API RU DATA (DataHub)

v2.0

Содержание

1. Введение

RU DATA API является REST API веб-сервисом, предоставляющим доступ к информации по ценным бумагам из хранилища данных Интерфакс и НРД.

RU DATA API позволяет получить следующую информацию:

- Данные по облигациям (параметры выпуска, купоны, оферты и др.);
- Данные по акциям, депозитарным распискам, фондам и ИСУ;
- Справки по эмитентам и их ценным бумагам;
- Архивы и итоги торгов финансовыми инструментами на Московской бирже;
- Рейтинги компаний и облигаций;
- Расчетные показатели рыночного риска, краткосрочной ликвидности и другие, вычисленные в соответствии с инструкциями ЦБ.

2. Доступ к RU DATA API

Сервис RU DATA API расположен по адресу https://dh2.efir-net.ru/v2.
Для доступа к тестовым данным необходимо использовать адрес https://test-dh2.efir-net.ru/v2.

Сервис поддерживает защищенный протокол ***HTTPS***, обеспечивающий конфиденциальность пользовательских данных при обращении к сервису.
Использование незащищенного протокола ***HTTP*** не рекомендовано и может быть ограничено службами информационной безопасности (ИБ) клиента.

Доступ к данным сервиса может быть обеспечен либо программными средствами, либо с использованием упрощенного пользовательского интерфейса swagger, расположенного по адресу:
https://dh2.efir-net.ru/swagger/index.html
для доступа к тестовым данным - https://test-dh2.efir-net.ru/swagger/index.html.

