ООО «АСТРАЛ-СОФТ»

ПП «Астрал.Платформа»

Руководство пользователя

Калуга, 2019 г.

СОДЕРЖАНИЕ

© ООО «АСТРАЛ-СОФТ», 2021

Введение

Для использования API Астрал Платформы предварительно необходимо настроить клиентскую часть защищенного канала передачи данных с использованием протокола TLS и зарегистрироваться в сервисе Астрал Платформы в качестве партнера. Регистрация осуществляется через менеджера ООО «АСТРАЛ-СОФТ». После регистрации партнеру присваивается уникальный токен, который при вызове методов обязательно передается в HTTP-заголовке “X-API-Key”.

© ООО «АСТРАЛ-СОФТ», 2021

1. Концепция работы

1.1. Концепция работы с событиями

Астрал.Платформа построена по принципу асинхронной ESB-системы, а взаимодействие осуществляется с помощью обмена событиями.

Участник системы — это субъект взаимодействия (партнерская система, сервис компании или другой поставщик услуг), интегрированный в систему и обладающий уникальным в рамках системы идентификатором.

Событие — это набор данных установленного формата, передаваемый системой между участниками и, возможно, связанный с другими событиями. События также могут содержать в себе идентификаторы пакетов и файлов.

Код события — это уникальный набор символов, идентифицирующий тип данных и способ их обработки участниками.

Отправитель (Producer) - идентификатор участника системы, который отправил новое событие. Генерируется автоматически при отправке, на основании идентифицирующего заголовка X-API-Key.

Получатель (Consumer) - идентификатор участника системы, который должен получить данное событие. Должен быть указан отправителем. Может быть указано несколько получателей для одного события, в этом случае его получат все участники, перечисленные в списке.

Событие может иметь ссылку на **родительское событие (Parent)**. Это требуется для выстраивания цепочки этапов процесса. Например, исходное событие запроса на проверку подписи файла не имеет родительского, но будет является родительским для дочернего события результата проверки.

В широком смысле, Астрал.Платформа — это особый менеджер для работы с очередями, реализующий **паттерн pub/sub**. Это определяет логику использования API. События, аналогично задачам в очереди, могут быть опубликованы их производителем по одному или группой. А потребитель создает одну или несколько подписок для получения списка новых задач.

Платформа предоставляет **long-polling API** по работе с событиями. Этот способ обеспечения асинхронного взаимодействия сочетает простоту использования и высокую скорость доставки сообщений в системе. Поскольку каждый новый запрос к API требует накладных расходов на сетевое взаимодействие, для повышения эффективности утилизации ресурсов рекомендуется использовать максимально длительный интервал ожидания, обеспечивающий стабильность соединения, чаще всего в интервале от 60 до 600 сек.

1.2. Концепция работы с файлами

Файл — это именованный набор произвольных данных, принадлежащий одному или нескольким пакетам. Система не накладывает ограничений на тип или имя файла (если это не противоречит логике обработки событий, в которых участвует файл), но ограничивает максимально возможный размер - 128 Мб.

Пакет — это набор из одного или нескольких файлов, связанных между собой некоторой логикой использования. Пользователь системы вправе сам определять логику группировки с помощью вызовов соответствующих функций API.

Связь между пакетом и файлом определяется двумя правилами:

© ООО «АСТРАЛ-СОФТ», 2021

1. Пакет не может существовать без файлов, т.е. в системе не может быть пустых пакетов. При удалении файла, если в пакете не осталось других файлов, пакет также удаляется.
2. Файл может участвовать в нескольких пакетах. При удалении пакета, файл удаляется только в случае, если не осталось других пакетов, где он используется.

Список контроля доступа (ACL) — это набор правил, определяющих соответствие участников системы и допустимых для них операций с файлами и пакетами. Каждый файл и пакет обладают этим списком, что делает возможным использование механизмов наследования и переопределения.

Как следствие из этих правил, **Идентификатор файла** в системе состоит из **пары**: собственного идентификатора (UUID) пакета и собственного идентификатора (UUID) файла, разделенных косой чертой (/). Таким образом, указывая полный идентификатор файла, также появляется возможность определить, в контексте списка контроля доступа какого пакета его обрабатывать.

