Home Graph - *A data platform for the home*

Introduction

Home Graph is based on SmartThings a cloud based platform designed to support IOT applications. SmartThings itself has changed little in terms of its value since its introduction in 2013. There have of course been updates but the core focus and user benefits have changed little. At it's core SmartThings is about devices connected to a hub or directly to the smartThings cloud organized by motivated users into Locations and Rooms. SmartThings also provides a developer workspace to enable extensions to the primary SmartThings user experience on a Mobile device via "smartApps"and automation services. SmartThings also provides a REST API for more general development.

The AIX team in SRA have been working fairly intensely with the SmartThings REST API over the past 18 months creating new user experiences. As with many REST interfaces getting at the information you really need can take multiple calls. This caused the AIX team a few issues when trying to write voice based and new client experiences. For example a voice request to *"turn on all the lights in the kitchen"* might seem simple enough but in order to pull all the information needed using the REST API requires at least two REST calls and some filtering. This requires the building of custom services to support individual voice requests. The situation was similar in an advanced AR experience where trying to parse out the devices in a room required a lot of client side logic dealing with, and navigating the information encapsulated by the smartThings API.

Given these kinds of issues a new approach was needed. It was clear that SmartThings had created a logical data structure for a home around rooms so viewing the smart home as a hierarchical data structure it was clear what was needed was an efficient way to navigate that structure.

Home Graph uses SmartThings data and is based on GraphQL, a query language developed and open-sourced by Facebook. It provides a powerful API to get exactly the data you need in a single request, seamlessly traversing and combining data sources. Using Graphql rather than REST for SmartThings allows applications and services to make complex queries or updates in one call. Home Graph also supports the seamless extension of the the SmartThings API allowing both service and data integration.

The use of Graphql to simplify access to smartThings data opened an entirely new way to view data in the home. One where new services could be added to augment the SmartThings structure or entirely separate services related to the home or family could be included. What Home Graph is, is less a smartthings simplification but more a general data model for the home.

The rest of this document discusses the use of Home Graph.

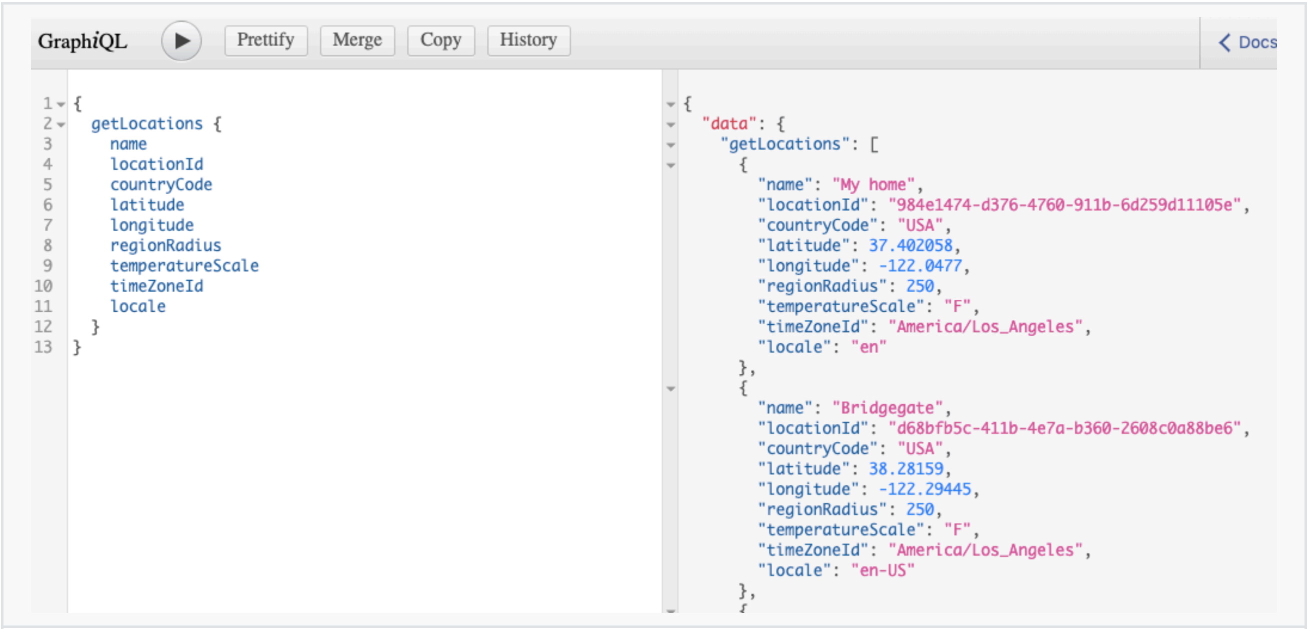
Home Graph Queries

SmartThings exposes a logical structure of Locations (Homes), Rooms and Devices so the Home Graph adopts this general structure and extends it.

Locations

A **Location** in general equates to a Home, or a significant geolocation managed by smartThings. From a user standpoint Locations are named using a familiar moniker, from a SmartThings standpoint Locations have a unique Id. A Location can be further organized into Rooms, such as "Kitchen" or "Bedroom".

The general model for Location and the default fields as defined my SmartThings is shown below

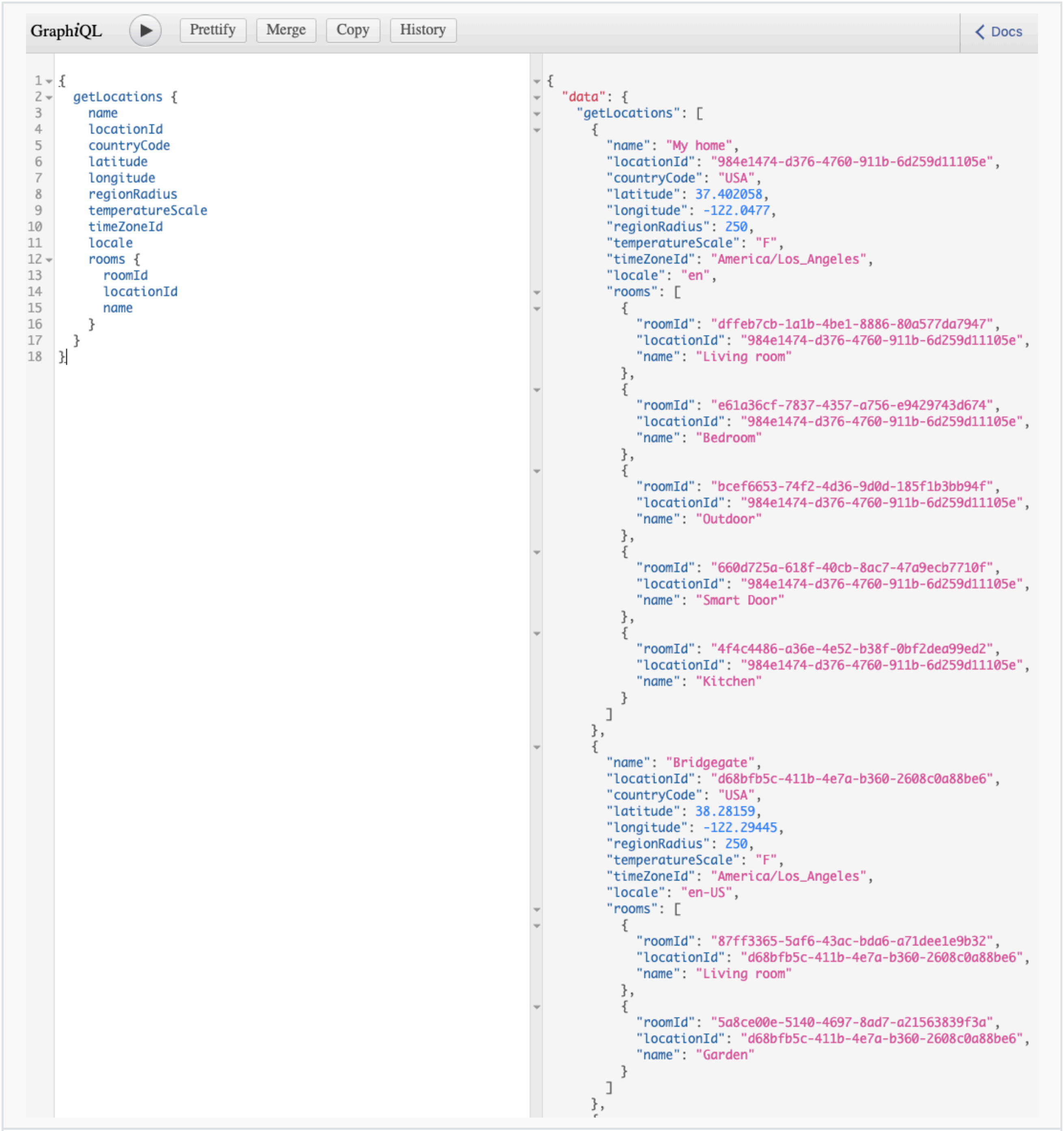


```
{
  getLocations {
    name
    locationId
    countryCode
    latitude
    longitude
    regionRadius
    temperatureScale
    timeZoneId
    locale
  }
}
```

```
{
  "data": {
    "getLocations": [
      {
        "name": "My home",
        "locationId": "984e1474-d376-4760-911b-6d259d11105e",
        "countryCode": "USA",
        "latitude": 37.402058,
        "longitude": -122.0477,
        "regionRadius": 250,
        "temperatureScale": "F",
        "timeZoneId": "America/Los_Angeles",
        "locale": "en"
      },
      {
        "name": "Bridgegate",
        "locationId": "d68bfb5c-411b-4e7a-b360-2608c0a88be6",
        "countryCode": "USA",
        "latitude": 38.28159,
        "longitude": -122.29445,
        "regionRadius": 250,
        "temperatureScale": "F",
        "timeZoneId": "America/Los_Angeles",
        "locale": "en-US"
      },
```

Query to get all Locations for account

The result fields shown (on right side) in the image above contain the typical fields of comparable SmartThings REST call however via Graphql it's possible to dive much deeper into the SmartThings Structure. The Location includes a `rooms` field which pulls back the room information for each location.

