ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ
«НЕОЛАНТ СЕРВИС»

Приложение Валидатор/загрузчик инженерных данных в Систему управления инженерными данными НЕОСИНТЕЗ

УСТАНОВКА И НАСТРОЙКА

Ставрополь, 2025 г.

СОДЕРЖАНИЕ

СПИСОК СОКРАЩЕНИЙ 3
1. ОБЩИЕ ПОЛОЖЕНИЯ 4
1.1. Наименование системы 4
1.2. Область применения 4
1.3. Краткое описание возможностей 4
1.4. Уровень подготовки администратора 5
1.5. Перечень эксплуатационной документации, с которой необходимо ознакомиться администратору 5
2. УСТАНОВКА И НАСТРОЙКА ПРИЛОЖЕНИЯ 6
2.1. Системные требования 6
2.2. Установка приложения Валидатор/загрузчик инженерных данных 6
2.3. Системный журнал 7
2.4. Настройка СУИД НЕОСИНТЕЗ для работы с приложением 7
3. Конфигурационные файлы 8
3.1. Настройки приложения для работы с сервером СУИД НЕОСИНТЕЗ 9
3.2. Конфигурационный файл схемы данных 10
3.3. Параметры настроек проекта и правил проверки пакетов 11
3.4. Конфигурационный файл правил проверки и настроек проекта 11
3.5. Работа с YAML файлами 26
3.6. Регулярные выражения 28
Лист регистрации изменений 30

СПИСОК СОКРАЩЕНИЙ

Сокращение	Наименование
API	Application Programming Interface – программный интерфейс приложения для доступа сторонних систем
АРМ	Автоматизированное рабочее место
ОС	Операционная система
ПК	Персональный компьютер
ПСД	Проектно-сметная документация
СУИД	Систему управления инженерными данными

1. ОБЩИЕ ПОЛОЖЕНИЯ

1.1. Наименование системы

Полное наименование приложения: Валидатор/загрузчик инженерных данных в Систему управления инженерными данными НЕОСИНТЕЗ.

Краткое наименование приложения: Валидатор/загрузчик инженерных данных (далее - Система).

1.2. Область применения

Настоящий документ предназначен для пользователей выполняющих проверку и загрузку инженерных данных в Систему управления инженерными данными НЕОСИНТЕЗ.

Пользователями приложения являются операторы/специалисты (далее Пользователи) предприятий и организаций, работающих с инженерными данными в нефтегазовой и химической отрасли, энергетике, металлургии, добыче полезных ископаемых и других отраслях промышленности для которых требуется управление инженерными данными на любых стадиях жизненного цикла инфраструктурного объекта.

В настоящем документе приведено описание основных возможностей приложения Валидатор/загрузчик инженерных данных и основных операций, необходимых Пользователям для выполнения своих функций.

1.3. Краткое описание возможностей

Валидатор/загрузчик инженерных данных расширяет функциональность СУИД НЕОСИНТЕЗ и позволяет формировать, валидировать на соответствие заданным правилам и загружать в СУИД НЕОСИНТЕЗ наборы данных инженерных моделей с определённой кодировкой. Правила кодировки настраиваются в конфигурационных файлах. При загрузке данных в СУИД НЕОСИНТЕЗ создаются новые объекты или обновляются существующие.

Компоненты ядра приложения инженерных данных могут быть использованы в составе сервисов представляющих API внешним приложениям, что может быть использовано для создания автоматизированного конвейера получения, валидации и загрузки различных видов инженерных данных в СУИД НЕОСИНТЕЗ.

1.4. Уровень подготовки администратора

Администратор должен обладать следующими знаниями и навыками:

- администрирование операционных систем семейства Windows;
- знаниями работы стека протоколов TCP/IP и принципов маршрутизации.

1.5. Перечень эксплуатационной документации, с которой необходимо ознакомиться администратору

Перед началом работы администратору необходимо ознакомиться с настоящим документом и документом «Описание продукта».

2. УСТАНОВКА И НАСТРОЙКА ПРИЛОЖЕНИЯ

2.1. Системные требования

Приложение Валидатор/загрузчик инженерных данных в Систему управления инженерными данными НЕОСИНТЕЗ является десктопным приложением, имеет русскоязычный графический интерфейс пользователя, и функционирует на ПК. Приложение может функционировать под управлением 64-х разрядных версиях ОС Windows 10 и более новых. Для работы на Astra Linux требуется дополнительно установить .NET SDK. Для осуществления загрузки пакетов проектной документации необходим доступ к серверу НЕОСИНТЕЗ по сети (используются стандартный HTTP/HTTPS протокол для доступа к REST API НЕОСИНТЕЗ).

Приложение предъявляет невысокие требования к производительности ПК, рекомендуется использование, как минимум, следующей конфигурации:

Таблица 1 Требования к конфигурации ПК

Элемент	**Требования**
Операционная система	Одна из операционных систем: Windows 10 с пакетом обновления 1 (SP1) (64-разрядная) Window 11 (64-разрядная) Astra Linux Common Edition 1.7+ Linux-дистрибутив, с поддержкой X11 или Wayland
Процессор	Intel® Core™ i3 Processors и лучше
Оперативная память	4 Гбайт или больше
Видеоадаптер	Интегрированные видеокарты Intel HD Graphics или лучше
Дисковая подсистема	200 МБ свободного места для установки приложения, дополнительный объём по объёму передаваемой проектной документации (уточняется в процессе эксплуатации), передаваемые пакеты ПСД могут размещаться на сетевых дисках
Сетевой интерфейс	не менее 100 Мбит/с

Приложение работает без инсталляции в системе, для начала использования приложения достаточно скопировать папку с приложением на локальный диск АРМ и запуска исполняемого файла или запускаться со съемного носителя/сетевого диска. Для работы приложение не требует привилегий администратора.

2.2. Установка приложения Валидатор/загрузчик инженерных данных

Для установки приложения Валидатор/загрузчик инженерных данных достаточно распаковать полученный архив с приложением на один из дисков АРМ, где будет осуществляться работа с приложением.

Для запуска приложения необходимо запустить исполняемый файл Neolant.VerificationUtility.AvaloniaUI.exe в папке с файлами приложения. При регулярном использовании приложения для удобства пользователя можно создать ярлык на рабочем столе, указывающий на исполняемый файл приложения.

