakstīta [Datu apmaiņas izveides vadlīniju](#) dokumentā

Try Out/Izmēģināt sadaļa

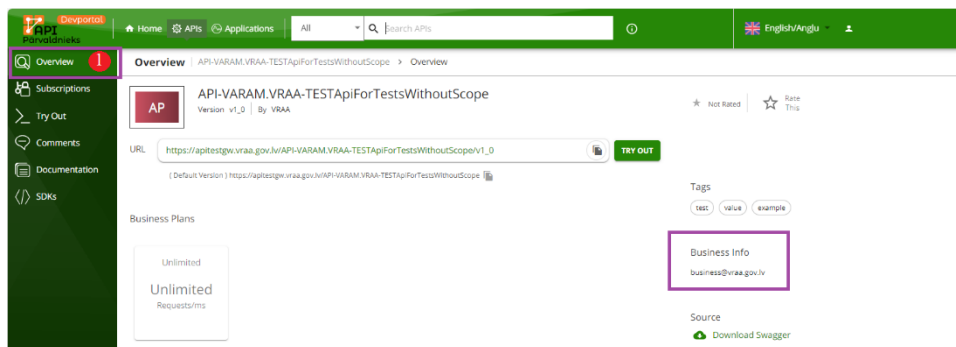
API sadaļā Izmēģināt/Try Out iespējams apskatīt konkrētajam servisam pieejamās metodes un izsaukuma piemērus:



Lūdzam ņemt vērā, ka šajā sadaļā **nav** iespējams testēt servisu darbību un veikt servisa izsaukumus. Informāciju par servisu testēšanu skatīt nodaļā 3. “*Servisu testēšana izmantojot Postman*”.

Tiesību pieprasīšana no Datu devējiem

Lai iegūtu piekļuves tiesības uz sevis izvēlētu servisu, nepieciešams sazināties ar tā īpašnieku. Kontaktinformācija atrodama pie izvēlētā API, sadaļā *Business Info*. Ja lietotājam jau ir izveidots API Pārvaldnieka lietotājs, tad tiesību iegūšanai izmantot servisu, VDAA iesaiste vairs nav nepieciešama.



Pieprasot atļauju datu devējam jānorāda:

- Servisa nosaukums, kuru vēlas izmantot
- Izveidotā Klientu lietojuma (Aplikācijas) nosaukumu, ar kuru abonēts serviss un tā Consumer Key.
- Atļaujas (scope), kuras vēlēšies izmantot

Pēc atļaujas saņemšanas iespējams sākt izmantot šos serviss/metodes

3. Servisu testēšana izmantojot Postman

Produkcijas un testa vidē, talonu izgūšana notiek tikai izmantojot sertifikāta autentifikāciju!

3.1. Piekļuves talona (access_token) iegūšana izmantojot sertifikātu

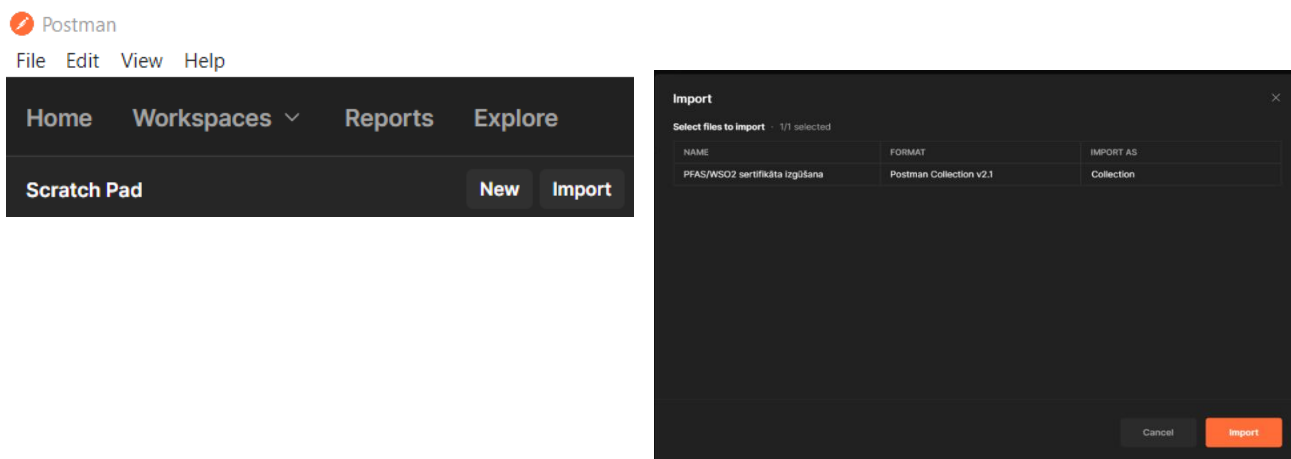
Nepieciešamais piekļuves talona iegūšanai:

- PFAS Sertifikāts (Iegūšana aprakstīta - <https://www.vdaa.gov.lv/lv/api-parvaldnieks#sertifikata-pieprasijums>
!!! Iegūto sertifikātu sūtīt uz e-pasta adresi - atbalsts@vdaa.gov.lv, kā arī pieminēt priekš kuras vides sertifikāts vajadzīgs - testa vai produkcijas.
- Aplikācija API Pārvaldniekā
- KeyStore Explorer vai līdzvērtīgs rīks
- Postman v9.5.0 vai jaunāks

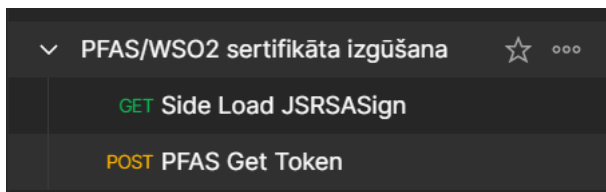
Soli:

1. Postman rīkā jāimportē kolekcija *PFAS-WSO2 sertifikāta izgūšana.postman_collection* (https://api.postman.com/collections/25378384-a96b179c-24f8-4b35-b7df-c8fde08b5dc0?access_key=PMAT-01HNFKB238WW6KAHJ4EN3YEXY5

Spogulis - <https://www.npoint.io/docs/6eeef13dc97dd42846c3> vai <https://api.npoint.io/6eeef13dc97dd42846c3>)



2. Kolekcija satur divus pieprasījumus - **Side Load JSRSASign** un **PFAS Get Token**. Šajā gadījumā nepieciešams atvērt PFAS Get Token.



3. Pre-request Script sadaļā jāveic 4 parametru konfigurāciju:

- var clientId – Klienta lietojuma identifikators (Consumer Key) (Iegūšana aprakstīta 2.2. nodaļā)
- var clientSecret – Klienta lietojuma atslēga (Consumer Secret) (Iegūšana aprakstīta 2.2. nodaļā)

- var privateKey – Privātā RSA atslēga, to iespējams iegūt ar KeyStore Explorer konvertējot izdoto PFX sertifikātu *.key formātā (Iegūšana aprakstīta 3.1. nodaļas 4. solī)
- var publicKey – Publiskā atslēga, iegūst izmantojot KeyStore Explorer no izdotā PFX sertifikāta (Iegūšana aprakstīta 3.1. nodaļas 5. solī)

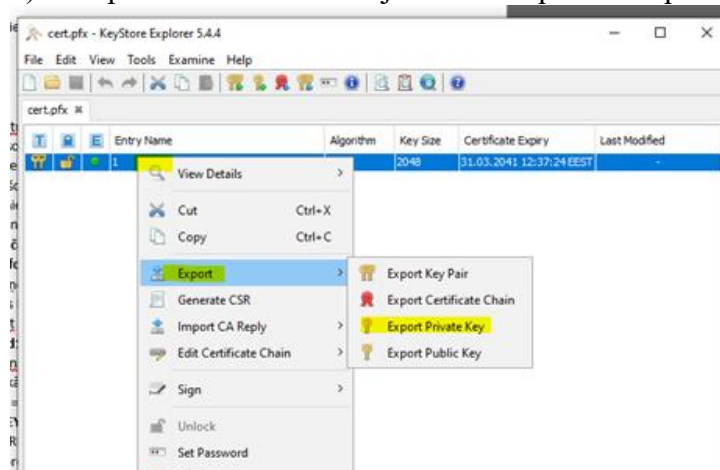
```

6 var clientId = "Klienta identifikators no API Pārvaldnieka";
7 var clientSecret = "Klienta atslēga no API Pārvaldnieka"; //environmental variable
8 var audience = "https://ha.vraa.gov.lv/STS/VISS.Pfas.STS/oauth2/token"; //environmental variable
9 var privateKey = "-----BEGIN RSA PRIVATE KEY-----\
10 RSA\
11 PRIVATE\
12 KEY\
13 -----END RSA PRIVATE KEY-----"; // privkey taken from certificate, see instruction
14 var publicKey = "Public key value"; //pubkey taken from certificate, see instruction

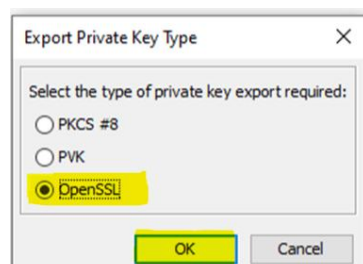
```