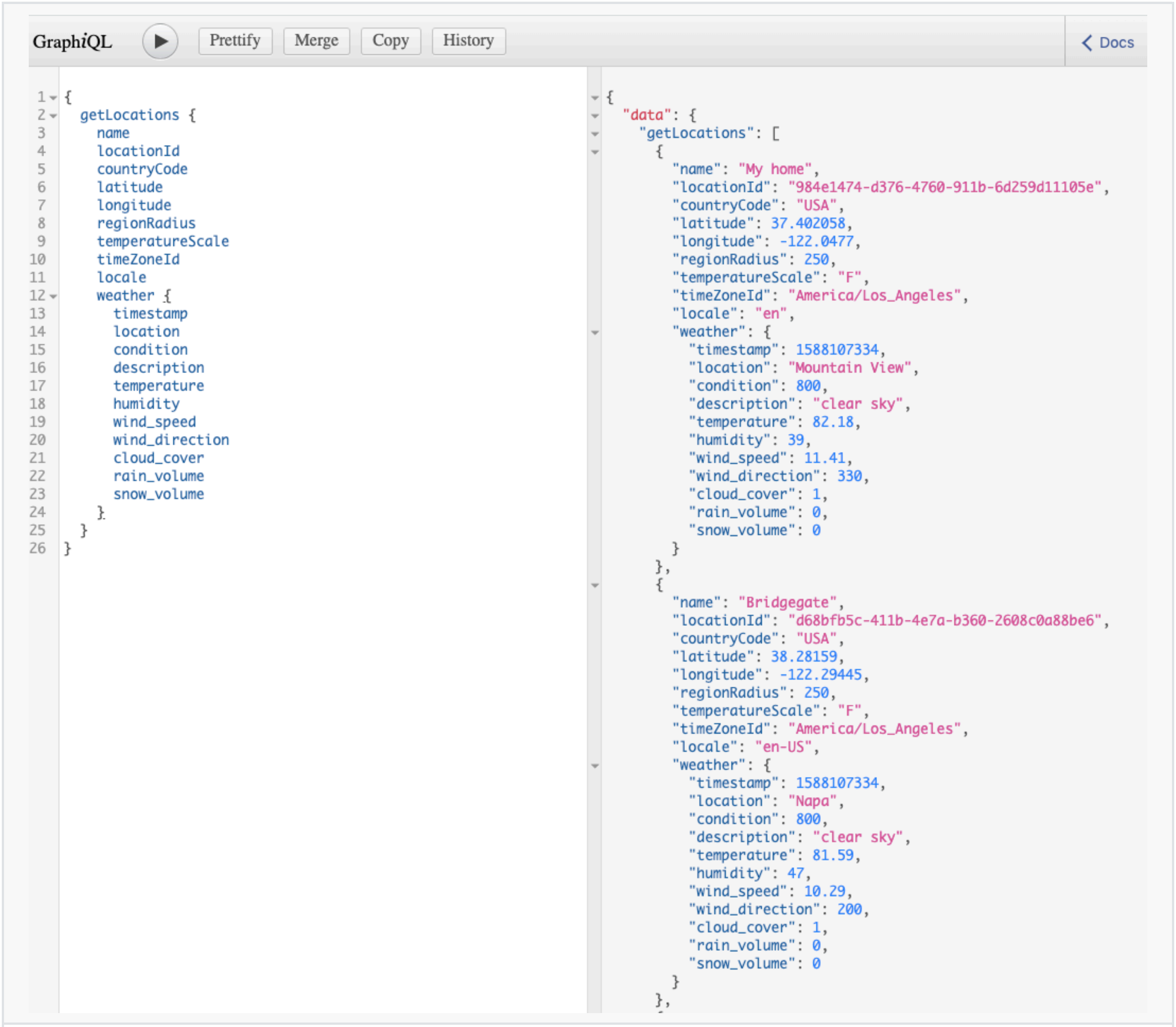


```
{
  getLocations {
    name
    locationId
    countryCode
    latitude
    longitude
    regionRadius
    temperatureScale
    timeZoneId
    locale
    rooms {
      roomId
      locationId
      name
    }
  }
}
```

```
{
  "data": {
    "getLocations": [
      {
        "name": "My home",
        "locationId": "984e1474-d376-4760-911b-6d259d11105e",
        "countryCode": "USA",
        "latitude": 37.402058,
        "longitude": -122.0477,
        "regionRadius": 250,
        "temperatureScale": "F",
        "timeZoneId": "America/Los_Angeles",
        "locale": "en",
        "rooms": [
          {
            "roomId": "dffeb7cb-1a1b-4be1-8886-80a577da7947",
            "locationId": "984e1474-d376-4760-911b-6d259d11105e",
            "name": "Living room"
          },
          {
            "roomId": "e61a36cf-7837-4357-a756-e9429743d674",
            "locationId": "984e1474-d376-4760-911b-6d259d11105e",
            "name": "Bedroom"
          },
          {
            "roomId": "bcef6653-74f2-4d36-9d0d-185f1b3bb94f",
            "locationId": "984e1474-d376-4760-911b-6d259d11105e",
            "name": "Outdoor"
          },
          {
            "roomId": "660d725a-618f-40cb-8ac7-47a9ecb7710f",
            "locationId": "984e1474-d376-4760-911b-6d259d11105e",
            "name": "Smart Door"
          },
          {
            "roomId": "4f4c4486-a36e-4e52-b38f-0bf2dea99ed2",
            "locationId": "984e1474-d376-4760-911b-6d259d11105e",
            "name": "Kitchen"
          }
        ]
      },
      {
        "name": "Bridgegate",
        "locationId": "d68bfb5c-411b-4e7a-b360-2608c0a88be6",
        "countryCode": "USA",
        "latitude": 38.28159,
        "longitude": -122.29445,
        "regionRadius": 250,
        "temperatureScale": "F",
        "timeZoneId": "America/Los_Angeles",
        "locale": "en-US",
        "rooms": [
          {
            "roomId": "87ff3365-5af6-43ac-bda6-a71dee1e9b32",
            "locationId": "d68bfb5c-411b-4e7a-b360-2608c0a88be6",
            "name": "Living room"
          },
          {
            "roomId": "5a8ce00e-5140-4697-8ad7-a21563839f3a",
            "locationId": "d68bfb5c-411b-4e7a-b360-2608c0a88be6",
            "name": "Garden"
          }
        ]
      },
```

Query to get all Locations plus rooms for account

One Graphql request returns what would have taken multiple REST calls to the SmartThings API. This is not all, since a smartThings Location includes geolocation new geo sevices can be added. For example the [Location]() includes fields for `weather` and `airQuality` which have nothing to do with the smartThings API at all. These are third party services that can easily be added integrated into the Home Graph API. For example this:



```
{
  getLocations {
    name
    locationId
    countryCode
    latitude
    longitude
    regionRadius
    temperatureScale
    timeZoneId
    locale
    weather {
      timestamp
      location
      condition
      description
      temperature
      humidity
      wind_speed
      wind_direction
      cloud_cover
      rain_volume
      snow_volume
    }
  }
}
```

```
{
  "data": {
    "getLocations": [
      {
        "name": "My home",
        "locationId": "984e1474-d376-4760-911b-6d259d11105e",
        "countryCode": "USA",
        "latitude": 37.402058,
        "longitude": -122.0477,
        "regionRadius": 250,
        "temperatureScale": "F",
        "timeZoneId": "America/Los_Angeles",
        "locale": "en",
        "weather": {
          "timestamp": 1588107334,
          "location": "Mountain View",
          "condition": 800,
          "description": "clear sky",
          "temperature": 82.18,
          "humidity": 39,
          "wind_speed": 11.41,
          "wind_direction": 330,
          "cloud_cover": 1,
          "rain_volume": 0,
          "snow_volume": 0
        }
      },
      {
        "name": "Bridgegate",
        "locationId": "d68bfb5c-411b-4e7a-b360-2608c0a88be6",
        "countryCode": "USA",
        "latitude": 38.28159,
        "longitude": -122.29445,
        "regionRadius": 250,
        "temperatureScale": "F",
        "timeZoneId": "America/Los_Angeles",
        "locale": "en-US",
        "weather": {
          "timestamp": 1588107334,
          "location": "Napa",
          "condition": 800,
          "description": "clear sky",
          "temperature": 81.59,
          "humidity": 47,
          "wind_speed": 10.29,
          "wind_direction": 200,
          "cloud_cover": 1,
          "rain_volume": 0,
          "snow_volume": 0
        }
      },
```

Query to get weather at all Locations for account

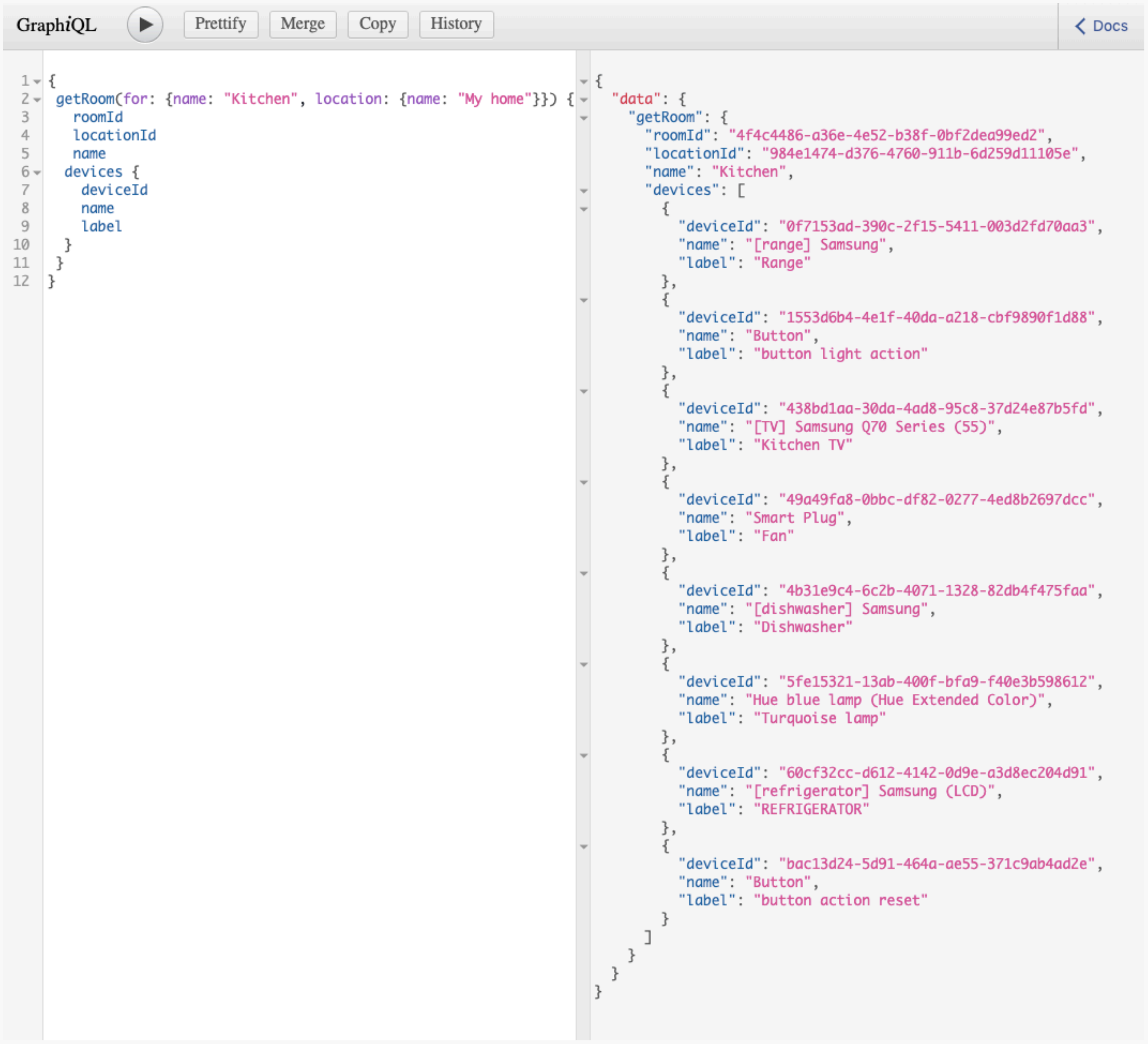
Demonstrates the user of the `weather` field which pull weather from a cloud service. This seamless integration demonstrates the power of Home Graph to create a comprehensive data layer for the home. There are many methods to enable such integration;

