INFOWATCH

# SDK. Загрузка выгрузок из БД

17/05/2018

© АО “ИнфоВотч”

Тел./Факс +7 (495) 229-00-22

http://www.infowatch.ru

# СОДЕРЖАНИЕ


1 Назначения программного интерфейса REST API ....4

2 Общее описание программного интерфейса REST API....5

2.1 Создание эталонной выгрузки REST API ....5

2.2 Обновление эталонной выгрузки REST API ....5

2.3 Добавление содержимого к эталонной выгрузке REST API ....5

2.4 Получение состояния эталонной выгрузки REST API....5

2.5 Инициирование распространения нового содержимого загруженных эталонных документов на модули анализа REST API ....5

2.6 Получение версии интерфейса, который поддерживается TrafficMonitor REST API . 6

3 Формат эталонной выгрузки REST API ....7

4 Регистрация стороннего компонента REST API ....8

4.1 Формат файла регистрации стороннего компонента REST API....8

5 Конфигурирование программного интерфейса REST API....10

6 Типовые сценарии использования API ....11

6.1 Типовой сценарий создания эталонной выгрузки....11

6.2 Типовой сценарий обновления эталонной выгрузки....12

6.3 Типовой сценарий добавления данных к выгрузке ....12

6.4 Реализация завершения скачивания эталонной выгрузки....13

7 Функциональное описание программного интерфейса REST API .....14

7.1 Общие заголовки GET/POST-запросов ....14

7.2 Работа с конфигурацией REST API ....14

7.2.1 Распространение конфигурации REST API ....14

7.3 Работа с эталонными выгрузками REST API ....14

7.3.1 Создание выгрузки REST API....15

7.3.2 Обновление данных выгрузки REST API....16

7.3.3 Добавление данных выгрузки REST API ....17

7.3.4 Получение состояния выгрузки REST API ....17

7.3.5 Получение текущей версии REST API ....19

7.4 Общие ошибки REST API ....20

7.4.1 Успешный ответ REST API ....20

7.4.2 Ошибка валидации данных/версии REST API ... 20
7.4.2.1 Формат ответа в случае неправильной версии API (JSON Scheme) REST API ... 20
7.4.2.2 Формат ответа в случае ошибки валидации запроса (JSON Scheme) REST API ... 21
7.4.2.3 Пример ошибки при указании неправильной версии REST API ... 23
7.4.2.4 Пример ошибки проверки на уникальность имени эталонной выгрузки REST API ... 23
7.4.3 Необходима авторизация REST API ... 24
7.4.4 Недостаточно прав REST API ... 24
7.4.5 Сущность в системе не найдена REST API ... 24
7.4.6 Тело запроса слишком большое REST API ... 24
7.4.7 Превышено количество запросов к серверу REST API ... 24
7.4.8 Ошибка системы REST API ... 25
7.4.9 Сервер временно не доступен REST API ... 26

# 1 Назначения программного интерфейса REST API

Данный программный интерфейс переназначен для интеграции сторонних компонент с продуктом InfoWatch Traffic Monitor. Сторонние компоненты могут самостоятельно формировать документы, представляющие собой «Эталонные выгрузки баз данных» и, используя предоставленный интерфейс, передавать их в Traffic Monitor. После этого переданные выгрузки будут детектироваться в анализируемых Traffic Monitor потоках данных. Правила детектирования выгрузок задаются в консоли Traffic Monitor. Сторонний компонент только загружает или обновляет содержимое выгрузки. Сторонний компонент должен быть зарегистрирован в системе согласно процедуре, которая описана в разделе «Регистрация стороннего компонента».

# 2 Общее описание программного интерфейса REST API

Программный интерфейс представляет собой REST API, т.е. последовательность запросов к Web-серверу. Программный интерфейс предоставляет следующие возможности:

- Создание эталонной выгрузки REST API;
- Обновление эталонной выгрузки REST API;
- Добавление содержимого к эталонной выгрузке REST API;
- Получение состояния эталонной выгрузки REST API;
- Инициирование распространения нового содержимого загруженных эталонных документов на модули анализа REST API;
- Получение версии интерфейса, который поддерживается TrafficMonitor REST API.

## 2.1 Создание эталонной выгрузки REST API

```
FingerPrintDsc Create(string name, string[] columns)
```

Функция создаёт эталонную выгрузку с заданным именем и списком имён колонок эталонной выгрузки. Возвращается описание созданного эталонного объекта (FingerPrint).

## 2.2 Обновление эталонной выгрузки REST API

```
FingerPrintDsc UpdateContent(string FingerPrintId,string[]content)
```

Функция позволяет загрузить в существующий эталонный объект (выгрузку) его содержимое. Эталонный объект определяется его идентификатором (поле структуры FingerPrintDsc). Возвращается описание эталонного объекта (FingerPrint). После загрузки содержимого начинается процесс преобразования и сохранения содержимого эталонного объекта. Данная функция целиком заменяет содержимое эталонного объекта.

