Документация

Выпуск 3.2

XSQUARE - REPORTS

мар. 14, 2024

Содержание

ГЛАВА 1

Общие сведения

1.1 Содержание документации

Список разделов документации сервера отчетов (report server) можно увидеть в разделе Содержание.

1.2 Назначение сервера отчетов

Основными функциями сервера отчетов являются:

- генерация отчетов в форматах DOCX, XLSX, PDF;
- объединение PDF-документов;
- формирование на основе ранее сформированного PDF-документа печатной формы в формате PDF со штампом.

1.3 Модули сервера отчетов

Сервер отчетов включает следующие модули:

- *Сервис формирования отчетов из документа-шаблона*
- *Генератор отчетов PDF*
- *Сервис объединения PDF-документов*
- *Сервис формирования печатной формы*

1.4 Операционные системы

Протестировано на следующих операционных системах x86-64:

- Astra Linux
- RED OS
- Alt Linux
- ROSA
- Ubuntu 20+
- Red Hat 8+
- Debian 10+

1.5 Установка сервера отчетов

Установка сервера отчетов описана в разделе *Установка*

1.6 Примеры использования

Примеры запросов и шаблонов приведены в разделе *Примеры*.

1.7 Получение первого отчета

Процесс создания простого отчета с помощью сервера отчетов описан в разделе *Мой первый отчет*.

ГЛАВА 2

Архитектура и системные требования

2.1 Архитектура

Архитектура приложения представляет собой веб-сервер, который обрабатывает клиентские HTTP-запросы посредством формирования отчетов на основе документов-шаблонов и возвращает результат клиенту.

2.2 Среда исполнения

Поддерживаемые архитектуры:

- x86-64
- ARM (в том числе байкал)
- e2k Эльбрус

Сертифицировано со следующими отечественными ОС (https://xsquare.ru/o-nas/):

- Astra Linux
- RED OS
- Alt Linux
- ROSA

Поддерживаемые ОС:

- Ubuntu 20+
- Red Hat 8+
- Debian 10+

2.3 Системные требования

CPU - 1 Ядро RAM - 100 Мб HDD - 100 Мб + Логи

Установка системы виртуализации/контейнеризации, операционной системы, базы данных осуществляется на усмотрение Администратора исходя из потребностей Организации.

ГЛАВА 3

Установка

Протестировано на следующих операционных системах x86-64:

- Astra Linux
- RED OS
- Alt Linux
- ROSA
- Ubuntu 20+
- Red Hat 8+
- Debian 10+

3.1 Установка

```
mkdir -p /usr/local/xsquare.xreports
```

3.1.1 Перенос контента

```
scp -r  xreports assets/ templates/  root@[host]/usr/local/xsquare.xreports
```

3.1.2 Установка Libre Office

Рекомендуем устанавливать дистрибутив Libre Office версии 7.4.7.2 с сайта производителя(не из репозитория OS):

```
tar -xvzf LibreOffice_7.4.7.2_Linux_x86-64_rpm.tar.gz
cd LibreOffice_7.4.7.2_Linux_x86-64_rpm/
cd RPMS/
yum install *.rpm
find  / -name "soffice"
vi /usr/local/xsquare.xreports/config.json
```

3.1.3 Настройка Systemctl

```
vi /etc/systemd/system/xsquare.xreports.service

[Unit]
Description=XSQUARE-Reports
After=syslog.target
After=network.target

[Service]
Type=simple
ExecStart=/usr/local/xsquare.xreports/xreports
WorkingDirectory=/usr/local/xsquare.xreports
User=root

[Install]
WantedBy=multi-user.target
```

Запуск служб

```
systemctl daemon-reload
systemctl enable xsquare.xreports
systemctl start xsquare.xreports
systemctl status xsquare.xreports
```

3.1.4 Установка Шрифтов для RED OS

```
dnf install msttcore-fonts-installer
fc-cache -fv
```

3.1.5 Файл конфигурации config.json

Для работы сервера отчетов необходимо, чтобы в директории с сервисом присутствовал файл конфигурации config.json. Файл конфигурации содержит 3 раздела:

1. Описатель App, где можно определить базовые настройки сервиса:

```
    {
"app": {
    "port": "8087",
    "debug": false
},
```

- «port» - строка. Определяет номер сетевого порта, на котором будет запущен сервис (по умолчанию - 8087)
- «debug» - логическое значение. Включает режим отладки, при котором доступен подробный лог обработки запросов, а также происходит сохранение всех запросов и готовых документов в локальной директории **reports_debug сервиса** (аналогично действию флага *enable-debug-report-save* в свойствах запроса). Значение по умолчанию - **false**.

2. Описатель formatConversion, где определяются настройки для приложения Libre Office soffice:

```
"formatConversion": {
        "format-conversion-dir": "",
        "soffice-max-process-count": 0,
        "soffice-path": ""
},
```

- «format-conversion-dir» - строка. Определяет директорию для хранения временных файлов (по умолчанию - «/tmp»)
- «soffice-max-process-count» - число. Определяет количество ядер процессора, которое может использовать soffice для конвертации (по умолчанию - «0», использовать все доступные ядра)
- «soffice-path» - строка. Определяет путь до исполняемого файла soffice (по умолчанию - путь, записанный в переменные окружения при установке Libre Office)

3. Описатель license, где определяются настройки в соответствии с лицензионным соглашением:

```
"license": {
        "TAX_ID": "tax id",
        "TAX_NAME": "name",
        "WORKER_COUNT": 5,
        "CORE_COUNT": 3,
        "LICENSE_ID": "61c06e5d",
        "LICENSE_EXPIRATION_DATE": "10.10.2073",
        "SUPPORT_EXPIRATION_DATE": "11.11.2023",
        "SIGNATURE": "yQLpwB54SSgDgVh1LX9sUxR7o7EyZC//JmEtlEYOvtM7XY="
        }
}
```

ГЛАВА 4

Функциональные характеристики

XSQUARE-REPORTS – это самостоятельный веб-сервис, предназначенный для формирования отчетов в различных форматах из документов-шаблонов по HTTP-запросу в форматах XML или JSON. на основе запросов с данными и шаблонов документов в различных форматах.

Основные функциональные характеристики и возможности сервера отчетов XSQUARE-REPORTS:

- формирование отчета в формате OOXML (документы *.docx и *.xlsx) из документа-шаблона по HTTP-запросу в формате XML или JSON
- формирование отчета в формате PDF по HTTP-запросу в формате JSON
- объединение документов в формате PDF в один документ по HTTP-запросу в формате JSON
- формирование печатной формы документа в формате PDF по HTTP-запросу в формате JSON

ГЛАВА 5

Жизненный цикл

5.1 Общие сведения

Описание процессов, обеспечивающих поддержание жизненного цикла программного обеспечения XSQUARE-REPORTS, в том числе устранение неисправностей, выявленных в ходе эксплуатации программного обеспечения, совершенствование программного обеспечения, а также информация о персонале, необходимом для такой поддержки.

Программа	Сервер отчетов XSQUARE-REPORTS
Разработчик	ООО «Хи-Квадрат»
Пользователь	Юридическое лицо, использующее Программу согласно договора с Разработчиком
Сайт	https://lcdp.xsquare.ru

5.2 Поддержание жизненного цикла Программы

1. Жизненный цикл Программы включает в себя следующие этапы:
2. Проектирование и разработка Программы, осуществляемые Разработчиком;
3. Тестирование и выявление неисправностей в работе Программы Разработчиком;
4. Установка, использование и обновление Программы Пользователем согласно лицензионному соглашению с Разработчиком;
5. Модернизация программы Разработчиком согласно собственному плану доработок и улучшений, а также по заявкам Пользователя;
6. Осуществление технической поддержки Пользователя Разработчиком по вопросам установки, интеграции и эксплуатации Программы;
7. Выпуск Разработчиком обновленных сборок модернизированной Программы.

8. Разработчик регулирует проведение всех этапов жизненного цикла программы за исключением процессов установки, интеграции и использования Программы Пользователем.

5.3 Устранение неисправностей, выявленных в ходе эксплуатации Программы

Неисправности, выявленные в ходе эксплуатации Программы могут быть устранены следующими способами:

1. Внесение исправлений в код Программы Разработчиком согласно своей дорожной карте разработки Программы;
2. Внесение исправлений в код Программы на основе обращения Пользователя;

Пользователь может сформировать следующие запросы:

- Отчёт об инциденте с приложением информации об условиях возникновения бага с использованием графической информации, лог-файлов, информации о программном окружении и номерах версий используемого программного обеспечения, включая версию и редакцию Программы. Запрос также должен содержать информацию об ожидаемом и фактическом поведении Программы и любую другую информацию, которая поможет диагностировать и устранить неисправность Программы Разработчиком;
- Запрос на улучшение Программы в целях изменения её поведения для достижения нужных результатов в решениях Пользователя;
- Запрос на предоставление дополнительной информации о функционировании и возможностях Программы.

Запросы могут быть отправлены Пользователем только с помощью трекинговой системы Разработчика - https://tracker.xsquare.ru. Доступ к трекенговой системе предоставляется по факту приобретения Программы.

Разработчик принимает и фиксирует все запросы Пользователя. Каждому запросу присваиваится уникальный номер, который позволяет отследить историю общения Пользователя и Разработчика в дальнейшем.

Разработчик информирует Пользователя о новом функционале Программы, либо о добавлении задачи по развитию в план разработки.

Разработчик оставляет за собой право запросить дополнительную информацию от Пользователя, которая может быть полезна для устранения неисправностей в работе Программы.

При непредоставлении либо недостаточном предоставлении Пользователем информации, требуемой Разработчиком, последний вправе приостановить внесение требуемых изменений в код Программы.

5.4 Совершенствование Программы

Программа непрерывно улучшается и модернизируется, выпускаются регулярные обновления, публикуются информационные материалы на Сайте Программы, Пользователи информируются об изменениях в Программе.

Пользователь может инициировать запрос на изменение или улучшение работы Программы с помощью формирования запроса к Разработчику.

Запросы могут быть отправлены Пользователем с помощью трекинговой системы https://tracker.xsquare.ru

Разработчик принимает и фиксирует все запросы Пользователя. Каждому запросу присваивается уникальный номер, который позволяет отследить историю общения Пользователя и Разработчика в дальнейшем.

Разработчик информирует Пользователя о внесенных изменениях в код Программы, либо о добавлении задачи по модернизации в план разработки.

Разработчик оставляет за собой право запросить дополнительную информацию от Пользователя, которая может быть полезна для улучшения работы Программы.

При непредоставлении либо недостаточном предоставлении Пользователем информации, требуемой Разработчиком, последний вправе приостановить внесение требуемых изменений в код Программы

5.5 Техническая поддержка Программы

Техническая поддержка Программы осуществляется с помощью формирования запросов Разработчику.

Запросы могут быть отправлены Пользователем с помощью трекинговой системы https://tracker.xsquare.ru.

Разработчик принимает и фиксирует все запросы Пользователя. Каждому запросу присваивается уникальный номер, который позволяет отследить историю общения Пользователя и Разработчика в дальнейшем.

Техническая поддержка Пользователя включает в себя:

- Помощь в установке Программы;
- Помощь в установке базовых общесистемных компонентов (операционной системы, HTTP Сервер, сервер баз данных и)
- Помощь в интеграции Программы в существующие решения Пользователя;
- Помощь в устранении неисправностей, возникающих в работе Программы;
- Консультации по функционированию Программы;
- Сбор информации о некорректной работе Программы для последующего выпуска модернизации Программы согласно плану доработок;
- Информирование Пользователя об обновлениях Программы

5.6 Информация о персонале, необходимом для обеспечения поддержки

Персонал, который будет осуществлять поддержку Программы со стороны Пользователя, должен обладать:

1. Базовыми навыками администрирование операционных систем семейства Unix
2. Базовыми навыками работы с офисными пакетами
3. Пользователи Программы должны обладать навыками работы с персональным компьютером и веб браузером.
4. В случае возникновения вопросов у персонала, им следует обратиться к Разработчику за получением технической поддержки

ГЛАВА 6

Эксплуатация

Данный раздел описывает порядок поддержания работоспособности приложения и порядок загрузки компонентов.

6.1 Сервер отчетов

Для загрузки xsquare.xreports пользователю необходимо убедиться в наличие правильно настроенного конфигурационного файла.

```
cat /usr/local/xsquare.xreports/config.json
```

Команда должна отобразить правильный конфигурационный файл, описанный в разделе “Установка”.

Далее необходимо запустить сервер отчетов выполнив команду:

```
systemctl start xsquare.xreports
```

После проверить состояние сервера приложений:

```
systemctl status xsquare.xreports
```

В случае возникновения ошибок они будут записаны в журнал. Проверить сообщения об ошибках можно выполнив команду:

```
journalctl -u xsquare.xreports
```

ГЛАВА 7

Модули

7.1 Сервис формирования отчетов из документа-шаблона

7.1.1 Общее описание сервиса

Сервис формирует отчет в формате OOXML (документы *.docx и *.xlsx) из документа-шаблона по HTTP-запросу в формате XML или JSON. Опционально возможна конвертация отчета в PDF.

Алгоритм работы сервиса: теги документа-шаблона заменяются конкретными данными, указанными в запросе.

Основные возможности сервиса

1. Генерация отчетов в формате DOCX на основе документа-шаблона. Могут быть сгенерированы:
   - простые отчеты в формате DOCX;
   - мульти-отчеты в формате DOCX (отчеты из одного шаблона на основе множества входных данных).
2. Генерация отчетов в формате XLSX на основе документа-шаблона.
3. Конвертация отчетов в формат PDF.

7.1.2 Работа с документами-шаблонами

Для формирования отчета необходимо создать шаблон в формате DOCX или XLSX.

Расположение файлов шаблонов

Документы шаблонов в форматах DOCX и XLSX хранятся в локальной в директории *templates*,расположенной в директории приложения сервиса.

Формат файлов шаблонов

Для создания документов-шаблонов необходимо использовать текстовый процессор (MS Word/Excel, Р7-Офис, LibreOffice Writer/Calc, Яндекс Документы и т.п.).

Важно:

- Шаблоны для формирования документов в формате DOCX должны быть сохранены в формате DOCX.
- Шаблоны для формирования документов в формате XLSX должны быть сохранены в формате XLSX.
- Шаблоны для последующей конвертации в PDF должны быть сохранены в формате DOCX (желательно с использованием Libre Office, так как результирующий PDF будет полностью соответствовать визуальному представлению документа в Libre Office).

Использование тегов в шаблоне

Шаблоны документов могут содержать теги, которые будут заменены входными данными из запроса. Тег задается в квадратных скобках. Пример: [задолженность]

В теле запроса входные данные для тегов содержатся в `input-data`. В запросе скобки не указываются (см. примеры из пункта “Структура запроса” подраздела “Формирование отчета в формате DOCX” и из пункта “Структура запроса” подраздела “Формирование отчета в формате XLSX”).

Работа с изображениями

Принцип работы с изображениями:

- в шаблон добавляется изображение, предназначаемое для последующей замены;
- при формировании документа оно заменяется файлом изображения, который передается в запросе. В запросе передается новое содержимое файла изображения, закодированное в BASE64.

Теги изображений в шаблоне задаются в свойстве изображения «Замещающий текст» («Alt text») без квадратных скобок.

Поддерживаемые форматы: JPEG, PNG.

Ограничения для изображений

1. Формат изображения (JPEG, PNG) в запросе должен совпадать с форматом замещаемого изображения в шаблоне.
2. Если необходима подмена нескольких изображений шаблона разными изображениями из входных данных, в шаблоне изображения также должны быть разными. Связано с тем, что изображение в теле документа ссылается на его контент (файл), откуда следует, что при замене контента в документе изменятся все изображения, которые на этот контент ссылаются.

Примеры шаблонов

С примерами шаблонов можно ознакомиться в архиве примеров *архиве примеров*

Примеры шаблонов с изображениями:

Для документов в формате DOCX:

- barcode_code-128.docx
- example_from_xml.docx
- example_from_json.docx

Для документов в формате XLSX:

- barcode_code-128.xlsx
- base_example.xlsx

7.1.3 Формирование отчета в формате DOCX

Описание сервиса

Наименование сервиса	Сервис формирования отчетов из документа-шаблона в формате **DOCX**
Путь к сервису	Простые отчеты в формате DOCX: • [host]:[port]/word_report_json • [host]:[port]/word_report_xml Мульти-отчеты в формате DOCX: • [host]:[port]/word_multi_report_1_N_json • [host]:[port]/word_multi_report_1_N_xml
Метод	POST
Параметры	Тело запроса должно содержать объект в формате JSON или XML. Подробнее про структуру тела запроса можно прочитать в подразделе “Структура тела запроса”. В ответ сервис отдаёт файл документа в формате DOCX, закодированный в base64. При конвертации отчета в формат PDF сервис возвращает файл документа в формате PDF, закодированный в base64.
Назначение	Сервис предназначен для формирования отчетов из документа-шаблона в формате **DOCX**. Возможна конвертация в PDF.

Структура тела запроса

Тело запроса содержит объект в формате JSON или XML, который включает:

1. Описатель шаблона.
2. Входные данные для подстановки в шаблон вместо тегов.
3. Опции запроса.
4. Формат ответа.

Описатель шаблона

Элемент (объект) описателя шаблона `template` содержит:

- `uri` - строка. Задает расположение файла документа-шаблона в зависимости от того, как трактуется параметр `id`. Поддерживаемое значение: *local*.
- `id` - строка. Идентификатор шаблона. Путь к файлу документа-шаблона относительно директории сервиса `templates`. Также используется для записи файла отчета в отладочном режиме и для идентификации запроса в логе.

Пример для формата XML:

```
<template uri="local" id="Информационное письмо"/>
```

Пример для формата JSON:

```
      "template":
{
    "uri": "local",
    "id": "template_example_1"
}
```

Входные данные

Элемент (объект) входных данных `input-data` содержит:

- Простые строковые данные для подстановки в шаблон вместо тегов.
- Значения для условных выражений.
- Описатели таблиц.
- Описатели списков.
- Данные для подстановки в теги изображений.
- Описатели блоков.

Примечание: Отличие структуры данных для формата JSON от XML в том, что вместо XML-элементов данные представлены как объекты и списки объектов. Имена тегов задаются именами полей объектов. Списки, таблицы, изображения являются дочерними объектами по отношению к объекту `input-data`.

Примечание: Для мульти-отчетов используется список (массив) `input-data-array`, содержащий элементы (объекты), по структуре совпадающие с `input-data`. Подробнее с мульти-отчетами можно ознакомиться в подразделе “Работа с документом мульти-отчета”.

Простые строковые данные для подстановки в шаблон вместо тегов

Для формата XML

Элемент `tags` содержит список дочерних элементов `tag` с атрибутом `name` и нужным конкретным значением, на который тег в шаблоне будет заменен при генерации отчета. Атрибут `name` совпадает с тегом в шаблоне.

Пример:

```
<input-data>
<tags>
        <tag name="ORGANIZATION">АО «xxxxyyyy»</tag>
</tags>
</input-data>
```

Для формата JSON

Имена тегов задаются именами объектов внутри `input-data`.

Пример:

```
 "input-data":
         {
"ORGANIZATION": "АО «xxxxyyyy»"
 }
```

Описание таблиц

Для формата XML

Элемент описателей таблиц tables содержит список дочерних элементов table с атрибутом name. Атрибут name совпадает с тегом в шаблоне. Тег в шаблоне будет заменен на таблицу, созданную согласно описанию таблицы в запросе.

Пример:

```
<input-data>
    <tables>
        <table name="TABLE-FORMATTED">
                <rows>
                    <row>
                        <cell tag="номер_пп">1</cell>
                        <cell tag="номер_договора">Пример номера 1</cell>
                        <cell tag="район">Пример района 1</cell>
                        <cell tag="предприятие">Пример предприятия 1</cell>
                        <cell tag="дата_отключения">Пример даты 1</cell>
                        <cell tag="адрес">Пример адреса 1</cell>
                        <cell tag="комплексное_поле">Пример суммы 1</cell>
                    </row>
                </rows>
            </table>
            <table name="TABLE-NO-FORMAT">
                <header>
                    <cell>№ п/п</cell>
                    <cell>№ дела в Арбитражном суде</cell>,
                    <cell>Период задолженности</cell>,
                    <cell>Основной долг, руб. с НДС</cell>,
```

(continues on next page)

(продолжение с предыдущей страницы)

```
                    <cell>Проценты, руб.</cell>,
                    <cell>Госпошлина, руб.</cell>,
                    <cell>Неустойка по решению суда</cell>,
                    <cell>Общая сумма погашения, руб.</cell>,
                    <cell>Срок погашения, не позднее</cell>
                </header>
                <rows>
                    <row>
                        <cell>1</cell>,
                        <cell>A12-1234/2030</cell>,
                        <cell>Октябрь 2017</cell>,
                        <cell>12 345,58</cell>,
                        <cell/>
                        <cell>23 456,00</cell>,
                        <cell>78 912,41</cell>,
                        <cell/>
                        <cell>31.08.2019</cell>
                    </row>
                </rows>
            </table>
        </tables>
</input-data>
```

Для формата JSON

Описатели таблиц являются дочерними объектами по отношению к объекту `input-data`. Именами полей объектов таблиц совпадают с тегами в шаблоне.

