Performance Analyser API

Documentation version 1.2

Contents

1 Introduction

This document is a reference guide to how the API can be used to extract data programmatically from Performance Analyser, or how you can run new jobs.

You could use the API, for example, to generate local dashboards or integrate Performance Analyser and performance checks into your continuous integration or continuous deployment solutions.

2 API Path

With the exception of authentication (which is detailed below), all Performance Analyser API requests are made this way. The API references in this document assume that the following path is being used:

PATH	`https://api.nccgroup-webperf.com/pa/1`

Also note that all requests must be via HTTPS.

3 Authentication

Access to the API will always require the use of an authentication token. Once you have this, you will use the token with each request you make to the API.

To request the API token, you currently need your standard login credentials (the same as when you log into the portal) and a ***client_id*** and ***client_secret***.

If you don't have these, please get in contact with us and we can provide them.

REQUEST

METHOD	`POST`
PATH	`https://api.nccgroup-webperf.com/authorisation/token`
HEADER	`Authorization: Basic` ***{base64 encoded client_id:client_secret}***
BODY	`username=`***{email_address}***`&password=`***{password}***`&grant_type=password`

The request needs to include a base64 encoded string of your ***client_id*** and ***client_secret***, separated by a colon character.

For example, if your ***client_id*** is **client** and your ***client_secret*** is **secret**, then you would base64 encode the whole string **client:secret** which is **Y2xpZW50OnNlY3JldA==**

RESPONSE

BODY	{
	"access_token": "gdjgurtyiuerytiuertyiuteyituruie",
	"token_type": "Bearer",
	"expires_in": 43200
	}

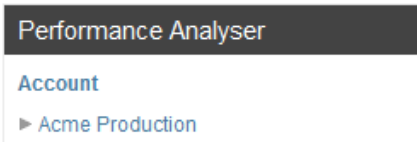
PARAMETERS

PARAM	DETAILS
`access_token`	This is the token you will use in all subsequent API requests, as documented in each of the requests
`expires_in`	This defines how long the token is valid for, in seconds

If you attempt to use a token that has expired, you will get an `HTTP 401 Unauthorized` response and will need to get a new token as described above.

4 Working with Accounts (Realms)

When you log into the Portal Hub and click through to your Performance Analyser account, this is equivalent to the **`realm`** that you will refer to when using the API.



All requests to the API need to include a **`realm`** to control where you want to get data from, or which account you want to run new jobs in.

REQUEST

METHOD	`GET`
PATH	`https://api.nccgroup-webperf.com/authorisation/user?service=6`
HEADER	`Authorization: Bearer` ***`{access_token}`***

When using the API for Performance Analyser, the **`service`** is always **`6`**, so use that to query for your realms (accounts).

RESPONSE

BODY

```
{
  "realms": {
    "1234": {
      "label": "Acme Production",
      "service": "6",
      ...
    },
    ...
  },
  "info": {
    "fullName": "Joe Bloggs",
    "emailAddress": "joe.bloggs@mydomain.com"
  },
  "roles": [
    "MONITORING",
    "USER_ADMIN",
    "ACCOUNT_ADMIN",
    "RT",
    "PA"
  ]
}
```

In the above example, if you wanted to use the API to work in your **`Acme Production`** account, you would specify the realm **`1234`** in all subsequent API requests.

5 Working with the API

5.1 Dates

All date/times will be in the ISO8601 format. This includes any returned data, as well as any queries that allow you to filter on date/time.

This ensures that all date/times can be used between responses and requests, as well as being in a standard format that is well supported across different programming languages, and is human readable.

The ISO standard insists on the timezone/offset being included, which again ensures there is no ambiguity as to which date/time is being used (i.e. the timezone/offset is mandatory: `2016-04-07T08:06:47+00:00`).

All dates will be returned by the API in UTC, for example: `2016-04-07T08:06:47+00:00`

5.2 Duration Metrics

To ensure consistency, all durations are specified in seconds when using the API.

There are some values that are specified differently in the portal (for human readability), but these have all been defined in seconds, to ensure consistency within the API.

For example the following are specified in seconds in the API, which is different from what you will see in the portal:

ITEM	METRIC IN PORTAL	METRIC IN API
Inactivity timeout	Milliseconds, e.g. 500	Seconds, e.g. 0.5
Latency	Milliseconds, e.g. 200	Seconds, e.g. 0.2
Crawl timeout	Minutes, e.g. 180	Seconds, e.g. 10800

5.3 URL Encode Parameters

All parameter values that are sent to the API in query strings need to be URL encoded.

Query strings are typically used to filter the query, for example to limit the search to a particular date/time period or to get something from the API that relates one thing to another.

More details can be found in each section below, but if we take an example of searching for all objects (network requests) that have been made within a test run, we are searching for **`objects`** for a given **`testRun`**, as follows:

PATH	`/objects?testRun=`***`{sref_of_test_run}`***

The **`testRun`** and ***`sref_of_test_run`*** are query strings sent as parameters to the **`/objects`** endpoint.

An example ***`sref_of_test_run`*** could be **`testRuns/12345`**. Only the value of the query string is encoded:

PATH	`/objects?testRun=testRuns/12345`

Do not URL encode this — URL encode this

Doing this, we end up making the following request:

PATH	`/objects?testRun=testRuns%2F12345`

Another common case (as documented later) is to include date/time parameters in your query. As these need to be encoded, you **cannot** issue a query like this:

✘

PATH	`/testRuns?fromDate=2016-07-01T00:00:00+00:00&toDate=2016-07-31T23:59:59+00:00`

You need to URL encode the date/time parameters values being passed as query strings, which ends up as:

✓

PATH	`/testRuns?fromDate=2016-07-01T00%3A00%3A00%2B00%3A00&toDate=2016-07-31T23%3A59%3A59%2B00%3A00`

Remember it is only the query parameter values that are encoded, which is why the `&toDate=` in the above example is not encoded whereas the value `2016-07-31T23:59:59+00:00` is encoded.

Most programming languages include methods for URL encoding the parameters for you, such as **`urlencode`** (PHP), **`UrlEncode`** (C#), **`URLEncoder.encode`** (Java) and **`encodeURI`** (JavaScript).

5.4 Pagination

By default, most endpoints will return a limited number of records (such as jobs, or test runs) in the response. This can be increased up to a maximum of 1,000.

The default page size (if not specified) is up to 25 records.

The following parameters are used to control the pagination:

- **`paginationPageSize`**: the maximum number of records to return
- **`paginationPage`**: which page to return (indexed from 1)

The following examples show how pagination is used with the `/jobs` endpoint, but the principle is the same for other endpoints:

REQUEST PARAMETERS	BEHAVIOUR
`/jobs`	Get the latest jobs using the default page size
`/jobs?paginationPageSize=500`	Get the latest 500 jobs in the account
`/jobs?paginationPage=3`	Get the latest jobs from the third page using the default page size
`/jobs?paginationPageSize=500&paginationPage=3`	Get the latest set of 500 jobs from the third page of 500
`/jobs?paginationPageSize=5000`	HTTP/400 Bad Request (1,000 is the maximum number of jobs that can be returned with each request)

All requests to the API include a **`meta`** section in the response which contains more information about the data being returned.

