GCE_API

Ce document décrit l'utilisation des objets GCE_API et GCE_Refresh dans un widget HTML.

Refresh Widgets from WebSocket

Les données du Web Socket sont récupérables dans les widgets HTML via le tableau window.GCE_Refresh. Pour les utiliser, il faut ajouter une fonction avec un paramètre à ce tableau. Cette fonction sera appelée automatiquement lors de chaque changement d'état remonté par le WebSocket.

```
window.GCE_Refresh.push(function(data) { //IMPORTANT: biding refresh function to the WebSocket
    // received all IO and ANA that change in the V5 as an object
    // ex for IO: { _id: 65536, on: true }
    // ex for ANA: { _id: 196608, value: 1 }
    switch (data['_id'])
    {
        case 65536:
            // do something when IO 65536 is update
            break;
    }
})
```

Methods

- GCE_API.get
- GCE_API.getIO
- GCE_API.getANA
- GCE_API.put
- GCE_API.putIO
- GCE_API.putANA
- GCE_API.request

get

Description

GET request to API GCE_API.get(path, query = null);

Parameters

- **path** (String): API road
- **query** (String): Query string (optional)

Quick Example

```
GCE_API.get("/api/core/io/65536")
.then((ret) => {
  console.log("OK ", ret)
}).catch((err) => {
  console.error("KO ", err)
});
```

Return

```
{_id: 65536, on: true }
```

getIO

Description

GET requested IO to API GCE_API.getIO(id, query = null);

Parameters

- **id** (Number): the ID of the IO to GET
- **query** (String): Query string (optional)

Quick Example

```
GCE_API.getIO(65536)
.then((ret) => {
  console.log("OK ", ret)
}).catch((err) => {
  console.error("KO ", err)
});
```

Return

```
{_id: 65536, on: true }
```

getANA

Description

GET requested ANA to API GCE_API.getANA(id, query = null);

Parameters

- **id** (Number): the ID of the ANA to GET
- **query** (String): Query string (optional)

Quick Example

```
GCE_API.getANA(196608)
.then((ret) => {
  console.log("OK ", ret)
}).catch((err) => {
  console.error("KO ", err)
});
```

Return

```
{_id: 196608, value: 100 }
```

put

Description

PUT request to API PUT functions are asynchronous. The update will be received in the websocket. GCE_API.put(path, body, query = null);

Parameters

- **path** (String): API route
- **value** (any): the asked Value
- **query** (String): Query string (optional)

Quick Example

```
GCE_API.put('api/core/io/65536', { toggle: true })  //  toggle IO state
GCE_API.put('api/core/io/65536', { on: true })  // set ON
GCE_API.put('api/core/io/65536', { on: false }) // set OFF
```

```
.then((ret) => {
  console.log("OK ", ret)
}).catch((err) => {
  console.error("KO ", err)
});
```

Return

```
void
```

putIO

Description

PUT requested IO to API PUT functions are asynchronous. The update will be received in the websocket. GCE_API.putIO(id, value, query = null);

Parameters

- **id** (Number): the ID of the IO to PUT
- **value** (any): the asked state for the IO could be true, false or 'toggle'
- **query** (String): Query string (optional)

Quick Example

```
GCE_API.putIO(65536, 'toggle')  //  toggle IO state
GCE_API.putIO(65536, true)  // set ON
GCE_API.putIO(65536, false) // set OFF
.then((ret) => {
  console.log("OK ", ret)
}).catch((err) => {
  console.error("KO ", err)
});
```

Return

```
void
```

putANA

Description

PUT requested ANA to API PUT functions are asynchronous. The update will be received in the websocket. GCE_API.putANA(id, value, query = null);

Parameters

- **id** (Number): the ID of the ANA to GET
- **value** (any): the asked value for the ANA
- **query** (String): Query string (optional)

Quick Example

```
GCE_API.putANA(196608, 100) // set value to 100
.then((ret) => {
  console.log("OK ", ret)
}).catch((err) => {
  console.error("KO ", err)
});
```

Return

```
void
```

Example Template

```
<script>
window.GCE_Refresh.push(function(data) { //IMPORTANT: biding refresh function to the WebSocket
        // received all IO and ANA that change in the V5 as an object
        // ex for IO: { _id: 65536, on: true }
        // ex for ANA: { _id: 196608, value: 1 }
        switch (data['_id'])
        {
            case 65536: // do something when IO 65536 is update
                changeIconFromState(data.on)
                break;
        }
    })


    function init() {
        GCE_API.getIO(65536).then((ret) => { // ask for the actual value
            changeIconFromState(ret.on)
        })
    }

    function changeIconFromState(state) {
```

```
        let myIcon = document.getElementById("IO_65536_State");
        myIcon.style.color = state ? "#1fdb49" : "#666565";
        myIcon.className = "gce-glyph "+(state ? "icon-lamp4_light" : "icon-lamp5_light");
    }

    init()
</script>

<!-- Requests toogle on value of the 65536 IO -->
<button onclick="GCE_API.putIO(65536, 'toggle')">Toggle IO</button>

<!-- Requests a set ON command on value of the 65536 IO -->
<button onclick="GCE_API.putIO(65536, true)">Set ON IO</button>

<!-- Requests a set OFF command on value of the 65536 IO -->
<button onclick="GCE_API.putIO(65536, false)">Set OFF IO</button>

<i id="IO_65536_State" class="gce-glyph" style="font-size:48px"></i>
```