Основные сведения о HAL NetroManager

NetroManager - служба, запускаемая из соответствующего исполняемого файла. Цель службы - обеспечение унифицированного интерфейса между модемами умного дома(Netro,Z-wave,bluetooth,gsm) и сервером умного дома(написанном на JAVAScript). Служба NetroManager принимает команды от сервера умного дома в формате json посредством tcp либо unix сокета и транслирует их в набор команд для определенного модема умного дома. По окончанию выполнения команды, либо в случае какой-либо ошибки, NetroManager отправляет ответ в json-формате на сервер умного дома посредством сокета. NetroManager также обрабатывает инициативные команды от модемов(от датчиков, пультов) и формирует соответсвующие json-команды, которые отправляются через ранее заданный сокет на сервер умного дома.

Формат команд NetroManager

Формат команд, посредством которых осуществляется обмен с NetroManager, представляет собой JSON-объекты. Каждый такой JSON-объект содержит 2 обязательных поля:

1. “class” типа string, содержащий класс команды

2. “command” типа string, содержащий название команды

Все остальные поля зависят от конкретного типа команды и заполняются исходя из ниже приведенных таблиц(Сеть Intro3, Сеть Z-Wave, Сеть GSM). Между командами отсутствует какой-либо разделитель, что нужно учитывать при потоковом разборе ответов от NetroManager

Все команды могут комплектоваться парой ключ:значение - “label”:someLabel. Ответы на каждую из команд содержат ту же пару “label”:someLabel, которая содержится и в команде. Сделано это для возможности отслеживания хода обработки команды клиентом, если в NetroManager от данного клиента отправляется сразу несколько команд. Ответ на каждую команду дополнительно содержит внутренний уникальный индекс команды “index”:uint32_value для возможности использования сервисных команд применительно к отправленной команде(например, есть возможность отменить команду, убрать ее из очереди выполнения при помощи cancelCommand). Команда для netro manager может также содержать поле “priority”:uint32_value для приоритетной обработки команды в очереди по сравнению с другими командами(минимальный приоритет 0 - выставляется командам без поля “priority”).

Пример алгоритма обработки команды от клиента для NetroManager:

1. Клиент передает в NetroManager команду управления устройством №1234

{“class”: “netro”,“command”: “controlDevice”,“id” : 1234,“action” : “loop”,“ack”:true}

2. SHNetroManager в ответ присылает подтверждение приема команды(“status”:“accepted”) и размещения во внутренней очереди команды(значение “queue”)

{“class”:“netro”,“command”:“answer”,“index”:1,“status”:“accepted”,“queue”:0,“waitTime”:4720}

3. Далее когда очередь доходит до данной команды, и NetroManager приступает к ее выполнению, он присылает сообщение серверу умного дома(“status”:“started”)

{“class”:“netro”,“command”:“answer”,“index”:1,“status”:“started”}

4. По мере выполнения команды SHNetroManager может присылать на сервер умного дома прогресс выполнения(“status”:“progress”)

{“class”:“netro”,“command”:“answer”,“index”:1,“status”:“progress”,“progress”:25}

{“class”:“netro”,“command”:“answer”,“index”:1,“status”:“progress”,“progress”:50}

{“class”:“netro”,“command”:“answer”,“index”:1,“status”:“progress”,“progress”:75}

5. И результат выполнения команды, например ошибку(“status”:“error”) связи с устройством умного дома(описание ошибки содержится в поле “error” и “errorDescription”)

{“class”:“netro”,“command”:“answer”,“index”:1,“status”:“error”,“error”:“device timeout”}

либо успешность выполнения команды(“status”:“completed”)

{“class”:“netro”,“command”:“answer”,“index”:1,“status”:“completed”}

Команды и ответы на них описаны в таблицах “Команды NetroManager” и “Ответы NetroManager”

Инициативные команды всем подключенным клиентам от NetroManager

NetroManager также может информировать всех подключенных к нему клиентов(по socket технологии) о различных событиях, происходящих в сети Netro,ZWave,GSM(и проч.)и на сервере(статусе батарейки и присутствия питания 220В, статусе wifi-wps). Клиент не должен формировать какой-либо ответ на данный вид полученной информации. Пример такой команды, дающий информацию клиентам о том, что закончено построение сети устройств netro:

{“class”:“netro”,“command”:“eventCommand”,“status”:“completed”,“event”:“buildingNetworkFinished”}

NetroManager API

Дополнительные команды netromanager

Класс команды (ключ “class”)	Мнемоника (ключ “command”)	Описание	Необходимые параметры	Возможные параметры	Ответ на команду
Восстановление системы					
netro	createBackup	Создать точку восстановления системы			Стандартный быстрый ответ, стандартный полный ответ
netro	restoreBackup	Восстановить систему согласно точки восстановления	**modemData** - данные для восстановления модема(тип: base64_string) **managerData** - данные для восстановления netroManager(тип: base64_string)		Стандартный быстрый ответ, стандартный полный ответ
netro	getModemParams	Запрос параметров модема	Хотя бы один из возможных параметров запроса	**snifferEnabled** - включение/отключение сниффера на модеме(тип bool)	Стандартный быстрый ответ, стандартный полный ответ
netro	setModemParams	Настройка парметров модема	Хотя бы один из возможных параметров для установки	**snifferEnabled** - включение/отключение сниффера на модеме(тип составной)	Стандартный быстрый ответ, стандартный полный составной ответ
netro	createRoute	создать пользовательский маршрут к устройству netro3	**id** - адрес устройства, к которому создается маршрут(тип: uint32) **path** - маршрут(массив адресов устройств типа uint32)		Стандартный быстрый ответ, стандартный полный ответ

Класс команды (ключ “class”)	Мнемоника (ключ “command”)	Описание	Необходимые параметры	Возможные параметры	Ответ на команду
service	setConnectionParams	Настройка параметров клиентского соединения	Хотя бы один из параметров	**peripheryEvents** - поддержка сокетом ивентов, адресованных серверной периферии(тип: bool, по умолчанию false) **smartDevicesEvents** - поддержка сокетом ивентов устройств умного дома(тип: bool,по умолчанию true) **snifferEvents** - поддержка сокетом ивентов от сниффера(тип: bool) **gsmEvents** - поддержка сокетом ивентов от сети gsm(тип: bool) **guiEvents** - поддержка сокетом ивентов для gui(тип: bool) **networkEvents** - поддержка сокетом ивентов сетевой организации wifi,eth,netro и zwave(тип: bool)	стандартный составной полный ответ

Система авторизации на сервере умного дома по локальной сети в случае наличия на нем ПО SHGate

Когда NetroManager на сервере запущен с параметром –loopback, либо в /smarthome/options.conf данный параметр “loopback_mode”:1, для получения доступа к NetroManager необходимо пройти авторизацию с помощью сервиса SHGate. У данного сервиса открыт порт 1333 с надстройкой ssl TLSv1_2. Авторизация происходит с помощью команды “class”:“auth”, “command”:“register”. После успешной регистрации SHGate используется для проброса шифрованных сообщений по локальной сети от клиента к NetroManager и обратно. При переподключении клиента к сервису SHGate требуется повторная авторизация

SHgate по умолчанию принимает стандартный логин (выдаваемый тех. поддержкой) в качестве данных для авторизации, если иное не указано в файле /smarthome/options.conf (параметры “service_login” и “service_password”)

В случае, когда авторизация не пройдена, на любые команды для NetroManager будет получен ответ от сервиса SHGate {“class”:“netro”,“command”:“answer”,“status”:“error”,“error”:“command denied”,“errorDescription”:“authorization is required”}

Для генерации сертификата и ключей шифрования необходимо выполнить из-под каталога /smarthome команду **openssl req -newkey rsa:2048 -nodes -keyout gate-key.pem -x509 -out gate-cert.pem -subj “/C=BY/ST=Denial/L=NeroElectroniucs/O=Dis/CN=www.nero-home.by [http://www.nero-home.by]“**

Класс команды (ключ “class”)	Мнемоника (ключ “command”)	Описание	Необходимые параметры	Возможные параметры	Пример использования
auth	register	Авторизация на сервере умного дома	**login** - имя пользователя для регистрации(тип: string) **password** - пароль для регистрации (тип: string}		**Клиент** {“class”:“auth”,”command”:“register”,“password”:“1111”,“login”:“superuser”} **Ответ** {“class”:“auth”,“command”:“answer”,“status”:“error”,“error”:“bad param”,“errorDescription”:“incorrect login\/password”} **либо** {“class”:“auth”,“command”:“answer”,“status”:“completed”}

Сеть INTRO 3

1. Описание команд

Класс команды (ключ “class”)	Мнемоника (ключ “command”)	Описание	Необходимые параметры	Возможные параметры	Пример использования
Устройства					

Класс команды (ключ “class”)	**Мнемоника (ключ “command”)**	**Описание**	**Необходимые параметры**	**Возможные параметры**	**Пример использования**
netro	controlDevice	Управление устройством либо группой устройств	Либо **netro3Group**, либо **id** - группа устройств/адрес устройства для управления(для группы значения от 1 до 511, для id - значения от 1 до 16777215) **action** - действие, которое должно выполнить устройство (тип: string, значения- {“up”(включить), “down”(выключить), “stop”, “loop”(управление по кольцу), “comfort1”(заданный режим комфорта), “comfort2”(заданный режим комфорта), “script”(выполнение специальной заранее заданной команды), “pos”(установить в положение), “upJal”(Вверх ламели жалюзи), “downJal”(Вниз ламели жалюзи),“extScriptN”(N- число от 1 до 16, расширенный шаблонный сценарий, прописанный в устройство),“lock”(закрыть замок),“unlock”(открыть замок)} **position** (если “action” == “pos”) - значение положения для установки (тип: float, значения: 0-1, шаг 0.0001)	**delayed** - параметр актуален при управлении группой устройств, когда необходимо последовательная сработка устройств(чтобы не создавать нагрузки на электрическую сеть, тип bool, по умолчанию false) **fromBorder** - установка в положение(“action” = “pos”) относительно верхней либо нижней границы для роллетного прибора (тип: string, значения {“top”,“bottom”}). По умолчанию позиции более 0.5 выставляются относительно верхней границы, меньше либо равно 0.5 - относительно нижней **ack** - необходимость подтверждения от устройства об успешном выполнении команды (тип: bool) **slots** - количество слотов ретрансляции при управлении netro3Group(тип uint8, default = min(64,max(кол-во устройств,4))) **brightness** (если “action” == “pos”) - значение яркости для установки на диммере (тип: float, значения: 0-1, точность 0.01)	**Клиент** {“class”: “netro”,“command”: “controlDevice”,“id” : 8755,“action” : “pos”,“position”:0.02,“ack”:true} **Ответы:** {“class”:“netro”, “command”:“answer”, “label”:10, “index”:221, “status”:“accepted” ,“queue”:0, “waitTime”:0} {“class”:“netro” ,“command”:“answer”, “label”:10, “index”:221, “status”:“started”} “class”:“netro”, “command”:“answer”, “label”:10, “index”:221, “status”:“completed”, “destinationPosition”:0, “timeToGo”:200, “direction”:“down”}
netro	getDevicePosition	Запрос положения устройства	**id** - адрес устройств для запроса (значения от 1 до 16777215)		**Клиент** {“class”:“netro”, “command”:“getDevicePosition”, “id”:8755, “label”:11} **Ответы:**{“class”:“netro”, “command”:“answer”, “label”:11, “index”:222, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”: “netro”, “command”:“answer”, “label”:11, “index”:222, “status”:“started”} {“class”:“netro”, “command”:“answer”, “label”:11, “index”:222, “status”:“completed”, “currentPosition”:{“value”:0}, “destinationPosition”:{“value”:0}, “brightness”: {“value”:0}}
netro	getDeviceParams	Запрос параметров устройства	**id**- адрес устройства(значения от 1 до 16777215) Хотя бы один из возможных параметров запроса	**workMode** - запрос режима работы ИУ(тип: {value: bool, channels:[uint8, ..,] (опционально default = [1])})	**Клиент** {“class”:“netro”, “comfort2”:true,“command”:“getDeviceParams”, “id”:8755, “label”:14, “version”:true,

