Загрузка событий в InfoWatch Traffic Monitor (pushAPI SDK)

Traffic Monitor 7.9

Exported on 05/14/2024

Table of Contents

Программный интерфейс pushAPI SDK предназначен для интеграции сторонних компонентов с Traffic Monitor для осуществления перехвата информации с различного рода каналов и их последующего анализа.
Сторонние компоненты самостоятельно формируют события перехвата данных и передают их в Traffic Monitor. После этого Traffic Monitor классифицирует события, анализирует их и выносит свой вердикт на основе настроенных политик защиты данных. Переданные события и вердикты можно просматривать в консоли Traffic Monitor.

1 Принципы использования pushAPI SDK

Для организации передачи событий в Traffic Monitor необходимо учитывать следующие принципы:

- События перехвата передаются в Traffic Monitor в виде объектов «событие»;
- Каждое «событие» имеет набор обязательных атрибутов, которые задаются пользователем программного интерфейса;
- Каждое «событие» может иметь набор необязательных атрибутов, которые задаются пользователем программного интерфейса;
- Каждое «событие» имеет «отправителя» и «получателя»;
- Каждый «отправитель» и «получатель» может иметь связанный с ним список «контактов», которые позволяют уникальным образом идентифицировать отправителя и получателя;
- Каждый «получатель» и «отправитель» может иметь связанный с ним набор атрибутов;
- Каждое «событие» имеет связанные с ним «данные», т.е. последовательность байт и/или символов текста, которые представляют собой содержимое перехваченного события;
- Каждый элемент «данных» может иметь связанный с ним набор атрибутов.

Все возможные «события», которые может принимать и анализировать Traffic Monitor, разбиты на группы – классы событий, которые объединены общими характеристиками. Класс события определяет бизнес-логику обработки события, а также правила его отображения в консоли Traffic Monitor. Собственные типы данных, т.е. классы событий, в Traffic Monitor создать нельзя. Пользователь SDK должен выбрать необходимые ему из списка поддерживаемых:

- **Почта** – любые события обмена электронными письмами, включая вложения. Характеризуется списком персон-отправителей и персон-получателей, датой отправки, темой, телом и вложениями к письму. Включает в себя электронную почту (IMAP, POP3, SMTP, MAPI, NRPC) и Веб-почту (HTTP, HTTPS). Название сервиса в Traffic Monitor: **Почта**
- **Беседа** – любые события беседы, представляющие собой обмен текстовыми сообщениями или файлами в рамках одного приложения. Приложение может являться как тонким клиентом на рабочих станциях и мобильных устройствах, так и web-приложением. Характеризуется списком реплик. Для каждой реплики должна быть указана персона-отправитель, время отправления и текст сообщения. Следующие сервисы уже предустановлены в этом классе: MS Lync, ICQ, Mail.Ru Агент, XMPP, Skype (передача SMS и обычных текстовых сообщений). Название сервиса в Traffic Monitor: **Мессенджеры**;
- **Печать, копирование и сканирование** – события преобразования данных на специализированных устройствах ввода и вывода изображений/документов (например: Принтер, Сканер). Характеризуется персоной-отправителем и устройством-получателем, количеством копий, а также файлом с оригиналом документа/образом документа/заданием на печать. В рамках этого класса предустановлен сервис Принтер. Название сервиса в Traffic Monitor: **Принтер и МФУ**;
- **Интернет-активность** – события формирования запросов из Web-браузеров и аналогичных программ, осуществляемое по протоколам HTTP, HTTPS. В событие данного типа попадают все данные, для которых не нашлось другого типа. Примером может служить отправка веб-сообщения. Характеризуется персоной-отправителем и ресурсом-получателем в случае POST-запросов или персоной-получателем и ресурсом-отправителем в случае GET-запросов, а также текстом "сырого" запроса. В рамках этого класса предустановлен сервис Интернет-приложение. Название сервиса в Traffic Monitor: **Интернет-активность**;
- **Фотосъемка** – события фотосъемки с помощью камеры устройства. Характеризуется персоной-отправителем и устройством-получателем, а также перехваченным изображением. Название сервиса в Traffic Monitor: **Запись мультимедиа**
- **Обмен файлами** – любые события, связанные с передачей файлов между произвольными отправителями и получателями. Характеризуется персоной-отправителем и персоной-/устройством-/ресурсом-получателем, а также перехваченным файлом. Включает в себя копирование на USB-устройство, в облачное хранилище, обмен файлами по FTP-протоколу.

Следующие сервисы уже предустановлены в этом классе: FTP, Съемное устройство, MS Lync, ICQ, Mail.Ru Агент, XMPP, Skype, Облачные хранилища, Терминальная сессия, Сетевые ресурсы. Название сервиса в Traffic Monitor: **Мессенджеры** (если отправители и получатели события являются персонами) или **Обмен файлами** (если хотя бы один отправитель или получатель события не является персоной);

- **Звонок** – любое событие голосового или видеоразговора между людьми, который может происходить по мессенджеру или телефону. Характеризуется персоной-отправителем (инициатор звонка) и персонами-получателями, а также файлом и длиной записи разговора. Примером могут служить отправка/получение SMS и MMS или перехват разговоров в сотовых сетях. Следующие сервисы уже предустановлены в этом классе: MS Lync,Mail.Ru Агент, XMPP, Skype. Название сервиса в Traffic Monitor: **Мессенджеры**;
- **Запись мультимедиа** – события создания аудио- и видеозаписи с помощью устройства. Характеризуются персоной-отправителем и устройством-получателем, а также перехваченным изображением. Примером могут служить записи с микрофонов/диктофонов или записи на видеокамеру, а также любые события, связанные с фотографированием потенциально конфиденциальных документов на камеру устройства. Название сервиса в Traffic Monitor: **Запись мультимедиа**.

Пользователь SDK должен обязательно определить класс событий и «Протокол» для своих перехваченных объектов. «Протокол» определяет, что именно было перехвачено с точки зрения пользовательского сервиса, который использовал «отправитель» для передачи информации. Добавление собственных «Протоколов» осуществляется через файл регистрации стороннего компонента (подробнее см. «Регистрация стороннего компонента pushAPI SDK(see page 24)»). На основании «Протокола» объекта и идентификатора компании-производителя перехватчика реализуются политики лицензирования внутри Traffic Monitor. Также «Протокол» перехваченного события может задаваться как одно из условий в формировании «политики» Traffic Monitor и аналитических запросах.

На основе классов событий пользователь SDK может определить и зарегистрировать свой сервис или выбрать из числа встроенных в Traffic Monitor.

Пример:

В Traffic Monitor есть встроенный сервис «file_copy_removable» для перехвата событий класса «Обмен файлами» при копировании файлов на съемные носители.

Пример:

Если пользователь SDK реализовал перехватчик для мессенджера Viber, он может определить и зарегистрировать сервис "Viber" для перехвата событий, например, таких классов, как «Беседа», «Голосовая беседа», «Обмен файлами». На каждый сервис одного производителя будет выдаваться отдельная лицензия.

У каждого «События» должны быть определены «отправитель(и)» и «получатель(и)». Поддерживаются следующие типы «получателей» и «отправителей»:

- «Персона» – представляет собой «пользователя», который определяется как создатель, отправитель или получатель перехваченного объекта. Для персоны можно определить набор «контактов» для ее идентификации. Тип контакта можно выбрать из списка, который поддерживается SDK. В качестве примера типа контакта можно привести почту, телефон или учетную запись Windows;
- «Компьютеры» – представляет собой компьютер, мобильное устройство или терминальный сервер, на котором был «создан» перехваченный объект. Для данного типа «отправителя» можно определить список «контактов»;
- «Ресурс» – представляет собой адрес в сети Интернет, на который были переданы данные;

- «Устройство» – представляет собой устройство, которое получает данные при копировании файлов на съемные носители.

Список контактов, связанных с «Персоной», позволяет идентифицировать ее по любому контакту. Таким образом, любые перехваченные действия «Персоны», в которых определен хотя бы один ее контакт, позволяют идентифицировать конкретного пользователя в Traffic Monitor.
Интерфейс SDK позволяет связывать различные типы контактов между собой логическими связями, например, отправителя «Персону» с отправителем «Компьютеры». Это позволяет точно определить контекст перехватываемой операции, т.е. кто и где инициировал действие по передаче данных.
Интерфейс SDK позволяет передавать данные в сеть по частям. Это позволяет внешним разработчикам самим определять стратегию ограничения нагрузки на сеть.
Общий алгоритм передачи события через интерфейс SDK выглядит так:

1. Создать объект «Событие» и установить его атрибуты;
2. Создать отправителей/получателей события, а также установить их контакты и атрибуты;
3. Создать данные события и установить его атрибуты;
4. Создать соединение с Traffic Monitor;
5. Проверить доступ к Traffic Monitor и лицензию плагина внешнего разработчика;
6. Передать «метаданные» события;
7. Передать по частям содержимое данных события;
8. Закрыть соединение;
9. Удалить объект «Событие».

Более подробно процесс передачи событий можно посмотреть в Примерах SDK.

2 Общее описание программного интерфейса pushAPI SDK

Программный интерфейс реализован в виде Thrift-сервиса. Подробнее о Thrift-сервисах можно прочитать на официальном сайте Apache Thrift[1] (доступен на английском языке).

Для работы pushAPI SDK установите последнюю доступную версию Thrift.

Для операционных систем, подобных Red Hat Linux, можно воспользоваться готовым пакетом из стороннего репозитория EPEL.

2.1 Установка Thrift на примере операционной системы Red Hat Enterprise Linux 7.x

1. Установите репозиторий EPEL, выполнив в терминале команду:
   `yum install epel-release`
2. Установите Thrift, выполнив команду:
   `yum -y install thrift`

В состав примеров SDK входит Thrift-схема, которая описывает сервис, реализованный в Traffic Monitor. Далее приведено описание этой схемы:

2.2 Типы данных pushAPI SDK

Код	Описание
Структура Identity	
`typedef i32 ItemId`	Идентификатор элемента в событии
`enum IdentityItemType { kPerson, kWorkstation, kDevice, kResource, }`	Вид элемента идентификации

1 https://thrift.apache.org

Код	Описание
`struct Attribute { 1: required string name, 2: required string value }`	Отдельный атрибут любой сущности события (самого события, идентификации, потока данных): 1. контакт; 2. метаинформация о контакте
`struct ContactWithMeta { 1: required Attribute contact, 2: optional string meta }`	Контакты идентификации с метаинформацией: 1. контакт; 2. метаинформация о контакте
`struct Identity { 1: required ItemId identity_id, 2: required IdentityItemType identity_type, 3: required list<Attribute> identity_contacts, 4: optional list<Attribute> identity_attributes, 5: optional list<ContactWithMeta> identity_contacts_with_meta }`	Идентификация получателя или отправителя объекта: 1. уникальный номер идентификации внутри объекта, используется для ссылок на идентификацию; 2. вид элемента идентификации; 3. набор контактов идентификации; 4. необязательный набор атрибутов идентификации; 5. набор контактов идентификации с метаинформацией
`struct EventData { 1: required ItemId data_id, 2: optional list<Attribute> data_attributes }`	Описание потока данных события: 1. уникальный идентификатор потока данных внутри события, используется для ссылок на поток; 2. необязательный набор атрибутов потока данных

Код	Описание

`struct ChaTraffic M{ toressage`
`1: required ItemId sender_id,`
`2: required string sent_time,`
`3: required string utf8_text,`
`4: optional ItemId mes_data_id`
`}`
	Описание одного сообщения класса "Беседа":
	1. ссылка на идентификацию, которая является отправителем сообщения;
	2. локальное время передачи сообщения в формате ГГГГ-ММ-ДДТЧЧ:ММ:СС+ЧЧ:ММ (ISO 8601 с форматированием yyyy-MM-ddTHH:mm:sszzz);
	3. текст сообщения в кодировке UTF-8;
	4. идентификатор данных объекта, в которых будет передано содержимое сообщения, если `utf8_text` пустой

`typedef string LinkNameType`
`struct ItemLink {`
`1: required ItemId id,`
`2: required ItemId link_to_id,`
`3: required LinkNameType link_name`
`}`
	Описание связи между различными сущностями события:
	1. идентификатор сущности события, от которой идёт связь;
	2. идентификатор сущности события, к которой идёт связь;
	3. имя (тип) связи

`const LinkNameType link_name_ws =` `"workstation_link"`	Список имен поддерживаемых связей между сущностями: имя для задания связи идентификаций типа Персона и Рабочая станция

Код	Описание
`struct Event { 1: required EventClass evt_class, 2: required Traffic MonitorServiceType evt_service, 3: optional list<Attribute> evt_attributes, 4: required list<Identity> evt_senders, 5: required list<Identity> evt_receivers, 6: required list<EventData> evt_data, 7: optional list<ChaTraffic Monitoressage> evt_messages, 8: optional list<ItemLink> evt_links, 9: optional Identity evt_source, 10: optional Identity evt_destination }`	Описание события, которое нужно передать в Traffic Monitor: 1. класс события; 2. сервис события; 3. атрибуты события; 4. отправители события; 5. получатели события; 6. данные события; 7. сообщения события для класса "Беседа"; 8. связи между сущностями в событии; 9. источник события; 10. приемник события
`struct Credentials { 1: required string company_name, 2: required string token }`	Структура, описывающая параметры доступа к Traffic Monitor: 1. название клиента (или компании). Этот параметр связывает плагин внешнего разработчика с лицензией; 2. токен авторизации. Токен авторизации. Используется для авторизации источника данных. Значение токена известно только плагину и Traffic Monitor. Так реализована авторизация источника данных
`typedef i64 EventId`	Идентификатор загружаемого события
`typedef i64 StreamId`	Идентификатор загружаемого потока данных события

Код	Описание

`typedef i32 InterfaceVersion`
`const InterfaceVersion`
`pushapi_version = 1`
	Версия PushAPI интерфейса

`enum EventClass {`
`kChat,`
`kEmail,`
`kMfp,`
`kWeb,`
`kFileExchange,`
`kPhoto,`
`kMultimedia,`
`kVoiceTalk`
`}`
	Поддерживаемые классы событий в PushAPI:
	• Беседа
	• Почта
	• Печать, копирование и сканирование
	• Интернет-активность
	• Обмен файлами
	• Фотосъемка
	• Запись мультимедиа
	• Звонок

Примечание:

Если необходимо реализовать сценарий для блокировки передачи данных по результатам анализа, то при отправке посредством PushAPI событий в Traffic Monitor используйте один из "классов" событий: `kChat, kEmail , kWeb.` Только для этих классов, соответствующих типам событий **Мессенджер**, **Почта**, **Интернет-активность** в Traffic Monitor реализована возможность назначения событию вердикта **Заблокировано** в политике защиты данных.

