Úvod do API Štíteknabalík.cz

Verze: **1.73**; 2025-11-05

API je založeno na RESTové architektuře. K jeho korektnímu používání budete pořebovat splnit následující podmínky.

1. Přistupovat pouze přes zabezpečený protokol **HTTPS**.
2. Získat oprávnění přístupu od společnosti zaregistrované na stránce https://id.app.stiteknabalik.cz/oauth-registration.
3. Autentizovat se pomocí **HTTP Bearer autentizace**.
4. Zasílat data pouze s kódováním **UTF-8**.

Prostředí API

	Produkční prostředí PROD
Rest API [1]	https://rest.stiteknabalik.cz
Registrace OAuth klienta [2]	https://id.app.stiteknabalik.cz/oauth-registration
Registrace účtu	https://id.app.stiteknabalik.cz
Klientská aplikace	https://id.app.stiteknabalik.cz/login

1] Rest API s verzemi /v2, /v3, /v4.

2] OAuth klient pro připojení API vyžaduje vytvoření firemního účtu.

Autentizace

Štíteknabalík.cz podporuje dva způsoby autentizace. **Basic** s využitím vygenerovaného tokenu a pokročilejší **Bearer** s protokolem OAuth 2.0., který navíc podporuje tzv scope (práva jednotlivým metodám)

Všechny požadavky kromě hlavní adresy rest.stiteknabalik.cz vyžadují HTTP autentizaci. K přístupu pomocí **Basic autentizace** potřebujete získat od podpory `token` a pro přístup přes **OAuth** potřebujete získat takzvaný `access_token`.

Vzhledem k povaze této autentizace je **NUTNÉ přistupovat vždy přes protokol HTTPS**. Jediný přístup k API přes HTTP s korektní hlavičkou Authorization může teoreticky vést k odposlechnutí vašich přístupových údajů. Více v sekci Autentizace

HTTP kódy

Prvotním ověřením na Vaší straně by vždy měla být kontrola vrácného HTTP kódu. Všechny tyto kódy odpovídají specifikaci HTTP. (Více o HTTP zde https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html).

Možné HTTP kódy odpovědí API:

HTTP kód	Význam	Podrobnosti
200	**OK**	Požadavek byl úspěšně zpracován
201	**Created**	Požadavek byl úspěšně zpracován a nový zdroj byl vytvořen (typická odpověď po úspěšném POST požadavku)
304	**Not Modified**	Odpověď se shoduje s cacheovaým obsahem, tělo odpovědi je prázdné, viz cacheování
400	**Bad Request**	Byl přijat neplatný požadavek, například nevalidní formát vstupních dat
401	**Unauthorized**	Autentizace selhala, chybí hlavička Authorization nebo obsahuje neplatné údaje
404	**Not Found**	Požadovaný zdroj (URL) neexistuje
405	**Method Not Allowed**	Byla použita neplatná HTTP metoda pro daný zdroj, v odpovědi lze nalézt platné metody
406	**Not Acceptable**	Byla použita nepodporovaná hlavička Accept
410	**Gone**	Přistupujete na starou verzi API
412	**Precondition Failed**	Pokus o aktualizaci změněného zdroje
414	**Request-URI Too Large**	Byl odeslán požadavek s příliš dlouhou URI
415	**Unsupported Media Type**	Byla přijata nepodporovaná hlavička Content-type
422	**Unprocessable Entity**	Data požadavku neprošla validací, například neplatná váha zásilky
426	**Upgrade Required**	Byl použit nezabezpečený protokol HTTP
500	**Internal Server Error**	Došlo k chybě v aplikaci na naší straně
503	**Service Unavailable**	Služba je dočasně nedostupná (probíhá údržba naší aplikace)

Podporované formáty komunikace

Abychom Vám co nejvíce ulehčili napojení na naše API, podporujeme několik formátů dat v komunikaci. Použití je snadné:

- Uvěďte MIME type `Content-Type` v hlavičce vašich dotazů, abyste specifikovali v jakém formátu k nám zasíláte data.
- Uvěďte MIME type `Accept` v hlavičce vašich dotazů, abyste získali odpověď ve Vámi požadovaném formátu. Podporované formáty jsou níže v tabulce.
- Uvěďte `Accept-language` v hlavičce vašich dotazů, abyste získali případné chybové hlášky ve Vámi požadovaném jazyce. Podporujeme angličtinu `en` (jako výchozí) a češtinu `cs`.

