AT&T / 2lemetry API V2 Usage Guide

[Updated: 9/10/2013 | Document Version: 2.5]

This article details how to retrieve and monitor data published by devices on 2lemetry's Device Cloud Platform (m2m.io) using API calls. Primarily, it discusses "Get" calls, which are those commands to retrieve data from specific devices. It also covers actions such as authentication, publishing, rule setting, data storage, key/value pairings, account management, and weather variables.

For example, let's say you have a devices that measures temperature. You can request easily retrieve past data about your devices ("things") by using that calls located under "Get Past about a Thing". Here, you have options to set parameters for the data you would like to retrieve: attributes, start times, end times, limits, and format.

Base API URL

The base URL for the m2m.io API, version 2 is:

```
http://att-api.m2m.io
```

Interactive API Documentation

Fully functional API documentation can be found at http://2lemetry.com/api

Notes

- Parameters containing [op] below are optional parameters
- When parameters are described in larger paragraphs, they will be specified with *italics*

- Example titles for certain parameters will be delineated by a preceding "my." For example: "mydomain", "mystuff", etc.

/auth

Operation allows generation of an API authentication token. Tokens have a 3 hour expiry. Tokens are used in a HTTPS header with the value Bearer [token].

Get Authentication Token

Retrieve a token.

Operation, URL

```
GET /2/auth
```

Parameters

none

Example

Return an authentication token.

```
http://att-api.m2m.io/2/auth
```

Response

```
{
  "token": "vzT2nGFUejXCJiK4zPF8w7DjsA61Fis7eFq3h1quOSKw"
}
```

/publish

Use the m2m.io API to publish data to the MQTT broker. The normal use case for publishing MQTT data is to use an MQTT client. This API endpoint can be used in place of a client when integrating a dedicated client would prove difficult. A possible example would be publishing infrequent messages from a web application.

Publish MQTT Data

Operation, URL

```
POST /2/account/domain/{domain}/stuff/{stuff}/thing/{thing}/publish
```

Parameters

- domain
- stuff
- thing
- topic [op]

If the *topic* parameter is left out, publish will occur on the topic: *domain/stuff/thing*. If *topic* is specified, domain must contain user's *domain* but *stuff* and *thing* parameters are not used.

Examples

Send {"message":"hello"} on com.example/mystuff/mything

```
http://att-
api.m2m.io/2/account/domain/com.example/stuff/mystuff/thing/mything/pub
lish?payload=%7B%22message%22%3A%22hello%22%7D
```

Response

```
{"topic" : "com.example/mystuff/mything", "payload" :  "
{"message":"hello"}"}
```

Send {"message":"hello"} on mytopic/mysubtopic

```
http://att-
api.m2m.io/2/account/domain/com.example/stuff/mystuff/thing/mything/pub
lish?
topic=mytopic%2Fmysubtopic&payload=%7B%22message%22%3A%22hello%22%7D
```

Response

```
{"topic" : "mytopic/mysubtopic", "payload" :  "{"message":"hello"}"}
```

/things

The *things* API calls provide access to the data published to the m2m.io platform via MQTT or by the publish API call. Data published to the platform on a three-level topic x/y/z will be stored and can be retrieved via the things API calls.

Domains are the highest level of organization on the platform. A *domain* would

contain all devices and data from one company or one higher-level project or application domain.

Typical domain examples:

- Single company monitoring humidity in walk-in freezers
- Organization monitors rainfall and controls industrial HVAC equipment. These separate solutions may better fit into more than one *domain*.

Multiple user accounts can be allocated under one *domain*, giving access to the data to multiple users. Rules can also be applied across all data in a *domain*.

stuff is the second layer of organization. Individual devices are at the third level. The *stuff* gives the capability to better organize those devices into a hierarchy rather than one flat "bucket" of devices.

Typical stuff examples:

- The walk-in freezer company (mentioned above) may have 10 freezers to monitor in every building of a manufacturing facility. They could use stuff to organize devices into their respective buildings.
- A trucking company tracks their in-city delivery vehicles in multiple cities. In this example a truck would be a device. The *stuff* level could be used to group their trucks by city or region.

things are the third topic level. Typically in a sensor data or command & control application this level would be a device. Think of the walk-in freezer example (see the above). The *thing* level would be each individual freezer monitor.