2.3 Исключения pushAPI SDK

Код	Описание

`exception EventNotFound {`
`1: string message,`
`}`
	EventId, указанный в списке параметров, не найден в рамках текущей сессии передачи события

`exception DataNotFound {`
`1: string message,`
`}`
	DataId для указанного EventId некорректный, т.е. указанный DataId не найден в рамках текущей сессии передачи события

Код	Описание
`exception StreamNotFound { 1: string message, }`	StreamId для указанного EventId некорректный, т.е. указанный StreamId не найден в рамках текущей сессии передачи события
`exception InvalidEventFormat { 1: string message, }`	Некорректный формат события, т.е. не прошла проверка на содержимое атрибутов события. Каждый класс события имеет набор обязательных атрибутов
`exception InvalidCredentials { 1: string message }`	Некорректные параметры доступа к интерфейсу. Нужно указать правильный токен
`exception LicenseError { 1: string message }`	На сервере Traffic Monitor нет правильной лицензии для передачи событий
Примечание: message – уточняет причину возникновения исключения	

2.4 Сервисы Traffic Monitor pushAPI SDK

Встроенные в Traffic Monitor сервисы, которые можно использовать для событий:

```
const TrafficMonitorServiceType service_email = "email" //!< Электронная почта
const TrafficMonitorServiceType service_email_web = "email_web" //!< Веб-почта
const TrafficMonitorServiceType service_im_icq = "im_icq" //!< ICQ
const TrafficMonitorServiceType service_im_skype = "im_skype" //!< Skype
const TrafficMonitorServiceType service_im_xmpp = "im_xmpp" //!< XMPP
const TrafficMonitorServiceType service_im_mmp = "im_mail_ru" //!< MMP
const TrafficMonitorServiceType service_im_lync = "im_lync" //!< MS Lync
const TrafficMonitorServiceType service_web = "web" //!< Веб-сообщения
const TrafficMonitorServiceType service_file_removable_storage = "file_copy_removable" //!< Обмен файлами с внешним устройством (съёмным устройством хранения)
const TrafficMonitorServiceType service_file_ftp = "file_ftp" //!< Обмен файлами по FTP
const TrafficMonitorServiceType service_print = "print" //!< Печать
const TrafficMonitorServiceType service_cloud_storage = "cloud_storage" //!< Обмен файлами с облачным хранилищем
```

```
const TrafficMonitorServiceType service_terminal_session = "terminal_session" //!< Обмен файлами через терминальную сессию
const TrafficMonitorServiceType service_network_resource = "network_resource" //!< Обмен файлами с сетевыми ресурсами
```

Наименование сервиса	Классы	Описание
email	«Почта»	Сервис перехвата корпоративной почты, который не зависит от используемого протокола или типа почтового сервера. При расширении класса «Почта» рекомендуется добавлять к названию сервиса префикс "email_", например "email_google"
email_web	«Почта»	Сервис перехвата веб-почты
im_icq	«Беседа», «Голосовая беседа», «Обмен файлами»	Сервис перехвата программы обмена сообщениями "ICQ" (протокол "OSCAR")
im_skype	«Беседа», «Голосовая беседа», «Обмен файлами»	Сервис перехвата программы обмена сообщениями "Skype"
im_xmpp	«Беседа», «Голосовая беседа», «Обмен файлами»	Сервис перехвата любой программы обмена сообщениями, которая работает по протоколу XMPP (Jaber)
im_mail_ru	«Беседа», «Голосовая беседа», «Обмен файлами»	Сервис перехвата любой программы обмена сообщениями, которая работает по протоколу MMP(Mail.ru Agent)
im_lync	«Беседа», «Голосовая беседа», «Обмен файлами»	Сервис перехвата программы обмена сообщениями "MS Lync"
web_common	«Интернет-активность»	Сервис перехвата неразобранных POST-запросов
file_copy_removable	«Обмен файлами»	Сервис, который создает копии файлов, копируемых на съёмные устройства хранения данных

Наименование сервиса	Классы	Описание
ftp	«Обмен файлами»	Сервис, который создает копии файлов, передаваемых по FTP-протоколу
print	«Принтеры и МФУ»	Сервис, который создает «образы» распечатываемых на принтере документов
cloud_storage	«Обмен файлами»	Сервис, который создает копии файлов, копируемых на облачные хранилища
terminal_session	«Обмен файлами»	Сервис, который создает копии файлов, передаваемых через терминальную сессию
network_resource	«Обмен файлами»	Сервис, который создает копии файлов, копируемых на сетевые ресурсы
Примечание: При добавлении нового сервиса, который представляет собой программу обмена сообщениями, рекомендуется добавить к его названию префикс "im_". Например "im_viber"		

Примечание:

Сервисы с другими типами событий могут быть зарегистрированы с помощью плагинов

2.5 Типы контактов Traffic Monitor pushAPI SDK

Типы контактов, поддерживаемые бизнес-логикой Traffic Monitor:

Код	Описание
`const TrafficMonitorContactType contact_type_email = "email"`	Почтовый адрес. Используется для идентификаций с типом «Персона»
`const TrafficMonitorContactType contact_type_auth = "auth"`	Логин в ОС в формате login@domain. Используется для идентификаций с типом «Персона»
`const TrafficMonitorContactType contact_type_dnshostname = "dnshostname"`	DNS имя. Используется для отправителей с типом «Рабочая станция»

Код	Описание
`const TrafficMonitorContactType contact_type_icq = "icq"`	Icq-номер. Используется для идентификаций с типом «Персона»
`const TrafficMonitorContactType contact_type_skype = "skype"`	Skype-логин. Используется для идентификаций с типом «Персона»
`const TrafficMonitorContactType contact_type_phone = "phone"`	Номер телефона. Используется для идентификаций с типом «Персона»
`const TrafficMonitorContactType contact_type_ip = "ip"`	IPv4-адрес. Используется для отправителей с типом «Рабочая станция»
`const TrafficMonitorContactType contact_type_sid = "sid"`	Security Identifier в операционной системе Windows, представленный в текстовом формате. Используется для идентификаций с типом «Персона»
`const TrafficMonitorContactType contact_type_webaccount = "webaccount"`	Название учетной записи на сетевом ресурсе
`const TrafficMonitorContactType contact_type_lotus = "lotus"`	Почтовый адрес в Lotus. Используется для идентификаций с типом «Персона»
`const TrafficMonitorContactType contact_type_domain = "domain"`	Домен

2.6 Поддерживаемые атрибуты Traffic Monitor pushAPI SDK

Список имён атрибутов, которые поддерживаются в Traffic Monitor:

Код	Описание атрибута
`typedef string EventAttributeType`	**Атрибуты события:**

Код	Описание атрибута
`const EventAttributeType event_attr_capture_date = "capture_ts"`	Локальное время перехвата события в формате ГГГГ-ММ-ДДТЧЧ:ММ:СС+ЧЧ:ММ (ISO 8601 с форматированием yyyy-MM-ddTHH:mm:sszzz) (*)
`const EventAttributeType event_attr_capture_server_ip = "capture_server_ip"`	IP-адрес сервера перехвата (позволяет различать различные экземпляры одного плагина) (*)
`const EventAttributeType event_attr_capture_server_fqdn = "capture_server_fqdn"`	DNS-имя сервера перехвата (позволяет различать различные экземпляры одного плагина) (*)
`const EventAttributeType event_attr_inteceptor_id = "interceptor_id"`	Идентификатор объекта в перехватчике
`const EventAttributeType event_attr_inteceptor_protocol = "protocol_mnemo"`	Протокол, по которому передавался объект
`const EventAttributeType event_attr_print_copies = "print_copies"`	Количество копий, целое число в строковом представлении
`const EventAttributeType event_attr_destination_type = "destination_type"`	Тип приёмника
`typedef string DataAttributesType`	**Общие атрибуты потока данных:**
`const DataAttributesType data_attr_mimetype = "mimetype"`	MIME-тип содержимого (заполняется только для узлов, которые уже разобраны клиентом)
`const DataAttributesType data_attr_size = "data_size"`	Размер данных, если он известен заранее

Код	Описание атрибута
`const DataAttributesType data_attr_sent_date = "sent_date"`	Дата отправки объекта в формате ГГГГ-ММ-ДДТЧЧ:ММ:СС+ЧЧ:ММ. Позволяет независимо устанавливать время формирования события и время реальной передачи данных
`const DataAttributesType data_attr_comment = "data_comment"`	Комментарий
	Атрибуты потока данных для класса "Почта":
`const DataAttributesType data_attr_email_subject = "subject"`	Тема письма
`const DataAttributesType data_attr_email_is_encrypted = "encrypted"`	Признак шифрования (строки "true" или "false")
`const DataAttributesType data_attr_email_is_signed = "is_signed"`	Признак наличия цифровой подписи (строки "true" или "false")
	Атрибуты потока данных для класса "Обмен файлами":
`const DataAttributesType data_attr_file_filename = "filename"`	Короткое имя файла (*)
`const DataAttributesType data_attr_file_source_file_path = "source_path"`	Полный путь к файлу источнику события
`const DataAttributesType data_attr_file_destination_file_path = "destination_path"`	Полный путь к переданному или скопированному файлу или полный адрес веб-ресурса
	Атрибуты потока данных для класса "Сотовая связь":

Код	Описание атрибута
`const DataAttributesType data_attr_voice_duration_sec = "call_duration"`	Длительность в секундах (целое число в строковом представлении)
	Атрибуты потока данных для класса "Принтер и МФУ":
`const DataAttributesType data_attr_print_job_name = "print_job_name"`	Название задания на печать
`const DataAttributesType data_attr_print_copies = "event_attr_print_copies"`	Количество копий (целое число в строковом представлении)
`typedef string IdentityAttrType`	**Общие атрибуты идентификаций:**
`const IdentityAttrType identity_attr_comment = "identity_comment"`	Комментарий
`typedef string TWorkstation`	**Атрибуты идентификаций типа "Компьютеры":**
`const IdentityAttrType identity_attr_ws_type = "workstation_type"`	Тип рабочей станции:
`const TWorkstation ws_type_computer = "ws_type_computer"`	Компьютер
`const TWorkstation ws_type_mobile = "ws_type_mobile"`	Мобильное устройство
`const TWorkstation ws_type_terminal = "ws_type_terminal"`	Терминальный сервер

Код	Описание атрибута
	Атрибуты идентификаций типа "Устройство":
`const IdentityAttrType identity_attr_device_name = "device_name"`	Имя устройства (*)
`const IdentityAttrType identity_attr_device_type = "device_type"`	Тип устройства
	Атрибуты идентификации типа "Устройство", "Ресурс", "Рабочая станция" для источника и приемника:
`const IdentityAttrType identity_attr_endpoint_name = "name"`	Имя устройства или ресурса (*)
`const IdentityAttrType identity_attr_endpoint_path = "path"`	Путь на заданном ресурсе или устройстве (*)
`const IdentityAttrType identity_attr_endpoint_type = "type"`	Тип ресурса или устройства. Заполняется мнемониками из справочника
`const IdentityAttrType identity_attr_endpoint_device_id`	ID съемного устройства
`const string identity_attr_print_location = "print_location"`	Расположение устройства принтера
`const string identity_attr_print_port_name = "print_port_name"`	Имя порта устройства принтера

Код	Описание атрибута
`const string identity_attr_print_server = "print_server"`	Имя сервера устройства принтера
	Атрибуты идентификаций типа "Ресурс":
`const IdentityAttrType identity_attr_resource_url = "res_destination_url"`	URL ресурса (*)
`const IdentityAttrType identity_attr_resource_address = "res_destination_host"`	DNS или IP адрес ресурса (*)
Примечание: (*) - обязательный атрибут	

Примечание:

Если внешняя система присылает в Traffic Monitor событие, в котором атрибут `destination_path` заполнен несколькими значениями, то в Traffic Monitor событие будет сохранено с первым значением этого атрибута, а все остальные значения будут отброшены.

2.7 Сервис EventProcessor pushAPI SDK

В состав pushAPI SDK входит сервис EventProcessor. Он предоставляет API, который позволяет создавать и пересылать события и объекты перехвата. Ниже приведено его описание.