1.3. Настройка клиента TLS 1.0 с поддержкой ГОСТ на базе КриптоПРО CSP 4.0

Основная идея процесса и план действий:

Взаимодействие ПО Партнера и API Платформы осуществляется прозрачно через защищенный туннель

Для организации туннеля используется компонент Stunnel из состава дистрибутива КриптоПРО CSP 4.0

Stunnel (Клиент TLS) является прокси-сервером, принимает запросы на локальный порт и перенаправляет на внешний Сервер TLS. Таким образом, ПО Партнера должно использовать обычный HTTP и обращаться на локальный порт Клиента TLS, в то время как канал передачи данных будет надежно защищен.

Общая схема работы *(рис. 1.3.1.)*:

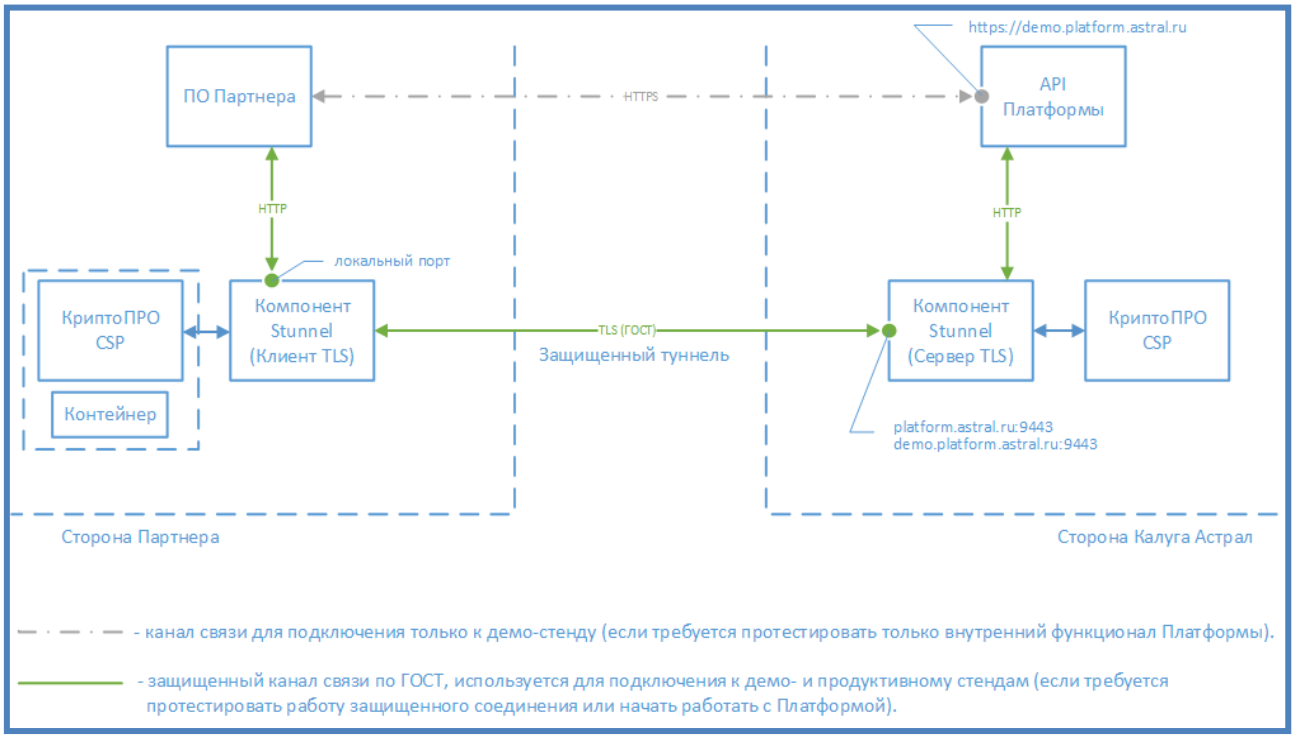


Рис. 1.3.1.

