

Dokumentacja API

wersja API: 4.3

Metryka

Tytuł dokumentu	Dokumentacja API
Nazwa projektu	API
Autorzy dokumentu	Tomasz Czerko, Kajetan Jurkowski, Marek Kugacz, Krzysztof Pawlak, Jarosław Tempki, Piotr Zejer
Określenie poufności	Wewnątrz firmy oraz dla klientów korzystających z API
Wersja dokumentu	3
Status dokumentu	draft

Historia zmian dokumentu

LP	Data	Wersja dokumentu	Dot. API	Opis	Rozdziały
1	2016-01-28	1.0	1.0	Utworzenie dokumentu	wszystkie
2	2016-02-12	1.1	1.0	drobne poprawki treści	9.4, 9.5.2, 9.5.9, 9.5.11
3	2016-04-12	1.1	1.0	Rozszerzenie opisu metody field	9.4
4	2016-05-19	1.1	1.0	Poprawienie adresów i przykładów	9.4, 9.5.7
5	2016-06-09	1.1	1.1	Nowa metoda	9.4
6	2016-08-01	1.1	1.2	Nowy kontroler	9.4
7	2016-09-06	1.1	1.3	Nowe metody	9.4
8	2016-09-12	1.1	x.x	Doprecyzowanie zasad odświeżania sesji	7
9	2016-10-04	1.1	1.3	Nowe metody	9.4
10	2016-12-02	1.1	2.0, 1.4	Nowe metody	9.4
11	2017-01-03	2	2.1	Aktualizacja sygnatur metod z Events-controller	9.4

12	2017-02-02	2	3.0	Zmiana zasad limitowania zapytań	8
13	2017-02-09	2	2.1	Nowa metoda /files/resource	9.4
14	2017-02-13	2	3.0	Usunięcie 3 metod	9.4
15	2017-02-14	2	3.0	Aktualizacja informacji o limitach	8
16	2017-06-09	3	3.3	Rozszerzenie informacji zwracanej przez metodę <code>fields</code>	9.5.2 9.5.8 9.5.9 9.5.11
17	2017-06-19	3	3.3	Uaktualnienie informacji o wyjątkach	10.3
18	2017-10-25	4	4.3	Nowy kontroler	9.4

Spis treści

[Spis treści:](#)

[Zawartość dokumentu](#)

[Podstawowe informacje](#)

[Kontrakt](#)

[Wersjonowanie](#)

[Założenia](#)

[Informacja o zmianie wersji](#)

[Zmiany w API powodujące zmianę wersji \(non-backwards-compatible\)](#)

[Zmiany w API nie powodujące zmiany wersji \(backwards-compatible\)](#)

[Dokumentacja](#)

[Komunikacja](#)

[Autoryzacja](#)

[Limity zapytań](#)

[Interface API](#)

[Adres](#)

[Zapytania](#)

[Obiekt odpowiedzi \(Response\)](#)

[Metody API](#)

[Common-controller](#)

[Concept-controller](#)

[Document-controller](#)

[Files-controller](#)

[Yearbooks-controller](#)

[Events-controller](#)

[User-folder-controller](#)

[Obiekt Search](#)

[Referencja czasowa zapytania \(pit\)](#)

[Zwracane pola](#)

[Wyszukiwanie frazy](#)

[Rankowanie wyników wyszukiwania](#)

[Zwracanie aktualnej wersji dokumentu](#)

[Paginacja](#)

[Filtrowanie](#)

[Sortowanie](#)

[Zapytanie facetowe po polach](#)

[Zapytanie facetowe typu pivot \(zagnieżdżone\)](#)

[Facety wyników podzapytań](#)

[Obsługa błędów](#)

[Kody HTTP zwracane przez serwer](#)

[Obiekt błędu odpowiedzi](#)

[Lista możliwych błędów w zapytaniach klienta \(4xx\)](#)

1. Zawartość dokumentu

Dokument opisuje budowę i zasady korzystania z API udostępniającego dane z bazy Wolters Kluwer. Zawiera zestaw informacji potrzebnych do poprawnego odpytywania API oraz interpretacji odpowiedzi lub błędów.

2. Podstawowe informacje

API działa w oparciu o komunikację JSON over HTTP (opracowana w oparciu o założenia REST). Zapytania i odpowiedzi przesyłane są w formacie JSON wykorzystując metody protokołu HTTP. Komunikacja jest szyfrowana w oparciu o protokół HTTPS.

3. Kontrakt

Każda aplikacja kliencka może korzystać z predefiniowanego zestawu metadanych dokumentów.¹ Zakres zestawu podlegać będzie zmianom i może wpłynąć na zamianę wersji API, natomiast dostęp do metadanych wynika bezpośrednio z posiadanej przez klienta licencji.

4. Wersjonowanie

4.1. Założenia

Dostęp do danej wersji API odbywa się poprzez odpowiedni adres URL w którym zawarty jest główny numer wersji (major) z której klient chce skorzystać.

Zmiany w API, które zrywają kompatybilność są wystawiane w postaci nowej głównej wersji API powodując tym samym obsługę pod nowym adresem URL zawierającym główny numer wersji. Zmiana podwersji (minor) nie powoduje zerwania kompatybilności i są one wprowadzane pod adresem URL bieżącej wersji głównej API. **Aplikacje klienckie muszą być przygotowane na zmianę podwersji (minor) przy zachowaniu kontraktu.**

Po wydaniu nowej głównej wersji API (n) poprzednia wersja (n-1) jest utrzymywana jeszcze przez 30-dni kalendarzowych po czym jest usuwana. W tym czasie aplikacje klienckie muszą dostosować się i przełączyć na nową wersję. **Proces wydawniczy aplikacji klienckich musi uwzględniać 30-dniowy okres przejściowy przy zmianie głównej wersji API (np. aktualizacje aplikacji klienckich, propagacja nowego modelu danych lub kolekcji metadanych).** Wszystkie zmiany objęte wersjonowaniem zostaną ujęte w dokumentacji API.

