VALSTS REĢIONĀLAS ATTĪSTĪBAS AĢENTŪRA

API Pārvaldnieka publicētāju un izstrādātāju portāls

Skaidrojumi un izmantošana

v2.6

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.

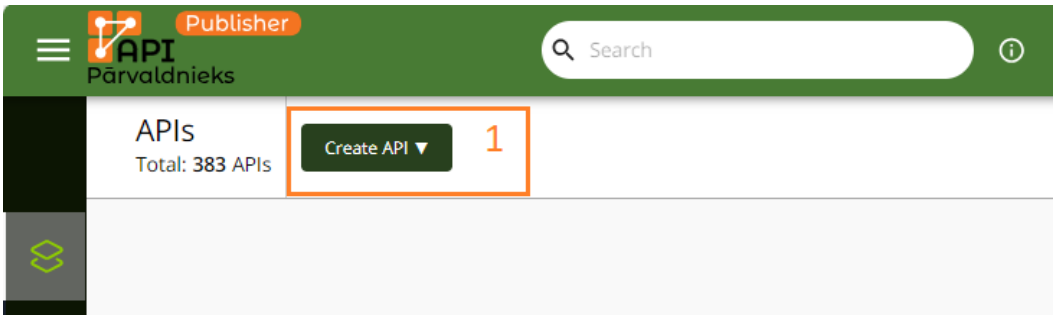
Kopā ar šo dokumentu aicinām izmantot Datu apmaiņas izveides vadlīniju dokumentu, kas ir pieejams viss.gov.lv portālā API Pārvaldnieka sadaļā (https://viss.gov.lv/lv/Informacijai/Dokumentacija/Koplietosanas_komponentes/API_Parvaldnieks), kur ir aprakstītas datu apmaiņas iespējas, to veidošana, izsaukumu piemēri un citi ieteikumi, kā arī informācija, kura sniedz atbalstu datu apmaiņu izstrādātājiem.

1. API Publicētāju portāls

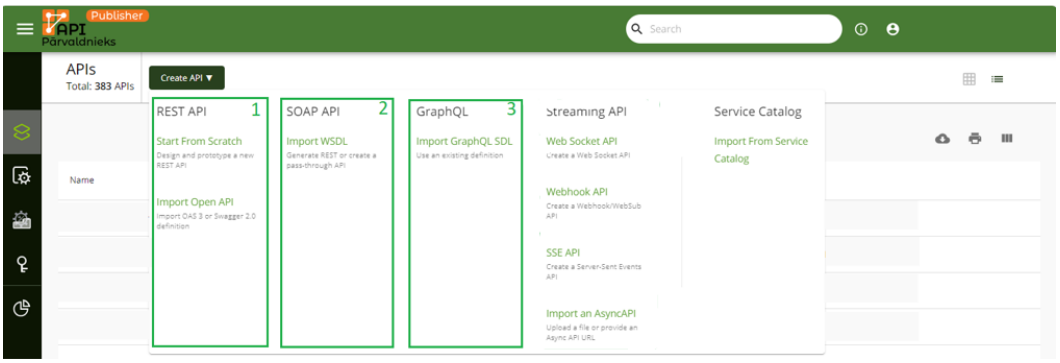
API Publicētāju (Publisher) portāls paredzēts API reģistrēšanai, publicēšanai un pārvaldībai. Portāls lietotājam tiek attēlots valodā, kāda ir iestatīta interneta pārlūkprogrammā.

1.1 Sadaļa "Create API"

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

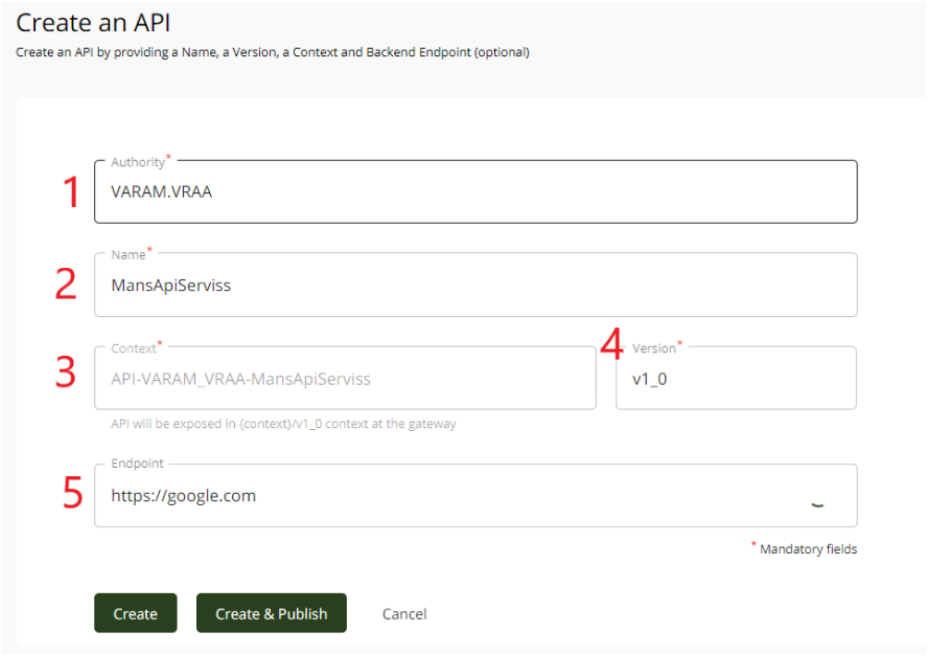


Izvēlas veidu, kā pievienot API, atzīmējot vienu no trim opcijām:



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;



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

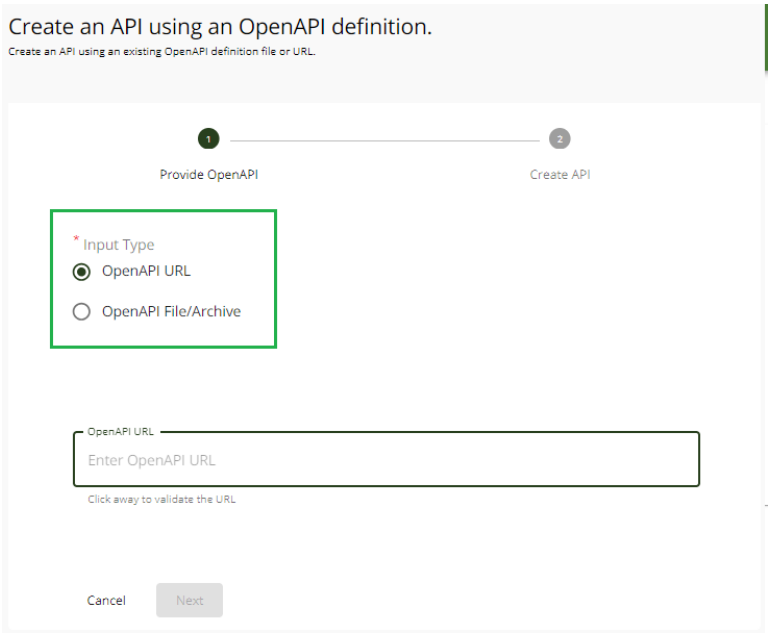
- Authority - Iestādes identifikators, kurš jānorāda tādā pat formātā kā ir klasifikatoru katalogā, piemēram, VARAM.VRAA
- Name – Servisa nosaukums, kurš nevar saturēt speciālos simbolus
- Context – Lauks aizpildās automātiski, ar informāciju no “Servisa veids”+”Authority”+”Name”. Šajā gadijumā API-VARAM_VRAA-MansApiServiss
- 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.

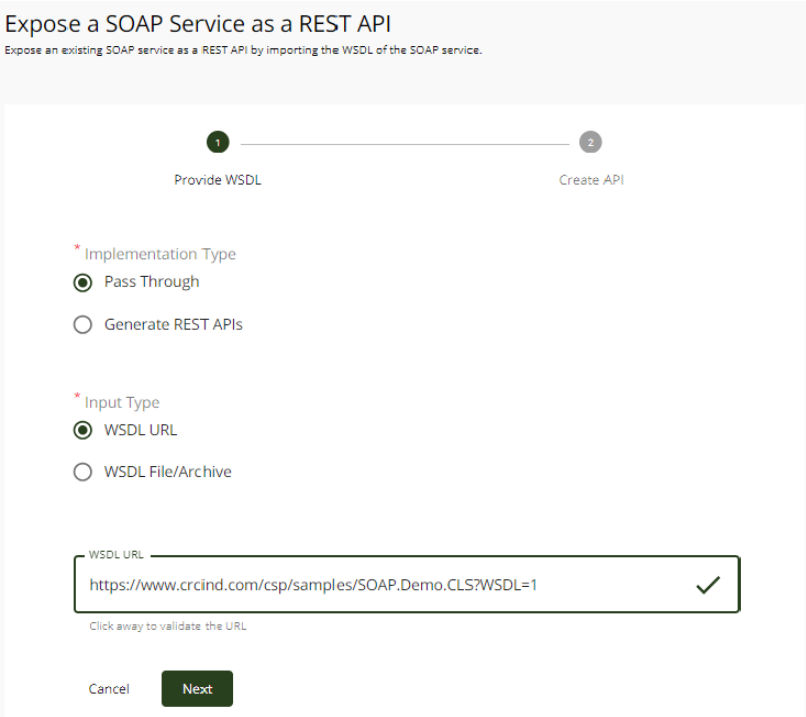
Swagger definīcija ir formāts, kādā apraksta REST API.



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



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).

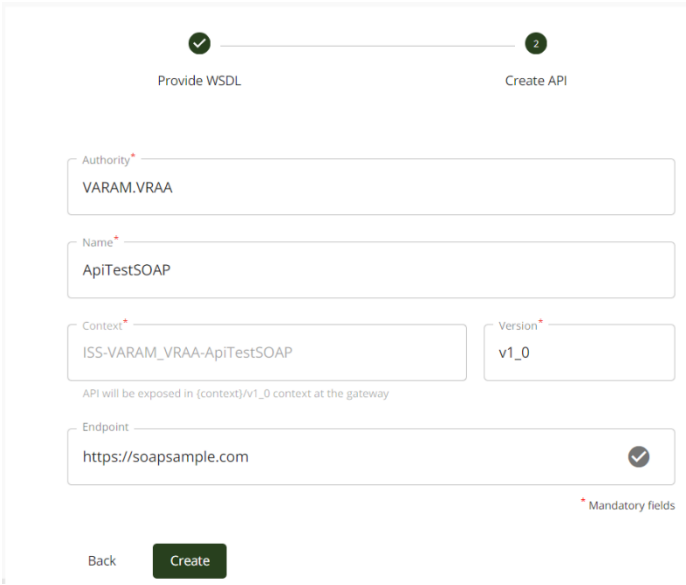
- Izvēloties "*Pass Through*" – tiek izveidots proxy SOAP pieprasījumiem, kas iet uz API Gateway.

