



Руководство программиста Сервисных решений

Предисловие

Любые фрагменты кода, которые могут быть встречены в этой документации, являются только демонстрационными примерами. Эти примеры служат только одной цели - проиллюстрировать применение продуктов МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер и МоиОтчеты Публикатор. Компания "Быстрые отчеты" не гарантирует правильность и полноту данных примеров и не несет ответственности за прямой или косвенный ущерб, который может быть ими принесен.

Документация может содержать гиперссылки на различные ресурсы в сети Интернет. Эти ссылки актуальны на момент написания документации. Компания "Быстрые отчеты" не несет ответственности за их доступность на момент прочтения документации или нанесенный ими ущерб.

Руководства

Этот раздел посвящен руководствам по работе с облаком и интеграции МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер и МоиОтчеты Публикатор в приложение. На данный момент доступны руководства для:

- [REST API](#)
- [C#/.NET](#)

Руководства для использования с другими языками программирования в планах.

Напишите нам если вас интересует какой-то конкретный язык программирования или платформа.[Написать!](#)

Инструменты разработки

Доступны SDK для следующих языков:

- [Haskell](#)
- [JavaScript](#)
- [C++](#)
- [Python](#)
- [Java](#)
- [Go](#)
- [C#/.NET](#) и [ASP.NET](#)

Обратите внимание, инструменты разработки выложены исходниками на Github, чтобы их использовать, из исходников необходимо будет собрать пакет или библиотеку самостоятельно.

REST API

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этом разделе приведены пошаговые инструкции и руководства для выполнения типовых задач по использованию Сервисных решений без привязки к конкретному языку программирования.

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Инструмент `curl`.

Подойдёт любой другой REST клиент, но примеры будут построены для `curl`.

2. Дизайнер отчётов [FastReport Community Designer](#).

Для некоторых руководств нужен будет дизайнер для создания отчёта.

3. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор.

4. Доступ в интернет.

Обратите внимание! Эти руководства рассчитаны на то, что вы уже знаете, как использовать `curl` или использовать другой REST клиент.

Примечание. Пункты выше описывают рекомендуемые инструменты.

Что дальше?

Советуем начать своё ознакомление со статей:

- [Статус коды](#).
- [Аутентификация и авторизация](#).

Статус коды

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье перечислены все статус-коды, которые могут быть возвращены сервисами МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер и МоиОтчеты Публикатор.

Запрос успешно обработан — 2xx

- 200 OK - запрос обработан успешно, используется в большинстве случаев, когда вместе с результатом возвращено тело ответа.
- 201 Created - запрос обработан успешно, в результате его выполнения была создана новая сущность.
- 204 No Content - запрос обработан успешно, возвращено пустое тело ответа.

Ошибка пользователя — 4xx

- 400 Bad Request - запрос содержит в себе ошибки, например:
 - Отсутствует параметр, необходимый для выполнения запроса.
 - Формат `Id` сущности отличается от Hex 24.
 - В запросе отрицательный параметр `skip` или `take`.
- 401 Unauthorized - запрос пришёл от неавторизованного пользователя.
- 402 Payment Required - запрашиваемый ресурс ограничен по плану подписки.
- 403 Forbidden - запрашиваемый ресурс найден, но у пользователя нет прав на действие.
- 404 Not Found - запрашиваемый ресурс не найден.
- 413 Request Entity Too Large - тело запроса превышает лимиты в настройках сервиса.

Ошибка сервера — 5xx

- 500 Internal Server Error - запрос прошёл все проверки на валидность, но в процессе его выполнения возникло исключение Exception. В этом случае просим сообщить, написав по адресу - support@fastreport.ru

Другое

Промежуточные сервисы между Сервисным решением и пользователем могут возвращать другие статус коды.

Аутентификация и авторизация

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

Процесс проверки подлинности пользовательских данных и выдача пользователю определённых прав в Сервисных решениях осуществляется по одному из двух доступных способов:

1. Через JWT token.

В этом случае аутентификацию нужно пройти лично, а токен будет действовать только 5 минут, за которые пользователь должен войти в своё приложение. При подключении к серверу браузер перенаправит на сервер аутентификации, после чего сгенерирует токен доступа. С точки зрения безопасности мы ограничиваем возможность получения JWT токена только лично пользователем.

Если в течении 5 минут пользователь не зашёл в приложение, то аутентификацию необходимо пройти заново. Если пользователь зашёл в приложение, повторная аутентификация не требуется.

2. Через API key.

В этом случае процесс получения прав доступа осуществляется для серверных приложений. Для получения ключа доступа (API key) необходимо присутствие пользователя. Однако сам ключ может действовать продолжительное время, например год.

Получение первого API key

Для получения первого API key воспользуйтесь [пользовательской панелью](#). Если по какой-то причине доступа к пользовательской панели нет, можно запросить ключ по описанию ниже.

1. Откройте ссылку в браузере: `<{host_name}/account/signin?r={host_name}/api/manage/v1/ApiKeys>`.

Переход по этой ссылке направит вас на автоматический процесс аутентификации браузера.

2. Теперь, когда аутентификация пройдена, необходимо запросить новый ключ.

Нажмите `F12` или `Ctrl+Shift+I`, чтобы открыть панель разработчика. Сочетание клавиш могут отличаться от стандартных, в этом случае откройте панель разработчика через меню браузера.

3. Скопируйте и выполните код в консоли JavaScript.

Этот код сделает `POST` запрос на URL `{host_name}/api/manage/v1/ApiKeys` для создания нового ключа доступа до 2030 года.

4. Обновите страницу браузера и заберите результат.

```
{
  "apiKeys": [
    {
      "value": "cc355oeu1z5d5wncayo33me6c1g5junqdk4pkupid7t8ynjshey",
      "description": "Generated by js develop panel",
      "expired": "2030-01-01T07:41:23.399Z"
    }
  ],
  "count": 1
}
```

Также ключ можно создать, открыв вкладку API ключи.

Теперь вы можете использовать API key, в примере выше был использован `быстрыеотчеты.рф`

```
cc355oeu1z5d5wncayo33me6c1g5junqdqk4pkupid7t8ynjshey
```

Повторно получать новый API key через браузер нет необходимости.

Как использовать API key

Ключ следует передавать с каждым запросом в заголовке `Authorization: Basic`. В качестве имени пользователя следует использовать `apikey`, а в качестве пароля - значение ключа. Например:

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

Где `Base64Encode` это функция преобразования строки в `base64` при использовании кодировки `UTF8`.

Получение нового API key

Для получения нового ключа сделайте `POST` запрос на точку входа `{host_name}/api/manage/v1/ApiKeys` и передайте JSON в теле запроса по схеме ниже.

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

Пример запроса.

```
curl -X POST "{host_name}/api/manage/v1/ApiKeys" -H "accept: text/plain" -H "authorization: Basic YXBpa2V5OmNjMzU1b2V1MXo1ZDV3bmNheW8zM21INmMxZzVqdW5xZHFrNHBrdXBpZDd0OHluanNoZXk=" -H "Content-Type: application/json-patch+json" -d '{"description": "Generated by js develop panel", "expired": "2030-01-01T07:41:23.399Z"}'
```

Схема ответа.

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

Также получить новый ключ можно через пользовательскую панель.

Что дальше?

- [Загрузка нового шаблона.](#)
- [Работа с группами.](#)
- [Добавление новых пользователей в рабочее пространство.](#)

Загрузка нового шаблона

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

Одна из основных возможностей Сервисных решений — это хранение шаблонов, отчётов и других данных в облаке. В этой статье будет рассмотрен способ загрузить свой готовый шаблон в хранилище шаблонов сервисного решения.

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. Инструмент curl.

Подойдёт любой другой REST клиент, но примеры будут построены для curl.

3. Шаблон отчёта.

Его можно подготовить с помощью бесплатной программы [FastReport Community Designer](#).

4. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор.

5. Доступ в интернет.

Инструкция

1. Вам необходимо получить идентификатор корневой папки рабочего пространства. Этот идентификатор никогда не будет меняться, поэтому его можно сохранить и не делать запрос каждый раз перед загрузкой нового шаблона.

Что бы запросить корневую директорию выполните GET запрос на `{host_name}/api/rp/v1/Templates/Root`.

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

В этом случае будет возвращена директория для рабочего пространство по умолчанию. Можно указать конкретное рабочее пространство передав параметр `subscriptionId`.

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

Пример ответа.


```
{
  "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. Теперь этот шаблон можно использовать для подготовки (построения) отчёта и экспорта.

Что дальше?

- [Управление правами доступа на примере шаблона.](#)
- [Построение отчёта.](#)

Добавление источника данных JSON

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье будет рассмотрен способ создать новый источник данных на основе JSON в Сервисных решениях.

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. Инструмент curl.

Подойдёт любой другой REST клиент, но примеры будут построены для curl.

3. JSON файл.

4. JSON схема.

5. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор.

6. Доступ в интернет.

Создание источника данных

Для того, чтобы создать источник данных, отправьте `CREATE` запрос на `{host_name}/api/data/v1/DataSources`.

Пример тела запроса:

```
{
  "name": "fakeAPI",
  "connectionString":
  "Json=aHR0cHM6Ly9qc29ucGxhY2Vob2xkZXludHlwZWVudGZlLnV2Y29tL3Bvc3Rz;JsonSchema=eyJ0eXBlljoib2JqZW50Iiwic2VudGZlLnV2Y29tL3Bvc3Rz;JsonSchema=eyJ0eXBlljoib2JqZW50Iiwic2VudGZlLnV2Y29tL3Bvc3Rz;Encoding=utf-8",
  "subscriptionId": "604f52e1261a3c19104c0e25",
  "connectionType": "JSON"
}
```

Где

- `name` — имя источника данных (будет отображаться в дизайнера при выборе).
- `connectionString` — строка подключения, в случае с JSON она состоит из 3-х элементов:
 - `Json` — JSON файл или http/https ссылка, закодированная в base64;
 - `JsonSchema` — схема, описывающая структуру JSON файла, закодированная в base64;
 - `Encoding` — кодировка, стоит всегда передавать `utf-8`.
- `subscriptionId` — id рабочего пространства (подписки), к которому будет прикреплен источник данных.
- `connectionType` — тип подключения, в этом руководстве используется `JSON`.

Пример запроса.

```
curl -X POST "{host_name}/api/data/v1/DataSources" -H "accept: text/plain" -H "Content-Type: application/json-patch+json" -d '{"name":
"fakeAPI", "connectionString":
"Json=aHR0cHM6Ly9qc29ucGxhY2Vob2xkZXludHlwZWVudGZlLnVzL3Bvc3Rz;JsonSchema=eyJ0eXBlljoiYXJyYXkiLCJpdGVtcyl6eyJ0eXB
Blljoib2JqZWNoIiwicHJvcGVydGllcyI6eyJ1c2VySWQiOnsidHlwZSI6Im51bWJlciJ9LCJpZCI6eyJ0eXBlljoibnVtYmVyn0slnRpdGxllj7InR5cGU
iOiJzdHJpbmciSwiYm9keSI6eyJ0eXBlljoic3RyaW5nIn19fX0=;Encoding=utf-8", "subscriptionId": "604f52e1261a3c19104c0e25",
"connectionType": "JSON"}'
```

Пример ответа.

```
{
  "id": "60648953db44d83f9c6da98f",
  "name": "fakeAPI",
  "connectionType": "JSON",
  "connectionString":
"Json=aHR0cHM6Ly9qc29ucGxhY2Vob2xkZXludHlwZWVudGZlLnVzL3Bvc3Rz;JsonSchema=eyJ0eXBlljoiYXJyYXkiLCJpdGVtcyl6eyJ0eXB
lljoib2JqZWNoIiwicHJvcGVydGllcyI6eyJ1c2VySWQiOnsidHlwZSI6Im51bWJlciJ9LCJpZCI6eyJ0eXBlljoibnVtYmVyn0slnRpdGxllj7InR5cGU
iOiJzdHJpbmciSwiYm9keSI6eyJ0eXBlljoic3RyaW5nIn19fX0=;Encoding=utf-8",
  "dataStructure": "<JsonDataSourceConnection
ConnectionString=\\rjcmIqrcq6OJBTPt0pNFvRuRtDUSHSHLQy/QINolifTTaTjsrExzdbf1ifpPblp655sducwkD1IEVzxVZF8qRuE0NT6UkyTr7kwj
GitFOwh7DBsOyL6QkQY4FOZ2ki8AI2R30gpXs6nMUGg1BRwCF0rj3+QvmXbj+2t8x5RerR5y7inP1R+oCuo0wvfcTeOMfYfZrjdE3whziFh5Qn
3mR7vaevmV9peDWQ3LYyK2ec3KpGveEXSqM+10Wyl4ahY7EHuQlzlZROFGKfW50cUYwdillhKy24gNdsUzi5klG66DDQtCKEOLbNutDv
A0xqCTW3MvRNORSbvckL6g3gM+cStJ5PQ2XUjF9yz9zdwrmrnXl6k+MK8V9lrMkc0XFkDMHOxDlfG2jHhkFuUTgmiKp7hQMg==\\>\\r\\n
<JsonTableDataSource Name=\\JSON" DataType=\\FastReport.Data.JsonConnection.JsonParser.JsonArray\\ Enabled=\\true"
TableName=\\JSON\\>\\r\\n <Column Name=\\index\\>\\r\\n <Column Name=\\item\\
DataType=\\FastReport.JsonBase\\>\\r\\n <Column Name=\\userId\\>\\r\\n <Column Name=\\id\\
DataType=\\System.Double\\>\\r\\n <Column Name=\\title\\>\\r\\n <Column Name=\\body\\
DataType=\\System.String\\>\\r\\n </Column>\\r\\n <Column Name=\\array\\>\\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
}
```

Управление правами доступа на примере шаблона

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

Ограничение доступа к приватным ресурсам очень важная возможность Сервисных решений. Гибкая система доступа позволяет задать ограничение или выдать права на каждый ресурс отдельно, задавая круг лиц, которые могут получить доступ.

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. Инструмент curl.

Подойдёт любой другой REST клиент, но примеры будут построены для curl.

3. Шаблон отчёта.

О том, как загрузить шаблон отчёта, можно узнать в статье [Загрузка нового шаблона](#).

4. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор.

5. Доступ в интернет.

Инструкция

1. Для просмотра текущих прав на ресурс выполните GET запрос на `{host_name}/api/rp/v1/Templates/File/{id}/permissions`, где вместо `{id}` следует использовать идентификатор шаблона.

Пример запроса.

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

Пример ответа.

```

{
  "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
  }
}

```

В этом примере владелец и остальные участники рабочего пространства имеют полный доступ к шаблону отчёта.

- Для того чтобы изменить разрешения сделайте `POST` запрос на `{host_name}/api/rp/v1/Templates/File/{id}/permissions`, где вместо `{id}` следует использовать идентификатор шаблона.

Пример модели.

```

{
  "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"
  }
}

```

Обратите внимание! В запросе на обновление прав в свойстве `administrate` были переданы значения `Owner` и `Other`, в этом случае будут обновлены только указанные категории прав, т.е. `owner` и `other`, а категории `anon` и `groups` изменены не будут.

Пример запроса, который убирает у владельца право на обновление отчёта (например - редактирование в онлайн дизайнера), а также все права на файл у остальных участников рабочего пространства.

```
curl -X POST "{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"}
```

Пример ответа.

```
200 OK
```

3. Теперь можно снова запросить права, чтобы убедиться, что они действительно обновились.

Пример запроса.

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

Пример ответа.

```
{
  "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
  }
}
```

Что дальше?

- [Построение отчёта.](#)
- [Работа с группами.](#)
- [Добавление новых пользователей в рабочее пространство.](#)

Построение отчёта

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрен процесс построения отчёта из шаблона с помощью процессора отчётов Сервисных решений.

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. Инструмент curl.

Подойдёт любой другой REST клиент, но примеры будут построены для curl.

3. Шаблон отчёта.

Как загрузить шаблон отчёта можно узнать в статье [Загрузка нового шаблона](#).

4. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор.

5. Доступ в интернет.

Инструкция

1. Вам необходим идентификатор шаблона для его построения. Сделайте GET запрос на `{host_name}/api/rp/v1/Templates/Root` для получения корневой директории.

Пример запроса.

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

Пример ответа.

```
{
  "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"
}
```

Идентификатор директории из примера выше `5fa919f9292a8300019349b9`.

2. Для получения списка файлов в директории выполните GET запрос на `{host_name}/api/rp/v1/Templates/Folder/{id}/ListFiles?skip=0&take=10`, где вместо `{id}` следует подставить идентификатор

директории.

Пример запроса.

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

Пример ответа.

```
[
  {
    "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"
  }
]
```

Идентификатор шаблона из примера выше `5fc9ece6b792c90001d94b13`.

3. Для построения отчёта вам понадобится директория, в которую можно положить отчёт. Запросите корневую директорию отчётов, для этого сделайте `GET` запрос на `{host_name}/api/rp/v1/Reports/Root`.

Пример запроса.

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

Пример ответа.

```
{
  "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
}
```

Идентификатор директории из примера выше `5fa919f9292a8300019349ba`.

4. Для построения отчёта сделайте `POST` запрос на `{host_name}/api/rp/v1/Templates/File/{id}/Prepare`, где вместо `{id}` следует подставить идентификатор шаблона.

В теле запроса передайте JSON по схеме ниже.

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

- `folderId` — идентификатор директории куда будет помещён отчёт.
- `name` — название результирующего файла. Добавьте расширение `.fpx` вручную.
- `reportParameters` — параметры, которые указаны в самом отчёте с помощью дизайнера отчёта, в этом примере они не нужны.

Если не указать `folderId`, то подготовленный отчёт будет сохранён в корневую папку.

Пример запроса.

```
curl -X POST "{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"}'
```

Пример ответа.

```
{
  "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"
}
```

Обратите внимание на статус файла `InQueue`, он означает, что была создана задача на построение и в дальнейшем попала в очередь построителя. Уже на этом этапе отчёт получил свой идентификатор для работы `5fe4614bcd7c55000148e4c6`.

Следует подождать некоторое время, пока отчёт будет построен. Можно вызывать метод получения отчёта в цикле каждые несколько миллисекунд и проверять статус.

- Для получения информации об отчёте сделайте `GET` запрос на `{host_name}/api/rp/v1/Reports/File/{id}`, где вместо `{id}` следует указать идентификатор отчёта.

Пример запроса.

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

Пример ответа.