Класс команды (ключ “class”)	Мнемоника (ключ “command”)	Описание	Необходимые параметры	Возможные параметры	Пример использования
				rcGroupMode - запрос режима работы группы пульта(тип: {value: bool, groups:[uint8, ..,]) **buttonsMode** - запрос режима работы кнопок (тип: bool, для netro3). **moveTime** - запрос времени движения (тип: bool). Применительно к 7513 micro **comfort1** - запрос комфорта (тип: bool). Это значение в процентах от максимального времени для 7522, уровня освещения и нарастания с длительностью освещения в 7521, времени движения в 7513 micro/IN. **comfort2** - запрос комфорта (тип: bool). Это значение в процентах от максимального времени 65535 сек. для 7522, уровня освещения и нарастания с длительностью освещения в 7521, времени движения в 7513 micro/IN. **extendedScript** - запрос сложного сценария (тип: {value:bool, numbers:[uint8, .., uint8]}, где numbers - номера расширенных сценариев) **version** - запрос версии прибора (тип: bool) **temperatureSensor** - режим работы датчика температуры в сценарных панелях (тип: bool). Заводская настройка - отправка сообщений на сервер при изменении температуры включена. **programmedRCs** - запрограммированные в исполнительное устройства пульты(тип bool) **routes** - созданные через устройство маршруты(тип bool) **intro3Channel** -номер канала intro3(тип составной) **rssi** - информация о качестве сигнала связи с устройством(тип bool)	“workMode”:{“value”:true}} **Ответы:** {“class”:“netro”, “command”:“answer”,“label”:14, “index”:225,“status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“netro”, “command”:“answer”, “label”:14, “index”:225, “status”:“started”} {“class”:“netro”, “command”:“answer”, “label”:14, “index”:225, “status”:“progress”, “progress”:33} {“class”:“netro”, “command”: “answer”, “label”:14, “index”:225, “status”:“progress”, “progress”:66} {“class”:“netro”, “command”:“answer”, “label”:14, “index”:225, “status”:“completed”, “version”:{“product”:{“hard”:2, “variant”:1, “id”:113}, “version”:113, “address”:4096}, “comfort2”:{“value”:0.01}, “workMode”: [{“mode”:“relay”, “memoryMode”:false, “clockwise”:false, “comfortOn”:false, “channel”:1}]}

Класс команды (ключ “class”)	Мнемоника (ключ “command”)	Описание	Необходимые параметры	Возможные параметры	Пример использования
				wakeup - информация о просыпании устройства на батарейках(тип bool) **groupContent** - запрос состава одной или нескольких определенных групп пультов (тип: {value:bool, groups:[uint16, .., uint16] либо group:uint16}) **groupsIncluded** - запрос сведений о всех группах, в которых находится исполнительное устройство (тип:bool) **riseTime**- время нарастания/ убывания яркости 7521(тип bool) **dimmingStep**- шаг диммирования 7521(тип bool)	

Класс команды (ключ “class”)	Мнемоника (ключ “command”)	Описание	Необходимые параметры	Возможные параметры	Пример использования
netro	setDeviceParams	Настройка параметров устройства	**id**-адрес устройства(значения от 1 до 16777215) Хотя бы один из возможных параметров настройки	**workMode** - настройка режима работы ИУ (тип: составной, для netro3) **rcGroupMode** - настройка режима работы группы пульта(тип: составной, для netro3) **buttonsMode** - настройка режима работы кнопок(тип: составной, для netro3) **moveTime** - настройка времени движения (тип: составной) **comfort1** - настройка комфорта (тип: составной) Это значение в процентах от максимального времени 65535 сек. для 7522, уровня освещения и нарастания с длительностью освещения в 7521, времени движения в 7513 micro/IN. **comfort2** - настройка комфорта (тип: составной). Это значение в процентах от максимального времени для 7522, уровня освещения и нарастания с длительностью освещения в 7521, времени движения в 7513 micro/IN. **extendedScript** - Настройка сложного сценария (тип: составной, для netro3) **temperatureSensor** - установка режима работы датчика температуры в сценарных панелях (тип: составной) **intro3Channel** -номер канала intro3(тип составной) **wakeup** - информация о просыпании устройства на батарейках(тип составной) **riseTime**- время нарастания/ убывания яркости 7521(тип составной) **dimmingStep**- шаг диммирования 7521(тип составной) **ack** - необходимость подтверждения от устройства об успешном выполнении команды (тип: bool)	**Клиент** {“ack”:true, “class”:“netro”, “comfort1”: {“value”:0.1234}, “command”:“setDeviceParams”, “id”:8755, “label”:15, “workMode”:[{“channel”:1, “mode”:“relay”}]} **Ответы:** {“class”:“netro”, “command”:“answer”, “label”:15, “index”:226, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“netro”, “command”:“answer”, “label”:15, “index”:226, “status”:“started”} {“class”:“netro”, “command”:“answer”, “label”:15, “index”:226, “status”:“progress”, “progress”:50} {“class”:“netro”, “command”:“answer”, “label”:15, “index”:226, “status”:“completed”, “comfort1”:{“set”:true}, “workMode”:[{“set”:true, “channel”:1}]}

Класс команды (ключ “class”)	**Мнемоника (ключ “command”)**	**Описание**	**Необходимые параметры**	**Возможные параметры**	**Пример использования**
netro	deleteDeviceGroup	Удаление группы из устройства	**id** - устройство, которому предназначена команда (значения от 1 до 16777215, **параметр не обязателен, в случае его отсутствия группа чистится целиком**) **deleteGroup** - группа для удаления (значения: 1-511)	**remoteId** - id пульта, группу которого нужно удалить(значения от 0 до 16777215, 0-сервер) **ack** - необходимость подтверждения от устройства об успешном выполнении команды (тип: bool)	**Клиент** {“ack”:true, “class”:“netro”, “command”:“deleteDeviceGroup”, “deleteGroup”:1, “id”:8755, “label”:16, “remoteId”:1234} **Ответы:** {“class”:“netro”, “command”:“answer”, “label”:16, “index”:227, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“netro”, “command”:“answer”, “label”:16, “index”:227, “status”:“started”} {“class”:“netro”, “command”:“answer”, “label”:16, “index”:227, “status”:“completed”}
netro	writeDeviceGroup	Запись дополнительной группы в устройство(одной командой), либо сценария	**id** - устройство, которому предназначена команда (тип: uint32) **writeGroup** - группа для записи (тип: значения: 1-511)	**remoteId** - id пульта, группу которого нужно записать(значения от 0 до 16777215, 0-сервер) **ack** - необходимость подтверждения от устройства об успешном выполнении команды (тип: bool) **action** - запись действия по умолчанию на группу (тип: string/float, значения- {“up”, “down”, “stop”, “loop”, “comfort1”, “comfort2”,“extScriptN”(N- число от 1 до 16)})	**Клиент** {“ack”:true, “action”:“comfort1”, “class”:“netro”, “command”:“writeDeviceGroup”, “id”:8755, “label”:13, “remoteId”:1234, “writeGroup”:2} **Ответы:** {“class”:“netro”, “command”:“answer”, “label”:13, “index”:224, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“netro”, “command”:“answer”, “label”:13, “index”:224, “status”:“started”} {“class”:“netro”, “command”:“answer”, “label”:13, “index”:224, “status”:“completed”}
Управление пультами/устройствами Intro3 в памяти умного дома					
netro	listenToRemote	Прослушивание пультов и датчиков			**Клиент** {“class”:“netro”, “command”:“listenToRemote”, “label”:17} **Ответы** {“class”:“netro”, “command”:“answer”, “label”:17, “index”:228, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“netro”, “command”:“answer”, “label”:17, “index”:228, “status”:“started”} {“class”:“netro”, “command”:“answer”, “label”:17, “index”:228, “status”:“progress”, “progress”:50, “warning”:“user action wait;”} {“class”:“netro”, “command”:“answer”, “label”:17, “index”:228, “status”:“progress”, “progress”:75, “warning”:“user action wait;”} {“class”:“netro”, “command”:“answer”, “label”:17, “index”:228, “status”:“completed”, “remoteID”:{“id”:214019, “class”:“7501remoteit”}}