- 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:



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

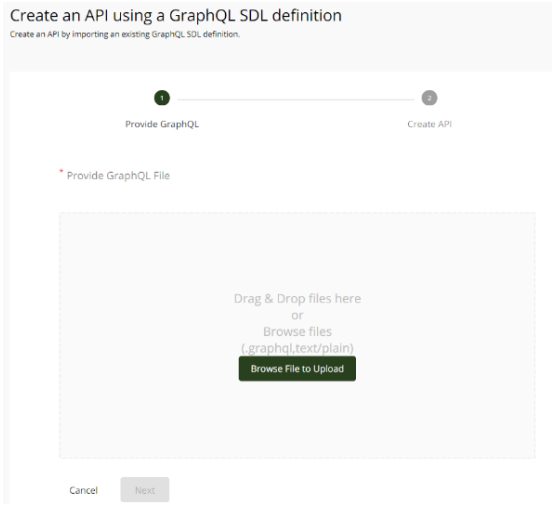
- Authority - Iestādes identifikators, kurš jānorāda tādā pat formātā kā ir klasifikatoru katalogā, piemēram, VARAM.VRAA
- Name – Servisa nosaukums, kurš nevar saturēt speciālos simbolus
- Context – Lauks aizpildās automātiski, ar informāciju no "Servisa veids"+"Authority"+"Name". Šajā gadijumā ISS-VARAM_VRAA-ApiTestSOAP
- 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'

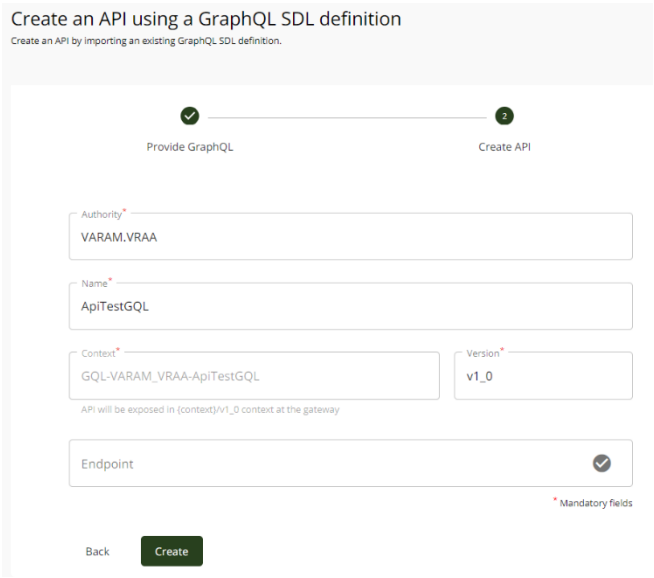
GraphQL - API vaicājumu valoda, kas strādā ar jūsu esošajiem datiem. **GraphQL** nodrošina saprotamu API datu aprakstu un pieprasījumos sniedz klientiem iespēju pieprasīt un saņemt tieši tos datus no servisa, kas viņiem nepieciešami.

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 gadijumā GQL-VARAM_VRAA-ApiTestGQL
- Version – Versijai jābūt norādītai noteiktā formāta notācijā – “v{Major}_{Minor}”, piemēram, v1_0
- Endpoint – Jāievada servisa galapunkts (Endpoint)

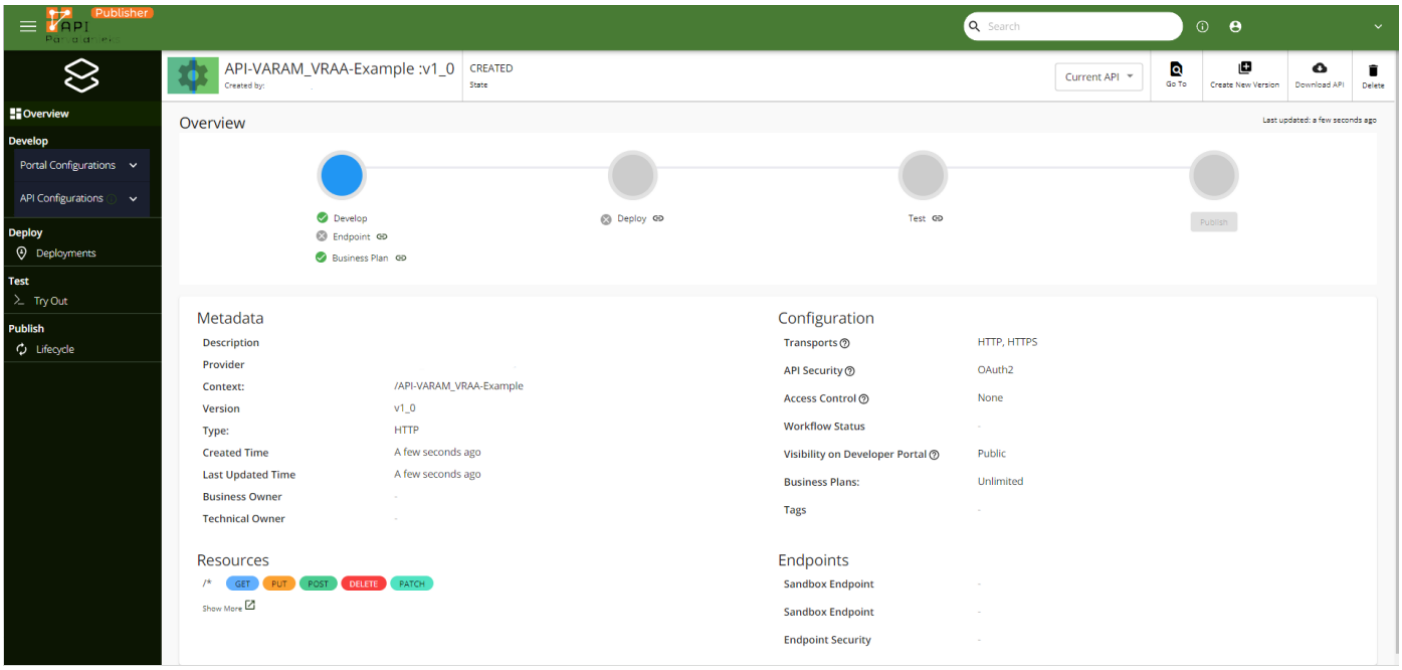
Servisa pilnais nosaukums būs tāds pats, kā šajā skatā redzamais ‘Context’

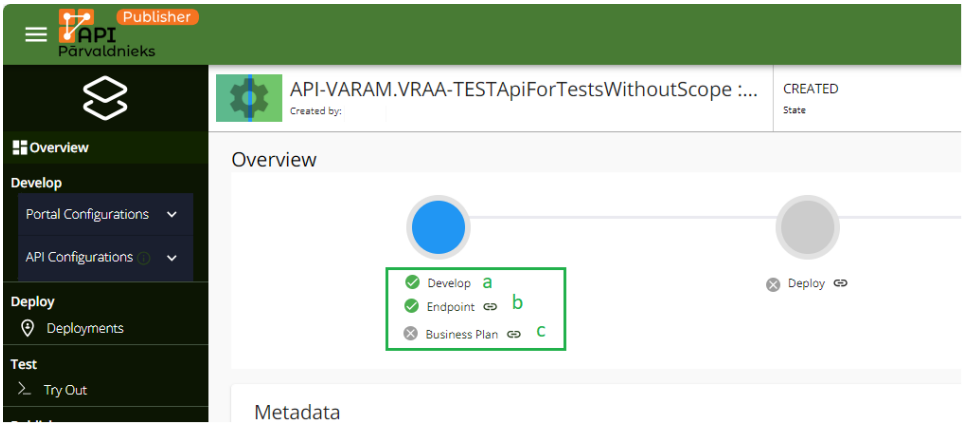
GraphQL servisu apraksts un noderīgas saites:
https://graphql.org/learn/
https://graphql.org/learn/schema/

1.2 Sadaļa “Overview”

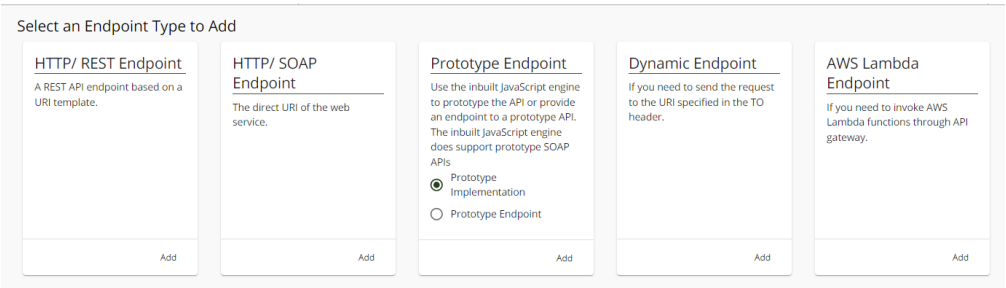
Vispārīgā informācija

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 principus, kas aprakstīti aktuālākajā *Datu apmaiņas izveides vadlīniju* dokumentā!**





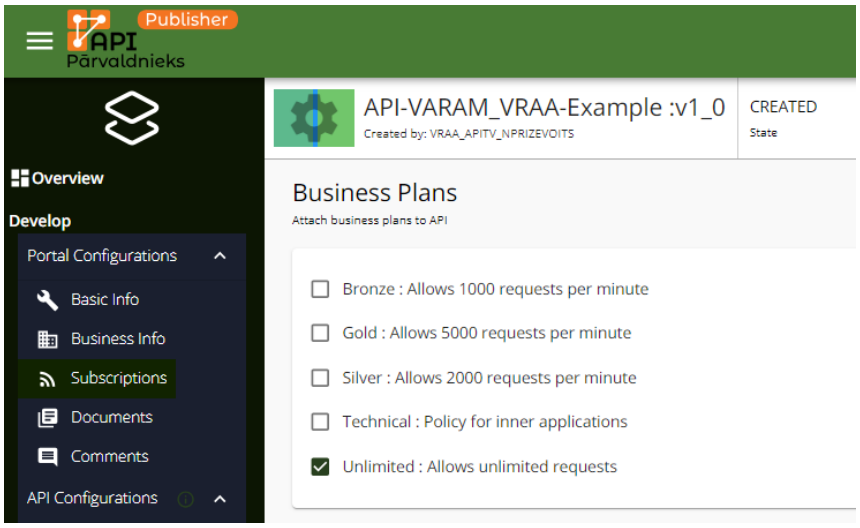
Izvēlne Develop (a) nav pieejama rediģēšanai, jo tā tiek aizpildīta pirmajā solī. Pievienojot jau eksistējošu API, API metodes tiek norādītas automātiski, jo tie aprakstītas Swagger definīcijā. Papildus tam, galapunktu iestatījumus var rediģēt sadaļā *Endpoint*. Ja API tiek veidots bez Swagger definīcijas,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



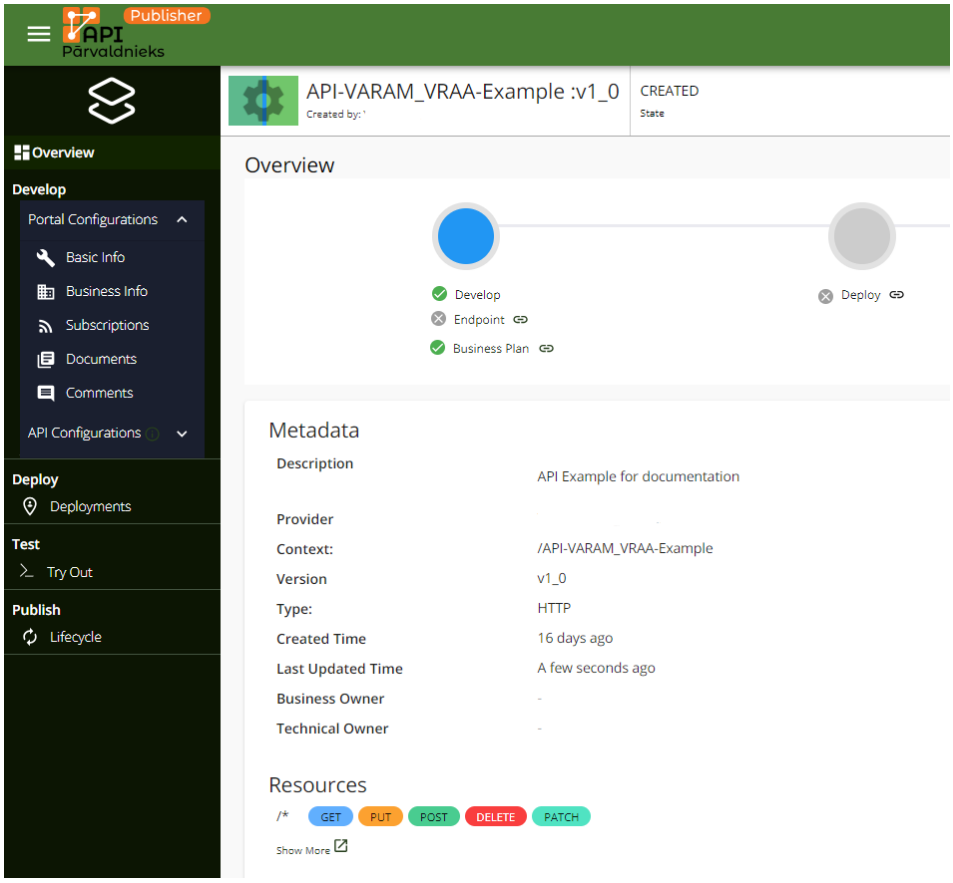
Izvēlnē *Endpoint*(b) jāizvēlas un jānokonfigurē atbilstošu galapunkta tipu un galapunkta saiti. Rediģējot un pievienojot galapunktu saites, obligāti kā pamata vidi ir jāizmanto *Production*. Pārējie tipi ir pieejami rediģēšanai, bet to izmantošana ir atkarīga no konkrēta klienta vajadzībām

