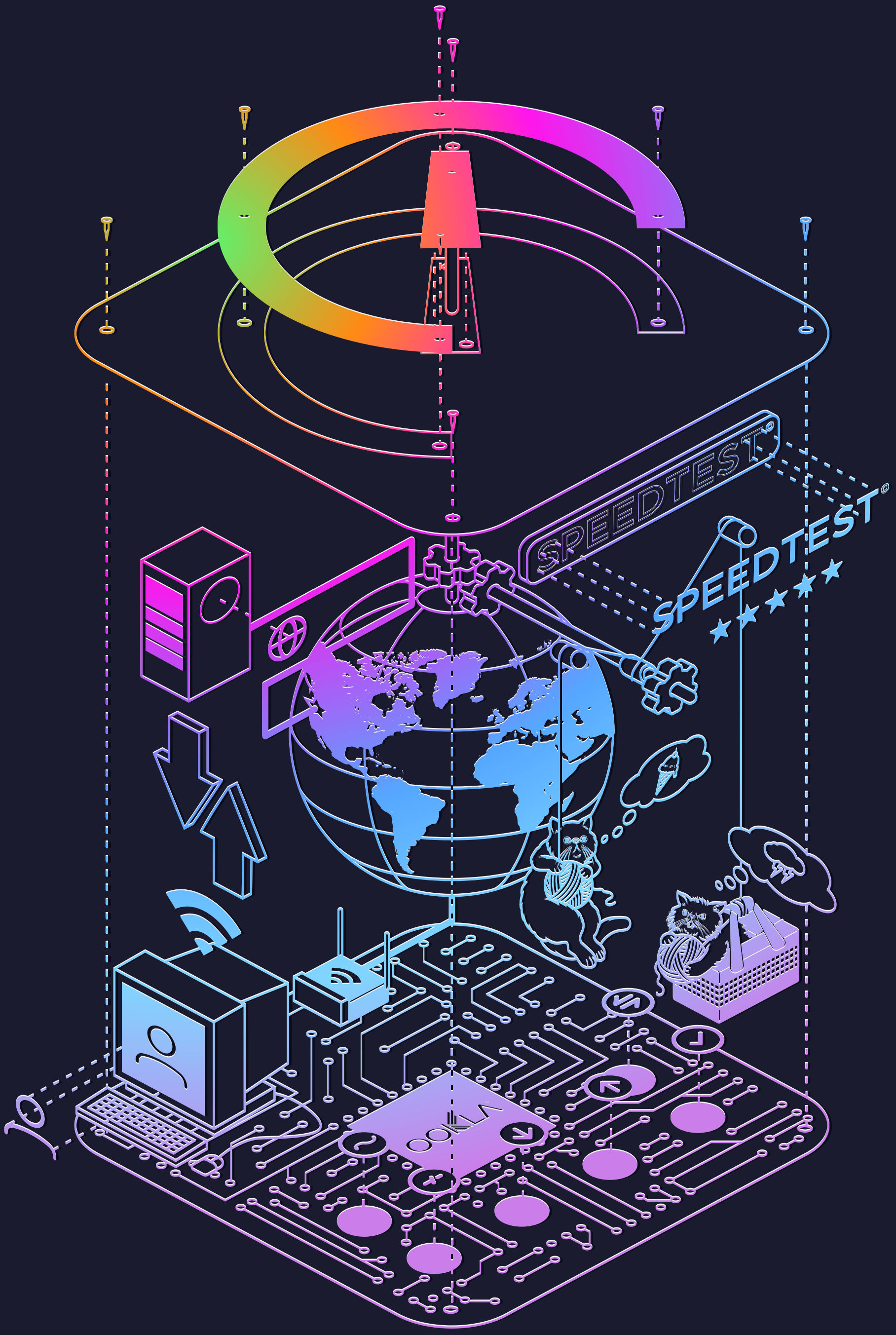
OOKLA

SPEEDTEST® Web SDK

Document Version: 1.3 (updated June 3, 2020)



Contents

Overview

The Speedtest Web SDK is designed to provide the Speedtest web client functionality in a headless form so that it can be incorporated into a custom UX. This document will describe how to include the SDK in a web app, including best practices on configuration, initiating a test, and receiving updates on the test status and results.

Including the SDK in Your Application

The JS SDK is provided as a gnu zipped tar archive which contains a npm bundle. This bundle can be installed in a npm project with the `npm i` command on the command line:

```
$ npm i ookla-speedtest-js-sdk-1.0.0.tgz
```

The EngineApi is expected to be imported into the application source code for use. Refer to the details below for the EngineApi interface details and callflows and code examples.

SDK Dependencies

The JS SDK dependencies are available in the SDK bundle's package.json file:

```
"dependencies": {
    "axios": "^0.19.0",
    "bowser": "^2.7.0",
    "build-url": "^2.0.0",
    "compare-versions": "^3.5.1",
    "config": "^3.2.4",
    "custom-event-polyfill": "^1.0.7",
    "detect-node": "^2.0.4",
    "js-md5": "^0.7.3",
    "lodash": "^4.17.15",
    "mini-signals": "^1.2.0",
    "qs": "^6.9.1"
  }
```

Configuring the Test Client

The test client can be configured by using the EngineApi.configure method. The Configuration options can be found here. See the Callflows and Code Examples for detailed examples for how to configure the client for specific use cases.

API Reference

Signals

Signals (currently implemented using the public mini-signals library) are a lightweight implementation of an observer pattern used to provide event updates from the Speedtest engine to the consuming application. For purposes of users of the Speedtest engine, all signals will be created and managed by the engine, and signals will be accessed as properties on the `EngineApi` or related objects.