- schema inclusion - as shown in the weather case
- schema stitching - where two schemas are stitched together at runtime from separate services (e.g Yelp Graphql API)
- federation - The use of two graphql services.

Using Home graph new services and other data models (e.g home blog, shopping list, tv guide, noticeboard) can be integrated very easily and able new client experiences as well as new device experiences to be created.

Rooms

Access to rooms from a Location has been shown above. In SmartThings Rooms contain devices and the Home graph Room type includes a property to get all the devices in a room



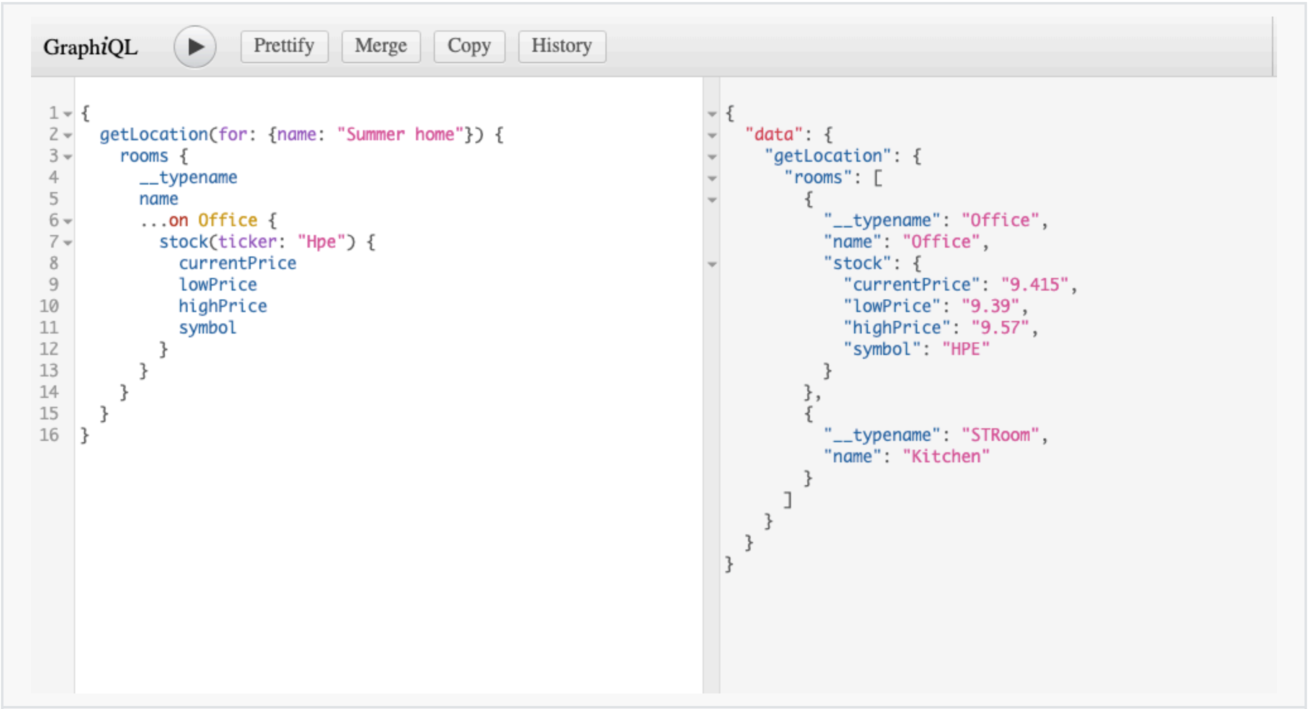
```
{
 getRoom(for: {name: "Kitchen", location: {name: "My home"}}) {
   roomId
   locationId
   name
  devices {
    deviceId
    name
    label
  }
 }
}
```

```
{
  "data": {
    "getRoom": {
      "roomId": "4f4c4486-a36e-4e52-b38f-0bf2dea99ed2",
      "locationId": "984e1474-d376-4760-911b-6d259d11105e",
      "name": "Kitchen",
      "devices": [
        {
          "deviceId": "0f7153ad-390c-2f15-5411-003d2fd70aa3",
          "name": "[range] Samsung",
          "label": "Range"
        },
        {
          "deviceId": "1553d6b4-4e1f-40da-a218-cbf9890f1d88",
          "name": "Button",
          "label": "button light action"
        },
        {
          "deviceId": "438bd1aa-30da-4ad8-95c8-37d24e87b5fd",
          "name": "[TV] Samsung Q70 Series (55)",
          "label": "Kitchen TV"
        },
        {
          "deviceId": "49a49fa8-0bbc-df82-0277-4ed8b2697dcc",
          "name": "Smart Plug",
          "label": "Fan"
        },
        {
          "deviceId": "4b31e9c4-6c2b-4071-1328-82db4f475faa",
          "name": "[dishwasher] Samsung",
          "label": "Dishwasher"
        },
        {
          "deviceId": "5fe15321-13ab-400f-bfa9-f40e3b598612",
          "name": "Hue blue lamp (Hue Extended Color)",
          "label": "Turquoise lamp"
        },
        {
          "deviceId": "60cf32cc-d612-4142-0d9e-a3d8ec204d91",
          "name": "[refrigerator] Samsung (LCD)",
          "label": "REFRIGERATOR"
        },
        {
          "deviceId": "bac13d24-5d91-464a-ae55-371c9ab4ad2e",
          "name": "Button",
          "label": "button action reset"
        }
      ]
    }
  }
}
```

Query room information a devices

A room within a home serves a function therefore the Home Graph Room type is infact a base-type (interface) that can be specialized for the function of a room. Most homes have a space to prepare food, a space to sleep and a space to be entertained or relax. Each room can have specialized services attached to them which support the function of that space. An online cookbook in the kitchen prehaps, a set of books for relaxing before bed, a playlist or watchlist in the living room.

This kind customization really demonstrates the power of Home Graph in supporting unique experiences in the home.



```
{
  getLocation(for: {name: "Summer home"}) {
    rooms {
      __typename
      name
      ...on Office {
        stock(ticker: "Hpe") {
          currentPrice
          lowPrice
          highPrice
          symbol
        }
      }
    }
  }
}
```

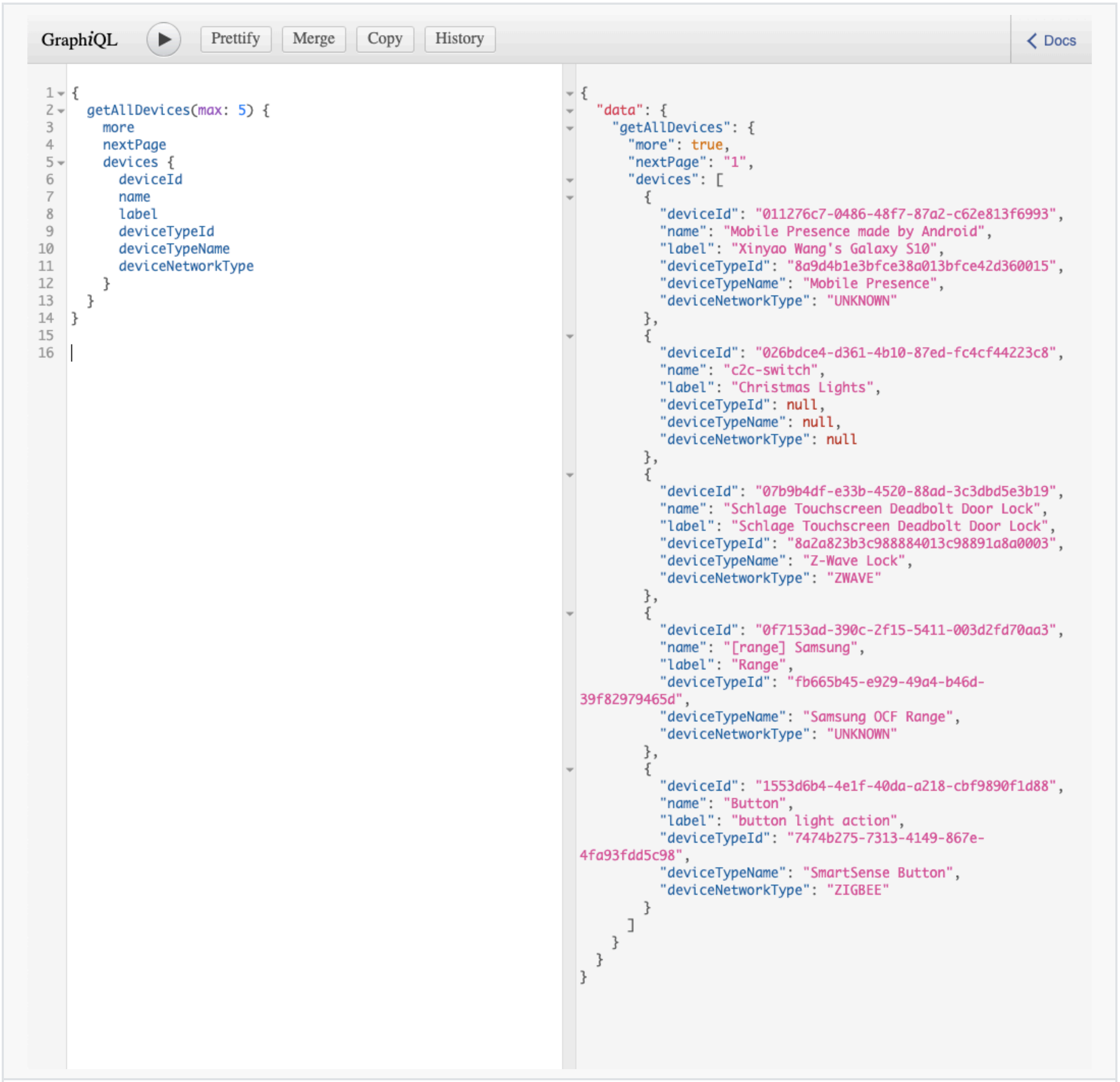
```
{
  "data": {
    "getLocation": {
      "rooms": [
        {
          "__typename": "Office",
          "name": "Office",
          "stock": {
            "currentPrice": "9.415",
            "lowPrice": "9.39",
            "highPrice": "9.57",
            "symbol": "HPE"
          }
        },
        {
          "__typename": "STRoom",
          "name": "Kitchen"
        }
      ]
    }
  }
}
```

