Загрузка эталонных документов и выгрузок из БД в Traffic Monitor (REST API SDK)

03/02/2025

Оглавление

1 Загрузка эталонных документов и выгрузок из БД в Traffic Monitor (REST API SDK)

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

Документ поможет реализовать механизм автоматической загрузки эталонных документов и выгрузок из БД средствами API. Документ предназначен для разработчиков - сотрудников организации, ее технологических партнеров или подрядных организаций.

1.1 Конфигурирование программного интерфейса REST API

В конфигурировании программного интерфейса участвуют следующие параметры:

- Адрес подключения с сервису SDK (сетевой адрес, порт);
- Токен авторизации. Строка символов, которая позволяет авторизовать компоненты, использующие SDK. Данная строка может быть получена в разделе "Управление" (вкладка "Токены") 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.

1.2 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 и эталонные документы указанного производителя. Лицензия разработчика позволяет загружать произвольные эталонные выгрузки и эталонные документы.
3. Создать файл регистрации стороннего компонента в описанном ниже формате. К этому моменту должны быть определены названия источника данных эталонной выгрузки и эталонного документа, которая будет формироваться сторонним компонентом. В файл регистрации добавляется лицензия, полученная на шаге 2.
4. Загрузить файл регистрации стороннего компонента в систему Traffic Monitor через интерфейс консоли Traffic Monitor. Раздел «Управление» - «Плагины». (см. "InfoWatch Traffic Monitor. Руководство пользователя").
5. Получить токен доступа в свойствах загруженного стороннего компонента.

Система готова принимать сформированные эталонные выгрузки и эталонные документы от заданной компании-производителя с указанным источником эталонной выгрузки или эталонного документа.

1.2.1 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;
- файл **manifest.json**, содержащий информацию о плагине.

> **Примечание:**
> Папки **licenses** и **icon** не являются обязательными. Файлы лицензий и файлы с используемыми пиктограммами могут находиться в корне.

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

Содержимое	Тип данных	Является обязательным	Описание
`{`			
`"PLUGIN_ID": "",`	Строка	Да	Уникальный идентификатор (UUID) плагина. До 40 символов. **Пример:** `"PLUGIN_ID": "189C38D390396EB6E0530100007F1CA200000001",`
`"DISPLAY_NAME": "",`	Строка	Да	Отображаемое имя плагина. **Пример:** `"DISPLAY_NAME": "Имя плагина",`
`"DESCRIPTION": "",`	Строка	Нет	Отображаемое описание плагина. **Пример:** `"DESCRIPTION": "Тестовый плагин для демонстрации",`
`"VERSION": "",`	Строка	Да	Версия плагина. **Пример:** `"VERSION": "1.0.0",`
`"VENDOR": "",`	Строка	Да	Идентификатор компании-разработчика, соответствующий названию компании в лицензии. **Пример:** `"VENDOR": "infowatch",`

`"LICENSE": [`
`{`
`"PATH":""`
`}`
`],`
	Массив
	
	Строка
		Да	Файлы лицензии. Должны быть указаны пути к файлам лицензий относительно корня архива.
			`"LICENSE": [`
			`{`
			`"PATH":"licenses/файл лицензии 1.license"`
			`},`
			`{`
			`"PATH":"licenses/файл лицензии 2.license"`
			`},`
			`],`
			Code Block 1 Пример:

Содержимое	Тип данных	Является обязательным	Описание

"PATTERN_SEARCH_LICENSE": "",	Строка	Нет	Шаблон для поиска загруженных ранее лицензий для привязки их к плагину. Имеет формат: *{operator:or|and,child[{name:value}]}*
			".
			"PATTERN_SEARCH_LICENSE": {
			"operator": "or",
			"conditions": [{
			"origin": "dm"
			}, {
			"origin": "dmmobile"
			}
			]
			},
			Code Block 2 Пример operator:or
			"PATTERN_SEARCH_LICENSE": {
			"operator": "or",
			"conditions": [
			{
			"operator": "and",
			"conditions": [
			{"key1": "value1"},
			{"key2": "value2"},
			{"key3": "value3"}
			]
			},
			{
			"operator": "and",
			"conditions": [
			{"key1": "value1"},
			{"key2": "value2"},
			{"key3": "value3"}
			]
			}
			]
			}
			Code Block 3 Пример operator:and

