Intellimeter API Quickstart Guide

V. 1.0

Getting Started

Intellimeter makes it easier to collect and customize how you receive your energy metering reads through the use of APIs. Continue reading on to find out how to connect to our API, some examples, and pull your first reading from this guide!

Should you need any technical support along the way, please contact support@intellimeter.ca for assistance.

Contents

For Starters

What is an API?

An application programming interface (commonly known as an API), is simply a communication protocol used to retrieve raw data from a server. For example, when accessing your social media accounts, an API request is sent to the server to retrieve your profile information and with this information, the website will then stylized to make it look pleasing to the user.

Example API Output

```
HTTP/1.1 200 OK
{

      "profile": [
      {
      "id":"123456789",
      "name":"Jane Doe",
"profile_img":"https://socialmedia.com/pf_imgs/fancy_img_JaneDoe_120px
.jpg",…
      "theme":"redlights"
      }
   ]
}
```

With a little bit of coding and some UI design, it will generate the page/interface in a more familiar view like the photo here. Another example is saving your game data to your iCloud or Gmail account. By accepting the app to use your account for saving your app data, the application will send and receive API requests on your behalf.

Without realizing, you have already interacted with an API!

When would I need this?

In terms of energy analysis, using an API is ideal for those who would like to integrate their data with their own system/interface (e.g. the customer already has a company portal they would like to integrate it on). This is also great for those who just need the values of the meter without all the fancy dashboards involved.

Who can use this?

Any Intellimeter customer who owns our equipment or software will be able to use our APIs.

Where can I download this?

There is no need to download any files or dependencies using our API.

If you have lingering questions that may not have been answered above, check out our API page on our website for more information. If that’s not enough, feel free to contact support@intellimeter.ca!

API Details

Intellimeter’s API is:

- REST based
- Has JSON and XML outputs available

Each of the URL routes can be accessed via **https://api.intellimeter.ca** followed by the route. Any route which returns JSON can be set to also return other data formats too! Simply include *responseType* as the URL argument, and set the value to any of the following: "xml", "json", or "yaml". The response will be the appropriate mime type as well.

The default is JSON.

If you use a type not included in this list, the response will be HTTP status code 400, however the request will still be executed.

Route	Method	Description

/session	GET	Returns information about the account you are currently logged into.
		JSON Example:
		`{"uuid": 1, "project": 1, "username": "test", "email": "test@example.com", "joined": "2019-10-29 16:07:17", "project_name": "myProject"}`
		XML Example:
		`<?xml version="1.0" encoding="UTF-8" ?><Response><uuid type="int">1</uuid><project type="int">1</project><username type="str">test</username><email type="str">test@example.com"</email><joined type="str">2019-10-29 16:07:17</joined><xml type="bool">True</xml><project_name type="str">myProject</project_name></Response>`

	PUT	Logs you into your Intellimeter Inc. API Account. There are 2 supported methods to log in:
		Method 1: Username/Password
		username: the username of your API Account
		password: the password of your API Account.
		Method 2: API Key (Recommended Method)
		key: The API Key of your Intellimeter Inc. API Account. (see */key* to generate)
		Both methods will return the UUID of the account you have logged into, or raise HTTP Error 401 if it can not log you in.
		JSON Example:
		`{"uuid": 1}`
		XML Example:
		`<?xml version="1.0" encoding="UTF-8" ?><Response><uuid type="int">1</uuid></Response>`

	DELETE	Logs you out of your Intellimeter Inc. API Account. Returns either an empty JSON string, or empty XML (depending on your specified response type)
		JSON Example:
		`{}`
		XML Example:
		`<?xml version="1.0" encoding="UTF-8" ?><Response></Response>`

/key	GET	Returns the API key of the account which is logged in
		JSON Example:
		`{"key": "f27a4f7186-d4a6635fe7-992d888955-284e0b037c"}`
		XML Example:
		`<?xml version="1.0" encoding="UTF-8" ?><Response><key type="str">5a683c7dd6-c73b2a4313-f0fb321137-9123b7cec0</key></Response>`

