API управления подпиской

- Сквозная авторизация на партнерском портале ООО “СкайДНС”
- Документация модуля skydns_website.api.subscription (API Подписки)
- Документация модуля skydns_website.api.application (API Агентов)
- Использование API подписки SkyDNS

Сквозная авторизация на партнерском портале ООО “СкайДНС”

Документация написана для партнеров ООО "СкайДНС". Описывает создание ссылки на партнёрский портал для прозрачной авторизации пользователей.

Принцип действия

Принцип основан на безопасной передаче данных через небезопасную среду, проще говоря с помощью криптографической подписи.

Рассмотрим на примере. Допустим есть партнер "Рога и Копыта", который предоставляет доступ в интернет. Есть кабинет партнера, через который клиенты партнера управляют услугами. Есть кабинет партнерского портала, через который пользователь может управлять сервисом ООО "СкайДНС". Требуется организовать бесшовную авторизацию на партнерском портале из кабинета партнера для его клиентов.

Для этого с помощью ключей доступа формируется подписанный URL, ведущий на партнерский портал. После перехода по URL пользователь авторизуется без ввода пароля. Если неавторизованный пользователь попадает на партнерский портал, он перенаправляется на кабинет партнера для получения ссылки для авторизации.

Требования

1. Ключи доступа к API ООО "СкайДНС" и доменное имя портала. Ключи и доменное имя можно получить, связавшись с отделом продаж.
2. Передать в ООО "СкайДНС" URL, где неавторизованный пользователь может получить ссылку для авторизации на партнерском портале.
3. Идентификатор пользователя на партнерском портале ООО "СкайДНС", его можно получить при создании пользователя, с помощью Provider Api ООО "СкайДНС".
4. Библиотека для криптографической подписи. Есть готовые реализации библиотеки:

1. python - в фреймворке Django "django.core.signing";
2. python - библиотека "itsdangerous";
3. php - библиотека "itsdangerous-php";
4. javascript - библиотека "nobi".

Если используемый вами язык программирования не перечислен выше, самостоятельно найдите или напишите аналог вышеописанных библиотек.

Создание URL

Для создания URL вам потребуется private_key и доменное имя портала. Далее пример кода для формирования URL. Так же потребуется идентификатор пользователя, для которого мы генерируем ссылку.

Далее следует код на python:

```
from django.utils.crypto import get_random_string
from django.core import signing

key = 'private key'
domain = 'skydns.example'
username = 'user@partner'

data = {'ident': username, 'token': get_random_string()}
signer = signing.Signer(key, salt='skydns')
json = signing.JSONSerializer().dumps(data)
b64 = signing.b64_encode(json)
token = signer.sign(b64)

print 'https://%s/welcome?%s' % (domain, token)
```

Где key это приватный ключ, domain доменное имя портала, username это идентификатор пользователя. Полученный URL можно разместить в месте, доступном пользователю.

Данным URL можно воспользоваться только один раз, и в течение некоторого времени, после этого данный токен становится не валидным.

Использование URL

Полученный URL нужно отобразить пользователю, что бы он перешел по нему на портал.

Так же необходимо передать нам URL, по которому пользователь, не авторизованный на нашем портале, мог бы авторизоваться. Как правило это URL личного кабинета партнера.

Пример кода на php

При возможности заменить функцию generateRandomString на криптографически безопасную random_bytes из модуля CSPRNG.

```
<?php
require("itsdangerous.php");

function generateRandomString($length = 10) {
    $characters = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
```

```
    $charactersLength = strlen($characters);
    $randomString = '';
    for ($i = 0; $i < $length; $i++) {
        $randomString .= $characters[rand(0, $charactersLength - 1)];
    }
    return $randomString;
}

$key = "private key";
$domain = "skydns.example";
$username = "user@partner";
$complex = array(
    "ident" => $username,
    "token" => generateRandomString(10)
);
$ser = new ItsDangerous\Signer\Serializer($key);
$c = $ser->dumps($complex);



$url = sprintf("https://%s/welcome?%s", $domain, base64_encode($c));

print $url;
```

Документация модуля skydns_website.api.subscription (API Подписки)

Набор представлений для методов Subscription API v 1.9.

```
skydns_website.api.subscription.views.activate_user(provider, data)
```

Активировать пользователя провайдера.

Параметры:	• **provider** - параметр нормализуется из ApiKey декоратором authenticate_provider • **ident** - login пользователя
Результат:	

```
skydns_website.api.subscription.views.add_ip(provider,user,profile,data)
```

Метод добавляет один или несколько статических адресов без замены.

Если при этом адрес находится на другом профиле этого же пользователя, происходит его перепривязка к указанному профилю. Если адрес привязан к профилю другого пользователя, выдается ошибка для этого или нескольких адресов. Добавление других адресов в наборе происходит без ошибки.

Параметр comment который будет применен ко всем ip

Параметры:	• **provider** - параметр нормализуется из ApiKey декоратором authenticate_provider • **ident** - login пользователя • **ip** - один или несколько IP адресов

Необязательные параметры:

Параметры:	• **profile** - числовой id профиля • **comment** - комментарий к ip
Результат:	

```
skydns_website.api.subscription.views.add_vpn(provider, data)
```

Метод пытается создать сертификат vpn-соединения.

Возможные ошибки:

- при попытке создать для пользователя впн-соединение с именем, уже использующимся в его впн-соединениях
- при попытке создать впн-соединений больше определённого для пользователя планфичей vpn
- при попытке создать соединение на несуществующий или принадлежащий другому пользователю профиль (по profile_id)
- при попытке реселлера создать соединение не своему пользователю

Параметры:	• **provider** - параметр нормализуется из ApiKey декоратором authenticate_provider • **ident** - login пользователя • **name** - название vpn • **profile_id** - идентификатор профиля пользователя
Результат:	{‘ovpn’: vpn.ovpn()}

```
skydns_website.api.subscription.views.clear_ip(provider, data)
```

Метод очищает все адреса, привязанные к профилю.

Параметры:	• **provider** - параметр нормализуется из ApiKey декоратором authenticate_provider • **ident** - login пользователя • **profile** - числовой id профиля
Результат:	

```
skydns_website.api.subscription.views.clear_vpn_for_profile(provider, data)
```

Метод удаляет все vpn-соединения для профиля пользователя

Параметры:	• **provider** - параметр нормализуется из ApiKey декоратором authenticate_provider • **profile_id** - идентификатор профиля пользователя
Результат:	

```
skydns_website.api.subscription.views.clear_vpn_for_user(provider,data>)
```

Метод удаляет все vpn соединения пользователя

Параметры:	• **provider** - параметр нормализуется из ApiKey декоратором authenticate_provider • **ident** - логин пользователя, для которого нужно удалить vpn соединения
Результат:	

```
skydns_website.api.subscription.views.create_profile(provider,data)
```

Метод создает профиль пользователю.

Параметры:	• **provider** - провайдер • **ident** - логин пользователя • **name** - имя профиля • **tls** - использовать ли TLS • **blockpage_id** - id страницы блокировки

```
skydns_website.api.subscription.views.deactivate_user(provider,data)
```

Деактивировать пользователя провайдера.

Параметры:	• **provider** - параметр нормализуется из ApiKey декоратором authenticate_provider • **ident** - login пользователя
Результат:	

```
skydns_website.api.subscription.views.get_active_users(provider,data)
```

Метод возвращает список активных пользователей реселлера и их тарифы.

