LEXICON DD8P

Control Protocol documentation

OVERVIEW

The control / monitoring of the DD8P device is done over TCP using the HTTP protocol to send / receive JSON packets. This protocol is primarily implemented to support the embedded HTTP server for use with a standard web browser, but non-browser clients can be developed to send / receive the HTTP requests / responses for control of the device, and requests can be sent with tools such as cURL. The configuration webpage gives a working example of the control protocol, which may be viewed using the developer tools built into the Chrome or Firefox web browsers for monitoring network activity.

The embedded webserver supports only one connection at a time. To improve performance, it supports and is intended to use persistent connections, where a TCP connection is kept open for multiple requests. It will drop a connection a few seconds after the last activity, and so must be polled once a second or so to keep a connection open.

HTTP SPEC

This section documents the HTML request / response packets with regards to the DD8P, as the HTTP protocol defines capabilities not used by the DD8P. The JSON structures will be discussed in later sections of this document.

HTTP STRUCTURE

The basic format structure of the HTTP request / response messages are similar and consist of the following:

- <Initial Line *Note: different for requests vs responses>*
- Zero or more header lines
- A blank line terminated with a carriage return/line feed sequence (\r\n)
- <Optional message body>

Note: The header and message body is always separated by a blank line, and all lines in the header end with a CRLF. HTTP 1.0 defines 16 headers, though none are required. HTTP 1.1 defines 46 headers, and one (Host:) is required in requests. For the purpose of controlling the DD8P the Content-Length header is also required, to define the length of the body.

HTTP GET REQUEST

GET requests are used to request information from the device, and consist of the GET command, the URI (Uniform Resource Identifier) of the information to get, and the HTTP version, followed by the headers and a blank line terminated by \r\n:

Example Request to get the STATUS from the DD8P: (Note: the DD8P IP address is 192.168.50.4)

```
GET /status HTTP/1.1\r\n
HOST: 192.168.50.4\r\n
Content-Type: application/json\r\n
\r\n
```

HTTP Response

The initial line of the response is of the form:

HTTP/1.1 200 OK\r\n

Common initial lines for response messages:

- 200 OK
- 404 Not Found
- 301 Moved Permanently
- 302 Moved Temporarily
- 500 Server Error

Example Response to the get STATUS from the DD8P: (Note: the DD8P IP address is 192.168.50.4)

```
HTTP/1.1 200 OK\r\n
Server: Lexicon Embedded Webserver\r\n
Connection: keep-alive\r\n
Access-Control-Allow-Origin: *\r\n
Access-Control-Allow-Methods: GET, POST, PUT\r\n
Content-Type: application/json\r\n
Content-Length: 92\r\n
\r\n
{"cfg_change_ctr":0,"sig_det":[0,0,0,0,0,0,0,0],"sig_clip":[0,0,0,0,0,0,0,0],"trigger_in":0}
```

The important parts of this above message are the initial line, which indicates that the request was successful. And the lines "Content-Type" and "Content-Length", which indicate that the response data is in JSON format and is 92 bytes long. ***Please Note: all messages will be in JSON format.***

HTTP Post

POST requests are used for sending data to the device, and consist of the POST command, the URI (Uniform Resource Identifier) of the information to send, and the HTTP version, followed by headers including at least the Content-Length header, a blank line terminated by \r\n, and a message body containing the JSON message:

Example POST to set setting "auto-powerdown" on the DD8P: (Note: the DD8P IP address is 192.168.50.4)

```
POST /settings HTTP/1.1\r\n
HOST: 192.168.50.4\r\n
Content-Type: application/json\r\n
Content-Length: 20\r\n
\r\n
{"auto_powerdown":1}
```

The important parts of this above message are the initial line, which indicates a POST message. And the lines "Content-Type" and "Content-Length", which indicate that the response data is in JSON format and is 20 bytes long. ***Please Note: all messages will be in JSON format.***

POST requests will respond with a message containing the cfg_change_ctr, which can be tracked to detect changes by other device controllers.

JSON Formats

JSON (JavaScript Object Notation) is a text format similar to XML however is simpler and lighter weight.

JSON Syntax Rules

- Data is in name/value pairs
- Data is separated by columns
- Curly braces denote "objects" consisting of name/value pairs
- Square brackets denote arrays of values

Data

The rules for the name/value pairs are field name (always in double quotes), followed by a colon, followed by a value. Possible values:

- A number (integer or floating point)
- A String (always double quoted)
- Boolean (true/1 or false/0)
- An array (in square brackets comma delimited)
- An object (in curly braces)
- Null

An example of a simple JSON object:

{"DeviceName": "DD8P-01234", "IP":[192, 158, 0, 1], "Debug":true}

The above JSON text defines an object with three data elements as follows:
(note: this is only an example).

