# Серверная часть

1. Задача серверной части принимать HTTP-запросы и отвечать на них. В процессе работы необходимо взаимодействовать с

СУБД PostgreSQL. СУБД может находиться физически на другой машине, поэтому надо предусмотреть такой вариант развёртывания

системы: может работать несколько серверных приложений на разных физических серверах и одна СУБД на отдельномом физическом сервере.

1.1 База данных хранит информацию о клиентах, файлах для них (команда /5/, пункт 3.3) и событиях связанных с клиентами.

Каждый клиент имеет ряд неотъемлемых свойств: Тег группы, Id клиента, операционная система, ip-адрес, geoip (страна),

Importance, userdefined, когда был зарегистрирован, когда выходил в последний раз в онлайн (команда /0/, пункт 3.1).

Importance и userdefined - это два целых числовых параметра (отрицательные числа, положительные числа и ноль),

устанавливаются или изменяются другими серверными приложениями или администраторм/оператором сервера.

2. Работа с сервером осуществляется через HTTP-запросы.

Они могут быть как GET, так и POST.

Между сервером и клиентом может стоять сколько угодно reverse-proxy, load-balancer и DNAT.

Таким образом, единственный способ узнать ip-адрес клиента который послал запрос - это параметр client-self-ip-address

в команде /0/ (пункт 3.1). Других способов нет, не будет и быть не может в принципе.

2.1 Каждый запрос содержит в URI три компонента разделённых символом "/"

```тег группы, id клиента, код команды. /<group-tag>/<clientid>/<ccode>/,```

где group-tag - тег группы,

clientid - id клиента,

ccode - код команды.

2.1.1 Тег группы - это произвольная строка состоящая из символов (a-z) и цифр (0-9). Параметр не чувствителен к регистру.

2.1.2 Id клиента - это строка состоящая из двух компонентов разделённых точкой.

Первая часть имеет формат <name>\_XYYYYYYY, где name это некоторое имя которое может как-то идентифицировать машину

(имя компьюетра или имя пользователя, в зависимости от типа операционной системы),

X - символ обозначающий тип системы на которой работает клиент (W - windows, L - linux, A - андроид, M - Mac OS),

YYYYYYY - 3-7 цифр содержащих major-version, minor-version и build операционной системы если таковые имеются у системы

(например, длЯ 6.1 build 7600 это будет 617600).

Вторая часть содержит 32 случайных символов 0-9, A-F.

Пример id клиента - ```QWERTY\_W617600.11223344556677889900AABBCCDDEEFF```.

Параметр не чувствителен к регистру.

2.1.3 Код команды - это число от 0 до 999.

2.2 Четрвёртый и последующий компоненты в URI, а также тело запроса и ответа зависят от кода команды

2.3 Кадлый ответ сервера может быть со следующими HTTP-кодами: 200, 403, 404

# 3. Команды

## 3.1 /0/ - инициализация цикла команд /1/.

Принимает 4 дополнительных параметра:

\* system-version - наименование операционной системы c версией, серви паком и архитектурой (32, 64 bit)

\* client-version - версия клиента, произвольное число больше 1000

\* client-self-ip-address - собственный внешний ip-адрес (ipv4 и ipv6) клиента в строковом представлении.

Добывается самим клиентом средствами выбранными по усмотрению разработчика этого клиента.

\* token-string - случайная строка длиной от 16 до 32 символов.

Может содержать большие и маленькие английские символы и цифры от 0 до 9. Чувствителен к регистру

\* devhash - идентификатор оборудования. Произвольная HEX-строка (символы 0-9, A-F) представляющая некоторый 256-битный хеш,

который зависит оборудования или операционной системы на которых работает клиент

Формат запроса:

```/<group-tag>/<clientid>/0/<system-version>/<client-version>/<client-self-ip-address>/<devhash>/<token-string>/```

