УТВЕРЖДЕН
ДССЛ.00111-01 31 01 - ЛУ

СПЕЦИАЛЬНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«3i NLP Platform»

Описание применения

ДССЛ.00111-01 31 01

Листов 75

Литера $О_1$

2015

АННОТАЦИЯ

Настоящий документ предназначен для ознакомления со специальным программным обеспечением (СПО) «3i NLP Platform» и содержит описание интерфейса программирования (API) для программистов, обеспечивающих использование 3i NLP Platform в качестве модуля, встраиваемого в другое программное обеспечение.

В разделе 1 приводятся сведения о назначении 3i NLP Platform.

В разделе 2 указаны требования к программно-техническим средствам, необходимым для работы 3i NLP Platform.

В разделе 3 указывается описание задач, решаемых 3i NLP Platform, даются сведения об используемых технологиях.

В разделе 4 даются сведения об установке 3i NLP Platform.

В разделе 5 приводится описание интерфейса программирования 3i NLP Platform.

В разделе 6 даются сведения о входных и выходных данных 3i NLP Platform.

В разделе 7 приводятся основные сообщения оператору при работе с 3i NLP Platform.

По всем вопросам, связанным с использованием СПО «3i NLP Platform» можно обращаться по электронной почте support@dss-lab.ru или по телефону +7 (495) 645-44-70 по будним дням с 10 до 18 часов, время московское.

СОДЕРЖАНИЕ

1. НАЗНАЧЕНИЕ ПРОГРАММЫ

Специальное программное обеспечение «3i NLP Platform» предназначено для решения задач обработки текстовых данных. 3i NLP Platform используется в качестве отдельного модуля, встраиваемого в другое программное обеспечение, предоставляя разработчику соответствующий функционал API. API (сокр. англ. Application Programming Interface, интерфейс программирования приложений, интерфейс прикладного программирования) – набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением для использования во внешних программных продуктах.

API 3i NLP Platform относится к классу REST (сокр. англ. Representational State Transfer, передача репрезентативного состояния) – метод взаимодействия компонентов распределённого приложения в сети Интернет, при котором вызов удаленной процедуры представляет собой обычный HTTP-запрос (обычно GET или POST; такой запрос называют REST-запрос), а необходимые данные передаются в качестве параметров запроса. Этот способ является альтернативой более сложным методам, таким как SOAP, CORBA и RPC.

К основным преимуществам приложения, предоставляющего HTTP REST API в качестве инструмента доступа к функциональным возможностям, можно отнести:

- надёжность (за счет отсутствия необходимости сохранять информацию о состоянии клиента, которая может быть утеряна);
- производительность (за счет использования кэша);
- прозрачность системы взаимодействия;
- легкость внесения изменений;
- масштабируемость.

2. УСЛОВИЯ ПРИМЕНЕНИЯ

2.1. Для функционирования 3i NLP Platform необходима вычислительная система, параметры которой не хуже:

– CPU частотой 2.4 ГГц (8 физических вычислительных ядер на 1 процессор);

– ОЗУ 16 ГБ;

– LAN 10 Мбит/с;

– дисковая подсистема совокупной емкостью не менее 1 Тб.

2.2. Для функционирования 3i NLP Platform на вычислительной системе должно быть установлено следующее общее программное обеспечение:

– операционная система – Linux CentOS 7.15;

– пакет типового программного обеспечения операционной системы семейства Linux (post-install) – «Everything» для Linux CentOS 7.15 (http://isoredirect.centos.org/centos/7/isos/x86_64/CentOS-7-x86_64-Everything-1503-01.iso);

– пакет библиотек C++ – Poco 1.6 или выше;

– пакет компонент поддержки библиотек C++ для Юникод – ICU-56.1 или выше.

3. ОПИСАНИЕ ЗАДАЧИ

3.1. Технологии обработки текстовых данных

3i NLP Platform предназначено для решения задач обработки структурированной и неструктурированной текстовой информации. Возможности 3i NLP Platform базируются на современных технологиях обработки естественного языка NLP (сокр. англ. Natural Language Processing) – общего направления искусственного интеллекта и математической лингвистики, изучающего проблемы компьютерного анализа и синтеза естественных языков.

3i NLP Platform на входе получает текст, а на выходе представляет его в виде структурированного набора данных. В ходе выполнения текстового анализа выявляется большой объем различной лингвистической информации. При поиске и классификации текстов учет этих параметров позволяет сильно повысить точность и релевантность результатов.

3i NLP Platform обеспечивает обработку поступающей информации «на проходе», позволяя обогащать ее дополнительными данными:

- морфологическая информация (часть речи, падеж, род, число и др.), предсказание морфологических форм для неизвестных слов;
- выявление сущностей (персоны, геолокации, организации);
- расчет тональностей для документов и сущностей.

Для выполнения эффективного анализа текстовой информации 3i NLP Platform используются развитые лингвистические ресурсы (словари, правила распознавания сущностей, сентиментные правила).

Технологии распознавания именованных сущностей

Именованная сущность – отдельное слово, соответствующее тому или иному смысловому классу. Под именованными сущностями, как правило, понимают достаточно устоявшиеся в информационном пространстве классы

текстовых данных (персоны, организации), в отношении которых предполагается проводить исследования или устанавливать мониторинг. Задача распознавания именованных сущностей сводится к отнесению фрагмента текста к одному из заранее заданных классов. Для распознавания именованных сущностей используются словари и правила, основанные на орфографическом написании слова или нескольких слов подряд. Логическая информационная модель именованной сущности представлена на рисунке 3.1.1.

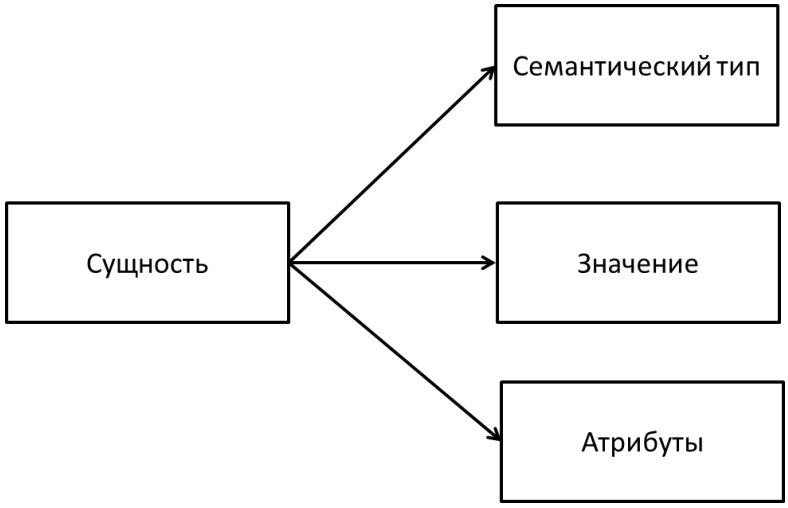

Рисунок 3.1.1 – Логическая информационная модель именованной сущности

Сущность – обобщенное название смысловой единицы определенного вида из текста.

Семантический тип – заранее заданный смысловой класс, к которому отнесена выделенная сущность.

Значение – часть исходного текста, которая соответствует заданным критериям.

Атрибуты – тип сущности (обобщение над семантическими типами), грамматическая информация.

Например, из текста может быть выделена следующая сущность:

Семантический тип: Имя Персоны

Значение: Иван Иванов

Атрибуты: тип сущности – Персона, грамматическая информация: нормальная форма – «Иван Иванов», орфографический тип – с заглавной буквы, позиция в предложении – не начало, часть речи – имя существительное, род – мужской, число – единственное, падеж – именительный.

В морфологических словарях словам, обозначающим имена, фамилии, отчества ставится соответствующий признак, который затем может быть преобразован в семантический тип. Во-первых, происходит обработка неизвестных слов с окончаниями и суффиксами, совпадающими с регулярными окончаниями и суффиксами фамилий. Сначала исключаются слова, контекст которых и написание в нижнем регистре говорят о том, что скорее всего они являются неизвестными именами прилагательными, а не фамилиями. Затем оставшиеся неизвестные слова, оканчивающиеся на регулярные суффиксы/окончания фамилий (-ов/-ова, -ев/-ева, -ин/-ина и некоторые другие в различных числах и падежах) получают соответствующее значение семантического типа. Затем происходит сбор в одну сущность сочетаний типа «господин Фамилия Имя Отчество», «мистер И.О.Фамилия» в различных сочетаниях и сокращениях.

Географическим названиям соответствует свой тег, который затем преобразуется в географические семантические типы. Имена существительные в морфологических словарях регистрозависимы: географические названия в морфологии могут получаться только из слов написанных с заглавной буквы.

Названиям организаций также соответствует свой тег, который затем преобразуется в семантические типы названий организаций. Названия организаций также могут получаться только из слов написанных с заглавной буквы.