Podporované formáty (pro `Content-Type:` a `Accept:`):

JSON	Doporučeno	Uveďte MIME type `application/json` v dané hlavičce.
XML	Podporu ukončujeme	Uveďte MIME type `application/xml` v dané hlavičce (podporu ukončujeme)
form-urlencoded	Podporu ukončujeme	Uveďte MIME type `application/x-www-form-urlencoded` v dané hlavičce (podporu ukončujeme)

Můžu hlavičku Accept vynechat?

Pokud hlavička Accept není uvedená, nebo používá generické `*/*`, je jako výchozí formát použit JSON. V případě neplatné či nepodporované hlavičky Accept je chybová zpráva vrácena také ve formátu JSON.

Jaký je kořenový tag XML odpověďi?

Pokud se rozhodnete využívat formát XML pro odpovědi z našeho serveru, kořenový tag je vždy `<root>`.

Jak poznám, že posílám špatný formát?

Pokud přijmeme nevalidní vstupní formát, je vrácen HTTP kód 400 s popisem chyby.

Jaký je v API formát datumů?

Pokud není uvedeno jinak, všechny datumové formáty jsou ve formátu ISO 8601, např. `2014-12-31T18:25:50+02:00`.
Více o ISO 8601 zde https://en.wikipedia.org/wiki/ISO_8601.

Jaký je v API formát desetinných čísel?

Lze používat desetinnou čárku i desetinnou tečku. Oddělovač tisíců nesmí být použit, např. `12500.00`, `330,50`.

> **Vyjímky**
>
> Pokud dojde k chybě na naší straně na nižší než aplikační úrovni, může být odpověd v jiném formátu, než bylo požadováno v hlavičce Accept. Tyto chyby mohou nastat při serverových chybách či údržbě aplikace, proto doporučujeme před parsováním odpovědi zkontrolovat vrácený HTTP kód. Příkladem může být i chyba 414. Pokud je vrácen kód 5xx, je možné, že odpověď není v očekávaném formátu.

Struktura odpovědí

Všechny odpovědi dodržují stejnou strukturu odpovědi. Konkrétní odpověď záleží na požadovaném formátu dat, nicméně struktura má vždy následující schéma.

Ukázka struktury odpovědí

Úspěšná odpověď:

```
root
--- code
--- status
--- message
--- data
```

Chybová odpověď:

```
root
--- code
--- status
--- message
--- errors
--- --- [0]
--- --- --- message
--- --- --- field
--- --- --- value
--- --- [1]
--- --- --- message
--- --- --- field
--- --- --- value
```

Příklady formát chybových odpovědí

JSON

```
{
"code": 422,
"status": "error",
"message": "Validation failed",
"errors":
    [
     {
        "message": "This value should be of type
float.",
        "field": "[0].packages[0].weight",
        "value": "3 kg"
     },
     {
        "message": "Invalid currency format,
expected ISO 4217",
        "field": "[0].valueCurrency",
        "value": "€"
     }
    ]
}
```