The mini-signals API is documented in its own dedicated github repo, but for convenience, the relevant portions (how to attach and manage event handler callbacks) are explained briefly here:

Signal API Details

```
signal.add(fn, [thisArg]) ⇒ MiniSignalBinding
```

Register a new listener. The provided `fn` callback will be invoked whenever a message is dispatched through the listener, using the optional `thisArg` to provide context if provided.

```
signal.once(fn, [thisArg]) ⇒ MiniSignalBinding
```

Register a new listener that is only invoked once.

```
signal.detach(MiniSignalBinding) ⇒ MiniSignal
```

Remove an existing listener (binding). Returns the instance on this method was called.

```
signal.detachAll() ⇒ MiniSignal
```

Remove all listeners (bindings) on a signal. Returns the instance on this method was called.

MiniSignalBinding API Details

A MiniSignalBinding is an opaque object that can be used to refer to an existing binding. They are created by the mini-signals library, but can be stored and later used to remove added listeners.

Class: EngineApi

Overview

Provides the primary interface for interacting with the Speedtest client.

Importing

The EngineApi is a singleton, and can be imported from the SDK:

As ES6 module import:

```
import EngineApi from ‘@ookla/speedtest-js-engine’;
```

Or using `require`:

```
const {EngineApi} = require(‘@ookla/speedtest-js-engine’);
```

Method Details

```
configure(options = {})
```

Used to configure the test client by overriding default settings as needed. See Configuration Properties for details.

```
getServers(num) ⇒ Promise[List[Server]]
```

Search for N local servers.

```
getServers(List[serverID]) ⇒ Promise[List[Server]]
```

Given a list of server IDs, retrieve the associated server records.

```
searchServers(name, num = ∞) ⇒ Promise[List[Server]]
```

Search for N servers by name.

```
sortServers() ⇒ Promise[List[Server]]
```

Select from default list and perform server selection. Also dispatches the result to `onServersSorted`.

```
sortServers(serverList) ⇒ Promise[List[Server]]
```

Given a server list, perform server selection. Also dispatches the result to `onServersSorted`.

```
startTest() ⇒ Promise[void]
```

If no server is specified as a parameter to `startTest()`, then by default, server selection is performed and the test is executed with the lowest latency server.Start a test against the provided server.

```
startTest(server) ⇒ Promise[void]
```

Start a test against the provided server.

```
nextStage(name) ⇒ void
```

Trigger the next stage to start. Only used when the engine is configured with `automaticStageProgression=false` in order to manually start each stage. If no stage name is provided, the next stage is automatically selected (see `config:stagesList`).

```
stopTest() ⇒ void
```

Stop a running test.

```
saveTest() ⇒ void
```

Save the test results. This API results in the same behavior as invoking `nextStage('save')`.

```
setSupplementalSaveData(data) ⇒ void
```

Set supplemental data to be saved when `saveTest()` is invoked. The supplemental data is stored in the engine until `setSupplementalSaveData` is invoked with parameter `data = null`. And the supplemental data stored in the engine is overwritten each time `setSupplementalSaveData` is invoked.

```
lookupSpeedtestResult(guid) ⇒ Promise[TestResult]
```

Retrieve saved test results. This API provides the test result based on the `guid`. The `guid` is a global unique identifier which is generated by `startTest()` and is included for lookup of the corresponding saved test results. The `guid` is available in the saved test result as `resultId` (refer to `onResultSaved` for retrieving the saved test result). **Note**: Results are enriched asynchronously, so some fields might take a few moments to be populated. Subsequent method calls should return fully populated results.

Signal Details

Signals are accessed as properties on the EngineApi object (see Signals reference for details). The available signals (and details of parameters passed to their handler callbacks) are:

```
onServersSorted
```

Triggered when a server sort is completed (all servers have either finished their latency test or timed out). Handler is invoked with:

> `serverList` - the list of servers, which is managed through the Speedtest Custom application, ordered based on latency test results

```
onStarted
```

Triggered after `startTest` has been called, when the test has actually started. Handler is invoked with no arguments.

`onStageStarted`

Triggered when a test stage has started. Handler is invoked with:

`stageName` - the name of the stage

`onProgress`

Triggered when a test stage provides a progress update. Handler is invoked with:

`stageName` - the name of the stage
`result` - an object representing the current test result (contents vary by stage)
`progress` - a float representing the current test progress percentage (valid range: 0.0 - 1.0)

`onError`

Triggered when an error occurs. Note that an error doesn't necessarily result in a stage or test failure. Handler is invoked with:

`errorKey` - a short representation of the error
`errorMessage` - an untranslated text representation of the error
`errorData` - any extra data associated with the error. This parameter is undefined when error data is not available.

Stage Errors:

Error Key	Description
latency-connectionError upload-connectionError download-connectionError	• WebSocket creation failure [exception in _doCreateWebsocketConnection()] • Connection ready, then failure to send data [exception thrown in rc.start()] • XMLHttpRequest object errors and WebSocket library 'ws' errors are provided as 'connectionError' if the following is true: hi/hello succeeds and the connection is not in any of the following states: • Connecting • Closing • Closed
latency-connectionFailed upload-connectionFailed download-connectionFailed	• The XMLHttpRequest object errors and WebSocket library 'ws' errors are sent as 'connectionFailed' if any of the following is true: • hi/hello failed or the connection is in any of the following states: • Connecting • Closing • Closed N/A - for latency stage
latency-allConnectionsFailed upload-allConnectionsFailed download-allConnectionsFailed	• Dispatched when 'connectionFailed' error occurs for all connections N/A - for latency stage
latency-timeout upload-timeout download-timeout	• Dispatched when timeout occurs for all desired connections (configurable ratio of totalDesiredConnections via 'maxTimeoutConnectionsPercent' which is currently 100 percent) N/A - for latency stage

`onStageFinished`
Triggered when a test stage has finished. Handler is invoked with:
  `stageName` - the name of the stage
  `result` - the results object from the stage (contents vary by stage)

`onResultSaved`
Triggered when the test result upload has been completed. Handler is invoked with:
  `resultID` - the ID of the saved result

`onFinished`
Triggered when the entire test is finished. Handler is invoked with no arguments.