Пример:

```
"input-data":
 {
     "TABLE-NO-FORMAT":
     {
         "header":
         [
             "№ п/п",
             "№ дела в Арбитражном суде",
             "Период задолженности",
             "Основной долг, руб. с НДС",
             "Проценты, руб.",
             "Госпошлина, руб.",
             "Неустойка по решению суда",
             "Общая сумма погашения, руб.",
             "Срок погашения, не позднее"
         ],
         "rows":
         [
             [
                 "1",
                 "A12-1234/2030",
                 "Октябрь 2017",
                 "12 345,58",
                 null,
                 "23 456,00",
                 "78 912,41",
                 null,
```

(continues on next page)

(продолжение с предыдущей страницы)

```
                "31.08.2019"
            ]
        ]
    },
    "TABLE-FORMATTED":
    {
        "rows":
        [
            {
                "номер_пп": "1",
                "номер_договора": "Пример номера 1",
                "район": "Пример района 1",
                "предприятие": "Пример предприятия 1",
                "дата_отключения": "Пример даты 1",
                "адрес": "Пример адреса 1",
                "комплексное_поле": "Пример суммы 1"
            }
        ]
    }
}
```

Описатели списков

Для формата XML

Элемент описателей списков `lists` содержит список дочерних элементов `list` с атрибутом name. Атрибут name совпадает с тегом в шаблоне. Тег в шаблоне будет заменен параграфом, отформатированным как список с данными элементов из описателя списка в запросе.

Пример:

```
<input-data>
    <lists>
       <list name="BULLET_LIST">
           <items>
               <item>bullet item 1</item>
               <item>bullet item 2</item>
               <item>bullet item 3</item>
           </items>
       </list>
        <list name="NUMBERED_LIST">
           <items>
               <item>numbered item 1</item>
               <item>numbered item 2</item>
               <item>numbered item 3</item>
           </items>
       </list>
   </lists>
</input-data>
```

Для формата JSON

Описатели списков являются массивами внутри объекта `input-data`. Именами полей объектов списков совпадают с тегами в шаблоне.

Пример:

```
"input-data":
   {
        "BULLET_LIST": ["bullet item 1", "bullet item 2", "bullet item 3"],
        "NUMBERED_LIST": ["numbered item 1", "numbered item 2", "numbered item 3"]
   }
```

Данные для подстановки в теги изображений

Для формата XML

Элемент `images` содержит список дочерних элементов `image` с атрибутом `name`. Атрибут `name` совпадает с тегом в шаблоне. Изображения в шаблоне будут заменены на данные изображений из запроса.

Пример:

```
<input-data>
    <images>
       <image name="IMAGE-1-JPEG">/9j/4AAQSkZJRgABAQEAYABgAAD/4Q…CigAooAKKACigD//Z</
↪image>
       <image name="IMAGE-2-JPEG">/9j/4AAQSkZJRgABAQEAYABg…KKACigAooAKKACigAooA//9k=</
↪image>
   </images>
</input-data>
```

Для формата JSON

Данные для подстановки в теги изображений находятся внутри объекта `images`, который является дочерним для объекта `input-data`.

Пример:

```
"input-data":
  {
      "images":
      {
          "IMAGE-1-JPEG":
          {
              "data": "/9j/4AAQSkZJRgABAQEAYABgAAD…igAooAKKACigAooAKKACigD//Z"
          },
          "IMAGE-2-JPEG":
          {
              "data": "/9j/4AAQSkZJRgABAQEAYAB…AooAKKACigAooAKKACigAooA//9k="
          }
      }
  }
```

Описатели блоков

Для формата XML

Элемент описателей блоков `blocks` содержит список дочерних элементов `block` с атрибутом `block-template` (совпадает с именем шаблона блока в документе-шаблоне). Блок имеет те же элементы, что и `input-data`, кроме элементов изображений и блоков. Шаблоны блоков в документе-шаблоне будут использованы для вставки в документ произвольного количества экземпляров блоков с разными наборами данных. Места вставки определяются тегами экземпляра шаблона в документе-шаблоне. Детали в подразделе “Работа с документом → Блоки”.

Для формата JSON

Список (массив) описателей блоков `blocks` находится внутри объекта `input-data`.

Пример:

```
"input-data": {
    "blocks": [
        {
            "block-template": "block1",
            "data": {
                "таблица1": {
                    "rows": [{
                                "дней": "670",
                                "доля": "130",
                                "пени": "12 564,46",
                                "дата_с": "11.02.2021",
                                "период": "Реализация (акт, накладная, УПД) уХан0000165␣
↪от 31.01.2021 23:59:59",
                                "ставка": "7,5",
                                "дата_по": "12.12.2022",
                                "формула": "32 505,07 * 670 * 0.08 * 1/130",
                                "задолженность": "32 505,07"
                            }
                    ]
                },
                "сумма_пени": "12 345,67",
                "сумма_долга": "78 910,23",
                "номер_договора": "1"
            }
        },
        {
            "block-template": "block2",
            "data": {
                "таблица1": {
                    "rows": [{
                                "дней": "670",
                                "пени": "12 564,46",
                                "дата_с": "11.02.2021",
                                "период": "Реализация (акт, накладная, УПД) уХан0000165␣
↪от 31.01.2021 23:59:59",
                                "ставка": "7,5",
                                "дата_по": "12.12.2022",
                                "задолженность": "32 505,07"
                            }
                    ]
                },
```

(continues on next page)

(продолжение с предыдущей страницы)

```
                "сумма_пени": "76 543,21",
                "сумма_долга": "987 654,32",
                "номер_договора": "2"
            }
        }
    ]
}
```

Опции запроса

Элемент (объект) опций запроса `options` содержит:

- `enable-debug-report-save` - логическое значение. Указывает создавать ли копию отчета документа и копию запроса (имя файла - идентификатор шаблона) в локальной директории `reports_debug` сервиса. Значение по умолчанию - `false`.
- `enable-binary-output` - логическое значение. Указывает, что сервис будет выдавать результат в виде бинарных данных, без кодирования в base64. В случае возникновения ошибки ответ будет соответствовать *используемому по умолчанию*. Значение по умолчанию - `false`.
- `formatting` - объект опций форматирования. Поддерживается только для запросов word. Поля объекта: `tables` - объект опций форматирования таблиц. Поля объекта: `enable-cells-auto-merge` - логическое значение. Указывает, объединять ли подряд идущие ячейки с одинаковым значением в одном столбце по вертикали. Значение по умолчанию - `true`.
- `report-format` - строка. Указывает формат файла отчета, если он отличен от формата, упомянутого в url запроса. Единственное непустое поддерживаемое значение: “`pdf`”. Значение по умолчанию: пустая строка. См. пункт “Конвертация отчета в PDF” ниже.

Пример для формата XML:

```
<options>
    <enable-debug-report-save>true</enable-debug-report-save>
    <enable-binary-output>false</enable-binary-output>
    <formatting>
        <tables>
            <enable-cells-auto-merge>true</enable-cells-auto-merge>
        </tables>
    </formatting>
    <report-format>pdf</report-format>
</options>
```

Пример для формата JSON:

```
"options":
  {
      "enable-debug-report-save": true,
      "enable-binary-output": false,
      "formatting": {
          "tables": {
              "enable-cells-auto-merge": true
          }
      },
      "report-format": "pdf"
  }
```

Формат ответа

Элемент (объект) формата ответа `response-format` может принимать значения `json` и `xml`.

Подробнее про то, как задается формат ответа можно прочитать в подразделе “Структура ответа”.

Пример для формата XML:

```
<response-format value="xml"/>
```

Пример для формата JSON:

```
{"response-format": "json"}
```

Примеры запросов

Примеры для формата XML

Пример №1

```
<?xml version="1.0" encoding="UTF-8"?>
<request>
    <template uri="…" id="…"/>
    <input-data>
        <tags>…</tags>
        <tables>…</tables>
        <lists>…</lists>
        <images>…</images>
        <blocks>…</blocks>
    </input-data>
    <options>…</options>
    <response-format value="…"/>
</request>>
```

Пример №2

```
<?xml version="1.0" encoding="UTF-8"?>
<request>
   <template uri="local" id="template_example_1"/>
   <input-data>
       <tags>
           <tag name="ORGANIZATION">АО «xxxxyyyy»</tag>
           <tag name="TAG_IN_HEADER_TEST">Пример верхнего колонтитула</tag>
           <tag name="TAG_IN_FOOTER_TEST">Пример нижнего колонтитула</tag>
           <tag name="CONDITIONAL_TAG_TRUE">true</tag>
           <tag name="CONDITIONAL_TAG_FALSE">false</tag>
       </tags>
       <tables>
           <table name="TABLE-FORMATTED">
               <rows>
                   <row>
                       <cell tag="номер_пп">1</cell>
                       <cell tag="номер_договора">Пример номера 1</cell>
                       <cell tag="район">Пример района 1</cell>
                       <cell tag="предприятие">Пример предприятия 1</cell>
                       <cell tag="дата_отключения">Пример даты 1</cell>
                       <cell tag="адрес">Пример адреса 1</cell>
```

(continues on next page)

(продолжение с предыдущей страницы)

```
                    <cell tag="комплексное_поле">Пример суммы 1</cell>
                </row>
            </rows>
        </table>
        <table name="TABLE-NO-FORMAT">
            <header>
                <cell>№ п/п</cell>
                <cell>№ дела в Арбитражном суде</cell>,
                <cell>Период задолженности</cell>,
                <cell>Основной долг, руб. с НДС</cell>,
                <cell>Проценты, руб.</cell>,
                <cell>Госпошлина, руб.</cell>,
                <cell>Неустойка по решению суда</cell>,
                <cell>Общая сумма погашения, руб.</cell>,
                <cell>Срок погашения, не позднее</cell>
            </header>
            <rows>
                <row>
                    <cell>1</cell>,
                    <cell>A12-1234/2030</cell>,
                    <cell>Октябрь 2017</cell>,
                    <cell>12 345,58</cell>,
                    <cell/>
                    <cell>23 456,00</cell>,
                    <cell>78 912,41</cell>,
                    <cell/>
                    <cell>31.08.2019</cell>
                </row>
            </rows>
        </table>
    </tables>
    <images>
        <image name="IMAGE-1-JPEG">/9j/
↪4AAQSkZJRgABAQEAYABgAAD…igAooAKKACigAooAKKACigD//Z</image>
        <image name="IMAGE-2-JPEG">/9j/
↪4AAQSkZJRgABAQEAYAB…AooAKKACigAooAKKACigAooA//9k=</image>
    </images>
</input-data>
<options>
    <enable-debug-report-save>true</enable-debug-report-save>
    <formatting>
        <tables>
            <enable-cells-auto-merge>true</enable-cells-auto-merge>
        </tables>
    </formatting>
    <!-- <report-format>pdf</report-format> -->
</options>
</request>
```

Пример для формата JSON

```
{
   "template":
   {
       "uri": "local",
       "id": "template_example_1"
   },
```

(continues on next page)

(продолжение с предыдущей страницы)


```
"input-data":
{
    "ORGANIZATION": "АО «xxxxyyyy»",
    "TAG_IN_HEADER_TEST": "Пример верхнего колонтитула",
    "TAG_IN_FOOTER_TEST": "Пример нижнего колонтитула",
    "CONDITIONAL_TAG_TRUE": "true",
    "CONDITIONAL_TAG_FALSE": "false",
    "TABLE-NO-FORMAT":
    {
        "header":
        [
            "№ п/п",
            "№ дела в Арбитражном суде",
            "Период задолженности",
            "Основной долг, руб. с НДС",
            "Проценты, руб.",
            "Госпошлина, руб.",
            "Неустойка по решению суда",
            "Общая сумма погашения, руб.",
            "Срок погашения, не позднее"
        ],
        "rows":
        [
            [
                "1",
                "А12-1234/2030",
                "Октябрь 2017",
                "12 345,58",
                null,
                "23 456,00",
                "78 912,41",
                null,
                "31.08.2019"
            ]
        ]
    },
    "TABLE-FORMATTED":
    {
        "rows":
        [
            {
                "номер_пп": "1",
                "номер_договора": "Пример номера 1",
                "район": "Пример района 1",
                "предприятие": "Пример предприятия 1",
                "дата_отключения": "Пример даты 1",
                "адрес": "Пример адреса 1",
                "комплексное_поле": "Пример суммы 1"
            }
        ]
    },
    "images":
    {
        "IMAGE-1-JPEG":
        {
            "data": "/9j/4AAQSkZJRgABAQEAYABgAAD…igAooAKKACigAooAKKACigD//Z"
        },
```


(continues on next page)

(продолжение с предыдущей страницы)

```
            "IMAGE-2-JPEG":
            {
                "data": "/9j/4AAQSkZJRgABAQEAYAB…AooAKKACigAooAKKACigAooA//9k="
            }
        }
    },
    "options":
    {
        "enable-debug-report-save": true,
        "formatting": {
            "tables": {
                "enable-cells-auto-merge": true
            }
        }
    }
}
```

Примеры запросов в форматах XML и JSON также можно посмотреть в *архиве примеров*.

Пример вызова сервиса

Для формата XML

```
curl --request POST --data-binary "@templates/examples/docx/example_from_xml.xml"␣
↪http://localhost:8087/word_report_xml
```

Для формата JSON

```
curl --request POST --data-binary "@templates/examples/docx/example_from_json.json"␣
↪http://localhost:8087/word_report_json
```

Структура ответа

Формат ответа (JSON или XML) может быть задан двумя способами:

1. в HTTP заголовке Accept. Поддерживаемые значения: `application/json` и `application/xml` .
2. в теле запроса в элементе (объекте) `response-format`. Поддерживаемые значения: `json` и `xml`.

Приоритет имеет формат, указанный в теле запроса.

Ответ сервиса содержит объект в формате JSON или XML, который включает:

1. Описание ошибки (код ошибки, сообщение об ошибке). В случае успешного ответа сервиса возвращается значение `null`.
2. Результат (закодированный в base64 файл документа отчета в формате docx/pdf).

Формат ответа

```
{
    error: {
        code
        message
    }
    result: "[base64 encoded docx/pdf file]"
}
```

Пример ответа


```
{
   "error": null,
   "result": "UEsDBBQACAAIAPdKo...JwAAAAA="
}
```


Работа с документом

Общая информация

Для документов в формате DOCX доступны:

- теги, которые будут заменены входными данными из запроса;
- таблицы;
- списки;
- изображения;
- условные выражения;
- создание единого отчета из одного шаблона на основе множества входных данных (мульти-отчеты);
- блоки.

Работа с таблицами

Поддерживаются следующие типы таблиц в шаблоне:

1. Таблица без форматирования. Задается только тегом. Таблица генерируется на ширину страницы с равной шириной столбцов.
2. Таблица с форматированием. Задается таблицей со следующими параметрами:

- опциональный произвольный заголовок.
- первая ячейка строки, следующей за заголовком содержит тег таблицы. Эта строка удаляется при генерации.
- строка следующая за строкой тега содержит теги столбцов, их форматирование будет применено к сгенерированным строкам. Эта строка удаляется при генерации.
- опциональные фиксированные строки с любыми тегами (источник данных замены - основные теги).

Примечание: таблицы в колонтитулах документа поддерживаются.

Управление слиянием ячеек

Базовый алгоритм слияния - автоматическое слияние ячеек с одинаковым содержимым по вертикали.

Изменить алгоритм слияния можно при помощи опции запроса `formatting.tables.enable-cells-auto-merge`.

Для запросов в формате JSON поддерживается расширенные опции вертикального и горизонтального слияния.

Объект, описывающий строку таблицы, имеет опциональный объект `formatting`. Пример:

```
"formatting": {
   "column tag value": {
       "vertical_merge": "restart",
       "column_span": 2
   }
}
```

Имена полей этого объекта являются названиями столбцов таблицы, для которых нужно применить опцию слияния. Значения полей: объект опции слияния. Данный объект имеет поля:

- `vertical_merge` - строка. Управляет вертикальным слиянием ячейки. Поддерживаемые значения: “restart” - предотвращает слияние соседней ячейкой, расположенной в предыдущей строке того же столбца; “continue” - форсирование слияния с ячейкой выше по столбцу; “unset” - опция слияния будет унаследована от ячейки выше по столбцу;
- `column_span` - целое число. Управляет горизонтальным слиянием ячейки. Значение задает количество ячеек для объединения справа от текущей ячейки (включительно).

Примечание: это прямое управление настройками слияния в документе. Чтобы быстро понять, какие значения нужно выставить тем или иным ячейкам, нужно сперва настроить желаемое поведение в текстовом редакторе, сохранить docx-документ, разархивировать его как zip архив и посмотреть, какие значения применяются в файле document.xml.

Списки

Поддерживаются одноуровневые списки типа «маркеры» и с нумерацией.

Пример для формата XML:

```
<lists>
<list name="BULLET_LIST">
 <items>
  <item>bullet item 1</item>
  <item>bullet item 2</item>
  <item>bullet item 3</item>
 </items>
</list>
</lists>
```

Подробный пример запроса с использованием списков (lists.json) и шаблон к нему (lists.docx) можно найти в *архиве примеров*.

Добавление элементов возможно в произвольное место списка в шаблона. В случае пустого списка во входных данных, список удаляется из отчета.

Изображения

См. в пункте “Работа с изображениями” подраздела “Работа с документами-шаблонами”.

Условные выражения

Части текста шаблона могут быть исключены из отчета в зависимости от истинности условия в условном теге.

Формат условного тега:

```
[#условие]условный текст шаблона[/условие]
```

Открывающий тег начинается с символа #, за которым следует условие. Закрывающий тег начинается с символа /, за которым следует условие открывающего тега.

Примечание: Условные выражения в колонтитулах не поддерживаются.

В запросе данные для подстановки в теги условных выражений находятся внутри элемента (объекта) `input-data`.

Пример для формата XML:

```
<input-data>
       <tags>
           <tag name="CONDITIONAL_TAG_TRUE">true</tag>
           <tag name="CONDITIONAL_TAG_FALSE">false</tag>
       </tags>
</input-data>
```

Пример для формата JSON:

```
"input-data":
 {
       "ORGANIZATION": "АО «xxxxyyyy»",
       "TAG_IN_HEADER_TEST": "Пример верхнего колонтитула",
       "TAG_IN_FOOTER_TEST": "Пример нижнего колонтитула",
       "CONDITIONAL_TAG_TRUE": "true",
       "CONDITIONAL_TAG_FALSE": "false"
}
```

Подробный пример запроса с использованием условных выражений (example_from_json.json) и шаблон к нему (example_from_json.docx) можно найти в *архиве примеров*.

Блоки

Блок - это фрагмент отчета, содержащий текст/таблицы/списки (все, кроме изображений, в текущей версии). Создается из шаблона-блока (определенного фрагмента документа-шаблона) произвольное количество раз в произвольном месте шаблона с разными наборами входных данных.

Шаблон блока в документе-шаблоне определяется тегами `[block-template:произвольное имя шаблона блока]`. Шаблоны блока полностью удаляются из документа-шаблона перед заменой тегов на входные данные.

Места вставки блоков определяются тегами экземпляров шаблонов блока `[block-instances:список имен шаблонов блоков через запятую]` документа-шаблона.

Контент шаблона блока вставляется в документ (с заменой тегов в нем) по месту нахождения тега экземпляра шаблона.

Каждый тег экземпляра шаблона блока указывает, какие именно шаблоны нужно взять за основу для создания блока в отчете (другими словами, инстанциировать шаблон блока). При этом рассматриваются все входные данные блоков по порядку. При замене тегов в шаблоне блока все значения тегов берутся только из объекта `data` блока (актуально для текущей версии сервиса отчетов).

В запросе данные блоков определяются массивом `blocks`, который содержит объекты `block` с полями `block-template` (совпадает с именем шаблона блока в документе-шаблоне) и полем объекта `data`. Объект `data` имеет те же дочерние объекты, что и объект `input-data`, кроме изображений и блоков.