¹ Listę dostępnych metadanych dokumentów dla danego klienta można pobrać za pomocą metody `fields` w API

4.2. Informacja o zmianie wersji

W momencie wydania nowej głównej wersji API, a więc zerwania kompatybilności wstecznej wszystkie zapytania, które korzystają z poprzedniej wersji API otrzymają informacje zwrotne dzięki którym klienci mogą niezwłocznie dowiedzieć się o konieczności przejścia na nową wersję API:

- a. ustawiany jest nagłówek odpowiedzi `X-API-Deprecated` z wartością `true`
- b. Obiekt odpowiedzi (`Response`) ma ustawioną w polu `deprecated` wartość `true`

Zmiany podwersji (minor) nie powodują dodanie nagłówka `X-API-Deprecated` i ustawienia pola `deprecated`.

4.3. Zmiany w API powodujące zmianę wersji (non-backwards-compatible)

- usunięcie/zmiana metody (adresu) API
- zmiana nazwy/usunięcie pola w obiekcie API
- zmiana nazwy/usunięcie pola w strukturze dokumentu
- dodanie/zmiana/usunięcie używanego kodu HTTP
- dodanie/zmiana/usunięcie zwracanego identyfikatora błędu (pole `exception`)
- zmniejszenie lub/i dodanie nowych limitów dla zapytań
- inne zmiany nieujęte w zmianach które nie powodują zmiany wersji API (poniżej)

4.4. Zmiany w API nie powodujące zmiany wersji (backwards-compatible)

- dodanie nowych metod (adresów) do API
- dodanie nowych opcjonalnych pól w istniejących zapytaniach
- dodanie nowych pól w odpowiedziach API
- zmiana kolejności zwracanych pól
- dodanie obsługi nowych metadanych dokumentu o które można odpytywać API
- zmiana zawartości zwracanych pól (np. imię i nazwisko autora, nazwa publikatora)
- zwiększenie (zluzowanie) limitów zapytań

- zmiana opisu szczegółów błędu (pola inne niż `exception`)

5. Dokumentacja

Oprócz niniejszego opracowania dokumentację API stanowi opis wygenerowany przez narzędzie Swagger znajdujący się pod adresem:

<https://api.lex.pl/swagger-ui.html>

Dostęp do dokumentu Swagger'a ograniczony jest hasłem, które klient otrzyma indywidualnie od Wolters Kluwer.

Narzędzie Swagger pozwala na wybór wersji API, której opis ma być zaprezentowany i na której mają być wykonane testy. Poprzednia wersja API opisana jest na liście wersji jako "DEPRECATED API".

6. Komunikacja

Komunikacja z API odbywa się poprzez przesyłanie obiektów JSON za pomocą protokołu HTTPS. Wykorzystano dwie metody protokołu:

- `GET` - stosowaną w przypadku prostych zapytań
- `POST` - stosowaną w przypadku złożonych zapytań, które wymagają przesłania obiektu JSON (obiekt `Search` opisany w dalszej części dokumentu)

Do autoryzacji używany jest nagłówek `Authorization` opisany w punkcie [Autoryzacja](#). Używane kody odpowiedzi HTTP zostały opisane w punkcie [Obsługa błędów](#).

7. Autoryzacja

W celu autoryzacji dostępu do API należy ustawić nagłówek zapytania `Authorization`, w którym należy wpisać identyfikator sesji uzyskany z Serwisu Logowania (BORG):

```
Authorization: {borgSessionId}
```

- `borgSessionId` - id sesji otrzymane w wyniku autentykacji w Serwisie Logowania (BORG)

W przypadku podania nieprawidłowego `borgSessionId` API zwróci kod HTTP 401 (`Unauthorized`) ze szczegółowym komunikatem (patrz [Obsługa błędów](#)).

Odpytywanie metod API nie powoduje odświeżania czasu życia sesji. Klient API jest odpowiedzialny za odświeżanie sesji w Serwisie Logowania (BORG).

8. Limity zapytań

API posiada limity:

- liczby zapytań wykonywanych dla konkretnego produktu w przeciągu doby
- zakresów zapytań i rozmiaru zwracanych list

Maksymalna liczba zapytań dla danego produktu w przeciągu doby zależna jest od formalnej umowy między Klientem a Wolters Kluwer. Domyślnie wynosi 50 000 zapytań na dobę, jednak jest wariant umowy, który przewiduje brak takiego ograniczenia.

Ustalono następujące limity dla zakresów zapytań i rozmiarów list :

- zapytanie o listę dokumentów zwraca w odpowiedzi maksymalnie 45 dokumentów.
- w przypadku zapytania o listę dokumentów zawierających pola z treścią, tj. content i plainTextContent obowiązuje bardziej restrykcyjny limit maksymalnie 1 dokument.
 - Wyjątkiem od powyższej zasady jest jawne zawężenie po polu contentLength, a jego wartość jest mniejsza niż 100 000.
- maksymalnie można pobrać 10 list (stron) zawierających 45 pozycji na stronie, a więc 450 pozycji listy wyników danego zapytania
- na liście zdarzeń (events) można uzyskać w odpowiedzi maksymalnie 45 pozycji zdarzeń.
- zakres dat w zapytaniu dla zdarzeń, jeżeli nie podajemy listy identyfikatorów dokumentów (nro^2) nie może być większy niż 7 dni. W przypadku podania identyfikatorów zakres nie może być większy niż 30 dni.
- maksymalna liczba dokumentów, którą można podać w zapytaniu o zdarzenia to 1

UWAGA - rekomendowanym sposobem pobierania treści dokumentów jest pobieranie treści każdego dokumentu z osobna.