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
    <code>422</code>
    <status>error</status>
    <message>Validation failed</message>
    <errors>
        <message>This value should be of type float.
</message>
        <field>[0].packages[0].weight</field>
        <value>3 kg</value>
    </errors>
    <errors>
        <message>Invalid currency format, expected ISO
4217</message>
        <field>[0].valueCurrency</field>
        <value>€</value>
    </errors>
</root>
```

Popis položek:

- `code` je obecné označení chyby totožné s HTTP kódem
- `status` značí stav požadavku, možné hodnoty jsou: success, error
- `message` message obecný popis chyby (anglicky)
- `data` data odpovědi, pouze pro úspěšné odpovědi, ne vždy přítomné, struktura se liší pro různé požadavky
- `errors` pole chyb, pouze pro chybové odpovědi, ne vždy přítomné
- `errors.message` je anglický popis konkrétní chyby (o české chyby lze zažádat hlavičkou accept-language: cs)
- `errors.field` je cesta k položce, kde k chybě došlo (viz příklady výše)
- `errors.value` je chybná hodnota (nemusí se přesně shodovat s odeslanou hodnotou)

> **Syntax**
>
> Všechny klíče používají **camelCase** syntax.

> **Další informace**
>
> Bližší popis možných chybových odpovědí, kódů a strukturu úspěšné odpovědi najdete přímo na stránce naší druhé služby Balíkonoše u konkrétních metod a zdrojů.
> Tyto metody a zdroje jsou shodné s Štíteknabalík.cz až na jednu důležitou vyjímku týkající se jiné url endpointu (https://rest.stiteknabalik.cz/api/v3/).

Verzování API

Použití jednotlivých verzí API je určeno v adrese zdroje. Aktuálně je API ve třetí verzi, tedy URL všech zdrojů začínají na `https://rest.stiteknabalik.cz/api/v3/`. O implementaci nových verzí Vás budeme informovat, přečemž původni verze bude po nějakou dobu stále funkčí. Minoritní změny v API, které neovlivňují dosavadní funkčnost se v adrese neprojeví.

Jednotlivé verze API:

- **v1** - iniciální verze API, již nepodporována
- **v2** - dobíhající verze, kompatibilata s API https://balikonos.cz/api/v2
- **v3** - aktuální verze, kompatibilata s API https://balikonos.cz/api/v3

- **v4** - aktuální verze, nejnovější verze, která je pravidelně rozšiřována.

Testování a debugování

Je velmi pravděpodobné, že při vývoji budete chtít ověřit jak se API chová a jaké odpovědi vrací na různé požadavky.

Vytvoření účtu

Registraci můžete vytvořit na odkazu pod tímto odstavcem.

V aplikaci Štíteknabalík.cz v sekci nastavení je zapotřebí přidat Vaše svozová místa. Pro testování konkrétního přepravce si vyberte z nabídky v nastavení a vložte libovolné testovací autentizační údaje.

Pro testování vložení a uzavření zásilek je zapotřebí **povolit svozové místo** přímo v nastavení konkrétního přepravce.
Bez tohoto povolení budou automaticky všechny zásilky označeny jako přímá zásilka z adresy na adresu, což je nežádoucí v případech kdy chcete testovat standardní odesílání zásilek ze svozového místa.

Odkazy pro vytvoření účtu naleznete v sekci Prostředí.

CURL

Asi nejjednodušší možnost jak testovat naše API je pomocí nástroje cURL z příkazové řádky. Ukázka možného použití:

```
curl -H "Accept: application/xml" -H "Authorization: Bearer bacffecfc1bd349b85e51b37a542aed57f457a8e"
https://rest.stiteknabalik.cz
```

Jiné nástroje

Existuje celá řada dalších nástrojů k testování RESTových API, například rozšíření Chrome prohlížeče Postman

Vlastnosti a funkce

Komprimace přenesených dat

Pokud v požadavku uvedete hlavičku `Accept-encoding: gzip`, výsledné tělo odpovědi Vám přijde v komprimované podobě **gzip**. Tato metoda ušetří u souborů XML kolem **80 % přenesených dat!** Pro JSON jde cca o 65 % dat.

Komprimovaná data se vrátí spolu s hlavičkou `Content-encoding: gzip`. Na vás je pouze dekódování přijatého obsahu, což je ve většině jazyků velmi snadné. Viz například PHP Stejně tak můžete i vy zasílat tělo zprávy komprimované pomocí gzip metody. Je však nutné uvést hlavičku `Content-encoding: gzip`.

Taktéž je možné nakonfigurovat Váš server tak, aby tuto komprimaci a dekomprimaci prováděl zcela automaticky, viz například apache

> **Doporučení**
> Jelikož množství přenášených dat může být skutečně velké, **výrazně doporučujeme používat komprimaci dat vždy a všude**.

HTTP Cache

Všechny `GET` metody vracejí odpovědi s hlavičkou `ETag`, která obsahuje hash aktuálně vrácené odpovědi. Tuto hodnotu je vhodné ukládat spolu s vrácenými daty a v případě potřeby stejného dotazu v budoucnu doplnit hash do hlavičky `If-None-Match`. Pokud se totiž odpověď od Vašeho posledního dotazu nezměnil, HTTP odpověď bude mít kód 304 a tělo bude prázdé – ušetříte tedy datový přenos obsahu, který již máte uložený z předchozího požadavku. Toto je obzvláště vhodné při dotazech vracejících větší množství dat (například PDF štítky). Chování si můžete prohlédnout i v prohlížeči na https://rest.stiteknabalik.cz

---

Další funkce

Přetěžování metody POST

Pokud vaše aplikace z nějakého důvodu neumožňuje zasílat PUT či DELETE HTTP požadavky, je možné přetížit metodu POST. Pokud nastavíte hlavičku `X-HTTP-Method-Override` na některou ze zmíněných hodnot (PUT či DELETE) při POST požadavku, bude vykonána stejná akce, jako kdyby se provedl PUT či DELETE požadavek.

Takto přetížit lze pouze metoda POST, nikoli GET. Metoda GET je totiž tzv. safe a nesmí měnit stav zdroje.

Formátování výstupu

Standardní JSON a XML odpovědi obsahují odřádkování i odsazení vnitřních elementů pro lepší čitelnost.Toto chování lze vypnout nastavením query parametru `prettyPrint` na `false`. Viz například ukázka kořenového zdroje https://rest.stiteknabalik.cz?prettyPrint=false. Tímto lze také mírně snížit objem přenášených dat

Nepodporované funkce

JSONP

Podpora pro JSONP není z technického hlediska možná z důvodu HTTP Bearer autentizace.

CORS

Podpora Cross-origin resource sharing není implementována a její nasazení není v plánu.

Autentizace

Štíteknabalík.cz API podporuje jednoduchý způsob autentizace **Basic** a pokročilý **Bearer** OAuth 2.0.
Podmínkou pro bezpečné přihlašování k API na adrese rest.stiteknabalik.cz je přístup přes protokol **HTTPS** jinak spojení bude odmítnuto.

Porovnání způsobů autentizace

	Basic	**Bearer (OAuth 2.0)**
Podpora scopes pro jednotlivé requesty	NE	ANO
Expirace tokenu	NE	ANO, 60 minut
Náročnost integrace	Malá	Vysoká
Způsob aktivace	Vygenerováním basic tokenu v nastavení aplikace Štíteknabalík.cz (sekce API) [1]	Vytvořením OAuth klienta přes formulář zde [2]

1] Basic token je po vytvoření okamžitě aktivní. Je ihned spárován s vaší registrací ve Štíteknabalík.cz a není tak potřeba dalších kroků k aktivaci. Basic token lze kdykolik přegenerovat.

2] OAuth klient není po vytvoření spárován s Vaší registrací ve Štíteknabalík.cz. Spárování se provede až v dalším kroku, viz "žádost o auth code" níže.

Basic autentizace

Pro získání Basic API tokenu prosíme kontaktujte podporu Štíteknabalík.cz.

Bude Vám vystaven token, který je nezbytné zaslat v hlavičce každého requestu přes HTTPS protokol. Token má neomezenou platnost a doporučujeme jednou za čas požádat o jeho opětovné vygenerování.

Basic token uvádějte v nezměněné podobě v hlavičce požadavku. Dejte si pozor na dodržení jedné mezery mehzi slovem Basic a API tokenem jinak nebude hlavička validní.

Ukázka hlavičky:

```
Authentication: Basic 684v98fd84vfd9845df55616d5f7d10d54ce9798ccd391735387ea2f6d9c64ce7d5
Content-Type: application/json
```

> **Info**
> Možnost manuálního vygenerování Basic API tokenu je momentálně ve vývoji.

OAuth 2.0

OAuth 2.0 protokol k autorizaci API klientů byl na našem webu vytvořen dle oficiální specifikace RFC 6749.

V této dokumentaci uvádíme konkrétní kroky k integraci a používání OAUTH 2.0, tak aby práce s protokolem byla co nejsrozumitelnější.
Detaily konkrétních částí protokolu zde neuvádíme, ale lze je nalézt přímo ve zmíněné specifikaci nebo v tomto článku.

Postup aktivace OAuth klienta ve zkratce

1. **Registrace OAuth klienta** - Na našem webu zaregistrujete nového OAuth 2.0 klienta pro Vaše API
2. **Povolení a autorizace (žádost o auth code)** - Přidáte povolení přístupu Vaší API k Vašemu účtu ve Štíteknabalík.cz a uložíte si `authorization code`
3. **Žádost o access token** - S auth kódem provedete requestem žádost o `access_token`
4. **Přístup ke zdrojům** - Začnete provádět požadované akce a metody dle specifikace v API Štíteknabalík.cz
5. **Přístup ke zdrojům - refresh access tokenu** Získání nového `access_tokenu`. Provádíte pouze pokud již máte `refresh_token`.

Krok 1. - Registrace OAuth klienta

Proveďte registraci nového OAuth 2.0 klienta uvedením základních údajů o Vaší aplikaci, která bude využívat API pro přístup k Vaší společnosti ve Štíteknabalík.cz:
PROD https://id.app.stiteknabalik.cz/oauth-registration

Nejdůležitějším údajem v tomto formuláři je **adresa pro přesměrování** (ve specifikaci jde o `redirect_uri`). Ta definuje **endpoint**, na který budou přesměrovávány všechny požadavky z naší aplikace. **Tato adresa by měla mít protokol https**, aby nebylo možné odcyhtit citlivé údaje!

Heslo je váš autentizační údaj, ve specifikaci vedený jako `client_secret`. Toto heslo se využívá při žádostech o `access_token`, podobně jako heslo v HTTP Basic autentizaci. Jako **přihlašovací jméno** v této OAuth autentizaci je použit vygenerovaný **identifikátor**, který získáte při dokončení registrace OAuth klienta. Ve specifikaci je tento údaj vedený jako `client_id`.

Kopii registrace obdržíte na uvedený email. Bude obsahovat zmíněný **identifikátor** a **heslo**, které jste si nastavili.

> **Pozor**
> Tato implementace nepodporuje přihlášení pomocí údajů v query parametrech či v těle požadavku. Ukázka požadavku s touto autentizací je uvedena níže. Současně nejsou podporováni **public klienti** obvykle implementovaní v JavaScriptu.

Krok 2. - Povolení a autorizace (žádost o auth code)

Již máte vytvořeného OAuth klienta a nyní mu musíte přidat povolení přístupu k Vašemu účtu ve Štíteknabalík.cz. Po schválení zároveň uložíte `authorization_code` pro využití v dalším kroku.

Toto se provádí na tzv autorizačním endpointu. Ten má adresu:
`https://app.stiteknabalik.cz/oauth/authorize/` pro **PROD**.

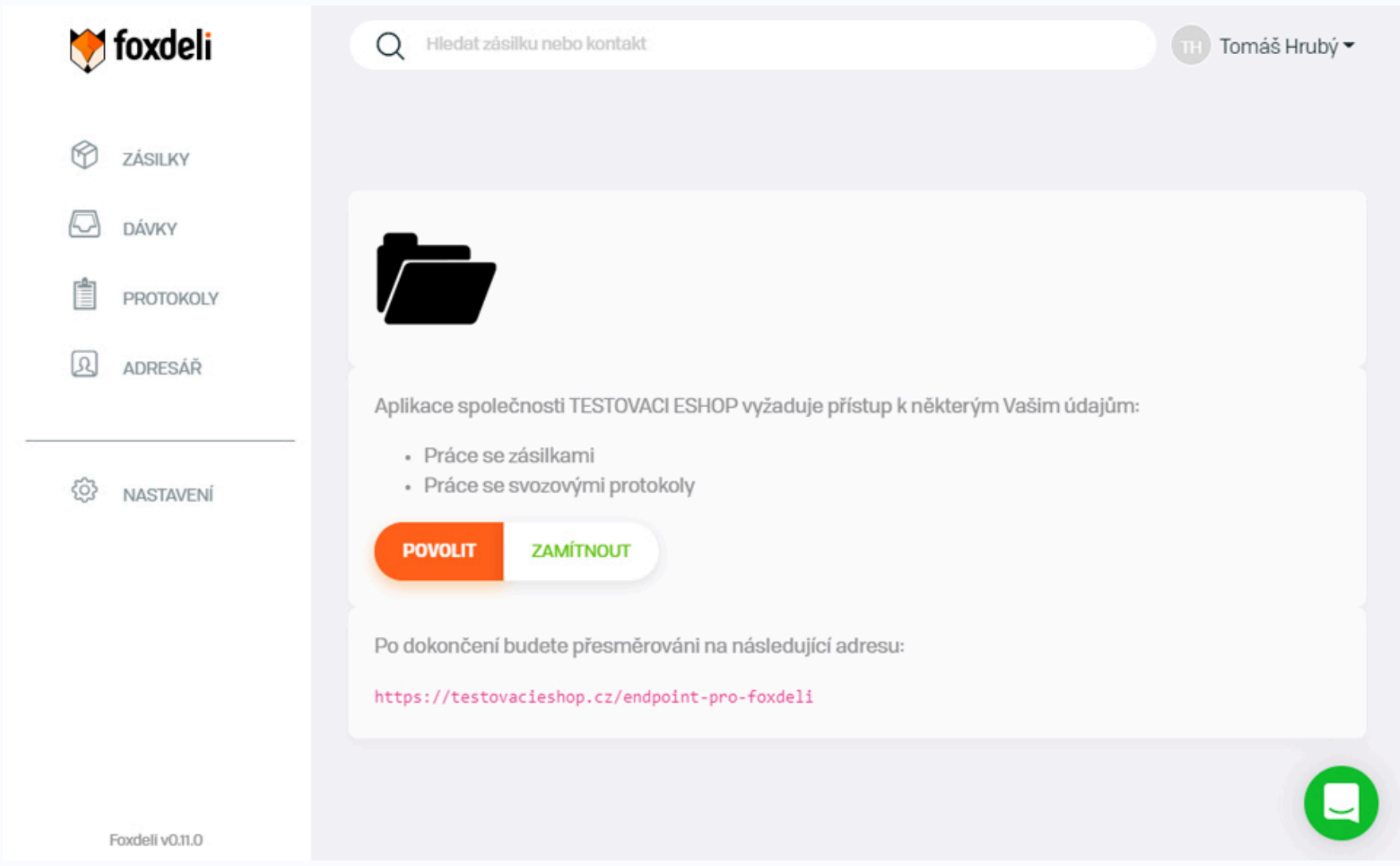
K této url adrese se do **query stringu** přidávají dle specifikace ještě tyto povinné parametry: `response_type`, `client_id` a také `scope` a `state`.
Scope určuje jaké metody půjde v OAuth klientem volat. Scope jednotlivých zdrojů jsou definovány v dokumentaci metod http://localhost:8880/cs/methods#methods-endpoints (viz OAuth Scope).
V případě, že chcete klientovi přidělit více scopes, spojte je znaménkem plus.
Dále je nutné, aby Váš klient ověřoval shodu **state** z query parametru v odpovědi z našeho serveru s Vámi odeslanou náhodnou hodnotou tohoto parametru. Tímto ověřením se předchází **CSRF útokům**.

V případě, že uvedete i parametr `redirect_uri`, je nutné, aby se přesně shodoval s adresou uvedenou při registraci Vašeho OAuth klienta.

Při úspěšném požadavku je uživateli zobrazen formulář s možností povolit či zamítnout přístup Vaší aplikace k zásilkám a dalším údajům. Spolu s tím je zobrazen název Vaší společnosti, logo, webová adresa, žádaný scope a adresa, na kterou bude posléze požadavek přesměrován.
Po přesměrování bude možné z url adresy zachytit **authorization_code**. Ten bude předaný metodou GET v query parametru `code`.

WEBVIEW

Žádost o **authorization_code** vyžaduje, abyste byli přihlášení ve Vašem účtu v aplikaci Štíteknabalík.cz. Proto je nezbytné, aby systém, ve kterém žádost provádíte uměl zobrazit HTML stránku a uměl si zapamatovat sessions. Protože se jedná o **GET** metodu, lze požadavek provést i v běžném prohlížeči. Následný redirect už může směřovat na Váš API endpoint, kde bude odchycen **authorization_code**, se kterým už bude automatizovaně pracovat Vás systém.
Přidělujete oprávnění a Vaše přihlášení + manuální potvrzení z Vašeho účtu ve Štíteknabalík.cz je jednou z částí zabezpečení.



Stránka s povolením

V případě povolení přístupu od Vás jako uživatele dojde k přesměrování na `redirect_uri` kde z query stringu v url adrese získáte `authorization_code` s platností 90 vteřin. Tento kód má vždy délku 40 znaků a obsahuje pouze malá písmena nebo čísla.

Upozornění

Žádost o `authorization_code` stačí provést jenom jednou nebo při změně `scope` daného klienta. Cílem je získat povolení a `authorization_code`, se kterým si v následujícím kroku vyžádáte čerstvý `access_token` a `refresh_token`.
Jakmile Vám po ukončení spojení `access_token` expiruje můžete jej znovu obnovit automatizovaně pomocí `refresh_tokenu`. **Autorizace přes "webview" se tedy podruhé již neprovádí**.

> Pokud máte více účtu v aplikaci Štíteknabalík.cz, prosíme, dejte si pozor, abyste při přidělování povolení OAuth klientovi byli přihlášení pod správným Štíteknabalík.cz účtem.

Ukázka požadavku autorizace

Detailní popis požadavku najdete ve specifikaci.

Endpoint otevřít