Код	Описание
`InterfaceVersion GetVersion()`	Функция получения текущей версии интерфейса
`void VerifyCredentials(1: Credentials cred) throws(1: InvalidCredentials ex);`	Функция проверки параметров доступа к интерфейсу pushAPI SDK. Позволяет проверить возможность передачи данных до их реальной передачи:

Код	Описание
`EventId BeginEvent (1: Event event, 2: Credentials cred) throws(1: InvalidEventFormat ex1, 2: InvalidCredentials ex2, 3: LicenseError ex3, 4: EventNotFound ex4, 5: DataNotFound ex5);`	Функция инициирует процесс передачи события перехвата на Traffic Monitor, используя pushAPI SDK, где: • `event` – заполненная структура содержащая все «метаданные», которые описывают событие перехвата. Атрибуты: отправитель, получатель, список потоков данных, которые связаны с событием; • `cred` – структура, которая позволяет авторизовать процесс передачи события. Функция возвращает идентификатор события с сессии передачи данных (в одной и той же TCP-сессии можно передавать несколько событий).
`StreamId BeginStream (1: EventId event_id, 2: ItemId event_data_id) throws(1: EventNotFound ex1, 2: DataNotFound ex2);`	Функция инициирует передачу потока данных, который связан с событием перехвата данных, где: • `event_id` – идентификатор события, с которым связан данный поток данных. Возвращается функцией `BeginEvent`; • `event_data_id` – идентификатор передаваемых данных. Этот идентификатор передается в составе «метаданных» события, в поле `evt_data`; Функция возвращает идентификатор загружаемого потока.
`void SendStreamData(1: EventId object_id, 2: StreamId stream_id, 3: binary chunk) throws(1: EventNotFound ex1, 2: StreamNotFound ex2, 3: InvalidEventFormat ex3);`	Функция передает буфер данных в составе потока данных, который связан с событием перехвата данных, где: • `object_id` – идентификатор события, с которым связан данный поток данных. Возвращается функцией `BeginEvent`; • `stream_id` – идентификатор потока данных, в составе которого передается буфер. Возвращается функцией `BeginStream`; • `chunk` – буфер данных. Пользователь SDK сам должен выбирать разумный размер буфера данных, чтобы обеспечить достаточную скорость загрузки событий, но не перегружать сетевую инфраструктуру

Код	Описание
`void EndStream(1: EventId event_id, 2: StreamId stream_id) throws(1: EventNotFound ex1, 2: StreamNotFound ex2, 3: InvalidEventFormat ex3);`	Функция фиксирует конец передачи потока данных, связанного с событием перехвата, где: • `event_id` – идентификатор события, с которым связан данный поток данных. Возвращается функцией `BeginEvent`; • `stream_id` – идентификатор потока данных, в составе которого передается буфер. Возвращается функцией `BeginStream`
`string GetEventDatabaseId(1: EventId object_id) throws(1: EventNotFound ex1);`	Функция получения уникального идентификатора события перехвата в системе Traffic Monitor, где: • `object_id` – идентификатор события, полученный вызовом `BeginEvent`. Это значение можно использовать для ссылки на загруженное событие. Функцию можно вызывать после удачного выполнения функции `BeginEvent`
`void EndEvent(1: EventId event_id, 2: bool abort) throws(1: EventNotFound ex1, 2: InvalidEventFormat ex2, 3: LicenseError ex3);`	Функция фиксирует конец передачи события перехвата, где: • `event_id` – идентификатор события, передача которого завершается; • `abort` – флаг прекращения обработки события в Traffic Monitor. Если выставлен флаг, то событие не попадает в БД

3 Регистрация стороннего компонента pushAPI SDK

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

1. Сторонний разработчик, регистрирует в компании InfoWatch строку идентификатор компании-производителя компонента. Это позволит связывать компанию-производителя с лицензией, которая будет устанавливаться в систему Traffic Monitor;
2. Он же получает лицензию разработчика, связанную с идентификатором компании. Это позволит загружать в Traffic Monitor события перехвата этого производителя. Лицензия разработчика позволяет загружать произвольные типы (Сервисы) событий;
3. Затем разработчик создает файл регистрации стороннего компонента в формате, описанном в статье "Формат файла регистрации стороннего компонента pushAPI SDK(see page 24)". К этому моменту должны быть определены названия сервисов, которые будут перехватываться сторонним компонентом. В файл регистрации добавляется лицензия, полученная на шаге 2;

   Важно!

   Крайне не рекомендуется в дальнейшем изменять названия сервисов. Каждому сервису соответствует тип события в Traffic Monitor. Изменение названия сервиса создаст новый тип события и может привести к дублированию типов события в Консоли управления Traffic Monitor. Подробнее об устранении дублирования см. "Обновление файла регистрации стороннего компонента(see page 51)".
   В файле регистрации стороннего компонента название сервиса указывается в поле "ADDS_SERVICES"->"SERVICE_TYPE" ->"SERVICE_MNEMO".
4. Заказчик, который внедряет интеграцию, загружает плагин с файлом регистрации стороннего компонента и лицензию в систему Traffic Monitor через интерфейс консоли Traffic Monitor. Раздел «Управление» - «Плагины». (см. "InfoWatch Traffic Monitor. Руководство пользователя");
5. Заказчик получает токен доступа в свойствах загруженного стороннего компонента.

Система готова принимать события от заданной компании производителя с указанным сервисом перехвата.

Информация об изменении файла регистрации уже зарегистрированного стороннего компонента приведена в статье "Обновление файла регистрации стороннего компонента(see page 51)".

3.1 Формат файла регистрации стороннего компонента pushAPI SDK

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, соответствующий стандарту RFC 4122. Идентификатор создается его разработчиком. Содержит до 40 символов (буквы используются только в верхнем регистре, не используется символ "-"). Идентификатор используется для обновления плагина. **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 ",`

Содержимое	Тип данных	Является обязательным	Описание
`"VERSION": "",`	Строка	Да	Версия плагина. **Пример:** `"VERSION": "1.0.0",`
`"IS_SYSTEM": ,`	Логический тип	Нет	Признак "Предустановленный". Входит плагин ли в состав Системы. **Пример:** `"IS_SYSTEM": false,`
`"VENDOR": "",`	Строка	Да	Идентификатор компании-разработчика, соответствующий названию компании в лицензии. **Пример:** `"VENDOR": "infowatch",`

Содержимое	Тип данных	Является обязательным	Описание

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

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|not,child[{name:value}]}"`.
			
			Пример operator:or
			
			`"PATTERN_SEARCH_LICENSE": { "operator": "or", "conditions": [{ "origin": "dm" }, { "origin": "dmmobile" } ] },`
			
			Пример operator:and
			
			`"PATTERN_SEARCH_LICENSE": { "operator": "or", "conditions": [ { "operator": "and",`

Содержимое	Тип данных	Является обязательным	Описание

```
            "conditions": [
                {"key1": "value1"},
                {"key2": "value2"},
                {"key3": "value3"}
            ]
        },
        {
            "operator": "and",
            "conditions": [
                {"key1": "value1"},
                {"key2": "value2"},
                {"key3": "value3"}
            ]
        }
    ]
}
```

Пример operator:not

```
"PATTERN_SEARCH_LICENSE": {
    "operator": "and",
    "conditions": [
        {"key1": "value1"},
        {"key2": "value2"},
        {"key3": "NONE"},
        {
            "operator": "not",
            "conditions": [
```

Содержимое	Тип данных	Является обязательным	Описание

			<pre>{"key4": null}
			]
			}
			]
			}</pre>

Содержимое	Тип данных	Является обязательным	Описание

"ADDS_EVENTS": {
"OBJECT_TYPE": [
{
"MNEMO": "",
"SERVICE_MNEMO": "",
"ICON": "",
"LOCALE": {
"": ""
}
}
],
"PROTOCOL": [
{
"MNEMO": "",
"LOCALE": {
"": ""
}
}
]
},
	ОбъектМассивСтрокаСтрокаОбъект	Да	Регистрируемые типы событий и протоколы.
			Массив типов регистрируемых событий с указанием типов сервисов, к которым они относятся.
			
			Тип события.
			Сервис.
			Файлы с иконками регистрируемых контактов.
			Локализации наименований типов контактов хотя бы на одном языке.
			
			Пример
			
			"ADDS_EVENTS": {
			"OBJECT_TYPE": [
			{
			"MNEMO": "cloud_storage",
			"SERVICE_MNEMO": "file",
			"LOCALE": {
			"rus": "Облачное хранилище ",
			"eng": "Cloud storage"
			}
			},
			{
			"MNEMO": "im_telegram",
			"SERVICE_MNEMO": "im",
			"ICON": "icon/event_icon_telegram.png",
			"LOCALE": {

Содержимое	Тип данных	Является обязательным	Описание

	Массив Строка Объект		<pre>"rus": "Telegram",
			"eng": "Telegram"
			}
			}
			],
			"PROTOCOL": [
			{
			"MNEMO": "telegram",
			"LOCALE": {
			"rus": "Telegram",
			"eng": "Telegram"
			}
			}
			]
			},</pre>

Содержимое	Тип данных	Является обязательным	Описание
`"USES_EVENTS": { "OBJECT_TYPE": [ { "MNEMO": "", "PROTOCOL": [ { "MNEMO": "" } ], "ORIGIN": [ "" ] } ] },`	Объект Массив Строка Массив Строка	Да	Существующие типы событий. Массив типов регистрируемых событий с указанием типов сервисов, к которым они относятся. Тип события. Массив протоколов, относящихся к данному типу события. Протоколы. Массив кодов типов перехватчиков для Плагинов InfoWatch. Коды. **Пример** `"USES_EVENTS": { "OBJECT_TYPE": [ { "MNEMO": "email", "PROTOCOL": [ { "MNEMO": "pop3" }, { "MNEMO": "imap" }, {`

Содержимое	Тип данных	Является обязательным	Описание

	МассивСтрока		<pre>"MNEMO": "mapi"
			},
			{
			"MNEMO": "smtp"
			}
			],
			"ORIGIN": ["dm", "dmmobile"]
			},
			{
			"MNEMO": "email_web",
			"PROTOCOL": [
			{
			"MNEMO": "http"
			},
			{
			"MNEMO": "https"
			}
			],
			"ORIGIN": ["dm", "dmmobile"]
			},
			{
			"MNEMO": "im_whatsapp",
			"PROTOCOL": [
			{
			"MNEMO": "whatsapp"
			}
			],
			"ORIGIN": ["dm", "dmmobile"]
			}
			]</pre>

Содержимое	Типданных	Являетсяобязательным	Описание
			},

Содержимое	Тип данных	Является обязательным	Описание

"ADDS_SERVICES": {
"SERVICE_TYPE": [
{
"SERVICE_MNEMO": "",
"DATA_CLASS": [""],
"ICON": "",
"LOCALE": {
"": "",
"": ""
}
}
]
},
	ОбъектМассивСтрокаМассивстрокСтрокаОбъе	Да	Регистрируемые типы событий через сервисы SDK.
			Регистрируемые типы перехватываемых сервисов.
			
			Имя перехватываемого сервиса.
			Список классов данных, которые перехватывает данный перехватчик в сервисе.
			Файл с иконкой регистрируемого сервиса.
			Локализации наименований типов регистрируемого сервиса хотя бы на одном языке.
			
			Пример
			
			"ADDS_SERVICES": {
			"SERVICE_TYPE": [{
			"SERVICE_MNEMO": "newservice",
			"DATA_CLASS": ["kChat",
			"kFileExchange"],
			"ICON": "icon/Traffic Monitor_icon.
			"LOCALE": {
			"rus": "сервис #5",
			"eng": "service#5"
			}
			},
			{
			"SERVICE_MNEMO": "newservice2",
			"DATA_CLASS": ["kPhoto"],
			"ICON": "icon/Traffic Monitor_icon.
			"LOCALE": {

Содержимое	Тип данных	Является обязательным	Описание

			<pre>"rus": "сервис #5+2",
			"eng": "service#5+2"
			}
			}
			]
			},</pre>

Содержимое	Тип данных	Является обязательным	Описание

`"OBJECT_HEADER": [`
`{`
`"NAME": "",`
 	
`"NOTE": {`
`"": "",`
`"": ""`
`},`
`"TYPE": "",`
`"FORMAT": "",`
 	
`"DATA_CLASS": [""],`
`"USE_IN_POLICY": "",`
`"USE_IN_NOTIFICATION": "",`
`"USE_IN_QUERY": "",`
`"USE_IN_LIST": "",`
`"USE_IN_SHOW": "",`
`"USE_IN_DETAIL": "",`
`"IS_MULTIPLE_VALUE": "",`
`"VALIDATION_RULE": [`
`{`
`"type": ""`
`}]`
`}`
`]`
	Массив
	
	Строка
	
	Объект Строка
		Нет	Пользовательские атрибуты событий.
			
			Уникальный код атрибута, который будет использован в заголовках события и для идентификации в xAPI/ PushAPI.
			Если идентификатор разработчика не "IW", код должен начинаться с префикса "`<Идентификатор компании – разработчика >_`".
			Пример: CISCO_CALL_DURATION
			Название атрибута, отображаемое в консоли Traffic Monitor.
			
			Тип значения.
			Формат записи. Возможны значения: число (целое, дробное), строка, дата и время в UTC + указание смещение часового пояса, длительность, гиперссылка, перечисление, логический тип.
			Сервисы, к событиям которых будет добавлен новый пользовательский атрибут.
			Использование атрибута в политиках. `"1"` - да, `"0"` - нет.
			Использование атрибута в почтовых уведомлениях. `"1"` - да, `"0"` - нет.
			
			Использование атрибута в запросах. `"1"` - да, `"0"` - нет.
			Использование атрибута в табличной форме просмотра списка событий. `"1"` - да, `"0"` - нет.
			Использование атрибута в краткой форме просмотра события. `"1"` - да, `"0"` - нет.
			Использование атрибута в детальной форме просмотра события. `"1"` - да, `"0"` - нет.

Содержимое	Тип данных	Является обязательным	Описание

	Перечисление Перечисление[] Масси		Множественное значение. `"1"` - да, `"0"` - нет.
			Правило проверки вводимых пользователем значений в запросах и политиках. Определяет возможные значения пользовательского атрибута. Можно задать следующие типы данных:
			• Логический
			• Строка
			• Целочисленный
			• Число с плавающей точкой
			• Длительность
			• Перечисление
			• Ссылка
			• Регулярное выражение
			Регулярное выражение задается в формате PCRE (Perl Compatible Regular Expressions).
			Примеры поля VALIDATION RULE
			`"VALIDATION_RULE": [ { "type": "boolean", "value": ["yes","no"] } ]`
			`"VALIDATION_RULE": [ {`

Содержимое	Тип данных	Является обязательным	Описание

	встрокПеречислениеПеречислениеП		<pre>"type": "string",
			"min": 1,
			"max": 100
			}
			]
			
			"VALIDATION_RULE": [
			{
			"type": "integer",
			"min": 0,
			"max": 100
			}
			]
			
			"VALIDATION_RULE": [
			{
			"type": "float",
			"min": 3,
			"max": 4
			}
			]
			
			"VALIDATION_RULE": [
			{
			"type": "duration",
			"value": [[1,31],[33,59]]
			}
			]
			
			"VALIDATION_RULE": [</pre>

Содержимое	Тип данных	Является обязательным	Описание

	перечислениеПеречислениеПеречисление		<pre>{
			"type": "enum",
			"enum": [
			"value1",
			"value2",
			"value3"
			]
			}
			]
			
			"VALIDATION_RULE": [
			{
			"type": "link",
			"link": "http://ya.ru"
			}
			]
			
			"VALIDATION_RULE": [
			{
			"type": "regexp",
			"pattern": "\d"
			}
			]</pre> **Пример описания пользовательского атрибута** <pre>"OBJECT_HEADER": [
			{</pre>

Содержимое	Тип данных	Является обязательным	Описание

	Перечисление Объект Перечисление		"NAME": "header_multi_string",
			"NOTE": {
			"rus": "Заголовок строк 1",
			"eng": "Header strings1"
			},
			"DATA_CLASS": ["kEmail"],
			"USE_IN_POLICY": "1",
			"USE_IN_QUERY": "1",
			"USE_IN_NOTIFICATION": "1",
			"USE_IN_LIST": "1",
			"USE_IN_SHOW": "1",
			"USE_IN_DETAIL": "1",
			"TYPE": "string",
			"FORMAT": "string",
			"IS_MULTIPLE_VALUE": "1"
			},
			{
			"NAME": "header_date",
			"NOTE": {
			"rus": "Заголовок дата 1",
			"eng": "Header date1"
			},
			"DATA_CLASS": ["kEmail"],
			"USE_IN_POLICY": "1",
			"USE_IN_QUERY": "1",
			"USE_IN_NOTIFICATION": "1",
			"USE_IN_LIST": "0",
			"USE_IN_SHOW": "1",
			"USE_IN_DETAIL": "1",
			"TYPE": "string",

Содержимое	Тип данных	Является обязательным	Описание

			`"FORMAT": "date",`
			`"IS_MULTIPLE_VALUE": "0"`
			`}`
			`]`

}			

Примечание:

Корневые поля `"ADDS_EVENTS"` и `"USES_EVENTS"` используются только для плагинов, разрабатываемых компанией InfoWatch. Для плагинов сторонних разработчиков используется поле `"ADDS_SERVICES".`

Полная JSON-схема входит в состав примеров SDK (см. раздел "Описание примеров PushAPI SDK(see page 56)"). Пример файла регистрации модуля входит в состав SDK, а также приведен в статье Пример файла регистрации стороннего компонента(see page 43).