Результат:	список активных пользователей:

```
[
{‘username’: ‘username1’, ‘plan_name’: ‘plan name’}, {‘username’: ‘username2’, ‘plan_name’: ‘pla
]
```

```
skydns_website.api.subscription.views.get_activity(provider,data)
```

Информация об активности пользователя за определенную дату. В зависимости от настроек реселлера возвращается старая статистика или аналитика.

Параметры:	• **provider** - параметр нормализуется из ApiKey декоратором authenticate_provider • **ident** - login пользователя, для которого необходимо получить информацию.

Необязательные параметры:

Параметры:	• **date** - дата, за которую необходимо получить отчет о действиях пользователя в формате YYYY-MM-DD. • **profile_id** - идентификатор профиля пользователя, для которого запрашивается отчет
Результат:	`{“requests”: “2600”, “blocks”: “581”}`

```
skydns_website.api.subscription.views.get_activity_report(provider,data)
```

Отчет об активности пользователя за определенный период времени.

Параметры:	• **provider** - параметр нормализуется из ApiKey декоратором authenticate_provider • **ident** - login пользователя, для которого необходимо получить информацию. • **start** - дата начала периода для получения статистики в формате YYYY-MM-DD. • **end** - дата окончания периода для получения статистики в формате YYYY-MM-DD.

Необязательный параметр: :param profile_id: идентификатор профиля пользователя, для которого запрашивается отчет

Результат:	словарь вида:

```
{
   “labels”: [
```

```
      “2016-06-29 14:00:00”, “2016-06-29 15:00:00”,

   ], “datasets”: [
      {“label”: “Requests”, “data”: [375, 275]}, {“label”: “Blocks”, “data”: [13, 0]}
   ]
}
```

```
skydns_website.api.subscription.views.get_category_report(provider,data)
```

Отчет по категориям пользователя за определенный период времени.

Параметры:	• **provider** - параметр нормализуется из ApiKey декоратором authenticate_provider • **ident** - login пользователя, для которого необходимо получить информацию. • **start** - дата начала периода для получения статистики в формате YYYY-MM-DD. • **end** - дата окончания периода для получения статистики в формате YYYY-MM-DD.

Необязательный параметр: :param profile_id: идентификатор профиля пользователя, для которого запрашивается отчет

Результат:	словарь вида:

```
{
“Movies & Video”: “5”, “File Storage”: “2”, “Home & Family”: “3”
}
```

```
skydns_website.api.subscription.views.get_daily_stat(provider,data)
```

Возвращает реселлеру статистику за день для указанного пользователя

Параметры:	• **provider** - параметр нормализуется из ApiKey декоратором authenticate_provider • **date** - день, за который требуется статистика • **ident** - пользователь, чья статистика требуется

Необязательные параметры:

Параметры:	• **email_to** - адрес, на который необходимо отправить ссылку со статистикой • **profile_id** - id профиля пользователя для которого запрашивается статистика
Return file_url:	
	ссылка на файл, содержащий статистику

Отчет содержит следующие столбцы: “Timestamp”, “Domain name”, “Visits”, “Blocks”, “Categories”, “Profile”, “Username”, “User’s timestamp”.

```
skydns_website.api.subscription.views.get_list_activity_report(provider,data)
```

Отчет об активности пользователя за определенный период времени.

Параметры:	• **provider** - параметр нормализуется из ApiKey декоратором authenticate_provider • **ident** - login пользователя, для которого необходимо получить информацию (в запросе может быть несколько). • **start** - дата начала периода для получения статистики в формате YYYY-MM-DD. • **end** - дата окончания периода для получения статистики в формате YYYY-MM-DD.
Результат:	словарь вида:

```
{
   “2019-08-01”: {
      “username”: {
         “visits”: 262972530, “blocks”: 567080
      },....
   },....
}
```

```
skydns_website.api.subscription.views.get_popular_report(provider,data)
```

Отчет о популярных запросах пользователя за определенный период времени.

Параметры:	• **provider** - параметр нормализуется из ApiKey декоратором authenticate_provider • **ident** - login пользователя, для которого необходимо получить информацию. • **start** - дата начала периода для получения статистики в формате YYYY-MM-DD. • **end** - дата окончания периода для получения статистики в формате YYYY-MM-DD.

Необязательный параметр: :param profile_id: идентификатор профиля пользователя, для которого запрашивается отчет

Результат:	словарь вида:

```
{
   “labels”: [“example.com”, “google.com”, “asdfg.com”], “datasets”: [

      {“label”: “Requests”, “data”: [630, 474, 290]}, {“label”: “NXdomain”, “data”: [0, 0, 290]}
      {“label”: “Blocks”, “data”: [630, 0, 0]}

   ]

}
```

```
skydns_website.api.subscription.views.get_vpn_list(provider,data)
```

Метод для получения списка vpn соединений пользователя

Параметры:	• **provider** - параметр нормализуется из ApiKey декоратором authenticate_provider • **ident** - логин пользователя, для которого нужно вернуть список vpn соединений
Результат:	список vpn соединений пользователя

```
skydns_website.api.subscription.views.list_ip(provider,data)
```

Параметры:	• **provider** - параметр нормализуется из ApiKey декоратором authenticate_provider • **ident** - login пользователя

Необязательный параметр: :param profile: числовой id профиля

Результат:	все IP адреса привязанные к профилю

```
skydns_website.api.subscription.views.method_not_found(*args,**kwargs)
```

Метод-пустышка-заглушка.

Нужен, чтобы, согласно спецификации jsonrpc, отдавать удобочитаемую ошибку. Ответ полностью соответствует спецификации: https://www.jsonrpc.org/specification.

```
skydns_website.api.subscription.views.profiles(provider,data)
```

Метод возвращает список активных профилей пользователя.

Параметры:	• **provider** - параметр нормализуется из ApiKey декоратором authenticate_provider • **ident** - логин пользователя, для которого нужно получить писок профилей
Return dict:	список профилей в формате как пример ниже `{ 70749: u'Основной', 93773: u'ДхтСкий', }`

```
skydns_website.api.subscription.views.prolongate(provider,data)
```

Включение и изменение платного тарифа для пользователя. Этот метод всегда должен вызываться после метода subscribe для включения услуги пользователю.

Параметры:	• **provider** - параметр нормализуется из ApiKey декоратором authenticate_provider • **ident** - login пользователя

Необязательный параметр: :param plan: код тарифного плана

Результат:	

```
skydns_website.api.subscription.views.remove_ip(provider,data)
```

Удаление IP адреса пользователя.

Параметры:	• **provider** - параметр нормализуется из ApiKey декоратором authenticate_provider • **ident** - login пользователя • **ip** - IP адрес пользователя
Результат:	

```
skydns_website.api.subscription.views.remove_vpn(provider, data)
```

Метод для удаления vpn соединения пользователя

Параметры:	• **provider** – параметр нормализуется из ApiKey декоратором authenticate_provider • **id** – id vpn соединения
Результат:	

```
send_monthly_stat(provider, data)
```

Отправка статистики за месяц пользователю по электронной почте.

Параметры:	• **provider** – параметр нормализуется из ApiKey декоратором authenticate_provider • **ident** – login пользователя, • **year** – год в формате YYYY, • **month** – месяц в формате MM, за который необходимо сформировать отчет,

Необязательный параметр: :param profile_id: идентификатор профиля пользователя, для которого запрашивается статистика.

Отчет содержит следующие столбцы: “Timestamp”, “Domain name”, “Visits”, “Blocks”, “Profile”, “Categories”.

```
skydns_website.api.subscription.views.subscribe(provider, data)
```

Регистрация пользователя в системе с переданными реквизитами и установка тарифа по умолчанию.

