Version 2025.2.1

FastReport Corporate Server Programmer's Guide

Introduction

Any code snippets in this documentation are demos only. These examples serve no other purpose than to illustrate the use of FastReport Corporate Server. Fast Reports Inc. does not warrant the accuracy or completeness of these examples and shall not be liable for any direct or indirect damages that may result from them.

The documentation may contain hyperlinks to various Internet resources. Such links are relevant at the time of publishing the documentation. Fast Reports Inc. shall not be liable for their availability at the time of reading the document or any damage caused by them.

Guides

This section is dedicated to guides for working and integrating FastReport Corporate Server into the application. At the moment the guides are available for:

- REST API
- C#/.NET

The guides for use with other programming languages are in the plans.

Contact us if you are interested in a specific programming language or platform. Contact!

Development tools

The SDKs are available for the following languages:

- Haskell
- JavaScript
- C++
- Python
- Java
- Go
- C#/.NET and ASP.NET

Please note, that the development tools are posted as sources on GitHub. To use them, you will need to create a package or library from the sources yourself.

REST API

This section provides step-by-step instructions and guidelines for performing typical tasks for using FastReport Corporate Server without being bound to a specific programming language.

Getting started

You will need the following tools and facilities:

1. Curl tool.

   Any other REST client will work, but the examples will be built for curl.

2. FastReport Community Designer.

   In some guides, you will need the designer to create the report.

3. Active FastReport Corporate Server subscription.

4. Internet access.

> Please note: These guides assume that you already know how to use curl or another REST client.

> Note. The items above describe the recommended tools.

What next?

We advise you to start reading the following articles:

- Status codes.
- Authentication and authorization.

Status codes

This article lists all status codes that can be returned by FastReport Corporate Server services.

Request Successfully Processed — 2xx

- 200 OK - the request was processed successfully. It is used in most cases when the response body is returned along with the result.
- 201 Created - the request was processed successfully. As a result of its execution, a new entity was created.
- 204 No Content - the request was processed successfully. An empty response body is returned.

User Error — 4xx

- 400 Bad Request - the request contains errors, for example:
  - There is no parameter required to run the request.
  - The entity `Id` format is different from Hex 24.
  - The negative skip or take parameter is in the request.
- 401 Unauthorized - the request was sent by an unauthorized user.
- 402 Payment Required - the requested resource is limited by subscription plan.
- 403 Forbidden - the requested resource is found, but the user does not have permission to perform the action.
- 404 Not Found - the requested resource is not found.
- 413 Request Entity Too Large - request body exceeds the limits in the service settings.

Server Error — 5xx

- 500 Internal Server Error - the request has passed all validity checks, but an Exception occurred during its execution. In this case, please inform us by emailing info@fast-report.com

Other

Intermediate services between FastReport Corporate Server and the user may return other status codes.

Authentication and authorization

The process of user data authentication and granting user certain rights in FastReport Corporate Server is carried out in one of two available ways:

1. Via JWT token.

   In this case, authentication must be done manually, and the token will only be valid for 5 minutes, during which time the user must log in to their application. When connecting to the server, the browser will redirect to the authentication server and then generate an access token. We restrict the possibility of retrieving the JWT token to the user personally from a security point of view.

   > If the user has not logged in to the application within 5 minutes, the authentication must be re-authenticated. If the user has logged in, re-authentication is not required.

2. Via API key.

   In this case, the obtaining of access rights is performed for server applications. To get an API key, a user must be present. However, the key itself can be valid for a long time, for example, a year.

Retrieve the first API key

To get the first API key, access the user panel. If for some reason you do not have access to the user panel, you can request a key as described below.

1. Open the link in your browser: <{your_host_name}/account/signin?r= {your_host_name}/api/manage/v1/ApiKeys>.

   If you click on this link, it will direct you to the automatic browser authentication process.

2. Now when authentication has passed, you need to request a new key.

   Press `F12` or `Ctrl+Shift+I` to open the developer panel. The key combinations may differ from the default, in that case open the developer panel via the browser menu

3. Copy and run the code in the JavaScript console.

   This code will make a `POST` request to the URL `{your_host_name}/api/manage/v1/ApiKeys` to create a new access key valid till 2030.

4. Refresh the browser page and get the result.

```
{
  "apiKeys": [
    {
      "value": "cc355oeu1z5d5wncayo33me6c1g5junqdqk4pkupid7t8ynjshey",
      "description": "Generated by js develop panel",
      "expired": "2030-01-01T07:41:23.399Z"
    }
  ],
  "count": 1
}
```

You can also create a key by opening the Api keys tab.

Now you can use the API key. In the example above, `cc355oeu1z5d5wncayo33me6c1g5junqdqk4pkupid7t8ynjshey` was used.

There is no need to retrieve a new API key through the browser again.

How to use the API key

The key should be passed with each request in the header `Authorization: Basic`. Use `apikey` as the username and the key value as the password. For example:

```
Authorization: Basic Base64Encode(apikey:cc355oeu1z5d5wncayo33me6c1g5junqdqk4pkupid7t8ynjshey);
```

Where `Base64Encode` is a function for converting a string to `base64` when using `UTF8` encoding.

Retrieve new API key

To get the new key, execute the `POST` request to the `{your_host_name}/api/manage/v1/ApiKeys` entry point and pass JSON in the request body according to the scheme below.

```
{
   "description": "string",
   "expired": "string($date-time)"
}
```

Example request:

```
curl -X POST "{your_host_name}/api/manage/v1/ApiKeys" -H "accept: text/plain" -H "authorization: Basic YXBpa2V5OmNjMzU1b2V1MXo1ZDV3bmNheW8zM21lNmMxZzVqdW5xZHFrNHBrdXBpZDd0OHluanNoZXk=" -H "Content-Type: application/json-patch+json" -d "{ \"description\": \"Generated by js develop panel\", \"expired\": \"2030-01-01T07:41:23.399Z\"}"
```

Response scheme:

```
{
  "value": "string",
  "description": "string",
  "expired": "2020-12-02T08:47:43.270Z"
}
```

You can also get a new key through the user panel.

What next?

- Upload new template.
- Work with groups.
- Add new users to the workspace.

Upload new template

One of the main features of FastReport Corporate Server is storing templates, reports, and other data in the cloud. This article will describe how to upload your own template to FastReport Corporate Server template storage.

Getting started

You will need the following tools and facilities:

1. Know how to use API key in FastReport Corporate Server.

   This article will skip over additional information on authentication and authorization.

2. Curl tool.

   Any other REST client will work, but the examples will be built for curl.

3. Report template.

   It can be created in the free FastReport Community Designer program.

4. Active FastReport Corporate Server subscription.

5. Internet access.

Instruction

1. You need to get the root folder ID of the workspace. This identifier will never change, so you can save it and you do not need to request the ID every time before uploading a new template.

   To request the root directory, execute the `GET` request to `{your_host_name}/api/rp/v1/Templates/Root`.

   ```
   curl -X GET "{your_host_name}/api/rp/v1/Templates/Root" -H "accept: text/plain"
   ```

   In this case, the directory for the default workspace will be returned. You can specify a particular workspace by passing the `subscriptionId` parameter.

   ```
   curl -X GET "{your_host_name}/api/rp/v1/Templates/Root?subscriptionId=your_workspace_id" -H "accept: text/plain"
   ```

   Example answer:

   ```
   {
    "name": "RootFolder",
    "parentId": null,
    "tags": [],
    "icon": "",
    "type": "Folder",
    "size": 16384,
    "subscriptionId": "5fa919fa292a8300019349bc",
    "status": "None",
    "id": "5fa919f9292a8300019349b9",
    "createdTime": "2020-11-09T10:29:13.928Z",
    "creatorUserId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491",
    "editedTime": "2020-11-13T15:58:45.69Z",
    "editorUserId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491"
   }
   ```

   Directory ID from the example above: `5fa919f9292a8300019349b9`.

2. To upload the desired template, execute the `POST` request to `{your_host_name}/api/rp/v1/Templates/Folder/{id}/File`, where the directory ID should be substituted for `{id}` and JSON should be passed in the request body according to the scheme below.

```
{
 "name": "string",
 "content": "base64encoded(template.frx)"
}
```

   - `name` — name of the file to be displayed in the user panel.

     Add the `.frx` extension to the file name manually.

   - `content` — file content encoded in base64.

   Example request:

```
curl -X POST "{your_host_name}/api/rp/v1/Templates/Folder/5fa919f9292a8300019349b9/File" -H "accept: text/plain" -H "Content-Type: application/json-patch+json" -d "{ \"name\": \"template.frx\", \"content\": \"77u/PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0idXRmLTgiPz4NCjxSZXBvcnQgU2NyaXB0TGFuZ3VhZ2U9IkNTaGFycCIgUmVwb3J0SW5mby5DcmVhdGVkPSIxMi8wNC8yMDIwIDEwOjU4OjU3IiBSZXBvcnRJbmZvLk1vZGlmaWVkPSIxMi8wNC8yMDIwIDExOjAwOjIwIiBSZXBvcnRJbmZvLkNyZWF0b3JWZXJzaW9uPSIyMC4yMC4wLjEiPg0KICA8RGljdGlvbmFyeS8+DQogIDxSZXBvcnRQYWdlIE5hbWU9IlBhZ2UxIiBXYXRlcm1hcmsuRm9udD0iQXJpYWwsIDYwcHQiPg0KICAgIDxDcm9zc0JhbmQgTmFtZT0iUmVwb3J0VGl0bGUxIiBXaWR0aD0iNzE4LjIiIEhlaWdodD0iMzcuOCIvPg0KICAgIDxQYWdlSGVhZGVyQmFuZCBOYW1lPSJQYWdlIEhlYWRlcjEiIFRvcD0iNDIuOCIgV2lkdGg9IjcxOC4yIiBIZWlnaHQ9IjI4LjM1Ii8+DQogICAgPERhdGFCYW5kIE5hbWU9IkRhdGExIiBUb3A9IjcyLjU1IiBXaWR0aD0iNzEuNiIvPg0KICAgIA8VGV4dE9iamVjdCBOYW1lPSJUZXh0MSIgV2lkdGg9Ijc1LjYiIFRleHQ9IkhlbGxvLCBGYXN0UmVwb3J0IENsb3VkISEhIiBIb3J6QWxpZ249IkNlbnRlciIgVmVydEFsaWduPSJDZW50ZXIiIEZvbnQ9IkFyaWFsLCAxMHB0Ii8+DQogICAgPC9EYXRhQmFuZD4NCiAgICA8UGFnZUZvb3RlckJhbmQgTmFtZT0iUGFnZUZvb3RlckEiIEhlaWdodD0iMTguNSIvPg0KICA8L1JlcG9ydFBhZ2U+DQo8L1JlcG9ydD4=\"}"
```

   Example answer:

```
{
 "reportInfo": {
   "author": null,
   "created": "2020-12-04T10:58:57",
   "creatorVersion": "20.20.4.1",
   "description": null,
   "modified": "2020-12-04T11:00:20",
   "name": null,
   "picture": null,
   "previewPictureRatio": 0,
   "saveMode": "All",
   "savePreviewPicture": false,
   "tag": null,
   "version": null
 },
 "name": "template.frx",
 "parentId": "5fa919f9292a8300019349b9",
 "tags": null,
 "icon": null,
 "type": "File",
 "size": 17159,
 "subscriptionId": "5fa919fa292a8300019349bc",
 "status": "Success",
 "id": "5fc9ece6b792c90001d94b13",
 "createdTime": "2020-12-04T08:01:42.7087229+00:00",
 "creatorUserId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491",
 "editedTime": "2020-12-04T08:01:42.7087112+00:00",
 "editorUserId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491"
}
```

3. Now this template can be used to prepare (create) a report and export it.

What next?

- Access rights management by template example.
- Create report.

Add JSON data source

This article will cover how to create a new JSON-based data source in FastReport Corporate Server.

Getting started

You will need the following tools and facilities:

1. Know how to use API key in FastReport Corporate Server.

   This article will skip over additional information on authentication and authorization.

2. Curl tool.

   Any other REST client will work, but the examples will be built for curl.

3. JSON file.

4. JSON scheme.

5. Active FastReport Corporate Server subscription.

6. Internet access.

Create a data source

To create a data source, send `CREATE` request to `{your_host_name}/api/data/v1/DataSources` .