```
{
  "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"
}
```

Обратите внимание на статус файла `Success`. Отчёт был успешно построен.

- Для скачивания отчёта сделайте `GET` запрос на `{host_name}/download/r/{id}`, где вместо `{id}` следует передать идентификатор отчёта.

Пример запроса.

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

В ответе будет получен файл.

Что дальше?

- [Экспорт отчёта в PDF.](#)

Обратите внимание! В примерах может не быть заголовка Authorization потому что используется модель аутентификации, основанная на cookie. Подробнее про авторизацию читайте в разделе [Аутентификация и авторизация](#)

Параметры отчёта

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрен процесс передачи параметров в отчёт, которые представляют собой словарь параметров, предоставляемых при формировании отчёта. Больше информации на эту тему можно найти в руководстве FastReport .NET в разделе [Параметры отчёта](#).

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. Инструмент curl.

Подойдёт любой другой REST клиент, но примеры будут построены для curl.

3. Шаблон отчёта с указанными параметрами отчёта.

Его можно подготовить с помощью бесплатной программы [FastReport Community Designer](#).

4. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор.

5. Доступ в интернет.

Передача параметров в отчёт

Параметры могут быть переданы в следующих случаях:

- при подготовке отчёта;
- при экспорте из шаблона;
- при работе с задачами.

Рассмотрим передачу параметров на примере подготовки отчёта. Подробная инструкция о том, как подготовить отчёт, содержится в разделе [Построение отчёта](#).

Для построения отчёта сделайте `POST` запрос на `{host_name}/api/rp/v1/Templates/File/{id}/Prepare`, где вместо `{id}` следует подставить идентификатор шаблона.

В теле запроса передайте JSON по схеме ниже.

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

- `folderId` — идентификатор директории куда будет помещён отчёт.
- `name` — название результирующего файла. Добавьте расширение `.fpx` вручную.
- `reportParameters` — параметры, которые указаны в самом отчёте с помощью дизайнера отчёта.

Если не указать folderId, то подготовленный отчёт будет сохранён в корневую папку.

Пример запроса.

```
curl -X POST "{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": "Значение1",  
    "Parameter2": "Значение2"  
  }  
}"
```

Пример ответа.

```
{  
  "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"  
}
```

Экспорт отчёта в PDF

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрен процесс экспорта отчёта с помощью процессора отчётов Сервисных решений.

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях. Как получить и использовать API key можно узнать в статье [Аутентификация и авторизация](#)

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. Инструмент curl.

Подойдёт любой другой REST клиент, но примеры будут построены для curl.

3. Отчёт.

Как построить отчёт можно узнать в статье [Построение отчёта](#).

4. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор.

5. Доступ в интернет.

Замечание

Обратите внимание, что экспорт отчёта можно сделать напрямую из шаблона, без промежуточного сохранения отчёта. Для этого выполните те же команды для шаблона отчёта, заменив `Report` в строках запроса на `Template`, также используйте идентификатор шаблона, а не отчёта.

Инструкция

1. Вам будет необходим идентификатор отчёта для экспорта в PDF. Сделайте `GET` запрос на `{host_name}/api/rp/v1/Report/Root` для получения корневой директории.

Пример запроса.

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

Пример ответа.

```
{
  "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
}
```

Идентификатор директории из примера выше `5fa919f9292a8300019349ba`.

- Получите список файлов в директории, для этого выполните `GET` запрос на `{host_name}/api/rp/v1/Reports/Folder/{id}/ListFiles?skip=0&take=10`, где вместо `{id}` следует подставить идентификатор директории.

Пример запроса.

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

Пример ответа.

```
[
  {
    "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"
  }
]
```

Идентификатор отчёта из примера выше `5fe4614bcd7c55000148e4c6`.

- Для экспорта отчёта понадобится директория, в которую следует положить файл экспорта.

Получите корневую директорию экспортов, для этого сделайте `GET` запрос на `{host_name}/api/rp/v1/Exports/Root`.

Пример запроса.

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

Пример ответа.

```
{
  "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
}
```

Идентификатор директории из примера выше `5fa919fa292a8300019349bb`.

4. Для экспорта отчёта сделайте `POST` запрос на `{host_name}/api/rp/v1/Reports/File/{id}/Export`, где вместо `{id}` следует подставить идентификатор отчёта.

В теле запроса передайте JSON по схеме ниже.

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

- `folderId` — идентификатор директории куда будет помещён экспорт. Если оставить пустым, то экспорт будет помещён в корневую папку для экспортов в рабочем пространстве.
- `fileName` — название результирующего файла. Если не указать расширение, или указать его неправильно - сервер заменит его самостоятельно.
- `locale` — локализация экспортированного отчёта. Эта опция поменяет форматы даты и чисел, на те, что соответствуют выбранному ISO - коду культуры (например - французский (Швейцария) выглядит так - "fr-CH"). Если оставить это поле пустым или указать несуществующую культуру, то подставится локаль по умолчанию из подписки или английский (США), если локаль по умолчанию не указана.
- `format` — формат экспорта.
- `exportParameters` — параметры экспорта. Задаются аналогично параметрам экспорта из библиотеки FastReport .NET. Более подробное описание хранится в документации пользователя в разделе [параметры экспорта](#).

Если не указать `folderId`, то подготовленный отчёт будет сохранён в корневую папку.

Пример запроса.

```
curl -X POST "{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\"}"
```

Пример ответа.

```
{
  "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"
}
```

Обратите внимание на статус файла `InQueue`, он означает, что была создана задача на экспорт и она попала в очередь построителя. Уже на этом этапе файл получил свой идентификатор для работы `5fe46a33cd7c55000148e4c7`.

Следует подождать некоторое время, пока экспорт будет построен.

- Для получения информации об файле сделайте `GET` запрос на `{host_name}/api/rp/v1/Exports/File/{id}`, где вместо `{id}` следует указать идентификатор экспорта.

Пример запроса.

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

Пример ответа.

```
{
  "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"
}
```

Обратите внимание на статус файла `Success`. Отчёт был успешно экспортирован.

- Для скачивания отчёта сделайте `GET` запрос на `{host_name}/download/e/{id}`, где вместо `{id}` следует передать идентификатор отчёта.

Пример запроса.

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

В ответе будет получен файл.

Что дальше?

- [Работа с группами.](#)
- [Добавление новых пользователей в рабочее пространство.](#)

Обратите внимание! В примерах может не быть заголовка Authorization потому что используется модель аутентификации, основанная на cookie. Подробнее про авторизацию читайте в разделе [Аутентификация и авторизация](#)

Добавление новых пользователей в рабочее пространство

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрен процесс добавления нового пользователя в рабочее пространство, получения списка пользователей в рабочем пространстве и удаление пользователя из рабочего пространства.

К каждому рабочему пространству всегда привязана подписка. Они имеют общий идентификатор.

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. Инструмент curl.

Подойдёт любой другой REST клиент, но примеры будут построены для curl.

3. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор, в которой есть хотя бы два слота для пользователя.

4. Доступ в интернет.

Инструкция

Есть два способа добавить пользователя в рабочее пространство:

- Напрямую, указав id пользователя.
- С помощью системы приглашений.

Добавляем пользователя через приглашение

1. Для создания приглашения необходим идентификатор рабочего пространства.

Получите идентификатор рабочего пространства сделав GET запрос на

```
{host_name}/api/manage/v1/Subscriptions?skip=0&take=10.
```

Пример запроса.

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

Пример ответа.

```

{
  "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://быстрыеотчеты.рф",
          "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
}

```

Идентификатор рабочего пространства (подписки) из примера выше `604f52e1261a3c19104c0e25`.

- Для создания приглашения выполните `POST` запрос на `{host_name}/api/manage/v1/Subscriptions/{subscriptionId}/invite`, где вместо `{subscriptionId}` нужно будет указать идентификатор рабочего пространства.

Указав в теле запроса свойства приглашения.

Пример тела.

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

- `usages` означает возможное количество использований,
- `durable` установленный в `true` указывает, что **количество** использований не ограничено,
- `expiredDate` указывает на срок окончания действия приглашение (свойство `durable` не делает приглашение вечно действующим).

Если оставить тело пустым, создастся одноразовое приглашение, которое перестает работать на следующий день.

Пример запроса.

```
curl -X POST "{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\"}"
```

Пример ответа.

```

{
  "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://быстрыеотчеты.рф",
      "unlimitedPage": true,
      "pageLimit": 0
    }
  },
  "old": null,
  "invites": [
    {
      "usages": 1,
      "durable": true,
      "accessToken": "fj3534g341ir7dyytfaap9z1r",
      "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
  }
}

```

В подписке появилось приглашение с токеном доступа `fj3534g341ir7dyytfaap9z1r`.

3. Нужно передать ссылку на принятие приглашения (или сам токен доступа) пользователю, которого вы хотите пригласить (любым удобным для вас способом). Итоговая ссылка будет выглядеть так - `{host_name}/api/manage/v1/Subscriptions/{subscriptionId}/invite/{accessToken}/accept`, где вместо `{subscriptionId}` нужно будет указать идентификатор рабочего пространства, а вместо `{accessToken}` - токен доступа.
4. Теперь приглашаемый пользователь может перейти по этой ссылке прямо в браузере или же отправить `GET` запрос на `{host_name}/api/manage/v1/Subscriptions/{subscriptionId}/invite/{accessToken}/accept`, где вместо `{subscriptionId}`

нужно будет указать идентификатор рабочего пространства, а вместо `{accessToken}` - токен доступа.

Пример запроса.

```
curl -X GET "https://xn--80ab2acne.xn--e1afliby2b0b.xn--p1ai/api/manage/v1/Subscriptions/6051f2a06c07a10001737394/invite/to9kxrxdz4iwbfyz3pq4fktocr" -H "accept: text/plain"
```

Удаление приглашения

- Когда у приглашения заканчивается использования, или срок его действия подходит к концу, оно не удаляется автоматически. Нужно будет сделать это вручную, отправив `DELETE` запрос на `{host_name}/api/manage/v1/Subscriptions/{subscriptionId}/invite/{accessToken}` Пример запроса.

```
curl -X DELETE "https://xn--80ab2acne.xn--e1afliby2b0b.xn--p1ai/api/manage/v1/Subscriptions/6051f2a06c07a10001737394/invite/to9kxrxdz4iwbfyz3pq4fktocr" -H "accept: text/plain"
```

В ответ будет получено пустое сообщение с кодом `NO CONTENT 204`.

Добавляем пользователя напрямую

1. Для добавления нового пользователя в рабочее пространство необходим идентификатор рабочего пространства.

Получите идентификатор подписки сделав `GET` запрос на `{host_name}/api/manage/v1/Subscriptions?skip=0&take=10`.

Пример запроса.

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

Пример ответа.

```

{
  "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://быстрыеотчеты.рф",
          "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
}

```

Идентификатор рабочего пространства (подписки) из примера выше `5fa919fa292a8300019349bc`.

2. Для добавления нового пользователя сделайте `PUT` запрос

`{host_name}/api/manage/v1/Subscriptions/{subscriptionId}/users/{userId}`, где вместо `{subscriptionId}` следует передать идентификатор рабочего пространства, а вместо `{userId}` следует передать идентификатор пользователя.

Пример запроса.

```

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

```

В ответ вы получите пустое сообщение со статус кодом `OK 200`.

3. Для получения списка пользователей сделайте `GET` запрос на `{host_name}/api/manage/v1/Subscriptions/{id}/users?skip=0&take=10`, где место `{id}` следует указать идентификатор рабочего пространства.

Пример запроса.

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

Пример ответа.

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

4. Для удаления пользователя из рабочего пространства сделайте `DELETE` запрос на `{host_name}/api/manage/v1/Subscriptions/{subscriptionId}/users/{userId}`, где вместо `{subscriptionId}` следует передать идентификатор рабочего пространства, а вместо `{userId}` следует передать идентификатор пользователя.

Пример запроса.

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

В ответ будет получено пустое сообщение с кодом `OK 200`.

Что дальше?

- [Работа с группами](#)
- [Помощь и обратная связь](#)

Работа с группами

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрен процесс создания новой группы, добавление пользователя в группу и получение списка пользователей группы.

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. Инструмент curl.

Подойдёт любой другой REST клиент, но примеры будут построены для curl.

3. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор, в которой есть два слота для пользователя.

4. Доступ в интернет.

Замечание

Обратите внимание! Добавить пользователя в группу возможно только если пользователь существует в рабочем пространстве.

Обратите внимание! Добавить пользователя в группу возможно только по его идентификатору.

Инструкция

1. Для создания новой группы необходим идентификатор рабочего пространства и название новой группы.

Получите идентификатор рабочего пространства сделав GET запрос на `{host_name}/api/manage/v1/Subscriptions?skip=0&take=10`.

Пример запроса.

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

Пример ответа.

```

{
  "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://быстрыеотчеты.рф",
          "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
}

```

Идентификатор рабочего пространства (подписки) из примера выше `5fa919fa292a8300019349bc`.

- Для создания новой группы сделайте `POST` запрос `{host_name}/api/manage/v1/Groups`, в тело запроса передайте JSON по схеме ниже.

```

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

```

Пример запроса.

```
curl -X POST "{host_name}/api/manage/v1/Groups" -H "accept: text/plain" -H "Content-Type: application/json-patch+json" -d '{"name": "Моя первая группа", "subscriptionId": "5fa919fa292a8300019349bc"}'
```

Пример ответа.

```
{
  "id": "5fe5d7866882ca0001760fcb",
  "name": "Моя первая группа",
  "subscriptionId": "5fa919fa292a8300019349bc"
}
```

Идентификатор группы из примера выше `5fe5d7866882ca0001760fcb`.

3. Для добавления нового пользователя в группу сделайте `PUT` запрос на `{host_name}/api/manage/v1/Groups/{groupId}/Users/{userId}`, вместо `{groupId}` следует ввести идентификатор группы, а вместо `{userId}` следует ввести идентификатор пользователя.

Пример запроса.

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

В ответе будет получено пустое сообщение с кодом `OK 200`.

4. Для получения списка пользователей в группе сделайте `GET` запрос на `{host_name}/api/manage/v1/Groups/{id}/Users?skip=0&take=10`, где вместо `{id}` следует указать идентификатор группы.

Пример запроса.

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

Пример ответа.

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

Что дальше?

- [Помощь и обратная связь](#)

Обратите внимание! В примерах может не быть заголовка `Authorization` потому что используется модель аутентификации, основанная на `cookie`. Подробнее про авторизацию читайте в разделе [Аутентификация и авторизация](#)

Работа с задачами

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

Задачи в МоиОтчеты Корпоративный Сервер - это действия по преобразованию и доставке документов потребителям. Они подробно описаны в разделе [Задачи](#).

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. Инструмент curl.

Подойдёт любой другой REST клиент, но примеры будут построены для curl.

3. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор.

4. Доступ в интернет.

Общие сведения

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрим типы задач и как с ними работать.

Классификация задач

Задачи в Сервисных решениях по типу запуска делятся на 2 типа:

1. Хранимые задачи;
2. Задачи, запускаемые из тела запроса.

Хранимые задачи - это задачи, которые сохраняются в базе данных и могут быть запущены позже. Они позволяют выполнять задачи по требованию или в нужное время по расписанию. Такие задачи отображаются в разделе "Задачи" в веб-интерфейсе Сервисного решения.

Задачи, запускаемые из тела запроса, не сохраняются в базе данных и выполняются немедленно. Они полезны при выполнении однократных запросов. Например, при построении PDF-документа и отправки его на почту по API из стороннего приложения. Таким образом можно быстро добавить функционал создания и отправки отчётов и документов в своё приложение, написанное на любом языке программирования.

Хранимые задачи

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

Как указано в [предыдущем разделе](#), в Сервисных решениях есть возможность создавать задачи, которые сохраняются в базе данных и могут быть запущены позже. Эти задачи называются хранимыми задачами.

Рассмотрим подробнее, как создавать, обновлять и запускать хранимые задачи с помощью REST API.

О ViewModel и типах

При работе вы будете передавать разные ViewModel (Модели Представления) в виде JSON. Важно, что первое поле любого ViewModel должно быть "\$t": "Название". Подробные примеры будут приведены ниже в этой статье.

Модели Представления для создания задач

Модели Представления для создания задач-преобразователей:

- CreatePrepareTaskVM
- CreateExportTemplateTaskVM
- CreateExportReportTaskVM

Модели Представления для создания задач-транспортов:

- CreateEmailTaskVM
- CreateWebhookTaskVM
- CreateFTPUploadTaskVM
- CreateS3UploadTaskVM

Модели Представления для создания иных задач:

- CreateFetchTaskVM

Модели Представления для обновления свойств сохранённых задач

Модели Представления для обновления задач-преобразователей:

- UpdatePrepareTaskVM
- UpdateExportTemplateTaskVM
- UpdateExportReportTaskVM

Модели Представления для обновления задач-транспортов:

- UpdateEmailTaskVM
- UpdateWebhookTaskVM
- UpdateFTPUploadTaskVM
- UpdateS3UploadTaskVM

Модели Представления для обновления иных задач:

- UpdateFetchTaskVM

Модель Представления для обновления прав задач:

- UpdateTaskPermissionsVM

Создание задачи

Для создания задач следует использовать метод CreateTask. Он принимает Модель Представления для создания задач.

Рассмотрим создание задачи экспорта отчёта и последующей отправкой по почте:

```
curl -X 'POST' \  
'https://{host_name}/api/tasks/v1/tasks' \  
-H 'accept: application/json' \  
-H 'Content-Type: application/json' \  
-d '{  
  "$t": "CreateExportTemplateTaskVM",  
  "subscriptionId": "23e0134c816935c1e11b3737",  
  "name": "SendViaWebhookExport",  
  "inputFile": {  
    "entityId": "61e0134c816935c1e11b3787"  
  },  
  "transports": [  
    {  
      "$t": "CreateEmailTaskVM",  
      "name": "Задача рассылки на почту",  
      "from": "admin@company.ru",  
      "to": ["user@company.ru", "seller@company.ru"],  
      "subject": "Важный отчет",  
      "body": "К письму прикреплен важный отчет.",  
      "EnableSsl": true,  
      "username": "admin@company.ru",  
      "password": "parol123",  
      "server": "smtp.company.ru",  
      "port": 587  
    }  
  ],  
  "format": "Pdf"  
}'
```

В поля EntityId и FolderId следует записывать реальные идентификаторы объектов. Иначе задача будет прервана с ошибкой.

Обратите внимание! Если не указать OutputFile, то он будет сохранён во временную папку. Если указать пустой OutputFile - в корневую папку. Если указать id папки - в неё.

Список всех доступных вьюмоделей доступен по ссылке - https://{host_name}/api/swagger/index.html

Запуск задач по расписанию

Чтобы настроить запуск задачи по расписанию, необходимо указать следующие поля:

- cronExpression
- startsOn
- ends

cronExpression - строка, содержащая cron-выражение, определяющее расписание запуска задачи.

startsOn - дата и время, с которого начинается выполнение задачи по расписанию (в часовом поясе пользователя).

ends - объект, содержащий дату и время, до которого будет выполняться задача по расписанию (on) или количество повторений (after). Поле on задаётся в часовом поясе пользователя. Если не указано ни поле after ни поле on, то задача будет выполняться бесконечно.

Пример создания задачи, запускаемой по расписанию:

```

curl -X 'POST' \
'https://{host_name}/api/tasks/v1/tasks' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
"$t": "CreateExportTemplateTaskVM",
"subscriptionId": "23e0134c816935c1e11b3737",
"name": "SendViaWebhookExport",
"inputFile": {
"entityId": "61e0134c816935c1e11b3787"
},
"transportIds": [
"68767aef184dcf225ca085e1"
],
"format": "Pdf",
"cronExpression": "*/*5 * * * *", # Запускать каждые 5 минут
"startsOn": "2025-10-01T00:00:00Z", # Начать с 1 октября 2025 года
"ends": {
"on": "2025-12-31T23:59:59Z" # Закончить 31 декабря 2025 года
}
}'

```

В этом примере задача будет запускаться каждые 5 минут, начиная с 1 октября 2025 года и заканчивая 31 декабря 2025 года.

Чтобы сделать так, чтобы задача выполнялась, например, 10 раз, нужно заменить поле `on` на поле `after`:

```

"cronExpression": "*/*5 * * * *", # Запускать каждые 5 минут
"startsOn": "2025-10-01T00:00:00Z", # Начать с 1 октября 2025 года
"ends": {
"after": 10 # Выполнить 10 раз"
}

```

Получение списка задач

```

// Получение первых десяти задач из рабочего пространства
curl -X 'GET' \
'https://{host_name}/api/tasks/v1/tasks?skip=0&take=10' \
-H 'accept: application/json'

```

Редактирование задачи

После того как задача создана можно изменить некоторые её параметры.

```

// Редактирование задачи
curl --location --request PUT 'https://{host_name}/api/tasks/v1/Tasks/{id задачи}' \
--header 'accept: application/json' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic {base64(apikey:api-key-value)}' \
--data '{
"$t": "UpdateExportTemplateTaskVM",
"reportParameters": {
"param1": "val1",
"param2": "val2"
},
"transportIds" : [{"id задачи-транспорта (рассылка на почту, отправка по FTP и др.)"}]
}'

```

Обратите внимание, что при обновлении задачи нужно указывать `transportIds` - список идентификаторов задач-транспортов (Email, FTP, Webhook, S3), а не сами задачи. Саму задачу можно создать заранее как отдельно так и при [быстрых отчетах](#). рф

создании задачи-трансформера. Подробнее про их создание читайте в разделе [Создание задач-транспортов](#).

Выполнение задачи по указанному идентификатору

```
curl -X 'POST' \  
'https://{host_name}/api/tasks/v1/tasks/42d134ae3130aaad37by345f/run' \  
-H 'accept: */*' \  
-d ""
```

Удаление задачи из хранилища

```
curl -X 'DELETE' \  
'https://{host_name}/api/tasks/v1/tasks/42d134ae3130aaad37by345f' \  
-H 'accept: */*'
```

Задачи, запускаемые из тела запроса

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

Как указано в [предыдущем разделе](#), в Сервисных решениях есть возможность запускать задачи без предварительного сохранения. Для этого следует выполнить POST запрос по пути `/api/tasks/v1/Tasks/run`, передав в запросе все необходимые для этого данные.

Например, для того, чтобы построить PDF-документ и отправить его по FTP нужно передать в теле запроса такие данные как идентификатор шаблона (`entityId`) или сам шаблон (`content`) в объект `inputFile`, а также данные для подключения к FTP-серверу.

Рассмотрим подробнее, как работать с такими задачами с помощью REST API.

О ViewModel и типах

При работе вы будете передавать разные ViewModel (Модели Представления) в виде JSON. Важно, что первое поле любого ViewModel должно быть "\$t": "Название". Подробные примеры будут приведены ниже в этой статье.

Модели Представления для запуска задач на лету (без сохранения)

Модели Представления для запуска задач-преобразователей на лету:

- RunPrepareTaskVM
- RunExportTemplateTaskVM
- RunExportReportTaskVM

Модели Представления для запуска задач-транспортов на лету:

- RunEmailTaskVM
- RunWebhookTaskVM
- RunFTPUploadTaskVM
- RunS3UploadTaskVM

Модели Представления для запуска иных задач на лету:

- RunFetchTaskVM

Запуск задачи на лету (из тела запроса)

Для запуска задачи на лету следует использовать метод RunTask.

Рассмотрим запуск задачи экспорта отчёта и последующим сохранением в хранилище S3:

```
curl --location 'https://{host_name}/api/tasks/v1/tasks/run' \  
--header 'accept: application/json' \  
--header 'Content-Type: application/json' \  
--header 'Authorization: Basic {base64(apikey:api-key-value)}' \  
--data '{  
  "$t": "RunExportTemplateTaskVM",  
  "subscriptionId": "{id рабочего пространства}",  
  "inputFile": {  
    "$t": "RunInputFileVM",  
    "entityId": "{id шаблона}",  
    "type": "Template",  
    "fileName": "Template.frx"  
  },  
  "format": "Pdf",  
  "transports": [  
    {  
      "$t": "RunS3UploadTaskVM",  
      "accessKey": "{публичный_ключ_доступа}",  
      "secretKey": "{секретный_ключ_доступа}",  
      "bucketName": "хранилище",  
      "destinationFolder": "test",  
      "useAws": true,  
      "enableSsl": true,  
      "region": "us-east-1"  
    }  
  ]  
}'
```

В поле EntityId следует записывать реальные идентификаторы объектов. Иначе задача будет прервана с ошибкой.

Также вместо него в объекте `inputFile` можно передать содержимое шаблона в поле `content` в виде Base64-строки.

Список всех доступных выюмоделей доступен по ссылке - https://{host_name}/api/swagger/index.html

Задача экспорта шаблона

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрим работу с задачей экспорта шаблона (ExportTemplateTask). Общие инструкции о том как работать с задачами описаны в разделе [Общие сведения](#).

Следует отметить что экспортировать шаблон можно не только с помощью системы задач, но и напрямую через запрос по пути `/api/rp/v1/Templates/File/{id}/Export`. Преимущество использования системы задач в том, что она позволяет не только сохранить документ в папку, но и передать во внешние системы (по Email, FTP, S3, Webhook).

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. Инструмент curl.

Подойдёт любой другой REST клиент, но примеры будут построены для curl.

3. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор.

4. Доступ в интернет.

5. Настроенный и доступный S3 сервер.

Обратите внимание! Пункты выше описывают рекомендуемые инструменты.

Создание задачи

Рассмотрим создание задачи экспорта шаблона в PDF:

```
// Создание задачи
curl --location 'https://{host_name}/api/tasks/v1/Tasks' \
--header 'accept: application/json' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic {base64(apikey:api-key-value)}' \
--data '{
  "$t": "CreateExportTemplateTaskVM",
  "name": "Задача экспорта в PDF",
  "subscriptionId": "{id рабочего пространства}",
  "inputFile": {
    "entityId": "{id шаблона}",
    "type": "Template"
  },
  "outputFile": {
    "fileName": "Отчёт"
  },
  "format": "Pdf",
  "reportParameters": {
    "param1": "val1",
    "param2": "val2"
  },
  "exportParameters": {
    "PrintOptimized": "true",
    "JpegQuality": "90"
  }
}'
```

Так как в `outputFile` не указана папка для сохранения, то при запуске задачи документ будет сохранён в корневую папку.

Зададим другую папку для сохранения:

```
"outputFile" : {
  "folderId": "61cb2226d24478b172103075",
  "fileName": "Отчёт"
}
```

Теперь при запуске задачи PDF-файл будет сохранён в указанную папку.

```
// Запуск задачи по идентификатору
curl -X 'POST' \
'https://{host_name}/api/tasks/v1/Tasks/{идентификатор задачи}/run' \
-H 'accept: */*' \
-d "
```

В поля `EntityId` и `SubscriptionId` следует записывать реальные идентификаторы объектов. Иначе задача будет прервана с ошибкой.

Отправка документа после экспорта

Для отправки документа после экспорта можно использовать задачу отправки по Email, FTP, S3 или Webhook. Рассмотрим пример создания задачи экспорта в PDF с последующей отправкой готового документа по Email.

```
// Создание задачи
curl --location 'https://{host_name}/api/tasks/v1/Tasks' \
--header 'accept: application/json' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic {base64(apikey:api-key-value)}' \
--data '{
  "$t": "CreateExportTemplateTaskVM",
  "name": "Задача экспорта в PDF",
  "subscriptionId": "{id рабочего пространства}",
  "inputFile": {
    "entityId": "{id шаблона}",
    "type": "Template"
  },
  "outputFile": {
    "fileName": "Отчёт"
  },
  "format": "Pdf",
  "reportParameters": {
    "param1": "val1",
    "param2": "val2"
  },
  "exportParameters": {
    "PrintOptimized": "true",
    "JpegQuality": "90"
  },
  "transports": [
    {
      "$t": "CreateEmailTaskVM",
      "name": "Задача рассылки на почту",
      "from": "admin@company.ru",
      "to": ["user@company.ru", "seller@company.ru"],
      "subject": "Важный отчет",
      "body": "К письму прикреплен важный отчет.",
      "EnableSsl": true,
      "username": "admin@company.ru",
      "password": "parol123",
      "server": "smtp.company.ru",
      "port": 587
    }
  ]
}'
```

Редактирование задачи

После того как задача создана можно изменить некоторые её параметры.

```
// Редактирование задачи
curl --location --request PUT 'https://{host_name}/api/tasks/v1/Tasks/{id задачи}' \
--header 'accept: application/json' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic {base64(apikey:api-key-value)}' \
--data '{
  "$t": "UpdateExportTemplateTaskVM",
  "reportParameters": {
    "param1": "val1",
    "param2": "val2"
  },
  "transportIds": ["{id задачи-транспорта (рассылка на почту, отправка по FTP и др.)}"]
}'
```

Обратите внимание, что при обновлении задачи нужно указывать `transportIds` - список идентификаторов задач-

транспортов(Email, FTP, Webhook, S3), а не сами задачи. Саму задачу можно создать заранее как отдельно так и при создании задачи-трансформера. Подробнее про их создание читайте в разделе [Создание задач-транспортов](#).

Выполнение задачи из тела запроса

```
curl --location 'https://{host_name}/api/tasks/v1/tasks/run' \  
--header 'accept: application/json' \  
--header 'Content-Type: application/json' \  
--header 'Authorization: Basic {base64(apikey:api-key-value)}' \  
--data '{  
  "$t": "RunExportTemplateTaskVM",  
  "subscriptionId": "{id рабочего пространства}",  
  "inputFile": {  
    "$t": "RunInputFileVM",  
    "entityId": "{id шаблона}",  
    "type": "Template",  
    "fileName": "Template.frx"  
  },  
  "format": "Pdf",  
  "transports": [  
    {  
      "$t": "RunS3UploadTaskVM",  
      "accessKey": "{публичный_ключ_доступа}",  
      "secretKey": "{секретный_ключ_доступа}",  
      "bucketName": "хранилище",  
      "destinationFolder": "test",  
      "useAws": true,  
      "enableSsl": true,  
      "region": "us-east-1"  
    }  
  ]  
}'
```

Таким образом в результате выполнения этого запроса будет создан PDF-документ и отправлен в S3-хранилище.

Обратите внимание! В этом случае создание PDF-документа и его отправка в S3 хранилище будет выполнена непосредственно этим запросом и задача не будет сохранена в базе данных.

Задача экспорта отчета

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрим работу с задачей экспорта отчета (ExportReportTask). Общие инструкции о том как работать с задачами описаны в разделе [Общие сведения](#).

Следует отметить что экспортировать отчёт можно не только с помощью системы задач, но и напрямую через запрос по пути [/api/rp/v1/Reports/File/{id}/Export](#). Преимущество использования системы задач в том, что она позволяет не только сохранить документ в папку, но и передать во внешние системы (по Email, FTP, S3, Webhook).

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. Инструмент curl.

Подойдёт любой другой REST клиент, но примеры будут построены для curl.

3. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор.

4. Доступ в интернет.

5. Настроенный и доступный S3 сервер.

Обратите внимание! Пункты выше описывают рекомендуемые инструменты.

Создание задачи

Рассмотрим создание задачи экспорта отчета в PDF:

```
// Создание задачи
curl --location 'https://{host_name}/api/tasks/v1/Tasks' \
--header 'accept: application/json' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic {base64(apikey:api-key-value)}' \
--data '{
  "$t": "CreateExportReportTaskVM",
  "name": "Задача экспорта отчёта в PDF",
  "subscriptionId": "{id рабочего пространства}",
  "inputFile": {
    "entityId": "{id отчета}",
    "type": "Report"
  },
  "outputFile": {
    "fileName": "Отчёт"
  },
  "format": "Pdf",
  "exportParameters": {
    "PrintOptimized": "true",
    "JpegQuality": "90"
  }
}'
```

Так как в `outputFile` не указана папка для сохранения, то при запуске задачи документ будет сохранён в корневую папку.

Зададим другую папку для сохранения:

```
"outputFile" : {  
  "folderId": "61cb2226d24478b172103075",  
  "fileName": "Отчёт"  
}
```

Теперь при запуске задачи PDF-файл будет сохранён в указанную папку.

```
// Запуск задачи по идентификатору  
curl -X 'POST' \  
  'https://{host_name}/api/tasks/v1/Tasks/{идентификатор задачи}/run' \  
  -H 'accept: */*' \  
  -d "
```

В поля `EntityId` и `SubscriptionId` следует записывать реальные идентификаторы объектов. Иначе задача будет прервана с ошибкой.

Отправка документа после экспорта

Для отправки документа после экспорта можно использовать задачу отправки по Email, FTP, S3 или Webhook. Рассмотрим пример создания задачи экспорта отчета в PDF с последующей отправкой готового документа по Email.

```
// Создание задачи
curl --location 'https://{host_name}/api/tasks/v1/Tasks' \
--header 'accept: application/json' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic {base64(apikey:api-key-value)}' \
--data '{
  "$t": "CreateExportReportTaskVM",
  "name": "Задача экспорта отчета в PDF",
  "subscriptionId": "{id рабочего пространства}",
  "inputFile": {
    "entityId": "{id отчета}",
    "type": "Report"
  },
  "outputFile": {
    "fileName": "Отчёт"
  },
  "format": "Pdf",
  "exportParameters": {
    "PrintOptimized": "true",
    "JpegQuality": "90"
  },
  "transports": [
    {
      "$t": "CreateEmailTaskVM",
      "name": "Задача рассылки на почту",
      "from": "admin@company.ru",
      "to": ["user@company.ru", "seller@company.ru"],
      "subject": "Важный отчет",
      "body": "К письму прикреплен важный отчет.",
      "EnableSsl": true,
      "username": "admin@company.ru",
      "password": "parol123",
      "server": "smtp.company.ru",
      "port": 587
    }
  ]
}'
```

Редактирование задачи

После того как задача создана можно изменить некоторые её параметры.

