Introduction Of Tuya Node-RED Flow

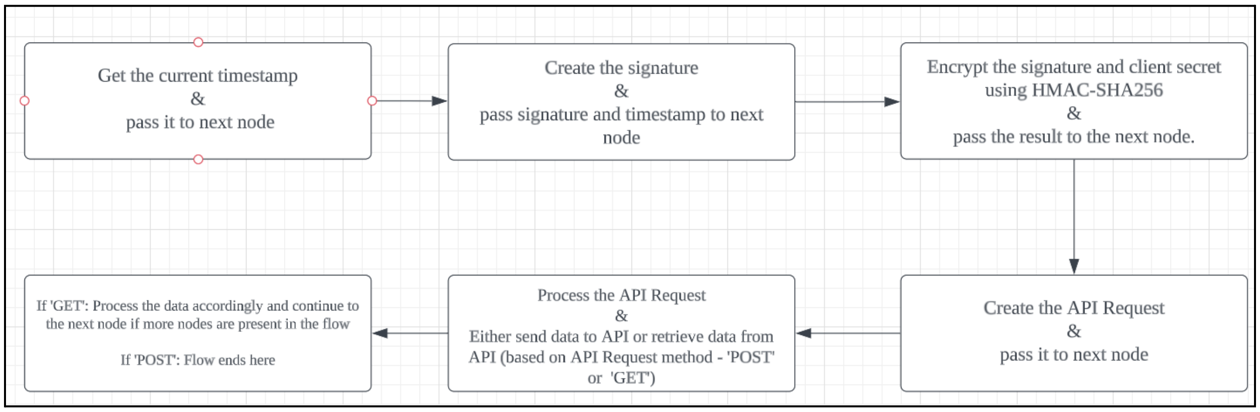
Table 1. List of APIs used in Tuya's Node-RED Flow

API Name	**Service Name**	**Capabilities (with relevance to project)**	**Limitations (with relevance to project)**
Get A Token	Authorization Token Management	Gets an access token which is used by all other APIs. The access token comes with a refresh token and expire time.	Access token is valid for 3 hours and needs to be refreshed before it expires.
Refresh Token	Authorization Token Management	Refreshes the access token every 3 hours, preemptively renewing it 5 seconds before expiration via the refresh token.	None
Get User's Device List	Smart Home Basic Service > Smart Home User Management	Gets the id, name and product name of each device registered under the user account.	None
Get The Specifications And Properties Of The Device	IoT Core > Device Control (Standard Instruction Set)	Gets the properties of 'cur_power' for each device. The properties consist of unit, min & max values, scale and step.	None
Query Device Log	Smart Home Basic Service > Smart Home User Management	Gets the 'cur_power' data logs of each device every 12 minutes.	Max of 100 logs per API call which includes logs for other types of data. Filtering process does not exclude other types of logs from being counted within the specified limit of 100 logs.
Device Control (Implemented but not used)	Smart Home Basic Service > Smart Home Device Control	Sends commands to turn on/off the grow lights of each rack.	None

This table consists of all the APIs that were used inside the Node-RED flow for Tuya. Each API request has a request structure - the request method, the full API url (consists of the endpoint & the API url) and the request header parameters [1]. The endpoint used is 'https://openapi.tuyaeu.com' as the data center chosen in the Tuya Cloud project is Central Europe Data Center. As for the API url and the request method, these can be found through the API Explorer by using the service name, followed by the API name (Refer to Table 1) and then, going to the 'View Docs' section. The API url also includes the required parameters (based on the red star) and any other parameters will depend on the usage of the API. For example, for ‘Get A Token’ API, the API url is ‘/v1.0/token?grant_type=1’ and the request method is ‘GET’.

Generally, the header parameters consist of the client id, signature, sign method, current timestamp and active access token. For the client id, it can be obtained from the Tuya Cloud project under the ‘Overview’ section. The signature is made of the client id, the current timestamp, the http method of the API, the Content-SHA256 and the API url [2]. The Content-SHA256 refers to the value of the request body and for our Tuya flow, most of the Content-SHA256 value is the default ‘empty body’ value provided in the signature documentation. Once the signature has been made, it goes through a HMAC-SHA256 algorithm that uses the signature and the client secret (obtained from our Tuya Cloud project) to create the message digest. This message digest is then converted to uppercase and used as the signature we needed inside the API request header.

