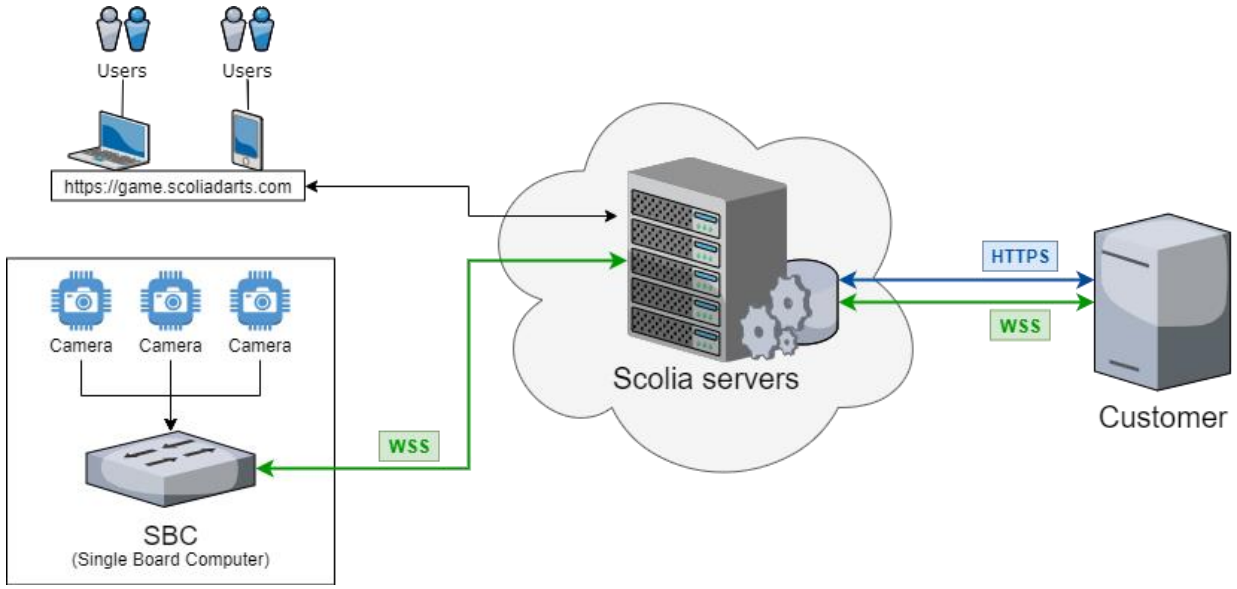
Technical Documentation

for Scolia Social API v1.2

1 Architecture overview

The following figure shows the overall architecture of Scolia. The system consists of two main parts. The first one is the Single Board Computer (hereinafter referred to as “SBC”) which is responsible to handle the hardware (including the cameras) and perform the image processing algorithm locally, then forward every detected information to the Scolia servers through a live *WebSocket* connection. The Scolia servers are responsible for the network communication between the halves, and other services in the cloud. In addition to these, include the databases, and handle the games which are accessible for users in a public domain. The customer is able to develop their own application which is able to use the purchased SBC, through our servers via *WebSocket* connections.



2 Authentication process for SBC

2.1 Authentication overview

The Scolia Social system uses a unified process for authentication based on the serial number of the board, and an access token for third party applications. Without these credentials, the purchased boards cannot be reached and controlled directly by the application of the **Customer** (hereinafter referred to as “you”).

2.2 Authenticating

The third-party app must be authenticated during the communication with the Scolia Social servers. This authentication can be done with the serial number of the board and the given access token.

These values should be provided in *URL query parameters* for the WebSocket API.

3 General information

3.1 SBC status and phase

The SBC is characterized by two distinct properties: status and phase. *Status* refers to the operational state of the SBC, while *phase* refers to a condition that affects the detectable game events. If the SBC is not in the Ready status the phase will be *null* .