```
// Редактирование задачи
curl --location --request PUT 'https://{host_name}/api/tasks/v1/Tasks/{id задачи}' \
--header 'accept: application/json' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic {base64(apikey:api-key-value)}' \
--data '{
  "$t": "UpdateExportTemplateTaskVM",
  "reportParameters": {
    "param1": "val1",
    "param2": "val2"
  },
  "transportIds": ["{id задачи-транспорта (рассылка на почту, отправка по FTP и др.)}"]
}'
```

Обратите внимание, что при обновлении задачи нужно указывать `transportIds` - список идентификаторов задач-транспортов (Email, FTP, Webhook, S3), а не сами задачи. Саму задачу можно создать заранее как отдельно так и при создании задачи-трансформера. Подробнее про их создание читайте в разделе [Создание задач-транспортов](#).

Выполнение задачи из тела запроса

```
curl --location 'https://{host_name}/api/tasks/v1/tasks/run' \  
--header 'accept: application/json' \  
--header 'Content-Type: application/json' \  
--header 'Authorization: Basic {base64(apikey:api-key-value)}' \  
--data '{  
  "$t": "RunExportReportTaskVM",  
  "subscriptionId": "{id рабочего пространства}",  
  "inputFile": {  
    "$t": "RunInputFileVM",  
    "entityId": "{id шаблона}",  
    "type": "Report",  
    "fileName": "Report.fpx"  
  },  
  "format": "Pdf",  
  "transports": [  
    {  
      "$t": "RunS3UploadTaskVM",  
      "accessKey": "{публичный_ключ_доступа}",  
      "secretKey": "{секретный_ключ_доступа}",  
      "bucketName": "хранилище",  
      "destinationFolder": "test",  
      "useAws": true,  
      "enableSsl": true,  
      "region": "us-east-1"  
    }  
  ]  
}'
```

Таким образом в результате выполнения этого запроса из отчета будет создан PDF-документ и отправлен в S3-хранилище.

Обратите внимание! В этом случае создание PDF-документа и его отправка по почте будет выполнена непосредственно этим запросом и задача не будет сохранена в базе данных.

Задача подготовки отчета

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрим работу с задачей подготовки отчета (PrepareTemplateTask). Общие инструкции о том как работать с задачами описаны в разделе [Общие сведения](#).

Следует отметить что подготавливать отчёт (преобразовывать шаблон в формате *.frx и данные в готовый документ в формате *.frx) можно не только с помощью системы задач, но и напрямую через запрос по пути [/api/rp/v1/Templates/File/{id}/Prepare](#). Преимущество использования системы задач в том, что она позволяет не только сохранить документ в папку, но и передать во внешние системы (по Email, FTP, S3, Webhook).

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. Инструмент curl.

Подойдёт любой другой REST клиент, но примеры будут построены для curl.

3. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор.
4. Доступ в интернет.
5. Настроенный и доступный S3 сервер.

Обратите внимание! Пункты выше описывают рекомендуемые инструменты.

Создание задачи

Рассмотрим создание задачи подготовки отчета:

```
// Создание задачи
curl --location 'https://{host_name}/api/tasks/v1/Tasks' \
--header 'accept: application/json' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic {base64(apikey:api-key-value)}' \
--data '{
  "$t": "CreatePrepareTemplateTaskVM",
  "name": "Задача подготовки отчета",
  "subscriptionId": "{id рабочего пространства}",
  "inputFile": {
    "entityId": "{id шаблона}",
    "type": "Template"
  },
  "outputFile": {
    "fileName": "Отчёт"
  },
  "reportParameters": {
    "param1": "val1",
    "param2": "val2"
  }
}'
```

Так как в `outputFile` не указана папка для сохранения, то при запуске задачи документ будет сохранён в корневую папку.

Зададим другую папку для сохранения:

```
"outputFile" : {
  "folderId": "{id папки с отчётами}",
  "fileName": "Отчёт"
}
```

Теперь при запуске задачи подготовленный отчет будет сохранён в указанную папку.

```
// Запуск задачи по идентификатору
curl -X 'POST' \
  'https://{host_name}/api/tasks/v1/Tasks/{идентификатор задачи}/run' \
  -H 'accept: */*' \
  -d "
```

В поля `EntityId` и `SubscriptionId` следует записывать реальные идентификаторы объектов. Иначе задача будет прервана с ошибкой.

Отправка документа после подготовки

Для отправки документа после подготовки можно использовать задачу отправки по Email, FTP, S3 или Webhook. Рассмотрим пример создания задачи подготовки отчета с последующей отправкой готового документа по Email.

```
// Создание задачи
curl --location 'https://{host_name}/api/tasks/v1/Tasks' \
--header 'accept: application/json' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic {base64(apikey:api-key-value)}' \
--data '{
  "$t": "CreatePrepareTemplateTaskVM",
  "name": "Задача подготовки отчета",
  "subscriptionId": "{id рабочего пространства}",
  "inputFile": {
    "entityId": "{id шаблона}",
    "type": "Template"
  },
  "outputFile" : {
    "fileName": "Отчёт"
  },
  "reportParameters": {
    "param1": "val1",
    "param2": "val2"
  },
  "transports": [
    {
      "$t": "CreateEmailTaskVM",
      "name": "Задача рассылки на почту",
      "from": "admin@company.ru",
      "to": ["user@company.ru", "seller@company.ru"],
      "subject": "Важный отчет",
      "EnableSsl": true,
      "username": "admin@company.ru",
      "password": "parol123",
      "server": "smtp.company.ru",
      "port": 587
    }
  ]
}'
```

Редактирование задачи

После того как задача создана можно изменить некоторые её параметры.

```
// Редактирование задачи
curl --location --request PUT 'https://{host_name}/api/tasks/v1/Tasks/{id задачи}' \
--header 'accept: application/json' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic {base64(apikey:api-key-value)}' \
--data '{
  "$t": "UpdateExportTemplateTaskVM",
  "reportParameters": {
    "param1": "val1",
    "param2": "val2"
  },
  "transportIds": [{"id задачи-транспорта (рассылка на почту, отправка по FTP и др.)}"]
}'
```

Обратите внимание, что при обновлении задачи нужно указывать `transportIds` - список идентификаторов задач-транспортов (Email, FTP, Webhook, S3), а не сами задачи. Саму задачу можно создать заранее как отдельно так и при создании задачи-трансформера. Подробнее про их создание читайте в разделе [Создание задач-транспортов](#).

Выполнение задачи из тела запроса

```
curl --location 'https://{host_name}/api/tasks/v1/tasks/run' \
--header 'accept: application/json' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic {base64(apikey:api-key-value)}' \
--data '{
  "$t": "RunPrepareTemplateTaskVM",
  "subscriptionId": "{id рабочего пространства}",
  "inputFile": {
    "$t": "RunInputFileVM",
    "entityId": "{id шаблона}",
    "type": "Template",
    "fileName": "Template.frx"
  },
  "transports": [
    {
      "$t": "RunS3UploadTaskVM",
      "accessKey": "{публичный_ключ_доступа}",
      "secretKey": "{секретный_ключ_доступа}",
      "bucketName": "хранилище",
      "destinationFolder": "test",
      "useAws": true,
      "enableSsl": true,
      "region": "us-east-1"
    }
  ]
}'
```

Таким образом в результате выполнения этого запроса будет создан *.frx документ и отправлен в S3-хранилище.

Обратите внимание! В этом случае создание *.frx-документа и его отправка по почте будет выполнена непосредственно этим запросом и задача не будет сохранена в базе данных.

Отправка на Email

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрим способ отправки отчёта на электронную почту. Инструкции о том как работать с задачами описаны в разделе [Общие сведения](#).

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. Инструмент curl.

Подойдёт любой другой REST клиент, но примеры будут построены для curl.

3. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор.
4. Доступ в интернет.
5. Настроенная и доступная учетная запись электронной почты.

Обратите внимание! Пункты выше описывают рекомендуемые инструменты.

Создание задачи

Рассмотрим создание задачи отправки шаблона по электронной почте:

```
// Создание задачи

curl -X 'POST' \
'https://{host_name}/api/tasks/v1/Tasks' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
"$t": "CreateEmailTaskVM",
"name": "Задача отправки по Email",
"subscriptionId": "{id рабочего пространства}",
"inputFile": {
"entityId": "{id шаблона}",
"type": "Template"
},
"from": "{почта отправителя}",
"to": ["{почта получателя 1}", "{почта получателя 2}"],
"subject": "Test message",
"EnableSsl": true,
"username": "{имя пользователя}",
"password": "{пароль}",
"server": "{адрес smtp-сервера}",
"port": 587
}'
```

```
// Запуск задачи по идентификатору
curl -X 'POST' \
  'https://{host_name}/api/tasks/v1/Tasks/{идентификатор задачи}/run' \
  -H 'accept: */*' \
  -d "
```

В поля EntityId и SubscriptionId следует записывать реальные идентификаторы объектов. Иначе задача будет прервана с ошибкой.

Выполнение задачи из тела запроса

```
curl -X 'POST' \
  'https://{host_name}/api/tasks/v1/Tasks/run' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "$t": "RunEmailTaskVM",
  "subscriptionId": "{id рабочего пространства}",
  "inputFile": {
    "content": "{файл шаблона в base64}",
    "type": "Template"
  },
  "from": "{почта отправителя}",
  "to": ["{почта получателя 1}", "{почта получателя 2}"],
  "subject": "Test message",
  "EnableSsl": true,
  "username": "{имя пользователя}",
  "password": "{пароль}",
  "server": "{адрес smtp-сервера}",
  "port": 587
}'
```

Обратите внимание! В этом случае отправка файла по почте будет выполнена непосредственно этим запросом и задача не будет сохранена в базе данных. Также в поле content можно передать отчёт и готовые документы в различных форматах, например, PDF, XLSX и т.д.

Обратите внимание! В примерах нет заголовка Authorization потому что используется модель аутентификации, основанная на cookie. Подробнее про авторизацию читайте в разделе [Аутентификация и авторизация](#)

Сохранение на FTP-сервер

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрим способ отправки отчёта на FTP сервер. Инструкции о том как работать с задачами описаны в разделе [Общие сведения](#).

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. Инструмент curl.

Подойдёт любой другой REST клиент, но примеры будут построены для curl.

3. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор.

4. Доступ в интернет.

5. Настроенный и доступный FTP сервер.

Обратите внимание! Пункты выше описывают рекомендуемые инструменты.

Обратите внимание! Данное руководство предполагает, что у вас есть настроенный FTP-сервер для принятия файлов от внешних источников.

Создание задачи

Рассмотрим создание задачи отправки шаблона на FTP-сервер:

```
// Создание задачи
curl -X 'POST' \
  'https://{host_name}/api/tasks/v1/Tasks' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "$t": "CreateFTPUploadTaskVM",
    "name": "Задача отправки по FTP",
    "subscriptionId": "23e0134c816935c1e11b3737",
    "ftpHost": "ftp://localhost",
    "ftpPort": 21,
    "ftpUsername": "FtpUser",
    "ftpPassword": "password",
    "archive": false,
    "archiveName": "Имя архива",
    "useSFTP": false,
    "inputFile": {
      "entityId": "61e0134c816935c1e11b3787",
      "type": "Template"
    },
    "destinationFolder": "/путь_назначения/"
  }'
```

```
// Запуск задачи по идентификатору
curl -X 'POST' \
'https://{host_name}/api/tasks/v1/Tasks/{идентификатор задачи}/run' \
-H 'accept: */*' \
-d "
```

В поля EntityId и SubscriptionId следует записывать реальные идентификаторы объектов. Иначе задача будет прервана с ошибкой.

Выполнение задачи из тела запроса

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

Обратите внимание! В этом случае отправка на FTP будет выполнена непосредственно этим запросом и задача не будет сохранена в базе данных.

Обратите внимание! В примерах нет заголовка Authorization потому что используется модель аутентификации, основанная на cookie. Подробнее про авторизацию читайте в разделе [Аутентификация и авторизация](#)

Сохранение по Webhook

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрим способ отправки отчёта по вебхуку. Инструкции о том как работать с задачами описаны в разделе [Общие сведения](#).

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. Инструмент curl.

Подойдёт любой другой REST клиент, но примеры будут построены для curl.

3. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор.

4. Доступ в интернет.

5. Клиентское приложение, принимающее запросы.

Обратите внимание! Пункты выше описывают рекомендуемые инструменты.

Обратите внимание! Данное руководство предполагает, что у вас есть клиентское приложение, принимающее и обрабатывающее входящие вебхуки.

Создание задачи

Рассмотрим создание задачи отправки шаблона по вебхуку:

```
// Создание задачи
curl -X 'POST' \
  'https://{host_name}/api/tasks/v1/Tasks' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "$t": "CreateWebhookTaskVM",
    "name": "Задача отправки по Webhook",
    "subscriptionId": "{идентификатор рабочего пространства}",
    "url": "https://example.com/",
    "headers": {
      "Authorization": "Bearer <token>",
      "Content-Type": "multipart/form-data"
    },
    "inputFile": {
      "entityId": "{идентификатор шаблона}",
      "type": "Template"
    }
  }'
```

```
// Запуск задачи по идентификатору
curl -X 'POST' \
'https://{host_name}/api/tasks/v1/Tasks/{идентификатор задачи}/run' \
-H 'accept: */*' \
-d "
```

Обратите внимание! В поля EntityId и SubscriptionId следует записывать реальные идентификаторы объектов. Иначе задача будет прервана с ошибкой.

Обратите внимание! В EntityId может передаваться идентификатор шаблона, отчёта и экспорта.

Выполнение задачи из тела запроса

```
curl -X 'POST' \
'https://{host_name}/api/tasks/v1/Tasks/run' \
-H 'accept: */*' \
-H 'Content-Type: application/json' \
-d '{
  "$t": "RunWebhookTaskVM",
  "name": "Задача отправки по Webhook",
  "subscriptionId": "{идентификатор рабочего пространства}",
  "url": "https://example.com/",
  "headers": {
    "Authorization": "Bearer <token>",
    "Content-Type": "multipart/form-data",
    "Content-Length": "1278"
  },
  "inputFile": {
    "entityId": "{идентификатор шаблона}",
    "type": "Template"
  }
}'
```

Обратите внимание! В этом случае отправка по вебхуку будет выполнена непосредственно этим запросом и задача не будет сохранена в базе данных.

Пример запроса, который придёт на эндпоинт вебхука:

```

{
  "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"
    }
  }
}

```

Обратите внимание! В примерах нет заголовка Authorization потому что используется модель аутентификации, основанная на cookie. Подробнее про авторизацию читайте в разделе [Аутентификация и авторизация](#).

Сохранение в S3-хранилище

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрим способ отправки отчёта на S3-хранилище. Инструкции о том как работать с задачами описаны в разделе [Общие сведения](#).

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. Инструмент curl.

Подойдёт любой другой REST клиент, но примеры будут построены для curl.

3. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор.

4. Доступ в интернет.

5. Настроенное и доступное S3-хранилище.

Обратите внимание! Пункты выше описывают рекомендуемые инструменты.

Обратите внимание! Данное руководство предполагает, что у вас есть настроенное S3-хранилище для принятия файлов от внешних источников.

Создание задачи

Рассмотрим создание задачи отправки шаблона в S3-хранилище:

```
// Создание задачи

curl -X 'POST' \
  'https://{host_name}/api/tasks/v1/Tasks' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "$t": "CreateS3UploadTaskVM",
    "name": "Задача отправки на S3",
    "subscriptionId": "23e0134c816935c1e11b3737",

    "accessKey": "{публичный_ключ_доступа}",
    "secretKey": "{секретный_ключ_доступа}",
    "url": "https://example.com",
    "bucketName": "{название_бакета}",
    "useAws": true,
    "enableSsl": true,
    "region": "us-east-1"
    "inputFile": {
      "entityId": "61e0134c816935c1e11b3787",
      "type": "Template"
    },
    "destinationFolder": "/путь_назначения/"
  }'
```

```
// Запуск задачи по идентификатору

curl -X 'POST' \
  'https://{host_name}/api/tasks/v1/Tasks/{идентификатор_задачи}/run' \
  -H 'accept: */*' \
  -d ''
```

В поля EntityId и SubscriptionId следует записывать реальные идентификаторы объектов. Иначе задача будет прервана с ошибкой.

Выполнение задачи из тела запроса

```
curl -X 'POST' \
  'https://{host_name}/api/tasks/v1/Tasks/run' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "$t": "RunS3UploadTaskVM",
    "name": "Задача отправки на S3",
    "subscriptionId": "23e0134c816935c1e11b3737",
    "accessKey": "{публичный_ключ_доступа}",
    "secretKey": "{секретный_ключ_доступа}",
    "url": "https://example.com",
    "bucketName": "{название_бакета}",
    "useAws": true,
    "enableSsl": true,
    "region": "us-east-1"
    "inputFile": {
      "entityId": "61e0134c816935c1e11b3787",
      "type": "Template"
    },
    "destinationFolder": "/путь_назначения/"
  }'
```

Обратите внимание! В этом случае отправка на S3 будет выполнена непосредственно этим запросом и задача не будет сохранена в базе данных.

Обратите внимание! В примерах нет заголовка Authorization потому что используется модель аутентификации, основанная на cookie. Подробнее про авторизацию читайте в разделе [Аутентификация и авторизация](#)

C# .NET

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этом разделе приведены пошаговые инструкции и руководства для выполнения типовых задач по применению Сервисных решений с использованием языка программирования C#/.NET и FastReport.Cloud.SDK.

Приступая к работе

Вам понадобятся следующие инструменты и возможности.

1. [.NET SDK](#).
2. Редактор C# кода или текстовый редактор, например [Visual Studio Code](#).
3. Дизайнер отчётов [FastReport Community Designer](#).

Для некоторых руководств нужен будет дизайнер для создания отчёта.

4. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор.
5. Доступ в интернет.

Обратите внимание! Эти руководства рассчитаны, что вы уже знаете, как разработать своё приложение на языке программирования C#.

Примечание. Пункты выше описывают рекомендуемые инструменты.

Установка и настройка

Для подключения SDK следует установить пакет `FastReport.Cloud.SDK` последней стабильной версии. Это можно сделать следующей командой.

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

Для добавления поддержки ASP.NET следует установить пакет `FastReport.Cloud.SDK.Web`.

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

Также в класс `Startup` необходимо добавить строчку.

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

Что дальше?

Советуем начать своё ознакомление со статей:

- [Статус коды](#).
- [Аутентификация и авторизация](#).

Аутентификация и авторизация

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

Процесс проверки подлинности пользовательских данных и выдача пользователю определённых прав в Сервисных решениях осуществляется по одному из двух доступных способов:

1. Через JWT token.

В этом случае аутентификацию нужно пройти лично, а токен будет действовать только 5 минут, за которые пользователь должен войти в своё приложение. При подключении к серверу браузер перенаправит на сервер аутентификации, после чего сгенерирует токен доступа. С точки зрения безопасности мы ограничиваем возможность получения JWT токена только лично пользователем.

Если в течение 5 минут пользователь не зашёл в приложение, то аутентификацию необходимо пройти заново. Если пользователь зашёл в приложение повторная аутентификация не требуется.

2. Через API key.

В этом случае процесс получения прав доступа осуществляется для серверных приложений. Для получения ключа доступа (API key) необходимо присутствие пользователя. Однако сам ключ может действовать продолжительное время, например год.

Получение первого API key

Для получения первого API key воспользуйтесь [пользовательской панелью](#). Если по какой-то причине доступа к пользовательской панели нет, можно запросить ключ по описанию ниже.

Самый простой вариант получения ключа - открытие вкладки "Апи ключи" и создание его на данной странице.

Вариант 2:

1. Откройте ссылку в браузере: <{host_name}/account/signin?r={host_name}/api/manage/v1/ApiKeys>.

Переход по этой ссылке направит вас на автоматический процесс аутентификации браузера.

2. Теперь, когда аутентификация пройдена, необходимо запросить новый ключ.

Нажмите **F12** или **Ctrl+Shift+I**, чтобы открыть панель разработчика. Сочетание клавиш могут отличаться от стандартных, в этом случае откройте панель разработчика через меню браузера.

3. Скопируйте и выполните код в консоли JavaScript.

Этот код сделает **POST** запрос на URL `{host_name}/api/manage/v1/ApiKeys` для создания нового ключа доступа до 2030 года.

4. Обновите страницу браузера и получите результат.

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

Теперь вы можете использовать API key. В примере выше им является