4. Privātās atslēgas iegūšana:

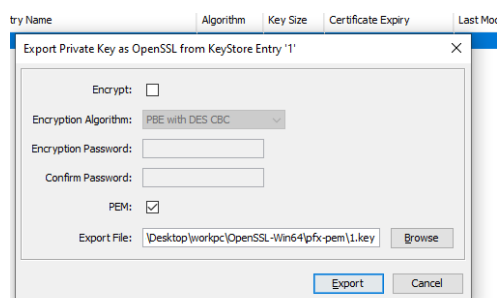
- a) KeyStore Explorer rīkā jānospiež File -> Open -> Jāizvēlas sertifikāts PFX formātā
- b) Jānospiež uz sertifikāta un jāizvēlas Export -> Export Private Key



c) Jāizvēlas OpenSSL



d) Jāsaglabā ar tādiem parametriem, kā parādīts attēlā



e) Ģenerēto *.key failu jāatver ar teksta redaktoru un jānokopē uzģenerēto **RSA PRIVATE KEY**

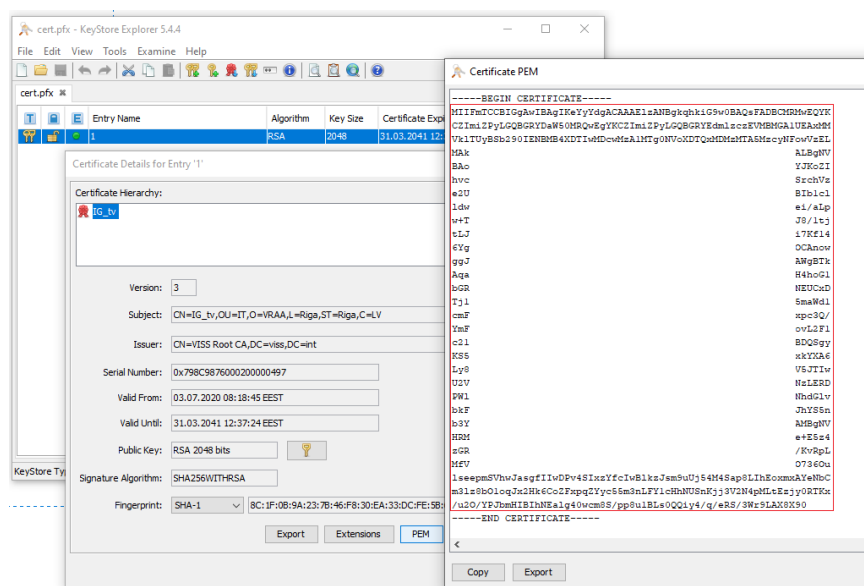
f) Iegūtā atslēga ir var privateKey parametrs, kurš jāiekopē Postman rīkā (3. solis šajā instrukcijā par piekļuves talona iegūšanu)

5. Publiskās atslēgas iegūšana

a) *KeyStore Explorer* rīkā divas reizes jānospiež uz sertifikāta, tad atvērsies logs, kurā jāizvēlas PEM

