Разработка

Конфиг устройства

Обобщенный вид конфига

```
{
  "network": NETWORK,
"identification": IDENTIFICATION,
"ntp": NTP,
"sensors": SENSORS,
"data_push": DATA_PUSH,
"meta_info": META
}
```

NETWORK

Настройки сети на устройстве

```
{
"ip": "IP",
"gateway": "GATEWAY",
"mask": "MASK",
"dns": "DNS",
"dns2": "DNS_2"
}
```

- IP - IP устройства
- GATEWAY - адрес шлюза
- MASK - маска подсети
- DNS - DNS сервер
- DNS_2 - второй DNS сервер. доступно только у hikvision и vivotek

---

IDENTIFICATION

Параметры идентификации устройства

```
```

```
{
 "name": "NAME",
 "device_group": "DEVICE_GROUP",
 "model": "MODEL",
 "firmware": "FIRMWARE",
 "serial": "SERIAL",
 "mac": "MAC",
 "serial_as_name": SERIAL_AS_NAME
}
```

- NAME - имя устройства. Можно изменять в конфигурационном файле
- DEVICE_GROUP - группа устройства. Можно изменять в конфигурационном файле
- MODEL - модель
- FIRMWARE - прошивка
- MAC - мак адрес
- SERIAL_AS_NAME - флаг использования серийника (или сгенерированного uuid) в качестве имени устройтва. доступно только для brickstream и xovis. Можно изменять в конфигурационном файле
  - 0 - не использовать
  - 1 - использовать

NTP

Настройки синхронизации времени

```
{
 "host": "HOST",
 "set_gateway_as_ntp": GATEWAY_AS_NTP,
 "timezone": TIMEZONE,
 "sync_protocol": "SYNC_PROTOCOL",
 "sync_period": "SYNC_PERIOD"
}
```

- HOST - адрес сервера синхронизации времени. Можно изменять в конфигурационном файле
- GATEWAY_AS_NTP - использовать шлюз как сервер синхронизации времени. Можно изменять в конфигурационном файле
  - true - использовать
  - false - не использовать (по умолчанию)
- TIMEZONE - сдвиг относительно UTC в часах. Можно изменять в конфигурационном файле
- SYNC_PROTOCOL - протокол синхронизации времени (только для brickstream). Можно

изменять в конфигурационном файле
  - 0 - countmax
  - 1 - countmax
  - 2 - daytime
  - 3 - sntp
- SYNC_PERIOD - периодичность синхронизации (только для vivotek). Можно изменять в конфигурационном файле
  - hour
  - day
  - week
  - month

SENSORS

Настройки линий детектирования сенсора

```
[
  ...,

    {
      "id": "SENSOR_ID",
      "height_filter_max": HEIGHT_FILTER_MAX,
      "height_filter_min": HEIGHT_FILTER_MIN,
      "human": HUMAN
    },

    ...,
]
```

- SENSOR_ID - ID сepсора
- HEIGHT_FILTER_MAX - максимальное значение фильтра высоты.
  - диапазон для hikvision - от 40 до 200
  - диапазон для brickstream - от 0 для 255
  - максимальное значение для vivotek - 250
  - максимальное значение для xovis - 200. 0 - при выключенном фильтре
- HEIGHT_FILTER_MIN - минимальное значение фильтра. только для xovis и vivotek
  - минимальное значение для vivokek - 80
  - минимальное значение для xovis - 80. 0 - при выключенном фильтре
- HUMAN - цель. только для vivotek
  - true - человек
  - false - none

DATA_PUSH

Настройки выгрузок данных с устройства

```
[
  ...,

    {
        "type": "TYPE",
        "host": "HOST",
        "intervals": INTERVALS,
        "options": OPTIONS,
        "auth": AUTH,
        "path": PATH
    },

...,
]
```

TYPE

Тип выгрузки

- count - выгрузка метрик
- record - выгрузка видео. только для xovis, vivotek, brickstream
- log - выгрузка логов. только для xovis, vivotek, brickstream
- passive_count - выгрузка метрик с датчика агентом (пассивный режим датчика)

HOST

URL для выгрузки данных в формате `SCHEME://HOST[:PORT][/PATH]`. . Можно изменять в конфигурационном файле

- SCHEME - обязательный элемент. доступные значения. http, https, ftp, sftp
- HOST - обязательный элемент. принимает ip адрес или доменное имя
- PORT - необязательные элемент. при отсутствии будут браться порты по умолчанию
- PATH - необязательный параметр

В выгрузках с TYPE==passive_count валидация поля не происходит.