```
cc355oeu1z5d5wncayo33me6c1g5junqdqk4pkupid7t8ynjshey
```

.

Повторно получать новый API key через браузер нет необходимости.

Как использовать API key

Ключ следует передавать с каждым запросом в заголовке `Authorization: Basic`. В качестве имени пользователя следует использовать `apikey`, а в качестве пароля - значение ключа. Например:

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

Где `Base64Encode` это функция кодировки строки в base64.

Для `FastReport.Cloud.SDK` есть специальный класс, который позволяет добавлять ключ к заголовку запроса `<xref:FastReport.Cloud.FastReportCloudApiKeyHeader>`.

Для добавления нужного заголовка создайте новый `HttpClient`.

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

Теперь этот `HttpClient` можно использовать для всех запросов.

Получение нового API key

Для получения нового ключа следует вызвать метод

```
<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);
```

По возможности используйте асинхронные аналоги методов вместо синхронных.

В результате выполнения этой функции будет получена модель `<xref:FastReport.Cloud.ApiKeyVM>`.

Что дальше?

- [Загрузка нового шаблона.](#)
- [Работа с группами.](#)

- [Добавление новых пользователей в рабочее пространство](#)

Загрузка нового шаблона

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

Одна из основных возможностей Сервисных решений — это хранение шаблонов, отчётов и других данных в облаке. В этой статье будет рассмотрен способ загрузить свой готовый шаблон в хранилище шаблонов сервисного решения.

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. [.NET SDK](#).

3. Редактор C# кода или текстовый редактор, например [Visual Studio Code](#).

4. Шаблон отчёта.

Его можно подготовить с помощью бесплатной программы [FastReport Community Designer](#).

5. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор.

6. Доступ в интернет.

Обратите внимание! Это руководство рассчитано, что вы уже знаете, как разработать своё приложение на языке программирования C#.

Примечание. Пункты выше описывают рекомендуемые инструменты.

Инструкция

1. Вам необходимо получить идентификатор корневой папки рабочего пространства. Этот идентификатор никогда не будет меняться, поэтому его можно сохранить и не делать запрос каждый раз перед загрузкой нового шаблона.

Что бы запросить корневую директорию вызовите метод

<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;
}
```

В этом примере параметр `subscriptionId` указывает идентификатор рабочего пространства (подписки), если он не задан (равен null), то будет возвращен идентификатор рабочего пространства пользователя по умолчанию.

2. Для загрузки нужного шаблона воспользуйтесь методом

<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;
}
```

В этом примере функция получает файл с диска и загружает его в директорию `folderId`, подробнее о параметрах:

- `folderId` — идентификатор директории шаблона.
- `model.Name` — название файла.

В названии файла должно быть расширение `.frx`, если его нет, то добавьте его вручную.

- `model.Content` — содержимое файла закодированное в base64.

Метод возвращает идентификатор загруженного шаблона.

Теперь этот шаблон можно использовать для подготовки (построения) отчёта и экспорта.

Что дальше?

- [Управление правами доступа на примере шаблона.](#)
- [Построение отчёта.](#)

Управление правами доступа на примере шаблона

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

Ограничение доступа к приватным ресурсам очень важная часть Сервисных решений. Гибкая система доступа позволяет задать ограничение или выдать права на каждый ресурс отдельно, задавая круг лиц, которые могут получить к ним доступ.

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. [.NET SDK](#).

3. Редактор C# кода или текстовый редактор, например [Visual Studio Code](#).

4. Шаблон отчёта.

Как загрузить шаблон отчёта можно узнать в статье [Загрузка нового шаблона](#).

5. Активная подписка на один из продуктов: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#) или [МоиОтчеты Публикатор](#).

6. Доступ в интернет.

Обратите внимание! Это руководство рассчитано, что вы уже знаете, как разработать своё приложение на языке программирования C#.

Примечание. Пункты выше описывают рекомендуемые инструменты.

Инструкция

1. Для просмотра текущих прав на ресурс воспользуйтесь методом `<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;
}
```

В этом примере метод возвращает права доступа для владельца шаблона.

Обратите внимание! Классы `<xref:FastReport.Cloud.FilePermissions>` и `<xref:FastReport.Cloud.FilePermission>` отличаются буквой **s** на конце, первый содержит полное описание прав доступа к сущности, второй – только описание для одной из категории.

2. Для изменения разрешений воспользуйтесь методом

<xref:FastReport.Cloud.ITemplatesClient.UpdatePermissionsAsync(System.String,FastReport.Cloud.UpdateFilePermissionsVM,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;
}
```

В этом примере функция даёт возможность анонимным пользователям просматривать информацию о шаблоне и скачивать его.

Обратите внимание! В данном примере мы передаём только те права, которые хотим отредактировать и указываем их в свойстве `Administrate`.

Если нужно изменить сразу несколько категорий прав, можно перечислить их через побитовый оператор ИЛИ (|)

Что дальше?

- [Построение отчёта.](#)
- [Работа с группами.](#)
- [Добавление новых пользователей в рабочее пространство.](#)

Построение отчёта

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрен процесс построения отчёта из шаблона с помощью процессора отчётов Сервисных решений.

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. [.NET SDK](#).
3. Редактор C# кода или текстовый редактор, например [Visual Studio Code](#).
4. Шаблон отчёта.

Как загрузить шаблон отчёта можно узнать в статье [Загрузка нового шаблона](#).

5. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор.
6. Доступ в интернет.

Обратите внимание! Это руководство рассчитано, что вы уже знаете, как разработать своё приложение на языке программирования C#.

Примечание. Пункты выше описывают рекомендуемые инструменты.

Инструкция

1. Вам необходим идентификатор шаблона для его построения. Для его получения воспользуйтесь методом `<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;
}
```

В этом примере функция запрашивает корневую директорию рабочего пространства пользователя по умолчанию, затем запрашивает 10 шаблонов и возвращает первый.

2. Для построения отчёта вам понадобится директория, в которую можно положить отчёт. Запросите корневую директорию отчётов, для этого воспользуйтесь методом `<xref:FastReport.Cloud.IReportFoldersClient.GetRootFolderAsync(System.String,System.Threading.CancellationToken)>`.

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

    FileVM result = await reportFoldersClient.GetRootFolderAsync(subscriptionId);

    return result.Id;
}
```

В этом примере функция запрашивает корневую директорию, идентификатор рабочего пространства можно не задавать. В этом случае будет возвращена корневая директория для рабочего пространства пользователя по умолчанию.

3. Для построения отчёта воспользуйтесь методом `<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;
}
```

В этом примере функция создаёт задачу на подготовку отчёта.

Обратите внимание! Отчёт ещё не построен, но ему уже присвоен идентификатор. Через некоторое время очередь построителя дойдёт до этой задачи и отчёт будет построен.

Если не указать FolderId, то подготовленный отчёт будет сохранён в корневую папку.

4. Для получения информации об отчёте воспользуйтесь методом `<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();
}
```

В этом примере функция запрашивает отчёт по его идентификатору и возвращает статус. Необходимо дождаться статуса `<xref:FastReport.Cloud.FileStatus.Success>`, проверяйте статус каждые несколько секунд.

5. Проверяем статус в цикле и скачиваем файл.

```
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. Для скачивания отчёта воспользуйтесь методом

`<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();
    }
}
```

В этом примере функция запрашивает файл и копирует его в память.

Что дальше?

- [Экспорт отчёта в PDF](#).

Параметры отчёта

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрен процесс передачи параметров в отчёт, которые представляют собой словарь параметров, предоставляемых при формировании отчёта. Больше информации на эту тему можно найти в руководстве FastReport .NET в разделе [Параметры отчёта](#).

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. [.NET SDK](#).

3. Редактор C# кода или текстовый редактор, например [Visual Studio Code](#).

4. Шаблон с указанными параметрами отчёта.

Его можно подготовить с помощью бесплатной программы [FastReport Community Designer](#).

5. Активная подписка на один из продуктов: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#) или [МоиОтчеты Публикатор](#).

6. Доступ в интернет.

Обратите внимание! Это руководство рассчитано, что вы уже знаете, как разработать своё приложение на языке программирования C#.

Обратите внимание! Пункты выше описывают рекомендуемые инструменты.

Передача параметров в отчёт

Параметры могут быть переданы в следующих случаях:

- при подготовке отчёта;
- при экспорте из шаблона;
- при работе с задачами.

Рассмотрим передачу параметров на примере подготовки отчёта. Подробная инструкция о том, как подготовить отчёт, содержится в разделе [Построение отчёта](#).

```
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", "Значение1" },
            { "Parameter2", "Значение2" }
        }
    };

    // Добавить новый параметр в словарь
    task.ReportParameters.Add("Parameter3", "Значение3");

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

Экспорт отчёта в PDF

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрен процесс экспорта отчёта с помощью процессора отчётов Сервисных решений.

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. [.NET SDK](#).

3. Редактор C# кода или текстовый редактор, например [Visual Studio Code](#).

4. Отчёт.

Как построить отчёт вы можете узнать в статье [Построение отчёта](#).

5. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор.

6. Доступ в интернет.

Обратите внимание! Это руководство рассчитано, что вы уже знаете, как разработать своё приложение на языке программирования C#.

Примечание. Пункты выше описывают рекомендуемые инструменты.

Замечание

Экспорт отчёта можно сделать напрямую из шаблона, без промежуточного сохранения отчёта. Для этого выполните те же команды для шаблона отчёта, заменив `Report` в методах на `Template`, также используйте идентификатор шаблона, а не отчёта.

Инструкция

1. Вам будет необходим идентификатор отчёта для экспорта в PDF. Для его получения воспользуйтесь методом `<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;
}

```

В этом примере функция запрашивает корневую директорию рабочего пространства пользователя по умолчанию, затем запрашивает 10 отчётов и возвращает первый.

- Для экспорта отчёта понадобится директория, в которую следует сохранить экспорт.

Получите корневую директорию экспортов, для этого воспользуйтесь методом [<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;
}

```

В этом примере функция запрашивает корневую директорию, идентификатор рабочего пространства можно не задавать. В этом случае будет возвращена корневая директория для рабочего пространства пользователя по умолчанию.

- Для экспорта отчёта воспользуйтесь методом [<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` — название результирующего файла. Если не указать расширение, или указать его неправильно - сервер заменит его самостоятельно.
- `FolderId` — идентификатор директории куда будет помещён экспорт. Если оставить пустым, то экспорт будет помещён в корневую папку для экспортов в рабочем пространстве.
- `Format` — формат экспорта.
- `ExportParameters` — параметры экспорта. Задаются аналогично параметрам экспорта из библиотеки `FastReport .NET`. Более подробное описание хранится в документации пользователя в разделе [параметры экспорта](#).

В этом примере функция создаёт задачу на экспорт отчёта.

Обратите внимание! Отчёт ещё не экспортирован, но уже сейчас экспорту присвоен идентификатор. Через некоторое время очередь построителя дойдёт до этой задачи и отчёт будет экспортирован.

Если не указать `FolderId`, то экспорт будет сохранён в корневую папку.

4. Для получения информации о файле воспользуйтесь методом `<xref:FastReport.Cloud.IExportsClient.GetFilesAsync(System.String,System.Threading.CancellationToken)>`.

```

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

    ExportVM result = await exportsClient.GetFilesAsync(exportId);

    return result.Status.GetValueOrDefault();
}

```

В этом примере функция запрашивает экспорт по его идентификатору и возвращает статус. Необходимо дожидаться статуса `<xref:FastReport.Cloud.FileStatus.Success>`. Проверяйте статус каждые несколько секунд.

5. Проверяем статус в цикле и скачиваем экспорт.

```
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. Для скачивания отчёта воспользуйтесь методом

`<xref:FastReport.Cloud.IDownloadClient.GetExportAsync(System.String,System.Threading.CancellationToken)>`.

```
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();
    }
}
```

В этом примере функция запрашивает файл и копирует его в память.

Что дальше?

- [Работа с группами.](#)
- [Добавление новых пользователей в рабочее пространство.](#)

Добавление новых пользователей в рабочее пространство

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрен процесс добавления нового пользователя в подписку, получения списка пользователей в подписке и удаления пользователя из подписки.

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. [.NET SDK](#).
3. Редактор C# кода или текстовый редактор, например [Visual Studio Code](#).
4. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор, в которой есть два слота для пользователя.
5. Доступ в интернет.

Обратите внимание! Это руководство рассчитано, что вы уже знаете, как разработать своё приложение на языке программирования C#.

Примечание. Пункты выше описывают рекомендуемые инструменты.

Замечание

Обратите внимание! Добавить пользователя в рабочее пространство возможно только по идентификатору пользователя.

Инструкция

1. Для добавления нового пользователя в рабочее пространство необходим идентификатор рабочего пространства(подписки).

Получите идентификатор рабочего пространства используя метод `<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;
}
```

В этом примере функция запрашивает первые 10 рабочих пространств (подписок) из список рабочих пространств пользователя, выбирает первое рабочее пространство и возвращает его идентификатор.

Рабочее пространство всегда соотносится с одной подпиской, поэтому у них совпадает идентификатор.

- Для добавления нового пользователя воспользуйтесь методом `<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);
}
```

В этом примере функция добавляет пользователя с идентификатором `userId` в рабочее пространство с идентификатором `subscriptionId`.

- Для получения списка пользователей рабочего пространства воспользуйтесь методом `<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);
}
```

В этом примере функция запрашивает 10 первых пользователей из рабочего пространства с идентификатором `subscriptionId`.

- Для удаления пользователя из рабочего пространства воспользуйтесь методом `<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);
}
```

В этом методе функция удаляет пользователя с идентификатором `userId` из рабочего пространства с идентификатором `subscriptionId`.

Что дальше?

- [Работа с группами](#)
- [Помощь и обратная связь](#)

Работа с группами

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрен процесс создания новой группы, добавление пользователя в группу и получение списка пользователей группы.

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. [.NET SDK](#).
3. Редактор C# кода или текстовый редактор, например [Visual Studio Code](#).
4. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор, в которой есть два слота для пользователя.
5. Доступ в интернет.

Обратите внимание! Это руководство рассчитано, что вы уже знаете, как разработать своё приложение на языке программирования C#.

Примечание. Пункты выше описывают рекомендуемые инструменты.

Замечание

Важно! Добавить пользователя в группу возможно только если пользователь существует в рабочем пространстве.

Обратите внимание! Добавить пользователя в группу возможно только по его идентификатору.

Инструкция

1. Для создания новой группы необходим идентификатор рабочего пространства и название новой группы.

Для получения идентификатора рабочего пространства воспользуйтесь методом `<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;
}
```

В этом примере функция запрашивает первые 10 рабочих пространств из списка рабочих пространств

пользователя, выбирает первое рабочее пространство и возвращает его идентификатор.

Идентификатор рабочего пространства и подписки совпадает, т.к. к каждому рабочему пространству соотносится одна подписка.

2. Для создания новой группы воспользуйтесь методом

<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;
}
```

В этом примере функция создаёт новую группу с именем группы `groupName` для рабочего пространства с идентификатором `subscriptionId`, в результате функция вернёт идентификатор созданной группы.

3. Для добавления нового пользователя в группу воспользуйтесь методом

<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);
}
```

В этом примере функция добавляет пользователя с идентификатором `userId` в группу с идентификатором `groupId`.