Параметры:	• **provider** – параметр нормализуется из ApiKey декоратором authenticate_provider • **password** – пароль пользователя

Необязательные параметры:

Параметры:	• **login** – login пользователя • **email** – адрес электронной почты пользователя • **customer** –
Результат:	

```
skydns_website.api.subscription.views.subscribe_plans(provider, data)
```

Метод возвращает список доступных тарифов, назначаемых при создании/изменении карточки пользователя.

Параметры:	**provider** – параметр нормализуется из ApiKey декоратором authenticate_provider

```
skydns_website.api.subscription.views.subscription_info(provider, data)
```

Информация о подписке. На текущий момент возвращает наступила ли дата окончания подписки.

Параметры:	• **provider** – параметр нормализуется из ApiKey декоратором authenticate_provider • **ident** – login пользователя
Результат:	{‘date_end’: true|false}

```
skydns_website.api.subscription.views.unsubscribe(provider, data)
```

Отключение пользователя. Пользователь переключается на бесплатный тариф.

Параметры:	• **provider** – параметр нормализуется из ApiKey декоратором authenticate_provider • **ident** – login пользователя
Результат:	

```
skydns_website.api.subscription.views.update_email(provider, data)
```

Изменение email пользователя для восстановления пароля и информационных сообщений.

Параметры:	• **provider** – параметр нормализуется из ApiKey декоратором authenticate_provider • **ident** – login пользователя • **email** – новый email пользователя
Результат:	

```
skydns_website.api.subscription.views.update_ip(provider, data)
```

Создать или обновить динамический IP адрес пользователя и привязать его к необходимому профилю фильтрации.

Параметры:	• **provider** – параметр нормализуется из ApiKey декоратором authenticate_provider • **ident** – login пользователя • **ip** – новый динамический IP адрес пользователя

Необязательные параметры:

Параметры:	• **hostname** – для привязки ip к hostname • **profile** – числовой id профиля к которому необходимо привязать IP адрес
Результат:	

```
skydns_website.api.subscription.views.update_nat(provider, data)
```

Метод для добавления или обновления привязки профиля пользователя реселлера к одному из NAT DNS

Параметры:	• **profile_id** – id профиля пользователя. • **address** – IP адрес нашего DNS(anycast-address).
Результат:	None

```
skydns_website.api.subscription.views.update_password(provider, data)
```

Обновление пароля пользователя.

Параметры:	• **provider** – параметр нормализуется из ApiKey декоратором authenticate_provider • **ident** – login пользователя • **password** – новый пароль пользователя
Результат:	

```
skydns_website.api.subscription.views.update_profile(provider, data)
```

Метод для изменения параметров профиля пользователя реселлера

Параметры:	• **provider** – параметр нормализуется из ApiKey декоратором authenticate_provider • **profile_id** – id профиля пользователя

Необязательные параметры:

Параметры:	• **name** – новое имя профиля пользователя • **tls** – булево значение для установки профилю, будет ли использоваться TLS.
Результат:	{‘id’: profile_id, ‘name’: profile_name, ‘tls’: true|false}

Метод для получения списка всех пользователей реселлера

Параметры:	• **key**

Пример запроса: `curl "https://www.skydns.ru/provider_api/users?key=<api_key>"`

Результат:

```
{
    "jsonrpc": "1.9",
    "status": "ok",
    "data": [{"is_active": true, "username": "kok_000001", "date_end": "", "plan": "Бизнес"},
{"is_active": false, "username": "test2", "date_end": "", "plan": "Школа"}, {"is_active":
true, "username": "test3", "date_end": "2025-05-01T00:00:00", "plan": "Домашний"}],
    "result": [{"is_active": true, "username": "kok_000001", "date_end": "", "plan":
"Бизнес"}, {"is_active": false, "username": "test2", "date_end": "", "plan": "Школа"},
{"is_active": true, "username": "test3", "date_end": "2025-05-01T00:00:00", "plan":
"Домашний"}]
}
```

Метод для получения установленных категорий фильтрации на профиле

Параметры:	• **key** • **profile_id**

Возможные ошибки:

- не передан profile_id - `profile_id param required`
- пользователь с profile_id не принадлежит реселлеру - `Wrong provider`

Пример запроса: `curl "https://www.skydns.ru/provider_api/categories?key=<api_key>&profile_id=<profile_id>"`

Результат:

```
{
    "status": "ok",
    "data": {"block_unknown_sites": true, "categories": [64, 68], "safe_search_enabled": false, "safe_youtube_enabled": true, "white_list_only": false},
    "result": {"block_unknown_sites": true, "categories": [64, 68], "safe_search_enabled": false, "safe_youtube_enabled": true, "white_list_only": false},
    "jsonrpc": "1.9"
}
```

Метод для установки категорий фильтрации на профиль

Параметры:	• **key** • **profile_id**

Необязательные параметры:

Параметры:	• **cat (multiple param)** • **safe_youtube_enabled** • **block_unknown_sites** • **safe_search_enabled** • **white_list_only** – для привязки ip к hostname

Возможные ошибки:

- не передан profile_id - `profile_id param required`
- пользователь с profile_id не принадлежит реселлеру - `Wrong provider`
- передача white_list_only с выключенной фичей white_list_mode - `white_list_only param is not available for this user`
- невалидные значения cat (не преобразуются к инту) - `Invalid cats`
- значения cat не могут быть установлены этому пользователя по его тарифному плану либо таких категорий не существует - `Invalid cats`
- переданы невалидные значения safe_youtube_enabled/block_unknown_sites/safe_search_enabled/white_list_only - `safe_youtube_enabled/block_unknown_sites/safe_search_enabled/white_list_only param must be True or False`

Пример запроса: `curl "https://www.skydns.ru/provider_api/set_categories?key=<api_key>&profile_id=<profile_id>&cat=<cat_id1>&cat=<cat_id2>&block_unknown_sites=<true|false>&safe_search_enabled=<true|false>&safe_youtube_en`

`abled=<true| false>&white_list_only=<true| false>"`

Результат:


```
{
    "status": "ok",
    "jsonrpc": "1.9",
    "data": {"safe_search_enabled": false, "white_list_only": false, "block_unknown_sites": true, "safe_youtube_enabled": true, "categories": [64, 68]},
    "result": {"safe_search_enabled": false, "white_list_only": false, "block_unknown_sites": true, "safe_youtube_enabled": true, "categories": [64, 68]}
}
```

Документация модуля skydns_website.api.application (API Агентов)

Доступные методы API

Модуль хранения представлений JSON-RPC API. Используется агентами.

Curl запросы для тестирования выглядит так:

```
curl -X POST http://localhost:8000/api/json/v2 -L –user user:password > -d ‘{“id”:0, “jsonrpc”:”
```

В простейшем случае связку кредитов можно использовать такую:

```
plantest-premium@skydns.ru:plantest
```

```
skydns_website.api.application.views.add_domain(request, profile, choice, domain, ip=None, comme
```

JSONRPC Method: addDomain

Добавить домен.

Параметры:	• **profile** – профиль пользователя • **choice** – тип домена • **domain** – домен • **ip** – IP адрес (если передать IP, то будет добавлен не домен, а алиас) • **comment** – комментарий к домену
Результат:	id добавленного домена.
Тип результата:	int

```
skydns_website.api.application.views.add_profile(request, name)
```

JSONRPC Method: addProfile

Метод создает профиль и возвращает словарь с данными о профиле.

