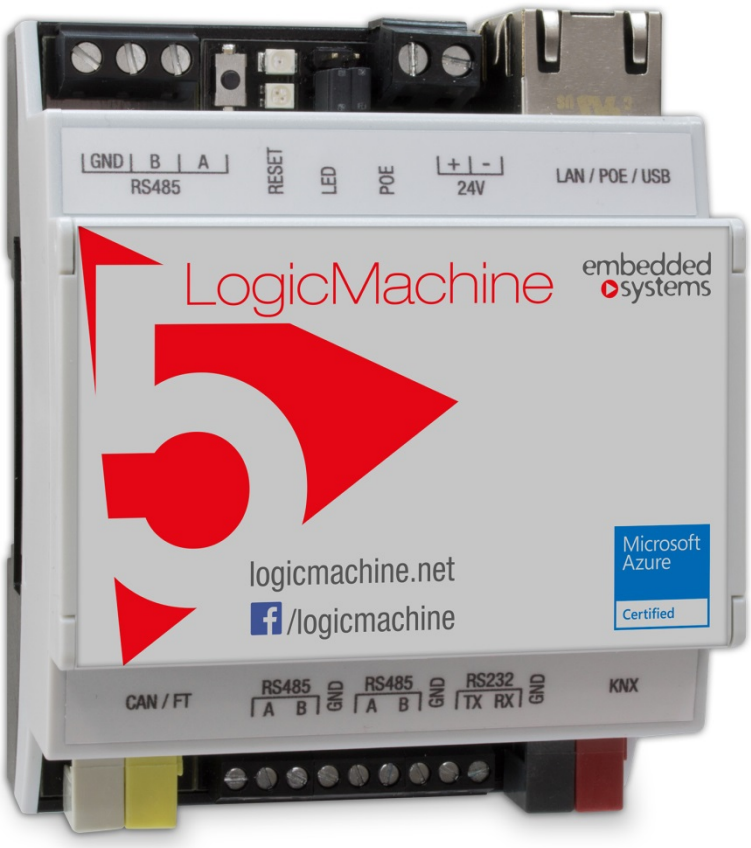
ИНСТРУКЦИЯ

по настройке взаимодействия

логического контроллера LogicMachine

и сервиса Яндекс «IoT Core»



Введение

Настоящая инструкция показывает, как обеспечить взаимодействие логического контроллера LogicMachine и устройств, подключенных к LM (например, датчиков) по протоколу MQTT к сервису Яндекс.Облако IoT Core. (IoT – Internet of Things – с английского: интернет вещей). В тексте также приводятся выдержки из документации на сервисы Яндекс.

1.1 Взаимосвязь ресурсов сервиса IoT Core

Основными элементами сервиса являются устройство[1] и реестр[2], которые могут обмениваться различными данными и командами с помощью MQTT-брокера[3].

Обмен данными происходит по протоколу MQTT версии 3.1.1. Это упрощенный сетевой протокол для взаимодействия между устройствами по принципу издатель-подписчик.

Чтобы начать обмен данными и командами между устройствами и реестрами, необходимо авторизоваться. Это можно сделать с помощью X.509-сертификатов или по логину и паролю. Замечания по подготовке программного обеспечения при том или ином способе авторизации будут описаны ниже.

Внутри сервиса устройства и реестры обмениваются данными и командами, представленными в виде сообщений с определенными топиками[4].

---

[1] Экземпляр физического устройства, например, датчика температуры, представленный в сервисе в виде имени устройства и его сертификата. Устройство создается в реестре, может отправлять телеметрические данные, получать команды и взаимодействовать с другими устройствами в реестре.

[2] Набор устройств, логически связанных между собой. Реестр может читать телеметрические данные устройств и отправлять им команды.

[3] Центральный узел для обмена MQTT-сообщениями между устройствами и реестрами. Обмен данными происходит только через брокер. Он отвечает за получение и обработку сообщений, а также контролирует их доставку клиентам (устройствам или реестрам), обеспечивая необходимый уровень качества обслуживания (QoS).

1.2 Начало работы

Устройство и реестр взаимодействуют с помощью X.509-сертификатов. Если у Вас есть сертификат, просто добавьте его к устройству в реестре. Если у Вас нет сертификата, его можно создать, например, с помощью программы OpenSSL в командной строке на компьютере:

```
$ openssl req -x509 \
-newkey rsa:4096 \
-keyout key.pem \
-out cert.pem \
-nodes \
-days 365 \
-subj '/CN=localhost'
```

Где:

- `-x509` — X.509-сертификат.
- `newkey` — алгоритм шифрования.
- `-keyout` — файл, в который будет записан приватный ключ.
- `-out` — файл, в который будет сохранен сертификат.
- `-nodes` — флаг, который устанавливается, когда не надо шифровать публичный ключ.
- `-days` — срок действия сертификата в днях.
- `-subj` — объект запроса

Дальнейшие операции выполняются в командной строке с помощью приложения Yandex CLI, либо в веб-интерфейса Яндекс.Облака. Последний вариант рассмотрен в этой инструкции.

2. Реестры.

2.1 Создание реестра

После перехода в сервис Яндекс.Облако IoT Core будет доступен первый раздел: Реестры. Необходимо перейти к созданию реестра, нажав на кнопку «Создать реестр» (рисунок 1).

[4] Тема сообщения, которая позволяет классифицировать данные для их выборочной отправки и получения устройствами и реестрами.