Rooms of type Office have a Stock Service Attached

In the above image a stock ticker sevice has been attached to an Office. This service supports the function of the Office and the service will supply stock quotes. An ambient display or TV in the Office could take advantage of this service and show a ticker tape for relevant stocks

Devices

The lowest level of the SmartThings system is the device layer, Devices belong to Locations and Rooms. The Rooms object has a field to go an retrieve the devices within a room as shown above. There is also a global query which can be used to find all the devices in an Account.

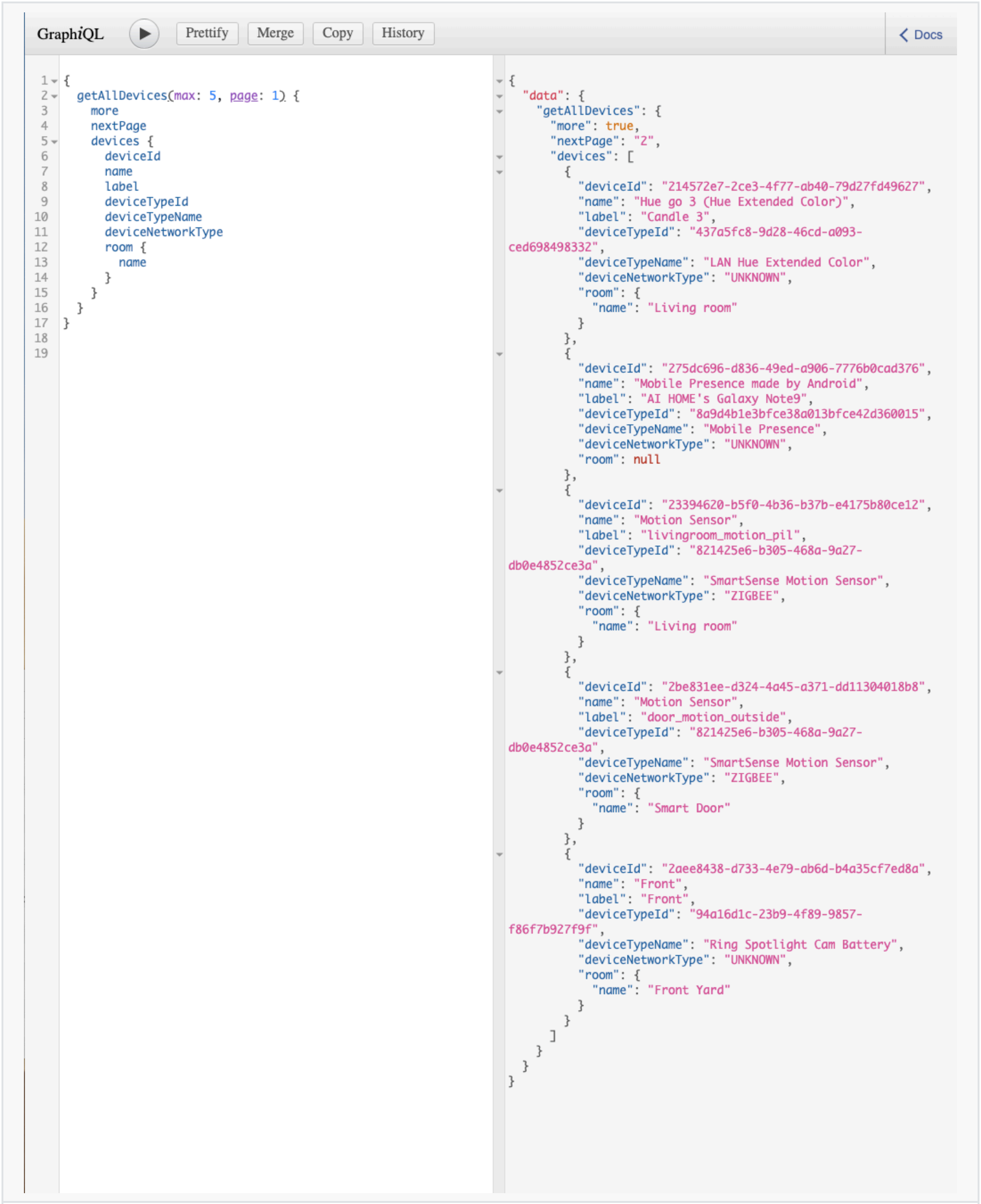


```
{
  getAllDevices(max: 5) {
    more
    nextPage
    devices {
      deviceId
      name
      label
      deviceTypeId
      deviceTypeName
      deviceNetworkType
    }
  }
}

```

```
{
  "data": {
    "getAllDevices": {
      "more": true,
      "nextPage": "1",
      "devices": [
        {
          "deviceId": "011276c7-0486-48f7-87a2-c62e813f6993",
          "name": "Mobile Presence made by Android",
          "label": "Xinyao Wang's Galaxy S10",
          "deviceTypeId": "8a9d4b1e3bfce38a013bfce42d360015",
          "deviceTypeName": "Mobile Presence",
          "deviceNetworkType": "UNKNOWN"
        },
        {
          "deviceId": "026bdce4-d361-4b10-87ed-fc4cf44223c8",
          "name": "c2c-switch",
          "label": "Christmas Lights",
          "deviceTypeId": null,
          "deviceTypeName": null,
          "deviceNetworkType": null
        },
        {
          "deviceId": "07b9b4df-e33b-4520-88ad-3c3dbd5e3b19",
          "name": "Schlage Touchscreen Deadbolt Door Lock",
          "label": "Schlage Touchscreen Deadbolt Door Lock",
          "deviceTypeId": "8a2a823b3c988884013c98891a8a0003",
          "deviceTypeName": "Z-Wave Lock",
          "deviceNetworkType": "ZWAVE"
        },
        {
          "deviceId": "0f7153ad-390c-2f15-5411-003d2fd70aa3",
          "name": "[range] Samsung",
          "label": "Range",
          "deviceTypeId": "fb665b45-e929-49a4-b46d-
39f82979465d",
          "deviceTypeName": "Samsung OCF Range",
          "deviceNetworkType": "UNKNOWN"
        },
        {
          "deviceId": "1553d6b4-4e1f-40da-a218-cbf9890f1d88",
          "name": "Button",
          "label": "button light action",
          "deviceTypeId": "7474b275-7313-4149-867e-
4fa93fdd5c98",
          "deviceTypeName": "SmartSense Button",
          "deviceNetworkType": "ZIGBEE"
        }
      ]
    }
  }
}
```

Query All Devices with Max Results

As can be seen from the above the query to get all devices can take a max input which enables paging. The the case above the query wants a max of 5 responses. The `nextPage` response indicates the index of the next page which can be passed into the query as shown below in the `page` param.

