# Moptim Device Auth

# Device Integration

## Functions

1. **Http Trigger** for handshaking of *Access Code*. This also returns the *Store Id* along with *Unique Id*.
2. **Timer Trigger** for cleaning up *Access Code* from CORE Db (5 minute cleanup). Even if the code is not cleaned up at exactly 5 minutes, it will invalidated by the backend using `CreatedAt` property.

## Core Backend

1. **Table** with `DeviceIntegration` schema. `DeviceIntegration.AccessCodes` is the table name.

## Columns and Types

| Column | Type |
|---|---|
| Id | `Guid` |
| StoreId | `int` |
| Access Code | `string[8]` |
| Created Date | `DateTime` |
| Status | `bool` |

2. **Table** with `DeviceIntegration` schema. `DeviceIntegration.RegisteredDevices`

| Column | Type |
|---|---|
| Id (Unique Id) | `Guid` |
| StoreId | `int` |
| Created Date | `DateTime` |

> Once the **Deregister Occurs** we simply delete the entry from *Table 2*

3. **Http Trigger** Function for Generates the Access Code.
    1. If the unique key table has an entry for the `storeid`, if its there then display already registered.

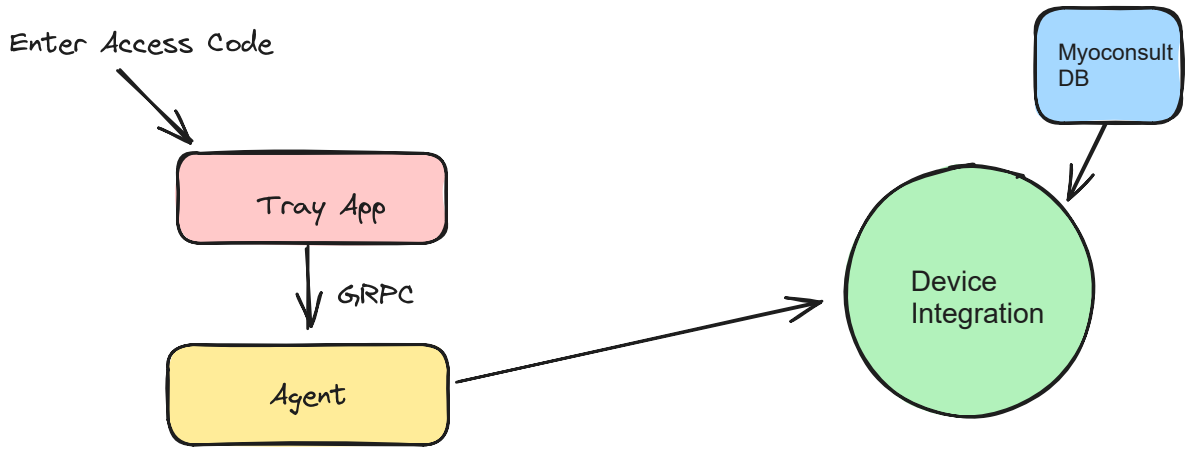
2. Otherwise create the access code.

4. **Http Trigger** Function for deleting the *Registered Device* Entry from Table 2.

## Moptim Agent/Tray

1. **UI** to enter the access code, with message if the ECP gives wrong Access Code.
2. Agent Makes the **API call** with the access code to Device Integration; *Unique ID*, a `Json` object with encrypted *Unique ID* and StoreID of the store.

```
body = {
        "UniqueID": "<AES Encrypted Base64 String>"
        "StoreID": <Store ID as int>"
}
```



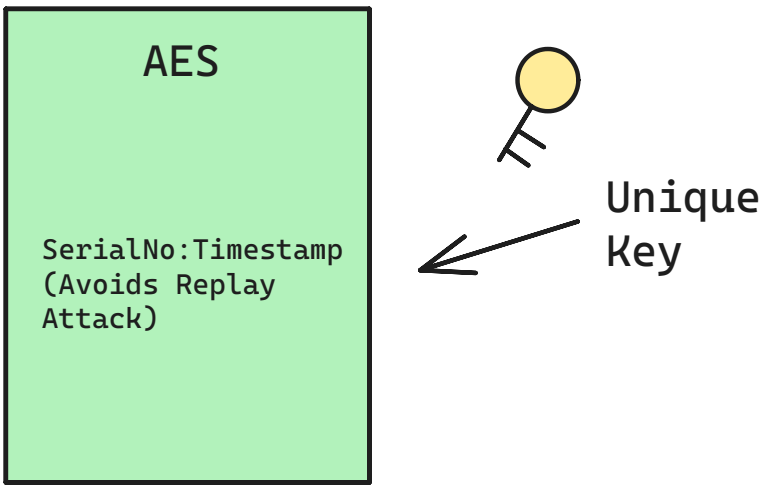

Tray Flow

3. Store the *Unique Id* in **credential manager**.

4. Pass this *Unique Id* to the **SAS generation** endpoint, along with device serial number.

- This HTTP post payload is expirable due the timestamp, this does not guarantee MITM attacks cannot be performed but makes it a bit more challenging.

-




## Encryption

Certain payloads are encrypted during exchange between the server and the device. The scheme used for encryption is `AES` encryption, which offers strong encryption while still being performant.
To derive the actual encryption key from the plain unique key PBKDF2 algorithm is used with salt size of 20 bytes, 1000 iterations and SHA256 as the hashing algorithm.

## Parameters

Below are the agreed upon parameters of importance, used in implementation.

| Parameter | Value |
| --- | --- |
| Access Code Expiry | 5 minutes |
| Unique Key Expiry | Until deregistration |
| SAS token expiry for Azure | 24 hours |