`onStageFailed`
Triggered when a test stage fails. Note that if a stage fails, `onStageFailed` will be invoked for the current stage only and then `onFailed` will be invoked. Handler is invoked with:
  `stageName` - the name of the stage

`onStageCanceled`
Triggered when a test is manually canceled. Note that if a test is canceled, `onStageCanceled` will be invoked for each remaining stage (including the current stage) and then `onCanceled` will be invoked. Handler is invoked with:
  `stageName` - the name of the stage

`onFailed`
Triggered when the test fails. Handler is invoked with no arguments.

`onCanceled`
Triggered when the test is manually canceled. Handler is invoked with no arguments.

Authorization

An API key uniquely associates the hostname where the SDK is deployed with an authorized SDK account. The API key is configured in a top-level configuration node named `apiKey`. All HTTP requests include the API key as a query string parameter for authorization. If an incorrect API key is provided, then the following error will be asynchronously thrown:

```
Error(‘Cannot start test. Ensure API key is configured properly: <API key>’);
```

State Machine Details

The JS SDK APIs are expected to be invoked in sequential order. A possible sequence of API calls could be:

```
1. sortServers()
2. startTest(serverUnderTest)
3. nextStage(‘download’)
4. nextStage(‘save’)
5. nextStage(‘upload’)
6. nextStage(‘save’)
7. stopTest()
```

After `sortServers` is invoked, wait for `onServersSorted` event before starting the test with a specific `serverUnderTest`. `startTest` is expected to be invoked once before individual test stages are invoked. And `startTest` can be invoked multiple times to restart the engine as long as there are no individual test stages executing (**note:** all test data will be lost when `startTest` is invoked, so be sure to properly save the test results first). After all test stages are performed, `stopTest` can be invoked once, however this is optional. In order to ensure each test stage is executed sequentially, wait for the corresponding `onStageFinished` event associated with each test stage. Test save is also considered a test stage and is also expected to be executed sequentially. Wait for the `onResultSaved` event to know when the save operation is completed. A test can only be saved after at least one test stage is executed (latency, download, or upload).

State Machine Errors

```
Error(‘Engine is currently executing <CURRENT> test stage, cannot restart engine now’)
```

The following errors are thrown by the JS SDK:

Use Case	Error Thrown

`startTest` is invoked while an individual test stage is executing	`Error(‘Engine is currently executing <CURRENT> test stage, cannot restart engine now’)`
	
	Note: this error contains the actual stage names in place of <CURRENT>. For example, if the test is being saved when `startTest` is invoked, then the error would be:
	
	`Error(‘Engine is currently executing save test stage, cannot restart engine now’)`

`stopTest` is invoked multiple times	`Error(‘Engine is already stopped, cannot stop engine again’)`
Individual test stages are attempted to be executed before `startTest` is invoked	`Error(‘Must start test before starting an individual test stage’)`
Attempt to save the test before executing a single test stage	`Error(‘Cannot save test before running latency, download, or upload first’)`

Attempt to execute test stages in parallel	`Error(‘Cannot start <NEXT> stage because <CURRENT> stage is running’)`
	
	Note: this error contains the actual stage names in place of <NEXT> and <CURRENT>. For example, if download stage is still running and upload stage is attempted to be executed in parallel, the error would be:
	
	`Error(‘Cannot start upload stage because download stage is running’)`

Note: when errors are thrown, the requested operation is not preformed and thus results in no-op.

Class: LatencyStage

Overview

Represents the latency portion of a speed test. Also used during server selection.

Result Object

The results object provided by `onProgress` and `onFinished` events.

> `latency` - the aggregate latency result from the test, as determined by the configured latency methodology (currently min)
> `jitter` - the aggregate jitter result from the latency test
> `serverVersion` - the reported version of the server being tested to
> `serverCapabilities` - the reported server capabilities of the server being tested to
> `clientAddress` - the ip address of this client, as reported by the server
> `connectionProtocol` - the connection protocol (WS, XHR) used for the stage
> `useSecureConnections` - boolean, represents whether the stage was performed across a TLS connection

Class: DownloadStage

Overview

Represents the download portion of a speed test.

Result Object

The results object provided by `onProgress` and `onFinished` events.

> `speed` - the final speed (throughput) calculation for the stage, in bytes per second
> `connectionProtocol` - the connection protocol (WS, XHR) used for the stage
> `useSecureConnections` - boolean, represents whether the stage was performed across a TLS connection

The results object provided by `onProgress` and `onFinished` events is just a number representing the measured download speed of the connection in bytes per second.

Class: UploadStage

Overview

Represents the upload portion of a speed test.

Result Object

The results object provided by `onProgress` and `onFinished` events.

> `speed` - the final speed (throughput) calculation for the stage, in bytes per second
> `connectionProtocol` - the connection protocol (WS, XHR) used for the stage
> `useSecureConnections` - boolean, represents whether the stage was performed across a TLS connection

Configuration Properties

In order to configure the engine beyond the default behavior, the EngineApi configuration method can be called with a nested object of desired configuration overrides, as detailed below:

Global Properties

Global settings not applicable to any particular stage, but to the Speedtest engine as a whole.

`apiKey [string]`
The API key for this deployment of the Speedtest SDK. Note: this value is required.
```
    default: null
```

`automaticStageProgression [boolean]`
Enables automatic progression to the next stage when each stage completes. Can be disabled to allow the application to handle animations or support UI interaction to progress to the next stage.
```
    default: true
```

`sdkBaseUrl [string]`
The bash URL for this deployment of the Speedtest SDK. Note: this value is required.
```
    default: null
```

`shortTests [boolean]`
When enabled, sets throughput stage durations to 3 seconds. Can be used in development deployments for quicker iteration. Note: this setting should not be enabled in production.
```
    default: false
```

`stagesList [array of string]`
The list and order of stages to use for the speedtest. Note that the ‘`save`’ stage can be repeated.
```
    default: [‘latency’, ‘download’, ‘upload’, ‘save’]
```

Section: `sampling`

Contains configuration that is applied to stages of the test that collect samples (latency and throughput, but not save).

