Session API

Session Capture API

Manage the Name and Description for a Session

Download Data for a Session

Session Tags API

Session Annotation API

Video Quality API

Visual Page Load Time API

Session Capture API

Get all sessions in your org

Fetch timestamps for a session

Create a session

Stop a session
Get the number of TLS exceptions for each host in a session

Get all sessions in your org

<table>
<thead>
<tr>
<th>Route</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>/v0/sessions?include_all={bool}&amp;num_sessions={int}&amp;tag={string}:{string}</td>
<td>GET</td>
</tr>
</tbody>
</table>

Optional Parameters

- `?include_all={true|false}`: If true, this will return both live (active) sessions and ended (inactive sessions). If this value is false, only live sessions will be included. The default value is false.
- `?num_sessions={n}`: The number of sessions you want the API request to return, sorted by most recent. The default value is 10 and will return up to 10 sessions.
- `?tag={key}:{value}`: Select sessions that contain the tag defined by tag key key and tag value value. Spaces in either the tag key or value may be represented using +. For example, sessions containing the tag my example: this is an example tag could be selected using the query argument tag=my+example:this+is+an+example+tag. This query argument may be provided multiple times to select sessions containing at least one of the tags provided.

Example

To get a list of the most recent 20 sessions, both active and inactive:
To get sessions containing at least one of the tags `my:tag` or `my other:example tag`:

```
curl 'https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions?include_all=true&num_sessions=20&tag=my:tag&tag=my+other:example+tag'
```

**Response**

The response is a JSON object with the `sessions` key, and it contains a list of sessions in your org matching the optional parameters `include_all` and `num_sessions`.

```
{"sessions":
 [  
   {  
     "companion_id": null,
     "session_type": "capture",
     "start_time": 1565291652.17,
     "state": "ended",
     "session_id": "<session_id>",
     "device_id": "<device_id>",
   },
   {  
     "companion_id": null,
     "session_type": "capture",
     "start_time": 1539154924.161,
     "state": "ended",
     "session_id": "<session_id>",
     "device_id": "<device_id>"
   }
  ]
}
```
Fetch timestamps for a session

Route

/v0/sessions/{session_id}/timestamps

Method

GET

Fetch all timestamps associated with capture for the target session.

Example

curl https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/{session_id}/timestamps

Response

If the request is successful, a HTTP 200 OK response is returned with a JSON object containing key-value pairs for any available timestamps associated with the capture session ID.

```json
{
  "capture-complete": 1579113687.702,
  "capture-ended": 1579113686.105,
  "capture-started": 1579113669.27
}
```

A HTTP 404 response is returned if no timestamps are available or the session does not exist.
Create a session

To start a session on a device, **you must first lock that device.** Lock the device through the HeadSpin UI or using the Android Device API or iOS Device API.

**Request Body**

The request body must contain a JSON object with a minimum of:

```json
{
    "session_type": "capture",
    "device_address": "<device_address>"
}
```

It can also optionally contain:

- **device_id**: The ID of the device to start the session on.
- **allow_replace**: If the target device is currently locked by the same user, and has an active capture session, then setting `allow_replace` to true will stop the current session and start a new one. If `allow_replace` is false, then the request will fail in that circumstance. Default false.
- **capture_video**: A boolean value that specifies whether to capture a video of the device screen. Default `true`.
- **capture_network**: A boolean value that specifies whether to capture network data. Default `true`.
- **mitm_ignore**: A JSON array of hosts to ignore. Default `[]` has no effect. See Actions for Hosts with TLS Exceptions for more details.
- **network_keep_in_session_patterns**: An optional JSON array containing patterns for hosts to keep in the network capture. Default `[]` has no effect. Has the opposite effect of `mitm_ignore`. See Actions for Hosts with TLS Exceptions for more details.
- **disable_http2**: A boolean value that specifies whether to prevent the device from using HTTP/2. Default `false`.

**Example**

```bash
curl -X POST https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions -d '{"session_type": "capture","device_address": "G070VM14941305AV@dev-us-pao-0-proxy-1.headspin.io"}'
```

**Response**

If the request is successful, the response is a JSON object with details of the created session, including `session_id`.

```json
{
    "companion_id": null,
    "session_type": "capture",
    "start_time": 1571175936.264,
    "endpoints": [],
}
```
If the request fails, you may get:

- `{"status": "Device is currently in use by someone else.", "status_code": 403}: Someone else is currently using the device. Choose another device to create a session, or wait until this device is free.
- `{"status": "Invalid device ID.", "status_code": 404}: The device ID specified doesn't exist. Double check that the device ID is correct.
- `{"status": "Device must be locked before capture.", "status_code": 403}: The device must be locked by you before a session can be created. Lock the device through the HeadSpin UI or an API call.

**Actions for Hosts with TLS Exceptions**

When using hosts with pinned TLS certificates, one of these two actions should be taken to enable accurate and comprehensive network capture:

- Add HeadSpin as a trusted cert per our documentation on [Android Network Security Configuration](https://ui-dev.headspin.io/docs/network-security), or
- Use HeadSpin's [Instrumentation API](https://ui-dev.headspin.io/docs/instrumentation-api) to enable profiling for already built apps

If it is not possible to trust the HeadSpin cert or enable profiling via instrumentation, hosts with pinned TLS certificates may fail to establish secure connections with our network capture setup, which can cause TLS Exceptions and undesirable app performance.
degradations. To prevent this from affecting your tests, hosts experiencing TLS exceptions may be filtered from the network capture using the `mitm_ignore` argument.

The `mitm_ignore` argument accepts an array of ignored `host:port` or `ip:port` regex patterns. If just the host name is provided, all ports will be matched. These hosts are ignored from the network capture session.

Example usage may include the following:

```
['abc.example.com:443',
'.*\.mydomain.com']
```

The `network_keep_in_session_patterns` argument controls whether a destination `host:port` (or `ip:port`) is kept in the session after an error. Add glob patterns here to keep a destination with an error in the session, to continue to try to parse the network protocol for that destination.

