SmartThings Web Services Implementation

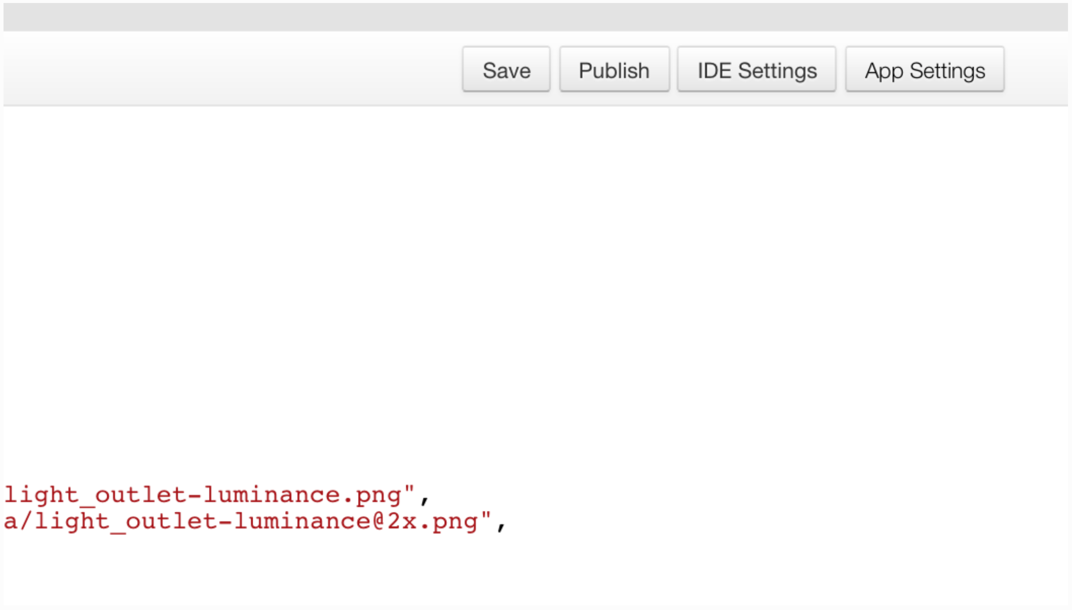
Developing an “API Access” Application

Step 1: Create a new Device Type in the IDE. Click on the tab “From Code” and paste the content of the file device_type.groovy.

Step 2: Self-publish the device type from the IDE by clicking on “Publish” -> “For Me”.

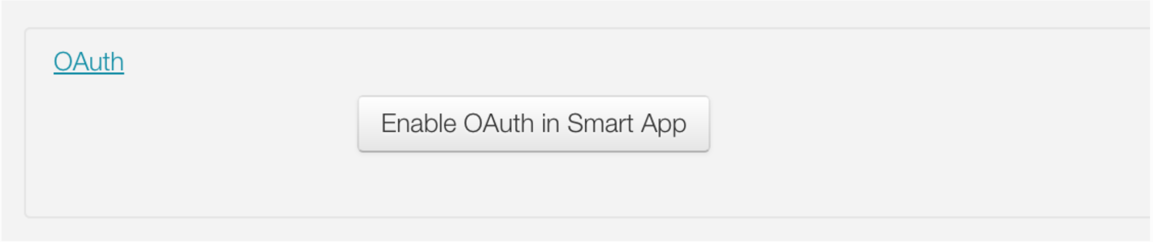
Step 3: Create a new SmartApp in the IDE. Click on the tab “From Code” and paste the content of the file service_manager.groovy.