Ответом сервера является файл (content-type: binary) следующего содержимого:

```/1/<group-tag>/<clientid>/<token-string>/<binary-content-length>/\r\n

<binary-content>\r\n

<server-sign>```

token-string - строка скопированная из GET-запроса

binary-content-length - длина блока binary-content в байтах записанная числов в виде строки

binary-content - блок бинарных данных

server-sign - цифровая подпись сервера, в неё включает содержимое ответа от первого байта до второго перевода строки \r\n (символы \r\n включаются в цифровую подпись)

Если при получении команды клиента нет в базе, то он должен быть добавлен в неё, если он есть, то информация в базе

(тег группы, ip-адрес, версия клиента, операционная система) должна быть обновлена.

Выборка файла.

Блок бинарных данных данном случае это файл extcfg, который выбирается по тем же правилам, что и в команде /5/. Если

Приём команды /0/ означает, что с текущего момента клиент находится в онлайне. Возможно получение несколько раз подряд

команды /0/ от одного клиента - это нормальная ситуация.

## 3.2 /1/ - выдача клиенту команды.

Имеет только один дополнительный параметр -

\* token-string (который будет продублирован в ответе).

Разрешается обрабатывать данную команду только в том случае если в последний раз команда /0/ приходила от данного клиента

не более суток назад (время может быть изменено через конфиг-файл серверного приложения),

в противном случае сервер должен ответить HTTP 403.

Формат ответа на данную команду:

```/<incode>/<group-tag>/<clientid>/<token-string>/<cmd-id>/\r\n

<command-params>\r\n

<server-sign>```

\* incode - код выданной команды

\* token-string - строка скопированная из параметра команды /1/ из GET-запроса

\* cmd-id - уникальный идентификатор выданной команды, произвольная строка с большими и маленькими английскими символами, цифрами от 0 до 9 и символомами -,+,точка,#. Чувствителен к регистру.

\* command-params - произвольная строка с параметрами (не должны содержаться символы \r\n так как это обозначает конец строки)

\* server-sign - цифровая подпись сервера, в неё включается содержимое ответа от первого байта до второго перевода строки \r\n (символы \r\n включаются в цифровую подпись)

## 3.3 /5/ - выдача клиенту файла.

Имеет только один дополнительный параметр - имя файла. Имя файл не чувствительно к регистру.

В ответ на эту команду сервер должен выдать файл (content-type: binary) предназначенный для этого клиента.

Во внутреннем хранилище серверного приложения (базе данных) должно находиться произвольное число файлов,

каждый файл может иметь фильтр по следующим параметрам:

фильтр группы (задаётся списком like-паттернов разделенных пробелами),

исключающий фильтр группы (задаётся списком like-паттернов разделенных пробелами, если есть хотя бы одно совпадание из этого списка, то файл считается не подходящим клиенту),

geoip,

операционная система,

importance (диапазон начало-конец),

userdefined (диапазон начало-конец),

id клиента (строгое соответствие),

приоритет.

Приоритет - это произвольное положительное целое число.

Тег группы может задаваться не только строгим наименованием группы, но и как like-паттерн из языка SQL.

Сервер может располагать несколькими файлами с одним и тем же именем, но с разными фильтрами.

Если клиенту подходит несколько файлов сразу, то должен быть выдан тот, у которого в фильтре значение приоритета больше

чем у остальных. База должна не допускать указание нескольких файлов с одинаковым приоритетом.

Если нет ни одного файла удовлетворяющего по условиям данному клиенту, то сервер должен ответить HTTP 404.

Если пришёл запрос от клиента от которого никогда не приходило запроса /0/, то у него надо проверить только тег группы

(так как другой информации ещё нет в базе).

## 3.4 /10/ - получение от клиента отчёта о выполненной им команды.

Формат запроса для команды /10/:

```/<group-tag>/<clientid>/10/<incode>/<cmd-id>/<result-code>/```

incode - код выполненной команды

cmd-id - идентификатор выполненной команды.

result-code - код результата выполненной команды. Число от 0 до 999

При получении этой команды, команда которая находится в очереди у данного клиента должна быть удалена из очереди

ВНЕ зависимости от того какой result-code прислал клиент.

Если в очереди нет команды с указанным кодом и идеентификаторорм указанных в параметрах incode и cmd-id (обязательна

проверка обеих значений), то это считается нештатной ситуацией должно быть занесено в некоторый журнал или другую

сущность, которую потом может посмотреть администратор сервера.

Ответ на данную команду всегда одинаковый HTTP 200, Content-type: text/plain, содержимое тела ответа "/1/"

## 3.5 /14/ - сохранение пары ключ-значение в БД.

Для возможности просмотра этих значений администратором или оператором.

Формат запроса:

```/<group-tag>/<clientid>/14/<name>/<value>/0/```

\* name - ключ. Произвольная строка с английскими заглавными и строчными буквами, цифрами и символами точка, "+" и "-".