Содержимое	Тип данных	Является обязательным	Описание

			<pre>"PATTERN_SEARCH_LICENSE": {
			"operator": "and",
			"conditions": [
			{"key1": "value1"},
			{"key2": "value2"},
			{"key3": "NONE"},
			{
			"operator": "not",
			"conditions": [
			{"key4": null}
			]
			}
			]
			}</pre> **Code Block 4 Пример operator:not**

*"DATATYPE": [
{
"VALUE": ""
}
]*
	Массив
	
	Строка
		Да	Список типов данных с указанием систем-источников.
			Пример:
			*"DATATYPE": [
			{
			"VALUE": "type1_s1","VALUE": "type2_s2"
			}
			],*

}			

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

1.3 Формат эталонной выгрузки и эталонного документа 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). Данные ограничения соответствуют 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 невалидной.

Эталонный документ представляет собой файл любого формата. Если файл является незашифрованным архивом, Система извлекает файлы из архива, обрабатывает их независимо друг от друга и создает отдельный эталонный документ на основе каждого извлеченного файла (эталонный документ на основе всего архива не создается). Все созданные эталонные документы помещаются в каталог, выбранный при создании эталонного документа. Если файл является зашифрованным архивом, Система не создает эталонный документ. Если файл является файлом типа "контейнер", Система создает один эталонный документа на основе файла.

1.4 Функциональное описание программного интерфейса REST API

Программный интерфейс представляет собой REST API, т.е. последовательность GET/POST-запросов к Web-серверу на определенный 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.

1.4.1 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».

1.4.2 Общие ошибки REST API

Описание возможных ошибок:

- not_unique_field – неуникальное 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 невалидно;
- 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 невалиден;
- invalid_cron_format – не правильный формат cron;
- read_only – объект доступен только для чтения;
- empty – значение не может быть пустым.

1.4.2.1 Успешный ответ REST API

Код возврата: «200».

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

Code Block 5 Формат ответа:

1.4.2.2 Ошибка валидации данных/версии REST API

Код возврата: «400».

```
{
    "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"
                            }
                        }
                    }
                }
            }
        }
    }
}
```

Code Block 6 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":       "validation"
                },
                "meta": {
                    "description": "Дополнительные данные об ошибке",
                    "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/m
eta/properties/validation"
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

Code Block 7 Формат ответа в случае ошибки валидации запроса (JSON Scheme) REST API

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

Code Block 8 Пример ошибки при указании неправильной версии REST API

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

Code Block 9 Пример ошибки проверки на уникальность имени эталонной выгрузки REST API

1.4.2.3 Необходима авторизация REST API

Код возврата: «401».

Описание: возвращается в любом запросе, где требуется авторизация.

Тело ответа: нет.

1.4.2.4 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.

1.4.2.5 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.

1.4.2.6 Тело запроса слишком большое REST API

Код возврата: «413».

Описание: возвращается, если передан слишком большой запрос.

Тело ответа: нет.

1.4.2.7 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.

```
{
    "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"
                        }
                    }
                }
            }
        }
    }
}
```

Code Block 10 Тело ответа (JSON Scheme):

1.4.2.8 Ошибка системы REST API

Код возврата: «500».

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

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

Code Block 11 Тело ответа (JSON Scheme):

1.4.2.9 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).

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

Code Block 12 Тело ответа (JSON Scheme):

1.4.3 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.

1.4.3.1 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.

1.4.4 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.

1.4.4.1 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*.

```
{
    "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"
    ]
}
```

Code Block 13 JSON Scheme тела запроса:

В случае успеха – код возврата «200»

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

Code Block 14 Тело ответа:

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

1.4.4.2 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.

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` (подробнее см. https://ru.wikipedia.org/wiki/Multipart/form-data).

Content-Disposition, который будет содержать файл, должен 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.

Важно!
Наполнение должно иметь целое число строк, минимум 2 строки.
Минимальное количество колонок – 2.
Минимальный байтовый размер загружаемого контента – 128 байт.
Максимальный байтовый размер загружаемого контента – 2 ГБ.