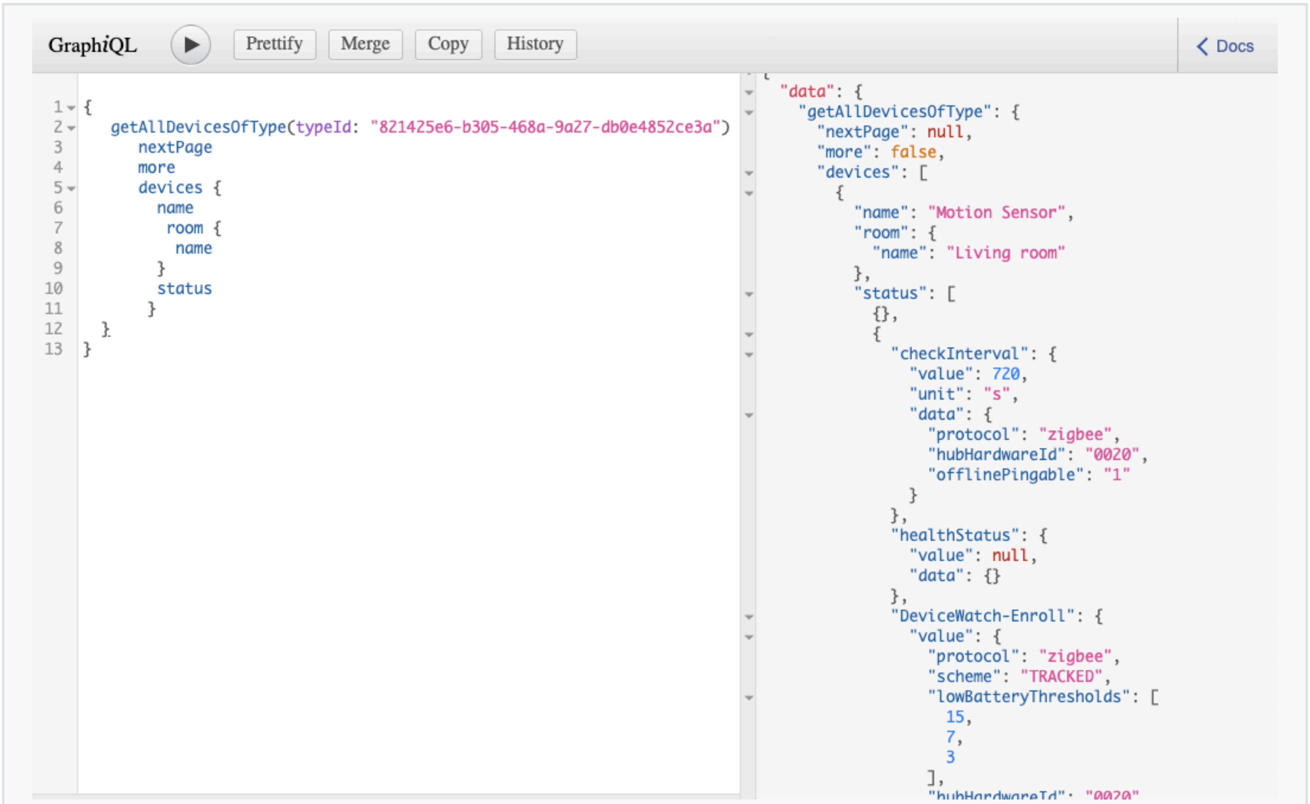


```
{
  getAllDevices(max: 5, page: 1) {
    more
    nextPage
    devices {
      deviceId
      name
      label
      deviceTypeId
      deviceTypeName
      deviceNetworkType
      room {
        name
      }
    }
  }
}
```

```
{
  "data": {
    "getAllDevices": {
      "more": true,
      "nextPage": "2",
      "devices": [
        {
          "deviceId": "214572e7-2ce3-4f77-ab40-79d27fd49627",
          "name": "Hue go 3 (Hue Extended Color)",
          "label": "Candle 3",
          "deviceTypeId": "437a5fc8-9d28-46cd-a093-
ced698498332",
          "deviceTypeName": "LAN Hue Extended Color",
          "deviceNetworkType": "UNKNOWN",
          "room": {
            "name": "Living room"
          }
        },
        {
          "deviceId": "275dc696-d836-49ed-a906-7776b0cad376",
          "name": "Mobile Presence made by Android",
          "label": "AI HOME's Galaxy Note9",
          "deviceTypeId": "8a9d4b1e3bfce38a013bfce42d360015",
          "deviceTypeName": "Mobile Presence",
          "deviceNetworkType": "UNKNOWN",
          "room": null
        },
        {
          "deviceId": "23394620-b5f0-4b36-b37b-e4175b80ce12",
          "name": "Motion Sensor",
          "label": "livingroom_motion_pil",
          "deviceTypeId": "821425e6-b305-468a-9a27-
db0e4852ce3a",
          "deviceTypeName": "SmartSense Motion Sensor",
          "deviceNetworkType": "ZIGBEE",
          "room": {
            "name": "Living room"
          }
        },
        {
          "deviceId": "2be831ee-d324-4a45-a371-dd11304018b8",
          "name": "Motion Sensor",
          "label": "door_motion_outside",
          "deviceTypeId": "821425e6-b305-468a-9a27-
db0e4852ce3a",
          "deviceTypeName": "SmartSense Motion Sensor",
          "deviceNetworkType": "ZIGBEE",
          "room": {
            "name": "Smart Door"
          }
        },
        {
          "deviceId": "2aee8438-d733-4e79-ab6d-b4a35cf7ed8a",
          "name": "Front",
          "label": "Front",
          "deviceTypeId": "94a16d1c-23b9-4f89-9857-
f86f7b927f9f",
          "deviceTypeName": "Ring Spotlight Cam Battery",
          "deviceNetworkType": "UNKNOWN",
          "room": {
            "name": "Front Yard"
          }
        }
      ]
    }
  }
}
```

Query All Devices with Paging

Device status provides access to the current state of all the Capabilities of the device.

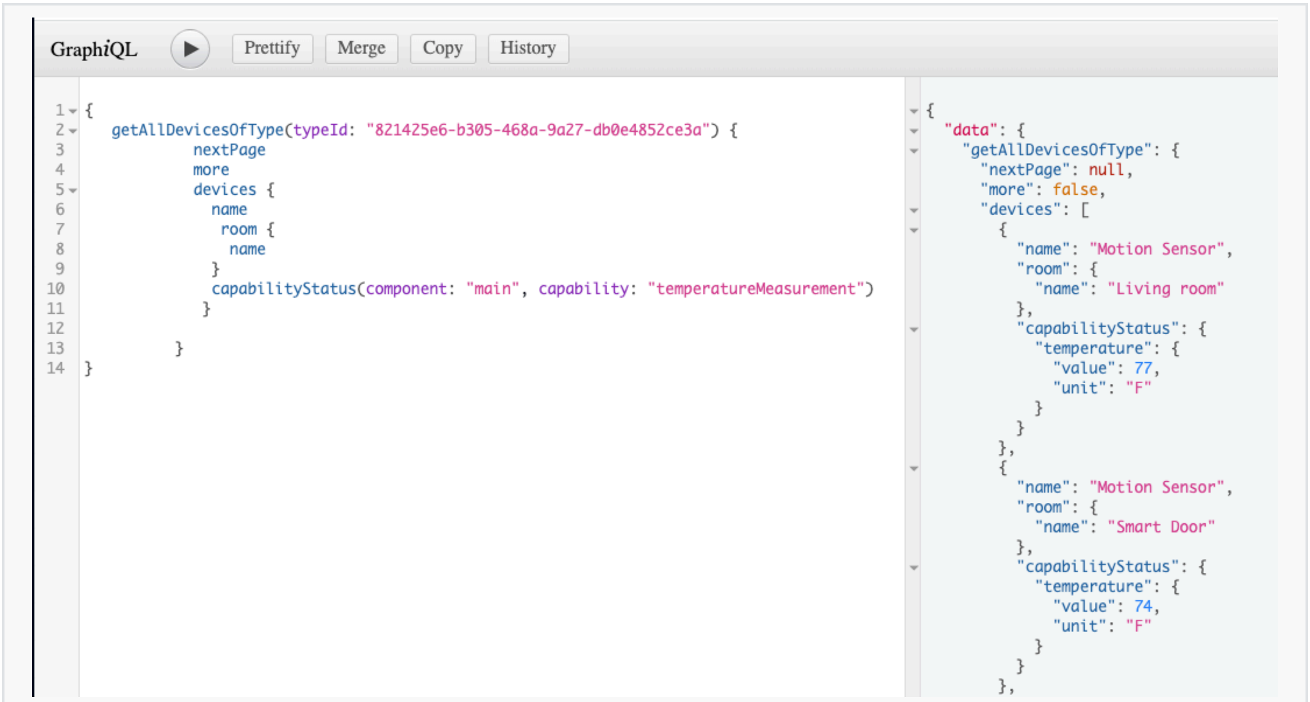


```
{
    getAllDevicesOfType(typeId: "821425e6-b305-468a-9a27-db0e4852ce3a")
        nextPage
        more
        devices {
          name
           room {
             name
          }
          status
        }
    }
}
```

```
  "data": {
    "getAllDevicesOfType": {
      "nextPage": null,
      "more": false,
      "devices": [
        {
          "name": "Motion Sensor",
          "room": {
            "name": "Living room"
          },
          "status": [
            {},
            {
              "checkInterval": {
                "value": 720,
                "unit": "s",
                "data": {
                  "protocol": "zigbee",
                  "hubHardwareId": "0020",
                  "offlinePingable": "1"
                }
              },
              "healthStatus": {
                "value": null,
                "data": {}
              },
              "DeviceWatch-Enroll": {
                "value": {
                  "protocol": "zigbee",
                  "scheme": "TRACKED",
                  "lowBatteryThresholds": [
                    15,
                    7,
                    3
                  ],
                  "hubHardwareId": "0020"
```

Get the status of all device capabilities

It is also possible to get the current status of a specific capability as shown below in getting the temperature of all the motion detectors.



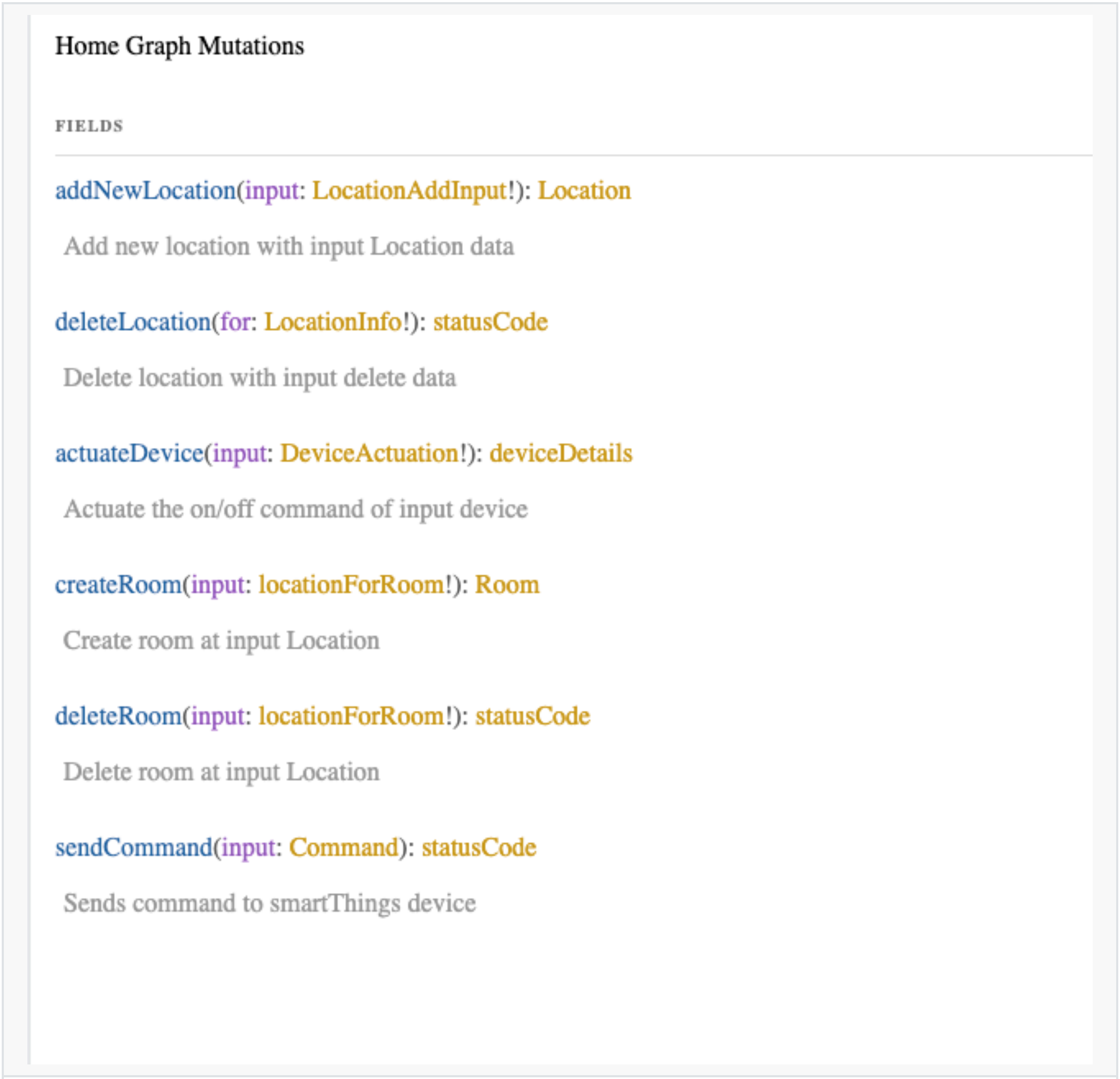
```
{
    getAllDevicesOfType(typeId: "821425e6-b305-468a-9a27-db0e4852ce3a") {
              nextPage
              more
              devices {
                name
                 room {
                   name
                }
                capabilityStatus(component: "main", capability: "temperatureMeasurement")
               }

           }
}
```

```
{
  "data": {
    "getAllDevicesOfType": {
      "nextPage": null,
      "more": false,
      "devices": [
        {
          "name": "Motion Sensor",
          "room": {
            "name": "Living room"
          },
          "capabilityStatus": {
            "temperature": {
              "value": 77,
              "unit": "F"
            }
          }
        },
        {
          "name": "Motion Sensor",
          "room": {
            "name": "Smart Door"
          },
          "capabilityStatus": {
            "temperature": {
              "value": 74,
              "unit": "F"
            }
          }
        },
```