Параметры:	**name** – имя создаваемого профиля

Результат:	данные созданного профиля в формате:
	{
	
	‘id’: id профиля,
	‘default’: является ли профиль профилем по умолчанию,
	‘token’: токен,
	‘name’: название профиля,
	‘white_list_only’: включена ли фильтрация только по белому списку,
	‘is_schedule_enabled’: включено ли расписание,
	‘safe_search_enabled’: статус безопасного поиска,
	‘safe_youtube_enabled’: статус фильтрации YouTube (для API v2),
	‘block_unknown_enabled’: статус фильтрации неизвестных сайтов (для API v2)
	
	}

Тип результата:	dict

```
skydns_website.api.application.views.categories(request, language='en')
```

JSONRPC Method: categories

Получить список сгруппированных категорий.

Параметры:	**language** – язык выдачи

Результат:	список сгруппированных категорий в формате: `[ { “title”: “group1”, “items”: [ { “title”: “item1”, “id”: 1 } ] } ]`
Тип результата:	list

`skydns_website.api.application.views.change_schedule(request, profile, segments)`

JSONRPC Method: setSchedule

Установить расписание фильтрации.

Параметры:	• **profile** – профиль пользователя • **segments** – список сегментов вида: [[<offset1>, <true|false>], [<offset2>, <true|false>]]
Результат:	[True]
Тип результата:	list

`skydns_website.api.application.views.clear_domains(request, profile, choice )`

JSONRPC Method: clearDomains

Удалить все домены определенного типа у профиля.

Параметры:	• **profile** – профиль пользователя • **choice** – тип домена
Результат:	None

`skydns_website.api.application.views.` `domains` (*request*, *profile*, *choice*)

JSONRPC Method: domains

Получить список доменов определенного типа для профиля.

Параметры:	• **profile** – профиль пользователя • **choice** – тип домена (‘black’, ‘white’ или ‘alias’)

Результат:	для запроса чёрных или белых доменов - список словарей в формате:
	`{ “domain”: домен, “id”: id домена ]`
	для запроса алиасов - список словарей в формате:
	`[ { “domain”: домен, “id”: id домена, “ip”: ip-адрес ]`

Тип результата:	list

```
skydns_website.api.application.views.enable_schedule(request, profile, flag)
```

JSONRPC Method: setScheduleEnabled

Включить/выключить расписание для профиля.

Параметры:	• **profile** – профиль пользователя • **flag** – флаг включить/выключить
Результат:	flag
Тип результата:	bool

```
skydns_website.api.application.views.feedback(request, title, message)
```

JSONRPC Method: feedback

Отправить сообщение обратной связи пользователя.

Параметры:	• **title** – заголовок сообщения для отправки • **message** – сообщение для отправки
Результат:	None

```
skydns_website.api.application.views.get_advertising(request)
```

JSONRPC Method: getAdvertising

Получить информацию об акции.

Результат:	[<strAdvUrl>, <strImageUrl>]
Тип результата:	list

```
skydns_website.api.application.views.get_agent_profile(request, uid, hostname, version, os_info, address)
```

JSONRPC Method: getProfile

Возвращает необходимые настройки (токен, UID) для работы агентского приложения. Если передан UID - метод вернёт информацию об уже существующем профиле с этим UID. Если UID пуст, но переданы все остальные параметры - будет создан новый профиль.

Параметры:	• **uid** – uid • **hostname** – имя хоста • **version** – версия • **os_info** – информация • **address** – IP адрес
Результат:	список с данными профиля агента: [uid, token, profile_id]
Тип результата:	list

```
skydns_website.api.application.views.get_categories_daily_stat(request, profile, lang='en', categories_list=None)
```

JSONRPC Method: getCategoriesDailyStats

Получить статистику по категориям за сутки для нужного профиля.

Параметры:	• **profile** – профиль пользователя • **lang** – язык выдачи • **categories_list** – список идентификаторов категорий. Если его нет, то будет получена статистика блокировок по всем категориям для профиля пользователя

Результат:	словарь формата: {
	`<category_name>: <value> ...`
	`}`

Тип результата:	dict

```
skydns_website.api.application.views.get_filtering_status(request)
```

JSONRPC Method: getFilteringStatus

Получить состояние фильтрации пользователя. :return: <состояние фильтрации> :rtype: bool

```
skydns_website.api.application.views.get_limited_agent_profile(request, uid, hostname, version, os_info, address)
```

JSONRPC Method: getLimitedProfile

Это модификация метода get_agent_profile (getProfile). Этот метод проверяет, сколько уже выдано токенов пользователю (мобильных или десктопных Linux), и если это число превышает максимум (ПФ max_mobile_agents и max_desktop_agents соответственно), отдаёт фейловые токены (‘’, 9999999999, <profile_id>) Если запрос не мобильного или десктопного Linux агента, токены генерируются без условий. Если передан UID - метод вернёт информацию об уже существующем профиле с этим UID. Если UID пуст, но переданы все остальные параметры - будет создан новый профиль.

Параметры:	• **uid** – uid • **hostname** – имя хоста • **version** – версия • **os_info** – информация • **address** – IP адрес
Результат:	список с данными профиля агента: [uid, token, profile_id]
Тип результата:	list

```
skydns_website.api.application.views.get_linux_supporting(request, tag=None, version=None)
```

JSONRPC Method: getLinuxSupporting

Получить версии Linux-агента: минимальную поддерживаемую (подлежащую принудительному обновлению) и текущую.

Параметры:	• **tag** – тег для получения версий (по умолчанию - самая новая без разбора версии ОС) • **version** – текущая версия установленного агента

Результат:	словарь вида:
	{
	‘minimalSupported’: <минимально поддерживаемая версия (целое неотрицательное число)>, ‘current’: <текущая версия мобильного агента (целое неотрицательное число)>, ‘url’: <url до пакета>, ‘checksum’: <контрольная сумма пакета>, ‘announces’: анонсы версий, от version до последней, если version не указан, то все
	[
	{
	‘version’: <номер версии> ‘announce’: <анонс для этой версии>
	]
	}
	}

Тип результата:	dict

```
skydns_website.api.application.views.get_plans_list(request, mobile=False)
```

JSONRPC Method: getPlans

Получить список тарифных планов.

Параметры:	**mobile** – True - выдавать только мобильные планы, False - все

Результат:	список тарифов в формате: [{ ‘code’: код тарифа, ‘name’: название тарифа, ‘price’: стоимость тарифа, ‘trialPeriod’: сколько дней триального периода на тарифе, ‘period’: на сколько месяцев покупается тариф, ‘isMobile’: является ли тариф мобильным, ‘description’: URL страницы сайта с описанием тарифа]
Тип результата:	list

```
skydns_website.api.application.views.get_popular_report(request, start, end, profile_id=None)
```

JSONRPC Method: getPopularReport

Отчет о популярных запросах пользователя за определенный период времени.

Параметры:	• **start** – дата начала периода для получения статистики в формате YYYY-MM-DD • **end** – дата окончания периода для получения статистики в формате YYYY-MM-DD • **profile_id** – id профиля. Если не задан, будет создан отчёт по всем профилям пользователя

Результат:	словарь вида:
	{
	“labels”: [“example.com”, “google.com”, “asdfg.com”], “datasets”: [
	{“label”: “Requests”, “data”: [630, 474, 290]}, {“label”: “NXdomain”, “data”: [0, 0, 290]}, {“label”: “Blocks”, “data”: [630, 0, 0]}
	]
	}

```
skydns_website.api.application.views.get_ssl_certificate(request, version=1)
```

JSONRPC Method: getSSLCertificate

Получить ссылку на корневой SSL-сертификат СкайДНС.

