DareBoost API User Documentation

The DareBoost API allows you to easily integrate your performance metrics into your continuous integration services and your applications.

URL / Access

DareBoost provides a base URL to access the API : https://www.dareboost.com/api/0.2/

All API calls begin with this base URL, from which you add the action you want.

Example : https://www.dareboost.com/api/0.2/analysis/launch

Note that the API requires the use of the HTTPS protocol and all the routes are **ONLY** reachable with the **HTTP POST** method (no GET, PUT or DELETE). The POST data is defined in the **JSON** format.

Authentication

Each request must be authenticated for being processed. So you need a token, that you can directly retrieve from your backoffice. You can generate it again if necessary.

This token is sent as a param of your POST requests.

The token looks like this : 9SD9FDS8FDS8DCX8V8CXV4

Quotas

The DareBoost API is based on a credit system. Each DareBoost plan gives you an amount of credits.

When you launch an analysis from the API that generates a valid report, this will cost you 1 credit.

If you want additional metrics like visual metrics (first / last visual change, speedindex..) this will cost you 2 credits!

Others requests (failed analysis, get configuration, get a report, etc) don't cost anything.

Response code

When you call the DareBoost API, the response always contains a status code that permits you to check if the request was successfully processed.

Overview

The table below lists the differents status codes and their explanation:

Status code	Explanation
>= 200 && < 400	the request was successfully processed. You get a different message depending on your action.

>= 400	the client's request could not be understood or treated by the server.

Details

- **200: OK**
- **202 : The analysis is currently processing**
- **204: No data available (e.g. you try to access to the last report of a monitoring, but no data was further processed)**
- **206: The analysis report is not complete (missing W3C result, or missing some rules)**
- **400: Missing argument or bad value for the argument (one of the required parameters is missing or doesn't respect the specifications)**
- **401: Authentication required (no token or invalid token)**
- **403: Action forbidden (no remaining credit, too many simultaneous analysis, unauthorized url)**
- **404: Page is unreachable (unknown url / check server response failed)**
- **406: Not a valid json format**
- **408: The analysis has timeout**
- **417: The last analysis of the monitoring results in error**
- **500: Internal server error (unknown error, contact us)**

API routes explanations

Routes overview

Action	Url
Launch an analysis	https://www.dareboost.com/api/0.2/analysis/launch
Get the report of a previously launched analysis	https://www.dareboost.com/api/0.2/analysis/report
Get the available configuration	https://www.dareboost.com/api/0.2/config
Get the last report for the requested monitoring	https://www.dareboost.com/api/0.2/monitoring/last-report
Get the list of your monitorings	https://www.dareboost.com/api/0.2/monitoring/list
Get a list of reports from one of your monitoring	https://www.dareboost.com/api/0.2/monitoring/reports

All these routes return the 200 HTTP code if they are successfully processed.

If you call a route that doesn't exists, you will get the 404 Not Found HTTP code.

Routes details

Launch an analysis

To facilitate the launch of an analysis, we provide default settings.

- Location : a random location is selected
- Browser : a random browser is selected
- Advanced settings : the default settings depends on the location/browser pair

It means that if you process two requests in a row, with the **default settings**, you are **not sure to get the same configuration** !

So the default analysis request, using the JSON format, is :

