title: Browser Monitor description: Browser monitor can monitor website performance and user journeys.

The browser monitor executes the configured script in a Chrome browser. It records various performance metrics during the execution. The script can extract & verify the page content using `assert` API during the execution. It can optionally collect screenshots.

Configuration

General

- **Name** - Name of the monitor. Max length is 255 characters.
- **Interval** - Monitor execution interval.
- **Locations** - List of locations to run the monitor.

Script

The browser monitor scripts are Node.js scripts that control a headless Chrome browser and use Google Puppeteer framework to drive the browser. For every run, the monitor will invoke the `testScript()` method with Puppeteer Page object as a parameter. The script content should be inside the `testScript()` method. For more information on specific use cases, refer to the `Browser Examples` section while creating a browser monitor. Use Node.js assert API to check if the values in the page match your requirements. If any assertion fails, the system declares the run as a failure. Monitor run logs, shown in Sematext Run details page, contain failure details.

Checkout awesome-puppeteer GitHub repository for more examples and tips for writing puppeteer scripts.

Conditions

Condition types supported in the browser monitor are:

- **Error** - During each run, if there are any errors like navigation timeout, assertion failed, etc., it will be recorded in the error field of the response. This does not include the error returned as part of the response body.
- **Metric** - Used to make sure the metrics are within the expected range.

By default, the UI adds the below conditions while creating a browser monitor. You can change them during the creation:

- Error equals empty
- Response Time metric Less Than 20000 ms

Screenshots

The browser monitor script allows the collection of page screenshots at any point during the execution. This can be collected using page.screenshot() Puppeteer API. JPEG and PNG image types are supported. Currently, the number of screenshots per run is limited to one. On failure of the script due to errors like navigation timeout, assertion failed, etc., a screenshot `error.png` will be collected for analysis.

Web Vitals

Web Vitals is an initiative by Google to provide unified guidance for quality signals that are essential to delivering a great user experience on the web. These are a set of performance metrics that they consider are essential for improving user experience. The Core Web Vitals are:

- Largest Contentful Paint (LCP)
- Total Blocking Time (TBT) - Synthetic equivalent for First Input Delay (FID)
- Cumulative Layout Shift (CLS)

The other Web Vital metrics are:

- First Contentful Paint (FCP)
- Time To First Byte (TTFB)

Synthetics Browser monitor collects all the above metrics except Total Blocking Time (TBT). The Web Vitals report under the Browser monitor displays these metrics. For each metrics, the recommended thresholds are also displayed for guidance. Since the Browser monitor tests the website in a desktop environment, these thresholds are for desktop devices.

You can filter the metrics based on locations and aggregate the results by average (default), percentile (99th, 95th, and 75th), min, and max.

Synthetics Web Vitals

Waterfall chart

For every run, the browser monitor collects all the resources fetched during the run. These resources are shown in a graphical waterfall chart in the individual run details page. The metrics shown for each resource are:

- **Total Time** - Total elapsed time of the request to fetch the resource in ms.
- **Started At** - Relative time when the fetch started in ms.
- **Blocked** - Time the request spent waiting before it could be sent in ms.
- **DNS** - Time taken for DNS resolution in ms.
- **Connect** - Socket connection time in ms.
- **SSL** - SSL handshake time in ms.

- **Send** - Time taken to send the request in ms.
- **Wait** - Time taken to receive the first byte of response from server in ms.
- **Receive** - Time taken to download the resource in ms.
- **Transfer Size** - Network size of the resource in bytes.
- **Content Size** - Actual uncompressed size of the resource.

Run environment

Each browser monitor run is executed in an isolated environment using a fresh instance of headless Google Chrome browser in a Node.js environment. Versions of various dependencies are:

- **Node.js** - 12.x
- **Google Chrome** - 83.0.4103.0
- **Puppeteer** - 3.1.0

Default runtime configuration values are:

- Chrome browser environment - Desktop
- Resolution - `1920x1080`
- Default Navigation timeout - 20 seconds
- Timeout for script execution - 1 minute
- Memory - 2048 MB
- CPU - 1 vCPU
- Network - Throttled using Chrome settings. Download speed - 20 Mbps, Upload speed - 5 Mbps, Latency - 4ms.