LaMetric Documentation

Release v1.8.0

Dmytro Baryskyy

Nov 18, 2022

Welcome to LaMetric developer documentation!

We are constantly improving and expanding the docs so check back often!

Tip: This documentation is opensource and available on GitHub. So if you find a bug, typo or maybe can explain things better – don't hesitate to contribute!

Overview

LaMetric is the open Internet of Things platform.

With LaMetric developers can:

- Create indicator apps that run on LaMetric and display key metrics or KPIs.

- Create button apps that run on LaMetric and can trigger HTTP requests.

- Deliver notifications right to the LaMetric Time in the local network.

- Integrate LaMetric Time into existing smart home or IoT systems.

1.1 Developer Highlights

LaMetric aims to be developer friendly platform. Key developer features:

- Simple REST API to send notifications to device in local network (no need to install apps on LaMetric) and get its state.

- Easy LaMetric app creation that does not require development (IDE can be found at https://developer.lametric. com)

- Web based simulator is available to help you test your app.

- Growing LaMetric Time community

1.2 How it Works

There are multiple ways of getting information onto LaMetric Time screen. Notification can be pushed in local network using Notification APP in combination with device REST API or to native LaMetric apps.

1.2.1 LaMetric Notification via Local REST API

This way of sending notifications to device is preferrable if:

- Fast response time is required.

- You are sending sensitive data and you don't want it to leave your local network.

- Data from notification should not stay on the display for too long.

1.2.2 Native LaMetric Apps

It is possible to create indicator or button apps. Create indicator app when:

- It is important to display numbers, charts or text messages that stay on LaMetric until new data has come and optionally notify the user about change.

- LaMetric Time must get fresh data by itself (via polling) or you want to push fresh data in local network or via LaMetric Cloud.

- Trigger action right from LaMetric Time in response to the data that is displayed.

- You are willing to distribute app to other LaMetric users.

Create button app when:

- It is required to trigger some action from LaMetric Time.

- User must be able to customize text and icon that is displayed on LaMetric when button app is active.

- You are willing to distribute app to other LaMetric users.

Tip: All public native LaMetric apps are available in our Appstore

---

1.3 What's Next?

To start developing for LaMetric, login as a developer to DevZone and create new app. Also check out LaMetric User Guide. In section 7 you will find answers to some frequently asked questions. Also check our *First Steps* guide to learn more.
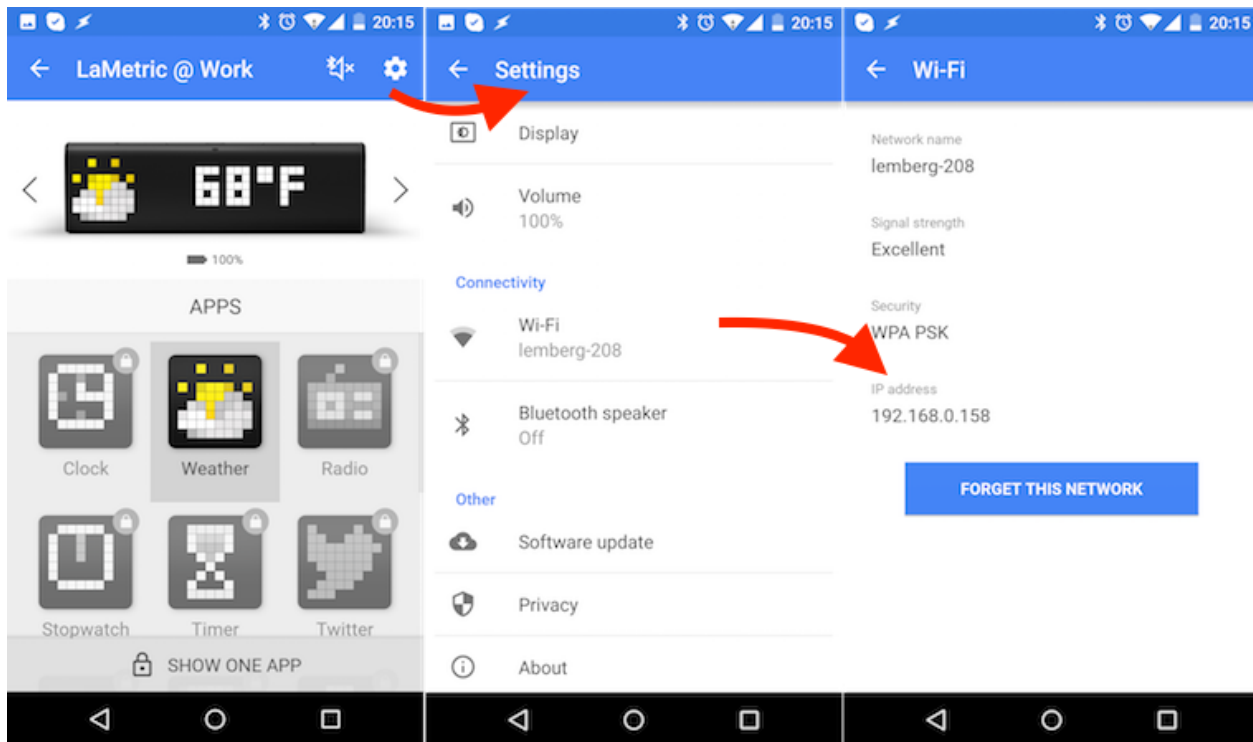
First Steps

2.1 First Local Notification

Let's send our first notification to the LaMetric Time device. In order to do that you need to follow few simple steps:

1. Discover LaMetric Time device's IP address

2. Discover API key

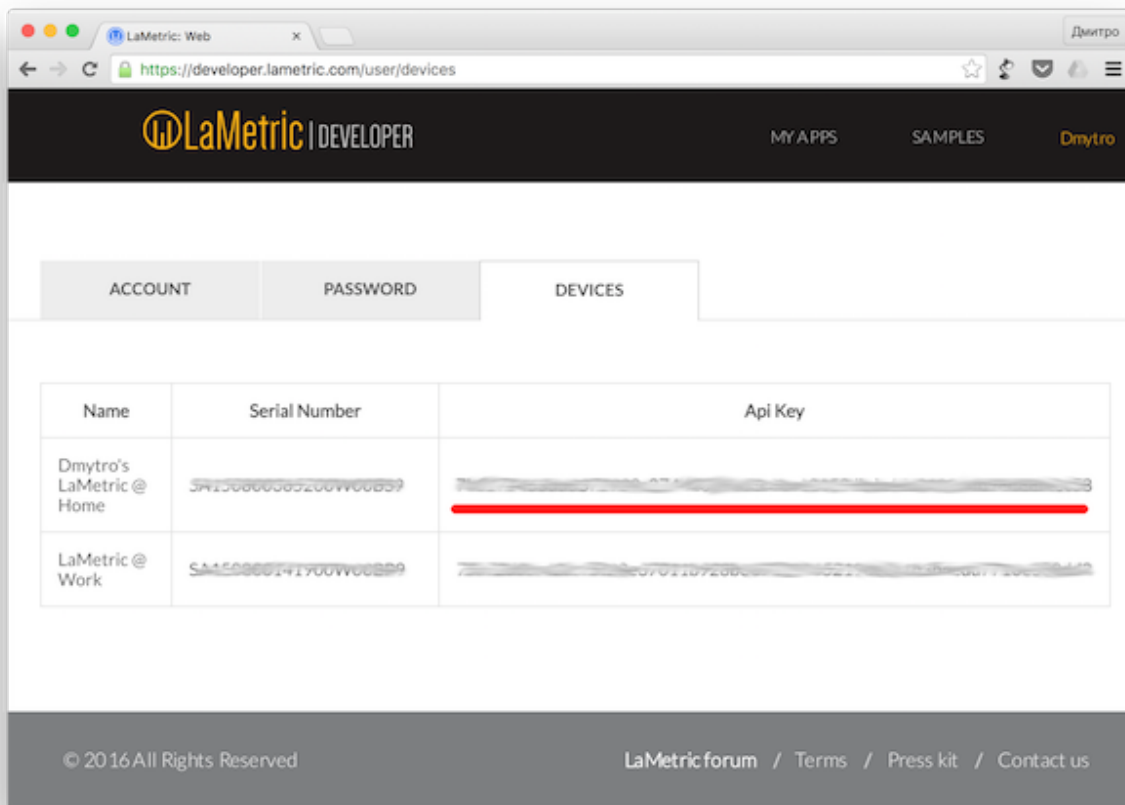3. Do simple authenticated POST REST API call

2.1.1 Discover IP address

IP address can be found in LaMetric Time app at Settings -> Wi-Fi -> IP Address.

2.1.2 Find API key

API key can be found in devices section of your developer account.

2.1.3 Send notification

In order to send a notification you must do HTTP POST request to http://<lametric_time_ip_address>:8080/api/v2/device/notifications endpoint with headers:

- Authorization: Basic <base64(dev:api_key)>

- Content-Type: application/json

and body:

```
{
    "model": {
        "frames": [
            {
                "icon":2867,
                "text":"Hello!"
            }
        ]
    }
}
```

Note: Copy paste curl example into your terminal window and don't forget to reaplace <your API key here> and <ip address> with valid values.

HTTP Example:

```
$ curl -X POST -u "dev:<your API key here>" -H "Content-Type: application/json" -d "
→{ \"model\": { \"frames\": [ { \"icon\":\"a2867\", \"text\":\"Hello\!\"} ] } }"
→http://<ip address>:8080/api/v2/device/notifications
```

HTTPS Example:

```
$ curl -X POST -u "dev:<your API key here>" -H "Content-Type: application/json" -d "
→{ \"model\": { \"frames\": [ { \"icon\":\"a2867\", \"text\":\"Hello\!\"} ] } }"
→https://<ip address>:4343/api/v2/device/notifications --insecure
```

Result:

```
200 OK
{ "success" : { "id" : "4" } }
```

2.1.4 What's next?

Now when you have learned how to make simple notification, it is time to create more complex one. Check out *Notification API reference* for more details. If you are interested in displaying data on the device permanently (like counter or metric) - check out *First Indicator App* section.

2.2 First Indicator App

Indicator app is a native LaMetric Time app that is useful for displaying icons, numbers, text and charts that must stay on the device at all times.

There are two ways you can deliver data to the app – push and poll. "Push" means you can send data to the app in your local network and it will be delivered immediately. "Poll" app will poll for the data with some predefined time interval in local network or from the Internet.

2.2.1 Local Push Indicator App

Use "Local Push" communication type when data is not changing too often and you want it to be updated as soon as it changes. Let's create the simplest indicator app and see how it works.

1. Create Indicator App

Start by logging in to your developer account and creating new indicator app.
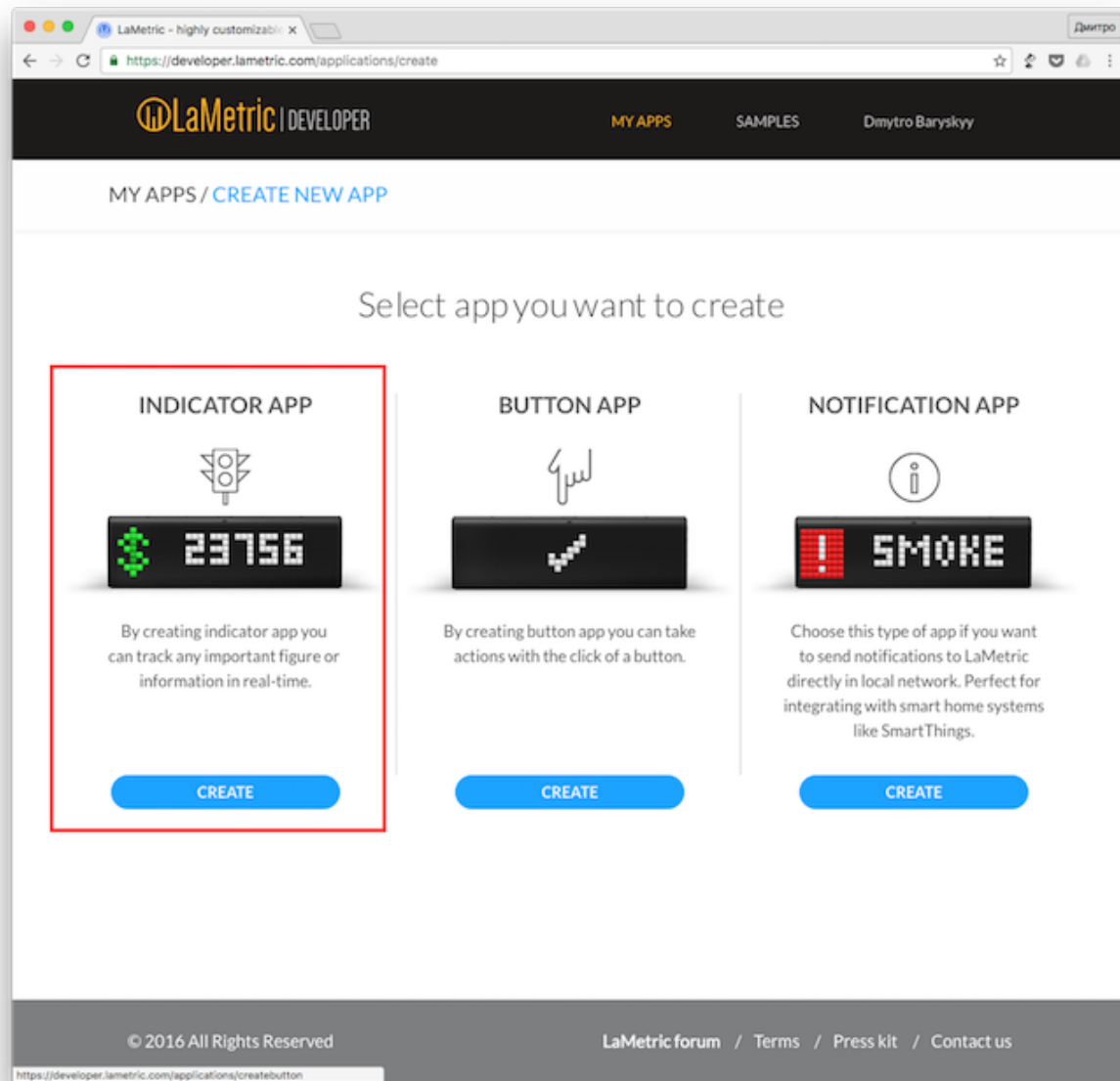
Choose an icon for your app. It will represent what we are trying to keep track of.
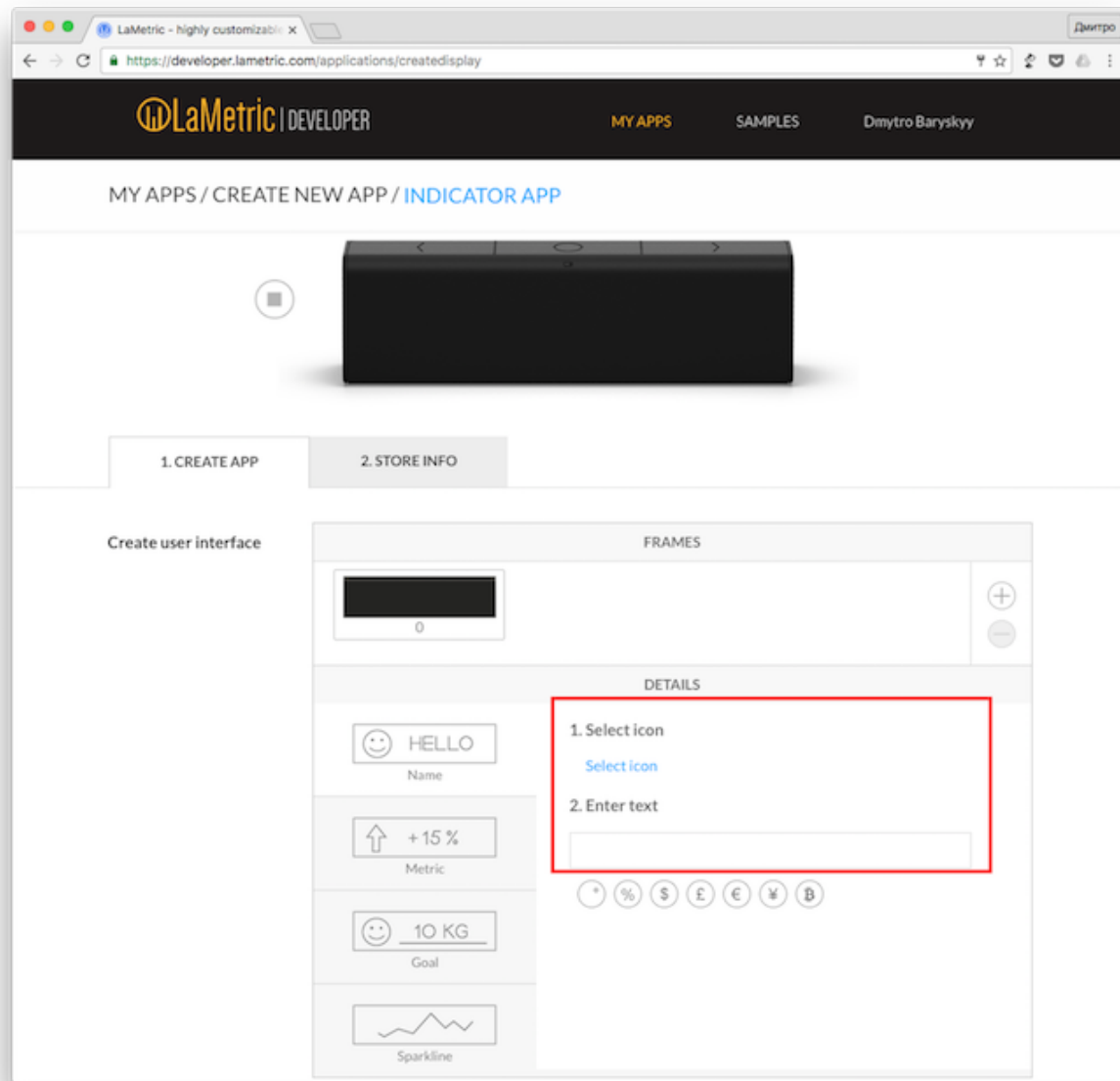
You can choose existing one or create the icon for yourself. It will be available to the whole community.
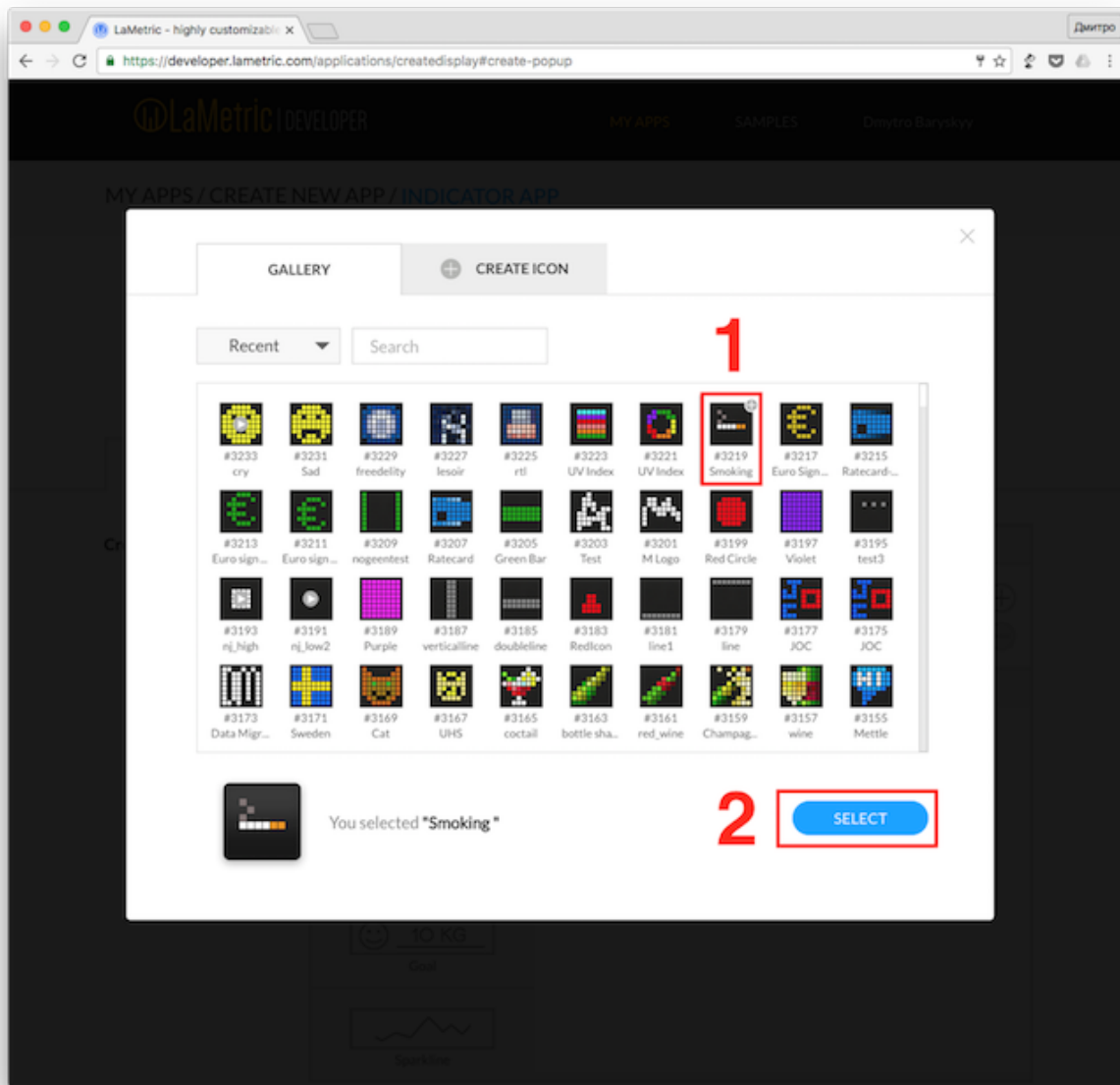
Enter initial value that will be displayed by default.
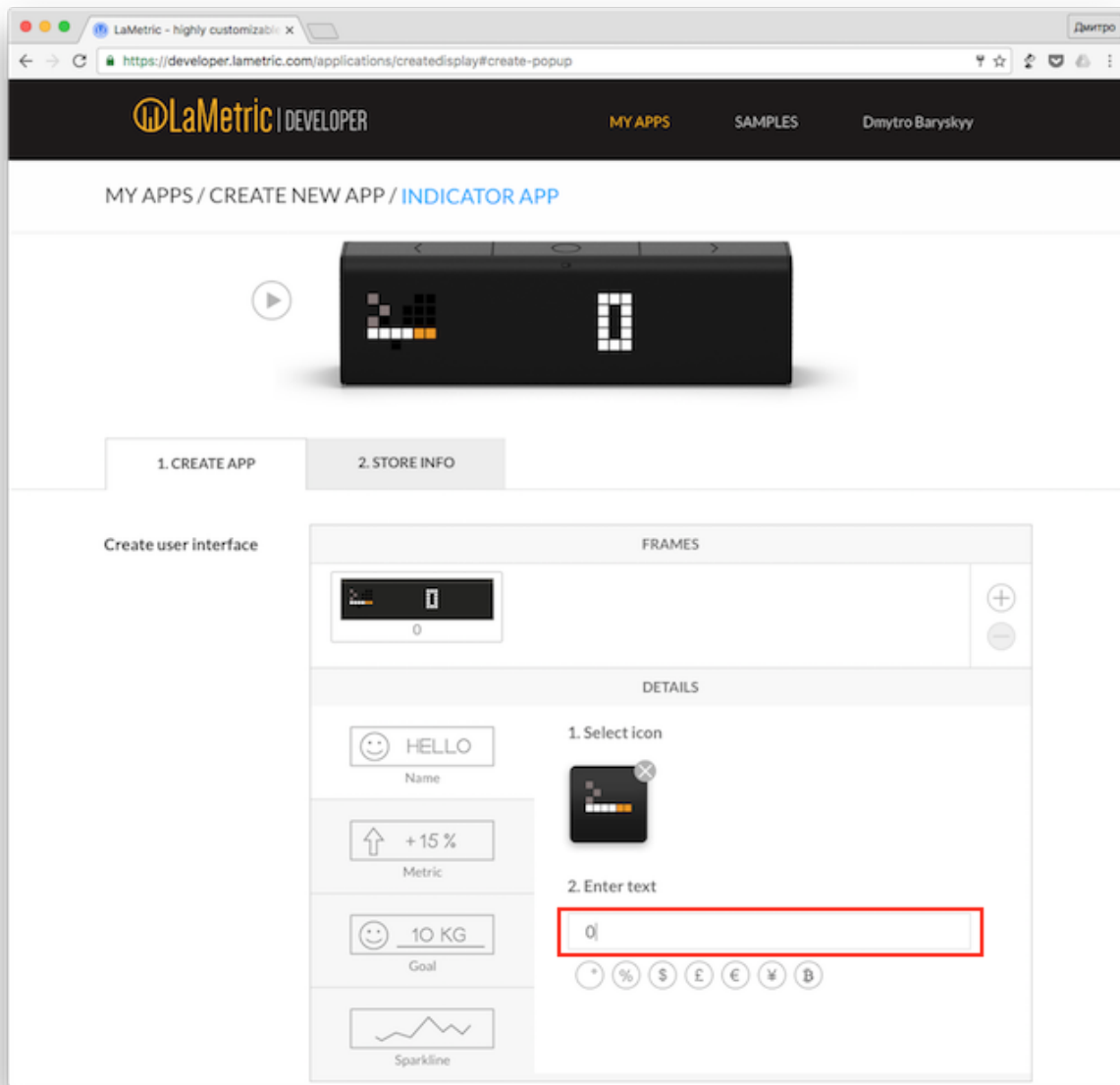
Choose "Local Push" communication type as we are going to push the data to the app.

Go to the next step by clicking on the "Next" button.
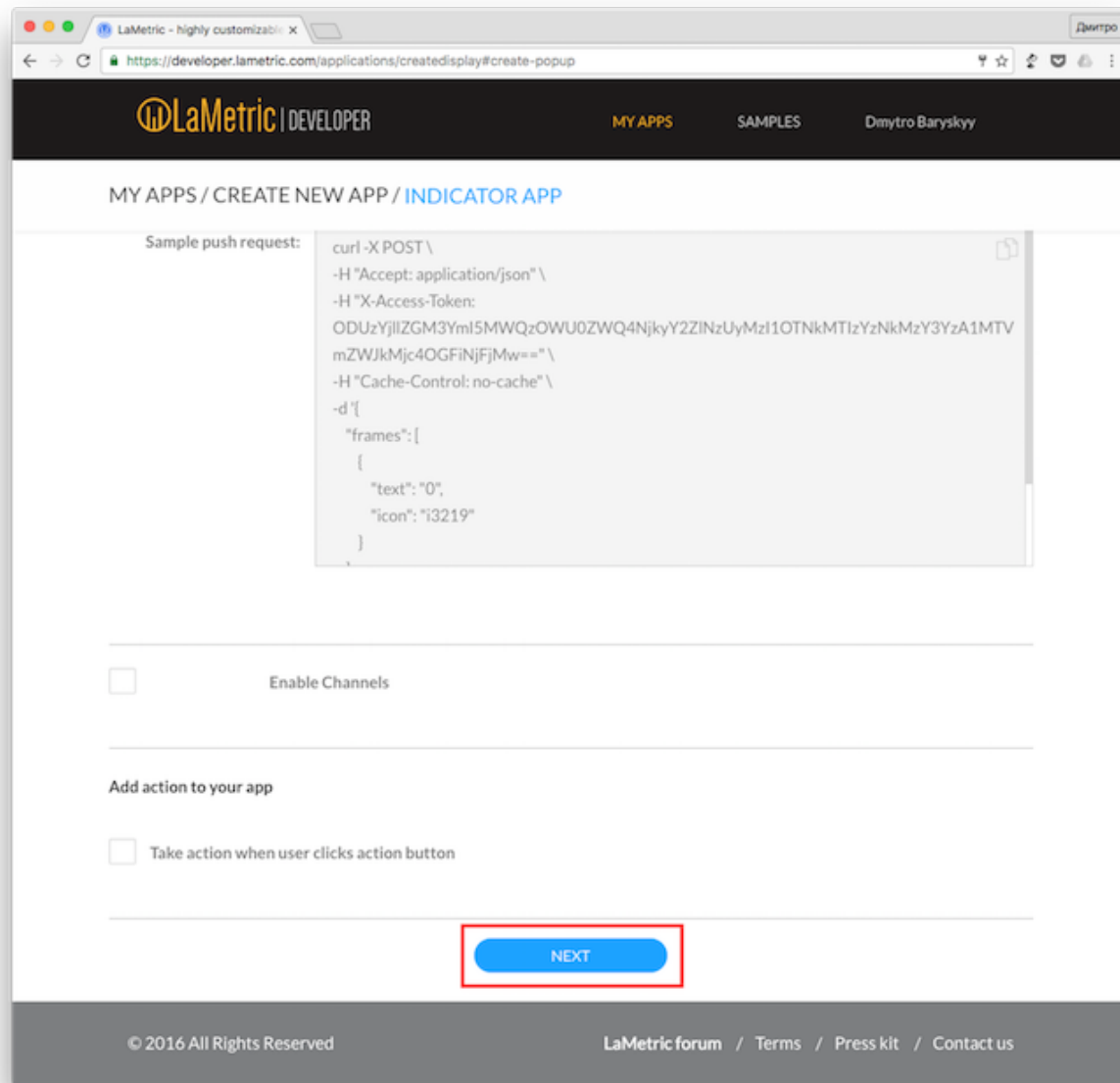
Now enter the name and description for your app. You'll find it in LaMetric Market and LaMetric Time app.