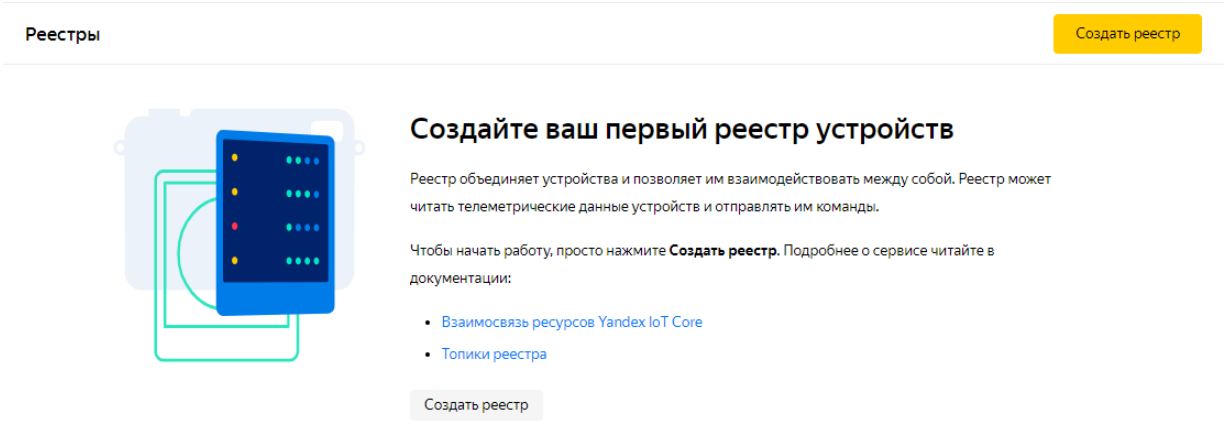


Рисунок 1 – Первый шаг – создание реестра

На странице создания реестра нужно ввести имя реестра и пароль (рисунок 2). Поля «Описание» и «Метки» созданы для удобства, их можно оставить пустыми.

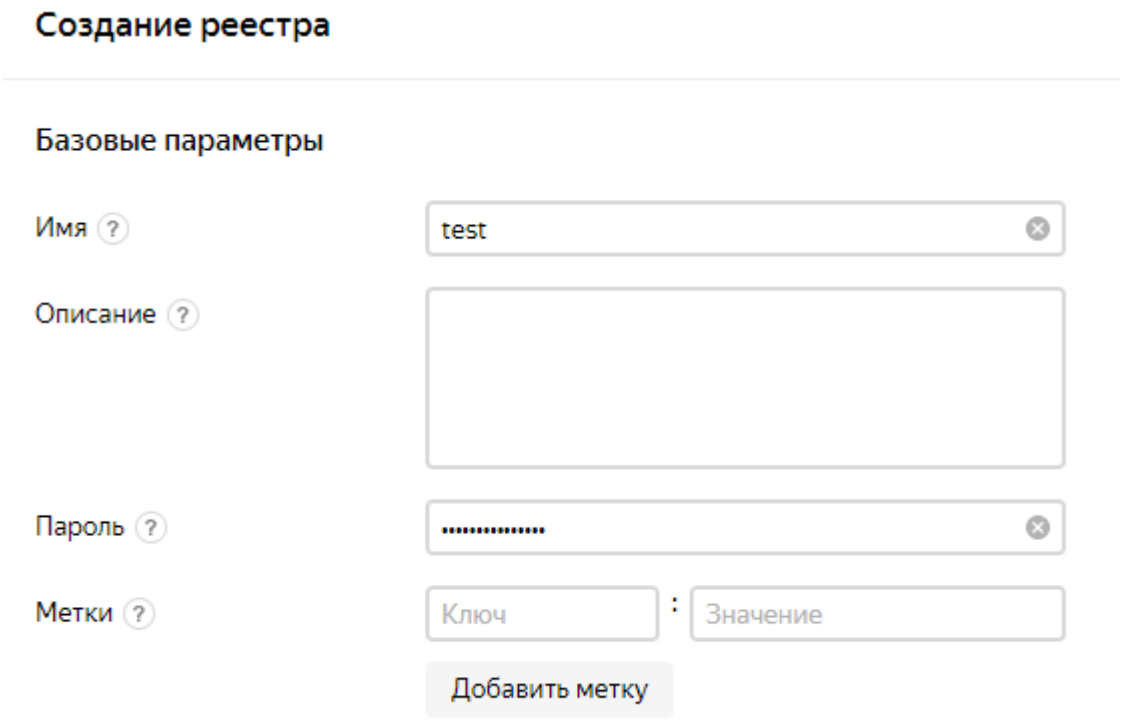


Рисунок 2 – Создание реестра в консоли (веб-интерфейс)

Далее необходимо добавить сертификат, сделанный в OpenSSL (рисунок 3).

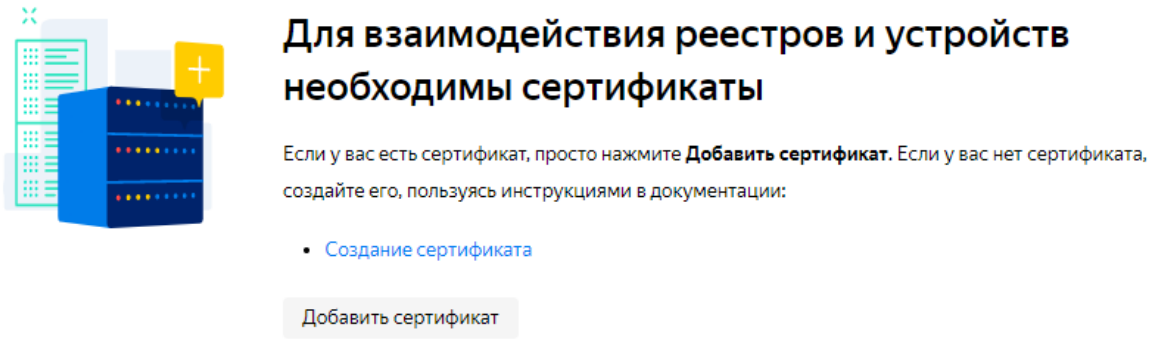


Рисунок 3 – Сообщение о необходимости добавить сертификат

После нажатия на кнопку «Добавить сертификат» станет доступна форма для его добавления (рисунок 4).