## 2.3 Добавление содержимого к эталонной выгрузке REST API

```
FingerPrintDsc AddContent(string FingerPrintId, string[] content)
```

Функция позволяет добавить в существующий эталонный объект (выгрузку) дополнительное содержимое. Эталонный объект определяется его идентификатором (поле структуры FingerPrintDsc). Возвращается описание эталонного объекта (FingerPrint). После загрузки содержимого начинается процесс преобразования и сохранения содержимого эталонного объекта. Данная функция целиком заменят содержимое эталонного объекта.

## 2.4 Получение состояния эталонной выгрузки REST API

```
FingerPrintDsc GetInfo(string FingerPrintId)
```

Функция позволяет получить информацию о существующей эталонной выгрузке. Возвращается описание эталонного объекта (FingerPrint). Важным полем результата вызова этой функции является поле статуса эталонной выгрузки.

## 2.5 Инициирование распространения нового содержимого загруженных эталонных документов на модули анализа REST API

```
Void Ditribute()
```

Функция инициирует процедуру распространения нового содержимого эталонных выгрузок по модулям анализа. Функция синхронная. После её выполнения система Traffic Monitor готова к детектированию нового содержимого в анализируемых потоках данных.

## 2.6 Получение версии интерфейса, который поддерживается TrafficMonitor REST API

```
VersionInfo GetInterfaceVersion()
```

Функция позволяет получить информация об интерфейсе, который поддерживает Traffic Monitor.

Каждая часть выгрузки, которая загружается в Traffic Monitor, проходит этапы преобразования во внутренний формат и сохранения в базе данных. После этого сторонний компонент должен дать команду о передаче новой выгрузки системам анализа Traffic Monitor. Таким образом, можно загрузить по частям достаточно большую выгрузку. Т.е. выгрузка загружается в несколько запросов, а затем одной командой распространяется по системам анализа.

Каждый этап обработки загруженной эталонной выгрузки занимает некоторое время. На каждом этапе могут произойти ошибки.

Программный интерфейс позволяет отслеживать этапы обработки выгрузки и реагировать на ошибки, которые могут произойти во время обработки.

# 3 Формат эталонной выгрузки REST API

Эталонная выгрузка представляет собой последовательность строк в кодировке UTF-8. Строки должны быть разделены символами CRLF или LF. Значения ячеек в строке должны быть разделены символами TAB или ‘,’(Comma). Фактически это соответствует формату TSV (Tab separated values) или CSV (Comma separated values). Число ячеек во всех строках должно совпадать. При создании выгрузки указывается число ячеек. Оно равно числу имён столбцов, которые передаются в функцию Create. В существующую эталонную выгрузку можно загружать только содержимое, где число ячеек в строке равно числу, которое было указано при создании. Любое другое число ячеек сделает выгрузку не валидной.

# 4 Регистрация стороннего компонента REST API

Для интеграции стороннего модуля перехвата данных с системой Traffic Monitor нужно выполнить следующие шаги:

1. Зарегистрировать в компании InfoWatch строку идентификатор компании производителя компонента. Это позволит связывать компанию производителя с лицензией, которая будет устанавливаться в систему Traffic Monitor;
2. Получить лицензию разработчика, связанную с идентификатором компании. Это позволит загружать в Traffic Monitor эталонные выгрузки этого производителя. Лицензия разработчика позволяет загружать произвольные эталонные выгрузки;
3. Создать файл регистрации стороннего компонента в описанном ниже формате. К этому моменту должны быть определены названия источника данных эталонной выгрузки, которая будет формироваться сторонним компонентом. В файл регистрации добавляется лицензия, полученная в пункте 2;
4. Загрузить файл регистрации стороннего компонента в систему Traffic Monitor через интерфейс консоли ТМ. Раздел «Управление» - «Плагины». (см. "Traffic Monitor. Руководство администратора");
5. Получить токен доступа в свойствах загруженного стороннего компонента;

Система готова принимать сформированные эталонные выгрузки от заданной компании-производителя с указанным источником эталонной выгрузки.

## 4.1 Формат файла регистрации стороннего компонента REST API

Плагин представляет собой архив в формате .zip. В состав архива входят:

- папка **licenses**, содержащая файлы лицензий;
- папка **icon**, содержащая файлы с используемыми пиктограммами для регистрируемых событий (необязательно);
- файл **manifest.json**, содержащий информацию о плагине.

Для внешних систем-источников событий файл **manifest.json** должен содержать следующую информацию:

| **Название** | **Является обязательным** | **Описание** |
|---|---|---|
| Версия | Да | Версия плагина |
| Идентификатор компании-разработчика | Да | Соответствует названию компании в лицензии |
| Название | Да | Название плагина |
| Описание | Нет | Описание плагина |
| Признак "Предустановленный" | Нет | Входит ли в состав Системы |

| Название | Является обязательным | Описание |
|---|---|---|
| Имя источника данных эталонной выгрузки | Да | Строка, выбранная производителем плагина для идентификации источника эталонной выгрузки |
| Уникальный идентификатор | Да | Уникальный 128-битный идентификатор (UUID) плагина |

**Важно!**

В состав файла входит просроченная лицензия. Для регистрации плагина на основании файла примера нужно будет получать новую лицензию.