Once the app is saved you will be presented with local push URL, access token and data format. Now you have all the information to start delivering some data.

2. Publish the app and install it to your LaMetric Time

Let's click that "Publish" button at the top and wait for a confirmation email. It should take no longer than two minutes for the e-mail to reach your inbox. Once the email is delivered - congratulations, your app is live.
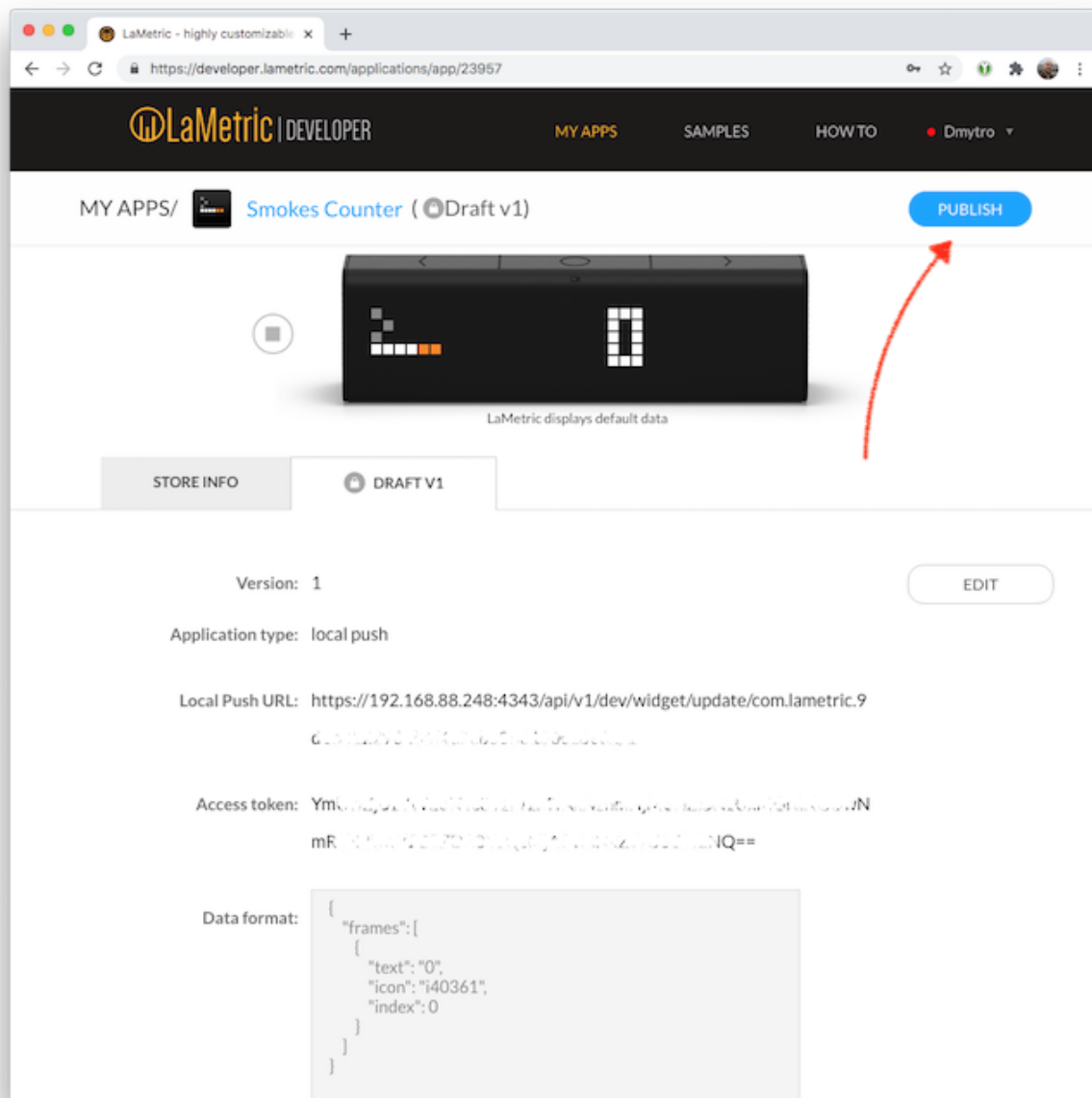
Now install your newly created app to your LaMetric Time device!

12:26 ◢   .ıll 🗢 🔋

Done

Private Apps

CREATED BY ME

Smokes Counter
User
♥ 0

ADD

Home   Categories   **Private**   Search
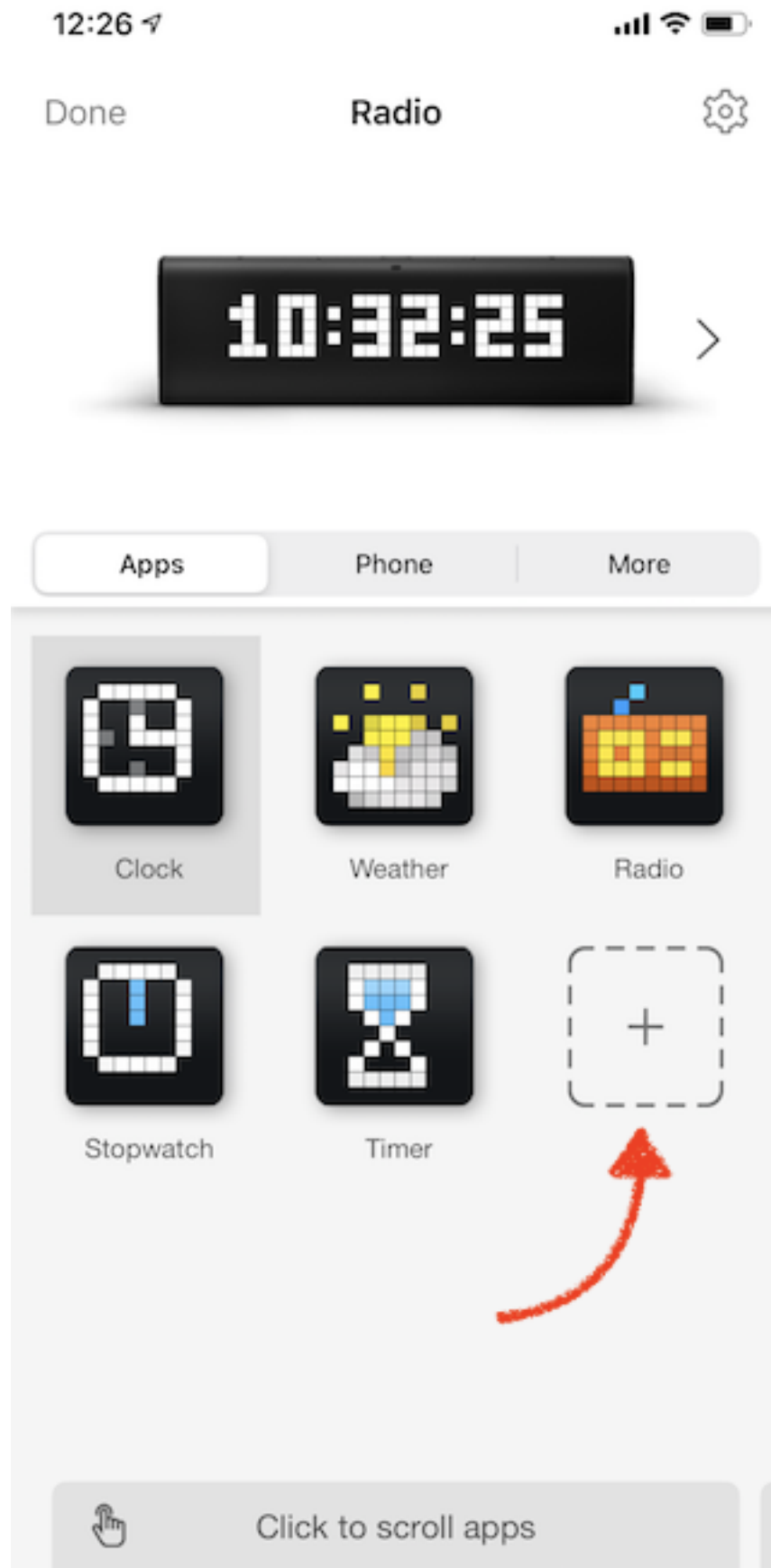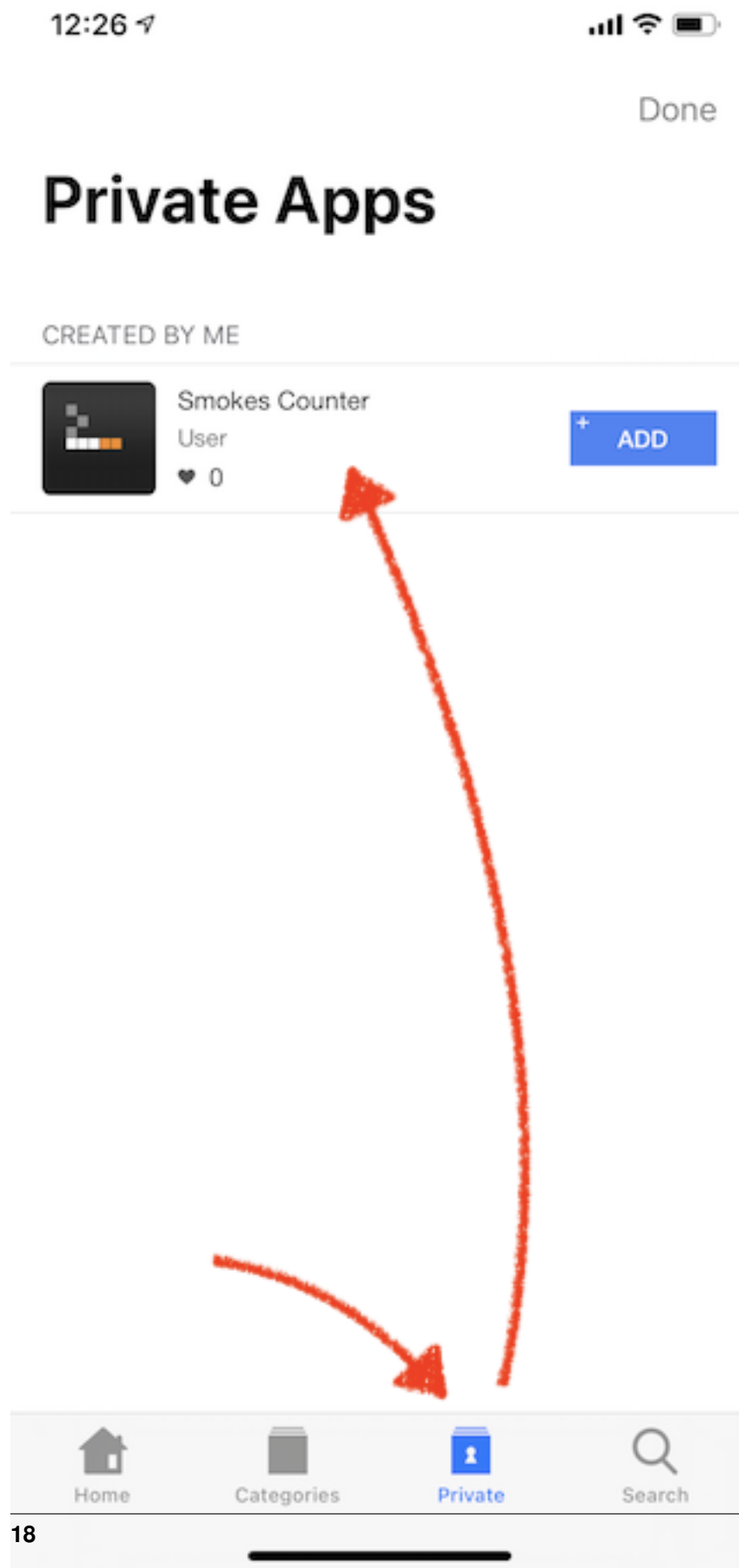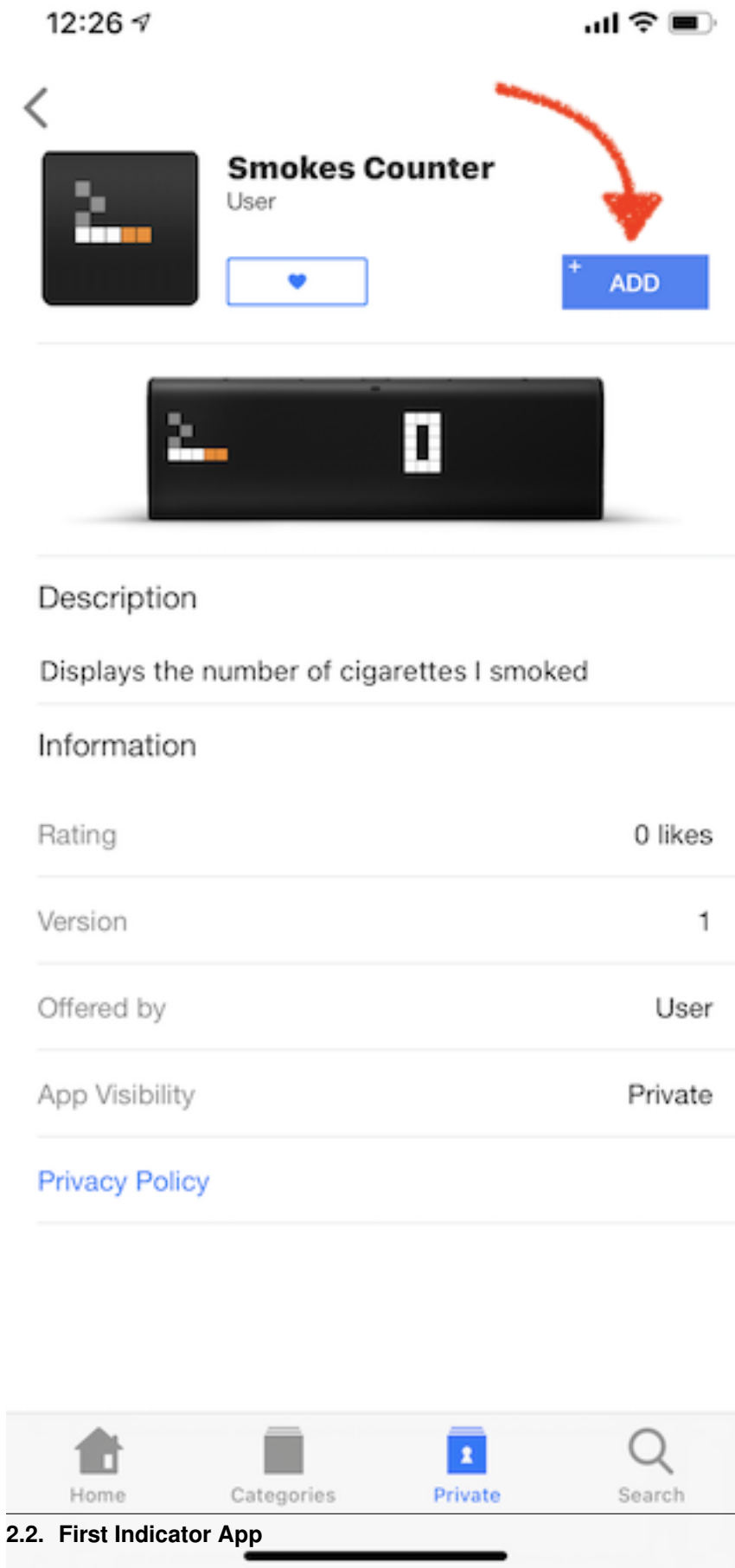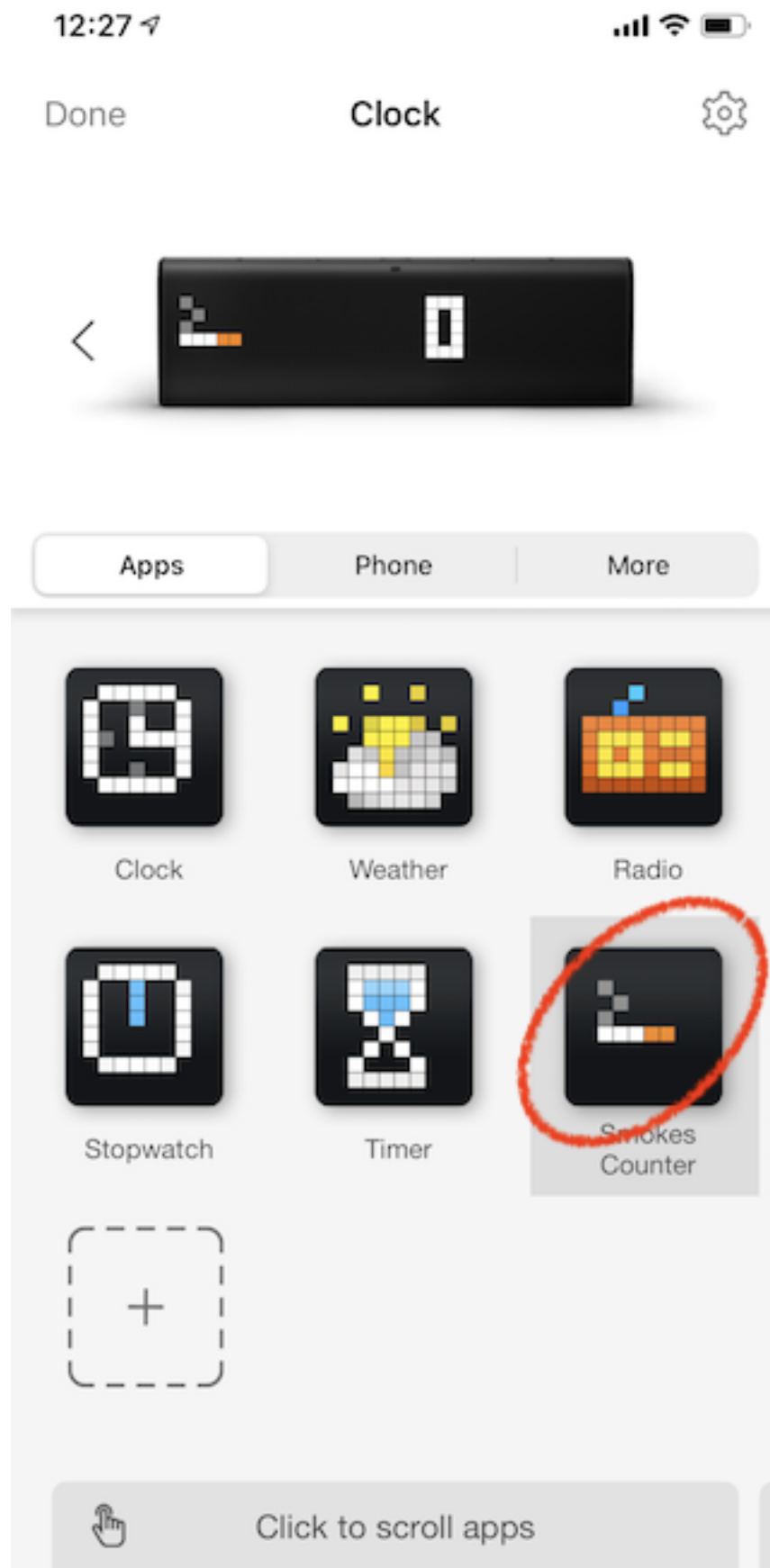
At this stage it should be active on LaMetric Time displaying default value.

3. Push some data!
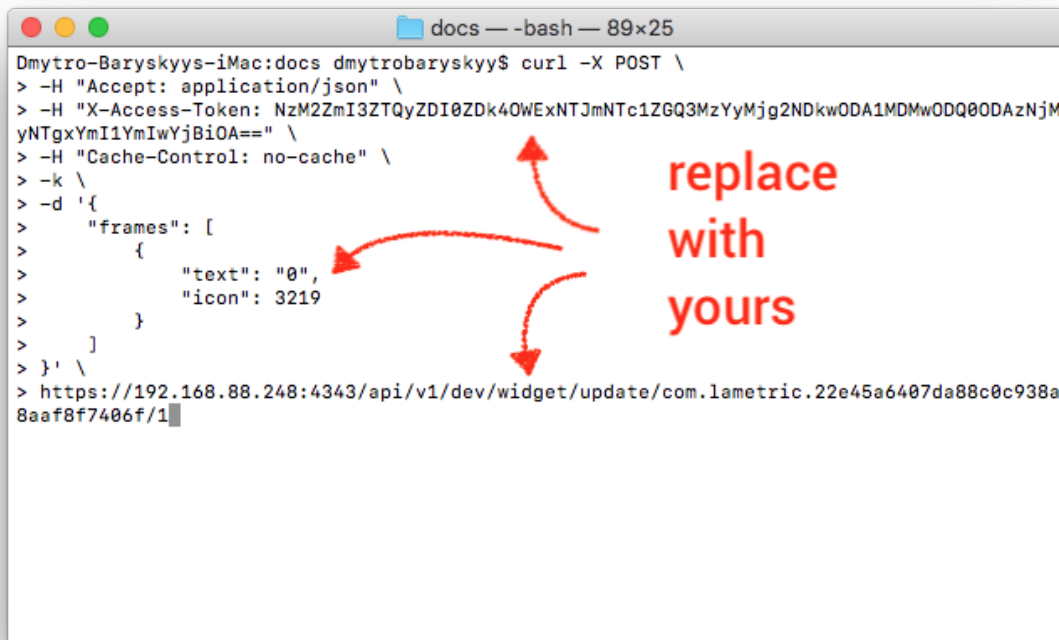
Tip: We will be using curl tool to make HTTP requests. While Linux and Mac users have it pre-installed, Windows users may want to download it.

Fortunately, there is curl example already on your app details page. Lets see how it looks in our case.

```
curl -X POST \
-H "Accept: application/json" \
-H "X-Access-Token:␣
↪NzM2ZmI3ZTQyZDI0ZDk4OWExNTJmNTc1ZGQ3MzYyMjg2NDkwODA1MDMwODQ0ODAzNjMyNTgxYmI1YmIwYjBiOA==
↪" \
-H "Cache-Control: no-cache" \
-k \
-d '{
    "frames": [
        {
            "text": "1",
            "icon": 3219
        }
    ]
}' \
https://192.168.88.248:4343/api/v1/dev/widget/update/com.lametric.
↪22e45a6407da88c0c938a8aaf8f7406f/1
```

Note: Don't forget to replace X-Access-Token and URL with values specific to your app. You can find them on the app details page on "Published V1" tab.

Let's copy and paste that into terminal app, change value of "text" attribute to "1" and run!

```
Dmytro-Baryskyys-iMac:docs dmytrobaryskyy$ curl -X POST \
> -H "Accept: application/json" \
> -H "X-Access-Token: NzM2ZmI3ZTQyZDI0ZDk4OWExNTJmNTc1ZGQ3MzYyMjg2NDkwODA1MDMwODQ0ODAzNjM
yNTgxYmI1YmIwYjBiOA==" \
> -H "Cache-Control: no-cache" \
> -k \
> -d '{
>     "frames": [
>         {
>             "text": "0",
>             "icon": 3219
>         }
>     ]
> }' \
> https://192.168.88.248:4343/api/v1/dev/widget/update/com.lametric.22e45a6407da88c0c938a
8aaf8f7406f/1
```

replace
with
yours

If value has changed - congratulations! You have successfully created your first LaMetric Time app! If it didn't, make sure X-Access-Token and URL are specific to your app and valid.

2.2.2 Poll Indicator App

Use "Poll" communication type when data you are trying to display changes often, requires authentication or additional user parameters. In this case indicator app will poll for data by itself with some predefined time interval. To simplify things a little, let's use DropBox as a backend.

Create Indicator App

Prepare our backend by creating and uploading file named *data.json* to DropBox with the following contents:

```
{
    "frames": [
        {
            "icon":3219,
            "text":"2"
        }
    ]
}
```