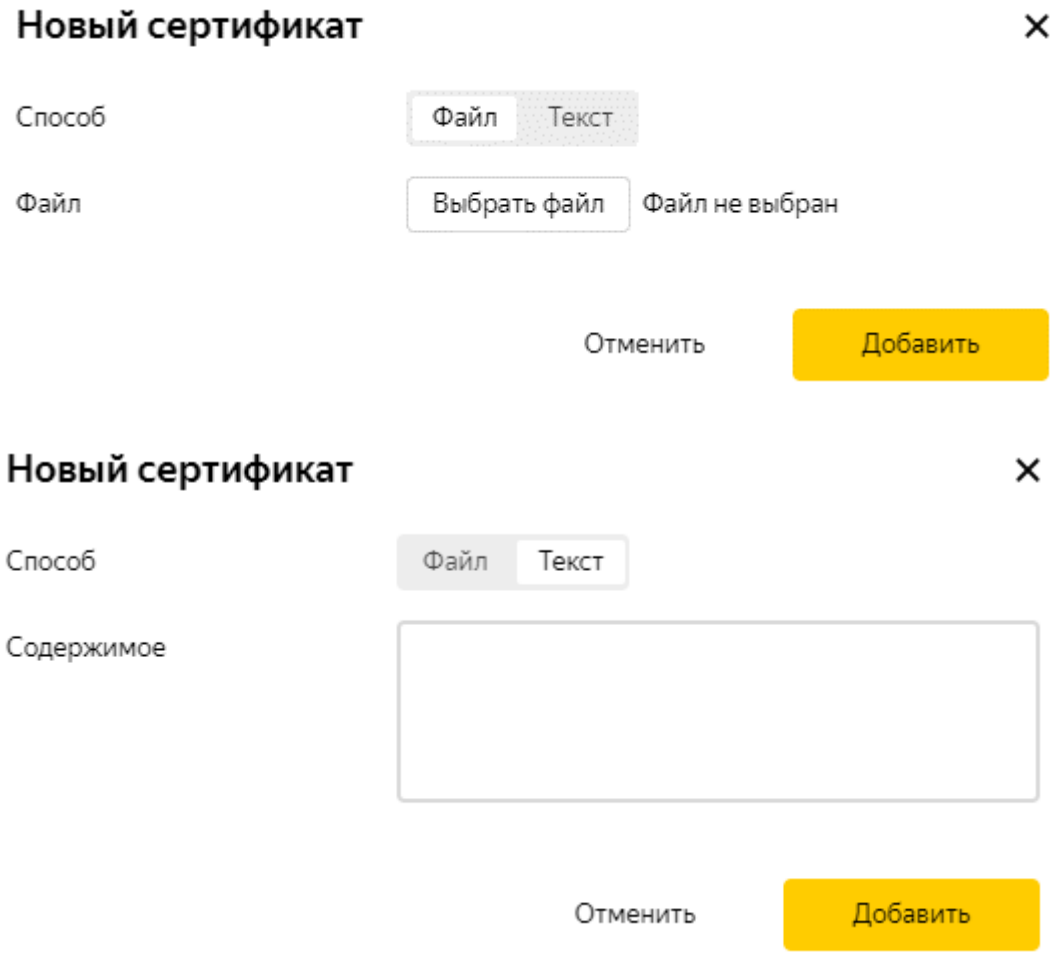


Рисунок 4 – Форма для выбора файла существующего сертификата (2 варианта добавления)

После создания реестра он появится в списке (рисунок 5):

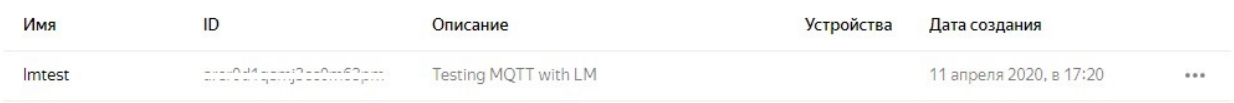


Рисунок 5 – Список реестров

2.2 Работа с реестром.

При нажатии на реестр в списке, Вы попадете в него. Во вкладке «Обзор» (рисунок 6) можно узнать ID реестра, который понадобится в скриптах, добавить новые пароли и сертификаты, а также удалить старые.