Basic request

```
POST https://www.dareboost.com/api/0.2/analysis/launch {
 // required parameters
 token: "", // String, The token to authenticate the user
 lang: "", // String,  Optional, The lang of the analysis.
           // Default value: "en". Possible values: "en", "fr".
 url: "", // String, The URL to analyze
 // All the following parameters are optional
 location: "", // String, Optional, The location to use.
              // Default value: random. Possible values: "paris", "new york"

 // browser
 browser: {
  name: "", // String, Optional, The browser name to use.
           // Default value: random. Possibles value: "firefox", "chrome"
  version: "" // String, Optional, The browser version to use.
            // Default value: random. Possibles value: depending on the browser
 },
 mobileAnalysis: false // Boolean, Optional, set to true if you want your page to
be analysed in a mobile context. This param is override if you define a "browser"
key. Default value: false
 isPrivate: boolean, // Boolean, Other users have access to the report, or not.
Default value: false

 visualMetrics: boolean, // Boolean, Default value: false. Set to true if you want
to have additional visual metrics ( start render, last visual change, speedindex)
Be careful activate this option will cost you more than one credit.
 // bandwidth
 bandwidth: {
  upstream: Number (integer 32 bits), // The upstream bandwidth
  downstream: Number (integer 32 bits), // The downstream bandwidth
 },
 // latency
 latency: Number (integer 32 bits), // The network latency

 // screen
 screen: {
  height: Number (integer 32 bits), // The height of the screen
  width: Number (integer 32 bits) // The width of the screen
 },

 // basic authentication
 basicAuth: {
  user:"", // String, The user
  password:"", // String,  The password
 }

 // post data
 postData: [ // Array, The posts data,
  {
   name: "",
   value: ""
  },
  ...
 ],

  // headers
 headers: [ // Array, The headers
  {
   name: "",
   value: ""
  },
  ...
 ],
   'cookie':[  // Array, list of cookies added on HTTP request (depend on domain
and path)
        {
            'name': '',
            'value': '',
            'domain': '',
            'path': ''
        }
    ],
 // blacklist and whitelist
    blacklist: [ // Array
  "", "" ... // Regex for blacklist request
 ],
    whitelist: [ // Array
```

```
  "", "" ... // Regex for whitelist request
 ]
}
```

Response

```
return: {
    status: Number (integer 32 bits), // The status code for the request itself
    message: "", // String, A message to specify the status
    reportId: "", // String, the report ID to use to get the report. Only present
if the request status is ok.
}
```

cURL example :

Launch an analysis to http://example.org with default option

```
curl --ssl-reqd -H "Content-Type: application/json" -X POST
 -d '{"token":"9SD9FDS8FDS8DCX8V8CXV4", "url":"http://example.org"}' \
 https://www.dareboost.com/api/0.2/analysis/launch
```

Explanation :

--ssl-reqd : Require SSL/TLS for the connection. Terminates the connection if the server doesn't support SSL/TLS.

-H "Content-Type: application/json": specify the request data format.

-X POST: Specifies a POST command

-d : Add post data in the JSON format, here we had the user token for authentication and the url to analyze.

And then come the DareBoost API url.

Launch an analysis to http://example.org with some option

```
curl --ssl-reqd -H "Content-Type: application/json" -X POST
 -d '{
   "token":"9SD9FDS8FDS8DCX8V8CXV4", \
    "url":"http://example.org", \
    "bandwidth": { \
    "upstream":3072, \
     "downstream":10240 \
    }, \
   "latency":40, \
    "headers":[ \
    {"name":"User-Agent", "value":"DareBoost"}, \
     {"name":"From", "value":"contact@dareboost.com"} \
   ] \
 }' \
https://www.dareboost.com/api/0.2/analysis/launch
```

Explanation :

In this example, we just added more option like the upstream & downstream used. Moreover, we added two header to the analysis : User-Agent and From.

Get the report of a previously launched analysis

Request

```
POST https://www.dareboost.com/api/0.2/analysis/report {
    token: "", // String, The token to authenticate the user
    reportId: "" // String, The report id to use to get the report, obtained when
launch an analysis from the API.
 summary: boolean // Boolean, Set to true to get a summary of the report with only
the metrics and without advices
}
```

Response

```
return: {
    status: Number (integer 32 bits), // The status code for the request itself
    message: "", // String, A message to specify the status
 missing: [], // Array : can contain the following values: rules, w3c HTML, w3c CSS
or performance_timings
    report: {} // Object, only present if the analysis is finished . The report
corresponding to the given reportId. Details are described below
}
```

cURL example :

Get the report from a previously launched analysis

```
curl --ssl-reqd -H "Content-Type: application/json" -X POST
 -d '{ \
    "token": "9SD9FDS8FDS8DCX8V8CXV4", \
    "reportId": "54084e3ce4b0f241b256c51a" \
 }' \
 https://www.dareboost.com/api/0.2/analysis/report
```

Explanation :

--ssl-reqd : Require SSL/TLS for the connection. Terminates the connection if the server doesn't support SSL/TLS.