\* value - значение. Произвольная строка, все специальные символы должны быть закодированы в urlencode.

\* Третий параметр всегда ноль.

Ответ на данную команду всегда одинаковый HTTP 200, Content-type: text/plain, содержимое тела ответа "/1/"

## 3.6 /15/ - чтение пары ключ-значение из БД.

```/<group-tag>/<clientid>/15/name```

Результат:

HTTP 200 -- тело: значение

HTTP 204 -- значение не найдено

## 3.7 /23/ - выдача клиенту конфига.

Имеет только один дополнительный параметр - версию текущего конфига клиента. Версия - это число от 0 до 2^32-1.

Пример запроса клиента:

```/<group-tag>/<clientid>/23/<current-config-version>/```

current-config-version - текущая версия конфига.

Пример ответа сервера:

```/23/<group-tag>/<clientid>/<config-version>/<binary-content-length>/\r\n

<binary-content>\r\n

<server-sign>```

binary-content - бинарное содержимое конфига

config-version - версия конфига в binary-content

binary-content-length - длина binary-content в байтах записанная в виде строки

Во внутреннем хранилище серверного приложения (базе данных) должно находиться произвольное число конфигов.

Подобно файлу конфиг имеет неотъемлемый параметр - версию - произвольное числовое значение, которое задаётся администратором или вышестоящей логикой.

Каждый конфиг может иметь фильтр по следующим параметрам:

\* тег группы (может задаваться как like-паттерн из языка SQL),

\* geoip,

\* операционная система,

\* importance (диапазон начало-конец),

\* userdefined (диапазон начало-конец),

\* id клиента (строгое соответствие).

Сервер может располагать несколькими конфигами с одной и той же версией, но с разными фильтрами.

Все конфиги с версией равной или меньшей чем та, которую указал клиент в запросе, отбрасываются.

Клиенту может быть выдан конфиг с версией строго большей чем, та которую он указал в запросе.

Если клиенту подходит несколько конфигов сразу, то должен быть выдан тот, у которого версия больше, чем у остальных.

База должна может допускать хранение конфигов с одинаковой версией.

Если нет ни одного файла удовлетворяющего по условиям данному клиенту, то сервер должен ответить HTTP 404.

Если пришёл запрос от клиента от которого никогда не приходило запроса /0/ (либо он приходил давно), то сервер отвечает сразу HTTP 403, без каких либо проверок.

## 3.8 /25/ - выдача клиенту ссылки.

Команда не имеет параметров.

Пример запроса клиента:

```/<group-tag>/<clientid>/25/<token-string>/```

token-string -идентичен параметру в команде /1/

Пример ответа сервера:

```/25/<group-tag>/<clientid>/<token-string>/\r\n

<link>\r\n

<server-sign>```

Во внутреннем хранилище серверного приложения (базе данных) должно находиться произвольное число ссылок.

Ссылка - это произвольная строка построенная по правилам URL.

Подобно файлам и конфигам, ссылка имеет неотъемлемый параметр - срок действия в минутах, с момента добавления администратором или вышестоящей логикой.

Каждая ссылка может иметь фильтр по следующим параметрам:

\* тег группы (может задаваться как like-паттерн из языка SQL),

\* geoip,

\* операционная система,

\* importance (диапазон начало-конец),

\* userdefined (диапазон начало-конец),

\* id клиента (строгое соответствие).

Сервер может располагать несколькими одинаковыми ссылкам, но с разными фильтрами.

При выборке ссылки для клиента должны быть отброшены всех ссылки, срок действия которых истёк.

Если клиенту подходит несколько ссылок сразу, то должна быть выдана та, актуальность которой истекает позже всех остальных (именно актуальность, т.е. дата-время добавления + срок действия в минутах).

Если нет ни одной ссылки удовлетворяющего по условиям данному клиенту, то сервер должен ответить HTTP 404.

Если пришёл запрос от клиента от которого никогда не приходило запроса /0/ (либо он приходил давно), то сервер отвечает сразу HTTP 403, без каких либо проверок.

# 3.9 /63/ - приём данных модуля.

Команда имеет следующие параметры:

\* имя модуля,

\* ctl к модулю,

\* результирующая строка ctl,

\* вспомогательный тег,

\* ctl\_OutData.

Причём результирующая строка ctl, вспомогательный тег и ctl OutData являются опциональными.

Параметр "ctl\_OutData" является блоком произвольных бинарных данных и передаётся в теле через multipart/form-data, имя параметра внутри multipart/form-data -- "noname".

Если в запросе присутствует ctl\_OutData, то запрос передаётся методом POST.

Пример запроса клиента

```/63/<module name>/<ctl>/<ctl-result-string>/<aux-tag>/```

\* module name - строка состоит только из английских букв, максималная длина 64 символа. Предполагается, что на этом столбце в таблице будет стоять индекс.