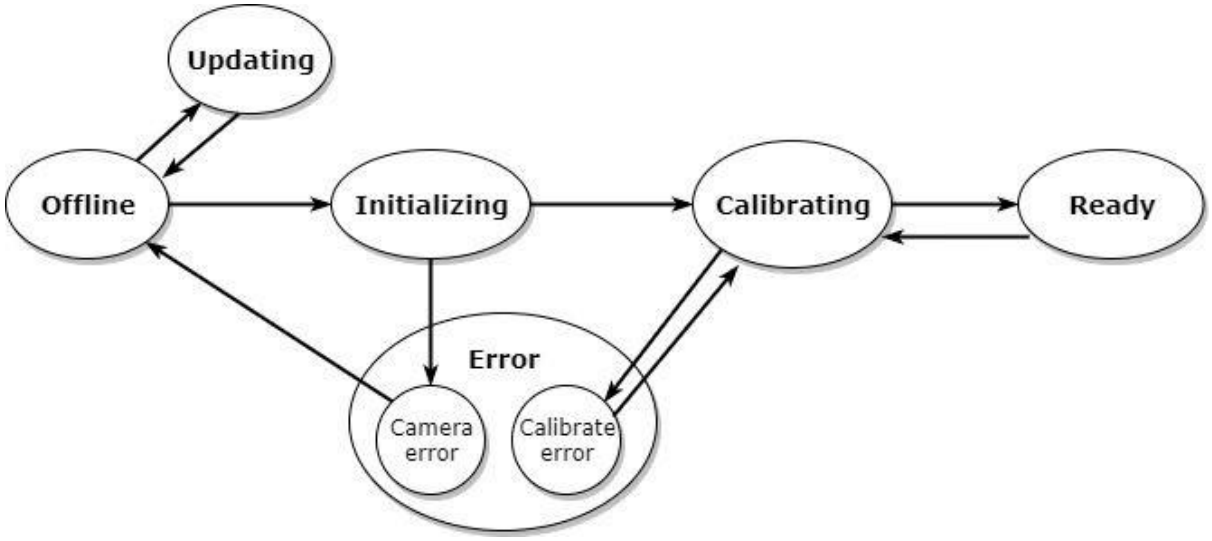
Status conditions:

- **Offline**: The SBC is offline or cannot establish a connection to our servers.
- **Updating**: The SBC received a firmware file from the server.
- **Initializing**: The SBC is initializing, so no message will be sent to it, and no throws will be detected or forwarded to the servers. This is the first status the SBC enters into at start-up.

- **Calibrating**: The SBC is calibrating the cameras, so no message will be sent to it, and no throws will be detected or forwarded to the servers. This is the status the SBC enters into after Initializing.

- **Ready**: The SBC is ready to detect throws and takeout events and forward the corresponding messages to the servers. Normally, after start-up, the SBC reaches the Ready state in about 50-60 seconds.
- **Error**: The SBC has an error, that prevents its normal functionality. No throws will be detected or forwarded to the servers, until the status becomes Ready.

Possible transitions between statuses during starting process in case of intended use:



Phase conditions (*null* if not in Ready status):

- **Throw**: The SBC is in Throw phase, which means it will detect the throws and forward them to the servers.
- **Takeout**: The SBC is in Takeout phase, which means it has detected the start of a Takeout process (removal of the thrown darts from the dartboard) and it is waiting for it to be ended. While being in Takeout phase, the SBC cannot detect any throws.

3.2 WebSocket error codes

These are the error codes the Scolia Social API will use, when closing a connection for some reason.

- **4000**: Pong response did not arrive within certain time.
- **4100**: Invalid serial number has been provided for the connection in the connection URL.
- **4101**: There is another established WebSocket connection with the provided serial number for this board.
- **4102**: Invalid access token has been provided.

4 WebSocket API documentation

4.1 Connecting to WebSocket API

Your application is able to establish one WebSocket connection for each purchased SBC separately (with the right authentication method, through a secure, *WSS* protocol) to communicate with the board in real- time. To open the WebSocket channel with Scolia Social API, you should initialize a WebSocket instance with the given URL, and append the serial number of the board as a query parameter named "`serialNumber`", and the given access token named "`accessToken`".

If you try to open a second WebSocket connection with the same serial number, the connection will be closed immediately with a 4101 error code.

```
// Example in JavaScript
const ws = new WebSocket('wss://game.scoliadarts.com/api/v1/social?serialNumber=SERIAL_NUMBER&accessToken=ACCESS_TOKEN');
```

After the WebSocket connection has been successfully opened, the format of the sent messages in the channel has to follow a strict JSON structure (see below). Every message has two required property in the root. The **`type`** contains the name of the operation, or the occurred event type represented by an uppercase string with underscore word separators.

