Developer Guide. Cbonds API

> December 2025

Definitions

Web service: A web service is a software system accessible via a URL. Its interface is described in a machine-readable format, enabling other software systems to discover it and interact with it by exchanging messages over standard Internet protocols. Cbonds API web service interfaces are provided in XML and JSON.

WSDL (Web Services Description Language): WSDL is an XML-based language used to describe the structure and behaviour of web service operations, including endpoints, request/response formats, and communication protocols.

SOAP (Simple Object Access Protocol): SOAP is a messaging protocol that uses XML to structure request and response messages exchanged between a client and a web service.

XML (Extensible Markup Language): XML is an extensible markup language used to represent structured data for storage, transmission, and processing.

JSON (JavaScript Object Notation): JSON is a lightweight, text-based data format used for transmitting structured data, commonly used in web APIs.

Web service operation/method/endpoint (hereinafter referred to as endpoint): An endpoint is a callable function of the web service that returns data to the client in either WSDL+SOAP or JSON format.

Endpoint catalogue https://cbonds.com/api/catalog/folders/

User: A user is a client of the Cbonds API service who is authorised with a login and password and has access to specific datasets.

Getting data from web service

User authorisation

In order to prevent unauthorised access to data for all endpoints described below, you must specify the username/password assigned to the service for the user.

To get acquainted with the technical aspect of the web service, you can use access of a preconfigured user (i.e. test user):

Login: Test

Password: Test

The test user has access to a limited subset of data, which does not reflect the full scope or depth of the complete database.

Getting the service scheme

An XSD/XML schema or JSON schema can be obtained by a GET request with a user's username and password specified in the URL:

Schema	Request URL
XSD/XML	https://ws.cbonds.info/services/wsdl?login=LOGIN&password=PASSWORD
JSON	https://ws.cbonds.info/services/json?login=LOGIN&password=PASSWORD

You can also get the schema using the POST request by specifying the username and password in the body of the POST request (not in the URL).

Two parameters in the schema that indicate the possibility of sorting and filtering:

sortable - indicates the possibility of sorting by field (1 - sorting is possible, 0 - not possible);

searchable – indicates the possibility of filtering by field (1 - filtration is possible, 0 - not possible).

Data request

Interaction with a web service for receiving data is carried out by sending POST requests of the HTTPS protocol. Depending on your preferred data, you should send queries to the following URLs:

Data format	Request URL
XSD/XML	https://ws.cbonds.info/services/wsdl/(endpoint name)*/?lang=eng
JSON	https://ws.cbonds.info/services/json/(endpoint name)*/?lang=eng

* For example: https://ws.cbonds.info/services/json/get_emissions
For the operation list, please refer to endpoint catalogue https://cbonds.com/api/catalog/folders/

The request body contains the request parameters. Request body structure in JSON format:

```
{
"auth":{"login":"LOGIN","password":"PASSWORD"},
"filters":[{"field":"...","operator":"...","value":...}, {}],
```

```
"quantity":{"limit":10,"offset":0},
"sorting":[{"field":"...","order":"..."}],
"fields":[{"field": "..."}, {"field": "..."}]
}
```

Where the main request parameters are:

“**auth”** - contains the user's login and password;

“**filters”** - contains the specified selection conditions;

“**quantity”** - contains parameters {"limit":..., "offset":...}

"limit" - number of records per page for a single request;

"offset" - page number. It must be a multiple of the limit, e.g. with a limit of 1000 offset can be 0, 1000, 2000, etc.

“**sorting”** - contains a list of fields for sorting;

“**fields”** - contains a list of fields to display in response.

Request body structure in WSDL+SOAP format:

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SOAP-ENV:Body
    <m:RequestMessage_emissions xmlns:m="https://ws.cbonds.info/">

      <m:AuthData>
        <m:Login><![CDATA[LOGIN]]></m:Login>
        <m:Password><![CDATA[PASSWORD]]></m:Password>
      </m:AuthData>

      <m:Filters>
        <m:NumericFilter>
          <m:FilterField>....</m:FilterField>
          <m:FilterOperator>...</m:FilterOperator>
          <m:FilterValue><![CDATA[...]]></m:FilterValue>
        </m:NumericFilter>
      </m:Filters>

      <m:QuantityData>
        <m:Limit>10</m:Limit>
        <m:Offset>0</m:Offset>
      </m:QuantityData>
```

```
        <m:SortRules>
          <m:SortRule>
            <m:SortField></m:SortField>
            <m:SortOrder>...</m:SortOrder>
          </m:SortRule>
        </m:SortRules>

        <m:ResponseItemFields>
          <m:ResponseItemField>
            <m:ItemField>...</m:ItemField>
          </m:ResponseItemField>
        </m:ResponseItemFields>