Полная JSON-схема входит в состав примеров SDK (см. раздел "Описание примеров PushAPI SDK"). Пример файла регистрации модуля входит в состав SDK.

# 5 Конфигурирование программного интерфейса REST API

В конфигурировании программного интерфейса участвуют следующие важные параметры:

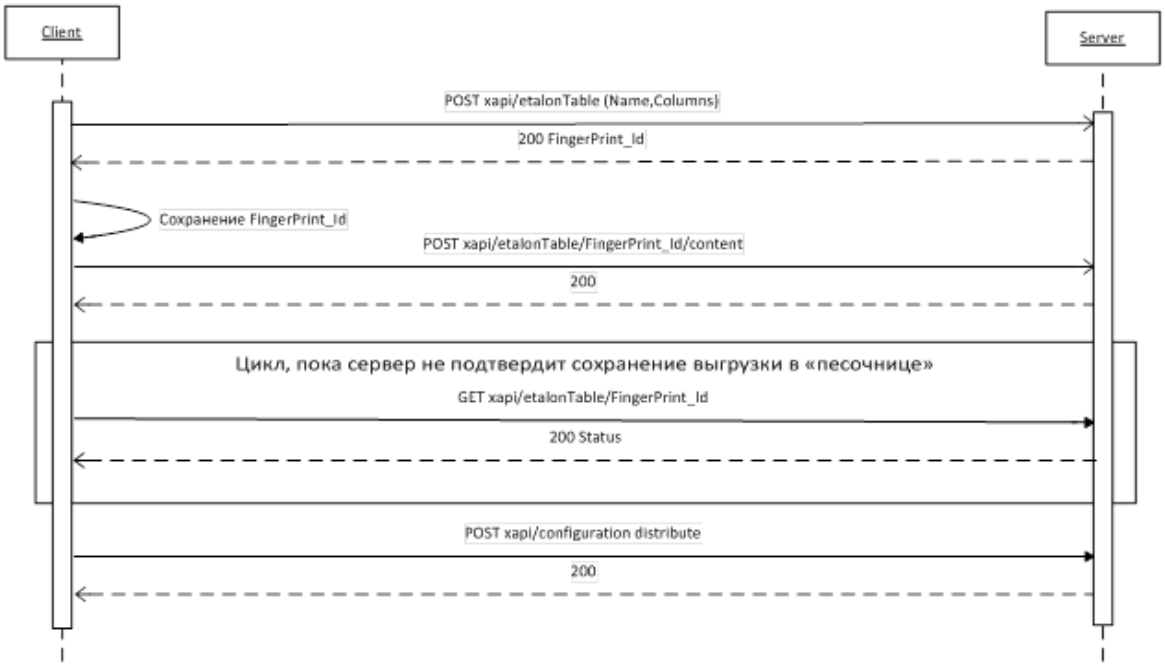
- Адрес подключения с сервису SDK (сетевой адрес, порт);
- Токен «авторизации». Строка символов, которая позволят авторизовать компоненты, использующие SDK. Данная строка может быть получена в панели управления внешними автоматизированными системами в консоли управления Traffic Monitor;
- Идентификатор передаваемых данных. Строка, которую производитель внешнего компонента получает от InfoWatch. Данная строка характеризует тип передаваемых данных и используется в политике лицензирования;
- Идентификатор компании-производителя внешнего компонента. Строка, которую производитель внешнего компонента получает от InfoWatch. Данная строка характеризует компанию-производителя и используется в политике лицензирования.