INTERVALS

Интервалы сбора/выгрузки данных

```
{
  "push": INTERVALS_PUSH,
    "aggregation": INTERVALS_AGGREGATION
}
```

- INTERVALS_PUSH - интервал выгрузки данных в секундах. Можно изменять в конфигурационном файле
  - brickstream
    - http выгрузки - 0, 60, 900, 1800, 3600
    - ftp выгрузки - 900, 3600, 86400
  - hikvision - 60, 300, 600, 900, 1200, 1800, 3600
  - vivotek - 60, 300, 900, 1800, 3600, 43200, 86400
  - xovis
    - для выгрузок с типом record - 0, 300, 900
    - для выгрузок с типом count - 60, 300, 900, 3600, 86400
    - для выгрузок с типом log - 5, 30, 60, 300, 900, 3600, 86400
- INTERVALS_AGGREGATION - интервал сбора данных в секундах. Можно изменять в конфигурационном файле
  - brickstream
    - http выгрузки - 60, 300, 900, 1800, 3600
    - ftp выгрузки - 300, 900, 1800, 3600
  - hikvision - не используется
  - vivotek - 60, 300, 900, 1800, 3600, 43200, 86400
  - xovis - 60, 300, 900, 1800, 3600, 21600, 43200, 86400

OPTIONS

Раздел с опциями отдельных устройств

```
{
  "format": "FORMAT",
  "upload_record": "UPLOAD_RECORD",
    "device_localtime": "DEVICE_LOCALTIME",
    "report_lite_mode": "REPORT_LITE_MODE"
}
```

- FORMAT - формат http выгрузок. только для vivotek и xovis. Можно изменять в конфигурационном файле
  - vivotek - 'xml', 'json', 'csv'. 'xml' - по умолчанию
  - xovis - 'xml_v1', 'xml_v2', 'json'. 'xml_v1' - по умолчанию

- UPLOAD_RECORD - используется только в hikvision. только для FTP выгрузок. Флаг выгрузки видео. Можно изменять в конфигурационном файле
  - 0 - не выгружать
  - 1 - выгружать
- DEVICE_LOCALTIME - используется только в vivotek. Флаг установки времени устройства в выгрузке. Можно изменять в конфигурационном файле
  - 0 - не устанавливать
  - 1 - устанавливать
- REPORT_LITE_MODE - используется только в vivotek. Флаг скрытия даты при нулевых счетчиках. Можно изменять в конфигурационном файле
  - 0 - выключено
  - 1 - включено

AUTH

Параметры для авторизации на сервере выгрузок. Пока только для FTP выгрузок

```
{
"user": "USER",
"password": "PASSWD"
}
```

- USER - имя пользователя. Можно изменять в конфигурационном файле
- PASSWD - пароль. Можно изменять в конфигурационном файле

PATH

Параметры сохранения на FTP сервере

```
{
"location": "LOCATION",
"filename": "FILENAME"
}
```

- LOCATION - папка сохранения выгрузки/видео. Можно изменять в конфигурационном файле
  - {{device_id}} - заменяется на id устройства
  - {{device_host}} - заменяется на адрес устройства
  - {{device_serial}} - заменяется на серийник устройтва
  - {{device_mac}} - заменяется на мак адрес устройства
- FILENAME - имя файла. только для brickscream. Можно изменять в конфигурационном файле

META

Раздел с мета-информацией устройства при команде запросе конфига. При команде установки конфига игнорируется

```
{
    "timezone_name": "TIMEZONE",
    "time_offset": TIME_OFFSET,
    "time_sync_enabled": TIME_SYNC_ENABLED,
    "log_enabled": LOG_ENABLED,
    "multisensor": MULTISENSOR,
    "slave_sensors": [
        ...,
        {
            "ip": "SLAVE_SENSOR_IP"
        },
        ...
    ],
    "device_heigth": DEVICE_HEIHTH
}
```

- TIMEZONE - имя таймзоны
- TIME_OFFSET - разница в часах с UTC
- TIME_SYNC_ENABLED - флаг включенной синхронизации времени
- LOG_ENABLED - флаг активной выгрузки логов
- MULTISENSOR - флаг включенного режима мультисенсора на устройстве
- SLAVE_SENSOR_IP - ip устройства добавленного в мультисенсор
- DEVICE_HEIGTH - высота устройства от пола

Команды от сервиса/1с

Процесс обработки команд

Механизм получения команд и отчета о выполнении команд

Шедулер агента по cron расписанию из параметра **CFG_AGENT_CHECK_CRON** запускает задачу запроса к сервису/1с для получения новых задач и отчета о выполнении более старых задач.