\* ctl - строка состоит только из английских букв и цифр, максималная длина 64 символа.

\* ctl-result-string - строка в формате base64. Бекенд должен сохранить эту строку в базе в декодированном виде. Максимальная длина после декодирования 1024 символа

\* aux-tag - вспомогательный тег, строка состоит только из английских букв и цифр, максималная длина 128 символов. Поле необходимо для удобства поиска в таблице, предполагается, что на этом столбце в таблице будет стоять индекс.

\* ctl\_OutData - передаётся в теле POST-запроса и содержит блок произвольных бинарных данных. Максимальный размер 32 МБ.

Ответ сервера либо всегда 200 - "/1/", либо 403 если клиента нет в базе или он слишком давно присылал /0/

Сервер должен хранить всю информацию полученную через команду /63/, а именно:

\* дата время,

\* сlientid,

\* ctl,

\* ctl-result-string,

\* aux-tag

\* ctl\_OutData.

Вся информация должна храниться в отдельных столбцах для возможности выборки и фильтрации (разумеется, кроме ctl\_OutData, для неё достаточно столбца с бинарным типом данных).

## 3.10 /64/ - отчет о событии в модуле.

Команда имеет следующие параметры:

\* имя модуля,

\* имя события,

\* информация о событии,

\* вспомогательный тег,

\* данные события.

Причём информация о событии, вспомогательный тег и данные являются опциональными.

Информация о событии -- это строка в кодировке UTF-8 максимальной длины 64 КБ (байтов, а не символов).

Параметр с данными является блоком произвольных бинарных данных и передаётся в теле через multipart/form-data,

Имя параметра внутри multipart/form-data -- "data", а имя параметра с информацией о событии -- "info".

Если в запросе присутствует данные или информация о событии, то запрос передаётся методом POST.