Для осуществления загрузки пакетов ПСД необходимо наличие доступа к СУИД НЕОСИНТЕЗ по протоколу HTTPS.

Приложение Валидатор/загрузчик инженерных данных осуществляет проверку корректности и целостности пакетов ПСД с использованием правил проверки пакетов ПСД и настроек проектов. Настройки проектов и правила проверки пакетов, используемые приложением, хранятся в директории %AppData%\Roaming\Neolant\VerificationUtility.

2.3. Системный журнал

Приложением Валидатор/загрузчик инженерных данных ведется системный журнал с технической информацией о работе приложения (логами), предназначенный для администратора или разработчика. Данный журнал ведется в виде текстовых лог-файлов и сохраняется в подпапку logs в папке приложения.

При возникновении критической ошибки или исключения в интерфейсе приложения отображается сообщение с информацией о проблеме. При этом в системные лог-файлы с именами вида YYYY-MM-DD_apperor.log, где YYYY-MM-DD – текущая дата, записывается более детальная информация о проблеме.

2.4. Настройка СУИД НЕОСИНТЕЗ для работы с приложением

На всех серверах СУИД НЕОСИНТЕЗ, на которые будет выполняться загрузка проектной документации Приложением Валидатор/загрузчик инженерных данных необходимо выполнить следующие настройки:

1. Добавить клиента API для приложения:
   a. Меню «Администрирование» - «Настройки» - «API».
   b. При помощи кнопки «Добавить» создать нового клиента API, пример возможных настроек приведен рисунке1.

.

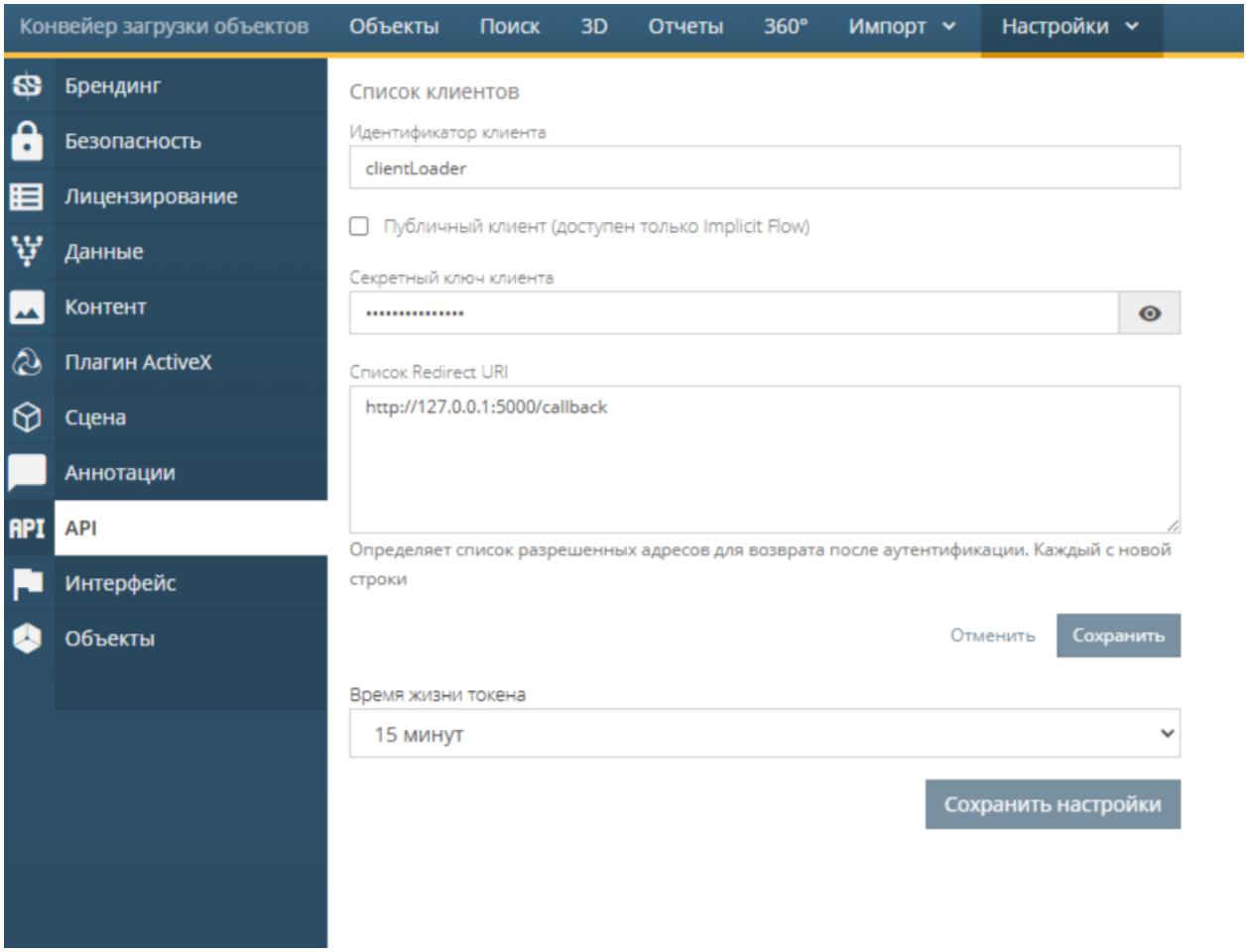


Рисунок 1 Настройка клиента API

2. Создать объект для хранения конфигурационного файла схемы данных Приложения Валидатор/загрузчик инженерных данных (при его отсутствии на сервере):

a. Меню «Объекты».
b. В ветке «Справочники» создать объект класса «Элемент справочника с файлом[1]» и именем «Конфигурационный файл».
c. В атрибут «Файл[2]» созданного объекта загрузить конфигурационный файл в формате JSON entities.json из комплекта поставки приложения Валидатор/загрузчик инженерных данных.

3. КОНФИГУРАЦИОННЫЕ ФАЙЛЫ

Для работы приложения Валидатор/загрузчик инженерных данных требуется как минимум три конфигурационных файла:

[1] В случае отсутствия класса «Элемент справочника с файлом» требуется его создать, подробное описание по созданию классов описано в руководстве администратора СУИД НЕОСИНТЕЗ
[2] В случае отсутствия атрибута «Файл» требуется его создать, подробное описание по созданию атрибутов описано в руководстве администратора СУИД НЕОСИНТЕЗ

- файл настроек приложения для работы с сервером СУИД НЕОСИНТЕЗ,
- файл схемы данных,
- файл с настройками проекта и правилами проверки пакета.