Example request body:

```
{
  "name": "fakeAPI",
  "connectionString":
"Json=aHR0cHM6Ly9qc29ucGxhY2Vob2xkZXIudHlwaWNvZGUuY29tL3Bvc3Rz;JsonSchema=eyJ0eXBlIjoiYXJyYXkiLCJpdGVtcyI
6eyJ0eXBlIjoib2JqZWN0IiwicHJvcGVydGllcyI6eyJ1c2VySWQiOnsidHlwZSI6Im51bWJlciJ9LCJpZCI6eyJ0eXBlIjoibnVtYmVyIn0sInR
pdGxlIjp7InR5cGUiOiJzdHJpbmcifSwiYm9keSI6eyJ0eXBlIjoic3RyaW5nIn19fX0=;Encoding=utf-8",
  "subscriptionId": "604f52e1261a3c19104c0e25",
  "connectionType": "JSON"
}
```

Where

- `name` — data source name (it will be displayed in the designer when selected).
- `connectionString` — connection string, in case of JSON, it consists of 3 elements:
  - `Json` — JSON file or http/https link encoded in base64;
  - `JsonSchema` — schema describing JSON file structure encoded in base64;
  - `Encoding` — encoding, it should always be passed to dfg `utf-8`.
- `subscriptionId` — ID of the workspace (subscription) to which the data source will be attached.
- `connectionType` — connection type, this guide uses `JSON`.

Example request:

```
curl -X POST "{your_host_name}/api/data/v1/DataSources" -H "accept: text/plain" -H "Content-Type: application/json-
patch+json" -d "{ \"name\": \"fakeAPI\", \"connectionString\":
\"Json=aHR0cHM6Ly9qc29ucGxhY2Vob2xkZXIudHlwaWNvZGUuY29tL3Bvc3Rz;JsonSchema=eyJ0eXBlIjoiYXJyYXkiLCJpdGVtcy
I6eyJ0eXBlIjoib2JqZWN0IiwicHJvcGVydGllcyI6eyJ1c2VySWQiOnsidHlwZSI6Im51bWJlciJ9LCJpZCI6eyJ0eXBlIjoibnVtYmVyIn0sInR
pdGxlIjp7InR5cGUiOiJzdHJpbmcifSwiYm9keSI6eyJ0eXBlIjoic3RyaW5nIn19fX0=;Encoding=utf-8\", \"subscriptionId\":
\"604f52e1261a3c19104c0e25\", \"connectionType\": \"JSON\"}"
```

Example answer:


```
{
  "id": "60648953db44d83f9c6da98f",
  "name": "fakeAPI",
  "connectionType": "JSON",
  "connectionString":
"Json=aHR0cHM6Ly9qc29ucGxhY2Vob2xkZXIudHlwaWNvZGUuY29tL3Bvc3Rz;JsonSchema=eyJ0eXBlIjoiYXJyYXkiLCJpdGVtcyI
6eyJ0eXBlIjoib2JqZWN0IiwicHJvcGVydGllcyI6eyJ1c2VySWQiOnsidHlwZSI6Im51bWJlciJ9LCJpZCI6eyJ0eXBlIjoibnVtYmVyIn0sInR
pdGxlIjp7InR5cGUiOiJzdHJpbmcifSwiYm9keSI6eyJ0eXBlIjoic3RyaW5nIn19fX0=;Encoding=utf-8",
  "dataStructure": "<JsonDataSourceConnection
ConnectionString=\"rijcmlqrcq6OJBTPt0pNFvRuRtDUSHSHLQy/QlNolifTTaTjsrExzdbf1ifpPblp655sducwkD1lEVzxVZF8qRuE0NT
6UkyTr7kwjGItFOwh7DBsOyL6QkQY4FOZ2ki8Al2R30gpXs6nMUGg1BRwCF0rj3+QvmXbj+2t8x5RerR5y7inP1R+oCuo0wvfcTe
OMfYfZrjdE3whziFh5Qn3mR7vaevmV9peDWQ3LYyK2ec3KpGVeEXSqM+10WyL4ahY7EHuQIzIZROGFGKfW50cUYwdillhKy24g
NdsUzi5klG66DDQtCKEOLbNutDvA0xqCTW3MvRNORSbvckL6g3gM+cStJ5PQ2XUjF9yz9zdwmramnXI6k+MK8V9lrMkc0XFkDM
HOxDlfG2jHhkFuUTgmiKp7hQMg==\">\r\n  <JsonTableDataSource Name=\"JSON\"
DataType=\"FastReport.Data.JsonConnection.JsonParser.JsonArray\" Enabled=\"true\" TableName=\"JSON\">\r\n    <Column
Name=\"index\" DataType=\"System.Int32\"/>\r\n    <Column Name=\"item\" DataType=\"FastReport.JsonBase\">\r\n
<Column Name=\"userId\" DataType=\"System.Double\"/>\r\n      <Column Name=\"id\" DataType=\"System.Double\"/>\r\n
<Column Name=\"title\" DataType=\"System.String\"/>\r\n      <Column Name=\"body\" DataType=\"System.String\"/>\r\n
</Column>\r\n    <Column Name=\"array\" DataType=\"FastReport.JsonBase\"/>\r\n
</JsonTableDataSource>\r\n</JsonDataSourceConnection>\r\n",
  "subscriptionId": "604f52e1261a3c19104c0e25",
  "editedTime": "2021-03-31T14:38:10.5792982Z",
  "editorUserId": "2df79f83-07f1-41ba-96b5-7757bbf377df",
  "createdTime": "0001-01-01T00:00:00",
  "creatorUserId": "2df79f83-07f1-41ba-96b5-7757bbf377df",
  "isConnected": true
}
```

Access rights management by template example

Restricting access to private resources is a very important feature of FastReport Corporate Server. The flexible access system allows you to restrict or grant rights to each resource separately, specifying the range of people who can access it

Getting started

You will need the following tools and facilities:

1. Know how to use API key in FastReport Corporate Server.

   This article will skip over additional information on authentication and authorization.

2. Curl tool.

   Any other REST client will work, but the examples will be built for curl.

3. Report template.

   To learn how to upload a report template, refer to the Upload a new template article.

4. Active FastReport Corporate Server subscription.

5. Internet access.

Instruction

1. To view the current rights to the resource, execute the `GET` request to `{your_host_name}/api/rp/v1/Templates/File/{id}/permissions`, where the template ID should be used instead of `{id}`.

   Example request:

   ```
   curl -X GET "{your_host_name}/api/rp/v1/Templates/File/5fc9ece6b792c90001d94b13/permissions" -H "accept: text/plain"
   ```

   Example answer:

```
{
 "permissions": {
  "ownerId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491",
  "owner": {
   "create": "All",
   "delete": "All",
   "execute": "All",
   "get": "All",
   "update": "All",
   "administrate": "All"
  },
  "groups": null,
  "other": {
   "create": "All",
   "delete": "All",
   "execute": "All",
   "get": "All",
   "update": "All",
   "administrate": "All"
  },
  "anon": null
 }
}
```

In this example, the owner and other members of the workspace have full access to the report template.

2. To change the permissions, execute the `POST` request to `{your_host_name}/api/rp/v1/Templates/File/{id}/permissions`, where the template ID should be used instead of `{id}`.

   Example model:

```
{
 "newPermissions": {
  "ownerId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491",
  "owner": {
   "create": -1,
   "delete": -1,
   "execute": -1,
   "get": -1,
   "update": 0,
   "administrate": -1
  },
  "other": {
   "create": 0,
   "delete": 0,
   "execute": 0,
   "get": 0,
   "update": 0,
   "administrate": 0
  }
 },
 "administrate": "Owner,Other"
}
```

   Important! In the request to update the rights in the `administrate` property, the values `Owner` and `Other` were passed in, in this case, only the specified rights categories, i.e., `other` and `owner` will be updated, and the categories `anon` and `groups` will not be changed.

   Example request that restricts the owner's right to update the report (e.g., editing in the online designer), as well as all workspace participants' rights to the file:

```
curl -X POST "{your_host_name}/api/rp/v1/Templates/File/5fc9ece6b792c90001d94b13/permissions" -H "accept: text/plain" -H "Content-Type: application/json-patch+json" -d "{ \"newPermissions\": { \"ownerId\": \"5af5a8dc-8cb0-40f9-ac99-ca2533fa4491\", \"owner\": { \"create\": -1, \"delete\": -1, \"execute\": -1, \"get\": -1, \"update\": 0, \"administrate\": -1 }, \"other\": { \"create\": 0, \"delete\": 0, \"execute\": 0, \"get\": 0, \"update\": 0, \"administrate\": 0 } }, \"administrate\": \"Owner,Other\"}
```

Example answer:

```
200 OK
```

3. Now you can request the permissions again to make sure they are indeed updated.

   Example request:

```
curl -X GET "{your_host_name}/api/rp/v1/Templates/File/5fc9ece6b792c90001d94b13/permissions" -H "accept: text/plain"
```

   Example answer:

```
{
 "permissions": {
   "ownerId": "2df79f83-07f1-41ba-96b5-7757bbf377df",
 "owner": {
     "create": "All",
     "delete": "All",
     "execute": "All",
  "get": "All",
     "update": "None",
  "administrate": "All"
   },
   "groups": null,
   "other": {
     "create": "None",
     "delete": "None",
     "execute": "None",
     "get": "None",
     "update": "None",
     "administrate": "None"
   },
   "anon": null
 }
}
```

What next?

- Create report.
- Work with groups.
- Add new users to the workspace.

Create report

This article describes the process of creating a report from a template using the FastReport Corporate Server report processor.

Getting started

You will need the following tools and facilities:

1. Know how to use API key in FastReport Corporate Server.

   This article will skip over additional information on authentication and authorization.

2. Curl tool.

   Any other REST client will work, but the examples will be built for curl.

3. Report template.

   To learn how to upload a report template, refer to the Upload a new template article.

4. Active FastReport Corporate Server subscription.

5. Internet access.

Instruction

1. To create a template, you need its ID. Execute `GET` request to `{your_host_name}/api/rp/v1/Templates/Root` to retrieve the root directory.

   Example request:

   ```
   curl -X GET "{your_host_name}/api/rp/v1/Templates/Root" -H "accept: text/plain"
   ```

   Example answer:

   ```
   {
    "name": "RootFolder",
    "parentId": null,
    "tags": [],
    "icon": "",
    "type": "Folder",
    "size": 16384,
    "subscriptionId": "5fa919fa292a8300019349bc",
    "status": "None",
    "id": "5fa919f9292a8300019349b9",
    "createdTime": "2020-11-09T10:29:13.928Z",
    "creatorUserId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491",
    "editedTime": "2020-11-13T15:58:45.69Z",
    "editorUserId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491"
   }
   ```

   Directory ID from the example above: `5fa919f9292a8300019349b9`.

2. To get a list of files in the directory, execute `GET` request to `{your_host_name}/api/rp/v1/Templates/Folder/{id}/ListFiles?skip=0&take=10`, where the directory ID should be used instead of `{id}`.

   Example request:

```
curl -X GET "{your_host_name}/api/rp/v1/Templates/Folder/5fa919f9292a8300019349b9/ListFiles?skip=0&take=10" -H "accept: text/plain"
```

Example answer:


```
[
  {
    "reportInfo": {
      "author": null,
      "created": "2020-12-04T10:58:57Z",
      "creatorVersion": "20.20.4.1",
      "description": null,
      "modified": "2020-12-04T11:00:20Z",
      "name": null,
      "picture": null,
      "previewPictureRatio": 0,
      "saveMode": "All",
      "savePreviewPicture": false,
      "tag": null,
      "version": null
    },
    "name": "template.frx",
    "parentId": "5fa919f9292a8300019349b9",
    "tags": null,
    "icon": null,
    "type": "File",
    "size": 17159,
    "subscriptionId": "5fa919fa292a8300019349bc",
    "status": "Success",
    "id": "5fc9ece6b792c90001d94b13",
    "createdTime": "2020-12-04T08:01:42.708Z",
    "creatorUserId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491",
    "editedTime": "2020-12-04T08:01:42.708Z",
    "editorUserId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491"
  }
]
```


Template ID from the example above: `5fc9ece6b792c90001d94b13`.

3. To create a report, you will need a directory to store the report in. Request the root report directory by executing `GET` request to `{your_host_name}/api/rp/v1/Reports/Root`.

   Example request:

   ```
   curl -X GET "{your_host_name}/api/rp/v1/Reports/Root" -H "accept: text/plain"
   ```

   Example answer:


   ```
   {
    "name": "RootFolder",
    "parentId": null,
    "tags": null,
    "icon": null,
    "type": "Folder",
    "size": 16384,
    "subscriptionId": "5fa919fa292a8300019349bc",
    "status": "None",
    "id": "5fa919f9292a8300019349ba",
    "createdTime": "2020-11-09T10:29:13.993Z",
    "creatorUserId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491",
    "editedTime": "0001-01-01T00:00:00Z",
    "editorUserId": null
   }
   ```


Directory ID from the example above: `5fa919f9292a8300019349ba`.

4. To create a report, execute the `POST` request to `{your_host_name}/api/rp/v1/Templates/File/{id}/Prepare`, where the template ID should be used instead of `{id}`.

