Collab-Hub Max Client 0.3.1

Overview

The Collab-Hub Max Client is a tool designed to faciliate communication with the Collab-Hub server from within Cycling 74's Max. This client will help you send/receive data over the internet between Max and/or other platforms.

The Max Client package consists of a set of modules (built as abstractions that run inside bpatchers) that can be used in different combinations depending on your needs.

The Collab-Hub Max Client is based on NodeForMax and Socket.IO.

Getting Started

1. If you've downloaded this package from Max Package Manager, skip to step #3.
2. Download the Collab-Hub-Max-Client folder from GitHub using the green Code button near the top of this page. Either choose "Download ZIP" or, if you're comfortable using git, you can clone the repo.
3. Add the Collab-Hub-Max-Client folder as an entry in your File Preferences in Max. Make sure that the "Subfolders" checkbox is checked.
4. Open the CH-Max-Demo Max patch for a quick introduction and/or watch the Getting Started video below.

DEMO VIDEO COMING SOON

Demo Patch

Modules

Each of the modules is designed to implement a specific function or group of related functions of Collab-Hub. They can be mixed and matched as needed. Each module has an inlet and outlet that can be used to send and receieve messages

to/from the server. Alternatively, outgoing messages can be connected to a *send toCH-Server* object, while incoming messages can be intercepted with a *receive fromCH-Server* object.

CH-Client

The **CH-Client** module is the bare minimum needed to connect to Collab-Hub. It is possible to build a setup using only this module if you do not require the functions of the other modules.

- **Change Username:** Click to change username from the automatically generated username (e.g. User000) to something else.
- **Connect:** Toggle the Connect button to connect/disconnect from the Collab-Hub server.
- **Connect Status:** The LED next to the **Connect** button will light up to confirm a successful connection.
- **Flags:** Toggling the **Flags** button on/off determines whether or not incoming controls and events are prepended with a username flag (e.g. for routing purposes).
- **Simple Chat:** As the name implies, this is a text box for sending chat messages to all othe users.
- **Users Display:** This window displays the usernames of all connected users.

CH-Chat

The **CH-Chat** module provides extended chat functionality. Incoming chat messages post to the Max console window.

- **all:** Chat messsages are sent to all users.
- **user:** Chat messages are sent to one specific user (dropdown menu with usernames is provided below the text box).
- **room:** Chat messages are sent to users in a room (dropdown menu with room names is provided below the text box).

CH-Rooms

The **CH-Rooms** module provides the ability to create, join, and leave rooms on the server. Rooms are a way to create groups of users that can receive controls, events, or chat messages directed only to them.

- **AvailableRooms:** A display that lists all rooms available to join (rooms you are not already in).
- **Create Room:** Click to create a new room (you will be prompted to provide a room name).
- **Details:** A toggle that switches between a minimal view (room names only) and detailed view (room names with all room members listed) in the AvailableRooms and MyRooms displays.
- **Join:** Select a room name from the dropdown menu to join an existing room that you are not already in.
- **Leave:** Select a room name from the dropdown menu to leave a room that you are currently in.
- **MyRooms:** A display that lists all of the rooms that you are currently in.

CH-Controls

The **CH-Controls** module provides a way to see and manage your control headers and choose controls to observe. This module is only really useful when sending/receiving controls using the **publish** mode.

- **AvailableControls:** A display that lists all of the available published controls (ones that you are not already observing).
- **Clear:** Choose one of your published control headers from the dropdown menu to clear it from the server.
- **Details:** A toggle that switches between a minimal view (control headers only)

and detailed view (control headers with all available info listed, including sender, observers, and format of values) in the AvailableControls, ObservedControls, and MyControls displays.
- **MyControls:** A display that lists all of the control headers that you have published to the server.
- **Observe:** Choose an available control header from the dropdown menu to start observing (you will start receiving this control data).
- **ObservedControls:** A display that lists the controls you are currently observing.
- **Stop:** Choose an available control header from the dropdown menu to stop observing (you will no longer receive this control data).

CH-Events

The **CH-Events** module provides a way to see and manage your event headers and choose events to observe. This module is only really useful when sending/receiving events using the **publish** mode.