Get the link to your *data.json* file by pressing "Share" button next to it and choosing "Copy Link".
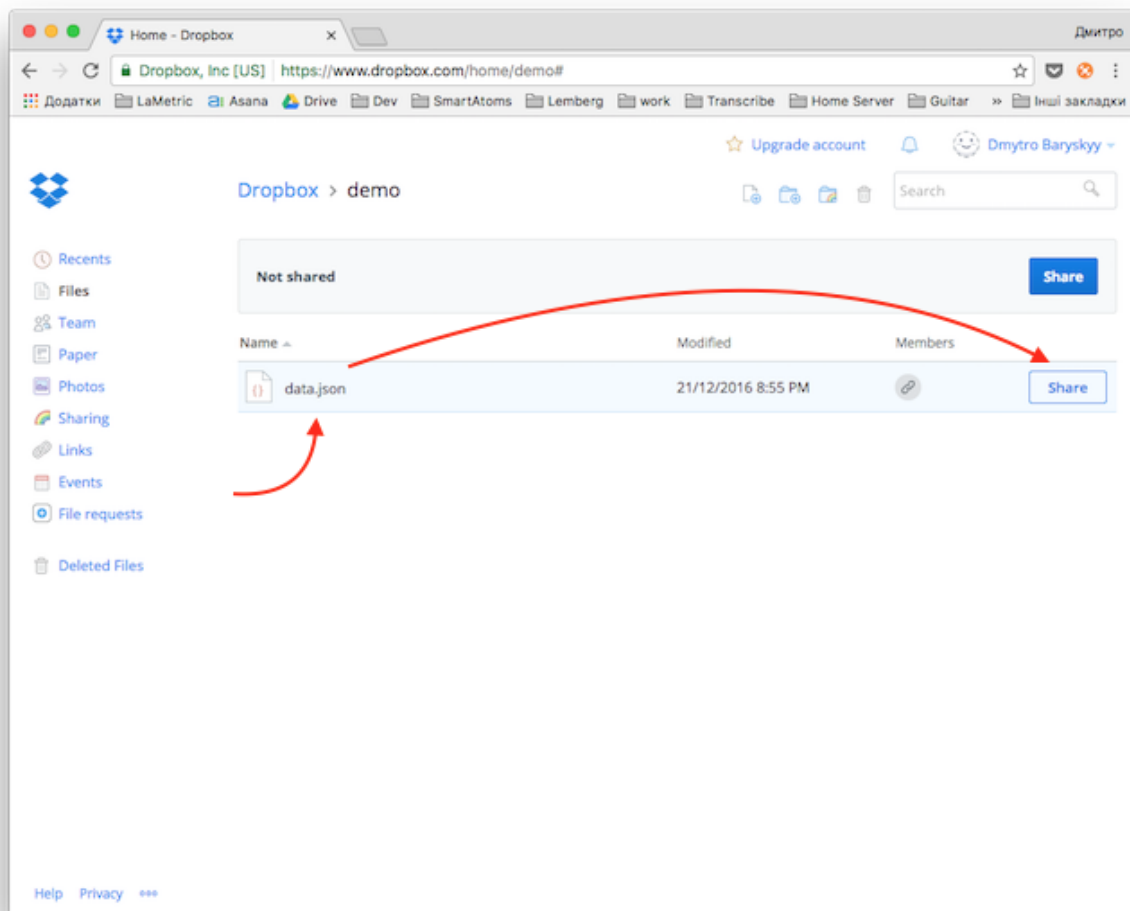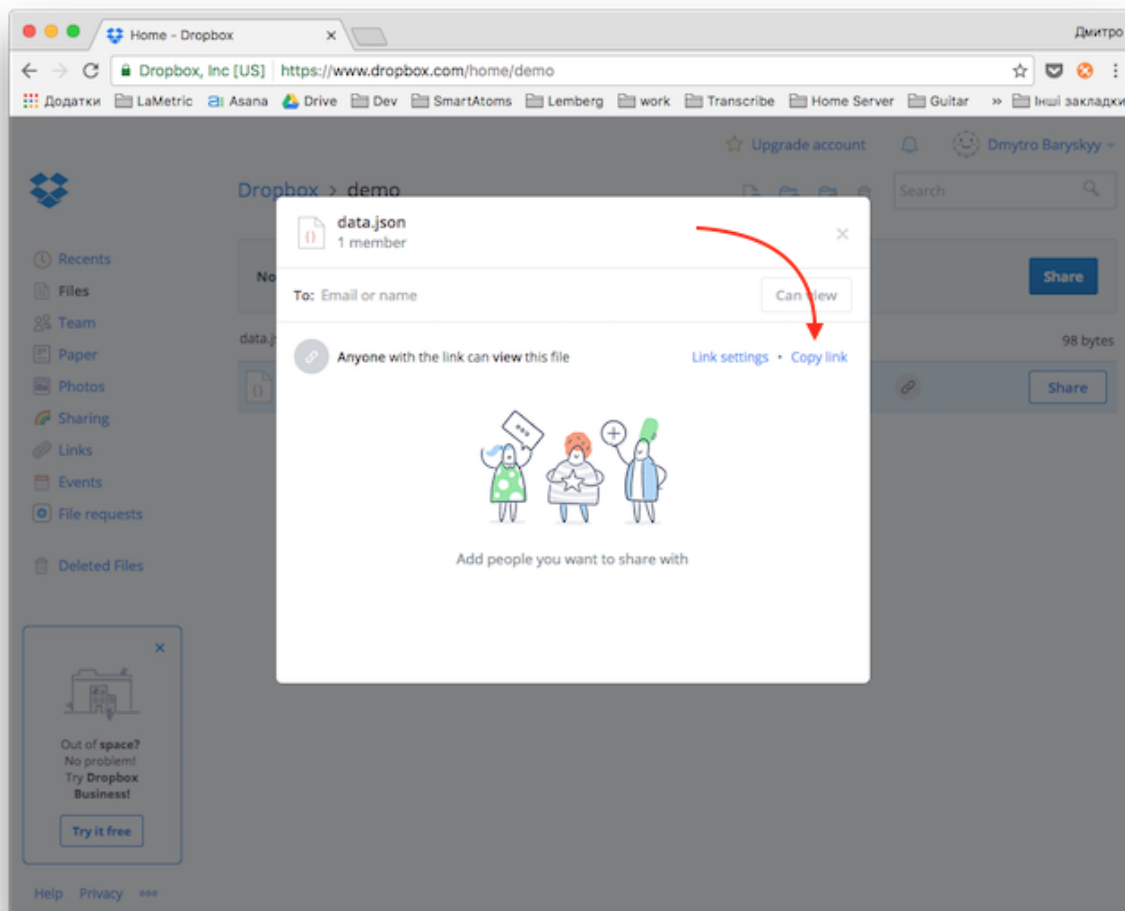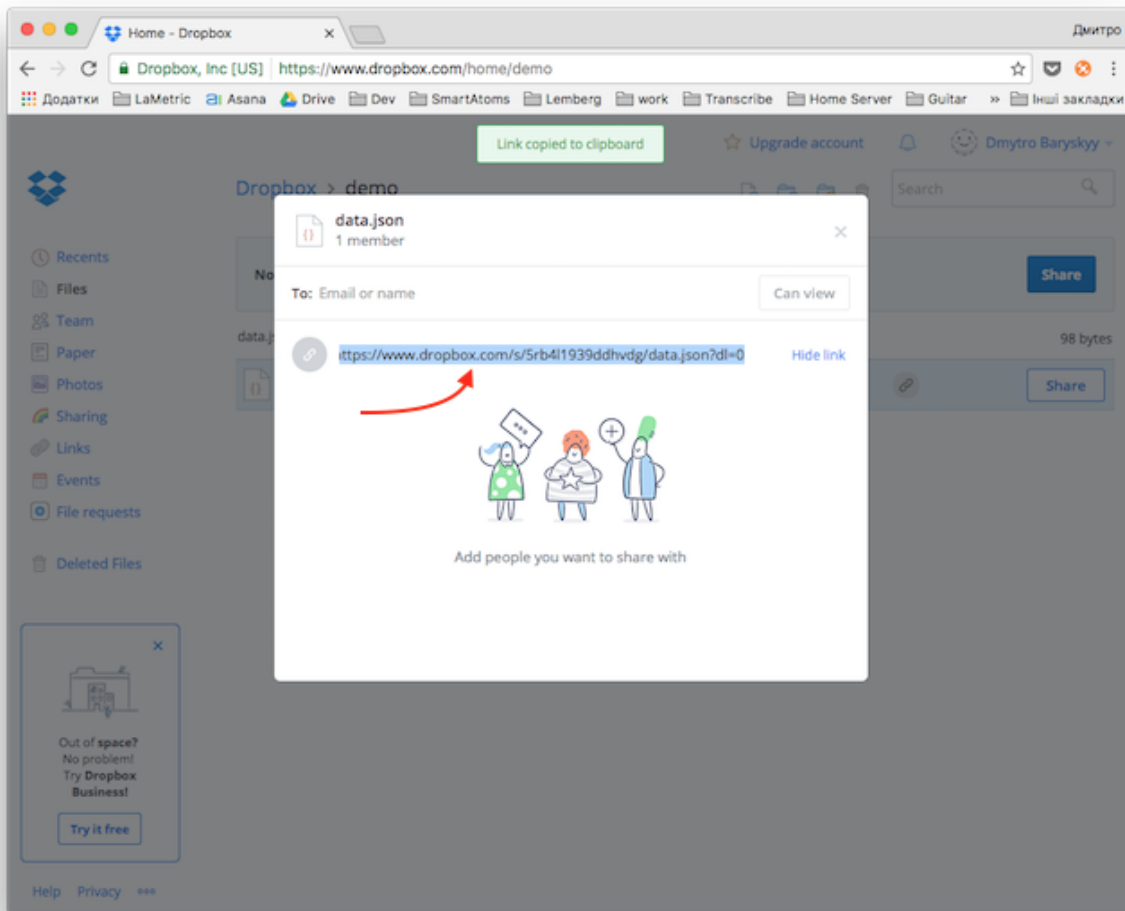
In our case it should look like this https://www.dropbox.com/s/5rb4l1939ddhvdg/data.json?dl=0. But, to make it work with LaMetric Time link must be direct. To make it direct let's change `dl=0` at the end to `dl=1` - https://www.dropbox.com/s/5rb4l1939ddhvdg/data.json?dl=1

Let's modify our "Push" app we made in previous section and make it poll for the data - just change the "Communication type" switch into "Poll" position.

Insert the link to *data.json* file into "URL to get data from" text field and choose "Poll frequency" to be set to some reasonable amount of time, for example 30 seconds.
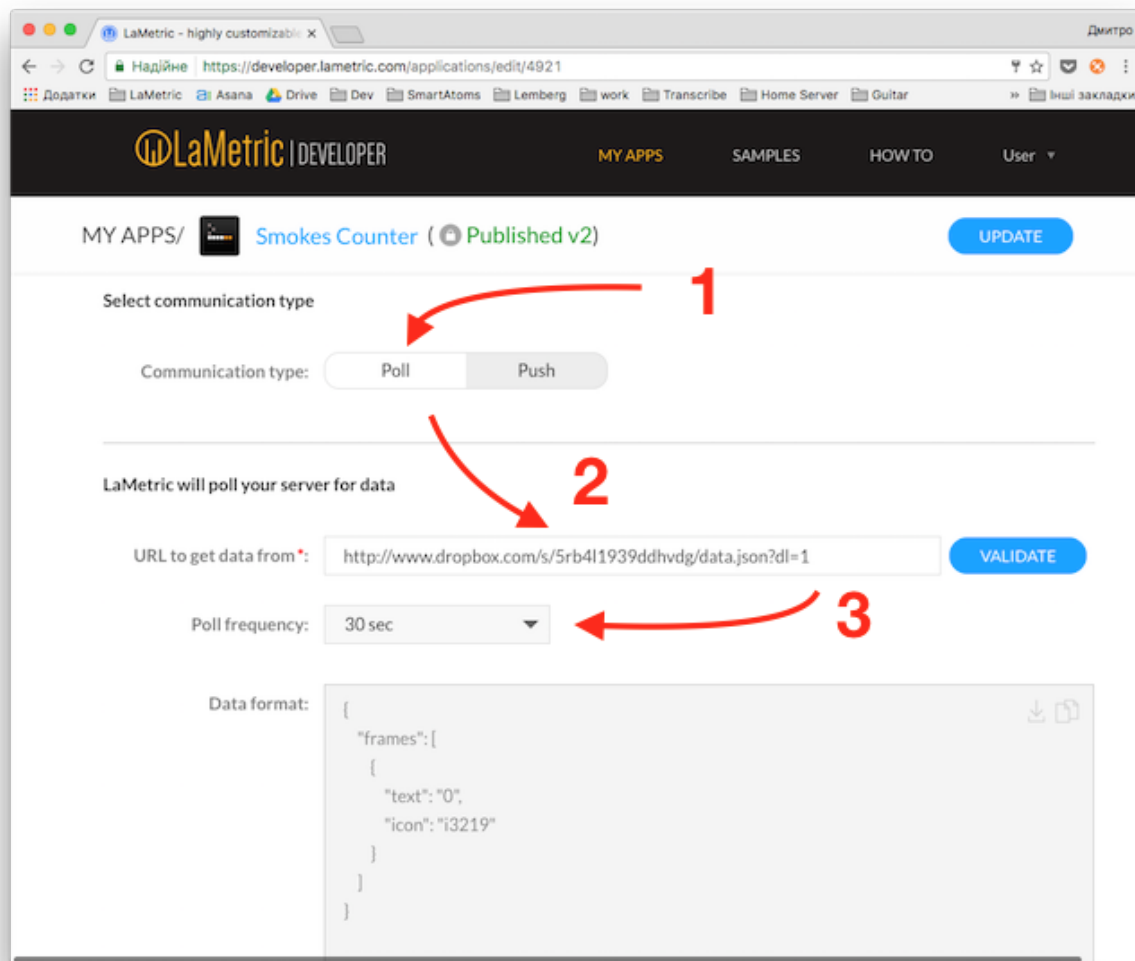
Publish app and install it to your LaMetric Time

Finally, lets publish our modified app by pressing "UPDATE" button at the top.

Once you received e-mail confirmation that your app is live - re-install it on your LaMetric Time device.

Note: To delete LaMetric Time app, press and hold it's icon for a second and drop to "Delete" area on top.

Pull some data!

Let's try to change the number displayed on the device. Modify data.json file and change text value to "3":

```
{
    "frames": [
        {
            "icon":"i3219",
            "text":"3"
```

```
            }
        ]
}
```

Save the file and upload it to the DropBox. Within 30 seconds the number will change!
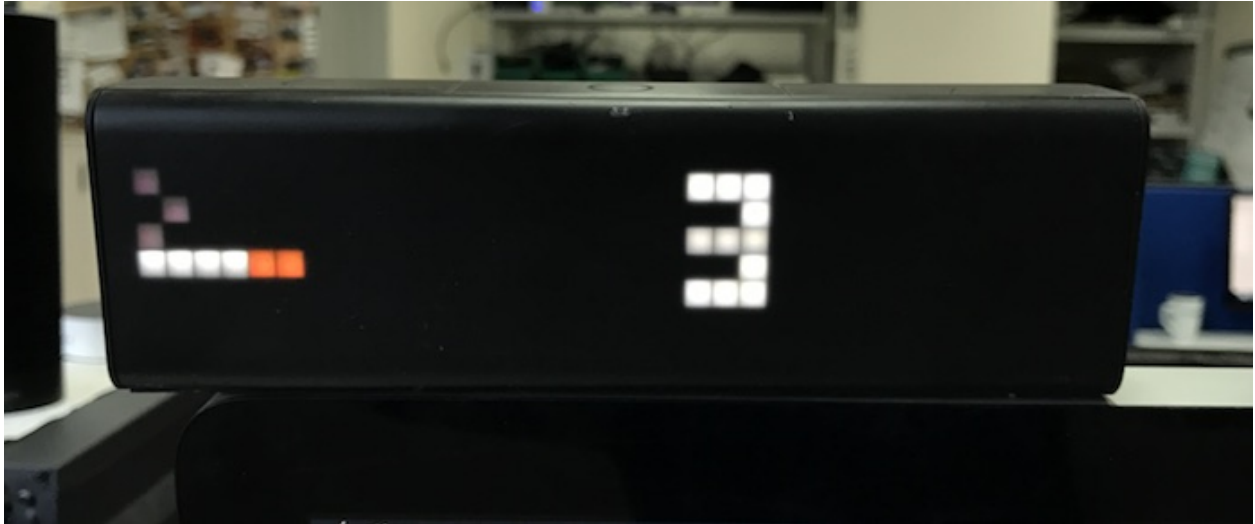
Congratulations! You have just learned how to create two types of indicator apps for your LaMetric Time!

2.2.3 What next?

To learn more about LaMetric Indicator apps check other tutorials here. Check *First Button App* section to learn more about Button apps.

2.3 First Button App

Button app is a native LaMetric Time app that allows to do things when you press an action button. We will use zapier.com to demonstrate how it works. Zapier is a service that can do something (like send email) in response to some event. We will use LaMetric Time button app to trigger such an event.

Let's create an app that will notify wife when you are going home.

To do that, we will need to:

1. Create "Button App" at developer.lametric.com

2. Create new Zapp (a.k. rule) at zapier.com

3. Publish you "Button App" and test how it works.

So, let's get started!

2.3.1 Create Button App

Start by logging in to your developer account and create new button app.
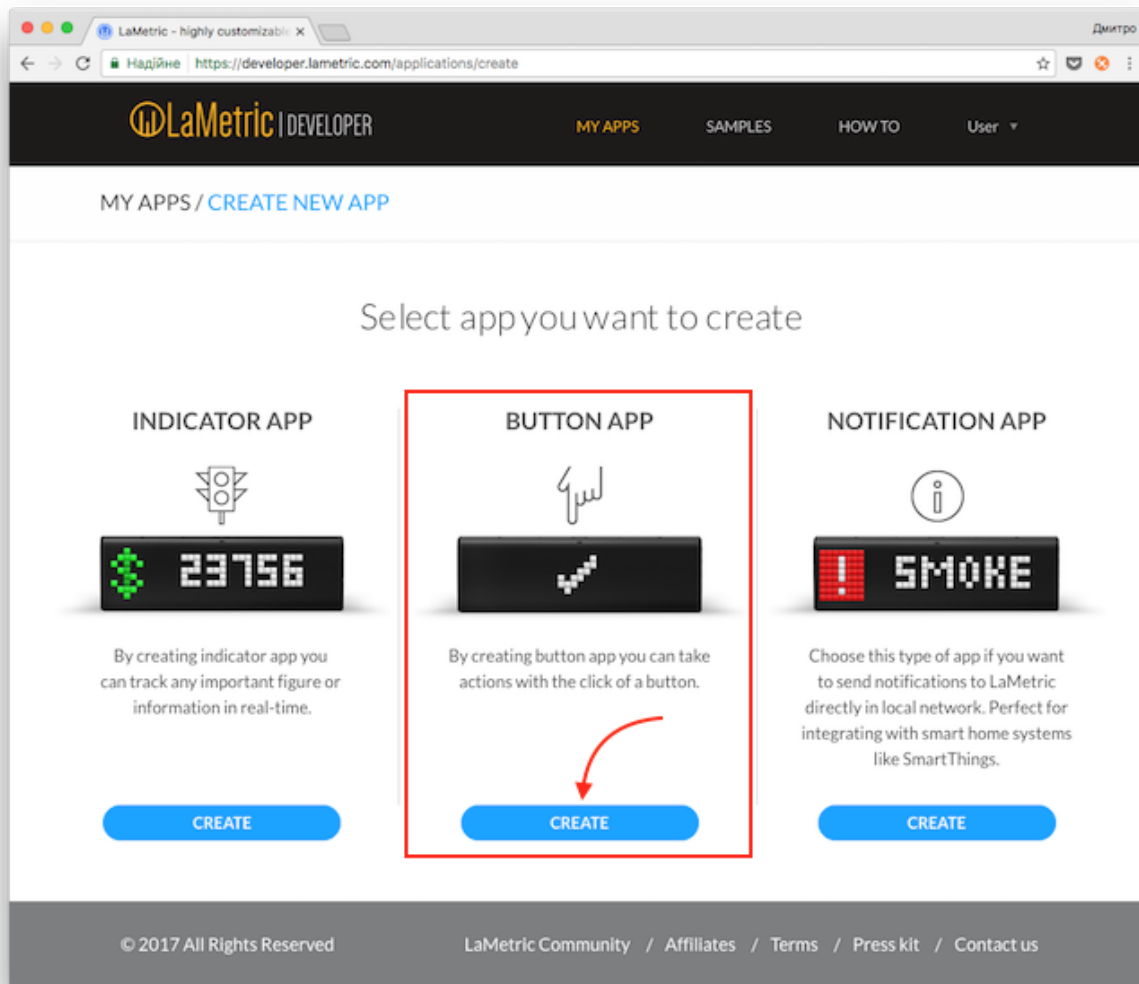
Now add "message" parameter that will allow us to change the text that will be sent. Choose icon and enter text that will be displayed on the LaMetric Time when app is active.

As you noticed, now we need URL to put into "URL to be triggerd" field. This is when Zapier comes into play.

Let's open zapier.com in separate tab and proceed with creating new zap.
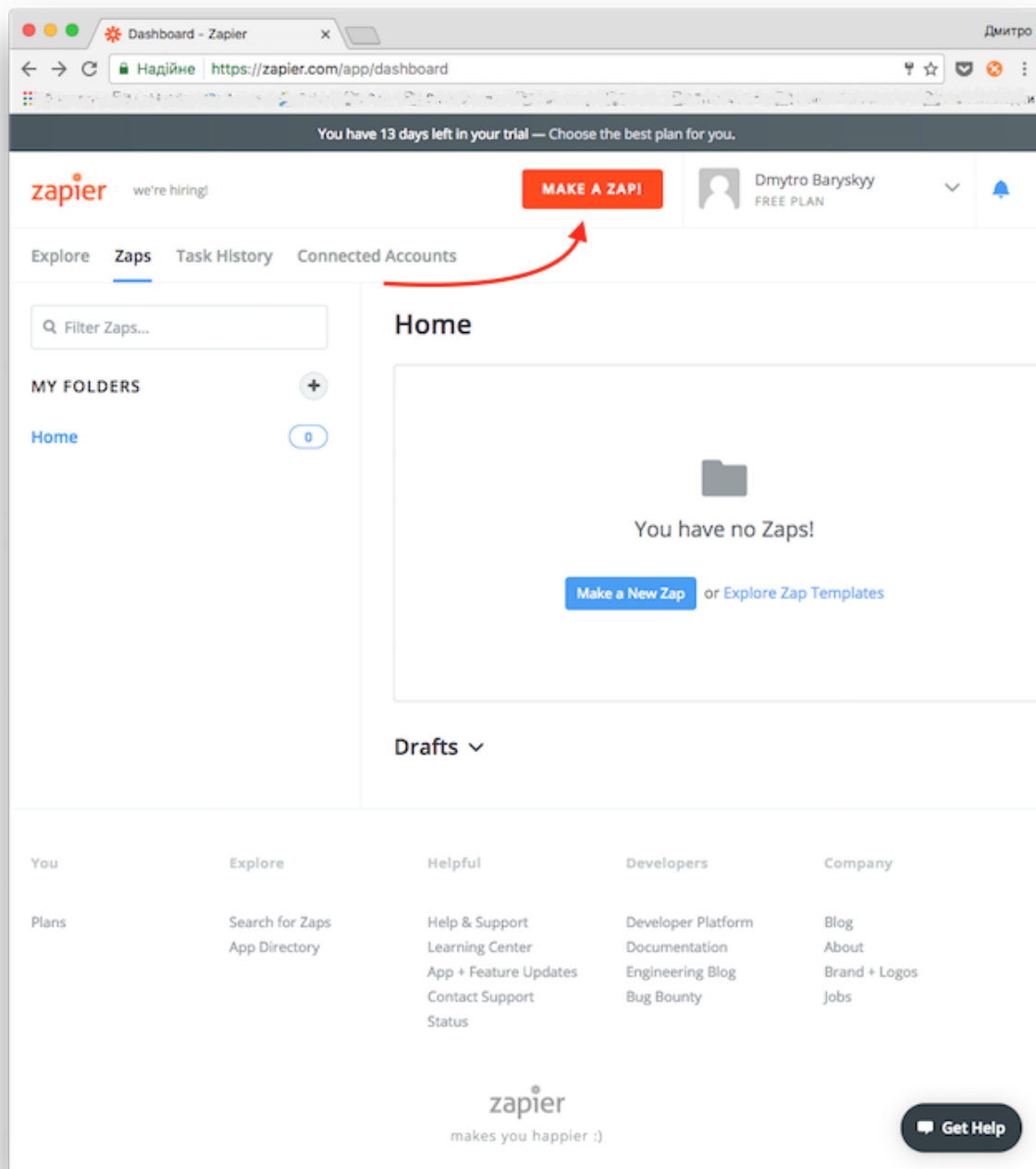
Let's search for "webhooks" triggers. Webhook is an API that we can call from our app to do some action.
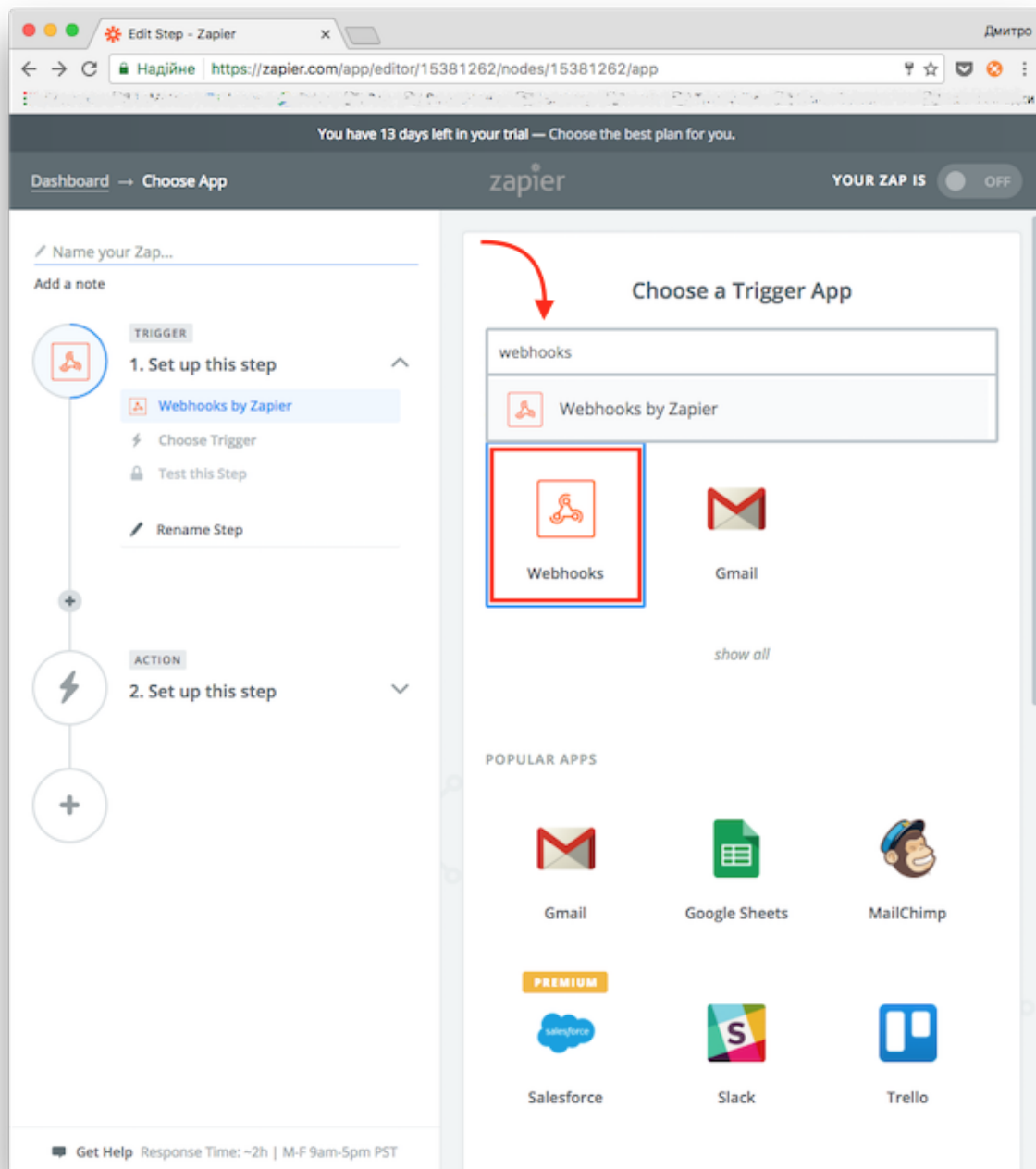
Now choose "Catch Hook" and press "Save + Continue" button. This will tell Zapier to create an API endpoint for us that we will be able to call.
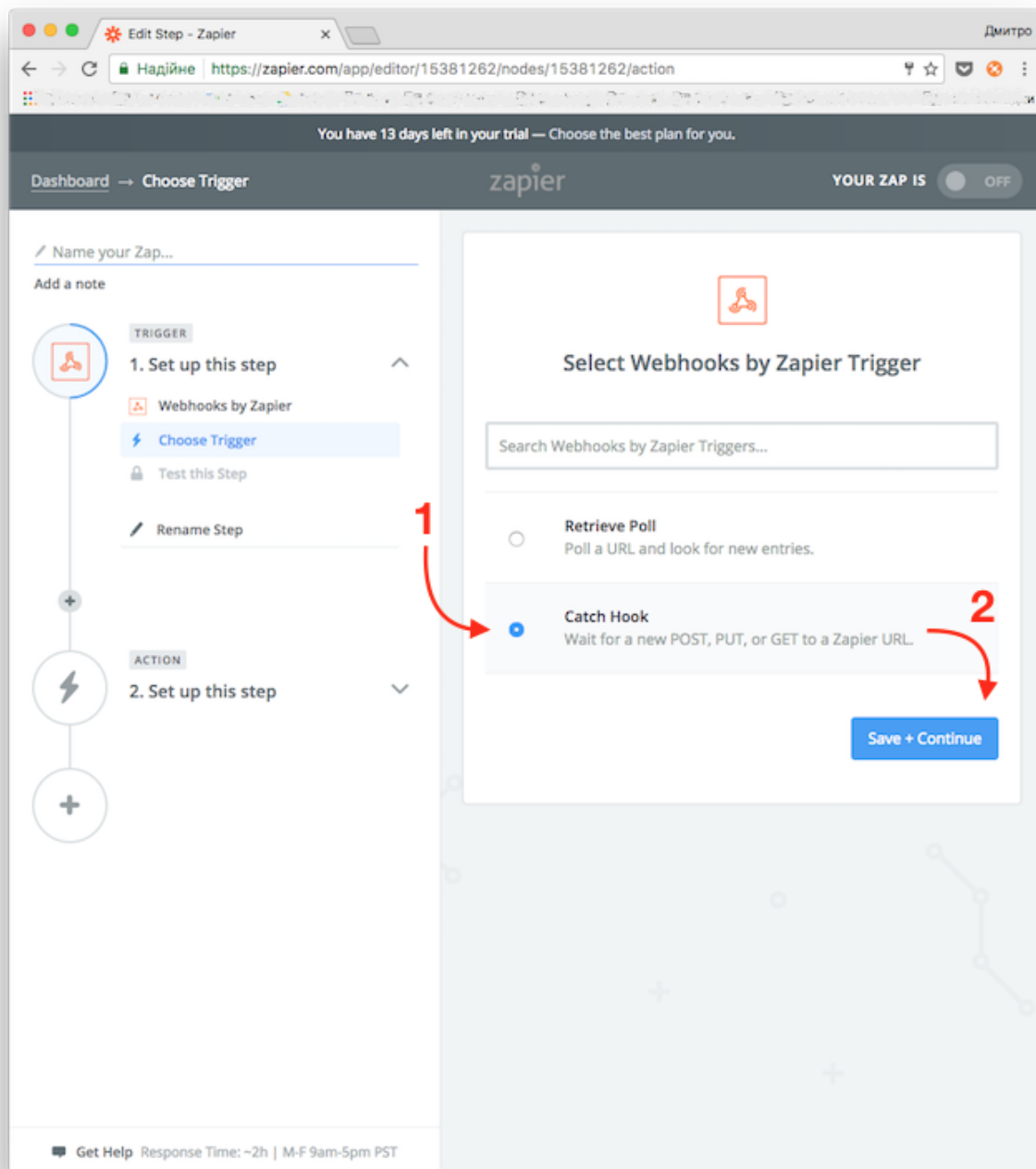
At this step enter parameter name we added to our button app - "message".
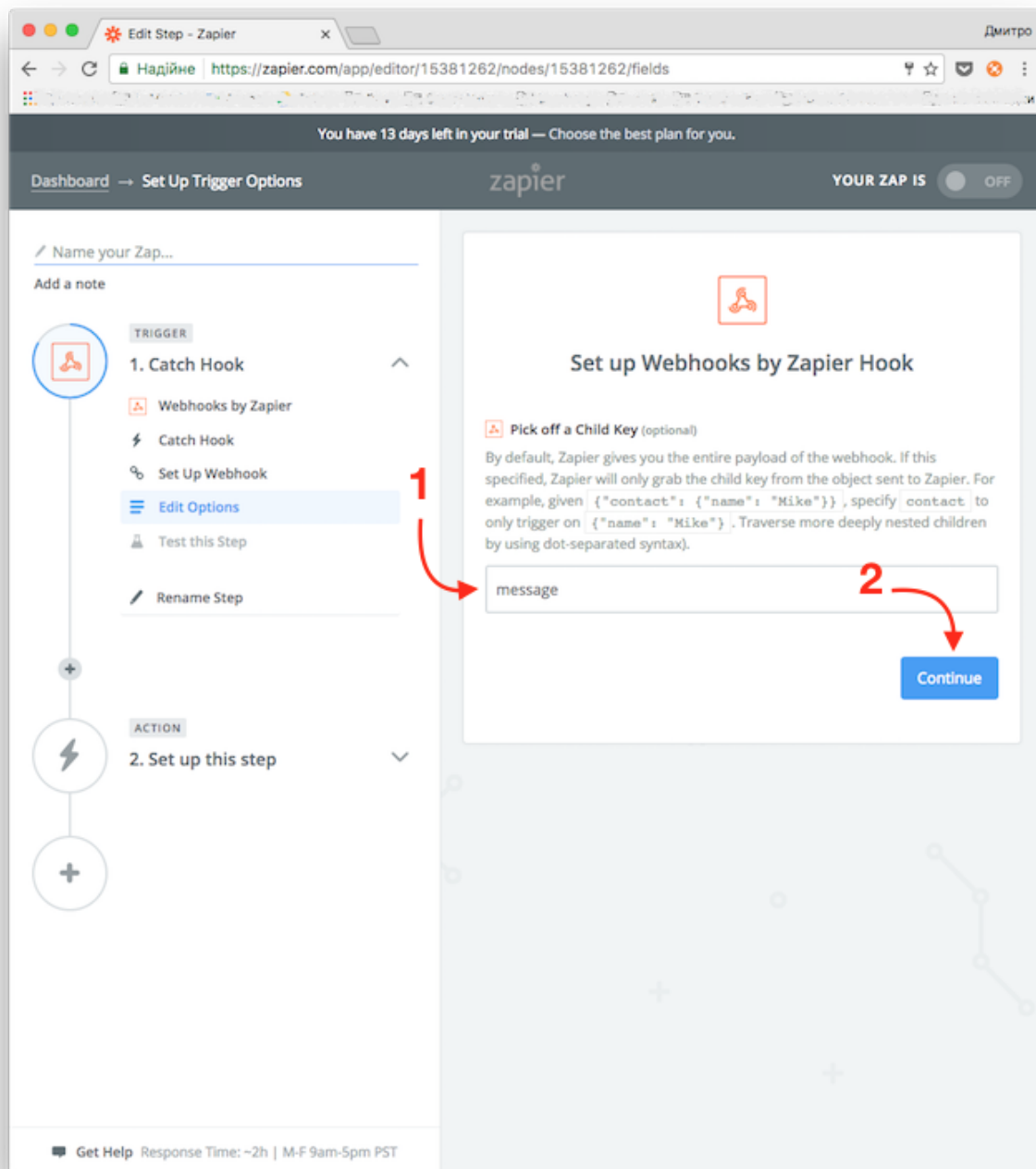
After this step you have the hook URL we can use in our button app. Copy it by pressing on "Copy to clipboard button".
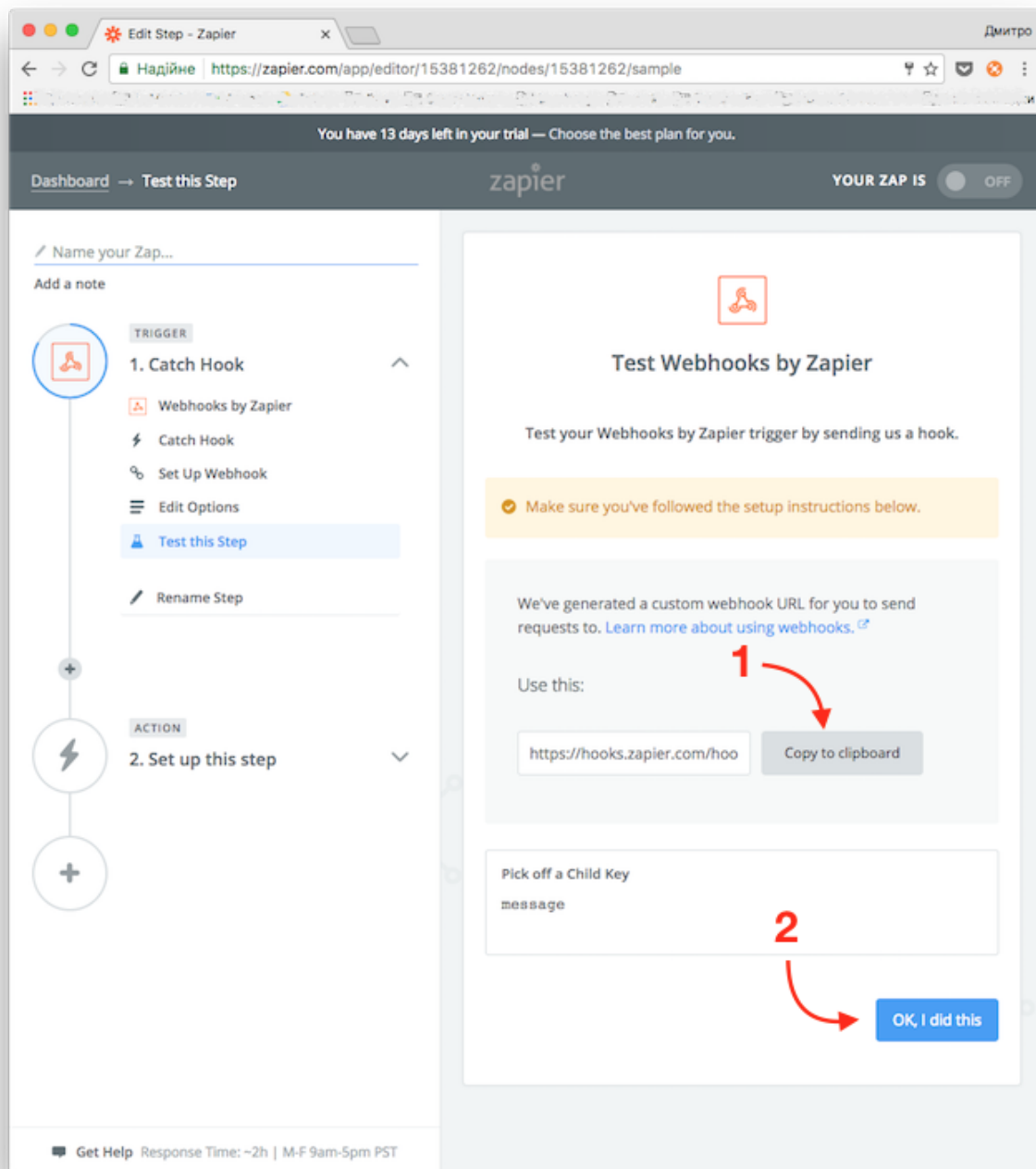
Lets switch to the tab where we have LaMetric Developer open and continue from step 5. Just paste the URL into "URL to be triggered" field and press "Next".