To complete the example, the *topic* on which devices would publish, and therefore the topic space used in a *things* API call, would be:

```
com.freezercompany/building4/freezer1
```

The *domain* is typically assigned to you when you create an account on the m2m.io platform. Stuff and things are completely up to you to define depending on your application and architecture.

Retrieving Data

There are two concepts of retrieving previously posted data from the platform: present and past.

A present call is used to retrieve the last published, or present, value. A present call could be used to build an application showing the current temperature for one of the sensors.

A past call is used to grab a series of data published sometime in the past. An example of using a past call is building a chart of a sensor value over time. A past call could be used to get humidity values over an hour period, starting at 12:00AM on June 12th.

Get a list of Stuff by Domain

Return a list of the stuff levels in a domain. If data has been published on:

- mydomain/greenthings/mything1
- mydomain/greenthings/mything2
- mydomain/bluethings/mything1

This call will return greenthings and bluethings.

Operation, URL

```
GET /2/account/domain/{domain}/stuff
```

Parameters

- domain

The *domain* is the domain in the m2m.io platform and is required.

Example

Retrieve stuff in domain:

```
http://att-api.m2m.io/2/account/domain/mydomain/stuff
```

Response

```
[
  "things",
  "morethigs"
]
```

Get things by Domain and Stuff

Returns all things within specified domain and stuff. If data has been published on:

- mydomain/greenthings/mything1
- mydomain/greenthings/mything2

- mydomain/bluethings/mything1

A call with domain=mydomain and stuff=greenthings would return mything1 and mything2.

Operation, URL

```
GET /2/account/domain/{domain}/stuff/{stuff}
```

Parameters

- domain
- stuff

domain is the domain in the m2m.io platform and is required. *stuff* is the desired second-level grouping and is required.

Example

Retrieve *things* in *domain* (mydomain) and *stuff* (mystuff) groupings

```
http://att-api.m2m.io/2/account/domain/mydomain/stuff/mystuff
```

Response


```
[
  {
    "domain": "mydomain",
    "stuff": "mythings",
    "thing": "thing001",
    "active": 1,
    "licenseid": "m5f7JJyV3h1pBHk:78dc1e34-7ee0-484d-afc8-
93609deaaa9c",
    "creation": 0,
    "whatevers": {
      "KEY": "mydomain:mythings:thing001"
    }
  },
  {
    "domain": "mydomain",
    "stuff": "mythings",
    "thing": "thing002",
    "active": 1,
    "licenseid": "m5f7JJyV3h1pBHk:78dc1e67-7ee0-484d-afc2-
```

```
93609deaba9c",
    "creation": 0,
    "whatevers": {
      "KEY": "mydomain:mythings:thing002"
    }
  }
]
```

Get Information About a Thing

This operation returns information about a thing such as its license id and ACLs.

Operation, URL

```
GET /2/account/domain/{domain}/stuff/{stuff}/thing/{thing}
```

Parameters

- domain
- stuff
- thing

domain is the domain in the m2m.io platform and is required. *stuff* is the desired second-level grouping and is required. *thing* is the device in question.

Example

Retrieve properties of a *thing*:

```
http://att-
api.m2m.io/2/account/domain/mydomain/stuff/mythings/thing/thing001
```

Response

```
{
  "domain": "mydomain",
  "stuff": "mythings",
  "thing": "thing001",
  "active": 1,
  "licenseid": "ni24Jq0BL6DKdp5:6def569d-cb54-406e-b1db-9D252a1e15e0",
  "creation": 0,
  "whatevers": {},
  "acl": {
    "list": [
```

```
      {
        "aclid": "e99c5b01-ab05-4abb-b31b-a7fa20f64983:api",
        "aclType": "api",
        "active": 1,
        "attributes": {
          "mydomain/#": "get:post:delete"
        }
      },
      {
        "aclid": "e99c5401-ab05-410b-b31b-a7fa30f64283:m2m",
        "aclType": "m2m",
        "active": 1,
        "attributes": {
          "mydomain/#": "pub:sub"
        }
      }
    ]
  }
}
```

Get Present data about a Thing

Present returns the most recently published values.

Operation, URL

```
GET /2/account/domain/{domain}/stuff/{stuff}/thing/{thing}/present
```

Parameters

- domain
- stuff
- thing
- whatever [op]

The *domain* is the domain in the m2m.io platform and is required. *stuff* is the desired second-level grouping and is required. *thing* is the device in question. *whatever* is the JSON object key.