      </m:RequestMessage_emissions>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Key response parameters

Response structure in JSON format:

```
{
"count":4,
"total":4,
"limit":10,
"offset":0,
"exec_time":0.4443,
"items":[...],
"meta":{ "cms_full_gen_time":0.4444, "user_id":33 }
}
```

Where the response parameters are:

"count" - the number of records found for the current page (for the last page, it may not be equal to limit);

"total" - the number of records found. To build pagination (a sequential request plan for selecting all the data in parts), we focus on the given field in the response (divide it by the limit and get the last page number in the whole part, if we count from zero). For example, if the limit is 1000 and the value total=2500, you will need to perform three queries with offsets 0*1000=0, 1*1000=1000, 2*1000=2000;

"limit" - takes the same value as at the request;

"offset" - the offset value that is not a multiple of the limit is reduced to a multiple to the lower side;

"exec_time" - this parameter is required for optimisation. If there are problems with the performance of the service (and there are opportunities), you can organise logging of this parameter for each operation, in the ideal case - statistics with aggregated values for the operation, filters, offset, etc.

"items" - sends all records for the corresponding page;

"cms_full_gen_time"- is used for optimisation;

"user_id" – your user identifier in our system, for the test user user_id = 33.

Response structure in WSDL+SOAP format:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="https://ws.cbonds.info/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <SOAP-ENV:Body>
      <ns1:ResponseMessage_emitents>
         <ns1:Count>4</ns1:Count>
         <ns1:Total>4</ns1:Total>
         <ns1:Limit>10</ns1:Limit>
         <ns1:Offset>0</ns1:Offset>
         <ns1:ExecTime>0.4139</ns1:ExecTime>
            <ns1:Items><ns1:Item>
               ...
            </ns1:Item></ns1:Items>
      </ns1:ResponseMessage_emitents>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

<!-- user_id: 33 -->
<!-- cms_full_gen_time: 0.4145 -->
<!-- lang: eng -->
<!-- strict_mode: 1 -->
<!-- performLogging: -1 -->
```

Data request using filters

When requesting data, you can create filters selecting only certain information that meets the specified criteria from all the data. **The list of fields for which a filter can be created is limited.**

List of possible operators for data filtering:

Operator	Decoding the operator	Operator value

'eq'	'equal'	**Equal to**
'ne'	'not equal'	**Not equal to**
'lt'	'less'	**Less than**
'le'	'less or equal'	**Less than or equal to**
'gt'	'greater'	**Greater than**
'ge'	'greater or equal'	**Greater than or equal to**
'in'	'is in'	**Included in the list**
'ni'	'is not in'	**Not included in the list**
'bw'	'begins with'	**Begins with (string prefix)**
'bn'	'does not begin with'	**Does not begin with (string prefix)**
'ew'	'ends with'	**Ends with (string suffix)**
'en'	'does not end with'	**Does not end with (string suffix)**
'cn'	'contains'	**Contains substring**
'nc'	'does not contain'	**Does not contain substring**
'nu'	'is null',	**Value is NULL**
'nn'	'is not null'	**Value is not NULL**

Samples

Examples of requests that can be applied to the *get_emissions* endpoint (issues static data).

The request will be sent to the address https://ws.cbonds.info/services/json/get_emissions/?lang=eng

1. Select an issue with a specific ISIN - US037833EL06

Request in JSON format:

```
{
"auth":{"login":"LOGIN","password":"PASSWORD"},
"filters":[{"field":"isin_code","operator":"eq","value":"US037833EL06"}],
"quantity":{"limit":10,"offset":0},
"sorting":[{"field":"id","order":"asc"}]
"fields":[{"field": "id"}, {"field": "maturity_date"}]
}
```

Request in WSDL+SOAP format:

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <SOAP-ENV:Body>
```

```
<m:RequestMessage_emissions xmlns:m="https://ws.cbonds.info/">
 <m:AuthData>
  <m:Login>LOGIN</m:Login>
  <m:Password>PASSWORD</m:Password>
 </m:AuthData>
 <m:Filters>
  <m:NumericFilter>
   <m:FilterField>isin_code</m:FilterField>
   <m:FilterOperator>eq</m:FilterOperator>
   <m:FilterValue><![CDATA[US037833EL06]]></m:FilterValue>
  </m:NumericFilter>
 </m:Filters>
 <m:QuantityData>
  <m:Limit>10</m:Limit>
  <m:Offset>0</m:Offset>
 </m:QuantityData>
 <m:SortRules>
  <m:SortRule>
   <m:SortField>id</m:SortField>
   <m:SortOrder>asc</m:SortOrder>
  </m:SortRule>
 </m:SortRules>
 <m:ResponseItemFields>
  <m:ResponseItemField>
   <m:ItemField>id</m:ItemField>
  </m:ResponseItemField>
  <m:ResponseItemField>
   <m:ItemField>maturity_date</m:ItemField>
  </m:ResponseItemField>
 </m:ResponseItemFields>
</m:RequestMessage_emissions>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Further, for brevity, all examples will be given only in JSON format.

2. Select multiple issues with specific ISINs, e.g. US037833EL06 and USU5876JAM72.

Request in JSON format:

```
{
"auth":{"login":"LOGIN","password":"PASSWORD"},
"filters":[{"field":"isin_code","operator":"in","value":"US037833EL06;USU5876JAM72"}],
"quantity":{"limit":10,"offset":0},
"sorting":[{"field":"","order":"asc"}],
```

```
"fields":[{"field": "id"}, {"fields": "maturity_date"}]
}
```

3. Select issues information on which were updated after August 21, 2025

Request in JSON format:

```
{
"auth":{"login":"LOGIN","password":"PASSWORD"},
"filters":[{"field":"updating_date","operator":"ge","value":"2025-08-21"}],
"quantity":{"limit":10,"offset":0},
"sorting":[{"field":"","order":"asc"}]
"fields":[]
}
```

4. Select issues information on which were updated between August 1 and August 21, 2025

Request in JSON format:

```
{
"auth":{"login":"LOGIN","password":"PASSWORD"},
"filters":[{"field":"updating_date","operator":"ge","value":"2025-08-01"},
                {"field":"updating_date","operator":"le","value":"2025-08-21"}],
"quantity":{"limit":10,"offset":0},
"sorting":[{"field":"","order":"asc"}]
"fields":[]
}
```

Limits and errors

Request volume

Maximum number of records in the response (items in the items field): 1000

To get all the data, you must run multiple queries by changing the **offset.** Each response contains information about the total number of records (the “total” field) that are available with the filters transferred in the request.

Data update frequency

Data in all endpoints, except for the *get_tradings_stocks_full_realtime_new* and *get_tradings_realtime* endpoints (intraday quotes), is updated 1 time per hour.

Queries frequency

Period	Limit
Allowed number of requests per endpoint **per minute**	No more than 30 times
Allowed number of requests per endpoint **per day**	No more than 10,000 times

Restrictions on the number of records

To ensure stable and uninterrupted work of the web service, a limit is introduced on the number of records in the result set.

The default maximum number of records is 1,000,000. If this value is exceeded, an error message with status 500100 will be broadcast.

List of possible errors in the web service response

1. HTTP Errors
    a. **HTTP/1.1 301 Moved Permanently** — The requested resource has been permanently moved to a new URL
    b. **HTTP/1.1 403 Forbidden** — Authentication or authorisation error:
    - Incorrect login or password;
    - Missing login/password or missing authorisation header;
    - The maximum allowed number of records in the response has been exceeded (see section *4. Result Set Size Exceeded*).
    c. **HTTP/1.1 500 Internal Service Error** — Internal service problem, such as:
    - Configuration or syntax error,
    - Request to an endpoint to which the user has no access.
    d. **HTTP/1.1 504 –** The service did not respond within the allowed time.
2. Application-Level Errors
    a. **https protocol required** — The service must be accessed via **HTTPS**.
    b. **Max requests per minute limit max_per_minute exceeded. Try to request the next minute.** — The per-minute request limit was exceeded.
    c. **Max requests per day limit max_per_day exceeded. Try to request next day.** — The daily request limit was exceeded.