   In the request body, pass JSON according to the scheme below.

```
{
  "name": "string",
  "folderId": "string",
  "reportParameters": {
    "additionalProp1": "string",
    "additionalProp2": "string",
    "additionalProp3": "string"
  }
}
```

   - `folderId` — ID of the directory where the report will be placed.
   - `name` — name of the resulting file. Add `.fpx` extension manually.
   - `reportParameters` — parameters that are specified in the report itself via the report designer, they are not needed in this example.

> If you do not specify folderId, the generated report will be saved to the root folder.

   Example request:

```
curl -X POST "{your_host_name}/api/rp/v1/Templates/File/5fc9ece6b792c90001d94b13/Prepare" -H "accept: text/plain" -H "Content-Type: application/json-patch+json" -d "{ \"name\": \"awesome_report.fpx\", \"folderId\": \"5fa919f9292a8300019349ba\"}"
```

   Example answer:

```
{
  "templateId": "5fc9ece6b792c90001d94b13",
  "reportInfo": null,
  "name": "awesome_report.fpx",
  "parentId": "5fa919f9292a8300019349ba",
  "tags": null,
  "icon": null,
  "type": "File",
  "size": 16384,
  "subscriptionId": "5fa919fa292a8300019349bc",
  "status": "InQueue",
  "id": "5fe4614bcd7c55000148e4c6",
  "createdTime": "2020-12-24T09:37:15.7169531+00:00",
  "creatorUserId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491",
  "editedTime": "2020-12-24T09:37:15.7169582+00:00",
  "editorUserId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491"
}
```

   Pay attention to the status of the `InQueue` file, it means that the task for generation has been created and then it has been added to the builder's queue. Already at this stage the report has received its ID for work: `5fe4614bcd7c55000148e4c6`.

You should wait a while for the report to be generated. You can call the method of getting the report in a loop every few milliseconds and check the report status.

5. For information about the report, execute the `GET` request to `{your_host_name}/api/rp/v1/Reports/File/{id}`, where the template ID should be used instead of `{id}`.

Example request:

```
curl -X GET "{your_host_name}/api/rp/v1/Reports/File/5fe4614bcd7c55000148e4c6" -H "accept: text/plain"
```

Example answer:


```
{
  "templateId": "5fc9ece6b792c90001d94b13",
  "reportInfo": null,
  "name": "awesome_report.fpx",
  "parentId": "5fa919f9292a8300019349ba",
  "tags": null,
  "icon": null,
  "type": "File",
  "size": 16927,
  "subscriptionId": "5fa919fa292a8300019349bc",
  "status": "Success",
  "id": "5fe4614bcd7c55000148e4c6",
  "createdTime": "2020-12-24T09:37:15.716Z",
  "creatorUserId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491",
  "editedTime": "2020-12-24T09:37:15.716Z",
  "editorUserId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491"
}
```


Note the status of the `Success` file. The report was successfully generated.

6. To download the report, execute the `GET` request to `{your_host_name}/download/r/{id}`, where the report ID should be passed instead of `{id}``.

   Example request:

   ```
   curl -X GET "{your_host_name}/download/r/5fe4614bcd7c55000148e4c6" -H "accept: text/plain"
   ```

   The file will be received in the response.

What next?

- Export report to PDF.

Report parameters

This article covers the process of passing parameters to a report, which are a dictionary of parameters provided when generating a report. More information on this can be found in the FastReport .NET guide in the Report parameters section.

Getting started

You will need the following tools and facilities:

1. Know how to use API key in FastReport Corporate Server.

   This article will skip over additional information on authentication and authorization.

2. Curl tool.

   Any other REST client will work, but the examples will be built for curl.

3. Report template.

   It can be created in the free FastReport Community Designer program.

4. Active FastReport Corporate Server subscription.

5. Internet access.

Pass parameters to the report

Parameters can be passed when:

- preparing a report;
- exporting from a template;
- working with tasks.

Let’s consider parameter passing on the example of report generation. Detailed instructions on how to create a report can be found in the Create report section.

To create a report, execute the `POST` request to the `{your_host_name}/api/rp/v1/Templates/File/{id}/Prepare`, where the template ID should be used instead of `{id}`.

In the request body, pass JSON according to the scheme below.

```
{
 "name": "string",
 "folderId": "string",
 "reportParameters": {
  "additionalProp1": "string",
  "additionalProp2": "string"
 }
}
```

- `folderId` — ID of the directory where the report will be placed.
- `name` — name of the resulting file. Add `.fpx` extension manually.
- `reportParameters` — parameters that are specified in the report itself using the report designer.

> If you do not specify folderId, the generated report will be saved to the root folder.

Example request:

```
curl -X POST "{your_host_name}/api/rp/v1/Templates/File/5fc9ece6b792c90001d94b13/Prepare"
-H "accept: text/plain"
-H "Content-Type: application/json-patch+json"
-d "{
 "name": "awesome_report.fpx",
 "folderId": "5fa919f9292a8300019349ba",
 "reportParameters": {
   "Parameter1": "Value1",
   "Parameter2": "Value2"
   }
 }"
```

Example answer:


```
{
  "templateId": "5fc9ece6b792c90001d94b13",
  "name": "awesome_report.fpx",
  "parentId": "5fa919f9292a8300019349ba",
  "type": "File",
  "size": 16384,
  "subscriptionId": "5fa919fa292a8300019349bc",
  "status": "InQueue",
  "statusReason": "None",
  "id": "5fe4614bcd7c55000148e4c6",
  "createdTime": "2020-12-24T09:37:15.7169531+00:00",
  "creatorUserId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491",
  "editedTime": "2020-12-24T09:37:15.7169582+00:00",
  "editorUserId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491"
}
```

Export report to PDF

This article covers the process of exporting a report using the FastReport Corporate Server report processor.

Getting started

You will need the following tools and facilities:

1. Know how to use API key in FastReport Corporate Server. How to retrieve and use API key you can learn in the Authentication and authorization article.

   This article will skip over additional information on authentication and authorization.

2. Curl tool.

   Any other REST client will work, but the examples will be built for curl.

3. Report template.

   To learn how to upload a report template, refer to the Upload a new template article.

4. Active FastReport Corporate Server subscription.

5. Internet access.

Annotation

ONote, that you can export the report directly from the template without saving the report. To do this, run the same commands for the report template, replacing `Report` in the query strings with `Template`, and use the template ID rather than the report ID.

Instruction

1. You will need the report ID to export to PDF. Execute the `GET` request to `{your_host_name}/api/rp/v1/Report/Root` to retrieve the root directory.

   Example request:

   ```
   curl -X GET "{your_host_name}/api/rp/v1/Reports/Root" -H "accept: text/plain"
   ```

   Example answer:

   ```
   {
    "name": "RootFolder",
    "parentId": null,
    "tags": null,
    "icon": null,
    "type": "Folder",
    "size": 16384,
    "subscriptionId": "5fa919fa292a8300019349bc",
    "status": "None",
    "id": "5fa919f9292a8300019349ba",
    "createdTime": "2020-11-09T10:29:13.993Z",
    "creatorUserId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491",
    "editedTime": "0001-01-01T00:00:00Z",
    "editorUserId": null
   }
   ```

   Directory ID from the example above: `5fa919f9292a8300019349ba`.

2. Retrieve a list of the files in the directory by executing `GET` request to `{your_host_name}/api/rp/v1/Reports/Folder/{id}/ListFiles?skip=0&take=10`, where the directory ID should be used instead of `{id}`.

   Example request:

   ```
   curl -X GET "{your_host_name}/api/rp/v1/Reports/Folder/5fa919f9292a8300019349ba/ListFiles?skip=0&take=10" -H "accept: text/plain"
   ```

   Example answer:

   ```
   [
     {
       "templateId": "5fc9ece6b792c90001d94b13",
       "reportInfo": null,
       "name": "awesome_report.fpx",
       "parentId": "5fa919f9292a8300019349ba",
       "tags": null,
       "icon": null,
       "type": "File",
       "size": 16927,
       "subscriptionId": "5fa919fa292a8300019349bc",
       "status": "Success",
       "id": "5fe4614bcd7c55000148e4c6",
       "createdTime": "2020-12-24T09:37:15.716Z",
       "creatorUserId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491",
       "editedTime": "2020-12-24T09:37:15.716Z",
       "editorUserId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491"
     }
   ]
   ```

   Report ID from the example above: `5fe4614bcd7c55000148e4c6`.

3. To export the report, you will need a directory in which to place the export file.

   Retrieve the root directory of the exports; to do this, execute `GET` request to `{your_host_name}/api/rp/v1/Exports/Root`.

   Example request:

   ```
   curl -X GET "{your_host_name}/api/rp/v1/Exports/Root" -H "accept: text/plain"
   ```

   Example answer:

   ```
   {
     "name": "RootFolder",
     "parentId": null,
     "tags": null,
     "icon": null,
     "type": "Folder",
     "size": 16384,
     "subscriptionId": "5fa919fa292a8300019349bc",
     "status": "None",
     "id": "5fa919fa292a8300019349bb",
     "createdTime": "2020-11-09T10:29:14.002Z",
     "creatorUserId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491",
     "editedTime": "0001-01-01T00:00:00Z",
     "editorUserId": null
   }
   ```

   Directory ID from the example above: `5fa919fa292a8300019349bb`.

4. To export the report, execute the `POST` request to `{your_host_name}/api/rp/v1/Reports/File/{id}/Export`, where report ID should be used instead of `{id}`.

   In the request body, pass JSON according to the scheme below.

```
{
 "fileName": "awesome_result.pdf",
 "folderId": "5fa919fa292a8300019349bb",
 "locale": "en-GB",
 "format": "Pdf",
 "exportParameters": {
  "additionalProp1": {},
  "additionalProp2": {},
  "additionalProp3": {}
 }
}
```

   - `folderId` — ID of the directory where the export will be placed. If left blank, the export will be placed in the root folder for exports in the workspace.
   - `fileName` — name of the resulting file. If you do not specify the extension, or specify it incorrectly, the server will replace it by itself.
   - `locale` — localization of the exported report. This option will change the date and number formats to those corresponding to the selected ISO code (e.g., French (Switzerland) looks like “fr-CH”). If you leave this field empty or specify a non-existent country, the default locale from the subscription or English (USA) will be substituted in case the default locale is not specified.
   - `format` — export format.
   - `exportParameters` — export parameters. They are set similarly to the export parameters from the FastReport .NET library. A more detailed description is stored in the user documentation in the export parameters section.

> If you do not specify folderId, the generated report will be saved to the root folder.

   Example request:

```
curl -X POST "{your_host_name}/api/rp/v1/Reports/File/5fe4614bcd7c55000148e4c6/Export" -H "accept: text/plain" -H "Content-Type: application/json-patch+json" -d "{ \"fileName\": \"awesome_result.pdf\", \"folderId\": \"5fa919fa292a8300019349bb\", \"locale\":\"ru-RU\", \"format\": \"Pdf\"}"
```

   Example answer:

```
{
 "format": "Pdf",
 "reportId": "5fe4614bcd7c55000148e4c6",
 "name": "awesome_result.pdf",
 "parentId": "5fa919fa292a8300019349bb",
 "tags": null,
 "icon": null,
 "type": "File",
 "size": 16384,
 "subscriptionId": "5fa919fa292a8300019349bc",
 "status": "InQueue",
 "id": "5fe46a33cd7c55000148e4c7",
 "createdTime": "2020-12-24T10:15:15.8039648+00:00",
 "creatorUserId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491",
 "editedTime": "2020-12-24T10:15:15.8039697+00:00",
 "editorUserId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491"
}
```

   Please note the status of the `InQueue` file, it means that the task for export has been created and it has

been added to the builder's queue. Already at this stage the file has received its ID for work `5fe46a33cd7c55000148e4c7`.

You should wait a while for the export to be completed.

5. For information about the file, execute the `GET` request to `{your_host_name}/api/rp/v1/Exports/File/{id}`, where the export ID should be used instead of `{id}`.

   Example request:

   ```
   curl -X GET "{your_host_name}/api/rp/v1/Exports/File/5fe46a33cd7c55000148e4c7" -H "accept: text/plain"
   ```

   Example answer:

   ```
   {
    "format": "Pdf",
    "reportId": "5fe4614bcd7c55000148e4c6",
    "name": "awesome_result.pdf",
    "parentId": "5fa919fa292a8300019349bb",
    "tags": null,
    "icon": null,
    "type": "File",
    "size": 41142,
    "subscriptionId": "5fa919fa292a8300019349bc",
    "status": "Success",
    "id": "5fe46a33cd7c55000148e4c7",
    "createdTime": "2020-12-24T10:15:15.803Z",
    "creatorUserId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491",
    "editedTime": "2020-12-24T10:15:15.803Z",
    "editorUserId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491"
   }
   ```

   Note the status of the `Success` file. The report was successfully exported.