W przypadku podania błędnych parametrów w zapytaniu, API zwróci kod HTTP 400 (Bad Request) ze szczegółowym komunikatem (patrz [Obsługa błędów](#)).

9. Interface API

9.1. Adres

API dostępne jest pod adresem: <https://api.lex.pl>

9.2. Zapytania

² nro to podstawowy unikalny identyfikator dokumentu w systemie WK API

Podstawowe zapytania, które wymagają podania kilku prostych parametrów wywoływane są za pomocą metody `GET` protokołu HTTP, bardziej skomplikowane wymagają przesłania obiektu `Search` ([Obiekt Search](#)) metodą `POST`.

Każde zapytanie wymaga przesłania nagłówka autoryzacyjnego (patrz [Autoryzacja](#)).

Wspólne parametry zapytań, które mogą wystąpić przy zapytaniach `GET`:

- `PIT` - data wg której ma być oceniana aktualność dokumentów. Domyślnie aktualna data.
- `fields` - lista pól danych po przecinku, które mają zostać zwrócone w odpowiedzi. Uwagi w [Zwracane pola](#).
`/documents/34234/54425?fields=nro,title,content`
- `from` - numer pozycji listy, od którego ma zostać zwrócona lista wyników. Domyślnie 0.
- `size` - maksymalna liczba pozycji w wyniku. Domyślnie 15.
- `sortField` - nazwa pola po którym mają być posortowane wyniki. Domyślnie puste.
- `sortOrder` - kierunek sortowania wyników. Możliwe wartości: `asc`, `desc`. Domyślnie `asc`.

Informację o tym, czy parametr jest używany przy wywołaniu można znaleźć w dokumentacji metod oraz w dokumentacji wygenerowanej przez Swaggera.

Wszystkie daty przesyłane do API jako parametry muszą mieć format `YYYY-MM-DD` lub `YYYY-MM-DD hh:mm:ss` w zależności od tego jakie pole chcemy obsługiwać. Pierwszy z formatów przeznaczony jest dla pól obsługujących datę, a drugi dla pól obsługujących datę i czas.

W przypadku pobierania plików z API użytkownik nie konstruuje samodzielnie wywołań, a jedynie posługuje się odnośnikami otrzymanymi w odpowiedziach API.

9.3. Obiekt odpowiedzi (**Response**)

Pola obiektu:

- `size` - liczba pozycji w wynikach
- `allDocumentCount` - liczba wszystkich dokumentów spełniających kryteria zapytania
- `results` - tablica z listą wyników
- `facets` - wyniki zapytań z wykorzystaniem mechanizmu facetingu³
- `facetsFilter` - wyniki facetingu podzapytań

³ faceting to jedna z metod kategoryzacji treści znalezionych w procesie wyszukiwania informacji np. do uzyskania unikalnej listy autorów publikacji spełniających określone kryteria zapytania.

- `deprecated` - informacja o tym, czy wykorzystywana jest najnowsza wersja API, jeżeli pole ma wartość `true`, oznacza to, że wersja API którą odpytujemy w najbliższym czasie zostanie usunięta i należy niezwłocznie przejść na nową wersję API

Każdy wynik prezentowany jest za pomocą mapy klucz - wartość (tablicy asocjacyjnej), gdzie kluczami są nazwy pól odpowiedzi. Wartość pola może być prostą wartością (np. liczbą czy ciągiem znaków), tablicą lub następną mapą klucz - wartość.

Przykład odpowiedzi:

```
{
  "size": 1,
  "allDocumentCount": 1,
  "results": [
    {
      "documentMainType": "ACT",
      "signature": "Dz.U.2014.121",
      "documentType": "ACT_DZUIMP",
      "nro": 16785996,
      "title": "Kodeks cywilny.",
      "version": 1917203
    }
  ],
  "facets": null,
  "facetsFilters": null
}
```

9.4. Metody API

Metody API:

Metoda “ping”

- **<https://api.lex.pl/ping>**
Jest to jedyna metoda, którą wywołujemy bez podania w adresie wersji API. Służy do sprawdzania połączenia z serwerem API.
 - Metoda: GET
 - Dane wejściowe: brak
 - Dane wyjściowe: tekst “OK”.

Common-controller

- **/fields**

Metoda zwraca listę możliwych metadanych dokumentu wraz z dostępnymi filtrami i typem danych.

- Metoda: GET
- Dane wejściowe: brak
- Dane wyjściowe: obiekt `Response` - lista możliwych pól dokumentów (metadanych) zwracanych przez API wraz z dostępnymi akcjami dla danego pola np. filtrowanie, faceting, zwracanie w wyniku. Dla filtrowania dodatkowo jest informacja z których filtrów można skorzystać dla danego pola oraz jaki typ danych należy przekazać do filtra. Jeżeli typ danych jest złożonym obiektem zwracana jest również lista pól tego obiektu.

- **/swaggerJSON**

Metoda zwraca dokument JSON, na podstawie którego można automatycznie wygenerować klienta API za pomocą narzędzia swagger-codegen.