# 6 Типовые сценарии использования API

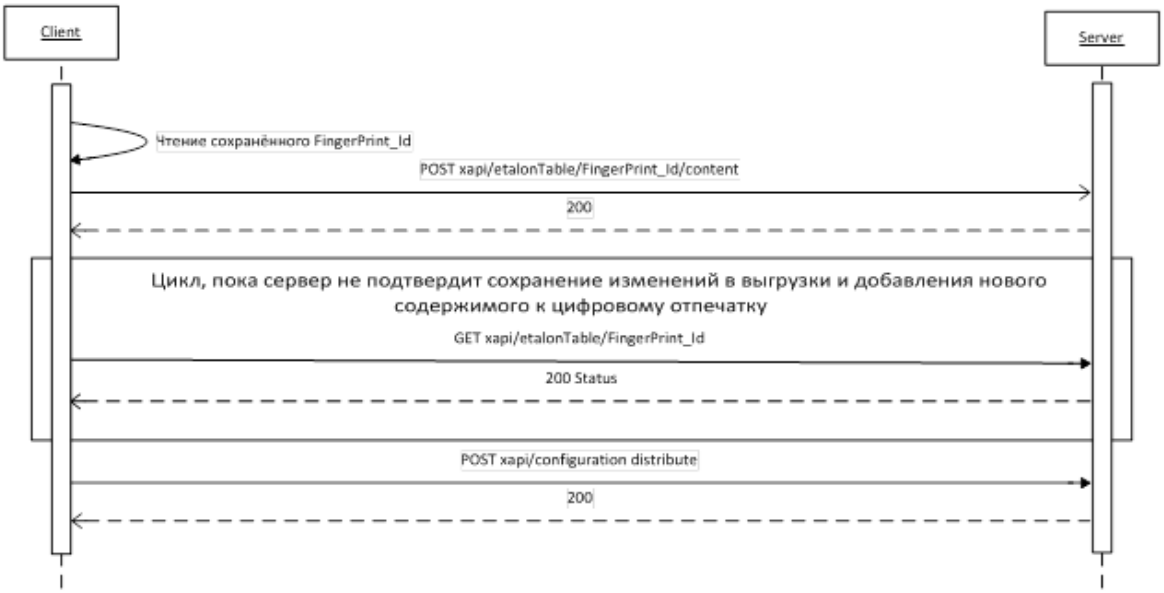
Ниже представлены диаграммы последовательностей, которые отражают типовые сценарии использования REST API:

- Типовой сценарий создания эталонной выгрузки;
- Типовой сценарий обновления эталонной выгрузки;
- Типовой сценарий добавления данных к выгрузке;
- Реализация завершения скачивания эталонной выгрузки.

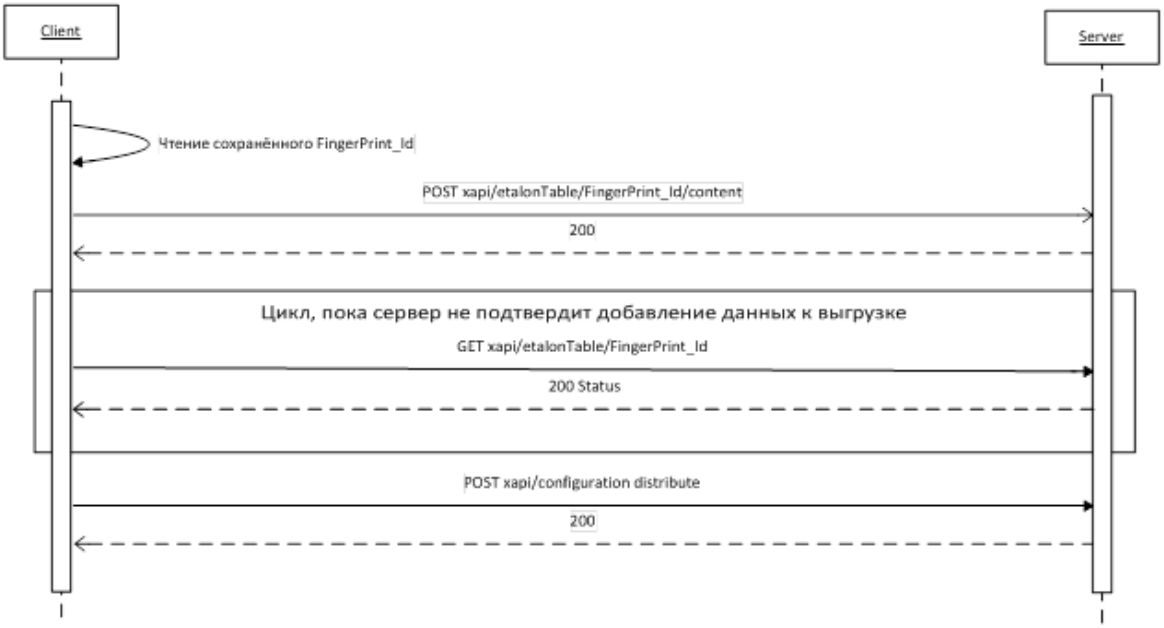
## 6.1 Типовой сценарий создания эталонной выгрузки