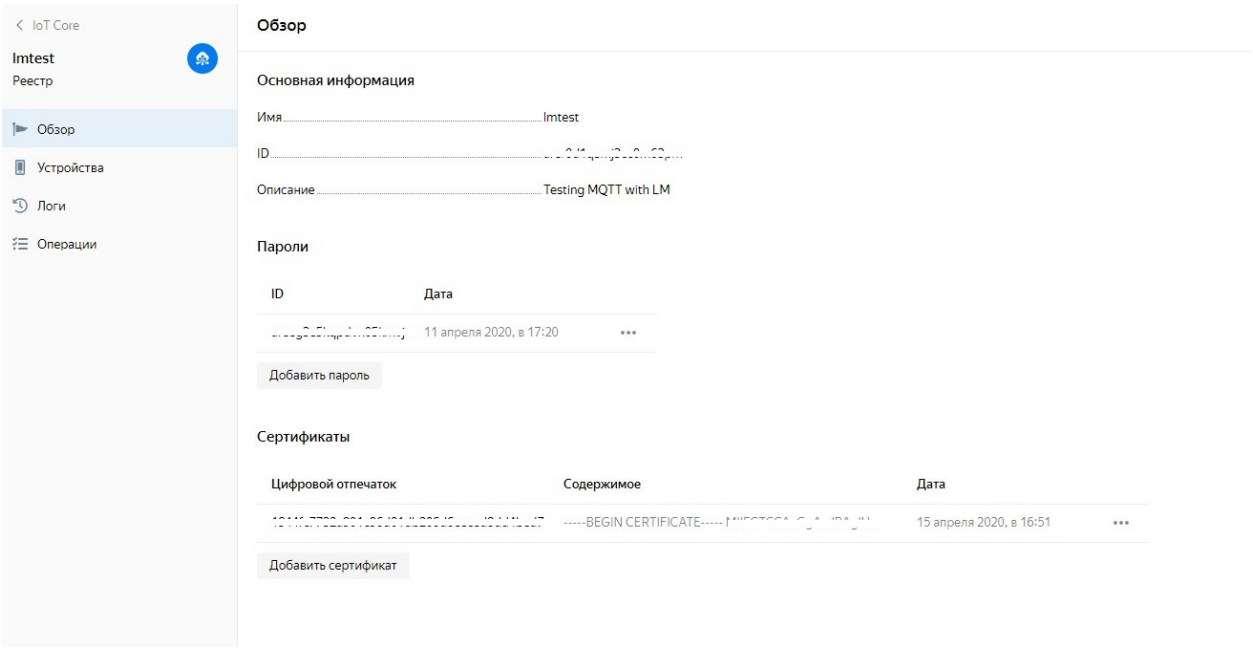


Рисунок 6 – Вкладка «Обзор» для работы с реестром

Во вкладке «Устройства» (рисунок 7) происходит добавление и изменение устройств.

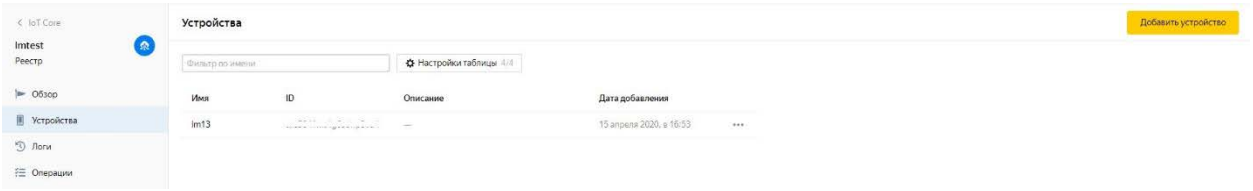


Рисунок 7 – Вкладка «Устройства»

Во вкладке «Логи» находятся сообщения о подключении к реестру с помощью MQTT (рисунок 8).

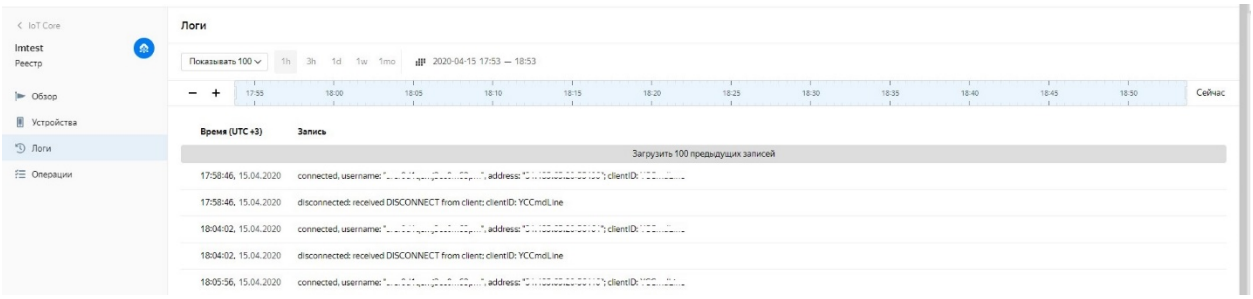


Рисунок 8 – Вкладка «Логи» для просмотра сообщений о подключении к реестру

Во вкладке «Операции» (рисунок 9) находятся сообщения о действиях с реестром: добавление сертификатов, устройств и т.п.

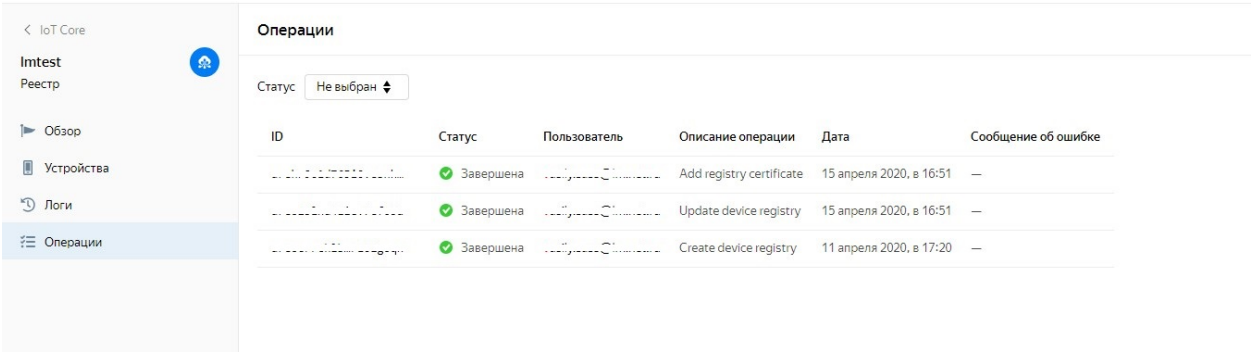


Рисунок 9 – Вкладка «Операции» о действиях с реестром

3. Создание устройства.

3.1 Создание устройства

Вернёмся к вкладке «Устройства», показанной на рисунке 7. После нажатия кнопки «Добавить устройство» Вам предлагается заполнить форму, представленную на рисунке 10. Здесь нужно ввести имя устройства и пароль. Поля «Описание» и список алиасов[5] созданы для удобства, их можно оставить пустыми на данном этапе.

[5] Псевдоним топика устройства, присвоенный пользователем. Алиасы можно присваивать стандартным топикам (реализованным сервисом) и топикам с произвольными сабтопиками.

Для этого действия также необходимо добавить сертификат.