Результат:	<прямая ссылка до сертификата>
Тип результата:	str

```
skydns_website.api.application.views.get_ssl_certificate_data_v2(request, version=1)
```

Получить ссылки на скачивание SSL-сертификатов СкайДНС.

```
 :return: {

“certs”: [
“cert/url”, “cert/url”
], “expires”: “YYYY-MM-DDTHH:MM:SS”

} :rtype: dict curl test request: curl –location –request POST
‘https://skynds.ru/api/json/v2‘ –header ‘Content-Type: application/json’ –data-raw ‘{
```

```
“id”: 0, “jsonrpc”: “2.0”, “method”: “getSSLCertificatesDataV2”, “params”: [“3”]
}’

>{
“id”:0,”jsonrpc”:”2.0”, “result”:{

“expires”:”2028-01-19T00:00:00”, “certs”:[

“https://www.skynds.ru/userfiles/certs/skydns_iss.crt”,
“https://www.skynds.ru/userfiles/certs/skydns_root.crt“
]

}

}
```

```
skydns_website.api.application.views.get_user_plan_info(request)
```

JSONRPC Method: getPlan

Получить информацию о тарифе пользователя.

Результат:	информация о тарифе в формате: {
	‘name’: название плана, ‘expired’: количество дней до истечения оплаченного периода, ‘isMobile’: поддерживает ли тариф мобильные устройства (true | false)
	}

Тип результата:	dict

```
skydns_website.api.application.views.get_version(request, tag=None)
```

JSONRPC Method: getAPCVersion

Получить версии мобильного приложения.

Параметры:	**tag** – тег для получения версий, по умолчанию ‘default’

Результат:	словарь вида: {
	‘minimalSupported’: <минимально поддерживаемая версия (целое неотрицательное число)>, ‘current’: <текущая версия мобильного агента (целое неотрицательное число)>
	}

Тип результата:	dict

```
skydns_website.api.application.views.intentional_crash(request)
```

JSONRPC Method: intentionalCrash

Специальный метод, всегда выдающий JsonRpcRuntimeError.

Результат:	None

```
skydns_website.api.application.views.multi_schedule(request, profile)
```

JSONRPC Method: multiSchedule

Получить расписание фильтрации.

Параметры:	**profile** – профиль пользователя
Результат:	расписание фильтрации в формате: `[ [<offset1>, <True|False>], [<offset2>, <True|False>], ... ]`
Тип результата:	list

```
skydns_website.api.application.views.my_ip(request)
```

JSONRPC Method: myip

Получение текущего IP адреса клиента.

Результат:	IP адрес клиента

Тип результата:	str

```
skydns_website.api.application.views.profile_schedule_activity(request, profile)
```

JSONRPC Method: profileScheduleActivity

Получить количество минут до изменения состояния фильтрации профилей.

Параметры:	**profile** – профиль пользователя
Результат:	[<флаг: работает ли фильтрация по профилю>, <количество минут до смены состояния>] количество минут = -1, если расписание выключено или нет других состояний.

Пример:

```
first segment state last segment current offset
change |  /

v v v v

|//////////|  False |///////////////X/|  0———260———1090————10k |
True |///////////|  True |
```

Тип результата:	list

```
skydns_website.api.application.views.profiles(request)
```

JSONRPC Method: profiles

Получение списка профилей.

Результат:	список профилей в формате: `[{ 'id': id профиля, 'default': является ли профиль профилем по умолчанию, 'token': токен, 'name': название профиля, 'white_list_only': включена ли фильтрация только по белому списку, 'is_schedule_enabled': включено ли расписание, 'safe_search_enabled': статус безопасного поиска, 'safe_youtube_enabled': статус фильтрации YouTube (для API v2), 'block_unknown_enabled': статус фильтрации неизвестных сайтов (для API v2) }]`
Тип результата:	list

```
skydns_website.api.application.views.register_user(request, username, password)
```

JSONRPC Method: register

Регистрация нового пользователя в облачном сервисе контент-фильтрации SkyDNS.

Параметры:	• **username** – имя пользователя • **password** – пароль создаваемого пользователя
Результат:	[True], если регистрация прошла успешно, [False, “error”] в противном случае
Тип результата:	list

```
skydns_website.api.application.views.remove_agent_profile(request, uid)
```

JSONRPC Method: removeAgentProfile

Метод удаляет AgentInfo отфильтрованный по заданному uid атрибуту.

Параметры:	**uid** – uid
Результат:	количество удалённых записей
Тип результата:	int

```
skydns_website.api.application.views.remove_domain(request, profile, choice, id)
```

JSONRPC Method: removeDomain

Удалить домен у заданного профиля.

TODO: параметр choice можно убрать и выбирать домен только по id.

Параметры:	• **profile** – профиль пользователя • **choice** – тип домена • **id** – id домена
Результат:	None

```
skydns_website.api.application.views.remove_profile(request, profile)
```

JSONRPC Method: removeProfile

Метод удаляет указанный профиль.

Параметры:	**profile** – числовой id удаляемого профиля
Результат:	None

```
skydns_website.api.application.views.rename_profile(request, profile_id, name)
```

JSONRPC Method: renameProfile

Метод устанавливает новое имя для профиля. Возвращает актуальное имя.

Параметры:	• **profile_id** – числовой id профиля, имя которого необходимо изменить • **name** – новое имя профиля
Результат:	новое имя профиля

Тип результата:	string

```
skydns_website.api.application.views.send_debug_info(request, uid, info)
```

Отправить логи для отладки.

Параметры:	**info** – информация лога формата [{“command_title”: <log_name>, “command_info”” <log_message>}, ...]
Результат:	количество созданных объектов

curl запрос для тестирования выглядит так:

```
$ curl -X POST http://localhost:8000/api/json/v2 -L –user plantest-premium@skydns.ru:plantest1 -d ‘{“id”:0, > “jsonrpc”:”2.0”, “method”:”sendDebugInfo”, > “params”: [“uid”, [{“command_title”: <log_name>, “command_info”” <log_message>}]}’
```

```
skydns_website.api.application.views.send_log(request, profile_id=None, log_content='', os_info=None)
```

JSONRPC Method: sendLog

Фиксация в лог-файл лога пользователя с агентского решения.

Параметры:	• **profile** – профиль пользователя • **log_content** – лог пользователя • **os_info** – версия системы
Результат:	True
Тип результата:	boolean

```
skydns_website.api.application.views.send_package(request, package, file_name, system_tag, version, checksum, announce)
```

JSONRPC Method: sendPackage

Загрузка новых версий агентов на сервер.

Параметры:	• **package** – содержимое пакета в Base64 • **file_name** – имя файла • **system_tag** – тег версии ОС • **version** – версия агента • **checksum** – контрольная сумма • **announce** – анонс версии
Результат:	True
Тип результата:	boolean

```
skydns_website.api.application.views.set_agent_profile(request, uid, profile_id)
```

JSONRPC Method: setProfile

Установить профиль AgentInfo.

Параметры:	• **uid** – uid AgentInfo • **profile_id** – id профиля, который требуется установить
Результат:	None

```
skydns_website.api.application.views.set_block_unknown_enabled(request, profile, flag)
```

JSONRPC Method: setBlockUnknownEnabled

Включить/выключить блокировку неизвестных сайтов.

Параметры:	• **profile** – профиль пользователя • **flag** – флаг включить/выключить
Результат:	None

```
skydns_website.api.application.views.set_categories(request, profile_id, cat_ids_list)
```

JSONRPC Method: setFilterCats

Метод привязывает список категорий фильтрации к профилю пользователя и возвращает актуальный список категорий.