Пример:

```
"blocks": [
    {
        "block-template": "block1",
        "data": {
            "таблица1": {
                "rows": [{
                    }
                ]
            },
            "сумма_пени": "12 345,67",
            "сумма_долга": "78 910,23",
            "номер_договора": "1"
        }
    },
    {
        "block-template": "block2",
        "data": {
            "таблица1": {
                "rows": [{
                    }
                ]
            },
            "сумма_пени": "76 543,21",
            "сумма_долга": "987 654,32",
            "номер_договора": "2"
        }
    },
    {
        "block-template": "block3",
        "data": {
            "номер_договора": "1",
            "название_договора": "Название договора 1",
            "дата_с": "01.01.2000",
            "дата_по": "01.01.2100"
        }
    },
    {
        "block-template": "block3",
        "data": {
            "номер_договора": "2",
            "название_договора": "Название договора 2",
            "дата_с": "02.02.2020",
            "дата_по": "02.02.2080"
        }
    }
```

(continues on next page)

(продолжение с предыдущей страницы)

```
]
```

Подробный пример запроса с использованием блоков (example_multiblocks.json) и шаблон к нему (example_multiblocks.docx) можно найти в *архиве примеров*.

Работа с мульти-отчетами

Общая информация

Доступно создание единого отчета из одного шаблона на основе множества входных данных.

Примечание:

- В текущей версии не поддерживаются нумерованные списки.
- В текущей версии не поддерживается создание единого отчета из множества шаблонов.
- Теги в колонтитулах нужно использовать разумно, т.к. нет возможности иметь собственные колонтитулы для каждого отчета.
- Замена изображений поддерживается только глобально. Во всех подотчетах будет набор изображений, примененный последним. Это ограничение текущей версии сервиса.

Примеры запросов

Пример для формата XML

```
<?xml version="1.0" encoding="UTF-8"?>
<request>
   <template uri="local" id="multi_report_1-N"/>
   <input-data-array>
       <input-data>
           <tags>
               <tag name="абонент">абонент1</tag>
               <tag name="адрес_абонента">адрес_абонента1</tag>
               <tag name="период">период1</tag>
               <tag name="договоры">договоры1</tag>
               <tag name="общая_сумма_долга">общая_сумма_долга1</tag>
               <tag name="сумма_задолженности1">сумма_задолженности1_1</tag>
               <tag name="сумма_задолженности2">сумма_задолженности2_1</tag>
               <tag name="дата_пени">дата_пени1</tag>
           </tags>
           <tables>
               <table name="TABLE-FORMATTED">
                   <rows>
                       <row>
                           <cell tag="номер_пп">1</cell>
                           <cell tag="номер_договора">Пример номера 1</cell>
                           <cell tag="район">Пример района 1</cell>
                           <cell tag="предприятие">Пример предприятия 1</cell>
                           <cell tag="дата_отключения">Пример даты 1</cell>
                           <cell tag="адрес">Пример адреса 1</cell>
                           <cell tag="комплексное_поле">Пример суммы 1</cell>
                       </row>
                   </rows>
```

(continues on next page)

(продолжение с предыдущей страницы)

```
                </table>
            </tables>
        </input-data>
        <input-data>
            <tags>
                <tag name="абонент">абонент2</tag>
                <tag name="адрес_абонента">адрес_абонента2</tag>
                <tag name="период">период2</tag>
                <tag name="договоры">договоры2</tag>
                <tag name="общая_сумма_долга">общая_сумма_долга2</tag>
                <tag name="сумма_задолженности1">сумма_задолженности1_2</tag>
                <tag name="сумма_задолженности2">сумма_задолженности2_2</tag>
            </tags>
            <tables>
                <table name="TABLE-FORMATTED">
                    <rows>
                        <row>
                            <cell tag="номер_пп">1</cell>
                            <cell tag="номер_договора">Пример номера 3</cell>
                            <cell tag="район">Пример района 3</cell>
                            <cell tag="предприятие">Пример предприятия 3</cell>
                            <cell tag="дата_отключения">Пример даты 3</cell>
                            <cell tag="адрес">Пример адреса 3</cell>
                            <cell tag="комплексное_поле">Пример суммы 3</cell>
                        </row>
                    </rows>
                </table>
            </tables>
        </input-data>
    </input-data-array>
    <options>
        <enable-debug-report-save>true</enable-debug-report-save>
        <formatting>
            <tables>
                <enable-cells-auto-merge>true</enable-cells-auto-merge>
            </tables>
        </formatting>
        <!-- <report-format>pdf</report-format> -->
    </options>
</request>
```

Пример для формата JSON

```
{
    "template":
    {
        "uri": "local",
        "id": "template_example_1"
    },
    "input-data-array":
    [
        {
            "ORGANIZATION": "АО «xxxxyyyy1»",
            "CLINAME": "МУНИЦИПАЛЬНОЕ ПРЕДПРИЯТИЕ ГОРОДА  \"XXXXXXXXXX\"",
            "CTRNUMBER": "1234",
            "TABLE-NO-FORMAT":
            {
```

(continues on next page)

(продолжение с предыдущей страницы)


```
                "header":
                [
                    "№ п/п1",
                    "№ дела в Арбитражном суде1",
                    "Период задолженности1",
                    "Основной долг, руб. с НДС1",
                    "Проценты, руб.1",
                    "Госпошлина, руб.1",
                    "Неустойка по решению суда1",
                    "Общая сумма погашения, руб.1",
                    "Срок погашения, не позднее1"
                ],
                "rows":
                [
                    [
                        "1_1",
                        "А12-1234/2030_1",
                        "Октябрь 2017_1",
                        "12 345,58_1",
                        null,
                        "23 456,00_1",
                        "78 912,41_1",
                        null,
                        "31.08.2019_1"
                    ]
                ]
            },
            "TABLE-FORMATTED":
            {
                "rows":
                [
                    {
                        "номер_пп": "1_1",
                        "номер_договора": "Пример номера 1_1",
                        "район": "Пример района 1_1",
                        "предприятие": "Пример предприятия 1_1",
                        "дата_отключения": "Пример даты 1_1",
                        "адрес": "Пример адреса 1_1",
                        "комплексное_поле": "Пример суммы 1_1"
                    }
                ]
            },
            "images":
            {
                "IMAGE-1-JPEG":
                {
                    "data": "/9j/4AAQSkZJRgABAQEAYABgAAD…AKKACigD//Z"
                },
                "IMAGE-2-JPEG":
                {
                    "data": "/9j/4AAQSkZJRgABAQEAYABg…oAKKACigAooA//9k="
                }
            }
        },
        {
            "ORGANIZATION": "АО «xxxxyyyy_2»",
            "CLINAME": "МУНИЦИПАЛЬНОЕ ПРЕДПРИЯТИЕ ГОРОДА\"ПАССАЖИРСКИЙ АВТОМОБИЛЬНЫЙ
↪ТРАНСПОРТ_2\"",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
            "CTRNUMBER": "1234_2",
            "RESTPENY": null,
            "TAG_IN_HEADER_TEST": "Пример верхнего колонтитула_2",
            "TAG_IN_FOOTER_TEST": "Пример нижнего колонтитула_2",
            "CONDITIONAL_TAG_TRUE": "true",
            "CONDITIONAL_TAG_FALSE": "false",
            "TABLE-NO-FORMAT":
            {
                "header":
                [
                    "№ п/п_2",
                    "№ дела в Арбитражном суде_2",
                    "Период задолженности_2",
                    "Основной долг, руб. с НДС_2",
                    "Проценты, руб._2",
                    "Госпошлина, руб._2",
                    "Неустойка по решению суда_2",
                    "Общая сумма погашения, руб._2",
                    "Срок погашения, не позднее_2"
                ],
                "rows":
                [
                    [
                        "1_2",
                        "А12-1234/2030_2",
                        "Октябрь 2017_2",
                        "12 345,58_2",
                        null,
                        "23 456,00_2",
                        "78 912,41_2",
                        null,
                        "31.08.2019_2"
                    ]
                ]
            },
            "TABLE-FORMATTED":
            {
                "rows":
                [
                    {
                        "номер_пп": "1_2",
                        "номер_договора": "Пример номера 1_2",
                        "район": "Пример района 1_2",
                        "предприятие": "Пример предприятия 1_2",
                        "дата_отключения": "Пример даты 1_2",
                        "адрес": "Пример адреса 1_2",
                        "комплексное_поле": "Пример суммы 1_2"
                    }
                ]
            }
        }
    ],
    "options":
    {
        "enable-debug-report-save": true,
        "formatting":
        {
```


(continues on next page)

(продолжение с предыдущей страницы)

```
            "tables":
            {
                "enable-cells-auto-merge": true
            }
        }
    }
}
```

Подробный пример запроса для формирования мульти-отчета (multi_report_1-N_json.json) и шаблон к нему (multi_report_1-N_json.docx) можно найти в *архиве примеров*.

Примеры вызова сервиса

Для формата XML

```
curl --request POST --data-binary "@templates/examples/docx/multi_report_1-N_xml.xml"␣
↪http://localhost:8087/word_multi_report_1_N_xml
```

Для формата JSON

```
curl --request POST --data-binary "@templates/examples/docx/multi_report_1-N_json.json
↪" http://localhost:8087/word_multi_report_1_N_json
```

7.1.4 Формирование отчета формате XLSX

Описание сервиса

Наименование сервиса	Сервис формирования мульти-отчетов из документа-шаблона в формате **XLSX**
Путь к сервису	• [host]:[port]/excel_report_json • [host]:[port]/excel_report_xml • [host]:[port]/excel_report_json/v2
Метод	POST
Параметры	Тело запроса должно содержать JSON или XML объект. Подробнее про структуру тела запроса можно прочитать в подразделе “Структура тела запроса”. В ответ сервис отдаёт файл документа в формате XLSX, закодированный в base64. При конвертации отчета в формат PDF сервис возвращает файл документа в формате PDF, закодированный в base64.
Назначение	Сервис предназначен для формирования мульти-отчетов из документа-шаблона в формате **XLSX**. Возможна конвертация в PDF.

Структура тела запроса

Тело запроса содержит объект в формате JSON или XML, который включает:

1. Описатель шаблона.
2. Входные данные для подстановки в шаблон вместо тегов.
3. Опции запроса.
4. Формат ответа.

Описатель шаблона

Элемент (объект) описателя шаблона `template` содержит:

- `uri` - строка. Задает расположение файла документа-шаблона в зависимости от того, как трактуется параметр `id`. Поддерживаемое значение: `local`.
- `id` - строка. Идентификатор шаблона. Путь к файлу документа-шаблона относительно директории сервиса `templates`. Также используется для записи файла отчета в отладочном режиме и для идентификации запроса в логе.

Пример для формата XML:

```
<template uri="local" id="Информационное письмо"/>
```

Пример для формата JSON:

```
      "template":
{
    "uri": "local",
    "id": "template_example_1"
}
```

Входные данные

Элемент (объект) входных данных `input-data` содержит:

- Простые строковые данные для подстановки в шаблон вместо тегов.
- Описатели таблиц.
- Данные для подстановки в теги изображений.

Примечание: Отличие структуры данных для формата JSON от XML в том, что вместо XML-элементов данные представлены как объекты и списки объектов. Имена тегов задаются именами полей объектов. Таблицы и изображения являются дочерними объектами по отношению к объекту `input-data`.

Простые строковые данные для подстановки в шаблон вместо тегов

Для формата XML

Элемент `tags` содержит список дочерних элементов `tag` с атрибутом `name` и нужным конкретным значением, на который тег в шаблоне будет заменен при генерации отчета. Атрибут `name` совпадает с тегом в шаблоне.

Пример:

```
<input-data>
    <tags>
        <tag name="ORGANIZATION">АО «xxxxyyyy»</tag>
    </tags>
</input-data>
```

Для формата JSON

Имена тегов задаются именами объектов внутри `input-data`.

Пример:

```
"input-data":
   {
       "ORGANIZATION": "АО «xxxxyyyy»"
}
```

Описатели таблиц

Отличие от формата DOCX:

- теги задаются в элементе (объекте) `cell-tags` один раз для всей таблицы;
- тег ячейки отображает тег столбца на индекс ячейки в массиве, задающем строку таблицы.

Для формата XML

Элемент описателей таблиц `tables` содержит список дочерних элементов `table` с атрибутом `name`. Атрибут `name` совпадает с тегом в шаблоне. Тег в шаблоне будет заменен на таблицу, созданную согласно описанию таблицы в запросе.

Пример:

```
<input-data>
     <tables>
        <table name="таблица_застрахованные_лица">
            <cell-tags>
                <cell-tag name="номер_пп" index="0"/>
                <cell-tag name="снилс" index="1"/>
                <cell-tag name="пол" index="2"/>
                <cell-tag name="фио" index="3"/>
                <cell-tag name="дата_рождения" index="4"/>
                <cell-tag name="дата_договора" index="5"/>
                <cell-tag name="номер_договора" index="6"/>
            </cell-tags>
            <rows>
                <row>
                    <cell>1</cell>
                    <cell>001-002-003 00</cell>
```

(continues on next page)

(продолжение с предыдущей страницы)

```
                    <cell>Ж</cell>
                    <cell>Горбунова Екатерина Васильевна</cell>
                    <cell>01.01.1970</cell>
                    <cell>01.01.1988</cell>
                    <cell>ABC-001-002-003-00</cell>
                </row>
            </rows>
        </table>
    </tables>
</input-data>
```

Для формата JSON

Описатели таблиц являются дочерними объектами по отношению к объекту `input-data`.

Пример:

```
"input-data":
   {
        "таблица_застрахованные_лица":
       {
           "cell-tags": {
               "номер_пп": 0,
               "снилс": 1,
               "пол": 2,
               "фио": 3,
               "дата_рождения": 4,
               "дата_договора": 5,
               "номер_договора": 6
           },
           "rows":
           [
               [
                   "1",
                   "001-002-003 00",
                   "Ж",
                   "Горбунова Екатерина Васильевна",
                   "01.01.1970",
                   "01.01.1988",
                   "ABC-001-002-003-00"
               ]
           ]
       }
   }
```

Данные для подстановки в теги изображений

Для формата XML

Элемент `images` содержит список дочерних элементов `image` с атрибутом `name`. Атрибут `name` совпадает с тегом в шаблоне. Изображения в шаблоне будут заменены на данные изображений из запроса. В запросе передается новое содержимое файла изображения, закодированное в BASE64.

Пример:

```
<input-data>
    <images>
       <image name="IMAGE-1-JPEG">/9j/4AAQSkZJRgABAQEAYABgAAD/4Q…CigAooAKKACigD//Z</
↪image>
       <image name="IMAGE-2-JPEG">/9j/4AAQSkZJRgABAQEAYABg…KKACigAooAKKACigAooA//9k=</
↪image>
   </images>
</input-data>
```

Для формата JSON

Данные для подстановки в теги изображений находятся внутри дочерних объектов объекта `input-data`.

Пример:

```
"input-data":
  {
      "images":
      {
          "IMAGE-1-JPEG":
          {
              "data": "/9j/4AAQSkZJRgABAQEAYABgAAD…igAooAKKACigAooAKKACigD//Z"
          },
          "IMAGE-2-JPEG":
          {
              "data": "/9j/4AAQSkZJRgABAQEAYAB…AooAKKACigAooAKKACigAooA//9k="
          }
      }
  }
```

Опции запроса

Элемент (объект) опций запроса `options` содержит:

- `enable-debug-report-save` - логическое значение. Указывает создавать ли копию отчета документа и копию запроса (имя файла - идентификатор шаблона) в локальной директории `reports_debug` сервиса. Значение по умолчанию - `false`.
- `enable-binary-output` - логическое значение. Указывает, что сервис будет выдавать результат в виде бинарных данных, без кодирования в base64. В случае возникновения ошибки ответ будет соответствовать *используемому по умолчанию*. Значение по умолчанию - `false`.
- `report-format` - строка. Указывает формат файла отчета, если он отличен от формата, упомянутого в url запроса. Единственное непустое поддерживаемое значение: “`pdf`”. Значение по умолчанию: пустая строка. См. пункт “Конвертация отчета в PDF” ниже.

Примечание: в шаблоне документа должны быть заданы области печати («Set Print Area»), в соответствии с ними будет произведена конвертация в PDF

Пример для формата XML:

```
<options>
    <enable-debug-report-save>true</enable-debug-report-save>
    <enable-binary-output>false</enable-binary-output>
    <report-format>pdf</report-format>
</options>
```

Пример для формата JSON:

«options»:

```
{
    "enable-debug-report-save": true,
    "enable-binary-output": false,
    "report-format": "pdf"
}
```

Формат ответа

Элемент (объект) формата ответа `response-format` может принимать значения `json` и `xml`.

Подробнее про то, как задается формат ответа можно прочитать в подразделе “Структура ответа”.

Пример для формата XML

```
<response-format value="xml"/>
```

Пример для формата JSON:

```
{"response-format": "json"}
```

Примеры запросов

Пример для формата XML

```
<?xml version="1.0" encoding="UTF-8"?>
<request>
   <response-format value="xml"/>
   <template uri="local" id="template_example"/>
   <input-data>
       <tags>
           <tag name="организация">Государственное учреждение-Отделение Пенсионного␣
↪фонда Российской Федерации</tag>
           <tag name="число">25</tag>
           <tag name="месяц">декабря</tag>
           <tag name="год">2018</tag>
           <tag name="номер_документа">01</tag>
           <tag name="число_договоров">3</tag>
           <tag name="сотрудник">Сидорова Мария Петровна</tag>
           <tag name="тел_сотрудника">+7 (123) 456-78-90</tag>
           <tag name="тег1">тег1_значение</tag>
           <tag name="тег2">тег2_значение</tag>
       </tags>
       <tables>
           <table name="таблица_застрахованные_лица">
               <cell-tags>
                   <cell-tag name="номер_пп" index="0"/>
                   <cell-tag name="снилс" index="1"/>
                   <cell-tag name="пол" index="2"/>
                   <cell-tag name="фио" index="3"/>
                   <cell-tag name="дата_рождения" index="4"/>
                   <cell-tag name="дата_договора" index="5"/>
```

(continues on next page)

(продолжение с предыдущей страницы)

```
                <cell-tag name="номер_договора" index="6"/>
            </cell-tags>
            <rows>
                <row>
                    <cell>1</cell>
                    <cell>001-002-003 00</cell>
                    <cell>Ж</cell>
                    <cell>Горбунова Екатерина Васильевна</cell>
                    <cell>01.01.1970</cell>
                    <cell>01.01.1988</cell>
                    <cell>ABC-001-002-003-00</cell>
                </row>
            </rows>
        </table>
    </tables>
    <images>
        <image name="IMAGE-1-JPEG">/9j/4AAQSkZJRgABAQEAYABgAAD/4QFQR0...
↪KACigAooAKKACigAooAKKACigD//Z</image>
        <image name="IMAGE-2-JPEG">/9j/4AAQSkZJRgABAQEAYABgAAD/4QFQR...
↪oAKKACigAooAKKACigAooA//9k=</image>
    </images>
</input-data>
<options>
    <enable-debug-report-save>true</enable-debug-report-save>
    <!-- <report-format>pdf</report-format> -->
</options>
</request>
```

Пример для формата JSON

```
{
   "response-format": "json",
   "template":
   {
       "uri": "local",
       "id": "template_example"
   },
   "input-data":
   {
       "организация": "Государственное учреждение-Отделение Пенсионного фонда␣
↪Российской Федерации",
       "число": "25",
       "месяц": "декабря",
       "год": "2018",
       "номер_документа": "01",
       "число_договоров": "3",
       "сотрудник": "Сидорова Мария Петровна",
       "тел_сотрудника": "+7 (123) 456-78-90",
       "тег1": "тег1_значение",
       "тег2": "тег2_значение",
       "таблица_застрахованные_лица":
       {
           "cell-tags": {
               "номер_пп": 0,
               "снилс": 1,
               "пол": 2,
               "фио": 3,
```

(continues on next page)

(продолжение с предыдущей страницы)

```
                "дата_рождения": 4,
                "дата_договора": 5,
                "номер_договора": 6
            },
            "rows":
            [
                [
                    "1",
                    "001-002-003 00",
                    "Ж",
                    "Горбунова Екатерина Васильевна",
                    "01.01.1970",
                    "01.01.1988",
                    "ABC-001-002-003-00"
                ]
            ]
        },
        "images":
        {
            "IMAGE-1-JPEG":
            {
                "data": "/9j/4AAQSkZJRgABAQEAYABgAAD/4QFQR0...
↪KACigAooAKKACigAooAKKACigD//Z"
            },
            "IMAGE-2-JPEG":
            {
                "data": "/9j/4AAQSkZJRgABAQEAYABgAAD/4QFQR...oAKKACigAooAKKACigAooA//
↪9k="
            }
        }
    },
    "options":
    {
        "enable-debug-report-save": true
    }
}
```

7.1.5 Пример вызова сервиса

Для формата XML

```
curl --request POST --data-binary "@templates/examples/xlsx/base_example_xml.xml"␣
↪http://localhost:8087/excel_report_xml
```

Для формата JSON

```
curl --request POST --data-binary "@templates/examples/xlsx/base_example.json" http://
↪localhost:8087/excel_report_json
```

7.1.6 Структура ответа

Формат ответа (JSON или XML) может быть задан двумя способами:

1. в HTTP заголовке Accept. Поддерживаемые значения: `application/json` и `application/xml`.
2. в теле запроса в элементе (объекте) `response-format`. Поддерживаемые значения: `json` и `xml`.