`connectionProtocol [string: ws|xhr]`
The protocol to use to complete the stage. This defaults to `‘ws’` for the latency stage (as it’s a low-overhead protocol) and `‘xhr’` for the throughput stages, as browsers generally show better performance.
```
      default [latency]: ‘ws’
      default [throughput]: ‘xhr’
```

`connectionHardTimeoutMillis [integer]`
The timeout (in milliseconds) to allow before failing the connection.
```
      default: 20000
```

`connectionSoftTimeoutMillis [integer]`
The timeout (in milliseconds) to allow before attempting to failover to another protocol or port. When

performing a soft timeout, the timed out connection is kept open until the hard timeout is reached, or one of the connections succeeds, whichever happens first.

```
default: 3000
```

`maintenanceIntervalMillis [integer]`
The period (in milliseconds) between maintenance tasks of the connection pool. Maintenance tasks involve primarily opening new connections as needed by scaling to support higher bandwidth clients.

```
default: 500
```

Section: `latency`

Contains configuration specific to the latency stage of the speedtest.

`maxFailedConnections [integer]`
The number of connection failures to support before failing the stage.

```
default: 3
```

`maxServers [integer]`
When performing server selection, the total number of servers to test to.

```
default: 10
```

`numSamples [integer]`
The number of samples to use for calculating latency (ping).

```
default: 10
```

Section: `throughput`

Contains configuration specific to the throughput (download, upload) stages of the speed test and includes any configuration applied to the `stage` and `sampling` sections.

`connectionScalingBandwidthThreshold [integer]`
The bandwidth per connection (in bytes per second) to allow before adding additional connections when scaling up the speed test to support higher bandwidth clients.

```
default: 750000
```

`bufferLengthMillis [integer]`
The period (in milliseconds) to attempt to keep filled with throughput messages for a given connection. If the current calculated message length (as a factor of current speed and number of connections) is less than `bufferLengthMillis`, pipelining may be used to queue up additional messages.

```
default [download]: 3000
default [upload]: 1000
```

`durationSeconds [integer]`
The period (in seconds) to run the throughput stage for, calculated from when `numInitialConnections` connections are marked as open and ready.

```
default: 15
```

`maxConnections [integer]`
The maximum number of connections to open when scaling up a throughput test to support higher

bandwidth clients.

```
default: 32
default [xhr]: 6
```

`maxConnections [integer]`

The period (in milliseconds) to allow between messages from the server for a single connection before marking the connection as timed out and closing it.

```
default: 10000
```

`maxTimeoutConnectionsPercent [integer]`

The percentage of timed out connections to require (as a function of the currently desired number of open connections) before failing the stage.

```
default: 100
```

`messageLengthMillis [integer]`

When calculating the size of a throughput message to generate (either when sending data during upload, or requesting download data from the server), the period (in milliseconds) to attempt to size the message to, based on current calculated bandwidth and current number of open connections.

```
default: 1000
```

`messageTimeoutMillis [integer]`

The period (in milliseconds) to allow between messages from the server for a single connection before marking the connection as timed out and closing it.

```
default: 10000
```

`numInitialConnections [integer]`

The number of connections to initially create and verify connected (with an initial “hello” interaction with the Speedtest server) before starting the stage.

```
default: 1
default [throughput]: 4
```

`progressIntervalMillis [integer]`

The period (in milliseconds) between progress events from the stage.

```
default: 50
```

`maxMessageSizeBytes [integer]`

The maximum message size (in bytes) to use for upload/download messages.

```
default: 50
```

`minMessageSizeBytes [integer]`

The minimum message size (in bytes) to use for upload/download messages.

```
default: 50
```

`startMessageSizeBytes [integer]`

The default message size (in bytes) to use for upload/download messages.

```
default: 50
```

Section: download

Contains configuration specific to the download stage of the speedtest and includes any configuration applied to the `stage`, `sampling`, and `throughput` sections. (Note that no configuration options are documented here specific to download, but any options documented in `stage`, `sampling`, and `throughput` could be applied specifically to download only).

Section: upload

Contains configuration specific to the download stage of the speedtest and includes any configuration applied to the `stage`, `sampling`, and `throughput` sections. (Note that no configuration options are documented here specific to upload, but any options documented in `stage`, `sampling`, and `throughput` could be applied specifically to upload only).

Section: connectionProtocols

Can be used to create overrides specific to a given connection protocol, where a key in the section is the protocol (possible values: `'xhr'`, `'ws'`) and the value is an object of the same format as the documented configuration above. For example:

```
EngineApi.configure({
    connectionProtocols: {
        xhr: {
            download: {
                maxConnections: 4,
            },
        },
    },
});
```

would set the maximum connections for an upload test to 4, but only when the connection protocol is `'xhr'` (in other words if, based on browser-specific settings, the connection protocol was set to websockets instead, the above configuration would not be applied).

Section: browsers

Can be used to create overrides specific to a given browser, where a key in the section is the detected browser alias and the value is an object of the same format as the documented configuration above. Operates in the same manner as the `connectionProtocols` section.

Callflows and Code Examples

All code examples below are available via a react application (speedtest-js-demo-1.0.0.tgz will be available with your SDK package) which can be used for additional reference.

Overview

The JS SDK engine is exported as the class EngineApi which is expected to be imported by the client code.

