Data Lantern Open Data Exchange Application Interfaces (API)

External API Document

Date: August 31, 2018
Draft Version 0.3

Table of Contents

1 INTRODUCTION 3

2 ABBREVIATION AND ACRONYMS 3

3 DATA LANTERN DEVICE DATA DEFINITIONS 3

4 APPLICATION INTERFACES (API) 5

4.1 SET DATA LANTERN SERVER 5
4.2 LOGIN TO THE DATA LANTERN SERVER 5
4.3 LOGOUT OF THE DATA LANTERN SERVER 6
4.4 QUERY DEVICES VISIBLE TO A USER 6
4.4.1 Get API Token to Send and Retrieve Data and Control 6
4.4.2 List of Devices Owned by User 6
4.4.3 Get Manufacturer and Configuration Information of a Specific Device 6
4.4.4 Get Device's Current List of Authorized User Access 6
4.4.5 Get List of Other Devices that a User Has Access to 6
4.4.6 Grant a User Data Access to My Device 6
4.4.7 Remove a User Data Access to My Device 7
4.4.8 Grant a User Control Access to My Device 7
4.4.9 Remove a User Control Access to My Device 7
4.5 POST DATA TO THE DATA LANTERN ECOSYSTEM 7
4.6 RETRIEVE DATA FROM A SPECIFIC DEVICE 8
4.7 RETRIEVE LATEST CONTROL SETTING REQUESTED 9
4.8 SEND A CONTROL SETTING CHANGE 9

1 Introduction

The Data Lantern Open Data Exchange is an architecture and ecosystem to aggregate data from smart sensors and devices. These devices and sensors are often referred to as Internet of Things (IoT). Some IoT devices and sensors may directly connect to the Internet, while others broadcast their data to the owner's / user's local data collector (an IoT gateway or a mobile device application).

While most IoT platforms and ecosystems focus on direct interconnection and passing data between supported devices (M2M), Data Lantern Open Data Exchange focuses on aggregating data into a centralized environment where users and manufacturers can control how their data are used, shared, or potentially sold. The Data Lantern Open Data Exchange utilizes a standard device data definition scheme that allows anyone or any device to query data from and/or control other devices independent of manufacturers.

Users can register their IoT devices and aggregate their data into the Data Lantern Open Data Exchange. Some benefits of the Data Lantern Open Data Exchange include:

- A simple monentization model that provides incentives for users and manufacturers to share their data with other devices / manufacturers, service providers, applications developers, data researcher / analytic, …etc
- A data-centric integration model between devices based on a standardized device data format / scheme
- An environment for IoT devices to query and retrieve data from other devices without directly connecting to them (some devices may not always be online)
- A new service model that gives servie providers insight into problematic devices without physically touching them

This API document covers, at very high level, different API's supported by Data Lantern. For detailed API syntax, type supported, and returned variables / code / status code; developers are encouraged to visit the Data Lantern public repository in GitHub for actual support libraries and sample program code.

2 Abbreviation and Acronyms

IoT	Internet of Things
IoE	Internet of Everything (Cisco)
JSON	JavaScript Object Notation
M2M	Machine to Machine
NO-SQL	Not Only SQL
REST	Representational State Transfer

3 Data Lantern Device Data Definitions

At the core of the Data Lantern Open Data Exchange, all devices basically has five categories of data:

1. Device Information (mandatory) – these are data that describe the manufacturer, model, version, current state of a device, and if device is and how to connect directly (such as REST over https);
2. Standard device data (mandatory) – based on the device type, these are basic mandatory data that the device can provide. Each device updates the ecosystem a fixed set of key-value pairs. Each key is a predefined string, and the value can be a single value, a set of values, or a complex structure of key value pairs.

3. Controllable device data (optional) – these are data that can be adjusted or controlled if the user permits. When a controllable variable is set, the device will read the changed setting(s) on its next data update interval (returned from the sendData API).
4. Extended device data (optional) – these are data beyond the standard device data that a device can provide. The extended device data is provided by manufacturer and must be consistent for a specific model. A newer version of a device model may expand the extended data set. For backward compatibility, extended data cannot be reduced unless the version of the device is no longer supported by the manufacturer.
5. Extended device controls (optional) – these are controls beyond the standard controls that a device can support. The extended controls are provided by manufacturer and must be consistent for a specific model. A newer version of a device model may expand the extended controls. For backward compatibility, extended controls cannot be reduced unless the version of the device is no longer supported by the manufacturer

Each device is stored into the Data Lantern Open Data Exchange relational database with the following information:

Name
Type
Description
Manufacturer's Name
Model Number
Serial Number
Version Number
Barcode (UPC code or QR code)
Configuration
Custom Fields (JSON object with key/value pairs)
Parent Device

When a device is created, a unique internal device ID is created. All API access to a device is based on the device ID and an API Token associated with the user.