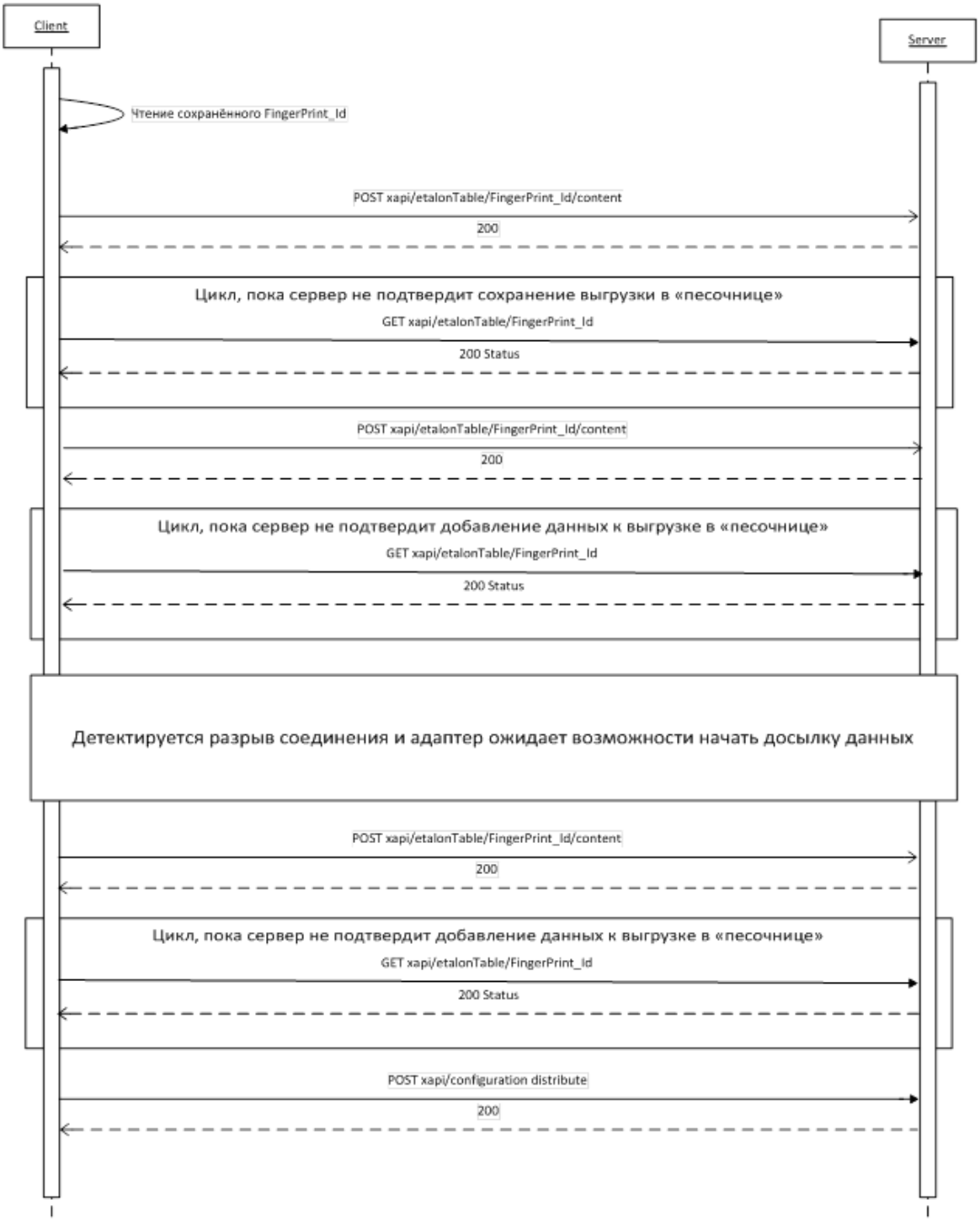
## 6.2 Типовой сценарий обновления эталонной выгрузки




## 6.3 Типовой сценарий добавления данных к выгрузке



## 6.4 Реализация завершения скачивания эталонной выгрузки



# 7 Функциональное описание программного интерфейса REST API

Программный интерфейс представляет собой REST API, т.е. последовательность GET/POST-запросов к Web-серверу на определённый ресурс и ответов на них. Формат оформления запросов и формат ответов представлен ниже. Защита данных во время их передачи осуществляется по протоколу SSL/TLS без авторизации клиента. Передача данных в открытом виде программным интерфейсом не поддерживается. При передаче данных используется кодировка UTF-8.

## 7.1 Общие заголовки GET/POST-запросов

Каждый запрос к серверу должен содержать следующие обязательные заголовки:

- X-API-Auth-Token – указывается токен авторизации;
- X-API-DataType – указывается идентификатор передаваемого типа данных;
- X-API-CompanyId – указывается идентификатор компании-производителя;
- X-API-Version – указывается номер версии программного интерфейса. Сейчас используется номер: «1».

## 7.2 Работа с конфигурацией REST API

После того, как содержимое эталонной выгрузки передано на сервер SDK и удачно «скомпилировано», можно дать команду на распространение изменённой конфигурации на системы анализа.

### 7.2.1 Распространение конфигурации REST API

Описание: метод нужен для распространения конфигурации на системы анализа.

Характеристики метода: POST, синхронный.

Ресурс: xapi/configuration..

JSON Scheme тела запроса:

```
{
    "STATUS": "distribute"
}
```

В случае успеха – код возврата «200» и тело ответа:

```
{
    "STATUS": "distributed"
}
```

В случае ошибки – код возврата «500». Возможные значение поля code:

- error – конфигурация не может быть применена.