The response returns:

- domain, stuff, thing for the message
- m2m_raw - the raw JSON message string

- m2m_nestedkey
- clock
- attributes - the parsed JSON objects along with a database clock value

Examples

Retrieve most recently published values

```
http://att-
api.m2m.io/2/account/domain/mydomain/stuff/mythings/thing/mything
```

Response

```
{
  "domain": "mydomain",
  "stuff": "mythings",
  "thing": "mything",
  "m2m_raw": "{\"hello\":\"world\"}",
  "m2m_nestedkey": [],
  "clock": 0,
  "attributes": {
    "hello": "world",
    "hello:clock": "1369518276120001"
  }
}
```

Retrieve most recently published values of a specific object

```
http://att-
api.m2m.io/2/account/domain/mydomain/stuff/mythings/thing/mything?
whatever=hello
```

Response

```
{
  "domain": "mydomain",
  "stuff": "mythings",
  "thing": "thing001",
  "m2m_raw": "{\"hello\":\"world\"}",
  "m2m_nestedkey": [],
  "clock": 0,
  "whatevers": {
```

```
    "hello": {
      "domain": "mydomain",
      "stuff": "mythings",
      "thing": "thing001:hello",
      "m2m_nestedkey": [],
      "clock": 0,
      "attributes": {}
    }
  },
  "attributes": {
    "hello": "world",
    "hello:clock": "1369518276120001"
  }
}
```

Get Past data about a Thing

Past returns previously published data.

Operation, URL

```
GET /2/account/domain/{domain}/stuff/{stuff}/thing/{thing}/past
```

Parameters

- domain
- stuff
- thing
- attributes
- start [op]
- end [op]
- limit [op]
- format [op]

domain is the domain in the m2m.io platform and is required. *stuff* is the desired second-level grouping and is required. *thing* is the device in question and is required.

Start and end parameters are used to slice the time series data into a time range. These values could be used to query values for one hour or one day to build charts or other visualizations.

limit is an optional number value to specify the number of results desired.

attributes are the JSON object key values published in the MQTT message payload. These can be specified or a "wildcard" (delineated by '_') can be given to return all attributes. The examples below illustrate this behavior.

Examples

In the following examples assume thing001 published temperature and humidity values on the MQTT topic mydomain/mythings/thing001. The MQTT payload would be formatted as:

```
{"temperature":100, "humidity":22}
```

Retrieve most recently published values (with wildcard for attributes)

```
http://att-
api.m2m.io/2/account/domain/mydomain/stuff/mythings/thing/thing001/past
?attributes=_
```

Response

```
{
  "domain": "mydomain",
  "stuff": "mythings",
  "thing": "thing001",
  "lastentry": "1369518276118890",
  "results": {
    "1369592198946353": {
      "temperature": 100
    },
    "1369592187210084": {
      "temperature": 101
    },
    "1369592182612328": {
      "temperature": 99
    },
    "1369590782704041": {
      "humidity": 22
    },
    "1369590779565053": {
      "humidity": 23
    },
    "1369518276118890": {
```

```
      "humidity": 25
    }
  }
}
```

Retrieve most recently published values (temperature)

```
http://att-
api.m2m.io/2/account/domain/mydomain/stuff/mythings/thing/thing001/past
?attributes=temperature
```

Response

```
{
  "domain": "mydomain",
  "stuff": "mythings",
  "thing": "thing001",
  "lastentry": "1369518276118890",
  "results": {
    "1369592198946353": {
      "temperature": 100
    },
    "1369592187210084": {
      "temperature": 101
    },
    "1369592182612328": {
      "temperature": 99
    }
  }
}
```

Limit to 2 records

```
http://att-
api.m2m.io/2/account/domain/mydomain/stuff/mythings/thing/thing001/past
?attributes=_&limit=2
```

Response

```
{
  "domain": "mydomain",
  "stuff": "mythings",
  "thing": "thing001",
```

```
  "lastentry": "1369518276118890",
  "results": {
    "1369592198946353": {
      "temperature": 100
    },
    "1369592187210084": {
      "temperature": 101
    }
  }
}
```

Create a New Thing

API endpoint to add *things*. Things are the third-level of the publish topic. These usually are devices in a typical device/sensor application.