1.	**Device Name = DD8P-01234**	**(string)**
2.	**IP = 192.168.0.1**	**(an array holding the IP address octets)**
3.	**Debug = true**	**(Bool flag)**

DD8+ Communication

To retrieve information the client will send a HTTP GET request to a specific path (defined later in this document). The DD8P will respond with an HTTP response packet with the JSON message in the body. To change a setting on the device, the client will send an HTTP POST requests with the body containing the JSON object with the new settings.

Any subset of the settings for a particular URI may be set in a single POST request, however, the message must be less than 512 bytes long.

Info

URI	Description
/info	General fixed device information. These values are read-only, this URI will not respond to POST requests.

```
{"device_type":"Lexicon DD8+","device_name":"DD8P-
000495","mac":"00:1c:e2:00:04:95","ip":"192.168.50.4","app_version":"1.0","b
oot_version":"1.0","build":"d75fd44bb5e84d33c6fbec77475f892459d3e857"}
```

Data Name	Data Type	Definition
`device_type`	String	“Lexicon DD 8+”
`device_name`	String	Name of the DD8P device
`mac`	String	Format: “xx:xx:xx:xx:xx:xx:”
`ip`	String	Current IP address assigned Format: “aaa.bbb.ccc.ddd”
`app_version`	String	Format: “x.y”
`boot_version`	String	Format: “x.y”
`build`	String	Git hash identifying the firmware build

Settings

URI	Description
/settings	Device-wide settings not part of a preset.

```
{"device_name":"DD8P-
000495","use_static_ip":"0","static_ip":"192.168.50.4","netmask":"255.255.25
5.0","gateway":"192.168.50.1","preset_settings_modified":1,"loaded_preset":0
,"config_name":"Default","preset_names":["Preset 1","Preset 2","Preset
3"],"auto_powerdown":1,"green_mode":1}
```

Data Name	Data Type	Definition
`device_name`	String	Name of the DD8P device. If the network supports it, this is handed to the DHCP server to allow identifying the device on the network by name instead of IP address.
`use_static_ip`	Boolean	If true, the device uses the static IP instead of attempting to obtain an IP via DHCP.
`static_ip`	String	Format: “aaa.bbb.ccc.ddd”
`netmask`	String	Format: “aaa.bbb.ccc.ddd”
`gateway`	String	Format: “aaa.bbb.ccc.ddd”
`preset_settings_modified`	Boolean	True if the current settings have been modified since loading or saving a preset. This value is read-only.
`loaded_preset`	Integer	0-3: Index of the loaded preset, 0 for default settings.

`config_name`	String	Name of the current configuration. No more than 12 characters.
`preset_names`	Array of strings	Names of the preset configurations
`auto_powerdown`	Boolean	0 = Off 1 = On
`green_mode`	Boolean	0 = Off 1 = On

STATUS

URI		Description
/status		Retrieve status values that may change at any time. These are given a separate resource identifier to minimize overhead in polling. These values are read-only, this URI will not respond to POST requests.
`{"cfg_change_ctr":484,"sig_present":[1,0,0,1,0,0,0,0],"sig_clip":[0,0,0,0,0,0,0,0],"trigger_in":0}`		
Returned JSON object		
Data Name	Data Type	Definition
`cfg_change_ctr`	Integer	Running count of configuration changes. Monitor this to detect changes in configuration by the web UI or other control applications.
`sig_present`	Array of Booleans	The values in this array correspond to the signal detector blocks connected to each output in the DSP.
`sig_clip`	Array of Booleans	The values in this array correspond to the clip indicator signals from the amplifier module.

INPUT

URI		Description
/input/INPUT_IDX		Settings for a specific input. INPUT_IDX ranges from 0 to 18: Inputs 0-7 are analog inputs 1-8. Inputs 8-15 are digital inputs 1-4 (2 input channels per physical input). Inputs 16 and 17 are the bus analog inputs. Input 18 is an internal pink noise source.
`{"input_id":0,"name": "ANALOG 1","input_mix":[1.000000,0.000000,0.000000,0.000000,0.000000,0.000000,0.000000,0.000000]}`		
Returned JSON object		
Data Name	Data Type	Definition
`input_id`	Integer	Read only, the index identifying the input.
`name`	String	The input name (12 characters max).
`input_mix`	Array of floats	The input_mix array corresponds to one row of the input mixing matrix, each element corresponding to one processing

		channel. The values are linear coefficients, and are typically either 0 or 1.