6. To download a report, execute the `GET` request to `{your_host_name}/download/e/{id}`, where the report ID should be passed instead of `{id}`.

   Example request:

   ```
   curl -X GET "{your_host_name}/download/e/5fe46a33cd7c55000148e4c7" -H "accept: text/plain"
   ```

   The file will be received in the response.

What next?

- Work with groups.
- Add new users to the workspace.

Add new users to the workspace

This article covers the process of adding a new user to a workspace, retrieving the list of users in the workspace, and removing a user from it.

> Each workspace always has a subscription associated with it. They have a common ID.

Getting started

You will need the following tools and facilities:

1. Know how to use API key in FastReport Corporate Server.

   This article will skip over additional information on authentication and authorization.

2. Curl tool.

   Any other REST client will work, but the examples will be built for curl.

3. Active FastReport Corporate Server subscription that has at least two user slots.

4. Internet access.

Instruction

There are two ways to add a user to a workspace:

- Directly by specifying the user ID.
- Using the invitation system.

Add a user via invitation

1. To create an invitation, you need a workspace ID.

   Retrieve the workspace ID by executing `GET` request to `{your_host_name}/api/manage/v1/Subscriptions?skip=0&take=10`.

   Example request:

   ```
   curl -X GET "{your_host_name}/api/manage/v1/Subscriptions?skip=0&take=10" -H "accept: text/plain"
   ```

   Example answer:

```
{
 "subscriptions": [
  {
   "id": "604f52e1261a3c19104c0e25",
   "name": "iwantpizza",
   "locale": null,
   "current": {
    "startTime": "2021-03-15T12:28:17.773Z",
    "endTime": "2121-03-15T12:28:17.773Z",
    "plan": {
     "id": "5f89b4d722d2d823440b6d10",
     "isActive": false,
     "displayName": "unlimited",
     "timePeriodType": "Year",
     "timePeriod": 100,
     "readonlyTimeLimitType": "Second",
     "readonlyTimeLimit": 0,
     "templatesSpaceLimit": 1048576000,
     "reportsSpaceLimit": 1048576000,
     "exportsSpaceLimit": 1048576000,
     "fileUploadSizeLimit": 1073741824,
     "dataSourceLimit": 10,
     "maxUsersCount": 10,
     "groupLimit": 5,
     "onlineDesigner": true,
     "isDemo": false,
     "urlToBuy": "https://www.fast-report.com/",
     "unlimitedPage": true,
     "pageLimit": 0
    }
   },
   "old": null,
   "invites": null,
   "templatesFolder": {
    "folderId": "604f52e1261a3c19104c0e22",
    "bytesUsed": 241247
   },
   "reportsFolder": {
    "folderId": "604f52e1261a3c19104c0e23",
    "bytesUsed": 16384
   },
   "exportsFolder": {
    "folderId": "604f52e1261a3c19104c0e24",
    "bytesUsed": 8059419
   }
  }
 ],
 "count": 1,
 "skip": 0,
 "take": 10
}
```


Workspace (subscription) ID from the example above: `604f52e1261a3c19104c0e25`.

2. To create an invitation, execute the `POST` request to `{your_host_name}/api/manage/v1/Subscriptions/{subscriptionId}/invite`, where the workspace ID will be specified instead of `{subscriptionId}`.

   By specifying the invitation properties in the request body.

   Example body:

```
{
  "usages": 1,
  "durable": true,
  "expiredDate": "2021-03-25T11:53:29.351Z"
}
```

- `usages` - possible number of uses;
- `durable` - if it is set to true, it indicates that the number of uses is not limited;
- `expiredDate` - it indicates the expiration date of the invitation (the the invitation valid forever);

  If you leave the body empty, it will create a one-time invitation valid until the next day.

Example request:

```
curl -X POST "{your_host_name}/api/manage/v1/Subscriptions/604f52e1261a3c19104c0e25/invite" -H "accept: text/plain" -H "Content-Type: application/json-patch+json" -d "{ \"usages\": 1, \"durable\": true, \"expiredDate\": \"2021-03-25T11:53:29.351Z\"}"
```

Example answer:

```
{
 "id": "604f52e1261a3c19104c0e25",
 "name": "iwantpizza",
 "locale": null,
 "current": {
  "startTime": "2021-03-15T12:28:17.773Z",
  "endTime": "2121-03-15T12:28:17.773Z",
  "plan": {
   "id": "5f89b4d722d2d823440b6d10",
   "isActive": false,
   "displayName": "unlimited",
   "timePeriodType": "Year",
   "timePeriod": 100,
   "readonlyTimeLimitType": "Second",
   "readonlyTimeLimit": 0,
   "templatesSpaceLimit": 1048576000,
   "reportsSpaceLimit": 1048576000,
   "exportsSpaceLimit": 1048576000,
   "fileUploadSizeLimit": 1073741824,
   "dataSourceLimit": 10,
   "maxUsersCount": 10,
   "groupLimit": 5,
   "onlineDesigner": true,
   "isDemo": false,
   "urlToBuy": "https://www.fast-report.com",
   "unlimitedPage": true,
   "pageLimit": 0
  }
 },
 "old": null,
 "invites": [
  {
   "usages": 1,
   "durable": true,
   "accessToken": "fj3534g341ir7dyytfiaap9z1r",
   "expiredDate": "2021-03-25T11:53:29.351Z",
   "addedUsers": [],
   "creatorUserId": "2df79f83-07f1-41ba-96b5-7757bbf377df"
  }
 ],
 "templatesFolder": {
  "folderId": "604f52e1261a3c19104c0e22",
  "bytesUsed": 241247
 },
 "reportsFolder": {
  "folderId": "604f52e1261a3c19104c0e23",
  "bytesUsed": 16384
 },
 "exportsFolder": {
  "folderId": "604f52e1261a3c19104c0e24",
  "bytesUsed": 8059419
 }
}
```


The invitation with access token appeared in the subscription: `fj3534g341ir7dyytfiaap9z1r`.

3. You need to pass the link to accept the invitation (or the access token itself) to the user you want to invite (in any way you like). The final link will look like this: `{your_host_name}/api/manage/v1/Subscriptions/{subscriptionId}/invite/{accessToken}/accept`, where the workspace ID should be specified instead of `{subscriptionId}`, and the access token should be specified instead of `{accessToken}`.

4. Now the invited user can click on this link directly in the browser or the execute `GET` request to

`{your_host_name}/api/manage/v1/Subscriptions/{subscriptionId}/invite/{accessToken}/accept`, where instead of `{subscriptionId}` the workspace ID should be specified, and the access token should be specified instead of `{accessToken}`.

Example request:

```
curl -X GET "https://xn--80ab2acne.xn--e1aflibyb2b0b.xn--p1ai/api/manage/v1/Subscriptions/6051f2a06c07a10001737394/invite/to9kxrxdz4iwbfyz3pq4fktdcr/accept" -H "accept: text/plain"
```

Delete the invitation

- When an invitation runs out of uses or expires, it is not automatically deleted. You will need to do this manually by sending `DELETE` request to `{your_host_name}/api/manage/v1/Subscriptions/{subscriptionId}/invite/{accessToken}`.

  Example request:

  ```
  curl -X DELETE "https://xn--80ab2acne.xn--e1aflibyb2b0b.xn--p1ai/api/manage/v1/Subscriptions/6051f2a06c07a10001737394/invite/to9kxrxdz4iwbfyz3pq4fktdcr" -H "accept: text/plain"
  ```

  An empty message with the code `NO CONTENT 204` will be received in response.

Add a user directly

1. To add a new user to a workspace, a workspace ID is required.

   Retrieve the subscription ID by executing `GET` request to `{your_host_name}/api/manage/v1/Subscriptions?skip=0&take=10`.

   Example request:

   ```
   curl -X GET "{your_host_name}/api/manage/v1/Subscriptions?skip=0&take=10" -H "accept: text/plain"
   ```

   Example answer:

```
{
 "subscriptions": [
  {
   "id": "5fa919fa292a8300019349bc",
   "name": "Awesome Corp",
   "current": {
    "startTime": "2020-11-17T10:22:58.584Z",
    "endTime": "2025-11-17T10:22:58.584Z",
    "plan": {
     "id": "5f43924b0231500001225686",
     "isActive": false,
     "displayName": "The greatest power",
     "timePeriodType": "Year",
     "timePeriod": 5,
     "readonlyTimeLimitType": "Second",
     "readonlyTimeLimit": 0,
     "templatesSpaceLimit": 1048576000,
     "reportsSpaceLimit": 1048576000,
     "exportsSpaceLimit": 1048576000,
     "fileUploadSizeLimit": 1048576000000,
     "dataSourceLimit": 10,
     "maxUsersCount": 10,
     "groupLimit": 5,
     "onlineDesigner": true,
     "isDemo": false,
     "urlToBuy": "https://www.fast-report.com",
     "unlimitedPage": true,
     "pageLimit": 15
    }
   },
   "old": [],
   "templatesFolder": {
    "folderId": "5fa919f9292a8300019349b9",
    "bytesUsed": 1668491
   },
   "reportsFolder": {
    "folderId": "5fa919f9292a8300019349ba",
    "bytesUsed": 6085990
   },
   "exportsFolder": {
    "folderId": "5fa919fa292a8300019349bb",
    "bytesUsed": 8336710
   }
  }
 ],
 "count": 1,
 "skip": 0,
 "take": 10
}
```


Workspace (subscription) ID from the example above: `5fa919fa292a8300019349bc`.

2. To add a new user, execute `PUT` request to `{your_host_name}/api/manage/v1/Subscriptions/{subscriptionId}/users/{userId}`, where the workspace ID should be passed instead of `{subscriptionId}`, and the user ID should be passed instead of `{userId}`.

   Example request:

   ```
   curl -X PUT "{your_host_name}/api/manage/v1/Subscriptions/5fa919fa292a8300019349bc/users/5af5a8dc-8cb0-40f9-ac99-ca2533fa4492" -H "accept: text/plain"
   ```

   In response you will receive an empty message with the `OK 200` status code.

3. To retrieve the list of users, execute `GET` request to `{your_host_name}/api/manage/v1/Subscriptions/{id}/users?skip=0&take=10`, where the workspace ID should be used instead of `{id}`.

   Example request:

   ```
   curl -X GET "{your_host_name}/api/manage/v1/Subscriptions/5fa919fa292a8300019349bc/users?skip=0&take=10" -H "accept: text/plain"
   ```

   Example answer:

   ```
   {
    "users": [
     {
      "userId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491"
     },
     {
      "userId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4492"
     }
    ],
    "count": 2,
    "take": 10,
    "skip": 0
   }
   ```

4. To remove a user from the workspace, execute `DELETE` request to `{your_host_name}/api/manage/v1/Subscriptions/{subscriptionId}/users/{userId}`, where instead of `{subscriptionId}` the workspace ID should be used, and instead of `{userId}` - user ID.

   Example request:

   ```
   curl -X DELETE "{your_host_name}/api/manage/v1/Subscriptions/5fa919fa292a8300019349bc/users/5af5a8dc-8cb0-40f9-ac99-ca2533fa4492" -H "accept: text/plain"
   ```

   In response you will receive an empty message with the `OK 200` status code.

What next?

- Work with groups
- Help and feedback

Work with groups

This article covers the process of creating a new group, adding a user to the group, and retrieving the list of users in it.

Getting started

You will need the following tools and facilities:

1. Know how to use API key in FastReport Corporate Server.

   This article will skip over additional information on authentication and authorization.

2. Curl tool.

   Any other REST client will work, but the examples will be built for curl.

3. Active FastReport Corporate Server subscription that has at least two user slots.

4. Internet access.

Annotation

Important! You can add a user to a group only if the user exists in the workspace.

Important! It is possible to add a user to a group only by their user ID.

Instruction

1. To create a new group, you need a workspace ID and the name of the new group.

   Retrieve the workspace ID by executing `GET` request to `{your_host_name}/api/manage/v1/Subscriptions?skip=0&take=10`.