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)

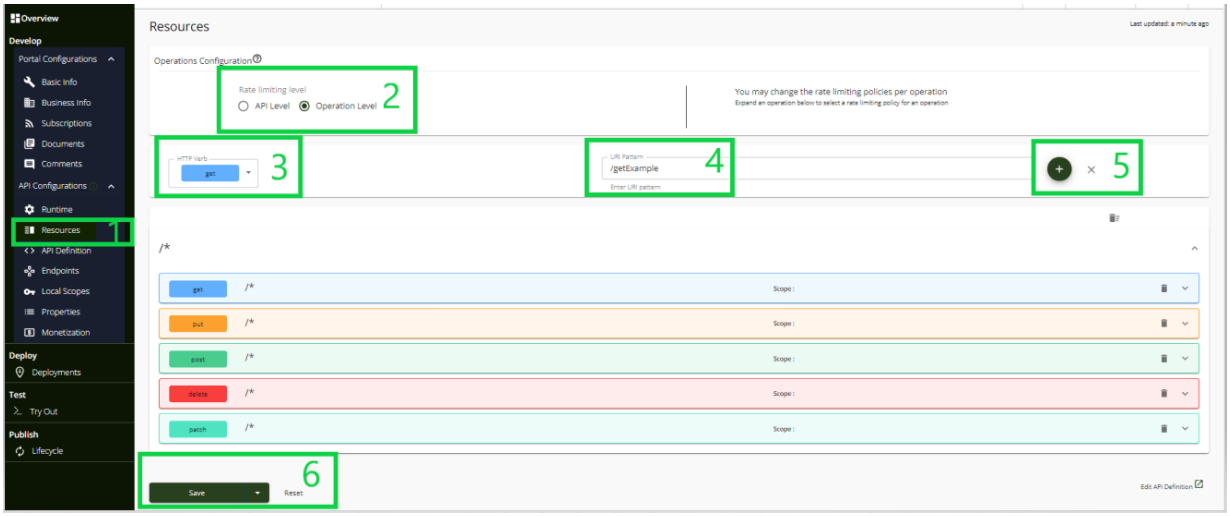


API definīciju, iespējams rediģēt vai tai pievienot jaunas metodes nospiežot pogu “Show More” pie Overview apakšpunkta *Resources*.



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:

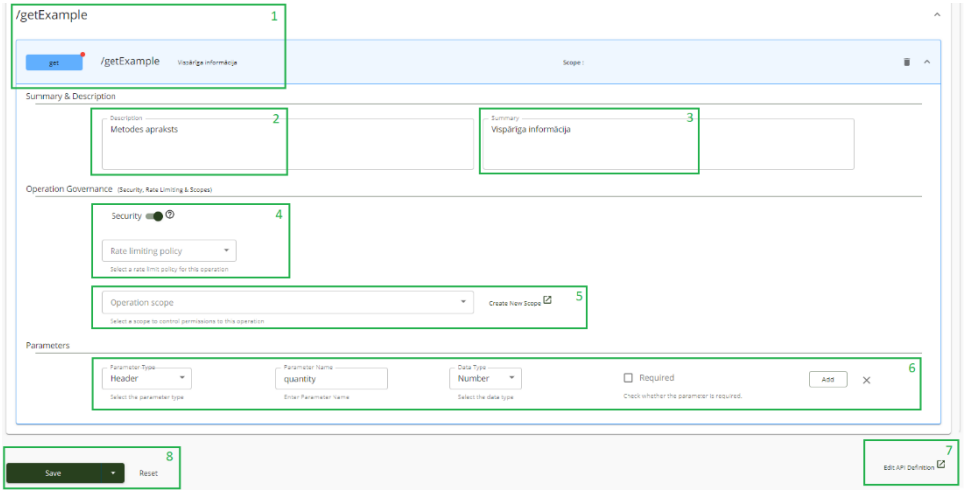
1. Atvērt sadaļu *Resources*;
2. Izvēlēties Operāciju limitēšanas līmeni (pēc noklusējuma ir *Operational Level*);
3. Izvēlēties nepieciešamo metode no saraksta;
4. Noradīt konkrēta galapunkta saites modeli (parasti "/foo" formātā);
5. Pievienot modeli servisam;
6. Saglabāt izmaiņas.



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.

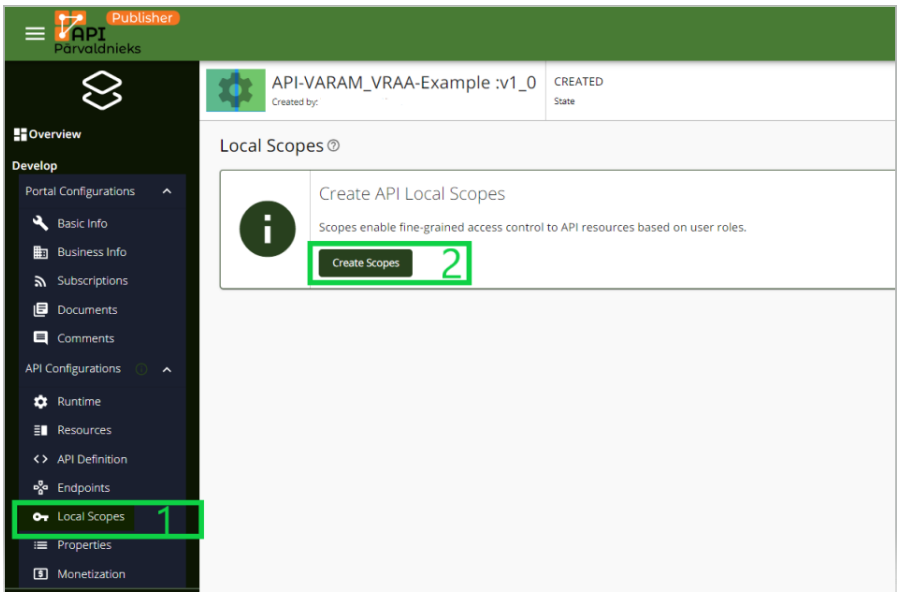
- **Data Type**– manuāli norāda padoto datu tipu, piemēram, integer (skaitļi) / string (simbolu virkne) / file / u.c.

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

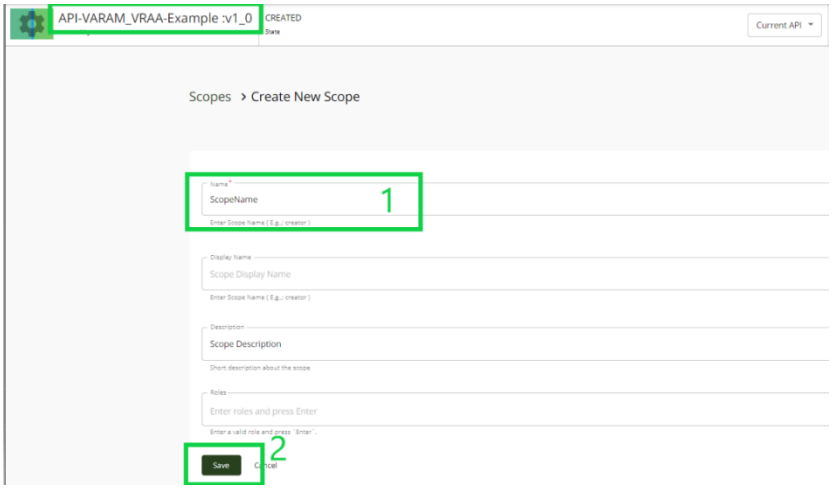
Atļauju (scope) pievienošana

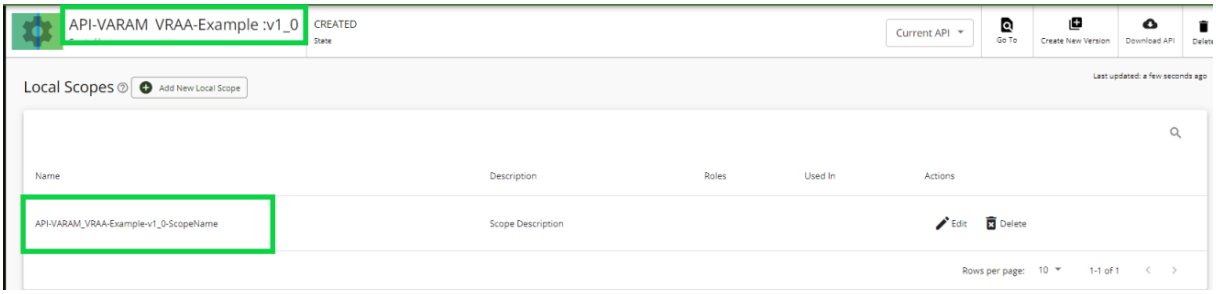
5. solī aprakstītajai servisa aizsargāšanai, jeb konkrētai metodes atļaujas (scope) piešķiršanai ir nepieciešams izveidot jaunu atļauju sadaļā *Local Scopes,* vai nospiežot pogu *Create New Scope* pie metodes modeļa izveides.

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



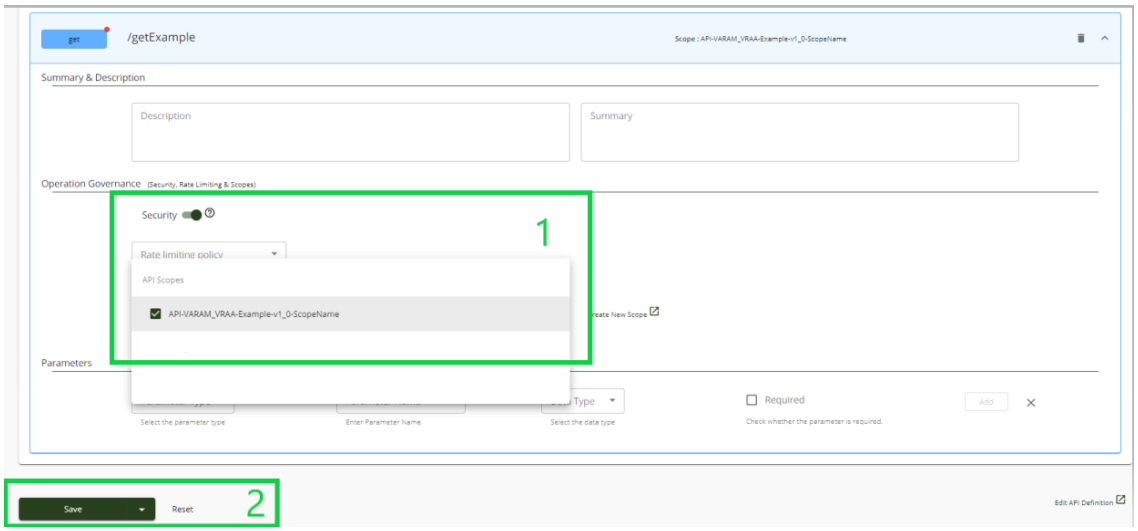
Atļaujas pilnais nosaukums, kurš būs arī jāizmanto atļauju piešķiršanai PFAS Uzticampo pušu sadaļā un API izsaukumos veidojas automātiski no "API nosauums + versija" + "Name".