В случае успеха – код возврата «200»

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

Code Block 15 Тело ответа:

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 со 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 невалидный 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.

1.4.4.3 Добавление данных выгрузки REST API

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

Характеристики метода: POST.

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

Подробнее см. статью "Обновление данных выгрузки REST API"

1.4.4.4 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`

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

Code Block 16 Тело ответа:

Поле STATUS 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 - превышено допустимое количество слов;
- max_columns - превышено допустимое количество столбцов.

1.4.4.5 Получение текущей версии REST API

Описание: метод нужен для получения поддерживаемых версий. Метод может использоваться для проверки лицензии, версии и авторизации. В случае успеха – код возврата «200», и можно продолжать работу. В случае неисправности возвращается код ошибки.

Характеристики метода: GET.

Ресурс: *xapi/version*.

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

Code Block 17 Тело ответа:

```
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"
                }
            ]
        }
    }
}
```

Code Block 18 Пример ответа при указании неправильной версии:

1.4.5 Работа с эталонными документами REST API

Глава содержит следующую информацию:

- Создание каталога эталонных документов REST API;
- Просмотр каталогов эталонных документов REST API;
- Редактирование каталога эталонных документов REST API;
- Удаление каталога эталонных документов REST API;
- Создание эталонного документа REST API;
- Просмотр эталонных документов REST API;
- Обновление и замена содержимого эталонного документа REST API;
- Удаление эталонного документа REST API.

1.4.5.1 Создание каталога эталонных документов REST API

Метод: `POST`

Ресурс: `etalonDocumentCategory`

Тело запроса :

```
{    "$schema": "http://json-schema.org/draft-04/schema#",    "definitions":
{},    "id": "http://infowatch.com/tm/etalonDocumentCategory/edit",    "type"
: "object",    "properties": {        "PARENT_CATEGORY_ID":
{            "examples":
[                "AC3DF073B60107890A4C704A10577FE700000000"            ],
        "title": "GUID категории родителя, если null, то проставляется папка
автоматических эталонных
документов",            "type": "string"        },        "DISPLAY_NAME":
{            "title": "Название",            "type": "string"        },
  "DIR_PATH (возможно изменение в названии)": {            "title": "Полное
имя
директории",            "type": "string"        },        "FP_BIN_VALUE_THRES
HOLD": {            "title": "Порог цитируемости бинарных
данных",            "type": "integer"        },        "FP_TEXT_VALUE_THRESHO
LD": {            "title": "Порог цитируемости текстовых
данных",            "type": "integer"        },        "NOTE":
{            "title": "Описание",            "type":
[                "null",                "string"            ]        }    }}
```

Тело ответа :

```
{    "$schema": "http://json-schema.org/draft-04/schema#",    "definitions":
{},    "id": "http://infowatch.com/tm/etalonDocumentCategory",    "type": "ob
ject",    "properties": {        "CATEGORY_ID": {            "examples":
[                "AC3DF073B60107890A4C704A10577FE700000000"            ],
        "title": "GUID
категории",            "type": "string"        },        "PARENT_CATEGORY_ID"
: {            "examples":
[                "AC3DF073B60107890A4C704A10577FE700000000"            ],
        "title": "GUID категории
родителя",            "type": "string"        },        "DISPLAY_NAME":
{            "title": "Название",            "type": "string"        },
  "DIR_PATH (возможно изменение в названии)": {            "title": "Полное
имя
директории",            "type": "string"        },        "FP_BIN_VALUE_THRES
HOLD": {            "title": "Порог цитируемости бинарных
данных",            "type": "integer"        },        "FP_TEXT_VALUE_THRESHO
LD": {            "title": "Порог цитируемости текстовых
```

```
данных",            "type": "integer"        },        "NOTE":
{            "title": "Описание",            "type":
[                "null",                "string"            ]        },
  "OWNER": {            "title": "ID
токена",            "type": "integer"        },        "CREATE_DATE":
{            "examples": [                "2017-07-04
10:45:45.000000"            ],            "title": "Дата
создания",            "type": "string"        },        "CHANGE_DATE":
{            "examples": [                "2017-07-04
10:45:45.000000"            ],            "title": "Дата
изменения",            "type": "string"        }    }}
```