   Example request:

   ```
   curl -X GET "{your_host_name}/api/manage/v1/Subscriptions?skip=0&take=10" -H "accept: text/plain"
   ```

   Example answer:

```
{
 "subscriptions": [
  {
   "id": "5fa919fa292a8300019349bc",
   "name": "Awesome Corp",
   "current": {
    "startTime": "2020-11-17T10:22:58.584Z",
    "endTime": "2025-11-17T10:22:58.584Z",
    "plan": {
     "id": "5f43924b0231500001225686",
     "isActive": false,
     "displayName": "The greatest power",
     "timePeriodType": "Year",
     "timePeriod": 5,
     "readonlyTimeLimitType": "Second",
     "readonlyTimeLimit": 0,
     "templatesSpaceLimit": 1048576000,
     "reportsSpaceLimit": 1048576000,
     "exportsSpaceLimit": 1048576000,
     "fileUploadSizeLimit": 1048576000000,
     "dataSourceLimit": 10,
     "maxUsersCount": 10,
     "groupLimit": 5,
     "onlineDesigner": true,
     "isDemo": false,
     "urlToBuy": "https://www.fast-report.com/",
     "unlimitedPage": true,
     "pageLimit": 15
    }
   },
   "old": [],
   "templatesFolder": {
    "folderId": "5fa919f9292a8300019349b9",
    "bytesUsed": 1668491
   },
   "reportsFolder": {
    "folderId": "5fa919f9292a8300019349ba",
    "bytesUsed": 6085990
   },
   "exportsFolder": {
    "folderId": "5fa919fa292a8300019349bb",
    "bytesUsed": 8336710
   }
  }
 ],
 "count": 1,
 "skip": 0,
 "take": 10
}
```


Workspace (subscription) ID from the example above: `5fa919fa292a8300019349bc`.

2. To create a new group, execute the `POST` request to `{your_host_name}/api/manage/v1/Groups`, pass JSON into the request body according to the scheme below.


```
{
 "name": "string",
 "subscriptionId": "string id"
}
```


Example request:

```
curl -X POST "{your_host_name}/api/manage/v1/Groups" -H "accept: text/plain" -H "Content-Type: application/json-patch+json" -d "{ \"name\": \"My first group\", \"subscriptionId\": \"5fa919fa292a8300019349bc\"}"
```

Example answer:

```
{
 "id": "5fe5d7866882ca0001760fcb",
 "name": "My first group",
 "subscriptionId": "5fa919fa292a8300019349bc"
}
```

Group ID from the example above: `5fe5d7866882ca0001760fcb`.

3. To add a new user to the group, execute `PUT` request to `{your_host_name}/api/manage/v1/Groups/{groupId}/Users/{userId}`, where the group ID should be specified instead of `{groupId}`, and user ID - instead of `{userId}`.

   Example request:

   ```
   curl -X PUT "{your_host_name}/api/manage/v1/Groups/5fe5d7866882ca0001760fcb/Users/5af5a8dc-8cb0-40f9-ac99-ca2533fa4492" -H "accept: text/plain"
   ```

   In response you will receive an empty message with the `OK 200` code.

4. To retrieve a list of users in the group, execute `GET` request to `{your_host_name}/api/manage/v1/Groups/{id}/Users?skip=0&take=10`, where group ID should be specified instead of `{id}`.

   Example request:

   ```
   curl -X GET "{your_host_name}/api/manage/v1/Groups/5fe5d7866882ca0001760fcb/Users?skip=0&take=10" -H "accept: text/plain"
   ```

   Example answer:

   ```
   {
    "users": [
      {
       "userId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4491",
       "userId": "5af5a8dc-8cb0-40f9-ac99-ca2533fa4492"
      }
    ],
    "count": 2,
    "take": 10,
    "skip": 0
   }
   ```

What next?

- Help and feedback

Work with tasks

Tasks in FastReport Corporate Server are actions to convert and deliver documents to consumers. They are described in detail in the Tasks section.

Getting started

You will need the following tools and facilities:

1. Know how to use API key in FastReport Corporate Server.

   This article will skip over additional information on authentication and authorization.

2. Curl tool.

   Any other REST client will work, but the examples will be built for curl.

3. Active FastReport Corporate Server subscription that has at least two user slots.

4. Internet access.

Work with tasks

Tasks in FastReport Corporate Server are actions to convert and deliver documents to consumers. They are described in detail in the Tasks section.

Getting started

You will need the following tools and facilities:

1. Know how to use API key in FastReport Corporate Server.

   This article will skip over additional information on authentication and authorization.

2. Curl tool.

   Any other REST client will work, but the examples will be built for curl.

3. Active FastReport Corporate Server subscription that has at least two user slots.

4. Internet access.

About ViewModels and types

On work you will pass different ViewModels in JSON. It is important that the first field of any ViewModel must be "$t": "Name". Detailed examples will be provided later in this article.

VMs for tasks creation:

- CreatePrepareTaskVM
- CreateExportTemplateTaskVM
- CreateExportReportTaskVM
- CreateFetchTaskVM
- CreateEmailTaskVM
- CreateWebhookTaskVM
- CreateFTPUploadTaskVM

VMs for raw tasks run:

- RunPrepareTaskVM
- RunExportTemplateTaskVM
- RunExportReportTaskVM
- RunFetchTaskVM
- RunEmailTaskVM
- RunWebhookTaskVM
- RunFTPUploadTaskVM

VMs for tasks' fields update:

- UpdatePrepareTaskVM
- UpdateExportTemplateTaskVM
- UpdateExportReportTaskVM
- UpdateFetchTaskVM
- UpdateEmailTaskVM
- UpdateWebhookTaskVM
- UpdateFTPUploadTaskVM

VM for tasks' permissions update:

- UpdateTaskPermissionsVM

Creating a task

To create tasks, use the CreateTask method. It can take any VM for task creation.

Let’s look at how to create a task of exporting a report and then sending it to Webhook:

```
curl -X 'POST' \
 '{your_host_name}/api/tasks' \
 -H 'accept: application/json' \
 -H 'Content-Type: application/json' \
 -d '{
 "subscriptionId": "23e0134c816935c1e11b3737",
 "type": "ExportTemplate",
 "name": "SendViaWebhookExport",
 "inputFile": {
   "entityId": "61e0134c816935c1e11b3787"
 },
 "transports": [
   {
    "type": "Webhook",
    "Endpoints": [
{
   "BearerToken": "allotoken",
         "Url": "https://localhost:7104/api",
         "Headers": {"header1":"val1", "header2":"val2"}
}
    ]
  }
 ],
 "format": "Pdf"
}'
```

Real object IDs should be written into the EntityId and FolderId fields. Otherwise, the task will be terminated with an error.

> Important! If you don't specify OutputFile, it will be saved to a temporary folder. If you specify an empty OutputFile, it will be saved to the root folder. If you specify a folder ID, it will be saved to the root folder.

A list of all available ViewModels is available at {your_host_name}/api/swagger/index.html

Retrieve task list

```
// Retrieve the first 10 tasks from the workspace
curl -X 'GET' \
 '{your_host_name}/api/tasks?skip=0&take=10' \
 -H 'accept: application/json'
```

Execute task by the specified ID

```
curl -X 'POST' \
 '{your_host_name}/api/tasks/42d134ae3130aaad37by345f/run' \
 -H 'accept: */*' \
 -d ''
```

Delete task from the storage

```
curl -X 'DELETE' \
  '{your_host_name}/api/tasks/42d134ae3130aaad37by345f' \
  -H 'accept: */*'
```

> Important! There is no Authorization header in the examples because a cookie-based authentication model is used. Read more about authorization in the Authentication and authorization section.

Save to FTP server

This article describes how to send a report to an FTP server. Instructions on how to work with tasks are described in the General information section.

Getting started

You will need the following tools and facilities:

1. Know how to use API key in FastReport Corporate Server.

   This article will skip over additional information on authentication and authorization.

2. Curl tool.

   Any other REST client will work, but the examples will be built for curl.

3. Active FastReport Corporate Server subscription that has at least two user slots.

4. Internet access.

5. Configured and available FTP server.

> Important! The items above describe the recommended tools.

> Important! This guide assumes that you have an FTP server configured to accept files from external sources.

Create task

Let’s consider creating a task to send a template to an FTP server:

```
curl -X 'POST' \
 '{your_host_name}/api/tasks/v1/Tasks' \
 -H 'accept: application/json' \
 -H 'Content-Type: application/json' \
 -d '{
 "$t": "CreateFTPUploadTaskVM",
 "name": "FTP send task",
 "subscriptionId": "23e0134c816935c1e11b3737",
 "ftpHost": "ftp://localhost",
 "ftpPort": 21,
 "ftpUsername": "FtpUser",
 "ftpPassword": "password",
 "archive": false,
 "archiveName": "Arcvive name",
 "useSFTP": false,
 "inputFile": {
   "entityId": "61e0134c816935c1e11b3787",
   "type": "Template"
 },
 "destinationFolder": "/destination_path/"
}'
```

```
// Start a task by ID
curl -X 'POST' \
  '{your_host_name}/api/tasks/v1/Tasks/{task ID}/run' \
  -H 'accept: */*' \
  -d ''
```

Real object IDs should be written to the EntityId and SubscriptionId fields. Otherwise, the task will be terminated with an error.

Execute task from the request body

```
curl -X 'POST' \
 '{your_host_name}/api/tasks/v1/Tasks/run' \
 -H 'accept: */*' \
 -H 'Content-Type: application/json' \
 -d '{
 "$t": "RunFTPUploadTaskVM",
 "name": "FTP send task",
 "subscriptionId": "23e0134c816935c1e11b3737",
 "ftpHost": "ftp://localhost",
 "ftpPort": 21,
 "ftpUsername": "FtpUser",
 "ftpPassword": "password",
 "archive": false,
 "archiveName": "Archive name",
 "useSFTP": false,
 "inputFile": {
   "entityId": "61e0134c816935c1e11b3787",
   "type": "Template"
 },
 "destinationFolder": "/destination_path/"
}'
```

> Important! In this case, sending to FTP will be done directly by this request and the task will not be saved in the database.

> Important! There is no Authorization header in the examples because a cookie-based authentication model is used. For more information about authorization, please see Authentication and authorization.

Save over Webhook

This article describes how to send a webhook report. Instructions on how to work with tasks are described in the General information section.

Getting started

You will need the following tools and facilities:

1. Know how to use API key in FastReport Corporate Server.

   This article will skip over additional information on authentication and authorization.

2. Curl tool.

   Any other REST client will work, but the examples will be built for curl.

3. Active FastReport Corporate Server subscription.

4. Internet access.

5. Client application receiving requests.

> Important! The items above describe recommended tools.

> Important! This guide assumes that you have a client application that receives and processes incoming webhooks.

Create task

Let’s look how to create the task of sending a template over a webhook:

```
// Create a task
curl -X 'POST' \
  '{your_host_name}/api/tasks/v1/Tasks' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "$t": "CreateWebhookTaskVM",
  "name": "Webhook sending task",
  "subscriptionId": "{workspace ID}",
  "url": "http://example.com/",
  "headers": {
      "Authorization": "Bearer <token>",
      "Content-Type": "multipart/form-data"
  },
  "inputFile": {
    "entityId": "{template ID}",
    "type": "Template"
  }
}'
```

```
// Start a task by ID
curl -X 'POST' \
  '{your_host_name}/api/tasks/v1/Tasks/{task ID}/run' \
  -H 'accept: */*' \
  -d ''
```

> Important! Real object identifiers should be written to the EntityId and SubscriptionId fields. Otherwise, the

task will be terminated with an error.

Important! The EntityId can be passed in the template, report, and export ID.

Execute task from the request body

```
curl -X 'POST' \
  '{your_host_name}/api/tasks/v1/Tasks/run' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "$t": "RunWebhookTaskVM",
  "name": "Webhook sending task",
  "subscriptionId": "{workspace ID}",
  "url": "http://example.com/",
  "headers": {
    "Authorization": "Bearer <token>",
    "Content-Type": "multipart/form-data",
    "Content-Length": "1278"
  },
  "inputFile": {
    "entityId": "{template ID}",
    "type": "Template"
  }
}'
```

Important! In this case, the webhook sending will be done directly by this request and the task will not be saved in the database.

Example request that comes up on webhook’s endpoint:

```
{
 "startedDateTime": "2024-06-07 08:37:09",
 "request": {
  "method": "POST",
  "url": "https://example.com/6e259560-25e5-482b-a7b2-8c5267ba6ae3/",
  "headers": [
   {
    "name": "connection",
    "value": "close"
   },
   {
    "name": "content-length",
    "value": "1421"
   },
   {
    "name": "content-type",
    "value": "multipart/form-data; boundary=\"62ec6524-9360-4e91-834f-e9531e2d2c30\""
   },
   {
    "name": "host",
    "value": "example.com"
   }
  ],
  "bodySize": 0,
  "postData": {
   "mimeType": "application/json",
   "text": ""
  }
 },
 "response": {
  "status": 200,
  "httpVersion": "HTTP/1.1",
  "headers": [
   {
    "name": "Content-Type",
    "value": "text/html"
   }
  ],
  "content": {
   "size": 145,
   "text": "This URL has no default content configured.",
   "mimeType": "text/html"
  }
 }
}
```


Important! There is no Authorization header in the examples because a cookie-based authentication model is used. For more information about authorization, please see Authentication and authorization.

C# .NET

This section provides step-by-step instructions and guides for performing typical FastReport Corporate Server application tasks using the C#/.NET programming language and FastReport.Cloud.SDK.