**Stop a session**

<table>
<thead>
<tr>
<th>Route</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/v0/sessions/{session_id}</code></td>
<td>PATCH</td>
</tr>
</tbody>
</table>

**Request Body**

The request body must be this JSON object:

```
{  "active": false }
```
Example

```
curl -X PATCH https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/{session_id} -d '{"active": false}'
```

Response

- A HTTP 200 OK response with `{"msg": "Video uploaded to https://api-dev.headspin.io/v0/sessions/{session_id}.mp4"}` is returned. This message contains a link to the captured video. The name of the video is the session ID.
- A HTTP 500 response if the request fails. Check that you have the right JSON object in the request body.
- A HTTP 404 Not Found response if the request fails. Make sure that the session id in your request is valid.

Get the number of TLS exceptions for each host in a session

<table>
<thead>
<tr>
<th>Route</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>/v0/sessions/{session_id}/tlsexceptions</td>
<td>GET</td>
</tr>
</tbody>
</table>

Example

```
curl https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/{session_id}/tlsexceptions
```
Response

The response is a JSON object where each key is a host name, and the value is the number of TLS exceptions for that host.

```
{
    "abc.example.com:443": 4,
    "www.mydomain.com:443": 2,
}
```

If the request contains an invalid session ID (such as one that doesn't exist), a 404 response will be returned.

Manage the Name and Description for a Session

Fetch the name and description associated with a session

Assign a name and description for a session

Reset the name and description of a session to the default values

Fetch the name and description associated with a session

<table>
<thead>
<tr>
<th>Route</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>/v0/sessions/{session_id}/description</td>
<td>GET</td>
</tr>
</tbody>
</table>
Example

curl https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/{session_id}/description

Response

{
    "name": "HeadSpin Session",
    "description": "HeadSpin Performance Dashboard/Analytics."
}

Assign a name and description for a session

If the session already has a name and description, this route will overwrite the current name and description.

Route     Method
/v0/sessions/{session_id}/description POST

Example

curl -X POST https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/{session_id}/description -d '{"name": "My name", "description": "My description."}'

Response
A HTTP 200 OK response if the request is successful
A HTTP 404 Not Found response if the request fails. Make sure that the session id in your request is valid.

Reset the name and description of a session to the default values

<table>
<thead>
<tr>
<th>Route</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>/v0/sessions/{session_id}/description</td>
<td>DELETE</td>
</tr>
</tbody>
</table>

Example

curl -X DELETE https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/{session_id}/description

Response

A HTTP 200 OK response if the request is successful
A HTTP 404 Not Found response if the request fails. Make sure that the session id in your request is valid.

Download Data for a Session

Download captured session data
**Download a summary of session Issues**

**Retrieve the most significant Issue by Issue Burst**

**Get the session video metadata**

**Download captured session data**

<table>
<thead>
<tr>
<th>Route</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/v0/sessions/{session_id}.{ext}</code></td>
<td>GET</td>
</tr>
</tbody>
</table>

{ext} (extension) specifies what type of data to download for this session. Extension must be one of:

- **mp4**: HeadSpin screen recording for this session. This extension supports a `fps` query parameter with a positive integer value greater than 1 and less than 120. If the `fps` parameters is provided, the screen recording video will be resampled to a constant frame rate at the specified frame rate.

- **har**: Standardized HTTP message archive, containing all HTTP traffic content as a JSON object. This extension supports an `enhanced` query parameter that (if `True`) will enhance the HTTP message archive with the actual HTTP body content extracted from the binary network traffic capture. Entries can be temporally filtered by supplying a `label_id` query parameter corresponding to a session label that spans the desired temporal range. Currently only supports HTTP/1.X.
- `mar`: HeadSpin's variant of HTTP message archive, containing additional metadata. This is also a JSON object.
- `csv`: `mar` data in `.csv` form
- `pcap`: Binary network traffic capture. See Reading Network Packet Data.
- `sslkeylog.txt`: TLS/SSL master secret key log for the `pcap` data. See Reading Network Packet Data.
- `appium.log.gz` or `appium.log`: Appium session log. The log is available for all sessions, including capture sessions and non-capture sessions.
- `selenium.log.gz` or `selenium.log`: Selenium driver session log. The log is available for all sessions, including capture sessions and non-capture sessions.
- `device.log.gz` or `device.log`: Device event log

**Examples**

To save the downloaded data as files on your computer, use the flag `--o {filepath}` in your `curl` commands.

To download the video screen recording:

```bash
curl https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/{session_id}.mp4 --o 'video.mp4'
```

To download the video screen recording resampled to a constant frame rate of 15 frames per second:

```bash
curl https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/{session_id}.mp4\?fps\=15 --o 'video_15fps.mp4'
```
To download the enhanced HAR data containing request and response bodies for HTTP/1.X traffic:

```bash
curl https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/{session_id}.har?enhanced=True -o 'data.har'
```

To download the pcap data:

```bash
curl https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/{session_id}.pcap -o 'data.pcap'
```

To download the TLS/SSL master secret:

```bash
curl https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/{session_id}.sslkeylog.txt -o 'sslkeylog.txt'
```

To download the Appium log:

```bash
curl https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/{session_id}.appium.log.gz -o 'appium.log.gz'
```

To download the Selenium driver log:

```bash
curl https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/{session_id}.selenium.log.gz -o 'selenium.log.gz'
```
To download the device log:

```
curl https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/{session_id}.device.log.gz -o 'device.log.gz'
```

**Response**

The response is a stream of the content of the data file. To save the data into a file on your disk, use the `-o {filepath}` flag in your `curl` request.

