# DeviceAuthenticator

Core microservice for devices to perform authentication and get tokens.

## Technologies

* Back: NodeJS
* Database: MongoDB
* Front: React

## Basic Response Model

* All responses are JSON objects
* All responses must have at least these two parameters
  + status: “success” or “failed”
  + message: String
* All responses must include response code (200 for success, non 200 for failures)

## Notes

* /libs/DeviceIdentitiesClient is referred as DeviceIdentities library in this document
* /libs/DeviceTokenRepoLib is referred as DeviceTokens library in this document

/server/getIdentity [POST, JSON]

Returns the server identity file.

### Parameters

* + none

### Return

* + server.id file

### Controller Class

* + server

### Affected Tables

* + server

### Steps

1. Return /identity/server.id file

/devices/createSession[POST, JSON]

Create a session for requested device.

### Parameters

* + dev\_serial: String
  + dev\_identity: JSON Object
  + secured: String
  + payload: String, Base64Encoded

### Return

* + JSON object of {session\_id, encrypted nonces}

### Controller Class

* + devices

### Steps

1. Call DeviceIdentities.verify(dev\_serial, dev\_identity) to get dev\_id
2. If verify failed
   1. Return Error message
3. Calculate payload\_json=RSA\_DECRYPT(payload, “/keys/pvt.key”)
4. Set skey = payload\_json.skey
5. Set iv = payload\_json.iv
6. Set device\_nonce = AES\_DECRYPT( payload\_json.nonce, skey, iv)
7. Set return\_nonce = device\_nonce+10
8. Set server\_nonce = new Random long integer (8 digit at-least)
9. Set response\_payload as JSON object:
   1. “return\_nonce”:return\_nonce
   2. “server\_nonce”:server\_nonce
10. Calculate enc\_response\_payload = BASE64ENC( RSA\_ENCRYPT(response\_payload, dev\_identity.pubKey) )
11. Call DeviceTokens.createSession(dev\_id, dev\_identity.dev\_serial, dev\_identity.dev\_usage, dev\_identity.dev\_type, dev\_identity.group\_id, server\_nonce, secured, skey, iv) to get session\_id
12. Return JSON Object:
    1. “session\_id”:session\_id
    2. “payload”: enc\_response\_payload

/devices/authenticate [POST, JSON]

Authenticated the device and returns the access token.

### Parameters

* + dev\_serial: String
  + session\_id: String
  + payload: String, Base64Encoded

### Return

* + Encrypted token info

### Controller Class

* + devices

### Steps

1. Call DeviceTokens.getSession(session\_id, dev\_serial) to get session
2. Calculate payload\_json = AES\_DECRYPT(payload, session.skey, session.iv)
3. Calculate session\_nonce = payload\_json.nonce – 10
4. Compare session\_nonce==session.nonce
   1. If not, Return Error: Not Allowed
5. Call DeviceTokens.createToken(session.dev\_id, session.dev\_serial, session.dev\_usage, session.dev\_type, session.group\_id, session.secured, session.skey, session.iv, Config.tokenLife) to get token\_info
6. Set response\_json as Json object:
   1. “access\_token”: token\_info.access\_token
   2. “refresh\_token”: token\_info.refresh\_token
   3. “expired”: token\_info.expired
7. Return BASE64ENC( AES\_ENC(response\_json, session.skey, session.iv) )