```
GET: https://app.stiteknabalik.cz/oauth/authorize/?
response_type=code&client_id=zjhygknkfk&scope=deliveries+collection-
protocols&redirect_uri=https://mujeshopik.cz/stiteknabalik-endpoint/&state=csjkhd5b1
```

Ukázka úspěšné odpovědi autorizace (redirect)

Detailní popis požadavku najdete ve specifikaci.

Endpoint otevřít

```
GET: https://mujeshopik.cz/stiteknabalik-endpoint/?code=87f90ba9fe97e3b43f11c37eea1e1b475dbd59b9&state=csjkhd5b1
```

Ukázka neúspěšné odpovědi autorizace

Detailní popis požadavku najdete ve specifikaci.

Endpoint otevřít

```
GET: https://mujeshopik.cz/stiteknabalik-endpoint/?
error=access_denied&error_description=The+user+denied+access+to+your+application&state=csjkhd5b1
```

Krok 3. - Žádost o access token

V tomto kroku vyměníte získaný `authorization_code` za nový `access_token` a `refresh_token`.

S `access_tokenem` budete moci přes API přistupovat ke zdrojům po dobu jeho platnosti (**60 minut**).

Po expiraci `access_tokenu` si vyžádáte nový pomocí `refresh_tokenu`. Ten má neomezenou platnost.

> **TIP**
>
> Vzhledem ke krátkodobé platnosti access tokenu (a nutnému získávání stále nových tokenů pomocí refresh tokenu) je vhodné uložit refresh token do databáze (pokud vytváříte aplikaci pro více společností registrovaných v systému Štíteknabalík.cz) či do konfigurace aplikace (pokud vytváříte vlastní aplikaci pouze pro Vás).
> Samotný access token lze uložit do session s platností daného tokenu a snížit tak overhead potřebný pro přístup ke zdrojům této API. Není tedy nutné ukládat access token do databáze.
> Zároveň je velmi nevhodné si vyžádat nový access token pro každý přístup (REQUEST) ke zdrojům!

Ukázka HTTP Požadavku o access token (REQUEST)

Detailní popis požadavku najdete ve specifikaci.

Endpoint

```
POST: https://rest.stiteknabalik.cz/oauth/token/
```

Ukázka hlavičky

```
Authorization: empoeWdrbmtmazphYmNkMTIzNA==
Content-Type: application/x-www-form-urlencoded
```

Ukázka těla

```
Array
(
    [code] => 87f90ba9fe97e3b43f11c37eea1e1b475dbd59b9
    [redirect_uri] => https://mujeshopik.cz/stiteknabalik-endpoint/
    [scope] => deliveries collection-places
    [grant_type] => authorization_code
)
```

Popis parametrů požadavku (REQUEST)

Parametr	Příklad hodnoty	Popis
`Authorization`	Basic empoeWdrbmtmazphYmNkMTIzNA==	HTTP Basic autentizace vaší aplikace. Vznikne spojením slova "Basic" a identifikátoru OAuth klienta s heslem. To celé zformátované do base64. Příklad: `"Basic ".base64_encode($client_id.":".$client_secret)`
`Content-type`	application/x-www-form-urlencoded	Vždy udávejte tuto hodnotu. Tělo POST požadavku musí obsahovat hodnoty ve formátu application/x-www-form-urlencoded.
`code`	87f90ba9fe97e3b43f11c37eea1e1b475dbd59b9	Autorizační kód získaný v předešlé odpovědi
`redirect_uri`	https://mujeshopik.cz/stiteknabalik-endpoint/	Zaregistrovaná adresa pro přesměrování
`scope`	collection-places deliveries	Seznam požadovaných přístupů oddělených mezerou (každý zdroj má svůj scope, více v dokumentaci metod)
`grant_type`	authorization_code	Vždy jen **authorization_code**

Ukázka HTTP odpovědi žádosti o access token (REQUEST)

Vrácený `access_token` má platnost 60 minut (je obsaženo i v odpovědi v sekundách) a `refresh_token` má neomezenou platnost

Oba tokeny mají vždy délku 40 znaků a obsahují pouze malá písmena nebo čísla. Při získávání nových access tokenů pomocí refresh tokenu získáte stejnou strukturu bez položky `refresh_token`. V případě, že uživatel zpětně zamítne přístup Vaší aplikace k jeho údajům, jsou odstraněny všechny Vaše tokeny přidružené k tomuto uživateli

Detailní popis požadavku najdete ve specifikaci.

Ukázka hlavičky

```
HTTP/1.1 200
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Encoding: gzip
```

Ukázka těla