**Reading Network Packet Data**

The downloaded packet capture (PCAP) is binary data, with the network traffic encrypted using TLS/SSL. To read this data, you must use a tool that can read `pcap` packets, and decrypt the traffic data with the `sslkeylog.txt` file. To do this:

1. Download [Wireshark](https://www.wireshark.org/).
2. Open Wireshark. In Wireshark, open the `pcap` file with `File > Open`, and select the `pcap` file.
3. Next, you'll need to specify the key log file. Go to `Wireshark > Preferences > Protocols > TLS`.
4. In `(Pre)-Master-Secret log filename`, select the downloaded `sslkeylog.txt` file.
5. The traffic data should now be decrypted. You can read the `pcap` data in Wireshark.

Alternatively you can open and decrypt a pcap directly using the `wireshark` command line interface with `wireshark -o ssl.keylog_file:path/to/sslkeylog.txt`
**Download a summary of session Issues**

This data is the same as that shown in the Waterfall UI issue card.

<table>
<thead>
<tr>
<th>Route</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>/v0/sessions/analysis/issues/{session_id}</td>
<td>GET</td>
</tr>
</tbody>
</table>

**Optional Parameters**

- **orient**: This parameter can have the value `column` (default) or `record`. This specifies the return format of the data:
  - The `column` orientation outputs the data by Issue category and then by column of data available for that issue (e.g. Status, Impact Time, Content Type...).
  - The `record` orientation outputs the data by Issue category and then by each instance of the issue.

**Example**

```bash
curl https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/analysis/issues/ed8ba23d-1092-11ea-a8da-acde48001122
```

To get the data by issue category and by issue:
curl https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/analysis/issues/ed8ba23d-1092-11ea-a8da-acde48001122?orient=record

Response

Example of a column response:

```
{
   "Low Frame Rate": {
      "Impact Time (sec)": [  
         "0.39"
      ],
      "Issue Start": [  
         "00:00:22.225"
      ],
      "Total Low FPS Time (sec)": [  
         "1.39"
      ],
      "Impact/Total (%)": [  
         "28.3"
      ],

   },
   "Domain Sharding": {
      "Domain": [  
         "example.com",
         "example2.com",
         "unity3d.com"
      ],
      "Total Connection Time (ms)": [  
         "675",
         "170"
      ],
      "Impact Time (ms)": [  
         "301",
         "55"
      ],
   }
}
```
"Impacted Connection Count": [
    "1",
    "1"
],
"Subdomain Count": [
    "2",
    "2"
],
"Impact/Total (%)": [
    "44.6",
    "32.4"
]
}

Example of a record response:

{
    "Low Frame Rate": [
        {
            "Impact Time (sec)": 0.39,
            "Issue Start": "00:00:22.225",
            "Total Low FPS Time (sec)": 1.39,
            "Impact/Total (%)": 28.3
        }
    ],
    "Domain Sharding": [
        {
            "Domain": "example.com",
            "Total Connection Time (ms)": 675,
            "Impact Time (ms)": 301,
            "Impacted Connection Count": 1,
            "Subdomain Count": 2,
            "Impact/Total (%)": 44.6
        },
        {
            "Domain": "example2.com",
            "Total Connection Time (ms)": 170,
            "Impact Time (ms)": 55,
        }
    ]
}
Retrieve the most significant Issue by Issue Burst

This endpoint is deprecated since November 2019. Please use the Retrieve a summary of issues endpoint instead.

Get the session video metadata

Example

curl https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/{session_id}/video/metadata
Response

```
{
    "session_id": "64df3cb7-7543-11eb-a85d-067dc7f9db8c",
    "video_duration_ms": 11560,
    "dimensions": {
        "width": 376,
        "height": 800
    },
    "fps": 25.0,
    "contains_audio": false
}
```

Session Tags API

Get all tags for a session

Create a tag for a session

Delete all tags in a session

Get all tags for a session

<table>
<thead>
<tr>
<th>Route</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/v0/sessions/tags/{session_id}</code></td>
<td>GET</td>
</tr>
</tbody>
</table>

Example
curl https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/tags/{session_id}

**Response**

The response is a list of key: value pairs for all the tags applied to that session. For example:

```json
[
    {
        "sessiontype": "demo",
        "tag_key": "test"
    }
]
```

**Create a tag for a session**

<table>
<thead>
<tr>
<th>Route</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>/v0/sessions/tags/{session_id}</td>
<td>POST</td>
</tr>
</tbody>
</table>

**Request Body**

The request body must be a JSON array of key, value tags:

```json
[
    {
        "<tag_key>": "<tag_value>"
    },
    {
        "<tag_key>": "<tag_value>"
    }
]
```
Example

curl -X POST https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/tags/{session_id} -d '[{"demo": "tag"},
{"hello": "world"}]'

Response

- HTTP 200 OK response if the request was successful and tags are added to the session.
- HTTP 404 if the session can't be found. Make sure you have the correct session id.

Delete all tags in a session

<table>
<thead>
<tr>
<th>Route</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>/v0/sessions/tags/{session_id}</td>
<td>DELETE</td>
</tr>
</tbody>
</table>

Example

curl -X DELETE https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/tags/{session_id}

Response
- HTTP 200 OK response if the request was successful. All tags in the session will be deleted.
- HTTP 404 if the session can't be found. Make sure you have the correct session id.