- Importing the engine:

```
import {EngineApi} from '@ookla/speedtest-js-engine';
```

Initialization

- Configure the engine before use:

```
const engineOptions = {
  automaticStageProgression: true,
  jsEngine: {
    savePath: '/report',
    saveType: 'sdk'
  },
  saveResults: false,
  apiKey: 'abcdefghijklmnopqrstuvwxyz0123456789',
  sdkBaseUrl: 'https://www.example.com'
};
EngineApi.configure(engineOptions);
```

- A bare minimum configuration requires the API key and SDK base URL:

```
const engineOptions = {
  apiKey: 'abcdefghijklmnopqrstuvwxyz0123456789',
  sdkBaseUrl: 'https://www.example.com'
};
EngineApi.configure(engineOptions);
```

Listeners and API Overview

After engine initialization is completed successfully, listeners in the form of callback function handlers can be added to events propagated from the engine; engine APIs can also be invoked. The order of the API invocation is important and should correspond to the following callflows for proper internal state machine transitions within the engine.

Listeners

- Listen for the server selection completed event and print the server list in the console:
- Listen for the test start event and print a notification in the console:

```
EngineApi.onServersSorted.add((servers) => {
  let serversString = [...servers].map((server) =>
    JSON.stringify(server, undefined, 2));
  console.log('onServersSorted: ' + [...serversString]);
});
```

- Listen for the test stop event and print a notification in the console:

```
EngineApi.onStarted.add(() => {
  console.log('onStarted!');
});
```

- Listen for the error events and print a notification in the console:

```
EngineApi.onStarted.add(() => {
  console.log('onCanceled!');
});
```

- Listen for the saved results and print a notification in the console:

```
EngineApi.onError.add((errorKey, errorMessage, errorObject) => {
      console.log(`onError: key = ${errorKey}, msg = ${errorMessage}, obj =
${JSON.stringify(errorObject)}`);
});
```

```
EngineApi.onResultSaved.add((saveResult, testResult) => {
  console.log(`onResultSaved saveResult = ${JSON.stringify(saveResult)}
testResult = ${JSON.stringify(testResult)}`);
});
```

APIs

- Start the server selection process:
- Start a test with a specific server:

```
EngineApi.sortServers();
```

Note: in this example, `serverUnderTest` is expected to be an object in the same form (see

```
EngineApi.startTest(serverUnderTest);
```

"JSON schemas" section of this document) as the server objects returned in the array of servers provided by the server selection.

- Stop an ongoing test:

```
EngineApi.stopTest();
```

Use Cases:

- Retrieve an instance of the engine:

```
EngineApi = EngineApi.configure(engineOptions);
```

- Perform server selection to get a list of nearby servers in ascending order by latency:

```
EngineApi.sortServers();
EngineApi.onServersSorted.add((servers) => {
  this.serverUnderTest = servers[0]; // minimum latency server
});
```

- Run a throughput test to a given server with automatic stage advancement
  a. Configure the engine to enable automatic tests

```
const engineOptions = {
  automaticStageProgression: true
};
EngineApi = EngineApi.configure(engineOptions);
```

  b. Start the test with a specific server

```
EngineApi.startTest(serverUnderTest);
```

  c. Listen to test progress

```
EngineApi.onProgress.add((stageName, result) => {
  const {ratioComplete} = result;
  const value = stageName === 'latency' ? result.latency : result.speed;
  console.log(`${stageName} ${value} (${Math.floor(ratioComplete * 100)}%)`);
});
```

- Run a throughput test to a given server using manual stage advancement
  a. Configure the engine to disable automatic tests

```
const engineOptions = {
  automaticStageProgression: false
};
EngineApi = EngineApi.configure(engineOptions);
```

  b. Start the test

```
EngineApi.startTest(serverUnderTest)
```

  c. Manually advance each stage. **Note:** stages are expected to execute sequentially, therefore only start the next stage after the previous stage has finished. The event `onStageFinished` can be used to know when a stage has finished.

  - Listen for each stage to finish

```
EngineApi.onStageFinished.add((stageName, result) => {
  console.log(`onStageFinished(stageName = ${stageName}, result = ${JSON.strin-
gify(result)})`);
});
```

- Manually advance to next stage after each previous stage is finished

```
EngineApi.nextStage('latency');
... // wait for onStageFinished event for latency stage

EngineApi.nextStage('download');
... // wait for onStageFinished event for download stage

EngineApi.nextStage('save');
... // wait for onResultSaved event for save stage

EngineApi.nextStage('upload');
... // wait for onStageFinished event for upload stage

EngineApi.nextStage('save');
... // wait for onResultSaved event for save stage
```

- Stop a running test

```
EngineApi.stopTest();
```

Note: A test can be manually stopped after the test is started and before the test finishes. A running test can be stopped during a test stage or in between test stages.

- Run a download only test to a given server with automatic stage advancement

a. Configure the engine to enable automatic tests

```
const engineOptions = {
  automaticStageProgression: true,
  stagesList: ['download', 'save']
};
EngineApi = EngineApi.configure(engineOptions);
```

b. Start the test with a specific server

```
EngineApi.startTest(serverUnderTest);
```

c. Listen to test progress

```
EngineApi.onProgress.add((stageName, result) => {
  const {ratioComplete, speed} = result;
  console.log(`${stageName} ${speed} (${Math.floor(ratioComplete * 100)}%)`);
});
```

- Search for a server

  Servers can be searched for based on an input string which is used to match relevant server information.

```
EngineApi.searchServers({
    search: document.getElementById('searchInput').value
}).then((servers) => {
  console.log(JSON.stringify(servers));
});
```

- Update Test Result with Additional Test Stage(s)

  The `saveTest()` API can be invoked multiple times between `startTest()` and `stopTest()` in order to append additional test information onto the same test result, but should only be invoked after at least one test stage. This is helpful if you are only running a single stage by default but allow for additional stages. E.g. only running the download stage but offering the user a button to run the upload stage.

```
EngineApi.startTest(serverUnderTest)

EngineApi.nextStage('latency');
... // wait until onStageFinished event for latency stage
EngineApi.saveTest();
... // wait for onResultSaved event for save stage

EngineApi.nextStage('download');
... // wait until onStageFinished event for download stage
EngineApi.saveTest();
... // wait for onResultSaved event for save stage

EngineApi.nextStage('upload');
... // wait until onStageFinished event for upload stage
EngineApi.saveTest();
... // wait for onResultSaved event for save stage
```