Now enter your app name and short description. As an example we will use "Wife Notifier" name.
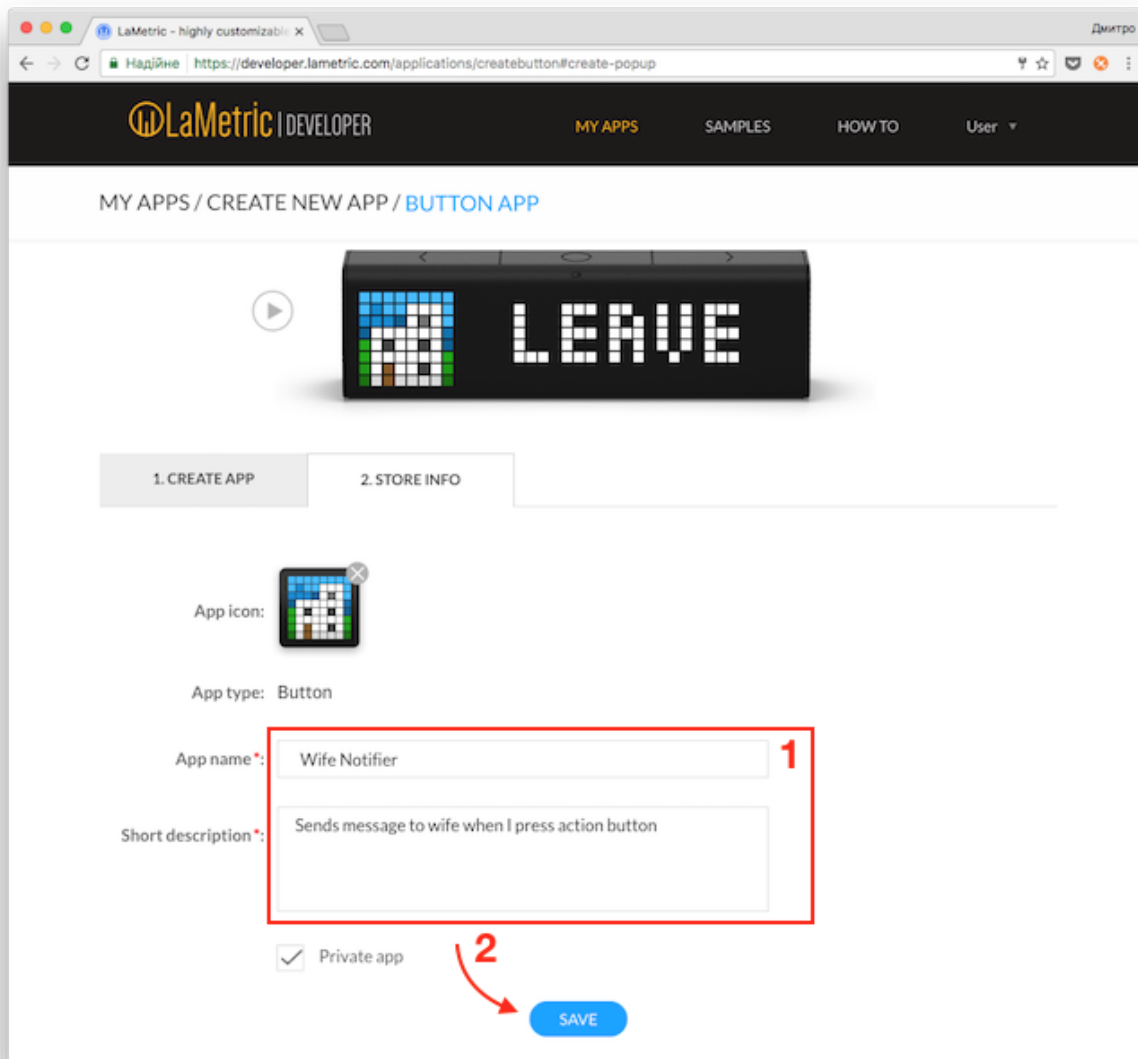
Ok, now publish app privately by clicking on "Publish" button and install it to your LaMetric Time device.
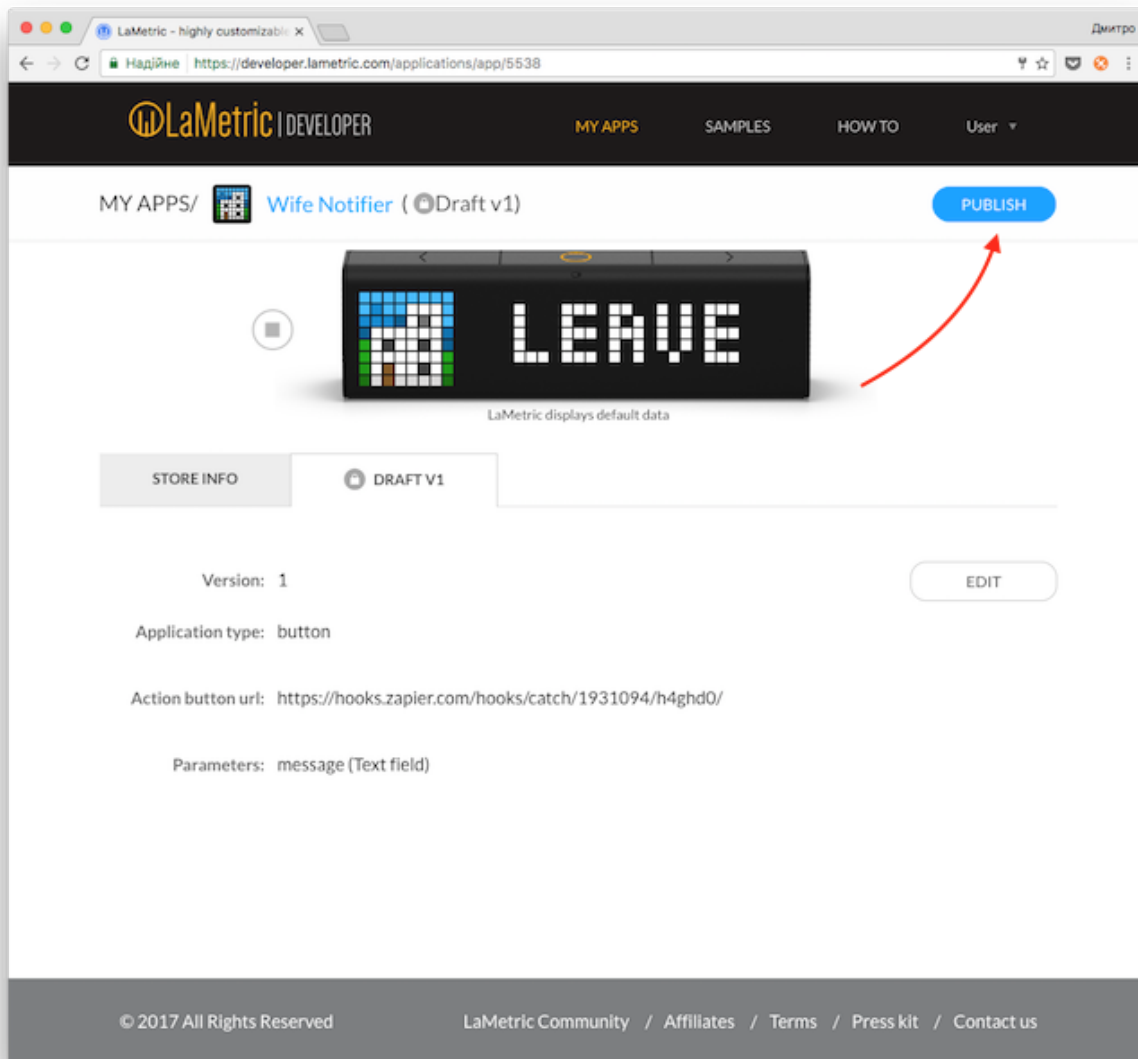
Open LaMetric Time app, go to your device settings and press on "+" button. Choose "Private" category, find your app and press "Add".
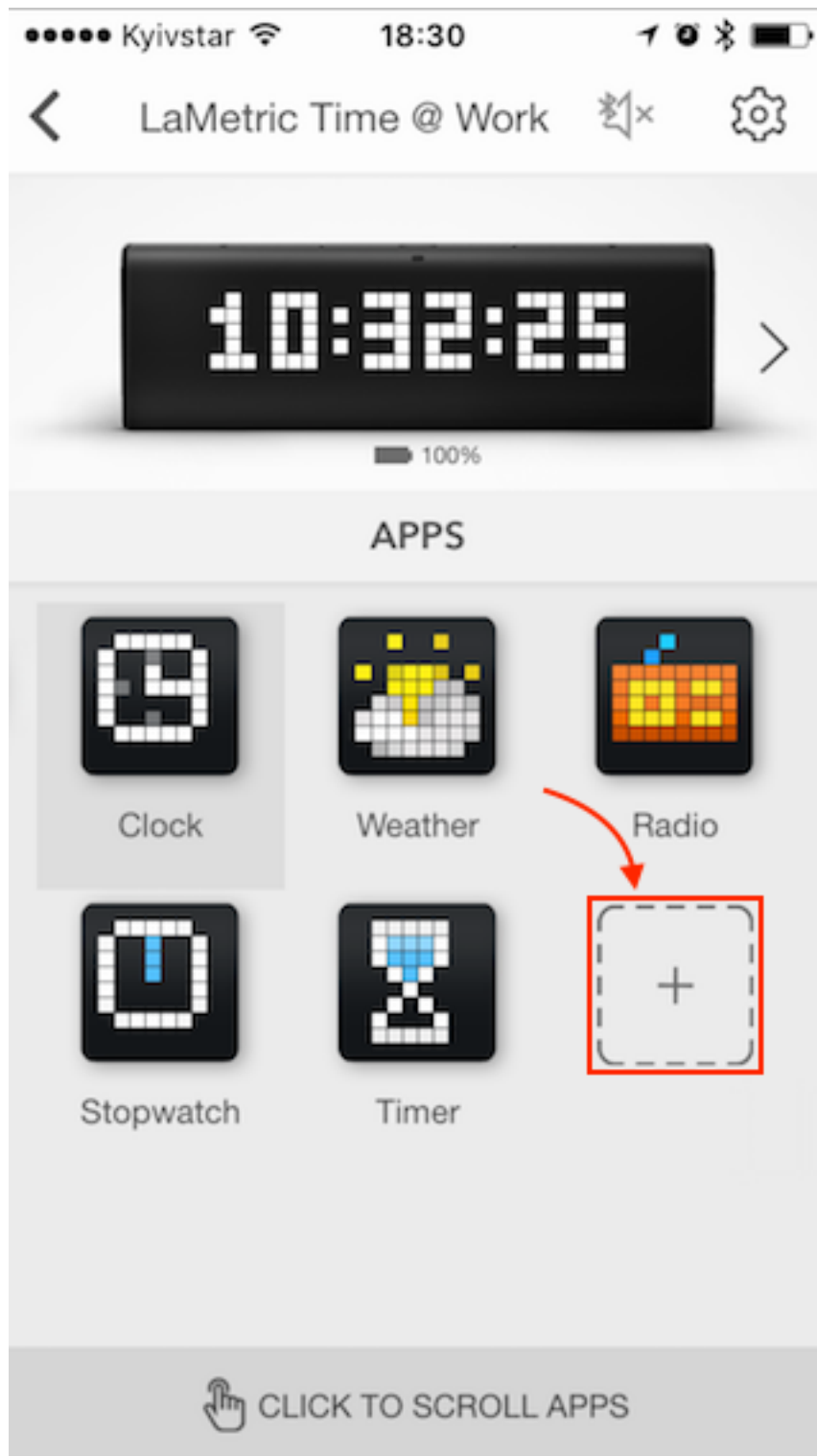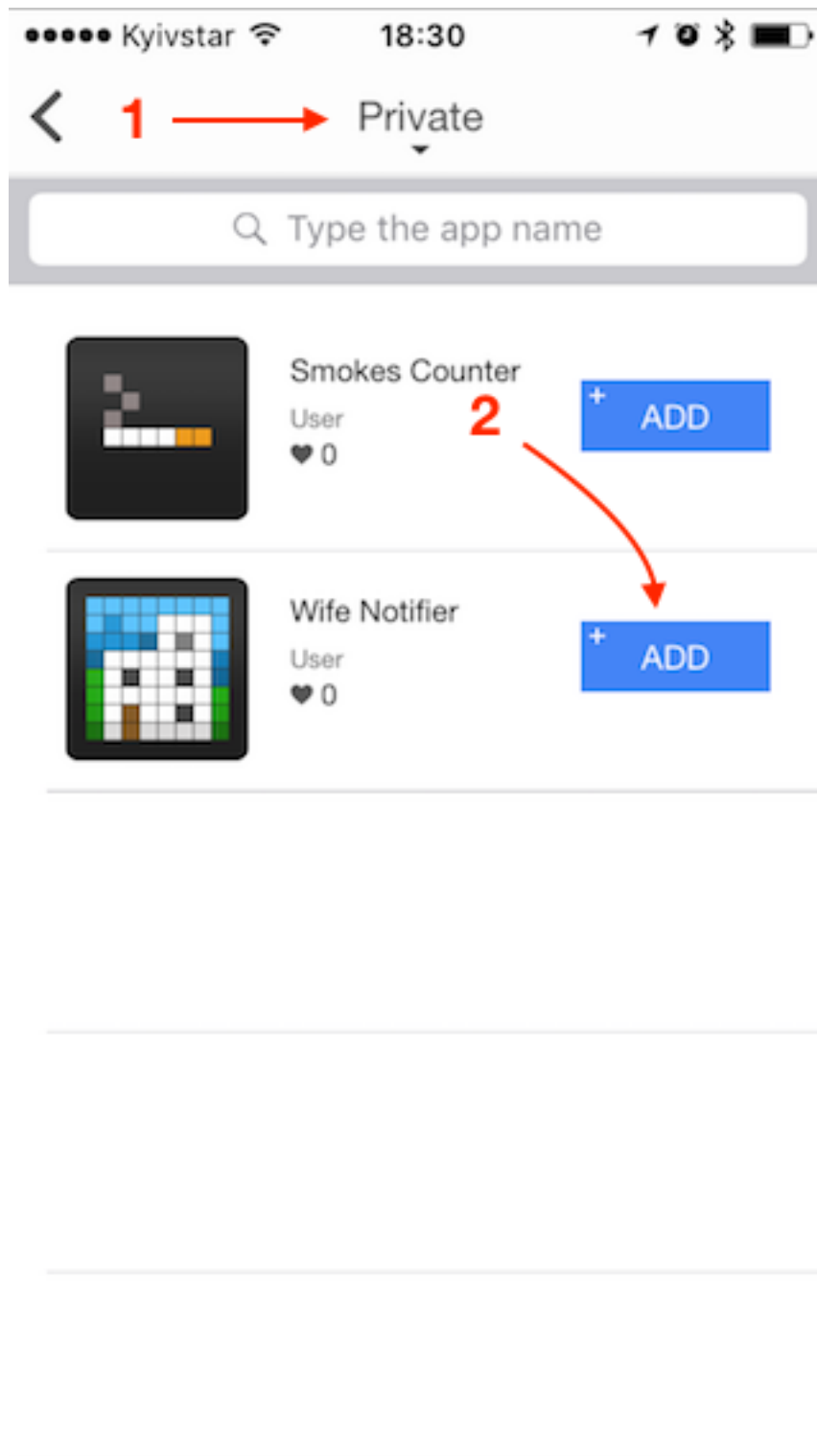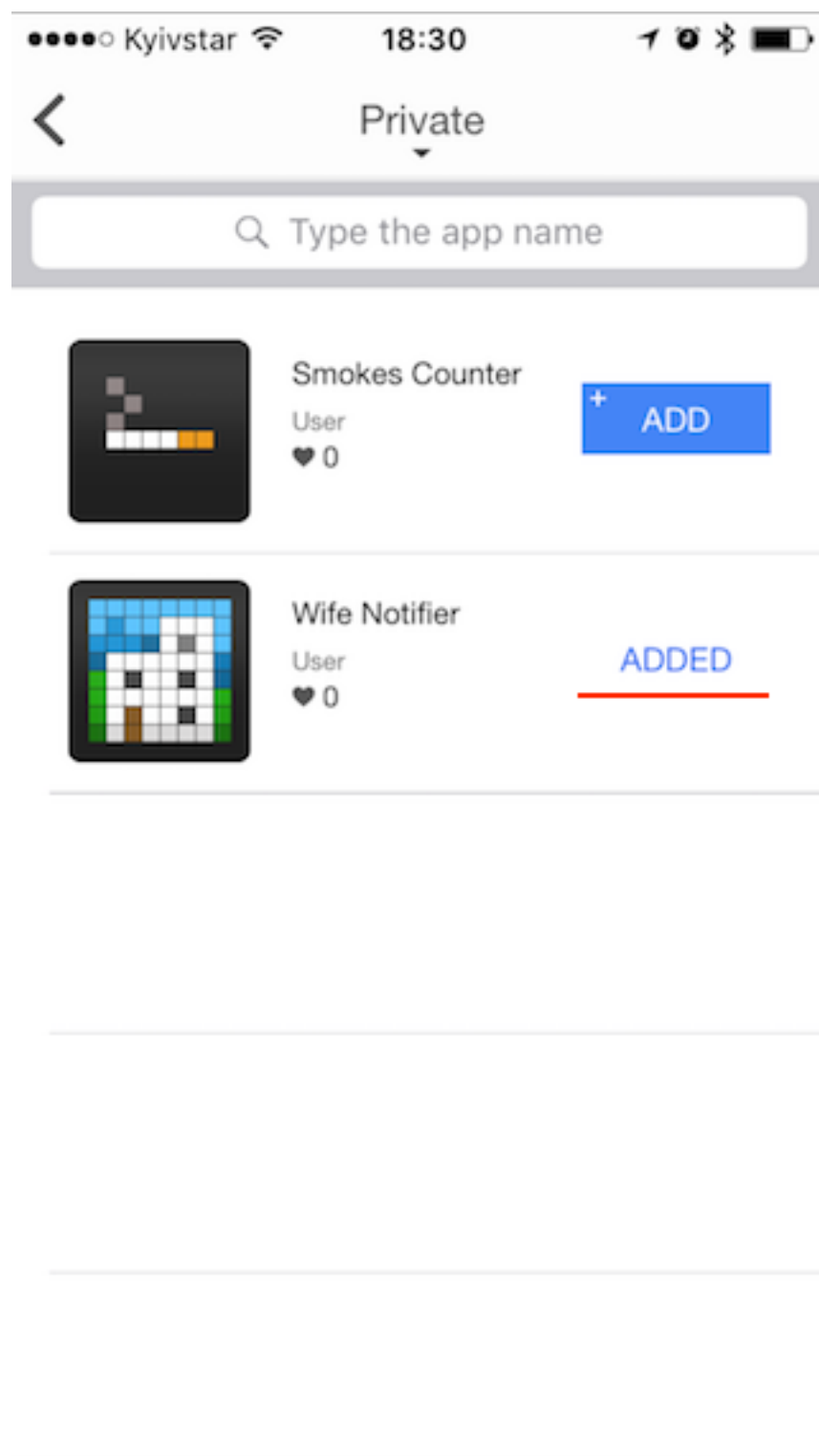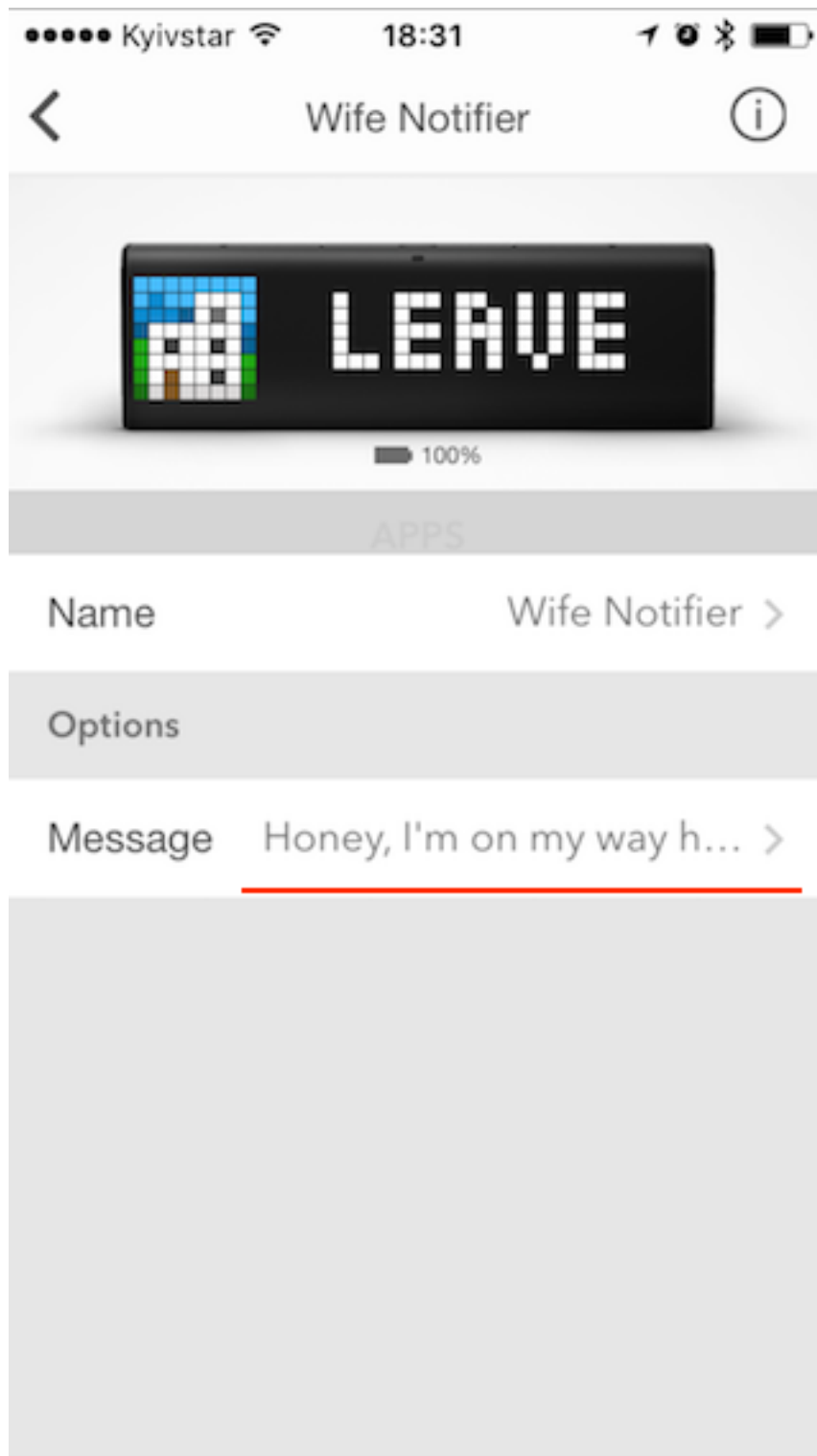
Now check your device - you should see "Leve" text (the one we added during app creation). Let's press "Action" button for a few times. This will cause hook URL to be triggered and zapier will know it works.