**Session Annotation API**

Label object

Time formatting

Available session label types

Add labels to a session

Get all labels associated with a session

Get a specific label

Get all labels associated with a label group

Update an existing label

Delete an existing label

Retrieve keyframe screenshots

Applying spatial filtering on video analyses

Run page load analysis
**Run audio activity analysis**

**Label object**

The Session Annotation API allows you create and manipulate labels for segments of a capture session. Labels are objects with the following properties:

<table>
<thead>
<tr>
<th>Session Label Object Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>label_id</code></td>
<td>A UUID that uniquely identifies the label.</td>
</tr>
<tr>
<td><code>session_id</code></td>
<td>The session UUID that the label is associated with.</td>
</tr>
<tr>
<td><code>name</code></td>
<td>A string name for a label.</td>
</tr>
<tr>
<td><code>start_time</code></td>
<td>The start time relative to session start. See <a href="https://ui-dev.headspin.io/docs/session-api">note on session time formatting</a>.</td>
</tr>
<tr>
<td><code>end_time</code></td>
<td>(optional) The end time relative to session start. See <a href="https://ui-dev.headspin.io/docs/session-api">note on session time formatting</a>.</td>
</tr>
<tr>
<td><code>ts_start</code></td>
<td>The absolute (UNIX) start time. See <a href="https://ui-dev.headspin.io/docs/session-api">note on session time formatting</a>.</td>
</tr>
<tr>
<td><code>ts_end</code></td>
<td>(optional) The absolute (UNIX) end time. See <a href="https://ui-dev.headspin.io/docs/session-api">note on session time formatting</a>.</td>
</tr>
</tbody>
</table>
### Session Label Object Property

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>label_type</td>
<td>(optional, default &quot;user&quot;) A string that is the type of the label. The label type determines how HeadSpin responds to the label. See the note on available session label types, below.</td>
</tr>
<tr>
<td>category</td>
<td>(optional) A string that is the category of the label. The label category is converted to lower case.</td>
</tr>
<tr>
<td>data</td>
<td>(optional) Any useful JSON-serializable content associated with this label.</td>
</tr>
<tr>
<td>pinned</td>
<td>(optional, default false) A boolean flag indicating whether the label is pinned to the session or not. Pinned labels are shown by default when viewing a session, under the right panel's Details section. Pinned labels cannot be deleted.</td>
</tr>
<tr>
<td>video_box</td>
<td>(optional) Coordinates of bounding boxes provided as nested arrays in the format <code>[[x0, y0, x1, y1], ...]</code>. See the note on applying spatial filtering in video analyses</td>
</tr>
</tbody>
</table>

### Time formatting

Segments are specified by start and end timestamps. Relative (to session start), absolute (UNIX), or an appropriate combination of both timestamps (e.g., `ts_start` and `end_time`) can be provided in a JSON object.

#### Relative timestamps (time since session start)

```
{
  "start_time": "<time>", // required
  "end_time": "[<time>]"  // optional
}
```
The `<time>` parameter must be provided either as integer milliseconds or as a string in "HH:MM:SS.milliseconds" format. For example, the `<time>` parameter representing 1.5 seconds after session start may be represented in integer milliseconds as 1500, and in "HH:MM:SS.milliseconds" as "00:00:01.5" or "1.5". The `end_time` key is optional. If no `end_time` is provided, the label will be considered instantaneous at the corresponding `start_time`.

**Absolute timestamps (UNIX time)**

```
{
  "ts_start": "<time>", // required
  "ts_end": "[<time>]"  // optional
}
```

The `<time>` parameter must be provided as seconds. The `ts_end` key is optional and if not provided, the label will be considered instantaneous at the corresponding `ts_start`.

**Available session label types**

Users can currently create labels of the following session label types:

Free-form annotations:

- "user": The default "label_type" for a free-form Session Annotation.

User Feedback label types (see our doc on [Providing Feedback on HeadSpin Performance Sessions](https://ui-dev.headspin.io/docs/session-api)):
- "suggestion": A tip, hint, or friendly request regarding ways HeadSpin could better meet your needs.
- "needs-improvement": Feedback that an existing solution is insufficient or inadequate and needs to be improved to be useful.
- "report-an-error": Feedback that an existing solution is broken or unusable.

Analysis label types:

- "page-load-request": Runs the visual page load time analysis on the region specified by the label (see our documentation on running page load analysis).
- "audio-activity-request": Runs the audio activity analysis on the region specified by the label (see our documentation on running audio activity analysis).
- "video-content": Runs the video quality analyses on the region specified by the label (see our documentation on User-Perceived Video Quality).

Add labels to a session

A single label or multiple labels can be added at a time.

<table>
<thead>
<tr>
<th>Route</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>/v0/sessions/{session_id}/label/add</td>
<td>POST</td>
</tr>
</tbody>
</table>
The body should be a JSON label object.

An example body for adding a single label:

```
{
    "name": "a helpful name",
    "category": "an optional category for the label",
    "start_time": "10.5",
    "end_time": "1:20.1",
    "data": {
        "optional": "data"
    },
    "pinned": true
}
```

An example body for adding multiple labels at once:

```
{
    "labels": [
        {
            "name": "test label 1", "start_time": 5000},
        {
            "name": "test label 2", "start_time": 10000
        }
    ]
}
```

**Example: Add a single label**

```
curl -X POST https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/{session_id}/label/add -d '{"name": "a helpful name", "category": "an optional category for the label", "start_time": "10.5", "end_time": "1:20.1", "data": {"optional": "data"}, "pinned": true}'
```

**Response**
If the request is successful, a HTTP 200 OK response is returned with the label ID as a JSON object:

```
{"label_id": "bf1434cf-99f4-11e9-b9f7-f21898a483e5"}
```

**Example: Add multiple labels at a time**

```bash
curl -X POST https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/{session_id}/label/add -d '{"labels": [{"name": "test label 1", "start_time": 5000}, {"name": "test label 2", "start_time": 10000}]}'
```

**Response**

If the request is successful, a HTTP 200 OK response is returned with the label IDs as a JSON object:

```
{"label_ids": ["be9fcccc-5b88-11eb-8aae-b5c31a9b9aee", "be9fcccd-5b88-11eb-8aae-b5c31a9b9aee"]}
```

**Get all labels associated with a session**

<table>
<thead>
<tr>
<th>Route</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>/v0/sessions/{session_id}/label/list</td>
<td>GET</td>
</tr>
</tbody>
</table>