## 7.3 Работа с эталонными выгрузками REST API

Глава содержит следующую информацию:

- Создание выгрузки REST API;

- Обновление данных выгрузки REST API;
- Добавление данных выгрузки REST API;
- Получение состояния выгрузки REST API;
- Получение текущей версии REST API.

### 7.3.1 Создание выгрузки REST API

Описание: метод нужен для добавления новой выгрузки без наполнения её данными. Сами данные будут добавлены через отдельный метод.

Характеристики метода: POST, синхронный.

Ресурс: xapi/etalonTable.

JSON Scheme тела запроса:

```
{
    "properties": {
        "DISPLAY_NAME": {
            "description": "Название выгрузки",
            "type": "string",
            "errors": ["required", "not_match", "not_unique_field"]
        },
        "CONDITION_COLUMNS": {
            "description": "строка, закодированный JSON-массив с названием столбцов,
например: [\"first_column\", \"second_column\"]",
            "type": "string",
            "errors": ["required", "not_valid_json"]
        },
        "NOTE": {
            "description": "Произвольное описание выгрузки",
            "type": "string"
        }
    },
    "required": [
        "DISPLAY_NAME",
        "CONDITION_COLUMNS"
    ]
}
```

В случае успеха – код возврата «200» и тело ответа:

```
{
    "FINGERPRINT_ID":    "string",
    "TYPE":              "string",
```

```
    "SOURCE":            "string",
    "DISPLAY_NAME":      "string",
    "NOTE":              "string",
    "CONDITION_COLUMNS": "string",
    "CREATE_DATE":       "string",
    "CHANGE_DATE":       "string",
    "STATUS":            "string"
}
```

**Важно!**

Число столбцов, которые указываются при создании выгрузки, не может быть изменено. При загрузке содержимого число ячеек в загружаемых стоках должно совпадать с тем значением, которое было указано при создании эталонной выгрузки.

### 7.3.2 Обновление данных выгрузки REST API

Описание: метод предназначен для обновления содержимого выгрузки. Все записи эталонной выгрузки будут удалены, и после этого добавится новая запись. Данный метод можно использовать для первичного наполнения.

**Важно!**

Наполнение должно иметь целое число строк.
Минимальное число строк наполнения – 2 штуки.
Минимальное количество колонок – 2 штуки.
Минимальный байтовый размер загружаемого контента – 128 байт.
Максимальный байтовый размер загружаемого контента – 2Gb

Характеристики метода: POST, асинхронный.

Ресурс: xapi/etalonTable/{id}/content/replace.

Параметры:

- id – fingerprint_id редактируемого объекта.

Например: enatalonTable/0123456789ABCDEF/content/replce

Content-type тела запроса должно быть не application/json, а multipart/form-data (более подробно можно узнать тут: https://ru.wikipedia.org/wiki/Multipart/form-data).

Content-Disposition, который будет содержать файл, должно иметь поле `name` со значением CONTENT и атрибут `filename` с названием файла с расширением tsv или csv в зависимости от формата файла, разделенного Tab или запятыми соответственно.

Возможные значение ошибок поля CONTENT:

- not_valid – файл имеет неверную сигнатуру;
- not_allowed_file_extension – неразрешенное расширение файла, на данный момент разрешены 2 формата: tsv и csv;
- too_long_size – слишком большой размер файла;

- duplicate – такой файл уже существует в системе.

В случае успеха – код возврата «200» и тело ответа:

```
{
    "FINGERPRINT_ID":    "string",
    "TYPE":              "string",
    "SOURCE":            "string",
    "DISPLAY_NAME":      "string",
    "NOTE":              "string",
    "CONDITION_COLUMNS": "string",
    "CREATE_DATE":       "string",
    "CHANGE_DATE":       "string",
    "STATUS":            "string"
}
```

После получения кода возврата «200» нужно взять поле FINGERPRINT_ID и по нему проверять состояние выгрузки, пока статус не перейдет в состояние ready.

**Важно!**

Число ячеек в загружаемых строках должно совпадать с тем значением числа столбцов, которое было указано при создании выгрузки.
При удалении выгрузки через консоль TrafficMonitor она считается существующей до момента распространения конфигурации, т.е. с этой выгрузкой возможны операции обновления и добавления содержимого.
При загрузке «ошибочного» содержимого, т.е. содержимого, которое является дубликатом уже загруженной выгрузки (части выгрузки) или содержимого, строки которого содержат неправильное количество ячеек, весь эталонный объект получает не валидный статус. Внешняя подсистема должна перестать загружать такое содержимое и перейти к следующему содержимому. Загрузка корректного содержимого исправляет общий статус эталонного объекта. Некорректное содержимое никогда не попадает на модули анализа.

### 7.3.3 Добавление данных выгрузки REST API

Описание: метод предназначен для добавления содержимого в выгрузку. Все записи содержимого будут сохранены, будет добавлена новая запись.

Характеристики метод: POST.

Ресурс: xapi/etalonTable/{id}/content.