	POST	Generates a new API key for the account which is logged in, and returns it. It is good practice to change your key once in a while for added security.
		Note that you will need to change your API key for each of your applications which depend on this API
		JSON Example:
		`{"key": "f27a4f7186-d4a6635fe7-992d888955-284e0b037c"}`
		XML Example:
		`<?xml version="1.0" encoding="UTF-8" ?><Response><key type="str">5a683c7dd6-c73b2a4313-f0fb321137-9123b7cec0</key></Response>`

/data	GET	Returns the most recent readings of each metering point (this updates every 5 minutes)
		JSON Example:
		`[{"MeterParam": "meter1", "TimeStamping": "2019-10-30 13:15:00", "value": 28540.24}, {"MeterParam": "meter2", "TimeStamping": "2019-10-30 13:15:00", "value": 35757.359}]`
		XML Example:
		`<?xml version="1.0" encoding="UTF-8" ?><Response><item type="dict"><MeterParam type="str">meter1</MeterParam><TimeStamping type="str">2019-10-30T13:15:00</TimeStamping><value type="float">28540.24</value></item><item type="dict"><MeterParam type="str">meter2</MeterParam><TimeStamping type="str">2019-10-30T13:15:00</TimeStamping><value type="float">35757.359</value></item></Response>`

/interval	GET	Returns the most recent interval readings of each metering point (this updates every 5 minutes)
		JSON Example:
		`[{"MeterParam": "meter1", "TimeStamping": "2019-10-30 14:10:00", "value": 0.5}, {"MeterParam": "meter2", "TimeStamping": "2019-10-30 14:10:00", "value": 0.08}]`
		XML Example:
		`<?xml version="1.0" encoding="UTF-8"?><Response><item type="dict"><MeterParam type="str">meter1</MeterParam><TimeStamping type="str">2019-10-30T14:10:00</TimeStamping><value type="float">0.5</value></item><item type="dict"><MeterParam type="str">meter2</MeterParam><TimeStamping type="str">2019-10-30T14:10:00</TimeStamping><value type="float">0.08</value></item></Response>`

/user	GET	Returns the public information of each user who is also in the same site
		JSON Example:
		`[{"uuid": 1, "username": "test", "email": "test@example.com", "joined": "2019-10-29 16:07:17"}, {"uuid": 2, "username": "test2", "email": "test2@example.com", "joined": "2019-10-31 14:45:12"}]`
		XML Example:
		`<?xml version="1.0" encoding="UTF-8"?><Response><item type="dict"><uuid type="int">1</uuid><username type="str">test</username><email type="str">test@example.com</email><joined type="str">2019-10-29T16:07:17</joined></item><item type="dict"><uuid type="int">2</uuid><username type="str">test2</username><email type="str">test2@example.com</email><joined type="str">2019-10-31T14:45:12</joined></item></Response>`

	POST	Updates your account's information. Omit any property you wish to not change:
		username: the new username. Note: This must be unique. If it is not, HTTP Status code 409 will be returned.
		password: the password of the account to log into (in plain text)
		email: the email address to attach to your account
		Returns the updated account information.
		JSON Example:
		`{"uuid": 1, "project": 1, "username": "test", "email": "test@example.com", "joined": "2019-10-29 16:07:17", "project_name": "myProject"}`
		XML Example:
		`<?xml version="1.0" encoding="UTF-8" ?><Response><uuid type="int">1</uuid><project type="int">1</project><username type="str">test</username><email type="str">test@example.com"</email><joined type="str">2019-10-29 16:07:17</joined><xml type="bool">True</xml><project_name type="str">myProject</project_name></Response>`

	PUT	Creates a new user into your project. Requires the following properties:
		username: the username of the new account. Note: This must be unique. If it is not, HTTP Status code 409 will be returned.
		password: the password of the new account
		email: the email address of the new account
		Returns the newly created account
		JSON Example:
		`{"uuid": 1, "project": 1, "username": "test", "email": "test@example.com", "joined": "2019-10-29 16:07:17", "project_name": "myProject"}`
		XML Example:
		`<?xml version="1.0" encoding="UTF-8" ?><Response><uuid type="int">1</uuid><project type="int">1</project><username type="str">test</username><email type="str">test@example.com"</email><joined type="str">2019-10-29 16:07:17</joined><xml type="bool">True</xml><project_name`