b) Sertifikāts atvērsies jaunā logā, kur tas redzams daļā starp

---BEGIN CERTIFICATE--- un ---END CERTIFICATE---, kā parādīts attēlā

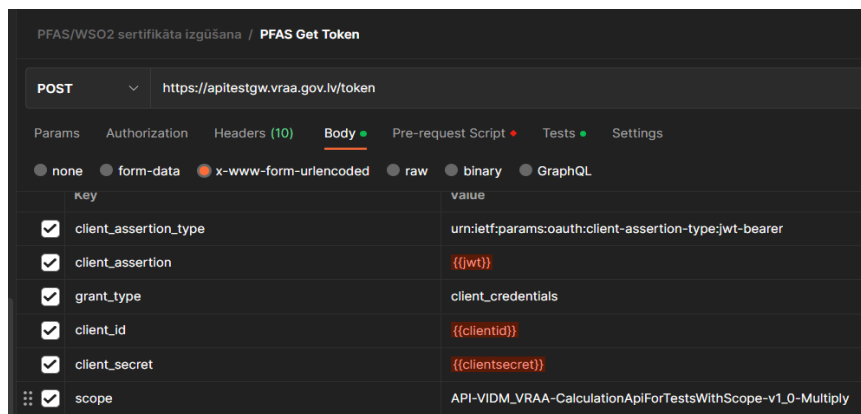


c) Iegūtā vērtība ir var publicKey parametrs, kurš jāiekopē Postman rīkā (3. solis šajā instrukcijā par piekļuves talona iegūšanu)

6. Pēc visu atribūtu aizpildes Pre-request Script sadaļā jāpalaiž Side Load JSRSASing un PFAS Get Token. Rezultātā tiks saņemts piekļuves talons (access_token).

```
"access_token": "urn:uuid:12ed2f0a-dase-2da2-bgas-f58b524c21ac:F2AA933282EE46BCF8FEAE5812D648DB79C78C6E",  
"expires_in": 7200,  
"token_type": "bearer"
```

Ja paredzēts izsaukt servisu, kura metodes ir aizsargātas ar atļauju (scope), tad iegūstot piekļuves talonu to ir nepieciešams norādīt "PFAS Get Token" pieprasījuma Body sadaļā.



Un atbilde saturēs piekļuves talonu un atļauju, ar kuru aizsargātās servisa metodes ar šo talonu iespējams izsaukt. Lai atbilde saturētu atļauju, norādītajai atļaujai jābūt pievienotai izmantotajai aplikācijai (Skatīt 1.6. sadaļu)



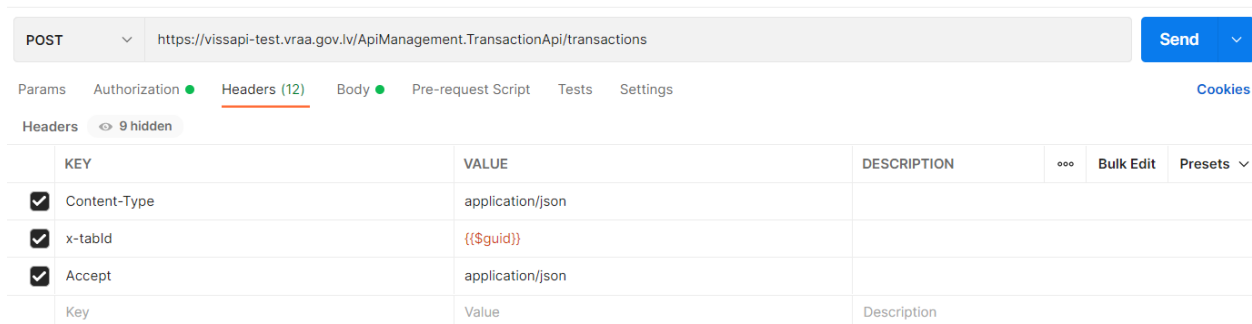
3.2. Transakcijas izveidošana servisa izsaukšanai

Izsaukuma metode – POST

Izsaukuma saite (Testa vidē) - <https://vissapi-test.vraa.gov.lv/ApiManagement.TransactionApi/transactions>

Izsaukuma saite (Produkcijas vidē) - <https://vissapi.viss.gov.lv/ApiManagement.TransactionApi/transactions>

Header sadaļā jānorāda šādi parametri:

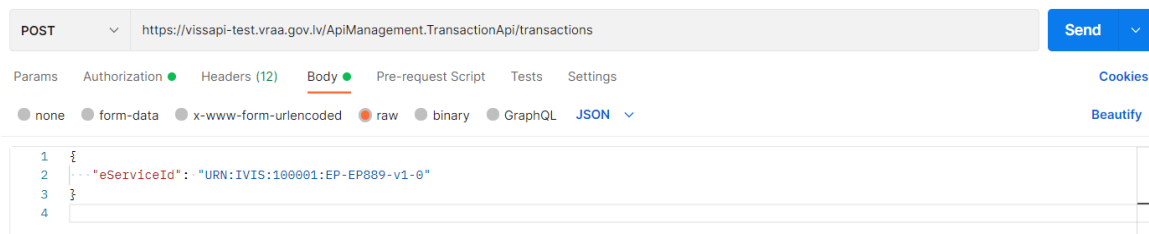


!Svarīgi: Ja transakcijas iegūšanai tiek izmantots Datu apmaiņas numurs, izsaukumā nepieciešams norādīt arī iepriekš iegūto piekļuves talonu