Operation, URL

```
POST /2/account/domain/{domain}/stuff/{stuff}/thing/{thing}
```

Parameters

- domain
- stuff
- thing

domain is the domain in the m2m.io platform and is required. *stuff* is the desired second-level grouping and is required. *thing* is the device in question and is required.

Example

Retrieve previously published data on com.example/things/device01

```
http://att-
api.m2m.io/2/account/domain/mydomain/stuff/mythings/thing/thing001
```

/messaging

Send messages via Email or SMS - a simple way to send notifications.

Email

Operation, URL

```
POST /2/account/domain/{domain}/email
```

Parameters

- domain
- to
- subject
- body
- alias [op]
- from [op]
- frompass [op]

  domain is the user's m2m.io domain and is required. *to, subject, body* and *alias* define the email message. *from* is an optional from email address and *frompass* is the password for that email address. Currently the API only supports *from* email addresses to be Google accounts.

Example

Send "This is an alert email" to me@example.com

```
http://att-api.m2m.io/2/account/domain/mydomain/email?
to=me%40mydomain.com&subject=Alert&body=This+is+an+alert+email&alias=Al
ert+Emailer
```

SMS

Operation, URL

```
POST /2/account/domain/{domain}/sms
```

Parameters

- domain
- to
- message
- topic [op]

domain is the user's m2m.io domain and is required. *to* and *message* define the

SMS message and are required. *topic* is optional and defines an MQTT topic on which the SMS status is published.

Example

Send "This is an SMS" to 3035551212

```
http://att-api.m2m.io/2/account/domain/mydomain/sms?
message=This+is+an+SMS&to=3035551212
```

/group

Get Groups

Operation, URL

```
GET /2/account/domain/{domain}/groups
```

Parameters

- domain

domain is the user's m2m.io domain and is required.

Example

Get groups in mydomain

```
http://att-api.m2m.io/2/account/domain/mydomain/group
```

Response

```
[
  {
    "rowkey": "903359e9-3338a-57f9-b034-476396213bd0",
    "domain": "mydomain",
    "name": "ABCDevices",
    "email": "me@mydomain.com",
    "creator": "me@mydomain.com",
    "active": 1,
    "things": {
      "things:0": "mydomain:mythings:thing001",
      "things:2": "mydomain:mythings:thing023",
      "things:1": "mydomain:mythings:thing050"
```

```
    }
  },
  {
    "rowkey": "11334329-73c3-4fff-a639-d814728d1732",
    "domain": "mydomain",
    "name": "XYZDevices",
    "email": "me@mydomain.com",
    "creator": "me@mydomain.com",
    "active": 1,
    "things": {
      "things:0": "mydomain:mythings:thing022"
    }
  }
]
```

Get Group by ID

Operation, URL

```
GET /2/account/domain/{domain}/id/{id}
```

Parameters

- domain
- ID

domain is the user's m2m.io domain and is required. *ID* is the group ID and is required.

Example

Retrieve group with specified ID

```
http://att-api.m2m.io/2/account/domain/mydomain/id/1c3fffff-7dd3-4522-a639-d81bafad1732
```

Response

```
{
  "rowkey": "1c3fffff-7dd3-4522-a639-d81bafad1732",
  "domain": "mydomain",
  "name": "ABCDevices",
  "email": "me@mydomain.com",
  "creator": "me@mydomain.com",
```

```
  "active": 1,
  "things": {
    "things:0": "mydomain:mythings:thing001",
    "things:2": "mydomain:mythings:thing023",
    "things:1": "mydomain:mythings:thing050"
  }
}
```

Create a new Group

Operation, URL

```
POST /2/account/domain/{domain}/group
```

Parameters

- domain
- email
- creator
- name
- info [op]
- things [op]

domain is the user's m2m.io domain and is required. *email* is the user to which this group applies. *creator* is the email of the person creating this group. *name* is the short text name of this group. *info* is an optional longer description of this group. *things* is an optional way to add *things* to the group as it is being created. This would be the shorthand way of doing the following: create group or add *things* to group.