Пример запроса клиента

```/64/<module name>/<event-name>/<aux-tag>/```

module name - строка состоит только из английских букв, максималная длина 64 символа. Предполагается, что на этом столбце в таблице будет стоять индекс.

event-name - строка состоит только из английских букв и цифр, максималная длина 64 символа.

aux-tag - вспомогательный тег, строка состоит только из английских букв и цифр, максималная длина 128 символов.

Поле необходимо для удобства поиска в таблице, предполагается, что на этом столбце в таблице будет стоять индекс.

Ответ сервера либо всегда 200 - "/1/", либо 403 если клиента нет в базе или он слишком давно присылал /0/

Сервер должен хранить всю информацию полученную через команду /64/, а именно:

\* дата время,

\* сlientid,

\* event name,

\* info,

\* aux-tag

\* data.

Вся информация должна храниться в отдельных столбцах для возможности выборки и фильтрации (разумеется, кроме data, для неё достаточно столбца с бинарным типом данных).

# 4. Каждый клиент должен иметь свою очередь команд.

Команды добавляются в очередь оператором или определёнными механизмами серверного приложения.

Команды удаляются из очереди только тогда, когда клиент присылает отчёт об их выполнении с помощью команды /10/.

Пока он отчёт не пришёл, команда будет стоять в очереди, ДАЖЕ если это приведёт к тому, что клиент запросит несколько

раз одну и ту же команду. При получении отчёта о выполении комада должна быть удалена из очереди ВНЕ зависимости от того

какой result-code прислал клиент.

Каждая команда имеет два атрибута:

\* код команды (число от 0 до 999)

\* параметр команды - ANSI-строка произвольной длины.

# 5. Цифровая подпись сервера

которая фигурирует в ответе некоторых команд является подписью ECDSA 256bit.

В текущей версии не используется, вместо неё должна быть строка "1234567890"

# 6 DataPost

- это отдельная версия серверного приложения которая принимает ТОЛЬКО команду /60/ и больше ничего.

6.1 /60/ - не используется в текущей версии системы

# 7 Функционал IdleCommands.

Сервер должен предоставить некоторое API (в виде хранимых процедур), которое будет создавать "ждущие команды".

Функционал ждущих команд подразумевает, что есть группа команд, которая изначально не имеет клиентов-получателей.

Каждая группа ждущих команд имеет следующие параметры:

\* код команды,

\* параметр команды,

\* количество команд,

\* фильтры geoip (до 10 стран)

\* фильтр операционной системы

\* фильтр группы. Фильтр может задаваться как like-паттерн из языка SQL, либо несколько like-паттернов разделенных пробелами

\* исключающий фильтр групп. Фильтр может задаваться как like-паттерн из языка SQL, либо несколько like-паттернов разделенных пробелами. Если тег группы клиента подходит хотя бы под один перечисленный в этом параметре паттерн, то ждущая команда считается как НЕ подходящая для него. Непротиворечивость этого параметра с предыдущим параметром обеспечивается администратором.

\* фильтр importance (начало и конец диапазона)

\* фильтр user defined (начало и конец диапазона).

Каждый приходящий запрос /1/ клиента, если у него нет в очереди невыполненных команд, проверяется по фильтрам групп "ждущих команд", при этом должны быть исключены те группы, из которых ранее были выданы команды этому клиенту.

Если такие фильтры находятся, то берётся первый фильтр (если таковых нашлось более одного), счётчик команд этой группы уменьшается на единицу, после чего код команды и её параметр добавляется в очередь команд этого клиента.

Важное требование: одному клиенту из каждой группы "ждущих команд" команда может быть выдана только один раз.

Также необходимо предоставить функционал управления группами "ждущих команд". Итоговый функционал (хранимые процедуры):

1. добавить группу

2. удалить группу

3. получить список групп (в списке у каждой группы должны быть все параметры, плюс количество оставшихся команд в группе и изначальное число команд в группе)

4. увеличить количество команд (позаботиться об атомарности операции и разрешении конфликта, когда из-за одновременности операций админа и бэкендов количество стало отрицательным)

5. изменить параметр команды (но не код команды)

6. добавить страну в фильтры (убирать нельзя)

# 8 API управления сервером.

Сервер на отдельном порте предоставляет API для управления некоторыми функциями сервера. Порт API должен настраиваться через конфиг сервера.

8.1 Доступ по API происходит

по протоколу HTTPS, вне зависимости от номера порта сервера. HTTP-запрос может быть как GET, так и POST. POST-запрос используется если один из аргументов API представлен в виде бинарного блока данных. Контроль доступа осущестсвляется по двум параметарам: apikey и apikeypass.

8.1.1 Формат запроса по API выглядит так:

```/<apikey>/<apikeypass>/<function>?param\_name1=<param1>&param\_name2=<param2>.....```

Если функция принимает некоторый бинарный блок данных, то запрос осуществляется методом POST, а блок данных передаётся внутри multipart/form-data с именем "bdata"

function - имя функции, строка чувствительна к регистру, может содержать большие и маленькие английские буквы и цифры

8.2 Каждый apikey имеет следующие атрибуты:

список функций которые ему разрешено выполнять,

диапазон ip-адресов с которых ему разрешён доступ.

Диапазон задаётся в формате CIDR. Например, 1.2.3.4/24 - диапазон от 1.2.3.0 до 1.2.3.255, 1.2.3.4/32 - единичный ip-адрес, 1.2.3.4/0 - любой ip-адрес.

Список всех apikey и apikeypass с их настройками (список функций и диапазон ip) должен задаваться через специальную таблицу базе данных

8.2.1 IP-адрес инициатора запроса извлекается из HTTP-заголовка "X-Forwarded-For". Если этого заголовка нет, то запрос должен быть отброшен с кодом HTTP 403.

## 8.3 Функции API

### 8.3.1 GetGroupData - получение информации по группам.

Принимает один параметр: период времени в минутах.

Функция должна выдавать отчёт об активности по группам за указанный период времени до настоящего момента.

Данные о клиенте включаются в отчёт если у него была активность в указанный период времени.

Отчёт по группам представляет из себя текстовый файл. Формат данных по группам :

```<group> <client\_count> <first\_created>```

\* group - имя группы

\* client\_count - количество уникальных клиентов

\* first\_created - время первой регистрции самого раннего клиента в этой группе. Формате времени: epoch time.

Строки с информаций о группах разделяются парой "\r\n".

Например:

```qwerty1 151 1451278532

test 12 1358237562

test2 1005 1428237531

test111 100 1438257732```

Функция в случае успеха отвечает HTTP 200 c текстовым файлом с данными по группам, в противном случае 403 (если параметры не указаны, неправильный формат, неправильные данные apikey/apikeypass или где-то ошибка).