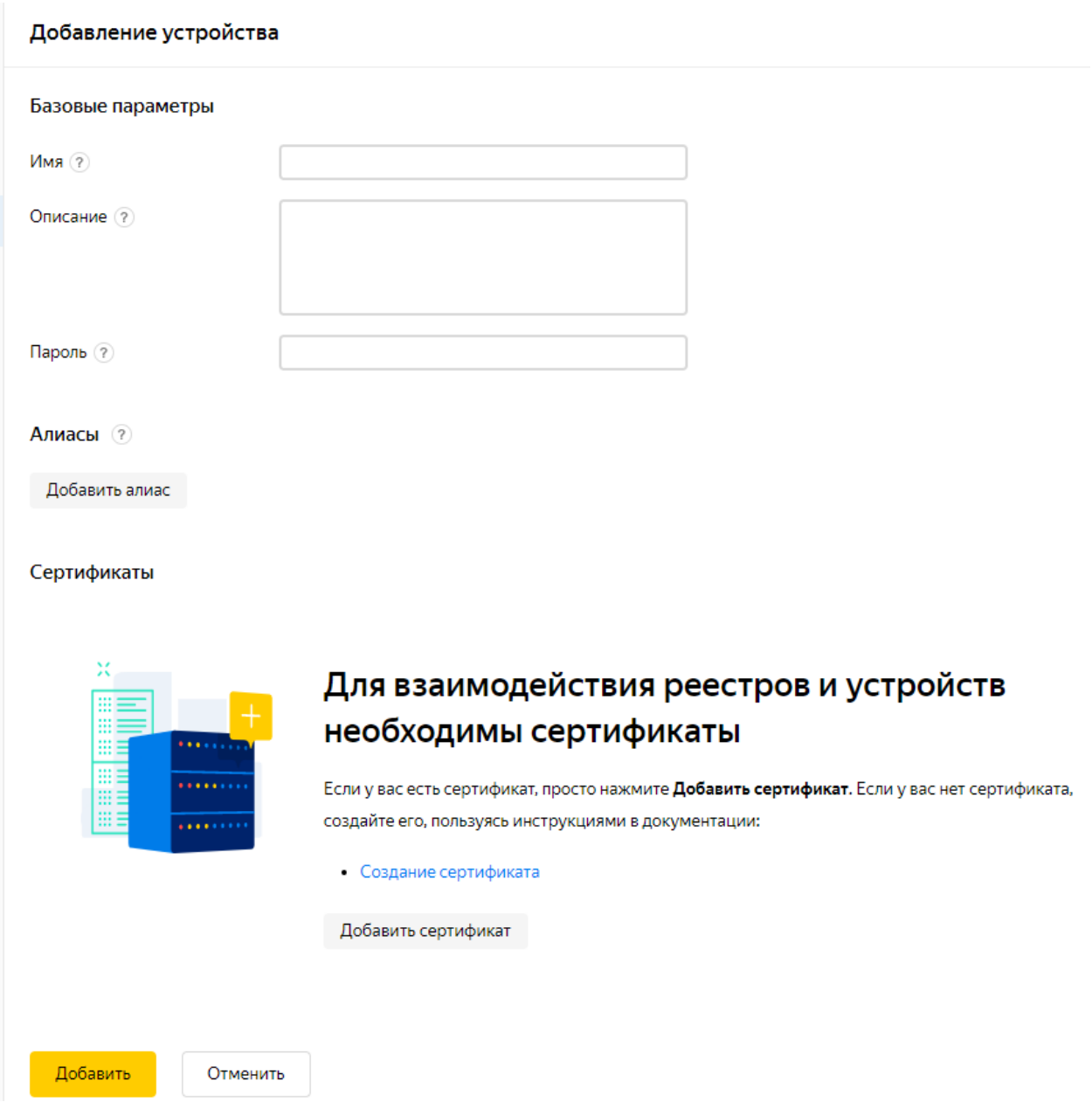


Рисунок 10 – Добавление устройства

3.2 Работа с устройством

При нажатии на устройство в списке устройств реестра (рисунок 7) откроется окно редактирования его параметров, в котором можно узнать ID устройства, добавить или удалить пароли и сертификаты для авторизации по MQTT (рисунок 11).

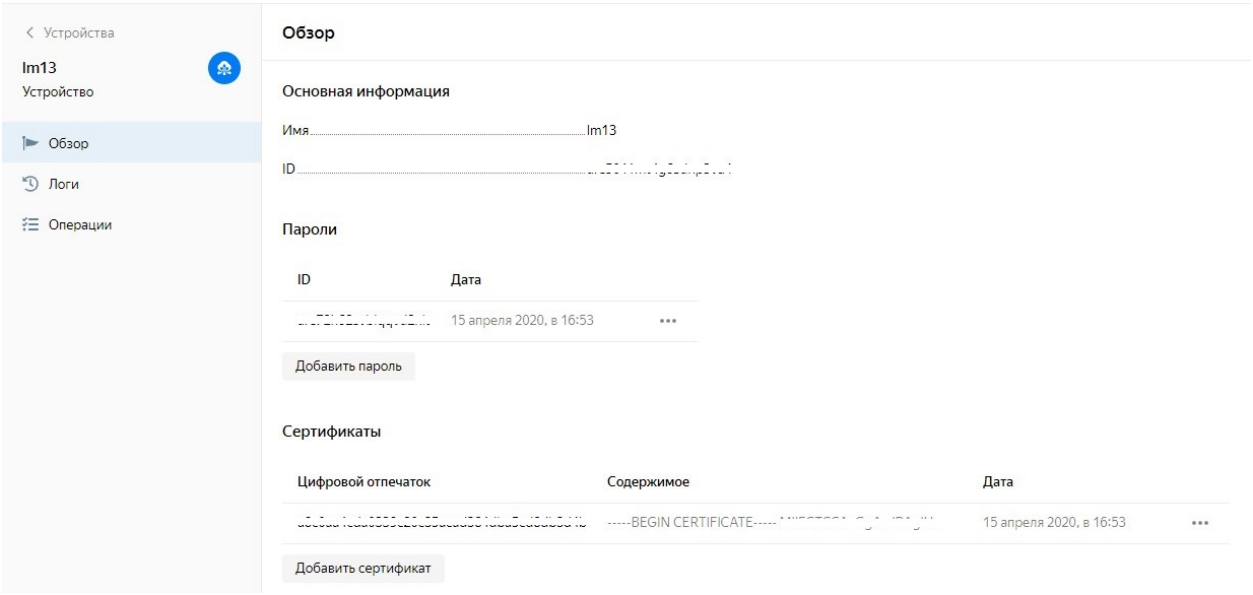


Рисунок 11 – Вкладка «Обзор» для работы с устройством

Во вкладке «Логи» находятся сообщения о подключении к реестру с помощью MQTT (рисунок 12).