Важно!

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

3.1.1 Пример файла регистрации стороннего компонента

Перед созданием файла регистрации стороннего компонента определите идентификатор компании-разработчика, соответствующий названию компании, зарегистрируйте его в компании InfoWatch, а

также определите название разрабатываемого сервиса.
В качестве исходных данных для примера возьмем:

- Идентификатор компании: `MY_COMPANY;`
- Название сервиса: `im_my_chat` – сервис перехвата сообщений в мессенджере.

Чтобы создать файл регистрации стороннего компонента:

1. Получите лицензию разработчика.
2. При необходимости выберите иконки для плагина и регистрируемых им событий.
3. Опишите manifest-файл плагина.

3.1.1.1 Получение лицензии разработчика

После определения названия сервиса получите в компании InfoWatch лицензию для плагина (подробнее см. статью "Запрос лицензии" в Руководстве пользователя). Для компании MY_COMPANY файл лицензии будет иметь следующий вид:

```
    "License": {
        "ActiveDays": 60,
        "Features": [
            {
                "common_name": "MY_COMPANY",
                "object_type": "im_my_chat",
                "protocol": "*"
            }
        ],
```

Где поле `protocol` может принимать следующие значения:

- `"none"` – отсутствие протокола;
- значение конкретного протокола;
- `"*"` – любой протокол.

3.1.1.2 Описание manifest-файла плагина

На основании исходных данных заполните все обязательные корневые поля `"PLUGIN_ID"`, `"DISPLAY_NAME"`, `"VERSION"`, `"VENDOR"`, `"LICENSE"` и `"ADDS_SERVICES"`.

Так как плагин `im_my_chat` предназначен для перехвата сообщений в мессенджере, значение поля `"DATA_CLASS"` в корневом поле `"ADDS_SERVICES"` будет равно `"kChat"`. Перехватываться будут события класса «**Беседа**».

Чтобы описать представление контакта пользователя в перехватываемых событиях добавьте поле `"CONTACT_TYPE"` в корневое поле `"ADDS_SERVICES"`.

Чтобы добавить пользовательские атрибуты события, используйте конструкцию `"OBJECT_HEADER"`. Название пользовательского атрибута события должно начинаться с названия компании-разработчика плагина. Количество атрибутов не ограничено. В данном примере добавим два атрибута:

- `"MY_COMPANY_FORWARDING_STATUS"` – статус пересылки сообщения, т.е. является ли сообщение пересланным.
  Может принимать значения:
  - `original`;
  - `forwarded`;
- `"MY_COMPANY_ORIGINAL_SENDER"` – оригинальный отправитель. Может принимать любое значение типа данных "Строка".

Manifest-файл плагина будет выглядеть следующим образом:

```
{
   "PLUGIN_ID": "189C38D390396EB6E0540100007F1CA200000001",
   "DISPLAY_NAME": "Test plugin",
   "DESCRIPTION": "Test plugin to demonstrate",
   "VERSION": "1.0.0",
   "IS_SYSTEM": false,
   "VENDOR": "MY_COMPANY",
   "ADDS_SERVICES": {
      "SERVICE_TYPE": [{
         "SERVICE_MNEMO": "im_my_chat",
              "DATA_CLASS": ["kChat"],
              "ICON": "icon/Traffic Monitor_icon.png",
              "LOCALE": {
                        "rus": "Новый сервис ",
                        "eng": "New service"
              },
              "CONTACT_TYPE" : [{
                                 "MNEMO" : "my_chat_Id",
                                 "SCOPE" : ["person"],
                                 "ICON": "icon/Traffic Monitor_icon.png",
                                 "LOCALE": {
                                           "rus": "Контакт тестового плагина  ",
                                           "eng": "Test plugin contact"
                                 }
                        }
              ]
```

```
      }]
   },
   "LICENSE": [
   { "PATH" : "licenses/Traffic Monitor_license_2023-07-24_.license" }
   ],
   "OBJECT_HEADER": [{
         "NAME": "MY_COMPANY_FORWARDING_STATUS",
         "NOTE": {
                  "rus": "Статус пересылки сообщения ",
                  "eng": "Forwarding Message Status"
         },
         "TYPE": "string",
         "FORMAT": "enum",
         "DATA_CLASS": ["kChat"],
         "USE_IN_POLICY": "1",
         "USE_IN_QUERY": "1",
         "USE_IN_NOTIFICATION": "1",
         "USE_IN_LIST": "1",
         "USE_IN_SHOW": "1",
         "USE_IN_DETAIL": "1",
         "IS_MULTIPLE_VALUE": "0",
         "VALIDATION_RULE": [{
                              "type": "enum",
                              "enum": [
                                       "original",
                                       "forwarded"
                              ]
   }]
   },
   {
         "NAME": "MY_COMPANY_ORIGINAL_SENDER",
         "NOTE": {
                  "rus": "Оригинальный отправитель  ",
                  "eng": "Original sender"
         },
         "TYPE": "string",
         "FORMAT": "string",
```

```
        "DATA_CLASS": ["kChat"],
        "USE_IN_POLICY": "0",
        "USE_IN_QUERY": "0",
        "USE_IN_NOTIFICATION": "1",
        "USE_IN_LIST": "0",
        "USE_IN_SHOW": "1",
        "USE_IN_DETAIL": "1",
        "IS_MULTIPLE_VALUE": "0"
    }]
}
```

В нашем случае файл лицензии будет располагаться в файле регистрации плагина, т.е. в zip-архиве, в папке licenses. Это описано в manifest-файле следующим образом:

```
    "LICENSE": [
   { "PATH" : "licenses/Traffic Monitor_license_2023-07-24_.license" }
    ],
```

Если файл лицензии отдельно загружен в Traffic Monitor в разделе **Управление** -> **Лицензии**, то в manifest-файле плагина необходимо задать шаблон для поиска загруженных ранее лицензий:

```
"PATTERN_SEARCH_LICENSE": { "operator": "and", "conditions": [{ "common_name":
"MY_COMPANY" }, { "object_type": "im_my_chat" }, { "protocol": "*" } ] }, "LICENSE": [] }
```

Полям `"VENDOR"` и `"SERVICE_MNEMO"` manifest-файла соответствуют наименования `"common_name"` и `"object_type"` в Traffic Monitor. Поэтому в шаблоне для поиска лицензий их значения равны соответственно `MY_COMPANY` и `im_my_chat`.

3.1.1.3 Работа с плагином

После того как лицензия получена, выбраны иконки и описан manifest-файл, создайте файл регистрации стороннего компонента и загрузите его в Traffic Monitor в разделе **Управление** -> **Плагины**.

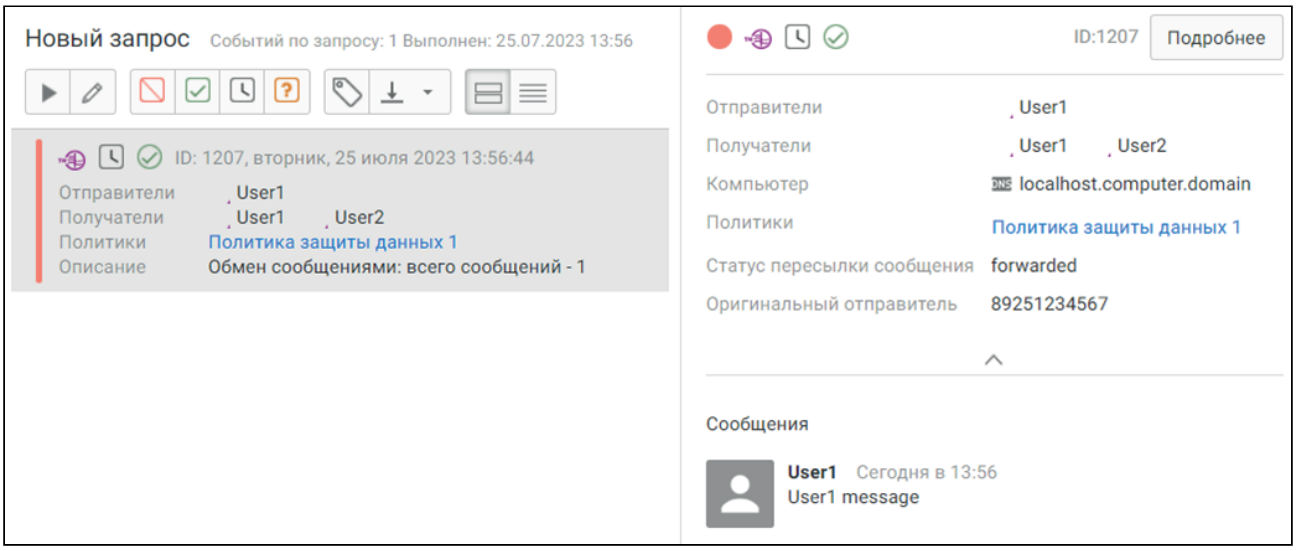
Для начала работы с зарегистрированным плагином можно использовать утилиту pushapi-util. Подробнее о ее использовании см. Описание примеров PushAPI SDK(see page 56).

Чтобы отправить в Traffic Monitor событие класса «**Беседа**» сервиса «**im_my_chat**», используйте команду вида:

```
pushapi-util \
--host Traffic Monitor_host \
--token im_my_chat_token_value \
--id MY_COMPANY \
--class imchat \
--service im_my_chat \
--evtattr event_name:"new service chat Event" \
```

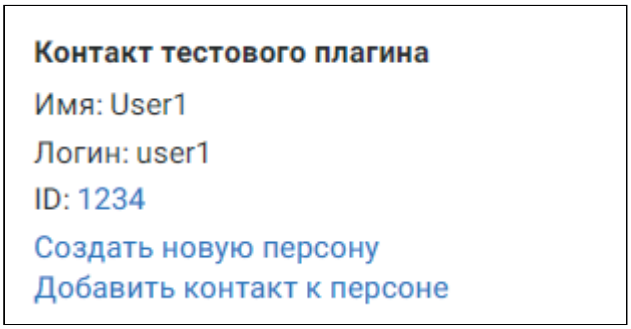
```
--sender "my_chat_Id:1234:'{\"display_name\":\"User1\",\"login\":\"user1\",
\"url\":\"https://my_chat.ru/user1\"}'" \
--receiver "my_chat_Id:76543:'{\"display_name\":\"User2\",\"login\":\"user1\",
\"url\":\"https://my_chat.ru/user2\"}'" \
--mes "my_chat_Id:1234:'{\"display_name\":\"User1\",\"login\":\"user1\",
\"url\":\"https://my_chat.ru/user1\"}'","User1 message" \
--evtattr MY_COMPANY_FORWARDING_STATUS:forwarded \
--evtattr MY_COMPANY_ORIGINAL_SENDER:89251234567
```

В Консоли Управления Traffic Monitor событие будет выглядеть следующим образом:



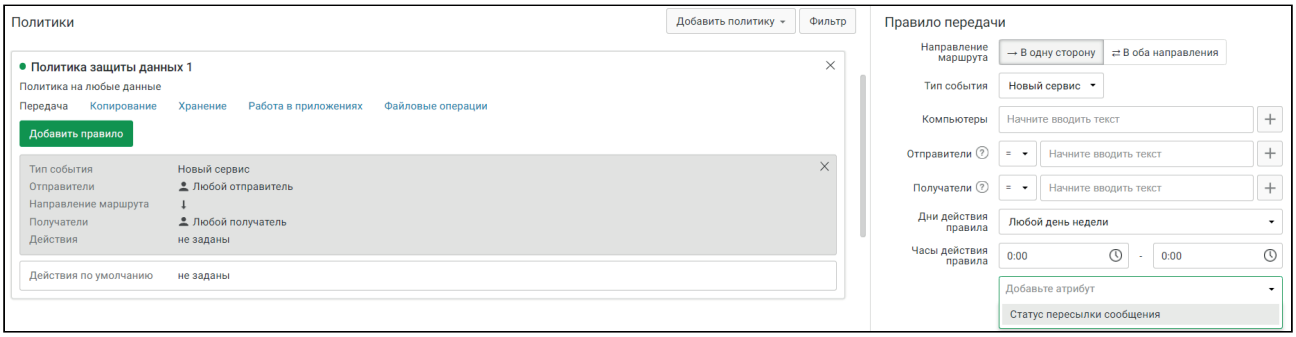
Заданные пользовательские атрибуты отображаются в описании события. Они не являются обязательными, а значит, если они не будут определены в команде, то и в событии они будут отсутствовать.