### 8.3.2 UploadFile - загрузка файла на сервер.

Принимает параметры в следующем порядке:

\* id - число (если адресуется конкретный файл для замены )

\* имя файла (filename)

\* group -- группа. (если не указан то "\*")

\* sys\_ver (если не указан то "\*")

\* country (если не указан то "\*")

\* client\_id (если не указан то 0)

\* importance\_low (если не указан то 0)

\* importance\_high (если не указан то 100)

\* userdefined\_low (если не указан то 0)

\* userdefined\_high (если не указан то 100)

\* priority (число)

\* bdata -- содержимое файла внутри multipart/form-data

Перед загрузкой файла в базу приоритет файла вычисляется следущим образом: вычисляется максимальное значение приоритета в таблице и прибавляется единица.

Функция в случае успеха отвечает HTTP 200 /1/, в противнос случае 403 (если параметры не указаны, неправильный формат, неправильные данные apikey/apikeypass или где-то ошибка).

### 8.3.3 UploadConfig - загрузка конфига на сервер.

\* id -- если нужно заменить линк

\* version версия конфига

\* group группа

\* country (если не указан то "\*")

\* client\_id (если не указан то 0)

\* importance\_low (если не указан то 0)

\* importance\_high (если не указан то 100)

\* userdefined\_low (если не указан то 0)

\* userdefined\_high (если не указан то 100)

\* sys\_ver

\* bdata -- содержимое конфига внутри multipart/form-data

Функция в случае успеха отвечает HTTP 200 /1/, в противнос случае 403 (если параметры не указаны, неправильный формат, неправильные данные apikey/apikeypass или где-то ошибка).

### 8.3.4 UploadLink - загрузка ссылки на сервер.

\* id -- если нужно заменить линк

\* expiry\_at -- unixtime (целое) - время протухания ссылки

\* sys\_ver -- версия системы

\* group -- группа

\* country -- страна

\* client\_id (если не указан то 0)

\* importance\_low (если не указан то 0)

\* importance\_high (если не указан то 100)

\* userdefined\_low (если не указан то 0)

\* userdefined\_high (если не указан то 100)

\* bdata -- полный текст ссылки внутри multipart/form-data

Функция в случае успеха отвечает HTTP 200 /1/, в противнос случае 403 (если параметры не указаны, неправильный формат, неправильные данные apikey/apikeypass или где-то ошибка).

### 8.3.5 PushBack - добавление команды в очередь клиента

\* cid - строковый идентификатор клиента. Часть идентификатора находящаяся после точки. Например, 4D2873436FA2371F319DD55C147DC9B2

\* code - код команды, число

\* param - параметр команды, строка. При передаче через ссылку в GET, она может быть в формате urlencode

Функция в случае успеха отвечает HTTP 200 /1/, если несуществующий cid - 404.

В противном случае 403 (если параметры не указаны, неправильный формат, неправильные данные apikey/apikeypass или где-то ошибка).

### 8.3.6 GetFilesList - получить список всех файлов со всеми параметрами.

Функция в случае успеха отвечает HTTP 200 text/plain с перечислением всех файлов, которые есть в таблице с файлами.

Одна строка - один файл, разделитель строк - "\r\n". Внутристроковый разделитель параметров - табуляция (9).

Параметры следуют в следующем порядке: id, имя, приоритет, группа, geo, importance\_low, importance\_high, system, userdefined\_low, userdefined\_high.

В противном случае 403 (если параметры не указаны, неправильный формат, неправильные данные apikey/apikeypass или где-то ошибка).

### 8.3.7 DeleteFile - удаление файла

\* id - идентификатор удаляемого файла

Функция в случае успеха отвечает HTTP 200 /1/, если несуществующий id - 404. В противном случае 403 (если параметры не указаны, неправильный формат, неправильные данные apikey/apikeypass или где-то ошибка).