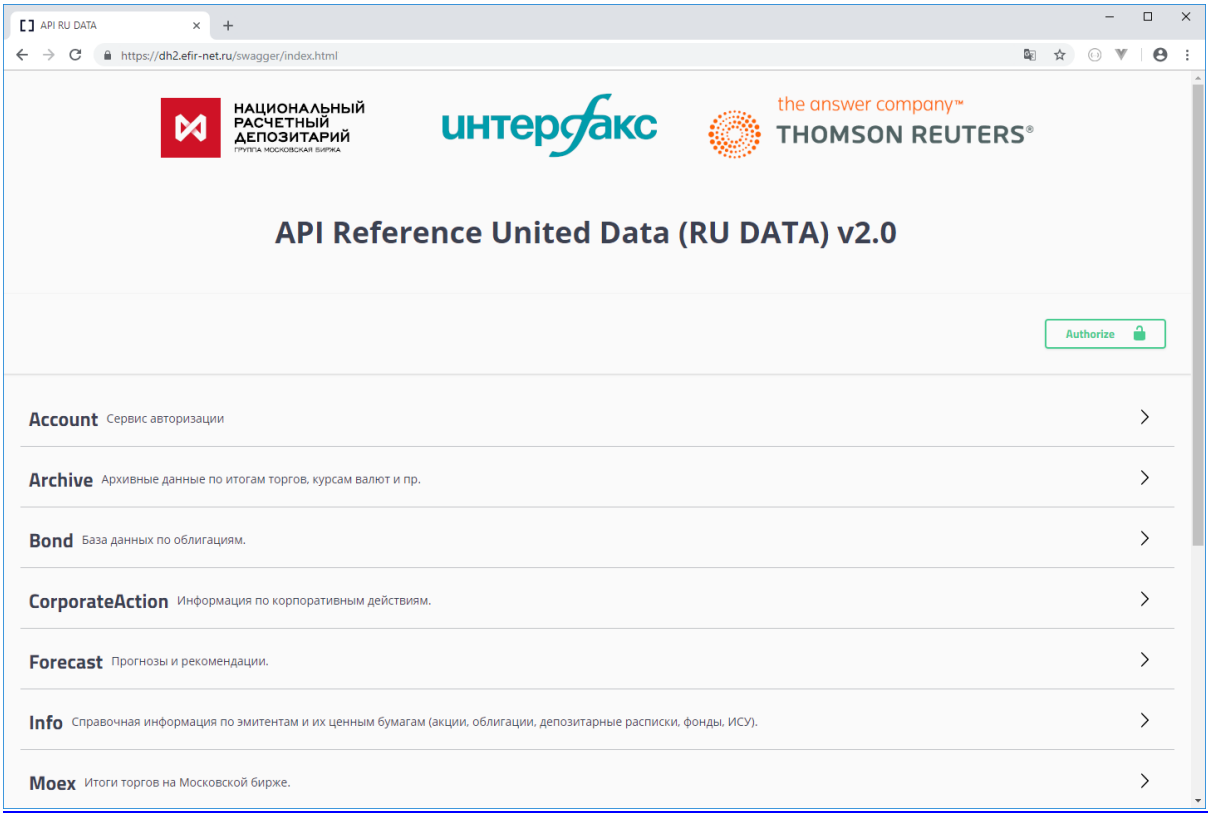


рис. 1 Интерфейс swagger

Интерфейс swagger создан для создания тестовых запросов к сервису, но он так же может быть использован и для повседневной работы с сервисом.

Сервис основан на ***HTTP REST*** протоколе данных.

При обращении к какому-либо интерфейсу сервиса необходимо сформировать HTTP запрос, содержащий необходимые заголовки и тело запроса.

Форматом данных запросов и ответов сервиса является ***application/json.***
Заголовок всех запросов (кроме запроса на авторизацию) должен содержать раздел ***Authorization***, в котором указывается метод авторизации ***Bearer*** и токен авторизации (получается при авторизации пользователя – см. раздел Авторизация).

Тело запроса представляет из себя JSON с данными запроса.

В случае успеха, в ответ на запрос возвращаются данные в формате ***JSON***.

Модели данных запросов и ответов собраны в виде библиотеки .net (***.NETFramework v4.5.2*** и ***.NETStandard v2.0***), доступной для загрузки по ссылке:

http://developer.efir-net.ru/NuGetFeed/Package/Efir.DataHub.Models

также все модели данных можно увидеть в разделе ***Models*** интерфейса swagger.

Описание полей моделей доступно в комментариях к свойствам моделей в указанной .net библиотеке либо их можно найти в интерфейсе swagger в виде комментариев на вкладке ***Model*** запроса и ответа