Так как заполнено поле `"CONTACT_TYPE"` в manifest-файле, контакт пользователя в событии отображается следующим образом:

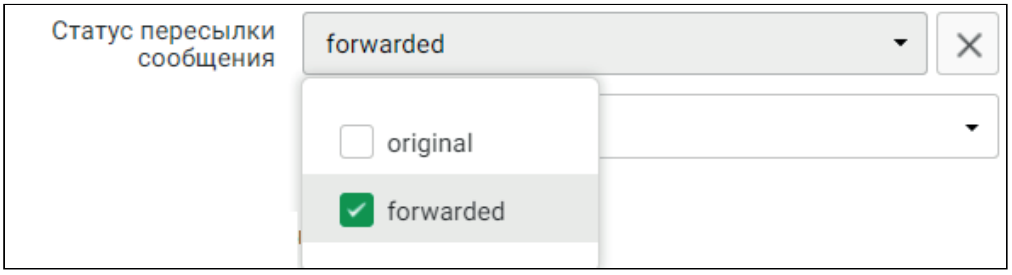


3.1.1.4 Использование пользовательского атрибута в политике защиты данных

У пользовательского атрибута `"MY_COMPANY_FORWARDING_STATUS"` в поле `"USE_IN_POLICY"` задано значение `"1"`. Это означает, что данный атрибут можно использовать при создании политики защиты данных.



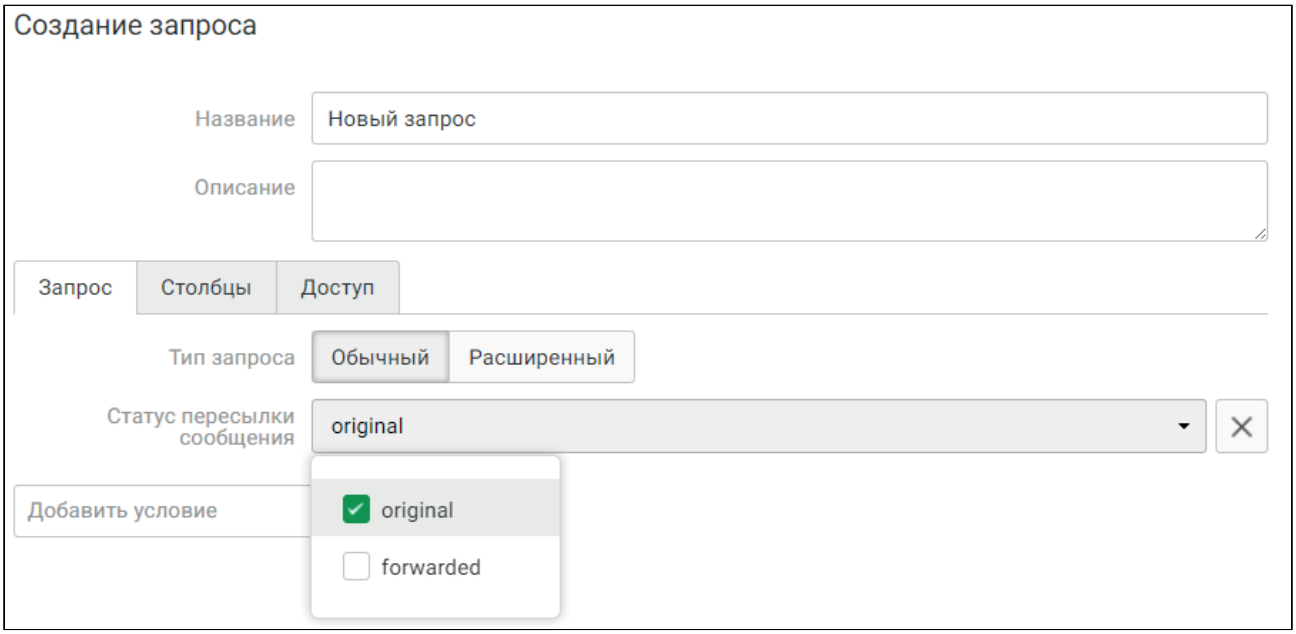
В параметрах Правила передачи отображается атрибут «Статус пересылки сообщения ». Для перехвата пересылаемых сообщений задайте значение «forwarded» для этого атрибута.



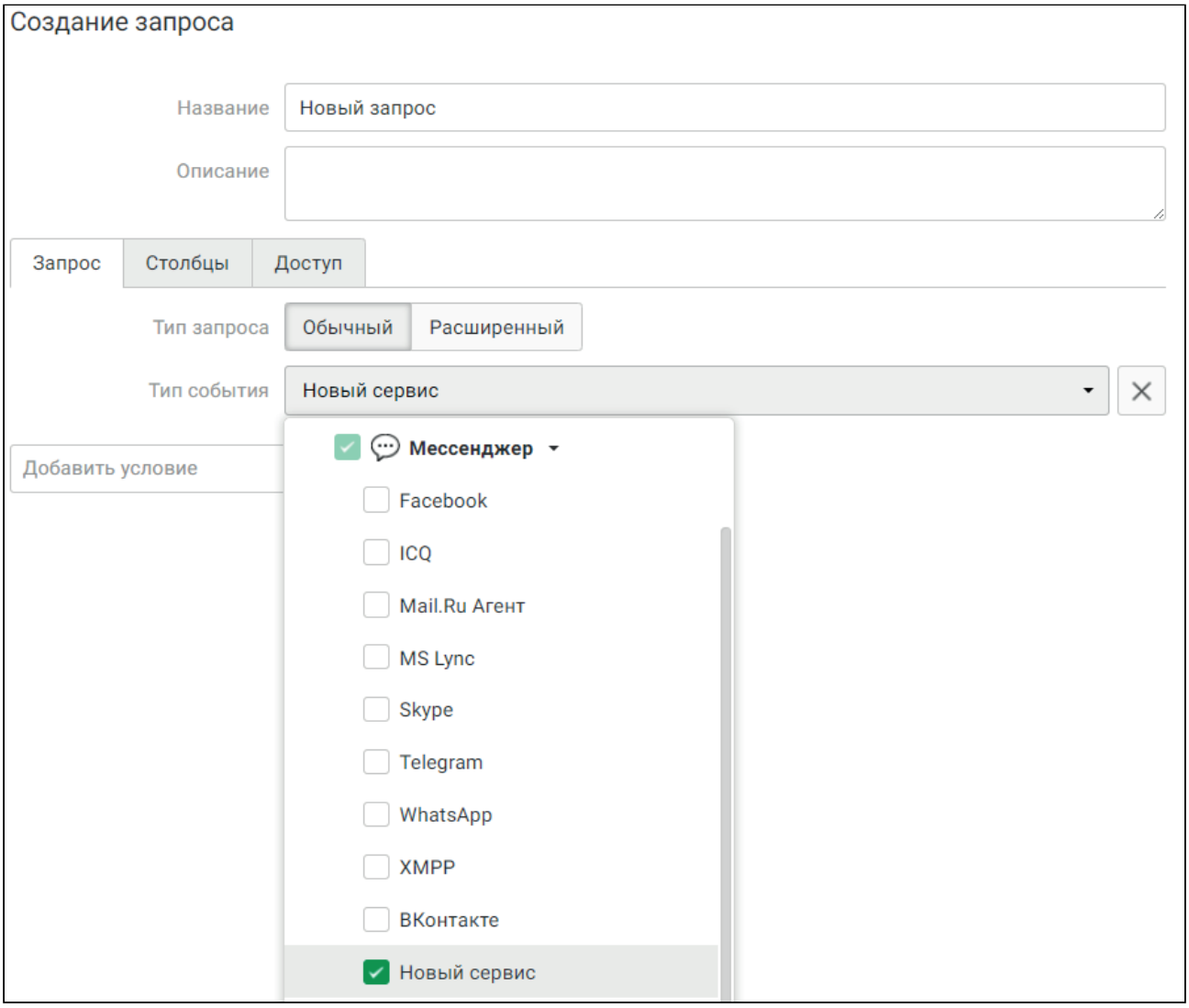
В результате политика будет срабатывать для событий с атрибутом `MY_COMPANY_FORWARDING_STATUS:forwarded`.

3.1.1.5 Создание запросов с новым сервисом

У пользовательского атрибута `"MY_COMPANY_FORWARDING_STATUS"` в поле `"USE_IN_QUERY"` задано значение `"1"`. Данный атрибут можно использовать при создании запросов в разделе **События**.



Также при создании запросов можно использовать и сам сервис. Он доступен при выборе типа события.



3.2 Обновление файла регистрации стороннего компонента

3.2.1 Чтобы обновить файл регистрации стороннего компонента:

1. Удостоверьтесь, что в Traffic Monitor уже зарегистрирован плагин со значением поля "PLUGIN_ID", равным аналогичному значению у нового плагина.
   Информацию обо всех зарегистрированных плагинах можно найти в таблице **plugin** базы данных. Подробнее о работе с БД см. в статье "Администрирование базы данных[2]".
2. Создайте новую версию manifest-файла плагина. Для этого внесите необходимые изменения в исходный manifest-файл.

   Важно!

2 https://kb.infowatch.com/pages/viewpage.action?pageId=217790230

При обновлении плагина не рекомендуется изменять название сервиса, т.е. значение поля "SERVICE_MNEMO", так как в этом случае будет создан новый тип события в Traffic Monitor, а события, перехваченные старым плагином, не будут ассоциированы с плагином с новым значением "SERVICE_MNEMO".
Если вы хотите изменить название типа события, отображаемое в веб-интерфейсе Traffic Monitor, отредактируйте поле "LOCALE".
Если вы измените поле "SERVICE_MNEMO" и при этом не измените поле "LOCALE", то в Traffic Monitor будут существовать два типа события с одинаковыми именами. В Консоли Управления Traffic Monitor их нельзя будет отличить друг от друга.

3. Загрузите новый плагин в Консоль Управления Traffic Monitor.

3.2.2 Удаление дублирующих типов событий

Если вы добавили в Traffic Monitor несколько типов события с разными именами сервиса ("SERVICE_MNEMO"), но с одинаковыми отображаемыми именами ("LOCALE"), вы можете устранить дублирование, удалив из базы данных информацию о неактуальных типах события. При этом можно удалить не весь плагин, а только созданный им тип события, который привел к дублированию.

Важно!

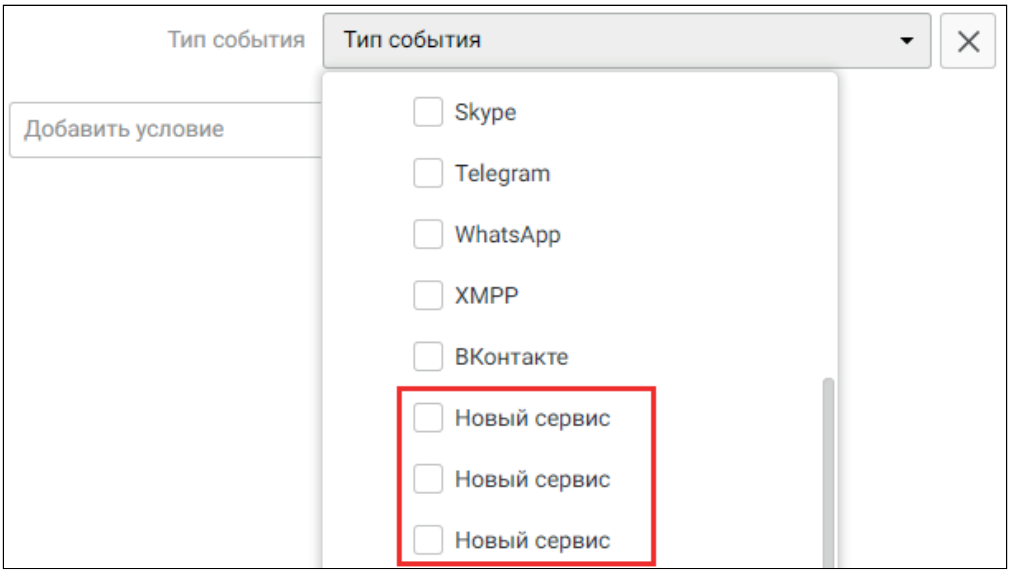
Выполняйте удаление информации из базы данных внимательно и осторожно. Ошибочное удаление информации может привести к некорректной работе Системы.
Перед удалением рекомендуется создавать резервные копии базы данных или таблиц.

Важно!

Удаление дублирующего типа события приведет к тому что, в Консоли Управления Traffic Monitor события этого типа и политики безопасности, использующие этот тип, будут отображаться некорректно. Результаты запросов, которые должны содержать события удаленного типа, не будут отображаться.
Если дублирующий тип события используется в Traffic Monitor, вы можете:

- Не удалять его, а изменить отображаемое имя нового типа события, корректно обновив актуальный плагин.
- Дождаться, когда истечет срок хранения событий дублирующего типа в Traffic Monitor, и тогда удалить этот тип из базы данных.

Рассмотрим процедуру удаления на примере плагина с типом события "im_my_chat" из статьи "Пример файла регистрации стороннего компонента(see page 43)". Для демонстрации плагин был некорректно обновлен дважды. Поле "SERVICE_MNEMO" имело значение "im_my_chat_new" и "im_my_chat_final". Поле "LOCALE" имело значение "Новый сервис" во всех случаях. В результате возникло дублирование. Например, при создании запроса в разделе **События** Консоли Управления Traffic Monitor стали отображаться три типа события Новый сервис:



Чтобы удалить дублирующие типы события:

1. В таблице **plugin** базы данных найдите идентификаторы плагинов, которые создали дублирование. Эти идентификаторы записаны в поле **plugin_ id**.
   В нашем примере всего одна строка с таким плагином:

	plugin_id [PK] character varying (40)	display_name character varying (4000)	description character varying (4000)
1		InfoWatch Device Monitor	InfoWatch Device Monitor
2	4B8038D390396EB6E0540100007F1CA200000001	Test plugin final	Test plugin to demonstrate
3		InfoWatch Data Discovery	Предустановленный плагин...
4		InfoWatch Sample documen...	Предустановленный плагин...