Get the status of single device capability

Home Graph Mutations

In Graphql Mutations unlike queries change state, HomeGraph exposes the following Mutations.



Home Graph Mutations

FIELDS

```
addNewLocation(input: LocationAddInput!): Location
```

Add new location with input Location data

```
deleteLocation(for: LocationInfo!): statusCode
```

Delete location with input delete data

```
actuateDevice(input: DeviceActuation!): deviceDetails
```

Actuate the on/off command of input device

```
createRoom(input: locationForRoom!): Room
```

Create room at input Location

```
deleteRoom(input: locationForRoom!): statusCode
```

Delete room at input Location

```
sendCommand(input: Command): statusCode
```

Sends command to smartThings device

List of Mutations

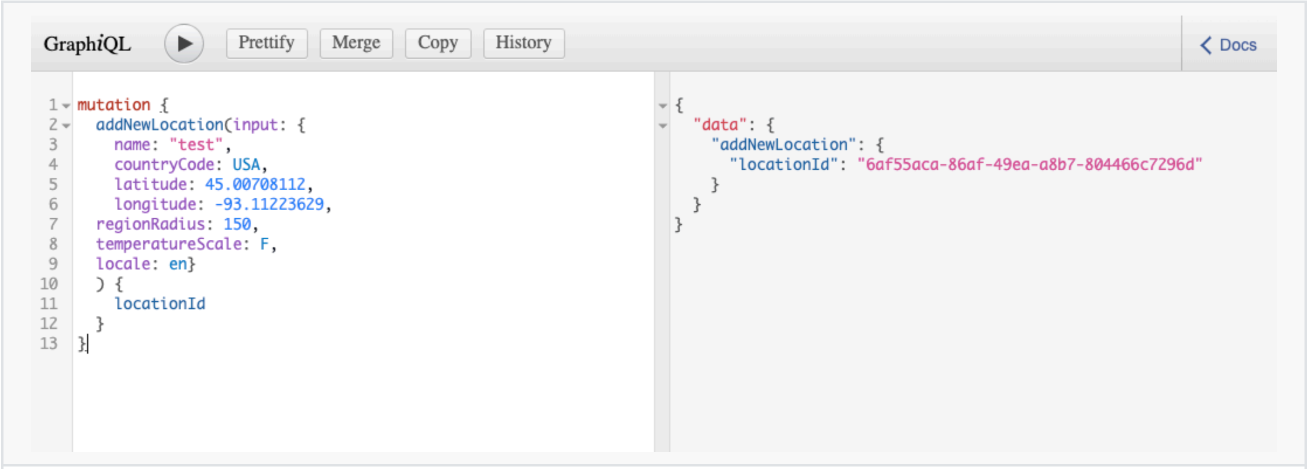
Add New Location

The `addNewLocation` mutation is used to create a new Location within a SmartThings account. Locations represent a Geolocation therefore the latitude and longitude of the location must be supplied along with a country code and name all other inputs are optional the full list of input arguments to this mutation are

- name: Is the human readable name of the newly created location

- countryCode: Schema type that represents a ISO Alpha 3 country
- latitude: Geospatial Latitude of the Location
- longitude: Geospatial Longitude of the Location
- regionRadius: The radius in meters around latitude and longitude which defines this location.
- temperatureScale: The desired temperature scale used within location. Value can be F or C.
- locale: Schema type value for an IETF BCP 47 language tag representing the chosen locale for this location.

This mutation returns a new Location type with unique `locationId` or null if an error occurs



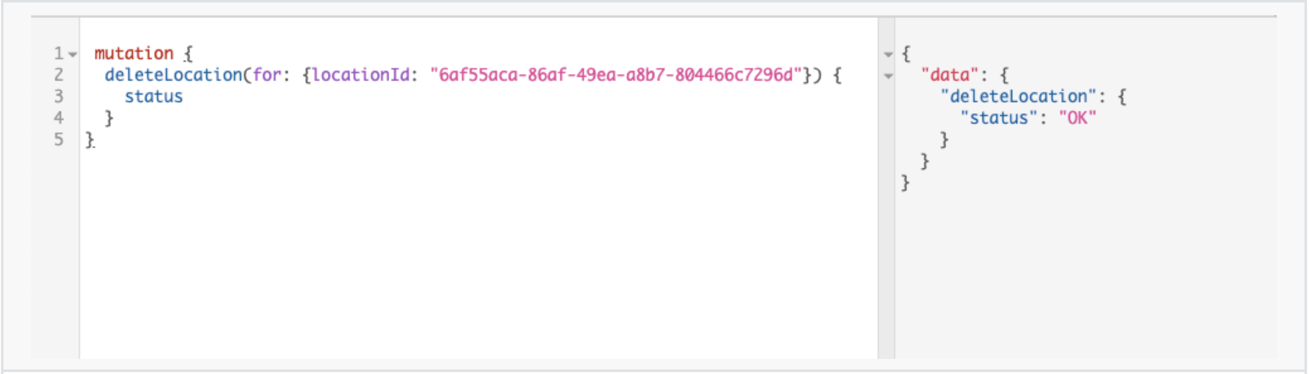
```
mutation {
  addNewLocation(input: {
    name: "test",
    countryCode: USA,
    latitude: 45.00708112,
    longitude: -93.11223629,
  regionRadius: 150,
  temperatureScale: F,
  locale: en}
  ) {
    locationId
  }
}
```

```
{
  "data": {
    "addNewLocation": {
      "locationId": "6af55aca-86af-49ea-a8b7-804466c7296d"
    }
  }
}
```

Creating a new location named Test

Delete Location

The `deleteLocation` mutation will delete an existing location along with all its room and detach any devices linked to that location. Devices are not deleted they are simply removed from the location. This mutation can use either Location Name or locationId to determine which location to delete. Since it is possible (though not advised) that two Locations associated with a SmartThings account may have the same name it is safer to use LocationId. The deleteLocation mutation returns a mutation result Status indication



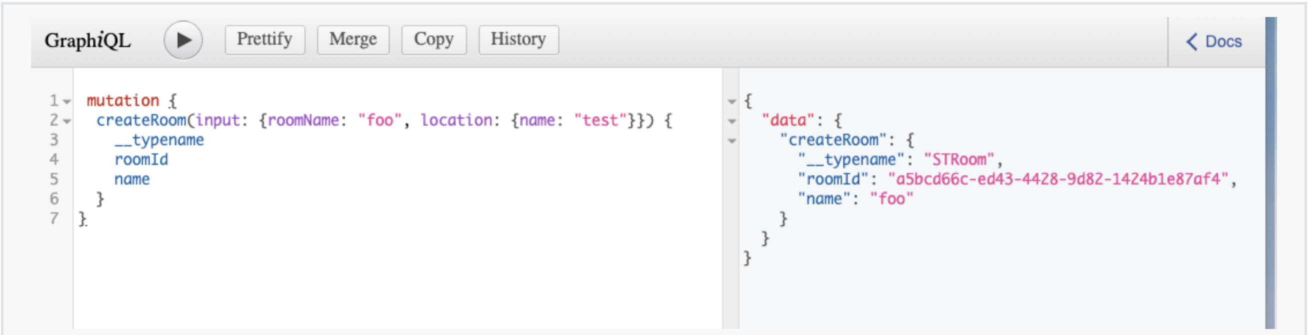
```
mutation {
  deleteLocation(for: {locationId: "6af55aca-86af-49ea-a8b7-804466c7296d"}) {
    status
  }
}
```

```
{
  "data": {
    "deleteLocation": {
      "status": "OK"
    }
  }
}
```

Deleting location using the LocationId

Create room

The `createRoom` mutation will create a room within a location. A Room in HomeGraph is a abstract type and currently the type of a Room is derived from the Room name. To create a room the mutation needs a Room name and either a location name and or locationId. The createRoom mutation returns a Room object.