Конфигурационные файлы описаны в формате JSON.

При запуске приложение проверяет наличие конфигурационного файла Packloader.json, расположенного в папке %AppData\Roaming\Neolant\VerificationUtility\при работе приложения на Windows или ~/.config/Neolant/VerificationUtility при работе на Linux.

Если файл существует, и его версия поддерживается приложением, Валидатор/загрузчик инженерных данных запускается в обычном режиме с настройками из данного файла.

Если файла Packloader.json в AppData нет, Валидатор/загрузчик инженерных данных откроет окно настроек для первоначальной настройки приложения.

Если конфигурационный файл Packloader.json есть, но его версия не поддерживается приложением или в нем содержится ошибка, Валидатор/загрузчик инженерных данных выведет ошибку проверки конфигурационного файла, при этом присутствует возможность повторной настройки приложения с записью корректного конфигурационного файла.

3.1. Настройки приложения для работы с сервером СУИД НЕОСИНТЕЗ

Параметры, необходимые для доступа приложения Валидатор/загрузчик инженерных данных к REST API СУИД НЕОСИНТЕЗ хранятся в конфигурационном файле appsetting.json расположенном в папке приложения Валидатор/загрузчик инженерных данных в секции "NeosyntezConfiguration" и содержат следующие поля:

Файл настроек приложения для работы с сервером СУИД НЕОСИНТЕЗ состоит из следующих полей:

- ClientId – идентификатор клиента,
- ClientSecret – секретный ключ клиента,
- BaseAddress – URL СУИД НЕОСИНТЕЗ
- ConfigurationFileName – наименование объекта для хранения конфигурационного файла описывающего схему данных,
- ConfigurationFileClassName – наименование класса, в котором хранится конфигурационный файл описывающий схему данных.

Параметры ClientId и ClientSecret должны соответствовать настройкам API сервера СУИД НЕОСИНТЕЗ.

Параметры ConfigurationFileName и ConfigurationFileClassName должны соответствовать созданному для хранения схемы данных объекту на сервере СУИД НЕОСИНТЕЗ.

Все приложения, работающие с одним набором серверов СУИД НЕОСИНТЕЗ используют одинаковые настройки в данном файле. В связи с этим, данный конфигурационный файл может быть подготовлен однократно, и включен в дистрибутив приложения, предназначенный для установки у всех пользователей, работающих с одними и теми же серверами СУИД НЕОСИНТЕЗ.

3.2. Конфигурационный файл схемы данных

Настройки привязки приложения Валидатор/загрузчик инженерных данных к схеме данных портала СУИД НЕОСИНТЕЗ, а также другие настройки, необходимые для работы приложения, хранятся в специальном конфигурационном файле.

Для обеспечения возможности работы приложения Валидатор/загрузчик инженерных данных конфигурационный файл должен быть загружен на СУИД НЕОСИНТЕЗ в атрибут типа «Файл» специального объекта для хранения конфигурации.

Для получения конфигурационного файла схемы данных приложение Валидатор/загрузчик инженерных данных использует указатель, в котором зафиксировано наименование объекта для хранения конфигурации и наименование атрибута, в который загружен конфигурационный файл. По умолчанию указатель представляет собой секцию JSON файла appsettings.json, расположенного в папке приложения. Пример настройки указателя:

```
{
…
"ConfigurationFileName": "Конфигурационный файл",
"ConfigurationFileClassName": "Элемент справочника с файлом"
}
```

Для корректной работы приложения Валидатор/загрузчик инженерных данных необходимо убедиться, что значения атрибутов ConfigurationFileName и ConfigurationFileClassName в конфигурационном файле приложения соответствуют размещённому на СУИД НЕОСИНТЕЗ конфигурационному файлу.

Типовой шаблон конфигурационного файла схемы данных entities.json поставляется в комплекте с приложением Валидатор/загрузчик инженерных данных, в случае отсутствия на настраиваемом узле СУИД НЕОСИНТЕЗ данного конфигурационного файла Перед использованием данного шаблона, необходимо

выполнить его подготовку для работы с актуальной схемой данных используемого экземпляра СУИД НЕОСИНТЕЗ.

3.3. Параметры настроек проекта и правил проверки пакетов

Наборы настроек проекта приложения Валидатор/загрузчик инженерных данных хранятся в конфигурационном файле UserSetting.json, расположенном в папке %AppData\Roaming\Neolant\VerificationUtility\ при работе приложения на Windows или ~/.config/Neolant/VerificationUtility при работе приложения на Linux, который загружается во внутреннюю базу наборов параметров проекта приложения Валидатор/загрузчик инженерных данных перед использованием.

Создание и управление наборами правил проверки пакетов и общими настройками проектов осуществляется с помощью настроек в интерфейсе приложения.

3.4. Конфигурационный файл правил проверки и настроек проекта

Для выполнения проверки пакета требуется как минимум два конфигурационных файла: файл глобальной конфигурации проекта и файл с правилами проверки пакета. Файлы могут иметь произвольные имена, разрешение файлов должно быть yaml. Конфигурационные файлы описаны в формате yaml. Приложение Валидатор/загрузчик инженерных данных может работать с несколькими экземплярами конфигурационных файлов каждого типа одновременно, при этом файлы должны иметь разные идентификаторы.

При запуске приложение проверяет наличие конфигурационного файла, расположенного в папке %AppData\Roaming\Neolant\VerificationUtility\ при работе прилоения на Windows или %.local/share/VerificationUtilityAvalonia при работе на Linux.

Если файл существует, и его версия поддерживается приложением, Валидатор/загрузчик инженерных данных запускается в обычном режиме с настройками из данного файла.

При первичном запуске используется исходный тестовый конфигурационный файл.

3.4.1. Глобальные настройки проекта

Глобальные настройки проекта описываются в отдельном файле конфигурации, который загружается во внутреннюю базу глобальных настроек проектов приложения перед использованием. Приложение Валидатор/загрузчик инженерных данных может работать с несколькими настройками проекта (например, для проверки соответствия

разным версиям требований на кодирование проектной документации или для работы с разными проектами). При работе с приложением пользователь осуществляет выбор глобальных настроек из загруженных в базу.

Все наименование блоков (полей), используемые в файлах конфигурации регистро-зависимые и должны писаться строго в соответствии с данной документацией.