1.4.5.2 Просмотр каталогов эталонных документов REST API

Метод: `GET`

Ресурс: `etalonDocumentCategory[/{category_id}]`

Параметры:

- `filter` — список фильтрации:
  - `create_date[from]` — начало диапазона для фильтрации по дате создания в формате UNIX-timestamp (включая передаваемую дату, можно использовать отдельно от `create_date[to]`)
  - `create_date[to]` — окончание диапазона для фильтрации по дате создания в формате UNIX-timestamp (включая передаваемую дату, можно использовать отдельно от `create_date[from]`)
  - `change_date[from]` — начало диапазона для фильтрации по дате изменения в формате UNIX-timestamp (включая передаваемую дату, можно использовать отдельно от `change_date[to]`)
  - `change_date[to]` — окончание диапазона для фильтрации по дате изменения в формате UNIX-timestamp (включая передаваемую дату, можно использовать отдельно от `change_date[from]`)
  - `display_name[]` — имя каталога, можно использовать * в конце, чтобы включить поиск по `LIKE`, например: `?filter[display_name][]=foo*`
  - `dir_path[]` — полный путь до каталога, можно использовать * в конце, чтобы включить поиск по `LIKE`, например: `?filter[dir_path][]=foo*`
  - `fingerprint_id[]` — GUID эталонного документа, например: `?filter[fingerprint_id][]=82EAE496A1686E407B1162E3C3159999F404F559&filter[fingerprint_id][]=AF8EAC3C3C8532C3C780481B7D8C5B8E68D0148F`
- `with[]` — список дополнительных сущностей, которые должны быть добавлены к объекту:
  - `etalonDocumentsCount` — количество эталонных документов в каталоге
- `sort[]` — поле, позволяющее сортировать результат. Возможные ключи:
  - `create_date` — `desc/asc`, например: `?sort[create_date]=desc`
  - `change_date` — `desc/asc, например: ?sort[change_date]=asc`

Ответ:

Массив объектов, отдаваемых при создании/редактировании

Примеры :

```
GET
/xapi/etalonDocumentCategory?filter[create_date][to]=1&filter[display_name][]
=foo*&filter[display_name][]=bar&with[]=etalonDocumentsCount&sort[create_date
```

```
]=descGET
/xapi/etalonDocumentCategory/FAB85F61531BB9E428088EFC81F266FA3C2959E6
```

1.4.5.3 Редактирование каталога эталонных документов REST API

Метод: PUT

Ресурс: *etalonDocumentCategory/{category_id}*

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "definitions": {},
    "id": "http://infowatch.com/tm/etalonDocumentCategory/edit",
    "type": "object",
    "properties": {
        "PARENT_CATEGORY_ID": {
            "examples": [
                "AC3DF073B60107890A4C704A10577FE700000000"
            ],
            "title": "GUID категории родителя, если null, то проставляется папка
автоматических эталонных документов",
            "type": "string"
        },
        "DISPLAY_NAME": {
            "title": "Название",
            "type": "string"
        },
        "DIR_PATH (возможно изменение в названии)": {
            "title": "Полное имя директории",
            "type": "string"
        },
        "FP_BIN_VALUE_THRESHOLD": {
            "title": "Порог цитируемости бинарных данных",
            "type": "integer"
        },
        "FP_TEXT_VALUE_THRESHOLD": {
            "title": "Порог цитируемости текстовых данных",
            "type": "integer"
        },
        "NOTE": {
            "title": "Описание",
            "type": [
                "null",
                "string"
            ]
        }
    }
}
```

Code Block 19 Тело запроса:

```

```

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "definitions": {},
    "id": "http://infowatch.com/tm/etalonDocumentCategory",
    "type": "object",
    "properties": {
        "CATEGORY_ID": {
            "examples": [
                "AC3DF073B60107890A4C704A10577FE700000000"
            ],
            "title": "GUID категории",
            "type": "string"
        },
        "PARENT_CATEGORY_ID": {
            "examples": [
                "AC3DF073B60107890A4C704A10577FE700000000"
            ],
            "title": "GUID категории родителя",
            "type": "string"
        },
        "DISPLAY_NAME": {
            "title": "Название",
            "type": "string"
        },
        "DIR_PATH (возможно изменение в названии)": {
            "title": "Полное имя директории",
            "type": "string"
        },
        "FP_BIN_VALUE_THRESHOLD": {
            "title": "Порог цитируемости бинарных данных",
            "type": "integer"
        },
        "FP_TEXT_VALUE_THRESHOLD": {
            "title": "Порог цитируемости текстовых данных",
            "type": "integer"
        },
        "NOTE": {
            "title": "Описание",
            "type": [
                "null",
                "string"
            ]
        },
        "OWNER": {
            "title": "ID токена",
            "type": "integer"
        },
        "CREATE_DATE": {
            "examples": [
                "2017-07-04 10:45:45.000000"
            ],
            "title": "Дата создания",
            "type": "string"
        },
        "CHANGE_DATE": {
            "examples": [
                "2017-07-04 10:45:45.000000"
            ],
            "title": "Дата изменения",
            "type": "string"
        }
    }
}
```


Code Block 20 Тело ответа:

```
curl 'https://example.com/xapi/etalonDocumentCategory/0123456789ABCDEF123400000000
' \
  -X PUT \
  -H 'X-API-Auth-Token: 1792jmclf7fer1ikuhby' \
  -H 'X-API-Version: 1' \
  -H 'X-API-CompanyId: IW' \
  -H 'X-API-DataType: sap' \
  -H 'Content-Type: application/json' \
  --data-
binary '{"PARENT_CATEGORY_ID":"0123456789ABCDEF123400000000","DISPLAY_NAME":"world
.txt","DIR_PATH":"c:\\hello\\world.txt","FP_BIN_VALUE_THRESHOLD":10,"FP_TEXT_VALUE
_THRESHOLD":10,"NOTE":null}'
```

Code Block 21 Пример редактирования каталога CURL

1.4.5.4 Удаление каталога эталонных документов REST API

Метод: *DELETE*
Ресурс: *etalonDocumentCategory/{category_id}*

```
curl -X 'DELETE'
'http://localhost:63342/xapi/etalonDocumentCategory/0123456789ABCDEF'
```

Code Block 22 Пример запроса

1.4.5.5 Создание эталонного документа REST API

Предполагается, что сначала создаётся эталонный документ, с помощью *POST /xapi/etalonDocument* со статусом new, а дальше происходит его компиляция, статус которой нужно проверять с помощью получения эталонного документа (*GET /xapi/etalonDocument/{id}*) и его отслеживания. При получении дубликатов эталонного документа (STATUS=duplicate), такие файлы будут удалены.

Метод: *POST*
Ресурс: *etalonDocument*

Поля запрса:

Название поля	Описание
DISPLAY_NAME	Имя эталонного документа (если не указано, то берётся из поля FILE)
FILE_PATH*	Путь к эталонному документу в источнике, например: • smb://example.com/top_secret/etalon.docx, • \\example\test\text.rtf, • ftp://example.com/foo/bar.txt, • oracle://example.com:1521/database/table/row, • https://examlpe.com/same/html/document По факту, это просто подсказка для автоматического загрузчика, по которой он может искать
NOTE	Описание эталонного документа

Название поля	Описание
BIN_VALUE_THRESHOLD	Порог цитируемости бинарных данных (по умолчанию берется из первого указанного каталога)
TEXT_VALUE_THRESHOLD	Порог цитируемости текстовых данных (по умолчанию берется из первого указанного каталога)
CATEGORY_ID*	Список GUID-ов категорий, разделенных запятой, например: "E77D893219A24EEAB3D049804FD86751,C743F88121B8449DAF37291474830679,A7AF8EFCF4D54E9393A9B3D750E5ABBB"
FILE*	Сам файл (передаётся сам файл, а не путь к нему. Сервер не сможет подключиться к вашему компьютеру и забрать его сам. Поэтому весь файл передаётся в запросе)
COMPILE_TYPE*	Тип цифрового отпечатка: • TEXT – извлекать только текст • ALL – извлекать и компилировать всё, что возможно