Output

URI	Description
/output/OUTPUT_IDX	Settings for a specific output. OUTPUT_IDX ranges from 0 to 7, corresponding with outputs 1-8.

```
{"output_id":0,"name": "OUTPUT
1","mute":0,"dim":1,"volume":0.000000,"turn_on_volume":0.000000,"max_volume"
:0.000000,"limiter":0,"output_mix":[1.000000,0.000000,0.000000,0.000000,0.00
0000,0.000000,0.000000,0.000000]}
```

Returned JSON object

Data Name	Data Type	Definition
`output_id`	Integer	Read only, the index identifying the output.
`name`	String	The output name (12 characters max).
`mute`	Boolean	Output mute state.
`dim`	Boolean	Output dim control (-20 dB attenuation).
`volume`	Float	Output volume. (in dB)
`turn_on_volume`	Float	Output turn-on volume. (in dB)
`max_volume`	Float	Output max volume. (in dB) The actual output volume setting applied will be the lower of max_volume and volume.
`limiter`	Float	Output limiter threshold. (in dB) A threshold of 0 dB disables the limiter.
`output_mix`	Array of floats	The output_mix array corresponds to one row of the output mixing matrix, each element corresponding to one processing channel. The values are linear coefficients, and are typically either 0 or 1.

Channel

URI	Description
/channel/CHANNEL_IDX	Settings for a processing channel. CHANNEL_IDX is zero-based and ranges from 0 to 7, corresponding with channels A-H.

```
{"ch_id":0,"trim_volume":0.000000,"volume":0.000000,"mute":0,"bass_boost":0.
000000,"treble_boost":0.000000,"hpf": {"mode":0, "freq":120},"lpf":
{"mode":0, "freq":1000},"delay":0}
```

Returned JSON object

Data Name	Data Type	Definition
`ch_id`	Integer	Read only, the index identifying the channel.
`trim_volume`	Float	The input trim volume, typically used for adjusting balance between channels in a pair. (in dB)
`volume`	Float	The channel volume. (in dB)
`mute`	Boolean	Channel mute

`bass_boost`	Float	The bass tone control. (in dB)
`treble_boost`	Float	The treble tone control. (in dB)
`hpf`	Object	An object containing the HPF settings.
`hpf.mode`	Integer	The filter mode. Valid settings are: 0 = disabled 1 = 6 dB/octave 2 = 12 dB/octave 3 = 18 dB/octave 4 = 24 dB/octave
`hpf.freq`	Integer	The filter corner frequency. (in Hz)
`lpf`	Object	An object containing the LPF settings.
`lpf.mode`	Integer	The filter mode. Valid settings are 0-4, and are identical to the HPF modes.
`lpf.freq`	Integer	The filter corner frequency. (in Hz)
`delay`	Integer	The delay in 48 kHz samples. Range is 0 to 720.

PARAMETRIC EQUALIZER

URI		Description
/channel/CHANNEL_IDX/peq/BAND_IDX		Settings for a processing channel's parametric equalizer bands. CHANNEL_IDX and BAND_IDX are zero based. CHANNEL_IDX ranges from 0 to 7, BAND_IDX ranges from 0 to 9.
`{"ch_id":0,"band_id":0,"enab":0,"gain":0.000000,"freq":1400,"q":1.000000}`		
Returned JSON object		
Data Name	Data Type	Definition
`ch_id`	Integer	Read only, the index identifying the channel.
`band_id`	Integer	Read only, the index identifying the band.
`enab`	Boolean	Band enable.
`gain`	Float	Band gain. (in dB)
`freq`	Integer	Band center frequency. (in Hz)
`q`	Float	Band Q.

PAIR

URI		Description
/pair/PAIR_IDX		Settings for a channel pair. PAIR_IDX is zero based and ranges from 0 to 3.
`{"pair_id":0,"mode":0}`		
Returned JSON object		
Data Name	Data Type	Definition
`pair_id`	Integer	Read only, the index identifying the pair.
`mode`	Integer	The pair mode. This is only used by the web configuration UI to indicate if the volume controls for the two channels in the pair should be linked.