© ООО «АСТРАЛ-СОФТ», 2021

Общий план действий (порядок действий сохранен):

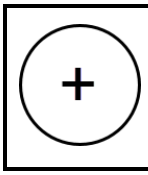
1. Установка КриптоПРО CSP
2. Установка компонента Stunnel (Клиент TLS)
3. Установка сертификата с закрытым ключом (Контейнер) в локальное хранилище
4. Настройка компонента Stunnel в качестве Клиента TLS
5. Настройка подключения к Серверу TLS

© ООО «АСТРАЛ-СОФТ», 2021

2. Требования к ресурсам

2.1. Принятые допущения:

- Клиент TLS настраивается на удаленной машине. Если это не так - те же инструкции актуальны и для локальных машин, нужно лишь пропускать команды SSH
- при подключении по SSH используется аутентификация по ключу
- вместо параметров подключения по SSH, пользователь (логин) и сервер (адрес) заменены на строку *user@host*



Криптоопераций, необходимые для поддержания TLS, потребуют вычислительных ресурсов, прежде всего CPU. В зависимости от нагрузки, может потребоваться 1-2 свободных ядра.

Доступ по SSH к серверу на базе Ubuntu 18.04.01 LTS

- Настройка на Ubuntu 16.04 может незначительно отличаться
- При настройке на других ОС адаптируйте шаги, сохраняя их суть и последовательность.

Доменное имя и порт сервера TLS

- Для продуктивного стенда Астрал.Платформы - *platform.astral.ru:9443*
- Для демо-стенда к Астрал.Платформе - *demo.platform.astral.ru:9443*

Контейнер (с сертификатом и закрытым ключом) сертификата ГОСТ Р 34.10/11-2012
Дистрибутив КриптоПРО CSP 4.0 и лицензия к нему

2.2. Установка КриптоПРО CSP

Предполагается, что есть архив с расширением *tar.gz* и он уже находится на сервере.

Распаковываем архив:

> tar xvf ./cpro4.tar.gz

- Установка lsb-core. Основная зависимость для CSP

> sudo apt update

> sudo apt install lsb-core

- Установка КриптоПРО

> cd /tmp/cpro-4.0

> chmod +x ./install.sh

> sudo ./install.sh

> sudo dpkg -i ./lsb-cprocsp-devel_4.0.0-4_all.deb

В процессе настройки может возникнуть ошибка *Error: Unknown non-lsb linux*. Это возможно, если команда */usr/bin/lsb_release* выдает сообщение вида *LSB Version: core-9.20170808ubuntu1-noarch:security-9.20170808ubuntu1-noarch.* На корректно настроенной машине результат обычно выглядит как *LSB Version: 1.4*. Данная ошибка возникает в процессе установки пакетов при выполнении скрипта */etc/init.d/cprocsp*, который должен перезапускать сервисы КриптоПРО для обновления. В целом - нам это не важно, т.к. это

© ООО «АСТРАЛ-СОФТ», 2021

влияет на корректную работу сервиса в исполнении КС2. Для КС1 сервис не нужен, поэтому можно не обращать внимания. Если нужен КС2, то нужно редактировать файл *sudo nano /etc/init.d/cprocsp*, и на строке 35 изменить выражение *uname -a | grep -i astra > /dev/null* на выражение *uname -a | grep -i -E 'astra|ubuntu' > /dev/null*. Для проверки - вызвать сначала *sudo systemctl daemon-reload*, затем *sudo service cprocsp restart*, а затем *sudo service cprocsp status*. В случае правильного выполнения команд там будет написано *Active: active (running).*

Если на машине установлена другая ОС (не 18.04), в зависимости от дистрибутива могут потребоваться различные пакеты для сборки. В случае, если система выдаст ошибку вида gcc not found, рекомендуется установить пакет *sudo apt install build-essential*, он должен входить в состав *lsb-core*.

2.3. Установка компонента Stunnel

> cd /tmp/cpro-4.0

> sudo dpkg -i ./cprocsp-stunnel-64_4.0.0-4_amd64.deb

В случае успешного выполнения всех настроек, должен появиться файл сервиса */opt/cprocsp/sbin/amd64/stunnel_thread* и файл конфига */etc/opt/cprocsp/stunnel.conf*.

2.4. Установка сертификата с закрытым ключом в локальное хранилище

Контейнер (сертификатом и закрытым ключом) представляет из себя архив, с названием вида *testupla.000.zip*. Внутри архива должен быть каталог с файлами *header.key, masks.key* и другими.

- Предполагается, что на сервере уже есть архив (контейнер) с расширением zip.
  Распаковываем архив и копируем файлы ключей в хранилище

Для распаковки понадобится утилита *unzip*. Её обычно нет по-умолчанию, поэтому можно поставить с помощью *sudo apt install unzip*. Если по каким-то причинам нет возможности её установить, можно распаковать архив на локальной машине, а после скопировать файлы на сервер, например с помощью *rsync -r*.