4. Для получения списка пользователей в группе воспользуйтесь методом

<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);
}
```

В этом примере функция запрашивает первые 10 пользователей из группы с идентификатором `groupId`.

Что дальше?

- [Помощь и обратная связь](#)

Работа с задачами

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

Задачи в Сервисных решениях - это действия по преобразованию и доставке документов потребителям. Они подробно описаны в разделе [Задачи](#).

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. [.NET SDK](#).

3. Редактор C# кода или текстовый редактор, например [Visual Studio Code](#).

4. Шаблон отчёта.

Его можно подготовить с помощью бесплатной программы [FastReport Community Designer](#).

5. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор.

6. Доступ в интернет.

Обратите внимание! Это руководство рассчитано, что вы уже знаете, как разработать своё приложение на языке программирования C#.

Обратите внимание! Пункты выше описывают рекомендуемые инструменты.

Общие сведения

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрим типы задач и как с ними работать.

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. [.NET SDK](#).

3. Редактор C# кода или текстовый редактор, например [Visual Studio Code](#).

4. Шаблон отчёта.

Его можно подготовить с помощью бесплатной программы [FastReport Community Designer](#).

5. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор.

6. Доступ в интернет.

Обратите внимание! Это руководство рассчитано, что вы уже знаете, как разработать своё приложение на языке программирования C#.

Обратите внимание! Пункты выше описывают рекомендуемые инструменты.

Во первых, нужно создать HttpClient, который мы далее будем использовать:

```
var httpClient = new HttpClient();
httpClient.BaseAddress = new Uri("https://облако.моиотчеты.рф");
httpClient.DefaultRequestHeaders.Authorization = new FastReportCloudApiKeyHeader(ApiKey);
```

Далее получим подписку, относительно которой будем работать с задачами:

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

Теперь нам нужно создать клиент для работы с задачами:

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

Теперь можно приступать непосредственно к работе с задачами.

Хранимые задачи

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

Как указано в [предыдущем разделе](#), в Сервисных решениях есть возможность создавать задачи, которые сохраняются в базе данных и могут быть запущены позже. Эти задачи называются хранимыми задачами.

Рассмотрим подробнее, как создавать, обновлять и запускать хранимые задачи с помощью C# SDK.

О ViewModel и типах

При работе вы будете передавать разные ViewModel (Модели Представления) в методы SDK. Рассмотрим какие модели представления могут использоваться.

Модели Представления для создания задач

Модели Представления для создания задач-преобразователей:

- CreatePrepareTaskVM
- CreateExportTemplateTaskVM
- CreateExportReportTaskVM

Модели Представления для создания задач-транспортов:

- CreateEmailTaskVM
- CreateWebhookTaskVM
- CreateFTPUploadTaskVM
- CreateS3UploadTaskVM

Модели Представления для создания иных задач:

- CreateFetchTaskVM

Модели Представления для обновления свойств сохранённых задач

Модели Представления для обновления задач-преобразователей:

- UpdatePrepareTaskVM
- UpdateExportTemplateTaskVM
- UpdateExportReportTaskVM

Модели Представления для обновления задач-транспортов:

- UpdateEmailTaskVM
- UpdateWebhookTaskVM
- UpdateFTPUploadTaskVM
- UpdateS3UploadTaskVM

Модели Представления для обновления иных задач:

- UpdateFetchTaskVM

Модель Представления для обновления прав задач:

- UpdateTaskPermissionsVM

Создание задачи

Для создания задач следует использовать метод `CreateTask` или `CreateTaskAsync`. Он может принимать любой класс-наследник `TaskBaseVM`:

- `CreatePrepareTaskVM`
- `CreateExportTemplateTaskVM`
- `CreateExportReportTaskVM`
- `CreateFetchTaskVM`
- `CreateEmailTaskVM`
- `CreateWebhookTaskVM`
- `CreateFTPUUploadTaskVM`
- `CreateS3UploadTaskVM`.

Рассмотрим создание задачи подготовки отчёта, сохранения его в папку и последующего экспорта в PDF с сохранением в папку:

```
await tasksClient.CreateTaskAsync(new CreatePrepareTaskVM
{
    Name = "Моя первая задача",
    Type = TaskType.Prepare,
    InputFile = new InputFileVM
    {
        EntityId = "{templateId}"
    },
    OutputFile = new OutputFileVM
    {
        FileName = "Мой первый отчёт сгенерированный задачей.frx",
        FolderId = "{идентификатор папки с отчётами}"
    },
    Exports = new List<CreateExportReportTaskVM>
    {
        new CreateExportReportTaskVM
        {
            Format = ExportFormat.Pdf,
            OutputFile = new OutputFileVM
            {
                FileName = "pdfИзFrxИзFrx.pdf",
                FolderId = "{идентификатор папки с экспортами}"
            }
        }
    }
});
```

В поля `EntityId` и `FolderId` следует записывать реальные идентификаторы объектов. Иначе задача будет прервана с ошибкой.

Обратите внимание! Если не указать `OutputFile`, то он будет сохранён во временную папку. Если указать пустой `OutputFile` - в корневую папку. Если указать `id` папки - в неё.

Далее рассмотрим создание задачи экспорта отчёта с последующей отправкой на почту:

```

var currentTask = await tasksClient.CreateTaskAsync(new CreateExportTemplateTaskVM
{
    SubscriptionId = subscriptionId,
    Name = "Задача отправки PDF по почте",
    InputFile = new InputFileVM
    {
        EntityId = template.Id
    },
    ReportParameters = new Dictionary<string, string>
    {
        { "param1", "val1" },
        { "param2", "val2" },
        { "param3", "val3" }
    },
    Format = ExportFormat.Pdf,
    ExportParameters = new Dictionary<string, string>
    {
        { "PrintOptimized", "true" },
        { "JpegQuality", "90" }
    },
    Transports = new List<CreateTransportTaskBaseVM>
    {
        new CreateEmailTaskVM
        {
            Name = "Задача рассылки на почту",
            From = "admin@company.ru",
            To = new List<string> { "user@company.ru", "seller@company.ru" },
            Subject = "Важный отчет",
            IsBodyHtml = false,
            Username = "admin@company.ru",
            Password = "parol123",
            Server = "smtp.company.ru",
            Body = "К письму прикреплен важный отчет.",
            Port= 587,
            EnableSsl = true
        }
    }
});

```

В поля EntityId и FolderId следует записывать реальные идентификаторы объектов. Иначе задача будет прервана с ошибкой.

Обратите внимание! Если не указать OutputFile, то он будет сохранён во временную папку. Если указать пустой OutputFile - в корневую папку. Если указать id папки - в неё.

Запуск задач по расписанию

Чтобы настроить запуск задачи по расписанию, необходимо указать следующие поля:

- CronExpression
- StartsOn
- Ends

CronExpression - строка, содержащая cron-выражение, определяющее расписание запуска задачи.

StartsOn - дата и время, с которого начинается выполнение задачи по расписанию (в часовом поясе пользователя).

Ends - объект, содержащий дату и время, до которого будет выполняться задача по расписанию (**On**) или количество повторений (**After**). Поле **On** задаётся в часовом поясе пользователя. Если не указано ни поле **After** ни поле **On**, то задача будет выполняться бесконечно.

Пример создания задачи, запускаемой по расписанию:

```
var currentTask = await tasksClient.CreateTaskAsync(new CreateExportTemplateTaskVM
{
    SubscriptionId = subscriptionId,
    Name = "Задача отправки PDF по почте",
    InputFile = new InputFileVM
    {
        EntityId = template.Id
    },
    Format = ExportFormat.Pdf,
    CronExpression = "**/5 * * * *",
    StartsOn = DateTime.Now.AddMinutes(1),
    Ends = new CreateTaskEndVM
    {
        On = DateTime.Now.AddDays(1)
    }
});
```

В этом примере задача будет запускаться каждые 5 минут, начиная с даты создания задачи и закончит выполнение через один день.

Чтобы сделать так, чтобы задача выполнялась, например, 10 раз, нужно заменить поле `On` на поле `After`:

```
CronExpression = "**/5 * * * *",
StartsOn = DateTime.UtcNow,
Ends = new CreateTaskEndVM
{
    After = 10
}
```

Получение списка задач

```
// Получить первые 100 задач из рабочего пространства
var tasks = await tasksClient.GetListAsync(0, 100, subscription.Id);
```

Редактирование задачи

После того как задача создана можно изменить некоторые её параметры.

```
// Редактирование задачи
var editedTask = await tasksClient.UpdateTaskAsync(currentTaskId, new UpdateExportTemplateTaskVM
{
    Format = ExportFormat.Docx,
    ReportParameters = new Dictionary<string, string>
    {
        { "param1", "val1" },
        { "param2", "val2" }
    },
    TransportIds = new List<string> { "{transportId}" }
});
```

Обратите внимание, что при обновлении задачи нужно указывать `TransportIds` - список идентификаторов задач-транспортов (Email, FTP, Webhook, S3), а не сами задачи. Саму задачу можно создать заранее как отдельно так и при создании задачи-трансформера. Подробнее про их создание читайте в разделе [Создание задач-транспортов](#).

Выполнение задачи по указанному идентификатору

```
// Запуск задачи по идентификатору  
await tasksClient.RunTaskByIdAsync(id);
```

Удаление задач из хранилища

```
// Удалить все задачи  
foreach(var t in tasks.Tasks)  
{  
    await tasksClient.DeleteTaskAsync(t.Id);  
}
```

Задачи, запускаемые из тела запроса

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

Как указано в [предыдущем разделе](#), в Сервисных решениях есть возможность запускать задачи без предварительного сохранения. Для этого следует использовать метод `RunTask`, передав в него модель, отнаследованную от `UpdateTaskBaseVM`.

Например, для того, чтобы построить PDF-документ и отправить его по FTP нужно передать в методе `RunTask` такие данные как идентификатор шаблона (`EntityId`) или сам шаблон (`Content`) в объект `InputFile`, а также данные для подключения к FTP-серверу.

Рассмотрим подробнее, как работать с такими задачами с помощью C# SDK.

О ViewModel и типах

При работе вы будете передавать разные ViewModel (Модели Представления) в методы SDK. Рассмотрим какие модели представления могут использоваться.

Модели Представления для запуска задач на лету (без сохранения)

Модели Представления для запуска задач-преобразователей на лету:

- `RunPrepareTaskVM`
- `RunExportTemplateTaskVM`
- `RunExportReportTaskVM`

Модели Представления для запуска задач-транспортов на лету:

- `RunEmailTaskVM`
- `RunWebhookTaskVM`
- `RunFTPUploadTaskVM`
- `RunS3UploadTaskVM`

Модели Представления для запуска иных задач на лету:

- `RunFetchTaskVM`

Запуск задачи на лету (из тела запроса)

Рассмотрим запуск задачи экспорта отчёта с последующей отправкой на почту:

```
var runTask = await tasksClient.RunTaskAsync(newRunExportTemplateTaskVM
{
    SubscriptionId = subscriptionId,
    InputFile = new RunInputFileVM
    {
        EntityId = template.Id,
        FileName = "Документ"
        //Content = new byte[] { }
    },
    Format = ExportFormat.Docx,
    Transports = new List<RunTransportTaskBaseVM>
    {
        new RunEmailTaskVM
        {
            From = "admin@company.ru",
            To = ["user@company.ru", "seller@company.ru"],
            Subject = "Важный отчет",
            Body = "К письму прикреплен важный отчет.",
            EnableSsl = true,
            Username = "admin@company.ru",
            Password = "parol123",
            Server = "smtp.company.ru",
            Port = 587
        }
    }
});
```

В поле EntityId следует записывать реальные идентификаторы объектов. Иначе задача будет прервана с ошибкой.

Также вместо него в объекте `InputFile` можно передать содержимое шаблона в поле `Content` в виде массива байт.

Задача экспорта шаблона

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрим работу с задачей экспорта шаблона (ExportTemplateTask). Общие инструкции о том как работать с задачами описаны в разделе [Общие сведения](#).

Следует отметить что экспортировать шаблон можно не только с помощью системы задач, но и напрямую через запрос к [контроллеру шаблонов](#). Преимущество использования системы задач в том, что она позволяет не только сохранить документ в папку, но и передать во внешние системы (по Email, FTP, S3, Webhook).

Создание задачи

Рассмотрим создание задачи экспорта шаблона в PDF:

```
// Создание задачи
var currentTask = await tasksClient.CreateTaskAsync(new CreateExportTemplateTaskVM
{
    SubscriptionId = subscriptionId,
    Name = "Задача отправки PDF по почте",
    InputFile = new InputFileVM
    {
        EntityId = template.Id
    },
    ReportParameters = new Dictionary<string, string>
    {
        { "param1", "val1" },
        { "param2", "val2" },
        { "param3", "val3" }
    },
    Format = ExportFormat.Pdf,
    ExportParameters = new Dictionary<string, string>
    {
        { "PrintOptimized", "true" },
        { "JpegQuality", "90" }
    },
    OutputFile = new OutputFileVM
    {
        FileName = "Отчёт"
    }
});
```

В поля EntityId и SubscriptionId следует записывать реальные идентификаторы объектов. Иначе задача будет прервана с ошибкой.

Так как в OutputFile не указана папка для сохранения, то при запуске задачи документ будет сохранён в корневую папку.

Зададим другую папку для сохранения:

```
"OutputFile" : {
  "FolderId": "61cb2226d24478b172103075",
  "FileName": "Отчёт"
}
```

Запуск задачи по идентификатору

Теперь при запуске задачи PDF-файл будет сохранён в указанную папку.

```
await tasksClient.RunTaskByIdAsync(currentTask.Id);
```

Отправка документа после экспорта

Для отправки документа после экспорта можно использовать задачу отправки по Email, FTP, S3 или Webhook.

Рассмотрим пример создания задачи экспорта в PDF с последующей отправкой готового документа по Email.

```
// Создание задачи
var currentTask = await tasksClient.CreateTaskAsync(new CreateExportTemplateTaskVM
{
    SubscriptionId = subscriptionId,
    Name = "Задача отправки PDF по почте",
    InputFile = new InputFileVM
    {
        EntityId = template.Id
    },
    ReportParameters = new Dictionary<string, string>
    {
        { "param1", "val1" },
        { "param2", "val2" },
        { "param3", "val3" }
    },
    Format = ExportFormat.Pdf,
    ExportParameters = new Dictionary<string, string>
    {
        { "PrintOptimized", "true" },
        { "JpegQuality", "90" }
    },
    Transports = new List<CreateTransportTaskBaseVM>
    {
        new CreateEmailTaskVM
        {
            Name = "Задача рассылки на почту",
            From = "admin@company.ru",
            To = new List<string> { "user@company.ru", "seller@company.ru" },
            Subject = "Важный отчет",
            IsBodyHtml = false,
            Username = "admin@company.ru",
            Password = "parol123",
            Server = "smtp.company.ru",
            Body = "К письму прикреплен важный отчет.",
            Port = 587,
            EnableSsl = true
        }
    }
});
```

Редактирование задачи

После того как задача создана можно изменить некоторые её параметры.

```
// Редактирование задачи
var editedTask = await tasksClient.UpdateTaskAsync(currentTask.Id, new UpdateExportTemplateTaskVM
{
    Format = ExportFormat.Docx,
    ReportParameters = new Dictionary<string, string>
    {
        { "param1", "val1" },
        { "param2", "val2" }
    },
    TransportIds = new List<string> { "{transportId}" }
});
```

Обратите внимание, что при обновлении задачи нужно указывать `TransportIds` - список идентификаторов задач-транспортов (Email, FTP, Webhook, S3), а не сами задачи. Саму задачу можно создать заранее как отдельно так и при создании задачи-трансформера. Подробнее про их создание читайте в разделе [Создание задач-транспортов](#).

Выполнение задачи из тела запроса

```
var runTask = await tasksClient.RunTaskAsync(new RunExportTemplateTaskVM
{
    SubscriptionId = subscriptionId,
    InputFile = new RunInputFileVM
    {
        EntityId = template.Id,
        FileName = "Документ"
        //Content = new byte[] { }
    },
    Format = ExportFormat.Docx,
    Transports = new List<RunTransportTaskBaseVM>
    {
        new RunEmailTaskVM
        {
            From = "admin@company.ru",
            To = ["user@company.ru", "seller@company.ru"],
            Subject = "Важный отчет",
            Body = "К письму прикреплен важный отчет.",
            EnableSsl = true,
            Username = "admin@company.ru",
            Password = "parol123",
            Server = "smtp.company.ru",
            Port = 587
        }
    }
});
```

Таким образом в результате выполнения этого запроса будет создан PDF-документ и отправлен по почте.

Обратите внимание! В этом случае создание PDF-документа и его отправка по почте будет выполнена непосредственно этим запросом и задача не будет сохранена в базе данных.

Задача экспорта отчета

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрим работу с задачей экспорта шаблона (ExportReportTask). Общие инструкции о том как работать с задачами описаны в разделе [Общие сведения](#).

Следует отметить что экспортировать шаблон можно не только с помощью системы задач, но и напрямую через запрос к [контроллеру отчетов](#). Преимущество использования системы задач в том, что она позволяет не только сохранить документ в папку, но и передать во внешние системы (по Email, FTP, S3, Webhook).

Создание задачи

Рассмотрим создание задачи экспорта отчета в PDF:

```
// Создание задачи
var currentTask = await tasksClient.CreateTaskAsync(new CreateExportReportTaskVM
{
    SubscriptionId = subscriptionId,
    Name = "Задача отправки PDF по почте",
    InputFile = new InputFileVM
    {
        EntityId = template.Id
    },
    Format = ExportFormat.Pdf,
    ExportParameters = new Dictionary<string, string>
    {
        { "PrintOptimized", "true" },
        { "JpegQuality", "90" }
    },
    OutputFile = new OutputFileVM
    {
        FileName = "Отчёт"
    }
});
```

В поля EntityId и SubscriptionId следует записывать реальные идентификаторы объектов. Иначе задача будет прервана с ошибкой.

Так как в OutputFile не указана папка для сохранения, то при запуске задачи документ будет сохранён в корневую папку.

Зададим другую папку для сохранения:

```
"OutputFile" : {
  "FolderId": "61cb2226d24478b172103075",
  "FileName": "Отчёт"
}
```

Запуск задачи по идентификатору

Теперь при запуске задачи PDF-файл будет сохранён в указанную папку.

```
await tasksClient.RunTaskByIdAsync(currentTask.Id);
```

Отправка документа после экспорта

Для отправки документа после экспорта можно использовать задачу отправки по Email, FTP, S3 или Webhook.

Рассмотрим пример создания задачи экспорта в PDF с последующей отправкой готового документа по Email.

```
// Создание задачи
var currentTask = await tasksClient.CreateTaskAsync(new CreateExportReportTaskVM
{
    SubscriptionId = subscriptionId,
    Name = "Задача отправки PDF по почте",
    InputFile = new InputFileVM
    {
        EntityId = template.Id
    },
    Format = ExportFormat.Pdf,
    ExportParameters = new Dictionary<string, string>
    {
        { "PrintOptimized", "true" },
        { "JpegQuality", "90" }
    },
    Transports = new List<CreateTransportTaskBaseVM>
    {
        new CreateEmailTaskVM
        {
            Name = "Задача рассылки на почту",
            From = "admin@company.ru",
            To = new List<string> { "user@company.ru", "seller@company.ru" },
            Subject = "Важный отчет",
            IsBodyHtml = false,
            Username = "admin@company.ru",
            Password = "parol123",
            Server = "smtp.company.ru",
            Body = "К письму прикреплен важный отчет.",
            Port= 587,
            EnableSsl = true
        }
    }
});
```

Редактирование задачи

После того как задача создана можно изменить некоторые её параметры.