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

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



1.3 Servisa atribūtu aizpilde

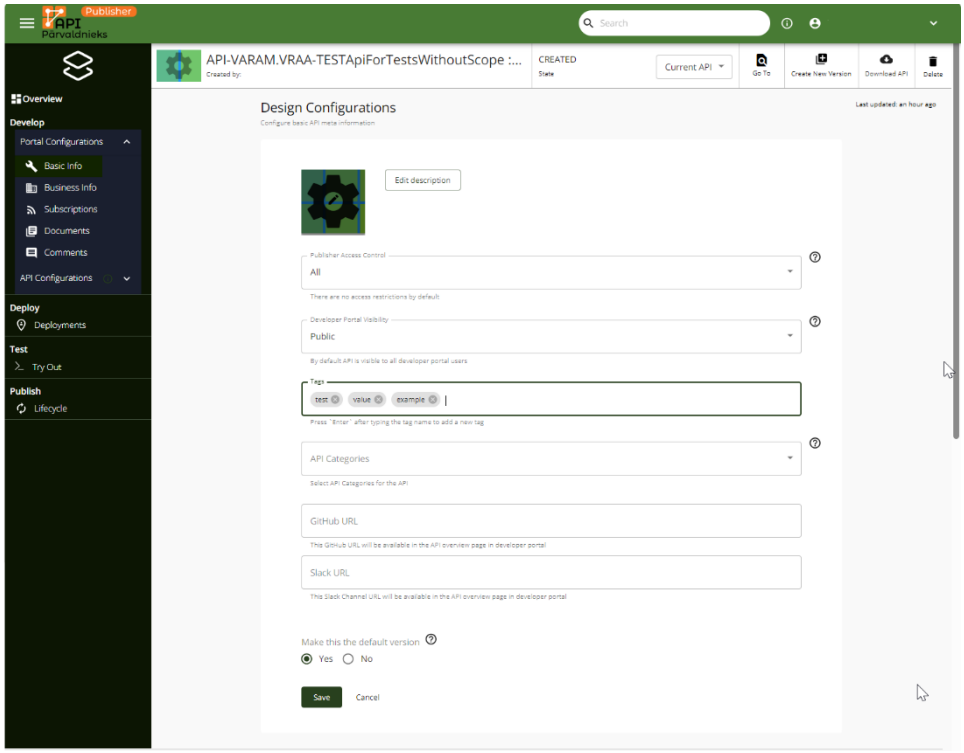
Pēc *Develop* soļa un pirms servisa izmitināšanas ir obligāti jāveic Servisa parametru aizpildi.

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** –

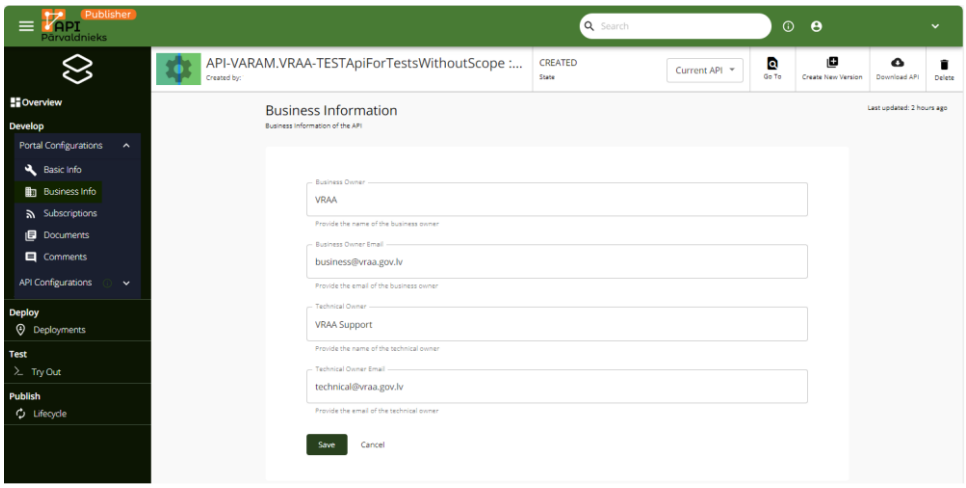
- **YES** – Servisa versija (ja ir vairākas) ar šo iezīmi tiks uzskatīta par primāro un pie vairākām servisa iterācijām var tikt izsaukta bez versijas apzīmējuma;
- **NO** – Servisa izsaukumā vienmēr būs jānorāda versiju.



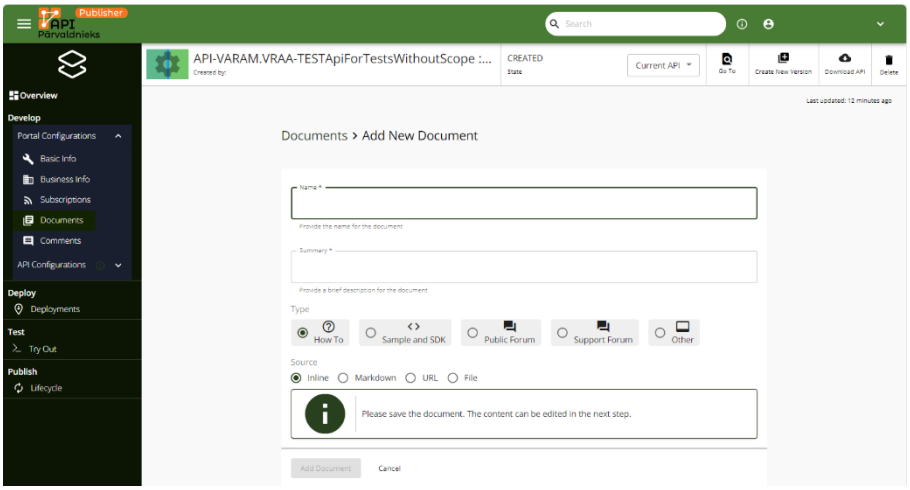
1.3.2 Business info sadaļa

Sadaļa ir radīta, lai servisa īpašnieks vai publicētājs varētu noradīt 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.

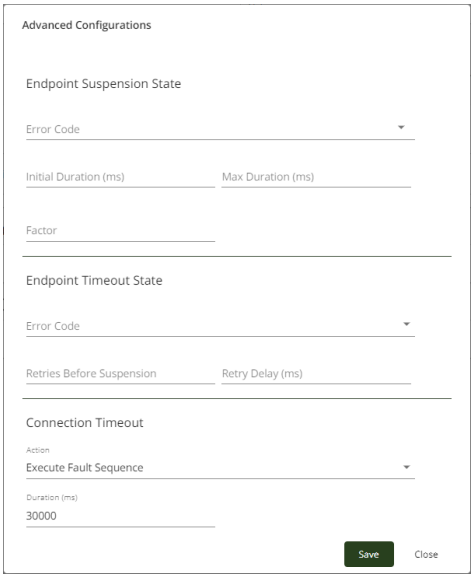


1.3.3 Documents sadaļa



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

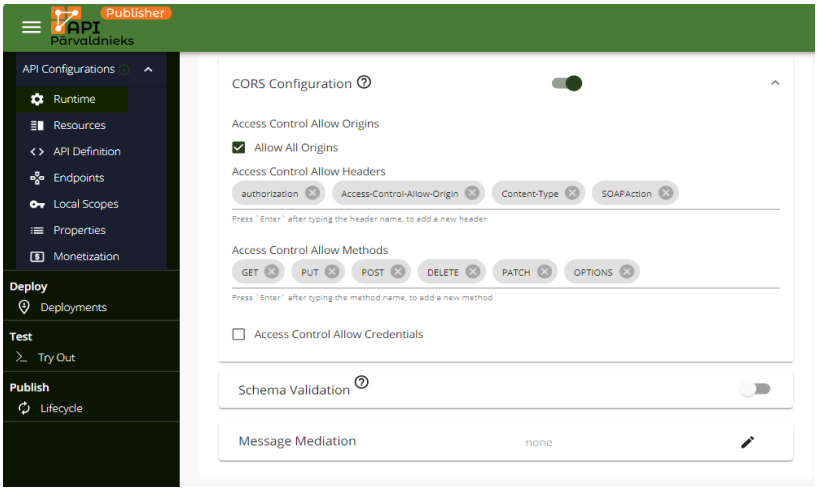
Izvēloties apakšsadaļu *Runtime* pie *API Configurations* sadaļas, iespējams konfigurēt norādītā galapunkta atgrieztās kļūdas aiztures un noildzes gadījumā. Iespējams izvēlēties vairākus kļūdu kodus.



1. *Endpoint Suspend State* – "aizturēts" galapunkts nesaņem pieprasījumus un neatgriež atbildes.
    a. **Initial Duration** – norāda, pēc cik ilga laika tiek sūtīts atkārtots ziņojums. Pēc noklusējama 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

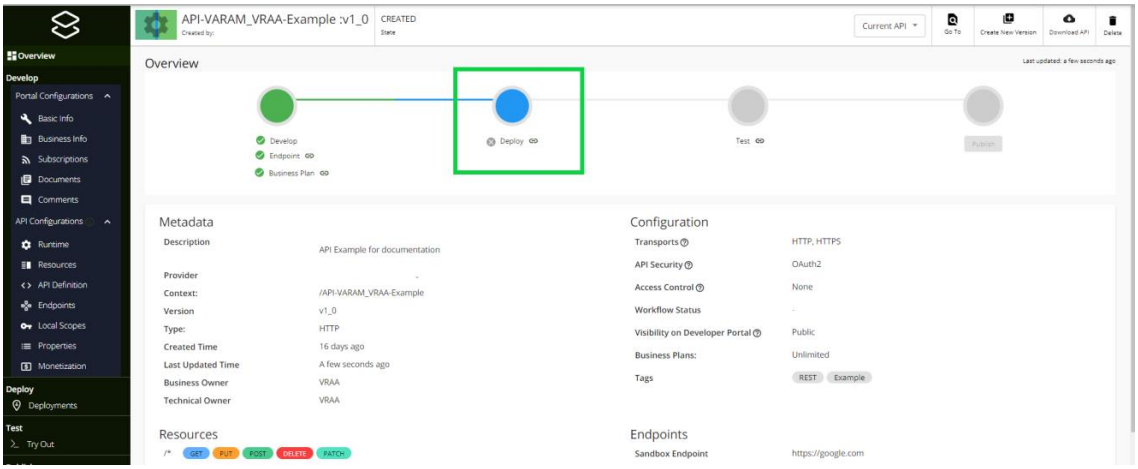
CORS (*Cross-Origin Resource Sharing*) ir mehānisms, kas ļauj piekļūt aizsargātiem resursiem no ārēja domēna (kas nav resursu domēns). CORS ir iespējams norādīt globāli visiem publicētajiem API. Parametra definēšanas pieejama sadaļā *Runtime*.