В следующих командах следует КРАЙНЕ ВНИМАТЕЛЬНО отнестись к написанию путей, символов слэша/звездочки на концах, и, верно, подставить все значения. Рекомендуется соблюдать неукоснительно, даже если shell по нажатию Tab дописывает иначе. Несоблюдение синтаксиса приведет к некорректным настройкам в файлах и правах, вплоть до потери возможности просмотра файлов в каталоге.

В путях файлов фигурирует имя пользователя. В данном примере, для визуальной идентификации, используется имя *astral*. При воспроизведении инструкции необходимо подставить нужное имя.

© ООО «АСТРАЛ-СОФТ», 2021

Эти команды следует выполнять от имени того пользователя, в хранилище которого будет установлен сертификат. Чтобы переключиться на другого пользователя, можно использовать, например, *sudo su astr*al.

> unzip ./testupla.000.zip
> mkdir -p /var/opt/cprocsp/keys/astral
> cp -R ./testupla.000 /var/opt/cprocsp/keys/astral/
> chmod 600 /var/opt/cprocsp/keys/astral/testupla.000/*

- Установка сертификата

Имя контейнера можно увидеть в выводе команды, оно начинается с префикса \\.*HDIMAGE*\, а дальше требуется выбрать нужное имя в списке. Оно либо частично совпадает с именем архива ключей, либо как-то иначе визуально идентифицируется (например, по доменному имени). Если это первая установка, то других ключей нет, и имя в списке будет единственным.

> /opt/cprocsp/bin/amd64/csptest -keyset -enum_cont -verifycontext -fqcn
> /opt/cprocsp/bin/amd64/certmgr -inst -cont '\\.\HDIMAGE\test.platform.astral'
> /opt/cprocsp/bin/amd64/certmgr --list

Последняя команда позволяет проверить, что сертификат корректно установлен. Важно убедиться, что есть строчка *PrivateKey Link: Yes*. Кроме того, нужно записать значение поля *SHA1 Hash*, оно пригодится позже для экспорта.

- Снятие пароля на закрытый ключ

Закрытый ключ не может быть зашифрован, так как *stunnel* не имеет возможности получить пароль от пользователя.

Чтобы проверить наличие пароля, надо выполнить команду ниже. Если в выводе будет строчка *No password needed*, то снятие пароля не требуется. Если выдается сообщение Passwd *needed*, то нужно убрать пароль, действуя дальше, а этой командой потом проверить результат.

> /opt/cprocsp/bin/amd64/csptest -passwd -pininfo –container '\\.\HDIMAGE\test.platform.astral'

Чтобы избавиться от пароля, нужно его сбросить в значение '' (пустая строка). Однако, для этого потребуется ввести действующий пароль от закрытого ключа.

> /opt/cprocsp/bin/amd64/csptest -passwd -change '' -container '\\.\HDIMAGE\test.platform.astral'

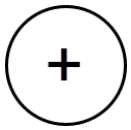
- Экспорт сертификата для использования в Stunnel

на предыдущих шагах был получен SHA1 - отпечаток установленного сертификата. Далее в параметр *thumbprint* нужно подставить его значение без префикса *0x*, который обозначает шестнадцатеричную запись. Т.е. если в выводе команды */opt/cprocsp/bin/amd64/certmgr –list* написано *SHA1 Hash: 0x9d261483b27718c8f9179300010d1c4b85a36b25*, то в параметр надо подставлять *9d261483b27718c8f9179300010d1c4b85a36b25*.

> sudo mkdir /stunnel
> sudo chown astral:astral /stunnel
> /opt/cprocsp/bin/amd64/certmgr -export -cert -thumbprint 9d261483b27718c8f9179300010d1c4b85a36b25 -dest /stunnel/test.platform.astral.ru.cer

- Установка цепочки корневых сертификатов

© ООО «АСТРАЛ-СОФТ», 2021



Всю цепочку корневых сертификатов всегда можно запросить в УЦ, который выдал ваш сертификат. В самом сертификате также должна быть ссылка на его корневой. Корневой сертификат можно скачать и установить командами ниже (пример для тестового УЦ АО "Калуга Астрал").