Подробности в статье "Обновление данных выгрузки".

### 7.3.4 Получение состояния выгрузки REST API

Описание: метод нужен для получения сведений о состоянии одной выгрузки.

Характеристики метода: GET.

Ресурс: xapi/etalonTable/{id}.

Параметры:

- id – fingerprint_id редактируемого объекта.

Например: etalonTable/0123456789ABCDEF

Тело ответа:

```
{
    "FINGERPRINT_ID":    "string",
    "TYPE":              "string",
    "SOURCE":            "string",
    "DISPLAY_NAME":      "string",
    "NOTE":              "string",
    "CONDITION_COLUMNS": "string",
    "CREATE_DATE":       "string",
    "CHANGE_DATE":       "string",
    "STATUS":            "string"
    "content":           [
        {
            "CONTENT_ID":   "string",
            "CHECKSUM":     "string",
            "CONTENT_SIZE": "integer",
            "CONTENT_MIME": "string"
        }
    ]
}
```

Поле статус может иметь следующие значения:

- sending – отправка данных на компилятор;
- sent – данные отправлены на компилятор, и начинается компиляция;
- compiling – идёт компиляция документа;
- compiled – документ готов;
- saving – идет сохранение;
- ready – документ сохранён, готов для дальнейшего внесения в него изменений и не блокирует применение конфигурации;
- fatal_error – произошла ошибка во время сохранения документа, и операция не может быть повторена или продолжена;
- duplicate – произошло дублирование данных content (можно продолжать загружать содержимое);
- not_match_delimiter – строки в выгрузке не содержат разделителя символов (COMMA или TAB);
- wrong_column_count – количество колонок в строке не совпадает с количеством колонок, которое было указано при создании выгрузки;
- error – временная ошибка, можно попробовать ещё раз через некоторое время (лучше через час);
- max_words - при превышении допустимого количества слов;

- max_columns - при превышении допустимого количества столбцов.

### 7.3.5 Получение текущей версии REST API

Описание: метод нужен для получения поддерживаемых версий.

Если код возврата будет «200», то можно продолжать работу.

Метод может использоваться для проверки лицензии, версии и авторизации. В случае успеха – код возврата «200». В случае неисправности возвращается код ошибки.

Характеристики метод: GET.

Ресурс: xapi/version.

Тело ответа:

```
{
    "data":       [
        {
            "NAME":    "string",
            "STATUS": "supported|deprecated|removed"
        }
    ]
}
```

Пример ответа при указании неправильной версии:

HTTP/1.1 400 Bad Request

Content-Type: application/json

```
{
    "error": {
        "code": "unsupported_version",
        "meta": {
            "versions": [
                {
                    "NAME":   "1.0.0",
                    "STATUS": "removed"
                },
                {
                    "NAME":   "2.0.0",
                    "STATUS": "deprecated"
                },
                {
                    "NAME":   "3.0.0",
                    "STATUS": "supported"
```

```
                }
            ]
        }
    }
}
```

## 7.4 Общие ошибки REST API

Описание возможных ошибок:

- not_unique_field – не уникальное значение;
- unique – уникальное значение;
- too_short – слишком короткое значение;
- too_long – слишком длинное значение;
- wrong_length – значение неправильной длины;
- repeat_exactly – значение не совпадает со значением в другом поле;
- required – у поля обязательно должно быть значение;
- not_exist_in_list – значение не входит в список;
- invalid – значение не валидно;
- contains_child – узел является не листовым;
- contains_items – узел категории содержит сущности;
- parent_own_child – родитель содержит детей;
- parent_self_set_current – попытка установить родителем самого себя;
- not_valid_json – JSON не валиден;
- invalid_cron_format – не правильный формат cron;
- read_only – объект доступен только для чтения;
- empty – значение не может быть пустым.

### 7.4.1 Успешный ответ REST API

Код возврата: «200».

Формат ответа:

```
{
    "data": "mixed",
    "meta": "mixed"
}
```

### 7.4.2 Ошибка валидации данных/версии REST API

Код возврата: «400».

#### Формат ответа в случае неправильной версии API (JSON Scheme) REST API

```
{
    "properties": {
        "error": {
```

```
            "description": "Контейнер ошибки",
            "properties":  {
                "code":       {
                    "description": "Общее описание ошибки",
                    "type":        "string",
                    "value":       "unsupported_version"
                },
                "meta": {
                    "description": "Дополнительные данные об ошибке",
                    "type":        "object",
                    "properties":  {
                        "supported_versions": {
                            "description": "Массив поддерживаемых версий",
                            "type":        "array",
                            "items": {
                                "description": "Поддерживаемая версия",
                                "type": "string"
                            }
                        }
                    }
                }
            }
        }
    }
}
```