Let's get back to Zapier and continue with our zap. If you see "Test Successful" message - press "Continue". If not - try to press "Action" button on the device few more times.
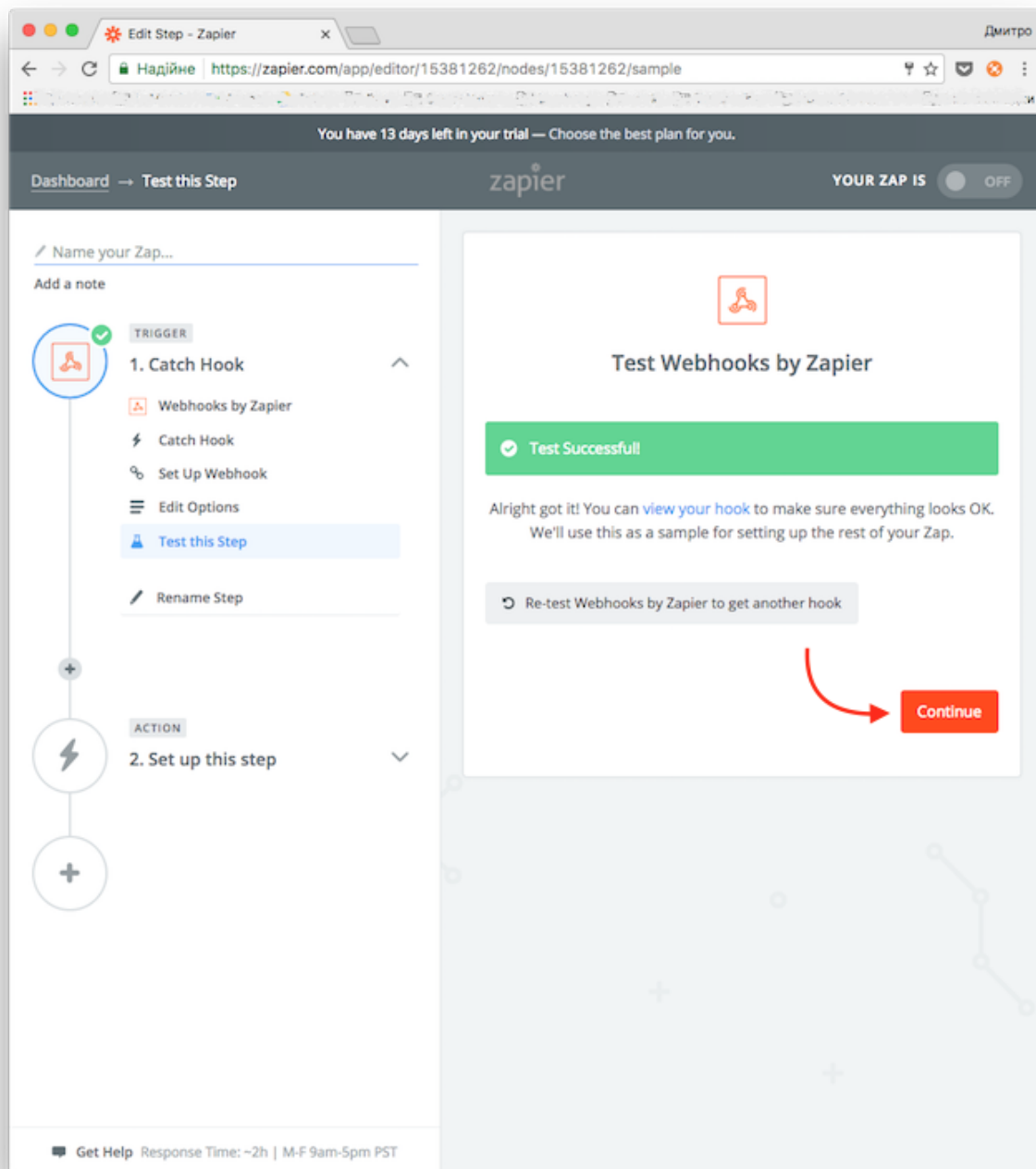
Now, when we are done creating trigger let's continue with action. Because we want to send an e-mail, let's find Gmail action by typing "Gmail" into action search field. Choose "Send Email" action and press "Save + Continue" button.
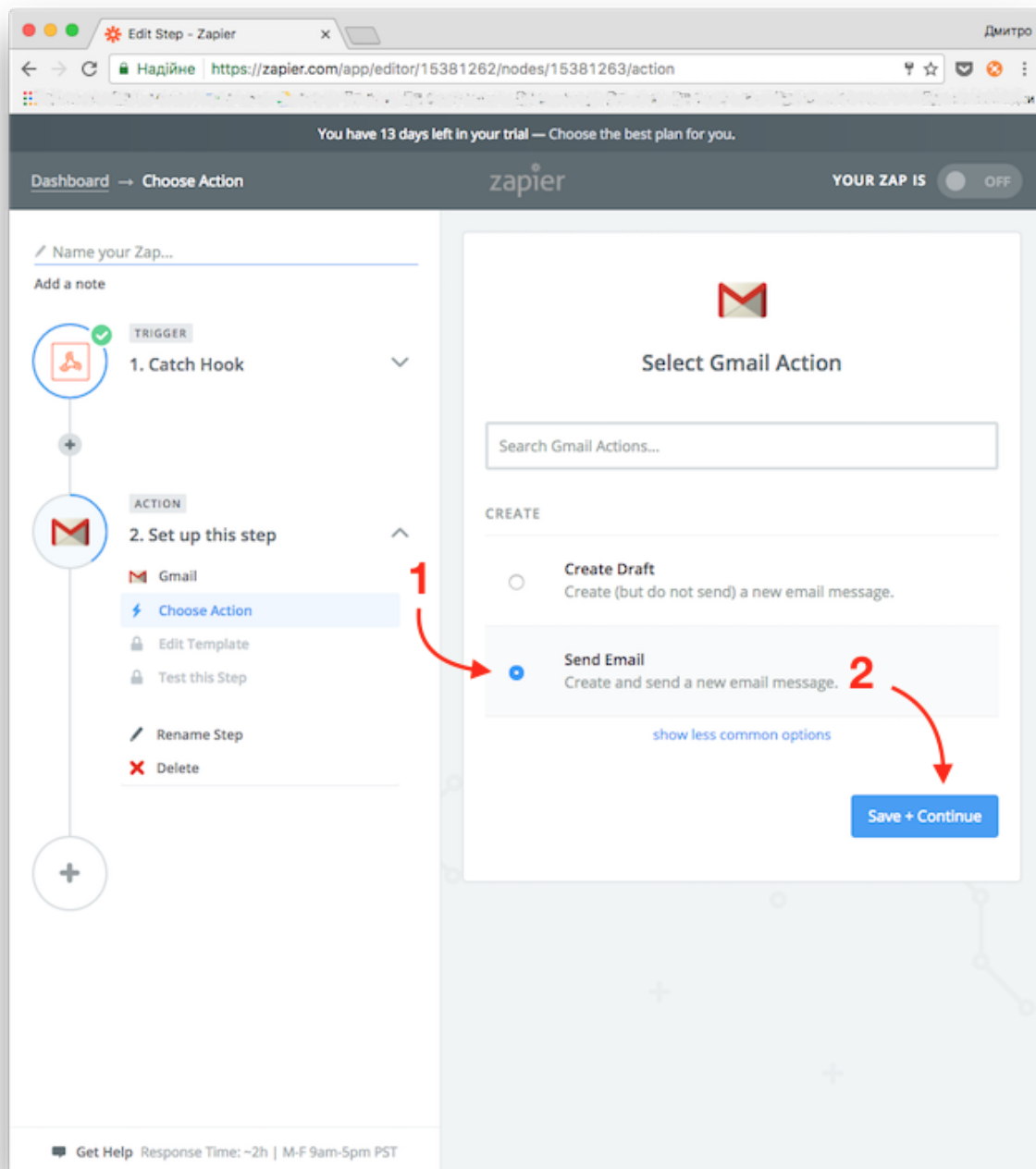
Login to you Gmail account and select it when asked, followed by "Save + Continue" button.
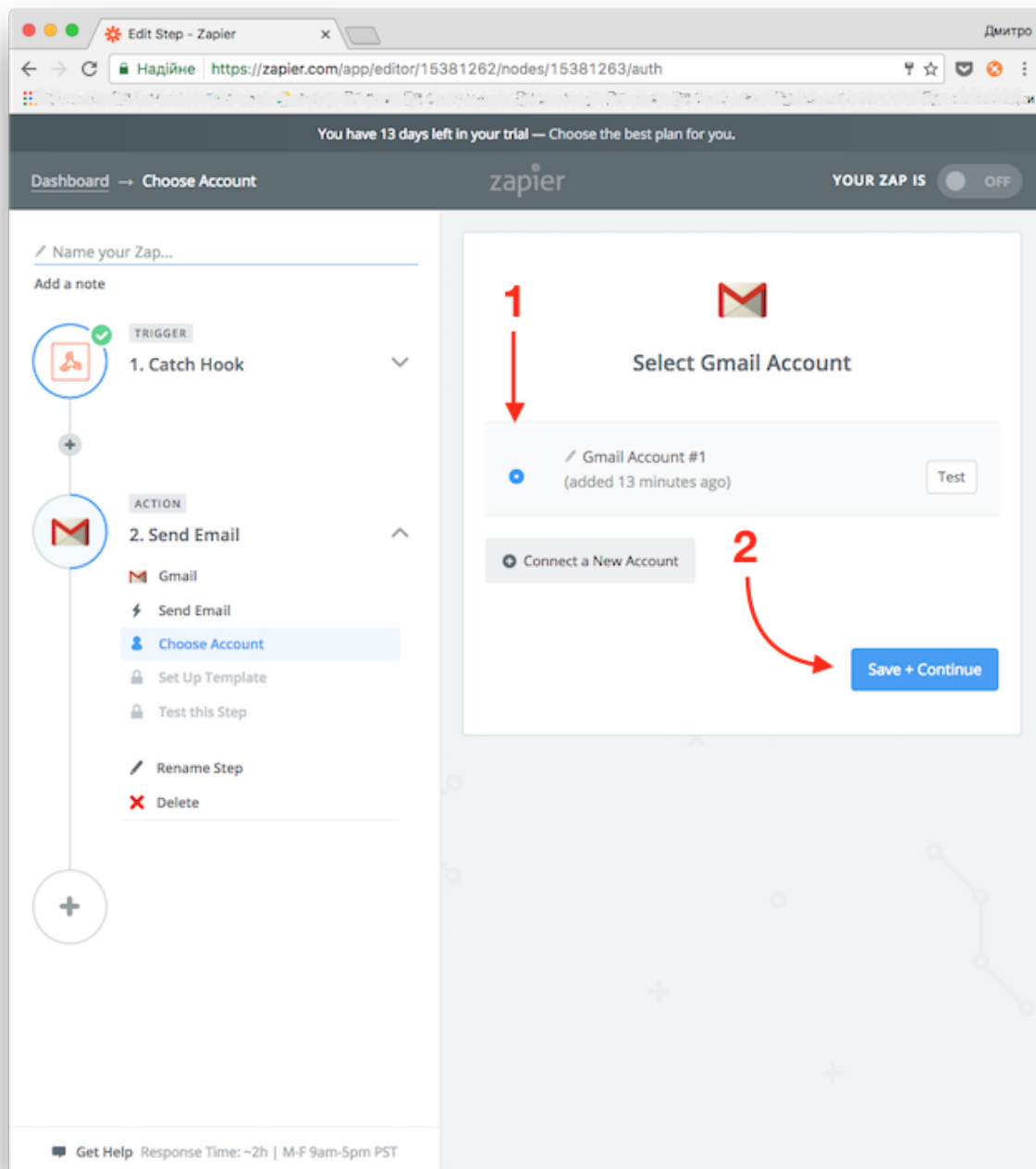
Lets fill out the e-mail form:

To wife@home.com (this may be your wives e-mail address)

Reply to me@work.com (this is your e-mail addres)

From name John (your name)

Subject I'm on my way! (subject of email)

Body here choose Querystring Message parameter. It will get replaced by the message from our app.
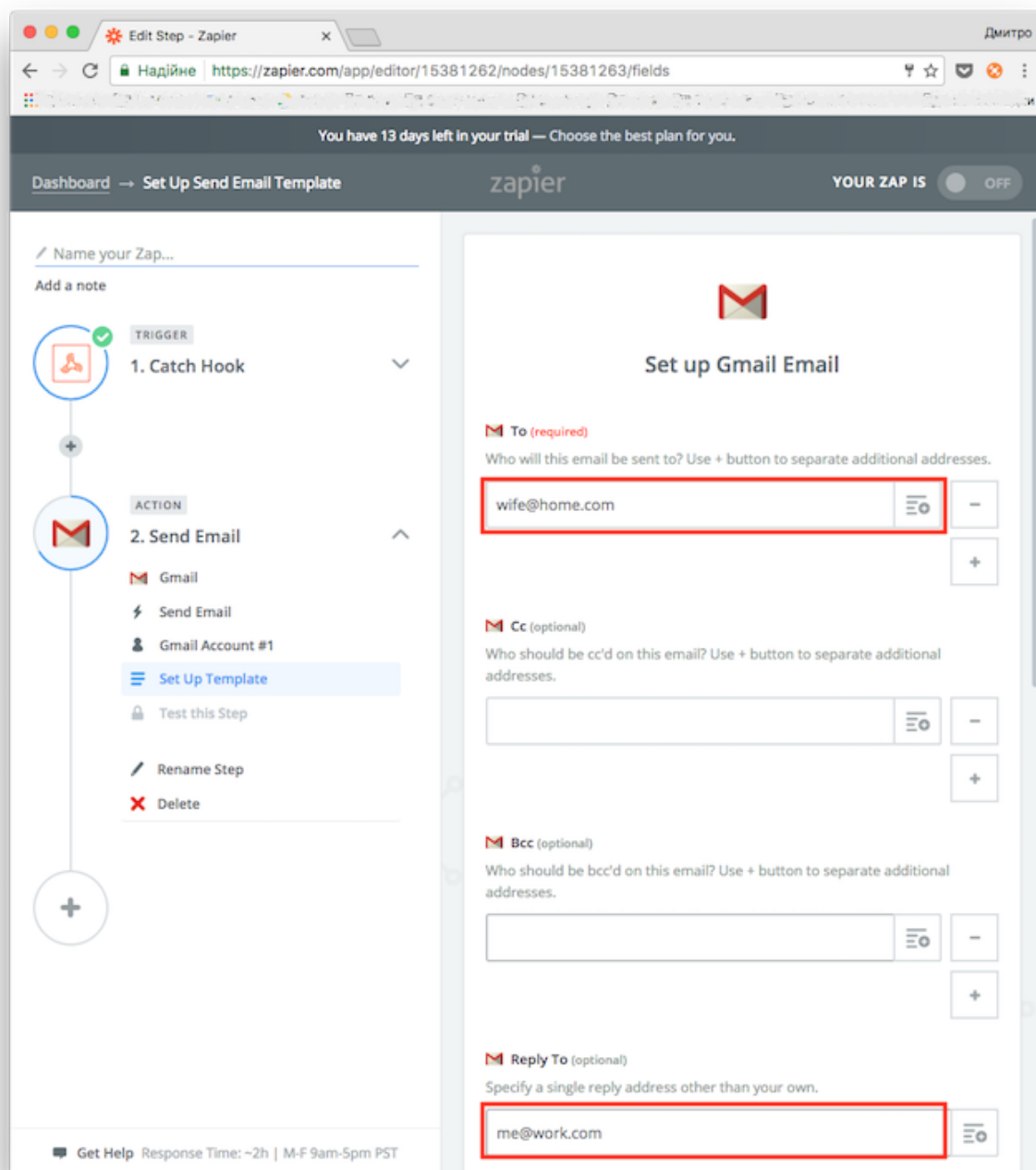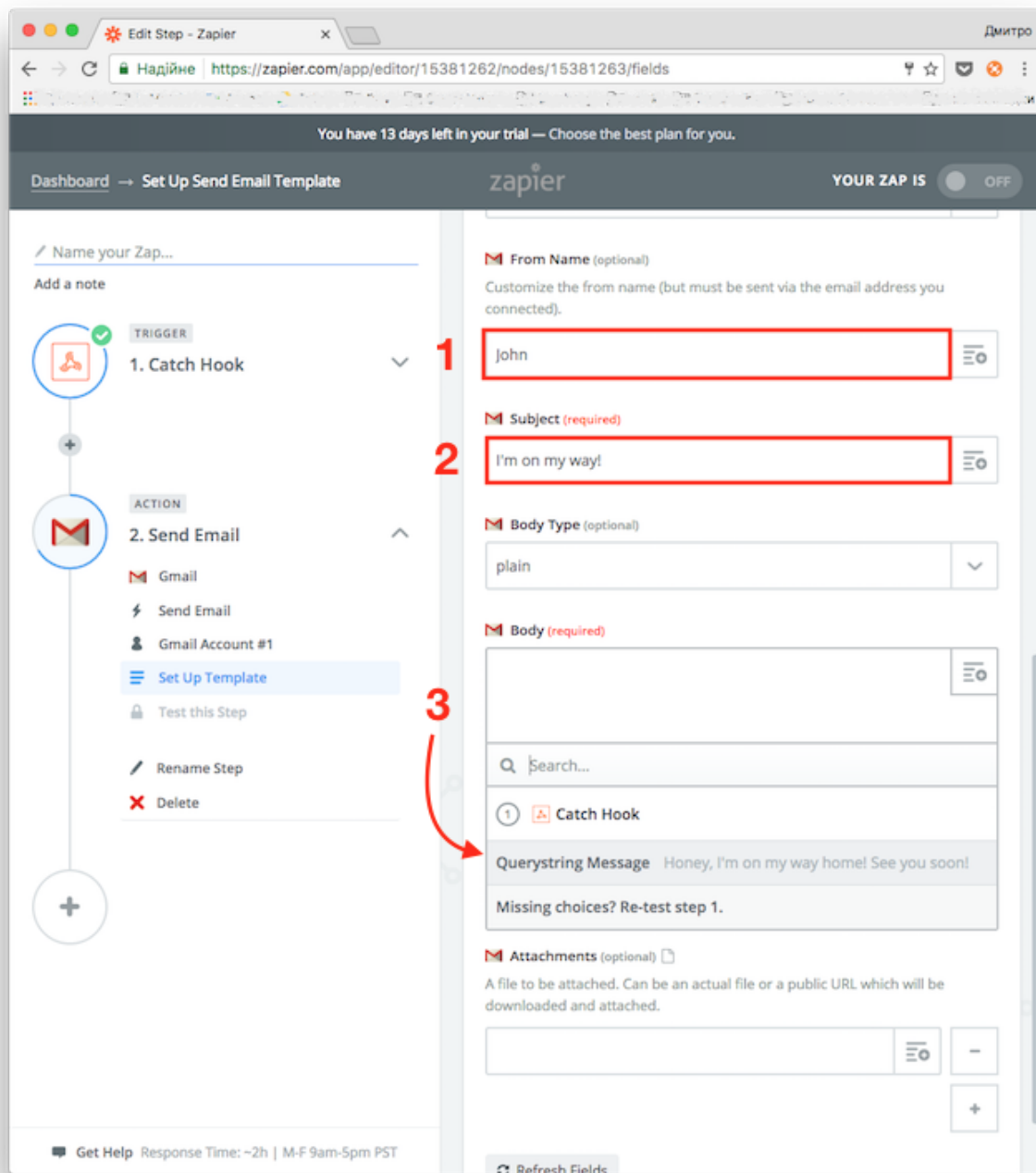
Now it is ok to click "Continue" button to proceed to the next step
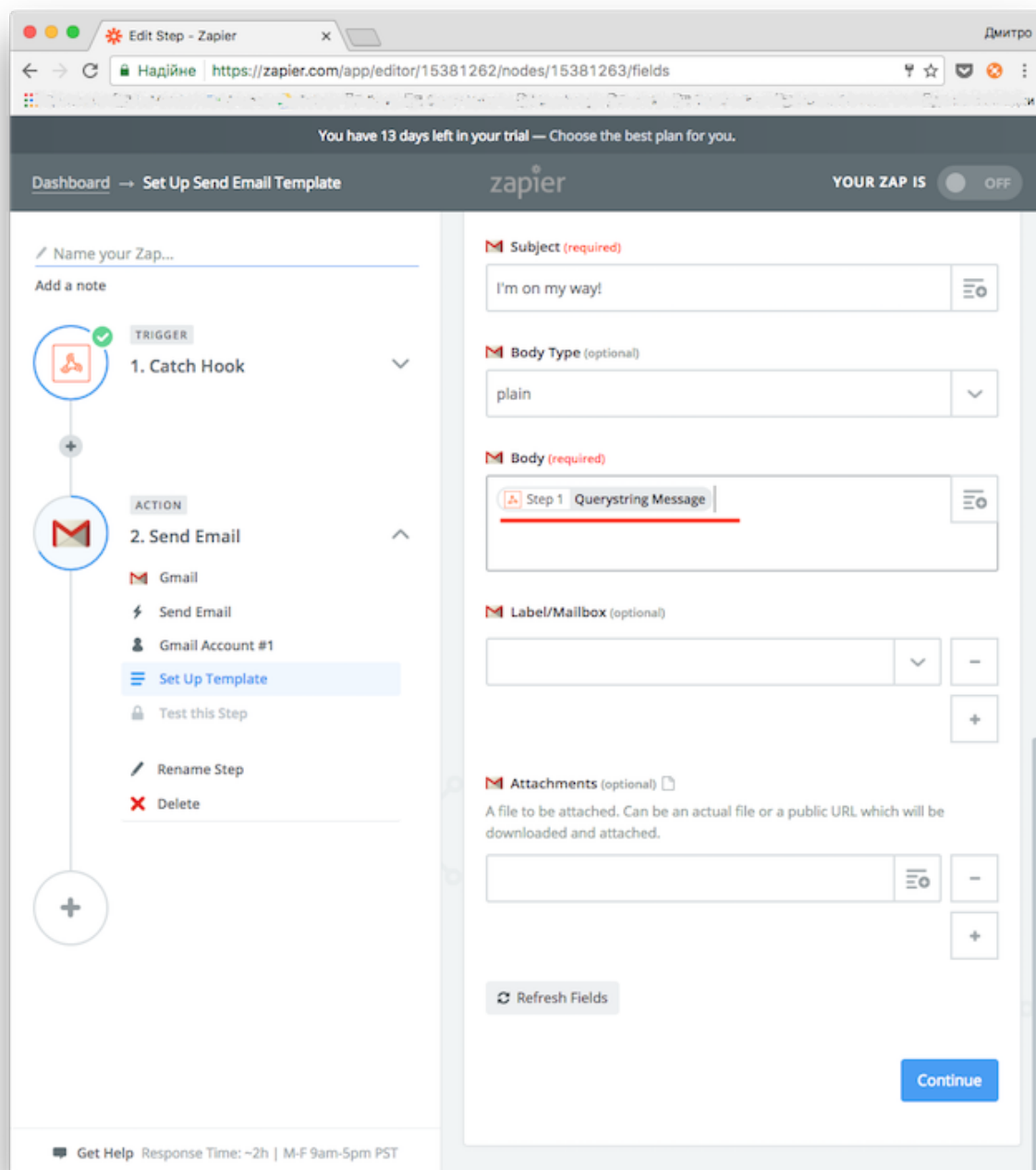
Check your e-mail and click on "Create & Continue" button.
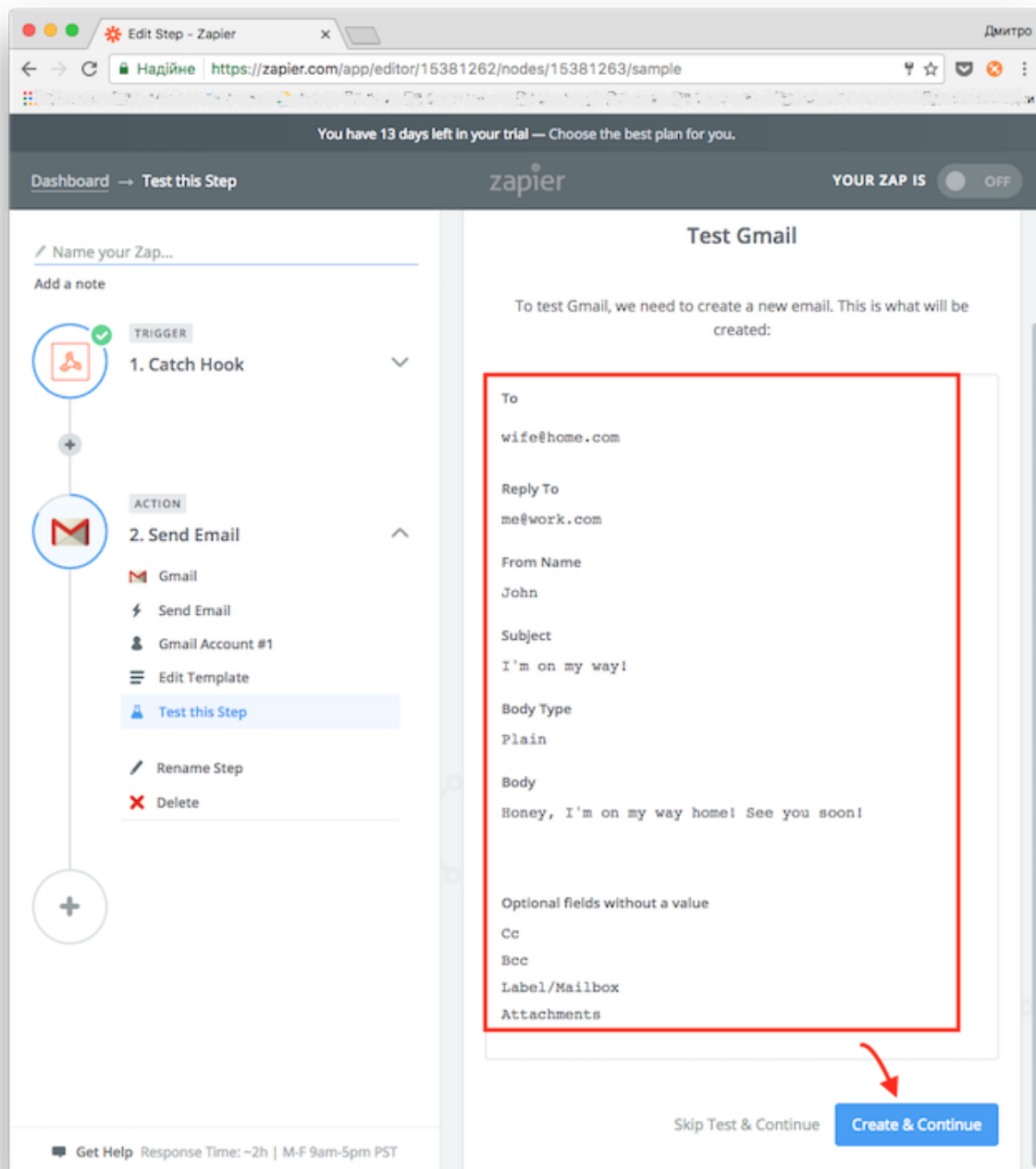
And finally here is our "Finish" button! Let's click it!
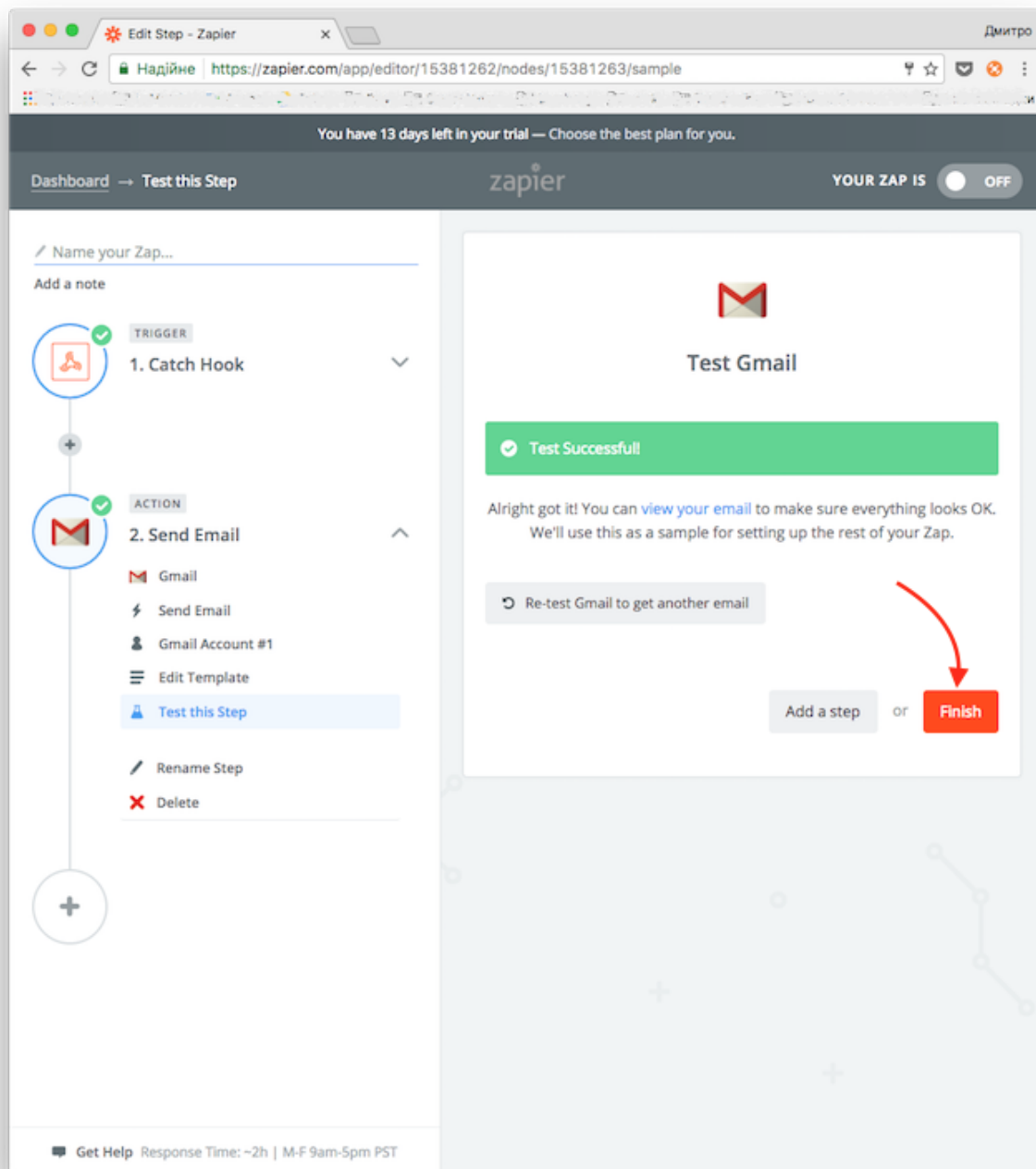
Final touch, lets choose the name for our zap - "Wife Notifier" would be good.
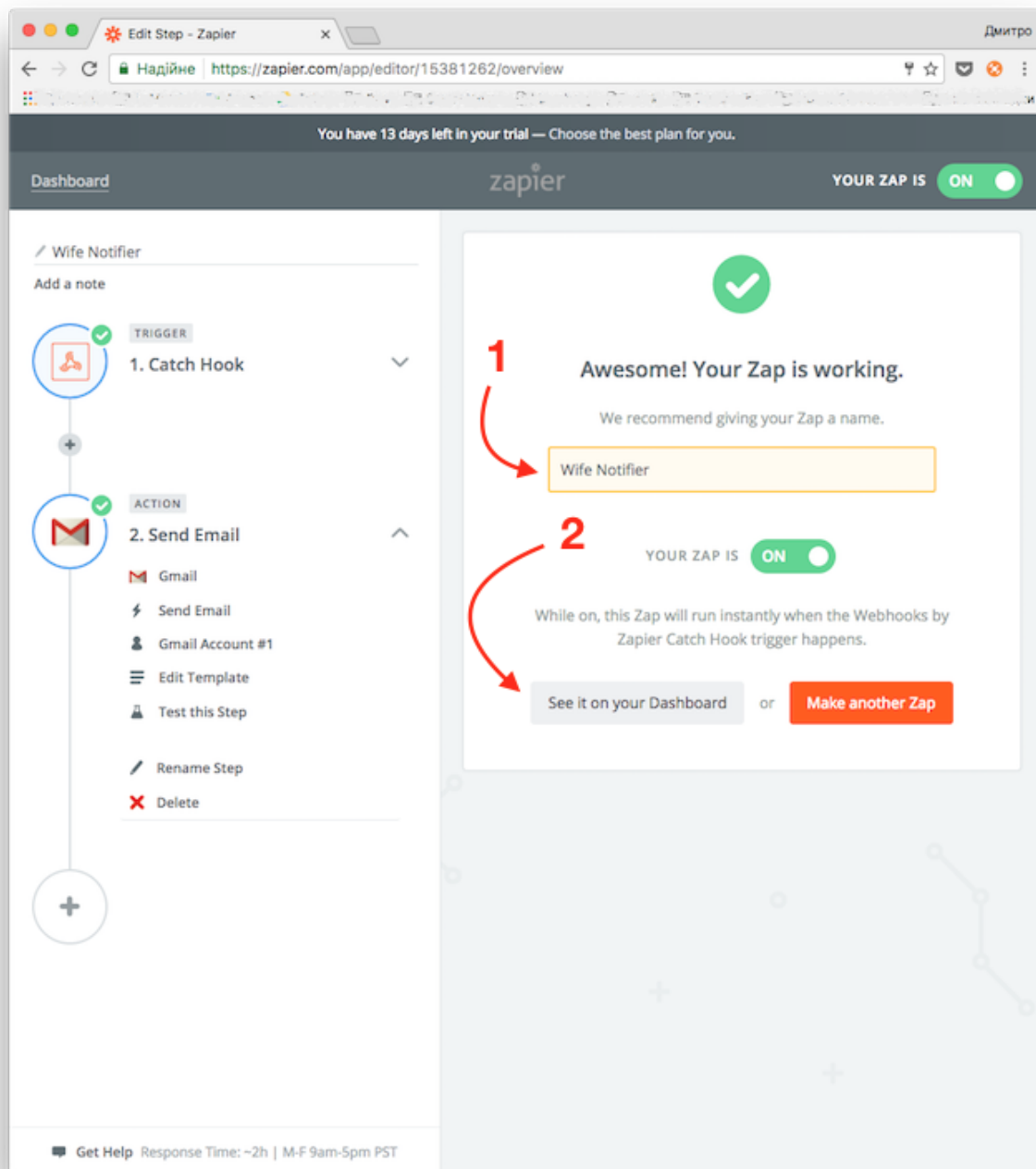
That's it! You have created your first zap - rule that sends e-mail to your wife when you click button on your LaMetric Time!
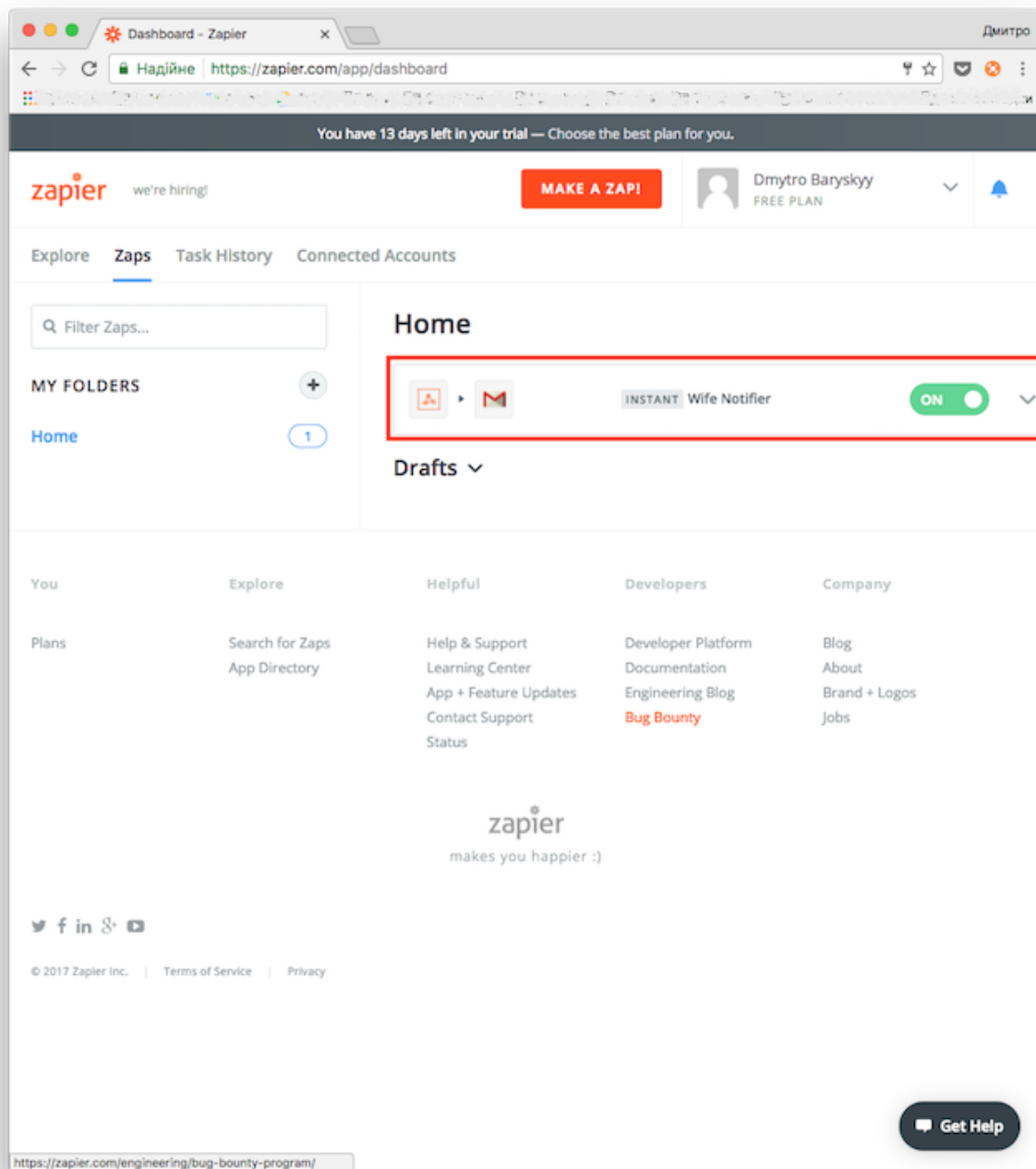
Let's press the action button. . . . .