Технологии распознавания тональности

Анализ тональности высказываний (сентиментный анализ) подразумевает классификацию предложений на нейтральные, позитивные и негативные. Под

«тональностью» понимается авторская оценка объекта или ситуации, выраженная в тексте. Сентиментный анализ традиционно происходит на основе выявления ключевых сентиментных слов и применения правил, которые модифицируют значение сентимента ключевого слова, или добавляют новое значение сентимента. В качестве ключевых слов используются слова, которые несут сентиментную окраску в любом контексте, или в подавляющем большинстве контекстов. В список ключевых сентиментных слов входят «хороший», «плохой», «неприятный», «трус», «успех», «провал», «угроза», «позитив», «оперативно», «своевременно», «слишком», а также многие другие. Если слово из этого списка встречается в тексте, система приписывает ему соответствующую тональность (см. Таблицу 3.1). Вышеозначенный подход к анализу сентиментов предложений используется для оценки сентимента текста в целом.

Таблица 3.1 Пример оценки тональности по ключевым словам

Предложение	**Экспертная оценка (+/-)**
Вася трус.	-
Зарплату платят с задержкой в 2-3 месяца	-

Сентиментные правила, в свою очередь, приписывают значение тональности, основываясь на сочетании определенных форм слов или синтаксической связи между ними. Отдельные выявленные именованные сущности также получают сентиментальную оценку.

Могут использоваться правила, модифицирующие сентимент ключевых слов. Например, при отрицании слова его значение в некотором смысле меняется на противоположное. При отрицании в сочетании с ключевым сентиментным словом его сентимент получает противоположное значение: если слово несет в себе позитив, то отрицание модифицирует его на негатив, и наоборот. Отрицание может выражаться несколькими способами. Основной способ выражения отрицания – это частица «не» и предикатив «нет». Также важную роль играют

такие слова, как «отсутствие, удаление, лишение, отрицание, устранение, отсутствовать, удалять, лишать, отрицать, устранять», и предлог «без». В сочетании «не хорош» сначала ключевое слово «хороший» получит положительный сентимент, а затем благодаря сочетанию с частицей «не» система припишет отрицание слову «хороший», которое в результате поменяет позитивный окрас на противоположный – негативный.

Обработка неизвестных слов

В текстах встречается до 3% неизвестных слов в зависимости от тематики. В специализированных технических, экономических текстах этот процент самый высокий, в то время как в популярных текстах он ниже. Иногда неизвестные слова из-за их редкой встречаемости не имеет смысла сохранять в словаре. Часто окончание или суффикс слова указывают однозначно на его грамматические характеристики, т.е. часть речи и другие атрибуты, необходимые для корректного синтаксического и семантического анализа. Особенно характерно это для т.н. «продуктивных» суффиксов – таких, с помощью которых порождаются или заимствуются новые слова в языке. Также новые слова часто формируются за счет присоединения продуктивных приставок. Такие приставки как «ультра-, супер-, экс-, гос-, микро-» не меняют грамматических характеристик слова и могут присоединяться к неограниченному списку слов. Поэтому нет смысла сохранять слова с такими приставками в морфологическом словаре: правильнее будет «отбрасывать» такую приставку и искать оставшееся исходное слово в морфологическом словаре.

Обработка неизвестных слов строится на основе «угадывания» грамматических атрибутов неизвестных слов в зависимости от окончания. Словоформы, для которых не нашлось точного соответствия в морфологии, распознаются по суффиксам и окончаниям. Таким образом, несмотря на ошибки внутри слова, оно будет разобрано в соответствии с несколькими последними буквами. Возможен вариант распознавания неизвестных слов с приставками. Если словоформа в тексте отличается от какой-то из словоформ в морфологическом

словаре на приставку, входящую в список таких продуктивных приставок, то она отбрасывается, и словоформе приписываются все грамматические характеристики соответствующей формы их словаря.

3.2. Структура 3i NLP Platform

Обобщенная структурная схема 3i NLP Platform представлена на рисунке 3.2

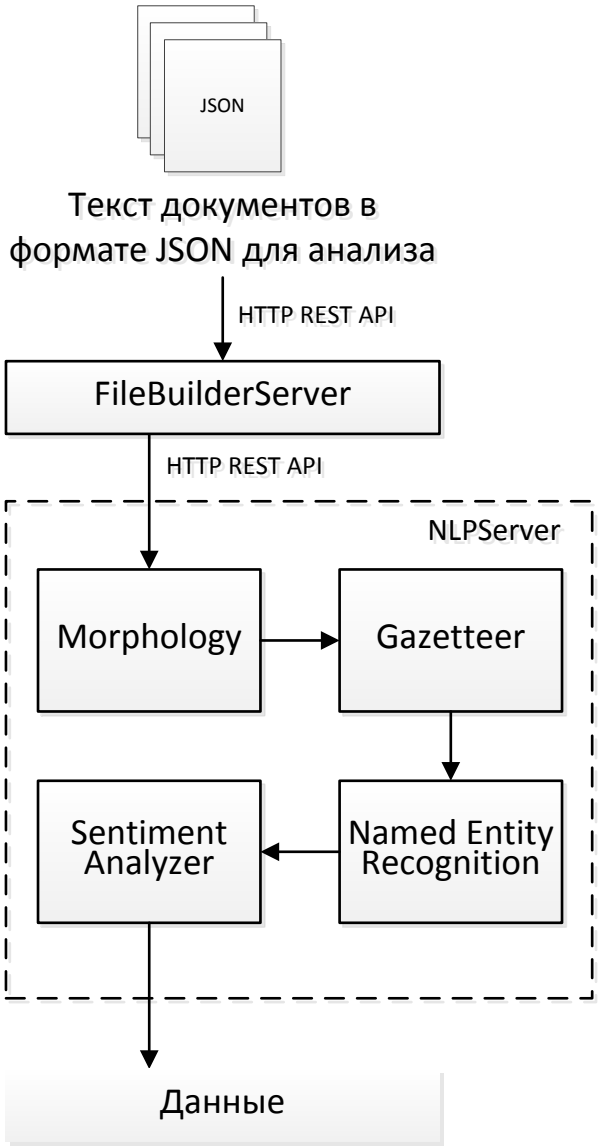

Рисунок 3.2 – Обобщенная структурная схема 3i NLP Platform

Структура 3i NLP Platform включает следующие компоненты:

– FileBuilderServer – обеспечивается сбор и импорт данных в систему из файловой сети или внутренних серверов. Основные функциональные возможности: автоматическое определение кодировки, перекодировка в UTF-8, определение языка документов, определение формата файла, выделение мета-данных, структуризация документов;

– PostProcessServer – обеспечивается обработка текстовых данных из поступающих в систему файлов, содержит следующие модули:

- Morphology – приведение термов текстовых данных к нормальной форме, предсказание морфологических форм;
- Gazetteer – добавление служебной семантической информации, определение кандидатов в сущности из текстовых данных;
- Named Entity Recognition – создание и обработка сложных (составных) сущностей;
- Sentiment Analyzer – обработка тональности для сущностей (простые и составные) и документов.

Технологическая цепочка обработки текстовых данных представлена на рисунке 3.3.

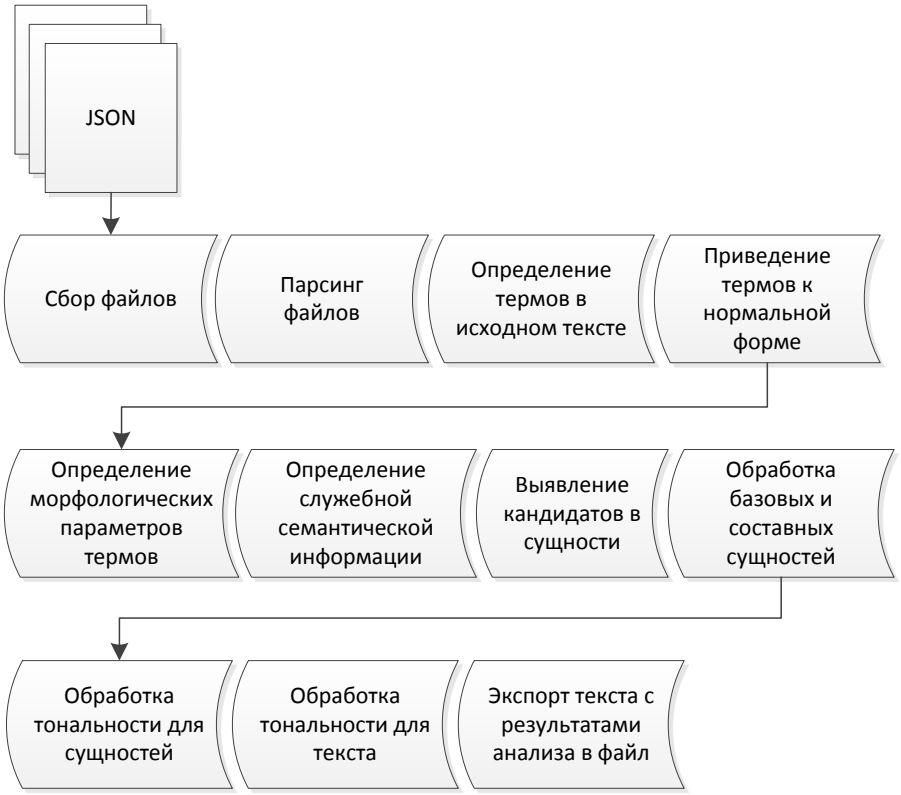

Рисунок 3.3 – Технологическая цепочка обработки текстовых данных

4. ВЫЗОВ И ЗАГРУЗКА ПРОГРАММЫ

Службы, необходимые для функционирования 3i NLP Platform, запускаются автоматически при старте серверов.