Класс команды (ключ “class”)	**Мнемоника (ключ “command”)**	**Описание**	**Необходимые параметры**	**Возможные параметры**	**Пример использования**
netro	saveRemote	Сохранение в модеме данных об услышанном ранее пульте, датчике либо устройстве Netro3(с этапа hello)	**id** - адрес услышанного пульта или устройства с этапа hello (значения от 1 до 16777215)	**code** - ключ доступа к устройству для сохранения(тип uint16)	**Клиент** {“class”:“netro”, “code”:55, “command”:“saveRemote”, “id”:214019, “label”:18} **Ответы:** {“class”:“netro”, “command”:“answer”, “label”:18, “index”:229, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“netro”, “command”:“answer”, “label”:18, “index”:229, “status”:“started”} {“class”:“netro”, “command”:“answer”, “label”:18, “index”:229, “status”:“completed”, idGroupsChanged:[1234,4321]}
netro	deleteRemote	Удаление из модема данных о сохраненном пульте либо устройстве Netro3	**id** - адрес сохраненного пульта/netro3 устройства (значения от 1 до 16777215)	**slots** -количество слотов флуда при удалении(uint8, default = 8) **broken**:bool{default:true}	**Клиент** {“ack”:true, “class”:“netro”, “command”:“deleteRemote”, “id”:214019, “label”:20} **Ответы:** {“class”:“netro”, “command”:“answer”, “label”:20, “index”:231, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“netro”, “command”:“answer”, “label”:20, “index”:231, “status”:“started”} {“class”:“netro”, “command”:“answer”, “label”:20, “index”:231, “status”:“progress”, “progress”:33} {“class”:“netro”, “command”:“answer”, “label”:20, “index”:231, “status”:“progress”, “progress”:66} {“class”:“netro”, “command”:“answer”, “label”:20, “index”:231, “status”:“completed”}
Управление модемом					
netro	setModemParams	Настройка парметров модема	Хотя бы один из параметров	**intro3Channel** -номер канала intro3(тип составной) **resetWatchDog** - сторожевой таймер сброса системы(тип составной) **transceiver** - выбор приемопередатчика для работы(тип составной)	**Клиент** {“class”:“netro”, “command”:“setModemParams”, “intro3Channel”:{“value”:5}, “label”:27,“transceiver”: {“value”:1}} **Ответы:** {“class”:“netro”, “command”:“answer”, “label”:27,“index”:252, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“netro”, “command”:“answer”, “label”:27, “index”:252, “status”:“started”} {“class”:“netro”, “command”:“answer”,“label”:27, “index”:252, “status”:“progress”, “progress”:50} {“class”:“netro”, “command”:“answer”, “label”:27, “index”:252, “status”:“completed”, “intro3Channel”: {“set”:true}, “transceiver”: {“set”:true}}

Класс команды (ключ “class”)	**Мнемоника (ключ “command”)**	**Описание**	**Необходимые параметры**	**Возможные параметры**	**Пример использования**
netro	resetModem	Сброс настроек и памяти модема(сброс всей информации о сети intro3)		**slots** -количество слотов флуда очистки устройств в эфире, для netro(uint8, default = 8)	**Клиент** {“class”:“netro”,“command”:“resetModem”, “slots”:12, “label”:23} **Ответы:** {“class”:“netro”, “command”:“answer”, “label”:23, “index”:234, “status”:“accepted”, “queue”:0, “waitTime”:0} событие начала очистки сети {“class”:“netro”, “command”:“eventCommand”, “status”:“completed”, “event”:“destroyingNetworkStarted”} {“class”:“netro”, “command”:“answer”, “label”:23, “index”:234, “status”:“started”} {“class”:“netro”, “command”:“answer”, “label”:23, “index”:234, “status”:“progress”, “progress”:25} {“class”:“netro”, “command”:“answer”, “label”:23, “index”:234, “status”:“progress”, “progress”:50} {“class”:“netro”, “command”:“answer” ,“label”:23, “index”:234, “status”:“progress”, “progress”:75} {“class”:“netro”, “command”:“answer”, “label”:23, “index”:234, “status”:“completed”} событие об окончании очистки сети {“class”:“netro”, “command”:“eventCommand”, “status”:“completed”, “event”:“destroyingNetworkFinished”}

Класс команды (ключ “class”)	**Мнемоника (ключ “command”)**	**Описание**	**Необходимые параметры**	**Возможные параметры**	**Пример использования**
netro	getModemParams	Запрос параметров модема	Хотя бы один из возможных параметров запроса	**modemMemory** - запрос состояния заполненности памяти модема данными об устройствах сети (тип: bool) **modemFunctions** - запрос функциональных возможностей модема (тип: bool) **version** - запрос текущей версии ПО и аппаратной части модема (тип: bool) **savedIds** - список сохраненных в модеме устройств/пультов(тип: bool) **intro3Channel** - номер частотного канала трансивера интро3(тип:bool) **resetWatchDog** - состояние сторожевого таймера сброса системы(тип:bool) **accelSensor** - показания датчика акселерометра модема(тип:bool) **lightSensor** - значение датчика освещенности сервера(тип:bool) **transceiver** - выбранный приемопередатчик для работы(тип bool) **rssi** - информация о качестве сигнала связи модема(тип bool)	**Клиент** {“class”:“netro”, “command”:“getModemParams”, “intro3Channel”:true, “label”:22, “version”:true} **Ответы:** {“class”:“netro”, “command”:“answer”, “label”:22, “index”:233, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“netro”, “command”:“answer”, “label”:22, “index”:233, “status”:“started”} {“class”:“netro”, “command”:“answer”, “label”:22, “index”:233, “status”:“progress”, “progress”:50} {“class”:“netro”, “command”:“answer”, “label”:22, “index”:233, “status”:“completed”, “version”: {“product”: {“hard”:22, “variant”:1, “id”:73}, “version”:3, “address”:4096}, “intro3Channel”: {“value”:5}}
Организация сети					

Класс команды (ключ “class”)	Мнемоника (ключ “command”)	Описание	Необходимые параметры	Возможные параметры	Пример использования
netro	startHello	Начать процесс определения незарегистрированных в сети устройств	**slots** - количество слотов для ожидания ответов от незарегистрированных устройств(тип uint8, значения 1-255)	**allowedIds** - список “своих” устройств(тип - массив чисел от 1 до 16777215) - если не задан, считает все найденные устройства “своими” **helloServer** - адрес сервера, который будет проводить процесс определения устройств(тип: uint32, если параметр не передается либо равен 0, то процесс определения производится с модема) **muteTime** - время молчания устройств после ответа на hello(тип uint8, 1-255 секунд, по умолчанию 255 секунд) **full** - проведение hello по всей сети устройств(тип bool, default: true) **clearBadRoutes** - после hello очистить все неподтвержденные с помощью hello маршрутные связи(value:bool, default:false)	**Клиент** {“allowedIds”:[11366,8755], “class”:“netro”, “command”:“startHello”, “helloServer”:0, “label”:24, “slots”:8} **Ответы:** {“class”:“netro”, “command”:“answer”, “label”:24, “index”:235, “status”:“accepted”, “queue”:0, “waitTime”:0} Событие начала построения сети {“class”:“netro”, “command”:“eventCommand”, “status”:“completed”, “event”:“buildingNetworkStarted”} {“class”:“netro”, “command”:“answer”, “label”:24, “index”:235, “status”:“started”} {“class”:“netro”, “command”:“answer”, “label”:24, “index”:235, “status”:“progress”, “progress”:0} сообщения с прогрессом построения {“class”:“netro”, “command”:“answer”, “label”:24, “index”:235, “status”:“completed”, “helloIds”: [{“id”:8755, “class”:“7522micro”}, {“id”:11366, “class”:“7513micro”}]}
netro	healNetwork	Ремонт маршрутов к устройствам в сети, восстановление групп из устройств		**id** - конкретное устройство, маршрут к которому нужно восстановить(число от 1 до 16777215)	**Клиент** {“ack”:true, “class”:“netro”, “command”:“healNetwork”, “label”:31} **Ответы:** {“class”:“netro”, “command”:“answer”, “label”:31, “index”:256, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“netro”, “command”:“answer”, “label”:31, “index”:256, “status”:“started”} {“class”:“netro”, “command”:“answer”, “label”:31, “index”:256, “status”:“progress”, “progress”:50} {“class”:“netro”, “command”:“answer”, “label”:31, “index”:256, “status”:“completed”, “warning”:“recovery of 11366 is failed;”}
Прошивки					

Класс команды (ключ “class”)	Мнемоника (ключ “command”)	Описание	Необходимые параметры	Возможные параметры	Пример использования
netro	updateFirmware	Обновление прошивки	**type** - тип прошивки(тип: string, значения - {“modem”,“device”}) **id**(если type == “device”)- id устройства для обновления прошивки(тип: uint32) **firmware** - прошивка модема (тип: составной)	**background** - фоновый режим задачи обновления(тип:bool, по умолчанию false) **wakeup** - режим отложенного обновления по просыпанию устройства(тип:bool, по умолчанию false, игнорируется, если background == false) **delay** - задержка в секундах для старта обновления(для background == true,тип, uint32, значения от 0 до 2 в степени 24)	**Клиент** {“background”:false, “class”:“netro”, “command”:“updateFirmware”, “delay”:0, “firmware”: {“high.dev.bin”:“bXbD83…”, “low.dev.bin”:“Bw4Ri3kwf…”}, “id”:8755, “label”:29, “type”:“device”, “wakeup”:false} **Ответы:** {“class”:“netro”, “command”:“answer”, “label”:29, “index”:254, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“netro”, “command”:“answer”, “label”:29, “index”:254, “status”:“started”} {“class”:“netro”, “command”:“answer”, “label”:29, “index”:254, “status”:“progress”, “progress”:0, “warning”:“Firmware update started;”} {“class”:“netro”, “command”:“answer”, “label”:29, “index”:254, “status”:“progress”, “progress”:0} сообщения с progress статусом {“class”:“netro”, “command”:“answer”, “label”:29, “index”:254, “status”:“completed”, “version”: {“product”:{“hard”:2, “variant”:1, “id”:113}, “version”:115, “address”:59392}}

2. Основные алгоритмы работы с сетью Intro3

а) первоначальная организация сети устройств: подключить исполнительные устройства к сети 220В, выполнить команду **startHello**. Результатом команды **startHello** будет список обнаруженных устройств **savedIds**. Если предварительно были просканированы qr коды исполнительных устройств взять коды для аутентификации устройств в сети и использовать их в командах **saveRemote** в качестве параметра **code**. Запустить поочередно команду **saveRemote** для каждого обнаруженного устройства. При отсутствии параметра **code** NetroManager с помощью **“warning”:“user action wait;”** сообщит о необходимости нажать на кнопку исполнительного устройства, чтобы завершить аутентификацию.

б) добавление пультов и датчиков в сеть: выполнить команду **listenToRemote**. NetroManager с помощью **“warning”:“user action wait;”** сообщит о необходимости нажать на комбинацию кнопок для аутентификации пульта/датчика в сети.(Если пользователь передумал добавлять пульт/датчик, необходимо послать сервисную команду **cancelCommand** для отмены команды **listenToRemote**) После нажатия необходимой комбинации NetroManager пришлет ответ со **“status”:“completed”** и полем **id**. Выполнить команду **saveRemote** для данного обнаруженного **id**, чтобы завершить аутентификацию. Данную команду необходимо послать не позже одной минуты после ответа на команду **listenToRemote**.