Step 4: Click on “App Settings”

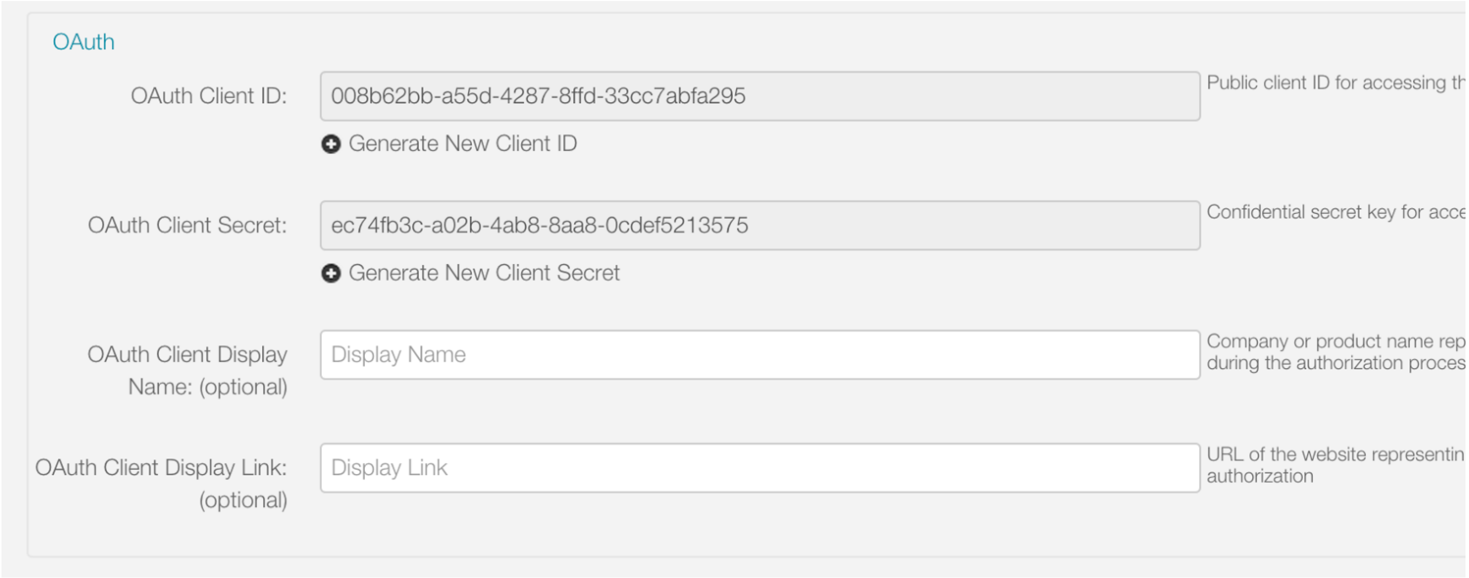


```
light_outlet-luminance.png",
a/light_outlet-luminance@2x.png",
```

Step 5: Enable OAuth in Smart App to receive an auto-generated Client ID and Client Secret.



For this example we'll assume myclient and mysecret. You should also set an **OAuth Client Display Name**, as it will show up in your authorization prompt later on.


OAuth

OAuth Client ID: 008b62bb-a55d-4287-8ffd-33cc7abfa295

⊕ Generate New Client ID

Public client ID for accessing th

OAuth Client Secret: ec74fb3c-a02b-4ab8-8aa8-0cdef5213575

⊕ Generate New Client Secret

Confidential secret key for acce

OAuth Client Display Name: (optional) Display Name

Company or product name rep during the authorization proces

OAuth Client Display Link: (optional) Display Link

URL of the website representin authorization

Step 6: Self-publish the app from the IDE by clicking on "Publish" -> "For Me".