The flow is divided into 4 sections - Token section, Device List section, Device Log section & Send Command section. Each section (may consists of more than 1 part) follows the general flow as shown in Figure 1.

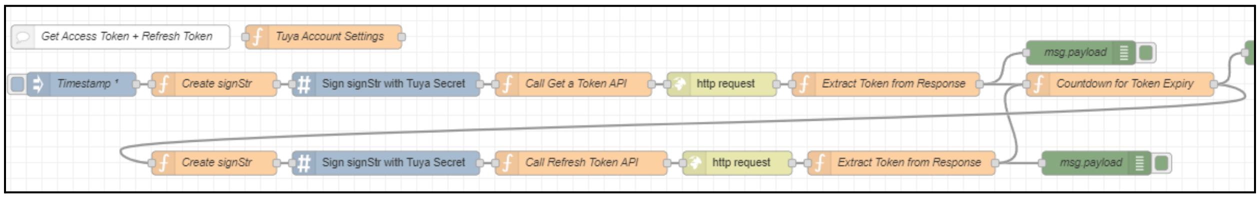


Get the current timestamp & pass it to next node
Create the signature & pass signature and timestamp to next node
Encrypt the signature and client secret using HMAC-SHA256 & pass the result to the next node.
Create the API Request & pass it to next node
Process the API Request & Either send data to API or retrieve data from API (based on API Request method - 'POST' or 'GET')
If 'GET': Process the data accordingly and continue to the next node if more nodes are present in the flow
If 'POST': Flow ends here

Figure 1. General Flow of Each Section

Token Section

In Figure 2, inside the ‘Tuya Account Settings’, the client id and the client secret is stored into a flow context which is like a temporary memory space that can hold these data so that it can be used for any API request later.



Get Access Token + Refresh Token
Tuya Account Settings
Timestamp
Create signStr
Sign signStr with Tuya Secret
Call Get a Token API
http request
Extract Token from Response
msg.payload
Countdown for Token Expiry
Create signStr
Sign signStr with Tuya Secret
Call Refresh Token API
http request
Extract Token from Response
msg.payload

Figure 2. Token Section of Tuya Flow

"The Token section consists of two parts - 'Get A Token' and 'Refresh Token'. This section handles token management, where the initial access token is obtained in 'Get a Token', and it is refreshed every 3 hours in 'Refresh Token'. To ensure seamless token renewal, the access token is refreshed 5 seconds before its expiration.

'Get a Token' covers the 1st inject node till the 1st 'Extract Token from Response' node. It gets the initial access token, a refresh token and the expire time using the 'Get A Token' API and stores these data into the flow context. Then, the next node which is 'Countdown for Token Expiry', begins a countdown loop on the expiry time begins which decreases the expire time every second. Once the expire time has reached 5 seconds, the loop stops and gets the current timestamp to pass it on to the next node which is the start of 'Refresh Token'.

'Refresh Token' covers the 2nd 'Create signStr' node till the 2nd 'Extract Token from Response' node. It refreshes the access token using the refresh token, and then generate a new access token, refresh token and expire time. These data is stored into the flow context, replacing the old data. Then, the last node of 'Refresh Token' is connected to the 'Countdown for Token Expiry' to ensure seamless renewal of access token every 3 hours. The nodes labelled 'msg.payload' serve debugging purposes.

This section is run immediately after initial Node-Red deployment to ensure that the other sections are able to work properly as Tuya APIs requires access token.