в) добавление новых исполнительных устройств в уже созданную сеть: выполняется аналогично пункту **a)**

г) создание групп: группой устройств называется такой перечень устройств, который реагирует одинаково и одновременно на одну команду **controlDevice** с полем **netro3Group**(либо на кнопку пульта). Сервер умного дома поддерживает настройку до 511 различных групп устройств(с 1 до 511). Количество групп пультов обычно варьируется от 1 до 8 в зависимости от типа пульта и настроек режима работы пульта. Для того, чтобы добавить устройство в группу, необходимо выполнить команду **writeDeviceGroup**, указав в ней номер группы(**group**), само устройство(**id**), которое хотим включить в группу, и владельца группы(**remoteId**). Если **remoteId** равен 0, то группа принадлежит серверу умному дому. Стоит отметить, что, допустим, группа 1 может быть как у сервера умного дома, так и у пультов, например, с **remoteId** #1234 либо #3646. Но это будут разные группы, так как они принадлежат разным владельцам.

д) удаление устройств из групп и очистка групп производится с помощью команды **deleteDeviceGroup**

е) группы со сценариями: представляют собой такие же группы как в пункте **г)**, но которым обычно посылается управляющая команда **script**, на которую они реагируют по разному(на другие команды такие как **up**,*down* и прочие устройства реагируют одинаково). Реакция на **script** задается в команде **writeDeviceGroup** с помощью параметра **action**. При этом, если необходимо изменить **action** с одной реакуии на

другую, нет необходимости предварительно выполнять команду **deleteDeviceGroup**, нужно всего лишь выполнить **writeDeviceGroup** той же группы но с другим полем **action**

ж) удаление устройств из сети: необходимо выполнить команду **deleteRemote**(Для полной очистки сети от всех устройств и пультов выполнить команду **resetModem**). При этом теряется вся информация в сервере умного дома об удаляемом устройстве: группы, в которых оно участвует, маршруты к нему или через него к другому устройству.

з) настройка параметров исполнительных устройств и пультов: выполняется с помощью команды **setDeviceParams**(запросить текущие значения параметров можно с помощью команды **getDeviceParams**). При настройке параметров исполнительных устройств они применяются мгновенно. В случае же с пультами(так как они большинство времени находятся в состоянии сна) для применения параметров необходимо нажать кнопку на пульте, о чем будет сообщении от NetroManager с **“warning”:“user action wait;”**. ВНИМАНИЕ: некоторые модели пультов, например 7501, 7510micro, не поддерживают настройку и запрос параметров.

и) восстановление работоспособности сети: в случае, когда устройство заведомо рабоче, но в следствии продолжительного отключения сервера утеряло с ним связь, можно восстановить его сопряженность с сервером с помощью команды **healNetwork**. Данным способом можно попробовать восстановить и сеть полностью

к) обновление устройств: для обновления устройств либо модема netro удомного длма необходимо иметь два файла прошивки(для двух областей памяти). Обновление выполняется с помощью команды **updateFirmware**

Сеть Z-Wave

1. Описание команд

Класс команды (ключ “class”)	**Мнемоника (ключ “command”)**	**Описание**	**Необходимые параметры**	**Возможные параметры**	**Пример использования**
zwave	controlDevice	Управление устройством	**id** - адрес устройства для управления(тип: число от 2 до 232) либо **group** - виртуальная группа устройств zwave(тип: число от 1 до 511) **action** - действие (тип: string, значения- {“up”, “down”, “pos”} **position** (если “action” == “pos”) - значение положения для установки (тип: float, значения: от 0 до 1)	**channel** - канал устройства для управления(по умолчанию = 1)	**НА ДАННЫЙ МОМЕНТ РАБОТАЕТ КРИВО ДЛЯ ДВУХКАНАЛЬНОГО РЕЛЕ** **Клиент** {“ack”:true, “action”:“up”, “class”:“zwave”, “command”:“controlDevice”, “id”:3, “label”:11} **Ответы:** {“class”:“zwave”, “command”:“answer”, “label”:11, “index”:2, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“zwave”, “command”:“answer”, “label”:11, “index”:2, “status”:“started”} {“class”:“zwave”, “command”:“answer”, “label”:11, “index”:2, “status”:“completed”}
zwave	getDevicePosition	Запрос положения устройства	**id** - адрес устройств для запроса (тип: uint8)	**channel** - канал устройства(по умолчанию = 1 для одноканальных устройств)	**Клиент** {“channel”:1, “class”:“zwave”, “command”:“getDevicePosition”, “id”:3, “label”:23} **Ответы:** {“class”:“zwave”, “command”:“answer”, “label”:23, “index”:1, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“zwave”, “command”:“answer”, “label”:23, “index”:1, “status”:“started”} {“class”:“zwave”, “command”:“answer”, “label”:23, “index”:1, “status”:“completed”, “Switch”:{“metering”:“”, “readAllow”:1, “writeAllow”:1, “value”:true}}

Класс команды (ключ “class”)	**Мнемоника (ключ “command”)**	**Описание**	**Необходимые параметры**	**Возможные параметры**	**Пример использования**
zwave	listenToRemote	Запись нового устройства в сеть Z-Wave			**Клиент** {“class”:“zwave”, “command”:“listenToRemote”, “label”:8} **Ответы:** {“class”:“zwave”, “command”:“answer”,“label”:8, “index”:1,“status”:“accepted”,“queue”:0, “waitTime”:0} {“class”:“zwave”, “command”:“answer”, “label”:8, “index”:1, “status”:“started”} {“class”:“zwave”, “command”:“answer”, “label”:8, “index”:1, “status”:“progress”, “progress”:100, “warning”:“user action wait;”} {“class”:“zwave”, “command”:“answer”, “label”:8, “index”:1, “status”:“completed”, “remoteID”: {“id”:2, “class”:“FGFS101 Zwave+ Flood Sensor”}}
zwave	deleteRemote	Удаление из сети Z-wave устройства(очистка всякой информации об устройстве на сервере)		**id** - адрес сохраненного устройства zwave(если не указано, пользователь должен будет нажать три раза на кнопку устройства, которое хочет удалить) (тип: uint8)	**Клиент** {“class”:“zwave”, “command”:“deleteRemote”, “label”:8} **Ответы:** {“class”:“zwave”, “command”:“answer”, “label”:8, “index”:1, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“zwave”, “command”:“answer”, “label”:8, “index”:1, “status”:“started”} {“class”:“zwave”, “command”:“answer”, “label”:8, “index”:1, “status”:“progress”, “progress”:100, “warning”:“user action wait;”} {“class”:“zwave”, “command”:“answer”, “label”:8, “index”:1, “status”:“completed”, “id”:3}

Класс команды (ключ “class”)	**Мнемоника (ключ “command”)**	**Описание**	**Необходимые параметры**	**Возможные параметры**	**Пример использования**
zwave	getDeviceParams	Запрос параметров устройства	**id**(тип uint8) Хотя бы один из возможных параметров запроса	**wakeup** - фоновая задача по триггеру перехода устройства в режим бодрствования из режима сна(тип bool, по умолчанию false) **Power** - мощность потребляемая устройством в данный момент времени (измеряется в Вт, тип: bool) **Energy** - энергия потребленная от сети за период времени (измеряется в кВт/ч,тип: bool) **ControlChannels** - Устройство имеет один канал управления, можно подключить только одну нагрузку. Только для розеточного модуля FIBARO WALL PLUG FGWPE/F-101 v.2.1 - v.2.3 **Wake-up Interval** - интервал выхода из сна (тип: bool) **Insensitiveness to temperature changes.** - чувствительность к изменению температуры (для Flood Sensor). При изменении изменяемого значения на эту величину приведет к передаче сообщения контроллеру(тип: bool). **Luminance report - threshold** - настройка значения освещенности при изменении на которое будет отправлено сообщение (Motion Sensor) (Тип: bool). **Temperature report - threshold** - настройка значения температуры при изменении на которое будет отправлено сообщение (Motion Sensor)(Тип: bool). **Time period between reports on power load and energy consumption.** - время периодической отправки отчетов мощности и энергопотребления для Wall Plug. **Standard power load reporting** - стандартный отчет о мощности для Wall Plug. Параметр определяет на сколько должна измениться мощность нагрузки, чтобы отправить отчет контроллеру. По умолчанию установлено - 15%. **Battery Level** - уровень заряда элемента питания. **Library Version** - версия библиотеки. **Protocol Version** - версия протокола. **Application Version** - версия приложения.	**Клиент** {“Battery Level”:true, “Wake-up Interval”:true, “class”:“zwave”, “command”:“getDeviceParams”, “id”:2, “label”:13} **Ответы:** {“class”:“zwave”, “command”:“answer”, “label”:13, “index”:1, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“zwave”, “command”:“answer”, “label”:13, “index”:1, “status”:“started”} {“class”:“zwave”, “command”:“answer”, “label”:13, “index”:1, “status”:“progress”, “progress”:50, “warning”:“user action wait;”} {“class”:“zwave”, “command”:“answer”, “label”:13, “index”:1, “status”:“completed”, “Wake-up Interval”: {“metering”:“Seconds”, “readAllow”:1, “writeAllow”:1, “value”:21600}, “Battery Level”: {“metering”:“%”, “readAllow”:1, “writeAllow”:0, “value”:73}}