- **AvailableEvents:** A display that lists all of the available published events (ones that you are not already observing).
- **Clear:** Choose one of your published event headers from the dropdown menu to clear it from the server.
- **Details:** A toggle that switches between a minimal view (event headers only) and detailed view (event headers with all available info listed, including sender and observers) in the AvailableEvents, ObservedEvents, and MyEvents displays.
- **MyEvents:** A display that lists all of the event headers that you have published to the server.
- **Observe:** Choose an available event header from the dropdown menu to start observing (you will start receiving this event).
- **ObservedEvents:** A display that lists the events you are currently observing.
- **Stop:** Choose an available event header from the dropdown menu to stop observing (you will no longer receive this event).

Message Formats

Outgoing data

All outgoing control and event communications between users on Collab-Hub operate in the following modes:

- **Publish** - Published controls/events are automatically registered to the server but will only be sent to users that choose to "observe" them. Their availablity is advertised in the AvailableControls and AvailableEvents displays in the CH-Controls and CH-Events modules, respectively.
- **Push** - Pushed controls/events are sent directly to the intended targets. They are not advertised in AvailableControls or AvailableEvents.

You may decide to use these two modes in the following example scenarios:

- If you have a Max patch that generates a lot of control data (e.g. using several LFOs) and you want to have an impromptu jam with some friends over the internet, you would **publish** those controls so that other users can selectively receive and map that data *à la minute*.
- If you are composing a piece with Max for laptop ensemble that has a known set of parameters and/or performers, you probably want to **push** controls and events since they will likely be routed and mapped the same way for each performance.

The max client expects to recieve outgoing control data in the following format:

```
mode target[if push] header value(s)
```

...and outgoing event data in the following format:

```
mode target[if push] header
```

For example, a valid published control message would look like this:

```
publish slider 5
```

...while a valid pushed event would look like this:

```
push all bang
```

For **pushed** controls/events, the target should be a username, room name, or the word 'all' (sent to everyone).

Incoming data

Incoming controls/events will be in this format (for controls):

```
header value(s)
```

...or this format (for events):

```
header
```

Optionally, if the *Flags* button is enabled in the **CH-Client** module in the receiver's patch, the sender's username will be prepended to all incoming controls and events:

```
sender header value(s)
```

...or:

```
sender header
```

This makes it easy in Max to route incoming data (e.g. using the *route* or *select* objects) by header and/or the sender's username.

Other Commands

Much of the functionality below is built into dropdown menus, buttons, and toggles of the modules. However, you can also implement them manually in your patch if you prefer by sending the following commands to any module inlet or a *send toCH-Server* object.

Change username:

```
addUsername 'username'
```

Toggle sender flags (prepends the sender's username to all incoming controls/events):

```
sender '0/1'
```

Toggle detailed view on/off for controls displays:

```
controlDetail '0/1'
```

Toggle detailed view on/off for events displays:

```
eventDetail '0/1'
```

Toggle detailed view on/off for room displays:

```
roomDetail '0/1'
```

Send chat messages (use 'all' for target to send to everyone):

```
chat 'target' 'message'
```

Start observing a published control:

```
observeControl 'header'
```

Stop observing a published control:

```
unobserveControl 'header'
```

Start/stop observing all published controls:

```
observeAllControl '0/1'
```

Clear a published control header from the server:

```
clearControl 'header'
```

Start observing a published event:

```
observeEvent 'header'
```

Stop observing a published control:

```
unobserveEvent 'header'
```

Start/stop observing all published events:

```
observeAllEvents 'header'
```

Clear a published event header from the server:

```
clearEvent 'header'
```

Join a room (will create the room if it does not already exist):

```
joinRoom 'roomname'
```

Leave a room:

```
leaveRoom 'roomname'
```

Going Further

The Client Script

The file called 'CH-ClientScript-vX.js' is the client script that loads in the *node.script* object inside the CH-Client module. This script is all that is really needed to communicate with the Collab-Hub server from within Max. Experienced users may wish to build their patches around just a *node.script* object running 'CH-ClientScript-vX.js', mitigating the need for the provided modules.

Initialization with config.json

Users can initialize some settings at the time of connection by changing entries in the 'config.json' file.

```
{
    "namespace": "hub",
    "username": ""
}
```

- **namespace:** The default Collab-Hub collective namespace is "hub". This should not be changed unless you are provided with a namespace by the Collab-Hub developers. Contact us if you think you or your group may have a need for your own namespace.
- **username:** By default, a random username is assigned in the format 'User000' if this entry is empty. You can have your username set to something else at the time of connection by entering it here (assuming that the username is not already in use by someone else). Usernames can always be changed after connection from within Max.