Ding! It works!
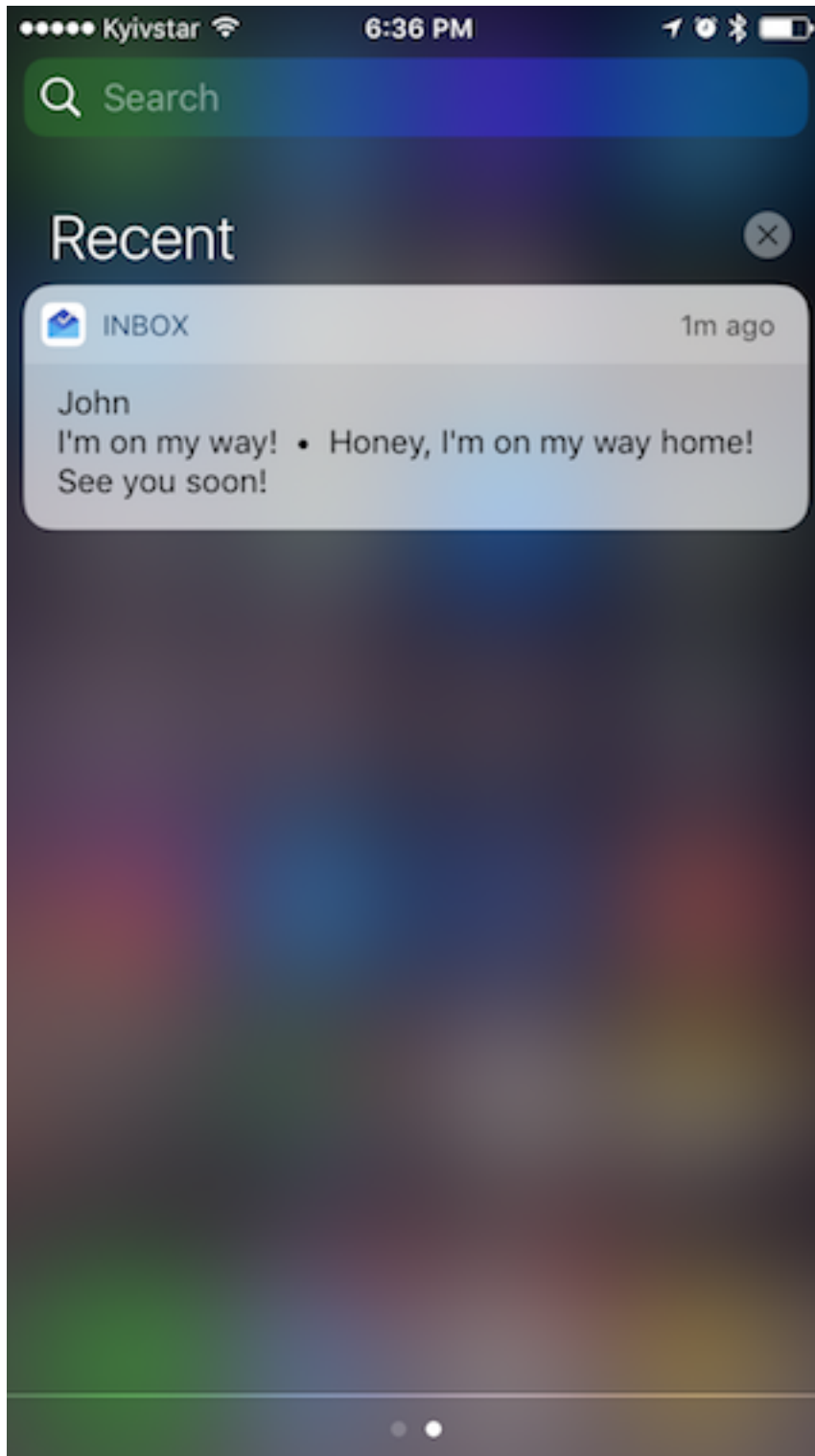
Hope you enjoyed the tutorial. Let us know if you find any issues, typos or other issues. Thanks!

Cloud API Documentation

This is where you can find API-level documentation for LaMetric Cloud API that allow to connect and manage LaMetric user profile and discover LaMetric Time devices.

API Contents

3.1 OAuth2 Authorization

3.1.1 Overview

The LaMetric API uses OAuth 2.0 protocol for simple but secure authentication and authorization. We support the most common OAuth 2.0 scenarios. Please keep in mind, that all requests to protected APIs must be made over SSL (https://).

The LaMetric API requires authentication for the requests made on behalf of a user. Each such request requires an **access_token**. Tokens are unique and should be stored securely.

Basic Steps

All applications should follow a basic pattern when accessing a LaMetric API using OAuth 2.0. At high level you should follow these four steps:

1. Obtain OAuth 2.0 credentials from the LaMetric Developer.

Visit LaMetric Developer to obtain OAuth 2.0 credentials (client id and client secret). You must create new Notification Source by pressing on "Create new app" button.

2. Obtain an access token from the LaMetric Cloud Server.

Your application must obtain an access token that grants access to specific API. Single access token can grant different degree of access to different API. This is controlled via *scope* parameter your application must set during request of access-token.

3. Send the access token to an API.

When access-token is obtained your application must send it to the LaMetric API in an HTTP authorization header.

4. Refresh the access token, if necessary.

Access tokens have limited lifetime. If your application must have access to the API beyond that lifetime it should request and store refresh token. Refresh token allows application to request new access tokens.

3.1.2 Server Side (Explicit) Flow
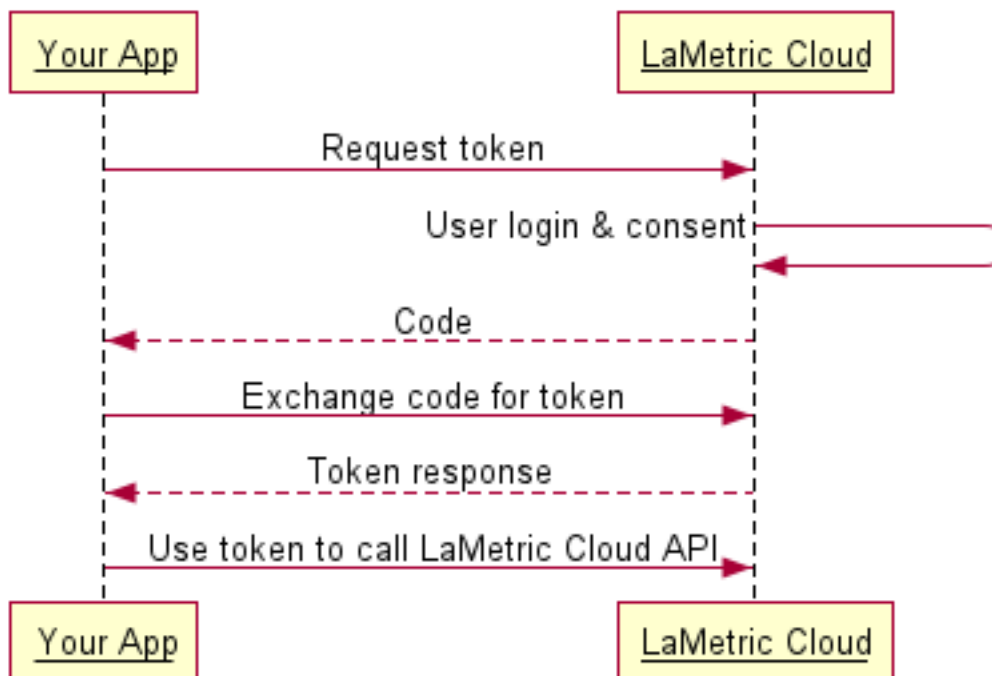
Step One: Direct your user to our authorization URL

```
https://developer.lametric.com/api/v2/oauth2/authorize/?client_id=CLIENT-ID&redirect_
→uri=REDIRECT-URI&response_type=code&scope=SCOPE&state=STATE
```

At this point, we present the user with a login screen and then a confirmation screen where to grant your app access to his/her LaMetric data.

Step Two: Receive the redirect from LaMetric

Once a user authorizes your application, we issue a redirect to your redirect_uri with a code parameter to use in step three.

```
http://your-redirect-uri?code=CODE&state=STATE
```

Note that the host and path components of your redirect URI must match exactly (including trailing slashes) your registered redirect_uri. You may also include additional query parameters in the supplied redirect_uri, if you need to vary your behavior dynamically.

If your request for approval is denied by the user, then we will redirect the user to your **redirect_uri** with the error message in parameters.

Step Three: Request the access_token

On this step you need to exchange the code you have received in previous step for the access token. To do that you just have to POST the code, along with app identification parameters to our token endpoint. Parameters are:

- **client_id**: your client id
- **client_secret**: your client secret
- **grant_type**: authorization_code
- **redirect_uri**: the redirect_uri you used in authorization request.
- **code**: the exact code you received during the authorization.

Sample request:

```
curl -F 'client_id=CLIENT_ID' \
     -F 'client_secret=CLIENT_SECRET' \
     -F 'grant_type=authorization_code' \
     -F 'redirect_uri=AUTHORIZATION_REDIRECT_URI' \
     -F 'code=CODE' \
     https://developer.lametric.com/api/v2/oauth2/token
```

If successfull this request will return OAuth 2.0 that you can use to make authenticated API requests. You will also get refresh_token, expiration interval and scopes. Example:

```
{
    "access_token":"0334613c79f116511fb81bc70f6dbaf20d002143",
    "expires_in":3600,
    "token_type":"Bearer",
    "scope":"basic devices_read",
    "refresh_token":"d7ab5645e7dcc9f227ee69f40962a19e15c80b04"
}
```

3.1.3 Client Side (Implicit) Flow
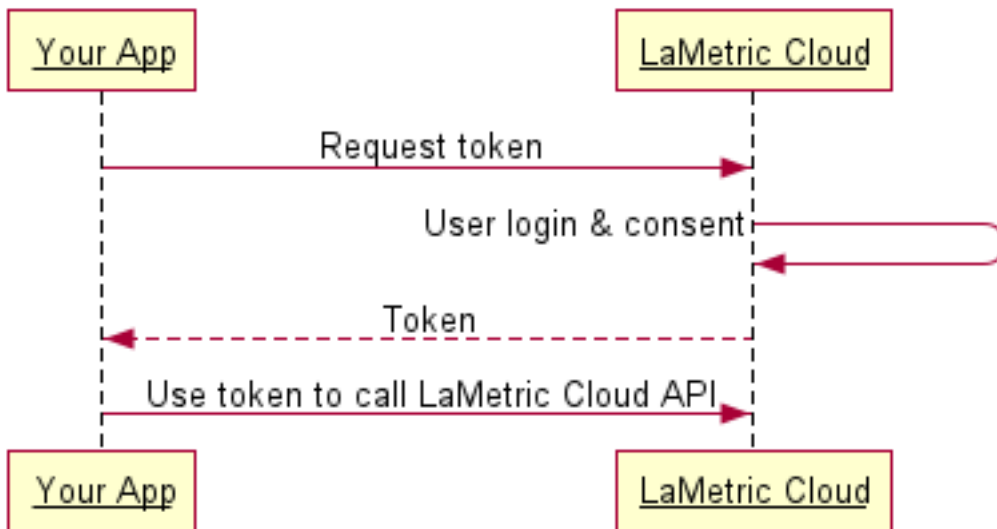
Step One: Direct your user to our authorization URL

```
https://developer.lametric.com/api/v2/oauth2/authorize/?client_id=CLIENT-ID&redirect_
↪uri=REDIRECT-URI&response_type=token&scope=SCOPE&state=STATE
```

User will be presented with login prompt and then a confirmation screen where they grant permissions to access their LaMetric Account data.

Step Two: Receive the access_token via the URL fragment

As soon as the user is authenticated and authorized your aplication, LaMetric Cloud will redirect them to your redirect_url with the access_token, token_type, scope and state in the URL fragment. It will look like this:

```
http://redirect-uri/#access_token=ACCESS-TOKEN&expires_in=3600&token_type=Bearer&
↪scope=SCOPE&state=STATE
```

3.1.4 Refreshing Access Token

You should refresh access token as soon as it is expired. This is the HTTP response you get when access token is not valid anymore:

```
HTTP/1.1 401 Unauthorized
Date: Fri, 17 Jun 2016 14:30:00 GMT
Server: Apache
X-Powered-By: PHP/5.4.45
Www-Authenticate: native
X-Powered-By: PleskLin
Connection: close
```

(continues on next page)

```
Transfer-Encoding: chunked
Content-Type: application/json; charset=UTF-8

{"errors":[{"message":"Unauthorized"}]}
```

To refresh the token do POST request to the token API:

```
curl -F 'client_id=CLIENT-ID' \
    -F 'client_secret=CLIENT-SECRET' \
    -F 'grant_type=refresh_token' \
    -F 'refresh_token=REFRESH-TOKEN' \
    https://developer.lametric.com/api/v2/oauth2/token
```

An example response JSON may look like this:

```
{
    "access_token":"65764004f094639190b93d7e75e2c4dfa343f3c3",
    "expires_in":3600,
    "token_type":"Bearer",
    "scope":"basic devices_read",
    "refresh_token":"b463576cc489e4a3f4b913a2505726f97635f5e7"
}
```

3.1.5 Scopes

Scope	Description
basic	Allows to read user's profile (name, e-mail, etc.)
devices_read	Allows to get information about devices that are connected to the user's account.
devices_write	Allows to update user's devices (rename for example)

It is possible to combine scopes like this:

```
scope=basic+devices_read+devices_write
```

3.2 Users

3.2.1 Endpoints

Method	Path	Description
GET	api/v2/users/me	Returns information about logged in user
GET	api/v2/users/me/devices	Returns list of user's devices
GET	api/v2/users/me/devices/:id	Returns information about specific device
PUT	api/v2/users/me/devices/:id	Updates device info

3.2.2 Get User

URL	/api/v2/users/me
Method	GET
Scope	basic

Description

Gets information about the logged in user.

Response

Returns User object.

Property	*Type*	*Description*
email	String	User's login and e-mail address
name	String	User's name
apps_count	Integer	Total number of apps created by the user
private_device_count	Integer	Number of devices connected to the user's account
private_apps_count	Integer	Number of private apps created by the user

Response Example

200 OK

```
{
    "id":1,
    "email":"user@mail.com",
    "name":"John Smith",
    "apps_count":1,
    "private_device_count":5,
    "private_apps_count":3
}
```

3.2.3 Get Devices

URL	/api/v2/users/me/devices
Method	GET
Scope	devices_read

Description

Gets the list of all devices connected to the user's account.

Response

Returns array of Device objects.

Property	Type	Description
id	Integer	Device id
name	String	Device name
state	String	Device state. Valid values are "new", "configured" or "banned".
serial_number	String	Device serial number
api_key	String	Key that is used as access token to access device's API in local network
ipv4_internal	String	IP address of the device in local network
mac	String	Mac address of the device
wifi_ssid	String	Name of the wi-fi access point the device is connected to

Response Example

200 OK

```
[
  {
    "id": 18,
    "name": "My LaMetric",
    "state": "configured",
    "serial_number": "SA140100002200W00BS9",
    "api_key": "8adaa0c98278dbb1ecb218d1c3e11f9312317ba474ab3361f80c0bd4f13a6749",
    "ipv4_internal": "192.168.0.128",
    "mac": "58:63:56:10:D6:30",
    "wifi_ssid": "homewifi",
    "created_at": "2015-03-06T15:15:55+02:00",
    "updated_at": "2016-06-14T18:27:13+03:00"
  }
]
```

3.2.4 Get Device By Id

URL	/api/v2/users/me/devices/:id
Method	GET
Scope	devices_read

Description

Gets device by id.

Response

Returns Device object.

Property	Type	Description
id	Integer	Device id
name	String	Device name
state	String	Device state. Valid values are "new", "configured" or "banned".
serial_number	String	Device serial number
api_key	String	Key that is used as access token to access device's API in local network
ipv4_internal	String	IP address of the device in local network
mac	String	Mac address of the device
wifi_ssid	String	Name of the wi-fi access point the device is connected to

Response Example

200 OK

```
{
  "id": 18,
  "name": "My LaMetric",
  "state": "configured",
  "serial_number": "SA140100002200W00BS9",
  "api_key": "8adaa0c98278dbb1ecb218d1c3e11f9312317ba474ab3361f80c0bd4f13a6749",
  "ipv4_internal": "192.168.0.128",
  "mac": "58:63:56:10:D6:30",
  "wifi_ssid": "homewifi",
  "created_at": "2015-03-06T15:15:55+02:00",
  "updated_at": "2016-06-14T18:27:13+03:00"
}
```

3.2.5 Update Device

URL	/api/v2/users/me/devices/:id
Method	PUT
Scope	devices_write

Description

Updates specific device by id.

Body

```
{
    "name": "Device @ Work"
}
```

Response

Returns success object with device id and name.

Response Example

200 OK

```
{
  "success": {
    "id": 18,
    "name": "Device @ Work"
  }
}
```

3.3 Icons

URL	/api/v2/icons
Method	GET

3.3.1 Description

Returns the list of the icons available at http://developer.lametric.com/icons. Supports pagination and sorting.

3.3.2 Parameters

Parameter	Description
page	Page number [0..n-1]
page_size	Number of items per page [1..total_icon_count]
fields	Comma separated field list to filter out fields that are not needed.
order	[popular, newest, title] • "popular" instructs the server to return icons sorted by popularity • "newest" instructs the server to return icons sorted by the creation time • "title" instructs the server to return icons sorted by title.

3.3.3 Response

Returns list of icon object inside "data" object and pagination information in "meta" object.
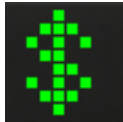
Meta Object

Property	Type	Description
total_icon_count	Integer	Total number of icons in database
page	Integer	Current page number
page_size	Integer	Page size (number of icons). Last page may have less
page_count	Integer	Total pages count

Data Object

Data object contains array of Icon objects.