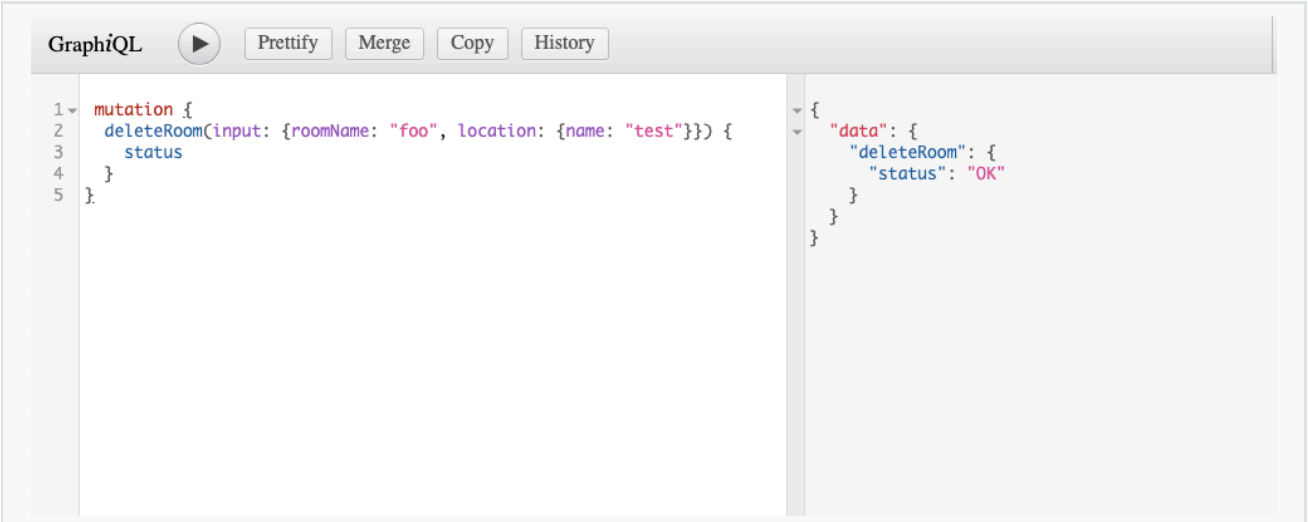
```
mutation {
  createRoom(input: {roomName: "foo", location: {name: "test"}}) {
    __typename
    roomId
    name
  }
}
```

```
{
  "data": {
    "createRoom": {
      "__typename": "STRoom",
      "roomId": "a5bcd66c-ed43-4428-9d82-1424b1e87af4",
      "name": "foo"
    }
  }
}
```

Creating a room called foo in test location

Delete room

The `deleteRoom` mutation will delete a room within a Location. To detele a room the mutation needs a Room name and either a location name and or locationId. The deleteRoom mutation mutation result Status indication.



```
mutation {
  deleteRoom(input: {roomName: "foo", location: {name: "test"}}) {
    status
  }
}
```

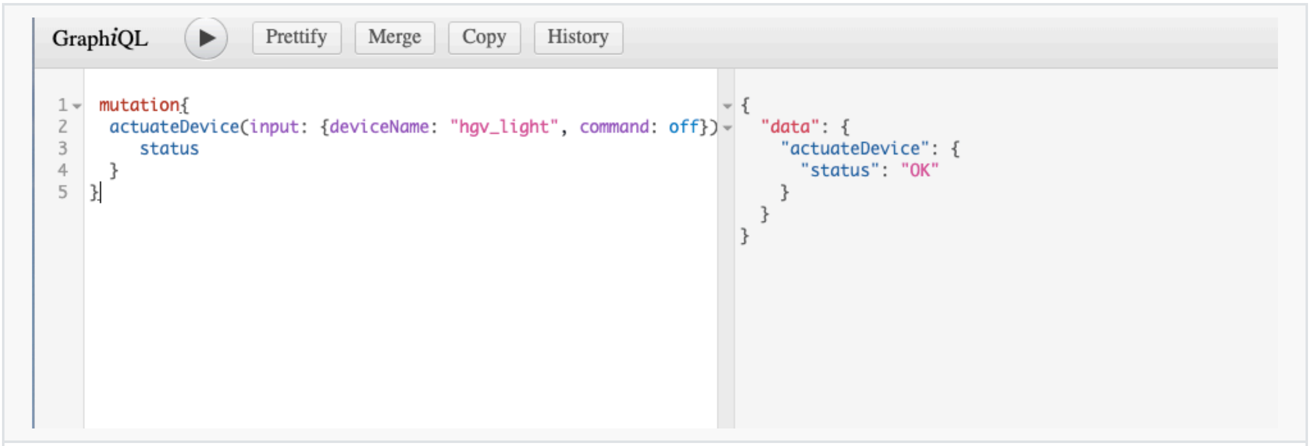
```
{
  "data": {
    "deleteRoom": {
      "status": "OK"
    }
  }
}
```

Deleting a room called foo in test location

Device commands

There are two mutations that deal with controlling devices `actuateDevice` and `sendCommand`. The former can be used for on off actuations of a device it was singled out because its the most common control on a device. The latter can be used to send any command offered by a device as specified here

The `actuateDevice` mutation accepts a device name and the the command is `on` or `off` as shown below



```
mutation{
 actuateDevice(input: {deviceName: "hgv_light", command: off}) {
    status
  }
}
```

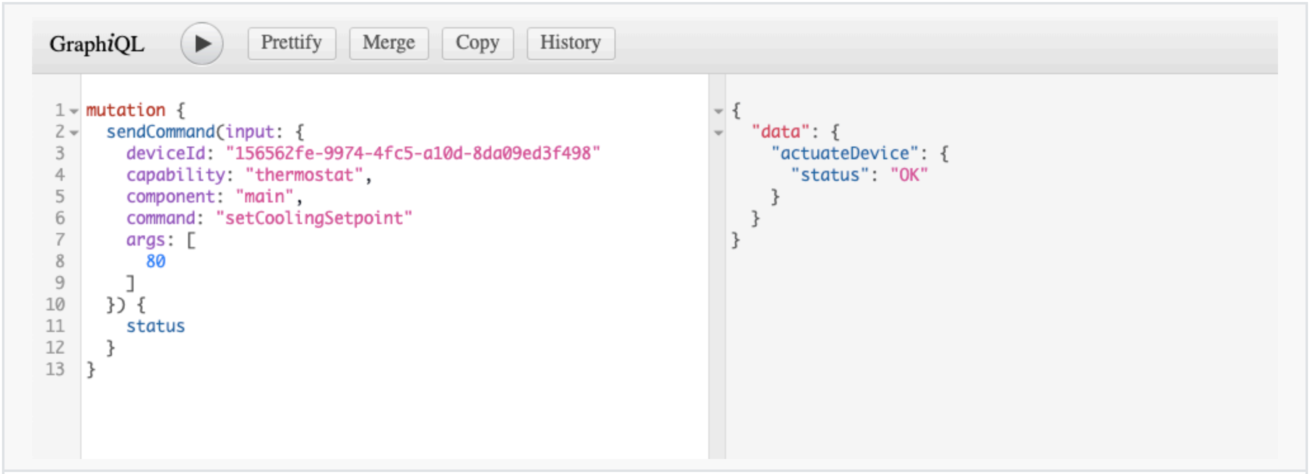
```
{
  "data": {
    "actuateDevice": {
      "status": "OK"
    }
  }
}
```

Use of actuateDevice Mutation turning off a device

The `sendCommand` mutation provides far more flexibity has been implemented to mirror the functionality of the SmartThings sendCommand so the input params are much the same as detailed below:

- deviceId : The unique ID of the device to which the command should be sent.
- capability : The capability to the command belongs as specified here
- component : The component name within the device.
- command : The name of the command to send
- args : a mixed type array of arguments to be sent with the command

Send Command returns a mutation result Status indication. An example of sending a `setCoolingSetpoint` command to a thermostat is shown below as an example.



```
mutation {
  sendCommand(input: {
    deviceId: "156562fe-9974-4fc5-a10d-8da09ed3f498"
    capability: "thermostat",
    component: "main",
    command: "setCoolingSetpoint"
    args: [
      80
    ]
  }) {
    status
  }
}
```

```
{
  "data": {
    "actuateDevice": {
      "status": "OK"
    }
  }
}
```

Sending the setCoolingSetpoint to thermostat

Deployment Options

There are two main deployment models supported by HomeGraph. These are explained below. There is some general configuration which is required for all modes. This is done via a `.env` file with the following

```
SMARTTHINGS_TOKEN=SmartThings API token, https://account.smartthings.com/tokens
FINHUB_KEY= API Key for https://finnhub.io/ used only for Stock service
CLIMACELL_API=API key for https://www.climacell.co/weather-api/pricing/ used for
airquality information
```

Serverless

The default deployment method is to deploy to AWS as a Lambda service. This is a very cost effective method since it has a generous free tier and when charged the billing is done on use.

To deploy to AWS ensure that the serverless environment has been fullyconfigured

Once configuration is completed the HomeGraph lambda can be deployed with `serverless deploy`

Node Library

HomeGraph used as a Node package (**Needs to be deployed to package service**) which can then be included into other projects to enable extensions through graphql federation and or schema stitching. This provides any number of integrations with other services or extension of Room types.

As a library HomeGraph can be repurposed as a standalone service for development purposes. The following segment demonstates configuration as an Express service

```
require('dotenv').config();
const path = require('path');
const fs = require('fs');
const express = require('express');
const graphqlHTTP = require('express-graphql');
const { makeExecutableSchema } = require('graphql-tools');
const bodyParser = require('body-parser');
const resolvers = require('../../lib/graphutils/resolvers');
const schema = require('HomeGraph');

const app = express();
app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

app.get('/', async (req, res) => {
  res.send(`Hello World!`);
});

app.use(
  '/graphql',
  graphqlHTTP({
    schema,
    graphiql: true,
  })
);

app.listen(4000);
```

Schema Types

▶ **Table of Contents**

Query

Home Graph Queries

Field	Argument	Type	Description
getLocations		[Location!]!	

Get all the Locations

getLocation
Location

Get Specific Location Information

for
LocationInfo!

getRoom
Room

Get data for a specific room in a location

for
RoomInfo!

getAllDevices
DeviceList!

Gets all devices across all locations

page
Int

max
Int

getAllDevicesOfType
DeviceList!

Gets all devices of a certain type

typeId
String!

getWeather
Weather

Get the weather at a Location

location
LocationInfo!

getAirQuality
AirQuality

Get the airQuality at a Location

location
LocationInfo!

getStockPrice
Stock

Get the stock price of underlying ticker

input
Symbol!

Mutation

Home Graph Mutations

Field	Argument	Type	Description
addNewLocation		Location	

Add new location with input Location data

input
LocationAddInput!

deleteLocation
statusCode

Delete location with input delete data

for
LocationInfo!

actuateDevice
statusCode

Actuate the on/off command of input device

input
DeviceActuation!

createRoom
Room

Create room at input Location

input
locationForRoom!

deleteRoom
statusCode

Delete room at input Location

input
locationForRoom!

sendCommand
statusCode

Sends command to smartThings device

input
Command

Objects

AirQuality

Air quality type based on Breezometer

Field	Argument	Type	Description
time		String	

Timestamp for when air quality reading was taken"

epaAqi
Float

Air Quality Index per US EPA

epaPrimaryPollutant
String

Primary Pollutant per US EPA

pm25
Float

Particulate Matter < 2.5 μm

pm10
Float

Particulate Matter < 10 μm

no2
Float

Nitrogen Dioxide

co
Float

Carbon Monoxide

so2
Float

Sulfur Dioxide

epaHealthConcern
String

Health concern level based on EPA standard

pollenTree
Int

ClimaCell pollen index for Trees

pollenWeed
Int

ClimaCell pollen index for Weed

pollenGrass
Int

ClimaCell pollen index for Grass

Capability

Type for single Capability

Field	Argument	Type	Description
id		String	

User readible label for capability

version
Int

Version number

Component

A device may have multiple sub-systems and this is represented by a Component type

Field	Argument	Type	Description
id		String	

Unique ID of the component

label
String

User readible label for component e.g main

capabilities
[Capability!]!