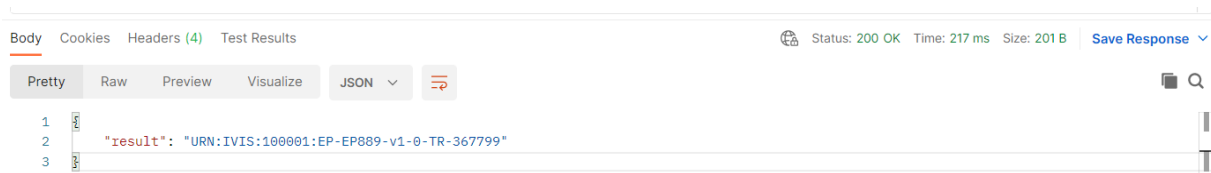
Body sadaļā jāizvēlas raw formāts un jāievada:

```
{
  "eServiceId": "URN:IVIS:100001:EP-EP889-v1-0"
}
```

Testa vidē testējot servisu šis numurs varbūt jebkāds, ja vien tas ir reģistrēts numuru katalogā. Datu apmaiņas nodrošināšanai šo numuru piešķir VDAA. **Produkcijas vidē ir atlaufs izmantot tikai iestādei izdoto numuru!**



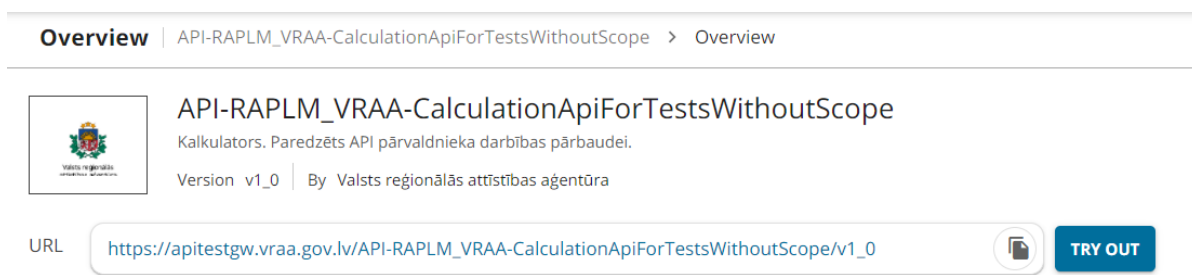
Saņemtā atbildē būs ģenerēta transakcijas numura vērtība, kura būs nepieciešams nākamajos soļos.



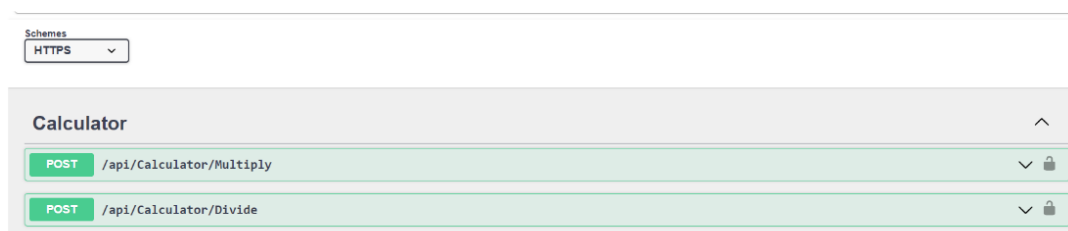
4. Servisa izsaukšana

Izsaukuma metode – POST

Izsaukuma saitē jānorāda ISS, API vai GraphQL Endpoints no API Pārvaldnieka.



Aiz pamata endpoint daļas jānorāda izvēlētas metodes endpoint. Tās iespējams apskatīt API Pārvaldnieka portālā pie abonētā API, sadaļā Try Out, piemēru skatīt bildē.