Файл глобальных настроек проекта состоит из следующих блоков (полей) описания проекта:

- name – наименование проекта
- info – свойства конфигурационного файла настроек проекта
- check_rules – ссылка на соответствующий данному проекту конфигурационный файл с набором правил проверки пакета
- global_value_parameters – свойства проекта.

Каждый блок должен присутствовать в конфигурационном файле. Все блоки обязательные.

3.4.1.1. Поле name

name - текстовое поле, содержащее наименование проекта.

3.4.1.2. Поле info

info – объект, описывающий характеристики конфигурационного файла проекта.

Объект info должен содержать следующие поля:

- type – текстовое поле, определяющее тип конфигурационного файла. Должно содержать значение “global_project_setting”, определяющее что данный файл является файлом глобальных настроек проекта.
- id – текстовое поле, содержащее уникальный идентификатор файла конфигурации,
- version – текстовое поле, определяющее версию структуры файла конфигурации.
- date – текстовое поле содержащее дату формирования конфигурационного файла в формате "ГГГГ-ММ-ДД",
- description – текстовое поле с кратким описанием данного файла конфигурации. Значение данного поля отображается в интерфейсе приложения Валидатор/загрузчик инженерных данных.

3.4.1.3. Поле check_rules

check_rules – объект, указывающий на соответствующий данному проекту конфигурационный файл с набором правил проверки пакета.

Объект check_rules должен содержать следующие поля:

- id – текстовое поле, определяющее уникальный идентификатор набора правил проверки пакета ПСД, соответствующий данному проекту,
- version – текстовое поле, содержит версию файла с набором правил проверки пакета. Поле является информационным и приложением в текущей версии не используется,
- description – текстовое поле, содержащее краткое описание связанного набора правил.

Приложение Валидатор/загрузчик инженерных данных на основании уникального идентификатора набора правил (поле “id”) автоматически выполняет поиск связанного с данным проектом набора правил. Возможность выбора набора правил у пользователя в интерфейсе приложения отсутствует.

3.4.1.4. Поле global_value_parameters

global_value_parameters – массив объектов, содержащий описание свойств проекта. Каждый объект массива global_value_parameters должен содержать следующие поля:

- Название блока – текстовое поле, определяющее название параметра. Может принимать строго определенные значения.
- description – текстовое поле с наименованием атрибута,
- value – текстовое поле, значение атрибута проекта.

3.4.1.5. Пример файла конфигурации

В общем виде файл глобальных настроек проекта выглядит следующим образом:

```
name: "ПНР-ПИ--20"
info:
  type: "global_project_setting"
  id: "M-01.03.03.07-12-v.2-modif-ESTIMATE-Z"
  min_app_version: "2.2.0.0"
  version: "1.4.0.3"
  date: "2021-11-23"
  description: "Модифицированные настройки. Общий набор настроек для входного
контроля"
check_rules:
  id: "M-01.03.03.07-12-v.2-modif-ESTIMATE-
Z"
  version: "1.4.0.2"
  description: "Используемые правила"
global_value_parameters:
  - project_name
       description: "Название проекта"
       value: "Типовой набор настроек для проверки проекта"
  - chg_authorization_num_param
```

```
      description: ""
      value: "595-19"
  - OL_TT_SI_inside_RD
      description: ""
      value: true
  - example_param_1
      description: "Числовой параметр"
      value: "00"
  - proj_org_name"
      description: "Название проектной организации"
      value: "АО \"Проектный институт\""
  - stage
      description: "Перечень допустимых стадий проектирования"
      value_set: ["ПД", "РД", "ОТР", "ЗУР", "КД", "ИИ", "ОЛ/ТТ/СИ"]
  - capital_obj_tag
      description: "Допустимые коды капитального проекта"
      value_set: []
  - view_only_ext"
      description: "Перечень форматов просмотра"
      value_set: ["PDF", "pdf"]
  - statuses"
      description: "Перечень допустимых состояний"
      value_set: ["IFR", "IFQ", "IFI", "FCA", "IFD", "AFD", "IPO", "AB", "IFC",
"Аннулированный/Отмененный документ"]
  - large_projects
      description: "Допустимые коды проекта"
      value_set: []
  - capital_projects
      description: "Допустимые коды проекта"
      value_set: []
  - proj_org_codes
      description: "Допустимые коды проектных организаций"
      value_set: []
```

3.4.2. Правила проверки пакетов

Правила проверки пакетов приложения Валидатор/загрузчик инженерных данных конфигурируемы и описываются в отдельном файле конфигурации правил проверки, который загружается во внутреннюю базу наборов правил приложения перед использованием. Приложение может работать с несколькими наборами правил (например, для проверки соответствия разным версиям требований на кодирование проектной документации). Выбор правил проверки осуществляется приложением автоматически без участия пользователя на основании параметров, указанных в глобальных настройках проекта.

По умолчанию все выполняемые проверки являются обязательными, т.е. при невыполнении условия проверки отображается ошибка, препятствующая формированию пакета для отправки. При этом администратором любая проверка может быть переведена в режим «предупреждение» - при невыполнении условия проверки отображается предупреждение, не препятствующее формированию пакета ПСД. Для

этого необходимо в объекте описания проверки добавить поле "error_type" установленным в значение "Warning". Фрагмент конфигурационного файла с данным полем приведен ниже:

```
checks:
  - type: “regex”
    regex: "[А-Я]{2,6}(Ф\\d|Ф\\d\\.\\d)",
    match: false,
    error: "Недопустимый формат кода крупного проекта",
    error_type: "Warning"
```

Файл с правилами проверки пакета состоит из следующих блоков (полей) описания проекта:

- info – свойства конфигурационного файла правил проверки пакета
- parametrization – содержит список параметров с описанием, используемых для в правила, параметры устанавливаются пользователем через интерфейс приложения, либо через настройки проекта.
- regex_elements - содержит элементы регулярных выражений, используемых для выделения и валидации полей кодировки в имени файла,
- entity_items – содержит шаблоны регулярных выражений, используемые для сопоставления с именами файлов пакета ПСД, содержат статическую часть и подставляемые элементы, сопоставляемые полям,
- dictionaries - содержит словари, используемые для проверки корректности элементов кодирования, проверяемое значение поля должно соответствовать одному из элементов массива, иначе ошибка,
- classes – поддерживаемые классы документов,
- matrix – сожержит матрицы сравнений, используемые для проверки корректности элементов кодирования, проверяется сочетание значений кодировки.
- optional_levels – содержит дополнительные уровни группировки файлов, а также проверки к ним,
- download_list - описание формата листа загрузки.