- Metoda: GET
- Dane wejściowe: brak
- Dane wyjściowe: `document` w formacie JSON

Concept-controller

- **/letters**

Metoda zwraca listę liter haseł jednolitego indeksu

- Metoda: GET
- Dane wejściowe: brak
- Dane wyjściowe: obiekt `Response` - lista liter haseł jednolitego indeksu

- **/nodes**

Metoda zwraca drzewo haseł indeksu jednolitego

- Metoda: GET
- Dane wejściowe: brak
- Dane wyjściowe: obiekt `Response` - drzewo haseł jednolitego indeksu

- **/nodes/{letter}**

Metoda zwraca drzewo haseł indeksu jednolitego dla podanej litery

- Metoda: GET
- Dane wejściowe:
 - w URL: litera
- Dane wyjściowe: obiekt `Response` - drzewo haseł indeksu jednolitego

- **/relatedConcept/{conceptId}**

Metoda zwraca drzewo powiązanych haseł indeksu jednolitego dla podanego identyfikatora hasła indeksu jednolitego

- Metoda: `GET`
- Dane wejściowe:
 - w URL: identyfikator hasła indeksu jednolitego
- Dane wyjściowe: obiekt `Response` - drzewo powiązanych haseł indeksu jednolitego

- **`/preferredPath/{conceptId}`**

Metoda zwraca strukturę ścieżki haseł do hasła jednolitego indeksu dla podanych parametrów.

- Metoda: `GET`
- Dane wejściowe w URL:
 - `conceptId` - identyfikator hasła jednolitego indeksu
- Dane wejściowe w Parametrach
 - `pit` - opcjonalny
 - `uiInstanceOfType` (rodzaj hasła) - opcjonalny
Domyślnie: `generalIssues`
Enum (wartość -> nazwa):
 - `generalIssues` -> Zagadnienia ogólne
 - `specificIssues` -> Pozostałe zagadnienia
 - `internationalAgreements` -> Umowy międzynarodowe
 - `uiPreferredPath` (ścieżka jednolitego indeksu, która ma zostać zwrócona) - opcjonalna;
Domyślnie: zostanie zwrócona ścieżka pierwsza w porządku alfabetycznym
Format: nazwy haseł z preferowanej ścieżki oddzielone średnikami (np. Administracja publiczna;Rada Ministrów;Zagadnienia ogólne)
- Dane wyjściowe: obiekt `Response` - pole `preferredPath` w wynikach facetingu. Kolejne wartości, to kolejne elementy ścieżki do hasła jednolitego indeksu .

Document-controller

- **`/documents`**

Metoda zwraca listę dokumentów na podstawie zapytania `Search`

- Metoda: `POST`
- Dane wejściowe: obiekt `Search` w ciele zapytania
- Dane wyjściowe: obiekt `Response` - lista dokumentów

- **`/documents/directHit`**

Metoda zwraca nro, wersję i jednostkę najlepiej pasującego dokumentu dla sygnatury lub skrótu i jednostki podanych w parametrze `term`

- Metoda: `GET`
- Dane wejściowe:

- w parametrach: `string term` - obowiązkowy
 - w parametrach: `pit` - opcjonalny
 - Dane wyjściowe: obiekt `Response` z jednym dokumentem zawierającym 3 pola: `nro`, `version`, `actContentUnit`
- **`/documents/{nro}`**

Metoda zwraca dokument na podstawie identyfikatora `nro`

 - Metoda: `GET`
 - Dane wejściowe:
 - w URL: `nro`
 - w parametrach: kolekcja metadanych `fields`, `pit` - opcjonalny
 - Dane wyjściowe: obiekt `Response` - dokument
- **`/documents/{nro}/{version}`**

Metoda zwraca dokument na podstawie jego `nro` i numeru wersji

 - Metoda: `GET`
 - Dane wejściowe:
 - w URL: `nro` i numer wersji dokumentu
 - w parametrach: kolekcja metadanych `fields`, `pit`
 - Dane wyjściowe: obiekt `Response` - dokument
- **`/documents/{nro}/relations`**

Metoda zwraca informację o typach relacji oraz ich licznosciach dla dokumentu o podanym `nro`

 - Metoda: `GET`
 - Dane wejściowe:
 - w URL: `nro`
 - w parametrach: `pit`
 - Dane wyjściowe: obiekt `Response` - kolekcja z typami relacji i ich licznosciami
- **`/documents/{nro}/relatedConcept`**

Metoda zwraca informację o słowach kluczowych przypisanych do dokumentu o podanym `nro`

 - Metoda: `GET`
 - Dane wejściowe:
 - w URL: `nro`
 - w parametrach: `pit`, `version` jako wersja dokumentu
 - Dane wyjściowe: obiekt `Response` - kolekcja słów kluczowych wraz z liczbą dokumentów powiązanych z danym słowem
- **`/documents/{nro}/relations/{relation}`**

Metoda zwraca dokument o podanym `nro` wraz z dokumentami znajdującymi się z nim w danej relacji

- Metoda: GET
- Dane wejściowe:
 - w URL: `nro`, słownikowy identyfikator relacji (np.: `relationChangeActive`)
 - w parametrach: kolekcja metadanych `fields`, `pit`, `from`, `size`
- Dane wyjściowe: obiekt `Response` - dokument wraz kolekcją dokumentów powiązanych w ramach danej relacji

File-controller

Użytkownik API nie konstruuje samodzielnie wywołań metod z tego kontrolera, a jedynie posługuje się wywołaniami otrzymanymi w odpowiedziach API.

- **`/files/author`**

Metoda zwraca zdjęcie do notki o autorze - **link zwracany dla dokumentu w polu `authorsPhotoLink`**

- Metoda: GET
- Dane wejściowe:
 - w parametrach: `nro` pliku do pobrania - `nro` - obowiązkowy
 - w parametrach: wersja pliku do pobrania - `version` - obowiązkowy
- Dane wyjściowe: zdjęcie - plik PNG

Uwaga! Swagger nie umożliwia podglądu/pobrania odpowiedzi w takiej postaci. W dokumentacji widoczne będą jedynie nagłówki i kod odpowiedzi.