The **`id`** field is a universally unique identifier (UUID version 4 - according to RFC 4122), which will be generated for the outgoing messages automatically, and required for the incoming messages. This field is mainly used for debugging purposes to clearly identify the incoming message in ACKNOWLEDGED and REFUSED messages.

The **`payload`** property contains the context dependent content of the message. It can be an object representing the message content, or it can be omitted if the message has no specific content to deliver.

```
{
  "type": "MESSAGE_TYPE",
  "id": "UUID",
  "payload": {
     "property": "value"
  }
}
```

4.2 Outgoing messages

These are the messages your application can send to the Scolia Social server through the WebSocket channel. The table below describes the interpretable states when you can send the specific message to the Scolia Social API. In the cases denoted with ✕, you will receive a REFUSED message as a response.

	Offline	Initializing	Calibrating	Ready	Error
GET_SBC_STATUS	✓	✓	✓	✓	✓
GET_CAMERA_IMAGES	✕	✕	✓	✓	✓
RECALIBRATE	✕	✕	✕	✓	✓

RESET_PHASE	✕	✕	✕	✓	✕
THROW_CORRECTED	✕	✕	✕	✓	✕
DELETE_THROW	✕	✕	✕	✓	✕

4.2.1. GET_SBC_STATUS

This message requests the current status and phase of the board.

Responses to this message:

- SBC_STATUS if the message is valid
- REFUSED if the message has invalid id

```
{
  "type": "GET_SBC_STATUS",
  "id": "UUID"
}
```

4.2.2. GET_CAMERA_IMAGES

This message requests the current images of the 3 attached cameras. The camera images will be sent over the WebSocket channel as Base64 (RFC 4648) encoded strings.

Responses to this message:

- CAMERA_IMAGES if the message is valid
- REFUSED if the message has invalid id or the SBC is in Offline or Initializing status

Note that the response to this message has a significantly larger payload, normally around 100 Kb. The request rate of the camera images is limited to 1 request / every 3 seconds by the Scolia Social API.

```
{
  "type": "GET_CAMERA_IMAGES",
  "id": "UUID"
}
```

4.2.3. RECALIBRATE

This message requests a calibration process on the SBC. During the calibration process, the status of the board is "Calibrating". Once the Calibration process is over, the status of the SBC will change to "Ready" or "Error", depending on the success of the calibration process. You will get SBC_STATUS_CHANGED messages about these status changes.

Responses to this message:

- ACKNOWLEDGED if the message is valid
- REFUSED if the message has invalid id or the SBC is in Offline, Initializing or Calibrating status

```
{
  "type": "RECALIBRATE",
  "id": "UUID"
}
```

4.2.4. RESET_PHASE

This message requests the SBC to reset its internal phase to “Throw” and remove any throws for the current round. This message should be typically used if the takeout process has not finished normally. Responses to this message:

- ACKNOWLEDGED if the message is valid
- REFUSED if the message has invalid id or the SBC is not in Ready status

```
{
  "type": "RESET_PHASE",
  "id": "UUID"
}
```

4.2.5. THROW_CORRECTED

This message sends back a corrected throw to the SBC if the user has corrected a throw in the application. The **`throwIndex`** property must contain a number which can be **`0`** , **`1`** or **`2`** representing the sequence number of the corrected throw in the round. Eg. 0 is the first throw in the round, and 2 is the third one.

Responses to this message:

- ACKNOWLEDGED if the message is valid
- REFUSED if the message payload or id is invalid or the SBC is not in Ready status

```
{
  "type": "THROW_CORRECTED",
  "id": "UUID",
  "payload": {
    "throwIndex": 0
  }
}
```

4.2.6. DELETE_THROW

This message should be sent when the application deleted a throw from the current round. (*This is necessary to keep the phase of the SBC in sync with the real state of the board.)* The **`throwIndex`** property must contain a number which can be **`0`** , **`1`** or **`2`** representing the position of the deleted throw in the round. Eg. 0 is the first throw in the round, and 2 is the third one.