Device List Section

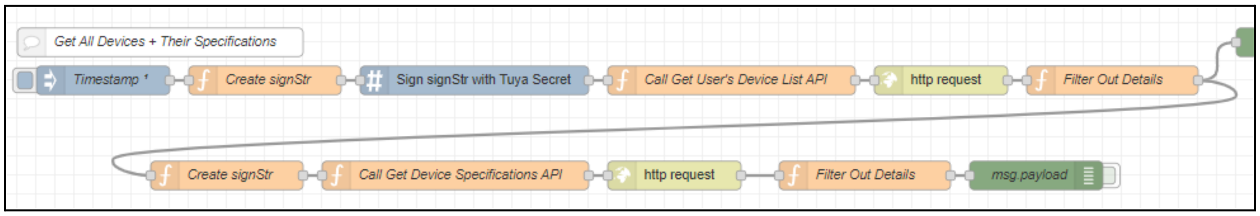

Get All Devices + Their Specifications
Timestamp
Create signStr
Sign signStr with Tuya Secret
Call Get User's Device List API
http request
Filter Out Details
Create signStr
Call Get Device Specifications API
http request
Filter Out Details
msg.payload

Figure 3. Device List Section of Tuya Flow

The Device List section consists of 2 parts - 'Get User's Device List' and 'Get Device Specifications'. This section retrieves information about each device registered under the Tuya Account in 'Get User's Device List' and the properties of 'cur_power' data of each device in 'Refresh Token'. The 'cur_power' data is the power consumption data that needs to be stored into InfluxDB and the value of 'cur_power' data is in whole number rather than its actual value which is in decimals. So, this is where the scale of 'cur_power' data of each device comes in. The scale indicates the number of decimal places to consider when interpreting the value of 'cur_power' data.

'Get User's Device List' covers the 1st inject node till the 1st 'Filter Out Details' node. It gets the device id, device name and product name of each device using the 'Get User's Device List' API and stores these data as a list in the flow context. It also gets the current timestamp and passes it to the next node which is the start of 'Get Device Specifications'.

'Get Device Specifications' covers the 2nd ‘Create signStr’ node till the 2nd ‘Filter Out Details’ node. It gets the unit, min, max, scale and step properties of ‘cur_power’ data of each device using the 'Get Device Specifications' API and stores these data into the same list from 'Get User's Device List' in the flow context. The nodes labelled ‘msg.payload’ serve debugging purposes.

This section is run once 5 seconds after initial Node-Red deployment. However, if there are changes such as changing a device’s name, removal of device and addition of device, this section will need to be run again to update the list that was stored in the flow context as this list is used in the Device Log & Command section.