CORS apraksts un noderīgas saites
[Enabling CORS for APIs]

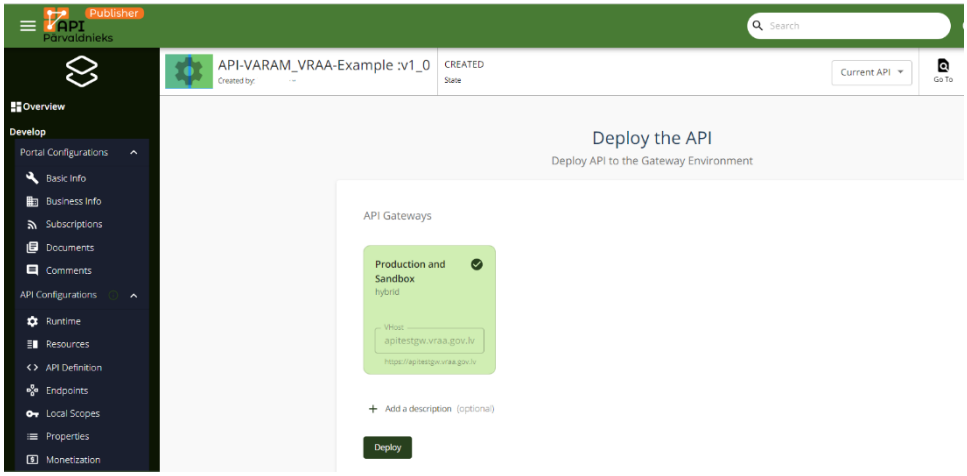
1.5 Sadaļa "Deploy"

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.

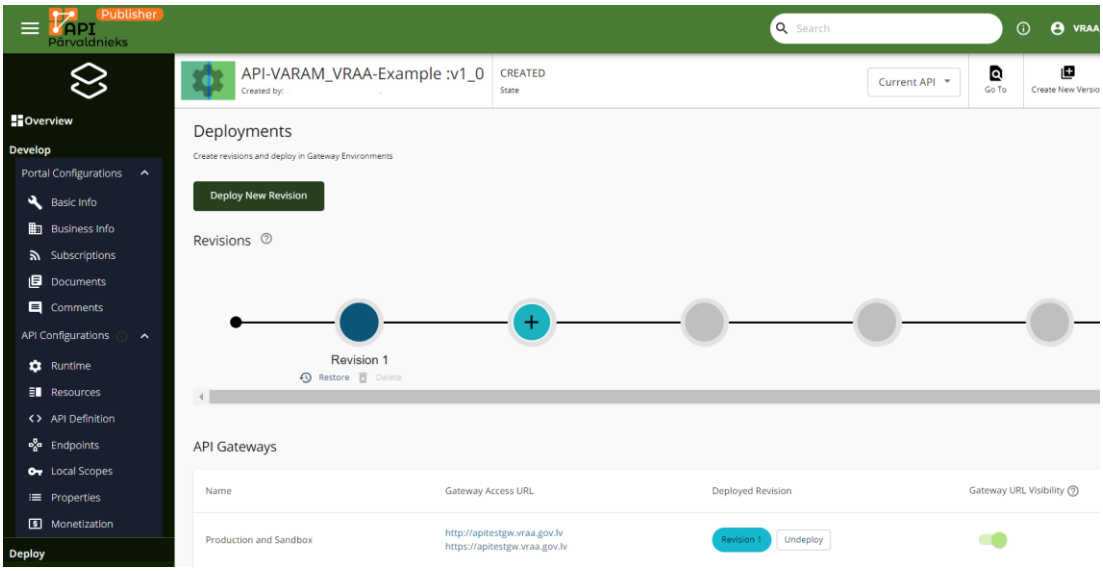


Turpmāk izvērsies Deploy sadaļas izvēlne, kur ir sekojoši lauki:

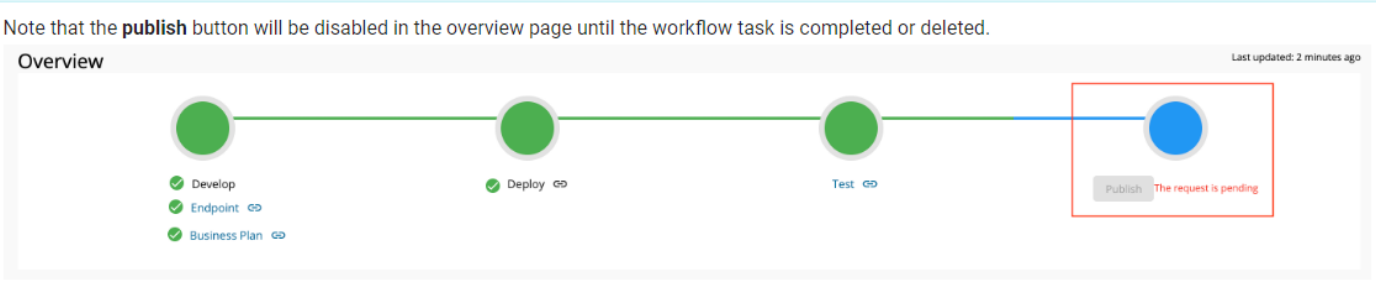
1. API publicēšanas vide, parasti paliek bez izmaiņām;
2. API galapunkta apraksts konkrētai servisa revīzijai;
3. Veikt izmitināšanu.



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



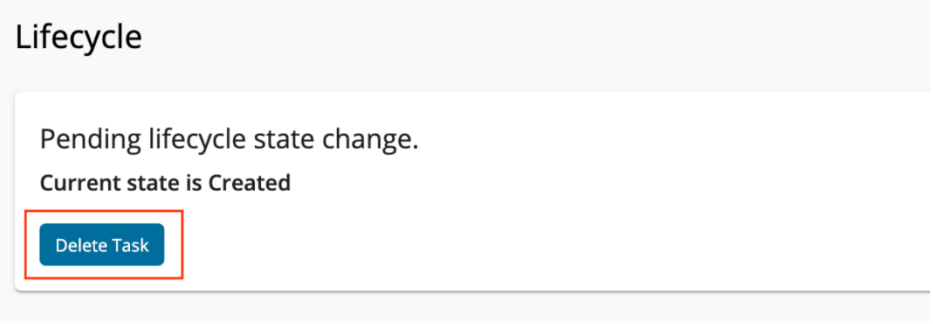
Pēc soļu izpildes nepieciešams pāriet uz Overview sadaļu. Produkcijas vidē, nospiežot 'Publish', VRAA tiks nosūtīts servisa publicēšanas pieprasījums, un tiks pārbaudīta servisa atbilstība vadlīnijām. Kamēr servisa publicēšanas pieprasījums nebūs apstiprināts, būs redzams statuss “The request is pending”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



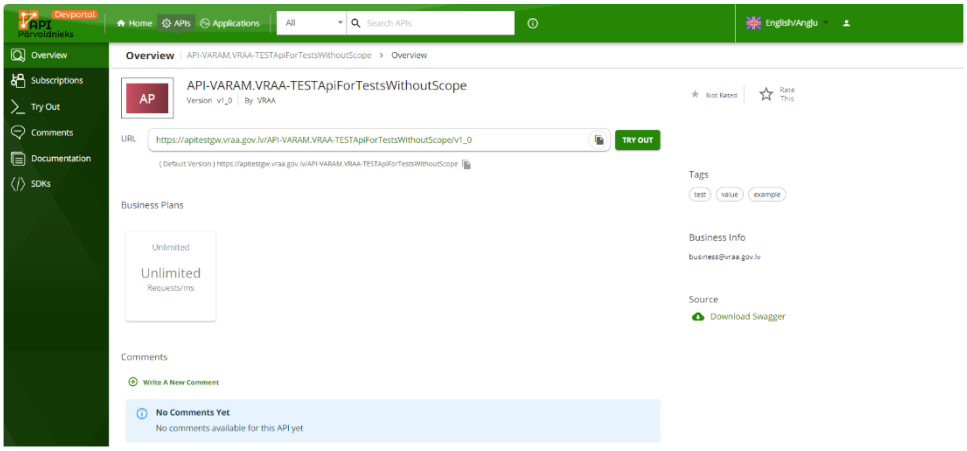
Lai serviss varētu tikt apstiptināts publicēšanai, tam jābūt izpildītiem šādiem kritērijiem:

- 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):



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 saņemts e-pastu ar ziņu, ka publicēšanas pieprasījums ir noraidīts ar uzskaitītiem trūkumiem, kurus nepieciešams izlabot pirms veikt atkārotu servisa publicēšanu.

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

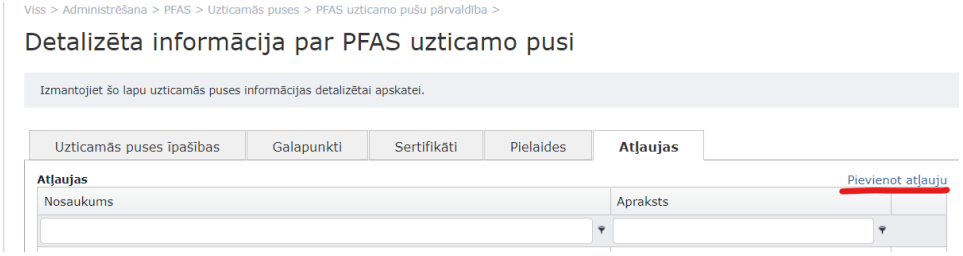
1.6 Uzticamo pušu pārvaldība

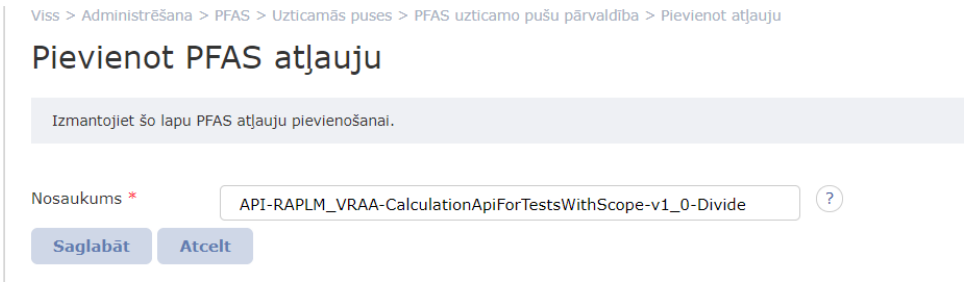
Atļauju piešķiršanas kārtība

Ja serviss ir aizsargāts ar atļauju (scope), tad to ir nepieciešams pievienot izveidotajam klienta lietojumam jeb aplikācijai, izmantojot visstv.vraa.gov.lv (testa vidē) vai viss.gov.lv (produkcijas vidē) portāla PFAS uzticamo pušu pārvaldības sadaļas. Tās atrodamas 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*.