- **`/files/pdf`**

Metoda zwraca treści dokumentu w formacie PDF - **link zwracany dla dokumentu w polu `pdfLink`**

- Metoda: GET
- Dane wejściowe:
 - w parametrach: nazwa pliku do pobrania - `fileName` - obowiązkowy
 - w parametrach: `nro` pliku do pobrania - `nro` - opcjonalny
 - w parametrach: wersja pliku do pobrania - `version` - opcjonalny
- Dane wyjściowe: odpowiedź zawierająca treść dokumentu w formacie pdf

Uwaga! Swagger nie umożliwia podglądu/pobrania odpowiedzi w takiej postaci. W dokumentacji widoczne będą jedynie nagłówki i kod odpowiedzi.

- **`/files/resource`**

Metoda zwraca dodatkowe pliki powiązane z dokumentem - np. grafiki w treści dokumentu, odnośniki z pola **`fileLinks`**.

- Metoda: GET
- Dane wejściowe:
 - w parametrach: nazwa pliku do pobrania - `fileName` - opcjonalny
 - w parametrach: typ pliku do pobrania - `typ` - obowiązkowy
 - w parametrach: identyfikator pliku do pobrania - `id` - obowiązkowy

- w parametrach: rozszerzenie pliku do pobrania - `extension` - obowiązkowy
 - Dane wyjściowe: odpowiedź zawierająca treść pliku
- Uwaga! Swagger nie umożliwia podglądu/pobrania odpowiedzi w takiej postaci. W dokumentacji widoczne będą jedynie nagłówki i kod odpowiedzi.**
- **/files/docx**
Metoda zwraca treści dokumentu w formacie DOCX - **link zwracany dla dokumentu w polu `docxLink`**
 - Metoda: `GET`
 - Dane wejściowe:
 - w parametrach: nazwa pliku do pobrania - `fileName` - obowiązkowy
 - w parametrach: nro pliku do pobrania - `nro` - obowiązkowy
 - w parametrach: wersja pliku do pobrania - `version` - obowiązkowy
 - Dane wyjściowe: odpowiedź zawierająca treść dokumentu w formacie docx
- Uwaga! Swagger nie umożliwia podglądu/pobrania odpowiedzi w takiej postaci. W dokumentacji widoczne będą jedynie nagłówki i kod odpowiedzi.**

Yearbooks-controller

- **/yearbooks**
Metoda zwraca dostępne kategorie roczników.
 - Metoda: `GET`
 - Dane wejściowe: brak
 - Dane wyjściowe: obiekt `Response` - kolekcja nazw kategorii dokumentów
- **/yearbooks/{category}**
Metoda zwraca wydawców publikacji z danej kategorii.
 - Metoda: `GET`
 - Dane wejściowe:
 - w URL: kategoria dokumentów
 - Dane wyjściowe: obiekt `Response` - kolekcja wydawców w ramach danej kategorii dokumentów związanych z rocznikami
- **/yearbooks/{category}/{publisher}**
Metoda zwraca roczniki publikacji dla wybranego wydawcy z danej kategorii
 - Metoda: `GET`
 - Dane wejściowe:
 - w URL: kategoria dokumentów, id wydawcy
 - Dane wyjściowe: obiekt `Response` - kolekcja roczników publikacji wydawcy z danej kategorii
- **/yearbooks/{category}/{publisher}/{year}**
Metoda zwraca pozycje danego rocznika publikacji dla wybranego wydawcy z danej kategorii
 - Metoda: `GET`

- Dane wejściowe:
 - w URL: kategoria roczników, id wydawcy, rok
- Dane wyjściowe: obiekt `Response` - lista dokumentów z podanego roku dla wydawcy i kategorii dokumentów

Events-controller

● `/events`

Metoda zwraca kolekcję zdarzeń związanych z dokumentami w określonym przedziale czasu.

- Metoda: `GET`
- Dane wejściowe:
 - `dateFrom` - data początkowa
 - `dateTo` - data końcowa. Maksymalna różnica między datą `{od}`, a datą `{do}` może wynieść 7 dni.
 - `from` - początek strony
 - `size` - rozmiar strony
 - `sort` - sortowanie, domyślnie malejące (`DESC`)
- Dane wyjściowe: obiekt `Response` - kolekcja zdarzeń związanych z dokumentami wraz z ich wskazaniem

● `/eventsForNros`

Metoda zwraca listę zdarzeń związanych z dokumentem.

- Metoda: `GET`
- Dane wejściowe:
 - tablica `nro` - maksymalny rozmiar tablicy `nro` wynosi w tej chwili 1 element
 - `dateFrom` - data początkowa
 - `dateTo` - data końcowa. Maksymalna różnica między datą `{od}`, a datą `{do}` może wynieść 30 dni.
 - `from` - początek strony
 - `size` - rozmiar strony
 - `sort` - sortowanie, domyślnie malejące (`DESC`)
- Dane wyjściowe: obiekt `Response` - lista zdarzeń dokumentu

User-folder-controller

● `/userFolders/documents/{actualizationIdentifier}`

Metoda zwraca dokumenty, które pojawiły się po dacie wskazanej w identyfikatorze aktualizacji

- Metoda: `GET`
- Dane wejściowe: `actualizationIdentifier`
- Dane wyjściowe: obiekt `UserDocumentsResponse` - mapa klucz-wartość zawierająca dokumenty użytkownika i dane aktualizacji (status odpowiedzi, liczba dokumentów, identyfikator następnej aktualizacji)

- **/userFolders/synchronize**

Metoda zwraca wszystkie dokumenty użytkownika. Jeśli podany zostanie identyfikator aktualizacji, to dokumenty zostaną zwrócone tylko w przypadku, gdy od podanej w nim daty nastąpiły jakiegokolwiek zmiany