Параметры:	• **profile_id** – числовой id профиля • **cat_ids_list** – список id категорий
Результат:	актуальный список id категорий
Тип результата:	list

```
skydns_website.api.application.views.set_category(request, profile, cat, state)
```

JSONRPC Method: setFilterCat

Метод позволяет изменить состояние фильтра для категории.

Параметры:	• **profile** – профиль пользователя • **cat** – числовой id категории • **state** – булево значение состояния категории (true/false)
Результат:	None

```
skydns_website.api.application.views.set_filtering_status(request, enabled)
```

JSONRPC Method: setFilteringStatus

Установить состояние фильтрации пользователя. :param enabled: состояние фильтрации для установки

Результат:	<состояние фильтрации>
Тип результата:	bool

```
skydns_website.api.application.views.set_networks(request, profile, hostname)
```

JSONRPC Method: updateNic

Установить динамический IP адрес. IP адрес определяется автоматически из запроса.

Параметры:	• **profile** – профиль пользователя • **hostname** – имя хоста
Результат:	установленный IP адрес
Тип результата:	str

```
skydns_website.api.application.views.set_safe_search(request, profile, flag)
```

JSONRPC Method: setSafeSearchEnabled

Включить/выключить работу по безопасному поиску.

Параметры:	• **profile** – профиль пользователя • **flag** – флаг включить/выключить
Результат:	None

```
skydns_website.api.application.views.set_safe_youtube(request, profile, flag)
```

JSONRPC Method: setSafeYoutubeEnabled

Включить/выключить безопасный Youtube.

Параметры:	• **profile** – профиль пользователя • **flag** – флаг включить/выключить
Результат:	None

```
skydns_website.api.application.views.set_whitelist(request, profile, flag)
```

JSONRPC Method: setWhiteListOnly

Включить/выключить работу по белому списку.

Параметры:	• **profile** – профиль пользователя • **flag** – флаг включить/выключить
Результат:	None

```
skydns_website.api.application.views.system_info(request)
```

JSONRPC Method: systemInfo

Получение информации о настройках системы.

Результат:	системная информация в формате:
	{
	‘public_dns’: DNS, ‘nxdomain’: NX domain, ‘blockpage’: blockpage, ‘blockpage_token’: blockpage token, ‘blockapi’: block api
	}

Тип результата:	dict

```
skydns_website.api.application.views.test_auth(request, username, password)
```

JSONRPC Method: testAuth

Аутентификация пользователя.

Результат:	результат аутентификации (true или false)
Тип результата:	bool

```
skydns_website.api.application.views.user_filter(request, profile)
```

JSONRPC Method: userFilter

Получить список категорий профиля пользователя.

Результат:	список id категорий профиля
Тип результата:	list

```
skydns_website.api.application.views.user_info(request)
```

JSONRPC Method: userInfo

Получить информацию о пользователе.

Результат:	информация о пользователе в формате:
	{
	‘username’: юзернейм, ‘plan’: {
	‘name’: название тарифного плана,
	‘code’: код тарифного плана, ‘date_end’: дата конца оплаченного периода,
	‘features’: {
	название_опции: значение, ...
	
	},
	
	}, ‘tz’: смещение от UTC в часах,
	‘tz_minutes’: смещение от UTC в минутах
	
	}

Тип результата:	dict

Использование API подписки SkyDNS

Провайдеру выдается публичный ключ для аутентификации и подписания запросов.

Все запросы выполняются через протокол HTTPS. С помощью GET запросов. Каждый запрос должен содержать публичный ключ провайдера с именем параметра **key**.

Например:

```
curl "https://www.skydns.ru/provider_api/{имя метода}?key=abcd&a=1&b=1"
```

Публичный ключ передается отдельным письмом и не должен публиковаться в открытом доступе!

Доступны следующие методы API:

- subscribe - зарегистрировать пользователя в системе с переданными реквизитами и установить тариф провайдера по умолчанию.
- subscribe_plans - получить список доступных тарифов для создания/изменения пользователя реселлером.
- activate - активировать пользователя провайдера.
- deactivate - деактивировать пользователя провайдера.
- update_email - обновить email пользователя для восстановления пароля и получения информационных сообщений.
- update_password - обновить пароль пользователя.
- prolongate - включить или изменить платный тариф для пользователя. Этот метод всегда должен вызываться после метода subscribe для включения услуги пользователю.
- unsubscribe - отключить пользователя и переключить на бесплатный тариф.
- subscription_info - получить информацию о дате окончания подписки.
- profiles – получить список активных профилей пользователя.
- create_profile - создать профиль.
- update_profile – изменить параметры профиля пользователя реселлера.
- add_ip - добавить один или несколько статических IP адресов пользователя.
- clear_ip - удалить все статические IP адреса на профиле пользователя.
- list_ip - получить список всех статических IP адресов пользователя.
- update_ip - создать или обновить динамический IP адрес пользователя и привязать его к необходимому профилю фильтрации.
- remove_ip - удалить IP адрес пользователя.

- add_vpn - создать VPN-сертификат для профиля и получить профиль настройки OpenVPN.
- clear_vpn_for_profile - удалить все vpn-соединения на профиле пользователя.
- clear_vpn_for_user - удалить все vpn-соединения на всех профилях пользователя.
- get_vpn_list - получить список всех vpn-соединений пользователя.
- remove_vpn - удалить vpn-соединение пользователя.
- update_nat - добавить или обновить привязку профиля к DNS адресам для пользователей за NAT.
- get_activity - получить информацию об активности пользователя за определенную дату.
- get_activity_report - получить отчет об активности пользователя за определенный период времени.
- get_list_activity_report - отчет об активности пользователя за определенный период времени.
- get_popular_report - получить отчет о популярных запросах пользователя за определенный период времени.
- get_category_report - получить отчет по категориям пользователя за определенный период времени.
- send_monthly_stat - отправить статистику за месяц пользователю по электронной почте.
- get_daily_stat - получить статистику пользователя за день.
- get_active_users - получить список активных пользователей и их тарифы.

API доступно по адресу: https://www.skydns.ru/provider_api/

Вызывается по схеме: https://www.skydns.ru/provider_api/{имя метода}?{GET параметры запроса}

Доступные методы

subscribe

Зарегистрировать пользователя в системе с переданными реквизитами и установить тариф по умолчанию.

Передается 1 обязательный параметр:

- **password** - пароль создаваемого пользователя.

Может передаваться 4 необязательных параметра:

- **login** - логин, по умолчанию стандарное наименование с уникальным префиксом провайдера.
- **plan** - тарифный план. Сисок доступных для вас планов уточняйте у менеджера СкайДНС или с помощью метода subscribe_plans. Если вы используете недопустимый код, в ответ будет выдана ошибка.
- **email** - адрес электронной почты пользователя (может использоваться для самостоятельного восстановления пароля пользователем через личный кабинет и получения служебных уведомлений).
- **customer** - наименование договора.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok",
        "data": {
            "username": "username",
            "email": "email",
            "plan": "tariff_name"
        }
    }
}
```

subscribe_plans

Получить список доступных тарифов для создания/изменения пользователя реселлером.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok",
        "data": {
            "plan1": "tariff_name1",
            "plan2": "tariff_name2"
        }
    }
}
```

activate

Активировать пользователя провайдера.

Передается 1 обязательный параметр:

- **ident** - login пользователя, которого необходимо активировать.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok"
    }
}
```

deactivate

Деактивировать пользователя провайдера.

Передается 1 обязательный параметр:

- **ident** - login пользователя, которого необходимо деактивировать.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok"
    }
}
```

update_email

Обновить email пользователя для восстановления пароля и получения информационных сообщений.