Есть возможность запуска задачи запроса к сервису/1с вне расписания. Для этого нужно сделать POST запрос (далее запрос-триггер) к агенту по адресу **AGENT_HOST:CFG_API_PORT/agent/commands_check**

- AGENT_HOST - ip адрес или доменное имя на котором запущен агент
- CFG_API_PORT - порт на котором запущен агент. значение параметра конфига

При получении запроса-триггера агент проверит дату последнего запроса к сервису. Если с момента прошлого запроса прошло более **CFG_AGENT_CHECK_UNCHEDULED_DELAY** секунд, то запрос к сервису выполнится сразу. Иначе запрос к сервису будет выполнен, когда после последнего запроса к сервису пройдет **CFG_AGENT_CHECK_UNCHEDULED_DELAY** секунд.

- CFG_AGENT_CHECK_UNCHEDULED_DELAY - параметр конфига задающий задержку между внеочередными запросами к сервиса. положительное целое число

Запрос к сервису/1с

Агент получает команды от сервиса/1с и отчитывается о выполнении команд путем POST запроса по адресу **CFG_COLLECTOR_SERVICE_BASE/agent/check**

(CFG_COLLECTOR_SERVICE_BASE берется из конфига агента)

В данных запроса в поле executed_commands передаются результаты выполнения команд агентом.

В ответе на запрос агент получает список актуальных команд, которые он должен выполнить и отчитаться.

Данные запроса

```
{
  "health": {
    "last_report_at": LAST_REPORT_AT,
    "status": "STATUS",
    "activity": "ACTIVITY",
    "up_time": UPTIME
  },
  "executed_commands": [
      ...,
      {
        "command_id": "COMMAND_ID",
        "device_id": [..., "DEVICE_ID", ...],
    "command": "COMMAND",
      "result": RESULT
      },
    ...
    ]
}
```

- LAST_REPORT_AT - время последнего отчета о статусе агента (поле пока не используется)
- STATUS - Усредненный статус по всем устройствам агента
  - ok - TODO
  - warning - TODO
  - error - TODO
- ACTIVITY - Последняя активность жизненного цикла агента (поле пока не используется)
- UPTIME - время работы агента в секундах
- COMMAND_ID - id команды
- DEVICE_ID - id устройства на котором должна была выполняться команда
- COMMAND - строка с командой
- RESULT - словарь с результатами выполнения команды (зависит от команды)

Ожидаемый ответ

```
{
    "commands":[
      ...,
      {
            "command_id": "COMMAND_ID",
            "device_id": [..., "DEVICE_ID", ...],
```

```
            "command": "COMMAND",
            "params": PARAMS
        },
□□...
    ]
}
```

- COMMAND_ID - id команды
- DEVICE_ID - id устройства на котором должна выполниться команда
- COMMAND - строка с командой
- PARAMS - словарь с параметрами команды (зависит от команды)

Механизм выполнения команды

После получения списка команд агент добавляет их в очередь команд и начинает их выполнение.

Команды выполняются асинхронно. Для каждого устройства внутри команды выполнение также асинхронно.

После завершения выполнения команды агент делает внеочередной запрос **agent/check** для отчета о выполнении команды.

Список команд

screenshot

получение скриншота с устройства.

params

```
{
    "host": "FTP_HOST",
    "user": "FTP_USERNAME",
    "password": "FTP_PASSWORD",
    "path": "FTP_PATH",
    "multisensor": "MULTISENSOR_FLAG"
}
```