One of the parameters returned is **`paginationTotalResults`** which describes how many records match the request.

For example if your response includes the following, then you know there are 7,724 records you can retrieve, up to 1,000 records at a time:

BODY	

```
"meta": {
   "paginationTotalResults": 7724,
   "response_in_sec": 0.13
},
```

6 Extracting Data

There are various endpoints for extracting data out of the API.

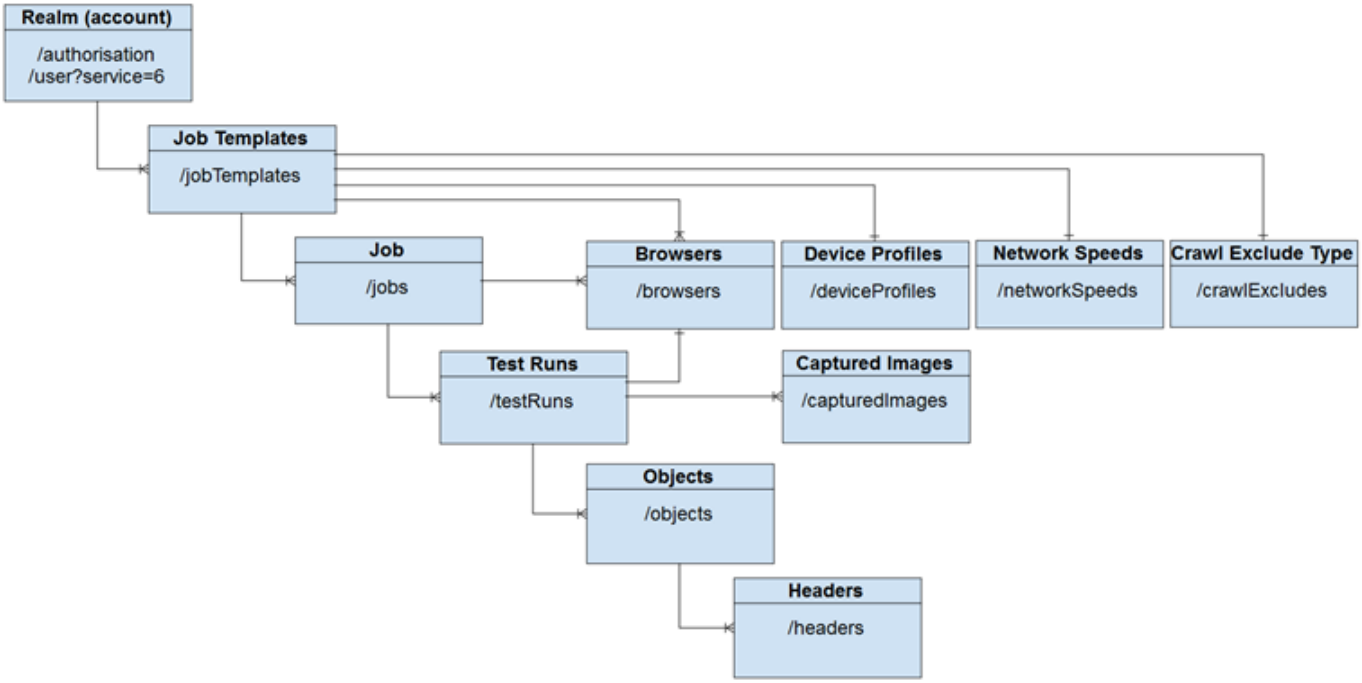
Endpoints typically include Uri paths (e.g. `jobUri`) in the responses that describe how records relate to each other for example, test runs will be related to the job that they were run from.

Similarly, every object you request or search for in the API will include an `sref` which is a self-reference URI for that object.

The `sref` is effectively a combination of the id of the object and the API endpoint for accessing it. In general, you will use the `sref` to query for a specific object.

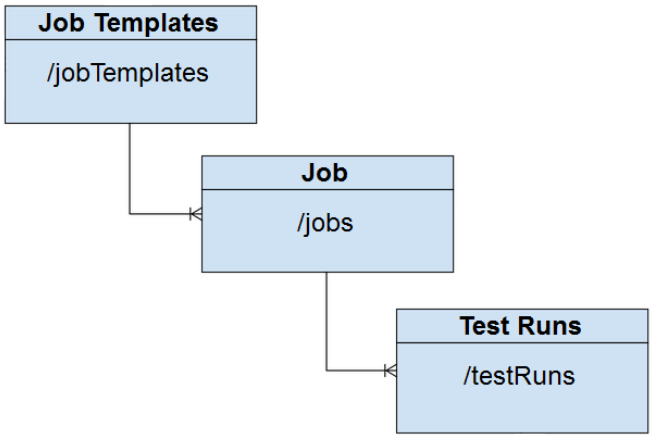
For example (as documented later), the endpoint for requesting data for a job is `/jobs`, so if a job has an id 123, the `sref` for it would be `/jobs/123`.

This shows how the various endpoints and objects relate to each other:



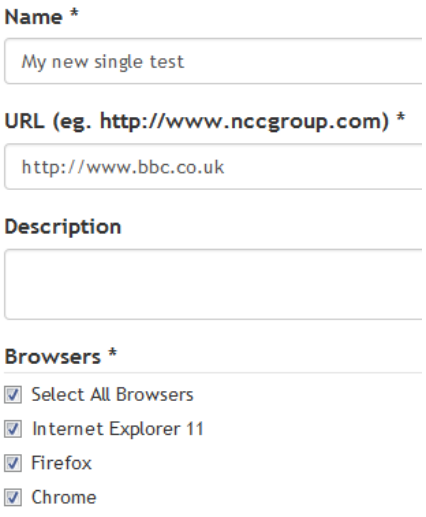
When using Performance Analyser you will be working with **Job Templates**, **Jobs** and **Test Runs**. While you may not be too familiar with these when using the portal, it's important to understand these when using the API.

Referring back to the diagram above and just focusing on these items, we can see how they are related:



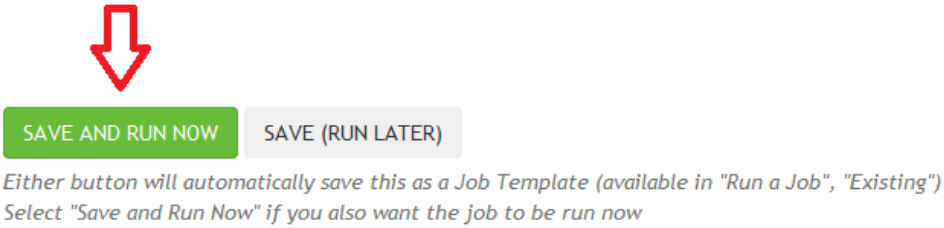
When you create a new **single**, **multi** or **crawl** in Performance Analyser, you are creating a new **Job Template**.

In this example we are creating a new single **Job Template** in Performance Analyser:



Notice that we have selected three browsers for this Job Template

Once we have configured the **Job Template** parameters, we click the **SAVE AND RUN NOW** button:



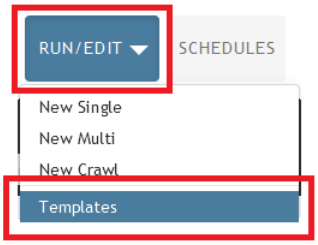
This will create a new **Job Template** and also run a **Job** of that **Job Template**.