4.1. Установка 3i NLP Platform

Установка 3i NLP Platform осуществляется переносом (копированием) архива с исполняемыми файлами с загрузочного диска в требуемую директорию на жестком диске сервера. После переноса (копирования) архива исполняемые файлы 3i NLP Platform разархивируются из него в требуемую директорию на жестком диске сервера.

4.2. Конфигурирование компонент 3i NLP Platform

Конфигурирование компонент 3i NLP Platform выполняется пользователем путем внесения изменений в соответствующие файлы конфигураций каждой установленной компоненты. Настройки компонент представлены в Таблице 4.1. Некоторые настройки приложений скрыты от пользователя и осуществляются автоматически.

Таблица 4.1 Настройки компонент 3i NLP Platform

№ п/п	Наименование компоненты	Перечень настроек
1.	FileBuildServer	Задается IP-адрес компоненты и порт, а так же следующие настройки необходимые для ее работы: build_thread_number –количество потоков обработчиков. output_server_list – список серверов(PostProcessServer), которым будут отправлены обработанные приложением

№ п/п	Наименование компоненты	Перечень настроек
		FileBuildServer данные для дальнейшей обработки.
2.	PostProcessServer	Задается IP-адрес компоненты и порт, а так же следующие настройки необходимые для ее работы: BuildThreadNumber–количество потоков обработчиков. NetworkServerList – список серверов, которым будут отправлены обработанные приложением PostProcessServer данные для дальнейшей обработки. Проверяется наличие лингвистических ресурсов для каждого модуля компоненты в папке resources в требуемой директории на жестком диске сервера, в которой располагаются исполняемые файлы 3i NLP Platform. В папке resources присутствуют отдельно ресурсы для каждого поддерживаемого языка (ru, en, uk и т.д.). В каждой подпапке языка есть папки словарей: dict для Morphology, ner для Named Entity Recognition и gazetteer для Gazetteer.

Дополнительно пользователю в зависимости от требований к обработке входящих данных необходимо изменить настройки в папке конфигурации компонента FileBuildServer для:

– правил обработки данных (парсеров), которые хранятся в файле «Parsers»;

– правил приведения данных (fieldconversions), которые хранятся в файле «FieldConversions».

5. ВЫПОЛНЕНИЕ ПРОГРАММЫ

5.1. Описание функционала API

Функции API 3i NLP Platform условно разделены на три крупных специализированных блока:

– Блок импорта и предварительной обработки данных (FileBuilderServer) – обеспечивается сбор и импорт данных в систему из файловой сети или внутренних серверов. Основные функциональные возможности: автоматическое определение кодировки, перекодировка в UTF-8, определение языка документов, определение формата файла, выделение мета-данных, структуризация документов;

– Блок обработки текстовых данных (PostProcessServer) – обеспечивается обработка текстовых данных из поступающих в систему файлов: приведение термов текстовых данных к нормальной форме, предсказание морфологических форм, добавление служебной семантической информации, выявление сущностей из текстовых данных, обработка тональности;

Далее для каждого блока приведено отдельное описание функций, типов полей, обработок и ошибок API.

5.1.1. Блок импорта и предварительной обработки данных (FileBuilderServer)

Обращение к блоку импорта и предварительной обработки данных (FileBuilderServer) происходит посредством HTTP-запросов. Тело запросов и ответов представлено в JSON формате.

Таблица 5.1.1.1 Полный список функций блока аналитической обработки

Наименование функции	**Пример запроса**
Добавление документов для	http://address:port/document

Наименование функции	Пример запроса
обработки (POST метод)	
Работа с правилами обработки и выделения данных из документов (parser)	
Добавление правила обработки данных (POST метод)	http://address:port/parser
Получения списка правил обработки данных (GET метод)	http://address:port/configuration/parser/list
Получение правила обработки данных по его идентификатору (GET метод)	http://address:port/configuration/parser/{id}
Удаление правила обработки данных по его идентификатору (DELETE метод)	http://address:port/configuration/parser/{id}
Обработка заданий (task)	
Добавление задания (POST метод)	http://address:port/parser
Получение списка заданий (GET метод)	http://address:port/configuration/task/list
Получение описания задания по его идентификатору (GET метод)	http://address:port/configuration/task/{id}
Удаление задания по его идентификатору (DELETE метод)	http://address:port/configuration/task/{id}
Работа с правилами приведения существующих значений полей данных документа к заданному значению (field conversion)	
Получение списка правил приведения данных (GET метод)	http://address:port/configuration/field_conversion/list
Получение описания правила приведения данных по его идентификатору (GET метод)	http://address:port/configuration/field_conversion/{id}
Удаление правила приведения	http://address:port/configuration/field_conver

Наименование функции	Пример запроса
данных по его идентификатору (DELETE метод)	sion/{id}

5.1.1.1. Добавление документов для обработки (POST метод)

Запрос

```
http://address:port/document
```

Параметры запроса:

Таблица 5.1.1.2 Параметры запроса

Параметр	Тип данных	Обязательность	Описание
taskName	String	Обязательный	Имя задачи (task), настройки которой будут использоваться для обработки получаемых документов
Type	String	Обязательный	Тип документа, в соответствии с которым будет выбран разметчик (parser) для обработки. Допустимые значения типа документа: XML, HTML, JSON
Content	String	Обязательный	Поле содержит документ в бинарном виде

Параметр	Тип данных	Обязательность	Описание
			закодированный в base64, если тип документа XML или HTML. Если тип документа в формате JSON, то кодировка не выполняется.
Action	String	Обязательный	Действие, которое будет производиться с документом. Допустимые значения действия: create, update
UniqueId	Integer	Обязательный	Уникальный идентификатор документа. Если данному полю не присвоено значение, то FileBuilderServer автоматически его создаст

Пример запроса:

JSON-структура в теле может содержать несколько документов для обработки.

```
{
  "apiVersion": "1.0",
  "taskName": "ArchiveTask",
  "documents": [
  {
     "url": "temp/File1.xml",
     "type": "JSON",
```

```
    "uniqueId": "23213",
  "action": "create",
    "content": {
      “text”:”json document”
   }
      },
      {
    "url": "temp/File2.xml",
    "type": "JSON",
    "uniqueId": "5IGJ5IGhpcyByZWFzb2",
  "action": "update",
    "content": {
      “text”:”json document”
   }
    }

        …

  ]
}
```

Успешный ответ

Возвращается JSON-структура, содержащая уникальные ID документов, загруженных для обработки.

Пример успешного ответа:

```
{"documentUniqueIdList":["c10c2681ab9f11e48531000c2901b740",","c10c2682ab9f11e4abe
6000c2901b740","c10c2683ab9f11e48881000c2901b740"]}
```

Ответ с ошибкой

Таблица 5.1.1.3 Коды состояния HTTP

Код	Описание
400	«Bad Request». Неверный запрос

Таблица 5.1.1.4 Параметры ответа

Параметр	Тип данных	Обязательность	Описание
error	string	Обязательный	Сообщение об ошибке
status	integer	Обязательный	Код состояния HTTP

Пример ответа с ошибкой:

```
Status: 400 Bad Request
Content-Type: application/json; charset=UTF-8
{
  "error": "Bad Request",
  "status": 400
}
```

5.1.1.2. Добавление правила обработки данных (POST метод)

Запрос

```
http://address:port/parser
```

Параметры запроса:

Таблица 5.1.1.5 Параметры запроса

Параметр	Тип данных	Обязательность	Описание
Id	Integer	Обязательный	Идентификатор правила обработки данных
Name	String	Обязательный	Тип постобработки. Допустимые значения: Reencoding, RegexReplace, CleanedFieldGenerate,

Параметр	Тип данных	Обязательность	Описание
			AutoDetectLanguage, FieldConversion, Merge, StringReplace.
Input_Field	String	Обязательный	Имя поля, из которого будут взяты данные. Имя поля должно соответствовать именам полей, описанным в FieldExtraction, либо системным полям.
Output_Field	String	Обязательный	Имя поля, в которое будут занесены данные после обработки. Если такого поля не существует, то оно будет создано.
Parameters.Name	String	Обязательный	Имя параметра. Допустимые значения: replaceWhat, replaceTo, sizeLimit, rewriteField, rewriteIfNotEmpty, rewriteLanguageField, getLangFromField, languagesNumber, fieldConversionId, inputEncoding.