The device data are stored in a NO-SQL database with JSON format. The JSON format to store the device data are:

```
"data":{
   "standard":{
      /* Standard data provided based on device categories / subcategories */
   },
   "extended":{
      /* Manufacturer specific non-standard data provided based on device model */
   }
},
"control":{
   "standard":{
      /* Devcie specific standard controls supported by this device category / subcategory */
   },
   "extended":{
      /* Manufacturer specific non-standard controls supported based on device model */
   }
}
```

To allow greatest flexibility in presenting the data, common units of measurement are included in the tag naming. For example, the temperature reading from a thermostat can use JSON tags "temperature_f" (in Fahrenheit) or "temperature_c" (Celsius). Only common units of measurement are supported in the Data Lantern Open Data Exchange. If a device or sensor uses unit of measurement not supported by Data Lantern, the device is expected to convert the reading / data to one of the supported unit of measurement. Device manufacturer can also partition Data Lantern to support the new unit of measurement.

4 Application Interfaces (API)

This section covers the different Python REST API's to interact with the Data Lantern Open Data Exchange.

For query API's that return a list of data or devices, the following JSON tags are also returned:

```
"from":          /* start of the element return in the page returned */
"last_page":     /* total number of pages of resultant */
"next_page":     /* full URL path of the next page */
"path":          /* full URL path of the request */
"per_page":      /* number of elements returned per page */
"prev_page_url": /* full URL path of the previous page */
"to":            /* end of the elements return in the page returned */
"total":         /* total number of elements from the request */
```

For example, the follow JSON tags are returned with the data for the 1st page (15 elements) of 23 pages (345 total elements) for device with ID 13:

```
"from": 1,
"last_page": 23,
"next_page": "https://datalantern.com/api/v1/data/13?page=2",
"path": "https://datalantern.com/api/v1/data/13",
"per_page": 15,
"prev_page_url": null,
"to": 15,
"total": 345
```

If the list exceeds more than 15, the API will return the 15 items of the list, and the number of pages of the total resultant.

4.1 Set Data Lantern Server

This API sets the name or IP address of the Data Lantern Open Data Exchange server:

API : setHostname(name)

4.2 Login to the Data Lantern Server

All API access to the Data Lantern Open Data Exchange server requires authenticating the user credential with password. Data Lantern uses the user email address as the credential:

API : login(email, password)

4.3 Logout of the Data Lantern Server

To guard against potential bridge of user data, applications using the Data Lantern API should terminate the session with the logout:

API : logout( )

4.4 Query Devices Visible to a User

4.4.1 Get API Token to Send and Retrieve Data and Control

This API returns the API token that is specific to the user. The API token is used for all store and retrieve operations.

API: getApiToekn()

4.4.2 List of Devices Owned by User

This API returns the list of devices currently owned by the authenticated user.

API : getMyDevices(page)

This API returns the 15 devices owned by the authenticated user. If the return list is longer than 15, the API will also return the total number of pages (list of 15) of devices that the user owns. The subsequent API call can use the page parameter to jump to specific page.

4.4.3 Get Manufacturer and Configuration Information of a Specific Device

This API returns the manufacturer information and configuration of the specified device:

API: getMyDevice(deviceId)

4.4.4 Get Device's Current List of Authorized User Access

This API returns a list of users who currently have access to a specific device. The caller must be owner of the device.

API: getUserAccess(deviceId)

4.4.5 Get List of Other Devices that a User Has Access to

This API returns a list of devices that the current user has access to. If the list is longer than 15, the API will also return the total number of pages (list of 15) of devices that the user does not own but has access to. The subsequent API call use the page parameter to jump to specific page.

API: getOtherDevices(page)

4.4.6 Grant a User Data Access to My Device

This API grants another user data access to a caller's device.

API: addUserAccessToDeviceData(deviceId, other_user_email_address)

4.4.7 Remove a User Data Access to My Device

This API removes a user data access to a caller's device.

API: removeUserAccessToDeviceData(DeviceId, other_user_email_address)

4.4.8 Grant a User Control Access to My Device

This API grants another user control access to a caller's device.

API: addUserAccessToDeviceControl(deviceId, other_user_email_address)

4.4.9 Remove a User Control Access to My Device

This API removes a user control access to a caller's device..