-H "Content-Type: application/json": specify the request data format.

-X POST: Specifies a POST command

-d : Add post data in the JSON format, here we had the user token for authentication and the report id.

And then come the DareBoost API url.

Get available configuration

Request

```
POST https://www.dareboost.com/api/0.2/config {
    token: "" // String, The token to authenticate the user
}
```

Response

```
return: {
    status: Number (integer 32 bits), // The status code for the request itself
    message: "", // String, A message to specify the status
 config :[ // Array, An array of location/browsers
            {
                location:"", // String, The location
    isPrivate: boolean, // Boolean if the location is a private location
                browsers: [ // Array, The available browsers for the location
                    {
                        name: "", // String, The name of the browser
                        version: "", // String, The version of the browser
                        isMobile: boolean, // Boolean, Is the browser mobile or not
                    },
                    ...
                ],
            },
            ...
    ],
}
```

cURL example :

Get locations and browsers available on DareBoost

```
curl --ssl-reqd -H "Content-Type: application/json" -X POST
 -d '{ \
    "token": "9SD9FDS8FDS8DCX8V8CXV4" \
  }' \
 https://www.dareboost.com/api/0.2/config
```

Explanation :

--ssl-reqd : Require SSL/TLS for the connection. Terminates the connection if the server doesn't support SSL/TLS.

-H "Content-Type: application/json": specify the request data format.

-X POST: Specifies a POST command

-d : Add post data in the JSON format, here we had the user token for authentication.

And then come the DareBoost API url.

Get available browser for a location

Request

```
POST https://www.dareboost.com/api/0.2/config {
    token: "", // String, The token to authenticate the user
 location: "" // String, The location
}
```

Response

```
return: {
    status: Number (integer 32 bits), // The status code for the request itself
    message: "", // String, A message to specify the status
 location: "",
 isPrivate: boolean, // Boolean if the location is a private location
 browsers: [ // Array, The available browsers for the requested location
         {
             name: "", // String, The name of the browser
                version: "", // String, The version of the browser
                isMobile: boolean, // Boolean, Is the browser mobile or not
   },
            ...
 ],
}
```

cURL example :

Get available browsers for Paris location

```
curl --ssl-reqd -H "Content-Type: application/json" -X POST
 -d '{ \
    "token": "9SD9FDS8FDS8DCX8V8CXV4", \
    "location": "paris" \
 }' \
 https://www.dareboost.com/api/0.2/config
```

Explanation :

--ssl-reqd : Require SSL/TLS for the connection. Terminates the connection if the server doesn't support SSL/TLS.

-H "Content-Type: application/json": specify the request data format.

-X POST: Specifies a POST command

-d : Add post data in the JSON format, here we had the user token for authentication and the location.

And then come the DareBoost API url.

Get your monitorings list

Request

```
POST https://www.dareboost.com/api/0.2/monitoring/list {
    token: "", // String, The token to authenticate the user
    name: "", // String, Optional, a string pattern to filter your monitoring and
return only those that contain the pattern in their name
    url: "", // String, Optional, a string pattern to filter your monitoring and
return only those that contains the pattern in their url
}
```

Response

```
return: {
    status: Number (integer 32 bits), // The status code for the request itself
    message: "", // String, A message to specify the status
    monitorings: [  // Array, Optional, An array listing your monitorings
            {
    id: Number (integer 64 bits), // The unique identifier of your monitoring
    url: "", // String, the url of the monitored webpage
    name: "", // String, the name of your monitoring
    state: "", // String, one of following values: OK, ALERT, ERROR
    errorMessage: "", // String, Optional, only if state is ERROR, that will
describe the error
    lastExecution: Number (integer 64 bits), // Optional, The timestamp of the last
execution. Not defined if the monitoring has not been executed yet.
    enabled: Boolean // Boolean, If the monitoring is enabled or not.
            }
    ],
}
```

cURL example :

Get the list of all your monitorings

```
curl --ssl-reqd -H "Content-Type: application/json" -X POST
 -d '{ \
    "token": "9SD9FDS8FDS8DCX8V8CXV4", \
    "url": "dareboost" \
 }' \
 https://www.dareboost.com/api/0.2/monitoring/list
```

Explanation :

--ssl-reqd : Require SSL/TLS for the connection. Terminates the connection if the server doesn't support SSL/TLS.