**Example**
curl https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/{session_id}/label/list

Response

The response is a JSON object with the `labels` key, and it contains a list of label objects. For example:

```
{"labels": [
  {
    "label_id": "bf1434cf-99f4-11e9-b9f7-f21898a483e5",
    "session_id": "4867b234-92c3-11e9-9362-02e56c2f74c2",
    "name": "a helpful name",
    "category": "an optional category for the label",
    "start_time": 10500,
    "end_time": 80100,
    "ts_start": 1585162717.054,
    "ts_end": 1585162786.654,
    "data": {"optional": "data"},
    "pinned": true
  }
]
```

Get a specific label

<table>
<thead>
<tr>
<th>Route</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>/v0/sessions/label/{label_id}</td>
<td>GET</td>
</tr>
</tbody>
</table>
Example

curl https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/label/bf1434cf-99f4-11e9-b9f7-f21898a483e5

Response

A successful request receives a JSON response with the specified label for the given session.

```json
{
    "label_id": "bf1434cf-99f4-11e9-b9f7-f21898a483e5",
    "session_id": "4867b234-92c3-11e9-9362-02e56c2f74c2",
    "name": "a helpful name",
    "category": "an optional category for the label",
    "start_time": 10500,
    "end_time": 80100,
    "ts_start": 1585162717.054,
    "ts_end": 1585162786.654,
    "data": {"optional": "data"},
    "pinned": true
}
```

Get all labels associated with a label group

Labels that are "linked" have the same label_group_id assigned to them, which can be used to fetch the group of labels. For example, a page-load-request label and a page-load-result label can be "linked", because the page-load-request label drives the page-load-result label.

Visual Page Load Analysis
### Example

```bash
curl https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/label/group/{label_group_id}
```

### Response

The response is a JSON object with the `labels` key, and it contains a list of label objects.

```json
{
"labels": [
{
"label_id": "6c4f23de-04b0-4550-b6e4-3fbfaf005e89",
"label_group_id": "1c8501bc-4ca0-4aa3-b73d-10dc5bdddba0b",
"label_type": "page-load-request",
"session_id": "446c0272-e329-11ea-b51a-bf4d9c36b3f2",
"name": "test",
"category": "page load",
"start_time": 14254,
"end_time": 28509,
"ts_start": 1597076919.267,
"ts_end": 1597076933.522,
"data": null,
"pinned": false,
},
{
"label_id": "6aec3050-e32b-11ea-b51a-bf4d9c36b3f2",
"label_group_id": "1c8501bc-4ca0-4aa3-b73d-10dc5bdddba0b",
"label_type": "page-load-result",
```

<table>
<thead>
<tr>
<th>Route</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>/v0/sessions/label/group/{label_group_id}</td>
<td>GET</td>
</tr>
</tbody>
</table>
"session_id": "446c0272-e329-11ea-b51a-bf4d9c36b3f2",
"name": "test",
"category": "page load",
"start_time": 15760,
"end_time": 27760,
"ts_start": 1597076920.773,
"ts_end": 1597076932.773,
"data": {
  "end_sensitivity": 0.975,
  "start_sensitivity": 0.835
},
"pinned": false,
}

Update an existing label

Route
/v0/sessions/label/{label_id}
Method
PATCH

Request Body

The body should be a JSON with the fields of the label to update.

```json
{
  "name": "new name",
  "category": "new category",
  "start_time": "1:00.7",
  "end_time": "1:30.5",
  "data": {
    "optional": "data",
    "more": "data"
  },
  "pinned": false
}
```
**Example**

curl -X PATCH https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/label/bf1434cf-99f4-11e9-b9f7-f21898a483e5 -d '{"name": "new name", "category": "new category", "start_time": "1:00.7", "end_time": "1:30.5", "data": {"optional": "data", "more": "data"}, "pinned": false}'

**Response**

- {"status": "OK", "status_code": 200} if the label is updated successfully.
- {"status": "Label not found.", "status_code": 404} if the label is not found.

Check that the label ID is correct.

**Delete an existing label**

<table>
<thead>
<tr>
<th>Route</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>/v0/sessions/label/{label_id}</td>
<td>DELETE</td>
</tr>
</tbody>
</table>

**Example**

curl -X DELETE https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/label/bf1434cf-99f4-11e9-b9f7-f21898a483e5

**Response**
- "status": "OK", "status_code": 200} if the label is deleted successfully.
- "status": "Label not found.", "status_code": 404} if the label is not found.

Check that the label ID is correct.

**Retrieve keyframe screenshots**

Keyframes are JPEG images corresponding to the start and end of a Session Label time interval. These routes allow you to download the corresponding screenshots.

<table>
<thead>
<tr>
<th>Route</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>/v0/sessions/label/{label_id}/keyframe/start</td>
<td>GET</td>
</tr>
<tr>
<td>/v0/sessions/label/{label_id}/keyframe/end</td>
<td>GET</td>
</tr>
</tbody>
</table>

**Example**

We use the `--o` option to specify the location and name for the downloaded screenshot.

```
curl https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/label/bf1434cf-99f4-11e9-b9f7-f21898a483e5/keyframe/start --o start-screenshot.jpeg
```

**Response**
A HTTP 200 OK response if the request is successful. The file is saved as specified by the -o option.

- `{"status": "Label not found.", "status_code": 404}` if the label is not found. Check that the label ID is correct.

Applying spatial filtering on video analyses

For video analyses that are driven by session labels, you can instruct where on the frames the analyses should run by specifying the `video_box` field in the request labels. `video_box` should be provided as nested arrays in the format `[[x0, y0, x1, y1], ...]` where `x0`, `y0`, `x1`, and `y1` are the coordinates of the bounding boxes on the video frames. The origin, `(0, 0)`, is placed at the upper left corner of a video frame. For example, `[[0, 0, 100, 200]]` specifies a rectangle on the frame that can be drawn with points `(0, 0)` (at the upper left corner) and `(100, 200)` (100 pixels to the right and 200 pixels below from `(0, 0)`). Note that if the width and height of the session video are 450 and 800, respectively, `[[0, 0, 450, 800]]` specifies the entire frame.