Формат ответа в случае ошибки валидации запроса (JSON Scheme) REST API

```
{
    "properties": {
        "error": {
            "description": "Контейнер ошибки",
            "properties":  {
                "code":       {
                    "description": "Общее описание ошибки",
                    "type":        "string",
                    "value":       "validation"
                },
```

```
"meta": {
    "description": "Дополнительные данные об ошибки",
    "type":        "object",
    "properties":  {
        "validation": {
            "description": "Название полей модели",
            "type":        "object",
            "properties":  {
                "field":     {
                    "description": "Поле модели",
                    "type":        "array",
                    "items":       {
                        "description": "Ошибка",
                        "type":        "string",
                        "values": [
                            "not_unique_field",
                            "too_short",
                            "too_long",
                            "wrong_length",
                            "repeat_exactly",
                            "required",
                            "not_exist_in_list",
                            "invalid",
                            "contains_child",
                            "contains_items",
                            "parent_own_child",
                            "unique",
                            "not_valid_json",
                            "invalid_cron_format",
                            "read_only",
                            "parent_self_set_current",
                            "empty"
                        ]
                    }
                },
                "sub_model": {
                    "description": "Название вложенной модели",
```

```
                                    "type":        "object",
                                    "$ref":       "#/properties/error/properties/meta/
properties/validation"
                                }
                            }
                        }
                    }
                }
            }
        }
}
```

Пример ошибки при указании неправильной версии REST API

```
{
    "error": {
        "code": "unsupported_version",
        "meta": {
            "supported_versions": ["2", "3"]
        }
    }
}
```

Пример ошибки проверки на уникальность имени эталонной выгрузки REST API

```
{
    "error": {
        "code": "validation",
        "meta": {
            "validation":    {
                "DISPLAY_NAME": ["unique"]
            },
            "unique_models": [
                {
                    "FINGERPRINT_ID": "0123456789ABCDEF",
                    "DISPLAY_NAME":   "test"
                }
            ]
```

```
        }
    }
}
```

### 7.4.3 Необходима авторизация REST API

Код возврата: «401».

Описание: возвращается в любом запросе, где требуется авторизация.

Тело ответа: нет.

### 7.4.4 Недостаточно прав REST API

Код возврата: «403».

Описание: возвращается в любом запросе, если у текущего пользователя недостаточно прав для выполнения данной операции.

Тело ответа: нет.

### 7.4.5 Сущность в системе не найдена REST API

Код возврата: «404».

Описание: возвращается, если определенная модель не найдена в системе. Например, не найдена эталонная выгрузка.

Тело ответа: нет.

### 7.4.6 Тело запроса слишком большое REST API

Код возврата: «413».

Описание: возвращается, если передан слишком большой запрос.

Тело ответа: нет.

### 7.4.7 Превышено количество запросов к серверу REST API

Код возврата: «429».

Описание: возвращается, если на сервер отправлено слишком много запросов за определенный промежуток времени. Может сопровождаться заголовком Retry-After, указывающим, через какое время можно повторить запрос.

Тело ответа (JSON Scheme):

```
{
    "properties": {
        "error": {
            "description": "Контейнер ошибки",
            "properties":  {
                "code": {
```

```
                    "description": "Общее описание ошибки",
                    "type":        "string",
                    "value":       "too_many_requests "
                },
                "meta": {
                    "description": "Дополнительные данные об ошибки",
                    "type":        "object",
                    "properties":  {
                        "time": {
                            "description": "HTTP-дата, либо целое число в секундах, через
которые можно повторить запрос",
                            "type":        "string"
                        }
                    }
                }
            }
        }
    }
}
```

### 7.4.8 Ошибка системы REST API

Код возврата: «500».

Описание: возвращается, если в системе произошла логическая ошибка, и далее невозможно выполнение запроса.

Тело ответа (JSON Scheme):

```
{
    "properties": {
        "error": {
            "description": "Контейнер ошибки",
            "properties":  {
                "code": {
                    "description": "Общее описание ошибки",
                    "type":        "string"
                },
                "meta": {
                    "description": "Дополнительные данные об ошибки",
                    "type":        "object"
                }
```

```
            }
        }
    }
}
```

### 7.4.9 Сервер временно не доступен REST API

Код возврата: «503».

Описание: информирует, что в данный момент невозможно выполнить запросы, и сервер находится на техническом обслуживании. Приблизительное время окончания технического обслуживания содержится в заголовке Retry-After (значением этого заголовка может быть либо HTTP-дата, либо целое число в секундах).

Тело ответа (JSON Scheme):

```
{
    "properties": {
        "error": {
            "description": "Контейнер ошибки",
            "properties":  {
                "code": {
                    "description": "Общее описание ошибки",
                    "type":        "string",
                    "value":       "maintains"
                },
                "meta": {
                    "description": "Дополнительные данные об ошибки",
                    "type":        "object",
                    "properties":  {
                        "time": {
                            "description": "HTTP-дата, либо цело число в секундах, через
которые можно повторить запрос",
                            "type":        "string"
                        }
                    }
                }
            }
        }
    }
}
```