-H "Content-Type: application/json": specify the request data format.

-X POST: Specifies a POST command

-d : Add post data in the JSON format, here we had the user token for authentication and filters.

And then come the DareBoost API url.

Get the last report of a monitoring

Request

```
POST https://www.dareboost.com/api/0.2/monitoring/last-report {
    token: "", // String, The token to authenticate the user
    monitoringId: Number (integer 64 bits), // The id of the monitoring to get the
last execution from
}
```

Response

```
return: {
    status: Number (integer 32 bits), // The status code for the request itself
    message: "", // String, A message to specify the status
 lastExecution: Number (integer 64 bits), // The timestamp of the last audit.
    alerts: [  // Array, Optional, An array listing the alerts triggered by the
audit
            {
                type: "", // String, The type of the alert. Can be : "GLOBAL_NOTE",
"REQUEST", "WEIGHT", "FIRST_BYTE", "START_RENDER", "VISUALLY_COMPLET", "TIME",
"SPEED_INDEX" or "RESOURCE_IN_ERROR"
                threshold: Number (integer 64 bits), // The threshold defined for
the alert
                value: Number (integer 64 bits), // The  value of the alert type
for the audit
            }
    ],
 missing: [], // Array : can contain the following values: rules, w3c HTML, w3c CSS
or performance_timings
    report: {}, // Object, The report corresponding to the last audit of the
monitoring. Optional if the analysis is not finished or has failed. Details are
described below
    }
}
```

cURL example :

Get the last monitoring report for a given id

```
curl --ssl-reqd -H "Content-Type: application/json" -X POST
 -d '{ \
    "token": "9SD9FDS8FDS8DCX8V8CXV4", \
    "monitoringId": 404 \
 }' \
 https://www.dareboost.com/api/0.2/monitoring/last-report
```

Explanation :

--ssl-reqd : Require SSL/TLS for the connection. Terminates the connection if the server doesn't support SSL/TLS.

-H "Content-Type: application/json": specify the request data format.

-X POST: Specifies a POST command

-d : Add post data in the JSON format, here we had the user token for authentication and the monitoring id.

And then come the DareBoost API url.

Get reports of one of your monitoring

Request

```
POST https://www.dareboost.com/api/0.2/monitoring/reports {
    token: "", // String, The token to authenticate the user
    monitoringId: Number (integer 64 bits), // The monitoring id to get the last
execution
 limit: Number (integer 32 bits), // Default 30. limit the number of results (0 =
no limit)
 date: Number (integer 64 bits), // Optional, Timestamp to precise from when you
want reports (from 'date' to now)
    error: Boolean, // Boolean, Optional, If true, returns only audits marked as
error, if false, only returns those that are not marked as error. If not precise,
both will be returned
}
```

Response

```
return: {
    status: Number (integer 32 bits), // The status code for the request itself
    message: "", // String, A message to specify the status
    monitoringData: [  // Array, Optional, An array listing the reports of the
monitoring (only overall metrics)
            {
    id: Number (integer 64 bits), // The unique identifier of the report
    date: Number (integer 64 bits), // Optional, Timestamp of the date of execution
    score: Number (integer 32 bits), // The score computed by DareBoost for the
analyzed url
    weight: Number (integer 64 bits), // The weight of the page
    requests: Number (integer 32 bits), // The number of requests
    unreachableRequests: Number (integer 32 bits), // The number of requests with a
bad status (ie: 4XX, 5XX )
       timings: {
           firstByte: Number (integer 64 bits), // The delay from the navigation
starts to the first byte received by the client side
           firstPaint: Number (integer 64 bits), // Optional. Only available on
Chrome and Mobile (Not in Firefox) The delay from the navigation start to the first
display. (returned by the browser)
           domInteractive: Number (integer 64 bits), // The delay between the
navigation start and the user can interact with the web page.
           loadEventEnd: Number (integer 64 bits), // The delay between the
navigation start and the load event of the current document is completed.
     startRender: Number (integer 64 bits), // Optional. The delay between the
navigation start and the first visual change (obtained through video)
     speedIndex: Number (integer 64 bits), // Optional. Performance Index
transcribing the rendering speed of the above the fold part of the page (page's
area visible  without having to scroll). The faster the rendering is, the smaller
the speedindex will be. Google recommendation: less than 1000. (obtained through
video)
     visuallyComplete: Number (integer 64 bits), // Optional. The delay between the
navigation start and the last visual change occur (obtained through video)
       },
    state: "", // String, one of following values: OK, ERROR
    errorMessage: "", // String, Optional, only if state is ERROR, this will
describe the error
            }
    ],
}
```