The job is visible on the main **Job List** page:

NAME	URL	STARTED	LAST UPDATED	RUN BY	TYPE	PAGES	STATUS	TOOLS
My new single test	http://www.bbc.co.uk/	03 Mar 2017 11:39:08	03 Mar 2017 11:41:43	Paul Bianciardi	Single	3	Completed	

As we configured the **Job Template** to test a URL in three browsers we have ended up creating three **Test Runs**.

You can see the new **Job Template** created by looking on the **Templates** page:

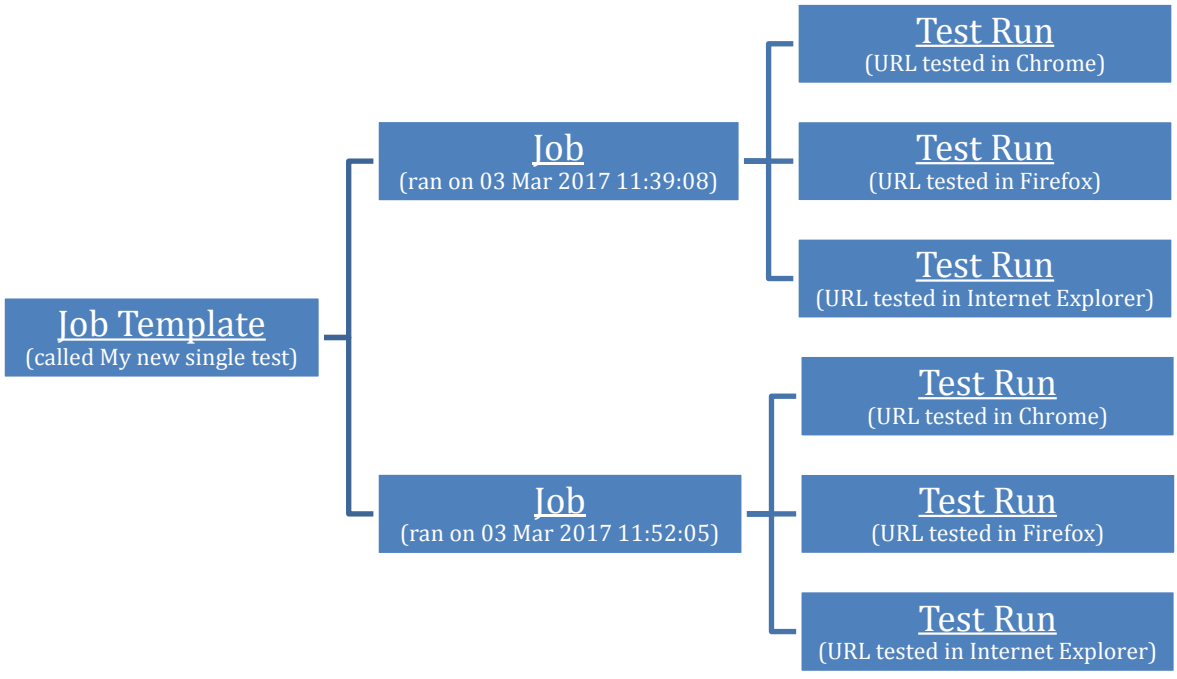


	NAME	URL	CREATED BY	TYPE	TOOLS
	My new single test	http://www.bbc.co.uk/	Paul Bianciardi	Single	

Running another one of these (from the scheduler, or by using the icon) will create another **Job** from the same **Job Template**.

NAME	URL	STARTED	LAST UPDATED	RUN BY	TYPE	PAGES	STATUS	TOOLS
My new single test	http://www.bbc.co.uk/	03 Mar 2017 11:52:05	03 Mar 2017 11:54:42	Paul Bianciardi	Single	3	Completed	
My new single test	http://www.bbc.co.uk/	03 Mar 2017 11:39:08	03 Mar 2017 11:41:43	Paul Bianciardi	Single	3	Completed	

In this scenario we have created 1x **Job Template** and 2x **Jobs**. Each **Job** has created 3x **Test Runs** (so, 6x **Test Runs** in total), which looks like this:



6.1 Jobs

6.1.1 Get All Jobs

This is equivalent to what you get on the default landing page in Performance Analyser, where you see a list of all jobs within your account.

REQUEST

METHOD	`GET`
PATH	`/jobs`
HEADER	`Realm: {realm_of_the_account}` `Authorization: Bearer {access_token}`

RESPONSE

The response for this request will always be an array of job objects:

BODY

```
"results": [
   {
      "name": "Homepage",
      "url": "http://www.mydomain.com",
      "startedAt": "2017-02-27T13:19:11+00:00",
      "updatedAt": "2017-02-27T13:19:53+00:00",
      "startedBy": "Joe Bloggs",
      "type": "Single",
      "pageCount": 1,
      "status": "completed",
      "sref": "jobs/12345",
      "jobTemplateUri": "jobTemplates/6789",
      "runType": "manual",
      "testRunsUri": "testRuns?job=jobs%2F12345"
   },
   {
      "name": "Sales Funnel",
      "url": "https://www.yourdomain.co.uk/",
      "startedAt": "2017-02-27T08:00:04+00:00",
      "updatedAt": "2017-02-27T08:00:57+00:00",
      "startedBy": "Joe Bloggs",
      "type": "Scripted",
      "pageCount": 3,
      "status": "completed",
      "sref": "jobs/23456",
      "jobTemplateUri": "jobTemplates/98765",
      "runType": "scheduled",
      "testRunsUri": "testRuns?job=jobs%2F23456"
   },
   ...
]
```

6.1.2 Get One Job

REQUEST

METHOD	`GET`
PATH	`/jobs/{id}`
HEADER	`Realm: {realm_of_the_account}` `Authorization: Bearer {access_token}`

You request an individual job using the **`sref`** of that job. This is typically found by searching for jobs via other requests (such as all jobs within a certain date range).

As mentioned earlier, the **`sref`** is a combination of the path and id, for example **`/jobs/123`**.

RESPONSE

As you are requesting an individual job, the response for this request will always be a single job object:

BODY

```
"results": {
   "name": "Homepage",
   "url": "http://www.mydomain.com",
   "startedAt": "2017-02-27T13:19:11+00:00",
   "updatedAt": "2017-02-27T13:19:53+00:00",
   "startedBy": "Joe Bloggs",
   "type": "Single",
   "pageCount": 1,
   "status": "completed",
   "sref": "jobs/12345",
   "jobTemplateUri": "jobTemplates/6789",
   "runType": "manual",
   "testRunsUri": "testRuns?job=jobs%2F12345"
}
```

6.1.3 Get Jobs from Related Job Templates

You may want to get all of the jobs that have been run from the same job template (or subsequent edits of that job template). This is useful to see how the performance of the results in that job changes over time.

For example, if you have a job template set up to test your website against your competitors every day, you may want to extract the results from the API to display on a dashboard. This is similar to what the Speed Analysis chart shows you in the portal.