		`type="str">myProject</project_name></Response>`

	DELETE	Deletes the account which is logged in. This will also cause you to be logged out. Returns either an empty JSON string, or empty XML (depending on your specified response type)
		JSON Example:
		`{}`
		XML Example:
		`<?xml version="1.0" encoding="UTF-8" ?><Response></Response>`

/graphics	GET	Generates and returns the graphic with the mimetype of "image/png" specified by the query parameters. Note that all parameters are optional, and their default values have also been listed. *type*: The type of chart you would like to generate. This can be one of the following items: "pie", "bar", or "line". Default is "pie". *meter*: Generates a line graph of the meter's historical data. This overrides the *type* property. Omit to see all meters together. *width*: The width of the graphic in thousands of pixels. Default is 16. *height*: The height of the graphic in thousands of pixels. Default is 9. Returns the graphic with the "image/png" mimetype.

/point	GET	Returns information about a specified meter. Meters can be specified one of two ways.
		Note that if both are specified, the ID will take priority.
		Method 1: By Meter Name
		name: The name of the meter to get
		Method 2: By Meter ID
		id: The ID of the meter to get
		JSON Example:
		`{"id": 1, "MeterId": "meter1", "ModbusId": 1, "Port": "USB0", "NoPhases": "3", "MeterType": "I-45/MFX/636", "ModbusMap": "Priority.ici", "SaveEvery": 5, "startAddress": 1100}`
		XML Example:
		`<?xml version="1.0" encoding="UTF-8" ?><Response><id type="int">1</id><MeterId type="str">meter1</MeterId><ModbusId type="int">1</ModbusId><Port type="str">USB0</Port><NoPhases type="str">3</NoPhases><MeterType type="str">I-45/MFX/636</MeterType><ModbusMap type="str">Priority.ici</ModbusMap><SaveEvery type="int">5</SaveEvery><startAddress type="int">1100</startAddress></Response>`

	POST	Allows you to rename a metering point. Note that this will not take effect on the DataDefault system, or CSV Generator which comes pre-installed with your setup. The meter which will be renamed can be specified one of two ways.
		Note that if both are specified, the ID will take priority.
		Method 1: By Meter Name
		name: The name of the meter to get
		Method 2: By Meter ID
		id: The ID of the meter to get
		
		new: The name the meter should be given
		This will return the new data of the meter
		JSON Example:
		`{"id": 1, "MeterId": "meter1", "ModbusId": 1, "Port": "USB0", "NoPhases": "3", "MeterType": "I-45/MFX/636", "ModbusMap": "Priority.ici", "SaveEvery": 5, "startAddress": 1100}`
		XML Example:
		`<?xml version="1.0" encoding="UTF-8" ?><Response><id type="int">1</id><MeterId type="str">meter1</MeterId><ModbusId type="int">1</ModbusId><Port type="str">USB0</Port><NoPhases type="str">3</NoPhases><MeterType type="str">I-45/MFX/636</MeterType><ModbusMap type="str">Priority.ici</ModbusMap><SaveEvery type="int">5</SaveEvery><startAddress type="int">1100</startAddress></Response>`

/file	GET	Generates a file of past readings using the respective mimetype based on the following parameters. **Mandatory Parameters** *start*: The day at the start of the collection range. Format: "YYYY-mm-dd". *end*: The day at the end of the collection range. Format: "YYYY-mm-dd". Note that the start and end date can be placed in either parameter, and the result will remain the same. **Optional Parameters** *type*: The file type of data to generate. This can be one of the following items: "csv", "json", "xml", or "yaml". Default is "csv". *meter*: The name of the specific meter to collect data from. If

		omitted, all meters will be included.