Класс команды (ключ “class”)	**Мнемоника (ключ “command”)**	**Описание**	**Необходимые параметры**	**Возможные параметры**	**Пример использования**
zwave	setDeviceParams	Установка параметров устройства	**id**(тип uint8) Хотя бы один из возможных параметров запроса	**wakeup** - фоновая задача по триггеру перехода устройства в режим бодрствования из режима сна(тип bool, по умолчанию false) **Wake-up Interval** - интервал выхода из сна для Motion Sensor (type: uint16, 0 - датчик не просыпается, допустимые значения 0 - 65535 сек). Заводская настройка - 7200 сек. **Wake-up Interval** - интервал выхода из сна для Door Sensor (type: {“value”:uint32}, 0 - датчик не просыпается, допустимые значения 3600 - 64800 сек.) **Wake-up Interval** - интервал выхода из сна для Flood Sensor (type: {“value”:uint32}, 0 - датчик не просыпается, допустимые значения 60 - 86400 сек) **Insensitiveness to temperature changes.** - чувствительность к изменению температуры (для Flood Sensor). При изменении изменяемого значения на эту величину приведет к передаче сообщения контроллеру. type={“value”:uint16}. Possible parameter settings:1 – 1000 [each 0.01oC] [0.01oC – 10.00oC]. **Luminance report - threshold** - настройка значения освещенности при изменении на которое будет отправлено сообщение (Motion Sensor) Тип {“value”:uint16}. 0 - освещенность не отправляется. Available settings: 0 - 32767 (1 - 32767 lux; 0 = reports are not sent). Заводская настройка - 200 lux. **Temperature report - threshold** - настройка значения температуры при изменении на которое будет отправлено сообщение (Motion Sensor). type: {“value”:uint8}. Available settings: 0 - 255 (0.1 - 25.5C; 0 = reports are not sent) Default setting: 10 (1C)	**Клиент** {“Temperature report - threshold”: {“value”:10}, “Wake-up Interval”: {“value”:7200}, “channel”:1, “class”:“zwave”, “command”:“setDeviceParams” ,“id”:2, “label”:14} {“class”:“zwave”, “command”:“answer”, “label”:14, “index”:2, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“zwave”, “command”:“answer”, “label”:14, “index”:2, “status”:“started”} {“class”:“zwave”, “command”:“answer”, “label”:14, “index”:2, “status”:“progress”, “progress”:50, “warning”:“user action wait;”} {“class”:“zwave”, “command”:“answer”, “label”:14, “index”:2, “status”:“completed”, “Wake-up Interval”: {“set”:true}}
zwave	deleteDeviceGroup	Удаление группы из устройства	**id** - устройство, которому предназначена команда (тип: uint8, **deleteGroup** - группа для удаления (тип: uint8, значения: 1-5, либо виртуальная группа сервера от 1 до 511)	**remoteId** - id пульта, группу которого нужно удалить(тип:uint8, default = 1-сервер)	**Клиент** {“class”:“zwave”, “command”:“deleteDeviceGroup”, “deleteGroup”:2, “id”:3, “label”:7, “remoteId”:2} **Ответы:** {“class”:“zwave”, “command”:“answer”, “label”:7, “index”:2, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“zwave”, “command”:“answer”, “label”:7, “index”:2, “status”:“started”} {“class”:“zwave”, “command”:“answer”, “label”:7, “index”:2, “status”:“completed”}

Класс команды (ключ “class”)	**Мнемоника (ключ “command”)**	**Описание**	**Необходимые параметры**	**Возможные параметры**	**Пример использования**
zwave	writeDeviceGroup	Запись дополнительной группы в устройство(одной командой), либо сценария	**id** - устройство, которому предназначена команда (тип: uint8) **writeGroup** - группа для записи (тип: uint8, значения: 1-5, либо виртуальная группа сервера от 1 до 511)	**remoteId** - id пульта, группу которого нужно записать(тип:uint8, default = 1-сервер)	**Клиент** {“class”:“zwave”, “command”:“writeDeviceGroup”, “id”:3, “label”:26, “remoteId”:2, “writeGroup”:2} {“class”:“zwave”, “command”:“answer”, “label”:26, “index”:1, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“zwave”, “command”:“answer”, “label”:26, “index”:1, “status”:“started”} {“class”:“zwave”, “command”:“answer”, “label”:26, “index”:1, “status”:“completed”}
zwave	healNetwork	Ремонт маршрутов к устройствам в сети		**НА ДАННЫЙ МОМЕНТ РАБОТАЕТ КРИВО id** - конкретное устройство, маршрут к которому нужно восстановить(тип: uint8)	**Клиенты** {“class”:“zwave”, “command”:“healNetwork”, “id”:3, “label”:17} **Ответы** {“class”:“zwave”, “command”:“answer”, “label”:17, “index”:3, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“zwave”, “command”:“answer”, “label”:17, “index”:3, “status”:“started”} {“class”:“zwave”, “command”:“answer”, “label”:17, “index”:3, “status”:“progress”, “progress”:100} {“class”:“zwave”, “command”:“answer”, “label”:17, “index”:3, “status”:“completed”}
zwave	getModemParams	Запрос параметров модема	Хотя бы один из возможных параметров запроса	**version** - запрос версии модема (тип: bool) **savedIds** - список сохраненных в модеме устройств/ пультов(тип: bool)	**Клиент** {“class”:“zwave”, “command”:“getModemParams”, “label”:15, “savedIds”:true, “version”:true} **Ответы:** {“class”:“zwave”, “command”:“answer”, “label”:15, “index”:1, “status”:“completed”, “version”: {“value”:“Z-Wave 4.05”}, “savedIds”:[{“id”:1, “route”:true, “sleep”:“unknown”, “saved”:true, “class”:“ZME_UZB1 USB Stick”}, {“id”:2, “route”:true, “sleep”:true, “saved”:true, “class”:“FGFS101 Zwave+ Flood Sensor”}, {“id”:3, “route”:true, “sleep”:“unknown” ,“saved”:true, “class”:“FGS222 Double Relay Switch 2×1.5kW”}]}
zwave	setModemParams	Настройка параметров модема	Хотя бы один из возможных параметров запроса	**frequency** - настройка частоты работы zwave модема(тип составной:{“country”:string(“EU”,“RU”,“US” и некоторые другие значения)})	**Клиент** {“class”:“zwave”, “command”:“setModemParams”, “label”:15, “frequency”:{“country”:“RU”}} **Ответы:** {“class”:“zwave”, “command”:“answer”, “label”:15, “index”:1, “status”:“completed”, “frequency”: {“set”:true}}
zwave	resetModem	Сброс настроек и памяти модема(сброс всей информации о сети zwave)			**Клиент** {“class”:“zwave”, “command”:“resetModem”, “label”:9} **Ответы:** {“class”:“zwave”, “command”:“answer”, “label”:9, “index”:2, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“zwave”, “command”:“answer”, “label”:9, “index”:2, “status”:“started”} Сетевое событие очистки сети zwave {“class”:“zwave”, “command”:“eventCommand”, “status”:“completed”, “event”:“destroyingNetworkFinished”} {“class”:“zwave”, “command”:“answer”, “label”:9, “index”:2, “status”:“completed”}

2. Основные алгоритмы работы с сетью Z-Wave

а) добавление устройств и датчиков в сеть: выполнить команду **listenToRemote**. NetroManager с помощью **“warning”:“user action wait;”** сообщит о необходимости нажать на комбинацию кнопок(обычно это быстрое троекратное нажатие программирующей кнопки на устройстве) для аутентификации устройства/датчика в сети.(Если пользователь передумал добавлять устройство/датчик, необходимо послать сервисную команду **cancelCommand** для отмены команды **listenToRemote**) После нажатия необходимой комбинации NetroManager пришлет ответ со **“status”:“completed”** и полем **id**. Устройство/датчик будет автоматически сохранен(о) в сервере умного дома

б) создание ассоциативных групп: Каждое исполнительное устройство Z-Wave поддерживает от 3 до 5 ассоциативных групп(согласно инструкции на устройство). На каждую ассоциативную группу можно добавить до 5 управляющих устройств(датчиков, пультов, прочего). Запись в ассоциативную группу устройства производится с помощью команды **writeDeviceGroup**

в) удаление устройств из ассоциативных групп производится с помощью команды **deleteDeviceGroup**

г) создание виртуальных групп сервера для управления группой с помощью **controlDevice** с адресацией **group**: ДЛя создания виртуальной группы **remoteId** должен быть равен 1 либо 0, номер группы от 1 до 511.

д) удаление устройств из сети: необходимо выполнить команду **deleteRemote**(Для полной очистки сети от всех устройств и пультов выполнить команду **resetModem**). При этом теряется вся информация в сервере умного дома об удаляемом устройстве: группы, в которых оно участвует, маршруты к нему или через него к другому устройству. Для удаления датчиков требуется нажать кнопку на датчике для подтверждения, либо задать явно **id** в команде, если датчик либо устройство физически не доступно(сломано, утеряно и проч.)

е) настройка параметров исполнительных устройств и датчиков: выполняется с помощью команды **setDeviceParams**(запросить текущие значения параметров можно с помощью команды **getDeviceParams**). При настройке параметров исполнительных устройств они применяются мгновенно. В случае же с датчиками(так как они большинство времени находятся в состоянии сна) для применения параметров необходимо нажать кнопку на датчике, о чем будет сообщении от NetroManager с **“warning”:“user action wait;”**.

ж) восстановление работоспособности сети: в случае, когда устройство заведомо рабоче, но в следствии продолжительного отключения сервера утеряло с ним связь, можно восстановить его сопряженность с сервером с помощью команды healNetwork. Данным способом можно попробовать восстановить и сеть полностью

Сеть GSM

1. Описание команд

Класс команды (ключ “class”)	**Мнемоника (ключ “command”)**	**Описание**	**Необходимые параметры**	**Возможные параметры**	**Пример использования**
gsm	sendSMS	Отправить СМС адресату	**phone** - номер телефона в международном формате(тип: string, например “+375297000000”) **text** - текст СМС сообщения(тип string)	**smsCenter** - номер центра смс(тип string)	**Клиент** {“class”:“gsm”, “command”:“sendSMS”, “label”:15, “phone”:“+375297000000”, “text”:“Привет”} **Ответы:** {“class”:“gsm”, “command”:“answer”, “label”:15, “index”:6, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“gsm”, “command”:“answer”, “label”:15, “index”:6, “status”:“started”} {“class”:“gsm”, “command”:“answer”, “label”:15, “index”:6, “status”:“progress”, “progress”:33} {“class”:“gsm”, “command”:“answer”, “label”:15, “index”:6, “status”:“progress”, “progress”:66} {“class”:“gsm”, “command”:“answer”, “label”:15, “index”:6, “status”:“completed”}
gsm	sendUSSD	Отправить USSD	**ussd** - (тип: string, например “*100#”)		**Клиент** {“class”:“gsm”, “command”:“sendUSSD”, “label”:16, “ussd”:“*100#”} **Ответы** {“class”:“gsm”, “command”:“answer”, “label”:16, “index”:7, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“gsm”, “command”:“answer”, “label”:16, “index”:7, “status”:“started”} {“class”:“gsm”, “command”:“answer”, “label”:16, “index”:7, “status”:“completed” ,“USSD Answer”:“IElz=”}