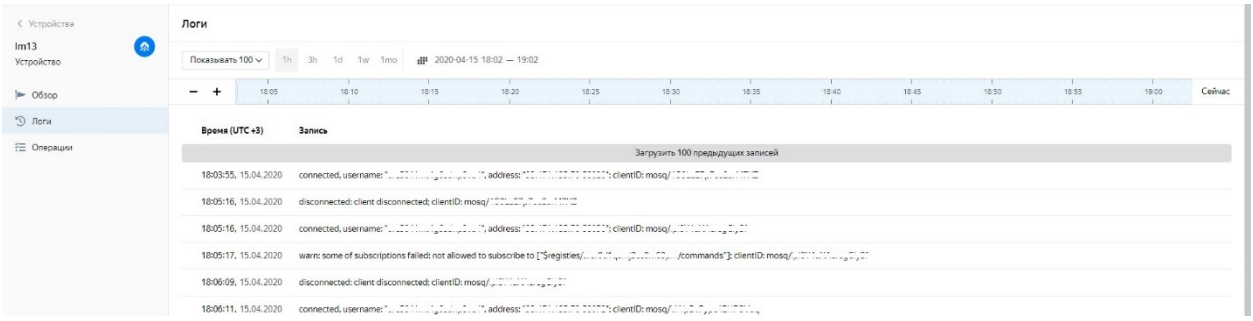


Рисунок 12 – Вкладка «Логи» с сообщениями о подключении устройства

Во вкладке «Операции» (рисунок 13) находятся сообщения о действиях с реестром: добавление сертификатов, устройств и т.п.

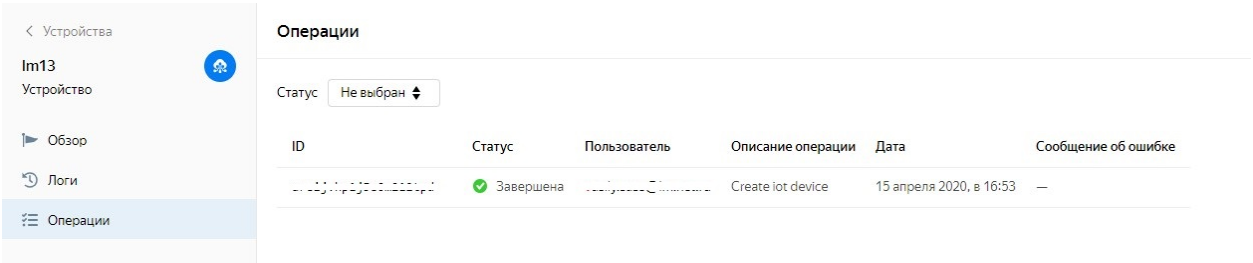


Рисунок 13 – Вкладка «Операции» с сообщениями о действиях с устройством

4. Добавление скриптов для обмена сообщениями в LogicMachine.

4.1 Отправка сообщения

В интерфейсе LogicMachine (рисунок 14) перейдите на вкладку «Scripting (Скрипты)», откройте «Event-based (Событийные)». Нажмите на кнопку «Add new script (Добавить скрипт)».

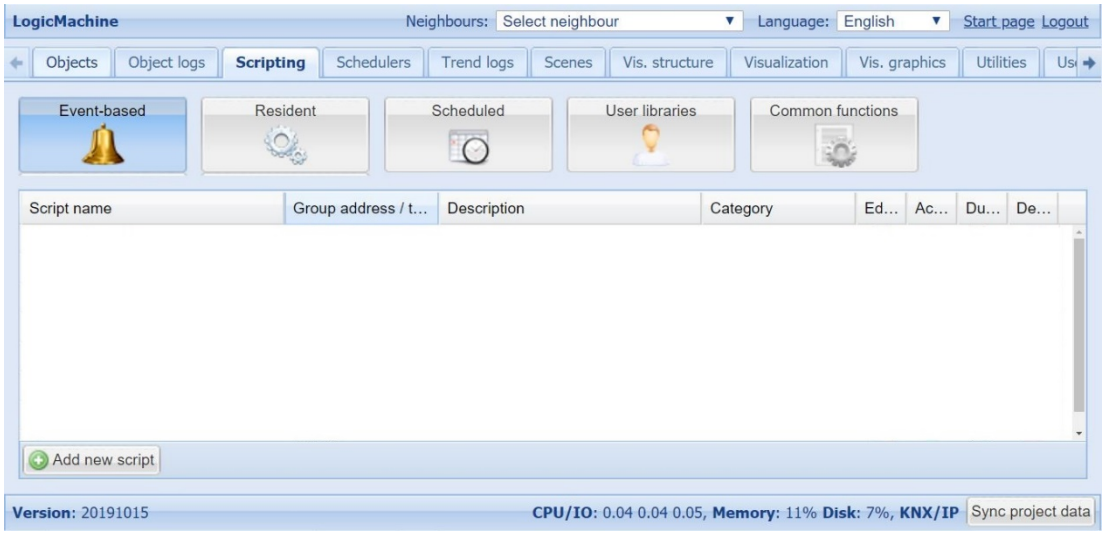


Рисунок 14 – Список событийных скриптов в интерфейсе LogicMachine

В появившемся окне настроек (рисунок 15) в поле «Script name (Название)» введите любое удобное для Вас название скрипта для *отправки* сообщений в облако, в списке «Group address / tag (Групповой адрес / Тег)» выберите групповой адрес или тэг источника события.