Responses to this message:

- ACKNOWLEDGED if the message is valid
- REFUSED if the message payload or id is invalid or the SBC is not in Ready status

```
{
  "type": "DELETE_THROW",
  "id": "UUID",
  "payload": {
    "throwIndex": 0
  }
}
```

4.2.7. CONFIGURE_SBC

With this message you can configure behaviour of the SBC, which belongs to the current WebSocket connection. The SBC can be configured at any time, independent of the status of that.

Responses to this message:

- ACKNOWLEDGED if the message is valid
- REFUSED if the message payload or id is invalid Properties:

- **enableMessageForwardToScolia** is a boolean property which can be used to enable or disable the message forwarding to Scolia application. The value is true by default. If you disable this behaviour, the messages will be delivered only to your server, meanwhile not handled by the Scolia application.

```
{
  "type": "CONFIGURE_SBC",
  "id": "UUID",
  "payload": {
    "enableMessageForwardToScolia": boolean
  }
}
```

4.2.8. GET_SBC_CONFIGURATION

This message requests the current configuration of the SBC.

Responses to this message:

- SBC_CONFIGURATION if the message is valid (see 4.3.10)
- REFUSED if the message is invalid

```
{
  "type": "GET_SBC_CONFIGURATION",
  "id": "UUID",
}
```

4.3 Incoming messages

These are the messages the Scolia Social server can send to your application through the WebSocket channel.

4.3.1. HELLO_CLIENT

This message will be sent to your application when a successfully authenticated WebSocket connection has been opened. It contains the current status, the phase of the board (see 3.1), and optionally include the detailed error type (“Camera” or “Calibrate””) if the status is Error.

```
{
  "type": "HELLO_CLIENT",
  "id": UUID,
  "payload": {
    "boardStatus": "Ready",
    "boardPhase": "Throw",
    "errorType"?: null,
  }
}
```

4.3.2. SBC_STATUS

This message contains the current status, the phase of the SBC (see 3.1), and and optionally include the detailed error type. This is a response to GET_SBC_STATUS message.

```
{
  "type": "SBC_STATUS",
  "id": UUID,
  "payload": {
    "boardStatus": "Error",
    "boardPhase": null,
    "errorType"?: "Camera",
  }
}
```

4.3.3. SBC_STATUS_CHANGED

You will receive this message when a status change happens on the SBC. Note that the phase change of the board does not trigger this message.

```
{
  "type": "SBC_STATUS_CHANGED",
  "id": UUID,
  "payload": {
    "boardStatus": "Calibrating",
    "boardPhase": null,
    "errorType"?: null,
  }
}
```

4.3.4. THROW_DETECTED

This is the most frequently used message, which contains every information about the detected throw. The payload contains information about the thrown sector, coordinates, and further throw-related data. The **`sector`** field is a string, which includes the multiplier (if can be interpreted) and the score of the thrown sector. 's' denotes the single area between the trebles ring and the 25/Bull area, while 'S' denotes the single area between the doubles ring and trebles ring.

Properties:

- **sector** ϵ { ["S"|"s"|"D"|"T"][1-20] | "25" | "Bull" | "None" }
  Regular expression for the field:
  /(([SsDT]{1})(20|1[0-9]|[1-9]))|25|Bull|None/
- **coordinates** is an array of numbers. The first value is the horizontal, and the second is the vertical value. The values are in the **-250** and **+250** interval, in units of millimeter.
- **angle** is an object with the horizontal and vertical angle of the thrown darts. The interpretable interval for the values is between **-90** and **+90** in units of degree.
- **bounceout** is a boolean property indicating whether the dart has landed outside the dartboard (either bounced out or landed on the surround).
- **sectorSuggestions** array may contain the nearest possible sector names, which can be used as suggestions for score correction in case of an erroneous detection. The length of this array is between 0 and 3.
- **detectionTime** is the UTC time of the detection, which represents a simplified extended ISO format in string (ISO 8601).

```
{
  "type": "THROW_DETECTED",
  "id": UUID,
  "payload": {
    "sector": "S1",
    "coordinates": [40, 138],
    "angle": {
       "vertical": 82.1234,
       "horizontal": 89.0012
    },
    "bounceout": false,
    "sectorSuggestions": ["S20", "T20", "T1"],
    "detectionTime": "2019-10-05T14:48:14.000Z"
  }
}
```

