**IoT Example APpSync communication protocol document**

**Change log**

| Document version | Author | Changes |
| --- | --- | --- |
| 1.0 | MN | Initial version |
|  |  |  |

[**Protocol requirements**](#_rf8ijoe22p21) **3**

[**Connectivity Details**](#_xmcn8eogmxiv) **3**

[**Sequence of interaction**](#_n62l57iggrla) **3**

[Registration](#_v6zje12nqrdi) 3

[Login](#_brmk8foeiu1j) 3

[Onboard an IoT device](#_qmlnvjytk9ib) 4

[Query data](#_hpt9lqk40cw0) 4

[Offboarding](#_iqckvwg849ah) 4

[**Queries & Mutations**](#_fymxebchc2x4) **4**

[Onboarding](#_dfzbujy2xz40) 4

[Request](#_ez7gbdjq63yf) 4

[Response](#_dsi0cvdz9uat) 4

[Offboarding](#_lhdet54vwfyj) 5

[Request](#_e7vfl9bx8vy0) 5

[Response](#_s09d60cab82z) 5

[**Get my device ID**](#_20x28rgghcnr) **5**

[Request](#_f3mcian4phjk) 5

[Response](#_fjtqtwam94qm) 6

[**Get one hour data**](#_n3hjk0m4um4r) **6**

[Request](#_5lzjs5my9w9z) 6

[Response](#_rs4ezrqqnecd) 6

[**Get 2 week data**](#_f18nwjpyhdzu) **7**

[Request](#_2hibttibmz0t) 7

[Response](#_swvhna8ko9lt) 8

[**GraphQL Schema**](#_g2g90oc3t0gf) **8**

## 

## Protocol requirements

The API is configured using AWS AppSync with AWS cognito for user management and authentication.

Clients must follow the authentication process in order to communicate with the cloud and interact with customer data.

https://docs.aws.amazon.com/appsync/latest/devguide/security-authz.html#amazon-cognito-user-pools-authorization

## Connectivity Details

In order for a client app to be developed the following details will need to be included in the app.

| **name** | **value** | **description** |
| --- | --- | --- |
| APIEndpoint | https://xxxxxxxxxxxxxxxxxxxxxxxxxx.appsync-api.us-east-1.amazonaws.com/graphql | URL for app to connect to when interfacing with AppSync |
| CognitoIdentityPoolId | us-east-1:xxxxxxxx-xxx-xxxx-xxxx-xxxxxxxxxxxx | ID number of the cognito identify pool |
| CognitoUserPoolClientId | xxxxxxxxxxxxxxxxxxxxxxxxxx | ID number of the cognito client user pool |
| CognitoUserPoolId | us-east-1\_XXXXXXXXX | ID number of the cognito user pool |

## Sequence of interaction

### Registration

A user is first required to register an account.

The registration process asks for the following details:

* Email address
* Password

An email will be sent to the address entered by the user with a link, the link must be clicked in order to validate the email address is owned by the user.

### Login

Once an account is active the user will need to login with their email address and password.

### Onboard an IoT device

A user is required to onboard a device using the device’s unique ID. This value must correspond to the device ID used by the IoT device to send data to the cloud.

### Query data

After login and periodically afterwards the client app is required to request for new data. There are 2 data sets available to the client:

* the last hour’s worth of data at a one minute interval
* the last 2 week’s worth of data at a one hour interval

### Offboarding

Should the user no longer be responsible for monitoring an IoT device the device can be removed from the users account by deleting the device.

## Queries & Mutations

### Onboarding

The createUserDevice mutation shall be called with the device ID as the parameter to the function

#### Request

| mutation createDevice {  createUserDevice(deviceID: "AAAAAAAAAAAA") {  userID  } } |
| --- |

#### Response

| {  "data": {  "createUserDevice": {  "userID": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"  }  } } |
| --- |

| **key** | **Data type** | **units** | **description** |
| --- | --- | --- | --- |
| userID | string | N/A | The unique ID of this users account |

### Offboarding

The deleteUserDevice mutation shall be called when removing the mapping between the user account and an IoT device