Параметр	Тип данных	Обязательность	Описание
Parameters.Value	String	Обязательный	Значение параметра
Field_Fixed_Value. Name	String	Обязательный	Параметр задает имя поля, в которое будут занесены данные.
Field_Fixed_Value. Value	String	Обязательный	Значение имени поля

Пример запроса:

```
{
"parser":{
    "document_process_stages": [
     {
      "id": "1",
      "input_fields": [
       "resultData"
      ],
      "name": "AutoDetectLanguage",
      "ouput_fields": [
       "documentLanguage"
      ],
      "parameters": [
       {
        "name": "rewriteLanguageField",
        "value": "true"
       },
       {
        "name": "getLangFromField",
        "value": "basicData/languageId"
       }
      ]
     }
    ],
    "fieldF_fixed_values": [
     {
      "name": "autodetectedDocumentLanguage",
      "value": "ru"
     }
    ],
```

```
    "field_extractions": [
     {
      "extract_expression": "internalPassport/resultData",
      "extract_type": "once",
      "name": "resultData",
      "type": "text"
     },
     {
      "extract_expression": "internalPassport/basicData/fileExtension",
      "extract_type": "once",
      "name": "basicData/fileExtension",
      "type": "text"
     },
     {
      "extract_expression": "internalPassport/basicData/languageId",
      "extract_type": "once",
      "name": "basicData/languageId",
      "type": "text"
     },
     {
      "extract_expression": "internalPassport/basicData/id",
      "extract_type": "once",
      "name": "basicData/id",
      "type": "text"
     },
     {
      "extract_expression": "internalPassport/basicData/receiverPhoneAddress/value",
      "extract_type": "multiple",
      "name": "basicData/receiverPhoneAddress",
      "type": "text"
     }
    ],
    "id": "KAOIParser",
    "name": "KAOIParser",
    "type": "xml"
   }

}
```


Таблица 5.1.1.6 Характеристики постобработок

Имя постобработки	Описание
Reencoding	Производит перекодировку информации хранимой в заданном поле в UTF-8 из заданной кодировки.

Имя постобработки	Описание
RegexReplace	Производит поиск значений в заданном поле с помощью регулярного выражения и заменяет их на заданное значение.
CleanedFieldGenerate	Берет из заданного поля информацию заданной длины и записывает в новое поле.
AutoDetectLanguage	Производит определение языка в заданном поле. И записывает результат в зависимости от параметров в заданное или системное поле.
FieldConversion	Заменяет значение заданного поля в соответствии с правилом преобразования описанным в заданном FieldConversion.
Merge	Производит слияние полей и записывает результат в заданное поле.
StringReplace	Производит замену подстроки на заданное значение в заданном поле документа.

Успешный ответ

Возвращается JSON-структура, содержащая уникальный ID правила обработки данных.

Пример успешного ответа:

```
{
   "parserId": "KAOIParser"
}
```

Ответ с ошибкой

Таблица 5.1.1.7 Коды состояния HTTP

Код	Описание
400	«Bad Request». Неверный запрос

Таблица 5.1.1.8 Параметры ответа

Параметр	Тип данных	Обязательность	Описание
error	string	Обязательный	Сообщение об ошибке
status	integer	Обязательный	Код состояния HTTP

Пример ответа с ошибкой:

```
Status: 400 Bad Request
Content-Type: application/json; charset=UTF-8
{
  "error": "Bad Request",
  "status": 400
}
```

5.1.1.3. Получение списка правил обработки данных (GET метод)

Запрос

```
http://address:port/configuration/parser/list
```

Параметры запроса: без параметров.

Пример запроса:

```
http://address:port/configuration/parser/list
```

Успешный ответ

Таблица 5.1.1.9 Параметры ответа

Параметр	Тип данных	Обязательность	Описание
Id	Integer	Обязательный	Идентификатор правила обработки данных
Name	String	Обязательный	Тип постобработки. Допустимые значения: Reencoding, RegexReplace, CleanedFieldGenerate, AutoDetectLanguage, FieldConversion, Merge, StringReplace.
Input_Field	String	Обязательный	Имя поля, из которого будут взяты данные. Имя поля должно соответствовать именам полей, описанным в FieldExtraction, либо системным полям.
Output_Field	String	Обязательный	Имя поля, в которое будут занесены данные после обработки. Если такого поля не существует, то оно будет создано.
Parameters.Name	String	Обязательный	Имя параметра. Допустимые значения:

Параметр	Тип данных	Обязательность	Описание
			replaceWhat, replaceTo, sizeLimit, rewriteField, rewriteIfNotEmpty, rewriteLanguageField, getLangFromField, languagesNumber, fieldConversionId, inputEncoding.
Parameters.Value	String	Обязательный	Значение параметра
Field_Fixed_Value. Name	String	Обязательный	Параметр задает имя поля, в которое будут занесены данные.
Field_Fixed_Value. Value	String	Обязательный	Значение имени поля

Таблица 5.1.1.10 Характеристики постобработок

Имя постобработки	Описание
Reencoding	Производит перекодировку информации хранимой в заданном поле в UTF-8 из заданной кодировки.
RegexReplace	Производит поиск значений в заданном поле с помощью регулярного выражения и заменяет их на заданное значение.
CleanedFieldGenerate	Берет из заданного поля информацию заданной длины и записывает в новое поле.

Имя постобработки	Описание
AutoDetectLanguage	Производит определение языка в заданном поле. И записывает результат в зависимости от параметров в заданное или системное поле.
FieldConversion	Заменяет значение заданного поля в соответствии с правилом преобразования описанным в заданном FieldConversion.
Merge	Производит слияние полей и записывает результат в заданное поле.
StringReplace	Производит замену подстроки на заданное значение в заданном поле документа.

Пример успешного ответа:


```
{
   "parsers": [
      {
         "document_process_stages": [
            {
               "id": 1,
               "input_fields": [
                  "resultData"
               ],
               "name": "AutoDetectLanguage",
               "ouput_fields": [
                  "documentLanguage"
               ],
               "parameters": [
                  {
                     "name": "rewriteLanguageField",
                     "value": "true"
                  },
                  {
                     "name": "getLangFromField",
                     "value": "basicData/languageId"
                  }
```

```
            ]
          }
        ],
        "fieldF_fixed_values": [
          {
            "name": "autodetectedDocumentLanguage",
            "value": "ru"
          }
        ],
        "field_extractions": [
          {
            "extract_expression": "internalPassport/resultData",
            "extract_type": "once",
            "name": "resultData",
            "type": "text"
          },
          {
            "extract_expression": "internalPassport/basicData/fileExtension",
            "extract_type": "once",
            "name": "basicData/fileExtension",
            "type": "text"
          },
          {
            "extract_expression": "internalPassport/basicData/languageId",
            "extract_type": "once",
            "name": "basicData/languageId",
            "type": "text"
          },
          {
            "extract_expression": "internalPassport/basicData/id",
            "extract_type": "once",
            "name": "basicData/id",
            "type": "text"
          },
          {
            "extract_expression": "internalPassport/basicData/receiverPhoneAddress/value",
            "extract_type": "multiple",
            "name": "basicData/receiverPhoneAddress",
            "type": "text"
          }
        ],
        "id": "KAOIParser",
        "name": "KAOIParser",
        "type": "xml"
      },
      {
        "document_process_stages": [
          {
```

```
        "id": 1,
        "input_fields": [
          "resultData"
        ],
        "name": "AutoDetectLanguage",
        "ouput_fields": [
          "documentLanguage"
        ],
        "parameters": [
          {
            "name": "rewriteLanguageField",
            "value": "true"
          },
          {
            "name": "getLangFromField",
            "value": "basicData/languageId"
          }
        ]
      },
      {
        "id": 2,
        "input_fields": [
          "basicData/fileExtension"
        ],
        "name": "FieldConversion",
        "parameters": [
          {
            "name": "fieldConversionId",
            "value": "firstFieldConversion"
          }
        ]
      }
    ],
    "fieldF_fixed_values": [
      {
        "name": "autodetectedDocumentLanguage",
        "value": "ru"
      }
    ],
    "field_extractions": [
      {
        "extract_expression": "$.internalPassport.resultData",
        "extract_type": "once",
        "name": "resultData",
        "type": "text"
      },
      {
        "extract_expression": "$.internalPassport.basicData.fileExtension",
```

```
            "extract_type": "once",
            "name": "basicData/fileExtension",
            "type": "text"
          },
          {
            "extract_expression": "$.internalPassport.basicData.languageId",
            "extract_type": "once",
            "name": "basicData/languageId",
            "type": "text"
          },
          {
            "extract_expression": "$.internalPassport.basicData.id",
            "extract_type": "once",
            "name": "basicData/id",
            "type": "text"
          },
          {
            "extract_expression": "$.internalPassport.basicData.senderPhoneAddress[*]",
            "extract_type": "multiple",
            "name": "basicData/receiverPhoneAddress",
            "type": "text"
          }
        ],
        "id": "JsonParser",
        "name": "JsonParser",
        "type": "json"
      }
    ]
}
```


Ответ с ошибкой

Таблица 5.1.1.11 Коды состояния HTTP

Код	Описание
400	«Bad Request». Неверный запрос

Таблица 5.1.1.12 Параметры ответа

Параметр	Тип данных	Обязательность	Описание
error	string	Обязательный	Сообщение об ошибке
status	integer	Обязательный	Код состояния HTTP

Пример ответа с ошибкой:

```
Status: 400 Bad Request
Content-Type: application/json; charset=UTF-8
{
  "error": "Bad Request",
  "status": 400
}
```

5.1.1.4. Получение правила обработки данных по его идентификатору (GET метод)

Запрос

```
http://address:port/configuration/parser/{id}
```

Параметры запроса:

id – идентификатор правила обработки данных.