Norādāmie parametri Header sadaļā ir atkarīgi no paša servisa. Obligātie ir norādīti zemāk:

x-transactionId - Iepriekš saņemtais transakcijas numurs

Authorization – Bearer + iepriekš saņemtais piekļuves talons (access token)

Content-Type un Accept - atkarīgs no katra servisa specifikas

POST ▼ https://apitestgw.vraa.gov.lv/API-RAPLM_VRAA-CalculationApiForTestsWithoutScope/v1_0/api/Calculator/Multiply Send ▼

Params Authorization Headers (13) Body ● Pre-request Script Tests Settings Cookies

Headers 8 hidden

KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets ▼
☒ Content-Type	application/json				
☒ x-tabId	{{ \$guid }}				
☒ Accept	application/json, text/json, application/xml, text/xml				
☒ x-transactionId	URN:IVIS:100001:EP-EP889-v1-0-TR-367799				
☒ Authorization	Bearer urn:uuid:b0ad4fbb-f2f8-472a-aad1-9c7889aedf5...				
Key	Value	Description			

Body sadaļa formējas atbilstoši servisa aprakstam. Konkrētajā piemērā parādīts raw + JSON Body datu tips. Atkarībā no servisa uzbūves tipi mēdz mainīties un tad jānorāda atbilstošs tips un formāts, kādā tiks veikts izsaukums. Ja izvēlētais serviss ir SOAP serviss, tad formāts vienmēr būs XML un metode POST, REST servisa gadījumā būs JSON vai kāds cits no rīkā piedāvātiem tipiem.

Piemērā tas ir JSON.

POST ▼ https://apitestgw.vraa.gov.lv/API-RAPLM_VRAA-CalculationApiForTestsWithoutScope/v1_0/api/Calculator/Multiply Send ▼

Params Authorization Headers (13) Body ● Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON ▼ Beautify

```

1  {
2    "Multiplier": 12,
3    "Multiplicand": 3
4  }

```

Body Cookies Headers (16) Test Results Status: 200 OK Time: 1420 ms Size: 633 B Save Response ▼

Pretty Raw Preview Visualize JSON ▼ 🔍

```

1  36.0

```

1. Pielikums

```

2. openapi: 3.0.1
3. info:
4.   title: API-RAPLM_VRAA-EmptyExample
5.   version: v1_0
6. servers:
7.   - url: /
8. security:
9.   - default: []
10. paths:
11.   /getExample:
12.     get:
13.       summary: 'Vispārīga informācija '
14.       description: Metodes apraksts
15.       parameters: []
16.       responses:
17.         '200':
18.           description: ok
19.       security:

```

```

20.     - default:
21.       - API-RAPLM_VRAA-EpmtyExample-v1_0-ScopeName
22.     x-auth-type: Application & Application User
23.     x-throttling-tier: Unlimited
24.     x-wso2-application-security:
25.       security-types:
26.         - oauth2
27.     optional: false
28. components:
29.   securitySchemes:
30.     default:
31.       type: oauth2
32.       flows:
33.         implicit:
34.           authorizationUrl: 'https://test.com'
35.           scopes:
36.             API-RAPLM_VRAA-EpmtyExample-v1_0-ScopeName: Scope to provide an example for users
37.           x-scopes-bindings:
38.             API-RAPLM_VRAA-EpmtyExample-v1_0-ScopeName: ''
39. x-wso2-auth-header: Authorization
40. x-wso2-cors:
41.   corsConfigurationEnabled: false
42.   accessControlAllowOrigins:
43.     - '*'
44.   accessControlAllowCredentials: false
45.   accessControlAllowHeaders:
46.     - authorization
47.     - Access-Control-Allow-Origin
48.     - Content-Type
49.     - SOAPAction
50.     - apikey
51.     - Internal-Key
52.   accessControlAllowMethods:
53.     - GET
54.     - PUT
55.     - POST
56.     - DELETE
57.     - PATCH
58.     - OPTIONS
59. x-wso2-production-endpoints:
60.   urls:
61.     - 'http://swagger.io'
62.   type: http
63. x-wso2-basePath: /example/v1_0
64. x-wso2-transport:
65.   - http
66.   - https
67. x-wso2-response-cache:
68.   enabled: false
69.   cacheTimeoutInSeconds: 300
70.

```

5. Kļūdas API Pārvaldniekā

Šajā dokumentā aprakstītas iespējamās kļūdas API Pārvaldniekā, to novēršana, un sagatavotas atbildes uz kļūdu pieteikumiem API Pārvaldnieka lietotājiem.

Kļūdas API Pārvaldnieka servisu izsaukumos

Kļūdas kods: 405 Method Not Allowed

Kļūdas paziņojums: "Method not allowed for given API resource"

Kļūdas cēlonis/novēršana: API izsaukums tiek veikts ar metodes tipu, kura nav reģistrēta šim API (GET/POST/PUT/utt.). Jāpārlicinās, kādu metodi norāda izsaukumā un kādas ir reģistrētas API Pārvaldniekā.