#### Request

| mutation deleteDevice {  deleteUserDevice(deviceID: "AAAAAAAAAAAA") {  userID  } } |
| --- |

#### Response

| {  "data": {  "deleteUserDevice": {  "userID": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",  "deviceID": "AAAAAAAAAAAA"  }  } } |
| --- |

If this is called and no device map exists, deleteUserDevice will return null.

| **key** | **Data type** | **units** | **description** |
| --- | --- | --- | --- |
| deviceID | string | N/A | The unique ID of the IoT device that is associated with this used account |
| userID | string | N/A | The unique ID of this users account |

## Get my device ID

Return the device ID that is associated with this used account. The device ID is not required to get data from the device.

#### Request

| query getUserDevices {  getUserDevices {  deviceID  userID  } } |
| --- |

#### Response

| {  "data": {  "getUserDevices": {  "deviceID": "AAAAAAAAAAAA",  "userID": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"  }  } } |
| --- |

| **key** | **Data type** | **units** | **description** |
| --- | --- | --- | --- |
| deviceID | string | N/A | The unique ID of the IoT device that is associated with this used account |
| userID | string | N/A | The unique ID of this users account |

## Get one hour data

Get the data that was recorded by the IoT device for the last hour.

#### Request

| query getSensorData {  getSensorData {  items {  temp  hum  timestamp  }  } } |
| --- |

#### Response

| {  "data": {  "getSensorData": {  "items": [  {  "temp": 25,  "hum": 87,  "timestamp": 1656406833  },  {  "temp": 25,  "hum": 87,  "timestamp": 1656406794  },  {  "temp": 25,  "hum": 87,  "timestamp": 1656405233  },  {  "temp": 22,  "hum": 86,  "timestamp": 1656404615  },  {  "temp": 21,  "hum": 85,  "timestamp": 1656404555  }  ]  }  } } |
| --- |

| **key** | **Data type** | **units** | **description** |
| --- | --- | --- | --- |
| timestamp | int | Seconds since epoch | The timestamp when the data point was transmitted to the cloud |
| temp | float | celsius | The average temperature recorded by the IoT device since the last transmit |
| hum | int | Relative humidity percent | The average relative humidity recorded by the IoT device since the last transmit |

## Get 2 week data

Get the data that was summarised by cloud into one hour data points from data that was reported by the IoT device.

#### Request

| query getTwoWeekData {  getTwoWeekSummary {  items {  temp  hum  timestamp  }  } } |
| --- |

#### Response

| {  "data": {  "getTwoWeekSummary": {  "items": [  {  "temp": 23.25,  "hum": 86,  "timestamp": 1656406800  }  ]  }  } } |
| --- |

| **key** | **Data type** | **units** | **description** |
| --- | --- | --- | --- |
| timestamp | int | Seconds since epoch | The timestamp when the data point was transmitted to the cloud |
| temp | float | celsius | The average temperature recorded by the IoT device since the last transmit |
| hum | int | Relative humidity percent | The average relative humidity recorded by the IoT device since the last transmit |

## GraphQL Schema

| schema {  query: Query  mutation: Mutation }  type Mutation {  createUserDevice(deviceID: String!): UserDevice @aws\_auth(cognito\_groups : ["UserGroup"])  deleteUserDevice: UserDevice @aws\_auth(cognito\_groups : ["UserGroup"]) }  type Query {  getSensorData: DeviceDataConnection @aws\_auth(cognito\_groups : ["UserGroup"])  getTwoWeekSummary: TwoWeekDataConnection @aws\_auth(cognito\_groups : ["UserGroup"])  getUserDevices: UserDevice @aws\_auth(cognito\_groups : ["UserGroup"]) }  type DeviceDataConnection {  items: [DeviceDataPoint]  nextToken: String }  type DeviceDataPoint {  hum: Int!  temp: Float!  timestamp: AWSTimestamp! }  type TwoWeekDataConnection {  items: [TwoWeekDataPoint]  nextToken: String }  type TwoWeekDataPoint {  hum: Int!  temp: Float!  timestamp: AWSTimestamp! }  type UserDevice {  deviceID: String!  error\_msg: String  userID: String } |
| --- |

### 