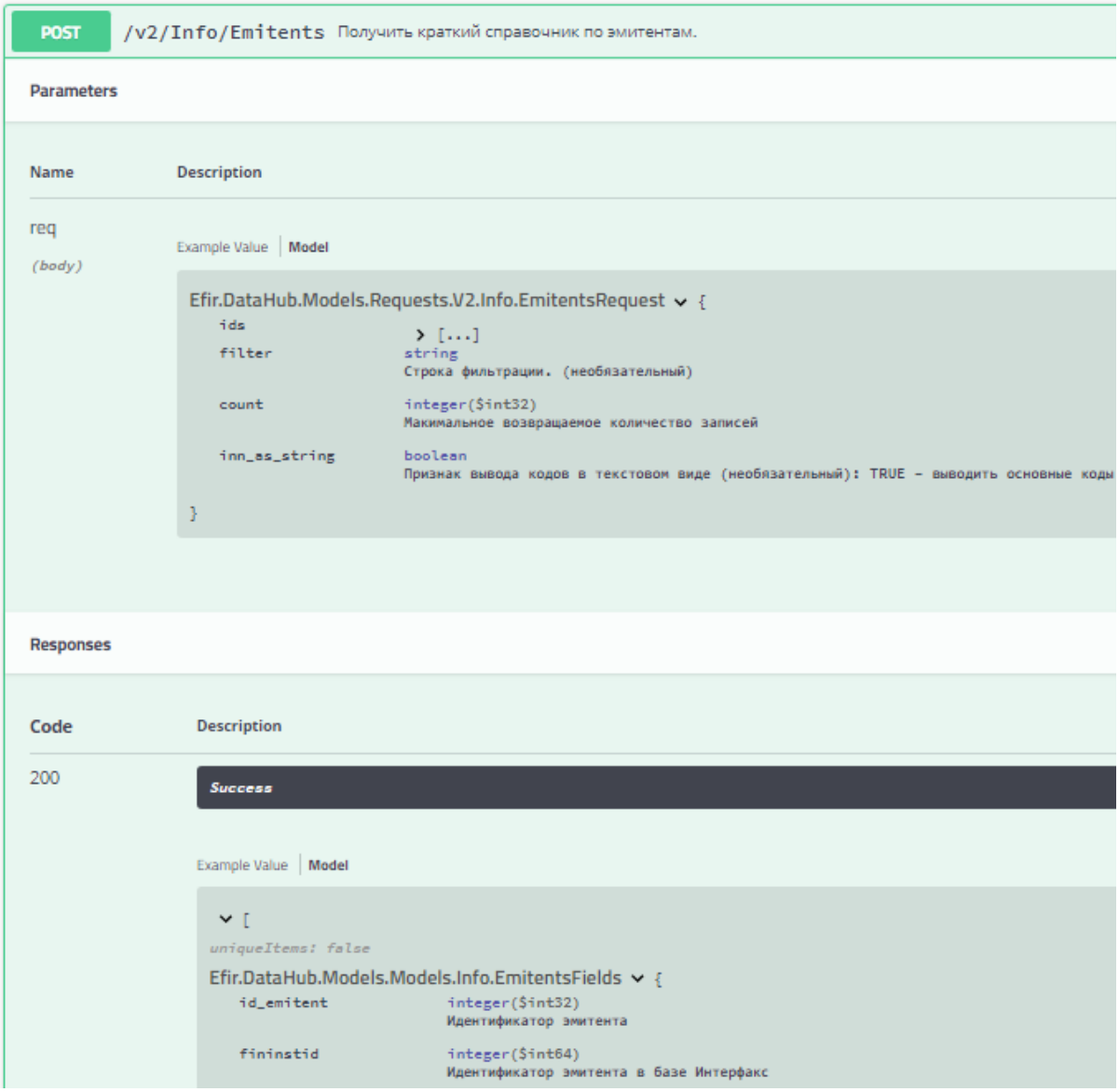


рис. 2 Описание полей в интерфейсе swagger

3. Авторизация

Для доступа к методам RU DATA API в первую очередь необходимо произвести авторизацию пользователя.

Авторизация производится с использованием логина и пароля пользователя, который необходимо получить у менеджера Интерфакс.

Для авторизации необходимо выполнить POST запрос к ресурсу https://dh2.efir-net.ru/v2/Account/Login,

В теле запроса передается информация о логине и пароле пользователя:

```
{
  "login": "login",
  "password": "password"
}
```

При успешном запросе пользователю вернется ответ с телом, содержащим авторизационный ***JWT токен***:

```
{
  "token":
        "eyJhbGciOiJ4GhoNiIsInR5cCI6IkpXVCJ9.eyJuYmYiOjE1MzkxNzY3NTEsImV4cCI6MTUzOTI2MzE1MSwiaXNzIjoiaHR0cDovL2VmaXItZG9ja2VyOjcwNzAiLCJhdWQiOlsiaHR0cDovL2VmaXItZG9ja2VyOjcwNzAvcmVzb3VyY2VzIiwiZGF0YUh1YjIiLCJuZXdzLnByb3h5Il0sImNsaWVudF9pZCI6ImRhdGFodWIiLCJzdWIiOiIyMjY0MSIsImF1dGhfdGltZSI6MTUzOTE3Njc1MSwiaW5ghyjoibG9jYWwiLCJVc2VySWQiOiIyMjY0MSIsIlNlc3Npb25JZCI6InVuaWNraWRfOylfMTc5OTQ3MjQxXzEwMTAxOCAxNjA1MzVfXyIsIlVzZXJJcEFkZHJlc3NQb3J0IjoiMTcyLjE3LjAuMTo1MjY2NiIsIkxvZ2luIjoiaWZheC1wb2RrYWx1ayIsInNjb3BlIjpbImRhdGFIdWIyIiwibmV3cy5wcm94eSJdLCJhbXIiOlsicHdkII19.3c5JhwrGl-3DRpqu5CvZ8-9WKPmo1LAasxVZi7htiD5teQXDP2nSovoo806FxyqHZ83dIvZ8OwfybOPZNNf3pEHBjDsi5lccUtGv3JY72UGFyQYHQfE7yWwTVk-G1XL_kkhyCdZ9obQV7HXpSTRtydpmhhTLYBAm1igddBRjTKgZlws2B_i1gzcIxL6YNWv61dOHFh4zeu5np5kh0I7HFE3K8gW2OU4eouoGomRQIT0J0f0kwqmB-ipGYlkTd3zYXWCjSUfzyaXc9ZgIEqvl-DvLiT0LxGsLtxBqAzjHxxudM8MuiqVJ4DOZXFyf75YNkOtzcrQFEnZfYRvz6eB3_Q"
}
```