Каждый блок должен встречаться в конфигурационном файле один раз. Все блоки кроме блоков «optional_levels» и «parametrization» обязательные.

3.4.2.1. Поле info

info – объект, описывающий общие свойства файла проекта.

Объект info должен содержать следующие поля:

- type – текстовое поле, определяющее тип конфигурационного файла. Должно содержать значение “check-rules”, определяющее что данный файл является файлом с правилами проверки пакета ПСД,
- id – текстовое поле, содержащее уникальный идентификатор файла конфигурации. Данный идентификатор используется приложением для поиска соответствующих проекту правил проверки пакета,
- version – текстовое поле, определяющее версию структуры файла конфигурации.
- date – текстовое поле, содержащее дату формирования конфигурационного файла в формате "ГГГГ-ММ-ДД",
- description – текстовое поле с кратким описанием данного файла конфигурации. Значение данного поля отображается в интерфейсе приложения Валидатор/загрузчик инженерных данных.

Пример описания объекта info:

```
info:
    type: check-rules
    id: M-01.03.03.07-12-v.4-modif-Z
    version: 1.6.0
    date: 2025.05.29
    description: Правила проверки ПСД по М-01.07.03.03-12 v.4
```

3.4.2.2. Поле parametrization

Содержит список заваленных параметров, которые могут быть использованы в правилах. Если параметр не описан в данной секции, но используется в настройках проекта или в правилах проверок, то система выдаст ошибку при валидации правил.

Каждый параметр должен содержать следующие поля:

- заголовок блока – название параметра, используется для обращения к параметру,
- role - Определяет возможность параметризации правил из интерфейса приложения возможные значения - project, internal_user, external_user.
- type – тип данных параметра, может принимать значения bool - логический, string - строка, enum - перечисление, decimap – числовой.
- description – строка, человек читаемое описание параметра, отображается в интерфейсе.
- default_value – значение по умолчанию.
- required – логический параметр, определят обязательность параметра при запылении параметров через интерфейс.

3.4.2.3. Поле regex_elements

regex_elements - массив, содержащий элементы регулярных выражений, используемых для выделения и валидации полей кодировки в имени файла. Каждый элемент массива описывает фрагмент кода пакета в имени файла. Данные элементы массива используются в качестве переменных при определении шаблонов имени файла, шаблона обозначения документа.

Каждый объект массива regex_elements должен содержать следующие поля:

- заголовок блока – текстовое поле, содержит идентификатор элемента, используемый в качестве переменной в последующих проверках,
- description - текстовое поле, содержит описание элемента,
- regex - текстовое поле, содержит правило для проверки содержимого элемента. Для формирования правила используются регулярные выражения;
- check - массив, содержащий набор правил проверки содержимого элемента на его соответствие установленным проверкам. Каждый объект массив содержит следующие поля:
    - type
        - regex - текстовое поле, определяет, что проверка содержимого элемента будет осуществляться с использованием регулярных выражений,
        - dictionary - текстовое поле, определяет, что проверка содержимого элемента будет осуществляться с использованием словаря
    - match – логическое значение, определяет тип проверки. При установке значения в true в случае соответствия значения элемента регулярному выражению приложение выдает ошибку, при установке значения в false - ошибка возникает в случае несоответствия значения элемента регулярному выражению;
    - conditions – Логические выражения, которые определяют будет ли применяться указанная проверка. Если результат - true - проверка выполняется. Подход формирования логические операторов предполагает, что сначала идет логически оператор, а затем операнды. Логические операторы:
        - AND - логическое И
        - OR - логическое ИЛИ
        - NOT - логическое НЕ
        - Унарные операторы для атрибутов:

- EQ - equal, совпадение атрибута
- IN - вхождение в приведенный массив
- REGEX - соответствие REGEX выражению
- DICT - вхождение значения в передаваемый словарь

- error - текстовое поле, содержит текст сообщения об ошибке,
- dictionary_name - текстовое поле, содержит название словаря, описанного в данном конфигурационном фале, для проверки по нему содержимого элемента,
- error_type - текстовое поле, применяется в случае, если проверяемое поле является пустой строкой. Содержит "Error" в случае, если данная проверка обязательна, содержит “Warning” в случае, если данной ошибкой можно пренебречь. В случае наличия ошибки приложение выводит сообщение об ошибке. Для обязательных проверок отображается красный кружок, означающий ошибку, для необязательных выводится желтый треугольник - предупреждение.

Пример описания массива regex_elements приведен в следующем блоке:

```
"regex_elements":
land_part_code:
    description: "Шифр комплекта ЗУР"
    regex: "[^-_\\d]{1,6}"
    type: "string"
    checks:
     - type: "regex"
       regex: "^[А-Я]{3,4}$"
       match: false
       error: "Недопустимый формат шифра комплекта ЗУР"
       required: true
     - type: "dictionary"
       dictionary_name: "land_part_codes"
       error: "Код раздела по ЗУР отличается от общепринятых"
       error_type: "Error"
       required: true
     - type: "regex"
       regex: "[a-zA-Z]"
       match: true
       error: "Использование латинских букв в шифре комплекта не допускается"
       required: true
```

Здесь описывается один элемент с именем (поле id) large_prj_code, в котором должен храниться код крупного проекта.

В поле regex верхнего уровня задан шаблон: "[^-_\\d]{1,6}". Для данного элемента определен определён массив check, состоящий из трех объектов, каждый из которых описывает одно правило проверки:

- Проверка с параметром regex и шаблоном ^[А-Я]{3,4}$ проверяет, что строка состоит только из 3 или 4 заглавных букв русского алфавита. Например, под

данный шаблон подходят аббревиатуры ТЗФ или АВТО. При несовпадении содержимого данного элемента указанному шаблону (поле match равно false) приложение выдаст ошибку, препятствующую формированию пакета для отправки;

- Проверка с параметром dictionary говорит, что проверка будет вестись по словарю с именем land_part_codes. В случае отсутствия в данном словаре содержимого элемента приложение выдаст ошибку, препятствующую формированию пакета для отправки.
- Проверка с параметром regex и шаблоном [a-zA-Z] определяет, что если в содержимом элемента имеются буквы английского алфавита (поле match равно true) то вывести ошибку препятствующую формированию пакета для отправки.