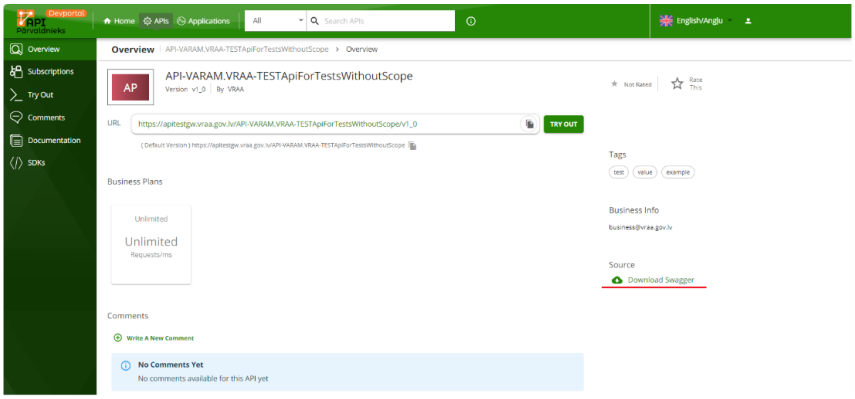
Nākošajā solī jāatrod API Izstrādātāju (Devportal) portālā izveidoto klienta lietojumu, ar kuru tika abonēts konkrēts serviss. To var atrast divos veidos: 1)Nosaukumā ievadot izveidotā klienta lietojuma nosaukumu. 2) Identifikatora laukā ievadot klienta lietojuma Consumer Key (Tā iegūšana aprakstīta 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)



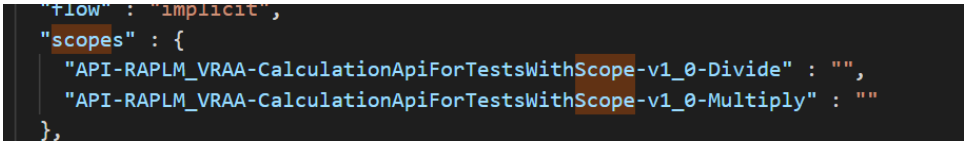


Piemērs ar atļaujas pievienošanu



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**.



2. API Izstrādātāju portāls

Lai piekļūtu servisu abonēšanas funkcionalitātei ir jāautentificējas 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 neautentificētiem lietotājiem.

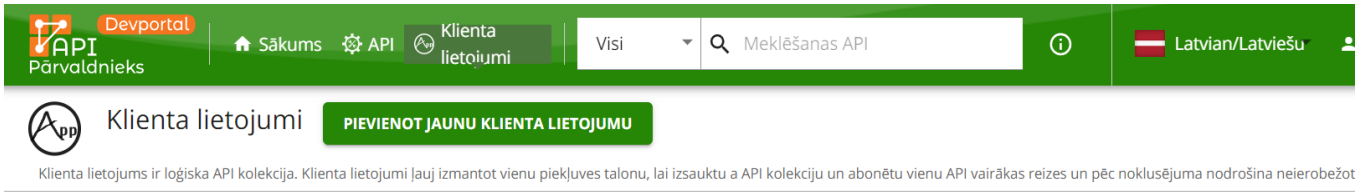
API Izstrādātāju portāls lietotājam attēlosies tādā valodā, kāda būs izvēlēta lapas galvenē

2.1. Klienta lietojuma izveide

Pirmais solis, pirms varētu sākt abonēt servisus 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, servisus 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).

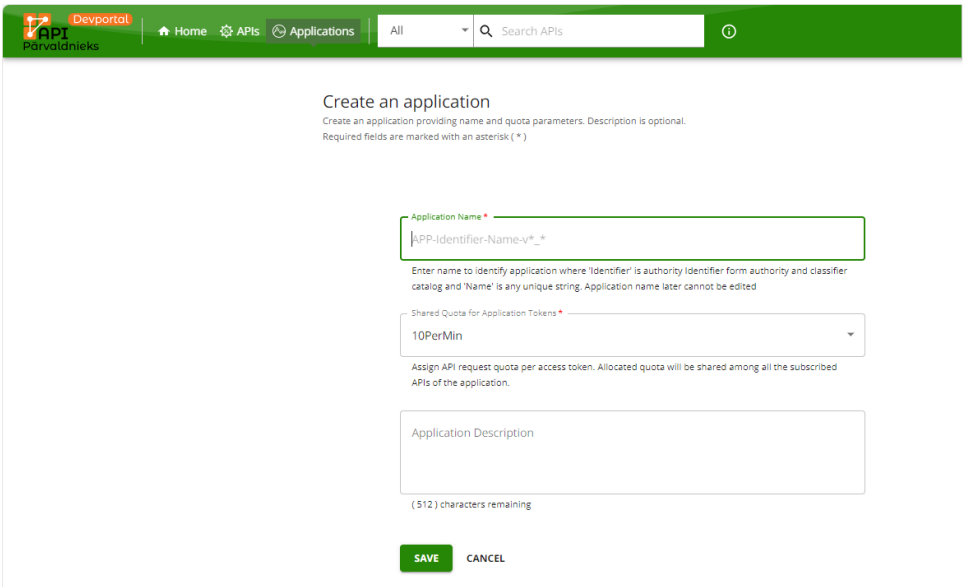
1. Lai pievienotu jaunu *Klienta lietojumu,* sadaļā *Klienta lietojumi* ir jāizvēlas *Pievienot jaunu klienta lietojumu.*



2. Nākošajā solī Klienta lietojuma nosaukums jāveido vadoties pēc Klienta lietojuma nosaukuma formāta notācijas: “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.



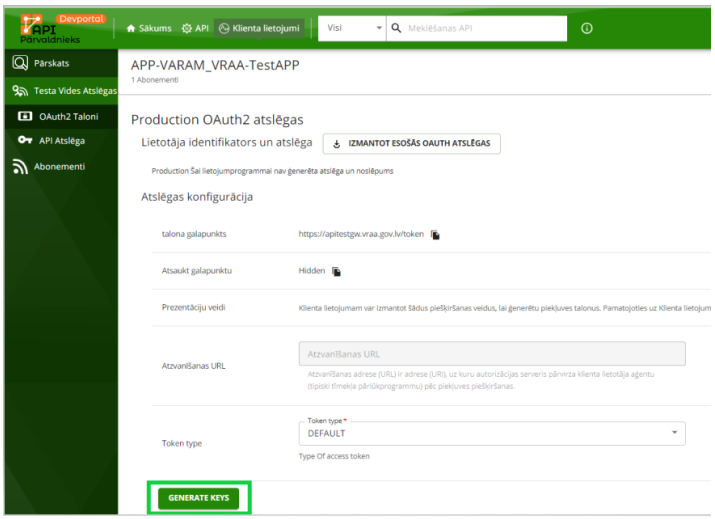
Klienta lietojuma īpašnieka maiņa

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@vraa.gov.lv

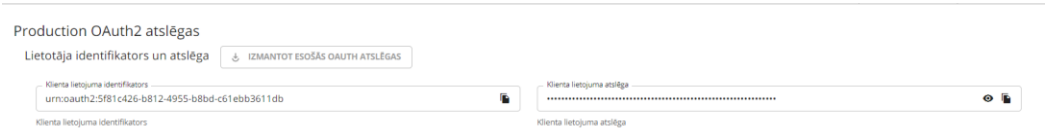
- Pieprasījumā jānorāda šāda informācija:
- Klienta lietojuma nosaukums un identifikators (ClientId)
- Esošā īpašnieka API Pārvaldnieka lietotājvārds
- Jaunā īpašnieka API Pārvaldnieka lietotājvārds
- Iestāde, kuras ietvaros tiks veikta aplikācijas īpašnieka maiņa

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".



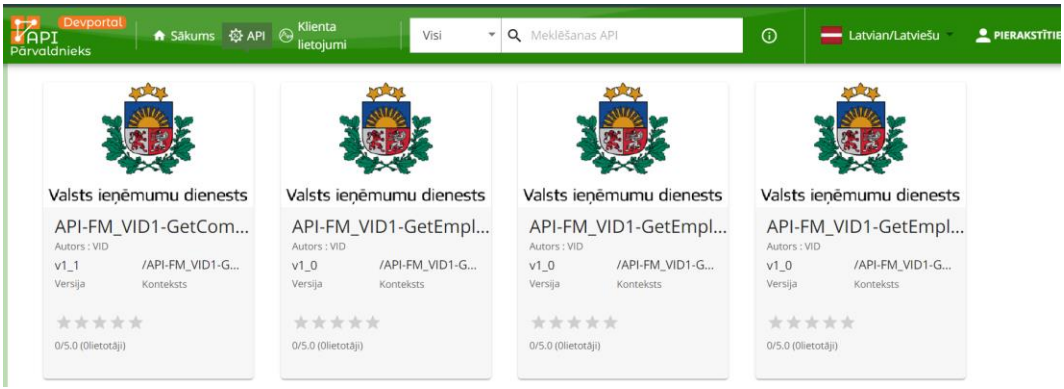
Ja klienta lietojuma nosaukums ir izveidots atbilstoši vadlīnijām, tiks uzģenerēts un parādīts Consumer Key un Consumer Secret, savukārt gadījumā, ja tiek attēlota kļūda par atslēgu ģenerēšanu – ir jāveic izveidotās aplikācijas nosaukuma precizēšana. (Skatīt 2.1. sadaļu)



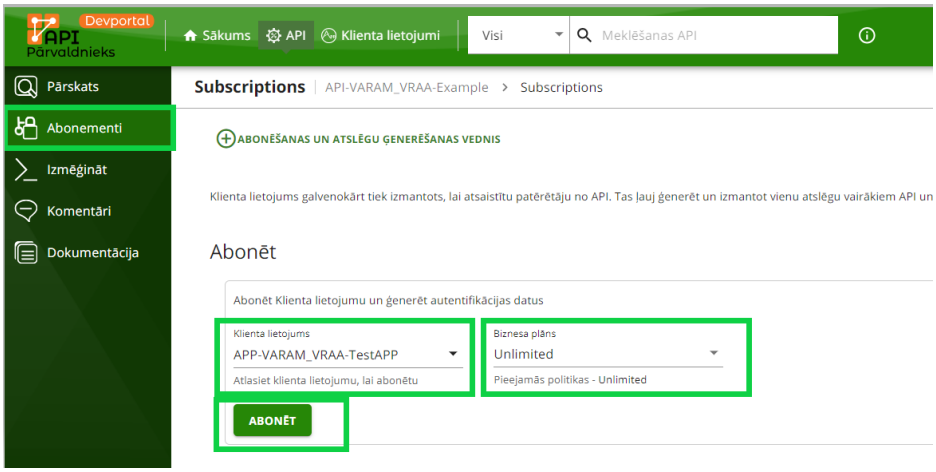
!Svarīgi: Piekļuves talona (access_token) iegūšanai **neizmantot** piedāvāto iespēju *CURL To Generate Access Token/CURL Piekļuves Talona Ģenerēšana***.** Tā vietā skatīt nodaļu 3.*"Servisu testēšana izmantojot Postman",* par piekļuves talona iegūšanu izmantojot sertifikāta autentifikāciju.

2.3. API abonēšana un izmantošana

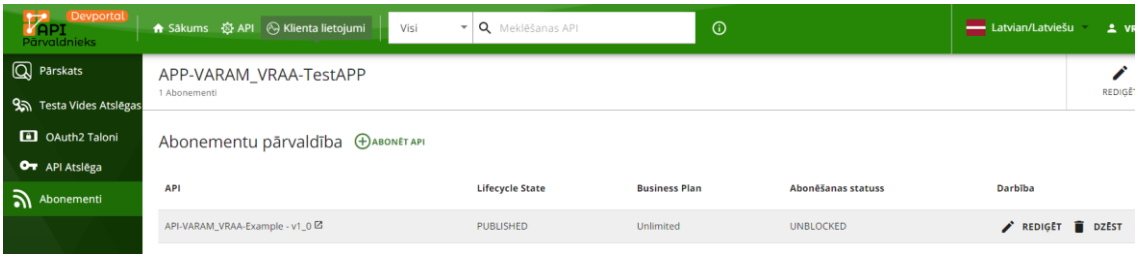
Pēc sekmīgas klienta lietojuma izveides ir iespējams izvēlēties un abonēt nepieciešamos servisus no API kataloga. Nepieciešamo servisu iespējams meklēt vai nu kopējā API katalogā, vai arī izmantojot meklētāja funkcionalitāti.