- Metoda: GET
- Dane wejściowe: `actualizationIdentifier` (opcjonalnie)
- Dane wyjściowe: obiekt `UserDocumentsResponse` - mapa klucz-wartość zawierająca dokumenty użytkownika i dane aktualizacji (status odpowiedzi, liczba dokumentów, identyfikator następnej aktualizacji)

Tree-structures-controller

- **/checklistCategory**

Metoda zwraca drzewo kategorii dla dokumentów typu CHECKLIST

- Metoda: GET
- Dane wejściowe:
 - w parametrach: `nrs` (opcjonalny) - nrs kategorii której poddrzewo chce pobrać użytkownik
- Dane wyjściowe: obiekt `Response` - drzewo kategorii w polu `tree`

9.5. Obiekt Search

Dokładna budowa obiektów zapytania jest prezentowana w dokumentacji wygenerowanej przez Swaggera. Niniejsza dokumentacja skupia się na sposobie korzystania z API.

Obiekt zawiera właściwości pozwalające na skonstruowanie złożonych zapytań do API wg potrzeb klienta np. zapytanie o daną frazę, filtrowanie zwracanych wartości, paginacja i sortowanie, zapytania korzystające z mechanizmu faceting.

9.5.1. Referencja czasowa zapytania (pit)

W celu zdefiniowania momentu w czasie względem, którego dokonana ma zostać ocena obowiązywania dokumentów, należy podać datę w polu `pit` (`pointInTime`). Domyślnie jako wartość `pit` przyjmowana jest aktualna data.

Przykład:

```
{
  "pit": "2012-10-04",
  "query" : {
    "fields": [ "editionPosition", "nro", "title",
    "authors" ]
  }
}
```

```
}  
}
```

9.5.2. Zwracane pola

Listę pól, które chcemy otrzymać w odpowiedzi należy podać jako tablicę w ramach właściwości `fields` obiektu `Query`.

Informacja które pola możliwe są do zwrócenia przez API w odpowiedzi zwracana jest przez metodę `fields` znacznik: `result`. Niektóre pola wymagają zastosowania filtra, aby uzyskać ich wartość w takim przypadku metoda `fields` zwraca dla tego pola dodatkowy znacznik: `filterRequired`.

Przykład:

```
{  
  "pit": "2012-10-04",  
  "query" : {  
    "fields": [ "editionPosition", "nro", "title",  
              "authors"]  
  }  
}
```

Jeżeli właściwość `fields` nie zostanie określona to do odpowiedzi zostaną dołączone pola domyślne: `nro, version, title, documentMainType, documentType, signature`

Uwagi:

- w przypadku podania nazwy metadanej, które nie istnieje w API (np. literówka) API zwróci wyjątek `NoSuchFieldException`
- w przypadku zapytania o metadana, która z definicji jest polem bezzwrotnym, czyli nie zwraca wartości (pole opisane jest flagą `result:false`, natomiast lista pól dostępna jest, jako wynik wywołania metody `fields`) API zwróci wyjątek `IncludeInResultNotAllowedException`
- jeżeli zapytanie dotyczy metadanych, które nie istnieją dla danego typu dokumentu, API zwróci je z wartością `null`

9.5.3. Wyszukiwanie frazy

Frazę, którą chcemy wyszukać podajemy jako właściwość `term` obiektu `Query`.

Przykład:

```
{  
  "pit": "2012-10-04",  
  "query" : {
```

```
    "fields": [ "editionPosition", "nro", "title",  
    "authors"],  
    "term": "podatek",  
    "boost": true  
  }  
}
```

9.5.4. Rankowanie wyników wyszukiwania

Włączenie mechanizmu sortowania wyników zapytania według trafności uzyskujemy ustawiając właściwość `boost` obiektu `Query`.

Przykład:

```
{  
  "pit": "2012-10-04",  
  "query" : {  
    "fields": [ "editionPosition", "nro", "title",  
    "authors"],  
    "term": "podatek",  
    "boost": true  
  }  
}
```

9.5.5. Zwracanie aktualnej wersji dokumentu

W celu uzyskania aktualnych wersji dokumentu dla danej wartości parametru `pit` (domyślnie aktualna data) należy przypisać właściwości `versionBasedOnPit` wartość `true` w ramach obiektu `Search`.

Przykład:

```
{  
  "pit": "2012-10-04",  
  "versionBasedOnPit": true,  
  "query" : {  
    "fields": [ "editionPosition", "nro", "title",  
    "authors"]  
  }  
}
```

9.5.6. Paginacja

Paginacja jest definiowana na podstawie dwóch właściwości obiektu `Query`.

- `from` - od której pozycji włącznie mają być zwrócone elementy wyniku
- `size` - liczba elementów w wyniku

Przykład:

```
{
  "query" : {
    "fields": [ "position", "nro", "title", "authors"],
    "from" : 10,
    "size" : 5
  }
}
```

9.5.7. Filtrowanie

Obiekt `Query` pozwala również na określenie dodatkowych kryteriów zapytania w postaci kolekcji warunków stanowiących wartość właściwości `filters`. Każdy element kolekcji posiada pole `field` określające nazwę pola, którego dotyczy filtrowanie oraz kryteria dla wartości wskazanego pola:

1. filtrowanie wg dokładnie jednej wartości uzyskamy dzięki polu `eq`, którego wartość może to być ciągiem znaków, liczbą lub obiektem - zgodnie z wytycznymi danego pola opisanymi w wyniku metody `fields`)
2. filtrowanie wartości wg zakresu ustawiamy odpowiednio polami:
 - a. `lt` - wartość mniejsza od
 - b. `lte` - wartość mniejsza od lub równa
 - c. `gt` - wartość większa od
 - d. `gte` - wartość większa od lub równa
3. filtrowanie wg grupy możliwych wartości z wykorzystaniem pola `in` w którym podajemy listę wartości (mogą to być ciągi znaków, liczby lub obiekty - zgodnie z wytycznymi danego pola opisanymi w wyniku metody `fields`)

Uwaga! Nie można ustawić w filtrze wartości z dwóch grup np. `eq` i dowolnej z wartości `lt`, `lte`, `gt`, `gte` lub `in`. Spowoduje to zwrócenie błędu `InvalidFilterException`.