Приоритет имеет формат, указанный в теле запроса.

Ответ сервиса содержит объект в формате JSON или XML, который включает:

1. Описание ошибки (код ошибки, сообщение об ошибке). В случае успешного ответа сервиса возвращается значение `null`.
2. Результат (закодированный в base64 файл документа отчета в формате xlsx).

Формат ответа

```
{
   error: {
       code
       message
   }
   result: "[base64 encoded xlsx file]"
}
```

Пример ответа

```
{
   "error": null,
   "result": "UEsDBBQACAAIAPdKo...JwAAAAA="
}
```

7.1.7 Работа с документом

Общая информация

Для документов в формате XLSX доступны:

- теги, которые будут заменены входными данными из запроса;
- таблицы;
- изображения.

Примечание:

- Формат входных данных ячеек - текст и числа (целые или вещественные)..
- Ссылки на ячейки в формулах при генерации отчета не корректируются. Формулы могут стать неверными в случае добавления/удаления строк в отчет.

- Условные выражения в текущей версии не поддерживаются.

Работа с таблицами

Существует два варианта работы с таблицами.

v1

API: `/excel_report_json`

Таблица с форматированием задается следующим образом:

- первая ячейка строки содержит тег с названием таблицы. Эта строка удаляется при генерации.
- «строка тегов», строка (строки, см ниже) следующая за строкой тега таблицы, содержит теги столбцов, их форматирование будет применено к сгенерированным строкам (если не задана строка форматирования, см ниже). Эта строка (строки) удаляется при генерации.
- опционально: строка, указывающая, что в следующей строке содержится строка форматирования. Содержит одну ячейку с текстом "[формат ячеек]". Применяется для таблиц, где числовые данные должны быть в ячейках с форматом число (или general), а не строками (иначе проще задать форматирование в строке тегов).
- опционально: "строка форматирования", строка, задающая формат ячеек вместо строки тегов. Значения ячеек игнорируются. Формат применяется к сгенерированным строкам. Должна быть полной копией строки тегов с точки зрения количества ячеек и их порядка.

Поддерживается возможность отображения одной строки входных данных на множество строк в документе. Это может понадобиться для таблиц, где в одной логической строке есть ячейки с объединением по вертикали. Количество строк шаблона на одну строку входных данных определяется как максимальное число строк объединенных по вертикали в любой ячейке из первой строки после строки с тегом самой таблицы (см. функцию `getNumberOfSheetRowsPerInputDataRow`).

Пример шаблона можно найти в *архиве примеров*.

Примечание: на данный момент применение форматирования к сгенерированным строкам не включает в себя настройки шрифта. Поддерживается цвет фона и настройки границ ячеек (например, горизонтальное слияние ячеек).

---

v2

API: `/excel_report_json/v2`

Таблица с форматированием задается следующим образом:

- первая ячейка строки содержит тег с названием таблицы. Эта строка удаляется при генерации.
- «строка тегов», строка следующая за строкой тега содержит теги столбцов, их форматирование будет применено к сгенерированным строкам(если не задана строка форматирования, см ниже). Эта строка удаляется при генерации.
- опционально: строка, указывающая, что в следующей строке содержится строка форматирования. Содержит одну ячейку с текстом "[формат ячеек]". Применяется для таблиц, где числовые данные должны быть в ячейках с форматом число(или general), а не строками(иначе проще задать форматирование в строке тегов).
- опционально: "строка форматирования", строка, задающая формат ячеек вместо строки тегов. Значения ячеек игнорируются. Формат применяется к сгенерированным строкам. Должна быть полной копией строки тегов с точки зрения количества ячеек и их порядка.

Примеры шаблонов можно найти в *архиве примеров*.

- `cell_format_options.xlsx`
- `cell_format_options_typed_cells.xlsx`

Примечание: на данный момент применение форматирования к сгенерированным строкам не включает в себя настройки шрифта. Поддерживается цвет фона и настройки границ ячеек (например, горизонтальное слияние ячеек).

Управление слиянием ячеек

Базовый алгоритм слияния - применение горизонтального слияния генерируемых ячеек на основе слияния ячеек таблицы шаблона.

Поддерживается расширенные опции вертикального слияния. Задается объектом с единственным полем:

- `vertical_span` - целое число. Управляет вертикальным слиянием ячейки. Значение задает количество ячеек для объединения вниз от текущей ячейки (включительно)

Опции могут быть привязаны к столбцам таблицы в шаблоне, задаваемым по индексу

```
"formatting": {
   "template-cells": [
       {
           "vertical_span": 2
       },
       ...
   ]
}
```

или к тегам ячеек (данный способ имеет приоритет, что удобно для избирательной коррекции опций слияния для части столбцов):

```
"formatting": {
   "tags": {
       "cell_tag_value": {
           "vertical_span": 2
       },
   }
}
```

Формула как значение ячейки таблицы

Формула в запросе задается объектом с полями

- `type`, строка, поддерживаемое значение: “formula”
- `format`, строка, поддерживаемое значение: “raw”
- `value`, строка, значения берутся из используемого редактора таблиц, без знака равенства.

```
{
   "type": "formula",
   "format": "raw",
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    "value": "SUM(D1:D20)"
},
```

Преобразований номеров строк не выполняется. Предполагается, что номера строк актуальны для финального отчета, то есть после вставки строк таблицы, идущих перед строкой, содержащей формулу. Номер строк можно рассчитать на основе номеров строк таблицы в документе-шаблоне и количества строк таблицы на момент генерации строки с формулой в запросе.

Пример запроса: `formula.json`. Его также можно найти в *архиве примеров*.

7.1.8 Добавление строк в конец документа шаблона

Строки, заданные списком `rows-to-add` будут добавлены в конец шаблона без сохранения/применения форматирования

См. пример запроса `add_rows_to_end.json` в *архиве примеров*.

7.1.9 Изображения

См. в пункте “Работа с изображениями” подраздела “Работа с документами-шаблонами”.

7.1.10 Конвертация отчета в PDF

Активация

В опциях запроса указать формат отчета «pdf».

Пример:

```
<options>
    <report-format>pdf</report-format>
</options>
```

Зависимости

Необходим установленный пакет Libre Office. Программа `soffice` должна быть доступна для вызова. Возможно задать путь к ней в параметрах командной строки сервера отчетов или через файл конфигурации.

Принцип работы

Сервер отчетов сохраняет отчет в файл и вызывает утилиту `soffice` со следующими параметрами, для примера

```
soffice --convert-to pdf report.docx --outdir temp_dir
soffice --convert-to pdf report.xlsx --outdir temp_dir
```

затем высылает вместо оригинала отчета полученный файл PDF.

Умолчания

Путь к утилите `soffice` ищется в директориях, заданных переменной среды `PATH`. Можно указать прямой путь в аргументах командной строки сервера отчетов или через файл конфигурации.

Файлообмен с `soffice` идет через временную папку системы. Можно указать рабочую директорию в аргументах командной строки сервера отчетов или через файл конфигурации.

См. вывод команды

```
xreports --help
```

Производительность

По умолчанию время исполнения команды

```
soffice --convert-to pdf
```

для пустого документа - 2.3 секунды (для cpu i7-3615M, ssd). Основное время тратится на инициализацию Libre Office.

Чтобы ускорить конверсию, можно воспользоваться режимом `quick start` , то есть предварительным исполнением команды

```
soffice --quickstart
```

В этом режиме Libre Office все время находится в оперативной памяти, что экономит примерно 2 секунды.

Примечание: для Libre Office 6.3.4.2 в Ubuntu 19.10 `soffice --quickstart` приводит к показу главного окна. Параметры `--minimized` и `--headless` не решают эту проблему, т.к. не позволяют пользоваться ускорением после однократной конверсии. В Windows 10 при таком запуске LO виден только в трее.

7.1.11 Работа со штрихкодом

Возможен вывод строки текста в виде штрихкода. Для корректной работы в отчетах Word и Excel в операционной системе должны быть установлены шрифты семейства Libre Barcode. Доступны в репозитории librebarcode.

Каждый шрифт представляет собой определенный формат штрихкода. Поддерживаются следующие форматы:

- Code-128
- Code-39
- EAN13

Шрифт может включать оригинальный текст непосредственно под штрихкодом (шрифты с суффиксом Text).

Рассмотрим применение на примере формата Code-128.

Code-128

Шрифты для установки:

- `LibreBarcode128Text-Regular.ttf`, будет доступен под именем «Libre Barcode 128 Text»
- `LibreBarcode128-Regular.ttf`, будет доступен под именем «Libre Barcode 128»

Как использовать

В шаблоне для обычного тега выбрать шрифт. Например, «Libre Barcode 128 Text».

Тестовые запросы

Документ в формате DOCX

```
curl --request POST --data-binary "@templates/examples/docx/barcode_code-128.json"␣
↪http://localhost:8087/word_report_json
```

Документ в формате XLSX

```
curl --request POST --data-binary "@templates/examples/xlsx/barcode_code-128.json"␣
↪http://localhost:8087/excel_report_json
```

7.2 Генератор отчетов PDF

7.2.1 Общее описание сервиса

Сервис формирует отчет в формате PDF по HTTP-запросу в формате JSON.

PDF-документ генерируется при помощи специальных команд генератора.

7.2.2 Формирование документов с помощью генератора PDF

Описание сервиса

Наименование сервиса	Генератор отчетов PDF
Путь к сервису	[host]:[port]/pdf_report_json
Метод	POST
Параметры	Тело запроса должно содержать объект в формате JSON. Подробнее про структуру тела запроса можно прочитать в подразделе “Структура тела запроса”. В ответ сервис отдаёт файл документа в формате PDF, закодированный в base64.
Назначение	Сервис предназначен для генерации документов в формате PDF.

Структура тела запроса

Тело запроса содержит объект в формате JSON, который включает:

1. Описатель генератора PDF.
2. Входные данные для подстановки в шаблон вместо тегов.
3. Опционально: генератор PDF.
4. Опции запроса.

Пример:

```
{
    "report-generator":
    {
        "uri": "local",
        "id": "example_1"
    },
    "input-data":
    {
        "ORGANIZATION": "Акционерное общество Негосударственный пенсионный фонд
↪«XXXXX»",
        "CLINAME": "Петров Петр Петрович"
    },
    "options":
    {
        "enable-debug-report-save": true
    }
}
```

Описатель генератора PDF

Объект описателя генератора PDF `report-generator` содержит:

- `uri` - строка. Задает положение источника команд генератора в зависимости от того, как трактуется параметр `id`. Поддерживаемые значения: `local` и `embedded`.
- `id` - строка. ID генератора.

Возможны два варианта использования генератора PDF:

- генератор как часть запроса (все команды генератора размещаются внутри объекта `embedded-report-generator` тела запроса);
- генератор как часть сервиса (все команды генератора размещаются внутри файла JSON в специальной папке `pdf_report_gen` на сервере).

Соответственно:

- При значении `uri`, равному `embedded`, `id` используется только для диагностических сообщений и идентификации запроса. Генератор должен находится в корневом объекте `embedded-report-generator` запроса. См. пункт «Генераторы как часть запроса» ниже.
- При значении `uri`, равному `local`, `id` - это имя файла в директории сервиса `pdf_report_gen`. См. пункт «Генераторы как часть сервиса» ниже.

Генератор как часть запроса

Опционально генератор может быть встроен в запрос.

В этом случае все команды генератора указываются в объекте `embedded-report-generator`.

Пример:

```
{
  "report-generator":{
     "id":"example",
     "uri":"embedded"
  },
  "embedded-report-generator":{
     "commands":[
        {
           "name":"NewPage"
        },
        {
           "name":"setCoordinateMode",
           "params":{
              "mode":"millimeters"
           }
        },
        {
           "name":"setCellMargin",
           "params":{
              "margin":1
           }
        }
     ]
  },
  "options":{
     "enable-debug-report-save":false
  }
}
```

Генератор как часть сервиса

Цель: не передавать данные команд внутри запросов, если они не меняются от запроса к запросу.

В этом случае все команды генератора указываются в файле формата JSON, который должен находиться в директории `pdf_report_gen`, расположенной в директории приложения сервиса.

Формат генератора внутри файла в директории `pdf_report_gen`:

```
{
    "report-generator":
    {
         "commands": [...]
    }
}
```

Пример генератора в теле запроса:

```
"report-generator":
   {
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        "uri": "local",
        "id": "example"
    }
```

, где `id` - это имя файла в директории сервиса `pdf_report_gen`.

Пример простого запроса для генератора внутри файла example.json в директории `pdf_report_gen`:

```
{
    "report-generator":
    {
        "uri": "local",
        "id": "example"
    },
    "options":
    {
        "enable-debug-report-save": true
    }
}
```

Входные данные

Объект входных данных `input-data` содержит входные, которые подставляются в теги.

Примечание: Команды генератора PDF, которые работают с текстом, могут содержать теги, которые будут заменены входными данными из запроса. Подробнее об использовании тегов можно прочитать в разделе “Теги” ниже.

Пример:

```
"input-data":
    {
        "ORGANIZATION": "Акционерное общество Негосударственный пенсионный фонд␣
↪«XXXXX»",
        "CLINAME": "Петров Петр Петрович"
    }
```

Опционально: генератор PDF

Объект генератора PDF `embedded-report-generator` содержит команды генератора. Используется опционально при значении `uri`, равному `local`.

Подробнее описан в пункте “Генераторы как часть запроса”.

Опции запроса

Объект опций запроса `options` имеет поля:

- `enable-debug-report-save` - логическое значение. Указывает создавать ли копию отчета документа и копию запроса (имя файла - идентификатор шаблона) в локальной директории `reports_debug` сервиса. Значение по умолчанию - `false`.
- `enable-binary-output` - логическое значение. Указывает, что сервис будет выдавать результат в виде бинарных данных, без кодирования в base64. Значение по умолчанию - `false`.
- `enable-debug-pdf-log` - логическое значение. Включает расширенный диагностический лог создания PDF. Значение по умолчанию - `false`.

Пример:

```
"options":
    {
        "enable-debug-merged-doc-save": true,
        "enable-binary-output": false,
        "enable-debug-pdf-log": true
    }
```

7.2.3 Пример вызова сервиса

```
curl --request POST --data-binary "@templates/examples/pdf/embedded-report-generator/
↪request_example.json" http://localhost:8087/pdf_report_json
```

Структура ответа

Ответ сервиса содержит объект в формате JSON, который включает:

1. Описание ошибки (код ошибки, сообщение об ошибке). В случае успешного ответа сервиса возвращается значение `null`.
2. Результат (закодированный в base64 сформированный PDF-файл документа).

Формат ответа

```
{
   error: {
       code
       message
   }
   result: "[base64 encoded pdf file]"
}
```

Пример ответа


```
{
   "error": null,
   "result": "UEsDBBQACAAIAPdKo...JwAAAAA="
}
```


Использование шаблонов страниц

Для формирования документа с помощью генератора PDF можно использовать готовые шаблоны страниц в формате PDF.

Для задания шаблона конкретной страницы существует специальная команда `setPageTemplate`. Подробнее про команду можно прочитать в подразделе “Команды генератора”.

Примечание: Поддерживаются многостраничные шаблоны.

Для расположения текста и других элементов на странице шаблона необходимо задавать координаты. Подробнее про задание координат элементов документа можно прочитать в подразделе “Координаты”.

Шаблоны страниц в формате PDF хранятся локально в директории `templates/pdf`, расположенной в директории приложения сервиса.

Служебные директории

Скрипты генерации PDF хранятся в директории `pdf_report_gen`, расположенной в директории приложения сервиса.

Шаблоны страниц в формате PDF хранятся локально в директории `templates/pdf`, расположенной в директории приложения сервиса.

Примеры запросов находятся в директории `request_examples/pdf`.

Шрифты для использования в генераторе, находятся в директории `assets/fonts`, расположенной в директории приложения сервиса. Загружаются при старте сервиса.

Примечание: При добавлении новых файлов шрифтов необходимо перезапустить сервис.

Примеры запросов и шаблонов

Примеры запросов находятся в директории `examples/pdf/embedded-report-generator` и `examples/pdf/report-generator` в *архиве примеров*.

Примеры шаблонов находятся в директории `examples/pdf`в *архиве примеров*.

7.2.4 Команды генератора

PDF-документ генерируется при помощи специальных команд генератора.

Команды располагаются внутри объекта `embedded-report-generator`, если генератор встраивается в запрос (см. пункт “Генератор как часть запроса” выше).

Команды также могут располагаться в файле формата JSON, который находится в директории `pdf_report_gen`, расположенной в директории приложения сервиса (см. пункт “Генератор как часть сервиса” выше).

Команды описываются внутри массива `commands`.

Пример:

```
"commands": [
        {
           "name":"NewPage"
        },
        {
           "name":"setCoordinateMode",
           "params":{
              "mode":"millimeters"
           }
        },
        {
           "name":"setCellMargin",
           "params":{
              "margin":1
           }
        }
     ]
```

Массив `commands` может включать команды, описанные ниже.

NewPage

Создает новую страницу в документе.

Параметры

- `orientation` - строка. Поддерживаемые значения:
  - «L» - альбомная ориентация
  - «P» - портретная ориентация

Значение по умолчанию - “P”.

Пример:

```
{
  "name":"NewPage",
  "params":{
      "orientation":"L"
   }
}
```

setCoordinateMode

Задает трактовку координат и размеров, указываемых в последующих командах. Режим по умолчанию «в миллиметрах”.

Параметры

- `mode` Поддерживаемые значения: «millimeters», «percents», «normalized».

setPageMargins

Устанавливает отступы от границ страницы.

Параметры

`left`, `right`, `top`, `bottom` - числа с плавающей точкой. Отступы от левого, правого, верхнего, нижнего краев страницы. При достижении нижнего отступа в командах типа `Row_Print` автоматически создается новая страница. Команда заменяет plpdf функции `setAllMargin` и `SetAutoNewPage`.

Пример:

```
{
  "name":"setPageMargins",
  "params":{
     "left":20,
     "right":20,
     "top":30,
     "bottom":15
    }
 }
```

setCurrentX

Устанавливает X координату курсора.

Параметры

- `value` - число с плавающей точкой. Координата X.

setCurrentY

Устанавливает Y координату курсора.

Параметры

- `value` - число с плавающей точкой. Координата Y.

setCurrentXY

Устанавливает координаты курсора.

Параметры

- `x` - число с плавающей точкой. Координата X.
- `y` - число с плавающей точкой. Координата Y.

saveX

Сохраняет координату X для последующего использования в параметрах координат команд через тег `[PDF:savedX]`.

Параметры

- `value` - опциональный параметр, по умолчанию равен текущей координате X. Может быть числом с плавающей точкой или строкой-выражением со ссылками на встроенные теги. Пример выражения: `"[PDF:currentX] - 50"`

saveY

Сохраняет координату Y для последующего использования в параметрах координат команд через тег `[PDF:savedY]`. Внимание: при последующем создании новой страницы значение может оказаться нерелевантным.

Параметры

- `value` - опциональный параметр, по умолчанию равен текущей координате Y. Может быть числом с плавающей точкой или строкой-выражением со ссылками на встроенные теги. Пример выражения: `"[PDF:currentY] - 50"`

saveCoordinate

Сохраняет произвольное значение координат для последующего использования в параметрах координат команд через тег `[SAVED:ключ]`. Дополняет команды `saveX/saveY` в случаях, когда нужен доступ к нескольким значениям.

Параметры

- `key` - строка. Произвольный ключ для использования в теге [SAVED:ключ]
- `value` - может быть числом с плавающей точкой или строкой-выражением со ссылками на встроенные теги. Пример выражения: `"[PDF:currentY] - 50"`

Пример:

```
{
  "name":"saveCoordinate",
  "params":{
     "key":"x1",
     "value":10
  }
}
```

setRotate

Задает поворот всех последующих объектов.

Параметры

- `angle` - число с плавающей точкой. Угол вращения в градусах.
- `x`- число с плавающей точкой. X координата точки вращения. Если null, то X координата курсора
- `y` - число с плавающей точкой. Y координата точки вращения. Если null, то Y координата курсора

Пример:

```
{
  "name":"setRotate",
  "params":{
     "angle":45,
     "x":105,
     "y":130,
   }
}
```

Отмена команды:

```
{
  "name":"setRotate",
  "params":{
     "angle":0
  }
}
```

startOpacity

Указывает степень прозрачности для всех последующих объектов. Действие отменяется командой `endOpacity`.

Параметры

- `val` - число с плавающей точкой. Допустимый диапазон от 0.0 до 1.0, где ноль - полная прозрачность.

endOpacity

Отменяет действие команды `startOpacity`.

Без параметров.

SetPrintFont

Задает шрифт и размер шрифта для последующих текстовых команд.