Передается 2 обязательных параметра:

- **ident** - login пользователя, у которого изменяется email.
- **email** - новый email пользователя. Если в базе уже есть пользователь с таким email, будет возвращена ошибка.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok"
    }
}
```

update_password

Обновить пароль пользователя.

Передается 2 обязательных параметра:

- **ident** - login пользователя.
- **password** - новый пароль пользователя.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok"
    }
}
```

prolongate

Включить или изменить платный тариф для пользователя. Этот метод всегда должен вызываться после метода subscribe для включения услуги пользователю.

Передается 1 обязательный параметр:

- **ident** - login пользователя, у которого включается платная подписка.

Может передаваться 1 необязательный параметр:

- **plan** с доступными значениями: PREMIUM, SCHOOL, BUSINESS. Список доступных для вас планов уточняйте у менеджера СкайДНС или с помощью метода subscribe_plans. Если вы используете недопустимый код, в ответ будет выдана ошибка.

Если параметр **plan** не передан, по умолчанию включается план PREMIUM.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok"
    }
}
```

unsubscribe

Отключить пользователя и переключить на бесплатный тариф.

Передается 1 параметр:

- **ident** - login пользователя, аккаунт которого переключается в бесплатный режим.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok"
    }
}
```

subscription_info

Получить информацию о дате окончания подписки.

Передается 1 параметр:

- **ident** - login пользователя, у которого проверяется срок подписки.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok",
        "data": {
            "date_end": 1390930655
        }
    }
}
```

Дата окончания в unix time.

В настоящее время не используется, зарезервировано для будущего использования!

profiles

Получить список активных профилей пользователя.

Передается 1 обязательный параметр:

- **ident** – login пользователя, у которого необходимо получить список профилей.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok",
        "data": {
            "ид профиля 1": "имя профиля",
            "ид профиля 2": "имя профиля",
            "ид профиля 3": "имя профиля"
```

```
        }
    }
}
```

“имя профиля” передается в кодировке utf-8.

create_profile

Создать профиль.

Передается 2 обязательных параметра:

- **ident** - login пользователя.
- **name** – имя профиля.

Дополнительно можно передать 2 необязательных параметра:

- **tls** – будет ли использоваться TLS. Принимает значения: True, False.
- **blockpage_id** – id страницы блокировки, которую необходимо установить на профиль.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok"
    }
}
```

update_profile

Изменить параметры профиля пользователя реселлера.

Передается 1 обязательный параметр:

- **profile_id** – id профиля пользователя, который необходимо изменить.

Дополнительно можно передать 2 необязательных параметра:

- **name** – новое имя профиля пользователя.
- **tls** – булево значение для установки профилю, будет ли использоваться TLS. Принимает значения: True, False.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok",
        "data": {
            "id": "profile_id",
            "name": "profile_name",
            "tls": "True| False"
        }
    }
}
```

add_ip

Добавить один или несколько статических IP адресов пользователя.

Передается 2 обязательных параметра:

- **ident** - login пользователя.
- **ip** - один или несколько IP адресов. При передаче нескольких адресов, каждое значение нужно указывать отдельным параметром.

Дополнительно можно передать 2 необязательных параметра:

- **profile** - числовой id профиля, к которому необходимо привязать IP адреса.
- **comment** - комментарий к IP адресу.

Если параметр **profile** не передан, IP адреса будут привязаны к профилю “По умолчанию”.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok",
        "data": {
            "added_addresses": ["1.1.1.1", "1.1.1.2"]
        }
    }
}
```

Если какие-то из переданных адресов не удалось добавить, дополнительно будет выведен их список:

```
{
    "response": {
        "status": "ok",
        "data": {
            "added_addresses": ["1.1.1.1", "1.1.1.2"],
            "invalid_adresses": [
```

```
                {"1.invalid.ip.adress": "IP address is invalid"},
                {"0.0.0.0": "This address is not public"}
            ]
        }
    }
}
```

clear_ip

Удалить все статические IP адреса на профиле пользователя.

Передается 2 обязательных параметра:

- **ident** - login пользователя.
- **profile** - числовой id профиля.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok"
    }
}
```

list_ip

Получить список всех статических IP адресов пользователя.

Передается 1 обязательный параметр:

- **ident** - login пользователя, у которого необходимо получить IP адреса.

Дополнительно можно передать 1 необязательный параметр:

- **profile** - числовой id профиля, для которого необходимо получить IP адреса.

Если параметр **profile** не передан, IP адреса будут получены для всех профилей.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok",
        "data": {
            "ip": [
                {"address": "1.2.3.4", "comment": "example.com"},
                {"address": "1.2.3.5", "comment": "example.com"}
```

```
            ],
        }
    }
}
```

update_ip

Создать или обновить динамический IP адрес пользователя и привязать его к необходимому профилю фильтрации.

Передается 2 обязательных параметра:

- **ident** - login пользователя.
- **ip** - новый динамический IP адрес пользователя.

Предоставленный IP адрес будет установлен на профиле Основной указанного пользователя. Если предоставленный IP уже привязан к другому пользователю провайдера, привязка будет убрана и IP адрес будет привязан к указанному в методе пользователю.

Дополнительно можно передать 2 необязательных параметра:

- **hostname** - позволяет привязать несколько IP адресов к одному профилю, а также профилям, отличным от профиля Основной. При последующих обновлениях адреса привязка к профилю будет сохраняться.
- **profile** - числовой id профиля к которому необходимо привязать IP адрес пользователя. Если этот параметр будет опущен, по умолчанию привязка произойдет на профиль Основной. Числовые коды профилей доступны в личном кабинете в списке профилей.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok"
    }
}
```

remove_ip

Удалить IP адрес пользователя.

Передается 2 обязательных параметра:

- **ident** - login пользователя.
- **ip** - IP адрес пользователя, который необходимо удалить.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok"
    }
}
```

add_vpn

Создать VPN-сертификат для профиля и получить профиль настройки OpenVPN.

Передается 3 обязательных параметра:

- **ident** - login пользователя, которому запрашивается профиль настройки OpenVPN.
- **name** - уникальное для пользователя название создаваемого VPN-подключения.
- **profile_id** - идентификатор профиля пользователя, для которого запрашивается профиль настройки OpenVPN. Если **profile_id** не соответствует ни одному из профилей пользователя, будет получена соответствующая ошибка.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok",
        "data": {
            "ovpn": "client\nremote vpn.skydns.ru 1194 udp\nnobind\ndev tun\npersist-tun\npersis
        }
    }
}
```

Если лимит VPN-соединений исчерпан, попытка добавить новое завершится ошибкой.

clear_vpn_for_profile

Удалить все vpn-соединения на профиле пользователя.

Передается 1 обязаетельный параметр:

- **profile_id** - числовой id профиля, у которого необходимо удалить все vpn-соединения.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok"
    }
}
```

clear_vpn_for_user

Удалить все vpn-соединения на всех профилях пользователя.

Передается 1 обязательный параметр:

- **ident** - login пользователя, у которого необходимо удалить vpn-соединения.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok"
    }
}
```

get_vpn_list

Получить список всех vpn-соединений пользователя.