Getting started

You will need the following tools and facilities.

1. .NET SDK.
2. C# code editor or text editor, e.g., Visual Studio Code.
3. FastReport Community Designer.

   For some guides, you will need a designer to create the report.
4. Active FastReport Corporate Server subscription.
5. Internet access.

> Important! These guides assume that you already know how to develop your application in the C# programming language.

> Note. The items above describe recommended tools.

Install and configure

To connect the SDK, you should install the `FastReport.Cloud.SDK` package of the latest stable version. This can be done by running the following command.

```
dotnet add package FastReport.Cloud.SDK
```

To add ASP.NET support, install the package `FastReport.Cloud.SDK.Web`.

```
dotnet add package FastReport.Cloud.SDK.Web
```

You should also add a line to the `Startup` class.

```
services.AddFastReportCloud();
```

What next?

We advise you to start reading the following articles:

- Status codes.
- Authentication and authorization.

Authentication and authorization

The process of user data authentication and granting user certain rights in FastReport Corporate Server is carried out in one of two available ways:

1. Via JWT token.

   In this case, authentication must be done manually, and the token will only be valid for 5 minutes, during which time the user must log in to their application. When connecting to the server, the browser will redirect to the authentication server and then generate an access token. We restrict the possibility of retrieving the JWT token to the user personally from a security point of view.

   > If the user has not logged in to the application within 5 minutes, the authentication must be re-authenticated. If the user has logged in, re-authentication is not required.

2. Via API key.

   In this case, the obtaining of access rights is performed for server applications. To get an API key, a user must be present. However, the key itself can be valid for a long time, for example, a year.

Retrieve the first API key

To get the first API key, access the user panel. If for some reason you do not have access to the user panel, you can request a key as described below.

The easiest way to retrieve a key is to open the Api keys tab and create one on this page.

Option 2:

1. Open the link in your browser: <{your_host_name}/account/signin?r= {your_host_name}/api/manage/v1/ApiKeys>.

   If you click on this link, it will direct you to the automatic browser authentication process.

2. Now when authentication has passed, you need to request a new key.

   Press `F12` or `Ctrl+Shift+I` to open the developer panel. The key combinations may differ from the default, in that case open the developer panel via the browser menu

3. Copy and run the code in the JavaScript console.

   This code will make a `POST` request to the URL `{your_host_name}/api/manage/v1/ApiKeys` to create a new access key valid till 2030.

4. Refresh the browser page and get the result.

```
{
  "apiKeys": [
    {
      "value": "cc355oeu1z5d5wncayo33me6c1g5junqdqk4pkupid7t8ynjshey",
      "description": "Generated by js develop panel",
      "expired": "2030-01-01T07:41:23.399Z"
    }
  ],
  "count": 1
}
```

Now you can use the API key. In the example above, `cc355oeu1z5d5wncayo33me6c1g5junqdqk4pkupid7t8ynjshey`

was used.

There is no need to retrieve a new API key through the browser again.

How to use API key

The key should be passed with each request in the `Authorization: Basic`. You should use apikey as the username and the key value as the password. For example:

```
Authorization: Basic Base64Encode(apikey:cc355oeu1z5d5wncayo33me6c1g5junqdqk4pkupid7t8ynjshey);
```

Where `Base64Encode` is a function to encode a string into base64.

For `FastReport.Cloud.SDK` there is a special class that allows you to add a key to the request header <xref:FastReport.Cloud.FastReportCloudApiKeyHeader>.

To add the necessary header, create a new `HttpClient`.

```
HttpClient httpClient = new HttpClient();
httpClient.BaseAddress = new Uri({your_host_name});
httpClient.DefaultRequestHeaders.Authorization = new FastReportCloudApiKeyHeader(apiKey);
```

Now this `HttpClient` can be used for all requests.

Retrieve new API key

To retrieve a new key, call the following method <xref:FastReport.Cloud.Management.ApiKeysClient.CreateApiKeyAsync(FastReport.Cloud.CreateApiKeyVM)>

```
CreateApiKeyVM model = new CreateApiKeyVM()
{
   Description = "Created by FastReport.Cloud.SDK",
   Expired = DateTime.UtcNow.AddYears(1)
};

IApiKeysClient apiKeysClient = new ApiKeysClient(httpClient);
await apiKeysClient.CreateApiKeyAsync(model);
```

> Whenever possible, use asynchronous method analogs instead of synchronous methods.

This function will result in a <xref:FastReport.Cloud.ApiKeyVM> model.

What next?

- Upload new template.
- Work with groups.
- Add new users to the workspace.

Upload new template

One of the main features of FastReport Corporate Server is storing templates, reports, and other data in the cloud. This article will describe how to upload your own template to FastReport Corporate Server template storage.

Getting started

You will need the following tools and facilities:

1. Know how to use API key in FastReport Corporate Server.

   This article will skip over additional information on authentication and authorization.

2. .NET SDK.

3. C# code editor or text editor, e.g., Visual Studio Code.

4. Report template.

   It can be generated in the free FastReport Community Designer program.

5. Active FastReport Corporate Server subscription.

6. Internet access.

> Important! These guides assume that you already know how to develop your application in the C# programming language.

> Note. The items above describe recommended tools.

Instruction

1. You need to get the root folder ID of the workspace. This ID will never change, so you can save it and not have to request it each time before loading a new template.

   To request the root directory, call the method <xref:FastReport.Cloud.ITemplateFoldersClient.GetRootFolderAsync(System.String,System.Threading.CancellationToken)>.

   ```
   public async Task<string> GetTemplatesRoot(HttpClient httpClient, string subscriptionId = null)
   {
      ITemplateFoldersClient templateFoldersClient = new TemplateFoldersClient(httpClient);

      FileVM result = await templateFoldersClient.GetRootFolderAsync(subscriptionId);

      return result.Id;
   }
   ```

   In this example, the `subscriptionId` parameter specifies the workspace (subscription) ID, if it is not specified (equal to null), the default user workspace ID will be returned.

2. To upload the necessary template, use the following method <xref:FastReport.Cloud.ITemplatesClient.UploadFileAsync(System.String,FastReport.Cloud.TemplateCreateVM,System.Threading.CancellationToken)>.

```
public async Task<string> UploadFrx(HttpClient httpClient, string folderId, string filePath)
{
   ITemplatesClient templatesClient = new TemplatesClient(httpClient);

   byte[] bytes = File.ReadAllBytes(filePath);

   TemplateCreateVM model = new TemplateCreateVM()
   {
      Name = Path.GetFileName(filePath),
      Content = Convert.ToBase64String(bytes)
   };

   TemplateVM result = await templatesClient.UploadFileAsync(folderId, model);

   return result.Id;
}
```

In this example, the function retrieves a file from disk and uploads it into the folderId directory. Learn more about the parameters:

- `folderId` — template directory ID.
- `model.Name` — file name, so it will be displayed inside FastReports Corporate Server.

  The file name should have the extension `.frx`, if it is not there, then add it manually.
- `model.Content` — Base64 encoded file content.

The method returns the identifier of the uploaded template.

Now this template can be used to prepare (create) a report and export it.

What next?

- Access rights management by template example.
- Create report.

Access rights management by template example

Restricting access to private resources is a very important feature of FastReport Corporate Server. The flexible access system allows you to restrict or grant rights to each resource separately, specifying the range of people who can access it.

Getting started

You will need the following tools and facilities:

1. Know how to use API key in FastReport Corporate Server.

   This article will skip over additional information on authentication and authorization.

2. .NET SDK.

3. C# code editor or text editor, e.g., Visual Studio Code.

4. Report template.

   To learn how to upload a report template, see the Upload new template article.

5. Active FastReport Corporate Server subscription.

6. Internet access.

> Important! These guides assume that you already know how to develop your application in the C# programming language.

> Note. The items above describe recommended tools.

Instruction

1. To view the current rights to the resource, use the following method <xref:FastReport.Cloud.ITemplatesClient.GetPermissionsAsync(System.String,System.Threading.CancellationToken)>.

   ```
   public async Task<FilePermission> GetOwnerPermissions(HttpClient httpClient, string templateId)
   {
    ITemplatesClient templatesClient = new TemplatesClient(httpClient);

      FilePermissions permissions = await
         templatesClient.GetPermissionsAsync(templateId);

      return permissions.Owner;
   }
   ```

   In this example, the method returns the permissions for the template owner.

   > Important! Classes <xref:FastReport.Cloud.FilePermissions> and <xref:FastReport.Cloud.FilePermission> differ by the letter `s` at the end, the first contains a full description of access rights to the entity, the second contains only a description for one of the categories.

2. To change the permissions, use the following method <xref:FastReport.Cloud.ITemplatesClient.UpdatePermissionsAsync(System.String,FastReport.Cloud.Update

FilePermissionsVM,System.Threading.CancellationToken)>.

```
public async Task<FilePermission> ShareTemplate(HttpClient httpClient, string templateId)
{
   ITemplatesClient templatesClient = new TemplatesClient(httpClient);

   UpdateFilePermissionsVM viewModel = new UpdateFilePermissionsVM()
   {
      NewPermissions = new FilePermissions() {
 Anon = new FilePermission() { Get = FilePermissionGet.Entity | FilePermissionGet.Download }
      },
      Administrate = UpdateFilePermissionsVMAdministrate.Anon
   };

   var result = await templatesClient.AddPermissionAsync(templateId, viewModel);

   return result.Other;
}
```

In this example, the function allows anonymous users to view information about the template and download it.

Important! In this example, we pass only the permissions we want to edit and specify them in the `Administrate` property.

If you need to change several categories of rights at once, you can list them using the bitwise operator OR (`|`).

What next?

- Create report.
- Work with groups.
- Add new users to the workspace.

Create report

This article describes the process of creating a report from a template using the FastReport Corporate Server report processor.

Getting started

You will need the following tools and facilities:

1. Know how to use API key in FastReport Corporate Server.

   This article will skip over additional information on authentication and authorization.

2. .NET SDK.

3. C# code editor or text editor, e.g., Visual Studio Code.

4. Report template.

   To learn how to upload a report template, see the Upload new template article.

5. Active FastReport Corporate Server subscription.

6. Internet access.

> Important! These guides assume that you already know how to develop your application in the C# programming language.

> Note. The items above describe recommended tools.

Instruction

1. You need the template ID to create the template. To get it, use the following method <xref:FastReport.Cloud.ITemplatesClient.GetFilesListAsync(System.String,System.Nullable{System.Int32},System.Nullable{System.Int32},System.Threading.CancellationToken)>

   ```
   public async Task<string> GetTemplateId(HttpClient httpClient)
   {
       ITemplateFoldersClient templateFoldersClient =
           new TemplateFoldersClient(httpClient);
       ITemplatesClient templatesClient = new TemplatesClient(httpClient);

       FileVM rootFolder = await templateFoldersClient.GetRootFolderAsync(null);

       IEnumerable<TemplateVM> templates =
           await templatesClient.GetFilesListAsync(rootFolder.Id, 0, 10);

       TemplateVM template = templates.First();

       return template.Id;
   }
   ```

   In this example, the function requests the root directory of the user's default workspace, then requests 10 templates and returns the first one.

2. To generate a report, you will need a directory to place the report in. Request the report root directory by using the following method <xref:FastReport.Cloud.IReportFoldersClient.GetRootFolderAsync(System.String,System.Threading.Cancella

tionToken)>.

```
public async Task<string> GetReportsRoot(HttpClient httpClient,
                                string subscriptionId = null)
{
   IReportFoldersClient reportFoldersClient = new ReportFoldersClient(httpClient);

  FileVM result = await reportFoldersClient.GetRootFolderAsync(subscriptionId);

   return result.Id;
}
```

In this example, the function requests the root directory, the workspace ID can be omitted. In this case, the root directory for the user's default workspace will be returned.

3. To create a report, use the following method <xref:FastReport.Cloud.ITemplatesClient.PrepareAsync(System.String,FastReport.Cloud.PrepareTemplateTaskVM,System.Threading.CancellationToken)>.

```
public async Task<string> BuildReport(HttpClient httpClient,
                              string folderId,
                              string templateId,
                              string fileName)
{
   ITemplatesClient templatesClient = new TemplatesClient(httpClient);

   PrepareTemplateVM task = new PrepareTemplateVM()
   {
      Name = Path.ChangeExtension(fileName, ".fpx"),
      FolderId = folderId
   };

   ReportVM result = await templatesClient.PrepareAsync(templateId, task);

   return result.Id;
}
```

In this example, the function creates a task to create a report.

Important! The report has not been generated yet, but it has already been assigned an identifier. After some time, the builder's turn will come to this task and the report will be created.

> If you do not specify FolderId, the generated report will be saved to the root folder.