Важно, чтобы корневой сертификат устанавливался именно в корневое хранилище машины, т.е. *mroot.*

В хранилище *mroot* можно устанавливать только доверенные корневые сертификаты, либо сертификаты, происхождение которых не вызывает сомнений (например, сертификаты тестовых УЦ).

> cd /tmp

> curl -L -O http://regservice.keydisk.ru/testca/root/testca3.crt

> sudo /opt/cprocsp/bin/amd64/certmgr -inst -all -store mroot -file ./testca3.crt

Настройка stunnel в качестве клиента

Настройка сервиса достаточно проста и заключается в правильном редактировании двух файлов, примеры которых приведены далее. Следует помнить, что это лишь примеры.

Для двусторонней аутентификации требуется передать свой клиентский сертификат (тот, что экспортировали) вашему менеджеру Калуга Астрал, для установки сертификата на сервере. Сертификат желательно передавать вместе с корневыми сертификатами Удостоверяющего Центра, выпустившего сертификат.

Описание всех доступных опций *Stunnel* можно прочитать командой *man /opt/cprocsp/share/man/man8/stunnel.ru.8*

- Пример файла конфигурации клиента:

> nano /stunnel/client.conf

```
; Файл сертификата, который экспортировали ранее
; Ключ КриптоПРО возьмет из хранилища пользователя
cert = /stunnel/client.cer

; Параметры рабочего окружения, для изоляции stunnel
setuid = user
setgid = user
pid = /stunnel/stunnel.pid

; Особые настройки сокетов, умолчания лучше не менять без необходимости
socket = l:TCP_NODELAY=1
socket = r:TCP_NODELAY=1

; Отладочный вывод. Для комфортной работы достаточно 3 или даже меньше
debug = 5
output = /stunnel/stunnel.log

; Включение режима клиента
client = yes
foreground = yes
```


© ООО «АСТРАЛ-СОФТ», 2021

[https]
; Какой порт слушать
; 127.0.0.1:8000 // только локальный
; 10.0.1.16:8000 // только определенный интерфейс
; 0.0.0.0:8000 // любой интерфейс
accept = 127.0.0.1:8000

; На какой внешний сервис проксировать
connect = platform.astral.ru:9443

; Если 0, то не проверяет сертификат сервера
; Если 1 или 2, то проверяет цепочку
; Если 3, то проверяет по локальному списку
verify = 3
TIMEOUTclose = 0

- Пример файла сервиса для запуска через *systemd*

> sudo nano /etc/systemd/system/tls_client.service
[Unit]
Description=Stunnel GOST TLS Client

[Service]
PIDFile=/stunnel/stunnel.pid
ExecStart=/opt/cprocsp/sbin/amd64/stunnel_thread /stunnel/client.conf
Restart=always
User=root
LimitNOFILE=65535

[Install]
WantedBy=multi-user.target

- Команды для запуска сервиса

> sudo systemctl daemon-reload
> sudo systemctl start tls_client
> sudo systemctl status tls_client

Настройка подключения к серверу

На предыдущем шаге в файле */stunnel/client.conf* мы написали конфигурацию подключения. Важно проверить, что в значении *connect* написан верный адрес и порт сервера:

- Для продуктивного стенда Астрал.Платформы - *platform.astral.ru:9443*
- Для демо-стенда к Астрал.Платформе - *demo.platform.astral.ru:9443*

При подключении к серверу производятся обоюдные проверки сертификатов. Если вы хотите упростить процесс подключения к демо-стенду, хотя это и не рекомендуется делать, можно отключить проверку сертификата сервера. Для этого в конфиге нужно установить значение *verify=0* и перезагрузить сервис.

© ООО «АСТРАЛ-СОФТ», 2021

Чтобы настроить проверку сертификата Сервера (двусторонний TLS), нужно добавить сертификат сервера и его корневые в хранилище:

- Установить серверный сертификат в *mTrustedUsers*

> /opt/cprocsp/bin/amd64/certmgr -inst -all -store mTrustedUsers -file ./demo.platform.astral.ru.cer

- Скачать и установить корневой серверного сертификата в *mroot*

> curl -L -O http://regservice.keydisk.ru/testca/root/testca3.crt

> sudo /opt/cprocsp/bin/amd64/certmgr -inst -all -store mroot -file ./testca3.crt