```
{
    "access_token": "f709547842cb9f5b891bb9dd3dcaf90d512f8ddd",
    "expires_in": "3600",
    "token_type": "bearer",
    "scope": "deliveries collection-places",
    "refresh_token": "406aabf0e1aedc3c600cbf7f591e9b439408f5c3"
}
```

Krok 4. - Přístup ke zdrojům

Všechny přístupy ke zdrojům vyžadují **HTTP Bearer autentizaci specifikovanou** v RFC 6750.
Použijte tedy `access_token` v HTTP hlavičce Authorization společně s prefixem **Bearer**.

Přihlašování pomocí tokenu v query parametru nebo v těle požadavku není podporováno.

Příklad požadavku (REQUEST)

Endpoint otevřít

```
GET: https://rest.stiteknabalik.cz/v4/deliveries?deliveryId=15023456
```

Ukázka hlavičky

```
Authorization: Bearer f709547842cb9f5b891bb9dd3dcaf90d512f8ddd
Accept: application/json
```

Příklad odpovědi expirovaného access tokenu (RESPONSE)

Je nutné ve Vaší aplikaci kontrolovat, zda používáte platný access token (v době jeho platnosti). Pokud dojde k požadavku s expirovaným či jinak neplatným tokenem, je vrácena chyba dle specifikace. Pro expirovaný token je vrácena následující odpověď:

Ukázka hlavičky

```
HTTP/1.1 401 Unauthorized
Authorization: Bearer f709547842cb9f5b891bb9dd3dcaf90d512f8ddd
WWW-Authenticate: Bearer realm="ClientApi", error="invalid_token", error_description="The access token provided has expired"
Accept: application/json
```

Ukázka těla

```
{
    "code": "401",
    "status": "error",
    "message": "The access token provided has expired",
    "errors": []
}
```

Krok 5. - Přístup ke zdrojům - refresh access tokenu

Tento krok provádíte vždy když expiruje platnost `access_tokenu`.

Pomocí `refresh_tokenu` si pro vašeho OAuth klienta vyžádáte náhradu původního `access_tokenu` za nový (s platností 60 minut)

Hlavička tohoto POST požadavku musí obsahovat totožné údaje jako v případě žádosti pomocí `authorization_code` (viz. bod 3.). V těle požadavku je nutné vyplnit `refresh_token`, `redirect_uri`, `scope`, `grant_type`.

Conten-type je opět **application/x-www-form-urlencoded**.

Popis parametrů požadavku (REQUEST)

Parametr	Příklad hodnoty	Popis
`Authorization`	Basic empoeWdrbmtmazphYmNkMTIzNA==	HTTP Basic autentizace vaší aplikace. Vznikne spojením slova "Basic" a identifikátoru OAuth klienta s heslem. To celé zformátované do base64. Příklad: `"Basic ".base64_encode($client_id.":".$client_secret)`
`Content-type`	application/x-www-form-urlencoded	Vždy udávejte tuto hodnotu. Tělo POST požadavku musí obsahovat hodnoty ve formátu application/x-www-form-urlencoded.
`refresh_token`	406aabf0e1aedc3c600cbf7f591e9b439408f5c3	Refresh token, který jste získali v žádosti o `access_token`
`redirect_uri`	https://mujeshopik.cz/stiteknabalik-endpoint/	Zaregistrovaná adresa pro přesměrování
`scope`	collection-places deliveries	Seznam požadovaných přístupů oddělených mezerou (každý zdroj má svůj scope, více v dokumentaci metod)
`grant_type`	refresh_token	Vždy jen `refresh_token`

Příklad požadavku s refresh tokenem (REQUEST)

Endpoint

```
POST: https://rest.stiteknabalik.cz/oauth/token
```

Ukázka hlavičky

```
Authorization: empoeWdrbmtmazphYmNkMTIzNA==
Content-Type: application/x-www-form-urlencoded
```

Ukázka těla

```
Array
(
    [refresh_token] => 406aabf0e1aedc3c600cbf7f591e9b439408f5c3
    [redirect_uri] => https://mujeshopik.cz/stiteknabalik-endpoint/
    [scope] => deliveries collection-places
    [grant_type] => refresh_token
)
```

Ukázka odpovědi

Ukázka hlavičky

```
HTTP/1.1 201
Content-Type: application/json
Content-Encoding: gzip
```

Ukázka těla

```
{
    "access_token": "943a8490c94e6750789e2bbb0a0272b8d3833869",
    "expires_in": "3600",
    "token_type": "bearer",
    "scope": "deliveries collection-places"
}
```

© 2018 - 2025 Foxdeli s.r.o.,