subscribe

Request a websocket data stream, and commands to manage the data stream.

Description

Request ticket and meta data:

```
https://api.eze.io/v1/subscribe/ticket
```

The `subscribe/ticket` call requests a websocket ticket and returns metadata for the group and ezeio units that are in context of the given API credentials.

Subsequent calls using the subscribe command are used to manage the data stream.

> This functionality requires firmware 21012701 or later in the device.

Parameters

The `subscribe/ticket` call has no parameters

Example usage

Using the subscribe API requires the following steps:

Step 1 : Request ticket and metadata

Call the following API endpoint using valid API credentials:

```
https://api.eze.io/v1/subscribe/ticket
```

This call will return a JSON object (see example below) with metadata listing the systems that will be accessible through the websocket.

(whitespaces added for readability)


```
{
    "reqtime": "2021-01-31T14:15:16Z",
    "ws": "wss:\/\/api.eze.io\/ws\/wstktXXXXXXXXXXXXXXXXXXXXXX",
```

```
    "api":
"https:\/\/api.eze.io\/v1\/subscribe\/wstktXXXXXXXXXXXXXXXXXXXXXX\/",
    "account": {
        "id": "123",
        "name": "eze System",
        "description": "eze System - Testaccount"
    },
    "systems": [{
            "serial": "ABC-123",
            "name": "Demo unit",
            "note": "This is a demo unit in our lab",
            "lastseen": "2021-01-31T14:10:11Z",
            "fields": [{
                "fieldno": "1",
                "name": "Tank Level: Distance from top",
                "unit": "in",
                "decimals": "2",
                "assettag": "TANKLVL",
                "loginterval": "0"
            }, {
                "fieldno": "2",
                "name": "Output flow",
                "unit": "gal/min",
                "decimals": "1",
                "assettag": "",
                "loginterval": "60"
            }]
        },
        {
            "serial": "ABC-456",
            "name": "Other demo unit",
            "note": "This is also a demo unit in our lab",
            "lastseen": "2021-01-31T14:09:56Z",
            "fields": [{
                "fieldno": "2",
                "name": "Air temperature",
                "unit": "F",
                "decimals": "1",
                "assettag": "",
                "loginterval": "300"
            }, {
                "fieldno": "3",
                "name": "Relative humidity",
                "unit": "percent",
                "decimals": "0",
                "assettag": "",
                "loginterval": "0"
            }, {
```

```
                "fieldno": "4",
                "name": "Voltage",
                "unit": "V",
                "decimals": "1",
                "assettag": "V",
                "loginterval": "0"
            }]
        }
    ],
    "status": "OK",
    "exec_time": 0.055
}
```

The `ws` property is the complete websocket URI. This URI is valid only for 10 seconds following the call to subscribe/ticket.

The `api` property should be saved until the websocket is disconnected, as it is required for subsequent calls to update the data flow.

Step 2 : Open websocket and start receiving data

Use the `ws` URI from step 1 to open a websocket connection. The websocket will automatically receive all updates to any system (ezeio) covered by the initial API call, as the data becomes available to the cloud servers.

The data received will have a `type`, describing what kind of data this is:

LOGDATA	The field data is from the 'fast log'. The interval is determined by the log interval setting on each field.
STATUS	The field data is from the 'status log'. The interval is fixed to 10 minutes.

Note that the ezeio normally buffers data before the data is uploaded to the cloud servers, so the data may be delayed with up to 20 minutes

The STATUS updates will be sent every 10 minutes even if a faster subscription is active.

This is an example of what a STATUS update looks like (whitespace added for readability):

```
{
  "channel": "export:123",
  "data": {
    "type": "STATUS",
    "serial": "ABC-123",
    "time": "2021-01-31T14:20:00Z",
    "fields": [
      {
        "1": 73.4
      },
```

```
      {
        "2": 8.112
      }
    ]
  }
}
```

Note that the fields array will include all configured fields for this unit - regardless of their log setting. All fields are always logged every 10 minutes.

A `LOGDATA` update has the following format (whitespace added for readability):

```
{
  "channel": "export:123",
  "data": {
    "type": "LOGDATA",
    "serial": "ABC-123",
    "time": "2021-01-31T14:21:15Z",
    "timeout": 0,
    "fields": [
      {
        "2": 9.021
      }
    ]
  }
}
```

Note that this message only includes the fields that are configured for fast logging (interval less than 10 minutes). Please see below for the meaning of the `timeout` property.

Step 3 : Request subscription changes

With an open websocket and while receiving data from the ezeio system, you can call the subscribe API to request immediate updates or to cancel updates from a certain ezeio.

To request unbuffered log updates, the call is:

```
https://api.eze.io/v1/subscribe/wstktXXXXXXXXXXXXXXXXXXXXXXXX/ABC123
```

The log updates will revert to normal (buffered) mode after 30 minutes.

To cancel updates, the call is:

```
https://api.eze.io/v1/subscribe/wstktXXXXXXXXXXXXXXXXXXXXXXXX/-ABC123
```

Multiple requests can be sent in the same command:

```
https://api.eze.io/v1/subscribe/wstktXXXXXXXXXXXXXXXXXXXXXXXX/-ABC123,ABC124,AB
```

```
C321,-ABC322
```

Up to 50 devices can be included in the same command

Example code to set up the websocket channel and receive data (PHP)

```php
<?php
    define("APIURI", "https://api.eze.io/v1/subscribe/ticket");

    // API keyID and key needs to be set up in eze.io under Groups->API.
    define("APIKeyID", "12345");
    define("APIKey", "XXXXXYYYYYXXXXXYYYYYXXXXXYYYYY");

    // Using the textalk/websocket client
    // https://github.com/Textalk/websocket-php
    require('vendor/autoload.php');
    use WebSocket\Client;

    // Request a websocket key and metadata using cURL
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, APIURI);

    // All API calls use Digest AUTH
    curl_setopt($ch, CURLOPT_HTTPAUTH, CURLAUTH_DIGEST);
    curl_setopt($ch, CURLOPT_USERPWD, APIKeyID.":".APIKey);

    // Set a 5s timeout, and return any received data
    curl_setopt($ch, CURLOPT_TIMEOUT, 5);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);

    // Execute the cURL request
    $response = curl_exec($ch);
    curl_close($ch);

    // Decode the json reply into an associative array
    $json = json_decode( $response, TRUE );

    // Status should be 'OK'
    if($json["status"] != "OK")
        die("ERROR:\n" . print_r($json, TRUE));

    // Open the websocket
    $wsclient = new WebSocket\Client( $json["ws"] );

    // Loop forever (until websocket disconnects)
    while ( true ) {
```

```
        try {
            $message = $wsclient->receive();

            // Process the data - here we just print it for testing
            print_r( json_decode( $message, TRUE ) );
        }
        catch (\WebSocket\ConnectionException $e) {
            // If the websocket was disconnected, exit the loop
            if( !$wsclient->isConnected() )
                break;
        }
    }
    $wsclient->close();
```