При использовании интерфейса swagger, данные о логине и пароле необходимо ввести в окне запроса (предварительно ***нажав кнопку Try it out***)

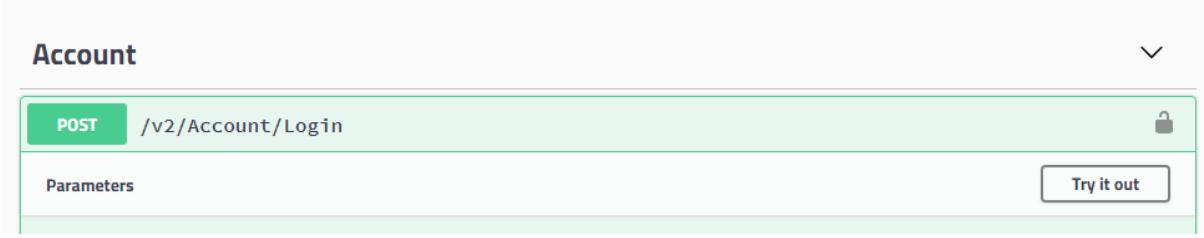


рис. 3 Авторизация в интерфейсе swagger

и далее нажав кнопку ***Execute***:

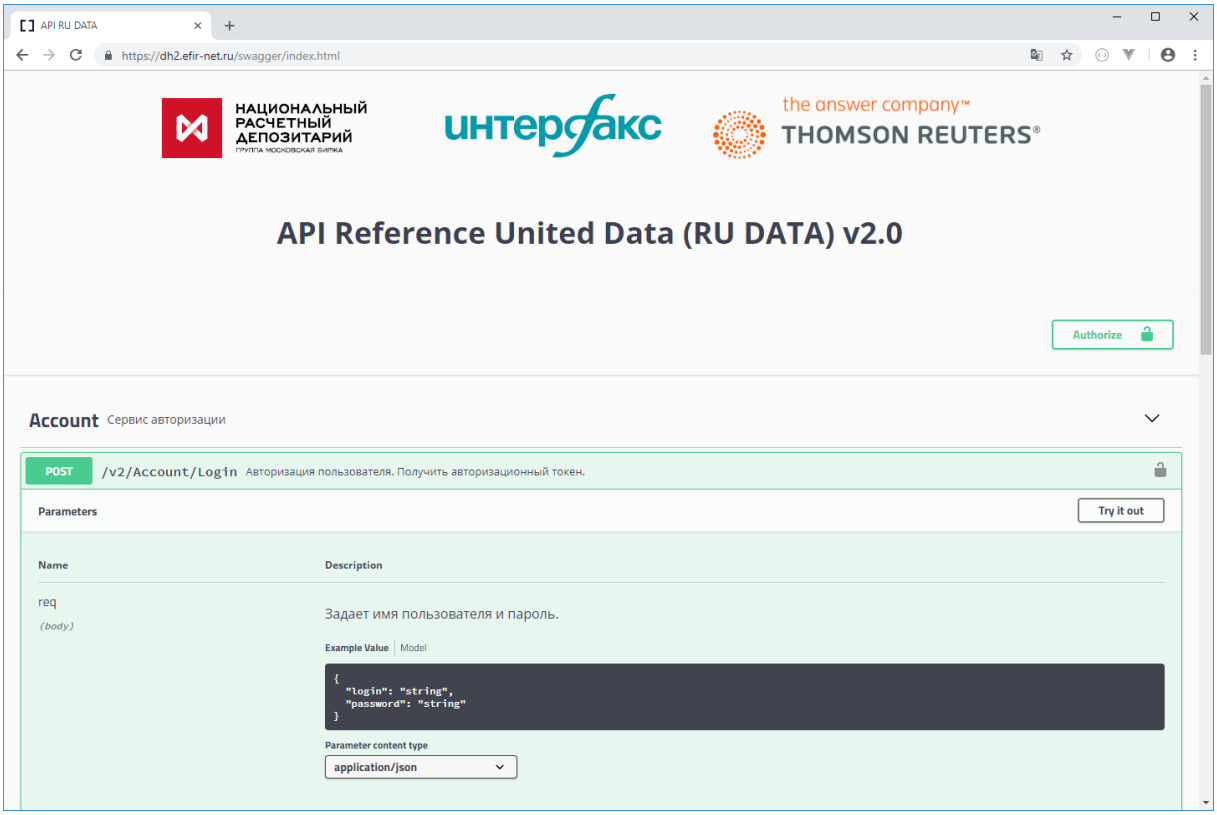


рис. 4 Заполнение полей запроса в интерфейсе swagger

Ответ на запрос будет отображен в разделе ***Responses*** в окне ***Response body***

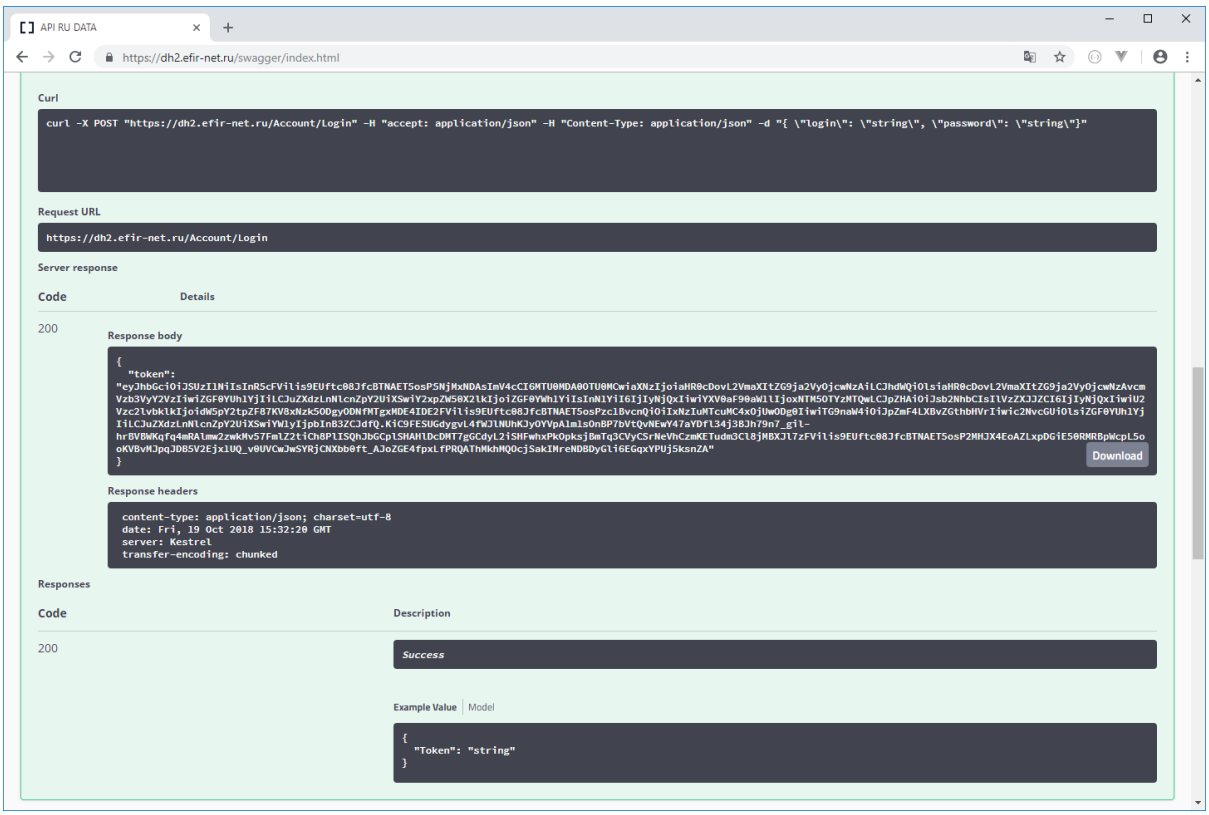
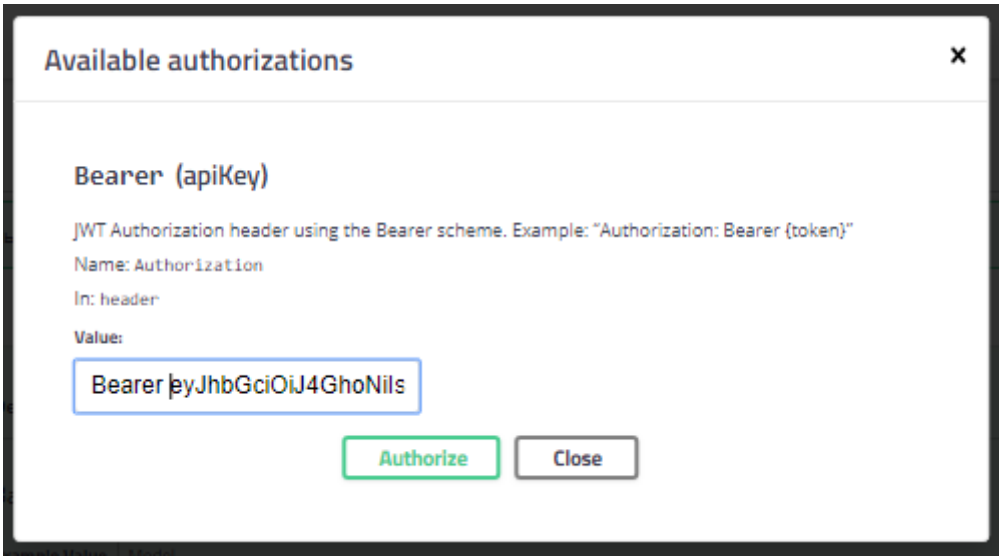