- FTP_HOST - адрес FTP сервера для загрузки скриншота. Обязательный параметр
- FTP_USERNAME - имя пользователя на FTP сервере. Обязательный параметр
- FTP_PASSWORD - пароль пользователя на FTP сервере. Обязательный параметр
- FTP_PATH - полный путь сохранения скриншота на FTP сервере. ({{datetime}}.png по умолчанию)
  - {{datetime}} - заменяется на текущую дату на агента в формате '%Y%m%d%H%M%S'
  - {{device_id}} - заменяется на id устройства к которому адресована команда
  - {{device_host}} - заменяется на IP адрес устройства к которому адресована команда
  - {{device_serial}} - заменяется на серийный номер устройства к которому адресована команда
  - {{device_mac}} - заменяется на MAC адрес устройства к которому адресована команда
  - {{command_id}} - заменяется на id команды
  - {{agent_id}} - заменяется на id агента
- MULTISENSOR_FLAG - флаг скриншота мультисенсора
  - 0 - обычный скриншот
  - 1 - скриншот мультисенсора (делается при поддержке устройством)

success payload

```
{
"screenshot_path": "REMOTE_PATH"
}
```

- REMOTE_FILE - полный путь скриншота на FTP сервере

---

upload_data

выгрузка исторических данных с датчика

params

```
```

```
{
  "from": "FROM",
    "to": "TO"
}
```

- FROM - время начала исторических данных для выгрузки в формате unixtime * 1000
- TO - время окончания исторических данных для выгрузки в формате unixtime * 1000

success payload

```
{}
```

---

reboot

перезагрузка устройства

params

```
{}
```

success payload

```
{}
```

---

get_config

Запрос конфига устройства

params

```
{}
```

success payload

```
DEVICE_CONFIG
```

- DEVICE_CONFIG - конфиг устройства

---

set_config

Запрос конфига устройства

params

```
{
"DEVICE_ID": DEVICE_CONFIG
}
```

- DEVICE_ID - ID устройства
- DEVICE_CONFIG - конфиг устройства

success payload

```
DEVICE_CONFIG
```

- DEVICE_CONFIG - конфиг устройства

---

reset_auth

установка логина и пароля для авторизации на устройстве

params

```
{
"user": "USERNAME",
  "pass": "PASSWORD"
}
```

- USERNAME - новое имя пользователя (используется только на brickstream и vivotek)
- PASSWORD - новый пароль

success payload

```
{
"new_user": "USERNAME",
  "new_pass": "PASSWORD"
```

```
}
```

- USERNAME - новое имя пользователя
- PASSWORD - новый пароль

---

executor_flash

обновление прошивки на устройстве

params

```
{
  "file_name": FILENAME,
    "image": IMAGE,
    "timeout": TIMEOUT
}
```

- FILENAME - имя файла прошивки
- IMAGE - содержимое файла (HEX)
- TIMEOUT - таймаут

success payload

```
{
  "firmware": "FIRMWARE"
}
```

- FIRMWARE - версия прошивки. возвращается только из brickstream и xovis

---

video_record

установка расписания записи видео. работает

params

```
{
  "from": "FROM",
    "to": "TO",
    "sd_card_format": "FORMAT_FLAG",
```

```
  "quality": "QUALITY"
}
```

- FROM - время начала записи в формате unixtime * 1000
- TO - время окончания записи в формате unixtime * 1000
- FORMAT_FLAG - флаг форматирования CD карты на устройстве. Используется только для brickstream и hikvision
  - 0 - не форматировать
  - 1 - форматировать
- QUALITY - уровень качества видео (1 - 100). Используется только для hikvision и vivotek

success payload

```
{}
```

Работа с API сервиса/1c

`GET` agent/device *?{page}=1&{per_page}=20*

Получает актуальный список устройств

```
{
  "new_devices": NEW_DEVICES,
    "devices":[
     ...

    {
            "id": DEVICE_ID,
            "is_new": DEVICE_IS_NEW,
            "serial": DEVICE_SERIAL,
            "mac"   : DEVICE_MAC,
            "host"  : DEVICE_HOST,
            "adapter_id" : DEVICE_ADAPTER,
            "login": DEVICE_LOGIN,
            "password": DEVICE_PASSWORD,
            "timezone": DEVICE_TIMEZONE,
            "sensors": [
                ...
                {
                    "sensor_id": SENSOR_ID,
                    "line_id": SENSOR_LINE_ID
                },
                ...
            ]
     },

       ...
],
    "pagination":{
```

```
        "page": 1,
        "per_page": 20,
        "pages": 2,
        "total": 40
    }
}
```