REQUEST

METHOD	`GET`
PATH	`/jobs?jobTemplate={sref_of_job_template}`
HEADER	`Realm: {realm_of_the_account}` `Authorization: Bearer {access_token}`

Note that you are searching for jobs based on the given **`sref`** for the job template. Refer to the section on Get One Job Template for more details on discovering the **`sref`** for job templates.

The **`sref`** for the job template must be URL encoded, so for example if you are searching for jobs that have run from the job template 6789 (which would have an **`sref`** of **`jobTemplates/6789`**), your request would look like this:

PATH	`/jobs?jobTemplate=jobTemplates%2F6789`

RESPONSE

The response for this request will always be an array of job objects:

BODY


```
"results": [
   {
      "name": "Homepage",
      "url": "http://www.mydomain.com",
      "startedAt": "2017-02-27T13:19:11+00:00",
      "updatedAt": "2017-02-27T13:19:53+00:00",
      "startedBy": "Joe Bloggs",
      "type": "Single",
      "pageCount": 1,
      "status": "completed",
      "sref": "jobs/12345",
      "jobTemplateUri": "jobTemplates/6789",
      "runType": "manual",
      "testRunsUri": "testRuns?job=jobs%2F12345"
   },
   {
      "name": "Homepage",
      "url": "http://www.mydomain.com",
      "startedAt": "2017-02-28T13:19:11+00:00",
```



```
        "updatedAt": "2017-02-28T13:19:53+00:00",
        "startedBy": "Joe Bloggs",
        "type": "Single",
        "pageCount": 1,
        "status": "completed",
        "sref": "jobs/12345",
        "jobTemplateUri": "jobTemplates/6789",
        "runType": "manual",
        "testRunsUri": "testRuns?job=jobs%2F12345"
    },
    ...
]
```

6.1.4 Job Parameters

Each job you request from the API will include the following parameters:

PARAM	DETAILS
`name`	The name given to the job
`url`	The main URL for the job
`startedAt`	The date/time the job started
`updatedAt`	The date/time the job last updated (if the job is still running, this will be updated as test run results become available)
`startedBy`	The person who ran the job (note that any jobs run from the API will show as having been run by the `Performance Analyser API` user)
`type`	The type of job; this will be one of: • `Single` • `Multi` • `Crawl` • `Scripted`
`pageCount`	The number of test run (page) results that are available for the job (if the job is still running, this will increase as results become available)
`status`	The status of the job; this will be one of: • `pending` – the job has been submitted but hasn't yet started to run • `running` – the job is in progress and results will start to become available • `failed` – the job has failed to run; try running it again or contact Support • `completed` – the job has finished, and all results are available for it • `cancelled` – a user stopped the job before it had finished • `expired` – the crawl has taken longer than allowed so has been stopped
`sref`	This is a self-reference path that describes how you can request the individual job
`jobTemplateUri`	The URI of the job template used to run this job, i.e. the job template's **`sref`** (the job template contains the parameters and behaviour of how the job should run)
`runType`	How the job ran, which will be one of: • `manual` – the job was run manually from the portal, or via the API • `scheduled` – the job was run from the scheduler
`testRunsUri`	The URI of how to discover the test runs (results) for this job

6.2 Test Runs

Test runs are the individual results (pages) you get when running jobs in Performance Analyser. For example, if you run a 1,000 page crawl, you will get 1,000 test run results.

Similarly, if you test a single URL in Chrome, Firefox and Internet Explorer, you will get three test run results.

6.2.1 Get One Test Run

REQUEST

You request an individual test run using the **`sref`** of that test run (which is a combination of the endpoint and id for the test run). This is typically found by searching for test runs via other requests, which include a **`testRunsUri`** for that test run.

METHOD	GET
PATH	/testRuns/***{id}***
HEADER	Realm: ***{realm_of_the_account}*** Authorization: Bearer ***{access_token}***

RESPONSE

As you are requesting an individual test run, the response for this request will always be a single test run object:

BODY


```
"results": {
   "sref": "testRuns/1001",
   "pageTitle": "My Homepage",
   "url": "http://my.url.com",
   "browser": "browsers/ff",
   "dataStartDuration": 1.234,
   "renderStart": 2.345,
   "domLoad": 3.456,
   "onLoad": 4.567,
   "visuallyComplete": 5.678,
   "downloadDuration": 6.789,
   "pageSize": 42345,
   "speedIndex": 3678,
   "resultCode": 1,
   "objectCount": 26,
   "severity": "ok",
   "ranAt": "2016-04-07T08:06:47+00:00",
   "objectsUri": "objects?testRun=testRuns%2F1001",
   "jobUri": "jobs/123"
}
```


6.2.2 Get All Test Runs for a Job

Use this to get all of the test run results for a specific job.

REQUEST

METHOD	`GET`
PATH	`/testRuns?job={sref_of_job}`
HEADER	`Realm: {realm_of_the_account}` `Authorization: Bearer {access_token}`

Note that you are searching for test runs, based on the given **`sref`** for the job. As shown above, job objects will include the **`sref`** parameter which is how you search for all test runs for that job.

The **`sref`** for the job must be URL encoded, so for example your request may look something like this:

PATH	`/testRuns?job=jobs%2F123`

RESPONSE

The response for this request will always be an array of test run objects:

BODY


```
"results": [
   {
      "sref": "testRuns/1001",
      "pageTitle": "My Homepage",
      "url": "http://my.url.com",
      "browser": "browsers/ff",
      "dataStartDuration": 1.234,
      "renderStart": 2.345,
      "domLoad": 3.456,
      "onLoad": 4.567,
      "visuallyComplete": null,
      "downloadDuration": 6.789,
      "pageSize": 42345,
      "speedIndex": null,
      "resultCode": 1,
      "objectCount": 26,
      "severity": "ok",
      "ranAt": "2016-04-07T08:06:47+00:00",
      "objectsUri": "objects?testRun=testRuns%2F1001",
      "jobUri": "jobs/123"
   },
      "sref": "testRuns/1002",
      "pageTitle": "Your Homepage",
      "url": "http://your.url.com",
      "browser": "browsers/ff",
      "dataStartDuration": 3.254,
      "renderStart": 1.463,
      "domLoad": 2.693,
      "onLoad": 3.683,
      "visuallyComplete": 4.293,
```



```
        "downloadDuration": 6.046,
        "pageSize": 83968,
        "speedIndex": 3678,
        "resultCode": 1,
        "objectCount": 32,
        "severity": "ok",
        "ranAt": "2016-04-07T08:08:12+00:00",
        "objectsUri": "objects?testRun=testRuns%2F1002",
        "jobUri": "jobs/123"
    },
    ...
]
```


6.2.3 Get Test Runs from Related Job Templates

You may want to get all of the test runs that have been run from jobs from the same job template (or subsequent edits of that job template). This is useful to see how the performance of the results in that job changes over time.

For example, if you have a job template set up to test your website against your competitors every day, you may want to extract the results from the API to display on a dashboard. This is similar to what the Speed Analysis chart shows you in the portal.