4.3.5. TAKEOUT_STARTED

The message is triggered when the user gets into the field of view of the cameras during a darts takeout action.

```
{
  "type": "TAKEOUT_STARTED",
  "id": UUID,
  "payload": {
    "time": "2019-10-05T14:48:14.000Z"
  }
}
```

4.3.6. TAKEOUT_FINISHED

The message is triggered when the user finishes the takeout process, and the field of view of the cameras is clear again. After this message is sent, the SBC is ready to detect the upcoming throws. The **`falseTakeout`** property means that the user got into the field of view of the cameras, but has not removed the darts from the dartboard.

```
{
  "type": "TAKEOUT_FINISHED",
  "id": UUID,
  "payload": {
    "falseTakeout": false,
    "time": "2019-10-05T14:48:14.000Z"
  }
}
```

4.3.7. CAMERA_IMAGES

This message contains the three camera images of the SBC as Base64 (RFC 4648) encoded JPEG images. This message is a response to GET_CAMERA_IMAGES.

Note that this message has a significantly larger payload, normally around 100 Kb. The request rate of the camera images is limited to 1 request / every 3 seconds by the Scolia Social API.

```
{
  "type": "CAMERA_IMAGES",
  "id": UUID,
  "payload": {
    "images": [
      "[...]fqxQ//9k=",
      "[...]fqxQ//9k=",
      "[...]fqxQ//9k="
    ]
  }
}
```

4.3.8. ACKNOWLEDGED

This is a general return message indicates that your previous message has been accepted by the server, but no specific response exists to that message, such as RESET_PHASE. The **`replyTo`** property refers to the message id of the message that the server has acknowledged.

```
{
  "type": "ACKNOWLEDGED",
  "id": UUID,
  "payload": {
    "replyTo": UUID
  }
}
```

4.3.9. REFUSED

This is a general return message which indicates that your previous message has been rejected by the server. This can be a validation error when you send invalid payload or message, or can be a conflict with the current SBC state. For instance

you cannot send a THROW_CORRECTED message while the SBC is not in Ready state.

The specific error message will be sent back to your application in the payload property of the message. The **`replyTo`** property refers to the message id of the message that the server has refused.

```
{
  "type": "REFUSED",
  "id": UUID,
  "payload": {
    "replyTo": UUID,
    "error": "ErrorMessage",
    "errorMessage": "Human readable error message."
  }
}
```

4.3.10. SBC_CONFIGURATION

This message contains the configuration properties belonging to the current SBC. It is the response message for the GET_SBC_CONFIGURATION (see 4.2.8) request.

```
{
  "type": "SBC_CONFIGURATION",
  "id": "UUID",
  "payload": {
    "enableMessageForwardToScolia": true
  }
}
```

5 REST API documentation

5.1 Authentication

The API uses the HTTP Bearer authentication strategy which authenticates requests based on a bearer token contained in the:

- Authorization header field where the value is in the format {scheme} {token} and scheme is "Bearer" in this case.
- or access_token body parameter
- or access_token query parameter

5.2 API endpoints

5.2.1 Get all the boards connected to the account.

```
GET https://game.scoliadarts.com/api/social/boards
```

Response:

- 200: Array of connected boards

```
[
  {
    "name": "board name",
    "serialNumber": "serial number",
    "isHomeSbc": false
  },
  …
]
```

5.2.2 Connect a board to the account.

```
PUT https://game.scoliadarts.com/api/social/boards
```

Payload:

```
{
    "serialNumber": "SERIAL_NUMBER"
}
```

Response:

- 200: board connected

```
{
  "name": "board name",
  "serialNumber": "serial number",
  "isHomeSbc": false
},
```

- 409: board has already been connected

```
{
    "error": "Board has already been connected."
}
```

- 404: invalid serial number

```
{
    "error": "Invalid serial number: SERIAL_NUMBER"
}
```

5.2.3 Disconnect a board from the account.

```
DELETE https://game.scoliadarts.com/api/social/boards/SERIAL_NUM
```

Response:

- 200: board deleted
- 404: invalid serial number

```
{
    "error": "Board is not connected to this service
              account."
}
```