- NEW_DEVICES - список устройств с флагом is_new. агентом не используется
- DEVICE_ID - uuid устройства. строка
- DEVICE_IS_NEW - флаг нового устройства. булево
- DEVICE_SERIAL - строка с серийником
- DEVICE_MAC - строка с mac адресом
- DEVICE_ADAPTER - строка с кодом типа устройства
- SENSOR_ID - uuid сенсора (линии подсчета)
- SENSOR_LINE_ID - id линии/счетчика от самого устройства
- DEVICE_HOST - ip устройства
- DEVICE_LOGIN - логин для доступа на устройство
- DEVICE_PASSWORD - пароль для доступа на устройство
- DEVICE_TIMEZONE - строка с таймзоной

При получении списка устройств агент проводит сверку полученного списка (блок devices) со своей локальной базой:

- Если устройство есть на агенте, но нет в списке от сервиса, флаг is_new не установлен на агенте, то такое устройство удаляется с агента
- Если устройство есть на агенте, флаг is_new установлен, в списке от сервиса оно есть без установленного флага, то ID устройства на агенте заменяется тем что прислал сервис, ID серсоров так же заменяется (с привязкой к line_id). Флаг is_new на агенте снимается. При этом в хранимых метриках по такому устройству так же заменяются ID устройства, ID серсора (и затем отправляются в сервис стандартным способом).
- Если устройство есть на агенте, флаг is_new установлен, и его нет в списке от сервиса. То с таким устройством ничего не делать и хранить метрики только на агенте.

`POST` agent/device

регистрация нового устройства в сервисе

тело запроса - строка json:

```
{
    "id": DEVICE_ID,
    "is_new": DEVICE_IS_NEW,
```

```
    "serial": SERIAL,
    "mac": MAC,
    "adapter_id": ADAPTER_ID,
    "sensors": [
    ...
      {
        'sensor_id": SENSOR_ID,
          'line_id": SENSOR_LINE_ID
        },
      ...
    ],
    "host": HOST,
    "login": LOGIN,
    "password": PASSWORD,
    "time_zone": TIMEZONE
}
```

- DEVICE_ID - uuid устройства. строка
- DEVICE_IS_NEW - флаг нового устройства. булево
- SERIAL - строка с серийником
- MAC - строка с mac адресом
- ADAPTER_ID - строка с кодом типа устройства
- SENSOR_ID - uuid сенсора (линии подсчета)
- SENSOR_LINE_ID - id линии/счетчика от самого устройства
- HOST - ip устройства
- LOGIN - логин для доступа на устройство
- PASSWORD - пароль для доступа на устройство
- TIMEZONE - строка с таймзоной

POST agent/check

Запрос

отчет о состоянии агента, отчет о выполненных командах

Тело запроса - строка json:

```
{
  'health': {
    'last_report_at": 0.0,
    'status": STATUS,
```

```
    "activity": "working",
    "up_time": UPTIME
},
'executed_commands': EXECUTED_COMMANDS
}
```

- STATUS - статус агента
  - ok - количество неактивных устройств равно нулю
  - warning - количество неактивных устройств менее установленного в конфиге
  - error - количество неактивных устройств более установленного в конфиге
- UPTIME - количество секунд с момента запуска агента
- EXECUTED_COMMANDS - список результатов отработанных команд

EXECUTED_COMMANDS:

```
{
    "command_id": COMMAND_ID
    "device_id": DEVICES
    "command": COMMAND
    "result": COMMAND_RESULT
}
```

- COMMAND_ID - uuid команды
- DEVICES - список с uuid устройств
- COMMAND - строка с названием команды
- COMMAND_RESULT - результат выполнения команды

COMMAND_RESULT:

```
{
"error_flag": ERROR_FLAG,
  "message": MESSAGE,
  "payload": [
    ...
      {
        "error_flag": PAYLOAD_ERROR_FLAG,
          "message": PAYLOAD_MESSAGE,
          "payload": PAYLOAD_PAYLOAD,
          "device_id": PAYLOAD_DEVICE_ID
        }

      ...
```

```
    ]
}
```

- ERROR_FLAG - флаг ошибки всей команды. значения 0 и 1
- MESSAGE - сообщения о результатах выполнения всей команды. строка или список строк
- PAYLOAD_ERROR_FLAG - флаг ошибки выполнения команды на конкретном устройстве. значения 0 и 1
- PAYLOAD_MESSAGE - сообщения о результатах выполнения всей команды на конкретном устройстве. строка или список строк
- PAYLOAD_PAYLOAD - результат выполнения команды на конкретном устройстве. список или словарь
- PAYLOAD_DEVICE_ID - id устройства