Device Log Section

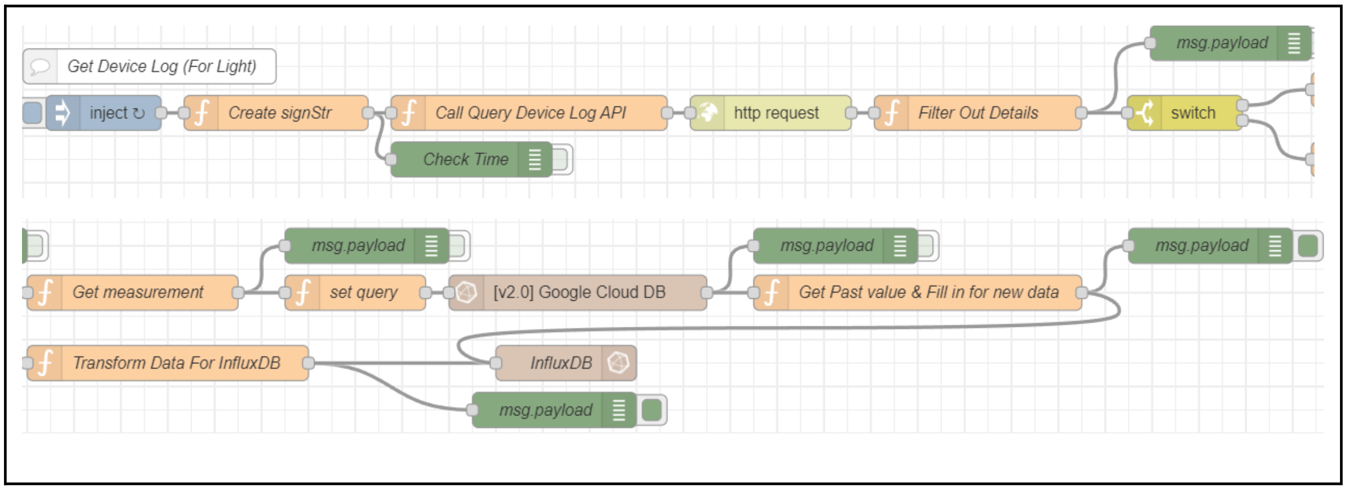


Figure 4. Device Log Section of Tuya Flow

The Device Log section consists of 4 of the same part but each part is catered to different category based on the device names - Light devices, Water devices, Other devices (doesn’t contain Light or Water its name) and All devices. This is to account for scalability as current solution only monitors the power consumption for 3/9 racks. This section retrieves the ‘cur_power’ logs of each device from the Tuya Cloud every 12 minutes using the ‘Query Device Log’ API and process them before storing into InfluxDB.

The ‘Query Device Log’ API requires the start time and end time as parameters and thus, in the ‘Create signStr’ node, a simple calculation is performed to get these data - the end time is the current timestamp and the start time is the end time - 720000 milliseconds (which is equivalent to 12 minutes). There are other parameters included such as the event type which is 7 and the code which is ‘cur_power’ according to the API documentation. So, these parameters ensure that the power consumption data sent from the device to the Tuya Cloud is retrieved.

Additionally, inside the ‘Create signStr’ node, a filter is added to only get devices according to the category of the part. For example, for the Light devices part in Figure 4, it filters for all devices with ‘Light’ in its name from the list in Device List section. Next, a loop is created to go through each device to get its device id, create and encrypt the signature individually as each device needs its own signature.

Then, all these data - start time, end time, event type, code, device id, signature and current timestamp are sent to the next node.

The nex node which is 'Call Query Device Log API' will then process the data sent and create the API Request separately for each device and pass the API Request to the next node which is the 'http request' node. The node processes the API Request and retrieve the specified logs from the 'Query Device Log' API and sent to the 'Filter Out Details' node. This node process the retrieved logs using a loop which calculates the cumulative power consumption based on the value and duration of each log and store the cumulative power consumption as a variable called 'totalPower'. Once the loop reaches the last log, a new log is created with these data - code which is 'cur_power', value which is the totalPower, timestamp of the last log, device id, device name, device unit (which is the unit for the value) and device scale (the decimal places). The new log is then sent to the next node which is a 'switch' node.

The 'switch' node is added for cases where there was no retrieved logs which happens when there is no power consumption data sent from the device to the Tuya Cloud in the 12 minutes interval. So, the node checks if the payload sent from the previous node is empty or not. If it is not empty, the node sends the log over to the 'Transform Data For InfluxDB' node. This node transforms the log based on InfluxDB's data point format and sends it over to be stored into InfluxDb via the 'influx batch' node.

However, if the 'switch' node detects that the payload is empty, it calls the 'Get measurement' node which gets the respective measurement in the Power Consumption bucket in InfluxDB according to the device name. The measurement name is similar to the device name. For example, the device name is 'Rack 3 Light' and the corresponding measurement that contains all data for this device is 'Rack_3_Light'. Once the node gets the measurement name, it sends the measurement name to the next node which is the 'Set Query' node. This node creates the query to retrieve the past hour records for the respective measurement from InfluxDB. Then, it passes this query to the next node which is the 'influxdb in' node. The 'influxdb in' node processes the query and retrieves the specified records and sends them to the next node which is 'Get Past value & Fill in for new data' node.This node processes the retrieved records and finds the latest record. Once it does, it takes the details of that record and creates a new data point based on those details. Then, it sends the newly created data point to be stored into InfluxDB via the 'influx batch' node.

This section is run 10 seconds after initial Node-Red deployment and is run every 12 minutes from 12am to 12pm. As of now, current solution uses the parts catered to light devices and water devices. Do note that the 12 minute interval works only with the current solution as the current solution only monitors the power consumption for 3/9 racks. Make the necessary modifications to this section or to the Tuya Cloud Project if more racks are being monitored as there's is a monthly quota of API calls per month which is 26000 for the trial edition. Exceeding this quota will lead to a suspension of the service until the quota gets refreshed the next month [3].