Shown below are the available video analyses and the label types for which specifying `video_box` is meaningful:

<table>
<thead>
<tr>
<th>Analysis</th>
<th>Label Type</th>
<th>Number of bounding boxes allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>page load analysis</td>
<td>&quot;page-load-request&quot;</td>
<td>1</td>
</tr>
</tbody>
</table>
You can get the dimensions of the session video to help with selecting the coordinates for the desired bounding boxes. Once you have added the labels with video_box specified, you can check what the filtered frames look like by retrieving the screenshot of the labels.

**Run page load analysis**

The page load analysis measures the amount of time it takes to offload from Screen 1 (the screen at the start of the annotated region) to reach Screen 2 (the screen at the end of the annotated region).* This is achieved by detecting where the first and last meaningful page changes occur in the annotated region, and you can tune how sensitive the analysis should be in detecting the page changes (see the section below on Tunable Sensitivities for details).

*For an illustration of the analysis, refer to the Page Load section in our documentation on measuring visual load time in HeadSpin Sessions.

**Example: Add page-load-request labels to a session**

To run the page load analysis, simply add labels on the regions of interest with the label_type specified to page-load-request. For each page-load-request label that you add, you can provide the following optional customizations:

- Specify custom sensitivities in the data field. See the section below Tunable Sensitivities.
- Specify the coordinates of a bounding box in the video_box field. See our documentation on applying spatial filtering on video analyses.
curl -X POST https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/{session_id}/label/add -d '{"labels": [{"name": "no customizations", "label_type": "page-load-request", "start_time": "24.37", "end_time": "29.39"}, {"name": "customized", "label_type": "page-load-request", "start_time": "24.37", "end_time": "29.39", "video_box": [[0, 600, 450, 700]], "data": {"start_sensitivity": 0.98, "end_sensitivity": 0.99}}],

{"label_ids": ["85e731d2-65cd-11eb-b3c0-e7adb5682894", "85e731f0-65cd-11eb-b3c0-e7adb5682894"]}

These are the label IDs of the two page-load-request labels that you added. If the page load regions are found, then page-load-result labels are created on them during the analysis.

**Example: Retrieve page-load-result labels**

You can check the session Waterfall UI to see if any page-load-result labels have been created.

You can also look up the labels with a label group ID since a page-load-request label and its resulting page-load-result label are "linked" by the same label group ID. First, look up the page-load-request label with its label ID to obtain its label group ID:

curl https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/label/85e731f0-65cd-11eb-b3c0-e7adb5682894

```
{
    "label_id": "85e731f0-65cd-11eb-b3c0-e7adb5682894",
    "label_group_id": "85e731f1-65cd-11eb-b3c0-e7adb5682894",
    "label_type": "page-load-request",
    "session_id": "2b1312f9-55ec-11eb-9382-067dc7f9db8c",
```
Then look up the labels that share the same label group ID to find the `page-load-result` label:

```json
{
    "labels": [
        {
            "label_id": "85e731f0-65cd-11eb-b3c0-e7adb5682894",
            "label_group_id": "85e731f1-65cd-11eb-b3c0-e7adb5682894",
            "label_type": "page-load-request",
            "session_id": "2b1312f9-55ec-11eb-9382-067dc7f9db8c",
            "name": "customized",
            "category": "page load",
            "start_time": 24370,
            "end_time": 29390,
            "ts_start": 1610575880.728,
            "ts_end": 1610575885.748,
            "data": {
                "start_sensitivity": 0.98,
                "end_sensitivity": 0.99,
                "pinned": false,
                "video_box": [[0, 600, 450, 700]]
            }
        },
        {
            "label_id": "87cfc70c-65cd-11eb-b3c0-e7adb5682894",
            "label_group_id": "85e731f1-65cd-11eb-b3c0-e7adb5682894",
            "label_type": "page-load-result",
            "session_id": "2b1312f9-55ec-11eb-9382-067dc7f9db8c",
            "name": "customized",
            "category": "page load",
            "start_time": 24370,
            "end_time": 29390,
            "ts_start": 1610575880.728,
            "ts_end": 1610575885.748,
            "data": {
                "start_sensitivity": 0.98,
                "end_sensitivity": 0.99,
                "pinned": false,
                "video_box": [[0, 600, 450, 700]]
            }
        }
    ]
}
```
Tunable Sensitivities

The following sensitivities can be tuned (default values shown) in the page load analysis:

```
{
  "start_sensitivity": 0.835,
  "end_sensitivity": 0.975
}
```

You can provide values between 0 (not at all sensitive) to 1 (extremely sensitive) for both `start_sensitivity` and `end_sensitivity`. The higher the sensitivity value, the more sensitive the analysis is to detecting minute screen changes between frames.

If you wish to use custom sensitivities on a region of interest, specify the sensitivities in the `data` field when adding a `page-load-request` label.

Run audio activity analysis
The audio activity analysis finds regions of audio activity based on the Audio Volume time series (which tracks the intensity of the audio waveform). This analysis is useful for finding the exact timing of audio events and is highly customizable. See the section below on Tunable Thresholds for details regarding customization.

**Example: Add audio-activity-request labels to a session**

To run the audio activity analysis, simply add labels on the regions of interest with the label_type specified to audio-activity-request. For each audio-activity-request label that you add, you can provide custom thresholds in the data field. If none are provided, then the default thresholds (see the section below Tunable Thresholds) are used.

```
curl -X POST https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/{session_id}/label/add -d '{"labels": [{"name": "use default thresholds", "label_type": "audio-activity-request", "start_time": 0, "end_time": 1000}, {"name": "use custom merge_ms threshold", "label_type": "audio-activity-request", "start_time": 0, "end_time": 1000, "data": {"thresholds": {"merge_ms": 150}}}]},
{"label_ids": ["25fd8346-6269-11eb-aa94-e5af0b819d09", "25fd8348-6269-11eb-aa94-e5af0b819d09"]}
```

These are the label IDs of the two audio-activity-request labels that you added. If the audio activity regions are found, then audio-activity-result labels are created on them during the analysis.