Передается 1 обязательный параметр:

- **ident** - login пользователя, у которого необходимо получить список vpn соеденений.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok",
        "data": [
            {"id": "vpn_id1", "profile": "profile_name", "ip": "vpn_hostname1", "name": "vpn_nam
            {"id": "vpn_id2", "profile": "profile_name", "ip": "vpn_hostname2", "name": "vpn_nam
        ]
    }
}
```

remove_vpn

Удалить vpn-соединение пользователя.

Передается 1 обязательный параметр:

- **id** - id vpn соединения, которое необходимо удалить.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok"
    }
}
```

update_nat

Добавить или обновить привязку профиля к DNS адресам для пользователей за NAT.

Передается 2 обязательных параметра:

- **profile_id** - id профиля пользователя.
- **address** - IP адрес NAT DNS.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok"
    }
}
```

get_activity

Получить информацию об активности пользователя за определенную дату.

Передается 1 обязательный параметр:

- **ident** - login пользователя, для которого необходимо получить информацию.

Дополнительно можно передать 2 необязательных параметра:

- **date** - дата, за которую необходимо получить отчет о действиях пользователя в формате YYYY-MM-DD. Если параметр не задан, отчет будет сформирован за сегодня.

- **profile_id** - идентификатор профиля пользователя, для которого запрашивается отчет.

Значение параметра **date** должно быть не ранее, чем 1 января предыдущего года.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok",
        "data": {
            "requests": "2600",
            "blocks": "581"
        }
    }
}
```

Где **requests** - общее количество запросов за указанную дату, **blocks** - количество блокировок.

get_activity_report

Получить отчет об активности пользователя за определенный период времени.

Передается 3 обязательных параметра:

- **ident** - login пользователя, для которого необходимо получить информацию.
- **start** - дата начала периода для получения статистики в формате YYYY-MM-DD.
- **end** - дата окончания периода для получения статистики в формате YYYY-MM-DD.

Дополнительно можно передать 1 необязательный параметр:

- **profile_id** - идентификатор профиля пользователя, для которого запрашивается отчет.

Если диапазон **start** - **end** охватывает более 30 суток, в отчет попадают только последние 30 из них.

Начало периода **start** должно быть не ранее, чем 1 января предыдущего года.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
```

```
        "status": "ok",
        "data": {
            "labels": [
                "2016-06-29 14:00:00",
                "2016-06-29 15:00:00"
            ],
            "datasets": [
                {
                    "label": "Requests",
                    "data": [375, 275]
                },
                {
                    "label": "Blocks",
                    "data": [13, 0]
                }
            ]
        }
    }
}
```


Где **labels** - список значений для оси времени. **datasets** - список словарей содержащих наборы данных по определенном параметрам отчета, *label* - название параметра отчета, *data* - список значений, соответстующих датам из списка **labels**.

В **datasets** используются наборы данных: **Requests** - количество запросов, **Blocks** - количество блокировок.

get_list_activity_report

Получить отчет об активности пользователя за определенный период времени.

Передается 3 обязательных параметра:

- **ident** - login пользователя, для которого необходимо получить информацию (в запросе может быть несколько).
- **start** - дата начала периода для получения статистики в формате YYYY-MM-DD.
- **end** - дата окончания периода для получения статистики в формате YYYY-MM-DD.

Пример запроса:

https://skydns.ru/provider_api/get_list_activity_report/?key=supersecretkey&start=2023-04-01&end=2023-04-17&ident=username1&ident=username2

Ответ возвращается в формате JSON следующего вида:

```
{
    "data": {
        "2023-04-03": {
            "username1": {
                "visits": 28251,
                "blocks": 1106
            },
            "username2": {
                "visits": 4854,
                "blocks": 760
            }
        },
    }
}
```

get_popular_report

Получить отчет о популярных запросах пользователя за определенный период времени.

Передается 3 обязательных параметра:

- **ident** - login пользователя, для которого необходимо получить информацию.
- **start** - дата начала периода для получения статистики в формате YYYY-MM-DD.
- **end** - дата окончания периода для получения статистики в формате YYYY-MM-DD.

Дополнительно можно передать 1 необязательный параметр:

- **profile_id** - идентификатор профиля пользователя, для которого запрашивается отчет.

Если диапазон **start** - **end** охватывает более 30 суток, в отчет попадают только последние 30 из них.

Начало периода **start** должно быть не ранее, чем 1 января предыдущего года.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok",
        "data": {
            "labels": ["example.com", "google.com", "asdfg.com"],
            "datasets": [
                {
                    "label": "Requests",
                    "data": [630, 474, 290]
```

```
                },
                {
                    "label": "NXdomain",
                    "data": [0, 0, 290]
                },
                {
                    "label":"Blocks",
                    "data":[630, 0, 0]
                }
            ]
        }
    }
}
```

Где **labels** - список значений для оси времени. **datasets** - список словарей, содержащих наборы данных по определенном параметрам отчета, *label* - название параметра отчета, *data* - список значений, соответстующих датам из списка **labels**.

В **datasets** используются наборы данных: **Requests** - количество запросов, **Blocks** - количество блокировок, **NXdomain** - количество неразрезолвенных доменов.

get_category_report

Получить отчет по категориям пользователя за определенный период времени.

Передается 3 обязательных параметра:

- **ident** - login пользователя, для которого необходимо получить информацию.
- **start** - дата начала периода для получения статистики в формате YYYY-MM-DD.
- **end** - дата окончания периода для получения статистики в формате YYYY-MM-DD.

Дополнительно можно передать 1 необязательный параметр:

- **profile_id** - идентификатор профиля пользователя, для которого запрашивается отчет.

Если диапазон **start** - **end** охватывает более 30 суток, в отчет попадают только последние 30 из них.

Начало периода **start** должно быть не ранее, чем 1 января предыдущего года.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
```

```
        "status": "ok",
        "data": {
            "Фильмы и видео онлайн": "5",
            "Файловые архивы": "2",
            "Дом, семья, хобби": "3"
        }
    }
}
```

Где **data** - представляет собой словарь, *ключ* - название категории, *значение* - количество посещений.

send_monthly_stat

Отправить пользователю по электронной почте статистику за месяц.

Передается 3 обязательных параметра:

- **ident** - login пользователя.
- **year** - год в формате YYYY.
- **month** - месяц в формате MM, за который необходимо сформировать отчет.

Дополнительно можно передать 1 необязательный параметр:

- **profile_id** - идентификатор профиля пользователя, для которого запрашивается статистика.

На email пользователя будет выслан отчет в формате CSV со статистикой за указанный месяц. Отчет содержит следующие столбцы: “Timestamp”, “Domain name”, “Visits”, “Blocks”, “Profile”, “Categories”.

Значение параметра **year** должно быть не меньше прошлого года.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok"
    }
}
```

get_daily_stat

Получить статистику пользователя за день.

Передается 2 обязательных параметра:

- **date** - дата в формате ‘YYYY-MM-DD’ определяет день, за который необходима статистика.
- **ident** - имя пользователя, для которого необходима статистика.

Дополнительно можно передать 2 необязательных параметра:

- **email_to** - email, на который будет отправлена статистика.
- **profile_id** - идентификатор профиля пользователя, для которого запрашивается статистика.

Значение параметра **date** должно быть не ранее, чем 1 января предыдущего года.

Ответ возвращается в формате JSON следующего вида:

```
{
    "response": {
        "status": "ok",
        "data": {
            "result": "ссылка на файл"
        }
    }
}
```

get_active_users

Получить список активных пользователей и их тарифы.

Ответ возвращается в формате JSON следующего вида:

```
{
    "data": [
        {
            "username": "username1",
            "plan_name": "Бизнес"
        },
        {
            "username": "username1",
            "plan_name": "Школа"
        },
```

```
    ]
}
```