Example

Create a Group, Adding a Thing

```
http://att-api.m2m.io/2/account/domain/mydomain/group?
email=me%40mydomain.com&creator=me%40mydomain.com&name=ABCDevices&thing
s=mydomain&things=mydomain%3Amythings%3Athing001
```

Response

```
{
  "rowkey": "1c77b329-ffff-4aaa-a639-d816363d1732",
```

```
  "domain": "mydomain",
  "name": "ABCDevices",
  "email": "me@mydomain.com",
  "creator": "me@mydomain.com",
  "active": 1,
  "things": {
    "things:0": "mydomain:mythings:thing001"
  }
}
```

Add Things to Group

Operation, URL

```
PUT /2/account/domain/{domain}/id/{id}
```

Parameters

- domain
- ID

domain is the user's m2m.io domain and is required. *ID* is the group ID to which the *thing* will be added and is required.

Example

Add thing mydomain:mythings:thing002 to group

```
http://att-api.m2m.io/2/account/domain/mydomain/id/1d33bfff-73dd-4abf-a333-d737373d1732?things=domain%3Amythings%3Athing002
```

Response

```
{
  "rowkey": "1d33bfff-73dd-4abf-a333-d737373d1732",
  "domain": "mydomain",
  "name": "ABCDevices",
  "email": "me@mydomain.com",
  "creator": "me@mydomain.com",
  "active": 1,
  "things": {
    "things:0": "mydomain:mythings:thing001",
    "things:1": "mydomain:mythings:thing002"
  }
```

```
}
```

Remove Things from Group

Operation, URL

```
DELETE /2/account/domain/{domain}/id/{id}
```

Parameters

- domain
- ID
- thing(s)

domain is the user's m2m.io domain and is required. *ID* is the group ID from which the *thing* will be removed and is required. *thing(s)* is a list of *things* to remove from the specified group.

Example

Remove thing mydomain:mythings:thing001 from group

```
http://att-api.m2m.io/2/account/domain/mydomain/id/1d33bfff-73dd-4abf-a333-d737373d1732?things=domain%3Amythings%3Athing001
```

Response

```
{
  "rowkey": "1d33bfff-73dd-4abf-a333-d737373d1732",
  "domain": "mydomain",
  "name": "ABCDevices",
  "email": "me@mydomain.com",
  "creator": "me@mydomain.com",
  "active": 1,
  "things": {
    "things:1": "mydomain:mythings:thing002"
  }
}
```

/rule

This API endpoint provides operations for adding, modifying and deleting rules.

Rules are associated with a domain and are applied to all MQTT messages sent by devices or other MQTT clients which connected using an account in that domain.

Get a list of Rules

Used to query the list of rules in a user's domain.

Operation, URL

```
GET /2/account/domain/{domain}/rule
```

Parameters

- domain
- stuff [op]
- thing [op]
- username [op]
- name [op]

domain is the user's m2m.io domain and is required. *name*, *username*, *stuff*, and *thing* are optional and further define the query to narrow down the list of rules returned.

Example

Return all rules in the *domain*, mydomain.

```
http://att-api.m2m.io/2/account/domain/mydomain/rule
```

Get a Rule by ID

Return the rule with the supplied ID.

Operation, URL

```
GET /2/account/domain/{domain}/rule/{ruleid}
```

Parameters

- domain
- ruleid

domain is the user's m2m.io domain and is required. *ruleid* is the ID of the specific rule and is required.

Example

Return rule with ID: abcdef-1234-5678-xyz-987654321

```
http://att-api.m2m.io/2/account/domain/mydomain/rule/abcdef-1234-5678-xyz-987654321
```

Save a New Rule

The API operation to create a rule.

Operation, URL

```
POST /2/account/domain/{domain}/rule
```

Parameters

- domain
- name
- rule
- description [op]
- stuff [op]
- thing [op]

domain is the user's m2m.io domain and is required. *name* is a short text name for the rule. *rule* is the actual text of the *rule* and is required. See the "Rule Engine Usage document" for more information on defining rules. *description* is an optional, longer, free-form description of the rule. *stuff* and *thing* are optional.

Example

Create rule "incoming/topic" {} -> !! "outgoing/topic"

```
http://att-api.m2m.io/2/account/domain/mydomain/rule?rule=%22incoming%2Ftopic%22+%7B%7D+-%3E+!!+%22outgoing%2Ftopic%22&name=MyRule
```

Note: the *rule* is URL encoded.