Kļūdas kods: 400 Bad Request

Kļūdas paziņojums: Obligātais header atribūts x-transactionId nav norādīts.

Kļūdas cēlonis/novēršana: API Servisa izsaukumā Header daļā jānorāda iepriekš izgūtais transakcijas numurs parametrā x-transactionId

Kļūdas kods: 403 Forbidden

Kļūdas paziņojums: "Access failure for API: nosaukums, version: v1_0 status:- Resource forbidden "

Kļūdas cēlonis/novēršana: Serviss, kuru vēlas izsaukt nav abonēts API Pārvaldnieka izstrādātāju (Developer) portālā. Nepieciešams Devportālā to abonēt (subscribe).

Ja it kā ir abonēts, tad jāsalīdzina, vai talona izgūšanā tiešām norāda tās pašas aplikācijas ClientId un Secret.

Kļūdas kods: 403 Forbidden

Kļūdas paziņojums: "The access token does not allow you to access the requested resource",

Kļūdas cēlonis/novēršana: Drošības talons, ar kuru veic servisa izsaukumu nesatur vajadzīgo atļauju (scope). Vai nu PFAS sadaļa pie klienta lietojuma nav pievienota aplikācija, vai arī izgūstot talonu nenorāda Body daļā šo vajadzīgo scope

Kļūdas kods: 401 Unauthorized

Kļūdas paziņojums: "Invalid Credentials. Make sure you have provided the correct security credentials"

Kļūdas cēlonis/novēršana: Tiek lietots nederīgs/nepareizs drošības talons. Jāpārlicinās vai nemēģina servisu testa vidē izsaukt ar produkcijas vides talonu, vai otrādi.

Kļūdas kods: 500 Internal Error

Kļūdas paziņojums: "Unclassified Authentication Failure"

Kļūdas cēlonis/novēršana: Izsaukumā tiek lietots drošības talons, kuram ir beidzies termiņš, nepieciešams izgūt jaunu talonu un tad veikt izsaukumu.

Kļūdas kods: 404 Not Found

Kļūdas paziņojums: "Runtime error" "No matching resource found for given API Request"

Kļūdas cēlonis/novēršana: Kļūdaini norādīts izsaukuma URL, var būt, ka nepareizi norādīta konkrēta metode, kuru vēlas izmantot. Jāpārbauda kā izsaukumā norādīts Context+metode.

Ja URL + metode norādīta korekti, tad, iespējams, kļūda pašā API Pārvalndiekā. Jāpārbauda statuss pie endpoint.

Kļūdas kods: 500 Internal Error

Kļūdas paziņojums: " com.netflix.zuul.exception.ZuulException"

Kļūdas cēlonis/novēršana: Kļūda veicot servisu izsaukumus. Kļūda servisa īpašnieka pusē, jāgaida kamēr viņi atrisina



5.1. Kļūdas TransactionApi izsaukumā

Kļūdas kods: 403 Forbidden

Kļūdas paziņojums: "Nepietiekamas tiesības"

Kļūdas cēlonis/novēršana: Kļūda tiek saņemta gadījumā, ja mēģina izgūt Transakciju izmantojot DA numuru. Izsaukumā nepieciešams papildus padod Bearer drošības talonu. Ja tas tiek jau darīts, jāpārliedz, vai sertifikāts un DA numurs reģistrēti zem vienas un tās pašas iestādes. Transakciju ar DA numuru iespējams uzsākt tikai ar sertifikāta autentifikāciju.

Kļūdas kods: 400 Bad Request

Kļūdas paziņojums: "Identifikators nav reģistrēts katalogā"

Kļūdas cēlonis/novēršana: Kļūda tiek saņemta gadījumā, ja izgūstot transakciju tiek lietots neeksistējošs vai kļūdainis EP vai DA numurs. Jāpārbauda vai tiek lietots testa/produkcijas videi atbilstošs numurs, un vai šāds URN ir reģistrēts Sitecore.