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).



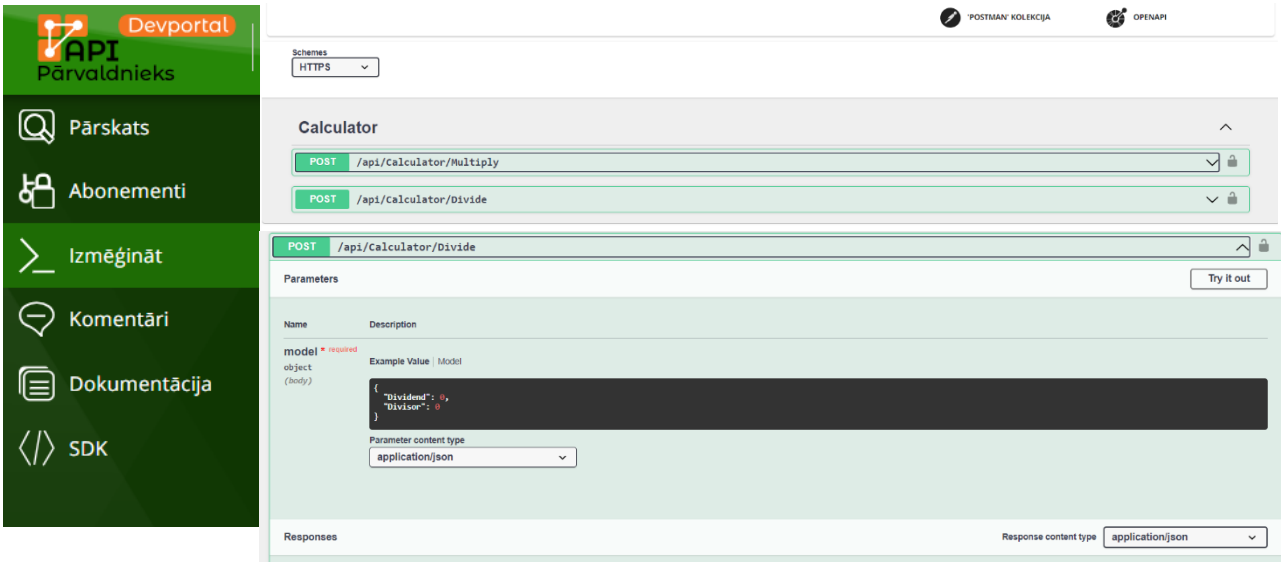
Savus abonētos API servisus iespējams redzēt *Klienta lietojumi* sadaļas *Abonementi* apakšsadaļā. Ja serviss ir pieejams šajā sadaļā, tas nozīmē, ka servisa abonēšana noritēja veiksmīgi.



Turpmāka servisu izmantošanas un izsaukšanas procedūra no tehniskā skatupunkta ir aprakstīta *Datu apmaiņas izveides vadlīniju* dokumentā

Try Out/Izmēģināt sadaļa

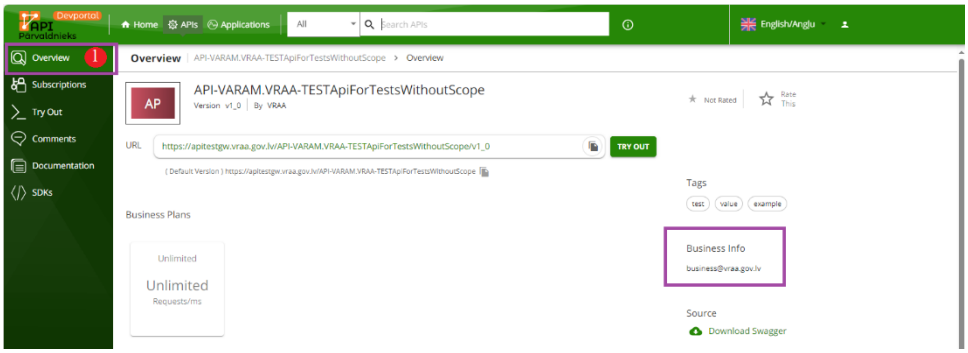
API sadaļā Izmēģināt/Try Out iespējams apskatīt konkrētajam servisam pieejamās metodes un izsaukuma piemērus:



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, VRAA 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 vēlēsies izmantot

Pēc atļaujas saņemšanas iespējams sākt izmantot šos servisus/metodes

3. Servisu testēšana izmantojot Postman

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

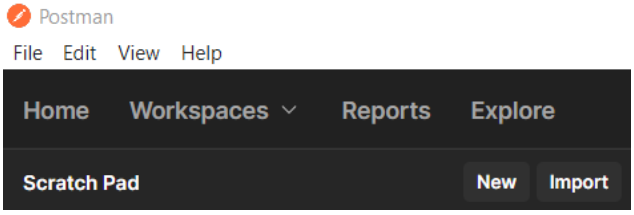
3.1. Piekļuves talona (access_token) iegūšana izmantojot sertifikātu

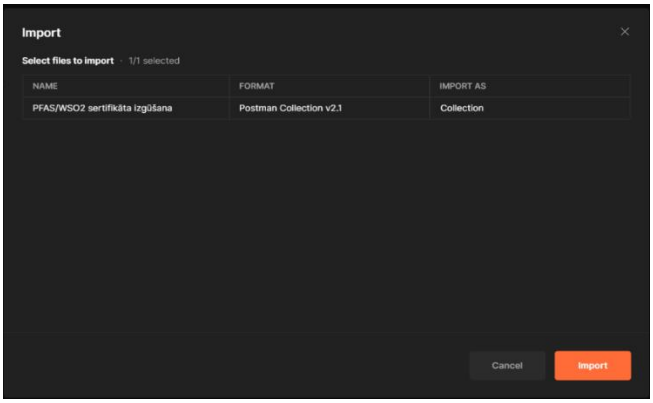
Nepieciešamais piekļuves talona iegūšanai:

a) PFAS Sertifikāts (Iegūšana aprakstīta - https://viss.gov.lv/lv/Informacijai/partneriem/Vadlinijas/Noradijumi-sert-parvaldibai/2-2)
b) Aplikācija API Pārvaldniekā
c) KeyStore Explorer vai līdzvērtīgs rīks
d) Postman v9.5.0 vai jaunāks

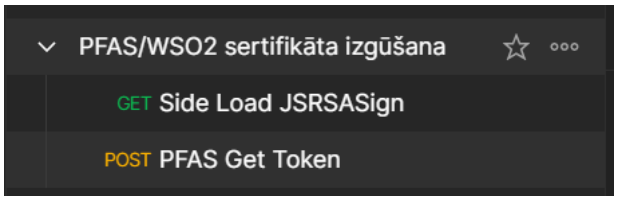
Soļi:

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:

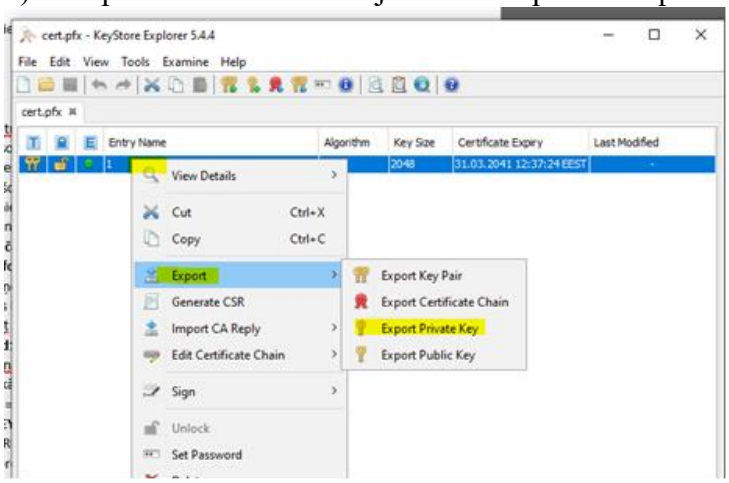
- var clientId – Klienta lietojuma identifikators (Consumer Key) (Iegūšana aprakstīta 2.2. nodaļā)
- var clientSecret – Klienta lietojuma atslēga (Consumer Secret) (Iegūšana aprakstīta 2.2. nodaļā)
- var privateKey – Privātā RSA atslēga, to iespējams iegūt ar KeyStore Explorer konvertējot izdoto PFX 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)

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

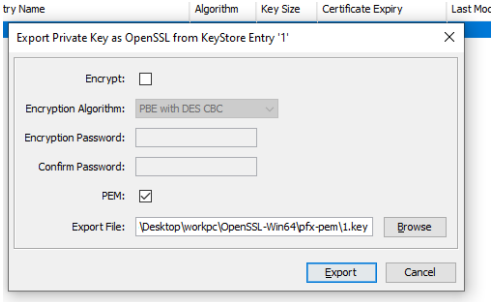
4. Privātās atslēgas iegūšana:

a)KeyStore Explorer rīkā jānospiež File -> Open -> Jāizvēlas sertifkāts 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

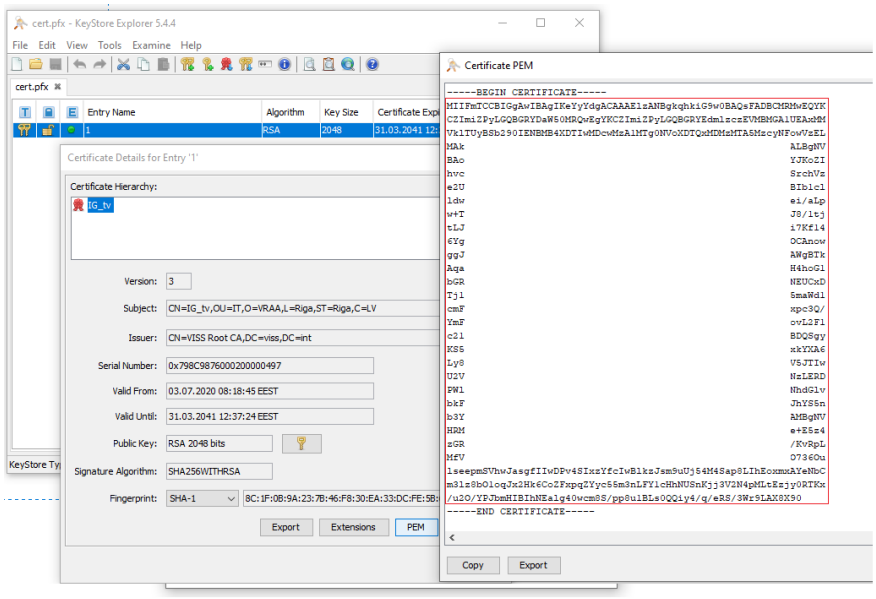
f) Iegūtā atslēga ir var privateKey parametrs, kurš jāiekopē Postman rīkā (3. solis šajā instrukcijā par piekļuves talona iegūšanu)

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

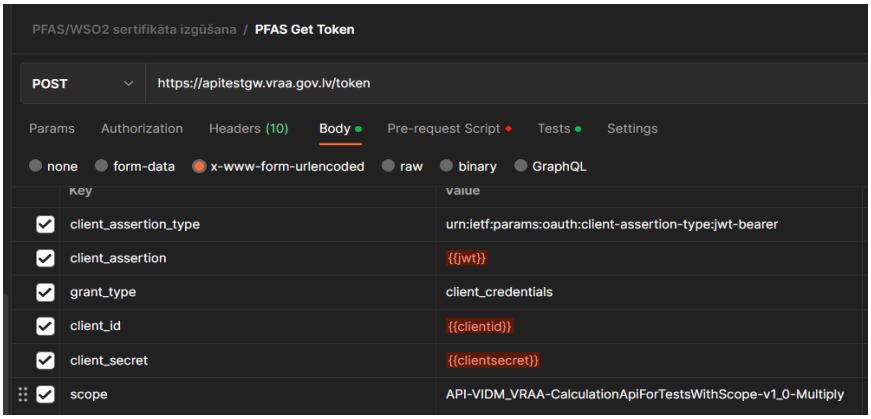


c)Iegūtā vērtība ir var publicKey parametrs, kurš jāiekopē Postman rīkā (3. solis šajā instrukcijā par piekļuves talona iegūšanu)