Icon Object

Property	Type	Description
id	Integer	Icon ID
title	String	Icon title
code	String	Code that can be used when sending icon to LaMetric.
type	Enum	Type of the image – picture or movie. • picture – static png image • movie – static or animated gif
url	String	Direct URL to download icon that has size of 8x8 pixels.
thumb	Object	Object that contains URLs to preview images. original (45px x 45px) small (40px x 40px) large (99px x 99px) xlarge (150px x 149px)

3.3.4 Examples

Request

```
GET https://developer.lametric.com/api/v2/icons
```

Response

```
200 OK

{
  "meta": {
    "total_icon_count": 2676,
    "page": 0,
    "page_size": 2676,
    "page_count": 1
  },
  "data": [
    {
      "id": 1,
      "title": "Button Error",
      "code": "a1",
      "type": "movie",
      "category": null,
      "url": "https://developer.lametric.com/content/apps/icon_thumbs/1.gif",
      "thumb": {
        "original": "https://developer.lametric.com/content/apps/icon_thumbs/1_icon_
↪thumb.gif",
        "small": "https://developer.lametric.com/content/apps/icon_thumbs/1_icon_
↪thumb_sm.png",
        "large": "https://developer.lametric.com/content/apps/icon_thumbs/1_icon_
↪thumb_lg.png",
        "xlarge": "https://developer.lametric.com/content/apps/icon_thumbs/1_icon_
↪thumb_big.png"
      }
    },
    {
      "id": 2,
      "title": "Button Success",
      "code": "i2",
      "type": "picture",
      "category": null,
      "url": "https://developer.lametric.com/content/apps/icon_thumbs/2.png",
      "thumb": {
        "original": "https://developer.lametric.com/content/apps/icon_thumbs/2_icon_
↪thumb.png",
        "small": "https://developer.lametric.com/content/apps/icon_thumbs/2_icon_
↪thumb_sm.png",
        "large": "https://developer.lametric.com/content/apps/icon_thumbs/2_icon_
↪thumb_lg.png",
        "xlarge": "https://developer.lametric.com/content/apps/icon_thumbs/2_icon_
↪thumb_big.png"
      }
    },

    // Result is truncated
  ]
}
```

Request

```
GET https://developer.lametric.com/api/v2/icons?page=1&page_size=2&fields=id,title,
→url&order=newest
```

Response

```
200 OK

{
  "meta": {
    "total_icon_count": 2676,
    "page": 1,
    "page_size": 2,
    "page_count": 1338
  },
  "data": [
    {
      "id": 2959,
      "title": "JulienBreux - CPU",
      "url": "https://developer.lametric.com/content/apps/icon_thumbs/2959.png"
    },
    {
      "id": 2957,
      "title": "JulienBreux - Memory",
      "url": "https://developer.lametric.com/content/apps/icon_thumbs/2957.png"
    }
  ]
}
```

Device API Documentation

This is where you can find API-level documentation for sending notifications to LaMetric Time devices in local network.

API Contents

4.1 Device Discovery

There are multiple ways to discover LaMetric devices in local network. Discovering works if 3rd party service, device or app is located in the same subnet.

4.1.1 1. Cloud API

To get the local IP address of the LaMetric Time device do these steps:

1. Login to LaMetric Cloud

2. Get list of user's devices

3. Get IP address and api_key of the device via API request.

For more details check Authorization section.

4.1.2 2. UPnP

1.1. Send SSDP discovery request (more info can be found here) and listen for incomming broadcast messages:

```
HTTP/1.1 200 OK
CACHE-CONTROL: max-age=1800
EXT:
LOCATION: http://192.168.88.153:60669/b2d83c5d-6012-4857-916e-6238ca6cab4e/device_
↪description.xml
```

(continues on next page)

```
SERVER: Linux/3.4.103 UPnP/1.1 HUPnP/1.0
ST: upnp:rootdevice
USN: uuid:b2d83c5d-6012-4857-916e-6238ca6cab4e::upnp:rootdevice
BOOTID.UPNP.ORG: 0
CONFIGID.UPNP.ORG: 0
```

1.2. Take the URL from `LOCATION` parameter and get device_description.xml:

```
curl -i http://192.168.88.153:60669/b2d83c5d-6012-4857-916e-6238ca6cab4e/device_
↪description.xml
```

```
HTTP/1.1 200 OK
DATE: Tue, 28 Jun 2016 19:59:36
Connection: close
HOST: 192.168.88.153:60669
content-length: 771

<?xml version="1.0" encoding="UTF-8"?>
  <root xmlns="urn:schemas-upnp-org:device-1-0">
    <specVersion>
      <major>1</major>
      <minor>0</minor>
    </specVersion>
    <URLBase>https://192.168.88.153:443</URLBase>
    <device>
      <deviceType>urn:schemas-upnp-org:device:LaMetric:1</deviceType>
      <friendlyName>LaMetric (LM1419)</friendlyName>
      <manufacturer>Smart Atoms Inc.</manufacturer>
      <manufacturerURL>http://www.smartatoms.com</manufacturerURL>
      <modelDescription>LaMetric - internet connected ticker witn UPnP SSDP support</
↪modelDescription>
      <modelName>LaMetric Battery Edition</modelName>
      <modelNumber>SA01</modelNumber>
      <modelURL>http://www.lametric.com</modelURL>
      <serialNumber>SA150800000100W00BP9</serialNumber>
      <serverId>1</serverId>
      <UDN>uuid:b2d83c5d-6012-4857-916e-6238ca6cab4e</UDN>
    </device>
  </root>
```

LaMetric Time public API is located on ports 8080 and 4343. You should have api_key to access it (see Authorization section for more details).

1.3. If you have api_key on this stage you can check if LaMetric Time API works.

```
curl -i http://192.168.88.153:8080/api/v2 --user dev:<api_key>

HTTP/1.1 200 OK
CONTENT-TYPE: application/json;charset=UTF8
Transfer-Encoding: chunked
Date: Tue, 28 Jun 2016 17:36:54 GMT
Server: lighttpd/1.4.35

{
  "api_version" : "2.0.0",
  "endpoints" : {
```

```
    "audio_url" : "http://192.168.88.153:8080/api/v2/device/audio",
    "bluetooth_url" : "http://192.168.88.153:8080/api/v2/device/bluetooth",
    "concrete_notification_url" : "http://192.168.88.153:8080/api/v2/device/
↪notifications{/:id}",
    "current_notification_url" : "http://192.168.88.153:8080/api/v2/device/
↪notifications/current",
    "device_url" : "http://192.168.88.153:8080/api/v2/device",
    "display_url" : "http://192.168.88.153:8080/api/v2/device/display",
    "notifications_url" : "http://192.168.88.153:8080/api/v2/device/notifications",
    "widget_update_url" : "http://192.168.88.153:8080/api/v2/widget/update{/:id}",
    "wifi_url" : "http://192.168.88.153:8080/api/v2/device/wifi"
  }
}
```

We recommend to use secure way of accessing the API via https using port 4343:

```
curl -i -H "Authorization: Basic␣
↪dXNlcjpiZTBmNTNhYTQ1NzdjMzUxMDE3OGY2Mzc3Yjk3NTEwY2U0ZTA2ZGQ3ZTBjYTlkMDRjNDMyMDRiY2RlZTllMjY2
↪"
https://192.168.88.153:4343/api/v2 --insecure

HTTP/1.1 200 OK
CONTENT-TYPE: application/json;charset=UTF8
Transfer-Encoding: chunked
Date: Tue, 28 Jun 2016 17:51:25 GMT
Server: lighttpd/1.4.35

{
  "api_version" : "2.0.0",
  "endpoints" : {
    "audio_url" : "https://192.168.88.153:4343/api/v2/device/audio",
    "bluetooth_url" : "https://192.168.88.153:4343/api/v2/device/bluetooth",
    "concrete_notification_url" : "https://192.168.88.153:4343/api/v2/device/
↪notifications{/:id}",
    "current_notification_url" : "https://192.168.88.153:4343/api/v2/device/
↪notifications/current",
    "device_url" : "https://192.168.88.153:4343/api/v2/device",
    "display_url" : "https://192.168.88.153:4343/api/v2/device/display",
    "notifications_url" : "https://192.168.88.153:4343/api/v2/device/notifications",
    "widget_update_url" : "https://192.168.88.153:4343/api/v2/widget/update{/:id}",
    "wifi_url" : "https://192.168.88.153:4343/api/v2/device/wifi"
  }
}
```

`--insecure` option must be added because of the random IP address LaMetric may have, and it is not possible to verify host stored inside the certificate.

4.2 Authorization

4.2.1 Overview

LaMetric Time uses basic authorization for simple access to the device via API. Authorization header consists of word "dev" as user name and API key as a password.

```
Authorization: Basic Base64(dev:api_key)
```

Steps To Get an API Key

There are 3 steps you should do to get API key.

Step 1. Authenticate on the Cloud

Please refer to the Cloud API Documentation / OAuth2 Authorization on this subject.

Step 2. Get API key for the device from the Cloud

Once authenticated please use GET https://developer.lametric.com/api/v2/users/me/devices API to get list of user's devices.

You should get something like this:

```
[
  {
    "id": 18,
    "name": "My LaMetric",
    "state": "configured",
    "serial_number": "SA150600000100W00BS9",
    "api_key": "8adaa0c98278dbb1ecb218d1c3e11f9312317ba474ab3361f80c0bd4f13a6749",
    "ipv4_internal": "192.168.0.128",
    "mac": "58:63:56:10:D6:30",
    "wifi_ssid": "homewifi",
    "created_at": "2015-03-06T15:15:55+02:00",
    "updated_at": "2016-06-14T18:27:13+03:00"
  }
]
```

You can find device API key in `api_key` property.

Step 3. Store API key and use it in every API request

Now store the API key in secure place and use it to authenticate each API call to device.

3.1 Concatenate "dev:" and api_key:

```
dev:8adaa0c98278dbb1ecb218d1c3e11f9312317ba474ab3361f80c0bd4f13a6749
```

3.2 Encode using Base64:

```
base64(dev:8adaa0c98278dbb1ecb218d1c3e11f9312317ba474ab3361f80c0bd4f13a6749) =
→ZGV2OjhhZGFhMGM5ODI3OGRiYjFlY2IyMThkMWMzZTExZjkzMTIzMTdiYTQ3NGFiMzM2MWY4MGMwYmQ0ZjEzYTY3NDk=
```

3.3 Use in HTTP header:

```
Authorization: Basic
→ZGV2OjhhZGFhMGM5ODI3OGRiYjFlY2IyMThkMWMzZTExZjkzMTIzMTdiYTQ3NGFiMzM2MWY4MGMwYmQ0ZjEzYTY3NDk=
```

4.3 Endpoints

Method	Path	Description
Introduced in API 2.0.0		
GET	/api/v2	Returns API version and endpoints available
GET	/api/v2/device	Returns full device state
POST	/api/v2/device/notifications	Sends new notification to device
GET	/api/v2/device/notifications	Returns the list of notifications in queue
GET	/api/v2/device/notifications/current	Returns current notification (notification that is visible)
GET	/api/v2/device/notifications/:id	Returns specific notification
DELETE	/api/v2/device/notifications/:id	Removes notification from queue or dismisses if it is visible
GET	/api/v2/device/display	Returns information about display, like brightness
PUT	/api/v2/device/display	Allows to modify display state (change brightness)
GET	/api/v2/device/audio	Returns current volume
PUT	/api/v2/device/audio	Allows to change volume
GET	/api/v2/device/bluetooth	Returns bluetooth state
PUT	/api/v2/device/bluetooth	Allows to activate/deactivate bluetooth and change name
GET	/api/v2/device/wifi	Returns wi-fi state
Added in API 2.1.0		
GET	/api/v2/device/apps	Returns the list of installed apps
GET	/api/v2/device/apps/:package	Returns info about installed app identified by package name
PUT	/api/v2/device/apps/next	Switches to next app
PUT	/api/v2/device/apps/prev	Switches to previous app
POST	/api/v2/device/apps/:package/widgets/:id/actions	Sends application specific action to widget
PUT	/api/v2/device/apps/:package/widgets/:id/activate	Activates specific widget (app instance)

4.3.1 Get API Version

URL	/api/v2
Method	GET
Authentication	basic
API Version	2.0.0

Description

Gets information about the current API version. Also returns object map with all API endpoints available on the device.

Response

Property	Type	Description
api_version	String	Current version of the api in format <major>.<minor>.<patch>
endpoints	Object	Map of available API endpoints

Response Example

```
HTTP/1.1 200 OK
CONTENT-TYPE: application/json;charset=UTF8
Transfer-Encoding: chunked
Date: Thu, 23 Jun 2016 15:56:42 GMT
Server: lighttpd/1.4.35

{
  "api_version": "2.1.0",
  "endpoints": {
    "apps_action_url": "http://192.168.3.13:8080/api/v2/device/apps/{:id}/widgets/
↪{:widget_id}/actions",
    "apps_get_url": "http://192.168.3.13:8080/api/v2/device/apps/{:id}",
    "apps_list_url": "http://192.168.3.13:8080/api/v2/device/apps",
    "apps_switch_next_url": "http://192.168.3.13:8080/api/v2/device/apps/next",
    "apps_switch_prev_url": "http://192.168.3.13:8080/api/v2/device/apps/prev",
    "apps_switch_url": "http://192.168.3.13:8080/api/v2/device/apps/{:id}/widgets/
↪{:widget_id}/activate",
    "audio_url": "http://192.168.3.13:8080/api/v2/device/audio",
    "bluetooth_url": "http://192.168.3.13:8080/api/v2/device/bluetooth",
    "concrete_notification_url": "http://192.168.3.13:8080/api/v2/device/
↪notifications/{:id}",
    "current_notification_url": "http://192.168.3.13:8080/api/v2/device/notifications/
↪current",
    "device_url": "http://192.168.3.13:8080/api/v2/device",
    "display_url": "http://192.168.3.13:8080/api/v2/device/display",
    "notifications_url": "http://192.168.3.13:8080/api/v2/device/notifications",
    "wifi_url": "http://192.168.3.13:8080/api/v2/device/wifi"
  }
}
```

4.4 Device

4.4.1 Get Device State

URL	/api/v2/device
Method	GET
Authentication	basic

Description

Returns information about the device like name, serial number, version of the firmware, model etc. Response also contains state of audio, display, bluetooth and wi-fi.

Parameters

Property	Description
fields	Comma separated list of field you want to receive in response. Possible values are: • id • name • serial_number • os_version • mode • model • audio • display • bluetooth • wifi

Response

Property	Type	Description
id	String	Id of the device on the cloud
name	String	User specified name of the device
serial_number	String	Device serial number
os_version	String	Software version in format <major>.<minor>.<patch>
update_available	Object	Since API 2.3.0. Optional. If present, means that firmware upgrade is available for the device.
model	String	Model number. Can be one of "LM 37X8", "sa8", "sa5" • *"LM 37X8"* stands for LaMetric TIME devices, produced before 2022. • *"sa8"* stands for LaMetric TIME devices, produced in 2022 and later • *"sa5"* snands for LaMetric SKY devices.
mode	String	Current device mode. Can be one of "auto", "manual", "schedule" or "kiosk" • *auto* - auto scroll mode, when device switch between apps automatically • *manual* - click to scroll mode, when user can manually switch between apps • *schedule* - mode when apps get switched according to a schedule • *kiosk* - kiosk mode when single app is locked on the device.
audio	Object	Audio state. Allows to get current volume
bluetooth	Object	Bluetooth state
display	Object	Display state
wifi	Object	Wi-Fi state

Examples

Request:

```
GET http://192.168.0.239:8080/api/v2/device

Authorization: Basic <Base64("dev:Device API Key")>
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF8

{
    "id" : "1",
    "name" : "LM0001",
    "serial_number" : "SA150600000100W00BS9",
    "mode" : "manual",
    "model" : "LM 37X8",
    "os_version" : "2.3.0",
    "update_available": {
        "version": "2.3.1"
    },
    "audio" : {
        "volume" : 100
    },
    "bluetooth" : {
        "available" : true,
        "name" : "LM0001",
        "active" : false,
        "discoverable" : false,
        "pairable" : true,
        "address" : "58:63:56:10:FD:2F"
        },
    "display" : {
        "brightness" : 67,
        "brightness_mode" : "auto",
        "width" : 37,
        "height" : 8,
        "type" : "mixed"
    },
    "wifi" : {
        "active" : true,
        "address" : "58:63:56:10:D6:1F",
        "available" : true,
        "encryption" : "WPA",
        "essid" : "home-wifi",
        "ip" : "192.168.0.233",
        "mode" : "dhcp",
        "netmask" : "255.255.255.0",
        "strength" : 100
    }
}
```

Request:

```
GET http://192.168.0.233:8080/api/v2/device?fields=name,wifi
```

Response:

```
HTTP/1.1 200 OK
CONTENT-TYPE: application/json;charset=UTF8
Transfer-Encoding: chunked
Date: Thu, 23 Jun 2016 17:06:14 GMT
```

```
Server: lighttpd/1.4.35

{
    "name" : "LM0001",
    "wifi" : {
        "active" : true,
        "address" : "58:63:56:10:D6:1F",
        "available" : true,
        "encryption" : "WPA",
        "essid" : "home-wifi",
        "ip" : "192.168.0.233",
        "mode" : "dhcp",
        "netmask" : "255.255.255.0",
        "strength" : 100
    }
}
```

4.4.2 Change Device Mode

URL	/api/v2/device
Method	PUT
Authentication	basic
API Version	2.3.1

Description

Allows changing device mode to one of four supported ones: manual, auto, schedule, kiosk.

Body

```
{
    "mode": "[manual|auto|schedule|kiosk]"
}
```

Response

```
{
    "success" : {
        "data" : { "mode": "[manual|auto|schedule|kiosk]" },
        "path" : "/api/v2/device"
    }
}
```

Example. Enable automatic app switching mode

Request

REST

```
PUT https://<device ip address>:4343/api/v2/device

Authorization: Basic <Base64("dev:Device API Key")>
Accept: application/json
Content-Type: application/json

{
    "mode": "auto"
}
```

cURL

```
$ curl --location --request PUT 'https://192.168.170.14:4343/api/v2/device' \
  -u "dev:<device API key>" -k \
  --header 'Accept: application/json' \
  --header 'Content-Type: application/json' \
  --data-raw '{ "mode": "auto" }'
```

Response

```
HTTP/1.1 200 OK

{
    "success": {
        "data": { "mode": "auto" },
        "path": "/api/v2/device"
    }
}
```

4.5 Apps

Apps API allows you to control some aspects of your LaMetric Time device that is related to apps that are running on it. For example, you will be able to switch between apps as well as configure alarm, control timers, radio and more.

4.5.1 Endpoints

Method	Path	Description
GET	*/api/v2/device/apps*	Returns the list of installed apps
PUT	*/api/v2/device/apps/next*	Switches to next app
PUT	*/api/v2/device/apps/prev*	Switches to previous app
GET	*/api/v2/device/apps/:package*	Returns info about installed app identified by package name
POST	*/api/v2/device/apps/:package/widgets/:id/actions*	Sends application specific action to a widget
PUT	*/api/v2/device/apps/:package/widgets/:id/activate*	Activates specific widget (app instance)

4.5.2 Get List of Apps

URL	/api/v2/device/apps
Method	GET
Authorization	basic
API Version	2.1.0

Description

Returns apps currently installed on LaMetric Time. Each app is identified by package name. Device can run multiple instances of an app. These instances are called widgets. There are at least one widget (instance) running for each app.

Response

```
{
    "<package1>" : {
      "package": "<package1>",
      "vendor" : "<vendor>",
      "version" : "<version>",
      "version_code" : "<version_code>",
      "widgets" : {
        "<widget_uuid>" : {
          "index":<position>,
          "package": "<package1>",
          "visible": <bool>     // since API 2.3.0
        }
      },
      "actions" : {
        "<action1>" : {
          "<action_name1>" : {
              "<param1>" : {
                  "data_type" : "<bool|int|string>",
                  "name" : "<param name>",
                  "format": "<regexp>",
                  "required": <bool>
              }
          }
        }
      }
    },
    ...
    "<package_n>" : <app_object_n>
}
```

Please refer to */api/v2/device/apps/:package* for more details on Application object.

Example

Request

REST:

```
GET https://<device ip address>:4343/api/v2/device/apps

Accept: application/json
```

cURL:

```
$ curl -X GET -u "dev" -k -H "Accept: application/json" \
  https://<device ip address>:4343/api/v2/device/apps
$ Enter host password for user 'dev': <device api key>
```

Response

```
{
    "com.lametric.clock": {
        "package": "com.lametric.clock",
        "vendor": "LaMetric",
        "version": "1.0.19",
        "version_code": "31",
        "actions": {
            "clock.alarm": {
                "enabled": {
                    "data_type": "bool",
                    "name": "enabled",
                    "required": false
                },
                "time": {
                    "data_type": "string",
                    "format": "[0-9]{2}:[0-9]{2}(?::[0-9]{2})?",
                    "name": "time",
                    "required": false
                },
                "wake_with_radio": {
                    "data_type": "bool",
                    "name": "wake_with_radio",
                    "required": false
                }
            }
        },
        "widgets": {
            "08b8eac21074f8f7e5a29f2855ba8060": {
                "index": 0,
                "package": "com.lametric.clock",
                "visible": true
            }
        }
    },
    "com.lametric.countdown": {
        ...
    },
    "com.lametric.radio": {
        ...
    },
    "com.lametric.stopwatch": {
        ...
    },
    "com.lametric.weather": {
        ...
```

(continues on next page)

```
        }
}
```

4.5.3 Switch to Next App

URL	/api/v2/device/apps/next
Method	PUT
Authorization	basic
API Version	2.1.0

Description

Allows to switch to the next app on LaMetric Time. App order is controlled by the user via LaMetric Time app.

Body

Does not require body.

Response

```
{
    "success": {
        "data" {},
        "path": "<endpoint>"
    }
}
```

or

```
{
    "errors" : [
      {
        "message" : "<Error message>"
      }
    ]
}
```

Example

Request

REST:

```
PUT https://<device ip address>:4343/api/v2/device/apps/next
```

cURL:

```
$ curl -X PUT -u "dev" -H "Accept: application/json"-k \
  https://<device ip address>:4343/api/v2/device/apps/next
$ Enter host password for user 'dev': <device api key>
```

Response:

```
{
    "success": {
        "data": {},
        "path": "/api/v2/device/apps/next"
    }
}
```

4.5.4 Switch to Previous App

URL	/api/v2/device/apps/prev
Method	PUT
Authorization	basic
API Version	2.1.0

Description

Allows to switch to the previous app on LaMetric Time. App order is controlled by the user via LaMetric Time app.

Body

Does not require body.

Response

```
{
    "success": {
        "data" {},
        "path": "<endpoint>"
    }
}
```

or

```
{
    "errors" : [
      {
        "message" : "<Error message>"
      }
    ]
}
```

Example

Request

REST:

```
PUT https://<device ip address>:4343/api/v2/device/apps/prev
```

cURL:

```
$ curl -X PUT -u "dev" -H "Accept: application/json" -k \
  https://<device ip address>:4343/api/v2/device/apps/prev
$ Enter host password for user 'dev': <device api key>
```

Response

```
{
    "success": {
        "data": {},
        "path": "/api/v2/device/apps/prev"
    }
}
```

4.5.5 Get Specific App Details

URL	/api/v2/device/apps/:package
Method	GET
Authorization	basic
API Version	2.1.0

Description

Returns information about currently installed app identified by the package.

Response

```
{
    "package": "<string>",
    "vendor": "<string>",
    "version": "<x.x.x>",
    "version_code": "<version code>",
    "actions": {
        "<action_id>": {
            "<parameter_id>": {
                "data_type": "[bool, int, string]",
                "name": "<string>",
                "required": <boolean>,
                "format": "<regexp>"
            }
        }
```

(continues on next page)

```
    },
    "widgets": {
        "<uuid>": {
            "index": <order no>,
            "package": "<string>",
            "visible": <bool>   // since API 2.3.0
        }
    }
}
```

Application Object		
Field	**Type**	**Description**
package	String	Unique identifier of LaMetric Time native app.
vendor	String	Name of the app creator
version	String	Version in format "<major>.<minor>.<patch>". For example 2.0.0 or 2.0.1
version_code	String	Version as number, like 1, 2, 3. Useful for easy comparison
actions	Map	Map of actions this app supports. For example, clock support action that allows to configure alarm.
widgets	Map	Map of Widgets. Widget is an instance of an app. For example, if you clone Clock app, you'll get two widgets representing each clock instance.

Parameter Object		
Field	**Type**	**Description**
data_type	String	One of [bool, int, string]
name	String	Name of the parameter
required	Boolean	true if parameter is required or false otherwise
format	String	Optional. Regext that defines the format of string parameter.

Widget Object		
Field	**Type**	**Description**
index	Integer	Position of the widget when switching between them with buttons or API. Can be -1.
package	String	Id of the LaMetric Time app this widget is an instance of
visible	Boolean	Since API 2.3.0. True if widget is currently displayed on the screen, false otherwise.

Example

Request

REST:

```
GET https://<device ip address>:4343/api/v2/device/apps/com.lametric.clock

Accept: application/json
```

cURL:

```
$ curl -X GET -u "dev" -H "Accept: application/json" -k \
  https://<device ip address>:4343/api/v2/device/apps/com.lametric.clock
$ Enter host password for user 'dev': <device api key>
```

Response:

```
{
    "package": "com.lametric.clock",
    "vendor": "LaMetric",
    "version": "1.0.19",
    "version_code": "31",
    "actions": {
        "clock.alarm": {
            "enabled": {
              "data_type": "bool",
              "name": "enabled",
              "required": false
            },
            "time": {
              "data_type": "string",
              "format": "[0-9]{2}:[0-9]{2}(?::[0-9]{2})?",
              "name": "time",
              "required": false
            },
            "wake_with_radio": {
              "data_type": "bool",
              "name": "wake_with_radio",
              "required": false
            }
        }
    },
    "widgets": {
        "08b8eac21074f8f7e5a29f2855ba8060": {
            "index": 0,
            "package": "com.lametric.clock",
            "visible": true
        }
    }
}
```

4.5.6 Interact With Running Widgets

URL	/api/v2/device/apps/:package/widgets/:id/actions
Method	POST
Authorization	basic
API Version	2.1.0

Description

Using this endpoint you can control LaMetric Time apps. Each app provides its own set of actions you can use. For example, you can start or stop radio playback, start, pause, reset timers, configure alarm clock etc. To execute an action just send an Action object in the body of the request to the endpoint like this:

```
{
  "id" : "<action_id>",
  "params": {},
  "activate": true|false
}
```

Parameter "activate" controls whether widget should become visible when the action is invoked.

Here are some actions of preinstalled apps:

App Name	Package	Action Id
Alarm Clock	`com.lametric.clock`	`clock.alarm` - configure alarm clock `clock.clockface` - sets or updates a clock face
Radio	`com.lametric.radio`	`radio.play` - start playback `radio.stop` - stop playback `radio.next` - next radio station `radio.prev` - previous radio station
Timer	`com.lametric.countdown`	`countdown.configure` - set time `countdown.start` - starts countdown `countdown.pause` - pauses countdown `countdown.reset` - resets timer
Stopwatch	`com.lametric.stopwatch`	`stopwatch.start` - starts stopwatch `stopwatch.pause` - pauses stopwatch `stopwatch.reset` - resets stopwatch
Weather	`com.lametric.weather`	`weather.forecast` - displays weather forecast

Some actions have parameters, for example `clock.alarm`, `clock.clockface` and `countdown.configure`.

Action "clock.alarm"

```
{
  "id":"clock.alarm",
  "params": {
    "enabled":true,
    "time":"10:00:00",
```

```
      "wake_with_radio":false
   }
}
```

Parameter	Format	Description
`enabled`	Boolean	Optional. Activates the alarm if set to `true`, deactivates otherwise.
`time`	String	Optional. Local time in format "HH:mm:ss".
`wake_with_radio`	Boolean	Optional. If true, radio will be activated when alarm goes off.

Action "clock.clockface"

```
{
  "id":"clock.clockface",
  "params": {
    "icon":"data:image/png;base64,
↪iVBORw0KGgoAAAANSUhEUgAAAAgAAAAICAYAAADED76LAAAAOklEQVQYlWNUVFBgwAeYcEncv//
↪gP04FMEmsCmCSiooKjHAFMEF0SRQTsEnCFcAE0SUZGBgYGAl5EwA+6RhuHb9bggAAAABJRU5ErkJggg==",
    "type" : "custom"
  },
  "activate": true
}
```

Parameter	Format	Description
`icon`	String	Optional. Icon data in format "data:image/png;base64,<base64 encoded png binary>" or "data:image/gif;base64,<base64 encoded gif binary>"
`type`	String	Optional. Specifies the clockface type. Possible values are "weather", "page_a_day", "custom" and "none".

Action "countdown.configure"

```
{
  "id":"countdown.configure",
  "params": {
    "duration":1800,
    "start_now":false
  }
}
```

Parameter	Format	Description
`duration`	Integer	Optional. Time in seconds.
`start_now`	Boolean	Optional. If set to `true` countdown will start immediately.

Body

```
{
  "id" : "<action_id>",
  "params": {
    "key": "value"
```

```
    }
}
```

Response

```
{
  "success": {
    "data": {},
    "path": "/api/v2/device/apps/<package>/widgets/<widget_id>/actions"
  }
}
```

Example

This request starts radio playback.

Request

REST:

```
POST https://<device ip addess>:4343/api/v2/device/apps/com.lametric.radio/widgets/
→589ed1b3fcdaa5180bf4848e55ba8061/actions

Content-Type: application/json

{ "id": "radio.play" }
```

cURL:

```
$ curl -X POST -u "dev" -H "Content-Type: application/json" -k \
  -d '{ "id":"radio.play"}' \
  https://<device ip addess>:4343/api/v2/device/apps/com.lametric.radio/widgets/
→589ed1b3fcdaa5180bf4848e55ba8061/actions
$ Enter host password for  user 'dev': <device api key>
```

Response

```
{
  "success": {
    "data": {},
    "path": "/api/v2/device/apps/com.lametric.radio/widgets/
→589ed1b3fcdaa5180bf4848e55ba8061/actions"
  }
}
```

4.5.7 Activate Specific Widget

URL	/api/v2/device/apps/:package/widgets/:id/activate
Method	PUT
Authorization	basic
API Version	2.1.0

Description

Allows to make any widget visible using widget id.

