JSONtypes used in this API

- **RECTANGLE:**

```
COORDINATE =
      {
          "up_left":
                {
                  "lat": 51.12356,
                  "long": 16.123456
                },
          "up_right":
                {
                  "lat": 51.12356,
                  "long": 16.123456
                },
          "down_right":
                {
                  "lat": 51.12356,
                  "long": 16.123456
                },
          "down_left":
                {
                  "lat": 51.12356,
                  "long": 16.123456
                },
          "center":
                {
                  "lat": 51.12356,
                  "long": 16.123456
                }
      }

      # Attn. See 'COORDINATE' below.
```

- **COORDINATE:**

```
COORDINATE =
      {
        "lat": 51.12356,
        "long": 16.123456
      }

      # Attn. 'lat' and 'long' are floats.
```

GENERAL

- Back-end will answer requests on `"response"` channel, see data contents for more information.
- Back-end will notify fron-end on `"notify"` channel, see data contents for more information.

  Front-end only have to listen to these two channels and based on the data contents
  decide what to do.

API calls from front-end to back-end:

Connect to back-end

When creating the socket to back-end it will automatically call the `connect` function, which will do the following:

1. Create a new client linked to this connection.
2. Client/this connection will join a `socketio room` linked to a `session`.

This function will then return a `client_id` which is a unique identifier for this client, the `client_id` must be saved since it is required for future calls to the back-end.

- **Event name:**
  `N/A`
- **Data to be sent (JSON format)**
  `N/A`
- **Response**
  - **Channel:**
    `"response"`
  - **Data Content:**

```
{
 "fcn" : "ack",
 "fcn_name" : "connect",
 "arg" :
    {
      "client_id" : "integer(1, -)" # Integer greater than 1.
    }
}
```

- **Example call:**
  `var socket = io.(''http://localhost:8080')`, call to `connect` is performed automatically.

  Listen to response:
  `socket.on("response", function(data){ #process response })`

Check that the connection to back-end is alive

Check so that the connection to the back-end is till alive and working.

- **Event Name**
  `"check_alive"`
- **Data to be sent (JSON format):** `N/A`
- **Response:**
  - **Channel:** `"response"`
  - **Data Content:**

```
{
 "fcn" : "ack",
 "fcn_name" : "check_alive",
}
```

- **Error Response:** No response

- **Example:**
  `socket.emit("check_alive")`

Disconnect from back-end

Disconnect from back-end which in turn disconnects from the RDS.
After this is called communication will come to a halt.

- **Event Name**
  `"quit"`

- **Data to be sent (JSON format)**
  `N/A`

- **Success Response:**

  - **Channel:** `"response"`

  - **Content:**

```
{
 "fcn" : "ack",
 "fcn_name" : "quit",
}
```

- **Example:**
  `socket.emit("quit")`

Define area

Define an area of interest (boundaries).
Must be called after `/function/connect` before back-end will listen to any other instructions.

- **Event Name**
  `"set_area"`

- **Data to be sent (JSON format)**

```
{
  "fcn" : "set_area",
  "arg" :
    {
      "client_id" : "integer(1,-)", # A unique client_id is given in the connect response.
      "coordinates" :
              {
                "waypoint_1" :     # A waypoint is a COORDINATE.
                        {
                          "lat": 51.12356,
                          "long": 16.123456
                        },
                "waypoint_2" :
                        {
                          "lat": 51.12356,
```

```
                        "long": 16.123456
                      },
                  ...           # Allow up to N waypoints.
               "waypoint_N" :
                      {
                        "lat": 51.12356,
                        "long": 16.123456
                      }
               }
      }

}
```

- **Success Response:**
  - **Channel:** `"response"`
  - **Content:**

```
      {
       "fcn" : "ack",
       "fcn_name" : "set_area",
      }
```

- **Example:**
  `socket.emit("set_area", data_to_be_sent)`

Request view

Request images from this area (non prioritized). Back-end will return image ID's which cover specified area (this is to allow front-end to cache images).

`set_area` must be called once before this function is called.

- **Event Name**
  `"request_view"`
- **Data to be sent (JSON format)**

```
{
  "fcn" : "request_view",
  "arg" :
    {
      "client_id" : "integer(1, -)",
      "coordinates" :
                {
                  "up_left":       # It is a COORDINATE.
                        {
                          "lat" : 58.123456,
                          "long":16.123456
                        },
               "up_right":       # It is a COORDINATE.
                        {
                          "lat":59.123456,
                          "long":17.123456
                        },
               "down_left":       # It is a COORDINATE.
                        {
                          "lat":60.123456,
                          "long":18.123456
```

```
                    },
                "down_right":       # It is a COORDINATE.
                    {
                      "lat":61.123456,
                      "long":19.123456
                    },
                "center":       # It is a COORDINATE.
                    {
                      "lat":61.123456,
                      "long":19.123456
                    }
                }
    }

}
```

- **Success Response:**
  - **channel:** `response`
  - **Content:** `json { "fcn" : "ack", "fcn_name" : "request_view", "arg" : { "image_ids" : ["image_id_1", "image_id_2", ..., "image_id_N"] } }`
- **Example:**
  `socket.emit("request_view", data_to_be_sent)`

Request priority view

Request prioritized images from specified area. Back-end will return image ID's which cover specified area (this is to allow front-end to cache images).

`set_area` must be called once before this function is called.