REQUEST

METHOD	`GET`
PATH	`/testRuns?jobTemplate={sref_of_job_template}`
HEADER	`Realm: {realm_of_the_account}` `Authorization: Bearer {access_token}`

Note that you are searching for test runs based on the given **`sref`** for the job template. Refer to the section on Get One Job Template for more details on discovering the **`sref`** for job templates.

The **`sref`** for the job template must be URL encoded, so for example your request may look something like this:

PATH	`/testRuns?jobTemplate=jobTemplates%2F6789`

RESPONSE

The response for this request will always be an array of test run objects:

BODY

```
"results": [
   {
      "sref": "testRuns/1001",
      "pageTitle": "My Homepage",
      "url": "http://my.url.com",
      "browser": "browsers/ff",
      "dataStartDuration": 1.234,
      "renderStart": 2.345,
      "domLoad": 3.456,
      "onLoad": 4.567,
      "visuallyComplete": null,
      "downloadDuration": 6.789,
      "pageSize": 42345,
      "speedIndex": null,
      "resultCode": 1,
      "objectCount": 26,
      "severity": "ok",
      "ranAt": "2016-04-07T08:06:47+00:00",
      "objectsUri": "objects?testRun=testRuns%2F1001",
      "jobUri": "jobs/123"
   },
      "sref": "testRuns/1002",
      "pageTitle": "Your Homepage",
```

```
        "url": "http://your.url.com",
        "browser": "browsers/ff",
        "dataStartDuration": 3.254,
        "renderStart": 1.463,
        "domLoad": 2.693,
        "onLoad": 3.683,
        "visuallyComplete": 4.293,
        "downloadDuration": 6.046,
        "pageSize": 83968,
        "speedIndex": 3678,
        "resultCode": 1,
        "objectCount": 32,
        "severity": "ok",
        "ranAt": "2016-04-07T08:08:12+00:00",
        "objectsUri": "objects?testRun=testRuns%2F1002",
        "jobUri": "jobs/123"
    },
    ...
]
```


6.2.4 Test Run Parameters

Each test run you request from the API will include the following parameters:

PARAM	DETAILS
`sref`	This is a self-reference path that describes how you can request the test run object
`pageTitle`	The title of the page tested (or the URL if no page title was set)
`url`	The URL tested
`browser`	The browser used for the test run
`dataStartDuration`	How long (in seconds) until the very first request started return data
`renderStart`	How long (in seconds) until the browser starts rendering the page
`domLoad`	How long (in seconds) until the DOM load event has fired in the browser
`onLoad`	How long (in seconds) until the page load event fires in the browser
`visuallyComplete`	How long (in seconds) until the visible page has finished rendering
`downloadDuration`	How long (in seconds) for all network activities take to load the page
`pageSize`	The size of the page in bytes (using the compressed size of objects if compressed)
`speedIndex`	The Speed Index for the test run (refer to this documentation for more details)
`resultCode`	The result code of the test run (refer to the portal documentation for more details)
`objectCount`	How many objects (network requests) were made for this page
`severity`	A categorisation of how the page ran (for example, `ok`, `warning`)
`ranAt`	The date/time the page was tested
`objectsUri`	The URI of how to query details about all objects for this test run
`jobUri`	The URI of the job that this test run is a member of

6.3 Captured Images (Screen Shots)

You can use the API to get the screen shots captured during the tests. First discover what screen shots are available for a test run (URIs for thumbnail and full size images) and then request the actual images.

6.3.1 Discovery

Use the following to discover what screen shots are available for a given test run.

REQUEST

METHOD	GET
PATH	/capturedImages?testRun=***{sref_of_test_run}***
HEADER	Realm: ***{realm_of_the_account}*** Authorization: Bearer ***{access_token}***

Note that you are searching for captured images, based on the given **sref** for the test run.

The **sref** for the test run must be URL encoded, so for example your request may look something like this:

PATH	/capturedImages?testRun=testRuns%2F123456789

RESPONSE

The response will always be an array of object detailing the images available (with no pagination):

BODY	"results": [
	{
	"thumbnailImageUri": "images/9cbb0fbc5a1d5c78b9f54fc17983661813",
	"interval": 0,
	"hasChanged": true,
	"fullSizeImageUri": "images/a535edf350e225534efbe87ef147496146"
	},
	{
	"thumbnailImageUri": "images/9cfbb0fb5a1d367c789f54fc1798366113",
	"interval": 0.1,
	"hasChanged": false,
	"fullSizeImageUri": "images/a5354df350e22534e6fbe8ef1f43796146"
	}
	...
	]

CAPTURED IMAGE PARAMETERS

PARAM	DETAILS
thumbnailImageUri	The URI of the thumbnail image for screen shot at that time interval
interval	The time interval (in seconds) that the image refers to
hasChanged	Whether or not the image at this time interval has changed from the prior interval
fullSizeImageUri	The URI of the full size image for screen shot at that time interval

6.3.2 Get the Image

To extract the actual PNG image of the thumbnail or full size image from the API, make a request to the `thumbnailImageUri` or `fullSizeImageUri` URIs you discovered as shown above.

The URI is effectively a combination of the API endpoint path and the **`{hash}`** of the image being requested:

METHOD	`GET`
PATH	`/images/`***`{hash}`***
HEADER	`Realm:` ***`{realm_of_the_account}`*** `Authorization: Bearer` ***`{access_token}`***

Be aware that the first captured image of most test runs (scripted jobs can be different) will be a blank white image (i.e. the initial blank page you see when the browser launches).

If do you request the first image, expect it to be a blank white PNG (this is not an error).

6.4 Objects

Objects refer to each of the network requests that are made when testing a URL in Performance Analyser. For example, each row you see on a waterfall chart corresponds to an object (such as a CSS file, PNG image etc.).

6.4.1 Get All Objects for a Test Run

You will typically get all of the objects for a given test run, as shown here. If you only want one object, use the **`sref`** to request it individually, as shown in the next section.

REQUEST

METHOD	`GET`
PATH	`/objects?testRun=`***`{sref_of_test_run}`***
HEADER	`Realm:` ***`{realm_of_the_account}`*** `Authorization: Bearer` ***`{access_token}`***

Note that you are searching for objects, based on the given **`sref`** for the test run.

The **`sref`** for the test run must be URL encoded, so for example your request may look something like this:

PATH	`/objects?testRun=testRuns%2F123456789`

RESPONSE

The response for this request will always be an array of objects:

BODY


```
"results": [
   {
      "url": "http://my.url.com/home.html",
      "mimeType": "text/html",
      "result": 301,
      "requestHeaderSize": 663,
      "responseHeaderSize": 659,
      "transmittedSize": 20,
      "uncompressedSize": 20,
      "offsetDuration": 0,
      "blockDuration": null,
      "dnsDuration": 0,
      "connectDuration": 0,
      "sslDuration": 0,
      "sendDuration": null,
      "dataStartDuration": 0.052,
      "receiveDuration": 0,
      "cacheDuration": null,
      "networkDuration": 0.052,
      "ttfbDuration": 0.052,
      "downloadDuration": 0.052,
      "responseBodyChecksum": "d91492d55d6e977a0bfe790238b24a76677b85",
      "sref": "objects/345678",
      "testRunUri": "testRuns/123456789",
      "headersUri": "headers?object=objects%2F345678"
   },
   {
      "url": " http://my.url.com/styles/main.css",
      "mimeType": "text/css",
      "result": 200,
      "requestHeaderSize": 358,
      "responseHeaderSize": 345,
      "transmittedSize": 519,
      "uncompressedSize": 1066,
      "offsetDuration": 0.055,
      "blockDuration": null,
      "dnsDuration": 0.005,
      "connectDuration": 0.004,
      "sslDuration": 0.048,
      "sendDuration": null,
      "dataStartDuration": 0.031,
      "receiveDuration": 0,
      "cacheDuration": null,
      "networkDuration": 0.088,
      "ttfbDuration": 0.088,
      "downloadDuration": 0.088,
      "responseBodyChecksum": "d91492d55d6e977a0bfe79022b384a76677b85",
      "sref": "objects/456789",
      "testRunUri": "testRuns/123456789",
      "headersUri": "headers?object=objects%2F456789"
   }
   ...
]
```

6.4.2 Get One Object

The **sref** of the object contains the full path of how to request each object individually. The **sref** is effectively the API endpoint path and the **id** of the object being requested.

REQUEST

METHOD	GET
PATH	/objects/***{id}***
HEADER	Realm: ***{realm_of_the_account}*** Authorization: Bearer ***{access_token}***

RESPONSE

As you are requesting an individual object, the response for this request will always be a single object:

BODY


```
"results": {
   "url": "http://my.url.com/home.html",
   "mimeType": "text/html",
   "result": 301,
   "requestHeaderSize": 663,
   "responseHeaderSize": 659,
   "transmittedSize": 20,
   "uncompressedSize": 20,
   "offsetDuration": 0,
   "blockDuration": null,
   "dnsDuration": 0,
   "connectDuration": 0,
   "sslDuration": 0,
   "sendDuration": null,
   "dataStartDuration": 0.052,
   "receiveDuration": 0,
   "cacheDuration": null,
   "networkDuration": 0.052,
   "ttfbDuration": 0.052,
   "downloadDuration": 0.052,
   "responseBodyChecksum": "d91492d55d6e977a0bfe790238b24a76677b85",
   "sref": "objects/345678",
   "testRunUri": "testRuns/123456789",
   "headersUri": "headers?object=objects%2F345678"
}
```


6.4.3 Object Parameters

PARAM	DETAILS
`url`	The URL of the object being requested
`mimeType`	The mime type (content type) of this object
`result`	The status code for the object (refer to the portal documentation for more details)
`requestHeaderSize`	The size of the request header in bytes
`responseHeaderSize`	The size of the response header in bytes
`transmittedSize`	The size in bytes of the response body (on the wire), showing the compressed size if relevant
`uncompressedSize`	The size in bytes of the response body, in its uncompressed state
`offsetDuration`	The number of seconds from when the test started to when the browser starts the request for the object (which may be network activity such as DNS)
`blockDuration`	This has been deprecated in Performance Analyser so will generally be `null`
`dnsDuration`	The number of seconds spent resolving a host name into an IP address
`connectDuration`	The number of seconds required to create a TCP connection to the web server (this does not includes SSL/TLS handshake time for HTTPS requests)
`sslDuration`	The number of seconds spent performing SSL/TLS handshake for HTTPS requests
`sendDuration`	This has been deprecated in Performance Analyser so will generally be `null`
`dataStartDuration`	The number of seconds spent waiting for a response from the web server
`receiveDuration`	The number of seconds taken to read the response from the web server
`cacheDuration`	This has been deprecated in Performance Analyser so will generally be `null`
`networkDuration`	The number of seconds spent on all network related operations for the request (this will typically be the same as `downloadDuration`)
`ttfbDuration`	Time To First Byte is the number of seconds from the initial network request to the first byte of data being received
`downloadDuration`	The total amount of time in seconds (including all network activity) that the request took to complete (typically be the same as `networkDuration`)
`responseBodyChecksum`	A hash of the body content of the object (this is useful to compare against other objects, for example to see if the content changes over time)
`sref`	This is a self-reference path that describes how you can request the object
`testRunUri`	The URI of the test run that this test object is from
`headersUri`	The URI for querying the HTTP request and response headers for this object

6.5 HTTP Headers

Request and response headers can obtained for each object requested.

As shown above, when requesting details for an object, the API includes the discovery attribute **`headersUri`** for where to get the header from.

REQUEST

METHOD	`GET`
PATH	`/headers?object=`***`{sref_of_the_object}`***
HEADER	`Realm:` ***`{realm_of_the_account}`*** `Authorization: Bearer` ***`{access_token}`***

The **`sref`** for the object must be URL encoded, so for example your request may look something like this:

PATH	`/headers?object=objects%2F456789`

RESPONSE

The response will contain a `requestHeader` and `responseHeader`, each of which contains an array of header items (split into `name` and `value` pairs):

BODY

```
"results": {
   "requestHeader": [
      {
         "name": "Request Line",
         "value": "GET / HTTP/1.1"
      },
      {
         "name" : "Accept",
         "value" : "text/html,application/xhtml+xml,application/xml"
      },
      ...
   ],
   "responseHeader": [
      {
         "name": "Status Line",
         "value": "HTTP/1.1 200 OK"
      },
      {
         "name" : "Accept-Ranges",
         "value" : "bytes"
      },
      ...
   ]
}
```

6.6 Job Templates

Job templates contain the configuration of how a job should run, for example which URLs to test, in which browser, and in the case of a crawl, how long it should run for, etc.

When querying for job templates, parameters that are common to all job types will be returned within the main object. Parameters that are specific to certain types of jobs will be included in the **meta** section.

For consistency across job types, browsers and URLs will always be in the common section and be returned as arrays (even through some job types only permit one of these).

6.6.1 Get All Job Templates

REQUEST

METHOD	GET
PATH	/jobTemplates
HEADER	Realm: ***{realm_of_the_account}*** Authorization: Bearer ***{access_token}***

RESPONSE

The response for this request will always be an array of job template objects:

BODY

```
"results": [
   {
      "sref": "jobTemplates/123456",
      "type": "Single",
      "name": "My single job template",
      "description": "",
      "urls": [
         "www.mydomain.com"
      ],
      "browsers": [
         "browsers/ff",
         "browsers/ch",
         "browsers/ie11",
      ]
      "createdBy": "Joe Bloggs",
      "capture": true,
      "advanced": {
         "inactivityTimeout": 0
      },
      "meta": {},
      "deviceProfile": "deviceProfiles/1",
      "networkSpeed": {
         "networkSpeedUri": "networkSpeeds/1",
         "meta": {
            "downloadKbps": 2010,
            "uploadKbps": 190,
            "latency": 0,
```



```
            "packetLossRate": 0
         }
      }
   },
   {
      "sref": "jobTemplates/456789",
      "type": "Multi",
      "name": "My multi job template",
      "description": "Comparison between us and main competitor",
      "urls": [
         "www.mydomain.com",
         "www.yourdomain.com",
      ],
      "browsers": [
         "browsers/ff"
      ],
      "createdBy": "Joe Bloggs",
      "capture": true,
      "advanced": {
         "inactivityTimeout": 0
      },
      "meta": {},
      "deviceProfile": null,
      "networkSpeed": {
         "networkSpeedUri": "networkSpeeds/1",
         "meta": {
            "downloadKbps": 2010,
            "uploadKbps": 190,
            "latency": 0,
            "packetLossRate": 0
         }
      }
   },
   {
      "sref": "jobTemplates/654321",
      "type": "Crawl",
      "name": "My crawl job template",
      "description": "",
      "urls": [
         "www.mydomain.com"
      ],
      "browsers": [
         "browsers/ch"
      ],
      "createdBy": "John Doe",
      "capture": false,
      "advanced": {
         "inactivityTimeout": 1
      },
      "meta": {
         "crawlExcludes": {
            "type": "regex",
            "data": [
               "%.*\.jpe?g%i",
               "%.*\.pdf%i"
            ]
         },
         "crawlTimeout": 10800
```

```
        },
        "deviceProfile": null,
        "networkSpeed": {
            "networkSpeedUri": "networkSpeeds/1",
            "meta": {
                "downloadKbps": 2010,
                "uploadKbps": 190,
                "latency": 0,
                "packetLossRate": 0
            }
        }
    }
]
```