- Чтобы убедиться, что всё прошло успешно, можно выполнить проверку соединения командой *curl*, например *curl -X GET "http://127.0.0.1:8000/events"*. Если TLS соединение установлено, то в ответ придет JSON с текстом *"Не указано значение заголовка X-API-Key"*. При этом ошибки соединения будут описаны в логе клиента */stunnel/stunnel*.log, а также подробнее в логах сервера. Команды на отслеживание записей:

> sudo tail -f /var/log/syslog

> tail -F /stunnel/stunnel.log

Если возникает ошибка 0x800b0109 (CERT_E_UNTRUSTEDROOT) с текстом Цепочка сертификатов обработана правильно, но обработка прервана на корневом сертификате, у которого отсутствует отношение доверия с поставщиком доверия, то значит корневой либо не установлен, либо установлен в uroot. Нужно установить его в mroot.

© ООО «АСТРАЛ-СОФТ», 2021

3. Методы API

3.1. POST /events

Метод публикации событий для сервисов Астрал Платформы.
Опубликовать можно одно или несколько событий одновременно. У них могут быть различные коды, получатели и родительские события. При публикации списка, каждое событие публикуется индивидуально, поэтому, в случае ошибки формирования одного из событий, другие всё равно будут опубликованы, и система продолжит работу. Система гарантирует, что события будут поступать получателю в порядке их публикации, но не гарантирует, что получатель обработает их в этом порядке.

Входным параметром является объект JSON, общая структура которого имеет вид:

```
{
   "events": [  список публикуемых событий
      {
         "code":  string – код типа события
         "parent":  string – необязательный идентификатор родительского события
         "consumer": [string] – список идентификаторов сервисов для обработки события
         "data": { }– параметры для обработки события в сервисе
      }
   ]
}
```

Выходным параметром является объект JSON, содержащий следующую структуру:

```
{
   "status": integer {100-600} – статус обработки запроса
   "type": string uri – ссылка на документацию по ошибки, about:blank, если ошибки нет
   "title": string – краткое описание результата запроса
   "events": [
      {
         "id": string string [a-f0-9]{32} – идентификатор события
         "code": string – код публикуемого события
         "producer": string – идентификатор отправителя события
         "consumer": [string] – список сервисов для обработки события
         "data": {  } – параметры для обработки события в сервисе
      }
   ]
}
```

3.2. GET /events

Метод получения входящих событий от других участников системы, например от сервисов Астрал.Платформы.

Входными параметрами являются:

- *integer* ***limit*** - максимальное количество новых событий, которое процесс готов получить

- *integer* ***timeout*** - время жизни запроса в секундах, от 10 до 600

© ООО «АСТРАЛ-СОФТ», 2021

- *boolean* ***unconfirmed*** **-** признак получения событий, факт получения которых не был подтвержден получателем

Поскольку этот метод работает по принципу **long poll**, его нужно использовать в “вечном” цикле, каждый раз открывая новый запрос сразу же после завершения предыдущего. Время ожидания **timeout** следует указывать настолько большим, насколько возможно - начать с 600 секунд и, при возникновении сетевых проблем, постепенно снижать. В минимальном и самом простом варианте, при наличии одного процесса обработки входящих событий, в каждый конкретный момент времени достаточно только одного ожидающего запроса. Он будет получать все входящие события.

При необходимости **горизонтально масштабировать** обработчики по нескольким процессам или машинам, можно запустить параллельно несколько запросов. При этом рекомендуется указывать параметр **limit** с небольшим значением, чтобы события равномерно распределялись по процессам. В данном случае значение этого параметра является весовым коэффициентом нагрузки на обработчик.

Например, пусть у нас есть 3 машины, две по 2 ядра и одна 4 ядра. Тогда на первых машинах запустим запросы со значением limit=2, а на более мощной со значением limit=4. В среднем, при равномерном поступлении новых событий, последняя машина будет получать столько же, сколько и две другие вместе.

При успешной обработке события, его получение нужно подтвердить с помощью метода POST /events/confirm. Если этого не сделать, еще не подтвержденные события будут скапливаться в отдельной очереди для переобработки. Получить события из этой очереди можно, указав значение аргумента **unconfirmed**=true. Эта возможность - один из инструментов обеспечения **гарантии доставки** и надежности системы. Если принимающая сторона по какой-то причине не смогла обработать и подтвердить часть событий, их всегда можно снова получить (неограниченное кол-во раз, вплоть до подтверждения). При этом ошибки не помешают обработке основного потока событий, не допустив деградации системы для пользователя. Для этого режима работы параметр **limit** также имеет значение, а вот аргумент **timeout** игнорируется.