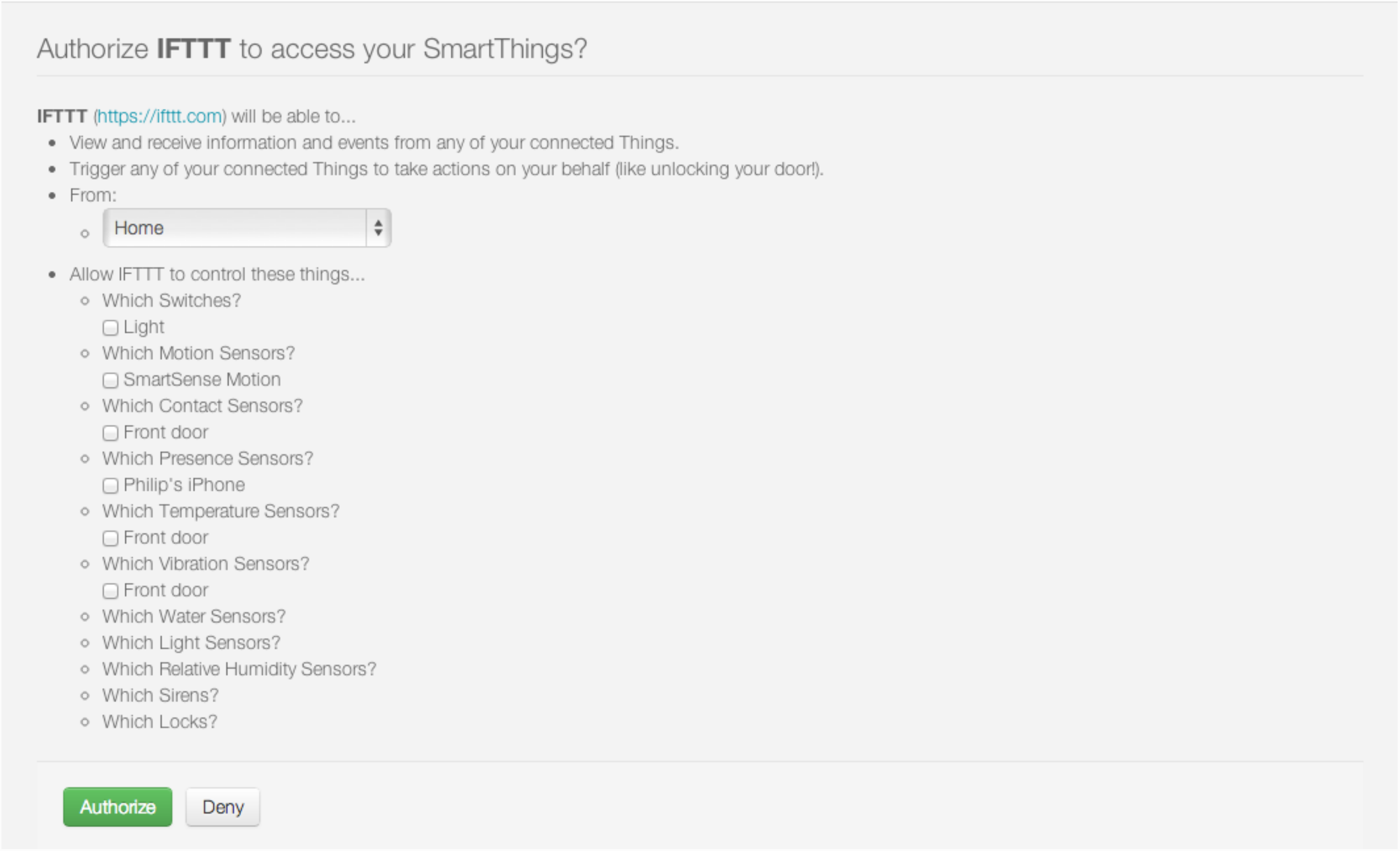
Step 7: Obtain an OAuth authorization code.

The authorization code (once returned) will take the place of a password in the next step of getting the OAuth access token.

Retrieve an authorization code by first authenticating with SmartThings. Set the client_id to the value specified in your SmartApp, and the redirect_uri to a location you have setup to handle these OAuth 2 requests. For this example, I’ve used a value that corresponds to a small Sinatra app defined later in this document.

```
Request:
GET
https://graph.api.smartthings.com/oauth/authorize?response_type=code&client_id=myclient&scope=app&red
irect_uri=https%3A%2F%2Fgraph.api.smartthings.com%2Foauth%2Fcallback
```

The user clicks “Authorize” when prompted.