рис. 5 Отображение данных ответа на запрос в интерфейсе swagger

Полученный JWT токен должен передаваться в заголовке каждого последующего обращения к ресурсам RU DATA API.

В случай использования интерфейса swagger, полученный токен необходимо зарегистрировать, нажав кнопку ***Authorize*** (см. рис. 1), в появившемся диалоговом окне ввести в поле Value слово Bearer, пробел, скопировать полученный авторизационный токен (без кавычек) и нажать кнопку ***Authorize***



Важно: время “жизни” токена ограничено по времени (в данный момент 30 минутами), поэтому, в случае более длительной работы с сервисом, необходимо авторизоваться повторно для получения нового токена.

4. Запрос данных RU DATA API

Запрос данных сервиса RU DATA API производится при помощи действий, аналогичных описанным в разделе Авторизация.

При формировании запроса необходимо указать авторизационный токен, сформировать данные запроса, отправить запрос и получить ответ.

При формировании запросов имеется возможность использовать ряд стандартных входных параметров, которые используются во многих методах и имеют одно и то же назначение: это параметры ***filter, count***.

4.1 Параметр filter

Необязательный параметр filter в методах нужно использовать для дополнительной фильтрации получаемых данных.

В качестве параметров в фильтре используются имена выходных полей, вызываемого метода API. Список полей соответствует списку полей возвращаемых данных запроса.

Также можно использовать специальное служебное поле ***rownum*** для того, чтобы указать количество данных/записей в результатах вызова: например, установив значение фильтра ***«rownum<10»***, метод будет возвращать первые 9 найденных записей:

```
{
  "filter": "rownum=1"
}
```

При работе с полями можно применять стандартный набор операторов: ***“>”, “<”, “=”, “IN”, круглые скобки*** и т.д.