Класс команды (ключ “class”)	Мнемоника (ключ “command”)	Описание	Необходимые параметры	Возможные параметры	Пример использования
gsm	getModemParams	Запрос параметров модема	Хотя бы один из возможных параметров запроса	**gsmStatus** - Наличие сим карты и зарегистрированность (тип: bool) **version** - запрос версии модема (тип: bool) **signalLevel** - уровень сигнала (тип: bool)	**Клиент** {“class”:“gsm”, “command”:“getModemParams”, “gsmStatus”:true, “label”:13, “signalLevel”:true} **Ответы:** {“class”:“gsm”, “command”:“answer”, “label”:13, “index”:4, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“gsm”, “command”:“answer”, “label”:13, “index”:4, “status”:“started”} {“class”:“gsm”, “command”:“answer”, “label”:13, “index”:4, “status”:“progress”, “progress”:33} {“class”:“gsm”, “command”:“answer”, “label”:13, “index”:4, “status”:“progress”, “progress”:66} {“class”:“gsm”, “command”:“answer”, “label”:13, “index”:4, “status”:“completed”, “gsmStatus”: {“SIM ready”:true, “protection”:“none”, “Operator name”:“life:) BY”}, “signalLevel”: {“value”:-56}}
gsm	resetModem	Перезагрузка gsm модема			**Клиент** {“class”:“gsm”, “command”:“resetModem”, “label”:17} **Ответы:** {“class”:“gsm”, “command”:“answer”, “label”:17, “index”:8, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“gsm”, “command”:“answer”, “label”:17, “index”:8, “status”:“started”} {“class”:“gsm”, “command”:“answer”, “label”:17, “index”:8, “status”:“progress”, “progress”:50} {“class”:“gsm”, “command”:“answer”, “label”:17, “index”:8, “status”:“completed”}
gsm	clearSecrets	Сброс пароля заблокированной sim карты	Хотя бы один из параметров:PIN,PUK	**PIN**- пин-код(тип string) **PUK**-puk код(тип string)	**Клиент** {“PIN”:“8121”, “class”:“gsm”, “command”:“clearSecrets”, “label”:12} **Ответы:** {“class”:“gsm”, “command”:“answer”, “label”:12, “index”:3, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“gsm”, “command”:“answer”, “label”:12, “index”:3, “status”:“started”} **событие о разблокировке sim-карты** {“class”:“gsm”, “command”:“eventCommand”, “status”:“completed”, “event”:“unlocked”} {“class”:“gsm”, “command”:“answer”, “label”:12, “index”:3, “status”:“completed”}

2. Основные алгоритмы работы с сетью GSM

а) NetroManager автоматически запрашивает состояние заблокированности sim-карты при ее наличии в сервере и сообщает клиентам данную информацию с помощью команды gsm **eventCommand**(о необходимости ввода pin(**event=“lockedByPIN”**) либо puk(**event=“lockedByPUK”**) кода)

б) Клиент в случае заблокированности sim-карты с помощью команды **clearSecrets** сбрасывает pin/puk код карточки. Карточка будет полностью разблокированной до тех пор, пока пользователь заново не задаст pin код(например, через мобильный телефон)

в) отправка sms, ussd возможна только на разблокированной карточке

г) с помощью **getModemParams** можно запросить качество сигнала связи с gsm. При качестве связи хуже -96dB отправка смс может не работать

Команды для управления периферией

1. Описание команд

Класс команды (ключ “class”)	**Мнемоника (ключ “command”)**	**Описание**	**Необходимые параметры**	**Возможные параметры**	**Пример использования**
periph	setModemParams	Настройки переферии модема	**Хотя бы один из опциональных параметров**	**WPSConnectionTimer** - время отображения wps подключения к роутеру(тип: составной) **showTime** - статус отображения часов на экране сервера в ждущем режиме(тип: составной)	
periph	getModemParams	Запрос основной информации о мненджере умного дома		**showTime** - статус отображения часов на экране сервера в ждущем режиме(тип: bool) **WPSConnectionTimer** -текущее время таймера wps при подключении к роутеру(тип: bool) **ethernet** - статус ethernet-сети(тип: bool) **wifi** - статус wifi-сети(тип: bool) **version** - версия модема periph (GUI) (тип: bool)	**Клиенты**{“class”:“periph”, “command”:“getModemParams”, “version”:true, “showTime”:true, “WPSConnectionTimer”:true, “ethernet”:true, “wifi”:true} **Ответы:**“class”:“periph”, “command”:“answer”, “label”:3, “index”:65, “status”:“completed”, “WPSConnectionTimer”:{“value”:0}, “ethernet”:{“ip”:“192.168.1.50”, “status”:“on”}, “showTime”:{“value”:true}, “version”:{“value”:“0.10-5”}, “wifi”:{“status”:“off”}}
periph	setImage	Установка изображения GUI	**image** - картинка (тип base64_string)	**time** - время отображения (тип: int, default - 4s) **mainText** - основная надпись (тип base64_string) **extText** - вторая надпись (тип base64_string) **algCancel** - алгоритм значимости. Будут ли выведены другие изображения во время показа текущего (тип: bool)	

2. Основные алгоритмы работы с командами управления периферией и gui

Сервисные команды

1. Описание команд

Класс команды (ключ “class”)	**Мнемоника (ключ “command”)**	**Описание**	**Необходимые параметры**	**Возможные параметры**	**Пример использования**

Класс команды (ключ “class”)	**Мнемоника (ключ “command”)**	**Описание**	**Необходимые параметры**	**Возможные параметры**	**Пример использования**
service	cancelCommand	Отмена выполнения команды	**index** - внутренний индекс команды для отмены (тип: uint32)		**Клиент** {“class”:“netro”, “command”:“listenToRemote”, “label”:1} **Ответы** {“class”:“netro”, “command”:“answer”, “label”:1, “index”:259, “status”:“accepted”, “queue”:0, “waitTime”:0} {“class”:“netro”, “command”:“answer”, “label”:1, “index”:259, “status”:“started”} {“class”:“netro”, “command”:“answer”, “label”:1, “index”:259, “status”:“progress”, “progress”:66, “warning”:“user action wait;”} **Клиент** {“class”:“service”, “command”:“cancelCommand”, “index”:259, “label”:35} **Ответы** {“class”:“service”, “command”:“answer”, “label”:35, “index”:261, “status”:“completed”} {“class”:“netro”, “command”:“answer”, “label”:1, “index”:259, “status”:“canceled”}
service	getManagerInfo	Запрос основной информации о мненджере умного дома		**netroMap** - карта netro сети(тип: bool) **commandQueue** - очередь команд netromanager **modems** -поддерживаемые менеджером модемы и их состояние **ethernet** - статус ethernet-сети **wifi** - статус wifi-сети **power** - статус питания сервера	**Клиент** {“class”:“service”, “command”:“getManagerInfo”, “commandQueue”:true, “ethernet”:true, “label”:32, “wifi”:true} **Ответ:** {“class”:“service”, “command”:“answer”, “label”:32, “index”:257, “status”:“completed”, “version”:“0.64-7”, “zwaveLibrary”:“1.4.-1”, “ethernet”:{“status”:“unknown”}, “wifi”: {“status”:“unknown”}, “commandQueue”:[{“class”:“netro”, “command”:“listenToRemote”, “label”:1, “index”:254, “queue”:0}]}
service	setManagerParams	Настройка параметров менеджера(для отладки, не для использования в продакшне)	Хотя бы один из параметров	**backgroundPeriod** - период фонового обновления для устройств(тип: {“value”:uint32}) **backgroundWindow** - временное окно фонового обновления(тип: {“value”:uint32}) **WPSConnectionTimer** - таймер отслеживания скрипта запуска wps(тип: {“value”:bool,“time”:uint32(в секундах)}) **wifiApConfig** - настройка точки доступа(тип: {“channel”:uint8(1-14),“txPower”:int})	**Клиенты** {“backgroundPeriod”: {“value”:6000}, “backgroundWindow”:{“value”:3000}, “class”:“service”, “command”:“setManagerParams”, “label”:33} **Ответ** {“class”:“service”, “command”:“answer”, “label”:33, “index”:258, “status”:“completed”, “backgroundPeriod”: {“set”:true} ,“backgroundWindow”: {“set”:true}}

2. Основные алгоритмы работы с сервисными командами

а) Для того, чтобы отменить текущую исполняемую команду, предназначенную одному из модемов netro,zwave либо gsm, необходимо отправить **cancelCommand** с полем **index** той команды, которую нужно отменить. Таким же способом можно отменять команды, которые находятся в очереди и ожидают своего выполнения.

б) чтобы узнать состояние очередей модемов(какие команды сейчас в очереди и на выполнении), необходимо запросить командой **getManagerInfo** с параметром **commandQueue**. Полученный список при желании можно отменить с помощью команды **cancelCommand**

Ответы от NetroManager клиенту

Класс команды (ключ “class”)	**Мнемоника (ключ “command”)**	**Описание**	**Необходимые параметры**	**Возможные параметры**
netro либо service	answer	Быстрый ответ на команду	**index** - уникальный внутренний индекс, присвоенный команде в менеджере netro (тип: uint32) **status** - результат выполнения/обработки команды (тип: string, значение {“accepted”}) **queue** - номер в очереди на обработку (тип: uint32) **waitTime** - приблизительное время ожидания до выполнения данной команды, в мс (тип:uint32) **label** - метка, которая содержалась в команде, на которую дается ответ	
netro либо service	answer	Стандартный ответ на команду	**index** - уникальный внутренний индекс, присвоенный команде в менеджере netro (тип: uint32) **status** - результат выполнения/обработки команды (тип: string, значения {“error”, “started”, “progress”, “completed”,“cancelled”}) **progress** (если “status” == progress) - прогресс выполнения команды в % (тип uint8, значение: 0-100) **error** (если “status” == “error”) - тип ошибки при выполнении команды (тип: string, значения {“bad param”, “bad command”,“no param”, “modem timeout”, “algorithmic error”, “modem answer error”, “modem inner error”, “modem incorrect answer”, “device timeout”, “no route”,“bad firmware”,“device error”}) **label** - метка, которая содержалась в команде, на которую дается ответ	**warning** - требующая внимания строка от SHNetroManager, касающаяся выполнения данной команды **errorDescription** - описание некоторых неочевидных ошибок
netro либо service	answer	Составной ответ на команду	**index** - уникальный внутренний индекс, присвоенный команде в менеджере netro (тип: uint32) **status** - результат выполнения/обработки команды (тип: string, значения {“error”, “completed”}) **error** (если “status” == “error”) - Общий тип ошибки при выполнении команды (тип: string, значения {**“bad param”** - неверный тип одного из параметров в команде от клиента, параметр не попадает в допустимый диапазон значений, **“no param”** - в команде отсутсвует параметр, который обязателен согласно API, **“modem timeout”** - таймат ожидания ответа от модема , **“algorithmic error”** - ошибка посреди алгоритма выполнения команды менеджером, “modem answer error”(на данный момент не используется), **“modem inner error”** - внутренний сбой модема , **“modem incorrect answer”** - модем ответил не точ, что мы ожидали, сбой usart интерфейса, **“device timeout”** - таймату ожидания ответ от устройства,**“bad firmware”**- в команде передано плохое ПО,**“device error”** - ошибка выполнения команды на самом устройстве, **“invalid device”** - обращение к несуществующему устройству, **“command denied”** - команда запрещена к исполнению на устройстве}) **Названия устанавливаемых или запрашиваемых частей** - составный(см. следующую таблицу) **label** - метка, которая содержалась в команде, на которую дается ответ	**warning** - требующая внимания строка от SHNetroManager, касающаяся выполнения данной команды **errorDescription** - описание некоторых неочевидных ошибок