Параметры

- `font_id` Поддерживаемые значения: имена файлов шрифтов из директории `assets/fonts` без расширения. Шрифты загружаются при старте сервиса.
- `size` Размер шрифта в точках в пользовательском пространстве документа pdf. Никаких отличий от PLPDF и других текстовых редакторов.

Примечание: Единственная поддерживаемая кодовая страница - cp1251.

setColor4Text

Задает цвет текста для последующих текстовых команд.

Варианты входных параметров

- `r`, `g`, `b` целочисленные десятичные компоненты RGB цвета в диапазоне от 0 до 255. Пример красного цвета: “r”:255, “g”:0, “b”:0.
- `color` строка в формате «#rrggbb», где rgb - шестнадцатеричные компоненты RGB цвета в диапазоне от 0 до ff. Пример красного цвета: «color»:“#ff0000”.

setColor4Drawing

Задает цвет границ для объектов, выводимых после исполнения данной команды. Примеры объектов: ячейки таблиц, окружности, линии, прямоугольники и т.д.

Параметры: аналогично команде `setColor4Text`.

setColor4Filling

Задает цвет заливки внутренних областей для объектов, выводимых после исполнения данной команды. Примеры объектов: ячейки таблиц, окружности, прямоугольники и т.д.

Параметры: аналогично команде `setColor4Text`.

setLineWidth

Устанавливает толщину линий, выводимых после исполнения данной команды. Примеры объектов: границы ячеек таблиц, линии, прямоугольники и т.д.

Параметры

- `width` - число с плавающей точкой. Толщина линии.

setCellMargin

Устанавливает отступы контента табличной ячейки от левой, верхней, правой и нижней границ.

Параметры

- `margin` - число с плавающей точкой.

setCellBottomMargin

Устанавливает отступ контента табличной ячейки от нижней границы.

Параметры

- `margin` - число с плавающей точкой.

setCellLeftMargin

Устанавливает отступ контента табличной ячейки от левой границы.

Параметры

- `margin` - число с плавающей точкой.

setCellTopMargin

Устанавливает отступ контента табличной ячейки от верхней границы.

Параметры

- `margin` - число с плавающей точкой.

setCellRightMargin

Устанавливает отступ контента табличной ячейки от правой границы.

Параметры

- `margin` - число с плавающей точкой.

PrintCell

Выводит прямоугольную ячейку с текстом внутри. Можно указать цвет и наличие границ.

Параметры

- `w` - число с плавающей точкой. Ширина прямоугольника ячейки.
- `h` - число с плавающей точкой. Высота прямоугольника ячейки.
- `txt` - текст
- `border` - строка. Задает видимые границы ячейки. Допустимые значения: 0 - нет границ, 1 - внешняя граница прямоугольника, L - левая, T - верхняя, R - правая, B - нижняя границы. Допускается комбинация значений L,T,R и B.
- `align` - строка. Горизонтальное выравнивание текста. Значения: L - по левому краю. R - по правому краю, C - по центру, J - равномерно по ширине ячейки.
- `vert_align` - строка. Вертикальное выравнивание текста. Значения: T - по верхнему краю. B - по нижнему краю, C - по центру. Значение по умолчанию: C.
- `fill` - логическое значение. Указывает заполнять ли ячейку текущим цветом для заполнения. См. `setColor4Filling`. Значение по умолчанию - false.
- `link` - строка. Ссылка, например, URL или внутренняя ссылка. В текущей версии не поддерживается.

- `clipping` - логическое значение. Указывает обрезать ли текст по границам ячейки. Значение по умолчанию - false.
- `ln` - целое число. Позиция курсора после отрисовки ячейки. Допустимые значения: 0 - рядом с ячейкой, 1 - новая строка, 2 - под ячейкой. Значение по умолчанию - 0.

Пример:

```
{
  "name":"PrintCell",
  "params":{
     "w":120,
     "h":7,
     "txt":"Персональные данные",
     "border":"0",
     "align":"L",
     "vert_align":"C",
     "fill":true,
     "clipping":false,
     "ln":2
   }
}
```

PrintMultiLineCell

Выводит многострочную ячейку

Параметры

- `w` - число с плавающей точкой. Ширина прямоугольника.
- `h` - число с плавающей точкой. Высота прямоугольника ячейки.
- `txt` - текст
- `border` - строка. Задает видимые границы ячейки. Допустимые значения: 0 - нет границ, 1 - внешняя граница прямоугольника, L - левая, T - верхняя, R - правая, B - нижняя границы. Допускается комбинация значений L,T,R и B.
- `align` - строка. Горизонтальное выравнивание текста. Значения L - по левому краю. R - по правому краю, C - по центру, J - равномерно по ширине ячейки.
- `vert_align` - строка. Вертикальное выравнивание текста. Значения: T - по верхнему краю. B - по нижнему краю, C - по центру. Значение по умолчанию: C.
- `fill` - логическое значение. Указывает заполнять ли ячейку текущим цветом для заполнения. См. `setColor4Filling`. Значение по умолчанию - false.
- `maxline` - массив числе с плавающей точкой. Элемент массива задает максимальное количество строк текста в многострочной ячейке.
- `link` - строка. Ссылка, например, URL или внутренняя ссылка. В текущей версии не поддерживается.
- `clipping` - логическое значение. Указывает обрезать ли текст по границам ячейки. Значение по умолчанию - false.
- `indent` - число с плавающей точкой. Отступ для первой строки текста. Значение по умолчанию - 0.
- `ln` - целое число. Позиция курсора после отрисовки ячейки. Допустимые значения: 0 - рядом с ячейкой, 1 - новая строка, 2 - под ячейкой. Значение по умолчанию - 0.

Пример:

```
{
  "name":"PrintMultiLineCell",
  "params":{
     "h":5,
     "w":95,
     "txt":"Сумма средств пенсионных накоплений, поступивших в фонд (с учетом␣
↪инвестиционного дохода, полученного от инвестирования средств пенсионных␣
↪накоплений) при вступлении договора об обязательном пенсионном страховании в силу",
     "border":"LTR",
     "align":"L",
     "vert_align":"T",
     "fill":true,
     "indent":0,
     "clipping":false,
     "ln":0
   }
}
```

Row_Print

Выводит строку в PDF документе. Строка состоит из многострочных ячеек(см. `PrintMultiLineCell`). Высота строки определяется ячейкой с наибольшей высотой.

Параметры

- `data` - массив строк. Элемент массива задает текст ячейки в строке.
- `input_data_tag` - альтернатива data, имя массива строк-значений ячеек во входных данных запроса.
- `width` - массив чисел с плавающей точкой. Элемент массива задает ширину ячейки.
- `border` - массив строк. Элемент массива описывает видимые границы ячейки. См. описание команды `PrintMultiLineCell`. По умолчанию все границы видимы.
- `maxline` - массив чисел с плавающей точкой. Элемент массива задает максимальное количество строк текста в многострочной ячейке. По умолчанию ограничения нет.
- `align` - массив строк. Элемент массива задает горизонтальное выравнивание в ячейке.
- `vert_align` - массив строк. Элемент массива задает вертикальное выравнивание в ячейке.
- `font` - массив объектов описателей шрифта. Элемент массива задает параметры шрифта в ячейке. Набор полей каждого описателя аналогичен параметрам команды `SetPrintFont`. Значение по умолчанию - null, что означает использования настроек шрифта, установленного на предыдущем вызове `SetPrintFont`.
- `h` - число с плавающей точкой. Высота ячейки. Значение по умолчанию - 5 единиц.
- `fill` - логическое значение. Указывает заполнять ли ячейку текущим цветом для заполнения. См. setColor4Filling. Значение по умолчанию - false.
- `min_height` число с плавающей точкой. Минимальная высота строки. Ноль означает, что параметр не используется. Значение по умолчанию - 0.
- `clipping` - логическое значение. Указывает обрезать ли текст по границам ячейки. Значение по умолчанию - false.

Пример:

```
{
 "name":"Row_Print",
 "params":{
    "h":4.5,
    "min_height": 27,
    "width":[85, 95],
    "data":[
       "1.4. Средства (часть средств) материнского (семейного) капитала (за вычетом
↪части средств материнского (семейного) капитала, возвращенных в Пенсионный фонд
↪Российской Федерации в случае отказа застрахованного лица от направления их на
↪формирование накопительной пенсии и выбора другого направления использования,
↪включая доход полученный от их инвестирования))",
       "0,00"
      ],
    "align":["L", "L"],
    "vert_align":["T", "T"],
    "border":["0", "0"],
    "font": [
     {
      "size":6,
      "font_id":"Arial-Bold"
     },
     {
     "size":6,
     "font_id":"Arial"
     }
   ],
    "fill":true,
    "clipping":false
 }
}
```

PrintText

Выводит текст по заданным координатам.

Параметры

- `x` - число с плавающей точкой. X координата начала текста
- `y`- число с плавающей точкой. Y координата начала текста
- `text` - текст

Пример:

```
{
  "name":"PrintText",
  "params":{
    "x":10,
    "y":40,
    "text":"Список документов:"
   }
}
```

underline

Подчеркивает текст, выведенный с помощью команды `PrintText`.

Параметры аналогичны `PrintText`, значения параметров должны совпадать.

```
{
  "name":"underline",
  "params":{
     "x":10,
     "y":40,
     "text":"Список документов:"
   }
}
```

LineBreak

Разрыв строки. Переносит курсор на начало следующей строки.

Параметры

- `h` - число с плавающей точкой. Отступ от текущей координаты Y курсора.

Line

Рисует прямую линию между двумя точками на странице.

Параметры

- `x1` - число с плавающей точкой. X координата начала линии.
- `y1` - число с плавающей точкой. Y координата начала линии.
- `x2` - число с плавающей точкой. X координата конца линии.
- `y2` - число с плавающей точкой. Y координата конца линии.
- `color` строка в формате «#rrggbb», где rgb - шестнадцатеричные компоненты RGB цвета в диапазоне от 0 до ff. Пример красного цвета: «color»:“#ff0000”. По умолчанию используется цвет, предварительно заданный командой `setColor4Drawing`.
- `width` - число с плавающей точкой. Толщина линии. По умолчанию используется значение, предварительно заданное командой `setLineWidth`.

Пример:

```
{
  "name":"Line",
  "params":{
       "x1":100,
       "y1":230,
       "x2":150,
       "y2":230,
       "color":"#000000",
       "width":0.2
    }
}
```

Circle

Рисует окружность на странице.

Параметры

- `x` - число с плавающей точкой. X координата центра окружности.
- `y` - число с плавающей точкой. Y координата центра окружности.
- `r` - число с плавающей точкой. Радиус окружности.
- `draw` - логическое значение. Управляет отрисовкой контура окружности. Значение по умолчанию - true.
- `fill` - логическое значение. Управляет заливкой цветом внутренней области окружности. Значение по умолчанию - false.
- `draw_color` цвет контура окружности. Строка в формате «#rrggbb», где rgb - шестнадцатеричные компоненты RGB цвета в диапазоне от 0 до ff. Пример красного цвета: «color»:“#ff0000”. По умолчанию используется цвет, предварительно заданный командой `setColor4Drawing`.
- `fill_color` цвет контура окружности. строка в формате «#rrggbb», где rgb - шестнадцатеричные компоненты RGB цвета в диапазоне от 0 до ff. Пример красного цвета: «color»:“#ff0000”. По умолчанию используется цвет, предварительно заданный командой `setColor4Filling`.
- `linewidth` - число с плавающей точкой. Толщина контура окружности. По умолчанию используется значение, предварительно заданное командой `setLineWidth`.

Пример:

```
{
  "name": "Circle",
  "params":
      {
        "x": 135,
        "y": 250,
        "r": 15,
        "draw": false,
        "fill": false,
        "draw_color": "#ff0000",
        "fill_color": "#ff0000",
        "linewidth": 0.1
    }
 }
```

putImage

Параметры

- `name` - строка. Имя изображения или ID изображения. Если имя совпадает с указанным на предыдущем вызове, используется изображение, указанное в том же вызове.
- `data` - строка. Закодированный в base64 файл изображения. Если имя `name` совпадает с указанным на предыдущем вызове, данный параметр не указывается.
- `x` - число с плавающей точкой. X координата изображения.
- `y` - число с плавающей точкой. Y координата изображения.
- `w` - число с плавающей точкой. Ширина изображения в единицах, заданных командой `setCoordinateMode`. Если параметр не указан, изображение выводится в натуральную ширину,

которая высчитывается на основе плотности пикселей PDF-документа (72 точки на дюйм) и ширины изображения согласно файлу изображения. *Пример*: изображение 72x72 точки будет иметь размер 1 дюйм на 1 дюйм в PDF-документе.

- `h` - число с плавающей точкой. Высота изображения. Единицы измерения: аналогично параметру w.
- `link` - строка. URL или ID внутренней ссылки. В текущей версии не поддерживается.

Пример:

```
{
  "name":"putImage",
  "params":{
      "name":"img",
      "data":"base64 encoded image",
      "x":100,
      "y":205,
      "w":50,
      "h":37.5
    }
}
```

setPageTemplate

Параметры

- `template` - объект в формате

```
{
 "uri": "local",
 "id": "template_example"
}
```

где идентификатор шаблона - это путь к документу pdf относительно директории `templates/pdf` без расширения.

- `page` - целое число. Номер страницы шаблона для вывода. Диапазон от единицы до количества страниц в шаблоне.

Пример команды для многостраничного шаблона:

```
{
 "name":"setPageTemplate",
 "params":{
    "template":{
       "id":"example_multipage_template",
       "uri":"local"
     },
    "page":1
    }
}
```

Примечание: Происходит дублирование ресурсов шрифтов в отчете при использовании команды `setPageTemplate`, когда шрифт полностью совпадает с загруженным командой `SetPrintFont`. Может увеличить размер файла отчета.

checkPageBreak

Если добавление указанной высоты к текущей координате Y приведет к переполнению страницы (то есть выходу за пределы нижнего отступа страницы), команда добавляет новую страницу и возвращает true (возвращаемое значение имеет значение только в команде `if`)

Параметры

- `h` - число с плавающей точкой. Высота для проверки на переполнение страницы.
- `newpage` - логическое значение. Указывает создавать ли новую страницу в документе при переполнении. Значение по умолчанию - true.

if

Команда позволяет выполнить набор подкоманд при истинности условия непосредственно в момент выполнения команды(в отличие от команды `register_auto_new_page_commands`).

Поддерживаемый набор команд для использования в качестве условия: `checkPageBreak`.

Параметры

- `condition` - объект, описывающий команду, возвращающую логическое значение.
- `commands` - массив команд для выполнения в случае, если команда указанная в параметре `condition` вернула true.

Пример:

```
{
 "name": "if",
 "params": {
     "condition": {
         "name":"checkPageBreak",
         "params": {
             "h": 4.5,
             "newpage": true
         }
     },
     "commands": [
         {
             "name":"setPageTemplate",
             "params":{
                "template":{
                   "id":"example_template",
                   "uri":"local"
                }
             }
         },
         {
             "name":"setCurrentXY",
             "params":{
                 "x":10,
                 "y":30
             }
         }
     ]
  }
}
```

Типичный вариант использования: перед попыткой вывода элемента или набора элементов в нижнюю часть страницы, когда есть риск выхода за нижний отступ страницы. Значение параметра `h` для проверки обычно берётся из размера элемента, который нужно вывести на экран.

Примечание: Команда `if` работает непосредственно по месту вызова, с текущей координатой Y. То есть, если ее вызвать в момент, когда на странице есть вертикальное пространство высотой `h`, то список подкоманд исполнен не будет.

Примечание: Не рекомендуется использовать команду `if` для вывода строк таблиц. Она может быть полезна для вывода уникального элемента, когда отрисовка происходит ближе к нижней части страницы. В случае таблиц лучше использовать `register_auto_new_page_commands`.

---

register_auto_new_page_commands

Позволяет зарегистрировать набор подкоманд для выполнения в случае, когда исполнение команд типа `Row_Print` приводит к автоматическому созданию новой страницы документа. Может быть полезно для рендера заголовка таблицы. Действует с момента вызова. Отмена возможна вызовом этой же команды с пустым списком подкоманд.

Параметры

- `commands` - массив команд для выполнения.

Пример:

```
{
  "name": "register_auto_new_page_commands",
  "params": {
      "commands": [
          {
            "name":"setPageTemplate",
            "params":{
                 "template":{
                    "id":"example_template",
                    "uri":"local"
                 }
              }
          },
          {
            "name":"setCurrentXY",
            "params":{
                 "x":10,
                 "y":30
           }
        }
      ]
   }
}
```

Отмена команды:

```
{
  "name": "register_auto_new_page_commands",
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  "params": {
      "commands": []
  }
}
```

7.2.5 Координаты

Команды, принимающие координаты и размеры в качестве параметров, трактуют значения на основании вызова команды `setCoordinateMode`. По умолчанию используются миллиметры.

Все подобные параметры могут быть заданы в числовом или текстовом варианте.

В текстовом виде поддерживаются выражения и специальные теги `PDF:currentX/PDF:currentY`

Пример:

```
{
    "name": "Line",
    "params":
    {
        "x1": "[PDF:currentX]",
        "y1": "[PDF:currentY]",
        "x2": "[PDF:currentX] + 50",
        "y2": "[PDF:currentY]",
        "color": "#ff0000",
        "width": 0.1
    }
}
```

7.2.6 Теги

Команды, которые работают с текстом (`PrintText`, `PrintCell` и др.), могут содержать теги, которые будут заменены входными данными из запроса.

Входные данные содержатся в теле запроса с объекте "`input-data`": `{...}` (см. пункт "Входные данные" подраздела "Структура запроса").

Формат тегов

Тег задается в квадратных скобках. Например, [ORGANIZATION].

Пример:

```
{
  "name":"PrintMultiLineCell",
  "params":{
     "h":4,
     "w":77,
     "ln":2,
     "txt":"[ORGANIZATION], именуемый в дальнейшем «Фонд», в лице генерального␣
↪директора Иванова Ивана Ивановича, действующего на основании доверенности №␣
↪11142 от 02.04.2018 г., с одной стороны, и [CLINAME], именуемый в дальнейшем␣
↪«Участник», с другой стороны, заключили настоящее Соглашение о нижеследующем:",
     "fill":false,
```

(continues on next page)

(продолжение с предыдущей страницы)

```
      "align":"J",
      "border":"0",
      "indent":15,
      "clipping":false
   }
}
```

В запросе скобки не указываются (см. пример из пункта “Входные данные” подраздела “Структура запроса”).

Встроенные теги

Некоторые теги встроены в генератор PDF.

1. `PDF:currPageNum`, номер текущей страницы.
2. `PDF:currentX/PDF:currentY`, X и Y координаты курсора.
3. `PDF:savedX/PDF:savedY`, доступно после вызова команд `saveX/saveY`.
4. `[SAVED:ключ]`, доступно после команды `saveCoordinate`.

Пример:

```
{
   "name":"Line",
   "params":{
      "x1":"[PDF:savedX]",
      "x2":"[SAVED:endLineX]",
      "y1":"[PDF:currentY]",
      "y2":"[PDF:currentY]",
      "color":"#E8E8E8",
      "width":0.8
   }
},
{
   "name":"PrintText",
   "params":{
      "x":"[SAVED:pageNumX]",
      "y":"[SAVED:pageNumY]",
      "text":"Номер страницы: [PDF:currPageNum]"
   }
}
```

7.2.7 Работа со штрихкодом

Возможен вывод информации в виде штрихкода посредством использования шрифтов семейства Libre Barcode. Шрифты доступны в репозитории librebarcode. Каждый шрифт представляет собой определенный формат штрихкода. Поддерживаются следующие форматы:

- Code-128
- Code-39
- EAN13

Шрифт может включать оригинальный текст непосредственно под штрихкодом (шрифты с суффиксом Text).

Рассмотрим применение на примере формата Code-128.

Code-128

Шрифты для помещения в папку `assets/fonts` сервиса:

- LibreBarcode128Text-Regular.ttf
- LibreBarcode128-Regular.ttf

Как использовать

В генераторе команд выбрать нужный шрифт, например:

```
{
    "name": "SetPrintFont",
    "params":
    {
        "font_id": "LibreBarcode128Text-Regular",
        "size": 30
    }
}
```

Тестовый запрос

```
curl --request POST --data-binary "@templates/examples/pdf/embedded-report-generator/
↪request_barcode_code-128.json" http://localhost:8087/pdf_report_json
```

Объект описателя генератора PDF `report-generator` содержит:

- `uri` - строка. Задает положение источника команд генератора в зависимости от того, как трактуется параметр `id`. Поддерживаемые значения: `local` и `embedded`.
- `id` - строка. ID генератора.

Возможны два варианта использования генератора PDF:

- генератор как часть запроса (все команды генератора размещаются внутри объекта `embedded-report-generator` тела запроса);
- генератор как часть сервиса (все команды генератора размещаются внутри файла JSON в специальной папке `pdf_report_gen` на сервере).