- Static List of Servers

  Customers can provide a static list of servers which can be sorted in ascending order by latency using the `sortServers` API. See the “JSON Schemas” section of this document for the details of the server objects which are elements of the array passed into `sortServers`.

```
export const STATIC_SERVER_LIST =
[
  {
    "url": "http://seattle.speedtest.centurylink.net:8080/speedtest/upload.php",
    "lat": "47.6062",
    "lon": "-122.3321",
    "distance": 1122,
    "name": "Seattle, WA",
    "country": "United States",
    "cc": "US",
    "sponsor": "CenturyLink",
    "id": "8864",
    "preferred": 0,
    "https_functional": 1,
    "hostname": "seattle.speedtest.centurylink.net",
    "port": 8080,
  }, {
    "url": "http://speedtest01.wowrack.com:8080/speedtest/upload.php",
    "lat": "47.6062",
    "lon": "-122.3321",
    "distance": 1122,
    "name": "Seattle, WA",
    "country": "United States",
    "cc": "US",
    "sponsor": "Wowrack",
    "id": "6199",
    "preferred": 0,
    "https_functional": 1,
    "hostname": "speedtest01.wowrack.com.prod.hosts.ooklaserver.net",
    "port": 8080,
  },
. . .
. . .
. . .
];

EngineApi.sortServers(STATIC_SERVER_LIST);
```

- Save Test with Supplemental Data

  Customers can provide optional supplemental JSON which is inserted into the test result under the “supplemental” JSON node. Any valid JSON can be provided as long as the length of the resulting encoded JSON string is less than or equal to 1000 characters. You can invoke `setSupplementalSaveData` any time before the test is saved and the data will be included in the save. Note: this data persists across tests, so if there is data specific to a particular test, that should be cleared before the next test is taken.

```
const SUPPLEMENTAL_DATA = {
  "id": 123456789,
  "ip": "127.0.0.1",
};

EngineApi.setSupplementalSaveData({
  supplementalData: SUPPLEMENTAL_DATA,
});
```

- Clear Supplemental Data

```
EngineApi.setSupplementalSaveData(null);
```

JSON Schemas

Server Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "$id": "http://example.com/example.json",
    "type": "object",
    "title": "Root",
    "description": "Comprises the entire JSON document",
    "default": {},
    "examples": [
        {
            "url": "http://201.171.245.5:8080/speedtest/upload.php",
            "lat": "32.6517",
            "lon": "-115.4789",
            "distance": 209,
            "name": "Mexicali",
            "country": "Mexico",
            "cc": "MX",
            "sponsor": "INFINITUM",
            "id": "3419",
            "preferred": 1,
            "https_functional": 1,
            "host": "201.171.245.5.prod.hosts.ooklaserver.net:8080"
        }
    ],
    "required": [
        "url",
        "lat",
        "lon",
        "distance",
        "name",
        "country",
        "cc",
        "sponsor",
        "id",
        "preferred",
        "https_functional",
        "host"
    ],
    "additionalProperties": true,
    "properties": {
        "url": {
            "$id": "#/properties/url",
            "type": "string",
            "title": "URL",
            "description": "URL used for legacy speedtests",
            "default": "",
            "examples": [
                "http://201.171.245.5:8080/speedtest/upload.php"
            ]
        },
        "lat": {
            "$id": "#/properties/lat",
            "type": "string",
                "title": "lat",
                "description": "Latitude",
                "default": "",
                "examples": [
                    "32.6517"
                ]
            },
            "lon": {
                "$id": "#/properties/lon",
                "type": "string",
                "title": "lon",
                "description": "Longitude",
                "default": "",
                "examples": [
                    "-115.4789"
                ]
            },
            "distance": {
                "$id": "#/properties/distance",
                "type": "integer",
                "title": "Distance",
                "description": "",
                "default": 0,
                "examples": [
                    209
                ]
            },
            "name": {
                "$id": "#/properties/name",
                "type": "string",
                "title": "Name",
                "description": "",
                "default": "",
                "examples": [
                    "Mexicali"
                ]
            },
            "country": {
                "$id": "#/properties/country",
                "type": "string",
                "title": "Country",
                "description": "",
                "default": "",
                "examples": [
                    "Mexico"
                ]
            },
            "cc": {
                "$id": "#/properties/cc",
                "type": "string",
                "title": "cc",
                "description": "Country Code",
                "default": "",
                "examples": [
                    "MX"
                ]
            },
            "sponsor": {
                "$id": "#/properties/sponsor",
                "type": "string",
                "title": "Sponsor",
                "description": "",
                "default": "",
                "examples": [
                    "INFINITUM"
                ]
            },
            "id": {
                "$id": "#/properties/id",
                "type": "string",
                "title": "id",
                "description": "Identifier",
                "default": "",
                "examples": [
                    "3419"
                ]
            },
            "preferred": {
                "$id": "#/properties/preferred",
                "type": "integer",
                "title": "preferred",
                "description": "",
                "default": 0,
                "examples": [
                    1
                ]
            },
            "https_functional": {
                "$id": "#/properties/https_functional",
                "type": "integer",
                "title": "HTTPS Functional",
                "description": "",
                "default": 0,
                "examples": [
                    1
                ]
            },
            "host": {
                "$id": "#/properties/host",
                "type": "string",
                "title": "Host",
                "description": "",
                "default": "",
                "examples": [
                    "201.171.245.5.prod.hosts.ooklaserver.net:8080"
                ]
            }
        }
    }
```

Download Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "$id": "http://example.com/example.json",
    "type": "object",
    "title": "Download",
    "description": "Download stage finished",
    "required": [
        "speed",
        "connectionProtocol",
        "useSecureConnections",
    ],
    "properties": {
        "speed": {
            "$id": "#/properties/speed",
            "type": "integer",
            "title": "Speed",
            "description": "Speed in bytes per second",
            "default": 0,
            "examples": [
                2054433
            ]
        },
        "connectionProtocol": {
            "$id": "#/properties/connectionProtocol",
            "type": "string",
            "title": "Connection Protocol",
            "description": "XML Http Request (xhr) or Web Socket(ws)",
            "default": "",
            "examples": [
                "xhr",
                "ws"
            ]
        },
        "useSecureConnections": {
            "$id": "#/properties/useSecureConnections",
            "type": "boolean",
            "title": "Use Secure Connections",
            "description": "TLS is Enabled (true) or Disabled (false)",
            "default": false,
            "examples": [
                false
            ]
        }
    }
}
```