3.4.2.4. Поле entity_items

entity_items - массив, содержащий шаблоны регулярных выражений, используемые для сопоставления с именами файлов, расположенными в папке с пакетом ПСД. Шаблон представляет собой строку – заготовку регулярного выражения, включающую в себя постоянную часть и символы подстановки ${имя_элемента_регулярного_выражения}, вместо которых подставляются значения «regex» из объектов regex_element, имя которого указано в фигурных скобках соответствующей группы подстановки.

Приложение Валидатор/загрузчик инженерных данных на основании указанных правил определяет тип документа (документация, накладная, лист загрузки и т.д.), определят стадию проектной документации, выполняет проверку на соответствие других кодов в имени файла данной стадии файла проекта. В результате проверки файлов, на основании указанного шаблона, приложение формирует поле «обозначение документа». Данное поле также используется для группировки файлов при формировании листа загрузки.

Для каждого типа документов, описанного в «entity_items» приложение формирует «длинное» регулярное выражение, путем подстановки указанных элементов регулярных выражений в постоянную часть строки шаблона.

Для всех имен файлов, находящихся в указанной папке, выполняется последовательное сопоставление со всеми допустимыми шаблонами. Если совпадение есть, приложение прекращает сопоставление с шаблонами и принимает тип документа, советующий шаблону регулярного выражения имени файла (например: «ИИ», «ЗУР» и т.д.).

Если совпадений имени файла ни с одним шаблоном нет, приложение выведет для данного файла ошибку.

Для всех групп соответствия, описанных в блоке regex_elements, совпавшего регулярного выражения извлекается значение (последовательность символов), выполняются все указанные для данного элемента в массиве «check» проверки. Если проверки успешны, полученное значение приравнивается значению соответствующего элемента массива regex_elements.

Если все проверки для элементов пройдены, выполняем проверки для всего имени документа, указанные в советующем entity_items.

При завершении всех проверок приложением формируется поле “обозначение документа“.

Объекты массива regex_elements должны содержать следующие поля:

- заголовок блока - поле с название сущности. Наименование полей могут быть произвольными. На данный момент используются текстовые doc_id, doc_id_full, doc_type_pack. Для формирования значения полей могут использоваться как произвольные значения, так и элементы имени файла, используемые в шаблоне файла. Пример заполнения: "doc_type_pack": "КЧ", "doc_id": "${large_prj_code}-${capital_prj_code}-ЗУР-${land_part_code}${land_part_num}-${doc_type}-${doc_number}"
  Ключевые слова, указанные в фигурных скобках (large_prj_code, capital_prj_code и т.д.) являются идентификаторами элементов, описанных в разделе *regex_elements*;
- description - текстовое поле, содержит описание для данного правила проверки,
- filenames - текстовое поле, содержит шаблон имени файла, проверяемый на соответствие приложением, может содержать несколько имен. Пример заполнения поля: ${code}-${status _code}-ПУР-${main_code}${part_num}-${doc_type}-${doc_number}_${revison_status}\\.${file_ext}
  Аналогично полю doc_id, ключевые слова, указанные в фигурных скобках шаблона (code, status _cod и т.д.) являются идентификаторами элементов, описанных в разделе *regex_elements*;
- attributes – список атрибутов присваиваемых данной сущности, также в атрибутами считаются элементы regex_elements из которых формируется filenames.
- check - массив объектов, содержит дополнительные правила проверки имени файла. Выполняется при соответствии имени файла и шаблона в поле filename_template. Поле может отсутствовать, или содержать null. В таком случае никакие дополнительные проверки не выполняются. Каждый объект массива check должен содержать следующие поля:

- type
  - regex - текстовое поле, определяет, что проверка содержимого элемента будет осуществляться с использованием регулярных выражений,
  - dictionary - текстовое поле, определяет, что проверка содержимого элемента будет осуществляться с использованием словаря
  - matrix - проверка корректности наличия значений для двух атрибутов. Матрицы находится в блоке matrix.
- Attribute - текстовое поле, содержит наименование атрибута, для которого необходимо выполнить проверку

– conditions – Логические выражения, которые определяют будет ли применяться указанная проверка. Если результат - true - проверка выполняется. Подход формирования логические операторов предполагает, что сначала идет логически оператор, а затем операнды. Логические операторы:
  - AND - логическое И
  - OR - логическое ИЛИ
  - NOT - логическое НЕ
- Унарные операторы для атрибутов:
  - EQ - equal, совпадение атрибута
  - IN - вхождение в приведенный массив
  - REGEX - соответствие REGEX выражению
  - DICT - вхождение значения в передаваемый словарь
- row - текстовое поле, содержит наименование проверяемого атрибута, необязательное поле, заполняется только в при типе matrix,
- matrix - текстовое поле, содержит наименование матрицы используемой для проверки, необязательное поле, заполняется только в при типе matrix,
- dictionary_name - текстовое поле, содержит наименование словаря используемой для проверки, необязательное поле, заполняется только в при типе matrix,
- match - логическое значение, определяет тип проверки. При установке значения в true в случае соответствия значения элемента регулярному выражению приложение выдает ошибку, при установке значения в false - ошибка возникает в случае несоответствия значения элемента регулярному выражению;
- error – текстовое поле, содержит текст сообщения об ошибки,

- error_type - текстовое поле, применяется в случае, если проверяемое поле является пустой строкой. Содержит "Error" в случае, если данная проверка обязательна, содержит “Warning” в случае, если данной ошибкой можно пренебречь. В случае наличия ошибки приложение выводит сообщение об ошибке. Для обязательных проверок отображается красный кружок, означающий ошибку, для необязательных выводится желтый треугольник - предупреждение.