2. Найдите идентификаторы типов события и сервисов, созданных этими плагинами. Для этого:
   a. В таблице **plugin_data** найдите все данные для этих плагинов. Это можно сделать с помощью SQL-запроса следующего вида:

```
SELECT * FROM iwTraffic Monitor.plugin_data
WHERE plugin_id in ('4B8038D390396EB6E0540100007F1CA200000001')
ORDER BY plugin_id
```

   Если дублирующих плагинов несколько, то укажите их значения поля **plugin_id** в скобках через запятую.
   Результат запроса для нашего примера:

	plugin_id character varying (40)	entry_type character varying (255)	entry_value character varying (4000)
1	4B8038D390396EB6E0540100007F1CA200000001	CONTACT_TYPE_MNEMO	my_chat_Id
2	4B8038D390396EB6E0540100007F1CA200000001	CONTACT_TYPE_SCOPE	person
3	4B8038D390396EB6E0540100007F1CA200000001	OBJECT_TYPE_OBJECT_TYPE_ID	FAF6B5B261204B31BEED5F67798A477C00000000
4	4B8038D390396EB6E0540100007F1CA200000001	OBJECT_TYPE_SERVICE_ID	DD6ECB5227DA11E2B507CBDB6088709B00000000
5	4B8038D390396EB6E0540100007F1CA200000001	OBJECT_TYPE_MNEMO	im_my_chat_final
6	4B8038D390396EB6E0540100007F1CA200000001	object_header	my_company_forwarding_status
7	4B8038D390396EB6E0540100007F1CA200000001	object_header	my_company_original_sender
8	4B8038D390396EB6E0540100007F1CA200000001	license	8fee32283d89c06fafae1424d52871469ae0ee5235ade72a42d512ed2016b6...

   b. В полученных данных найдите для каждого значения **plugin_ id** те строки, в которых значение поля **entry_ type** равно OBJECT_TYPE_OBJECT_TYPE_ID или

OBJECT_TYPE_SERVICE_ID. Запомните пары значений этих полей. Это идентификатор типа события, созданного дублирующим плагином, и идентификатор сервиса.
В таблице из примера была найдена одна пара. В ней:
OBJECT_TYPE_OBJECT_TYPE_ID = "FAF6B5B261204B31BEED5F67798A477C00000000"
OBJECT_TYPE_SERVICE_ID = "DD6ECB5227DA11E2B507CBDB6088709B00000000"

3. В таблице **object_ type** найдите записи с парами идентификаторов из предыдущего пункта. Вы можете использовать для этого запрос следующего вида:

```
SELECT * FROM iwTraffic Monitor.object_type
WHERE object_type_id = 'FAF6B5B261204B31BEED5F67798A477C00000000'
  AND service_id = 'DD6ECB5227DA11E2B507CBDB6088709B00000000'
```

	object_type_id [PK] character varying (40)	service_id character varying (40)	language [PK] character varying (3)	display_name character varying (4000)	mnemo character varying (256)
1	FAF6B5B261204B31BEED5F67798A477C00000000	DD6ECB5227DA11E2B507CBDB6088709B00000000	eng	New service	im_my_chat_final
2	FAF6B5B261204B31BEED5F67798A477C00000000	DD6ECB5227DA11E2B507CBDB6088709B00000000	rus	Новый сервис	im_my_chat_final

4. По полю **mnemo** определите пару идентификаторов, которая относится к актуальному плагину. Если эта пара единственная, то перейдите к п.6.
В нашем примере поле **mnemo** у найденных записей имеет значение "im_my_chat_final", т.е. записи относятся к актуальному плагину.
5. Для каждой пары, не относящейся к актуальному плагину, выполните следующие действия:
   a. Убедитесь, что в таблице **object** нет событий с типом события из этой пары. Для этого выполните запрос, используя значения пары идентификаторов – OBJECT_TYPE_ID и SERVICE_ID:

```
SELECT count(*) FROM object
WHERE object_type_code='<OBJECT_TYPE_ID>'
AND service_code='<SERVICE_ID>'
```

   Если значение запроса равно нулю, то событий нет.

> **Важно!**
>
> Если в таблице **object** есть события этого типа, но вы все равно хотите удалить его, рекомендуется сначала удалить ежедневные табличные пространства, содержащие события этого типа, или обратиться в службу технической поддержки.
> Подробнее об удалении ежедневных табличных пространств см. в разделе "Администрирование базы данных[3]".

   b. Из таблицы **object_type** удалите записи, связанные с данным типом события. Это записи, найденные в п.3.
   c. Из таблицы **plugin_ data** удалите записи, связанные с данным типом события. Например, это могут быть:
      - идентификатор типа события – "OBJECT_TYPE_OBJECT_TYPE_ID"
      - идентификатор сервиса – "OBJECT_TYPE_SERVICE_ID"
      - название типа события – "OBJECT_TYPE_MNEMO"
6. В таблице **object_ type** найдите записи, относящиеся к дублирующим типам события. Для этого выполните следующий запрос:

3 https://kb.infowatch.com/pages/viewpage.action?pageId=217790230

```
SELECT * FROM iwTraffic Monitor.object_type
WHERE display_name = '<EVENT_TYPE_DISPLAY_NAME>'
```

где `<EVENT_TYPE_DISPLAY_NAME>` – отображаемое имя типа события.

В нашем примере запрос будет иметь следующий вид:

```
SELECT * FROM iwTraffic Monitor.object_type
WHERE display_name = 'Новый сервис '
```

	object_type_id [PK] character varying (40)	service_id character varying (40)	language [PK] character varying (3)	display_name character varying (4000)	mnemo character varying (256)
1	76505E31A1DE46698339D6AC00DD0DE8000000…	DD6ECB5227DA11E2B507CBDB6088709B00000000	rus	Новый сервис	im_my_chat_new
2	083030E950F948B891CCE1B048A8E70300000000	DD6ECB5227DA11E2B507CBDB6088709B00000000	rus	Новый сервис	im_my_chat
3	FAF6B5B261204B31BEED5F67798A477C00000000	DD6ECB5227DA11E2B507CBDB6088709B00000000	rus	Новый сервис	im_my_chat_final

7. По полю **mnemo** определите дублирующие типы события.
   В нашем случае "im_my_chat_final" – это тип события актуального плагина. Значит "im_my_chat" и "im_my_chat_new" – это старые дублирующие типы, которые мы и хотим удалить.
8. Для каждого дублирующего типа события:
   a. Убедитесь, что в таблице **object** нет событий с этим типом. Порядок действий аналогичен п.5а данной инструкции.
   b. В таблице **object_ type** найдите и удалите все записи, относящиеся к данному типу. Они будут отличаться только локализацией, т.е. значениями полей **language** и **display_ name**. Для этого можно использовать запрос:

      ```
      SELECT * FROM iwTraffic Monitor.object_type
      WHERE mnemo = '<MNEMO>'
      ```

      Например, для типа "im_my_chat" будут выведены следующие записи:

	object_type_id [PK] character varying (40)	service_id character varying (40)	language [PK] character varying (3)	display_name character varying (4000)	mnemo character varying (256)
1	083030E950F948B891CCE1B048A8E70300000000	DD6ECB5227DA11E2B507CBDB6088709B00000000	rus	Новый сервис	im_my_chat
2	083030E950F948B891CCE1B048A8E70300000000	DD6ECB5227DA11E2B507CBDB6088709B00000000	eng	New service	im_my_chat

9. Убедитесь, что дублирующие типы события отсутствуют в Консоли Управления Traffic Monitor.

4 Описание примеров PushAPI SDK

Примеры представляют собой исходные коды консольных утилит, которые реализуют загрузку событий в Traffic Monitor, используя Thrift-интерфейс pushAPI SDK.

Первая утилита – pushapi-cpp. Реализована на языке C++. Подготовлена к сборке в операционных системах Windows и Linux. Данная утилита демонстрирует сценарии загрузки всех типов событий, которые уже зарегистрированы в системе Traffic Monitor.

Вторая утилита – pushapi-csharp. Реализована на языке C#. Подготовлена к сборке в операционной системе Windows. Данная утилита демонстрирует сценарии загрузки всех типов событий, которые уже зарегистрированы в системе Traffic Monitor.

Третья утилита – pushapi-demo.py. Реализована на языке Python. Подготовлена к сборке в операционных системах Windows и Linux. Демонстрирует сценарии загрузки всех типов событий, которые уже зарегистрированы в системе Traffic Monitor.

Четвертая утилита – pushapi-util. Реализована на языке C++. Подготовлена к сборке в операционных системах Windows и Linux. Данная утилита имеет развитый интерфейс командной строки, который позволяет загружать произвольные события в систему Traffic Monitor, в том числе те, которые зарегистрированы внешней системой. Вот описание командной строки:

```
./pushapi-util
--host <pushapi address > [--port <pushapi port>]
--token <plugin token>
--id <company>
--class <event class>
--service <event service>
--evtattr <attr_name:attr_value,…, attr_name:attr_value>
--sender
<contact_type:contact_value,attr_name:attr_value,…,attr_name:attr_value>
--receiver
<contact_type:contact_value,attr_name:attr_value,…,attr_name:attr_value>
--data file:<data_falie_name>, attr_name:attr_value,…, attr_name:attr_value
--mes <contact_type:contact_value,"message text"
[--addctx]
[--show_event]
```

Где:

- <pushapi address > – адрес pushAPI-сервера;
- <pushapi port> – порт pushAPI-сервера, если не указан, используется значение по умолчанию;
- <plugin token> – токен авторизации плагина. Скопируйте его значение в разделе данного плагина консоли управления Traffic Monitor;
- <company> – идентификатор компании-производителя плагина. Произвольная строка, обязательно совпадающая с той, что использовалась для генерации лицензии для плагина;
- <event class> – класс передаваемого события;
- <event service> – сервис передаваемого события. Плагин может зарегистрировать свои перехватываемые сервисы по имени и передавать их;

- --evtattr – атрибуты передаваемого события. Посмотреть можно в списке команд с примером. Некоторые имена атрибутов имеют фиксированные значения и бизнес-смысл;
- --sender – отправитель события. Первая пара параметров определяет контакт отправителя. Далее указываются атрибуты отправителя. Существуют три «особых» типа контактов: dev (отправитель/получатель является устройством), res (отправитель/получатель является ресурсом), ws (отправитель/получатель является «Рабочей станцией»). Если не задан ни один отправитель, то будет сгенерировано два отправителя с типом «Персона» и контактом типа «auth»; и с типом «Рабочая станция» и контактом dnshostname. Задан может быть только один отправитель;

  Пример:

  Для события с отправителем с аккаунтом Вконтакте и указанием мета-информации:
  `--sender "vkontakte:349051292:'{\"display_name\":\"some name\",\"url\":\"some_url\",\"login\":\"some_login\"}'"`
- --receiver – получатель события. Параметры аналогичны отправителю;
- --data – данные события. Первая пара атрибутов указывает на путь к файлу, где содержится тело события. Остальные параметры описывают атрибуты этих данных. Для некоторых типов событий некоторые атрибуты обязательны;
- --mes – передача сообщений класса «Беседа». Указывается контакт отправителя сообщения и его содержимое. Сообщений может быть несколько;
- --addctx – необязательный параметр. Автоматически добавляет к событию двух отправителей с типом «Персона» и контактом типа «auth»; и с типом «Рабочая станция» и контактом типа dnshostname;
- --show_event – необязательный параметр. Выводит информацию о структуре События (Event) в консоль в формате JSON.

Утилиту pushapi-util можно установить из пакета QATOOLS. Для этого:

1. На рабочей станции с установленным Traffic Monitor извлеките pushapi-util в каталог `/opt/iw/Traffic`. Утилита pushapi-util находится в пакете `iwTraffic Monitor-qatools-x.x.x.xxx-release.xrpm/iwTraffic Monitor-qatools-x.x.x.xxx-release.x86_64.cpio/./opt/iw/Traffic`

   Здесь `x.x.x.xxx` – это номер сборки.

   Примечание:

   Если pushapi-util извлечена в другой каталог, то добавьте его в переменные окружения:
   `export PATH=$PATH:<directory>`
2. Перейдите в каталог, куда извлечена утилита pushapi-util.
3. Сделайте файл утилиты исполняемым:
   `chmod u+x pushapi-util`

Примеры вызова утилиты pushapi-util есть в исходных кодах примеров, а также приведены в статье Примеры использования утилиты pushapi-util(see page 58).

Особенности сборки примеров PushAPI SDK приведены в следующих статьях:

- Сборка примеров под Linux pushAPI SDK(see page 66)
- Сборка примеров под Windows pushAPI SDK(see page 67)

В состав примеров SDK также входят:

- файл plugin.json – JSON-схема файла manifest.json, который входит в состав файла регистрации внешнего плагина;
- файл plugin.zip – пример файла регистрации внешнего плагина.

> **Важно!**
>
> В состав файла входит просроченная лицензия. Для регистрации плагина на основании файла примера нужно получить новую лицензию.

4.1 Примеры использования утилиты pushapi-util

С помощью тестовой утилиты pushapi-util можно отправить в Traffic Monitor события для любых встроенных сервисов и классов, используя встроенный плагин Device Monitor.

> **Примечание:**
>
> В командах в качестве значения атрибута token используйте значение токена Device Monitor. Его можно скопировать в разделе **Управление** -> **Плагины** Консоли Управления Traffic Monitor.

4.1.1 Примеры отправки событий класса «Почта»

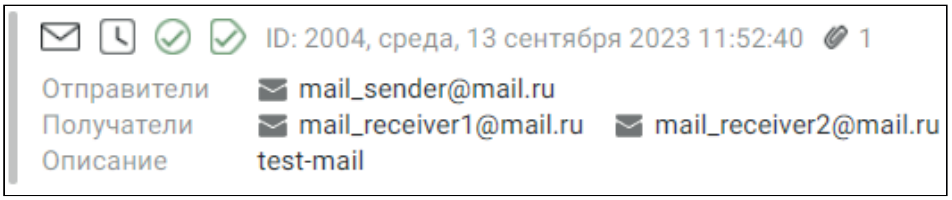
Пример 1:

Чтобы отправить в Traffic Monitor событие класса «**Почта**» сервиса «**Электронная почта**», используйте команду вида:

```
pushapi-util \
--host Traffic Monitor_host \
--token device_monitor_token_value \
--id iw \
--class email \
--service email \
--evtattr event_name:"Send email" \
--sender email:"mail_sender@mail.ru" \
--receiver email:"mail_receiver1@mail.ru" \
--receiver email:"mail_receiver2@mail.ru" \
--data file:/root/test_email.eml
```

Где `test_email.eml` – это файл, содержащий передаваемое письмо. Атрибуты `data`, `sender` и `receiver` являются обязательными.