Kļūdas kods: 400 Bad Request

Kļūdas paziņojums: "Neatbilstošs URN formāts"

Kļūdas cēlonis/novēršana: Kļūda tiek saņemta gadījumā, ja transakcijas izsaukumā eServiceId ir norādīts neatbilstoši vajadzīgajam formātam. Nepieciešams lietot:

EP gadījumā – “URN:IVIS:100001:EP-EPXX-V1-0”

DA gadījumā - "URN:IVIS:100001:EP.DA-DAXXX-V1-0"

5.2. Kļūdas drošības talona (access_token) izsaukumā

Kļūdas kods: 401 Unauthorized

Kļūdas paziņojums:

"error": "invalid_client",

"error_description": "urn:oauth2:{kļūdainais ClientId}"

Kļūdas cēlonis/novēršana: Drošības talona /token izsaukumā kļūdaini norādīts ClientId un/vai ClientSecret. Iespējams, ka tiek izmantoti testa vides ClientId un ClientSecret produkcijas vides drošības talonu izgūšanas adresei vai otrādi. ClientId jānorāda be “urn:oauth2:”

Kļūdas kods: 400 Bad Request

Kļūdas paziņojums: “The user ‘{PFAS lietotājs}’ failed validation using the PFAS”

Kļūdas cēlonis/novēršana: Kļūdaini norādīts PFAS lietotājvārds un/vai parole vai arī beidzies lomas termiņš. Iespējams, ka ārējie lietotāji norāda API Pārvaldnieka lietotājvārdu/paroli, ja tā, tad jāinformē, ka jāizmanto sertifikāta autentifikācija.

Kļūdas kods: 400 Bad Request

Kļūdas paziņojums: "The input is not a valid Base-64 string as it contains a non-base 64 character, more than two padding characters, or an illegal character among the padding characters. "

Kļūdas cēlonis/novēršana: Veicot drošības talona izgūšanu ar sertifikātu, publiskā daļa tiek norādīta ar "-----BEGIN CERTIFICATE-----\ un -----END CERTIFICATE-----". Pie publiskās atslēgas nepieciešams norādīt tikai pašu teksta virkni, bet pie privātās nepieciešams saglabāt arī -----BEGIN/END RSA PRIVATE KEY-----.

Kļūdas kods: -

Kļūdas paziņojums: “There was an error in evaluating the Pre-request Script:undefined: undefined”

Kļūdas cēlonis/novēršana: Kļūdaini aizpildīta Pre-request script sadaļa. Iespējams, ka nav norādīts -----BEGIN/END RSA PRIVATE KEY----- pie privateKey vērtības. Ja tas ir, tad jāpārbauda visa sintakse, vai nav kāda drukas kļūda.

Kļūdas kods: 400 Bad Request

Kļūdas paziņojums: "IDX10214: Audience validation failed. Audiences: 'https://ha.viss.gov.lv/STS/Viss.Pfas.STS/oauth2/token'. Did not match: validationParameters.ValidAudience...."

Kļūdas cēlonis/novēršana: Veicot izsaukumu ar Postman nesakrīt URL adrese ar to, kura norādīta “var audience”. Vai arī <https://ha.vraa.gov.lv/STS/VISS.Pfas.STS/oauth2/token> vietā tiek izmantots <https://ha.vraa.gov.lv/STS/Viss.Pfas.STS/oauth2/token>

Kļūdas kods: 400 Bad Request

Kļūdas paziņojums: “Unknown_error”; "ID4058"

Kļūdas cēlonis/novēršana: Sertifikāts, kurš tiek izmantots izsaukumā ir vai nu deaktivizēts/ nav aktīvs vai arī tam beidzies termiņš. Otrs variants, ka tiek lietots produkcijas vides sertifikāts testa vides talona iegūšanai vai arī otrādāk.

5.3. Kļūdas API Pārvaldnieka portālos

Kļūdas paziņojums: “Error! You have partially-created keys. Please click `Clean Up` button and try again.”
“Error occurred when generating application keys, General Error”

“Kļūda! Jūs esat daļēji izveidojis atslēgas. Lūdzu, noklikšķiniet uz pogas 'Notīrīt' un mēģiniet vēlreiz. Atslēgu ģenerēšanas kļūda. Aplikācijas nosaukums neatbilst sintaksei. Skatīt dokumentāciju.” “Ģenerējot Klienta lietojuma atslēgas, radās kļūda, General Error”.

Kļūdas cēlonis/novēršana: Klienta lietojuma aplikācijas nosaukums nav izveidots atbilstoši vadlīnijās noteiktajam formātam. Nepieciešams rediģēt aplikācijas nosaukumu, lai tas atbilstu šādas formātam: APP-{Iestādes_identifikators}-{Aplikācijas_nosaukums}