Upload Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "$id": "http://example.com/example.json",
    "type": "object",
    "title": "Upload",
    "description": "Upload stage finished",
    "required": [
        "method",
        "speed",
        "connectionProtocol",
        "useSecureConnections"
    ],
    "properties": {
        "method": {
            "$id": "#/properties/method",
            "type": "string",
            "title": "Method",
            "description": "Upload can be measured from the client (local) or
server (remote)",
            "default": "",
            "examples": [
                "remote"
            ]
        },
        "speed": {
            "$id": "#/properties/speed",
            "type": "integer",
            "title": "Speed",
            "description": "Speed in bytes per second",
            "default": 0,
            "examples": [
                230400
            ]
        },
        "connectionProtocol": {
            "$id": "#/properties/connectionProtocol",
            "type": "string",
            "title": "Connection Protocol",
            "description": "XML Http Request (xhr) or Web Socket (ws)",
            "default": "",
            "examples": [
                "xhr",
                "ws"
            ]
        },
        "useSecureConnections": {
            "$id": "#/properties/useSecureConnections",
            "type": "boolean",
            "title": "Use Secure Connections",
            "description": "TLS is Enabled (true) or Disabled (false)",
            "default": false,
            "examples": [
                false
            ]
        }
    }
}
```

Save Response Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "$id": "http://example.com/example.json",
    "type": "object",
    "title": "Save Result",
    "description": "",
    "required": [
        "resultid",
        "save"
    ],
    "properties": {
        "resultid": {
            "$id": "#/properties/resultid",
            "type": "string",
            "title": "Result Id",
            "description": "GUID used for persisted test result lookup",
            "default": "",
            "examples": [
                "ff87af4a-eaed-4b5b-9433-6100b3bd7382"
            ]
        },
        "save": {
            "$id": "#/properties/save",
            "type": "object",
            "title": "Save",
            "description": "",
            "default": {},
            "examples": [
                {}
           ]
       }
    }
}
```