6. Pēc visu atribūtu aizpildes Pre-request Script sadaļā jāpalaiž Side Load JSRSASing un PFAS Get Token. Rezltātā 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)

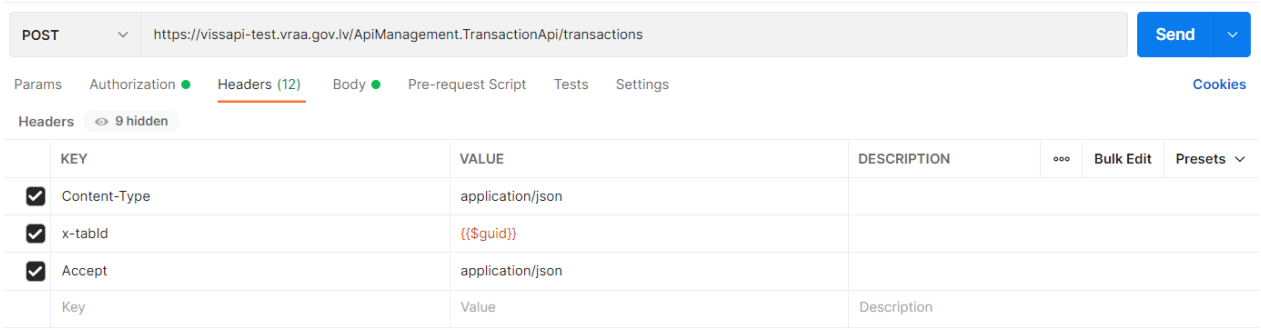
3.2. Transakcijas izveidošana servisa izsaukšanai

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 transasakcijas iegūšanai tiek izmantots Datu apmaiņas numurs, izsaukumā nepieciešams norādīt arī iepriekš iegūto piekļuves talonu

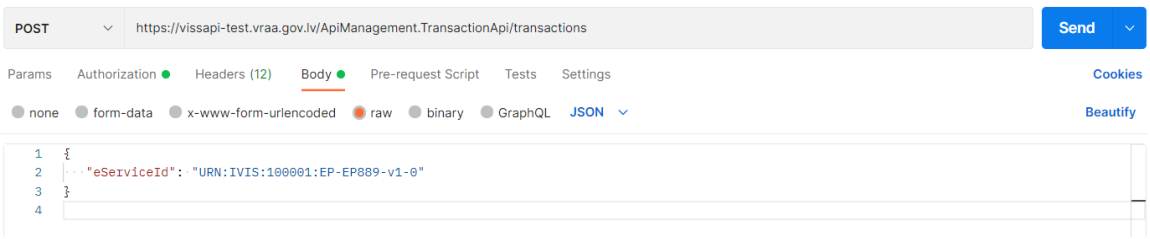
Body sadaļā jāizvēlas raw formāts un JSON, un jāievada:

{

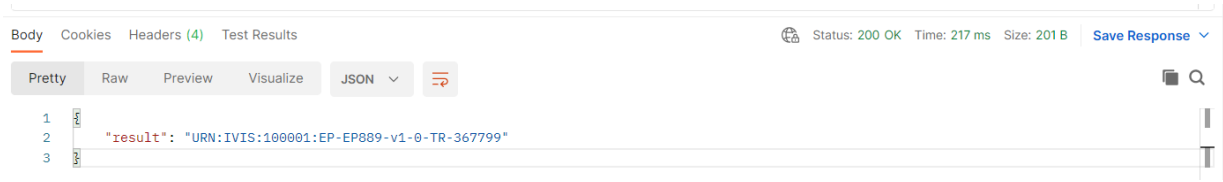
"eServiceId": "URN:IVIS:100001:EP-EP889-v1-0"

}

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



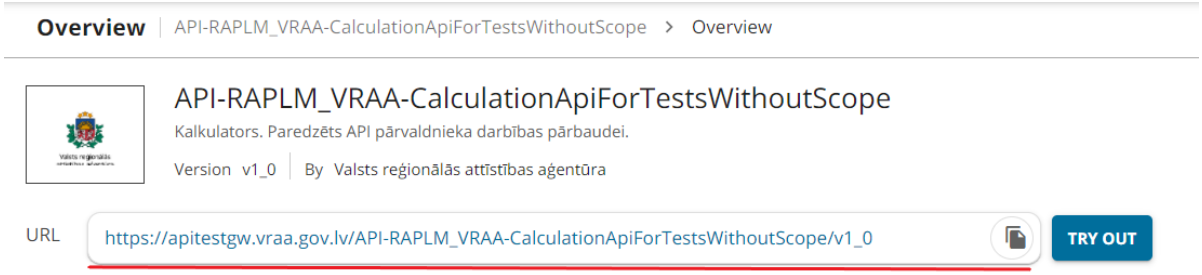
Saņemtā atbildē būs ģenerēta transakcijas numura vērtība, kura būs nepieciešams nākamajos soļos.



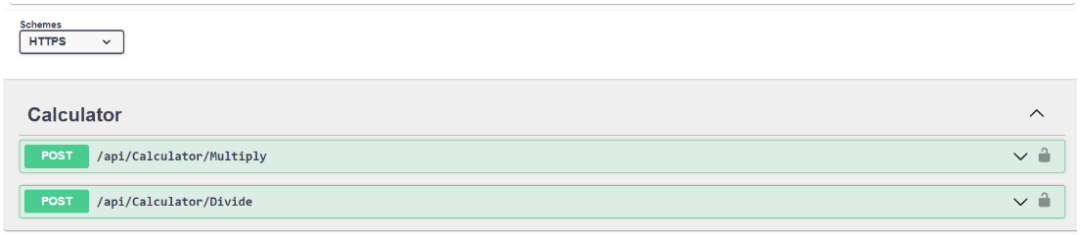
4. Servisa izsaukšana

Izsaukuma metode – POST

Izsaukuma saitē jānorada ISS, API vai GQL 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.

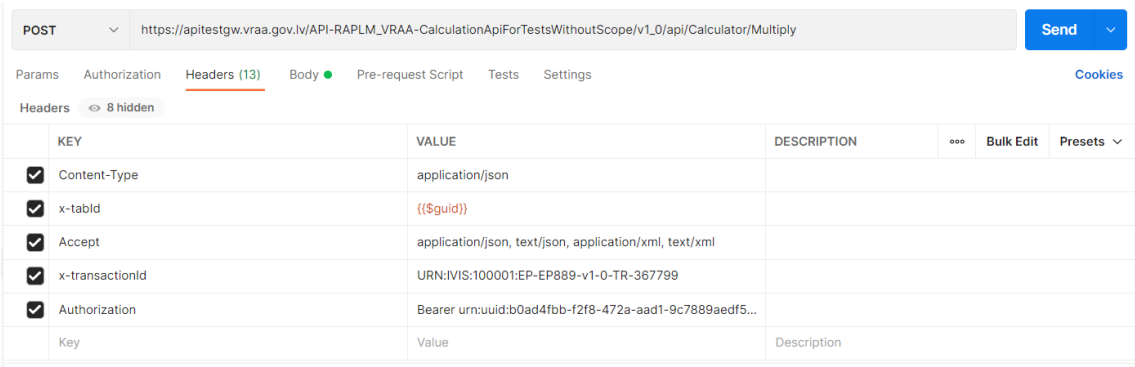


Norādāmie parametri Header sadaļā ir atkarīgi no paša servisa. Obligātie ir noradīti zemāk:

x-transactionId - Iepriekš saņemtais transakcijas numurs

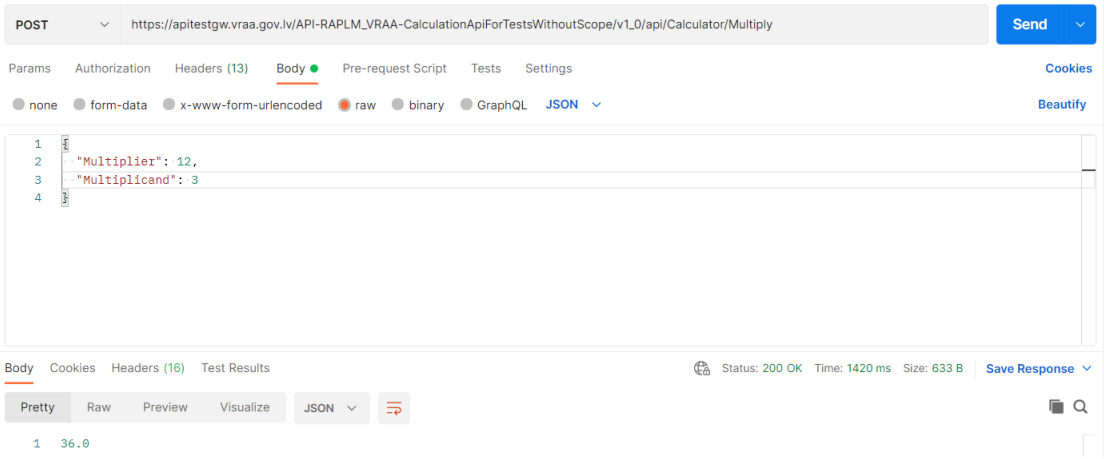
Authorization – Bearer + iepriekš saņemtais piekļuves talons (access token)

Content-Type un Accept - atkarīgs no katra servisa specifikas



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.



1. Pielikums

```yaml
openapi: 3.0.1
info:
  title: API-RAPLM_VRAA-EmptyExample
  version: v1_0
servers:
  - url: /
security:
  - default: []
paths:
  /getExample:
    get:
      summary: 'Vispārīga informācija '
      description: Metodes apraksts
      parameters: []
      responses:
        '200':
          description: ok
      security:
        - default:
            - API-RAPLM_VRAA-EpmtyExample-v1_0-ScopeName
      x-auth-type: Application & Application User
      x-throttling-tier: Unlimited
      x-wso2-application-security:
        security-types:
          - oauth2
        optional: false
components:
  securitySchemes:
    default:
      type: oauth2
      flows:
        implicit:
          authorizationUrl: 'https://test.com'
          scopes:
```

```
            API-RAPLM_VRAA-EpmtyExample-v1_0-ScopeName: Scope to provide an example for users
          x-scopes-bindings:
            API-RAPLM_VRAA-EpmtyExample-v1_0-ScopeName: ''
x-wso2-auth-header: Authorization
x-wso2-cors:
  corsConfigurationEnabled: false
  accessControlAllowOrigins:
    - '*'
  accessControlAllowCredentials: false
  accessControlAllowHeaders:
    - authorization
    - Access-Control-Allow-Origin
    - Content-Type
    - SOAPAction
    - apikey
    - Internal-Key
  accessControlAllowMethods:
    - GET
    - PUT
    - POST
    - DELETE
    - PATCH
    - OPTIONS
x-wso2-production-endpoints:
  urls:
    - 'http://swagger.io'
  type: http
x-wso2-basePath: /example/v1_0
x-wso2-transports:
  - http
  - https
x-wso2-response-cache:
  enabled: false
  cacheTimeoutInSeconds: 300

```