Информация о событиях от NetroManager клиенту

Класс команды (ключ “class”)	**Мнемоника (ключ “command”)**	**Описание**	**Необходимые параметры**	**Возможные параметры**	**Пример использования**

Класс команды (ключ “class”)	**Мнемоника (ключ “command”)**	**Описание**	**Необходимые параметры**	**Возможные параметры**	**Пример использования**
netro	snifferCommand	Команда, словленная модемом с эфира Netro(от пульта, ИУ, датчика)	**id** - адрес сохраненного пульта/устройства (тип: число от 1 до 16777215) **action** - тип команды от пульта/устройства (тип: string, значения- {“up”, “down”, “stop”, “loop”, “comfort1”, “comfort2”, “script”, “pos”,“hello”}) **status** - прогресс обработки команды с помощью SHNetroManager(тип string, значения “completed”,“progress”)	**group** - группа пульта (обязателен для netro, число от 1 до 8 в зависимости от типа пульта) **channel** - канал устройства(обязателен для многоканальных intro3 устройств) (тип uint8) **position** - текущая позиция устройства (тип float: значения от 0 до 1) **time** - время нажатия/отжатия кнопки пульта(тип: string) **failedIds** - массив номеров устройств, не прореагировавших на команду, которую отправил NetroManager в сеть (массив чисел от 1 до 16777215, [устройство1,устройство2,..,устройствоn])	**NetroManager клиентам:** {“class”:“netro”, “command”:“snifferCommand”, “status”:“completed”, “id”:214019, “group”:1, “action”:“up”, “time”:“0sec on”}
netro	eventCommand	Событие от модема	**event** - событие (тип: string значения- {“batteryPowerOn”,“batteryLowPower”, “acPowerOn”,“buildingNetworkStarted”, “buildingNetworkFinished”, “destroyingNetworkStarted”, “destroyingNetworkFinished”,“touchPadPress”, “touchPadRelease”, “touchPadSlide”, “powerButtonPress”, “powerButtonRelease”}	-	**NetroManager клиентам:** {“class”:“netro”, “command”:“eventCommand”, “status”:“completed”, “event”:“powerButtonPress”}
netro	alarmCommand	Сообщение-тревога (сигнализаия о критическом событии)	**alarm** - (тип: string значения- {“highTemperature”, “lowVoltage”, “lowBattery”, “modemFlashError”(отладочный),“deviceCleared”,“deviceSetDefault”})	**id** - номер устройства, от которого поступила тревога	**NetroManager клиентам:** {“class”:“netro”, “command”:“alarmCommand”, “status”:“completed”, “id”:8755, “alarm”: “highTemperature”}
netro	sensorCommand	Показание датчика, словленное модемом с эфира Netro(ИУ, датчика)	**id** - id датчика/устройства (тип: uint32) **НАЗВАНИЕ ДАТЧИКА** - тип сработавшего датчика (тип: составной, зависит от датчика внутреннее поле **value**, поле **metering**)	**group** - группа датчика (обязателен для netro, тип: uint16) **channel** - канал устройства(обязателен для netro многоканальных)(тип uint8)	
netro	networkStatusCommand	Команда состояния сети intro3	**id** - id датчика/устройства/пульта intro3 (тип: число от 1 до 16777215) **ТИП СОСТОЯНИЯ СЕТИ** - один из типов состояния(dead,sleep,saved и проч.) (тип: bool)	**dead** - мертвость/недоступность в эфире устройства(type: bool) **sleep** - устройство находится в состоянии сна и не слушает радиоэфир (type: bool) **saved** - сохранено ли устройство либо удалено из сети	**NetroManager клиентам:** {“class”:“netro”, “command”:“networkStatusCommand”, “status”:“completed”, “id”:2, “saved”:true}

Класс команды (ключ “class”)	Мнемоника (ключ “command”)	Описание	Необходимые параметры	Возможные параметры	Пример использования
zwave	snifferCommand	Команда, словленная модемом с эфира Z-wave(от ИУ, датчика)	**id** - id устройства z-wave (тип: число от 2 до 232) **action** - тип события, произошедший на устройстве(тип: string) **position** - текущая позиция устройства (тип float: значения от 0 до 1)		**NetroManager клиентам:** {“class”:“zwave”, “command”:“snifferCommand”, “status”:“completed”, “id”:3, “channel”:1, “action”:“pos”, “position”:0}
zwave	alarmCommand	Сообщение-тревога (сигнализаия о критическом событии)	**alarm** : {“тип_тревоги”:значение(число)} либо string(“deviceCleared”) **id** - номер устройства, от которого поступила тревога **channel** - канал устройства, на котором сработала тревога		**NetroManager клиентам:** {“class”:“zwave”, “command”:“alarmCommand”, “status”:“completed”, “id”:2, “channel”:1, “alarm”: {“Burglar”:3}}
zwave	sensorCommand	Показание датчика, словленное с z-wave сети	**id** - адрес датчика(число от 2 до 232)	**Power** - отчет отправляется автоматически при изменении подключенной нагрузки на 80%, 15 %, каждые 3600 с (Вт) **Energy** - отчет отправляется автоматически при изменении энергопотребления на 0,1 кВт/ч и каждые 3600 с. Только для розеточного модуля FIBARO WALL PLUG FGWPE/F-101 v.2.1 - v.2.3 **Sensor** - сообщение о срабатывании датчика (type: bool, true - датчик сработал, false - отмена тревоги) **basic** - сообщение о срабатывании датчика (type: bool, true - датчик сработал, false - отмена тревоги) **Temperature** - сообщение со значением температуры, единица измерения oC (type: short) **Luminance** - сообщение со значением освещенности, единица измерения “lux” (type: uint16). **Battery Level** - сообщение текущего заряда батареи датчика (тип {“value”:uint8(число от 0 до 100)}). **General** - сообщение о вскрытии (тип: uint8, датчик вскрыт - 255, норм. сост. - 0)	**NetroManager клиентам:** {“class”:“zwave”, “command”:“sensorCommand”, “status”:“completed”, “id”:2, “channel”:1, “Battery Level”: {“value”:73}}
zwave	paramCommand	Команда с параметром прибора z-wave	**id** - id датчика/устройства z-wave (тип: uint32) **НАЗВАНИЕ ПАРАМЕТРА** - тип параметра (тип: составной, зависит от параметра внутреннее поле **value**, поле **metering**, поле **readAllow**, **writeAllow**) **channel** - канал устройства(обязателен для z-wave многоканальных)		

Класс команды (ключ “class”)	Мнемоника (ключ “command”)	Описание	Необходимые параметры	Возможные параметры	Пример использования
zwave	networkStatusCommand	Команда состояния сети z-wave	**id** - id датчика/устройства z-wave (тип: число от 2 до 232) **ТИП СОСТОЯНИЯ СЕТИ** - один из типов состояния(dead,sleep,saved и проч.) (тип: bool)	**dead** - мертвость/недоступность в эфире устройства(type: bool) **sleep** - устройство находится в состоянии сна и не слушает радиоэфир (type: bool) **saved** - сохранено ли устройство либо удалено из сети	**NetroManager клиентам:** {“class”:“zwave”, “command”:“networkStatusCommand”, “status”:“completed”, “id”:2, “sleep”:false}
gsm	smsCommand	Команда sms от gsm модема	**phone** - номер телефона с которого пришла sms (тип: string, например “+375297000000”) **text** - содержание sms сообщения(тип base64_string)	-	**NetroManager клиентам:** {“class”:“gsm”, “command”:“smsCommand”, “status”:“completed”, “phone”:“+375297097611”, “text”:“Q3plc2Mh”}
gsm	eventCommand	gsm событие	**event** - тип события (тип: string,значения- {“lockedByPIN”, “lockedByPUK”,“unlocked”})	**retries** -количество оставшихся попыток разблоировки сим карты с помощью ПИН-кода или ПУК-кода(тип int)	**NetroManager клиентам:** {“class”:“gsm”, “command”:“eventCommand”, “status”:“completed”, “event”:“lockedByPIN”, “retries”:2}
service	eventCommand	Внутрисерверное событие	**event** - тип события (тип: string,значения- {“acPowerOn”, “batteryPowerOn”, “WPSButtonPressed”, “WPSConnectionEstablished”, “WPSConnectionFailed”, “WPSTimerStarted”, “WPSTimerStopped”, “ethernetConnected”, “ethernetDisconnected”, “wifiClientConnected”, “wifiOff”, “wifiApCreated”})	**time** - время в секундах некоторых событий **ip** - айпи для событий подключения eth и wifi	**NetroManager клиентам:** {“class”:“netro”, “command”:“eventCommand”, “status”:“completed”, “event”:“batteryPowerOn”}
periph	eventCommand	событие периферии	**event** - тип события (тип: string,значения- {“WPSButtonPressed”, “WPSConnectionEstablished”, “WPSConnectionFailed”, “ethernetConnected”, “ethernetDisconnected”, “wifiClientConnected”, “wifiOff”, “wifiApCreated”})	**ip** - айпи для событий подключения eth и wifi	**NetroManager клиентам:** {“class”:“periph”, “command”:“eventCommand”, “status”:“completed”, “event”:“wifiOff”}

Описание формата сложных параметров

Класс команды	Тип сообщения	Параметр	Состав
netro	setModemParams, getModemParams answer	intro3Channel	{**value**:uint8(от 0 до 6)}
netro	setModemParams, getModemParams answer	resetWacthDog	{**value**:uint32(0 - отключить watchdog сброса системы, от 1 до 42949672 в секундах)}
netro	getModemParams answer	modemMemory	{**freeCells**:uint16(количество свободных ячеек для записи устройств),**totalCells**:uint16(общее количество ячеек для записи устройств)} либо {**error**:string}