3. Error Messages Returned by the Service:
    a. Undefined index: xxxx
    b. Invalid field (xxxx) for filtering
    c. The request JSON string is not valid. The collection "%services_xxxxxx%" is not available for user id "yy"

4. Result Set Size Exceeded:

   If the number of records returned exceeds the maximum allowed value, the service returns the following message:
   "You exceeded the maximum number of records in the result set total size. Please use filtering in your request or refine existing filters to reduce the volume of results. The number of records in the result set is n, the maximum possible number is 1000000."

Demo version of the web service

The demo version serves as an assistant for developers. The tool contains the same functionality as the full version and differs only in the presence of a visual interface.

To start working with the demo version of the web service, click one of the links:

1. https://ws.cbonds.info/services/json/demo/
2. https://ws.cbonds.info/services/wsdl/demo/

Below is a description of working only with the demo version for JSON, working with the demo version for WSDL is similar.

1. Enter username and password for access (or test user credentials) and click the "Request service schema"

2. After requesting the service schema, the "Request" and "Response" windows will be filled in.

The "Response" window contains the service schema, in this example, a JSON schema.

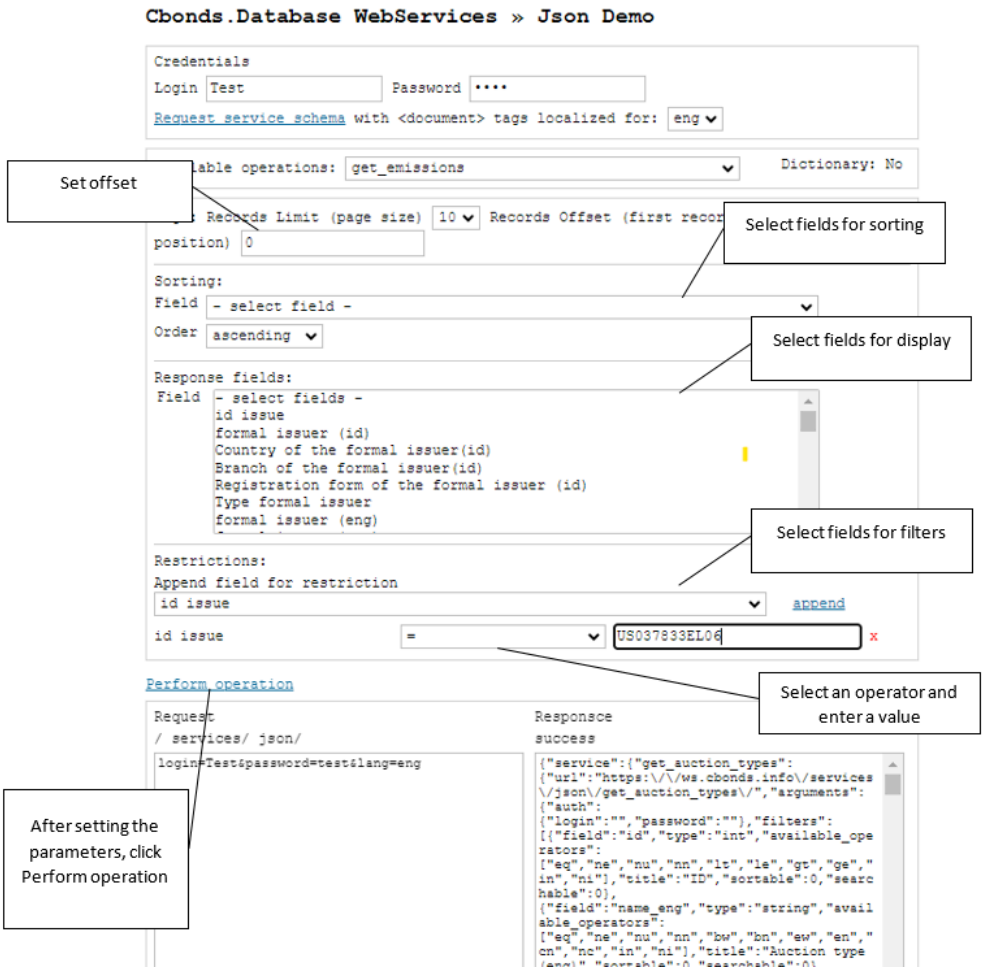
3. Select one of the available endpoints from the "Available operation" drop-down list. For example, let's select the "get_emissions" endpoint.

4. After selecting the endpoint, the following request parameter settings will be available:

- Select the number of records to send;
- Sort by a field in received responses;

- Filters on the data to be sent.

These options can be applied to all endpoints except reference endpoints. To work with filters, select a field, click the "append" button, select an operator and a value.



5. After the request is completed, the body of the POST request is automatically generated in the "Request" window. The "Response" window will display the response in the appropriate format.

Thus, when developing a product, you can consult the data returned by the web service using a demo version.

Example of accessing a web service in Python 3.x

Below there is an example of Python 3.x code performing:

1. Data request with filtering;
2. Writing data to pandas DataFrame;
3. Saving a table to csv.

```
import requests
import pandas as pd

url_emissions = 'https://ws.cbonds.info/services/json/get_emissions/?lang=eng'

json_data_emissions = {"auth":{"login":"Test","password":"Test"},
"filters":[{"field":"isin_code","operator":"in","value":"US037833EL06;USU5876JAM72"}],
 "quantity":{"limit":10,"offset":0},"sorting":[{"field":"","order":"asc"}]}

response_emissions = requests.get(url_emissions, json=json_data_emissions).json()
print(response_emissions)

initial_dict = response_emissions['items']

df = pd.DataFrame.from_dict(initial_dict)
df

df.to_csv(index=False)
```