- **Event Name**
  `"request_priority_view"`
- **Data to be sent (JSON format)**

```
{
  "fcn" : "request_priority_view",
  "arg" :
    {
      "client_id" : "integer(1, -)",
      "coordinates" :
                {
                  "up_left":       # It is a COORDINATE.
                        {
                          "lat" : 58.123456,
                          "long":16.123456
                        },
                "up_right":       # It is a COORDINATE.
                        {
                          "lat":59.123456,
                          "long":17.123456
                        },
                "down_left":       # It is a COORDINATE.
                        {
                          "lat":60.123456,
                          "long":18.123456
                        },
                "down_right":       # It is a COORDINATE.
                        {
                          "lat":61.123456,
                          "long":19.123456
                        },
```

```
                "center":       # It is a COORDINATE.
                        {
                          "lat":61.123456,
                          "long":19.123456
                        }
                }
        }

    }
```

- **Success Response:**
  - **Channel:** `"response"`
  - **Content:**
    `json { "fcn" : "ack", "fcn_name" : "request_priority_view", "arg" : { "force_que_id" : "integer(1,-)" } } # Note that prioritized images will arrive later when RDS starts transmitting # images to back-end.`
- **Example:**
  `socket.emit("request_priority_view", data_to_be_sent)`

Clear que of prioritized views.

Clear the queue of previously prioritized views.

- **Event Name**
  `"clear_queue"`
- **Data to be sent (JSON format)**
  `N/A`
- **Success Response:**
  - **Channel:** `"response"`
  - **Content:**

```
      {
       "fcn" : "ack",
       "fcn_name" : "clear_queue",
      }
```

- **Example:**
  `socket.emit("clear_queue")`

Change settings (mode).

Change the mode to `AUTO` or `MAN`

- **Event Name**
  `"set_mode"`
- **Data to be sent (JSON format)**

```
{"fcn" : "set_mode",
 "arg" :
    {
      "mode" : #Choise: "AUTO/MAN",
      "zoom" :  # Can be omitted if mode == "MAN"
```

```
            {
              "up_left":       # It is a COORDINATE.
                    {
                      "lat" : 58.123456,
                      "long":16.123456
                    },
            "up_right":       # It is a COORDINATE.
                    {
                      "lat":59.123456,
                      "long":17.123456
                    },
            "down_left":       # It is a COORDINATE.
                    {
                      "lat":60.123456,
                      "long":18.123456
                    },
            "down_right":       # It is a COORDINATE.
                    {
                      "lat":61.123456,
                      "long":19.123456
                    },
            "center":       # It is a COORDINATE.
                    {
                      "lat":61.123456,
                      "long":19.123456
                    }
            }
        }
    }
```

- **Success Response:**
  - **Channel:** `"response"`
  - **Content:**
    `json { "fcn" : "ack", "fcn_name" : "set_mode", }`
- **Example:**

`socket.emit("set_mode", data_to_be_sent)`

Request image by id.

Request images by id, can request several images at once.
Id is received from calling `functions/request_view` and `functions/request request_priority_view`.

- **Event Name**
  `"get_image_by_id"`

- **Data to be sent (JSON format)**

```
{"fcn" : "get_image_by_id",
 "arg" :
    {
      "client_id" : "integer(1,-)"
      "ids" : ["image_id_1", "image_id_2", ...m "image_id_N"] # List of integers, 1 or more.
    }
}
```

- **Success Response:**

  - **Channel:** `"response"`
  - **Content:**

```
    {
     "fcn" : "ack",
     "fcn_name" : "get_image_by_id",
     "arg":
        {
          "data" :
          [ # ATTN. List of this dict structure:
            {
              "encoded_image_data" : "image (encoded)",
              "type" : #Choise "RGB/IR",
              "force_que_id" : "integer(0,-)",  # 0 means not prioritized.
              "drone_id" : #Choise "one/two/three/...",
              "coordinates" :
                        {
                          "up_left":       # It is a COORDINATE.
                                {
                                  "lat" : 58.123456,
                                  "long":16.123456
                                },
                        "up_right":       # It is a COORDINATE.
                                {
                                  "lat":59.123456,
                                  "long":17.123456
                                },
                        "down_left":       # It is a COORDINATE.
                                {
                                  "lat":60.123456,
                                  "long":18.123456
                                },
                        "down_right":       # It is a COORDINATE.
                                {
                                  "lat":61.123456,
                                  "long":19.123456
                                },
                        "center":       # It is a COORDINATE.
                                {
                                  "lat":61.123456,
                                  "long":19.123456
                                }
                        }

            }
          ]
        }

    }</code></pre></li>
```

- **Example:**

  ```
  socket.emit("get_image_by_id", data_to_be_sent)
  ```

Get info

Get info about drones.

- **Event Name**
  `"get_info"`
- **Data to be sent (JSON format)**
  `N/A`
- **Success Response:**

- **Channel:** `"response"`
- **Content:**

```
{
 "fcn" : "ack",
 "fcn_name" : "get_info",
 "arg" : {"drone-id" : "one",
          "time2bingo" : 15}
}
```

- **Example:**
`socket.emit("get_info")`

API CALLS FROM BACK-END TO FRONT-END:

Notify about new images (including prioritized).

Notifies front-end when new images are sent by the RDS, including prioritized images.

- **Channel front-end listen to:** `"notify"`

- **Data to be sent (JSON format)**

```
{"fcn" : "new_pic",
 "arg" :
    {
      "type" : #Choice "RGB/IR",
      "prioritized" : #Choice: "True/False",
      "image_id" : "integer(1, -)"  # Id of image, which can be requested calling
                                    # functions/get_image_by_id
    }
}
```

- **Success Response:**

  On success, call the following functions.

  - **Event Name**
    `"ack"`

  - **Data to be sent (JSON format)**
    `{"fcn": "ack", "fcn_name" : "new_pic"}`

- **Example:**
`socket.on("notify", function(new_pic_data_is_received_here) {#Do whatever here});`