**Example: Retrieve audio-activity-result labels**
You can check the session Waterfall UI to see if any audio-activity-result labels have been created.

You can also look up the labels with a label group ID since a audio-activity-request label and its resulting audio-activity-result label(s) are "linked" by the same label group ID. First, look up the audio-activity-request label with its label ID to obtain its label group ID:

```
curl https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/label/25fd8348-6269-11eb-aa94-e5af0b819d09
```

```
{
  "label_id": "25fd8348-6269-11eb-aa94-e5af0b819d09",
  "label_group_id": "6cd99b5b-6280-11eb-aa94-e5af0b819d09",
  "label_type": "audio-activity-request",
  "session_id": "362fe37c-6263-11eb-aa94-e5af0b819d09",
  "name": "use custom merge_ms threshold",
  "category": "audio activity request",
  "start_time": 0,
  "end_time": 1000,
  "ts_start": 1611959920.347,
  "ts_end": 1611959921.347,
  "data": {"thresholds": {"merge_ms": 150}},
  "pinned": false
}
```

Then look up the labels that share the same label group ID to find the audio-activity-result label(s):

```
curl https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/label/group/6cd99b5b-6280-11eb-aa94-e5af0b819d09
```
Tunable Thresholds

https://ui-dev.headspin.io/docs/session-api
The following thresholds can be tuned (default values shown) in the audio activity analysis:

```json
{
    "thresholds": {
        "volume": 0.1,
        "duration_ms": 15,
        "merge_ms": 50
    }
}
```

The thresholds are used in the following ways:

1. Regions of the session timeline where the audio volume time series values are above the `volume` threshold for more than `duration_ms` are identified.

2. If the gaps between any of the consecutive regions are less than `merge_ms`, then the consecutive regions are merged into one.

For example, let's say two regions, 100 - 200 ms and 300 - 500 ms, are identified after Step 1. Since the gap between these two regions is greater than 50 ms (the default `merge_ms` threshold), these regions are recognized as two separate audio activity regions, creating two `audio-activity-result` labels. If the `merge_ms` threshold had been set to 120, then these regions would be merged into one, and only one `audio-activity-result` label would be created.

If you wish to use custom thresholds on a region of interest, specify the thresholds in the `data` field when adding a `audio-activity-request` label.
Video Quality API

The video quality API uses reference-free metrics to analyze the quality of video content in a session screen recording. These analyses are applied automatically to time intervals in a session where the device is in the landscape orientation. To apply these analyses to other sections of a session, you can use the endpoints documented in this section.

Each route expects a body consisting of a session ID, and optionally a set of regions or timestamps relative to the session start in which to run the corresponding analysis. If no regions or timestamps are provided, the analysis will be run on the entire video.

Analyze blockiness

Analyze blurriness

Analyze video for screen freezing

Analyze blockiness

Run blockiness analysis on the entire video or a region specified by timestamps and plot a time series of blockiness scores in the Waterfall UI.

<table>
<thead>
<tr>
<th>Route</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>/v0/sessions/analysis/blockiness</td>
<td>POST</td>
</tr>
</tbody>
</table>
Request Body

The request body must be a JSON object with the `session_id` field. It can also optionally contain the timestamps, as seen in the example below.

```json
{
    "session_id": "<session_id>",
    "regions": // optional
    [
        {
            "start_time": "<time>",
            "end_time": "<time>" // optional
        }
    ]
}
```

See [Note on Time Formatting](#) for a description of allowable `<time>` formatting. The "end_time" argument is optional for the last time interval, and if not provided the analysis will run to the end of the session. If the `regions` argument is omitted, the API will run the analysis over the full session.

Example

```bash
curl -X POST https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/analysis/blockiness -d '{"session_id": <session_id>, "regions": [{"start_time": "10.5", "end_time": "20.1"}]}'
```

Response

If the request is successful:
A HTTP 404 response is returned if the session is not found. Check that the `session_id` is correct.

**Analyze blurriness**

Run blurriness analysis on the entire video or a region specified by timestamps and plot a time series of blurriness scores in the Waterfall UI.

### Route

```
/v0/sessions/analysis/blurriness
```

### Method

POST

### Request Body

The request body must be a JSON object with the `session_id` field. It can also optionally contain the timestamps, as seen in the example below.

```
{
    "session_id": "<session_id>",
    "regions":  // optional
}
```
See Note on Time Formatting for a description of allowable `<time>` formatting. The "end_time" argument is optional for the last time interval, and if not provided the analysis will run to the end of the session. If the `regions` argument is omitted, the API will run the analysis over the full session.

**Example**

```bash
curl -X POST https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/analysis/blurriness -d '{"session_id": <session_id>, "regions": [{"start_time": "10.5", "end_time": "20.1"}]}'
```

**Response**

If the request is successful:

```json
{
   "_headspin_progress": "..",
   "status": {
      "blurriness": "success"
   },
   "session_id": "ed8ba23d-1092-11ea-a8da-acde48001122"
}
```
A HTTP 404 response is returned if the session is not found. Check that the session_id is correct.

**Analyze video for screen freezing**

Run video screen freezing analysis on the entire video or a region specified by timestamps and render Screen Freezing Issue card and highlights in the Waterfall UI. Requires the analysis: screen freezing session tag to be included in the session data. Tags may be added to the session via the Session Tag API.