All the capabilities supported by the component

Device

Representation of a device

Field	Argument	Type	Description
deviceId		String	

Unique identifier for a device

name
String

User friendly name of a device

label
String

User friendly name of a device

location
Location

The location a device belongs too

room
Room

The room a device is in

deviceTypeId
String

Unique type ID for device

deviceTypeName
String

User friendly name of device type

deviceNetworkType
String

The type used by the device to connect

components
[Component!]!

Retrieves components with schema

capabilityStatus
Anything

Generic Conponents status

component
String!

capability
String!

status
[Anything!]

The current status of capabilities

capability
String

DeviceList

Paginated List of Devices

Field	Argument	Type	Description
devices		[Device!]!	

Page list of Devices

more
Boolean

Indication if there are more devices

nextPage
String

Next Page number

Location

A Location is a Home or Geographic area

Field	Argument	Type	Description
name		String	

The human readable name of a location

locationId
String

A unique ID that represents a Location

countryCode
String

The international country code for a Location

latitude
Float

The latitude for the location

longitude
Float

The longitude for the location

regionRadius
Int

This is the radius in meters around the latitude and longitude which defines this Location

temperatureScale
String

This indicates if temperatures should be listed in Centigrade or Fahrenheit

timeZoneId
String

The international timezone label

locale
String

The international two character locale label

rooms
[Room!]!

Gets all the rooms within a location

weather
Weather

Gets the weather at the location

airQuality
AirQuality

Gets the airquality at the location

fireHazard
Float

Fire hazard Index

Office

Field	Argument	Type	Description
roomId		String!	

Unique Room id

locationId
String!

Location ID which the Room is in

name
String!

User friendly name of the room

devices
[Device!]!

The devices in a room

stock
Stock

Gets the underlying stock price

ticker

String!

STRoom

Field	Argument	Type	Description
roomId		String!	

Unique Space id"

locationId
String!

Location ID which the space is in

name
String!

User friendly name of the room

devices
[Device!]!

The devices in a room

Stock

Underlying Finincial Stock object

Field	Argument	Type	Description
currentPrice		String	

Current price

highPrice
String

High price of the day

lowPrice
String

Low price of the day

openingPrice
String

Open price of the day

previousClose
String

Previous close price

time
String

Timestamp

symbol
String

Stock symbol

Weather

The weather type based on openweather

Field	Argument	Type	Description
timestamp		Int	

Timestamp of weather reading

location
String

Location name

condition
Int

Symbol indicated weather icon to use

description
String

Text description of the weather

temperature
Float

Temperature in the unit scale defined by Location type

pressure
Float

The pressure at the location

humidity
Float

The humidity information

wind_speed
Float

The current windspeed

wind_direction
Int

The current wind direction

cloud_cover
Float

The percentage of cloud cove

rain_volume
Float

The current rain volume if any

snow_volume
Float

The current rain volume if any

deviceDetails

Device details

Field	Argument	Type	Description
deviceName		String	

deviceName

command
String

command

statusCode

General status reponse

Field	Argument	Type	Description
status		String	

Status code

Inputs

Command

Input type to send a command to a device

Field	Type	Description
deviceId	String!	

device Id

component
String!

component name (typically "main")

capability
String!

Capability nae

command
String!

Command

args
[Anything!]

Array of Args

DeviceActuation

Device Actuation type

Field	Type	Description
deviceName	String	

Device name

command
onOff

Actuation command

LocationAddInput

Input to add a location

Field	Type	Description
name	String!	

Name of new Location

countryCode
CountryCode!

Country Code

latitude
Float!

Latitude of Location

longitude
Float!

Longitude of Location

regionRadius
Int

regionRadius

temperatureScale
TemperatureScale

Temperature scale unit

locale
Locale

Location

LocationDeleteInput

Location delete Input

Field	Type	Description
locationId	String	

locationId

LocationInfo

LocationInfo

Field	Type	Description
locationId	String	

LocationId

name
String

Location name

Page

Page

Field	Type	Description
number	Int	

Page Number

max
Int

Max items in Page

RoomInfo

Room info Input

Field	Type	Description
location	LocationInfo!	

Location ID

roomId
String

Room Id

name
String

Room name

Symbol

Symbol

Field	Type	Description
ticker	String	

Ticker

locationForRoom

Input to specify a room

Field	Type	Description
roomName	String!	

Name of room

location
LocationInfo!

Location

Enums

CountryCode

Country code

Value	Description
ABW	
AFG	
AGO	
AIA	
ALA	
ALB	
AND	
ANT	
ARE	
ARG	
ARM	
ASM	
ATA	
ATF	

ATG	
AUS	
AUT	
AZE	
BDI	
BEL	
BEN	
BFA	
BGD	
BGR	
BHR	
BHS	
BIH	
BLM	
BLR	
BLZ	
BMU	
BOL	
BRA	
BRB	
BRN	
BTN	
BVT	
BWA	
CAF	
CAN	
CCK	

CHE	
CHL	
CHN	
CIV	
CMR	
COD	
COG	
COK	
COL	
COM	
CPV	
CRI	
CUB	
CXR	
CYM	
CYP	
CZE	
DEU	
DJI	
DMA	
DNK	
DOM	
DZA	
ECU	
EGY	
ERI	

ESH	
ESP	
EST	
ETH	
FIN	
FJI	
FLK	
FRA	
FRO	
FSM	
GAB	
GBR	
GEO	
GGY	
GHA	
GIB	
GIN	
GLP	
GMB	
GNB	
GNQ	
GRC	
GRD	
GRL	
GTM	
GUF	
GUM	

GUY	
HKG	
HMD	
HND	
HRV	
HTI	
HUN	
IDN	
IMN	
IND	
IOT	
IRL	
IRN	
IRQ	
ISL	
ISR	
ITA	
JAM	
JEY	
JOR	
JPN	
KAZ	
KEN	
KGZ	
KHM	
KIR	

KNA	
KOR	
KWT	
LAO	
LBN	
LBR	
LBY	
LCA	
LIE	
LKA	
LSO	
LTU	
LUX	
LVA	
MAC	
MAF	
MAR	
MCO	
MDA	
MDG	
MDV	
MEX	
MHL	
MKD	
MLI	
MLT	

MMR	
MNE	
MNG	
MNP	
MOZ	
MRT	
MSR	
MTQ	
MUS	
MWI	
MYS	
MYT	
NAM	
NCL	
NER	
NFK	
NGA	
NIC	
NIU	
NLD	
NOR	
NPL	
NRU	
NZL	
OMN	
PAK	
PAN	

PCN	
PER	
PHL	
PLW	
PNG	
POL	
PRI	
PRK	
PRT	
PRY	
PSE	
PYF	
QAT	
REU	
ROU	
RUS	
RWA	
SAU	
SDN	
SEN	
SGP	
SGS	
SHN	
SJM	
SLB	
SLE	

SLV	
SMR	
SOM	
SPM	
SRB	
STP	
SUR	
SVK	
SVN	
SWE	
SWZ	
SYC	
SYR	
TCA	
TCD	
TGO	
THA	
TJK	
TKL	
TKM	
TLS	
TON	
TTO	
TUN	
TUR	
TUV	
TWN	

TZA	
UGA	
UKR	
UMI	
URY	
USA	
UZB	
VAT	
VCT	
VEN	
VGB	
VIR	
VNM	
VUT	
WLF	
WSM	
YEM	
ZAF	
ZMB	
ZWE	

Locale

Locale two code

Value	Description
aa	
ab	
ae	

af	
ak	
am	
an	
ar	
as	
av	
ay	
az	
ba	
be	
bg	
bh	
bm	
bi	
bn	
bo	
br	
bs	
ca	
ce	
ch	
co	
cr	
cs	
cu	

cv	
cy	
da	
de	
dv	
dz	
ee	
el	
en	
eo	
es	
et	
eu	
fa	
ff	
fi	
fj	
fo	
fr	
fy	
ga	
gd	
gl	
gn	
gu	
gv	
ha	

he	
hi	
ho	
hr	
ht	
hu	
hy	
hz	
ia	
id	
ie	
ig	
ii	
ik	
io	
is	
it	
iu	
ja	
jv	
ka	
kg	
ki	
kj	
kk	
kl	

km	
kn	
ko	
kr	
ks	
ku	
kv	
kw	
ky	
la	
lb	
lg	
li	
ln	
lo	
lt	
lu	
lv	
mg	
mh	
mi	
mk	
ml	
mn	
mr	
ms	
mt	

my	
na	
nb	
nd	
ne	
ng	
nl	
nn	
no	
nr	
nv	
ny	
oc	
oj	
om	
or	
os	
pa	
pi	
pl	
ps	
pt	
qu	
rm	
rn	
ro	

ru	
rw	
sa	
sc	
sd	
se	
sg	
si	
sk	
sl	
sm	
sn	
so	
sq	
sr	
ss	
st	
su	
sv	
sw	
ta	
te	
tg	
th	
ti	
tk	
tl	

tn	
to	
tr	
ts	
tt	
tw	
ty	
ug	
uk	
ur	
uz	
ve	
vi	
vo	
wa	
wo	
xh	
yi	
yo	
za	
zh	
zu	

TemperatureScale

Temperature Scale

Value	Description
F	
C	

onOff

Actuation Value

Value	Description
on	
off	

Scalars

Anything

Any value.

Boolean

The `Boolean` scalar type represents `true` or `false`.

Float

The `Float` scalar type represents signed double-precision fractional values as specified by IEEE 754.

Int

The `Int` scalar type represents non-fractional signed whole numeric values. Int can represent values between -(2^31) and 2^31 - 1.

String

The `String` scalar type represents textual data, represented as UTF-8 character sequences. The String type is most often used by GraphQL to represent free-form human-readable text.

Interfaces

Room

The Room type

Field	Argument	Type	Description
roomId		String!	

Unique room id

locationId
String!

Location ID which the space is in

name
String!

User friendly name of the room

devices
[Device!]!

The devices in a room