3.3. GET /events/{event_uuid}

Метод получения конкретного события по идентификатору. Может быть полезен для уточнения данных события, повторной обработки, проверки корректности публикации. Текущий статус события не влияют на работу метода: данные можно получить в любой момент, как до его обработки, так и после подтверждения.

Входным параметром является идентификатор события - *string [a-f0-9]{32}* ***event_uuid***

Выходным параметром по получению результата является объект JSON, содержащий следующую структуру:

```
{
   "status": integer{100-600} – статус обработки запроса
   "type": string uri – ссылка на документацию по ошибки, about:blank, если ошибки нет
   "title": string – краткое описание результата запроса
   "events": [
     {
        "id": string [a-f0-9]{32} – идентификатор события
        "code": string – код публикуемого события
```


© ООО «АСТРАЛ-СОФТ», 2021

"producer": *string [a-f0-9]{32}* - идентификатор отправителя события
"parent": *string [a-f0-9]{32}* – идентификатор родительского обработанного события

"consumer": [*string*] – список сервисов для обработки события
"data": { } – результат выполненного события
}
]
}

3.4. POST /events/confirm

Метод подтверждения получения событий.

Отправка идентификатора события в этот метод сигнализирует системе о том, что событие успешно получено и обработано, получатель несет полную ответственность за дальнейшую судьбу данных, а событие пропадает из списка GET /events?unconfirmed=true. Тем не менее, при необходимости, можно получить данные события по идентификатору GET /events/{event_uuid}.

Входным параметром является объект JSON, который имеет следующую структуру:

{
"events_id": [*array* - список идентификаторов событий
string [a-f0-9]{32}
]
}

Выходным параметром по получению результата является объект JSON, содержащий следующую структуру:

{
"status": *integer {100-600}* – статус обработки запроса
"type": *string uri* – ссылка на документацию по ошибки, about:blank, если ошибки нет
"title": *string* – краткое описание результата запроса
"count": *integer* - количество подтвержденных событий
}

3.5. POST /packet

Метод для загрузки файлов в сервис Астрал.Платформа.

Входным параметром является форма с одним или несколькими файлами. Все загруженные файлы автоматически объединяются в пакет. Порядок файлов в пакете определяется порядком в загружаемой форме. Форма должна быть сформирована как **multipart/form-data**. Имя поля в форме (параметр name) не имеет рекомендуемого значения - обработаны будут все файловые поля. Несколько файлов можно загружать как с разными именами полей (например: file1, file2, ...), так и с одинаковыми (file, file, …). Тем не менее, в случае **с разными именами** полей, **порядок** их перебора **не определен!** Поэтому в случае, когда в рамках использования пакета необходимо сохранить порядок, **загружать файлы следует с одинаковым именем поля**.

Выходным параметром является объект JSON, содержащий следующую структуру:
{

© ООО «АСТРАЛ-СОФТ», 2021

"status": *integer{100-600}* – статус обработки запроса
"type": *string uri* – ссылка на документацию по ошибки, about:blank, если ошибки нет
"title": *string* – краткое описание результата запроса
"detail": *string* – развернутое описание результата запроса
"packet": {
"id": *string [a-f0-9]{32}* – идентификатор пакета
"created": *string [0-9]{4})-([0-9]{2})-([0-9]{2}* – дата создания пакета
"updated": *string [0-9]{4})-([0-9]{2})-([0-9]{2}* – дата обновления пакета
"files": [
{
"id": *string [a-f0-9]{32}* – уникальный идентификатор файла в пакете
"name": *string* – имя файла в пакете
"size": *integer* – размер файла в пакете (в байтах)
"mime": *string* – тип файла в пакете
"created": *string [0-9]{4})-([0-9]{2})-([0-9]{2}* – дата создания файла в пакете
"updated": *string [0-9]{4})-([0-9]{2})-([0-9]{2}* – дата обновления файла в пакете
"acl": {
"owner": *string [a-f0-9]{32}* – владелец файла в пакете
"access": { } – карта доступов к использованию файла
}
}
]
"acl": {
"owner": *string [a-f0-9]{32}* – владелец пакета с файлами
"access": { } – карта доступа к использованию пакета с файлами
}
}
}

3.6. GET /packet/{packet_uuid}.tar.gz

Метод для скачивания пакета.
Входным параметром является ***packet_uuid*** *string [a-f0-9]{32}* - идентификатор пакета
Результатом выполнения запроса является получение *gzip* архива.