Соответственно:

- При значении `uri`, равному `embedded`, `id` используется только для диагностических сообщений и идентификации запроса. Генератор должен находится в корневом объекте `embedded-report-generator` запроса. См. пункт «Генераторы как часть запроса» ниже.
- При значении `uri`, равному `local`, `id` - это имя файла в директории сервиса `pdf_report_gen`. См. пункт «Генераторы как часть сервиса» ниже.

Генератор как часть запроса

Опционально генератор может быть встроен в запрос.

В этом случае все команды генератора указываются в объекте `embedded-report-generator`.

Пример:

```
{
  "report-generator":{
     "id":"example",
     "uri":"embedded"
  },
  "embedded-report-generator":{
     "commands":[
        {
           "name":"NewPage"
        },
        {
           "name":"setCoordinateMode",
           "params":{
              "mode":"millimeters"
           }
        },
        {
           "name":"setCellMargin",
           "params":{
              "margin":1
           }
        }
     ]
  },
  "options":{
     "enable-debug-report-save":false
  }
}
```

Генератор как часть сервиса

Цель: не передавать данные команд внутри запросов, если они не меняются от запроса к запросу.

В этом случае все команды генератора указываются в файле формата JSON, который должен находиться в директории `pdf_report_gen`, расположенной в директории приложения сервиса.

Формат генератора внутри файла в директории `pdf_report_gen`:

```
{
    "report-generator":
    {
         "commands": [...]
    }
}
```

Пример генератора в теле запроса:

```
"report-generator":
   {
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        "uri": "local",
        "id": "example"
    }
```

, где `id` - это имя файла в директории сервиса `pdf_report_gen`.

Пример простого запроса для генератора внутри файла example.json в директории `pdf_report_gen`:

```
{
    "report-generator":
    {
        "uri": "local",
        "id": "example"
    },
    "options":
    {
        "enable-debug-report-save": true
    }
}
```

Входные данные

Объект входных данных `input-data` содержит входные, которые подставляются в теги.

Примечание: Команды генератора PDF, которые работают с текстом, могут содержать теги, которые будут заменены входными данными из запроса. Подробнее об использовании тегов можно прочитать в разделе “Теги” ниже.

Пример:

```
"input-data":
    {
        "ORGANIZATION": "Акционерное общество Негосударственный пенсионный фонд␣
↪«XXXXX»",
        "CLINAME": "Петров Петр Петрович"
    }
```

Опционально: генератор PDF

Объект генератора PDF `embedded-report-generator` содержит команды генератора. Используется опционально при значении `uri`, равному `local`.

Подробнее описан в пункте “Генераторы как часть запроса”.

Опции запроса

Объект опций запроса `options` имеет поля:

- `enable-debug-report-save` - логическое значение. Указывает создавать ли копию отчета документа и копию запроса (имя файла - идентификатор шаблона) в локальной директории `reports_debug` сервиса. Значение по умолчанию - `false`.
- `enable-binary-output` - логическое значение. Указывает, что сервис будет выдавать результат в виде бинарных данных, без кодирования в base64. Значение по умолчанию - `false`.
- `enable-debug-pdf-log` - логическое значение. Включает расширенный диагностический лог создания PDF. Значение по умолчанию - `false`.

Пример:

```
"options":
    {
        "enable-debug-merged-doc-save": true,
        "enable-binary-output": false,
        "enable-debug-pdf-log": true
    }
```

7.2.8 Пример вызова сервиса

```
curl --request POST --data-binary "@templates/examples/pdf/embedded-report-generator/
↪request_example.json" http://localhost:8087/pdf_report_json
```

Структура ответа

Ответ сервиса содержит объект в формате JSON, который включает:

1. Описание ошибки (код ошибки, сообщение об ошибке). В случае успешного ответа сервиса возвращается значение `null`.
2. Результат (закодированный в base64 сформированный PDF-файл документа).

Формат ответа

```
{
   error: {
       code
       message
   }
   result: "[base64 encoded pdf file]"
}
```

Пример ответа


```
{
   "error": null,
   "result": "UEsDBBQACAAIAPdKo...JwAAAAA="
}
```


Использование шаблонов страниц

Для формирования документа с помощью генератора PDF можно использовать готовые шаблоны страниц в формате PDF.

Для задания шаблона конкретной страницы существует специальная команда `setPageTemplate`. Подробнее про команду можно прочитать в подразделе “Команды генератора”.

Примечание: Поддерживаются многостраничные шаблоны.

Для расположения текста и других элементов на странице шаблона необходимо задавать координаты. Подробнее про задание координат элементов документа можно прочитать в подразделе “Координаты”.

Шаблоны страниц в формате PDF хранятся локально в директории `templates/pdf`, расположенной в директории приложения сервиса.

Служебные директории

Скрипты генерации PDF хранятся в директории `pdf_report_gen`, расположенной в директории приложения сервиса.

Шаблоны страниц в формате PDF хранятся локально в директории `templates/pdf`, расположенной в директории приложения сервиса.

Примеры запросов находятся в директории `request_examples/pdf`.

Шрифты для использования в генераторе, находятся в директории `assets/fonts`, расположенной в директории приложения сервиса. Загружаются при старте сервиса.

Примечание: При добавлении новых файлов шрифтов необходимо перезапустить сервис.

Примеры запросов и шаблонов

Примеры запросов находятся в директории `examples/pdf/embedded-report-generator` и `examples/pdf/report-generator` в *архиве примеров*.

Примеры шаблонов находятся в директории `examples/pdf`в *архиве примеров*.

7.2.9 Команды генератора

PDF-документ генерируется при помощи специальных команд генератора.

Команды располагаются внутри объекта `embedded-report-generator`, если генератор встраивается в запрос (см. пункт “Генератор как часть запроса” выше).

Команды также могут располагаться в файле формата JSON, который находится в директории `pdf_report_gen`, расположенной в директории приложения сервиса (см. пункт “Генератор как часть сервиса” выше).

Команды описываются внутри массива `commands`.

Пример:

```
"commands": [
        {
           "name":"NewPage"
        },
        {
           "name":"setCoordinateMode",
           "params":{
              "mode":"millimeters"
           }
        },
        {
           "name":"setCellMargin",
           "params":{
              "margin":1
           }
        }
     ]
```

Массив `commands` может включать команды, описанные ниже.

NewPage

Создает новую страницу в документе.

Параметры

- `orientation` - строка. Поддерживаемые значения:
  - «L» - альбомная ориентация
  - «P» - портретная ориентация

Значение по умолчанию - “P”.

Пример:

```
{
  "name":"NewPage",
  "params":{
      "orientation":"L"
   }
}
```

setCoordinateMode

Задает трактовку координат и размеров, указываемых в последующих командах. Режим по умолчанию «в миллиметрах”.

Параметры

- `mode` Поддерживаемые значения: «millimeters», «percents», «normalized».

setPageMargins

Устанавливает отступы от границ страницы.

Параметры

`left`, `right`, `top`, `bottom` - числа с плавающей точкой. Отступы от левого, правого, верхнего, нижнего краев страницы. При достижении нижнего отступа в командах типа `Row_Print` автоматически создается новая страница. Команда заменяет plpdf функции `setAllMargin` и `SetAutoNewPage`.

Пример:

```
{
  "name":"setPageMargins",
  "params":{
     "left":20,
     "right":20,
     "top":30,
     "bottom":15
    }
 }
```

setCurrentX

Устанавливает X координату курсора.

Параметры

- `value` - число с плавающей точкой. Координата X.

setCurrentY

Устанавливает Y координату курсора.

Параметры

- `value` - число с плавающей точкой. Координата Y.

setCurrentXY

Устанавливает координаты курсора.

Параметры

- `x` - число с плавающей точкой. Координата X.
- `y` - число с плавающей точкой. Координата Y.

saveX

Сохраняет координату X для последующего использования в параметрах координат команд через тег `[PDF:savedX]`.

Параметры

- `value` - опциональный параметр, по умолчанию равен текущей координате X. Может быть числом с плавающей точкой или строкой-выражением со ссылками на встроенные теги. Пример выражения: `"[PDF:currentX] - 50"`

saveY

Сохраняет координату Y для последующего использования в параметрах координат команд через тег `[PDF:savedY]`. Внимание: при последующем создании новой страницы значение может оказаться нерелевантным.

Параметры

- `value` - опциональный параметр, по умолчанию равен текущей координате Y. Может быть числом с плавающей точкой или строкой-выражением со ссылками на встроенные теги. Пример выражения: `"[PDF:currentY] - 50"`

saveCoordinate

Сохраняет произвольное значение координат для последующего использования в параметрах координат команд через тег `[SAVED:ключ]`. Дополняет команды `saveX/saveY` в случаях, когда нужен доступ к нескольким значениям.

Параметры

- `key` - строка. Произвольный ключ для использования в теге [SAVED:ключ]
- `value` - может быть числом с плавающей точкой или строкой-выражением со ссылками на встроенные теги. Пример выражения: `"[PDF:currentY] - 50"`

Пример:

```
{
  "name":"saveCoordinate",
  "params":{
     "key":"x1",
     "value":10
  }
}
```

setRotate

Задает поворот всех последующих объектов.

Параметры

- `angle` - число с плавающей точкой. Угол вращения в градусах.
- `x`- число с плавающей точкой. X координата точки вращения. Если null, то X координата курсора
- `y` - число с плавающей точкой. Y координата точки вращения. Если null, то Y координата курсора

Пример:

```
{
  "name":"setRotate",
  "params":{
     "angle":45,
     "x":105,
     "y":130,
   }
}
```

Отмена команды:

```
{
  "name":"setRotate",
  "params":{
     "angle":0
  }
}
```

startOpacity

Указывает степень прозрачности для всех последующих объектов. Действие отменяется командой `endOpacity`.

Параметры

- `val` - число с плавающей точкой. Допустимый диапазон от 0.0 до 1.0, где ноль - полная прозрачность.

endOpacity

Отменяет действие команды `startOpacity`.

Без параметров.

SetPrintFont

Задает шрифт и размер шрифта для последующих текстовых команд.

Параметры

- `font_id` Поддерживаемые значения: имена файлов шрифтов из директории `assets/fonts` без расширения. Шрифты загружаются при старте сервиса.
- `size` Размер шрифта в точках в пользовательском пространстве документа pdf. Никаких отличий от PLPDF и других текстовых редакторов.

Примечание: Единственная поддерживаемая кодовая страница - cp1251.

setColor4Text

Задает цвет текста для последующих текстовых команд.

Варианты входных параметров

- `r`, `g`, `b` целочисленные десятичные компоненты RGB цвета в диапазоне от 0 до 255. Пример красного цвета: “r”:255, “g”:0, “b”:0.
- `color` строка в формате «#rrggbb», где rgb - шестнадцатеричные компоненты RGB цвета в диапазоне от 0 до ff. Пример красного цвета: «color»:“#ff0000”.

setColor4Drawing

Задает цвет границ для объектов, выводимых после исполнения данной команды. Примеры объектов: ячейки таблиц, окружности, линии, прямоугольники и т.д.

Параметры: аналогично команде `setColor4Text`.

setColor4Filling

Задает цвет заливки внутренних областей для объектов, выводимых после исполнения данной команды. Примеры объектов: ячейки таблиц, окружности, прямоугольники и т.д.

Параметры: аналогично команде `setColor4Text`.

setLineWidth

Устанавливает толщину линий, выводимых после исполнения данной команды. Примеры объектов: границы ячеек таблиц, линии, прямоугольники и т.д.

Параметры

- `width` - число с плавающей точкой. Толщина линии.

setCellMargin

Устанавливает отступы контента табличной ячейки от левой, верхней, правой и нижней границ.

Параметры

- `margin` - число с плавающей точкой.

setCellBottomMargin

Устанавливает отступ контента табличной ячейки от нижней границы.

Параметры

- `margin` - число с плавающей точкой.

setCellLeftMargin

Устанавливает отступ контента табличной ячейки от левой границы.

Параметры

- `margin` - число с плавающей точкой.

setCellTopMargin

Устанавливает отступ контента табличной ячейки от верхней границы.

Параметры

- `margin` - число с плавающей точкой.

setCellRightMargin

Устанавливает отступ контента табличной ячейки от правой границы.

Параметры

- `margin` - число с плавающей точкой.

PrintCell

Выводит прямоугольную ячейку с текстом внутри. Можно указать цвет и наличие границ.

Параметры

- `w` - число с плавающей точкой. Ширина прямоугольника ячейки.
- `h` - число с плавающей точкой. Высота прямоугольника ячейки.
- `txt` - текст
- `border` - строка. Задает видимые границы ячейки. Допустимые значения: 0 - нет границ, 1 - внешняя граница прямоугольника, L - левая, T - верхняя, R - правая, B - нижняя границы. Допускается комбинация значений L,T,R и B.
- `align` - строка. Горизонтальное выравнивание текста. Значения: L - по левому краю. R - по правому краю, C - по центру, J - равномерно по ширине ячейки.
- `vert_align` - строка. Вертикальное выравнивание текста. Значения: T - по верхнему краю. B - по нижнему краю, C - по центру. Значение по умолчанию: C.
- `fill` - логическое значение. Указывает заполнять ли ячейку текущим цветом для заполнения. См. `setColor4Filling`. Значение по умолчанию - false.
- `link` - строка. Ссылка, например, URL или внутренняя ссылка. В текущей версии не поддерживается.

- `clipping` - логическое значение. Указывает обрезать ли текст по границам ячейки. Значение по умолчанию - false.
- `ln` - целое число. Позиция курсора после отрисовки ячейки. Допустимые значения: 0 - рядом с ячейкой, 1 - новая строка, 2 - под ячейкой. Значение по умолчанию - 0.

Пример:

```
{
  "name":"PrintCell",
  "params":{
     "w":120,
     "h":7,
     "txt":"Персональные данные",
     "border":"0",
     "align":"L",
     "vert_align":"C",
     "fill":true,
     "clipping":false,
     "ln":2
   }
}
```

PrintMultiLineCell

Выводит многострочную ячейку

Параметры

- `w` - число с плавающей точкой. Ширина прямоугольника.
- `h` - число с плавающей точкой. Высота прямоугольника ячейки.
- `txt` - текст
- `border` - строка. Задает видимые границы ячейки. Допустимые значения: 0 - нет границ, 1 - внешняя граница прямоугольника, L - левая, T - верхняя, R - правая, B - нижняя границы. Допускается комбинация значений L,T,R и B.
- `align` - строка. Горизонтальное выравнивание текста. Значения L - по левому краю. R - по правому краю, C - по центру, J - равномерно по ширине ячейки.
- `vert_align` - строка. Вертикальное выравнивание текста. Значения: T - по верхнему краю. B - по нижнему краю, C - по центру. Значение по умолчанию: C.
- `fill` - логическое значение. Указывает заполнять ли ячейку текущим цветом для заполнения. См. `setColor4Filling`. Значение по умолчанию - false.
- `maxline` - массив числе с плавающей точкой. Элемент массива задает максимальное количество строк текста в многострочной ячейке.
- `link` - строка. Ссылка, например, URL или внутренняя ссылка. В текущей версии не поддерживается.
- `clipping` - логическое значение. Указывает обрезать ли текст по границам ячейки. Значение по умолчанию - false.
- `indent` - число с плавающей точкой. Отступ для первой строки текста. Значение по умолчанию - 0.
- `ln` - целое число. Позиция курсора после отрисовки ячейки. Допустимые значения: 0 - рядом с ячейкой, 1 - новая строка, 2 - под ячейкой. Значение по умолчанию - 0.

Пример:

```
{
  "name":"PrintMultiLineCell",
  "params":{
     "h":5,
     "w":95,
     "txt":"Сумма средств пенсионных накоплений, поступивших в фонд (с учетом␣
↪инвестиционного дохода, полученного от инвестирования средств пенсионных␣
↪накоплений) при вступлении договора об обязательном пенсионном страховании в силу",
     "border":"LTR",
     "align":"L",
     "vert_align":"T",
     "fill":true,
     "indent":0,
     "clipping":false,
     "ln":0
   }
}
```

Row_Print

Выводит строку в PDF документе. Строка состоит из многострочных ячеек(см. `PrintMultiLineCell`). Высота строки определяется ячейкой с наибольшей высотой.

Параметры

- `data` - массив строк. Элемент массива задает текст ячейки в строке.
- `input_data_tag` - альтернатива data, имя массива строк-значений ячеек во входных данных запроса.
- `width` - массив чисел с плавающей точкой. Элемент массива задает ширину ячейки.
- `border` - массив строк. Элемент массива описывает видимые границы ячейки. См. описание команды `PrintMultiLineCell`. По умолчанию все границы видимы.
- `maxline` - массив чисел с плавающей точкой. Элемент массива задает максимальное количество строк текста в многострочной ячейке. По умолчанию ограничения нет.
- `align` - массив строк. Элемент массива задает горизонтальное выравнивание в ячейке.
- `vert_align` - массив строк. Элемент массива задает вертикальное выравнивание в ячейке.
- `font` - массив объектов описателей шрифта. Элемент массива задает параметры шрифта в ячейке. Набор полей каждого описателя аналогичен параметрам команды `SetPrintFont`. Значение по умолчанию - null, что означает использования настроек шрифта, установленного на предыдущем вызове `SetPrintFont`.
- `h` - число с плавающей точкой. Высота ячейки. Значение по умолчанию - 5 единиц.
- `fill` - логическое значение. Указывает заполнять ли ячейку текущим цветом для заполнения. См. setColor4Filling. Значение по умолчанию - false.
- `min_height` число с плавающей точкой. Минимальная высота строки. Ноль означает, что параметр не используется. Значение по умолчанию - 0.
- `clipping` - логическое значение. Указывает обрезать ли текст по границам ячейки. Значение по умолчанию - false.

Пример:

```
{
 "name":"Row_Print",
 "params":{
    "h":4.5,
    "min_height": 27,
    "width":[85, 95],
    "data":[
       "1.4. Средства (часть средств) материнского (семейного) капитала (за вычетом
↪части средств материнского (семейного) капитала, возвращенных в Пенсионный фонд
↪Российской Федерации в случае отказа застрахованного лица от направления их на
↪формирование накопительной пенсии и выбора другого направления использования,
↪включая доход полученный от их инвестирования))",
       "0,00"
      ],
    "align":["L", "L"],
    "vert_align":["T", "T"],
    "border":["0", "0"],
    "font": [
     {
      "size":6,
      "font_id":"Arial-Bold"
     },
     {
     "size":6,
     "font_id":"Arial"
     }
   ],
    "fill":true,
    "clipping":false
 }
}
```

PrintText

Выводит текст по заданным координатам.

Параметры

- `x` - число с плавающей точкой. X координата начала текста
- `y`- число с плавающей точкой. Y координата начала текста
- `text` - текст

Пример:

```
{
  "name":"PrintText",
  "params":{
    "x":10,
    "y":40,
    "text":"Список документов:"
   }
}
```

underline

Подчеркивает текст, выведенный с помощью команды `PrintText`.

Параметры аналогичны `PrintText`, значения параметров должны совпадать.

```
{
  "name":"underline",
  "params":{
     "x":10,
     "y":40,
     "text":"Список документов:"
   }
}
```

LineBreak

Разрыв строки. Переносит курсор на начало следующей строки.

Параметры

- `h` - число с плавающей точкой. Отступ от текущей координаты Y курсора.

Line

Рисует прямую линию между двумя точками на странице.

Параметры

- `x1` - число с плавающей точкой. X координата начала линии.
- `y1` - число с плавающей точкой. Y координата начала линии.
- `x2` - число с плавающей точкой. X координата конца линии.
- `y2` - число с плавающей точкой. Y координата конца линии.
- `color` строка в формате «#rrggbb», где rgb - шестнадцатеричные компоненты RGB цвета в диапазоне от 0 до ff. Пример красного цвета: «color»:“#ff0000”. По умолчанию используется цвет, предварительно заданный командой `setColor4Drawing`.
- `width` - число с плавающей точкой. Толщина линии. По умолчанию используется значение, предварительно заданное командой `setLineWidth`.

Пример:

```
{
  "name":"Line",
  "params":{
       "x1":100,
       "y1":230,
       "x2":150,
       "y2":230,
       "color":"#000000",
       "width":0.2
    }
}
```

Circle

Рисует окружность на странице.

Параметры

- `x` - число с плавающей точкой. X координата центра окружности.
- `y` - число с плавающей точкой. Y координата центра окружности.
- `r` - число с плавающей точкой. Радиус окружности.
- `draw` - логическое значение. Управляет отрисовкой контура окружности. Значение по умолчанию - true.
- `fill` - логическое значение. Управляет заливкой цветом внутренней области окружности. Значение по умолчанию - false.
- `draw_color` цвет контура окружности. Строка в формате «#rrggbb», где rgb - шестнадцатеричные компоненты RGB цвета в диапазоне от 0 до ff. Пример красного цвета: «color»:“#ff0000”. По умолчанию используется цвет, предварительно заданный командой `setColor4Drawing`.
- `fill_color` цвет контура окружности. строка в формате «#rrggbb», где rgb - шестнадцатеричные компоненты RGB цвета в диапазоне от 0 до ff. Пример красного цвета: «color»:“#ff0000”. По умолчанию используется цвет, предварительно заданный командой `setColor4Filling`.
- `linewidth` - число с плавающей точкой. Толщина контура окружности. По умолчанию используется значение, предварительно заданное командой `setLineWidth`.