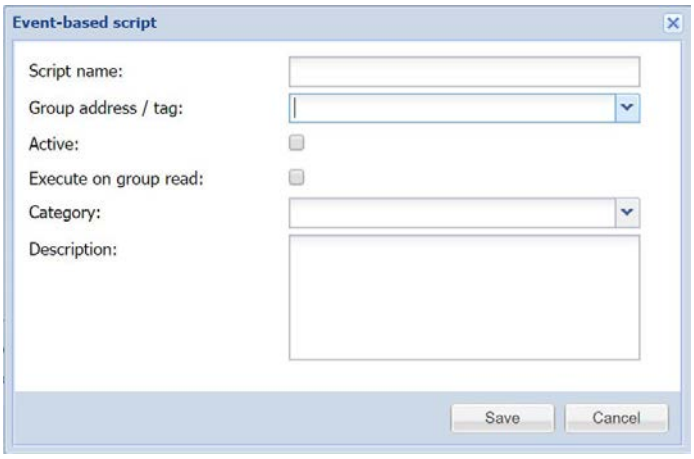


Рисунок 15 – Настройки для событийного скрипта

Откройте скрипт в редакторе, скопируйте в основное поле текст, приведённый в листинге «MQTT Publish.lua» в конце инструкции. Отредактируйте значения переменных id_dev, id_reg, username, password в соответствии с данными, приведёнными на вкладках «Обзор» интерфейса Яндекс.Облако (рисунки 6, 11).

4.2 Подписка устройства или реестра на получение сообщений

В интерфейсе LogicMachine (рисунок 16) на вкладке «Scripting (Скрипты)», откройте «Resident (Резидентные)». Нажмите на кнопку «Add new script (Добавить скрипт)».

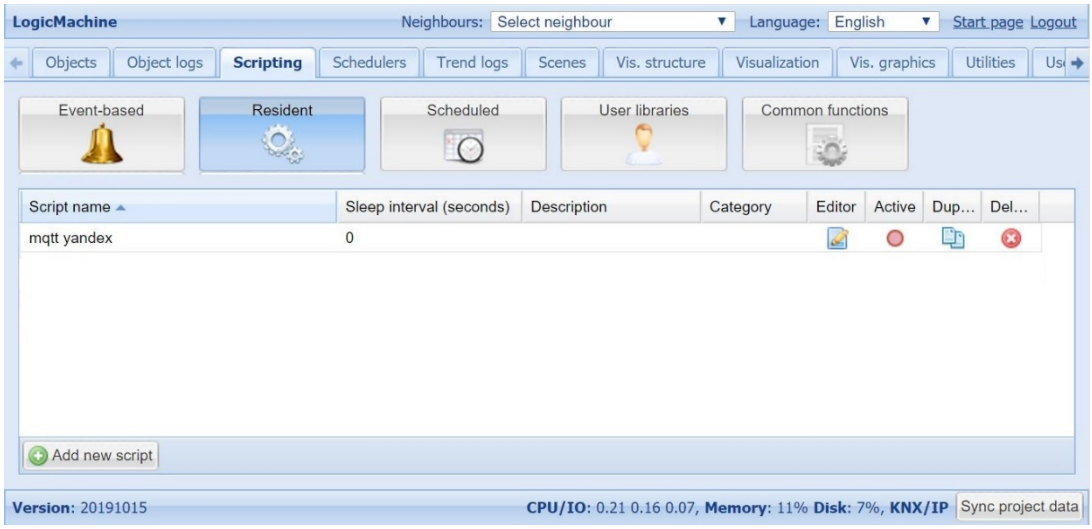


Рисунок 16 – Список резидентных скриптов в интерфейсе LogicMachine

В появившемся окне настроек (рисунок 17) в поле «Script name (Название)» введите любое удобное для Вас название скрипта для *получения* сообщений из облака, в списке «Sleep interval (seconds) (Интервал запуска (секунды))» выберите значение «0». Не забудьте активировать скрипты.

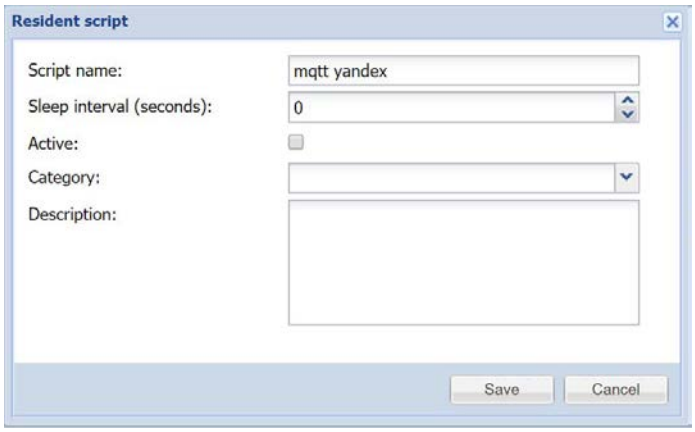


Рисунок 17 – Настройки для резидентного скрипта

Откройте скрипт в редакторе, скопируйте в основное поле текст, приведённый в листинге «MQTT Subscribe.lua» в конце инструкции. Отредактируйте значения переменных id_dev, id_reg, username, password в соответствии с данными, приведёнными на вкладках «Обзор» интерфейса Яндекс.Облако (рисунки 6, 11).

5. Способы авторизации

Как указывалось выше, существует два способа авторизации. Предыдущий раздел описывал действия, характерные для авторизации с помощью логина и пароля.

Если Вы желаете использовать сертификаты для этого действия, следует повторить те же действия, обращая внимание на следующие особенности:

- В интерфейсе LogicMachine следует воспользоваться разделом загрузки сертификата и ключа на LM через FTP.
- В разделе «Скрипты» следует создать скрипты *отправки* и *подписки*, используя заготовки, приведённые в листинге. При этом в каждом из них оставить один из двух фрагментов скрипта. **Фрагменты** заключены в листинге между многострочными комментариями с характерными обозначениями:
  - >>> – начало фрагмента
  - <<< – конец фрагмента

6. Примеры работы с отправкой и получения сообщений с помощью Yandex CLI.