Пример запроса:

```
http://address:port/configuration/parser/{35478}
```

Успешный ответ

Таблица 5.1.1.13 Параметры ответа

Параметр	**Тип данных**	**Обязательность**	**Описание**
Id	Integer	Обязательный	Идентификатор правила обработки данных
Name	String	Обязательный	Тип постобработки. Допустимые значения:

Параметр	Тип данных	Обязательность	Описание
			Reencoding, RegexReplace, CleanedFieldGenerate, AutoDetectLanguage, FieldConversion, Merge, StringReplace.
Input_Field	String	Обязательный	Имя поля, из которого будут взяты данные. Имя поля должно соответствовать именам полей, описанным в FieldExtraction, либо системным полям.
Output_Field	String	Обязательный	Имя поля, в которое будут занесены данные после обработки. Если такого поля не существует, то оно будет создано.
Parameters.Name	String	Обязательный	Имя параметра. Допустимые значения: replaceWhat, replaceTo, sizeLimit, rewriteField, rewriteIfNotEmpty, rewriteLanguageField, getLangFromField,

Параметр	Тип данных	Обязательность	Описание
			languagesNumber, fieldConversionId, inputEncoding.
Parameters.Value	String	Обязательный	Значение параметра
Field_Fixed_Value. Name	String	Обязательный	Параметр задает имя поля, в которое будут занесены данные.
Field_Fixed_Value. Value	String	Обязательный	Значение имени поля

Таблица 5.1.1.14 Характеристики постобработок

Имя постобработки	Описание
Reencoding	Производит перекодировку информации хранимой в заданном поле в UTF-8 из заданной кодировки.
RegexReplace	Производит поиск значений в заданном поле с помощью регулярного выражения и заменяет их на заданное значение.
CleanedFieldGenerate	Берет из заданного поля информацию заданной длины и записывает в новое поле.
AutoDetectLanguage	Производит определение языка в заданном поле. И записывает результат в зависимости от параметров в заданное или системное поле.
FieldConversion	Заменяет значение заданного поля в

Имя постобработки	Описание
	соответствии с правилом преобразования описанным в заданном FieldConversion.
Merge	Производит слияние полей и записывает результат в заданное поле.
StringReplace	Производит замену подстроки на заданное значение в заданном поле документа.

Пример успешного ответа:


```
{
  "document_process_stages": [
    {
      "id": 35478,
      "input_fields": [
        "resultData"
      ],
      "name": "AutoDetectLanguage",
      "ouput_fields": [
        "documentLanguage"
      ],
      "parameters": [
        {
          "name": "rewriteLanguageField",
          "value": "true"
        },
        {
          "name": "getLangFromField",
          "value": "basicData/languageId"
        }
      ]
    }
  ],
  "fieldF_fixed_values": [
    {
      "name": "autodetectedDocumentLanguage",
      "value": "ru"
    }
  ],
  "field_extractions": [
    {
      "extract_expression": "internalPassport/resultData",
```

```
        "extract_type": "once",
        "name": "resultData",
        "type": "text"
      },
      {
        "extract_expression": "internalPassport/basicData/fileExtension",
        "extract_type": "once",
        "name": "basicData/fileExtension",
        "type": "text"
      },
      {
        "extract_expression": "internalPassport/basicData/languageId",
        "extract_type": "once",
        "name": "basicData/languageId",
        "type": "text"
      },
      {
        "extract_expression": "internalPassport/basicData/id",
        "extract_type": "once",
        "name": "basicData/id",
        "type": "text"
      },
      {
        "extract_expression": "internalPassport/basicData/receiverPhoneAddress/value",
        "extract_type": "multiple",
        "name": "basicData/receiverPhoneAddress",
        "type": "text"
      }
    ],
    "id": "KAOIParser",
    "name": "KAOIParser",
    "type": "xml"
}
```


Ответ с ошибкой

Таблица 5.1.1.15 Коды состояния HTTP

Код	Описание
400	«Bad Request». Неверный запрос

Таблица 5.1.1.16 Параметры ответа

Параметр	Тип данных	Обязательность	Описание
error	string	Обязательный	Сообщение об ошибке
status	integer	Обязательный	Код состояния HTTP

Пример ответа с ошибкой:

```
Status: 400 Bad Request
Content-Type: application/json; charset=UTF-8
{
  "error": "Bad Request",
  "status": 400
}
```

5.1.1.5. Удаление правила обработки данных по его идентификатору (DELETE метод)

Запрос

```
http://address:port/configuration/parser/{id}
```

Параметры запроса:

id – идентификатор правила обработки данных.

Пример запроса:

```
http://address:port/configuration/parser/{2378}
```

Успешный ответ

Таблица 5.1.1.17 Коды состояния HTTP

Код	Описание
200	«OK». Успешный ответ

Пример успешного ответа:

```
Status: 200 OK
```

Ответ с ошибкой

Таблица 5.1.1.18 Коды состояния HTTP

Код	Описание
400	«Bad Request». Неверный запрос

Таблица 5.1.1.19 Параметры ответа

Параметр	Тип данных	Обязательность	Описание
error	string	Обязательный	Сообщение об ошибке
status	integer	Обязательный	Код состояния HTTP

Пример ответа с ошибкой:

```
Status: 400 Bad Request
Content-Type: application/json; charset=UTF-8
{
  "error": "Bad Request",
  "status": 400
}
```

5.1.1.6. Добавление задания (POST метод)

Запрос

```
http://address:port/task
```

Параметры запроса:

Таблица 5.1.1.20 Параметры запроса

Параметр	**Тип данных**	**Обязательность**	**Описание**
Id	Integer	Обязательный	Идентификатор задания.
Task_Name	String	Обязательный	Наименование задания. Используется для статистики с данным именем. Строковый параметр не более 128 символов.
Default_Encoding	String	Обязательный	Данный параметр указывает кодировку, в которой документы будут поступать на индексирование (UTF-8, ср-1251 и др.)
Default_Language	String	Обязательный	Язык по умолчанию. Если не указана функция определения языка из поля документа, то языком документа будет являться значение

Параметр	Тип данных	Обязательность	Описание
			данного параметра. Допустимые значения: ru, en, pt, zh, ko, ja, ar, he, es, fr, it, de, tr.
Define_language_from_field	Boolean	Обязательный	Параметр автоматического определения языка документа. Допустимые значения: true, false
File_size_limit	Integer	Обязательный	Максимальный размер документа для данного задания.
Default_Xml_Parser_Name	String	Обязательный	Имя парсера для XML документов, который будет взят из файла с парсерами для структуризации документов данного задания. Строковый параметр не более 128 символов.
Default_json_parser_name	String	Обязательный	Имя парсера для для документов JSON-формата, который будет взят из файла с парсерами

Параметр	Тип данных	Обязательность	Описание
			для структуризации документов данного задания. Строковый параметр не более 128 символов.
Index	String	Обязательный	Наименование индекса, в котором хранятся документы данного задания.
index_mapping	String	Обязательный	Наименование маппинга индекса, в котором хранятся документы данного задания.

Пример запроса:

```
{
   "task": {
      "default_encoding": "utf-8",
      "default_json_parser_name": "InternalJsonParser",
      "default_language": "ru",
      "define_language_from_field": "true",
      "file_size_limit": "1048576",
      "id": "123",
      "index": "news_archive",
      "index_mapping": "news",
      "task_name": "Json"
   }
}
```

Успешный ответ

Возвращается JSON-структура, содержащая уникальный ID задания обработки данных.

Пример успешного ответа:

```
{
   "taskId":"123"}
```

Ответ с ошибкой

Таблица 5.1.1.21 Коды состояния HTTP

Код	**Описание**
400	«Bad Request». Неверный запрос

Таблица 5.1.1.22 Параметры ответа

Параметр	**Тип данных**	**Обязательность**	**Описание**
error	string	Обязательный	Сообщение об ошибке
status	integer	Обязательный	Код состояния HTTP

Пример ответа с ошибкой:

```
Status: 400 Bad Request
Content-Type: application/json; charset=UTF-8
{
  "error": "Bad Request",
  "status": 400
}
```

5.1.1.7. Получение списка заданий (GET метод)

Запрос

```
http://address:port/configuration/task/list
```

Параметры запроса: без параметров.