```
// Редактирование задачи
var editedTask = await tasksClient.UpdateTaskAsync(currentTask.Id, new UpdateExportReportTaskVM
{
    Format = ExportFormat.Docx,
    TransportIds = new List<string> { "{transportId}" }
});
```

Обратите внимание, что при обновлении задачи нужно указывать `TransportIds` - список идентификаторов задач-транспортов (Email, FTP, Webhook, S3), а не сами задачи. Саму задачу можно создать заранее как отдельно так и при создании задачи-трансформера. Подробнее про их создание читайте в разделе [Создание задач-транспортов](#).

Выполнение задачи из тела запроса

```
var runTask = await tasksClient.RunTaskAsync(new RunExportReportTaskVM
{
    SubscriptionId = subscriptionId,
    InputFile = new RunInputFileVM
    {
        EntityId = template.Id,
        FileName = "Документ"
        //Content = new byte[] { }
    },
    Format = ExportFormat.Docx,
    Transports = new List<RunTransportTaskBaseVM>
    {
        new RunEmailTaskVM
        {
            From = "admin@company.ru",
            To = ["user@company.ru", "seller@company.ru"],
            Subject = "Важный отчет",
            Body = "К письму прикреплен важный отчет.",
            EnableSsl = true,
            Username = "admin@company.ru",
            Password = "parol123",
            Server = "smtp.company.ru",
            Port = 587
        }
    }
});
```

Таким образом в результате выполнения этого запроса будет создан PDF-документ и отправлен по почте.

Обратите внимание! В этом случае создание PDF-документа и его отправка по почте будет выполнена непосредственно этим запросом и задача не будет сохранена в базе данных.

Задача подготовки отчета

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрим работу с задачей подготовки отчета (PrepareTemplateTask). Общие инструкции о том как работать с задачами описаны в разделе [Общие сведения](#).

Следует отметить что подготовить отчет можно не только с помощью системы задач, но и напрямую через запрос к [контроллеру шаблонов](#). Преимущество использования системы задач в том, что она позволяет не только сохранить документ в папку, но и передать во внешние системы (по Email, FTP, S3, Webhook).

Создание задачи

Рассмотрим создание задачи подготовки отчета:

```
// Создание задачи
var currentTask = await tasksClient.CreateTaskAsync(new CreatePrepareTemplateTaskVM
{
    SubscriptionId = subscriptionId,
    Name = "Задача подготовки отчета",
    InputFile = new InputFileVM
    {
        EntityId = template.Id
    },
    OutputFile = new OutputFileVM
    {
        FileName = "Отчет.fpx"
    }
});
```

В поля EntityId и SubscriptionId следует записывать реальные идентификаторы объектов. Иначе задача будет прервана с ошибкой.

Так как в OutputFile не указана папка для сохранения, то при запуске задачи документ будет сохранён в корневую папку.

Зададим другую папку для сохранения:

```
"OutputFile" : {
  "FolderId": "61cb2226d24478b172103075",
  "FileName": "Отчёт"
}
```

Запуск задачи по идентификатору

Теперь при запуске задачи PDF-файл будет сохранён в указанную папку.

```
await tasksClient.RunTaskByIdAsync(currentTask.Id);
```

Экспорт отчета после подготовки

Для экспорта отчета после подготовки можно использовать задачу экспорта.

Рассмотрим пример создания задачи подготовки отчета с последующим экспортом в PDF.

```
// Создание задачи
var currentTask = await tasksClient.CreateTaskAsync(new CreatePrepareTemplateTaskVM
{
    SubscriptionId = subscriptionId,
    Name = "Задача подготовки отчета",
    InputFile = new InputFileVM
    {
        EntityId = template.Id
    },
    OutputFile = new OutputFileVM
    {
        FileName = "Отчет.fpx"
    },
    Exports = new List<CreateExportReportTaskVM>()
    {
        new CreateExportReportTaskVM
        {
            SubscriptionId = subscriptionId,
            Name = "Задача экспорта в PDF",
            Format = ExportFormat.Pdf,
            OutputFile = new OutputFileVM
            {
                FileName = "Документ.pdf"
            }
        }
    }
});
```

Редактирование задачи

После того как задача создана можно изменить некоторые её параметры.

```
// Редактирование задачи
var editedTask = await tasksClient.UpdateTaskAsync(currentTask.Id, new UpdatePrepareTemplateTaskVM
{
    ReportParameters = new Dictionary<string, string>
    {
        { "param1", "val1" },
        { "param2", "val2" }
    },
    TransportIds = new List<string> { "{transportId}" }
});
```

Обратите внимание, что при обновлении задачи нужно указывать `TransportIds` - список идентификаторов задач-транспортов (Email, FTP, Webhook, S3), а не сами задачи. Саму задачу можно создать заранее как отдельно так и при создании задачи-трансформера. Подробнее про их создание читайте в разделе [Создание задач-транспортов](#).

Выполнение задачи из тела запроса

```
var runTask = await tasksClient.RunTaskAsync(new RunPrepareTemplateTaskVM
{
    SubscriptionId = subscriptionId,
    InputFile = new RunInputFileVM
    {
        EntityId = template.Id,
        FileName = "Документ"
        //Content = new byte[] { }
    },
    Transports = new List<RunTransportTaskBaseVM>
    {
        new RunEmailTaskVM
        {
            From = "admin@company.ru",
            To = ["user@company.ru", "seller@company.ru"],
            Subject = "Важный отчет",
            Body = "К письму прикреплен важный отчет.",
            EnableSsl = true,
            Username = "admin@company.ru",
            Password = "parol123",
            Server = "smtp.company.ru",
            Port = 587
        }
    }
});
```

Таким образом в результате выполнения этого запроса будет подготовлен отчет и отправлен по почте.

Обратите внимание! В этом случае создание отчета и его отправка по почте будет выполнена непосредственно этим запросом и задача не будет сохранена в базе данных.

Отправка на Email

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрим способ отправки отчёта на электронную почту. Инструкции о том как работать с задачами описаны в разделе [Общие сведения](#).

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. [.NET SDK](#).

3. Редактор C# кода или текстовый редактор, например [Visual Studio Code](#).

4. Шаблон отчёта.

Его можно подготовить с помощью бесплатной программы [FastReport Community Designer](#).

5. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор.

6. Доступ в интернет.

7. Настроенный и доступный почтовый сервер.

Обратите внимание! Это руководство рассчитано, что вы уже знаете, как разработать своё приложение на языке программирования C#.

Обратите внимание! Пункты выше описывают рекомендуемые инструменты.

Обратите внимание! Данное руководство предполагает, что у вас есть опыт настройки подключения к почтовому серверу.

Создание задачи

Рассмотрим создание задачи отправки шаблона по электронной почте:

```
// Инициализация объекта
CreateEmailTaskVM emailTaskVM = new CreateEmailTaskVM
{
    Name = "Задача отправки по Email",
    SubscriptionId = "{идентификатор рабочего пространства}",
    InputFile = new InputFileVM
    {
        EntityId = "{идентификатор шаблона}",
        Type = FileKind.Template
    },
    From = "admin@company.ru",
    To = ["user@company.ru", "seller@company.ru"],
    Subject = "Важный отчет",
    Body = "К письму прикреплен важный отчет.",
    EnableSsl = true,
    Username = "admin@company.ru",
    Password = "parol123",
    Server = "smtp.company.ru",
    Port = 587
};

// Создание задачи
TaskBaseVM emailTask = await tasksClient.CreateTaskAsync(emailTaskVM);
```

```
// Запуск задачи по идентификатору
await tasksClient.RunTaskByIdAsync(emailTask.Id);
```

В поля EntityId и SubscriptionId следует записывать реальные идентификаторы объектов. Иначе задача будет прервана с ошибкой.

Выполнение задачи из тела запроса

```
RunEmailTaskVM emailTaskVM = new RunEmailTaskVM
{
    SubscriptionId = "{идентификатор рабочего пространства}",
    InputFile = new RunInputFileVM
    {
        EntityId = "{идентификатор шаблона}",
        Type = FileKind.Template
    },
    From = "admin@company.ru",
    To = ["user@company.ru", "seller@company.ru"],
    Subject = "Важный отчет",
    Body = "К письму прикреплен важный отчет.",
    EnableSsl = true,
    Username = "admin@company.ru",
    Password = "parol123",
    Server = "smtp.company.ru",
    Port = 587
};

// Выполнение задачи
TaskMessageIdVM emailTask = await tasksClient.RunTaskAsync(emailTaskVM);
```

Обратите внимание! В этом случае отправка файла по почте будет выполнена непосредственно этим запросом и задача не будет сохранена в базе данных. Также в поле content можно передать отчёт и готовые документы в различных форматах, например, PDF, XLSX и т.д.

Обратите внимание! В emailTask будет возвращён идентификатор сообщения.

Сохранение на FTP-сервер

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрим способ отправки отчёта на FTP сервер. Инструкции о том как работать с задачами описаны в разделе [Общие сведения](#).

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. [.NET SDK](#).

3. Редактор C# кода или текстовый редактор, например [Visual Studio Code](#).

4. Шаблон отчёта.

Его можно подготовить с помощью бесплатной программы [FastReport Community Designer](#).

5. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор.

6. Доступ в интернет или интрасеть, где работает корпоративный сервер.

7. Настроенный и доступный FTP сервер.

Обратите внимание! Это руководство рассчитано, что вы уже знаете, как разработать своё приложение на языке программирования C#.

Обратите внимание! Пункты выше описывают рекомендуемые инструменты.

Обратите внимание! Данное руководство предполагает, что у вас есть опыт настройки и конфигурации FTP сервера для принятия файлов от внешних источников.

Создание задачи

Рассмотрим создание задачи отправки шаблона на FTP-сервер и ее последующий запуск:

```

// Инициализация объекта
CreateFTPUploadTaskVM ftpUploadTaskVM = new CreateFTPUploadTaskVM
{
    Name = "Задача отправки на FTP",
    InputFile = new InputFileVM
    {
        EntityId = "{идентификатор шаблона}",
        Type = FileKind.Template
    },
    FtpHost = "{адрес FTP-сервера}",
    FtpPort = 21,
    FtpUsername = "{имя пользователя FTP-сервера}",
    FtpPassword = "{пароль}",
    UseSFTP = false,
    DestinationFolder = "{/путь_к_папке/}",
    Archive = false,
    ArchiveName = "имя архива",
    SubscriptionId = "{идентификатор рабочего пространства}"
};

// Создание задачи
TaskBaseVM ftpUploadTask = await tasksClient.CreateTaskAsync(ftpUploadTaskVM);

// Запуск задачи по идентификатору
await tasksClient.RunTaskByIdAsync(ftpUploadTask.Id);

```

В поля EntityId и SubscriptionId следует записывать реальные идентификаторы объектов. Иначе задача будет прервана с ошибкой.

Выполнение задачи из тела запроса

```

// Запуск задачи из тела запроса
await tasksClient.RunTaskAsync(new RunFTPUploadTaskVM
{
    InputFile = new RunInputFileVM
    {
        EntityId = "{идентификатор шаблона}",
        Type = FileKind.Template
    },
    FtpHost = "{адрес FTP-сервера}",
    FtpPort = 21,
    FtpUsername = "{имя пользователя FTP-сервера}",
    FtpPassword = "{пароль}",
    UseSFTP = false,
    DestinationFolder = "{/путь_к_папке/}",
    Archive = false,
    ArchiveName = "Архивированный шаблон",
    SubscriptionId = "{идентификатор рабочего пространства}"
});

```

Обратите внимание! В этом случае отправка на FTP будет выполнена непосредственно этим запросом и задача не будет сохранена в базе данных.

Обновление задачи по идентификатору

```
await tasksClient.UpdateTaskAsync("{идентификатор задачи}", new UpdateFTPUploadTaskVM()
{
    InputFile = new RunInputFileVM
    {
        EntityId = "{обновлённый идентификатор шаблона}",
        Type = FileKind.Template
    },
    FtpHost = "{обновлённый адрес FTP-сервера}",
    FtpPort = 21,
    FtpUsername = "{обновлённое имя пользователя FTP-сервера}",
    FtpPassword = "{пароль}",
    UseSFTP = false,
    DestinationFolder = "{обновлённый_путь_к_папке}",
    Archive = false,
    ArchiveName = "обновлённое имя архива"
});
```

Сохранение по Webhook

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрим способ отправки отчёта по вебхуку. Инструкции о том как работать с задачами описаны в разделе [Общие сведения](#).

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. [.NET SDK](#).

3. Редактор C# кода или текстовый редактор, например [Visual Studio Code](#).

4. Шаблон отчёта.

Его можно подготовить с помощью бесплатной программы [FastReport Community Designer](#).

5. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор.

6. Доступ в интернет или интрасеть, где работает корпоративный сервер.

7. Клиентское приложение, принимающее запросы.

Обратите внимание! Это руководство рассчитано, что вы уже знаете, как разработать своё приложение на языке программирования C#.

Обратите внимание! Пункты выше описывают рекомендуемые инструменты.

Обратите внимание! Данное руководство предполагает, что у вас есть клиентское приложение, принимающее и обрабатывающее входящие вебхуки.

Создание задачи

Рассмотрим создание задачи отправки шаблона по вебхуку и ее последующий запуск:

```
// Инициализация объекта
CreateWebhookTaskVM webhookTaskVM = new CreateWebhookTaskVM
{
    Name = "Задача отправки по Webhook",
    InputFile = new InputFileVM
    {
        EntityId = "{идентификатор шаблона}",
        Type = FileKind.Template
    },
    Url = new Uri("https://example.com/"),
    Headers = new Dictionary<string, string> {
        { "Authorization", "Bearer <token>" },
        { "Content-Length", "1238" }
    },
    SubscriptionId = "{идентификатор рабочего пространства}"
};

// Создание задачи
TaskBaseVM webhookTask = await tasksClient.CreateTaskAsync(webhookTaskVM);

// Запуск задачи по идентификатору
await tasksClient.RunTaskByIdAsync(webhookTask.Id);
```

Обратите внимание! В поля EntityId и SubscriptionId следует записывать реальные идентификаторы объектов. Иначе задача будет прервана с ошибкой.

Обратите внимание! В EntityId может передаваться идентификатор шаблона, отчёта и экспорта.

Выполнение задачи из тела запроса

```
// Запуск задачи из тела запроса
await tasksClient.RunTaskAsync(new RunWebhookTaskVM
{
    InputFile = new RunInputFileVM
    {
        EntityId = "{идентификатор шаблона}",
        Type = FileKind.Template
    },
    Url = new Uri("https://example.com/"),
    Headers = new Dictionary<string, string> {
        { "Authorization", "Bearer <token>" },
        { "Content-Type", "multipart/form-data" }
    },
    SubscriptionId = "{идентификатор рабочего пространства}"
});
```

Обратите внимание! В этом случае отправка по вебхуку будет выполнена непосредственно этим запросом и задача не будет сохранена в базе данных.

Обновление задачи по идентификатору

```

await tasksClient.UpdateTaskAsync("{идентификатор старой задачи}", new UpdateWebhookTaskVM()
{
    InputFile = new RunInputFileVM
    {
        EntityId = "{обновлённый идентификатор шаблона}",
        Type = FileKind.Template
    },
    Url = new Uri("{Обновлённый адрес}"),
    Headers = new Dictionary<string, string> {
        { "{Обновлённый заголовок}", "{Обновлённое значение}" },
    }
});

```

Пример запроса, который придёт на эндпоинт вебхука:

```

{
  "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"
    }
  }
}

```

Сохранение в S3-хранилище

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В этой статье рассмотрим способ отправки отчёта на S3-хранилище. Инструкции о том как работать с задачами описаны в разделе [Общие сведения](#).

Приступая к работе

Вам понадобятся следующие инструменты и возможности:

1. Знания по использованию API key в Сервисных решениях.

В этой статье будет пропущена дополнительная информация по аутентификации и авторизации.

2. [.NET SDK](#).

3. Редактор C# кода или текстовый редактор, например [Visual Studio Code](#).

4. Шаблон отчёта.

Его можно подготовить с помощью бесплатной программы [FastReport Community Designer](#).

5. Активная подписка на один из продуктов: МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор.

6. Доступ в интернет или интрасеть, где работает корпоративный сервер.

7. Настроенное и доступное S3-хранилище.

Обратите внимание! Это руководство рассчитано, что вы уже знаете, как разработать своё приложение на языке программирования C#.

Обратите внимание! Пункты выше описывают рекомендуемые инструменты.

Обратите внимание! Данное руководство предполагает, что у вас есть настроенное S3-хранилище для принятия файлов от внешних источников.

Создание задачи

Рассмотрим создание задачи отправки шаблона на S3-хранилище и ее последующий запуск:

```

// Инициализация объекта
CreateS3UploadTaskVM s3UploadTaskVM = new CreateS3UploadTaskVM
{
    Name = "Задача отправки на S3",
    InputFile = new InputFileVM
    {
        EntityId = "{идентификатор шаблона}",
        Type = FileKind.Template
    },
    AccessKey = "{публичный_ключ_доступа}",
    SecretKey = "{секретный_ключ_доступа}",
    Url = "https://example.com",
    BucketName = "{название_бакета}",
    UseAws = true,
    EnableSsl = true,
    Region = "us-east-1",
    DestinationFolder = "{/путь_к_папке/}",
    SubscriptionId = "{идентификатор рабочего пространства}"
};

// Создание задачи
TaskBaseVM s3UploadTask = await tasksClient.CreateTaskAsync(s3UploadTaskVM);

// Запуск задачи по идентификатору
await tasksClient.RunTaskByIdAsync(s3UploadTask.Id);

```

В поля EntityId и SubscriptionId следует записывать реальные идентификаторы объектов. Иначе задача будет прервана с ошибкой.

Выполнение задачи из тела запроса

```

// Запуск задачи из тела запроса
await tasksClient.RunTaskAsync(new RunS3UploadTaskVM
{
    InputFile = new RunInputFileVM
    {
        EntityId = "{идентификатор шаблона}",
        Type = FileKind.Template
    },
    AccessKey = "{публичный_ключ_доступа}",
    SecretKey = "{секретный_ключ_доступа}",
    Url = "https://example.com",
    BucketName = "{название_бакета}",
    UseAws = true,
    EnableSsl = true,
    Region = "us-east-1",
    DestinationFolder = "{/путь_к_папке/}",
    SubscriptionId = "{идентификатор рабочего пространства}"
});

```

Обратите внимание! В этом случае отправка на S3 будет выполнена непосредственно этим запросом и задача не будет сохранена в базе данных.

Обновление задачи по идентификатору

```
await tasksClient.UpdateTaskAsync("{идентификатор задачи}", new UpdateS3UploadTaskVM()
{
    InputFile = new RunInputFileVM
    {
        EntityId = "{обновлённый идентификатор шаблона}",
        Type = FileKind.Template
    },
    AccessKey = "{обновленный публичный_ключ_доступа}",
    SecretKey = "{обновленный секретный_ключ_доступа}",
    Url = "обновлённый адрес S3-хранилища",
    BucketName = "{обновлённое название_бакета}",
    UseAws = true,
    EnableSsl = true,
    Region = "{обновлённый регион}",
    DestinationFolder = "{/обновлённый_путь_к_папке/}"
});
```

Руководство по интеграции статического предварительного просмотра

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

Это руководство предоставляет подробные инструкции по интеграции отчетов Сервисных решений в ваши веб-приложения с использованием iframe.

Базовая настройка iframe

Сервис статического предварительного просмотра Сервисных решений разработан таким образом, чтобы его было легко внедрить в ваши веб-приложения с помощью iframe. Это позволяет легко интегрировать предварительный просмотр отчетов в существующие пользовательские интерфейсы.