Update an Existing Rule

Operation, URL

```
PUT /2/account/domain/{domain}/rule/{ruleid}
```

Parameters

- domain
- ruleid
- name [op]
- rule [op]
- description [op]
- stuff [op]
- thing [op]

domain is the user's m2m.io domain and is required. *ruleid* is the ID of the rule that will be modified and is required. *rule* is the actual text of the rule. This is where the *rule* is modified. See the "Rule Engine Usage" document for more information on defining rules. *name* is a short text name for the rule. *description* is an optional, longer, free-form description of the rule. *stuff* and *thing* are optional.

Example

Update rule ID abcdef-1234-5678-xyz-987654321 to "incoming/topic" {} -> !! "outgoing/topic"

```
http://att-api.m2m.io/2/account/domain/mydomain/rule/abcdef-1234-5678-xyz-987654321?rule=%22incoming%2Ftopic%22+%7B%7D+-%3E+!!+%22outgoing%2Ftopic%22
```

Note: the *rule* is URL encoded.

Deactivate a Rule

Remove the rule with the supplied ID.

Operation, URL

```
DELETE /2/account/domain/{domain}/rule/{ruleid}
```

Parameters

- domain
- ruleid

domain is the user's m2m.io domain and is required. *ruleid* is the ID of the specific rule and is required.

Example

Remove rule with ID: abcdef-1234-5678-xyz-987654321

```
http://att-api.m2m.io/2/account/domain/mydomain/rule/abcdef-1234-5678-xyz-987654321
```

/store

Use the m2m.io API to store generic, user-defined, key/value pairs. Users have the ability to create ROWS of key/value pairs. An example: User 1 could create a row called KEYS, and in that row store a set of key/value pairs, say k1=1, k2=2. That user then could create a second row DIFFERENTKEYS and in that row store k100=100, k200=200. The user could have even stored the same key, with different values, say k1=100, k2=200. The user then could query KEYS,k1 and get 1 and DIFFERENTKEYS,k1 and get 100.

Typically rowkeys are in the format "user:key", for example: user@mydomain.com:KEYS.

Stores have the concept of being "user protected". If a store is user protected only users in that domain can read the rowkey & key/value pairs. If the store is not user protected the rowkey & key/value pairs are shared across all users.

Get User Keys from Store

Get a list of user rowkeys.

Operation, URL

```
GET /2/account/domain/{domain}/store
```

Parameters

- domain

domain is the user's m2m.io domain and is required.

Example

Retrieve user's keys from the domain mydomain

```
http://att-api.m2m.io/2/account/domain/mydomain/store
```

Save or Update Generic Key/Value Pairs

Operation, URL

```
POST /2/account/domain/{domain}/store/{key}
```

Parameters

- domain
- key
- protect

domain is the user's m2m.io domain and is required. *key* is the rowkey to use when storing the desired key/value pairs. *protect* is a boolean specifying if the store is user protected.

Example

Add/update key/value pairs (k1, k2) from the rowkey user@mydomain.com:KEYS

```
http://att-
api.m2m.io/2/account/domain/mydomain/store/user%40mydomain.com%3AKEYS?
protect=false

using post parameters "k1=1,k2=2"
```

Get data saved in the Store

Operation, URL

```
GET /2/account/domain/{domain}/store/{key}
```

Parameters

- domain
- key
- protect

domain is the user's m2m.io domain and is required. *key* is the rowkey to use when storing the desired key/value pairs. *protect* is a boolean specifying if the store is user protected.

Example

Retrieve key/value pairs from the rowkey user@mydomain.com:KEYS

```
http://att-
api.m2m.io/2/account/domain/mydomain/store/user%40mydomain.com%3AKEYS?
protect=false
```

Delete an object saved in the Store

This operation removes rowkeys, which removes all key/value pairs associated with that rowkey.

Operation, URL

```
DELETE /2/account/domain/{domain}/store/{key}
```

Parameters

- domain
- key
- protect

domain is the user's m2m.io domain and is required. *key* is the rowkey to use when storing the desired key/value pairs. *protect* is a boolean specifying if the store is user protected.

Example

Remove the rowkey user@mydomain.com:KEYS

```
http://att-
api.m2m.io/2/account/domain/mydomain/store/user%40mydomain.com%3AKEYS?
protect=false
```

/acl

The ACL (Access Control List) defines what access a user has for API access and MQTT publish/subscribe access. ACLs are used for security to prohibit access to resources in the 2lemetry platform. ACLs are related to users as well as things.