Пример запроса:

```
http://address:port/configuration/task/list
```

Успешный ответ

Таблица 5.1.1.23 Параметры ответа

Параметр	**Тип данных**	**Обязательность**	**Описание**
Id	Integer	Обязательный	Идентификатор задания.
Task_Name	String	Обязательный	Наименование задания. Используется для статистики с данным именем. Строковый параметр не более 128 символов.
Default_Encoding	String	Обязательный	Данный параметр указывает кодировку, в которой документы будут поступать на индексирование (UTF-8, ср-1251 и др.)
Default_Language	String	Обязательный	Язык по умолчанию. Если не указана функция

Параметр	Тип данных	Обязательность	Описание
			определения языка из поля документа, то языком документа будет являться значение данного параметра. Допустимые значения: ru, en, pt, zh, ko, ja, ar, he, es, fr, it, de, tr.
Define_language_from_field	Boolean	Обязательный	Параметр автоматического определения языка документа. Допустимые значения: true, false
File_size_limit	Integer	Обязательный	Максимальный размер документа для данного задания.
Default_Xml_Parser_Name	String	Обязательный	Имя парсера для XML документов, который будет взят из файла с парсерами для структуризации документов данного задания. Строковый параметр не более 128 символов.

Параметр	Тип данных	Обязательность	Описание
Default_json_parser_name	String	Обязательный	Имя парсера для для документов JSON-формата, который будет взят из файла с парсерами для структуризации документов данного задания. Строковый параметр не более 128 символов.
Index	String	Обязательный	Наименование индекса, в котором хранятся документы данного задания.
index_mapping	String	Обязательный	Наименование маппинга индекса, в котором хранятся документы данного задания.

Пример успешного ответа:


```
{
   "task_list": [
      {
         "default_encoding": "utf-8",
         "default_json_parser_name": "VkPostsParser",
         "default_language": "ru",
         "define_language_from_field": "true",
         "file_size_limit": "1048576",
         "id": "VkPosts",
         "index": "socialnetworks_posts",
         "index_mapping": "posts",
         "task_name": "VkPosts"
```

```
    },
    {
      "default_encoding": "utf-8",
      "default_json_parser_name": "VkComentsParser",
      "default_language": "ru",
      "define_language_from_field": "true",
      "file_size_limit": "1048576",
      "id": "VkComments",
      "index": "socialnetworks_posts",
      "index_mapping": "posts",
      "task_name": "VkComments"
    }

…

  ]
}
```


Ответ с ошибкой

Таблица 5.1.1.24 Коды состояния HTTP

Код	Описание
400	«Bad Request». Неверный запрос

Таблица 5.1.1.25 Параметры ответа

Параметр	Тип данных	Обязательность	Описание
error	string	Обязательный	Сообщение об ошибке
status	integer	Обязательный	Код состояния HTTP

Пример ответа с ошибкой:


```
Status: 400 Bad Request
Content-Type: application/json; charset=UTF-8
{
  "error": "Bad Request",
  "status": 400
}
```

5.1.1.8. Получение описания задания по его идентификатору (GET метод)

Запрос

```
http://address:port/configuration/task/{id}
```

Параметры запроса:

id – идентификатор задания.

Пример запроса:

```
http://address:port/configuration/task/{123}
```

Успешный ответ

Таблица 5.1.1.26 Параметры ответа

Параметр	Тип данных	Обязательность	Описание
Id	Integer	Обязательный	Идентификатор задания.
Task_Name	String	Обязательный	Наименование задания. Используется для статистики с данным именем. Строковый параметр не более 128 символов.
Default_Encoding	String	Обязательный	Данный параметр указывает кодировку, в которой документы будут поступать на

Параметр	Тип данных	Обязательность	Описание
			индексирование (UTF-8, ср-1251 и др.)
Default_Language	String	Обязательный	Язык по умолчанию. Если не указана функция определения языка из поля документа, то языком документа будет являться значение данного параметра. Допустимые значения: ru, en, pt, zh, ko, ja, ar, he, es, fr, it, de, tr.
Define_language_from_field	Boolean	Обязательный	Параметр автоматического определения языка документа. Допустимые значения: true, false
File_size_limit	Integer	Обязательный	Максимальный размер документа для данного задания.
Default_Xml_Parser_Name	String	Обязательный	Имя парсера для XML документов, который будет взят из файла с парсерами для структуризации

Параметр	Тип данных	Обязательность	Описание
			документов данного задания. Строковый параметр не более 128 символов.
Default_json_parser_name	String	Обязательный	Имя парсера для для документов JSON-формата, который будет взят из файла с парсерами для структуризации документов данного задания. Строковый параметр не более 128 символов.
Index	String	Обязательный	Наименование индекса, в котором хранятся документы данного задания.
index_mapping	String	Обязательный	Наименование маппинга индекса, в котором хранятся документы данного задания.

Пример успешного ответа:

```
{
  "task_list": [
    {
      "default_encoding": "utf-8",
```

```
        "default_json_parser_name": "VkPostsParser",
        "default_language": "ru",
        "define_language_from_field": "true",
        "file_size_limit": "1048576",
        "id": "VkPosts",
        "index": "socialnetworks_posts",
        "index_mapping": "posts",
        "task_name": "VkPosts"
      },
  ]
}
```


Ответ с ошибкой

Таблица 5.1.1.27 Коды состояния HTTP

Код	Описание
400	«Bad Request». Неверный запрос

Таблица 5.1.1.28 Параметры ответа

Параметр	Тип данных	Обязательность	Описание
error	string	Обязательный	Сообщение об ошибке
status	integer	Обязательный	Код состояния HTTP

Пример ответа с ошибкой:


```
Status: 400 Bad Request
Content-Type: application/json; charset=UTF-8
{
  "error": "Bad Request",
  "status": 400
}
```

5.1.1.9. Удаление задания по его идентификатору (DELETE метод)

Запрос

```
http://address:port/configuration/task/{id}
```

Параметры запроса:

id – идентификатор задания.

Пример запроса:

```
http://address:port/configuration/task/{123}
```

Успешный ответ

Таблица 5.1.1.29 Коды состояния HTTP

Код	Описание
200	«OK». Успешный ответ

Пример успешного ответа:

```
Status: 200 OK
```

Ответ с ошибкой

Таблица 5.1.1.30 Коды состояния HTTP

Код	Описание
400	«Bad Request». Неверный запрос

Таблица 5.1.1.31 Параметры ответа

Параметр	Тип данных	Обязательность	Описание
error	string	Обязательный	Сообщение об ошибке

status	integer	Обязательный	Код состояния HTTP

Пример ответа с ошибкой:

```
Status: 400 Bad Request
Content-Type: application/json; charset=UTF-8
{
  "error": "Bad Request",
  "status": 400
}
```

5.1.1.10. Получение списка правил приведения данных (GET метод)

Запрос

```
http://address:port/configuration/field_conversion/list
```

Параметры запроса: без параметров.

Пример запроса:

```
http://address:port/configuration/field_conversion/list
```

Успешный ответ

Таблица 5.1.1.32 Параметры ответа

Параметр	**Тип данных**	**Обязательность**	**Описание**
Id	Integer	Обязательный	Идентификатор правила приведения данных
Keys	String	Обязательный	Задает строку, которая будет заменена на

Параметр	Тип данных	Обязательность	Описание
			значение заданное в свойстве Value.
Value	String	Обязательный	Задает строку, на значение, которой должно быть заменено поле документа, если оно равно одному из значений, перечисленных в параметре Key.

Пример успешного ответа:

```
{
   "field_conversions": [
      {
         "id": "firstFieldConversion",
         "keys": [
            "one",
            "two",
            "html"
         ],
         "value": "russian"
      }
   ]
}
```

Ответ с ошибкой

Таблица 5.1.1.33 Коды состояния HTTP

Код	Описание
400	«Bad Request». Неверный запрос

Таблица 5.1.1.34 Параметры ответа

Параметр	**Тип данных**	**Обязательность**	**Описание**
error	string	Обязательный	Сообщение об ошибке
status	integer	Обязательный	Код состояния HTTP

Пример ответа с ошибкой:

```
Status: 400 Bad Request
Content-Type: application/json; charset=UTF-8
{
  "error": "Bad Request",
  "status": 400
}
```

5.1.1.11. Получение описания правила приведения данных по его идентификатору (GET метод)

Запрос

```
http://address:port/configuration/field_conversion/{id}
```

Параметры запроса:

id – идентификатор правила приведения данных.

Пример запроса:

```
http://address:port/configuration/field_conversion/{845}
```

Успешный ответ

Таблица 5.1.1.35 Параметры ответа

Параметр	**Тип данных**	**Обязательность**	**Описание**
Id	Integer	Обязательный	Идентификатор правила приведения данных
Keys	String	Обязательный	Задает строку, которая будет заменена на значение заданное в свойстве Value.
Value	String	Обязательный	Задает строку, на значение, которой должно быть заменено поле документа, если оно равно одному из значений, перечисленных в параметре Key.

Пример успешного ответа:


```
{
   "id": "firstFieldConversion",
   "keys": [
      "one",
      "two",
      "html"
   ],
   "value": "russian"
}
```

Ответ с ошибкой

Таблица 5.1.1.36 Коды состояния HTTP

Код	Описание
400	«Bad Request». Неверный запрос

Таблица 5.1.1.37 Параметры ответа

Параметр	Тип данных	Обязательность	Описание
error	string	Обязательный	Сообщение об ошибке
status	integer	Обязательный	Код состояния HTTP

Пример ответа с ошибкой:

```
Status: 400 Bad Request
Content-Type: application/json; charset=UTF-8
{
  "error": "Bad Request",
  "status": 400
}
```

5.1.1.12. Удаление правила приведения данных по его идентификатору (DELETE метод)

Запрос

```
http://address:port/configuration/field_conversion/{id}
```

Параметры запроса:

id – идентификатор правила приведения данных.