В Консоли Управления Traffic Monitor событие будет выглядеть следующим образом:



Пример 2:

Чтобы отправить в Traffic Monitor событие класса «**Почта**» сервиса «**Веб-почта**», используйте команду вида:

```
pushapi-util \
--host Traffic Monitor_host \
--token device_monitor_token_value \
--id iw \
--class email \
--service email_web \
--evtattr event_name:"Send email_web" \
--sender email:"mail_sender@mail.ru" \
--receiver email:"mail_receiver1@mail.ru" \
--data file:/root/test_email.eml
```

4.1.2 Примеры отправки событий класса «Беседа»

Пример 1:

Чтобы отправить в Traffic Monitor событие класса «**Беседа**» для мессенджера ICQ, используйте команду вида:

```
pushapi-util \
--host Traffic Monitor_host \
--token device_monitor_token_value \
--id iw \
--class imchat \
--service im_icq \
--evtattr event_name:"ICQ chat event" \
--receiver icq:IcqReceiver \
--mes icq:IcqSender,"Confidential information"
```

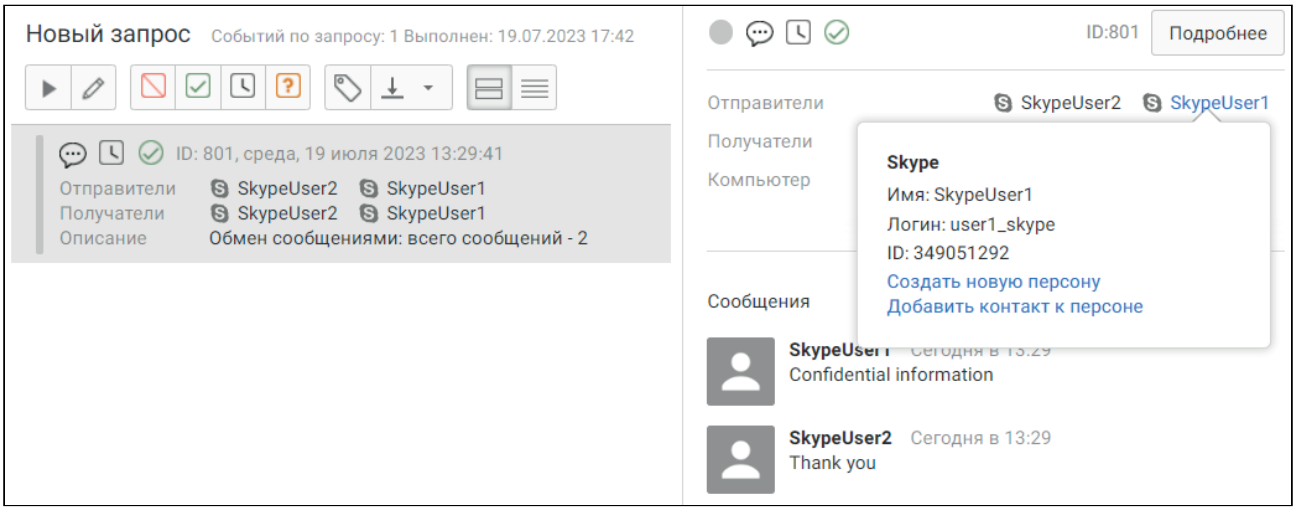
Атрибуты `mes` и `receiver` являются обязательными.

Пример 2:

Чтобы отправить в Traffic Monitor событие класса «**Беседа**» для мессенджера Skype, содержащее метаинформацию об отправителях, используйте команду вида:

```
pushapi-util \
--host Traffic Monitor_host \
--token device_monitor_token_value \
--id iw \
--class imchat \
--service im_skype \
--evtattr event_name:"Skype event" \
--mes "skype:349051292:'{\"display_name\":\"SkypeUser1\",\"login\":
\"user1_skype\"}'","Confidential information" \
--mes "skype:349051100:'{\"display_name\":\"SkypeUser2\",\"login\":
\"user2_skype\"}'","Thank you"
```

Здесь для отправителей сообщений указана метаинформация, включающая в себя отображаемое имя и логин отправителя. В Консоли Управления Traffic Monitor событие будет выглядеть следующим образом:



4.1.3 Примеры отправки событий класса «Печать, копирование и сканирование»

Чтобы отправить в Traffic Monitor событие класса «**Печать, копирование и сканирование**» сервиса «**Принтер и МФУ**», используйте команды следующего вида:

```
pushapi-util \
--host Traffic Monitor_host \
--token device_monitor_token_value \
--id iw \
--class mfp \
--service print \
--evtattr event_name:"Print event" \
```

```
--receiver dev:"Some Printer Name" \
--data file:/root/test.png,filename:test.png
```

и

```
pushapi-util \
--host Traffic Monitor_host \
--token device_monitor_token_value \
--id iw \
--class mfp \
--service print \
--evtattr event_name:"Print event" \
--receiver dev:"Some Printer Name",print_port_name:"Some print port
name",print_location:"Some print location" \
--data file:/root/test.png,print_copies:3,print_job_name:"Some print job name"
```

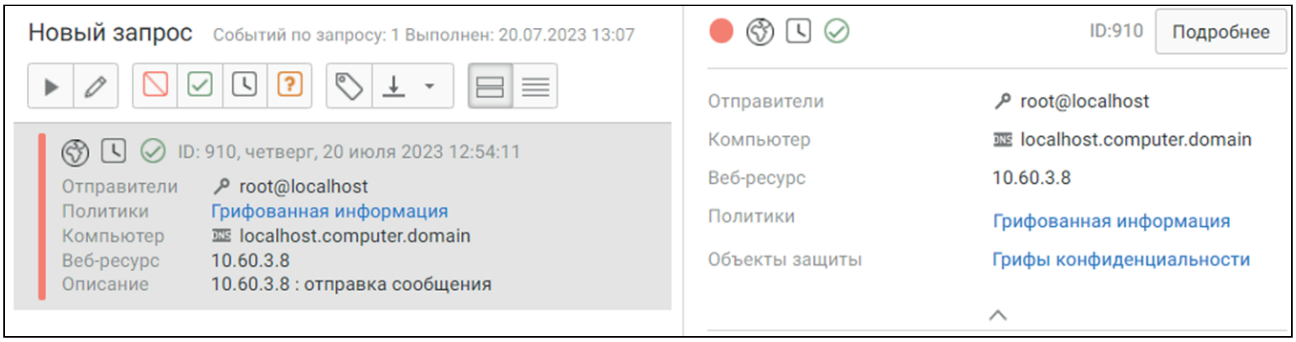
Атрибуты `receiver` и `data` являются обязательными. Также команда должна содержать один из атрибутов `filename` и `print_job_name`.

4.1.4 Примеры отправки событий класса «Интернет-активность»

Чтобы отправить в Traffic Monitor событие класса «**Интернет-активность**» одноименного сервиса, используйте команду следующего вида:

```
pushapi-util \
--host Traffic Monitor_host \
--token device_monitor_token_value \
--id iw \
--class web \
--service web_common \
--evtattr event_name:"Web event1" \
--receiver res:"drive.google.com",res_destination_url:"/file/test" \
--data file:/root/test
```

Где `test` – это файл, содержащий POST-запрос. В нашем примере в передаваемом запросе присутствует фраза «строго конфиденциально». При настроенных политиках защиты данных событие будет выглядеть следующим образом:



Атрибуты `data` и `receiver` являются обязательными.

4.1.5 Пример отправки событий класса «Фотосъемка»

Для отправки событий класса «**Фотосъемка**» нет встроенных сервисов, поэтому для этого необходимо зарегистрировать свой сервис и использовать команду следующего вида:

```
pushapi-util \
--host Traffic Monitor_host \
--token your_plugin_token_value \
--id your_company \
--class photo \
--service your_multimedia_service \
--evtattr event_name:"Photo event" \
--receiver ws:"ws1.domain.org",workstation_type:"ws_type_computer" \
--data file:/root/test.png,filename:test.png
```

4.1.6 Примеры отправки событий класса «Обмен файлами»

Атрибуты `receiver`, `data` и `destination_file_path` обязательны для событий сервисов «**Облачные хранилища**», «**Терминальная сессия**», «**Сетевые ресурсы**», «**FTP**» и «**Съемное устройство**».

Атрибуты `receiver` и `data` обязательны для событий класса **"Обмен файлами"** сервиса «**Мессенджеры**».

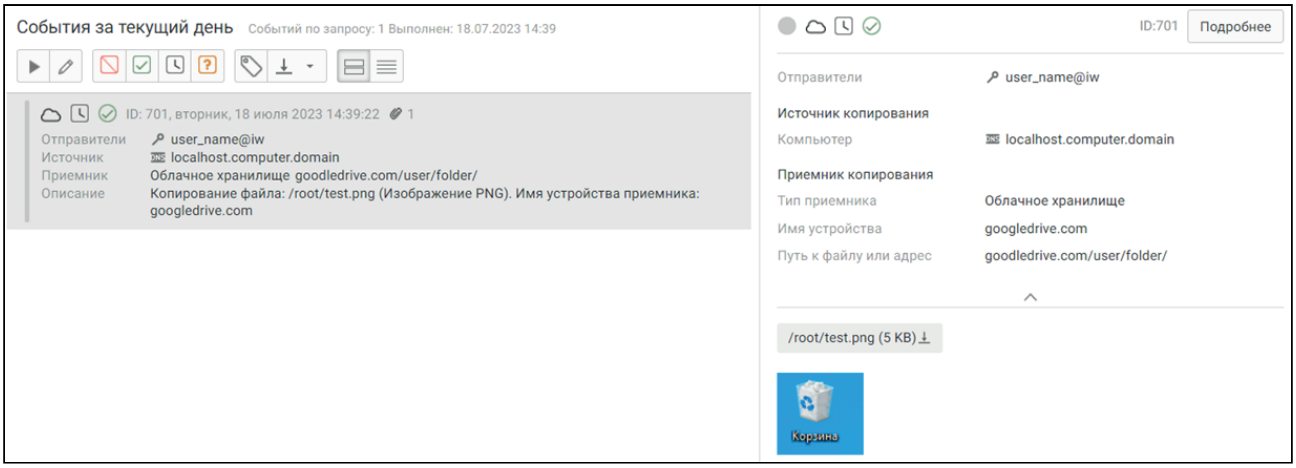
Пример 1:

Чтобы отправить в Traffic Monitor событие класса «**Обмен файлами**» сервиса «**Облачные хранилища**», используйте команду вида:

```
pushapi-util \
--host Traffic Monitor_host \
--token device_monitor_token_value \
--id iw \
--class fileexch \
```

```
--service cloud_storage \
--evtattr event_name:"cloud_storage event" \
--sender auth:user_name@iw \
--receiver res:googledrive.com,res_destination_url:"goodledrive.com/user/
folder/"  \
--data file:/root/test.png,destination_file_path:"Send file here"
```

Событие в Консоли Управления Traffic Monitor:



Пример 2:

Чтобы отправить в Traffic Monitor событие класса «**Обмен файлами**» сервиса «**Терминальная сессия**», используйте команду вида:

```
pushapi-util \
--host Traffic Monitor_host \
--token device_monitor_token_value \
--id iw \
--class fileexch \
--service terminal_session \
--evtattr event_name:"terminal_session event" \
--sender phone:+79057777777 \
--receiver res:root@test.com \
--data file:/root/test.png,destination_file_path:"test.com/folder"
```

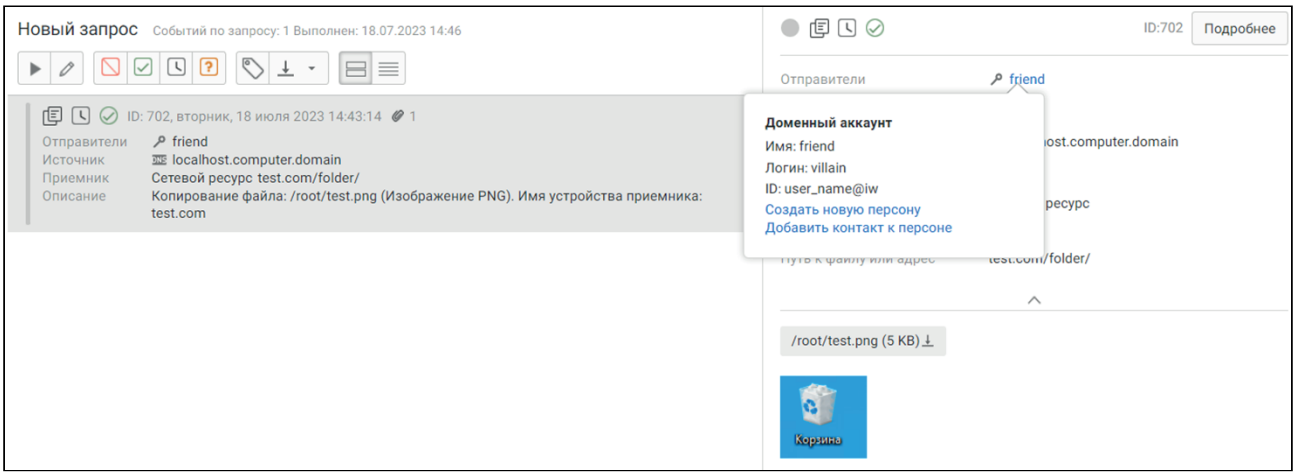
Пример 3:

Чтобы отправить в Traffic Monitor событие класса «**Обмен файлами**» сервиса «**Сетевые ресурсы**», используйте команду вида:

```
pushapi-util \
--host Traffic Monitor_host \
--token device_monitor_token_value \
```

```
--id iw \
--class fileexch \
--service network_resource \
--evtattr event_name:"network_resource event" \
--sender "auth:user_name@iw:'{\"display_name\":\"friend\",\"login\":
\"villain\"}'" \
--receiver res:test.com,res_destination_url:"test.com/folder/ "  \
--data file:/root/test.png,destination_file_path:"Send file here"
```

Событие в Консоли Управления Traffic Monitor:



Пример 4:

Чтобы отправить в Traffic Monitor событие класса «**Обмен файлами**» сервиса «**FTP**», используйте команду вида:

```
pushapi-util \
--host Traffic Monitor_host \
--token device_monitor_token_value \
--id iw \
--class fileexch \
--service ftp \
--evtattr event_name:"FTP Event" \
--sender login:villain \
--receiver res:ftp.site.org,res_destination_url:"catalog\catalog\res.file" \
--data file:/root/test.png,destination_file_path:"Send file here"
```

Пример 5:

Чтобы отправить в Traffic Monitor событие класса «**Обмен файлами**» сервиса «**Съемное устройство**», используйте команду вида:

```
pushapi-util \
--host Traffic Monitor_host \
--token device_monitor_token_value \
--id iw \
--class fileexch \
--service file_copy_removable \
--evtattr event_name:"Copy file on device" \
--receiver dev:"Removable device" \
--data file:/root/test.png,source_file_path:"Copy from
here",destination_file_path:"Send file here"
```

Пример 6:

Чтобы отправить в Traffic Monitor событие класса «**Обмен файлами**» для мессенджера ICQ, используйте команду вида:

```
pushapi-util \
--host Traffic Monitor_host \
--token device_monitor_token_value \
--id iw \
--class fileexch \
--service im_icq \
--evtattr event_name:"ICQ File Transfer Event" \
--receiver icq:IcqReceiver \
--data file:/root/test.png
```

Пример 7:

Чтобы отправить в Traffic Monitor событие класса «**Обмен файлами**» для мессенджера Skype, используйте команду вида:

```
pushapi-util \
--host Traffic Monitor_host \
--token device_monitor_token_value \
--id iw \
--class fileexch \
--service im_skype \
--evtattr event_name:"Skype File Transfer Event" \
--sender skype:SkypeSender \
--receiver skype:SkypeReceiver1 \
--receiver skype:SkypeReceiver2 \
--data file:/root/test.png,filename:test.png
```

4.1.7 Пример отправки события класса «Звонок»

Чтобы отправить в Traffic Monitor событие класса «**Звонок**» сервиса «**Мессенджеры**», используйте команду вида:

```
pushapi-util \
--host Traffic Monitor_host \
--token device_monitor_token_value \
--id iw \
--class voicetalk \
--service im_skype \
--evtattr event_name:"Skype Voice Talk" \
--sender skype:SkypeSender \
--receiver skype:SkypeReceiver \
--data file:/root/sound_msg.wav,call_duration:7
```

Событие содержит аудиофайл `sound_msg.wav`, а длительность разговора составляет 7 секунд. В Консоли Управления Traffic Monitor аудиофайл будет доступен для скачивания. Атрибуты `receiver`, `data` и `call_duration` обязательны для событий класса «**Звонок**» сервиса «**Мессенджеры**».