Объединение условий фильтров должно производится при помощи ключевого слова ***AND (логическое И).***

Если в параметре filter необходимо указать дату, то она должна вводиться в формате «round-trip» (в C# это формат "O" или "o") и обрамляться знаком решётки #, например, фильтр запроса данных измененных после 2 сентября 2015 года:

```
{
  "filter": "UPDATE_DATE > #2015-09-02T00:00:00.0000000Z#"
}
```

также может использоваться формат даты:

```
{
  "filter": "UPDATE_DATE > #2018-08-30#"
}
```

Пример объединения нескольких фильтров

```
{
  "filter": "UPDATE_DATE > #2018-08-30# AND rownum=1"
}
```

4.2 Параметр count

Многие методы API имеют входной параметр “***count***”, который задает максимальное количество возвращаемых методом записей. Сейчас, если это значение не задано, то будет использоваться значение по умолчанию – 1000 записей. Этот параметр дублирует функции специального служебного поля ”rownum” в параметре filter.

Пример получения рейтингов по облигации с ISIN RU000A0JQXG0 на 01.11.2014 с использованием метода */Rating/SecurityRatings* с параметром count:

```
{
  "isin": "RU000A0JQXG0",
  "date": "2014-11-01",
  "count": 5
}
```

4.3 Параметры запросов типа дата

Параметры с типом данных Дата в описании метода обозначаются как ***string($date-time)***.
Формат строкового преставления даты должен соответствовать правилу, приведенному в описании параметра filter.

Например, для получения рейтингов по облигации с ISIN RU000A0JQXG0 на 01.11.2014 можно использовать метод */Rating/SecurityRatings*, задав параметры:

```
{
  "isin": "RU000A0JQXG0",
  "date": "2014-11-01T00:00:00.0000000Z"
}
```

либо с коротким форматом даты:

```
{
  "isin": "RU000A0JQXG0",
  "date": "2014-11-01"
}
```

5. Пример кода вызова метода RU DATA API на языке C#

Ниже приведен исходный текст программы на языке C# для вызова метода API ***/info/fintoolReferenceData***

```
using System;
using System.Text;
using System.Collections.Generic;
using System.Threading.Tasks;
using System.Net.Http;
using System.Net.Http.Headers;

using Newtonsoft.Json;

using Efir.DataHub.Models.Models.Info;
using Efir.DataHub.Models.Requests.V2.Info;
using Efir.DataHub.Models.Models.Account;
using Efir.DataHub.Models.Requests.V2.Account;

namespace ApiExample
{
	public class Dh2Example
	{
		private string Url { get; }
		private string Login { get; }
		private string Password { get; }

		public static async Task DoTest()
		{
			var example = new Dh2Example("https://dh2.efir-net.ru/v2", "xxx", "yyy");
			var token = await example.LoginAsync();

			Console.WriteLine($"Token: {token}");

			var data = await example.GetBondsByEmitentAsync(6496, token);

			Console.WriteLine($"Data: {JsonConvert.SerializeObject(data)}");
		}

		public Dh2Example(string url, string login, string password)
		{
			Url = url;
			Login = login;
			Password = password;
		}

		public async Task<string> LoginAsync()
		{
			var query = new LoginRequest
			{
				login = Login,
				password = Password
			};
			var res = await ApiPostRequestAsync<LoginRequest, LoginResponse>($"{Url}/account/login", query);

			return res?.Token;
		}

		public async Task<List<FintoolReferenceDataFields>> GetBondsByEmitentAsync(long issuerId, string token)
		{
			var query = new FintoolReferenceDataRequest
			{
				filter = $"ISSUERUID={issuerId} AND FINTOOLTYPE IN ('Облигация')"
			};
			return await ApiPostRequestAsync<FintoolReferenceDataRequest, List<FintoolReferenceDataFields>>(
				$"{Url}/info/fintoolReferenceData",
				query,
				token);
		}

		private static async Task<Response> ApiPostRequestAsync<Request, Response>(string url, Request request, string token = null)
		{
			var client = new HttpClient();
			client.DefaultRequestHeaders.Add("Accept", "application/json");
			if (!string.IsNullOrEmpty(token))
```

```
                client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", token);

            var jsonData = JsonConvert.SerializeObject(request);
            var response = await client.PostAsync(url, new StringContent(jsonData, Encoding.UTF8, "application/json"));

            if (!response.IsSuccessStatusCode) //error
                throw new System.Exception("api error");

            return await response.Content.ReadAsAsync<Response>();
        }

    }
}
```