Body

Does not require body.

Example

Request

REST:

```
PUT https://<device ip address>:4343/api/v2/device/apps/com.lametric.clock/widgets/
↪08b8eac21074f8f7e5a29f2855ba8060/activate

Accept: application/json
```

cURL:

```
$ curl -X PUT -u "dev" -H "Accept: application/json" \
  https://<device ip address>:4343/api/v2/device/apps/com.lametric.clock/widgets/
↪08b8eac21074f8f7e5a29f2855ba8060/activate
$ Enter host password for  user 'dev': <device api key>
```

Response

```
{
  "success" : {
    "data" : {},
    "path" : "/api/v2/device/apps/com.lametric.clock/widgets/
↪08b8eac21074f8f7e5a29f2855ba8060/activate"
  }
}
```

4.6 Notifications

4.6.1 Endpoints

Method	Path	Description
POST	*/api/v2/device/notifications*	Sends new notification to device
GET	*/api/v2/device/notifications*	Returns the list of notifications in queue
DELETE	*/api/v2/device/notifications/:id*	Removes notification from queue or dismiss if it is visible

4.6.2 Display Notification

URL	/api/v2/device/notifications
Method	POST
Authorization	basic
Version	2.0.0

Description

Sends notification to the device.

Body

In order to send notification to a device you must post a Notification object.

```
{
  "priority": "[info|warning|critical]",
  "icon_type":"[none|info|alert]",
  "lifeTime":<milliseconds>,
  "model": {
   "frames": [
    {
       "icon":"<icon id or base64 encoded binary>",
       "text":"<text>"
    },
    {
      "icon": 298,
      "text":"text"
    },
    {
       "icon": 120,
       "goalData":{
           "start": 0,
           "current": 50,
           "end": 100,
           "unit": "%"
       }
    },
    {
       "chartData": [ <comma separated integer values> ]
    }
    ],
    "sound": {
      "category":"[alarms|notifications]",
       "id":"<sound_id>",
       "repeat":<repeat count>
    },
    "cycles":<cycle count>
  }
}
```

Since API 2.3.0 it is possible to use custom sounds

```
...
"sound": {
    "url": "<sound URL>",
    "type": "[mp3]",
    "fallback": {
        "category": "[alarms|notifications]",
        "id": "<sound_id>"
    }
}
...
```

Notification object

Property	Type	Description
model	Object	Message structure and data.
priority	Enum	*Optional.* Valid values are `info`, `warning`, `critical`
icon_type	Enum	*Optional.* Valid values: `none`, `info`, `alert`
lifetime	Integer	*Optional.* Lifetime of the message in milliseconds.

Detailed Properties Description

- **model** Object that represents message structure and data. It consists of "frames", "sound" and "cycles" properties.

  - *frames* – array, that describes the notification structure. Single frame can be *simple*, *goal* or *spike chart*.

    * *simple* frame consists of icon and text. Example:

      ```
      {
          "icon":"<icon_id or binary>",
          "text":"Message"
      }
      ```

      Icon can be defined as ID or in binary format. Icon ID looks like <prefix>XXX, where <prefix> is "i" (for static icon) or "a" (for animation). XXX - is the number of the icon and can be found at https://developer.lametric.com/icons or via Icons API. Binary icon string must be in this format (png):

      ```
      "data:image/png;base64,<base64 encoded png binary>"
      ```

      or gif:

      ```
      "data:image/gif;base64,<base64 encoded gif binary>"
      ```

    * *goal* frame consists of icon and goal data. Example:

      ```
      {
        "icon":120,
        "goalData":{
          "start": 0,
          "current": 50,
          "end": 100,
          "unit": "%"
        }
      }
      ```

* *spike chart* consists of array of numbers and is displayed as graph. Example:

```
{
    "chartData": [ 1, 2, 3, 4, 5, 6, 7 ]
}
```

– *sound* – object that describes the notification sound to play when notification pops on the LaMetric
Time's screen. Example:

```
{
    "category":"notifications",
    "id":"cat",
    "repeat":1
}
```

* *category* – sound category. Can be *notifications* or *alarms*.

* *id* – sound ID. Full list of notification ids:

```
bicycle
car
cash
cat
dog
dog2
energy
knock-knock
letter_email
lose1
lose2
negative1
negative2
negative3
negative4
negative5
notification
notification2
notification3
notification4
open_door
positive1
positive2
positive3
positive4
positive5
positive6
statistic
thunder
water1
water2
win
win2
wind
wind_short
```

Full list of alarm ids:

```
alarm1
alarm2
alarm3
alarm4
alarm5
alarm6
alarm7
alarm8
alarm9
alarm10
alarm11
alarm12
alarm13
```

* *repeat* – defines the number of times sound must be played. If set to 0 sound will be played until notification is dismissed. By default the value is set to 1.

Since API 2.3.0 Sound object can also look like this:

```
{
    "url":"<URL of an mp3 file>",
    "type":"mp3",
    "fallback": {
        "category" : "notifications",
        "id" : "cat"
    }
}
```

* *url* - can be any URL to the MP3 file. Currently we support the following audio formats:

* Sample Rate: 44100, 32000, 24000, 22050, 16000, 12000, 11025, 8000

* Channels: mono or stereo

* Sample Size: 16bit

* *type* - is an optional property, defines a type of the audio file. Currently *mp3* is supported. If it is missing, device will try to detect the media type using Content-Type HTTP header.

* *fallback* - object defines a built-in sound to play in case if there are any issues playing MP3 from the URL.

– *cycles* – the number of times message should be displayed. If *cycles* is set to 0, notification will stay on the screen until user dismisses it manually or you can dismiss it via the API (DELETE /api/v2/device/notifications/:id). By default it is set to 1.

• **priority** Priority of the message

– *info* – this priority means that notification will be displayed on the same "level" as all other notifications on the device that come from apps (for example facebook app). This notification will not be shown when screensaver is active. By default message is sent with "info" priority. This level of notification should be used for notifications like news, weather, temperature, etc.

– *warning* – notifications with this priority will interrupt ones sent with lower priority ("info"). Should be used to notify the user about something important but not critical. For example, events like "someone is coming home" should use this priority when sending notifications from smart home.

– *critical* – the most important notifications. Interrupts notification with priority *info* or *warning* and is displayed even if screensaver is active. Use with care as these notifications can pop in the middle of the night. Must be used only for really important notifications like notifications from smoke detectors, water leak sensors, etc. Use it for events that require human interaction immediately.

- **icon_type** Represents the nature of notification.

    - *none* – no notification icon will be shown.

    - *info* – "i" icon will be displayed prior to the notification. Means that notification contains information, no need to take actions on it.

    - *alert* – "!!!" icon will be displayed prior to the notification. Use it when you want the user to pay attention to that notification as it indicates that something bad happened and user must take immediate action.

- **lifetime** The time notification lives in queue to be displayed in milliseconds. Default lifetime is 2 minutes. If notification stayed in queue for longer than *lifetime* milliseconds – it will not be displayed.

Response

Returns success object with notification id.

```
{
  "success": {
    "id": "<notification id>"
  }
}
```

Example 1. Send notification

Request

REST:

```
POST https://<device ip address>:4343/api/v2/device/notifications

Authorization: Basic <Base64("dev:<device API key>")>
Content-Type: application/json
Accept: applciation/json

{
    "priority": "warning",
    "model": {
        "cycles": 1,
        "frames": [
            {
                "icon": "data:image/png;base64,
↪iVBORw0KGgoAAAANSUhEUgAAAAgAAAAICAYAAADED76LAAAAUklEQVQYlWNUVFBgYGBgYBC98uE/
↪AxJ4rSPAyMDAwMCETRJZjAnGgOlAZote+fCfCV0nOmA0+yKAYTwygJuAzQoGBgYGRkUFBQZ0dyDzGQl5EwCTE$NpFb6zEwAAAAI
↪",
                "text": "HELLO!"
            }
        ],
        "sound": {
            "category": "notifications",
            "id": "cat"
        }
    }
}
```

cURL:

```
$ curl -X POST -H -u "dev" -k \
  -H "Accept: application/json" \
  -H "Content-Type: application/json" \
  -d '{
      "priority": "warning",
      "model": {
          "cycles": 1,
          "frames": [
          {
              "icon": "data:image/png;base64,
↪iVBORw0KGgoAAAANSUhEUgAAAAgAAAAICAYAAADED76LAAAAUklEQVQYlWNUVFBgYGBgYBC98uE/
↪AxJ4rSPAyMDAwMCETRJZjAnGgOlAZote+fCfCV0nOmA0+yKAYTwygJuAzQoGBgYGRkUFBQZ0dyDzGQl5EwCTE$NpFb6zEwAAAA
↪",
              "text": "HELLO!"
          } ],
          "sound": {
              "category": "notifications",
              "id": "cat"
          }
      }
  }' \
  https://<device ip address>:4343/api/v2/device/notifications
$ Enter host password for user 'dev': <device API key>
```

Response

```
HTTP/1.1 200 OK

{ "success": { "id": "1" } }
```

Example 2. Send notification with custom sound

REST:

```
POST https://<device IP address>:4343/api/v2/device/notifications

Authorization: Basic <Base64("dev:<device API key>")>
Content-Type: application/json
Accept: application/json

{
    "priority": "warning",
    "icon_type": "info",
    "model": {
        "cycles": 1,
        "frames": [
            {
                "icon": "data:image/png;base64,
↪iVBORw0KGgoAAAANSUhEUgAAAAgAAAAICAYAAADED76LAAAAUklEQVQYlWNUVFBgYGBgYBC98uE/
↪AxJ4rSPAyMDAwMCETRJZjAnGgOlAZote+fCfCV0nOmA0+yKAYTwygJuAzQoGBgYGRkUFBQZ0dyDzGQl5EwCTE$NpFb6zEwAAAA
↪",
                "text": "SOUND IS PLAYING!"
            }
        ],
        "sound": {
            "url":"https://dl.espressif.com/dl/audio/gs-16b-2c-44100hz.mp3",
```

(continues on next page)

```
            "fallback": {
                "category": "notifications",
                "id": "cat"
            }
        }
    }
}
```

cURL:

```
curl --request POST 'https://<device IP address>:4343/api/v2/device/notifications' \
-u "dev:<device API key>" -k \
--header 'Accept: application/json' \
--header 'Content-Type: application/json' \
--data-raw '{
  "priority": "warning",
  "icon_type":"info",
  "model": {
      "cycles": 1,
      "frames": [
          {
            "icon":"data:image/png;base64,
↪iVBORw0KGgoAAAANSUhEUgAAAAgAAAAICAYAAADED76LAAAAUklEQVQYlWNUVFBgYGBgYBC98uE/
↪AxJ4rSPAyMDAwMCETRJZjAnGgOlAZote+fCfCV0nOmA0+yKAYTwygJuAzQoGBgYGRkUFBQZ0dyDzGQl5EwCTE$NpFb6zEwAAAAl
↪",
            "text":"SOUND IS PLAYING!"
          }
      ],
      "sound": {
        "url": "https://dl.espressif.com/dl/audio/gs-16b-2c-44100hz.mp3",
        "fallback" : {
            "category": "notifications",
            "id": "cat"
        }
      }
    }
  }'
```

Response

```
HTTP/1.1 200 OK

{ "success" : { "id" : "5" } }
```

4.6.3 Get Notification Queue

URL	/api/v2/device/notifications
Method	GET
Authorization	basic
Version	2.0.0

Description

Returns the list of all notifications in the queue. Notifications with higher priority will be first in the list.

Response

Returns array of *Notification* objects with additional fields like *created*, *exporation_date* and *type*.

```
[
  {
    "id": "<id>",
    "type": "[internal|external]",
    "priority": "[info|warning|critical]",
    "created": "<isotime>",
    "expiration_date": "<isotime>",
    "model": {...}
  }
]
```

Property	Type	Description
id	String	Notification id
type	Enum	**Notification type: `internal` or `external`.** • `External` ones come from API • `Internal` ones come from native LaMetric Time apps
priority	Enum	**Notification priority:** • `info` - put into notification queue along with internal notifications • `warning` - has higher priority than internal notifications • `critical` - interrupts other notifications and wakes the device from sleep (when screensaver is running)
created	String	Time when notification was created in ISO format.
expiration_date	String	Time when notification expires in ISO format.

Example

Request

REST:

```
GET https://<device ip address>:4343/api/v2/device/notifications

Accept: application/json
```

cURL:

```
$ curl -X GET -H -k -u "dev" \
  -H "Accept: application/json" \
  https://<device ip address>:4343/api/v2/device/notifications
$ Enter host password for user 'dev': <device API key>
```

Response

```
HTTP/1.1 200 OK

[
  {
    "id": "50",
    "type": "external",
    "priority": "info",
    "created": "2016-06-28T14:52:55",
    "expiration_date": "2016-06-28T14:54:55",
    "model": {
      "frames": [
        {
          "text": "HI!"
        }
      ]
    }
  }
]
```

4.6.4 Cancel or Dismiss a Notification

URL	/api/v2/device/notifications/:id
Method	DELETE
Authorization	basic
Version	2.0.0

Description

Removes notification from the queue or in case if it is already visible - dismisses it.

Response

Returns object with result.

```
{
  "success": true
}
```

or

```
{
  "errors": [
    {
        "message": "<error message>"
    }
  ]
}
```

Example

Request

REST:

```
DELETE https://<device ip address>:4343/api/v2/device/notifications/5
```

cURL:

```
$ curl -X DELETE -u "dev" -k https://<device ip address>:4343/api/v2/device/
↪notifications/5
$ Enter host password for user 'dev': <device API key>
```

Response

```
HTTP/1.1 200 OK

{
  "success": true
}
```

4.7 Display

4.7.1 Endpoints

Method	Path	Description
GET	*/api/v2/device/display*	Returns display state
PUT	*/api/v2/device/display*	Changes display state

4.7.2 Get Display State

URL	/api/v2/device/display
Method	GET
Authentication	basic
Version	2.0.0

Description

Returns information about the display like brightness, mode and size in pixels. Since version 2.1.0 returns information about screen saver settings.

Response

```
{
  "brightness": <0-100>,
  "brightness_mode": "[auto|manual]",
  "height": 8,
  "width": 37,
  "type": "mixed"
}
```

Since version 2.1.0 `screensaver` node has been added:

```
{
  "brightness": <0-100>,
  "brightness_mode": "[auto|manual]",
  "brightess_limit": {
    "max": 75,
    "min": 2
  }
  "brightness_range": {
   "max": 100
   "min": 0
  }
  "height": 8,
  "width": 37,
  "type": "mixed",
  "screensaver": {
    "enabled": [true|false],
    "modes": {
      "time_based": {
        "enabled": [true|false]
        "start_time": "<time in GMT>"
        "end_time": "<time in GMT>",
        "local_start_time": "<local time>",
        "local_end_time": "<local time>"
      },
      "when_dark": {
        "enabled": [true|false]
      }
    },
    "widget": "<widget_uuid>"
  }
}
```

Property	Type	Description
`brightness`	Integer	Brightness of the display. Valid values [0..100]
`brighrness_mode`	Enum	**["auto", "manual"]** • *auto* – automatically adjust brightness • *manual* – brightness is configured by the user
`width`	Integer	Width of the display in pixels
`height`	Integer	Height of the display in pixels
`type`	Enum	Display type. Valid values are : ["monochrome", "grayscale", "color", "mixed"]

Since API version 2.1.0

`screensaver`	Object	Object for screensaver configuration (off, when dark, timebased).
`brightness_range`	Object	*Optional.* The minimum and the maximum values of a brightness. When setting new brightness, the new value must be in this range. Usually valid values fall between 0 and 100.
`brightness_limit`	Object	*Optional.* Brightnesss limitations for current conditions. When a device is powered from a computer, we can limit the maximum brightness value in order to not exceed the maximum power consumption of a device to avoid unexpected power-offs. In this case the new value of brightness can not exceed the *max* value of this limit.

Screensaver

Property	Type	Description
`enabled`	Boolean	Enables or disables the screensaver.
`modes`	Map	Map of modes supported by the device. Currently "time_based" and "when_dark" are supported • *time_based* – activates if time is between start_time and end_time • *when_dark* – activates when light sensor senses darkness.
`widget`	String	UUID of the widget that should be activated when screensaver is active. Currently clock is supported.

brightness_range

Property	Type	Description
`max`	Integer	Maximum possible brightness value
`min`	Integer	Minimum possible brightness value

brightness_limit

Property	Type	Description
`max`	Integer	Current maximum brightness value limit. Can be less than *max* value of `brightness_range` (for example, due to the power comsumption limitation).
`min`	Integer	Current minimum brightness value limit

Example

Request:

```
GET https://<device ip address>:4343/api/v2/device/display
```

cURL:

```
$ curl -X GET -u "dev" -k \
  -H "Accept: application/json" \
  https://<device ip address>:4343/api/v2/device/display
$ Enter host password for user 'dev': <device API key>
```

Response:

```
HTTP/1.1 200 OK
CONTENT-TYPE: application/json;charset=UTF8
Transfer-Encoding: chunked
Date: Wed, 29 Jun 2016 13:47:05 GMT
Server: lighttpd/1.4.35
```

(continues on next page)

```json
{
  "brightness": 100,
  "brightness_limit": {
    "max": 100,
    "min": 2
  },
  "brightness_range": {
    "max": 100,
    "min": 0
  },
  "brightness_mode": "auto",
  "height": 8,
  "width": 37,
  "type": "mixed",
  "screensaver": {
    "enabled": true,
    "modes": {
      "time_based": {
        "enabled": true,
        "end_time": "18:42:56",
        "start_time": "18:41:53",
        "start_local_time": "20:41:53",
        "end_local_time": "20:42:56"
      },
      "when_dark": {
        "enabled": false
      }
    },
    "widget": "08b8eac21074f8f7e5a29f2855ba8060"
  }
}
```

4.7.3 Update Display State

URL	/api/v2/device/display
Method	PUT
Authentication	basic
Version	2.0.0

Description

Updates display state. It is possible to change brightness, mode and screen saver settings. If `brightness_mode` is set to "auto", brightness value still can be changed but this will not affect the actual brightness of the display. Brightness will be changed as soon as `brightness_mode` is set to "manual". Since API 2.1.0 it is possible to configure screensaver settings.

Body

```
{
    "brightness": <0-100>,
    "brightness_mode": "[auto|manual]"
}
```

Since API 2.1.0:

```
{
    "brightness": <0-100>,
    "brightness_mode": "[auto|manual]",
    "screensaver": {
        "enabled": [true|false],
        "mode": ["when_dark"|"time_based"]
        "mode_params": {
            "enabled": [true|false],
            "start_time": "21:00:00",
            "end_time": "06:00:00"
        }
    }
}
```

Examples

Example 1. Let's set auto brightness and enable screensaver in mode "when dark":

Request

REST:

```
PUT https://<device ip address>:4343/api/v2/device/display

Content-Type: application/json
Accept: application/json

{
    "brightness_mode": "auto",
    "screensaver": {
        "enabled" : true,
        "mode": "when_dark",
        "mode_params": {
            "enabled": true
        }
    }
}
```

cURL:

```
$ curl -X PUT -k -u "dev" \
  -H "Accept: application/json"
  -H "Content-Type: application/json" \
  -d '{
        "brightness_mode": "auto",
        "screensaver": {
            "enabled" : true,
            "mode": "when_dark",
```

<div align="right">(continues on next page)</div>

```
            "mode_params": {
                "enabled": true
            }
        }
    }' https://<device ip address>:4343/api/v2/device/display
$ Enter host password for user 'dev': <device API key>
```

Response

```
HTTP/1.1 200 OK
CONTENT-TYPE: application/json;charset=UTF8
Transfer-Encoding: chunked
Date: Wed, 29 Jun 2016 14:25:48 GMT
Server: lighttpd/1.4.35


{
    "success": {
        "data": {
            "brightness": 40,
            "brightness_mode": "auto",
            "height": 8,
            "type": "mixed",
            "width": 37
        },
        "path": "/api/v2/device/display"
    }
}
```

Since API 2.1.0:

```
HTTP/1.1 200 OK
CONTENT-TYPE: application/json;charset=UTF8
Transfer-Encoding: chunked
Date: Thu, 02 Mar 2017 16:24:18 GMT
Server: lighttpd/1.4.35


{
    "success": {
        "data": {
            "type": "mixed",
            "width": 37
            "height": 8,
            "brightness": 40,
            "brightness_mode": "auto",
            "brightness_range": {
                "max": 100,
                "min": 0
            },
            "brightness_limit": {
                "max": 100,
                "min": 2
            },
            "screensaver": {
              "enabled": true,
                "modes": {
                    "time_based": {
                        "enabled": true,
```

```
                    "end_time": "09:21:00",
                    "local_end_time": "12:21:00",
                    "local_start_time": "12:20:00",
                    "start_time": "09:20:00"
                },
                "when_dark": {
                    "enabled": false
                }
            },
            "widget": "486f5adeee1511ebab78273c1141c9c8"
        },
    },
    "path": "/api/v2/device/display"
    }
}
```

Example 2. Let's set time based screensaver from 9:00 PM to 6:00 AM. First of all, we need to convert our times to GMT. For example, if we are in GMT+2 timezone, we need to set activation time to 19:00 and deactivation time to 4:00 (subtract 2 hours from 9 pm and 6 am)

Request

REST:

```
PUT https://<device ip address>:4343/api/v2/device/display

Content-Type: application/json
Accept: application/json

{
    "screensaver": {
        "enabled" : true,
        "mode": "time_based",
        "mode_params": {
            "enabled": true,
            "start_time": "19:00:00",
            "end_time": "04:00:00"
        }
    }
}
```

cURL:

```
$ curl -X PUT -k -u "dev" \
  -H "Accept: application/json"
  -H "Content-Type: application/json" \
  -d '{
        "screensaver": {
            "enabled" : true,
            "mode": "time_based",
            "mode_params": {
                "enabled": true,
                "start_time": "19:00:00",
                "end_time": "4:00:00"
            }
        }
    }' https://<device ip address>:4343/api/v2/device/display
```

```
$ Enter host password for user 'dev': <device API key>
```

Response

```
HTTP/1.1 200 OK
CONTENT-TYPE: application/json;charset=UTF8
Transfer-Encoding: chunked
Date: Wed, 29 Jun 2016 14:25:48 GMT
Server: lighttpd/1.4.35

{
    "success": {
        "data": {
            "brightness": 40,
            "brightness_mode": "auto",
            "height": 8,
            "type": "mixed",
            "width": 37
        },
        "path": "/api/v2/device/display"
    }
}
```

Since API 2.1.0:

```
HTTP/1.1 200 OK
CONTENT-TYPE: application/json;charset=UTF8
Transfer-Encoding: chunked
Date: Thu, 02 Mar 2017 16:24:18 GMT
Server: lighttpd/1.4.35

{
    "success": {
        "data": {
            "type": "mixed",
            "width": 37
            "height": 8,
            "brightness": 100,
            "brightness_mode": "auto",
            "brightness_range": {
                "max": 100,
                "min": 0
            },
            "brightness_limit": {
                "max": 100,
                "min": 2
            },
            "screensaver": {
                "enabled": true,
                "modes": {
                    "time_based": {
                        "enabled": true,
                        "start_time": "19:00:00"
                        "end_time": "4:00:00",
                        "local_start_time": "6:00:00",
                        "local_end_time": "21:00:00"
                    },
```

```
                "when_dark": {
                    "enabled": false
                }
            },
            "widget": "08b8eac21074f8f7e5a29f2855ba8060"
        },

    },
    "path": "/api/v2/device/display"
    }
}
```

Note: All timestamps must be given in GMT.

4.8 Audio

4.8.1 Get Audio State

URL	/api/v2/device/audio
Method	GET
Authentication	basic

Description

Returns audio state such as volume.

Response

Property	Type	Description
volume	Integer	Current volume [0..100]
volume_range	Object	*Optional*. **Minimum and maximum volume values.** • min is the minimum volume value • max is the maximum volume value
volume_limit	Object	*Optional*. **Volume value limits** • min stands for *lower* volume limit • max stands for *upper* volume limit Device can limit its volume when it is powered from a computer in order to limit its power consumption and avoid unexpected power-offs. You will get error when try to set volume value that does not fall into this range.

Examples

Request:

```
GET http://192.168.0.239:8080/api/v2/device/audio
```

Response:

```
HTTP/1.1 200 OK
CONTENT-TYPE: application/json;charset=UTF8
Transfer-Encoding: chunked
Date: Wed, 29 Jun 2016 14:56:31 GMT
Server: lighttpd/1.4.35

{
    "volume" : 69,
    "volume_range": {
        "max": 100,
        "min": 0
    },
    "volume_limit": {
        "max": 69,
        "min": 0
```

(continues on next page)

```
    }
}
```

4.8.2 Update Audio State

URL	/api/v2/device/audio
Method	PUT
Authentication	basic

Description

Updates audio state.

Body

```
{
    "volume" : 69
}
```

Response

```
HTTP/1.1 200 OK
CONTENT-TYPE: application/json;charset=UTF8
Transfer-Encoding: chunked
Date: Wed, 29 Jun 2016 14:59:15 GMT
Server: lighttpd/1.4.35

{
    "success" : {
        "data" : {
            "volume" : 69,
            "volume_range": {
                "max": 100,
                "min": 0
            },
            "volume_limit": {
                "max": 69,
                "min": 0
            }
        },
        "path" : "/api/v2/device/audio"
    }
}
```

4.9 Bluetooth

4.9.1 Get Bluetooth State

URL	/api/v2/device/bluetooth
Method	GET
Authentication	basic

Description

Returns Bluetooth state.

Response

Property	Type	Description
available	Boolean	Indicates whether device has Bluetooth module on board or not.
active	Boolean	Indicates whether Bluetooth module is enabled or not.
discoverable	Boolean	Indicates whether Bluetooth module is visible for other Bluetooth devices or not.
pairable	Boolean	Indicates whether other devices can pair with LaMetric Time.
name	String	Name of the LaMetric visible via Bluetooth discovery.
mac	String	LaMetric Time Bluetooth MAC address.

Examples

Request:

```
GET http://192.168.0.239:8080/api/v2/device/bluetooth
```

Response:

```
HTTP/1.1 200 OK
CONTENT-TYPE: application/json;charset=UTF8
Transfer-Encoding: chunked
Date: Wed, 29 Jun 2016 15:11:42 GMT
Server: lighttpd/1.4.35

{
    "active" : false,
    "available" : true,
    "discoverable" : false,
    "mac" : "58:63:56:23:95:6C",
    "name" : "LM0001",
    "pairable" : true
}
```

4.9.2 Update Bluetooth State

URL	/api/v2/device/bluetooth
Method	PUT
Authentication	basic

Description

Updates Bluetooth state.

Body

Property	Type	Description
active	Boolean	*Optional.* True – activates Bluetooth module, false – deactivates.
name	String	*Optional.* Sets new Bluetooth name

Example

```
{
    "active" : true,
    "name" : "LaMetric Time"
}
```

Response

```
HTTP/1.1 200 OK
CONTENT-TYPE: application/json;charset=UTF8
Transfer-Encoding: chunked
Date: Wed, 29 Jun 2016 15:23:07 GMT
Server: lighttpd/1.4.35

{
    "success" : {
        "data" : {
            "active" : true,
            "available" : true,
            "discoverable" : false,
            "mac" : "58:63:56:23:95:6C",
            "name" : "LaMetric Time",
            "pairable" : true
        },
        "path" : "/api/v2/device/bluetooth"
    }
}
```

4.10 Wi-Fi

4.10.1 Get Wi-Fi State

URL	/api/v2/device/wifi
Method	GET
Authentication	basic

Description

Returns Wi-Fi state.

Response

Property	Type	Description
available	Boolean	Indicates whether device has Wi-Fi module on board or not.
active	Boolean	Indicates whether Wi-Fi is active or not.
encryption	Enum	Wi-Fi encryption – ["OPEN", "WEP", "WPA", "WPA2"]
ipv4	String	IP address of LaMetric Time in local network
mac	String	Mac address of Wi-Fi module
mode	Enum	["static", "dhcp"]
netmask	String	Network mask
signal_strength	Integer	Quality of Wi-Fi signal in range between [0..100]
ssid	String	Name of the Wi-Fi hotspot device is connected to

Examples

Request:

```
GET http://192.168.0.239:8080/api/v2/device/wifi
```

Response:

```
HTTP/1.1 200 OK
CONTENT-TYPE: application/json;charset=UTF8
Transfer-Encoding: chunked
Date: Wed, 29 Jun 2016 15:31:24 GMT
Server: lighttpd/1.4.35

{
    "available" : true,
    "active" : true,
    "encryption" : "WPA",
    "ipv4" : "192.168.0.225",
    "mac" : "58:63:56:23:6E:5C",
    "mode" : "dhcp",
    "netmask" : "255.255.255.0",
    "signal_strength" : 100,
    "ssid" : "homewifi"
}
```