cURL example :

Get the last monitoring report for a given id

```
curl --ssl-reqd -H "Content-Type: application/json" -X POST
 -d '{ \
    "token": "9SD9FDS8FDS8DCX8V8CXV4", \
    "monitoringId": 404 \
 }' \
 https://www.dareboost.com/api/0.2/monitoring/reports
```

Explanation :

--ssl-reqd : Require SSL/TLS for the connection. Terminates the connection if the server doesn't support SSL/TLS.

-H "Content-Type: application/json": specify the request data format.

-X POST: Specifies a POST command

-d : Add post data in the JSON format, here we had the user token for authentication and filters.

And then come the DareBoost API url.

Report details

The report object is the same for standalone analysis and audits from monitoring. See details:

```
report:{
 publicReportUrl: "https://www.dareboost.com/[lang]/report/[reportId]",
    harFileUrl: "https://www.dareboost.com/download/harFile/[reportId]",
    date: Timestamp, // Timestamp, the timestamp when the audit has been done
    url: "", // String, the analyzed URL
 lang: "", // String, the lang used for the analysis
    config: { // Object, this object describe the settings used for the analysis
        location:"", // String, the location used for the analysis
        browser: { // Object, describe the browser used for the analysis
            name: "", // String, the name of the browser
            version: "", // String, the version of the browser
        },
        isMobile: boolean, // Boolean, is the analysis  mobile or not?
        bandwidth: { // Object, It describes the bandwidth
            upstream: Number (integer 32 bits), // The upstream bandwidth
            downstream: Number (integer 32 bits), // The downstream bandwidth
        },
  latency: Number (integer 32 bits), // the network latency
  isPrivate: boolean, // Boolean, The report is private or not?
  screen: { // Object
   height: Number (integer 32 bits), // The height of the screen
   width: Number (integer 32 bits), // The width of the screen
  },
  basicAuth: { // Object, Optional not present if no basic auth defined
   user:"", // String, The user
   password:"", // String, The encoded password
  },
  postData: [ // Object, Optional not present if no post data defined
   {
    key:"", // String, The post data key
    value:"",  // String, The post data value
   },
   ...
  ],
  header: [ // Object, Optional not present if no header defined
   {
    key:"", // String, The header key
    value:"",  // String, The header value
   },
   ...
  ],
    },
    summary:{
        loadTime: Number (integer 64 bits), // The loadtime of the analyzed url
        score: Number (integer 32 bits), // The score computed by DareBoost for the
analyzed url
        requestsCount: Number (integer 32 bits), // The number of requests
        weight: Number (integer 64 bits), // The weight of the page
    },
    categories: [ // Array, show each score by category
        {
            mark: Number (integer 32 bits), // The score for the category
            name: "", // String, The name of the category
        },
        ...
    ],
    tips: [ // Array, The list of tips from DareBoost
        {
            advice: "", //  String, The contents of the advice
            category: "", // String, The category of the advice
            score: Number (integer 32 bits), // The score for this advice, note
that -1 matches an information tip
            name: "", // String, The title of the advice
   priority: Number (integer 32 bits), // The priority for this adivce, higher
value means higher priority
        },
        ...
    ],
    w3cValidators: { // Object, the result of the W3C
        CSS: { // CSS part
   status: "", // String, The status too explain if the page is valid or not, Value
: "OK", "KO", "UNAIVALABLE"
            errorsCount: Number (integer 32 bits), // Optional if status is
UNAVAILABLE. The number of CSS errors
            warningsCount: Number (integer 32 bits), // Optional if status is
UNAVAILABLE. The number of CSS warnings
            warnings: [ // Array, Optional if no warning. The list of warnings.
    {