Get ACL By ID

Returns an array of information relating to this aclid. Important properties include aclType and the attributes object. An aclType would either be m2m (device related) or api (RESTful webservice)

Operation, URL

```
GET /2/account/domain/{domain}/acl/{aclid}
```

Parameters

- domain
- aclid

Example

Retrieve the acl properties of com.example

```
http://att-api.m2m.io/2/account/domain/com.example/acl/xxxxx-xxx-xxxx-xxx-xxxxxxxxxx
```

Details of Response

Understanding ACL topic strings. The previous example would result in:

```
[
  {
    "aclid": "xxxxx-xxx-xxxx-xxx-xxxxxxxxxx:api",
    "aclType": "api",
    "active": 1,
    "attributes": {
      "com.example/#": "get:post:delete"
    }
  },
  {
    "aclid": "xxxxx-xxx-xxxx-xxx-xxxxxxxxxx:m2m",
    "aclType": "m2m",
    "active": 1,
    "attributes": {
      "com.example/#": "pub:sub"
    }
  }
]
```

The API that ACL defines as a topic structure is *com.example/#* and an action *get:post:delete* - Allowing RESTful PUT/POST, GET and DELETE. This would allow anything that connects to the API to retrieve RESTful data from the domain 'com.example'. The could be restricted by only allowing data from *com.example/mystuff/mything*. If this were the only ACL a user would only be able

to retrieve data that may have been published on *com.example/mystuff/mything*.

The same would go for the m2m ACL. In this example a device would be allowed to publish and subscribe to anything that starts with *com.example/* in the topic structure. If this ACL was only defined as *"com.example/mystuff/mything":"sub"* then this client would only be able to subscribe on this specific topic.

**Note: MQTT does not define any error or message that is the result of unauthorized topics. Therefore the client would not be notified and would either not receive the subscription or not deliver the "Publish".*

Update ACL by ID

Allows custom definitions to existing ACLs.

Operation, URL

```
PUT /2/account/domain/{domain}/acl/{aclid}
```

Parameters

- domain
- aclid
- api - true or false
- m2m - true or false
- get - true or false
- post - true or false
- delete - true or false
- subscribe - true or false
- publish - true or false
- topic

The *domain* is the current user's domain. The *aclid* is the ACL that you want to update. The *api* defines an API's ACL. *m2m* defines an m2m (device) ACL. *get* allows RESTful GET access. *post* allows RESTful PUT/POST access. *delete* allows RESTful DELETE access. *subscribe* allows MQTT subscriptions. *publish* allows MQTT publishes. *topic* is the string to add or modify these actions.

**Note: the user's domain will automatically be prepenned to the topic string for security purposes.*

Remove an ACL type

Allows you to clear specific ACL's based on ID and topic string.

Operation, URL

```
PUT /2/account/domain/{domain}/acl/{aclid}/remove
```

Parameters

- domain
- aclid
- api
- m2m
- topic

The *domain* is the current user's domain. The *aclid* is the ACL that you want to update. *api* defines an API ACL. *m2m* defines an m2m (device) ACL. The *topic* is the topic string that you want to clear. For instance, if the following topic string is currently defined *"com.example/#": "pub:sub"* you would post a topic of *"com.example/#"* to clear its state.

Get My ACL

```
GET/2/account/acl
```

/domain

A domain is the highest level of grouping or organization in the m2m.io platform. Multiple users can belong under one domain and share access to devices and the resulting data.

Get the DomainModel (of the Authenticated User)

A user returns the domain properties of the user making the call. Important properties include the license limit of the domain and the number of currently consumed licenses.

Operation, URL

```
GET /2/account/domain
```

Parameters

none

Example

Retrieve the domain of the users authenticating against the call.

```
http://att-api.m2m.io/2/account/domain
```

Get a DomainModel by Domain (based on Key)

Returns the domain properties of a domain given the domain ID. Important properties include the license limit of the domain and the number of currently consumed licenses.

Operation, URL

```
GET /2/account/domain/{rowkey}
```

Parameters

- rowkey

rowkey is the domain ID. In an organization this is typically their reverse domain, for example: com.example. For developer/test accounts, this will be a MD5 hash of the email used to sign up for the account.