6.6.2 Get One Job Template

REQUEST

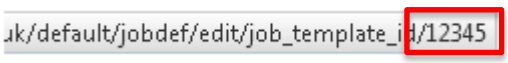
METHOD	GET
PATH	/jobTemplates/***{id}***
HEADER	Realm: ***{realm_of_the_account}*** Authorization: Bearer ***{access_token}***

The **sref** is typically found by searching for job templates via other requests (e.g. requesting all job templates as shown above). This is the recommended approach to ensure you are referencing the correct job template.

You can however also use the Performance Analyser Portal as a helper to find the **id** which you can then use to construct the **sref**.

Mouse over the edit template icon from the job templates page:

The address for the icon shown in the bottom left of your browser will include the **id** of the job template:



You can combine the **id** of the job template with the path of the API endpoint to construct the **sref** for the individual job template, for example **/jobTemplates/12345**.

RESPONSE

As you are requesting an individual job template, the response for this request will always be a single job template object:

BODY

```
"results": {
   "sref": "jobTemplates/123456",
   "type": "Single",
   "name": "My single job template",
   "description": "",
   "urls": [
      "www.mydomain.com"
   ],
   "browsers": [
      "browsers/ff",
      "browsers/ch",
      "browsers/ie11",
   ]
   "createdBy": "Joe Bloggs",
   "capture": true,
   "advanced": {
      "inactivityTimeout": 0
   },
   "meta": {},
   "deviceProfile": null,
   "networkSpeed": {
      "networkSpeedUri": "networkSpeeds/1",
      "meta": {
         "downloadKbps": 2010,
         "uploadKbps": 190,
         "latency": 0,
         "packetLossRate": 0
      }
   }
}
```

6.6.3 Job Template Parameters

PARAM	DETAILS
`sref`	This is a self-reference path that describes how you can request the individual job template
`type`	The type of job; this will be one of: • `Single` • `Multi` • `Crawl` • `Scripted`
`name`	The name given to the job template
`description`	The description given to the job template
`urls`	An array of URLs for the job template (this will always be an array for all job types)
`browsers`	An array of browsers that jobs of this job template will run in (as can be discovered using the `/browsers` request, see below)
`createdBy`	The name of the person who created (or last edited) the job template
`capture`	Whether or not screen capture has been enabled for the job template (Boolean)
`advanced`	Contains any advanced parameters for the job template, such as the inactivity timeout
`meta`	Contains anything configured for the job template that is not standard for all types of job template (typically, this will contain additional details about a crawl)
`deviceProfile`	Defines the device profile that jobs of this job template will run in, or `null` if no device profile has been set (these can be discovered using the `/deviceProfiles` request, see below)
`networkSpeed`	Defines the network speed that jobs of this job template will run in (as can be discovered using the `/networkSpeeds` request, see below), where `meta` includes individual elements of the network speed

6.7 Limit Searches by Date/Time

It is possible to restrict the search for test runs or jobs by the date/time they happened. You can specify **either** the **`fromDate`** or the **`toDate`** or **both** together.

All date/times are inclusive and are specified using the standard date/time format we return data in, but must be URL encoded and sent as a query string.

The returned pagination will honour the date/time range you have specified (meaning you can retrieve all of the data from within that date/time range, even if it spans multiple paginated pages). The response will include the date ranges specified as well as the total number of results in the **`meta`** section:

BODY	<pre>"meta": {
	  "fromDate": "2017-03-01T00:00:00+00:00",
	  "toDate": "2017-03-01T23:59:59+00:00",
	  "paginationTotalResults": 66,
	  "response_in_sec": 0.28
	},</pre>

6.7.1 Limit Test Runs by Date/Time

REQUEST

METHOD	GET
PATH	/testRuns?fromDate=***{from_date}***&toDate=***{to_date}***
HEADER	Realm: ***{realm_of_the_account}*** Authorization: Bearer ***{access_token}***

All date/times must be URL encoded, so for example if you wanted to search for all test runs that have run since the 1st January 2017 (which is represented as **`2017-01-01T00:00:00+00:00`** in the ISO date format), you would make the following request:

PATH	/testRuns?fromDate=2017-01-01T00%3A00%3A00%2B00%3A00

6.7.2 Limit Jobs by Date/Time

REQUEST

METHOD	GET
PATH	/jobs?fromDate=***{from_date}***&toDate=***{to_date}***
HEADER	Realm: ***{realm_of_the_account}*** Authorization: Bearer ***{access_token}***

All date/times must be URL encoded, so for example if you wanted to search for all jobs that ran before 1 January 2017 (which is represented as **`2017-01-01T00:00:00+00:00`** in the ISO date format), you would make the following request:

PATH	/jobs?toDate=2017-01-01T00%3A00%3A00%2B00%3A00

6.8 Discovery

Some of the parameters and values you will use within the API can be discovered by querying the API.

For example, for details about the browsers within Performance Analyser (for tests that have already run, or for running a new test), you can query the API to find out more details about those browsers.

6.8.1 Get Browser List

REQUEST

METHOD	`GET`
PATH	`/browsers`
HEADER	`Realm: {realm_of_the_account}` `Authorization: Bearer {access_token}`

RESPONSE

BODY

```
"results": [
   {
      "sref": "browsers/ch",
      "shortName": "CH",
      "longName": "Chrome",
      "deprecated": false
   },
   {
      "sref": "browsers/ie6",
      "shortName": "IE 6",
      "longName": "Internet Explorer 6",
      "deprecated": true
   },
   ...
]
```

BROWSER PARAMETERS

PARAM	DETAILS
`sref`	This is a self-reference path that describes how you can request the specific browser
`shortName`	The short name label for the browser
`longName`	The long (full) name label for the browser
`deprecated`	Whether or not the browser is deprecated (Boolean) If a browser is no longer available (`"deprecated": true`), it cannot be used for creating new job templates in Performance Analyser, but is included to describe the browser that may have been used for an older test, when the browser was available.