* – поле обязательно для заполнения

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "definitions": {},
    "id": "http://infowatch.com/tm/etalonDocument",
    "type": "object",
    "properties": {
        "FINGERPRINT_ID": {
            "examples": [
                "139C7026ED48BE13459669024786F7517E5687AE"
            ],
            "title": "Идентификатор эталонного документа",
            "type": "string"
        },
        "DISPLAY_NAME": {
            "title": "Название",
            "type": "string"
        },
        "FILE_PATH": {
            "title": "Полное имя файла",
            "type": "string"
        },
        "FILE_SIZE": {
            "title": "Размер файла в байтах",
            "type": "integer"
        },
        "MIME": {
            "title": "Mime-type файла",
            "type": "string"
        },
        "BIN_VALUE_THRESHOLD": {
            "title": "Порог цитируемости бинарных данных",
            "type": "integer"
        },
        "TEXT_VALUE_THRESHOLD": {
            "title": "Порог цитируемости текстовых данных",
            "type": "integer"
        },
        "NOTE": {
            "title": "Описание",
            "type": [
                "null",
                "string"
            ]
        },
        "OWNER": {
            "title": "ID токена",
            "type": "integer"
        },
        "STATUS": {
            "examples": [
                "new",
                "compiled",
                "compiling",
                "sending",
                "sent",
                "saving",
                "ready",
                "fatal_error",
                "duplicate",
                "error"
            ],
            "title": "Статус обработки",
            "type": "string"
        },
        "CREATE_DATE": {
            "examples": [
                "2017-07-04 10:45:45.000000"
            ],
            "title": "Дата создания",
```

```
            "type": "string"
        },
        "CHANGE_DATE": {
            "examples": [
                "2017-07-04 10:45:45.000000"
            ],
            "title": "Дата изменения",
            "type": "string"
        },
        "categories": {
            "type": "array",
            "title": "Массив категорий",
            "items": {
                "type": "object",
                "properties": {
                    "CATEGORY_ID": {
                        "examples": [
                            "AC3DF073B60107890A4C704A10577FE700000000"
                        ],
                        "title": "GUID категории",
                        "type": "string"
                    },
                    "DISPLAY_NAME": {
                        "title": "Название",
                        "type": "string"
                    },
                    "DIR_PATH (возможно изменение в названии)": {
                        "title": "Полное имя директории",
                        "type": "string"
                    },
                    "FP_BIN_VALUE_THRESHOLD": {
                        "title": "Порог цитируемости бинарных данных",
                        "type": "integer"
                    },
                    "FP_TEXT_VALUE_THRESHOLD": {
                        "title": "Порог цитируемости текстовых данных",
                        "type": "integer"
                    },
                    "NOTE": {
                        "title": "Описание",
                        "type": [
                            "null",
                            "string"
                        ]
                    },
                    "OWNER": {
                        "title": "ID токена",
                        "type": "integer"
                    },
                    "CREATE_DATE": {
                        "examples": [
                            "2017-07-04 10:45:45.000000"
                        ],
                        "title": "Дата создания",
                        "type": "string"
                    },
                    "CHANGE_DATE": {
                        "examples": [
                            "2017-07-04 10:45:45.000000"
                        ],
                        "title": "Дата изменения",
                        "type": "string"
                    }
                }
            }
        }
    }
}
```

Code Block 23 JSON-Schema ответа:

```
POST /xapi/etalonDocument HTTP/1.1
Host: tm.example.com
X-API-Auth-Token: brse99dugp1cdbr69axz
X-API-Version: 1
X-API-DataType: cap
X-API-CompanyId: liw
Content-Type: multipart/form-data; boundary=Asrf456BGe4h
Content-Length: (суммарный объём, включая дочерние заголовки)
(пустая строка)
(отсутствующая преамбула)
--Asrf456BGe4h
Content-Disposition: form-data; name="CATEGORY_ID"
(пустая строка)
2ABECC84F2360B94E0533D003C0A80AE00000000
--Asrf456BGe4h
Content-Disposition: form-data; name="COMPILE_TYPE"
(пустая строка)
ALL
--Asrf456BGe4h
Content-Disposition: form-data; name="FILE"; filename="foo.doc"
Content-Type: application/msword
(пустая строка)
(двоичное содержимое документа)
```

Code Block 24 Пример запроса:

```
curl -i -X POST \
   -H "Content-Type:multipart/form-data" \
   -H "X-API-Auth-Token:brse99dugp1cdbr69axz" \
   -H "X-API-Version:1" \
   -H "X-API-DataType:cap" \
   -H "X-API-CompanyId:liw" \
   -F "CATEGORY_ID=2ABECC84F2360B94E0533D003C0A80AE00000000" \
   -F "FILE=@\"./foo.doc\";filename=\"foo.doc\"" \
   -F "COMPILE_TYPE=ALL" \
 'https://tm.example.com/xapi/etalonDocument'
```

Code Block 25 Пример создания файла с помощью CURL :

1.4.5.6 Просмотр эталонных документов REST API

Метод: `GET`
Ресурс: `etalonDocument[/{fingerprint_id}]`
Параметры:

- `filter` — список фильтрации:
    - `create_date[from]` — начало диапазона для фильтрации по дате создания в формате UNIX-timestamp (включая передаваемую дату, можно использовать отдельно от `create_date[to]`)
    - `create_date[to]` — окончание диапазона для фильтрации по дате создания в формате UNIX-timestamp (включая передаваемую дату, можно использовать отдельно от `create_date[from]`)
    - `change_date[from]` — начало диапазона для фильтрации по дате изменения в формате UNIX-timestamp (включая передаваемую дату, можно использовать отдельно от `change_date[to]`)

- `change_date[to]` — окончание диапазона для фильтрации по дате изменения в формате UNIX-timestamp (включая передаваемую дату, можно использовать отдельно от `change_date[from]`)
- `display_name[]` — имя файла, можно использовать * в конце, чтобы включить поиск по `LIKE`, например: `?filter[display_name][]=foo*`
- `file_path[]` — полный путь до файла, можно использовать * в конце, чтобы включить поиск по `LIKE`, например: `?filter[file_path][]=foo*`
- `category_id[]` — GUID категории, например: `?filter[category_id][]=82EAE496A1686E407B1162E3C3159999F404F559&filter[category_id][]=AF8EAC3C3C8532C3C780481B7D8C5B8E68D0148F`
- `category_path[]` — полный путь до папки категории, можно использовать * в конце, чтобы включить поиск по `LIKE`, например: `?filter[category_path][]=bar*`

- `with[]` — список дополнительных сущностей, которые должны быть добавлены к объекту:
  - `categories` — список категорий, в которые входит эталонный документ
- `sort[]` — поле, позволяющее сортировать результат. Возможные ключи:
  - `create_date` — `desc/asc`, например: `?sort[create_date]=desc`
  - `change_date` — `desc/asc, например: ?sort[change_date]=asc`

Ответ:

Массив объектов, отдаваемых при создании/редактировании

Примеры :