Пример:

```
{
  "name": "Circle",
  "params":
      {
        "x": 135,
        "y": 250,
        "r": 15,
        "draw": false,
        "fill": false,
        "draw_color": "#ff0000",
        "fill_color": "#ff0000",
        "linewidth": 0.1
    }
 }
```

putImage

Параметры

- `name` - строка. Имя изображения или ID изображения. Если имя совпадает с указанным на предыдущем вызове, используется изображение, указанное в том же вызове.
- `data` - строка. Закодированный в base64 файл изображения. Если имя `name` совпадает с указанным на предыдущем вызове, данный параметр не указывается.
- `x` - число с плавающей точкой. X координата изображения.
- `y` - число с плавающей точкой. Y координата изображения.
- `w` - число с плавающей точкой. Ширина изображения в единицах, заданных командой `setCoordinateMode`. Если параметр не указан, изображение выводится в натуральную ширину,

которая высчитывается на основе плотности пикселей PDF-документа (72 точки на дюйм) и ширины изображения согласно файлу изображения. *Пример*: изображение 72x72 точки будет иметь размер 1 дюйм на 1 дюйм в PDF-документе.

- `h` - число с плавающей точкой. Высота изображения. Единицы измерения: аналогично параметру w.
- `link` - строка. URL или ID внутренней ссылки. В текущей версии не поддерживается.

Пример:

```
{
  "name":"putImage",
  "params":{
      "name":"img",
      "data":"base64 encoded image",
      "x":100,
      "y":205,
      "w":50,
      "h":37.5
    }
}
```

setPageTemplate

Параметры

- `template` - объект в формате

```
{
 "uri": "local",
 "id": "template_example"
}
```

где идентификатор шаблона - это путь к документу pdf относительно директории `templates/pdf` без расширения.

- `page` - целое число. Номер страницы шаблона для вывода. Диапазон от единицы до количества страниц в шаблоне.

Пример команды для многостраничного шаблона:

```
{
 "name":"setPageTemplate",
 "params":{
    "template":{
       "id":"example_multipage_template",
       "uri":"local"
     },
    "page":1
    }
}
```

Примечание: Происходит дублирование ресурсов шрифтов в отчете при использовании команды `setPageTemplate`, когда шрифт полностью совпадает с загруженным командой `SetPrintFont`. Может увеличить размер файла отчета.

checkPageBreak

Если добавление указанной высоты к текущей координате Y приведет к переполнению страницы (то есть выходу за пределы нижнего отступа страницы), команда добавляет новую страницу и возвращает true (возвращаемое значение имеет значение только в команде `if`)

Параметры

- `h` - число с плавающей точкой. Высота для проверки на переполнение страницы.
- `newpage` - логическое значение. Указывает создавать ли новую страницу в документе при переполнении. Значение по умолчанию - true.

if

Команда позволяет выполнить набор подкоманд при истинности условия непосредственно в момент выполнения команды(в отличие от команды `register_auto_new_page_commands`).

Поддерживаемый набор команд для использования в качестве условия: `checkPageBreak`.

Параметры

- `condition` - объект, описывающий команду, возвращающую логическое значение.
- `commands` - массив команд для выполнения в случае, если команда указанная в параметре `condition` вернула true.

Пример:

```
{
 "name": "if",
 "params": {
     "condition": {
         "name":"checkPageBreak",
         "params": {
             "h": 4.5,
             "newpage": true
         }
     },
     "commands": [
         {
             "name":"setPageTemplate",
             "params":{
                "template":{
                   "id":"example_template",
                   "uri":"local"
                }
             }
         },
         {
             "name":"setCurrentXY",
             "params":{
                 "x":10,
                 "y":30
             }
         }
     ]
  }
}
```

Типичный вариант использования: перед попыткой вывода элемента или набора элементов в нижнюю часть страницы, когда есть риск выхода за нижний отступ страницы. Значение параметра `h` для проверки обычно берётся из размера элемента, который нужно вывести на экран.

Примечание: Команда `if` работает непосредственно по месту вызова, с текущей координатой Y. То есть, если ее вызвать в момент, когда на странице есть вертикальное пространство высотой `h`, то список подкоманд исполнен не будет.

Примечание: Не рекомендуется использовать команду `if` для вывода строк таблиц. Она может быть полезна для вывода уникального элемента, когда отрисовка происходит ближе к нижней части страницы. В случае таблиц лучше использовать `register_auto_new_page_commands`.

---

register_auto_new_page_commands

Позволяет зарегистрировать набор подкоманд для выполнения в случае, когда исполнение команд типа `Row_Print` приводит к автоматическому созданию новой страницы документа. Может быть полезно для рендера заголовка таблицы. Действует с момента вызова. Отмена возможна вызовом этой же команды с пустым списком подкоманд.

Параметры

- `commands` - массив команд для выполнения.

Пример:

```
{
  "name": "register_auto_new_page_commands",
  "params": {
      "commands": [
          {
            "name":"setPageTemplate",
            "params":{
                 "template":{
                    "id":"example_template",
                    "uri":"local"
                 }
              }
          },
          {
            "name":"setCurrentXY",
            "params":{
                 "x":10,
                 "y":30
           }
        }
      ]
   }
}
```

Отмена команды:

```
{
  "name": "register_auto_new_page_commands",
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  "params": {
      "commands": []
  }
}
```

7.2.10 Координаты

Команды, принимающие координаты и размеры в качестве параметров, трактуют значения на основании вызова команды `setCoordinateMode`. По умолчанию используются миллиметры.

Все подобные параметры могут быть заданы в числовом или текстовом варианте.

В текстовом виде поддерживаются выражения и специальные теги `PDF:currentX/PDF:currentY`

Пример:

```
{
    "name": "Line",
    "params":
    {
        "x1": "[PDF:currentX]",
        "y1": "[PDF:currentY]",
        "x2": "[PDF:currentX] + 50",
        "y2": "[PDF:currentY]",
        "color": "#ff0000",
        "width": 0.1
    }
}
```

7.2.11 Теги

Команды, которые работают с текстом (`PrintText`, `PrintCell` и др.), могут содержать теги, которые будут заменены входными данными из запроса.

Входные данные содержатся в теле запроса с объекте "`input-data`": `{...}` (см. пункт "Входные данные" подраздела "Структура запроса").

Формат тегов

Тег задается в квадратных скобках. Например, [ORGANIZATION].

Пример:

```
{
  "name":"PrintMultiLineCell",
  "params":{
     "h":4,
     "w":77,
     "ln":2,
     "txt":"[ORGANIZATION], именуемый в дальнейшем «Фонд», в лице генерального␣
↪директора Иванова Ивана Ивановича, действующего на основании доверенности №␣
↪11142 от 02.04.2018 г., с одной стороны, и [CLINAME], именуемый в дальнейшем␣
↪«Участник», с другой стороны, заключили настоящее Соглашение о нижеследующем:",
     "fill":false,
```

(continues on next page)

(продолжение с предыдущей страницы)

```
      "align":"J",
      "border":"0",
      "indent":15,
      "clipping":false
   }
}
```

В запросе скобки не указываются (см. пример из пункта “Входные данные” подраздела “Структура запроса”).

Встроенные теги

Некоторые теги встроены в генератор PDF.

1. `PDF:currPageNum`, номер текущей страницы.
2. `PDF:currentX/PDF:currentY`, X и Y координаты курсора.
3. `PDF:savedX/PDF:savedY`, доступно после вызова команд `saveX/saveY`.
4. `[SAVED:ключ]`, доступно после команды `saveCoordinate`.

Пример:

```
{
   "name":"Line",
   "params":{
      "x1":"[PDF:savedX]",
      "x2":"[SAVED:endLineX]",
      "y1":"[PDF:currentY]",
      "y2":"[PDF:currentY]",
      "color":"#E8E8E8",
      "width":0.8
   }
},
{
   "name":"PrintText",
   "params":{
      "x":"[SAVED:pageNumX]",
      "y":"[SAVED:pageNumY]",
      "text":"Номер страницы: [PDF:currPageNum]"
   }
}
```

7.2.12 Работа со штрихкодом

Возможен вывод информации в виде штрихкода посредством использования шрифтов семейства Libre Barcode. Шрифты доступны в репозитории librebarcode. Каждый шрифт представляет собой определенный формат штрихкода. Поддерживаются следующие форматы:

- Code-128
- Code-39
- EAN13

Шрифт может включать оригинальный текст непосредственно под штрихкодом (шрифты с суффиксом Text).

Рассмотрим применение на примере формата Code-128.

Code-128

Шрифты для помещения в папку `assets/fonts` сервиса:

- LibreBarcode128Text-Regular.ttf
- LibreBarcode128-Regular.ttf

Как использовать

В генераторе команд выбрать нужный шрифт, например:

```
{
    "name": "SetPrintFont",
    "params":
    {
        "font_id": "LibreBarcode128Text-Regular",
        "size": 30
    }
}
```

Тестовый запрос

```
curl --request POST --data-binary "@templates/examples/pdf/embedded-report-generator/
↪request_barcode_code-128.json" http://localhost:8087/pdf_report_json
```

7.3 Сервис объединения PDF-документов

7.3.1 Общее описание сервиса

Сервис объединяет документы в формате PDF в один документ по HTTP-запросу в формате JSON.

Примечание: Формат XML не поддерживается в текущей версии.

7.3.2 Объединение PDF-документов

Описание сервиса

Наименование сервиса	Сервис объединения PDF-документов
Путь к сервису	[host]:[port]/pdf_merge_json
Метод	POST
Параметры	Тело запроса должно содержать объект в формате JSON. Подробнее про структуру тела запроса можно прочитать в подразделе “Структура тела запроса”. В ответ сервис отдаёт файл документа в формате PDF, закодированный в base64.
Назначение	Сервис предназначен для объединения нескольких документов в формате PDF в один.

Структура тела запроса

Тело запроса содержит объект в формате JSON, который включает:

1. Идентификатор запроса.
2. Входные данные: список описателей документов для объединения.
3. Опции запроса.

Пример:

```
{
    "request-id": "…",
    "input-data": {
        "document-descriptors": [
            {…},
            {…}
        ]
    },
    "options": {…}
}
```

Идентификатор запроса

Идентификатор запроса `request-id` служит для записи объединенного файла в отладочном режиме и для идентификации запроса в логе.

Входные данные

Объект входных данных `input-data` содержит:

Список (массив) `document-descriptors` описателей документов для объединения.

Пример:

```
"input-data": {
        "document-descriptors": [
            {…},
            {…}
        ]
    }
```

Описатель входного документа

Объект описателя входного документа поддерживает поля:

- `id` - строка. Идентификатор документа. Используется для ссылок на входной параметр в сообщениях об ошибках и диагностическом логе.
- `content-type` - строка. Указывает способ трактовки параметра `content`. Поддерживаемое значение: base64.
- `content` - строка. При значении типа контента base64 - закодированный в base64 PDF-документ. Примечание: Ограничения на количество страниц в документе нет.

Пример:

```
{
    "id": "doc1",
    "content-type": "base64",
    "content": "[doc 1 in base64 format]"
}
```

Опции запроса

Объект опций запроса `options` имеет поля:

- `enable-debug-merged-doc-save` - логическое значение. Указывает создавать ли копию отчета документа и копию запроса (имя файла - идентификатор шаблона) в локальной директории `reports_debug` сервиса. Значение по умолчанию - `false`.
- `enable-binary-output` - логическое значение. Указывает, что сервис будет выдавать результат в виде бинарных данных, без кодирования в base64. Значение по умолчанию - `false`.
- `enable-debug-pdf-log` - логическое значение. Включает расширенный диагностический лог создания PDF. Значение по умолчанию - `false`.

Пример:

```
"options":
    {
        "enable-debug-merged-doc-save": true,
        "enable-binary-output": false,
        "enable-debug-pdf-log": true
    }
```

Пример тела запроса

```
{
    "request-id": "pdf_merge_example1",
    "input-data": {
        "document-descriptors": [
            {
                "id": "doc1",
                "content-type": "base64",
                "content": "[doc 1 in base64 format]"
            },
            {
                "id": "doc2",
                "content-type": "base64",
                "content": "[doc 2 in base64 format]"
            }
        ]
    },
    "options":
    {
        "enable-debug-merged-doc-save": true,
        "enable-debug-pdf-log": true
    }
}
```

Примеры запросов находятся в директории `examples/pdf/pdf_merge` в *архиве примеров*.

Пример вызова сервиса

```
curl --request POST --data-binary "@templates/examples/pdf_merge/request_example.json
↪" http://localhost:8087/pdf_merge_json
```

Структура ответа

Ответ сервиса содержит объект в формате JSON, который включает:

1. Описание ошибки (код ошибки, сообщение об ошибке). В случае успешного ответа сервиса возвращается значение `null`.
2. Результат (закодированный в base64 PDF-файл объединенного документа).

Формат ответа

```
{
   error: {
       code
       message
   }
   result: "[base64 encoded pdf file]"
}
```

Пример ответа

```
{
   "error": null,
   "result": "UEsDBBQACAAIAPdKo...JwAAAAA="
}
```

Примечание: Объединенный документ может быть большего размера, чем сумма объединяемых документов. Связано с дублированием одинаковых ресурсов (например, шрифтов), используемых во входящих документах.

7.4 Сервис формирования печатной формы

7.4.1 Общее описание сервиса

Сервис формирует печатную форму документа в формате PDF по HTTP-запросу в формате JSON:

- На последней странице документа генерируется таблица подписантов.
- На промежуточных страницах отображается колонтитул с краткой информацией о документе.

Примечание: Формат XML не поддерживается в текущей версии.

7.4.2 Формирование печатной формы документа в формате PDF

Описание сервиса

Наименование сервиса	Сервис формирования печатной формы
Путь к сервису	[host]:[port]/print_form_pdf_json
Метод	POST
Параметры	Тело запроса должно содержать объект в формате JSON. Подробнее про структуру тела запроса можно прочитать в подразделе “Структура тела запроса”. В ответ сервис отдаёт файл печатной формы документа в формате PDF, закодированный в base64.
Назначение	Сервис предназначен для формирования печатной формы документа в формате PDF, включающей колонтикулы с краткой информацией о документе и таблицу подписантов на последней странице.

Структура тела запроса

Тело запроса содержит объект в формате JSON, который включает:

1. Идентификатор запроса.
2. Входные данные:
   - Описатель входного документа для создания печатной формы (оригинал документа PDF для создания печатной формы).
   - Описатель таблицы подписантов и колонтитула промежуточных страниц (данные для подстановки в таблицу подписантов и колонтитул промежуточных страниц с данными о подписываемом документе).
   - Опции вида генерируемых таблиц (цвет/размер шрифтов).
3. Опции запроса.

Пример:

```
{
    "request-id": "…",
    "input-data":
    {
        "input-document": {…},
        "signatures-info": {…},
        "draw-options": {…}
    },
    "options": {…}
  }
```

Идентификатор запроса

Идентификатор запроса `request-id` служит для записи файла в отладочном режиме и для идентификации запроса в логе.

Входные данные

Объект входных данных `input-data` содержит:

- Описатель входного документа для создания печатной формы.
- Описатель таблицы подписантов и колонтитула промежуточных страниц.
- Опции вида генерируемых таблиц.

Пример:

```
"input-data":
   {
        "input-document": {…},
        "signatures-info": {…},
        "draw-options": {…}
   }
```

Описатель входного документа

Объект описателя входного документа для создания печатной формы `input-document` имеет поля:

- `id` - строка. Идентификатор документа. Используется для ссылок на входной параметр в сообщениях об ошибках и диагностическом логе.
- `content-type` - строка. Указывает способ трактовки параметра content. Поддерживаемое значение: base64.
- `content` - строка. При значении типа контента base64 - закодированный в base64 PDF-документ. Примечание: Ограничения на количество страниц в документе нет.

Пример:

```
"input-document":
        {
         "id": "doc1",
         "content-type": "base64",
         "content": "[base64 encoded pdf file]"
        }
```

Описатель таблицы подписантов и колонтитула промежуточных страниц

Объект описателя таблицы подписантов и колонтитула промежуточных страниц `signatures-info` имеет поля:

- `signature-summary-text` - строка. Общее описание таблицы подписантов для вывода перед данными подписантов.
- `table-header-texts`- массив строк. Список заголовков таблицы подписантов. Примечание: в текущей версии может быть строго 4 столбца (ограничение элемента массива `signatures`, описанного ниже).
- `table-header-column-width-list` - массив чисел с плавающей точкой. Управляет шириной столбцов таблицы. Сумма всех нормализованных ширин должна равняться единице. Размер массива должен совпадать с размером поля `table-header-texts`.
- `signatures` - массив объектов с данными подписантов. Детали см. в пункте "Описатель данных подписанта" ниже.
- `intermediate-page-footer-info` - данные для вывода в колонтитуле на всех страницах документа, кроме последней, где находится таблица подписантов. Детали см. в пункте "Описатель колонтитула промежуточных страниц с данными о подписываемом документе" ниже.
- `logo-asset-name` - строка. Опциональный параметр. Имя файла изображения (с расширением) для вывода в правом верхнем углу таблицы подписантов. Путь задается относительно директории `assets/images/print_forms` в рабочем каталоге сервиса.

Пример:

```
"signatures-info":  {
    "signature-summary-text": "…",
    "table-header-texts": […],
    "table-header-column-width-list": […],
    "signatures": […],
    "intermediate-page-footer-info": {…},
    "logo-asset-name": "…"
}
```

Описатель данных подписанта

Элемент массива описателя данных подписанта `signatures` имеет поля:

- `description` - строка. Общее описание принадлежности подписи. Текст выводится в первый столбец таблицы подписантов.
- `certificate-owner` - объект со строковыми полями `organization` и `employee`, описывающий организацию, ФИО и должность владельца сертификата. Текст выводится во второй столбец.
- `certificate` - объект со строковыми полями `serial-number` и `validity`, описывающий серийный номер и период действия сертификата, используемого для подписи. Текст выводится в третий столбец.
- `signing-timestamp` - строка. Описание точного времени подписи. Текст выводится в четвертый столбец.

Пример:

```
{
     "description": "Подпись отправителя",
     "certificate-owner":
     {
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        "organization": "ООО \"ФИ-КРУГ\"",
        "employee": "Петров Иван Сидорович, Генеральный Директор"
    },
    "certificate":
    {
        "serial-number": "11255807011ABCBBE4DBF93ECCCB96C45",
        "validity": "с 01.02.2001 18:48 по 01.02.2031 18:48 GMT+03:00"
    },
    "signing-timestamp": "01.02.2021 19:00 GMT+03:00"
}
```

Описатель колонтитула промежуточных страниц с данными о подписываемом документе

Объект описателя колонтитула промежуточных страниц с данными о подписываемом документе `intermediate-page-footer-info` имеет поля:

- `lines`- массив строк. Произвольные данные, описывающие подписываемый документ. Выводятся в виде таблицы с невидимыми границами. Последней строкой будет описание номера текущей страницы, если это включено полем `print-page-number`.
- `logo-asset-name` - строка. Опциональный параметр. Имя файла изображения (с расширением) для вывода в правой части данных о документе. Путь задается относительно директории `assets/images/print_forms` в рабочем каталоге сервиса.
- `print-page-number` - логической значение. Указывает выводить ли номер текущей страницы документа под данными о документе.

Пример:

```
{
    "lines": [
        "Передан через XXX 01.02.2003 15:20 GMT+03:00",
        "abcdef00-0000-1111-2222-abcdefabcdef"
    ],
    "logo-asset-name": "golang-gopher.png",
    "print-page-number": true
}
```

Опции вида генерируемых таблиц

Объект опций вида генерируемых таблиц `draw-options` имеет поля:

- `font-color` - строка в формате «#rrggbb», где rgb - шестнадцатеричные компоненты RGB цвета в диапазоне от 0 до ff. Пример красного цвета: «font-color»:“#ff0000”.
- `font-size` - целое число. Размер шрифта данных подписантов в таблице подписантов. Единица равна 1/72 дюйма.
- `large-font-size` - целое число. Размер шрифта вывода текста `signature-summary-text` в таблице подписантов. Единица равна 1/72 дюйма.
- `table-border-color` - строка в формате «#rrggbb», где rgb - шестнадцатеричные компоненты RGB цвета в диапазоне от 0 до ff. Пример красного цвета: «font-color»:“#ff0000”. Цвет границ таблицы подписантов.