Имена файлов в архиве совпадают с собственными идентификаторами файлов (UUID), без расширений. Это сделано для того, чтобы сделать разбор пакета по метаданным более простым и надежным. Формат `tar` позволяет передавать вместе с файлами особые заголовки (PAX extended header records). Эта особенность используется Платформой для передачи оригинальных имен файлов. Имя записано в заголовке `ASTRAL.platform.file.name`. Это позволяет избежать дополнительных запросов к API в ряде случаев, когда в процессе обработки пакета не нужны никакие метаданные файлов, кроме идентификаторов и оригинальных имен.

3.7. GET /packet/{packet_uuid}.json

Метод получения метаданных по пакету.
Входным параметром является ***packet_uuid*** *string [a-f0-9]{32}* - идентификатор пакета
Выходным параметром является объект JSON, содержащий следующую структуру:
{
"status": *integer{100-600}* – статус обработки запроса
"type": *string uri* – ссылка на документацию по ошибки, about:blank, если ошибки нет

© ООО «АСТРАЛ-СОФТ», 2021

"title": *string* – краткое описание результата запроса
"detail": *string* – развернутое описание результата запроса
"packet": { метаданные по сформированному пакету
"id": *string* – уникальный идентификатор пакета
"created": *string date-time* – дата создания пакета
"updated": *string date-time* – дата обновления пакета
"files": [массив файлов в пакете
{
"id": *string* – уникальный идентификатор файла в пакете
"name": *string* – имя файла в пакете
"size": *integer* – размер файла в пакете
"mime": *string* – тип файла в пакете
"created": *string date-time* – дата создания файла в пакете
"updated": *string date-time* – дата обновления файла в пакете
"acl": { список контроля доступа на модификацию и чтение файла в пакете
"owner": *string* – владелец файла в пакете
"access": { } – карта доступов к использованию файла
}
}
],
"acl": {
"owner": *string* – владелец пакета с файлами
"access": { } – карта доступа к использованию пакета с файлами
}
}
}

3.8. GET /packet/{packet_uuid}/{file_uuid}

Метод скачивания файла из пакета. Файл передается без изменений, в несжатом виде

Входными параметрами являются:

packet_uuid *string [a-f0-9]{32}* - идентификатор пакета
file_uuid *string[a-f0-9]{32}* - идентификатор файла

Результатом выполнения запроса является получение отдельного файла.

3.9. DELETE /packet/{packet_uuid}/{file_uuid}

Метод удаления файла. Если в пакете не остается других файлов, пакет также удаляется.

Входными параметрами являются:

packet_uuid *string [a-f0-9]{32}* - идентификатор пакета
file_uuid *string[a-f0-9]{32}* - идентификатор файла

Результатом выполнения запроса является удаление файла из пакета.

3.10. GET /packet/{packet_uuid}/{file_uuid}.json

Метод получения метаданных по файлу.

Входным параметром является:

packet_uuid *string [a-f0-9]{32}* - идентификатор пакета
file_uuid *string[a-f0-9]{32}* - идентификатор файла в пакете

Выходным параметром является объект JSON, содержащий следующую структуру:
{

© ООО «АСТРАЛ-СОФТ», 2021

"status": *integer{100-600}* – статус обработки запроса
"type": *string uri* – ссылка на документацию по ошибки, about:blank, если ошибки нет
"title": *string* – краткое описание результата запроса
"file": {
"id": *string [a-f0-9]{32}* – уникальный идентификатор файла в пакете
"name": *string* – имя файла в пакете
"size": *integer* – размер файла в пакете
"mime": *string* – тип файла в пакете
"storage" *string* – тип хранилища файла
"created": *string date-time* – дата создания файла
"updated": *string date-time* – дата обновления файла
"acl": { список контроля доступа на модификацию и чтение файла в пакете
"owner": *string* – владелец файла в пакете
"access": { } – карта доступов к использованию файла
}
}
}

© ООО «АСТРАЛ-СОФТ», 2021