Пример описания массива entity_items приведен в следующем блоке:

```
Документ_РД:
      description: "Рабочая документация"
      filenames:
        - "^${project_phase}-${building_code}-${section_type}-${drawing_cat}-
${revision_status}\\.${file_ext}$"
      attributes:
     doc_id: "${project_phase}-${building_code}-${section_type}-${drawing_cat}"
doc_id_full: "${project_phase}-${building_code}-${section_type}-${drawing_cat}-${revision_status}"
        stage: "РД"
        class: "Файл"  # new
        sub_class: "Doc_rd"  # new
      checks:
        - attribute: building_code
          type: matrix
          row: doc_type
          error: "Сочетание типа документа и дисциплины проектирования отличаются от
рекомендуемых"
          matrix: " matrix"
          error_type: "warning"
          required: true
        - attribute: building_code
          type: matrix
          row: doc_type
          error: "Недопустимое сочетание типа документа и дисциплины проектирования"
          matrix: "matrix_asu"
          error_type: "warning"
          required: true
```

В данном примере описан один шаблон имени файла. Если имя файла соответствует заданному в поле filename_template шаблону, то выполняются проверки элементов, указанных в шаблоне: project_phase, building_code, section_type и т.д.

После проверки элементов выполняется проверка из блока check: проверяется соответствие поля doc-type (тип документа) полю complect_discipline (дисциплина проекта) с использованием значений из словаря disc-compat_matrix. В случае отсутствия в словаре соответствия указанным полям, приложение выведет сообщение об ошибке.

Полученному документа присваивается обозначение документа, в соответствии с шаблоном из поля doc_id.

3.4.2.5. Секция dictionaries

Блок описания dictionaries содержит описание используемых приложением словарей, используемых на разных стадиях проверки. Приложение использует два вида словарей, описанные в разделах plain.

В разделе plain содержатся словари, состоящие из одномерного массива. Проверяемое по словарю значение поля должно соответствовать одному из элементов массива. Пример словаря: "codes": ["ААЗУ", "ППТ", "ПМТ", "ПУК "АВМ"]. В данном примере словарь codes содержит массив из возможных шифров.

3.4.2.6. Секция matrix

В разделе matrix содержаться словари, представляющие собой двумерные массивы т.е. таблицу. Данные словари содержать несколько ключей (строка таблицы), каждому ключу соответствуют массив значений. Фрагмент такого словаря приведен ниже:

```
matrix:
  ССТ: [АК, "ПС],
  СТХ: [ТХ],
  СМТ: [ТХ, ТК, АК]
```

Словарь matrix используется приложением для проверки сочетания двух атрибутов. Фрагмент содержит три ключа ССТ, СТХ и СТМ. Каждому ключу соответствует массив возможных сочетания с другим атрибутом.

3.4.2.7. Секция optional_levels

Секция optional_levels представляет собой массив, содержащий описание проверяемых приложением дополнительных сущностей. Сущности формируются из сущностей низшего уровня сформированных в секции entity_items Для каждой новой сущности могут быть добавлены новые проверки. Секция представляет собой список сущностей, где для каждой необходимо заполнить следующие поля:

- заголовок блока - текстовое поле, содержит наименование группировочной сущности:
- from_entities - текстовое поле, содержит название сущности которые группируются в сущность данного уровня, может быть указанно сущность уровня optional_levels или низшая сущность entity_items.
- entities – Промежуточный уровень, внутри которого содержатся типы данного уровня группировки. Содержит экземпляры данной сущности. Например, в качестве сущности optional_levels может быть сформирована сущность documents, а в качестве промежуточных уровней entities могут быть документы: Накладная, смета и т.д.Для каждая сущность на промежуточном уровне необходимо заполнить:

- conditions - Признаки по которым выполняется отнесение сущностей предыдущего уровня к типу сущности на текущем уровне.В блоке conditions могут применяться логически конструкции подход формирования логические операторов предполагает, что сначала идет логически оператор, а затем операнды. Логические операторы:
  - AND - логическое И
  - OR - логическое ИЛИ
  - NOT - логическое НЕ
  - Унарные операторы для атрибутов:
  - EQ - equal, совпадение атрибута
  - IN - вхождение в приведенный массив
  - REGEX - соответствие REGEX выражению
  - DICT - вхождение значения в передаваемый словарь
- group_by - Блок определяет признаки по которым сущности предыдущего уровня группируются на текущем уровне.
- attributes – список атрибутов, который присваивается текущей сущности. Примеры определения атрибута:
  - large_prj_code: from_child
  - nakl_date: data_source.nakl_date
  - revision_code:params.params1

Значение from_child означает что атрибут устанавливается согласно значению у дочерних сущностей. Если значение данного атрибута у дочерних сущностей разное, то система вернет ошибку. data_source – значение источника данных, на данный момент источником данных является интерфейс запускаемого приложения. paprams. – означает что значение должно быть взято из параметров название которого указанно после «paprams.»

- checks - массив, содержащий коллекцию объектов, описывающих проверки, выполняемые над документом после сопоставления. Данный блок аналогичен блоку checks применяемому к остальным сущностям или regex_elements..

3.4.2.8. Секция download_list

Секция содержит информацию о структуре и проверках листа загрузки, а также содержит полный список атрибутов с описанием и маппингом.

Секция содержит следующие блоки:

- download_list_attributes – содержит список атрибутов с описанием
- download_list_config - информацию о структуре и проверках листа загрузки

Блок download_list_attributes содержит два следующих блока:

- downloadlist_requred_attributes - блок содержит перечисление атрибутов, которые должны быть обязательно записаны в лист загрузки в самом его начале (и в XLSX и в JSON), в том порядке, который указан в конфиге. При этом, это не означает, что этот атрибут будет заполнен у загружаемой сущности.
- download_list_attribute_mapping – блок содержит полный список атрибутов, где для каждого атрибута указан код используемый в правилах проверки и название атрибута используемое при записи в лист загрузки.

Для каждого атрибута из секции download_list_attribute_mapping должны быть заполнены следующие поля:

- attribute – код атрибута, используемый в правилах проверки,
- name – название атрибута, используемое при записи в лист загрузки
- column – столбец Excel файла в который записывается данный атрибут
- source – источник получения значения атрибута. Если значение установлено как user, то данный атрибут может быть заполнен пользователем через интерфейс и можно указать дополнительную валидацию значения, введенного пользователем. Валидация выполняется с помощью необязательного блока проверок check. Блок check аналогичен блоку check описанному выше.