Informacja z których typów filtrów można skorzystać przy danym polu jest zwracana przez metodę `fields`.

Przykład:

```
{
  "query" : {
    "fields": [ "editionPosition", "nro", "title", "authors"],
    "from" : 10,
    "size" : 5,
    "filters": [
      {
        "eq": "ACT",
        "field": "documentMainType"
      },
    ],
  }
}
```

```
{
  "gt": "22-05-2014",
  "lt": "22-05-2015",
  "field": "enactmentDate"
},
{
  "in": ["ACT_DZUIMP", "ACT_MIEJSCOWY"],
  "field": "documentType"
}
]
}
```

9.5.8. Sortowanie

Sortowanie listy wynikowej uzyskujemy poprzez zastosowanie właściwości `sorts`, jako kolekcję wskazującą pole sortowania oraz kierunek:

- `field` - nazwa pola, po którym ma być sortowany wynik
- `sort` - kierunek sortowania: `asc` - rosnąco, `desc` - malejąco

Informacja po których polach możliwe jest sortowanie zwracana jest przez metodę `fields`, znacznik: `sort`.

Przykład:

```
{
  "query" : {
    "fields": [
      "editionPosition", "nro", "title", "authors"
    ],
    "from": 0,
    "size": 10,
    "sorts": [
      {
        "field": "title",
        "sort": "desc"
      }
    ]
  }
}
```

9.5.9. Zapytanie korzystające z mechanizmu faceting określonych pól

Zapytanie korzystające z mechanizmu faceting pozwala na uzyskanie listy możliwych wartości określonych metadanych. Konstrukcja zapytania musi zawierać obiekt `Facet` ze zdefiniowaną listą pól `fields` (lista obiektów `FacetField`)

Informacja na których polach możliwe jest stosowanie mechanizmu facetingu zwracana jest przez metodę `fields`, znacznik: `facet`.

`FacetField` posiada pola:

- `field` - nazwa pola, po którym ma być wykonane zapytanie - pole obowiązkowe
- `minCount` - minimalna liczność od której zwracany jest wynik, czyli ile minimalnie elementów musi zawierać wynik wyszukiwania, aby można było je zwrócić - domyślnie 0
- `limit` - maksymalna liczba zwracanych elementów - domyślnie -1 (brak limitu)
- `sort` - słownikowe kryterium sortowania: `count` - wg liczności, `index` - alfabetycznie - domyślnie `count`

Przykład:

```
{
  "facet":
  {
    "fields" : [
      {
        "field": "actValidity"
      },
      {
        "field" : "objectValidity",
        "minCount" : 0,
        "limit" : 10,
        "sort" : "count"
      }
    ]
  }
}
```

9.5.10. Zapytanie wg mechanizmu faceting typu pivot (zagnieżdżone)

Zapytanie pivot uzyskujemy wstawiając obiekt `Facet` wraz z podaniem pola `pivot` (obiekt `Pivot`)

`Pivot` posiada pola:

- `fields` - tablica nazw pól, po którym ma być wykonane zapytanie pivot - pole obowiązkowe
- `minCount` - minimalna liczność od której zwracany jest wynik - domyślnie 0

Przykład:

```
{
  "facet":
```

```
{
  "pivot" : {
    "fields": ["documentMainType", "documentType"],
    "minCount":5
  }
}
```

9.5.11. Mechanizm faceting dla wyników podzapytań

Wykonując podzapytania i korzystając z mechanizmu faceting można uzyskać licznosci elementów w wynikach objętych dodatkowym filtrowaniem np. liczbę dokumentów z danego miesiąca.

Konstrukcja zapytania polega na zdefiniowaniu listy filtrów (obiektów `FacetSubQuery`) w ramach pola `subQueries` obiektu `facet`.

Informacja na których polach możliwe jest stosowanie mechanizmu `subQueries` zwracana jest przez metodę `fields`, znacznik: `subQuery`.

`FacetSubQuery` działa analogicznie i posiada te same pola co standardowy filtr obiektu `query` oraz dodatkowo zawiera pole `label` w którym należy podać własną etykietę dla wyniku zapytania:

Przykład:

```
{
  "facet":{
    "subQueries":[
      {
        "field":"noveltyDate",
        "label":"YESTERDAY",
        "eq":"2015-10-13"
      },
      {
        "field":"noveltyDate",
        "label":"LAST_WEEK",
        "gte":"2015-10-07"
      },
      {
        "field":"noveltyDate",
        "label":"LAST_TWO_WEEKS",
        "gte":"2015-09-30"
      },
      {
        "field":"noveltyDate",
        "label":"LAST_MONTH",
        "gte":"2015-09-13"
      }
    ]
  }
}
```



```
    }  
  ]  
}  
}
```

10. Obsługa błędów

10.1. Kody HTTP zwracane przez serwer

Przy braku błędów serwer zwraca kod 200 oraz dane odpowiedzi.