### Route

**Method**

/v0/sessions/analysis/screenfreezing  

**POST**

### Request Body

The request body must be a JSON object with the session_id field. It can also optionally contain the timestamps, as seen in the example below.

```json
{
    "session_id": "<session_id>",
    "regions": // optional
    [
        {
            "start_time": "<time>",
            "end_time": "<time>" // optional
        }
    ]
}```
See Note on Time Formatting for a description of allowable <time> formatting. The "end_time" argument is optional for the last time interval, and if not provided the analysis will run to the end of the session. If the regions argument is omitted, the API will run the analysis over the full session.

Example

```bash
curl -X POST https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/analysis/screenfreezing -d '{"session_id": <session_id>, "regions": ["start_time": "10.5", "end_time": "20.1"]}'
```

Response

If the request is successful:

```json
{
    "_headspin_progress": "..",
    "status": {
        "screenfreezing": "success"
    },
    "session_id": "ed8ba23d-1092-11ea-a8da-acde48001122"
}
```

A HTTP 404 response is returned if the session is not found. Check that the session_id is correct.

Visual Page Load Time API
The visual page load time analysis measures the amount of time it takes to transition from a starting page to an ending page. This API can be used for several purposes:

- Measuring app launch time: Transition time from the Home page or Menu screen to the fully loaded app page
- Measuring single page transition, e.g. the time between clicking on a shop item until the item page is fully loaded
- Measuring the time it takes to perform an entire user flow, e.g. the time between a shopping app has been fully loaded to a successful check out

This API contains the following assumptions:

- The first major visual change in the session is assumed to be the start of the page load.
- The end of the session (or the session at end_time if running the analysis on a region of interest) is in a fully loaded state and traces back to the time that first achieves the fully loaded state, estimating this as the end of the page load.

Note that this API runs via Session Annotation API, and you can run the same analysis via adding session labels of the label type page-load-request. When the API to run the analysis is called, a page-load-request label is first created on the region of interest, on which the analysis is run. Then a label on the resulting page load region from the analysis is created. This means that for every successful analysis request, a pair of page load request and result labels is created.

Retrieving a Previously Estimated Page Load Event
Retrieve any previously generated page load results for a session.

**Route**

/v0/sessions/analysis/pageloadtime/{session_id}  
GET

**Example**

```bash
curl https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/analysis/pageloadtime/b0d36a38-a296-11e9-8169-f21898a483e52
```

**Response**

Example response for a successful request with a valid session:

```json
{
    "session_id": "b0d36a38-a296-11e9-8169-f21898a483e52",
    "page_load_regions": [
        {
            "request_label_id": "83b0ff91-7480-11ea-aa54-9cb6d08b256d",
            "request_label_name": "Page A to B",
            "request_start_time": 20000,
            "request_end_time": 30000,
            "request_ts_start": 1585227174.565,
            "request_ts_end": 1585227184.565,
            "start_sensitivity": 0.835,
            "end_sensitivity": 0.99,
            "result_label_id": "632e8f8e-72e7-11ea-aa54-9cb6d08b256d"
        }
    ]
}
```
Run User-Perceived Load Time Analysis

Run an analysis to estimate the user-perceived visual load time on app launch or for any single page change in a region of interest and render a highlight in the Waterfall UI.

Route

/v0/sessions/analysis/pageloadtime/{session_id}  POST

Request Body

The request body is optional. If omitted, the API will run the analysis over the full session with the default values. If provided, the request body must be a JSON object with the following optional keys:

- **regions**: An array of JSON objects with the following keys:
  - **name**: (required) A string name for the region.
- **start_time** or **ts_start**: (optional) The start time (see **Note on Time Formatting** for allowed formatting) for the region of interest. Defaults to the start of the session.
- **end_time** or **ts_end**: (optional) The end time (see **Note on Time Formatting** for allowed formatting) for the region of interest. Defaults to the end of the session.
- **start_sensitivity**: (optional) A value between 0 (not sensitive) and 1 (extremely sensitive) used to tune the analysis for identifying the start of the page load. Applies to this region only. Defaults to 0.835.
- **end_sensitivity**: (optional) A value between 0 (not sensitive) and 1 (extremely sensitive) used to tune the analysis for identifying the end of the page load. Applies to this region only. Defaults to 0.975.

An example body:

```json
{
  "regions": [
    {
      "name": "Page A to B",
      "start_time": "20.0",
      "end_time": "30.0"
    }
  ]
}
```
```json
{
    "name": "Page B to C",
    "start_time": "40.0",
    "end_time": "50.0",
    "start_sensitivity": 0.95
}
,
"end_sensitivity": 0.99
}
```

**Example**

```bash
curl -X POST https://7c7e47e8567444fabe743cbfd558c30e@api-dev.headspin.io/v0/sessions/analysis/pageloadtime/b0d36a38-a296-11e9-8169-f21898a483e52 -d
'{"regions": [{"name": "Page A to B", "start_time": "20.0", "end_time": "30.0"},
{"name": "Page B to C", "start_time": "40.0", "end_time": "50.0", "start_sensitivity": 0.95}],
"end_sensitivity": 0.99'}
```

**Response**

An example response is shown here where a visual load event was identified for the first request region but not for the second one:

```json
{
    "units": "milliseconds since session start for start_time and end_time, seconds since epoch for ts_start and ts_end",
    "page_load_regions": [
        {
            "request_label_id": "83b0ff91-7480-11ea-aa54-9cb6d08b256d",
            "request_label_name": "Page A to B",
            "request_start_time": 20000,
```
"request_end_time": 30000,
"request_ts_start": 1585227174.565,
"request_ts_end": 1585227184.565,
"start_sensitivity": 0.835,
"end_sensitivity": 0.99,
"result_label_id": "632e8f8e-72e7-11ea-aa54-9cb6d08b256d",
"start_time": 20280,
"end_time": 24000,
"ts_start": 1585227174.845,
"ts_end": 1585227178.565
},
{
"request_label_id": "a4ef103e-1518-11e9-a3be-0242756d4341",
"request_label_name": "Page B to C",
"request_start_time": 40000,
"request_end_time": 50000,
"request_ts_start": 1585227194.565,
"request_ts_end": 1585227204.565,
"start_sensitivity": 0.95,
"end_sensitivity": 0.99,
"error_msg": "Unable to identify a page change."
}
],
"session_id": "b0d36a38-a296-11e9-8169-f21898a483e52",
"_headspin_progress": "."