Ответ

```
{
  ...
  "payload": {
    "commands": [
        ...
          {
            "command_id": COMMAND_ID,
              "device_id": DEVICES,
              "command": COMMAND,
              "params": COMMAND_PARAMS
            }
          ...
        ]
    }
  ...
}
```

- COMMAND_ID - uuid команды
- DEVICES - список устройств (uuid)
- COMMAND - строка команды
- COMMAND_PARAMS - словарь с параметрами команды

описание команд и их параметров

Локальная база агента

Поддержка пассивной выгрузки метрик

Датчик	Поддержка пассивной выгрузки
3db	✘
3dd	✔
3dh	✔
3dm	✘
3dt	✘
3dv	✔
3dx	✔
cm*	✔[1]

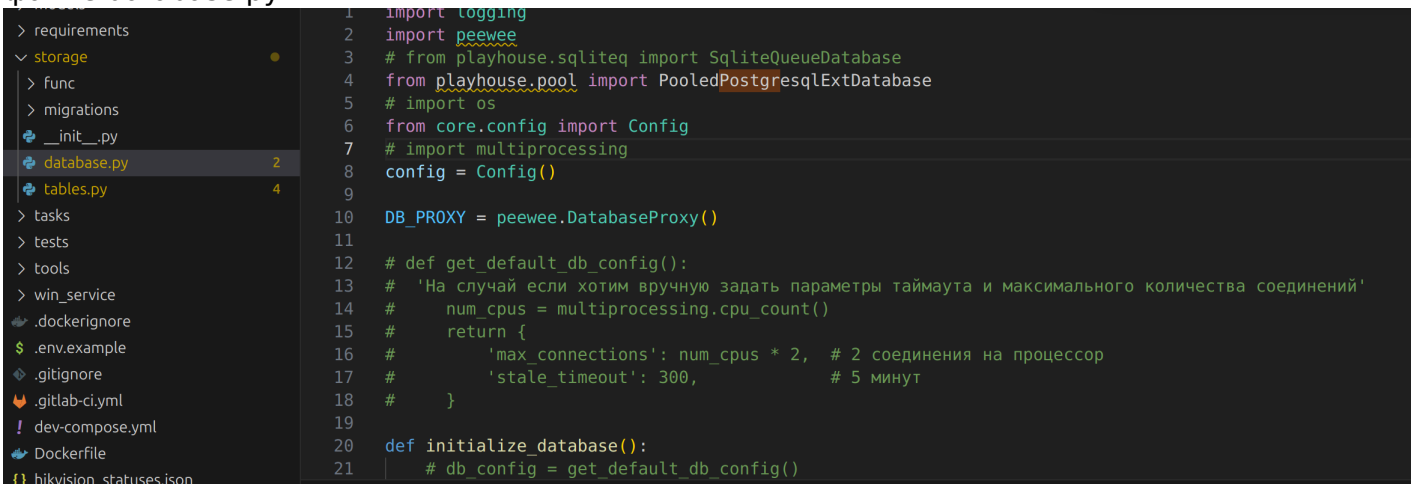
1. Только пассивная выгрузка

Опциональное подключение и настройка PostgreSQL

Для корректной работы Postgres необходимо заполнить поля в .env:

CFG_POSTGRES_MODE=1 # При 0 - деактивируем мод
CFG_POSTGRES_DB=postgres
CFG_POSTGRES_USER=postgres
CFG_POSTGRES_PASSWORD=postgres
CFG_POSTGRES_PORT=5432
CFG_POSTGRES_HOST=localhost

Также был заложен функционал настройки одновременного количества подключений в файле database.py



Настройка режима делегата

Делегат настраивается с двух сторон: в нашем контуре и в контуре клиента. Связываются агенты между собой по websocket и команды нужно направлять на агент который в нашем контуре

Конфигурация .env файла:

В нашем контуре:

CFG_PRODTESTING_DELEGATE_MODE - Активация принимающего вебсокет эндпоинта

В контуре клиента:

- **CFG_AGENT_WORK_MODE** = DELEGATE - Включаем режим работы делегата
- **CFG_DELEGATE_MODE_TOKEN -** токен агента
- **CFG_DELEGATE_MODE_AGENT_URL** = wss://agent.rarus-spp.ru/**область_агента** /agent/delegates - URL эндпоинт нашего принимающего агента (в нашем контуре), важно что это не **http** или **https**, а **wss**