4. For information about the report, use the following method <xref:FastReport.Cloud.IReportsClient.GetFileAsync(System.String,System.Threading.CancellationToken)>.

```
public async Task<ReportVMStatus> CheckStatus(HttpClient httpClient, string reportId)
{
   IReportsClient reportsClient = new ReportsClient(httpClient);

   ReportVM result = await reportsClient.GetFileAsync(reportId);

   return result.Status.GetValueOrDefault();
}
```

In this example, the function requests a report by its ID and returns the status. You must wait for the <xref:FastReport.Cloud.FileStatus.Success> status, check it every few seconds.

5. Check the status in the loop and download the file.

```
    int tries = 10;
    FileStatus status;
    do
    {
        status = await CheckStatus(httpClient, reportId);
        tries--;
    } while (status != FileStatus.Success && tries > 0);

    var report = await DownloadReport(httpClient, reportId);
```

6. To download the report, use the following method <xref:FastReport.Cloud.IDownloadClient.GetReportAsync(System.String,System.Threading.CancellationToken)>.

```
public async Task<byte[]> DownloadReport(HttpClient httpClient, string reportId)
{
   IDownloadClient downloadClient = new DownloadClient(httpClient);

 FileResponse file = await downloadClient.GetReportAsync(reportId);

   using(MemoryStream ms = new MemoryStream())
   {
      file.Stream.CopyTo(ms);

      return ms.ToArray();
   }
}
```

   In this example, the function requests a file and copies it to memory.

What next?

- Export report to PDF.

Report parameters

This article covers the process of passing parameters to a report, which are a dictionary of parameters provided when generating a report. More information on this can be found in the FastReport .NET guide in the Report parameters section.

Getting started

You will need the following tools and facilities:

1. Know how to use API key in FastReport Corporate Server.

   This article will skip over additional information on authentication and authorization.

2. .NET SDK.

3. C# code editor or text editor, e.g., Visual Studio Code.

4. Report template.

   It can be generated in the free FastReport Community Designer program.

5. Active FastReport Corporate Server subscription.

6. Internet access.

> Important! These guides assume that you already know how to develop your application in the C# programming language.

> Note. The items above describe recommended tools.

Pass parameters to the report

Parameters can be passed when:

- preparing a report;
- exporting from a template;
- working with tasks.

Let’s consider parameter passing by example of report generation. Detailed instructions on how to create a report can be found in the Create report section.

```
public async Task PassingReportParameters(HttpClient httpClient,
                        string folderId,
                        string templateId,
                        string fileName)
{
   ITemplatesClient templatesClient = new TemplatesClient(httpClient);
   PrepareTemplateVM task = new PrepareTemplateVM()
   {
    Name = Path.ChangeExtension(fileName, ".fpx"),
    FolderId = folderId,
    ReportParameters = new Dictionary<string, string>
    {
       { "Parameter1", "Value1" },
       { "Parameter2", "Value2" }
    }
};

// Add new parameter to dictionary
task.ReportParameters.Add("Parameter3", "Value3");

ReportVM result = await templatesClient.PrepareAsync(templateId, task);}
```

Export report to PDF

This article covers the process of exporting a report using the FastReport Corporate Server report processor.

Getting started

You will need the following tools and facilities:

1. Know how to use API key in FastReport Corporate Server.

   This article will skip over additional information on authentication and authorization.

2. .NET SDK.

3. C# code editor or text editor, e.g., Visual Studio Code.

4. Report.

   You can learn how to create a report in the Create report article.

5. Active FastReport Corporate Server subscription.

6. Internet access.

> Important! These guides assume that you already know how to develop your application in the C# programming language.

> Note. The items above describe recommended tools.

Annotation

You can export a report directly from the template, without intermediate saving of the report. To do this run the same commands for the report template, replacing `Report` in methods with `Template` use the template ID, not the report ID.

Instruction

1. You will need a report ID to export to PDF. To retrieve it, use the following method <xref:FastReport.Cloud.ITemplatesClient.GetFilesListAsync(System.String,System.Nullable{System.Int32},System.Nullable{System.Int32},System.Threading.CancellationToken)>.

   ```
   public async Task<string> GetReportId(HttpClient httpClient)
   {
      IReportFoldersClient reportFoldersClient = new ReportFoldersClient(httpClient);
      IReportsClient reportsClient = new ReportsClient(httpClient);

      FileVM rootFolder = await reportFoldersClient.GetRootFolderAsync(null);

      IEnumerable<ReportVM> reports =
         await reportsClient.GetFilesListAsync(rootFolder.Id, 0, 10);

      ReportVM report = reports.First();

      return report.Id;
   }
   ```

   In this example, the function requests the root directory of the user's default workspace, then requests 10 reports and returns the first one.

2. To export the report, you will need a directory to save the export to.

   Retrieve the root directory of exports by using the method <xref:FastReport.Cloud.IExportFoldersClient.GetRootFolderAsync(System.String,System.Threading.CancellationToken)>.

```
public async Task<string> GetExportsRoot(HttpClient httpClient,
                                string subscriptionId = null)
{
   IExportFoldersClient exportFoldersClient = new ExportFoldersClient(httpClient);

   FileVM result = await exportFoldersClient.GetRootFolderAsync(subscriptionId);

   return result.Id;
}
```

   In this example, the function requests the root directory, the workspace ID can be omitted. In this case, the root directory for the user's default workspace will be returned.

3. To export the report, use the following method <xref:FastReport.Cloud.IReportsClient.ExportAsync(System.String,FastReport.Cloud.ExportReportTaskVM,System.Threading.CancellationToken)>.

```
public async Task<string> ExportReport(HttpClient httpClient,
                                string folderId,
                                string reportId,
                                string fileName)
{
   IReportsClient reportsClient = new ReportsClient(httpClient);

   ExportReportVM task = new ExportReportVM()
   {
      FileName = Path.ChangeExtension(fileName, ".pdf"),
      FolderId = folderId,
      Format = ExportReportTaskVMFormat.Pdf,
      ExportParameters = new Dictionary<string, string>
      {
         { "additionalProp1", ""},
         { "additionalProp2", ""},
         { "additionalProp3", ""}
      }
   };

   ExportVM result = await reportsClient.ExportAsync(reportId, task);

   return result.Id;
}
```

   - `FileName` — name of the resulting file. If you do not specify the extension, or specify it incorrectly, the server will replace it by itself.
   - `FolderId` — ID of the directory where the export will be placed. If left blank, the export will be placed in the root folder for exports in the workspace.
   - `Format` — export format.
   - `ExportParameters` — export parameters. They are set similarly to the export parameters from the FastReport .NET library. A more detailed description is stored in the user documentation in the export parameters section.

   In this example, the function creates a task to export a report.

   Important! The report has not been exported yet, but an ID has already been assigned to the export. After some time, the builder's turn will come to this task and the report will be exported.

> If you do not specify FolderId, the export will be saved to the root folder.

4. For information about the file, use the following method <xref:FastReport.Cloud.IExportsClient.GetFileAsync(System.String,System.Threading.CancellationToken)>.

```
public async Task<ExportVMStatus> CheckStatus(HttpClient httpClient, string exportId)
{
   IExportsClient exportsClient = new ExportsClient(httpClient);

   ExportVM result = await exportsClient.GetFileAsync(exportId);

   return result.Status.GetValueOrDefault();
}
```

   In this example, the function requests an export by its ID and returns the status. You must wait for the <xref:FastReport.Cloud.FileStatus.Success> status. Check it every few seconds.

5. Check the status in the loop and download the export.

```
int tries = 10;
FileStatus status;
do
{
status = await CheckStatus(httpClient, reportId);
   tries--;
} while (status != FileStatus.Success && tries > 0);
var report = await DownloadExport(httpClient, reportId);
```

6. Use the following method to download the report <xref:FastReport.Cloud.IDownloadClient.GetExportAsync(System.String,System.Threading.CancellationTok
   n)>.

```
public async Task<byte[]> DownloadExport(HttpClient httpClient, string exportId)
{
   IDownloadClient downloadClient = new DownloadClient(httpClient);

   FileResponse file = await downloadClient.GetExportAsync(exportId);

   using (MemoryStream ms = new MemoryStream())
   {
      file.Stream.CopyTo(ms);

      return ms.ToArray();
   }
}
```

   In this example, the function requests a file and copies it to memory.

What next?

- Work with groups.
- Add new users to the workspace.

Add new users to the workspace

This article covers the process of adding a new user to a subscription, retrieving the list of users, and removing a user from it.

Getting started

You will need the following tools and facilities:

1. Know how to use API key in FastReport Corporate Server.

   This article will skip over additional information on authentication and authorization.

2. .NET SDK.

3. C# code editor or text editor, e.g., Visual Studio Code.

4. Active FastReport Corporate Server subscription that has at least two user slots.

5. Internet access.

> Important! These guides assume that you already know how to develop your application in the C# programming language.

> Note. The items above describe recommended tools.

Annotation

Important! It is only possible to add a user to a workspace by user ID.

Instruction

1. To add a new user to a workspace, a workspace (subscription) ID is required.

   To retrieve the workspace ID, use the following method <xref:FastReport.Cloud.ISubscriptionsClient.GetSubscriptionsAsync(System.Nullable{System.Int32},System.Nullable{System.Int32},System.Threading.CancellationToken)>.

   ```
   public async Task<string> GetSubscriptionId(HttpClient httpClient)
   {
      ISubscriptionsClient subscriptionsClient = new SubscriptionsClient(httpClient);
    SubscriptionsVM subscriptions =
          await subscriptionsClient.GetSubscriptionsAsync(0, 10);
    SubscriptionVM subscription = subscriptions.Subscriptions.First();
      return subscription.Id;
   }
   ```

   In this example, the function requests the first 10 workspaces (subscriptions) from the user's list of workspaces, selects the first workspace, and returns its ID.

   > A workspace always is associated with the one subscription, so they have the same ID.

2. To add a new user, use the following method <xref:FastReport.Cloud.ISubscriptionUsersClient.AddUserAsync(System.String,System.String,System.Threading.CancellationToken)>.

```
public async Task AddUser(HttpClient httpClient, string subscriptionId, string userId)
{
    ISubscriptionUsersClient subscriptionUsersClient =
        new SubscriptionUsersClient(httpClient);
    await subscriptionUsersClient.AddUserAsync(subscriptionId, userId);
}
```

In this example, the function adds the user with the `userId` ID to the workspace with the `subscriptionId` ID.

3. To retrieve the list of workspace users, use the following method <xref:FastReport.Cloud.ISubscriptionUsersClient.GetUsersAsync(System.String,System.Nullable{System.Int32},System.Nullable{System.Int32},System.Threading.CancellationToken)>.

```
public async Task<IEnumerable<string>> GetUsers(HttpClient httpClient, string subscriptionId)
{
    ISubscriptionUsersClient subscriptionUsersClient =
        new SubscriptionUsersClient(httpClient);
  SubscriptionUsersVM users =
        await subscriptionUsersClient.GetUsersAsync(subscriptionId, 0, 10);
  return users.Users.Select(m => m.UserId);
}
```

In this example, the function requests the first 10 users from a workspace with the `subscriptionId` ID.

4. To remove a user from the workspace, use the following method <xref:FastReport.Cloud.ISubscriptionUsersClient.RemoveUserAsync(System.String,System.String,System.Threading.CancellationToken)>.

```
public async Task RemoveUser(HttpClient httpClient, string subscriptionId, string userId)
{
    ISubscriptionUsersClient subscriptionUsersClient =
        new SubscriptionUsersClient(httpClient);
  await subscriptionUsersClient.RemoveUserAsync(subscriptionId, userId);
}
```

In this method, the function removes the user with the `userId` ID from the workspace with the `subscriptionId` ID.

What next?

- Work with groups
- Help and feedback

Work with groups

This article covers the process of creating a new group, adding a user to the group, and getting a list of the group's users.

Getting started

You will need the following tools and facilities:

1. Know how to use API key in FastReport Corporate Server.

   This article will skip over additional information on authentication and authorization.

2. .NET SDK.

3. C# code editor or text editor, e.g., Visual Studio Code.

4. Active FastReport Corporate Server subscription that has at least two user slots.

5. Internet access.

> Important! These guides assume that you already know how to develop your application in the C# programming language.

> Note. The items above describe recommended tools.

Annotation

Important! Adding a user to a group is only possible if the user exists in the workspace.

Important! It is possible to add a user to a group only by their ID.

Instruction

1. To create a new group, you need a workspace ID and the name of the new group.

   To retrieve the workspace ID, use the following method <xref:FastReport.Cloud.ISubscriptionsClient.GetSubscriptionsAsync(System.Nullable{System.Int32},System.Nullable{System.Int32},System.Threading.CancellationToken)>.

