Backend API

This page gathers all the API calls that can be used by the front end.

> Front end -> Backend

Global configuration

Collection of all functions/API calls available to the front end that handles the global variables.

saveConfigParams(configParams)

saveConfigParams(configParams)

Saves the config parameters to the persistent file

React usage example

```
import { saveConfigParams } from "./api/backend-api";

saveConfigParams(globalConfig);
```

loadConfigParams()

loadConfigParams()

Loads the config parameters from the persistent file

React usage example

```
import { loadConfigParams } from "./api/backend-api";

globalConfig = loadConfigParams();
```

Data base

Collection of all functions/API calls available to the front end that handles the communication with the data base, such as fetching and storing data.

getMaterialList()

getMaterialList()

TODO

React usage example

```
import { getMaterialList } from "./api/backend-api";

globalConfig = getMaterialList();
```

getMaterialAt(index)

getMaterialAt(index)

Returns the material at an `index` from the database.

React usage example

```
import { getMaterialAt } from "./api/backend-api";

const elem21 = getMaterialAt(21);
```

getExperimentAt(index)

getExperimentAt(index)

Returns the experiment at an `index` from the database.

React usage example

```
import { getExperimentAt } from "./api/backend-api";

const elem21 = getExperimentAt(21);
```

getDataPointArrayAt(index)

getDataPointArrayAt(index)

Returns an array of `DataPoint` at an `index` from the database.

React usage example

```
import { getDataPointArrayAt } from "./api/backend-api";
import { DataPointType } from "types/DataPointTypes";

const dataPointArrya: DataPointType[] = getDataPointArrayAt(21);
```

Core

startExperimentRoutineJS()

startExperimentRoutineJS()

This function calls the `start_experiment_routine()` on the backend.

Usually it should be used to handle when the user press a "start experiment" button or something similar.

React usage example

```
import { getMaterialList } from "./api/backend-api";

onClick(()=>{
   startExperimentRoutineJS();
};)
```

endExperimentRoutineJS()

endExperimentRoutineJS()

This function calls the `end_experiment_routine()` on the backend.

Usually it should be used to handle when the user press a "end experiment" button or something similar.

React usage example

```
import { getMaterialList } from "./api/backend-api";

onClick(()=>{
   endExperimentRoutineJS();
};)
```

setCustomMovementDistanceJS()

setCustomMovementDistanceJS()

Warning

DEPRECATED

This function calls the `set_custom_movement_distance(new_movement_distance)` on the backend.

Sets the movement distance that the z-axis moves when the user is controlling the machine manually.

This distance is set in MILLIMETERS

Returns 1 if succeeded.

React usage example

```
import { setCustomMovementDistanceJS } from "./api/backend-api";

onClick(()=>{
   // Sets the movement distance to 50 mm
   setCustomMovementDistanceJS(50);
};)
```

returnZAxisJS()

returnZAxisJS()

This function calls the `return_z_axis()` on the backend.

Returns the z-axis to the origin.

Returns 1 if succeeded (if the function was acknowledged).

React usage example

```
import { returnZAxisJS } from "./api/backend-api";

onClick(()=>{
   returnZAxisJS();
};)
```

stopZAxisJS()

stopZAxisJS()

This function calls the `stop_z_axis()` on the backend.

Stops the z-axis.

Returns 1 if succeeded (if the function was acknowledged).

React usage example

```
import { stopZAxisJS } from "./api/backend-api";

onClick(()=>{
   stopZAxisJS();
};)
```

moveZAxisMillimetersJS()

moveZAxisMillimetersJS()

This function calls the `move_z_axis_millimeters()` on the backend. Moves the z-axis [distance]mm. This distance is set in MILLIMETERS Returns 1 if succeeded (if the function was acknowledged).

React usage example

```
import { moveZAxisMillimetersJS } from "./api/backend-api";

onClick(()=>{
   moveZAxisMillimetersJS(10);
};)
```

getAvailablePortsListJS()

getAvailablePortsListJS()

This function calls the `get_available_ports_list()` on the backend. Returns a JSON object containing the available COM ports:

JSON

```
{
   "port": x,
   "desc": y,
}
```

React usage example

```
import { getAvailablePortsListJS } from "./api/backend-api";

onClick(()=>{
   getAvailablePortsListJS().then((availablePorts)=>{
      if(availablePorts) console.log(availablePorts);
   });
};)
```

connectToPortJS()

connectToPortJS()

This function calls the `connect_to_port()` on the backend. Connects to a port. The port argument is a string like `COM4`

Returns 1 connection was successful

React usage example

```
import { connectToPortJS } from "./api/backend-api";

onClick(()=>{
   connectToPortJS("COM3");
};)
```