Блок download_list_config содержит описывает структуру и дополнительные проверки при формировании листа загрузки и самого листа загрузки. Блок представляет собой массив сущностей, которые будут включены в лист загрузки, данные сущности должны быть указан в секции optional_levels, но если блок optional_levels отсутствует, то система может сформировать лист по низшей сущности entity_items. Для каждой сущности из секции optional_levels так указываются типы промежуточных сущностей из секции optional_levels. Для каждой промежуточной сущности из блока download_list_config должны быть заполнены следующие поля:

- requred_attributes – массив названий атрибутов, которые обязательно должны быть заполнены у сущности
- optional_attributes – массив названий атрибутов, которые опционально могут быть заполнены у сущности
- onloadcheck - перечень проверок, применяемых при проверке пакета содержащего сформированный ЛЗ
    - На данный момент используется одна проверка other_column_must_be_empty которая указывает, что колонки листа загрузки в формате XSLX должны быть пусты, кроме тех что указаны в requred_attributes и optional_attributes.

- Для ошибки необходимо указать текст в поле error.

Пример заполнения сущности в блоке download_list_config:

```
documents:
    types:
     - Документ:
       mapping_attributes:
        stage: РД
       required_attributes:
        - class
        - filename
        - project_code
        - project_number
        - section_code
        - doc_number

       optional_attributes:
        - stage
        - doc_id
       onloadcheck:
        other_column_must_be_empty: true
        error: "Заполнены ячейки, не относящиеся к документу стадии РД"
```

3.5. Работа с YAML файлами

YAML (YAML Ain't Markup Language) - простой формат обмена данными, удобный для чтения и написания как человеком, так и компьютером. Это человеко-читаемый формат сериализации данных, который использует отступы и минимальное количество специальных символов..

YAML - текстовый формат, полностью независимый от языка реализации. Он использует интуитивно понятный синтаксис, знакомый пользователям различных языков программирования, таких как Python, Ruby, JavaScript, Java и многих других

YAML файл использует отступы (пробелы) для обозначения структуры данных, а не фигурные или квадратные скобки. В отличие от JSON, YAML не требует обязательного использования кавычек для ключей и строк, а также запятых для разделения элементов.

Значение ключа в YAML можно задать только в одном из следующих форматов:

- строка – значение заключается в кавычки,
- число - значение указывается без кавычек,
- объект – набор пар «ключ: значение», где вложенность обозначается отступами. Запятые между элементами не требуются;
- массивом – упорядоченное множество значений, обозначается дефисами с пробелом (-).. Массив может быть пустым, т.е. не содержать ни одного значения;
- логическим значением true (правда) или false (ложь),
- null - обозначает отсутствие какого-либо объектного значения.

Пример YAML файла приведен в следующем фрагменте:

```
name: Иван
age: 37
mother:
  name: Ольга
  age: 58
children:
  - Маша
  - Игорь
  - Таня
married: true
dog: null
```

В данном файле:

- ключ name имеет строковое значение
- age - числовое значение
- mother - вложенный объект, содержащий пары name и age
- children - массив строковых значений
- married - логическое значение
- dog - пустое значение (null)

Для работы с коллекцией объектов в YAML используются словари (ассоциативные массивы). Словарь представляет собой неупорядоченную коллекцию произвольных объектов с доступом по ключу. Пример простого словаря:

```
divisons:
  - ППРП
  - ГПНЯ
  - МНГ
  - ОООП
  - ВПНВ
  - НЕПР
  - НППР
```

Словарь может состоять из нескольких пар ключей/значение, как в следующем примере:

```
stage-compat_matrix:
  ОТР: ["ПП", "ТТ", "НН"],
  П: ["ОЛ", "ТТ", "НН"]
```

В данном примере словарь stage-compat_matrix содержит два ключа ОТР и П. Для каждого ключа указано несколько значений связанных с ключем.

В отличие от JSON, YAML поддерживает комментарии, которые начинаются с символа «#».

```
key: value  # Комментарий после значения
```

AML использует отступы пробелами (обычно 2) для обозначения вложенности, при этом табуляции для отступов не допускаются, а также не требует обязательного использования кавычек для строк (если они не содержат специальных символов). Этот

формат более удобен для чтения и редактирования человеком благодаря своей наглядной структуре и поддержке многострочных строк с сохранением форматирования.

Для работы с YAML файлами можно использовать различные инструменты: Notepad++ (с подсветкой синтаксиса), Visual Studio Code (с расширениями для YAML), онлайн YAML валидаторы или специализированные IDE (PyCharm, RubyMine и др.). Ключевые преимущества YAML перед JSON включают более читаемый синтаксис, поддержку комментариев, гибкие возможности форматирования и расширенные типы данных (даты, якоря и ссылки), что делает его особенно удобным для работы с конфигурационными файлами.

3.6. Регулярные выражения

Регулярные выражения — это шаблоны, используемые для сопоставления последовательностей символов в строках. Регулярные выражения предоставляют мощный, гибкий и эффективный способ обработки текста. Комплексная нотация сопоставления шаблонов регулярных выражений позволяет быстро анализировать большие объемы текста в следующих целях:

- поиск определенных шаблонов символов;
- проверка текста на соответствие предопределенному шаблону (например, адресу электронной почты);
- извлечение, изменение, замена или удаление текстовых подстрок;
- добавление извлеченных строк в коллекцию для создания отчета.

Регулярное выражение представляет собой текстовою строку, содержащую специальные символы. Описание синтаксиса регулярных выражений, а также перечень доступных специальных символов, доступно по следующей ссылке: *Элементы языка регулярных выражений — краткий справочник (https://docs.microsoft.com/ru-ru/dotnet/standard/base-types/regular-expression-language-quick-reference).*

Сайт с онлайн-инструментом по работе с регулярными выражениями*:* *https://regexr.com* или *regex101.com*.

При использовании символа \ в строках YAML требуется его экранирование — перед ним нужно указать ещё один \. Это важно при работе с регулярными выражениями, где часто применяются служебные последовательности (\d, \w и т. д.), а также символы, совпадающие с обычными символами строки поиска (например, точка в имени файла). В итоге в YAML одна обратная косая черта записывается как \\.

Пример проверяемой строки: АКИ3.1 или АК.1

Пример регулярного выражения: [А-Я]{2,4}(\d)?\.\d

Пример этого выражения в строке YAML с экранированием: [А-Я]{2,4}(\\d)?\\.\\d

В YAML существует несколько способов задания строк (например, в кавычках или блочными литералами), и правила экранирования зависят от выбранного варианта.

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Изм	*Номера листов (страниц)*				*Всего листов (страниц) в докум.*	*№ документа*	*Входящий № сопроводи-тельного документа*	*Подпись*	*Дата*
	изменен-ных	*заменен-ных*	*новых*	*аннулиро-ванных*					