### 8.3.8 GetLastEventData - получить последние данные из указанного модуля и события

\* cid - строковый идентификатор клиента. Часть идентификатора находящаяся после точки. Например, 4D2873436FA2371F319DD55C147DC9B2

\* module - имя модуля

\* event - имя события

Функция в случае успеха отвечает HTTP 200 и octet-stream с содержимым данных присланных последний раз по указанному модулю и событию, если несуществующий cid - 404, если нет ни одного события удовлетворящего требованиям (module, event) - 404, если событие есть, но данных нет, то 204. В противном случае 403 (если параметры не указаны, неправильный формат, неправильные данные apikey/apikeypass или где-то ошибка).

### 8.3.9 GetOnline - получить список всех клиентов по последней активности.

\* period - период активности в секундах.

Функция в случае успеха отвечает HTTP 200 text/plain с перечислением строковых идентификаторов клиентов, которые были активны в указанный периол времени.

Одна строка - один идентификатор, разделитель строк - "\r\n". В идентификаторе указывать только часть после точки.

В противном случае 403 (если параметры не указаны, неправильный формат, неправильные данные apikey/apikeypass или где-то ошибка).

### 8.3.10 GetLastActivity - получить время последней активности клиента

\* cid - строковый идентификатор клиента. Часть идентификатора находящаяся после точки. Например, 4D2873436FA2371F319DD55C147DC9B2

Функция в случае успеха отвечает HTTP 200 и text/plain в котором в виде строки находится EPOCH-time с последней активностью указанного клиента, если несуществующий cid - 404. В противном случае 403 (если параметры не указаны, неправильный формат, неправильные данные apikey/apikeypass или где-то ошибка).

### 8.4 По всей активности API должен вестись журнал.

Должна храниться следующая информация: дата-время, apikey, искомый адрес, функция. Хранить информацию только я успешно выполненных запросов, все остальные логировать в общий текстовый лог сервера.

# 9 Лог клиента.

У каждого клиента должен быть свой журнал активности.

В этот журнал активности должна заноситься информация обо всех командах серверу кроме команды /1/.

Также должна заноситься информация обо всех выданных командах клиенту.

Если была выдана команда из групп commands\_idle, то должен быть сохранен идентификатр группы commands\_idle.

В журнале активности должна храниться следующая информация:

\* дата время

\* client\_id

\* код команды, входящей или исходящей.

\* Дополнительная информация в виде строки.

\*\* Для исходящих команд информации не будет.

\*\* При получении команды /10/ в данное поле надо сохранить код выполненной команды.

\*\* При "не холостом" выполнении команды /1/, в это поле должен быть сохранён код выданной команды.

\*\* Для commands\_idle это будет идентификатр группы.

\*\* Для /63/ это будет имя модуля и ctl (через пробел).

\*\* Для /64/ это будет имя модуля и имя события и aux-tag (через пробел).

\*\* Для /23/ это будет версия выданного конфига.

\*\* Для /5/ это будет id выданного файла.

\*\* Для /25/ это будет текст ссылки.

\*\* Для /10/ это будет идентификатор выполненной команды (cmdid) и код результата.

# 10 Importance.

Механизм Importance должен позволять формировать целочисленное значение от 0 до 100 включительно на основе событий связанных с клиентом.

## 10.1 Каждое событие имеет модификаторы значения Importance, их три: preplus, mul, postplus.

Все имеют тип float с точностью как минимум 4 знака после запятой и не могут принимать значение null и .

Preplus и postplus могут принимать значения от -100, до 100 включительно, mul может принимать значения от 0 до 100 включительно.

Также событие имеет класс и дополнительный параметр зависящий от класса.

Может быть сколько угодно событий с одним и тем же классом, но не может быть двух событий у которых класс и дополнительный параметр одинаковые.

У каждого клиента событие может сработать только один раз.

После того, как сработало событие значение importance перевычисляется по следующей формуле:

newimportance = (oldimportance+preplus)\*mul+postplus

Если newimportance получается менее 0 или более 100, то ему присваивается значение соответственно 0 или 100.

Также перед сохранением в базу значение newimportance округлятся до целого значения с использованием стандартной функции округления.

Помимо прочего у каждого клиента должен параметр (флаг), который запрещает автоматическое изменение значения importance.

По-умолчанию, он у всех снят.

Списоб событий глобален для всех клиентов.

ОН должен меняться через СУБД, а не через конфиги.

При удалении события из списка (таблицы) с событиями, никаких пересчётов importance не происходит.

Сам список событий изменяется редко (не чаще чем раз в 30 минут), поэтому допускается кеширование списка событий.

Регулярность обновления должна задаваться через конфиг.