Example

Retrieve the domain properties of com.example

```
http://att-api.m2m.io/2/account/domain/com.example
```

Create a new DomainModel

This endpoint allows creation of a domain.

Operation, URL

```
POST /2/account/domain
```

Parameters

- domain
- name

domain is the fully qualified domain name. For example: com.example. The *name* is

a short-text name to be given to the domain in the data store.

Example

Create the com.example domain

```
http://att-api.m2m.io/2/account/domain

with post parameters, domain and name.
```

Gets Published Topics

```
GET /2/account/domain/{rowkey}/topics
```

Gets Published Topics

```
GET /2/account/domain/topics
```

/account

API endpoint for account maintenance.

Gets AccountModel

Fetches information, such as domain, about an account used in the request.

Operation, URL

```
 GET /2/account
```

Parameters

none

Example

```
http://att-api.m2m.io/2/account
```

Response

```
{
  "rowkey": "me@mydomain.com",
  "email": "me@mydomain.com",
```

```
  "password": "CBZSf3zXl6k=",
  "domain": "mydomain",
  "active": 1,
  "aclid": "525ca6f33-cb3c-44df-6626-ffff9bd096a3",
  "creation": 0,
  "attributes": {}
}
```

Gets AccountModel by email

Fetches information, such as domain, about a different account.

Operation, URL

```
GET /2/account/{rowkey}
```

Parameters

none

Example

```
http://att-api.m2m.io/2/account/user%40domain.com
```

Response

```
{
  "rowkey": "me@mydomain.com",
  "email": "me@mydomain.com",
  "password": "CBZSf3zXl6k=",
  "domain": "mydomain",
  "active": 1,
  "aclid": "525ca6f33-cb3c-44df-6626-ffff9bd096a3",
  "creation": 0,
  "attributes": {}
}
```

Creates a new user account

This operation adds a user account.

Operation, URL

```
POST /2/account/domain/{domain}
```

Parameters

- email
- password
- domain [op]

Rowkey is the *email* address used as a username on the m2m.io platform. The *password* is the desired password for that user. *domain* is the user's m2m.io domain and is optional. The *domain* must exist before creating a user in that domain.

Example

Add user, user@mydomain.com with password: fakepassword

```
http://att-api.m2m.io/2/account?
email=user%40mydomain.com&password=fakepassword
```

Update user account

This operation updates properties of a user account.

Operation, URL

```
PUT /2/account/{rowkey}
```

Parameters

- user email
- username [op]
- email [op]
- password [op]
- githubtoken [op]
- twittertoken [op]
- facebooktoken [op]
- sfdctoken [op]
- googletoken [op]
- linkedintoken [op]
- role [op]

Rowkey is the email address used as an *username/ID* on the m2m.io platform.

Example

Update user, user@mydomain.com

```
http://att-api.m2m.io/2/account/user%40mydomain.com?
username=FaceSmooch&email=u%40example.com&password=mypass&githubtoken=g
&twittertoken=t&facebooktoken=f&sfdctoken=s&googletoken=g&linkedintoken
=l&role=role
```

/weather

Weather endpoint: Used to retrieve weather data given a location parameter.

Retrieve Weather Data

Operation, URL

```
GET /2/weather
```

Parameters

- q

The parameter *q* is the location parameter. This parameter can be:

- US zip code
- UK postcode
- Canada postal code
- IP address
- Latitude/Longitude (decimal degree)
- City name

Example

Weather in Aspen, CO.

```
http://att-api.m2m.io/2/weather?q=Aspen%2C+CO
```

Response

```
[
  {
```

```
    "cloudcover": "25",
    "FeelsLikeC": "22",
    "FeelsLikeF": "72",
    "humidity": "9",
    "observation_time": "09:34 PM",
    "precipMM": "0.0",
    "pressure": "1021",
    "temp_C": "22",
    "temp_F": "72",
    "visibility": "16",
    "weatherCode": "116",
    "weatherDesc": [
      {
        "value": "Partly Cloudy"
      }
    ],
    "weatherIconUrl": [
      {
        "value":
"http://www.worldweatheronline.com/images/wsymbols01_png_64/wsymbol_0002_sunny_intervals.png"
      }
    ],
    "winddir16Point": "S",
    "winddirDegree": "180",
    "windspeedKmph": "22",
    "windspeedMiles": "14"
  }
]
```