Send Command Section

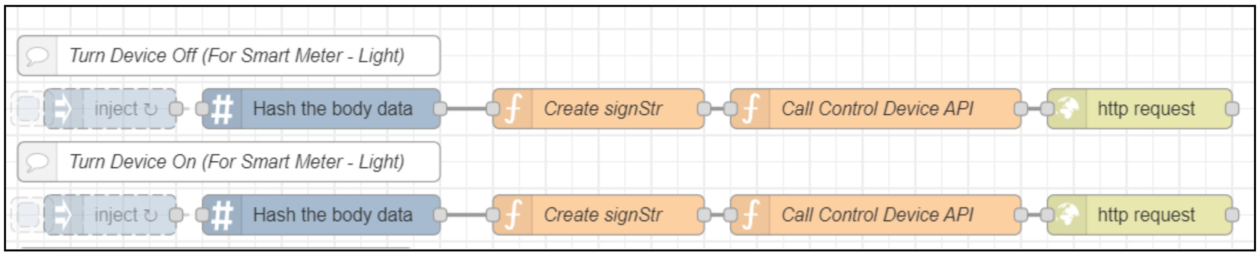


Figure 5. Send Command Section of Tuya Flow

The Send Command section consists of 4 parts - Turn Device Off (For Smart Meter - Light), Turn Device On (For Smart Meter - Light), Turn Device Off (For Smart Plug - Light), Turn Device On (For Smart Plug - Light). This section send command to the 'Device Control' API to either turn off or turn on the respective device at the specified time. Each part only caters to devices with 'Light' in its name and the parts are divided into 2 category - 'For Smart Meter' which is the Atorch socket and 'For Smart Plug' which is the Smatrul plug. The division is necessary as the socket and the plug has different commands to turn off and turn on the connected device.

So, each part in Figure 5, it starts off with an 'inject' node that consists of the current timestamp and the respective command. For example, the Atorch socket's commands to turn on or to turn off the connected device are {"commands":[{"code":"r_mode","value":"open"}]} and {"commands":[{"code":"r_mode","value":"close"}]} respectively. Then, the timestamp and command are sent to the next node which encrypts the command using HMAC-SHA256. The encrypted command and the timestamp is sent to the 'Create signStr' node. So, in this case, the Content-SHA256 needed for the signature is using the encrypted command instead of the default empty body value. Then, in 'Create signStr' node, a loop which filters through the list in Device List section and gets only the devices that contains light in its name and whether the product name of the device is 'Smart Meter' or 'Smart Plug'. The loop also creates a signature is for each device and the node passes this signature, the devic id and the timestamp to the next node which is the 'Call Control Device API' node. This node processes the data it gets from the previous node and creates the API Request for each device and sends this API Request to the next node which is the 'http request' node. This node processes the API Request of each device and send the command to the API which then, turns off or turns on the device connected to the socket or the plug accordingly.

As of now, current solution does not deploy this section as it is not within the project scope. However, this section can be considered for future work as a replacement for the timer that is used to set the growlight schedule of each rack. Currently, each part is set to run at 7am for Turn On Device and 7pm for Turn Off Device.

Do note that this section requires the Atorch socket or the Smatrul plug to be directly connected to the growlights to be able to turn them on or off. Also, to know which code corresponds with turning on or off the connected device for the socket or the plug, the information can be found in the 'Device Debugging' section on Tuya IoT platform. To get to the 'Device Debugging' section, the steps are go to the Tuya

Cloud project > click on ‘Devices’ > click on ‘Debug Device’ of a device > click on ‘Device Debugging’ tab. Most of the time, the code is ‘switch_1’ but for some cases like the Atorch socket, the code is ‘r_mode’. Get in touch with Tuya’s tech support if unsure of which code to use [4].

References

[1] https://developer.tuya.com/en/docs/iot/api-request?id=Ka4a8uuo1j4t4
[2] https://developer.tuya.com/en/docs/iot/new-singnature?id=Kbw0q34cs2e5g
[3] https://developer.tuya.com/en/docs/iot/membership-service?id=K9m8k45jwvg9j
[4] https://service.console.tuya.com/