Test Result Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "$id": "http://example.com/example.json",
    "type": "object",
    "title": "Test Result Schema",
    "description": "",
    "required": [
        "serverid",
        "testmethod",
        "source",
        "configs",
        "ping",
        "jitter",
        "guid",
        "closestPingDetails",
        "uploadMeasurementMethod",
        "upload",
        "download",
        "hash",
        "clientip"
    ],
    "properties": {
        "serverid": {
            "$id": "#/properties/serverid",
            "type": "string",
            "title": "Server Id",
            "description": "Server Unique Identifier",
            "default": "",
            "examples": [
                "14232"
            ]
        },
        "testmethod": {
            "$id": "#/properties/testmethod",
            "type": "string",
            "title": "Test Method",
            "description": "A comma separated list of connection protocols used
during test",
            "default": "",
            "examples": [
                "ws,xhr,xhr"
            ]
        },
        "source": {
            "$id": "#/properties/source",
            "type": "string",
            "title": "Source",
            "description": "Application Source",
            "default": "",
            "examples": [
                "sdk"
            ]
        },
        "configs": {
            "$id": "#/properties/configs",
            "type": "object",
            "title": "Configuration",
            "description": "Configuration used by the test",
            "default": {},
            "examples": [
                {
                    "downloadProtocol": "xhr",
                    "sdkBaseUrl": "http://www.example.com",
                    "latencyProtocol": "ws",
                    "automaticStageProgression": true,
                    "saveResults": false,
                    "engine": "js",
                    "serverBuild": "2019-09-17.2311.4b191db",
                    "apiKey": "abcdefghijklmnopqrstuvwxyz0123456789",
                    "host": "seattle.wa.speedtest.frontier.com",
                    "serverVersion": "2.7.4",
                    "port": 8080.0,
                    "uploadProtocol": "xhr",
                    "jsEngine": {
                        "savePath": "/report",
                        "saveType": "sdk"
                    }
                }
            ],
            "required": [
                "engine",
                "automaticStageProgression",
                "jsEngine",
                "saveResults",
                "apiKey",
                "sdkBaseUrl",
                "latencyProtocol",
                "downloadProtocol",
                "uploadProtocol",
                "host",
                "port",
                "serverVersion",
                "serverBuild"
            ],
            "properties": {
                "engine": {
                    "$id": "#/properties/configs/properties/engine",
                    "type": "string",
                    "title": "Engine Configuration",
                    "description": "",
                    "default": "",
                    "examples": [
                        "js"
                    ]
                },
                "automaticStageProgression": {
                    "$id": "#/properties/configs/properties/automaticStageProgres-
sion",
                    "type": "boolean",
                    "title": "Automatic Test",
                    "description": "",
                    "default": false,
                    "examples": [
                        true
                    ]
                },
                "jsEngine": {
                    "$id": "#/properties/configs/properties/jsEngine",
                    "type": "object",
                    "title": "Javascript Engine Configuration",
                    "description": "",
                    "default": {},
                    "examples": [
                        {
                            "savePath": "/report",
                            "saveType": "sdk"
                        }
                    ],
                    "required": [
                        "savePath",
                        "saveType"
                    ],
                    "properties": {
                        "savePath": {
                            "$id": "#/properties/configs/properties/jsEngine/proper-
ties/savePath",
                            "type": "string",
                            "title": "Save Path",
                            "description": "",
                            "default": "",
                            "examples": [
                                "/report"
                            ]
                        },
                        "saveType": {
                            "$id": "#/properties/configs/properties/jsEngine/proper-
ties/saveType",
                            "type": "string",
                            "title": "Save Type",
                            "description": "",
                            "default": "",
                            "examples": [
                                "sdk"
                            ]
                        }
                    }
                },
                "saveResults": {
                    "$id": "#/properties/configs/properties/saveResults",
                    "type": "boolean",
                    "title": "Save Results",
                    "description": "",
                    "default": false,
                    "examples": [
                        false
                    ]
                },
                "apiKey": {
                    "$id": "#/properties/configs/properties/apiKey",
                    "type": "string",
                    "title": "API Key",
                    "description": "",
                    "default": "",
                    "examples": [
                        "abcdefghijklmnopqrstuvwxyz0123456789"
                    ]
                },
                "sdkBaseUrl": {
                    "$id": "#/properties/configs/properties/sdkBaseUrl",
                    "type": "string",
                    "title": "SDK Base URL",
                    "description": "",
                    "default": "",
                    "examples": [
                        "http://www.example.com"
                    ]
                },
                "latencyProtocol": {
                    "$id": "#/properties/configs/properties/latencyProtocol",
                    "type": "string",
                    "title": "Latency Protocol",
                    "description": "",
                    "default": "",
                    "examples": [
                        "ws"
                    ]
                },
                "downloadProtocol": {
                    "$id": "#/properties/configs/properties/downloadProtocol",
                    "type": "string",
                    "title": "Download Protocol",
                    "description": "",
                    "default": "",
                    "examples": [
                        "xhr"
                    ]
                },
                "uploadProtocol": {
                    "$id": "#/properties/configs/properties/uploadProtocol",
                    "type": "string",
                    "title": "Upload Protocol",
                    "description": "",
                    "default": "",
                    "examples": [
                        "xhr"
                    ]
                },
                "host": {
                    "$id": "#/properties/configs/properties/host",
                    "type": "string",
                    "title": "Host",
                    "description": "",
                    "default": "",
                    "examples": [
                        "seattle.wa.speedtest.frontier.com"
                    ]
                },
                "port": {
                    "$id": "#/properties/configs/properties/port",
                    "type": "integer",
                    "title": "Port",
                    "description": "",
                    "default": 0,
                    "examples": [
                        8080
                    ]
                },
                "serverVersion": {
                    "$id": "#/properties/configs/properties/serverVersion",
                    "type": "string",
                    "title": "Server Version",
                    "description": "",
                    "default": "",
                    "examples": [
                        "2.7.4"
                    ]
                },
                "serverBuild": {
                    "$id": "#/properties/configs/properties/serverBuild",
                    "type": "string",
                    "title": "Server Build",
                    "description": "",
                    "default": "",
                    "examples": [
                        "2019-09-17.2311.4b191db"
                    ]
                }
            }
        },
        "ping": {
            "$id": "#/properties/ping",
            "type": "integer",
            "title": "Ping",
            "description": "",
            "default": 0,
            "examples": [
                46
            ]
        },
        "jitter": {
            "$id": "#/properties/jitter",
            "type": "number",
            "title": "Jitter",
            "description": "",
            "default": 0,
            "examples": [
                0.3333333333333333
            ]
        },
        "guid": {
            "$id": "#/properties/guid",
            "type": "string",
            "title": "Guid",
            "description": "",
            "default": "",
            "examples": [
                "ff87af4a-eaed-4b5b-9433-6100b3bd7382"
            ]
        },
        "closestPingDetails": {
            "$id": "#/properties/closestPingDetails",
            "type": "string",
            "title": "Closest Ping Details",
            "description": "",
            "default": "",
            "examples": [
                "[{\"server\":14232,\"jitter\":3,\"latency\":44},{\"-
server\":5904,\"jitter\":1,\"latency\":45},{\"server\":8864,\"jitter\":3,\"laten-
cy\":46},{\"server\":1782,\"jitter\":0.5,\"latency\":46},{\"server\":11329,\"jit-
ter\":1,\"latency\":46},{\"server\":10395,\"jitter\":3,\"latency\":47},{\"serv-
er\":6199,\"jitter\":3.5,\"latency\":48},{\"server\":5033,\"jitter\":0.5,\"laten-
cy\":48},{\"server\":22168,\"jitter\":0.5,\"latency\":48},{\"server\":6057,\"jit-
ter\":1,\"latency\":86}]"
            ]
        },
        "uploadMeasurementMethod": {
            "$id": "#/properties/uploadMeasurementMethod",
            "type": "string",
            "title": "Upload Measurement Method",
            "description": "",
            "default": "",
            "examples": [
                "remote"
            ]
        },
        "upload": {
            "$id": "#/properties/upload",
            "type": "integer",
            "title": "Upload",
            "description": "Upload speed in kilobits per second",
            "default": 0,
            "examples": [
                2113
            ]
        },
        "download": {
            "$id": "#/properties/download",
            "type": "integer",
            "title": "Download",
            "description": "Download speed in kilobits per second",
            "default": 0,
                "examples": [
                    36961
                ]
            },
            "hash": {
                "$id": "#/properties/hash",
                "type": "string",
                "title": "Hash",
                "description": "",
                "default": "",
                "examples": [
                    "cffde8a0812351424db9c30ebcd978e1"
                ]
            },
            "clientip": {
                "$id": "#/properties/clientip",
                "type": "string",
                "title": "Client IP",
                "description": "",
                "default": "",
                "examples": [
                    "71.223.231.180"
                ]
            }
        }
}
```