Horus System V2 Socket.io

Setup

First download the latest Horus Linking Lab from http://embed.horus.nu/. Start Horus Linking Lab (from here on referenced as HLL). This can take a few seconds, because the software loads all available components. When the HLL screen appears, select 'This system' (this should be the default) and click 'Connect'.
Extract the basic-server.tar.gz, change to the extracted directory and run '`npm install`', followed by '`npm start`'. This is the socket.io server that also runs on the Picostreamer. There is a basic web interface included to test messages and see responses from the system. The socket.io server listens on port 3105 and does not need authentication.

Minimal pipeline

In HLL, open the Graph Builder. On the left side there is a list of components, on the right is your canvas. At the top of the screen are buttons to get, set and start/stop the pipeline, save the pipeline to the remote device (not used in this guide) and to start and stop a recording.
Search for the Socket.io component and drag it onto the canvas. You can filter the components using the text field above the list to find it more easily.
At the top, first click `SET` to send the pipeline to the backend, then click the play button to start it. Open the Socket.io web page in your browser at http://127.0.0.1:3105/. At the bottom are inputs to sent control and data messages. In the control input, type PING and press enter or click the 'Send ctrl' button. At the top you should see a reply from your Horus System V2 prefixed by '**[ctrl]**'.
The commands you can send are defined in the `horus.pb.controlmessages.v1.MessageType` enum in the `horus-types` package.

Adding a sensor

Back in the Graph Builder, look for the Timer component. Drag it onto the canvas and select it. On the right side of the window there should appear a list of Properties for the selected component. Set the 'Interval Duration' to 2 (seconds, fractional values are allowed), 'Output Message Type' to 'sensor' and 'Payload' to any number you like. Now back on the canvas, the Timer component should have a circle at the top right. Drag this circle onto the Socket.io component to connect them. Then SET and start the pipeline and switch back to your browser to see a message prefixed with '**[data]**' every two seconds.

Adding a toggle

Next, add a toggle component and drag the ‘data 1’ output of the Socket.io component to the toggle and the ‘Both’ output back to the Socket.io component. Set and start the pipeline and in the browser enter the following JSON in the ‘Data message’ field:

```
{"SourceId":"PicoStreamer-client","Commands":[{"MyType":100,"horus.pb.container.v1.commands.PTZ.ptzCommand":{"Id":"Toggle","ButtonIdPressed":["Toggle"]}}]}
```

Press Enter or click ‘Send data’ and you should get a reply with the new toggle status. Look for the `Sensors[].DoubleValue` field in the JSON message. Every time you send that message, the value should toggle between 0.0 and 1.0.

JSON explanation

Base message

Key	**Description**
SourceId	This identifies where the message came from. When you receive a message from the backend, this is the name of the component the message originates from.
Commands	Array of commands to send. You can send multiple commands to multiple components at the same time, as long as they are connected to the same output of the Socket.io component

Command

Key	**Description**
MyType	The command type. Defined in `horus.pb.container.v1.commands.Type`
horus.pb.container.v1.commands.PTZ.ptzCommand	This is a PTZ command, normally used for Pan-Tilt-Zoom controllers, but can also be used for Toggles

horus.pb.container.v1.commands.PTZ.ptzCommand

Key	Description
Id	The Id of the component you want to send the command to
ButtonIdPressed	When a component has multiple buttons, this identifies which to activate. The toggle component responds to any value.

Retrieving the pipeline

The control message GET_PIPELINE retrieves the running pipeline from the backend. Send this command and analyse the JSON returned. This contains all of the components in your pipeline, their properties and how they are connected.

Pipeline connection statistics

With the GET_CONNECTION_STATISTICS command you can get statistics about data flow between components. This returns the packets (messages) per second and bytes per second transferred between components.

Next steps

In HLL open the Component Browser and read the descriptions of the components to learn more about them and their properties. Try to add other components to the pipeline and