Рекомендации по получению информации о казначейских счетах и иных реквизитах для перевода денежных средств в бюджетную систему Российской Федерации

Версия 1.3 (08.12.2025)

Оглавление

Шаг 1. Используемые справочники

Для получения казначейских счетов необходимо использовать:

- справочник Книга регистрации казначейских счетов (обновляется в режиме реального времени);
- Справочник БИК – для получения данных о банковских идентификационных кодах (БИК) и наименовании банка (обновляется ежедневно).

[1]:

```
import requests
import xml.etree.ElementTree as ET
from datetime import datetime, timedelta
from IPython.display import clear_output
import os
import json
import csv
import zipfile
import time

ts_start = time.time()
print('Начато: {}'.format(datetime.fromtimestamp(ts_start).strftime('%d.%m.%Y
%H:%M:%S')))
```

Шаг 2. Загрузка Справочника казначейских счетов

С использованием API необходимо постранично загрузить данные о казначейских счетах.

В зависимости от выбранного подхода возможно применение двух вариантов загрузки справочника:

- **Вариант 1**: периодическая загрузка полного актуального справочника;
- **Вариант 2**: разовая загрузка полного актуального справочника с периодической загрузкой изменений к нему. Такой вариант позволит загружать информацию быстрее, но требует реализации дополнительной логики обработки получаемых записей:
    - необходимо добавлять новые и исключать устаревшие записи во внутреннем справочнике на основании полученных изменений;
    - необходимо хранить и учитывать дату начала и окончания функционирования казначейского счета, который может быть открыт или закрыт будущей датой.

> **ⓘ Обратите внимание**
>
> Вне зависимости от выбранного варианта рекомендуется обновлять справочник с периодичностью *не менее* одного раза в сутки по рабочим дням.
>
> Справочник казначейских счетов ведется с поддержанием истории изменений, то есть одному номеру казначейского счета соответствует одна или несколько записей справочника, но только одна из этих записей может являться актуальной (**status** = `ACTIVE`).

2.1. Загрузка полного справочника

> ✔ **Шаг применим к Вариантам 1 и 2**

Для получения полного справочника, актуального на момент загрузки, следует использовать следующие фильтры:

- **pageSize** – число элементов на одной странице. Рекомендуется выводить 20 элементов;
- **pageNum** – номер загружаемой страницы;
- **filteractual** – признак актуальности записи. Поле может принимать значения `true` (запись актуальна) или `false` (запись неактуальна);
- **filterstatus** – статус записи. Одному казначейскому счету могут соответствовать несколько записей справочника в случае внесения изменений, например, в наименование или КПП организации – владельца счета. Но всегда у одного казначейского счета одна активная запись. Поле может принимать значения:
  - `ACTIVE` - актуальная запись,
  - `ARCHIVE` - архивная запись;
- **filter_start_accountnum** – маска номера казначейского счета, отбор проводится начиная слева;
- **filter_not_stateks** – статус казначейского счета, отбор проводится с условием "не". Поле может принимать значения:
  - `KS1` - открыт,
  - `KS2` - условно закрыт (прекращены операции по счету),
  - `KS3` - закрыт.

[2]:

```
url = 'http://budget.gov.ru/epbs/registry/7710568760-
KRKS/data?pageSize=20&pageNum=%page%&filteractual=true&filterstatus=ACTIVE&fi
lter_start_accountnum=40102810&filter_not_stateks=KS3'

pagenum = 1
pagecount = 1
recordscount = 0
data = []

while True:
    data_json = requests.get(url.replace('%page%', str(pagenum))).json()
    if not data_json['data']:
        break
    if pagenum == 1:
        pagecount = data_json['pageCount']
        recordsCountTotal = data_json["recordCount"]
    data.extend(data_json['data'])
    recordsCount = len(data)
    clear_output(wait=True)
    print(f'Обработано страниц: {pagenum} из {pagecount}')
    print(f'            записей: {recordsCount} из {recordsCountTotal}
({round(recordsCount/recordsCountTotal*100, 1)}%)')
    pagenum += 1
```

2.2. Форматирование и отбор данных

> ✔ **Шаг применим к Вариантам 1 и 2**

Приводим к нужному формату отдельные поля справочника и оставляем минимально необходимые данные:

- **ksnumber** - номер казначейского счета;
- **clientname** - полное наименование участника системы казначейских платежей, которому открыт казначейский счет;
- **accountnum** - номер единого казначейского счета;
- **tofkbik** - БИК территориального органа Федерального казначейства;
- **dateopen** - дата открытия казначейского счета. *Необходимо учитывать*, что дата открытия может превышать дату включения записи в справочник;
- **opentofkcode** - код территориального органа Федерального казначейства по месту открытия казначейского счета (может использоваться для группировки записей);
- **signls** - признак открытия лицевого счета, может принимать значения:
    - `LS1` – лицевой счет открыт в территориальном органе Федерального казначейства,
    - `LS2` – лицевой счет открыт в финансовом органе (органе управления государственным внебюджетным фондом).