```
GET /xapi/etalonDocument?filter[create_date][to]=1&filter[display_name][]=foo*&filter[display_name][]=bar&with[]=categories&sort[create_date]=descGET /xapi/etalonDocument/FAB85F61531BB9E428088EFC81F266FA3C2959E6
```

1.4.5.7 Обновление и замена содержимого эталонного документа REST API

Метод: POST

Ресурс: `etalonDocument/{fingerprint_id}/update`

Поля:

Название поля	Описание
DISPLAY_NAME	Имя эталонного документа (если не указано, то берётся из поля FILE)
FILE_PATH	Путь к эталонному документу в источнике, например: smb://example.com/top_secret/etalon.docx, \\example\test\text.rtf, ftp://example.com/foo/bar.txt, oracle://example.com:1521/database/table/row, https://examlpe.com/same/html/document
NOTE	Описание эталонного документа
BIN_VALUE_THRESHOLD	Порог цитируемости бинарных данных (по умолчанию берется из первого указанного каталога)
TEXT_VALUE_THRESHOLD	Порог цитируемости текстовых данных (по умолчанию берется из первого указанного каталога)

Название поля	Описание
CATEGORY_ID	Список GUID-ов категорий, разделенных запятой, например: "E77D893219A24EEAB3D049804FD86751,C743F88121B8449DAF37291474830679,A7AF8EFCF4D54E9393A9B3D750E5ABBB"
FILE	Сам файл (Передаётся сам файл, а не путь к нему. Сервер не сможет подключиться к вашему компьютеру и забрать его сам. Поэтому весь файл передаётся в запросе)
COMPILE_TYPE	Тип цифрового отпечатка: • TEXT — Извлекать только текст • ALL — Извлекать и компилировать всё что возможно

Передавать нужно только те поля, которые нужно изменить. Остальные поля нужно опускать.

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "definitions": {},
    "id": "http://infowatch.com/tm/etalonDocument",
    "type": "object",
    "properties": {
        "FINGERPRINT_ID": {
            "examples": [
                "139C7026ED48BE13459669024786F7517E5687AE"
            ],
            "title": "Идентификатор эталонного документа",
            "type": "string"
        },
        "DISPLAY_NAME": {
            "title": "Название",
            "type": "string"
        },
        "FILE_PATH": {
            "title": "Полное имя файла",
            "type": "string"
        },
        "FILE_SIZE": {
            "title": "Размер файла в байтах",
            "type": "integer"
        },
        "MIME": {
            "title": "Mime-type файла",
            "type": "string"
        },
        "BIN_VALUE_THRESHOLD": {
            "title": "Порог цитируемости бинарных данных",
            "type": "integer"
        },
        "TEXT_VALUE_THRESHOLD": {
            "title": "Порог цитируемости текстовых данных",
            "type": "integer"
        },
        "NOTE": {
            "title": "Описание",
            "type": [
                "null",
                "string"
            ]
        },
        "OWNER": {
            "title": "ID токена",
            "type": "integer"
        },
        "STATUS": {
            "examples": [
                "new",
                "compiled",
                "compiling",
                "sending",
                "sent",
                "saving",
                "ready",
                "fatal_error",
                "duplicate",
                "error"
            ],
            "title": "Статус обработки",
            "type": "string"
        },
        "CREATE_DATE": {
            "examples": [
                "2017-07-04 10:45:45.000000"
            ],
            "title": "Дата создания",
```

```
            "type": "string"
        },
        "CHANGE_DATE": {
            "examples": [
                "2017-07-04 10:45:45.000000"
            ],
            "title": "Дата изменения",
            "type": "string"
        },
        "categories": {
            "type": "array",
            "title": "Массив категорий",
            "items": {
                "type": "object",
                "properties": {
                    "CATEGORY_ID": {
                        "examples": [
                            "AC3DF073B60107890A4C704A10577FE700000000"
                        ],
                        "title": "GUID категории",
                        "type": "string"
                    },
                    "DISPLAY_NAME": {
                        "title": "Название",
                        "type": "string"
                    },
                    "DIR_PATH (возможно изменение в названии)": {
                        "title": "Полное имя директории",
                        "type": "string"
                    },
                    "FP_BIN_VALUE_THRESHOLD": {
                        "title": "Порог цитируемости бинарных данных",
                        "type": "integer"
                    },
                    "FP_TEXT_VALUE_THRESHOLD": {
                        "title": "Порог цитируемости текстовых данных",
                        "type": "integer"
                    },
                    "NOTE": {
                        "title": "Описание",
                        "type": [
                            "null",
                            "string"
                        ]
                    },
                    "OWNER": {
                        "title": "ID токена",
                        "type": "integer"
                    },
                    "CREATE_DATE": {
                        "examples": [
                            "2017-07-04 10:45:45.000000"
                        ],
                        "title": "Дата создания",
                        "type": "string"
                    },
                    "CHANGE_DATE": {
                        "examples": [
                            "2017-07-04 10:45:45.000000"
                        ],
                        "title": "Дата изменения",
                        "type": "string"
                    }
                }
            }
        }
    }
}
```

Code Block 26 Ответ:

1.4.5.8 Удаление эталонного документа REST API

Метод: DELETE
Ресурс: *etalonDocument/{fingerprint_id}*

```
curl -X 'DELETE' 'http://localhost:63342/xapi/etalonDocument/0123456789ABCDEF'
```

Code Block 27 Пример запроса