4.1.8 Примеры отправки событий класса «Запись мультимедиа»

Для отправки событий класса «**Запись мультимедиа**» нет встроенных сервисов, поэтому для этого необходимо зарегистрировать свой сервис и использовать команду следующего вида:

```
pushapi-util \
--host Traffic Monitor_host \
--token your_plugin_token_value \
--id your_company \
--class multimedia \
--service your_multimedia_service \
--evtattr event_name:"Camera Event" \
--receiver ws:"ws1.domain.org",workstation_type:"ws_type_computer" \
--data file:/root/video.avi
```

4.2 Сборка примеров под Linux pushAPI SDK

Для сборки примеров под Linux нужно иметь установленный пакет **cmake** и компилятор с поддержкой стандарта C++11. Все остальные зависимости определяются **cmake**-скриптом. Для сборки может понадобиться установка Thrift Runtime-библиотек, библиотеки Boost и OpenSSL, а именно:

- boost 1.62
- openssl 1.0.1
- thrift 0.11.0

4.3 Сборка примеров под Windows pushAPI SDK

Для сборки примеров под Windows нужно иметь установленную программу Visual Studio 2013. В состав примеров входит Toolchain, который содержит boost, OpenSSL, а также исходники Thift Runtime-библиотек, которые собираются в составе примера. В Toolchain входит утилита thrift, которая компилирует thift-схему в соответствующий исходный код. Пользователь может использовать свою реализацию boost, OpenSSL и Thrift – для этого нужно определить свои переменные окружения:

`$(BOOST_ROOT)`

`$(OPENSSL_ROOT_DIR)`

`$(THRIFT_INSTALL_DIR)`

> **Важно!**
>
> Самостоятельную сборку Thrift необходимо осуществлять с OpenSSL, т.к. pushAPI работает по SSL-соединению.

5 Диагностика ошибок при отправке событий

Проблемы с отправкой событий по Push API могут быть связаны с:

1. Неполадками в веб-интерфейсе, Чтобы включить режим отладки на сервере:
   a. Зайдите на сервер по SSH.
   b. Откройте на редактирование файл `/opt/iw/Traffic Monitor5/etc/web.conf .`
   c. Включите режим отладки, установив значение: `"debug": true .`
   d. Сохраните файл.
   e. Обновите страницу в веб-интерфейсе.
   f. Перейдите в отладочную консоль с подробными логами по запросам к API. Для этого введите url: `https://<IP_сервера>/api/debug/default .`
2. Отсутствие доверенных сертификатов. Добавьте самоподписанные сертификаты в доверенные. Пути до сертификатов:
   - Веб: `/opt/iw/Traffic Monitor5/etc/certification/web-server.pem`
   - XAPI: `/opt/iw/Traffic Monitor5/etc/cert/server.pem`
3. Файловыми очередями. Чтобы посмотреть файловые очереди и очереди ошибок на сервере TM, перейдите в:
   - `opt/iw/Traffic Monitor5/queue`
   - `opt/iw/Traffic Monitor5/errors`
4. Файлами с ошибками экстракции. Они находятся в папке, указанной в параметре "Dreamcatcher" для каждого экстрактора в конфигурационном файле `/opt/iw/Traffic Monitor5/etc/extractors.con`
5. Работой служб iw_xapi_xapi и iw_xapi_puppy. Чтобы получить информацию от логгеров разных программных компонентов:
   a. На сервере Traffic Monitor откройте на редактирование конфигурационный файл `/opt/iw/Traffic Monitor5/etc/xapi.conf`
   b. Установите следующие значения логгеров:

```
"GlobalLevel": "debug",
"Loggers":
{
"Filequeue": "debug",
"RawMailDump": "fatal",
"Root": "debug",
"puppy": "debug",
"thrift_handler": "warning"
}
```

   где:
   - Filequeue - логирование действий файловой очереди.
   - RawMailDump - сырые данные почты
   - puppy - логгер Push API
   - thrift_handler - данные thrift-структур, поступивших по сети, для проверки трафика
   - Root - корневой логгер, отвечающий за чтение из БД, создание индексов, архивирование, удаление индексов

   c. Сохраните файл.

d. Выполните команды:
```
iwTraffic Monitor restsrt xapi_xapi
iwTraffic Monitor restart xapi_puppy
```
e. Отправьте событие в Traffic Monitor по Push API (подробнее см. "Примеры использования утилиты pushapi-util(see page 58)"). Лог-файлы будут сохранены в директории `/var/log/infowatch`

6 Загрузка событий в InfoWatch Traffic Monitor по протоколу HTTP

Для отправки события в Traffic Monitor по HTTP-протоколу реализован метод REST API `JsonPush`, который преобразует запрос в PushAPI-формат. Подробнее о REST API см. "Функциональное описание программного интерфейса REST API[4]".

Версия API: ≥ 1.8

Метод: `POST`

Ресурс: `/push/event`

Описание: Метод отправляет событие в Traffic Monitor по протоколу HTTP.

Пример:

Запрос на получения событий с типом Облачные хранилища

```
curl -X 'POST' \
  'https://server.company.ru/xapi/push/event' \
   'X-API-Auth-Token: brse99dugp1cdbr69axz'
   'X-API-CompanyId: iw'
   'X-API-Version: 1.8'
  -F 'event={
  "evt_class": 4,
  "evt_service": "cloud_storage",
  "evt_attributes": [
    {
      "name": "capture_ts",
      "value": "2023-10-26T20:07:51+03:00"
    },
    {
      "name": "capture_server_ip",
      "value": "127.0.0.1"
    },
    {
      "name": "capture_server_fqdn",
      "value": "localhost"
    }
  ],
  "evt_senders": [
    {
      "identity_id": 1,
      "identity_type": 0,
      "identity_contacts": [
        {
          "name": "auth",
```

4 https://kb.infowatch.com/pages/viewpage.action?pageId=217790392

```
          "value": "user_name@iw"
        }
      ]
    },
    {
      "identity_id": 2,
      "identity_type": 1,
      "identity_contacts": [
        {
          "name": "dnshostname",
          "value": "localhost.computer.domain"
        }
      ]
    }
  ],
  "evt_receivers": [
    {
      "identity_id": 3,
      "identity_type": 3,
      "identity_contacts": []
    }
  ],
  "evt_data": [
    {
      "data_id": 1,
      "data_attributes": [
        {
          "name": "filename",
          "value": "res.txt"
        },
        {
          "name": "destination_file_path",
          "value": "goodledrive.com/user/folder/"
        }
      ]
    }
  ],
  "evt_destination": {
    "identity_id": 4,
    "identity_type": 3,
    "identity_contacts": [],
    "identity_attributes": [
      {
        "name": "name",
        "value": "googledrive.com"
      }
    ]
  }
}' \
  -F 'files[]='
```

Запрос на получение событий с типом Печать

```
curl -X 'POST' \
  'https://server.company.ru/xapi/push/event' \
   'X-API-Auth-Token: brse99dugp1cdbr69axz'
   'X-API-CompanyId: iw'
   'X-API-Version: 1.8'
  -F 'event=
{
  "evt_class": 2,
  "evt_service": "print",
  "evt_attributes": [
    {"name": "capture_ts", "value": "2024-03-06T08:45:00+03:00"},
    {"name": "capture_server_ip", "value": "127.0.0.1"},
    {"name": "capture_server_fqdn", "value": "captureservername.domain"},
    {"name": "print_copies", "value": "3"}
  ],
  "evt_senders": [
    {
      "identity_id": 3,
      "identity_type": 0,
      "identity_contacts": [
        {"name": "auth", "value": "userlogin@domain"},
        {"name": "email", "value": "test@example.com"}
      ]
    },
    {
      "identity_id": 4,
      "identity_type": 1,
      "identity_contacts": [
        {"name": "dnshostname", "value": "computername.domain"}
      ],
      "identity_attributes": [
        {"name": "workstation_type", "value": "ws_type_computer"}
      ]
    }
  ],
  "evt_receivers": [
    {
      "identity_id": 2,
      "identity_type": 2,
      "identity_contacts": [],
      "identity_attributes": [
        {"name": "device_name", "value": "PrinterName"},
        {"name": "print_port_name", "value": "Some print port name"},
        {"name": "print_server", "value": "Printer Xerox"},
        {"name": "print_location", "value": "Kitchen"}
      ]
    }
  ],
  "evt_data": [
```

```
    {
      "data_id": 1,
      "data_attributes": [
        {"name": "print_copies", "value": "3"},
        {"name": "print_job_name", "value": "Print job name"}
      ]
    }
  ]
}' \
  -F 'files[]='
```

Ответ

```
{
  "data": {
    "document_id": "string"
  }
}
```

Запрос на получение событий с типом Web-сообщение

```
curl -X 'POST' \
'https://109.73.44.69/xapi/push/event' \
-H 'accept: application/json' \
-H 'X-API-Auth-Token: 198igrnlnuonix672uq9' \
-H 'X-API-CompanyId: Yandex' \
-H 'X-API-Version: 1.8' \
-H 'Content-Type: multipart/form-data' \
-H 'X-Enable-OpenAPI: 1' \
-F 'event=
{
  "evt_class": 3,
  "evt_service": "web_common",
  "evt_attributes": [
      { "name": "event_name", "value": "Web Event Name" },
      { "name": "capture_ts", "value": "2024-03-25T23:41:21+03:00" },
      { "name": "capture_server_ip", "value": "127.0.0.1" },
      { "name": "capture_server_fqdn", "value": "captureservername.domain" }
  ],
  "evt_senders": [
      {
          "identity_id": 3,
          "identity_type": 0,
          "identity_contacts": [],
          "identity_contacts_with_meta": [
              {
              "contact": { "name": "auth", "value": "userlogin@domain" }
```

```
                }
            ]
        },
        {
            "identity_id": 4,
            "identity_type": 1,
            "identity_contacts": [],
            "identity_attributes": [
                { "name": "workstation_type", "value": "ws_type_computer" }
            ],
            "identity_contacts_with_meta": [
                {
                "contact": { "name": "dnshostname", "value": "computername.domain" }
                }
            ]
        }
    ],
    "evt_receivers": [
        {
            "identity_id": 2,
            "identity_type": 3,
            "identity_contacts": [],
            "identity_attributes": [
                { "name": "res_destination_host", "value": "testhost.com" },
                { "name": "res_destination_url", "value": "\/file\/test.php" }
            ]
        }
    ],
    "evt_data": [
        {
            "data_id": 1
        }
    ]
}
```

Значение поля `"evt_class" 3` соответствует константе kWeb. В месте с ним необходимо передать POST-запрос, например, в формате multipart.

Пример запроса для отправляемых текстовых событий

```
POST /file/test.php HTTP/1.1
Content-Type: multipart/form-data; boundary="----1234567890----"
Content-Length: 110
Host: testhost.com

------1234567890----
Content-Type: text/plain
Content-Disposition: inline

Hello World
```

```
------1234567890------
```

Пример запроса для отправляемых файлов

```
POST /file/test.php HTTP/1.1
Content-Type: multipart/form-data; boundary="----1234567890----"
Content-Length: 156
Host: testhost.com

------1234567890----
Content-Type: text/plain
Content-Disposition: attachment; filename="file.txt"

Hello World.
This is text file.

------1234567890------
```

Рекомендации по производительности API при отправке событий в Traffic Monitor:

- Для улучшения производительности NGINX:
  a. В файле `/etc/nginx/nginx.conf` измените значение параметра `worker_processes` на `2`.
  b. Сохраните изменения и перезагрузите сервис с помощью команды: `systemctl restart nginx`
- При высокой нагрузке в журнале ошибок NGINX `/var/log/nginx/error.log` могут возникать сообщения вида:

  `socket() failed (24: Too many open files) while connecting to upstream.`

  В этом случае требуется увеличить ограничение сервиса на количество открытых файлов, например, до 10000:

  a. Войдите в режим редактирования службы NGINX с помощью команды: `systemctl edit nginx.service`
  b. Вставьте следующие строки в поле редактора:
  `[Service]`
  `LimitNOFILE=10000`
  c. Сохраните изменения (**Ctrl**+**O**) и выйдите из режима редактирования (**Ctrl**+**X**).
  d. Перезагрузите сервис с помощью команды: `systemctl restart nginx`