     url: "", // String, The url of the page containing the warning
     warnings: [ // Array, The list of CSS warnings for the url
      {
       line: Number (integer 32 bits), // The line of the warning
       source: "", // String, The source of the warning
       message: "", // String, The explanation of the warning
      },
     ]
    },
    ...
   ],
            errors:[ // Array, Optional if no errors. The list of errors.
    {
     url: "", // String, The url of the page containing the error
     errors: [ // Array, The list of CSS errors for the url
      {
       line: Number (integer 32 bits), // The line of the error
       source: "", // String, The source of the error
       type: "", // String, the type of the error (example : parse-error)
       message: "", // String, The explanation of the error
      },
     ]
    },
    ...
   ],
        },
        HTML: { // HTML part
   status: "", // String, The page status: is valid or not, Value : "OK", "KO",
"UNAIVALABLE"
   errorsCount: Number (integer 32 bits),  // Optional if status is UNAVAILABLE.
The number of HTML errors
            warningsCount: Number (integer 32 bits),  // Optional if status is
UNAVAILABLE. The number of HTML warnings
   charset: "", // String, The charset identify for the page (example: UTF-8)
   doctype: "", // String, The doctype of the document
   url: "", // String, The url of the page containing the warning
            warnings: [ // Array, Optional if no warning. The list of warnings.
    {
     line: Number (integer 32 bits), // The line of the warning
     col: Number (integer 32 bits), // The column on the line of the warning
     source: "", // String, The source of the warning
     message: "", // String, The quick explanation of the warning
     details: "", // String, The long explanation of the warning
    },
    ...
   ],
            errors:[ // Array, Optional if no errors. The list of errors.
    {
     line: Number (integer 32 bits), // The line of the error
     col: Number (integer 32 bits), // The column on the line of the error
     source: "", // String, The source of the error
     message: "", // String, The quick explanation of the error
     details: "", // String, The long explanation of the error
    },
    ...
   ],
        },
    },
    performanceTimings: { // Object, Optional if DareBoost failed to get it, The
performance timings results
  navigationStart: Number (integer 64 bits), // The time when the browser has
finished to unload the previous page
        firstByte: Number (integer 64 bits), // The time when the client receive
the first byte
        firstPaint: Number (integer 64 bits), // Optional. Only available on Chrome
and Mobile (Not in Firefox) The time when the browser will first display something.
        domInteractive: Number (integer 64 bits), // The time when the user can
interact with the web page.
        loadEventEnd: Number (integer 64 bits), // The time when the load event of
the current document is completed.
  startRender: Number (integer 64 bits), // Optional. The first visual change
  speedIndex: Number (integer 64 bits), // Optional. Performance Index transcribing
the rendering speed of the above the fold part of the page (page's area visible
without having to scroll). The faster the rendering is, the smaller the speedindex
will be. Google recommendation: less than 1000.
  visuallyComplete: Number (integer 64 bits), // Optional. The time when the last
visual change occur
    },
    technos: [ // Array, the list of each technology detected on the page
        {
   name: "", // String, the name of the technology
   version: "" //  String, Optional if not found, The version of the technology
  },
        ...
    ],
}
```

Advanced params : units, minimum and maximum authorized values

Some settings have limitations and define a unit :

Setting	unit	minimum	maximum	forbidden characters
bandwidth (upstream and downstream)	kilo-octet	50	-	-
latency	ms	0	5 000	-
screen width	pixel	150	1920	-
screen height	pixel	150	3000	-
headers	-	-	-	=,;:space\t\r\n\v\f
post data	-	-	-	-
basic authentication	-	-	-	-