# Spider Protocol

This file contain the protocol for the project Spider. The following protocol must be followed in both parts of the project, server and client.

# Connection initialization

## Connection opening and accepting

The client must open a secure connection to the server, using the following criteria:

- The communication MUST be using TLS 1.2 standard
- You must sign your certificate with your own certification authority
- Your key must be at least 2048 bits
- You MUST use a Diffie Hellman exchange key
- You will only accept these following ciphers:

> ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256

> **Notes:**
>
> Your key should use strong algorithms such as ECDHE and respect the Forward Secrecy

## Client accepting

When the connection to your server is accepted, these steps are essentials to etablish the program running as well.

- You MUST send machine informations to the server in this JSON formatting:

```
{
    "os": "",
    "mac": "",
    "antivirus": ""
}
```

Where the key 'os' will contain the operating system of the client, mac will contain the physical address of the client and antivirus will contain the process name of the client antivirus.

You will have to generate an unique identifier with these informations, and store it in your database.

# Logging

Once the client is accepted you will have to log all his keystroke and mouse movement and click on position x and y.

- You MUST send the logs to the server in json format like this for keystroke:

```
{
    "timestamp": "",
    "process": "",
    "data": "",
    "type": "KEYSTROKE"
}
```

and like this for mouse movement or clicks:

```
{
  "timestamp": "",
  "process": "",
  "data": {
    "click": "",
    "x": 0,
    "y": 0
  },
  "type": "MOUSE"
}
```

Timestamp will handle the date where the logs was started, Process will handle the form where the input was. Data will be a string containing the keystroke, or a subdocument containing the click type and x/y positions. click type is a string enum:

> None
>
> RC
>
> LC
>
> SCROLLUP
>
> SCROLLDOWN
>
> MC

> **Notes:**
>
> RC stands for Right click
>
> LC stands for Left click
>
> MC stands for Middle click
>
> SCROLLUP stands for mouse wheel going up
>
> SCROLLDOWN stands for mouse wheel going down

# Ping

Client need to send a ping to the server to say he is alive. The ping must follow the following format:

```
{
  "type": "PING",
  "data": ""
```

```
}
```

You have to send the local timestamp to the server.

> **data** is the local timestamp of the machine (in unix timestamp), this would help the server to determine the latency
>
> **Example:**
>
> 1506433136
>
> Stands for **Tuesday, September 26, 2017 1:38:56 PM**

# Pong

By the way, server need to answer to the server to say the connection is ok. The pong must follow the following format:

```
{
  "type": "PONG",
  "data": ""
}
```

You have to send the local timestamp to the server.

> **data** is the local timestamp of the machine (in unix timestamp), this would help the server to determine the latency
>
> **Example:**
>
> 1506433136
>
> Stands for **Tuesday, September 26, 2017 1:38:56 PM**

# Server to Client commands

The server can send some commands to the client using this format:

```
{
  "type": "",
  "data": ""
}
```

**If your client don't support the command, he must ignore it.**

> **type** is a string specifying the type of the action you want to run on the client.
>
> **data** is a string or a subdocument you can design how you want in your implementation

# Awnser from the client about sent commands

**This part is not mandatory and depends on what command you implemented**

# Remote execution

If the command is a remote shell/powershell this json must be sent by the client respecting this format.

```
{
  "type": "command",
  "timestamp": "",
  "command": "",
  "stdout": "",
  "stderr": "",
  "exitcode": 0
}
```

> **timestamp** is a string and stand for the time where the execution finished
>
> **command** is the full command the client executed
>
> **stdout** is what the command wrote on stdout
>
> **stderr** is what the command wrote on stderr
>
> **exitcode** is the return/exit status of the command, in integer

**If you have any questions about any parts of this document, please ask Julien HOUZET**