## 10.2 В текущей реализации сервер должен поддерживать следующие классы событий

### 10.2.1 Класс - "online". Команда /0/ .

Дополнительный параметр - количество полученных команд /0/ . Например, указаны следующие события:

id1 - online(1), preplus=xx, mul= xx, postplus=xx

....

id16 - online(4)), preplus=xx, mul= xx, postplus=xx

....

id23 - online(7)), preplus=xx, mul= xx, postplus=xx

Таким образом при первой полученной команде /0/ сработает событие "id1",

при второй полученной команде событие "id1" уже НЕ сработет,

при третьей тоже не сработает,

а при четвёртой полученной команде /0/ сработает событие "id16",

при пятой и шестой командах /0/ события "id1" и "id16" не сработают,

при седьмой полученной команде /0/ сработает событие "id23".

Если параметр равен 0 или null, то он расценивается как единица.

### 10.2.2 Класс - "age". Время после регистрации.

Дополнительный параметр - количество минут прошеднее после самой первой команды /0/, которая послужила причиной регистрации клиента в базе.

Количество минут сравниваниется согласно операции "больше или равно" (">=").

Проверка на событие происходит только при получении команды /0/ или /1/.

Например, имеются следующие события:

id2- age(30), preplus=0, mul= 1.0, postplus=10

....

id15- age(90), preplus=0, mul= 0.2, postplus=0

Таким образом, при полученни команды /0/ или /1/ получилось так, что прошло 37 минут с момента регистрации клиента, то срабатывает событие "id2".

Потом если при получении /0/ или /1/ получилось так, что прошло 123 минуты, то сработает "id15".

В итоге значение importance будет равно 2 (в начале importance равно 0, потом после "id2" будет (0+0)\*1+10 = 10, а потом после "id15" будет (10+0)\*0.2+0=2)

Также возможет другой вариант: в какой-то момент при полученни команды /0/ или /1/ получилось так, что с момента регистрации клиента прошла 201 минута,

то сработают одновременно и событие "id2", и событие "id15". В этом случае мы получим одновременное срабатывание обоих событий, те же вычисления и значение importance = 2.

### 10.2.3 Класс - "geo". Страна.

Дополнительный параметр - наименование страны, строка. Проверка на событие осуществляется только при получении команды /0/.

### 10.2.4 Класс - "devhash\_dup". Имеется другой клиент, у которого devhash точно такой же.

Дополнительный параметр не используется.

Проверка на событие осуществляется только при получении команды /0/. Событие срабатывает в случае сли в при получении команды /0/ при сохранении его в таблицу выясняется, что точно такой же в devhash имеется у другого клиента.

### 10.2.5 Класс - "command\_complete". Успешное выполнение исходящей команды.

Дополнительный параметр - количество выполненных команд.

Проверка на событие осуществляется только при получении команды /10/.

Если параметр равен 0 или null, то он расценивается как единица.

### 10.2.6 Класс - "geo\_change".

Смена страны. Никаких дополнительных параметров нет.

Проверка на событие осуществляется только при получении команды /0/.

## 10.3 По каждому сработанному событию в логе искомого клиента (пункт 9) должна быть запись с идентификатором сработавшего события

класс online - счетчик индивидуальный

класс age - глобальный, время отсчитывается от времени указанном в столбце created\_at

класс geo - счетчик не нужен

класс devhash\_dup - счетчик не нужен

класс command\_complete - счетчик индивидуальный

## 11 Механизм автодобавления команд клиенту

Реализовать механизм автодобавления команд клиенту при получении отчетов о событиях (команда 64).

Триггер автодобавления должен реагировать на имя модуля, имя события и поле info, при их возникновении сервер должен добавлять команду с указанным кодом и параметром.

Поле info должно проверяться по регулярному выражению.

Таким образом, если поле info содержит "qrrr45werty", то проверка по "r45" должна дать положительный результат, а проверка по "^r45" отрицательный.

Помимо прочего триггер должен содержать целочисленный параметр ограничивающий минимальную частоту срабатывания этого триггера на каждом конкретно взятом клиенте.

Параметр задается в секундах.

Если он равен 0, то ограничения нет.

Например, если равен 300 секунд, то если у данного клиента триггер сработал 180 секунд назад, то команду вставлять в очередь не надо.

Управление списком автодобавления должно осуществляться через таблицу.

Таблицу можно кешировать, регулярность обновления кеша должна задаваться через конфиг в секундах.