MySkiTool Socket.IO API Guide

A. Connecting to the Server

To connect to the MySkiTool SocketIO server, use:

Python:
```
sio.connect('https://skilaketech.myskitool.com:5443')
```

B. SocketIO Structure

Event Types:
- connect
- connect_error
- disconnect

Interaction Types:
- message
- response

Refer to official SocketIO docs for details on handling events.

C. Message Format

Example Message:
```
{
  "event": "EXCHANGE",
  "source": "MST_Tracker",
  "target": "ShortlineBM1",
  "command": "stop",
  "comment": "38.6966124,-121.0537761",
  "skierName": "PaulCrawford",
  "skierPass": "",
  "skierSpeed": "",
  "skierRope": ""
}
```

Key Descriptions:
```
event     - Message category (EXCHANGE, 25W077, etc.)
source    - Message origin (e.g., MST_Tracker)
target    - Intended recipient (e.g., ShortlineBM1)
command   - e.g., start, stop, ping, record, etc.
comment   - Free text (location, notes)
skierName, skierPass, skierSpeed, skierRope - Metadata
```

D. Getting Started

1. Install SocketIO library (Python):
```
   pip install "python-socketio[client]"
```

2. Connect to the server and set up listeners.

MySkiTool Socket.IO API Guide

Python Sample:

```
import socketio

sio = socketio.Client()

@sio.on('connect')
def on_connect():
    print("Connected")

@sio.on('message')
def on_message(data):
    print("Received:", data)

sio.connect('https://skilaketech.myskitool.com:5443')
sio.emit('message', {
    'event': 'EXCHANGE',
    'source': 'ClientApp',
    'target': 'ShortlineBM1',
    'command': 'ping',
    'comment': 'Test connection',
    'skierName': '',
    'skierPass': '',
    'skierSpeed': '',
    'skierRope': ''
})
```

JavaScript Sample (Node.js):

```
const io = require("socket.io-client");
const socket = io("https://skilaketech.myskitool.com:5443");

socket.on("connect", () => {
  console.log("Connected");
});

socket.on("message", (data) => {
  console.log("Received:", data);
});

socket.emit("message", {
  event: "EXCHANGE",
  source: "ClientJS",
  target: "ShortlineBM1",
  command: "ping",
  comment: "Test connection"
});
```

Kotlin (Android):

MySkiTool Socket.IO API Guide

```
val socket = IO.socket("https://skilaketech.myskitool.com:5443")
socket.on(Socket.EVENT_CONNECT) {
    Log.d("SocketIO", "Connected")
}
socket.on("message") { args ->
    Log.d("Message", args[0].toString())
}
socket.connect()
socket.emit("message", JSONObject().apply {
    put("event", "EXCHANGE")
    put("source", "AndroidApp")
    put("target", "ShortlineBM1")
    put("command", "ping")
})

Java (Android):
-----------------
IO.Options options = new IO.Options();
Socket socket = IO.socket("https://skilaketech.myskitool.com:5443", options);

socket.on(Socket.EVENT_CONNECT, args -> Log.d("SocketIO", "Connected"));
socket.on("message", args -> Log.d("Message", args[0].toString()));
socket.connect();

JSONObject message = new JSONObject();
message.put("event", "EXCHANGE");
message.put("source", "JavaApp");
message.put("target", "ShortlineBM1");
message.put("command", "ping");
socket.emit("message", message);

iOS (Swift):
------------
let           manager           =           SocketManager(socketURL:           URL(string:
"https://skilaketech.myskitool.com:5443")!, config: [.log(true), .compress])
let socket = manager.defaultSocket

socket.on(clientEvent: .connect) {data, ack in
  print("socket connected")
}

socket.on("message") { data, ack in
  print("Received: \(data)")
}

socket.connect()

let message: [String: Any] = [
  "event": "EXCHANGE",
  "source": "iOSApp",
```

```
  "target": "ShortlineBM1",
  "command": "ping"
]
socket.emit("message", message)
```

E. iOS with Expo Go (React Native / JavaScript)

```
1. Install Dependencies (Expo Go Compatible):
   npm install socket.io-client
   npm install react-native-websocket

Note: Expo Go has built-in WebSocket support, so no native linking needed.

2. Sample Code (React Native with Expo):
----------------------------------------
import React, { useEffect } from 'react';
import { View, Text } from 'react-native';
import io from 'socket.io-client';

export default function App() {
  useEffect(() => {
    const socket = io('https://skilaketech.myskitool.com:5443', {
      transports: ['websocket'], // important for Expo Go
    });

    socket.on('connect', () => {
      console.log('Connected to server');

      // Send a ping test message
      socket.emit('message', {
        event: 'EXCHANGE',
        source: 'iOS_ExpoApp',
        target: 'ShortlineBM1',
        command: 'ping',
        comment: 'Testing from Expo Go',
      });
    });

    socket.on('message', (data) => {
      console.log('Received message:', data);
    });

    socket.on('disconnect', () => {
      console.log('Disconnected from server');
    });

    return () => {
      socket.disconnect();
    };
  }, []);
```

```
  return (
    <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
      <Text>MySkiTool Expo Go Client</Text>
    </View>
  );
}
```

3. Notes:
 - Use transports: ['websocket'] to ensure reliable connection.
 - HTTP long-polling may not work well in Expo Go.
 - No native modules or ejecting required - fully JS-based.