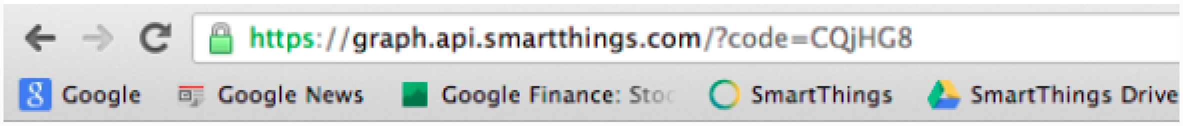


Authorize **IFTTT** to access your SmartThings?

IFTTT (https://ifttt.com) will be able to...

- View and receive information and events from any of your connected Things.
- Trigger any of your connected Things to take actions on your behalf (like unlocking your door!).
- From:
  - Home
- Allow IFTTT to control these things...
  - Which Switches?
    □ Light
  - Which Motion Sensors?
    □ SmartSense Motion
  - Which Contact Sensors?
    □ Front door
  - Which Presence Sensors?
    □ Philip's iPhone
  - Which Temperature Sensors?
    □ Front door
  - Which Vibration Sensors?
    □ Front door
  - Which Water Sensors?
  - Which Light Sensors?
  - Which Relative Humidity Sensors?
  - Which Sirens?
  - Which Locks?

A request with a parameter called “code” will now be sent to redirect_uri, which can be used to exchange for an access token in a separate request, shown below:



https://graph.api.smartthings.com/?code=CQjHG8

Step 8: Obtain an access token using the grant.

Now that you have the code (CQjHG8 in the URL above), you can make an HTTP request from the SmartThings OAuth endpoint using: + The Code + The OAuth Client ID + The OAuth Client Secret

This request will return a JSON document that contains the OAuth2 access token that you will use to make subsequent requests.

```
Request:
GET
https://graph.api.smartthings.com/oauth/token?grant_type=authorization_code&client_id=myclient&client
_secret=mysecret&redirect_uri=https%3A%2F%2Fgraph.api.smartthings.com%2Foauth%2Fcallback&scope=app&co
de=XXXXX

Response:
200 OK
{
  "access_token": "43373fd2871641379ce8b35a9165e803",
  "expires_in": 1576799999,
  "token_type": "bearer"
}
```

Note that while the normal flow of the mechanics of getting the OAuth2 access token are invisible to the enduser, the entire flow can be done as shown above using nothing but a web browser. This allows you to get an access token for testing purposes.

Step 9: Discover the endpoint URL in the following way, passing the token to the specified URL.

```
Request:
https://graph.api.smartthings.com/api/smartapps/endpoints/myclient?access_token=43373fd2871641379ce8b
```

```
35a9165e803
```

```
Response:
200 OK
[
  {
  "oauthClient": {
    "clientId": "myclient",
    "authorizedGrantTypes": "authorization_code"
  },
  "url": "/api/smartapps/installations/8a2aa0cd3df1a718013df1ca2e3f000c"
  }
]
```

Step 10: Make API calls to your app as follows:

```
Request:
```

https://graph.api.smartthings.com/api/token/43373fd2871641379ce8b35a9165e803**/smartapps/installations/**8a2aa0cd3df1a718013df1ca2e3f000c**/on**

https://graph.api.smartthings.com/api/token/43373fd2871641379ce8b35a9165e803**/smartapps/installations/**8a2aa0cd3df1a718013df1ca2e3f000c**/off**

https://graph.api.smartthings.com/api/token/43373fd2871641379ce8b35a9165e803**/smartapps/installations/**8a2aa0cd3df1a718013df1ca2e3f000c**/value/33**

Where “33” is the target_temp but it can be any numerical number.

```
Response:
200 OK
```

For more information please visit:
http://docs.smartthings.com/en/latest/smartapp-web-services-developers-guide/implementation.html