```
<iframe
  src="https://yourdomain.ru/staticpreview/t/your-template-id?apikey=your-api-key"
  width="100%"
  height="600px"
  frameborder="0"
  style="border: 1px solid #ccc;">
</iframe>
```

Структура URL

URL-адрес источника iframe следует такому шаблону:

```
https://yourdomain.ru/staticpreview/{type}/{report-id}?{parameters}
```

Пример: `https://xn--80ab2acne.xn--e1afliby2b0b.xn--p1ai/staticpreview/{type}/{report-id}?{parameters}`

Где:

- `{type}`: `t` для шаблонов или `r` для отчетов
- `{report-id}`: идентификатор шаблона или отчета
- `{parameters}`: параметры строки запроса для аутентификации и конфигурации

Параметры аутентификации

Аутентификация по API-ключу

```
<!-- Для аутентификации по API-ключу -->
<iframe src="https://yourdomain.ru/staticpreview/t/template-123?apikey=your-api-key-here"></iframe>
```

Параметр: `apikey`

- Обязательный:** Да (для аутентификации по API-ключу)
- Описание:** Ваш API-ключ МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор
- Пример:** `apikey=dw3ized7i7i97qjsrg4mesn3n8r1bpi54wbyhywpfier7er3nqjo`

Аутентификация по ключу доступа

```
<!-- Для аутентификации по ключу доступа -->
<iframe src=" https://yourdomain.ru/staticpreview/r/report-456?accessKey=your-access-key"></iframe>
```

Параметр: `accessKey`

- **Обязательный:** Да (для аутентификации по ключу доступа)
- **Описание:** Временный ключ доступа для конкретного отчета
- **Пример:** `accessKey=temp-key-abc123def456`

Параметры отчета

Для шаблонов, требующих параметры, используйте формат параметра `rp`:

```
<iframe src="https://yourdomain.ru/staticpreview/t/sales-report?apikey=your-key&rp=StartDate:2023-01-01&rp=EndDate:2023-12-31&rp=Region:North"></iframe>
```

Параметр: `rp` (Параметры отчета)

- **Формат:** `rp=ИмяПараметра:ЗначениеПараметра`
- **Множественные:** Используйте несколько параметров `rp` для разных значений
- **Описание:** Устанавливает начальные значения параметров отчета
- **Примеры:**
 - Дата: `rp=StartDate:2023-01-01`
 - Строка: `rp=CustomerName:John Doe`
 - Число: `rp=MinAmount:1000`
 - Логическое значение: `rp=IncludeInactive:true`

Параметры должны быть указаны внутри шаблона отчета.

Локализация пользовательского интерфейса

Сервис автоматически определяет язык из:

1. Конфигурации сервера (если доступно)
2. Настроек языка браузера

Локализация отчета

Локализация отчета будет использоваться из самих параметров отчета. Или из параметров рабочей области.

Примеры адаптивных iframe

Адаптивный на всю ширину

```
<div style="position: relative; width: 100%; height: 0; padding-bottom: 75%;">
  <iframe
    src=" https://yourdomain.ru/staticpreview/t/template-123?apikey=your-key"
    style="position: absolute; top: 0; left: 0; width: 100%; height: 100%; border: none;"
    allowfullscreen>
  </iframe>
</div>
```

Интеграция в модальное окно/диалог

```

<div class="modal" id="reportModal">
  <div class="modal-content" style="width: 90%; height: 90%;">
    <iframe
      id="reportFrame"
      src=" https://yourdomain.ru/staticpreview/t/template-123?apikey=your-key"
      style="width: 100%; height: 100%; border: none;">
    </iframe>
  </div>
</div>

<script>
function openReport(templateId, apiKey, parameters = {}) {
  const baseUrl = ' https://yourdomain.ru/staticpreview/t/ ' + templateId;
  const params = new URLSearchParams({ apiKey: apiKey });

  // Добавление параметров отчета
  Object.entries(parameters).forEach(([key, value]) => {
    params.append('rp', `${key}:${value}`);
  });

  document.getElementById('reportFrame').src = baseUrl + '?' + params.toString();
  document.getElementById('reportModal').style.display = 'block';
}

// Использование
openReport('sales-report', 'your-api-key', {
  StartDate: '2023-01-01',
  EndDate: '2023-12-31',
  Region: 'North America'
});
</script>

```

Рекомендуемые размеры iframe

- **Минимальная ширина:** 400px (для совместимости с мобильными устройствами)
- **Минимальная высота:** 300px (для отображения хотя бы одной страницы)
- **Рекомендуемые:** 800px × 600px или больше
- **На весь экран:** Используйте адаптивный дизайн для лучшего опыта

Соображения безопасности

1. **Защита API-ключа:** Никогда не раскрывайте API-ключи в клиентском коде в продакшене
2. **Ключи доступа:** По возможности используйте временные ключи доступа
3. **HTTPS:** Всегда используйте HTTPS для источников iframe в продакшене
4. **Заголовки CSP:** Настройте политику безопасности контента для разрешения источников iframe

```
<iframe src="..."></iframe>
```

Справочник параметров

ПАРАМЕТР	ТИП	ОБЯЗАТЕЛЬНЫЙ	ОПИСАНИЕ	ПРИМЕР
apikey	Строка	Да*	API-ключ МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор	apikey=your-api-key
accessKey	Строка	Да*	Временный ключ доступа	accessKey=temp-key-123

ПАРАМЕТР	ТИП	ОБЯЗАТЕЛЬНЫЙ	ОПИСАНИЕ	ПРИМЕР
rp	Строка	Нет	Параметр отчета в формате <code>Имя:Значение</code>	<code>rp=StartDate:2023-01-01</code>

* Требуется либо `apiKey`, либо `accessKey` в зависимости от метода аутентификации.

Руководство по интеграции Online Designer

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

Это руководство предоставляет подробные инструкции по интеграции Online Designer в ваши веб-приложения с использованием iframe.

Базовая настройка iframe

Online Designer можно встроить в ваше веб-приложение с помощью iframe. Это позволяет открывать и редактировать шаблоны отчетов прямо внутри существующего пользовательского интерфейса.

```
<iframe
  src="https://yourdomain.ru/designer/?uuid=your-template-id&apikey=your-api-key&lang=ru"
  width="100%"
  height="800px"
  frameborder="0"
  style="border: 1px solid #ccc;">
</iframe>
```

Структура URL

URL-адрес источника iframe следует такому шаблону:

```
https://yourdomain.ru/designer/?uuid={template-id}&apikey={api-key}
```

Пример: `https://xn--80ab2acne.xn--e1afliby2b0b.xn--p1ai/designer/?uuid={template-id}&apikey={api-key}`

Где:

- `{template-id}`: идентификатор шаблона отчета
- `{api-key}`: API-ключ для авторизации в сервисе

Если шаблон открывается по ключу доступа, используйте формат:

```
https://yourdomain.ru/designer/?uuid={template-id}:{access-key}
```

Параметры аутентификации

Аутентификация по API-ключу

```
<!-- Для аутентификации по API-ключу -->
<iframe src="https://yourdomain.ru/designer/?uuid=template-123&apikey=your-api-key-here"></iframe>
```

Параметр: `apikey`

- Обязательный:** Да (для аутентификации по API-ключу)
- Описание:** Ваш API-ключ МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор
- Пример:** `apikey=dw3ized7i7i97qjsrg4mesn3n8r1bpi54wbyhywpfier7er3nqjo`
- Особенность:** после загрузки страницы скрипт `cloud-inject.js` автоматически добавляет заголовок `Authorization: Basic ...` ко всем `same-origin` запросам `fetch` и `XMLHttpRequest`, которые выполняет Online Designer

Аутентификация по ключу доступа

```
<!-- Для аутентификации по ключу доступа -->
<iframe src="https://yourdomain.ru/designer/?uuid=template-123:your-access-key"></iframe>
```

Параметр: `uuid`

- **Обязательный:** Да
- **Формат:** `uuid=ИдентификаторШаблона:КлючДоступа`
- **Описание:** Открывает шаблон по временному ключу доступа без передачи API-ключа
- **Пример:** `uuid=66f1d8a5b7c4a1250d9f1234:temp-key-abc123def456`

Параметры шаблона

Для открытия существующего шаблона используйте параметр `uuid`:

```
<iframe src="https://yourdomain.ru/designer/?uuid=template-123&apikey=your-key"></iframe>
```

Параметр: `uuid`

- **Формат:** `uuid=ИдентификаторШаблона`
- **Описание:** Определяет шаблон, который будет открыт в Online Designer
- **Примеры:**
 - По API-ключу: `uuid=66f1d8a5b7c4a1250d9f1234`
 - По ключу доступа: `uuid=66f1d8a5b7c4a1250d9f1234:temp-key-abc123def456`

Для редактирования существующего шаблона параметр `uuid` должен указывать на шаблон отчета, доступный текущему пользователю или ключу доступа.

Локализация пользовательского интерфейса

Сервис автоматически определяет язык из:

1. Конфигурации сервера (если доступно)
2. Настроек языка браузера

Локализация отчета

Локализация текста, ресурсов и значений внутри шаблона определяется самим шаблоном отчета и параметрами рабочей области.

Примеры адаптивных iframe

Адаптивный на всю ширину

```
<div style="position: relative; width: 100%; height: 0; padding-bottom: 75%;">
  <iframe
    src="https://yourdomain.ru/designer/?uuid=template-123&apikey=your-key"
    style="position: absolute; top: 0; left: 0; width: 100%; height: 100%; border: none;"
    allowfullscreen>
  </iframe>
</div>
```

Интеграция в модальное окно/диалог

```

<div class="modal" id="designerModal">
  <div class="modal-content" style="width: 95%; height: 95%;">
    <iframe
      id="designerFrame"
      src="https://yourdomain.ru/designer/?uuid=template-123&apikey=your-key"
      style="width: 100%; height: 100%; border: none;">
    </iframe>
  </div>
</div>

<script>
function openDesigner(uuid, apiKey, accessKey = null) {
  const fullUuid = accessKey ? `${uuid}:${accessKey}` : uuid;
  const baseUrl = 'https://yourdomain.ru/designer/index.html';
  const params = new URLSearchParams({ uuid: fullUuid });

  if (apiKey) {
    params.set('apikey', apiKey);
  }

  document.getElementById('designerFrame').src = baseUrl + '?' + params.toString();
  document.getElementById('designerModal').style.display = 'block';
}

// Использование
openDesigner('template-123', 'your-api-key');
</script>

```

Рекомендуемые размеры iframe

- **Минимальная ширина:** 800px
- **Минимальная высота:** 600px
- **Рекомендуемые:** 1200px × 800px или больше
- **На весь экран:** Используйте полноэкранный или почти полноэкранный режим для комфортной работы с дизайнером

Соображения безопасности

1. **Защита API-ключа:** API-ключ в query string виден на стороне клиента, поэтому в продакшене используйте только ключи с минимально необходимыми правами
2. **Ключи доступа:** Для временного доступа к конкретному шаблону по возможности используйте ключи доступа
3. **HTTPS:** Всегда используйте HTTPS для источников iframe в продакшене
4. **Заголовки CSP:** Настройте политику безопасности контента для разрешения `frame-src` и `connect-src` к домену Online Designer

```
<iframe src="..."></iframe>
```

Справочник параметров

ПАРАМЕТР	ТИП	ОБЯЗАТЕЛЬНЫЙ	ОПИСАНИЕ	ПРИМЕР
<code>uuid</code>	Строка	Да	Идентификатор шаблона или идентификатор шаблона с ключом доступа	<code>uuid=66f1d8a5b7c4a1250d9f1234</code>

ПАРАМЕТР	ТИП	ОБЯЗАТЕЛЬНЫЙ	ОПИСАНИЕ	ПРИМЕР
<code>apiKey</code>	Строка	Да*	API-ключ МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер или МоиОтчеты Публикатор	<code>apiKey=your-api-key</code>
<code>accessKey</code>	Строка	Да*	Временный ключ доступа, передаваемый в составе <code>uuid</code>	<code>uuid=66f1d8a5b7c4a1250d9f1234:temp-key-abc123def456</code>

* Требуется либо `apiKey`, либо ключ доступа в составе `uuid`.

Безопасность пользователя и CORS

Продукты: [МоиОтчеты Облако](#), [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер и МоиОтчеты Публикатор серьезно относятся к безопасности пользователей и реализует строгую политику CORS (Cross-Origin Resource Sharing) для защиты данных и предотвращения потенциальных атак. В данной статье подробно рассмотрены механизмы работы CORS и Origin в контексте МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер и МоиОтчеты Публикатор, а также объясняются принципы безопасности браузера.

Что такое CORS и почему это важно?

CORS (Cross-Origin Resource Sharing) — это механизм безопасности браузера, который определяет, может ли веб-страница запрашивать ресурсы с другого домена. Без CORS браузеры блокируют кросс-доменные запросы по умолчанию в соответствии с политикой Same-Origin Policy.

Политика Same-Origin

Браузер считает, что два URL принадлежат одному источнику (origin), если у них совпадают:

- Протокол (https/http)
- Доменное имя
- Порт

Например:

- <https://облако.моиотчеты.рф> и <https://магазин.моиотчеты.рф> — разные источники
- <https://example.com> и <https://subdomain.example.com> — разные источники

Политика CORS и МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер и МоиОтчеты Публикатор

МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер и МоиОтчеты Публикатор реализует дифференцированный подход к CORS в зависимости от состояния авторизации пользователя:

Неавторизованные запросы

Для неавторизованных пользователей CORS-политика наиболее лояльна:

- Разрешены кросс-доменные запросы к кластеру
- Не требуется указание заголовка Origin
- Это позволяет использовать публичные API и функции без ограничений

Авторизованные запросы с Origin

Для авторизованных пользователей применяются строгие правила:

- Basic Authentication (`Authorization: Basic ...`)
- Bearer Token (`Authorization: Bearer ...`)
- Обязательно указание заголовка Origin для получения разрешения от браузера

Авторизация по cookies

Для повышения безопасности, использование cookies в кросс-доменных запросах (в том числе при загрузке содержимого в iframe с другого origin) ограничено. Это предотвращает автоматическую отправку сессионных cookies в потенциально нежелательных запросах. Для встраивания отчетов через iframe рекомендуется использовать ApiKey, который позволяет обойти эти ограничения.

Механизмы безопасности браузера

МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер и МоиОтчеты Публикатор корректно обрабатывает запросы и возвращает соответствующие CORS-заголовки.

Заголовки CORS

Сервер возвращает следующие заголовки:

```
Access-Control-Allow-Origin: https://your-domain.com
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: *
Access-Control-Allow-Methods: GET, PUT, POST, DELETE, PATCH, OPTIONS
```

Настройка и использование

Конфигурация Origins

Для корректной работы системы необходимо:

1. Указать список разрешенных Origins (Domains) в рабочем пространстве
 - Максимум 10 доменов
 - Требуется указать только домен (например, `your-domain.com`). МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер и МоиОтчеты Публикатор автоматически сопоставляет входящий заголовок `Origin` с этим списком. Подстановочные знаки (например, `*.example.com`) не поддерживаются; необходимо указывать конкретные домены.
 - Протокол (`https`) указывать не нужно, протокол тоже будет определён автоматически.
2. Использование списка Origins (Domains)
 - Список разрешенных доменных имён настраивается администратором.
 - При получении запроса от авторизованного пользователя кластер проверяет заголовок `Origin` запроса по этому списку и, при совпадении, возвращает соответствующие CORS-заголовки.

Примеры конфигурации

Для веб-приложений:

```
Origin: https://your-app.com
```

Для stand-alone приложений и интеграций:

Нужно передавать заголовок только если вы реализуете стандартную безопасность браузера.

Для API-запросов с авторизацией:

Практические рекомендации

Для разработчиков

1. Всегда указывайте заголовок `Origin` для авторизованных запросов (на самом деле, браузер делает это автоматически при кросс-доменных запросах)
2. Проверяйте список разрешенных Origins в настройках рабочего пространства
3. Используйте HTTPS для всех доменов в списке Origins
4. Тестируйте CORS-запросы в браузере, а не только через инструменты командной строки

Для администраторов

1. Регулярно проверяйте список Origins (Domains)
2. Удаляйте неиспользуемые домены
3. Следите за изменениями доменных имен
4. Обновляйте Origins (Domains) при миграции приложений

Защита от распространенных атак

CSRF (Cross-Site Request Forgery)

CORS-политика предотвращает выполнение вредоносных запросов от имени авторизованного пользователя с других сайтов.

XSS (Cross-Site Scripting)

Ограничение Origins снижает риск использования уязвимостей XSS для кражи данных через кросс-доменные запросы.

Clickjacking

Использование правильных заголовков `X-Frame-Options` и CORS помогает предотвратить атаки кликджекинга.

Заключение

Политика CORS в МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер и МоиОтчеты Публикатор представляет собой сбалансированное решение, обеспечивающее:

- Максимальную безопасность авторизованных пользователей
- Удобство использования для неавторизованных запросов
- Гибкость настройки для различных сценариев использования
- Совместимость с современными браузерами и их защитными механизмами

Правильная настройка Origins и соблюдение правил CORS позволяет безопасно интегрировать МоиОтчеты Облако, МоиОтчеты Корпоративный Сервер и МоиОтчеты Публикатор в любые веб-приложения, обеспечивая защиту данных пользователей и соответствие современным стандартам безопасности веб-приложений.