API: removeUserAccessToDeviceControl(deviceId, other_user_email_address)

4.5 Post Data to the Data Lantern Ecosystem

Devices own by a user has a uniquely assigned token. Only callers with the valid token may store data for the associated devices. Before sending data to the Data Lantern Open Data Exchange, the application must call the API to set the token. Only one call to the set token API is needed for the entire application session.

API : setApiToken(token)

It's expected that both device data and current control settings are sent / stored into the Data Lantern Open Data Exchange. The API to send data for a particular device to the Data Lantern Open Data Exchange:

API : sendData(deviceID, data)

A Date / Timestamp of when the data was created is expected as part of the JSON data (with JSON tag of "creation_time"). A Python example of packaging data, control, and timestamp before sending to the Data Lantern Open Data Exchange:

```
#   get the now date and time and pacakge into Data Lantern object
    now = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    dataObj = { 'data' : data, 'control' : control, 'creation_time' : now }
```

When the data is actually stored into the Data Lantern Open Data Exchange NO-SQL database, an internal NO-SQL database entry ID (with JSON tag of "_id", and a Date / Timestamp on when the data is actually stored into the database (with JSON tag of "created_at") are created:

```
"_id": "5b18efa606de0e10134ad8ab",
"created_at": "2018-06-07 08:41:10",
```

The response to sendData call is either success or fail with the JSON tag of "success" with value of either true or false. If the sendData request is successful, new requested control settings (both standard and extended) are returned. These new control settings indicate change(s) to the control settings have been requested, and the application can act based on these new control settings. Only requested changes to control are returned (ie not all control settings). If there are no new control setting change requested, the "Control" and "Extended Control" will be empty.

The format of the response are:

Successful store data request:

```
{
        "success": true,
        "Control" : {
                /* Standard control changes requestd */
        },
        "Extended Control" : {
                /* Extended control changes requested */
        }
}
```

Failed store data request:

```
{
        "success" : false,
        "error" : {
                "code" : 123,
                "message" : "123 – error message"
        }
}
```

4.6 Retrieve Data from a Specific Device

This API retrieves the data from a particular device. The "page" parameter is used to control the page of data to be retrieved. If page is not specified, the first page (page = 1) is assumed. A page of -1 retrieves the last page of the data, which may not be a full page (namely, length of list mod page size). The "limit" parameter can be used to control the number of data item returned per page. The default "limit" is 15 and maximum is 100.

API : getData(deviceId, page, limit)

An example of page 3 of device data from device ID of 15 returned:


```
{
    "data": {
        "current_page": 3,
        "data": [
            {
                "_id": "5b1956f806de0e2c3f42ab35",
                "created_at": "2018-06-07 16:02:00",
                "data":
"{\"control\":{\"extended\":{\"hold_until\":\"0\"},\"standard\":{\"heatset_f\":\"63\",\"fan
status\":\"off\",\"cool_set_f\":\"99\",\"coolstatus\":\"auto\",\"heatstatus\":\"auto\"}},\"
creation_time\":\"2018-06-07
09:02:00\",\"data\":{\"extended\":{\"humidity_%\":\"43\"},\"standard\":{\"temperature_f\":\
"71\"}}}",
```

```
                "device_id": 15,
                "updated_at": "2018-06-07 16:02:00"
            },
                 :
                 :
                 :
            {
                "_id": "5b19886c06de0e2c3951b8ea",
                "created_at": "2018-06-07 19:33:00",
                "data":
"{\"control\":{\"extended\":{\"hold_until\":\"0\"},\"standard\":{\"heatset_f\":\"63\",\"fan
status\":\"off\",\"cool_set_f\":\"99\",\"coolstatus\":\"auto\",\"heatstatus\":\"auto\"}},\"
creation_time\":\"2018-06-07
12:33:00\",\"data\":{\"extended\":{\"humidity_%\":\"43\"},\"standard\":{\"temperature_f\":\
"72\"}}}",
                "device_id": 15,
                "updated_at": "2018-06-07 19:33:00"
            }
        ],
        "from": 31,
        "last_page": 11,
        "next_page_url": "https://lantern.com/api/v1/data/15?page=4",
        "path": "https://lantern.com/api/v1/data/15",
        "per_page": 15,
        "prev_page_url": "https://lantern.com/api/v1/data/15?page=2",
        "to": 45,
        "total": 156
    },
    "status": "success"
}
```


4.7 Retrieve Latest Control Setting Requested

This API retrieves the latest control setting requested for a particular device:

API : getLatestControl(deviceId)

4.8 Send a Control Setting Change

This API sends a control data change request for a particular device:

API : sendControl(deviceId, controlData)