Пример:

```
{
    "font-color": "#0054BE",
    "font-size": 8,
    "large-font-size": 14,
    "table-border-color": "#0054BE"
}
```

Опции запроса

Объект опций запроса `options` имеет поля:

`enable-debug-merged-doc-save` - логическое значение. Указывает создавать ли копию отчета документа и копию запроса (имя файла - идентификатор шаблона) в локальной директории `reports_debug` сервиса. Значение по умолчанию - `false`. `enable-binary-output` - логическое значение. Указывает, что сервис будет выдавать результат в виде бинарных данных, без кодирования в base64. Значение по умолчанию - `false`. `enable-debug-pdf-log` - логическое значение. Активирует расширенный диагностический лог создания PDF. Значение по умолчанию - `false`.

Пример:

```
"options":
   {
       "enable-debug-doc-save": true,
       "enable-binary-output": false,
       "enable-debug-pdf-log": true
   }
```

Пример тела запроса

```
{
   "request-id": "print_form_pdf_example1",
   "input-data":
   {
       "input-document":
       {
           "id": "doc1",
           "content-type": "base64",
           "content": "JVBERi0xLjcKJYGBgYEKCjYgMCB….VmCjI1ODI5CiUlRU9G"
       },
       "signatures-info":
       {
           "signature-summary-text": "Документ подписан и передан через оператора АО␣
↪«Пример»",
           "table-header-texts": ["", "Владелец сертификата: организация, сотрудник",
↪"Сертификат: серийный номер, период действия", "Дата и время подписания"],
           "table-header-column-width-list": [0.15, 0.33, 0.32, 0.20],
           "signatures": [
               {
                   "description": "Подпись отправителя",
                   "certificate-owner":
                   {
                       "organization": "ООО \"ФИ-КРУГ\"",
                       "employee": "Петров Иван Сидорович, Генеральный Директор"
```

(continues on next page)

(продолжение с предыдущей страницы)

```
                    },
                    "certificate":
                    {
                        "serial-number": "11255807011ABCBBE4DBF93ECCCB96C45",
                        "validity": "с 01.02.2001 18:48 по 01.02.2031 18:48 GMT+03:00"
                    },
                    "signing-timestamp": "01.02.2021 19:00 GMT+03:00"
                },
                {
                    "description": "Подпись получателя",
                    "certificate-owner":
                    {
                        "organization": "ООО «СНАБЖЕНИЕ КОНЦЕССИЙ»",
                        "employee": "Сидоров Петр Иванович, Директор"
                    },
                    "certificate":
                    {
                        "serial-number": "3610D7230004000503BF",
                        "validity": "с 04.05.2006 16:07 по 31.12.2021 23:59 GMT+03:00"
                    },
                    "signing-timestamp": "02.02.2021 13:00 GMT+03:00"
                }
            ],
            "intermediate-page-footer-info":
            {
                "lines": [
                    "Передан через XXX 01.02.2003 15:20 GMT+03:00",
                    "8f109134-403b-4de5-aa09-6d10462ec071"
                ],
                "logo-asset-name": "golang-gopher.png",
                "print-page-number": true
            },
            "logo-asset-name": "golang-gopher.png"
        },
        "draw-options":
        {
            "font-color": "#0054BE",
            "font-size": 8,
            "large-font-size": 14,
            "table-border-color": "#0054BE"
        }
    },
    "options":
    {
        "enable-debug-doc-save": true,
        "enable-debug-pdf-log": true
    }
}
```

Примеры запросов находятся в директории `examples/pdf/print_form` в *архиве примеров*.

Пример вызова сервиса

```
curl --request POST --data-binary "@templates/examples/pdf/print_form/request_
↪example1.json" http://localhost:8087/print_form_pdf_json
```

Структура ответа

Ответ сервиса содержит объект в формате JSON, который включает:

1. Описание ошибки (код ошибки, сообщение об ошибке). В случае успешного ответа сервиса возвращается значение `null`.
2. Результат (закодированный в base64 PDF-файл печатной формы документа).

Формат ответа

```
{
   error: {
       code
       message
   }
   result: "[base64 encoded pdf file]"
}
```

Пример ответа

```
{
   "error": null,
   "result": "UEsDBBQACAAIAPdKo...JwAAAAA="
}
```

Служебные директории

Изображения для использования в качестве логотипа, находятся в директории `assets/images/print_forms`, расположенной в директории приложения сервиса.

7.5 Сервис конвертации XLSX документов в JSON, XML, CSV

7.5.1 Общее описание сервиса

Сервис конвертирует XLSX документ в форматы JSON, XML, CSV по HTTP-запросу в формате JSON или по HTTP-запросу в формате multipart/form-data.

В ответ сервис отдает содержимое входного документа представленное в одном из целевых форматов.

Примечание: Сервис конвертации доступен в сервере отчетов, начиная с версии 3.3.1.1

Описание сервиса

Наименование сервиса	Сервис конвертации XLSX документов в JSON, XML, CSV
Путь к сервису	Для multipart/form-data запросов • [host]:[port]/xlsx_convert Для json запросов • [host]:[port]/xlsx_convert_json
Метод	POST
Параметры	Тело запроса должно содержать объект в формате JSON или объекты в формате multipart/form-data. Подробнее про структуру тела запроса можно прочитать в подразделе “Структура тела запроса”. В ответ сервис отдаёт содержимое документа в целевом формате.
Назначение	Сервис предназначен для конвертирования XLSX документа в форматы JSON, XML, CSV

Структура тела запроса

В общем виде запросы в формате JSON или multipart/form-data включают:

Описатель документа. Опции запроса. Пример для JSON запроса:

```
    {
"document":
{
    "name": "",
    "data": ""
},
"options": {…}
    }
```

Описатель документа

Описатель документа для формата JSON имеет поля:

- `name` - строка. Имя - идентификатор документа. Используется для ссылок на входной параметр в сообщениях об ошибках и диагностическом логе.
- `data` - строка. Закодированный в BASE64 XLSX-документ.

Описатель документа для формата multipart/form-data состоит из параметра **xlsx=»@[путь к XLSX-документу]**

Например:

```
curl --request POST -F xlsx="@convert_example.xlsx" -F output-format="xml" http://
↪localhost:8087/xlsx_convert
```

Опции запроса

Запрос может содержать следующие опции запроса:

- `output-format` - строка. Целевой формат конвертации. Поддерживаемые значения: “JSON”, “XML”, “CSV”. Значение по умолчанию - `JSON`.
- `process-sheets-numbers` - массив целых чисел. Определяет порядковые номера страниц для конвертации из входного документа (Например: [1,2,3] - для обработки первых трех страниц). Нумерация начинается с “1”. Значение по умолчанию - `все страницы` для JSON и XML, `первая страница` для CSV.
- `process-from-start-of-sheet` - логическое значение. Указывает начать обработку документа с самой первой ячейки (A1) независимо есть ли в ней данные или нет. Иначе обработка будет производиться с первой ячейки с данными. Значение по умолчанию - `false`.
- `no-extend-line` - логическое значение. Данная опция отвечает за дополнение каждой строки исходного документа до максимальной длины строки в документе с целью получения двумерного массива данных (именно такое поведение конвертора предусмотрено по умолчанию). При значении true опция указывает не дополнять строки и на выходе будет массив массивов данных. Значение по умолчанию - `false`.
- `process-start-cell` - строка. Данная опция позволяет указать стартовую ячейку для обработки (в формате именования ячеек, например “E5”), то есть задает левую верхнюю границу обработки по строке-столбцу, что позволяет конвертировать только интересующий диапазон данных. Значение по умолчанию - `первая ячейка с данными` (будет определена автоматически).
- `process-end-cell` - строка. Данная опция позволяет указать конечную ячейку для обработки (в формате именования ячеек, например “G8”), то есть задает правую нижнюю границу обработки по строке-столбцу, что позволяет конвертировать только интересующий диапазон данных. Значение по умолчанию - `последняя ячейка с данными` (будет определена автоматически).
- `encoding` - строка. Данная опция позволяет указать целевую кодировку результата обработки. Поддерживаемые значения: WIN1250, WIN1251, WIN1252, WIN1253, WIN1254, WIN1255, WIN1256, WIN1257, WIN874, WIN866, KOI8R, ISO_8859_5. Значение по умолчанию - `UTF8`.
- `enable-debug-report-save` - логическое значение. Указывает создавать ли копию запроса в локальной директории `reports_debug` сервиса. Значение по умолчанию - `false`.

Для запросов в формате JSON опции передаются в объекте `options`

Пример:

```
"options":
{
  "output-format": "json",
  "no-extend-line": false,
  "process-sheets-numbers":[1,2],
  "process-from-start-of-sheet": false,
  "process-start-cell":"C2",
  "process-end-cell":"G5",
  "enable-debug-report-save": true
}
```

Для запросов в *multipart/form-data* формате каждая опция передается как отдельный параметр, все параметры задаются как строки.

Пример:

```
curl --request POST -F "xlsx=@convert_example.xlsx" -F output-format="json" -F no-
↪extend-line="false" -F process-sheets-numbers="1,2" -F process-from-start-of-sheet=
↪"false"
-F process-start-cell="C2" -F process-end-cell="G5" http://localhost:8087/xlsx_convert
```

Примеры запросов находятся в директории `examples/xslx` в *архиве примеров*.

Пример вызова сервиса

Для запросов в формате JSON:

```
curl --request POST --data-binary "@convert_example.json" http://localhost:8087/xlsx_
↪convert_json > convert_example_from_json.json
```

Для запросов в формате multipart/form-data:

```
curl --request POST -F "xlsx=@convert_example.xlsx" -F output-format="xml" http://
↪localhost:8087/xlsx_convert > convert_example1.xml

curl --request POST -F "xlsx=@convert_example.xlsx" -F output-format="csv" -F process-
↪from-start-of-sheet="false" -F process-start-cell="C2" -F process-end-cell="G5"
http://localhost:8087/xlsx_convert > convert_example2.csv

curl --request POST -F "xlsx=@convert_example.xlsx" -F output-format="json" -F no-
↪extend-line="true" -F process-sheets-numbers="1,2" -F process-from-start-of-sheet=
↪"false"
http://localhost:8087/xlsx_convert > convert_example2.json
```

Структура ответа

Ответ сервиса содержит текстовый поток данных в целевой кодировке

Пример конвертации

Рассмотрим в качестве примера конвертацию диапазона “D2”:»G4» второй страницы документа `convert_example.xlsx` из *архива примеров*:

	№ п/п	Номер страхового свидетельства обязательного пенсионного страхования застрахованного лица	Пол застрахованного лица	Фамилия, имя, отчество (при наличии) застрахованного лица	Число, месяц, год рождения застрахованного лица	Дата заключения договора об обязательном пенсионном страховании	Номер договора об обязательном пенсионном страховании
3	1	001-002-003 00	Ж	Горбунова Екатерина Васильевна	01.01.1970	01.01.1988	АВС-001-002-003-00
4	2	001-002-004 00	М	Харитонов Степан Эдуардович	02.01.1970	02.01.1988	АВС-001-002-004-00
5	3	001-002-005 00	Ж	Шестакова Анна Львовна	03.01.1970	03.01.1988	АВС-001-002-005-00

Запрос в формате multipart/form-data:

```
curl --request POST -F "xlsx=@convert_example.xlsx" -F output-format="csv" -F process-
↪sheets-numbers="2" -F process-start-cell="D2" -F process-end-cell="G4"
http://localhost:8087/xlsx_convert
```

Результат конвертации:

```
Пол застрахованного лица;Фамилия, имя, отчество (при наличии) застрахованного лица;
 ↪Число, месяц, год рождения застрахованного лица;Дата заключения договора об␣
 ↪обязательном пенсионном страховании
Ж;Горбунова Екатерина Васильевна;01.01.1970;01.01.1988
М;Харитонов Степан Эдуардович;02.01.1970;02.01.1988
```

ГЛАВА 8

Примеры

8.1 Архив примеров

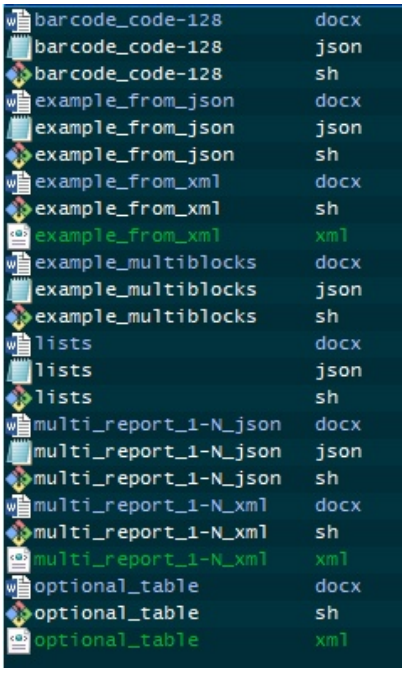
В данном разделе можно ознакомиться с примерами запросов и шаблонов для сервиса.

`Архив примеров` необходимо распаковать в директорию **templates** в корне сервиса. Распакованный архив содержит следующие каталоги:

- – `pdf` - содержит PDF - шаблоны страниц, используемых в *генераторе отчетов PDF*
- – `examples` - представляет собой следующее дерево каталогов:

```
├──docx
├──pdf
│   ├──embedded-report-generator
│   ├──pdf_merge
│   ├──print_form
│   └──report-generator
│       └──pdf_report_gen
└──xlsx
```

Каталоги **docx, xlsx** содержат набор запросов (в форматах json/xml), документов - шаблонов (*.docx и *.xlsx) и *скриптов (*.sh) для получения отчетов.



Каталог **pdf** содержит следующие каталоги:

- `embedded-report-generator` -запросы (.json) и скрипты (*.sh) для формирования отчетов в формате .pdf через *Генератор отчетов PDF*.
- `pdf_merge` -запросы (.json) и скрипты (*.sh) для *бъединения PDF-документов*.
- `print_form` -запросы (.json) и скрипты (*.sh) для *формирования печатной формы*.
- `report-generator` -запросы (.json) и скрипты (*.sh) для формирования отчетов через *Генератор отчетов PDF* с примером генератора как часть сервиса. Каталог `pdf_report_gen` с примерами генераторов необходимо разместить в корневом каталоге сервиса.

8.2 Формирование отчета из архива примеров

Рассмотрим генерацию отчета из каталога docx на основе запроса lists.json

```
{
    "template":
    {
        "uri": "local",
        "id": "examples/docx/lists"
    },
    "input-data":
    {
        "CONDITIONAL_TAG_TRUE": "true",
        "CONDITIONAL_TAG_FALSE": "false",
        "BULLET_LIST": ["bullet item 1", "bullet item 2", "bullet item 3"],
        "NUMBERED_LIST": ["numbered item 1", "numbered item 2", "numbered item 3"],
        "EMPTY_BULLET_LIST": [],
        "EMPTY_NUMBERED_LIST": []

    },
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    "options":
    {
        "enable-debug-report-save": false,
      "enable-binary-output": true,
        "formatting":
        {
            "tables":
            {
                "enable-cells-auto-merge": true
            }
        }
    }
}
```

В результате запуска скрипта lists.sh

```
curl --request POST --data-binary "@lists.json" http://localhost:8087/word_report_
↪json --output "lists_report.docx"
```

в каталоге templates/examples/docx был записан файл отчета lists_report.docx

---

Примечание: скриптам может понадобиться назначить права на исполнение. Это можно сделать следующей командой: chmod +x *.sh

---

ГЛАВА 9

Мой первый отчет

9.1 Общее описание

В данном подразделе будет рассмотрен процесс создания простого отчета с помощью сервера отчетов. Для работы необходимо удостовериться, что установка сервера отчетов прошла успешно.

9.2 Первый шаблон

Для формирования отчета необходимо создать шаблон документа в формате DOCX или XLSX с помощью MS Word/Excel, Р7-Офис, LibreOffice Writer/Calc и т.п.

Шаблоны документов могут содержать теги, которые будут заменены входными данными из запроса. Тег задается в квадратных скобках. Пример: [задолженность]. Подробнее про создание шаблонов в формате DOCX и XLSX можно прочитать в разделе Сервис формирования отчетов из документа-шаблона.

Для первого отчета создадим простой шаблон в формате DOCX, который содержит несколько тегов: first_report_template.docx. Примеры более сложных шаблонов можно найти в архиве примеров.

Созданный шаблон размещаем на сервере в директории `templates`, расположенной в директории приложения сервиса.

Примечание: Шаблоны страниц в формате PDF хранятся в директории `templates/pdf`, расположенной в директории приложения сервиса.

9.3 Первый запрос

Далее формируем HTTP-запрос.

Запрос для первого отчета в формате JSON будет выглядеть следующим образом: first_report.json. Документ-шаблон, созданный ранее, указан в атрибуте `id` элемента `template`. Примеры более сложных запросов можно найти в архиве примеров.

Примечание. Также для запросов в формате DOCX или XLSX может быть использован формат XML.

Для получения файла отчета используем Сервис формирования отчетов из документа-шаблона.

Для получения первого отчета на основе созданного шаблона DOCX используем URI http://localhost:8087/word_report_json.

9.3.1 Описание

Наименование сервиса	Сервис формирования печатной формы
Путь к сервису	[host]:[port]/print_form_pdf_json
Метод	POST

Параметры	Тело запроса должно содержать JSON объект:
	`{`
	`"template":`
	`{`
	`"uri": "local",`
	`"id": "first_report_template"`
	`},`
	`"input-data":`
	`{`
	`"ORGANIZATION": "АО «Пример»",`
	`"DATE": "01.01.2023",`
	`"EMP": "Иванов Иван Иванович"`
	`},`
	`"options":`
	`{`
	`"formatting": {`
	`"tables": {`
	`"enable-cells-auto-merge": true`
	`}`
	`}`
	`}`
	`}`
	Объект `template` имеет атрибуты (поля):
	• `uri` - строка. Задает расположение файла документа-шаблона в зависимости от того, как трактуется параметр `id`. Поддерживаемое значение: `local`.
	• `id` - строка. Идентификатор шаблона. Путь к файлу документа-шаблона относительно директории сервиса `templates`. Также используется для записи файла отчета в отладочном режиме и для идентификации запроса в логе.

continues on next page

Таблица 1 – продолжение с предыдущей страницы

	`input-data` - входные данные для подстановки в шаблон. Объект `input-data` может содержать: • теги простых строковых данных; • теги описателей таблиц; • теги описателей списков; • теги изображений; • теги блоков. `options` - опции запроса. Подробнее про структуру JSON можно прочитать в разделе Сервис формирования отчетов из документа-шаблона. В ответ сервис отдаёт файл документа отчета в формате DOCX, закодированный в BASE64. В случае, если необходимо получить результат без кодирования в BASE64, в опциях запроса необходимо указать флаг enable-binary-output, равный true. `{ "template": { "uri": "local", "id": "first_report_template" }, "input-data": { "ORGANIZATION": "АО «Пример»", "DATE": "01.01.2023", "EMP": "Иванов Иван Иванович" }, "options": { "enable-binary-output": true, "formatting": { "tables": { "enable-cells-auto-merge": true } } } }`
Назначение	Сервис предназначен для формирования печатной формы документа в формате PDF, включающей колонтикулы с краткой информацией о документе и таблицу подписантов на последней странице.

9.3.2 Пример запроса

```
curl -X POST \
'http://localhost:8087/word_report_json' \
-H 'Content-Type: application/json' \
-d '{
   "template":
   {
       "uri": "local",
       "id": "first_report_template"
   },
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    "input-data":
    {
        "ORGANIZATION": "АО «Пример»",
        "DATE": "01.01.2023",
        "EMP": "Иванов Иван Иванович"
    },
    "options":
    {
        "formatting": {
            "tables": {
                "enable-cells-auto-merge": true
            }
        }
    }
}'
```

9.3.3 Пример запроса для сохранения результата в файл без кодирования в base64

```
curl -X POST \
'http://localhost:8087/word_report_json' \
-H 'Content-Type: application/json' \
-d '{
    "template":
    {
        "uri": "local",
        "id": "first_report_template"
    },
    "input-data":
    {
        "ORGANIZATION": "АО «Пример»",
        "DATE": "01.01.2023",
        "EMP": "Иванов Иван Иванович"
    },
    "options":
    {
        "enable-binary-output": true,
        "formatting": {
            "tables": {
                "enable-cells-auto-merge": true
            }
        }
    }
}'
-o first_report.docx
```

9.4 Получение первого ответа

9.4.1 Формат возвращаемого ответа

```
{
   error: {
       code:
       message:
   }
   result: "[base64 encoded docx/pdf file]"
}
```

9.4.2 Пример ответа

```
{
   "error": null,
   "result": "UEsDBBQACAAIAPdKo...JwAAAAA="
}
```

9.5 Завершение

В итоге получаем файл документа первого отчета в формате DOCX, закодированный в base64, или сообщение об ошибке.