W przypadku wystąpienia błędów zwracane są następujące kody:

- kod 4xx – oznacza problem po stronie klienta API. Klient powinien poprawić zapytanie i wysłać je ponownie. Obsługa błędów tego typu nie wymaga angażowania zespołu API i powinna być rozwiązana przez klienta.
 - 400 - Bad Request – nieprawidłowe zapytanie np. błędna nazwa lub użycie filtra, próba zawężania po wartości nienumerycznej dla pola numerycznego itp.
 - 401 – Unauthorized – brak lub nieprawidłowy borgSessionId
 - 403 – Forbidden – próba pobrania zasobu do którego dany klient nie ma dostępu – np. zapytanie o pole, którego klient nie ma w kontrakcie, brak dostępu do API
 - 429 - Too Many Request - przekroczony został dozwolony dobowy limit zapytań
- kod 500 - Internal server error – oznacza problem po stronie serwera np. błąd połączenia ze źródłem danych. Błędy tego typu muszą zostać obsłużone i rozwiązane przez zespół API.

Kod 404 jest zarezerwowany dla sytuacji gdy serwer nie obsługuje danego adresu (typowe HTTP NOT FOUND), a nie dla sytuacji, gdy nie znaleziono obiektu wg zadanych kryteriów np. pobieranie obiektu po identyfikatorze.

10.2. Obiekt błędu odpowiedzi

Zawartość odpowiedzi błędu stanowi JSON z dodatkowymi informacjami:

- `exception` – nazwa wyjątku będąca jednocześnie identyfikatorem wyjątku
- `message` – informacja ze szczegółowym opisem problemu

- `correlationId` - unikalny identyfikator zdarzenia - pozwala połączyć zapytanie klienta z wyjątkiem w logach serwera
- `path` – wywoływana ścieżka
- `stackTrace` – tylko w trybie debug – zrzut stosu wywołań

Opisy i nazwy błędów są podawane w języku angielskim.

10.3. Lista możliwych błędów w zapytaniach klienta (4xx)

- `ApiAccessException` - brak dostępu do API dla użytkownika
- `UnknownBorgSessionIdException` - brak lub nieprawidłowe `borgSessionId`
- `LimitExceedException` - osiągnięto dzienny limit zapytań dla wybranego produktu
- `InvalidParameterValueException` - błędna wartość parametru w podanego w adresie
- `FieldForFilterNotSupportedException` - nie można użyć filtra dla danego pola metadanych
- `InvalidFilterException` - błędna składnia pola filter np. użycie jednocześnie `ól` `eq` oraz `in`
- `InvalidFilterValueException` - błędna wartość w polu filtra np. nieprawidłowa nazwa zdarzenia do filtrowania w kalendarium
- `InvalidFilterValueTypeException` - błędna typ wartości w polu filtra np. nieprawidłowy format daty, ciąg znaków w polu numerycznym itp.
- `IncompleteObjectForFilterException` - obiekt przekazany jako wartość w polu filtra nie posiada wszystkich wymaganych pól
- `InvalidRangeException` - przedział podany w filtrze jest nieprawidłowy np. podano zarówno wartość `lt` jak i `eq`
- `DateRangeNotAllowedException` - użyto wartości do filtrowania po zakresie dat do pola, dla którego takie filtrowanie nie jest dozwolone
- `FacetForFieldNotAllowedException` - zapytano o facet do pola, dla którego nie jest to dozwolone
- `FieldNotSetException` - nie ustawiono pola, dla którego ma zostać wykonane filtrowanie
- `IncludeInResultNotAllowedException` - poproszono o zwrócenie w wyniku pola, dla którego nie jest to dozwolone

- `IncludeInResultWithoutFilterNotAllowedException` - poproszono o zwrócenie w wyniku pola, dla którego obowiązkowe jest filtrowanie po tym polu w zapytaniu (np. wynik pola zależy od wartości filtru)
- `SortForFieldNotAllowedException` - nie można sortować po podanym polu
- `SortWithoutFilterNotAllowedException` - ustawiono sortowanie po polu, dla którego obowiązkowe jest filtrowanie po tym polu w zapytaniu
- `FilterNotAllowedException` - próbowano filtrować po polu, dla którego nie jest to dozwolone
- `NoSuchFieldException` - wykorzystano w zapytaniu pole które nie istnieje w systemie
- `DocumentsListSizeExceedException` - zapytanie dotyczyło zbyt dużej liczby dokumentów (np. wartość w polu `size`)
- `DocumentsListStartNumberExceedException` - przekroczona została maksymalna wartość liczby wyniku od którego pobrane zostaną następane dokumenty (np. wartość w polu `from`)
- `DocumentsWithContentListSizeExceedException` - zapytanie dotyczyło zbyt dużej liczby wyników z polem `content` należy zmniejszyć liczbę oczekiwanych dokumentów do zwrócenia, albo usunąć pole `content` z listy pól wynikowych `fields`
- `QueryListSizeExceedException` - przekroczono dopuszczalną liczbę elementów listy w zapytaniu
- `DateRangeNotAllowedException` - zapytanie obejmuje zbyt duży zakres dat
- `NoSuchYearbookCategoryException` - brak podanej kategorii dla roczników
- `TooManyNrosTemporaryException` - zbyt wiele podanych wartości pola `nro` w zapytaniu
- `FacetSubQueryForFieldNotAllowedException` - zapytano o facet wyników podzapytań dla pola dla którego nie jest to dozwolone
- `DeprecatedFieldException` - użyte pole zostanie usunięte w kolejnej wersji API i nie powinno być już używane
- `FileNotFoundException` - nie znaleziono żądanego pliku - należy sprawdzić poprawność ścieżki wywołania API
- `PointInTimeNotAllowedException` - parametr `pointInTime` poza dozwolonym zakresem dat
- `RelationsPerRequestExceedException` - przekroczona dopuszczalna liczba relacji w zapytaniu
- `IPAddressNotAuthorizedException` - adres IP z którego następuje połączenie z API nie jest w puli dozwolonych adresów

- `EmptyNarrowingListException` – próba zawężenia do pustej listy wartości