   ```
   public async Task<string> GetSubscriptionId(HttpClient httpClient)
   {
      ISubscriptionsClient subscriptionsClient = new SubscriptionsClient(httpClient);

      SubscriptionsVM subscriptions =
       await subscriptionsClient.GetSubscriptionsAsync(0, 10);

      SubscriptionVM subscription = subscriptions.Subscriptions.First();

      return subscription.Id;
   }
   ```

   In this example, the function requests the first 10 workspaces from the user's list of workspaces, selects the first workspace, and returns its ID.

   > The ID of a workspace and a subscription are the same because one subscription is associated with each workspace.

2. To create a new group, use the following method <xref:FastReport.Cloud.IGroupsClient.CreateGroupAsync(FastReport.Cloud.CreateGroupVM,System.Threading.CancellationToken)>.

```
public async Task<string> CreateGroup(HttpClient httpClient,
                                      string subscriptionId,
                                      string groupName)
{
   IGroupsClient groupsClient = new GroupsClient(httpClient);

   CreateGroupVM viewModel = new CreateGroupVM()
   {
      Name = groupName,
      SubscriptionId = subscriptionId
   };

   GroupVM group = await groupsClient.CreateGroupAsync(viewModel);

   return group.Id;
}
```

   In this example, the function creates a new group with the `groupName` name for the workspace with the `subscriptionId` ID, as a result the function will return the ID of the created group.

3. To add a new user to the group, use the following method <xref:FastReport.Cloud.IGroupUsersClient.AddUserToGroupAsync(System.String,System.String,System.Threading.CancellationToken)>.

```
public async Task AddUser(HttpClient httpClient, string groupId, string userId)
{
   IGroupUsersClient groupUsersClient = new GroupUsersClient(httpClient);

   await groupUsersClient.AddUserToGroupAsync(groupId, userId);
}
```

   In this example, the function adds a user with the `userId` ID to the group with `groupId` ID.

4. To retrieve the list of users in the group, use the following method <xref:FastReport.Cloud.IGroupUsersClient.GetUsersInGroupAsync(System.String,System.Nullable{System.Int32},System.Nullable{System.Int32},System.Threading.CancellationToken)>.

```
public async Task<IEnumerable<string>> GetUsers(HttpClient httpClient, string groupId)
{
   IGroupUsersClient groupUsersClient = new GroupUsersClient(httpClient);

   GroupUsersVM users =
    await groupUsersClient.GetUsersInGroupAsync(groupId, 0, 10);

 return users.Users.Select(m => m.UserId);
}
```

   In this example, the function requests the first 10 users from the group with the `groupId`.

What next?

- Help and feedback

Work with tasks

Tasks in FastReport Corporate Server are actions to convert and deliver documents to consumers. They are described in detail in the Tasks section.

Getting started

You will need the following tools and facilities:

1. Know how to use API key in FastReport Corporate Server.

   This article will skip over additional information on authentication and authorization.

2. .NET SDK.

3. C# code editor or text editor, e.g., Visual Studio Code.

4. Report template.

   It can be created in the free FastReport Community Designer program.

5. Active FastReport Corporate Server subscription.

6. Internet access.

> Important! These guides assume that you already know how to develop your application in the C# programming language.

> Note. The items above describe recommended tools.

General information

This article will describe how to work with tasks.

Getting started

You will need the following tools and facilities:

1. Know how to use API key in FastReport Corporate Server.

   This article will skip over additional information on authentication and authorization.

2. .NET SDK.

3. C# code editor or text editor, e.g., Visual Studio Code.

4. Report template.

   It can be generated in the free FastReport Community Designer program.

5. Active FastReport Corporate Server subscription.

6. Internet access.

> Important! These guides assume that you already know how to develop your application in the C# programming language.

> Note. The items above describe recommended tools.

First, HttpClient must be created, which we are going to use below:

```
var httpClient = new HttpClient();
httpClient.BaseAddress = new Uri("{your_host_name}");
httpClient.DefaultRequestHeaders.Authorization = new FastReportCloudApiKeyHeader(ApiKey);
```

Next, we will receive a subscription in which we will work with tasks:

```
var subscriptions = new SubscriptionsClient(httpClient);
var subscription = (await subscriptions.GetSubscriptionsAsync(0, 10)).Subscriptions.FirstOrDefault();
```

Now we need to create a client to work with tasks:

```
TasksClient tasksClient = new TasksClient(httpClient);
```

Finally, you can start working directly with tasks.

Create task

The CreateTask or CreateTaskAsync method should be used to create tasks. It can accept any TaskBaseVM child class:

- CreatePrepareTaskVM
- CreateExportTemplateTaskVM
- CreateExportReportTaskVM
- CreateFetchTaskVM
- CreateEmailTaskVM
- CreateWebhookTaskVM

- CreateFTPUploadTaskVM.

Let’s consider creating a task to generate a report, saving the result to a folder, and then exporting it to PDF:

```
await tasksClient.CreateTaskAsync(new CreatePrepareTaskVM
        {
           Name = "My first task",
           Type = TaskType.Prepare,
           InputFile = new InputFileVM
           {
              EntityId = "{templateId}"
           },
           OutputFile = new OutputFileVM
           {
              FileName = "My first report generated by the task.fpx",
              FolderId = "{report folder ID}"
           },
           Exports = new List<CreateExportReportTaskVM>
           {
              new CreateExportReportTaskVM
              {
                 Format = ExportFormat.Pdf,
                 OutputFile = new OutputFileVM
                 {
                    FileName = "pdffromFpxfromFrx.pdf",
                    FolderId = "{export folder ID}"
                 }
              }
           }
        });
```

Real object identifiers should be written into the EntityId and FolderId fields. Otherwise, the task will be terminated with an error.

> Important! If OutputFile is not specified, it will be saved to a temporary folder. If you specify an empty OutputFile, it will be saved to the root folder. If you specify a folder ID, it will be saved to it.

Retrieve task list

```
// Retrieve the first 100 tasks from the workspace
var tasks = await tasksClient.GetListAsync(0, 100, subscription.Id);
```

Execute a task by the specified ID

```
// Start a task by ID
await tasksClient.RunTaskByIdAsync(id);
```

Delete tasks from the storage

```
// Delete all tasks
foreach(var t in tasks.Tasks)
{
   await tasksClient.DeleteTaskAsync(t.Id);
}
```

Save to FTP server

In this article, we will consider the method of sending a report to an FTP server. Instructions on how to work with tasks are described in the General information section.

Getting started

You will need the following tools and facilities:

1. Know how to use API key in FastReport Corporate Server.

   This article will skip over additional information on authentication and authorization.

2. .NET SDK.
3. C# code editor or text editor, e.g., Visual Studio Code.
4. Report template.

   It can be generated in the free FastReport Community Designer program.

5. Active FastReport Corporate Server subscription.
6. Internet access.
7. Configured and available FTP server.

> Important! These guides assume that you already know how to develop your application in the C# programming language.

> Note. The items above describe recommended tools.

> Important! This guide assumes that you have experience setting up and configuring an FTP server to receive files from external sources.

Create task

Let’s consider creating a task of sending a template to an FTP server and its subsequent launching:

```
// Object initialization
 CreateFTPUploadTaskVM ftpUploadTaskVM = new CreateFTPUploadTaskVM
 {
    Name = "Task of sending to FTP",
    InputFile = new InputFileVM
    {
       EntityId = "{template ID}",
       Type = FileKind.Template
    },
    FtpHost = "{FTP server address}",
    FtpPort = 21,
    FtpUsername = "{FTP server user name}",
    FtpPassword = "{password}",
    UseSFTP = false,
    DestinationFolder = "{/older_path/}",
    Archive = false,
    ArchiveName = "Archive name",
    SubscriptionId = "{workspace ID}"
 };

// Create tasks
 TaskBaseVM ftpUploadTask = await tasksClient.CreateTaskAsync(ftpUploadTaskVM);

// Start a task by ID
 await tasksClient.RunTaskByIdAsync(ftpUploadTask.Id);
```

Real object IDs should be written to the EntityId and SubscriptionId fields. Otherwise, the task will be terminated with an error.

Execute task from the request body

```
// Start a task from the request body
 await tasksClient.RunTaskAsync(new RunFTPUploadTaskVM
 {
    InputFile = new RunInputFileVM
    {
       EntityId = "{template ID}",
       Type = FileKind.Template
    },
    FtpHost = "{FTP server address}",
    FtpPort = 21,
    FtpUsername = "{FTP server username}",
    FtpPassword = "{password}",
    UseSFTP = false,
    DestinationFolder = "{/folder_path/}",
    Archive = false,
    ArchiveName = "Archived template",
    SubscriptionId = "{workspace ID}"
 });
```

> Important! In this case, this request will send to FTP directly and the task will not be saved in the database.

Update task by ID

```
await tasksClient.UpdateTaskAsync("{old task ID}", new UpdateFTPUploadTaskVM()
{
    InputFile = new RunInputFileVM
    {
        EntityId = "{updated template ID}",
        Type = FileKind.Template
    },
    FtpHost = "{updated FTP server address}",
    FtpPort = 21,
    FtpUsername = "{updated FTP server username}",
    FtpPassword = "{password}",
    UseSFTP = false,
    DestinationFolder = "/updated_folder_path",
    Archive = false,
    ArchiveName = "updated archive name"
});
```

Save over Webhook

In this article, let's look at how to send a webhook report. Instructions on how to work with tasks are described in the General information section.

Getting started

You will need the following tools and facilities:

1. Know how to use API key in FastReport Corporate Server.

   This article will skip over additional information on authentication and authorization.

2. .NET SDK.
3. C# code editor or text editor, e.g., Visual Studio Code.
4. Report template.

   It can be generated in the free FastReport Community Designer program.

5. Active FastReport Corporate Server subscription.
6. Internet access.
7. Client application receiving requests.

> Important! These guides assume that you already know how to develop your application in the C# programming language.

> Note. The items above describe recommended tools.

> Important! This guide assumes that you have a client application, that receives and processes incoming webhooks.

Create task

Let’s look at creating a task to send a template over webhook and then running it:

```
// Object initialization
CreateWebhookTaskVM webhookTaskVM = new CreateWebhookTaskVM
{
   Name = "Webhook send task",
   InputFile = new InputFileVM
   {
      EntityId = "{template ID}",
      Type = FileKind.Template
   },
   Url = new Uri("https://example.com/"),
   Headers = new Dictionary<string, string> {
      { "Authorization", "Bearer <token>" },
      { "Content-Length", "1238" }
   },
   SubscriptionId = "{workspace ID}"
};

 // Create a task
 TaskBaseVM webhookTask = await tasksClient.CreateTaskAsync(webhookTaskVM);

 // Start a task by ID
 await tasksClient.RunTaskByIdAsync(webhookTask.Id);
```

> Important! Real object identifiers should be written to the EntityId and SubscriptionId fields. Otherwise, the task will be terminated with an error.

> Important! The EntityId can be passed in the template, report, and export ID.

Execute task from the request body

```
// Run a task from the request body
 await tasksClient.RunTaskAsync(new RunWebhookTaskVM
 {
    InputFile = new RunInputFileVM
    {
       EntityId = "{template ID}",
       Type = FileKind.Template
    },
    Url = new Uri("https://example.com/"),
    Headers = new Dictionary<string, string> {
       { "Authorization", "Bearer <token>" },
       { "Content-Type", "multipart/form-data" }
    },
    SubscriptionId = "{workspace ID}"
 });
```

> Important! In this case, this request will send the webhook and the task will not be saved in the database.

Update task by ID

```
await tasksClient.UpdateTaskAsync("{old task ID}", new UpdateWebhookTaskVM()
 {
    InputFile = new RunInputFileVM
    {
       EntityId = "{updated template ID}",
       Type = FileKind.Template
    },
    Url = new Uri("{updated address}"),
    Headers = new Dictionary<string, string> {
       { "{updated header}", "{updated value}" },
    }
 });
```

Example request that will come to the webhook endpoint:


```
{
  "startedDateTime": "2024-06-07 08:37:09",
  "request": {
    "method": "POST",
    "url": "https://example.com/6e259560-25e5-482b-a7b2-8c5267ba6ae3/",
    "headers": [
      {
        "name": "connection",
        "value": "close"
      },
      {
        "name": "content-length",
        "value": "1421"
      },
      {
        "name": "content-type",
        "value": "multipart/form-data; boundary=\"62ec6524-9360-4e91-834f-e9531e2d2c30\""
      },
      {
        "name": "host",
        "value": "example.com"
      }
    ],
    "bodySize": 0,
    "postData": {
      "mimeType": "application/json",
      "text": ""
    }
  },
  "response": {
    "status": 200,
    "httpVersion": "HTTP/1.1",
    "headers": [
      {
        "name": "Content-Type",
        "value": "text/html"
      }
    ],
    "content": {
      "size": 145,
      "text": "This URL has no default content configured.",
      "mimeType": "text/html"
    }
  }
}
```