Класс команды	Тип сообщения	Параметр	Состав
netro	getModemParams answer, getDeviceParams answer	version	{**version**:uint8(версия ПО), **product**(версия аппаратной части устройства):{**variant**:uint8, **hard**:uint8, **id**:uint8}, **address**:uint32} либо {**error**:string}
netro	getModemParams answer	accelSensor	{x: uint16, y:uint16, z:uint16}(показания акселерометра по координатам) либо {**error**:string}
netro	getModemParams answer	lightSensor	{value: uint32}(освещенность) либо {**error**:string}
netro	getDeviceParams answer	programmedRCs	Список запрограмиированных в устройство пультов [{**rc**:uint32(значения от 0 до 16777215),**group**:uint16(значения от 1 до 1021),**action**:string(Действие по умолчанию на группу),**class**(класс пульта):string, **local**(вручную записанный пульт или с помощью программы):bool,**slot**(номер слота ответа при групповой посылке управления):uint8},{..},{..}] либо {**error**:string}
netro	getDeviceParams answer	routes	[{**id**(адрес конечного устройства на маршруте):uint32(от 1 до 16777215),**route**(номер маршрута к устройству):uint8(от 0 до 31),**hop**(на каком участке маршрута находится устройство):uint8(от 0 до 7), **local**(вручную записанный маршрут):bool},{..},{..}] либо {**error**:string}
netro	getDeviceParams answer	rssi	{**hard**(аппаратное качество связи):uint8,**soft**(программное качество связи):uint8} либо {**error**:string}
netro	getDeviceParams answer	groupContent	[{**id**:uint32(id устройства, записанного в группу),**group**:uint16,**action**:string(Действие по умолчанию на группу), **ok**(корректно ли прописана группа в устройстве):bool},{..},{..}] либо {**error**:string}
netro	getDeviceParams answer	groupsIncluded	[{**rc**:uint32(id пульта, в который записано данное устройство),**class**:string(класс пульта),**group**:uint16,**action**:string(Действие по умолчанию на группу), **ok**(корректно ли прописана группа в устройстве):bool},{..},{..}] либо {**error**:string}
netro	getDeviceParams answer restoreBackup answer saveRemote answer	idGroupsChanged	[uint32(от 1 до 16777215),uint32…uint32] - перечень ИУ, в которых изменился состав групп
netro	setDeviceParams, getDeviceParams answer	moveTime	{**value**:uint16}(от 0 до 65535 сек) либо {**error**:string}
netro	setDeviceParams, getDeviceParams answer	riseTime	{**value**:float}(от 0 до 2,5 сек, шаг 0.1) либо {**error**:string}
netro	setDeviceParams, getDeviceParams answer	dimmingStep	{**value**:float}(от 0 до 1 сек, шаг 0.01) либо {**error**:string}
netro	setDeviceParams, getDeviceParams answer	comfort1 comfort2	Хотя бы одно значение {**value**:float(от 0 до 1, с шагом 0.0001) либо “undefined” - необязательный к настройке,**brightness**:float(от 0 до 1, с шагом 0.01) либо “undefined” - необязательный к настройке,**riseTime**:float(от 0 до 1800 секунд,шаг 0,2секунды) либо “undefined” - необязательный к настройке} либо {**error**:string}

Класс команды	Тип сообщения	Параметр	Состав
netro	setDeviceParams, getDeviceParams answer	wakeup	{**period**:uint16 (от 1 до 65535, в минутах)} либо {**error**:string}
netro	setDeviceParams, getDeviceParams answer	workMode	[{ **channel**(номер канала управления устройства, на данный момент существуют только исполнительные устройства с управлением одним каналом) : uint8, **memoryMode**(режим работы с памятью, при пропадании и появлении питания 220В будет ли восстановлено прибором прошлое состояние):bool, **clockwise**(управление двигателем по часовой и против часовой стрелке):bool, **comfortOn**(при поступлении команды up выставляется уровень яркости(либо другой какой уровень), который был до прошлого выключения):bool,**mode**:string(“1,5sec” - для совместимости, не использовать,“rollShutters” - роллетный режим,“blinds” - жалюзийный режим,“infinity” - режим бексконечности(не использовать),“rollShuttersDelayed” - не реализован,“doorLock” - режиим замка,“relay” - режим реле,“dimmer” - режим диммера, “unknown” - возвращается в ответе, если текущий режим не известен)}, ..,] либо [{**channel**:uint8,**error**:string}, ..,] При установке параметра необходимо поле channel и хотя бы один подпараметр
netro	setDeviceParams, getDeviceParams answer	rcGroupMode	[{ **group**(номер группы пульта) : uint8(обычно от 1 до 8 в зависимости от типа пульта), **mode**(режим отправки команд при нажатии кнопок):string(“rollShutters” - команда для роллет,“blinds” - команда для жалюзи, “relay” - команда для реле,“dimmer” - команда для диммера,“unknown” - возвращается в ответе, если текущий режим не известен), **action**:string (“stop”, “up”, ”down”, ”comfort1”, ”comfort2”, ”loop”,”script”, ”loopSimulate”), **buttonMode**:string(“delayed”,“unfixed”,“latching”,“doubleThrow”,“unknown”), **feedback**:bool(необходимость обратной связи), **enabled**(включена ли текущая группа или заблокирована):bool}, ..,] либо [{**group**:uint8,**error**:string}, ..,] При установке параметра необходимо поле group и хотя бы один подпараметр
netro	setDeviceParams, getDeviceParams answer	buttonsMode	{**mode**:string(“singleChannel” - кнопки на пульте отправляют разные команды для одной группы,“multiChannel” - кнопки на пульте отправляют команды(чаще всего script) для разных групп)} либо {**error**:string}
netro	setDeviceParams, getDeviceParams answer	extendedScript	[{**number**:uint8,**repeat**: uint16/string (“infinity” - бесконечно, 0 - без повтора), **sequence**:[{**action**:string, **position**:float, **time**:uint16}, .., {**action**:string, **position**:float, **time**:uint16}]}, .., ..] либо [{**number**:uint8, **error**:string}, .., ..]
netro	setDeviceParams, getDeviceParams answer	temperatureSensor	{**enabled**(включен ли замер температуры):bool, **period**(период считывания показания температуры, в секундах):uint32} либо {error:string}
netro	listenToRemote answer	remoteID	{id: uint32(адрес обнаруженного устройства),class: string(класс обнаруженного устройства)}
netro	upateFirmware	firmware	{**file1_name**:base64_string(прошивка устройства, закодированная в base64),…,**filen_name**:base64_string(прошивка устройства, закодированная в base64)}
netro	startHello answer	helloIds	[{**id**:uint32(адрес обнаруженного устройства, от 1 до 16777215),**class**(класс обнаруженного устройства):string},{**id**:uint32,**class**:string} … {**id**:uint32,**class**:string}]
netro	eventCommand	event	string {“batteryPowerOn” - питание сервера от батареи,“acPowerOn” - питание сервера от 220В}
netro	snifferCommand	ids	[{“id”:uint32(адрес устройства, от 1 до 16777215),“destinationPosition”(конечное положение устройства):float(от 0 до 1 с шагом 0.01),“timeToGo”(время до достижения destinationPosition в секундах):uint16,“direction”(текущее направление движения):string(“up”,“down”)},…,{…}]
netro	getManagerInfo answer	все параметры	{**version**:string, **modems**:{“netro”:status,“zwave”:status,“gsm”:status,“bluetooth”:status} где status = {**support**:bool,**online**:bool, **netroMap**:base64_string} перечисляется поддержка нетроменеджером модемов(support) и наличие текущей связи с модемами(support)
service	getManagerInfo answer	commandQueue	список команд в очереди на выполнение для всех активных модемов [{**class**(тип модема):string, **command**(команда):string **index**(индекс команды):int, **queue**(порядковый номер команды в очереди):int, **label**(метка клиента):client_type]

Класс команды	Тип сообщения	Параметр	Состав
netro	getManagerInfo answer	ethernet	{**status**:string(“on”(подключение по ethernet активно),“off”(подключение по ethernet отсутствует),“unknown”(подключение по ethernet не было запрошено)), **ip**:string}
netro	getManagerInfo answer	wifi	{**status**:string(“ap”(текущий режим wifi- точка доступа),“client”(текущий режим wifi- подключение к роутеру),“off”(текущий режим wifi- отключен),“unknown”(режим wifi не запрашивался либо неизвестен)), **ip**:string(текущий выделенный серверу ip адрес)}
netro	getManagerInfo answer	power	{**status**:string(“ac”(питание сервера от 220в),“battery”(питание сервера от аккумулятора),“unknown”(режим питания еще не запрашивался))}
netro zwave	getModemParams answer	savedIds	список устройств в сервере [{id: uint32, saved:bool(сохранено ли устройство в сервере), route:bool(либо “unknown”, есть ли доступ к устройству), sleep:bool(либо “unknown”,опционально, находится ли устройство в состоянии сна),class: string(класс устройства)}, {….},…] либо {**error**:string}
netro zwave	setDeviceParams answer	все параметры	{**set**:true} либо {**set**:false, **error**:string} Указывается, настроен ли параметр и что могло послужить ошибкой его настройки
gsm	getModemParams answer	gsmStatus	{**SIM ready**(наличие sim карты в сервере):bool либо {**error**:string},**protection**(тип защиты sim карты):string(“none”,“PIN”,“PUK”),“Operator name”(Имя оператора связи):string}
gsm	sendUSSD answer	USSD Answer	строка в формате base64
gsm	smsCommand answer	text	строка в формате base64
periph	setModemParams, getModemParams answer	WPSConnectionTimer	{**value**:uint32 (в секундах, 0 - отключить)} либо {**error**:string}
periph	setModemParams, getModemParams answer	showTime	{**value**:bool (true -включить отображение времени на экране ожидания, false -отключить)} либо {**error**:string}
periph	getModemParams answer	ethernet	{**status**:string (“unknown”- текущее состояние неизвестно, “on” - ethernet подключен, “off” - ethernet отключен), **ip**:string} либо {**error**:string}
periph	getModemParams answer	wifi	{**status**:string (“unknown”- текущее состояние неизвестно, “client” - подключен к роутеру, “ap” - включен в режиме точки доступа, “off” - wifi отключен), **ip**:string} либо {**error**:string}