Пример запроса:

```
http://address:port/configuration/field_conversion/{845}
```

Успешный ответ

Таблица 5.1.1.38 Коды состояния HTTP

Код	Описание
200	«OK». Успешный ответ

Пример успешного ответа:

```
Status: 200 OK
```

Ответ с ошибкой

Таблица 5.1.1.39 Коды состояния HTTP

Код	Описание
400	«Bad Request». Неверный запрос

Таблица 5.1.1.40 Параметры ответа

Параметр	Тип данных	Обязательность	Описание
error	string	Обязательный	Сообщение об ошибке
status	integer	Обязательный	Код состояния HTTP

Пример ответа с ошибкой:

```
Status: 400 Bad Request
Content-Type: application/json; charset=UTF-8
{
  "error": "Bad Request",
  "status": 400
}
```

5.1.2. Блок обработки текстовых данных (**PostProcessServer**)

Обработка поступающих текстовых данных выполняется в автоматическом режиме при помощи PostProcessServer, который вызывает функции соответствующих модулей.

Таблица 5.1.2.1 Полный список модулей блока обработки текстовых данных

Наименование блока обработки текстовых данных	**Характеристика**
Morphology	Приведение термов текстовых данных к нормальной форме. Предсказание морфологических форм.
Gazetteer	Добавление служебной семантической информации. Определение кандидатов в сущности из текстовых данных.
Named Entity Recognition	Создание и обработка сложных (составных) сущностей.
Sentiment Analyzer	Обработка тональности для сущностей (простые и составные) и документов.

5.1.2.1. Модуль Morphology

Модуль предназначен для приведения термов из поступающих текстовых данных к нормальной форме, предсказания морфологических форм для неизвестных термов.

Таблица 5.1.2.2 Полный список функций модуля Morphology

Функция	**Описание**
Morphology()	Конструктор класса.

Функция	Описание
bool loadBinaries(std::string utf8Path)	Загрузка словарей в бинарном виде. Utf8Path – путь к словарям. Функция возвращает true в случае успеха и false в случае неудачи.
void unloadBinaries()	Освобождение памяти от словарей.
std::string getLastError()	Возвращает последнюю ошибку.
langUtils::Languages::LangCode getLanguage()	Возвращает код языка, используемого в экземпляре класса морфологии.
~Morphology()	Деструктор класса.
ResValue getNorm(const char* wordForm, char* norm, bool properNounFlag)	Возвращает нормальную форму слова. Входные параметры: wordForm – входное слово, properNounFlag - параметр, принимающий значение true в случае, если слово пишется с большой буквы и false, если с маленькой. Выходные параметры: norm – возвращаемая нормальная форма, ResValue (принимает значение: NOT_FOUND, FOUND, PREDICTED).
ResValue getNorm(const char* wordForm, char* norm, bool properNounFlag, int64_t &morphAttrs)	Возвращает нормальную форму слова. Входные параметры: wordForm – входное слово, properNounFlag - параметр, принимающий значение true в случае, если слово пишется с большой буквы и false, если с маленькой. Выходные параметры: norm – возвращаемая нормальная форма, ResValue (принимает значение: NOT_FOUND,

Функция	Описание
	FOUND, PREDICTED). Дополнительно возвращает morphAttrs – морфологические аттрибуты исходной словоформы.
ResValue getNormVector(const char* wordForm, std::vector<std::string>& outVector, std::vector<int64_t>& morphAttrs, std::vector<int64_t>& groupIds)	Возвращает список вариантов нормальной формы слова. Входные параметры: wordForm – входное слово Выходные параметры: outVector – список вариантов нормальной формы слова, morphAttrs – список возможных морфологических атрибутов словоформы, соответствующий списку вариантов нормальной формы, groupIds - список возможных групп склонения словоформы, соответствующий списку вариантов нормальной формы.
ResValue getNormVector(const char* wordForm, std::vector<std::string>& outVector, std::vector<int64_t>& morphAttrs, std::vector<int64_t>& groupIds, std::vector <uint8_t> &resValues)	Возвращает список вариантов нормальной формы слова. Входные параметры: wordForm – входное слово Выходные параметры: outVector – список вариантов нормальной формы слова, morphAttrs – список возможных морфологических атрибутов словоформы, соответствующий списку вариантов нормальной формы, groupIds - список возможных групп склонения словоформы, соответствующий списку вариантов нормальной формы.

Функция	Описание
	Дополнительно возвращает resValues – список значений, соответствующий списку вариантов нормальной формы, определяет – была ли предсказана нормальная форма или содержится в словаре.

5.1.2.2. Модуль Gazetteer

Модуль предназначен для обогащения термов дополнительной служебной семантической информацией, определения кандидатов в сущности из текстовых данных.

Таблица 5.1.2.3 Полный список функций модуля Gazetteer

Функция	Описание
Gazetteer ()	Конструктор класса
~Gazetteer()	Деструктор класса
void init(std::map <std::string, inmemory::utils::BitStructure> &semanticAttrs)	Инициализация экземпляра класса. Производится считывание словарей, соответствующих установленному языку. Возвращается semanticAttrs – список семантических атрибутов, полученный из конфигурации словарей.
void run(const inmemory::parse::ParsedField* inputParsedField, std::map <std::string, inmemory::utils::BitStructure>	Основная функция, запуск работы модуля. Входные параметры: inputParsedField – входной документ, semanticAttrs – список семантических атрибутов. Выходные параметры:

Функция	Описание
&semanticAttrs, std::vector <inmemory::utils::Entity> &entities)	entities – список выделенных на данном этапе сущностей.
void setLanguage(const std::string &lang)	Установка языка для экземпляра класса.

5.1.2.3. Модуль Named Entity Recognition

Модуль предназначен для обогащения термов дополнительной служебной семантической информацией, определения кандидатов в сущности из текстовых данных, создания и обработки сложных (составных) сущностей.

Таблица 5.1.2.4 Полный список функций модуля Gazetteer

Функция	Описание
Ner()	Конструктор класса
~Ner()	Деструктор класса
void setLanguage(const std::string &lang)	Установка языка для экземпляра класса
void init(std::map <std::string, BitStructure> &semanticAttrsMap)	Инициализация экземпляра класса. Принимает semanticAttrs – список семантических атрибутов. Производится считывание словарей правил, соответствующих установленному языку.
void run(inmemory::parse::ParsedField &parsedField, std::vector <inmemory::utils::Entity> inputVec, std::vector	Основная функция, запуск работы модуля. Входные параметры: parsedField – входной документ, semanticAttrsMap – список семантических аттрибутов, inputVec – список сущностей,

Функция	Описание
<inmemory::utils::Entity> &entityVec, std::map <std::string, BitStructure> &semanticAttrsMap)	полученный на предыдущем этапе. Выходные параметры: entityVec – дополненный в процессе работы модуля список сущностей.

5.1.2.4. Модуль Sentiment Analyzer

Модуль предназначен для определения эмоциальной окраски слов из документа, численной оценки сентимента для документа в целом.

Таблица 5.1.2.5 Полный список функций модуля Sentiment Analyzer

Функция	Описание
SentimentAnalyzer()	Конструктор класса
~SentimentAnalyzer()	Деструктор класса
float computeScore(std::map <std::string, inmemory::utils::SentimentSemanticInfo> &measuredEntities)	Возвращает вычисленную численную оценку сентимента для документа. Принимает measuredEntities – список оцененных модулем сущностей документа.
inmemory::utils::ObjectPolarity computePolarity(std::map <std::string, inmemory::utils::SentimentSemanticInfo> &measuredEntities)	Возвращает полярность документа, принимает measuredEntities – список оцененных модулем сущностей документа.
bool init(std::map <std::string, inmemory::utils::BitStructure> &semanticAttrs)	Инициализация экземпляра класса. Принимает semanticAttrs – список семантических атрибутов.
void	Основная функция, запуск работы модуля.

Функция	Описание
run(inmemory::parse::ParsedField &parsedField, std::map <std::string, inmemory::utils::SentimentSemanticInfo> &measuredEntities)	Входные параметры: parsedField – входной документ. Выходные параметры: measuredEntities – список сущностей с приписанной им дополнительной информацией – сентиментом, полярностью и т.д.
void setLanguage(const langUtils::Languages::LangCode&)	Установка языка для экземпляра класса
std::vector <inmemory::utils::SentimentInfo>& getSentiments()	Возвращает список эмоциально окрашенных слов из документа.

6. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

В качестве основного формата представления входных и выходных данных используется JSON (англ. JavaScript Object Notation) – текстовый формат обмена данными, основанный на JavaScript (происходит от подмножества языка стандарта ECMA-262 1999 года). Формат считается независимым от языка и может использоваться практически с любым языком программирования.

JSON-текст представляет собой (в закодированном виде) одну из двух структур:

– Набор пар ключ: значение. В различных языках это реализовано как объект, запись, структура, словарь, хэш-таблица, список с ключом или ассоциативный массив. Ключом может быть только строка, значением — любая форма.