> ⓘ **Обратите внимание**
>
> Можно сохранить и другие поля при необходимости. Полный перечень полей приведен в описании по ссылке на Шаге 1.
>
> Поля **clientname** и **signls** помогут при заполнении реквизита «Плательщик» распоряжения. Согласно Положению Банка России «О ведении Банком России и кредитными организациями (филиалами) банковских счетов территориальных органов Федерального казначейства» в реквизите «Получатель» указывается полное или сокращенное наименование органа, которому открыт казначейский счет, в скобках полное или сокращенное наименование организации - получателя средств.

[3]:

```
keep = ['ksnumber', 'clientname', 'accountnum', 'tofkbik', 'dateopen',
'opentofkcode', 'signls']
newdata = []

for d in data:
    d = (dict(filter(lambda x: x[0] in keep, d.items())))
    d['dateopen'] = datetime.strptime(d['dateopen'], '%Y-%m-%d
%H:%M:%S.%f').strftime('%d.%m.%Y')
    d['clientname'] = d['clientname'].replace('\n', '')
    d = {k: d[k] for k in keep if k in d}
    newdata.append(d)

print(f'Прошло времени: {round(time.time()-ts_start,1)} сек.')
```

2.3. Загрузка изменений

> ✔ **Шаг применим к Варианту 2**

Для получения изменений следует использовать один из следующих фильтров по дате включения записи в справочник:

- **filterloaddate** - отбор записей, включенных в справочник в определенную дату;
- **filter_more_loaddate** - отбор записей, включенных в справочник в даты, позднее указанной (включительно);
- **filter_less_loaddate** - отбор записей, включенных в справочник в даты, ранее указанной (включительно).

Используя комбинацию фильтров **filter_more_loaddate** и **filter_less_loaddate** можно отобрать записи за диапазон дат.

Значение фильтра указывается в формате `ДД.ММ.ГГГГ`.

[4]:

```
## Для отбора записей, включенных в справочник в конкретный день:
url = 'http://budget.gov.ru/epbs/registry/7710568760-
KRKS/data?pageSize=20&pageNum=%page%&filter_start_accountnum=40102810&filterl
oaddate=%date%'
##
## Для отбора записей, включенных в справочник в даты, позднее указанной
(включительно):
##  url = 'http://budget.gov.ru/epbs/registry/7710568760-
KRKS/data?pageSize=20&pageNum=%page%&filter_start_accountnum=40102810&filter_
more_loaddate=%date%'
##
## Для отбора записей, включенных в справочник в даты, ранее указанной
(включительно):
##  url = 'http://budget.gov.ru/epbs/registry/7710568760-
KRKS/data?pageSize=20&pageNum=%page%&filter_start_accountnum=40102810&filter_
less_loaddate=%date%'
##
## Для отбора записей, включенных в справочник в диапазоне дат:
##  url = 'http://budget.gov.ru/epbs/registry/7710568760-
KRKS/data?pageSize=20&pageNum=%page%&filter_start_accountnum=40102810&filter_
more_loaddate=%date1%&filter_less_loaddate=%date2%'

pagenum = 1
pagecount = 1
recordscount = 0
yesterday = datetime.today() - timedelta(days=1)
loaddate = yesterday.strftime("%d.%m.%Y")
data = []

while True:
    data_json = requests.get(url.replace('%page%',
str(pagenum)).replace('%date%', loaddate)).json()
    if not data_json['data']:
        break
    if pagenum == 1:
        pagecount = data_json['pageCount']
        recordsCountTotal = data_json["recordCount"]
    data.extend(data_json['data'])
    recordsCount = len(data)
    clear_output(wait=True)
    print(f'Обработано страниц: {pagenum} из {pagecount}')
```

```
    print(f'           записей: {recordsCount} из {recordsCountTotal}
({round(recordsCount/recordsCountTotal*100, 1)}%)')
    pagenum += 1
```

2.4. Форматирование и отбор данных

> ✔ **Шаг применим к Варианту 2**

Приводим к нужному формату отдельные поля справочника и оставляем минимально необходимые данные.

При этом по сравнению с Шагом 2.2 необходимо дополнительно сохранить поля, исходя из которых добавлять новые или исключать ранее загруженные записи, а также проверять актуальность казначейского счета на дату совершения платежа:

- **status** - статус записи, может принимать значения:
  - `ACTIVE` - актуальная запись,
  - `ARCHIVE` - архивная запись;
- **stateks** - статус казначейского счета, может принимать значения:
  - `KS1` – открыт,
  - `KS2` – условно закрыт (прекращены операции по счету),
  - `KS3` – закрыт;
- **closedate** - дата закрытия казначейского счета.

Для этого необходимо в блоке кода [3] задать следующее значение для переменной `keep`:

```
keep = ['ksnumber', 'clientname', 'accountnum', 'tofkbik', 'dateopen',
'opentofkcode', 'signls', 'status', 'stateks', 'closedate']
```

> ⓘ **Обратите внимание**
>
> Можно сохранить и другие поля при необходимости. Полный перечень полей приведен в описании по ссылке на Шаге 1.