Для отправки и принятия сообщений воспользуйтесь интерфейсом командной строки Яндекс.Облака.

6.1 Отправьте данные с устройства в топик устройства

Отправьте данные, используя логин и пароль:

```
$ yc iot mqtt publish \
--username <ID устройства> \
--password <пароль для устройства> \
--topic '$devices/<ID устройства>/events' \
--message 'Test data' \
--qos 1
```

6.2 Подписка реестра на одно устройство

Подпишите реестр, используя логин и пароль:

```
$ yc iot mqtt subscribe \
--username <ID реестра> \
--password <пароль для реестра> \
--topic '$devices/<ID устройства>/events' \
--qos 1
```

В приведённых примерах используются следующие ключевые слова:

- `--username` и `--password` — параметры для авторизации с помощью логина и пароля.
- `--topic` — топик устройства для отправки данных.
- `--message` — текст сообщения.
- `--qos` — уровень качества обслуживания (QoS).

Больше примеров, в том числе для других способов авторизации, Вы можете найти в документации по работе с сервисом Яндекс.Облако IoT Core в разделах «Отправка сообщения» и «Подписка устройства или реестра на получение сообщений»

7. Листинг скриптов

7.1 MQTT Publish.lua

```
broker = "mqtt.cloud.yandex.net"
port = 8883
id_dev = '<id устройства>'
id_reg = '<id реестра>'
username = id_dev -- Логин — это идентификатор устройства или реестра.
password = "<пароль>" -- Пароль — это заданная вами комбинация символов. Также пароль может быть сгенерирован с помощью YC CLI.
-- Минимальная длина пароля — 14 символов.
-- Пароль должен содержать 3 из 4 групп символов: строчные латинские буквы, прописные латинские буквы, цифры, специальные символы.

dev_topic = "$devices/" .. id_dev .. "/events"
reg_topic = "$registries/" .. id_reg .. "/events"
-- $devices/<ID устройства>/commands
-- $registries/<ID реестра>/events
-- $registries/<ID реестра>/commands
--[[
  events: устройство отправляет/реестр получает
  commands: реестр отправляет/устройство получает
  $devices: устройство известно
  $registries: устройство неизвестно
--]]

payload = event.getvalue() --100 -- отправляемое

mqtt = require("mosquitto")
client = mqtt.new()

client.ON_CONNECT = function(status, rc, msg)
  if status then
    log("mqtt connected")
    client:publish(topic, tostring(payload))
  else
    log("mqtt connect failed " .. tostring(msg))
    client:disconnect()
  end
end
```

```
    client.ON_PUBLISH = function()
      client:disconnect()
    end
    --[[>>> фрагмент скрипта, если используются логин и
пароль]]--
      client:login_set(username, password)
      client:tls_set('/home/ftp/rootCA.crt')
    --[[<<<]]--
    --[[>>> фрагмент скрипта, если используется
сертификат]]--
    client:tls_set('/home/ftp/rootCA.crt',         nil,
'/home/ftp/client.crt', '/home/ftp/client.key')
    --[[<<<]]--
    -- сертификат Яндекса с
https://storage.yandexcloud.net/mqtt/rootCA.crt
    status, rc, msg = client:connect(broker, port)

    if status then
      client:loop_forever()
    else
      log("connect failed: " .. tostring(msg))
    end
```

7.2 MQTT Subscribe.lua

```
    if not broker then
      broker = "mqtt.cloud.yandex.net"
      port = 8883
      id_dev = '<id устройства>'
      id_reg = '<id регистра>'
      username = id_dev -- Логин — это идентификатор
устройства или реестра.
      password = "<пароль устройства>" -- Пароль — это
заданная вами комбинация символов. Также пароль может
быть сгенерирован с помощью YC CLI.
      -- Минимальная длина пароля — 14 символов.
      -- Пароль должен содержать 3 из 4 групп символов:
строчные латинские буквы, прописные латинские буквы,
цифры, специальные символы.

      dev_topic = "$devices/" .. id_dev .. "/commands"
      reg_topic   =   "$registries/"   ..   id_reg   ..
"/commands"
      -- $devices/<ID устройства>/commands
      -- $registries/<ID реестра>/events
      -- $registries/<ID реестра>/commands
```

```
        --[[
          events: устройство отправляет/реестр получает
          commands: реестр отправляет/устройство получает
          $devices: устройство известно
          $registries: устройство неизвестно
        --]]

        mqtt = require("mosquitto")
        client = mqtt.new()

        client.ON_CONNECT = function(status, rc, msg)
          if status then
            log("mqtt connected")
            client:subscribe(reg_topic)
            client:subscribe(dev_topic)
          else
            log("mqtt connect failed " .. tostring(msg))
            client:disconnect()
          end
        end

        client.ON_PUBLISH = function()
          client:disconnect()
        end

          client.ON_MESSAGE   =   function(mid,   topic,
payload)
          log(mid, topic, payload)
        end

          client.ON_DISCONNECT = function(...)
          log('mqtt disconnect', ...)
        end

        client.ON_SUBSCRIBE = function(...)
          log('subcribe: ', ...)
        end
      --[[>>> фрагмент скрипта, если используются логин и
пароль]]--
        client:login_set(username, password)
        client:tls_set('/home/ftp/rootCA.crt')
      --[[<<<]]--
      --[[>>> фрагмент скрипта, если используется
сертификат]]--
```

```
    client:tls_set('/home/ftp/rootCA.crt',          nil,
'/home/ftp/client.crt', '/home/ftp/client.key')
    --[[<<<]]--
      status, rc, msg = client:connect(broker, port)
      if status then
        client:loop_forever()
      else
        log("connect failed: " .. tostring(msg))
      end
    end
```