– Упорядоченный набор значений. Во многих языках это реализовано как массив, вектор, список или последовательность.

В качестве значений в JSON используются структуры:

– Объект – это неупорядоченное множество пар ключ: значение, заключённое в фигурные скобки «{ }». Ключ описывается строкой, между ним и значением стоит символ «:». Пары ключ-значение отделяются друг от друга запятыми.

– Массив (одномерный) – это упорядоченное множество значений. Массив заключается в квадратные скобки «[ ]». Значения разделяются запятыми.

– Значение может быть строкой в двойных кавычках, числом, объектом, массивом, одним из литералов: true, false или null. Таким образом, структуры могут быть вложены друг в друга.

– Строка – это упорядоченное множество из нуля или более символов юникода, заключенное в двойные кавычки. Символы могут быть

указаны с использованием escape-последовательностей, начинающихся с обратной косой черты «\».

Входными данными являются структурированные и неструктурированные наборы текстовых данных, представленные отдельными файлами (txt, doc, docx, rtf, pdf, odf, chm и др.). Размещение требуемых файлов осуществляется в общедоступной для вычислительной системы файловой сети или внутренних серверов.

Выходными данными являются структурированные наборы данных, представленные файлами в JSON формате. Данные файлы размещаются в распределённой файловой системе HDFS. Пример выходных данных – текст с добавленной семантической информацией в файле формата JSON:

{ "index" : { "_index" : "news_actual_test", "_type" : "news", "_id" : "1f8805aa91f811e5ad68e06995504cfc" } }
{"_internal_data":{"extension":"json","language":"ru","processed_date":1448303857},"entities":[{"count":1,"data_id":0,"sentiment":"NEUTRAL","type":"ORGANIZATION","value":"Twitter"},{"count":3,"data_id":1,"sentiment":"NEGATIVE","type":"PERSON","value":"Виталий Кличко"},{"count":2,"data_id":2,"sentiment":"NEUTRAL","type":"PERSON","value":"Дмитрий Рогозин"},{"count":2,"data_id":3,"sentiment":"NEUTRAL","type":"LOCATION","value":"Киев"},{"count":1,"data_id":4,"sentiment":"NEUTRAL","type":"LOCATION","value":"Москва"},{"count":1,"data_id":5,"sentiment":"NEUTRAL","type":"LOCATION","value":"РФ"}],"sentiment":"NEUTRAL","text":"<p>Российский политик выразил сожаления киевлянам в связи с их выбором градоначальника<\/p><p> <span data-ids=\"|4|\">МОСКВА<\/span>, 11 ноября 2015, 09:22 — REGNUM Вице-премьер <span data-ids=\"|5|\">России<\/span> <span data-ids=\"|2|\">Дмитрий<\/span> <span data-ids=\"|2|\">Рогозин<\/span> заявил, что ему жаль жителей <span data-ids=\"|3|\">Киева<\/span> в связи с их выбором мэра города <span data-ids=\"|1|\">Виталия<\/span> <span data-ids=\"|1|\">Кличко<\/span>, свои слова политик подкрепил записью пародийного выступления, где российские комики с точностью передали манеру киевского градоначальника вести разговор.<\/p><p> «<span data-to-ids=\"|1|\" data-attr=\"NEG\">Бедные<\/span> киевляне. .Теперь они за таких голосуют. Мне, честно говоря, грустно на это смотреть, поскольку это надолго», — написал <span data-ids=\"|2|\">Рогозин<\/span> в своем официальном микроблоге в <span data-ids=\"|0|\">Twitter<\/span>.<\/p><p> Напомним, бывший боксер и мэр <span data-ids=\"|3|\">Киева<\/span> <span data-ids=\"|1|\">Виталий<\/span> <span data-ids=\"|1|\">Кличко<\/span> не раз становился центром обсуждений в сети за свои не всегда понятные высказывания, за что пользователи с иронией прозвали его «Звездой YouTub». Так, самым знаменитой цитатой <span data-ids=\"|1|\">Кличко<\/span>, быстро разлетевшейся по сети, стало бессмертное:<\/p><p> «А сегодня в завтрашний день не все могут смотреть, вернее смотреть могут не только лишь все, мало, кто может это делать».<\/p>","title":"Европейские лидеры помолчат об антироссийских санкциях","url":"http:\/\/www.kommersant.ru\/doc\/2753762"}

Таблица 6.1 Параметры представления выходных данных в файле формата JSON

Параметр	Тип данных	Обязательность	Описание
_id	Integer	Обязательный	Уникальный идентификатор обновляемого документа
_type	String	Обязательный	Тип маппинга данных индекса, который используется для обновления документов
_index	String	Обязательный	Наименование индекса, в котором хранятся данные обновляемых документов
_internal_data. extension	String	Обязательный	Расширение обработанного документа. По умолчанию – json
_internal_data. language	String	Обязательный	Язык обработанного документа
_internal_data. processed_date	Integer	Обязательный	Дата и время обработки документа в форме уникального идентификатора
entities.data_id	Integer	Обязательный	Уникальный идентификатор выявленной в тексте сущности
entities.count	Integer	Обязательный	Количество встреч выявленной сущности в

Параметр	Тип данных	Обязательность	Описание
			тексте
entities.sentiment	String	Обязательный	Значение тональности выявленной сущности (NEGATIVE, POSITIVE, NEUTRAL)
entities.type	String	Обязательный	Тип выявленной сущности (PERSON, LOCATION, ORGANISATION)
entities.value	String	Обязательный	Значение выявленной сущности
text	String	Обязательный	Текст обработанного документа
title	String	Обязательный	Заголовок текста обработанного документа

7. СООБЩЕНИЯ ОПЕРАТОРУ

Обращение к REST API происходит посредством HTTP-запросов. Тело запросов и ответов представлено в JSON формате. Ошибки выполнения функций предоставляются оператору в виде соответствующих кодов состояния HTTP – (англ. HTTP status code) – часть первой строки ответа сервера при запросах по протоколу HTTP. Он представляет собой целое число из трёх десятичных цифр. Первая цифра указывает на класс состояния. За кодом ответа обычно следует отделённая пробелом поясняющая фраза на английском языке, которая разъясняет человеку причину именно такого ответа (см. Таблицу 7.1).

Таблица 7.1 Коды состояния HTTP, используемые для обозначения ошибок

Код	**Описание**	**Рекомендуемые действия для устранения ошибки**
400	«Bad Request». Сервер обнаружил в запросе клиента синтаксическую ошибку. Относится к группе ошибок со стороны клиента.	Проверить содержание требуемого запроса на наличие синтаксических ошибок и внести соответствующие правки.
404	«Not Found». Сервер понял запрос, но не нашёл соответствующего ресурса по указанному URL. Относится к группе ошибок со стороны клиента.	Проверить содержание требуемого запроса на правильность запрашиваемого адреса URL и внести соответствующие правки.
500	«Internal Server Error». Внутренняя ошибка сервера.	Требуется детальная диагностика системы (анализ логов, инспекция

Код	Описание	Рекомендуемые действия для устранения ошибки
	Относится к группе ошибок со стороны сервера.	данных и др.).
503	«Service Unavailable». Сервер временно не имеет возможности обрабатывать запросы по техническим причинам (обслуживание, перегрузка и прочее). Относится к группе ошибок со стороны сервера.	Рекомендуется повторить запрос через некоторое время.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

В настоящем документе приняты следующие условные обозначения:

СПО	Специальное программное обеспечение
3i NLP Platform	СПО «3i NLP Platform»
API	сокр. англ. Application Programming Interface, интерфейс программирования приложений – набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением для использования во внешних программных продуктах
IP-адрес	сокр. англ. Internet Protocol Address – уникальный сетевой адрес узла в компьютерной сети, построенной по протоколу IP
JSON	сокр. англ. JavaScript Object Notation – текстовый формат обмена данными, основанный на JavaScript (происходит от подмножества языка стандарта ECMA-262 1999 года)
NLP	сокр. англ. Natural Language Processing, обработка естественного языка – общее направление искусственного интеллекта и математической лингвистики, изучающее проблемы компьютерного анализа и синтеза естественных языков
Hadoop	Свободно распространяемый набор утилит, библиотек и средство разработки и выполнения распределённых программ, работающих на кластерах из сотен и тысяч узлов.
HDFS	сокр. англ. Hadoop Distributed File System – файловая система, предназначенная для хранения файлов

	больших размеров, поблочно распределённых между узлами вычислительного кластера
HTTP	сокр. англ. HyperText Transfer Protocol, протокол передачи гипертекста — протокол прикладного уровня передачи данных на технологии «клиент-сервер»
REST API	сокр. англ. Representational State Transfer, передача репрезентативного состояния – метод взаимодействия компонентов распределённого приложения в сети Интернет, при котором вызов удаленной процедуры представляет собой обычный HTTP-запрос

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в документе	№ документа	Входящий № сопроводительного документа и дата	Подпись	Дата
	Измененных	замененных	новых	аннулированных					