2.5. Обработка изменений

> ✔ **Шаг применим к Варианту 2**

Полученные на Шаге 2.4 записи необходимо обработать, внеся изменения во внутренний справочник потребителя в следующей последовательности:

1. Исключить (или перенести в архив) запись из внутреннего справочника, если (один из вариантов):
   - поле **status** принимает значение `ARCHIVE` (архивная);
   - поле **stateks** принимает значение `KS2` (условно закрыт) или `KS3` (закрыт) и поле **closedate** меньше либо равно дате обновления;
2. Включить запись во внутренний справочник, если поле **status** принимает значение `ACTIVE` (актуальная) и поле **stateks** принимает значение `KS1` (открыт). При этом необходимо учитывать, что казначейские счета могут быть открыты будущей датой.

Шаг 3. Формирование Справочника ЕКС

3.1. Выделяем единые казначейские счета в отдельный справочник

[5]:

```
keep = ['accountnum', 'tofkbik', 'opentofkcode']
trsa = []

for d in newdata:
    d = (dict(filter(lambda x: x[0] in keep, d.items())))
    d = {k: d[k] for k in keep if k in d}
    trsa.append(d)

trsa = [i for n, i in enumerate(trsa) if i not in trsa[n + 1:]]

print('Записей в справочнике ЕКС: {}'.format(len(trsa)))
print(f'Прошло времени: {round(time.time()-ts_start,1)} сек.')
```

3.2. Объединяем справочник с данными из Справочника БИК

Объединяем справочник с данными из Справочника БИК (см. Шаг 1) для связки с наименованиями банков в формате, определенном Положением Банка России «О ведении Банком России и кредитными организациями (филиалами) банковских счетов территориальных органов Федерального казначейства» *(наименование банка, обслуживающего территориальный орган Федерального казначейства, знак «//», сокращенное наименование и место нахождения территориального органа Федерального казначейства)*.

[6]:

```
bikFile = 'bik.zip'
ns = '{urn:cbr-ru:ed:v2.0}'

r = requests.get('https://www.cbr.ru/s/newbik', allow_redirects=True,
verify=False)
open(bikFile, 'wb').write(r.content)

zip_file = zipfile.ZipFile(bikFile, 'r')
zip_file.extractall()
xmlfile = zip_file.namelist()[0]
zip_file.close()

f = open(xmlfile)
xml = ET.fromstring(f.read())
f.close()

for a in trsa:
    bic = a['tofkbik']
    for entry in xml.findall(f'{ns}BICDirectoryEntry/[@BIC="{bic}"]'):
        NameCBR = ''
        el_info = entry.find(f'{ns}ParticipantInfo')
        NameP = el_info.attrib['NameP']
        Tnp = el_info.attrib['Tnp']
        Nnp = el_info.attrib['Nnp']
        el_acc = entry.findall(f'{ns}Accounts')
        hasAccounts = len(el_acc) > 0
        if hasAccounts:
            AccountCBRBIC = el_acc[0].attrib['AccountCBRBIC']
            for entry in
xml.findall(f'{ns}BICDirectoryEntry/[@BIC="{AccountCBRBIC}"]'):
                el_info = entry.find(f'{ns}ParticipantInfo')
                NameCBR = el_info.attrib['NameP']
```

```
    if hasAccounts:
        a['BankName'] = (f'{NameCBR}//{NameP}, {Tnp} {Nnp}')
    else:
        a['BankName'] = (f'{NameP}, {Tnp} {Nnp}')

os.remove(bikFile)
os.remove(xmlfile)

print(f'Прошло времени: {round(time.time()-ts_start,1)} сек.')
```

Шаг 4. Сохранение данных

4.1. Сохраняем в JSON Справочник казначейских счетов

[7]:

```
f = open('ks.json', 'w', encoding='utf-8')
f.write(json.dumps(newdata, ensure_ascii=False))
f.close()
```

4.2. Сохраняем в JSON Справочник ЕКС

[8]:

```
f = open('trsa.json', 'w', encoding='utf-8')
f.write(json.dumps(trsa, ensure_ascii=False))
f.close()
```

4.3. Сохраняем в CSV Справочник казначейских счетов

[9]:

```
f = open('ks.csv', 'w', newline='', encoding='utf-8')
csv_writer = csv.writer(f)
count = 0

for d in newdata:
    if count == 0:
        header = d.keys()
        csv_writer.writerow(header)
        count += 1
    csv_writer.writerow(d.values())

f.close()
```

4.4. Сохраняем в CSV Справочник ЕКС

[10]:

```
f = open('trsa.csv', 'w', newline='', encoding='utf-8')
csv_writer = csv.writer(f)
count = 0

for d in trsa:
    if count == 0:
        header = d.keys()
        csv_writer.writerow(header)
        count += 1
    csv_writer.writerow(d.values())

f.close()
```

[11]:

```
print('Завершено:
{}'.format(datetime.fromtimestamp(time.time()).strftime('%d.%m.%Y
%H:%M:%S')))
print(f'Процесс занял: {round(time.time()-ts_start,1)} сек.')
```