6.8.2 Get Network Speeds

REQUEST

METHOD	GET
PATH	/networkSpeeds
HEADER	Realm: ***{realm_of_the_account}*** Authorization: Bearer ***{access_token}***

RESPONSE

BODY

```
"meta": {
   "default": "networkSpeeds/1",
},
"results": [
   {
      "sref": "networkSpeeds/1",
      "name": "Original 2Mbps",
      "downloadKbps": 2010,
      "uploadKbps": 190,
      "latency": 0,
      "packetLossRate": 0,
      "default": true
   },
   {
      "sref": "networkSpeeds/2",
      "name": "3G",
      "downloadKbps": 19200,
      "uploadKbps": 128,
      "latency": 0.2,
      "packetLossRate": 0,
      "default": false
   }
]
```

As well as the main `results`, the response will include a `meta` section that describes which of the network speeds is the default for the account.

This is for convenience so you do not need to iterate through the `results` to find the default.

NETWORK SPEED PARAMETERS

PARAM	DETAILS
meta	The top level `meta` JSON object describes the URI of the default network speed
sref	This is a self-reference path that describes how you can request the specific network speed
name	The descriptive name of the network speed (as seen in the portal)
downloadKbps	The download throttle speed, in kbps
uploadKbps	The upload throttle speed, in kbps

`latency`	The additional network latency (round trip time) added to network requests, in seconds (note the portal is in ms)
`packetLossRate`	The packet loss rate to apply to the network, as a percentage (i.e. percentage packets to be dropped)
`default`	Whether or not the device profile is the default for the account (only one network speed will be `default: true` which will match with what is shown in the `meta` section

6.8.3 Get Device Profiles

REQUEST

METHOD	`GET`
PATH	`/deviceProfiles`
HEADER	`Realm: {realm_of_the_account}` `Authorization: Bearer {access_token}`

RESPONSE

BODY

```
"results": [
   {
      "sref": "deviceprofiles/1",
      "name": "Phone - Apple - iPhone 4s - Portrait",
      "characteristics": [
         "- UA string for iPhone 4s",
         "- viewport 320x480"
      ],
      "deprecated": false
   },
   {
      "sref": "deviceprofiles/2",
      "name": "Tablet - Apple - iPad - Landscape",
      "characteristics": [
         "- UA string for iPad",
         "- viewport 1024x768"
      ],
      "deprecated": false
   },
   {
      "sref": "deviceprofiles/3",
      "name": "Viewport size 1280x1024",
      "characteristics": [
         "- viewport 1280x1024"
      ],
      "deprecated": false
   }
]
```

DEVICE PROFILE PARAMETERS

PARAM	DETAILS
`sref`	This is a self-reference path that describes how you can request the specific device profile
`name`	The descriptive name of the device profile (as seen in the portal)
`characteristics`	An array of JSON strings that describe the behaviour of the device profile (e.g. the user agent string and/or viewport size)
`deprecated`	Whether or not the device profile is deprecated (Boolean) If a profile is no longer available (`"deprecated": true`), it cannot be used for creating new job templates, but is included to describe the profile that may have been used for an older test

6.8.4 Get Crawl Exclude Types

Use this to discover the types of excludes that can be used within crawl job types.

REQUEST

METHOD	`GET`
PATH	`/crawlExcludes`
HEADER	`Realm: {realm_of_the_account}` `Authorization: Bearer {access_token}`

RESPONSE

BODY	`"results": [`
	`{`
	`"type": "regex"`
	`},`
	`{`
	`"type": "robot"`
	`}`
	`]`

CRAWL EXCLUDE TYPE PARAMETERS

PARAM	DETAILS
`type`	A description of the type of crawl exclude

7 Run New Tests via the API

7.1 Run a New Job

When you run a new job in Performance Analyser, you do so by providing the job template that describes the details of the job you want to run. This is the same as when you use the portal.

As jobs are created from job templates, you do this with a `POST` to the jobs endpoint. This creates a distinction between a `GET` and a `POST` to `/jobs`, but this is necessary because jobs cannot be created on their own – they have to be made from a job template.

If you want to run a test for something completely new that doesn't have a job template, first create the template (as documented below) and then create a job using the new job template.

REQUEST

METHOD	`POST`
PATH	`/jobs`
HEADER	`Realm: {realm_of_the_account}` `Authorization: Bearer {access_token}`

BODY	`{`
	`  "jobTemplateUri": "{sref_of_the_job_templates}"`
	`}`

An example body payload would be: `"jobTemplateUri": "jobTemplates/1234"`

RESPONSE

The response will include the URI of the job that has been created.

BODY	`"results": {`
	`  "jobUri": "jobs/98765"`
	`}`

You can use the **`jobUri`** to find out the status of the job (e.g. when it has finished), as documented above.

7.2 Create a New Job Template

You can create new job templates and specify the main parameters needed to create it.

Crawl parameters cannot currently be configured. New crawls will set the values for max pages, etc. with the defaults that you see in the portal when creating a crawl.

For consistency across job types, browsers and URLs will always be specified as array elements (even through some job types only permit one of these).

You can create job templates with a `type` of `Single`, `Multi` or `Crawl` (you cannot create scripted jobs).

JOB TEMPLATE PARAMETERS

Here are the body parameters you can set when creating new job templates (with examples shown in the next section).

PARAM	Mandatory	DETAILS
`type`	Mandatory	The type of job template to create, this can be set to `Single`, `Multi` or `Crawl`
`name`	Mandatory	The name of the job template
`urls`	Mandatory	An array of URLs to be defined for the job template As in the portal, only one URL can be provided for `Single` or `Crawl` types, and multiple URLs can be given for a `Multi` type
`browsers`	Mandatory	An array of browser URIs the job template should run tests in (as can be discovered using the `/browsers` request, as shown above) As in the portal, only one browser can be provided for `Multi` or `Crawl` types, and multiple browsers can be given for a `Single` type
`description`	Optional	A description to assign to the job template
`deviceProfile`	Optional	The URI of a device profile to be given to this job template, (as can be discovered using the `/deviceProfiles` request, as shown above)
`networkSpeed`	Optional	This is used to set the network speed for the job template (as with the portal you can either use a predefined speed, or a custom one) The `networkSpeed` either contains the `networkSpeedUri` for a predefined speed (which can be discovered using the `/networkSpeeds` request, as shown above), or a `meta` object which defines each of the network speed options as follows: • `downloadKbps` the download throttle speed, in kbps • `uploadKbps` the upload throttle speed, in kbps • `latency` the latency in seconds (note the portal is in ms) • `packetLossRate` the percentage packet loss rate to apply If `meta` is being used to specify a custom speed, then either omit `networkSpeedUri` or set it to `null`

REQUEST

Example showing how to create a new single job in multiple browsers:

METHOD	POST
PATH	/jobTemplates
HEADER	Realm: ***{realm_of_the_account}*** Authorization: Bearer ***{access_token}***

BODY	<pre>{
	  "type": "Single",
	  "name": "My job template",
	  "urls": [
	  "http://my.url.com"
	  ],
	  "browsers": [
	  "browsers/ch",
	  "browsers/ff",
	  "browsers/ie11"
	  ],
	  "description": "Homepage tested in all browsers"
	}</pre>

Example showing how to create a new multi job in Chrome:

METHOD	POST
PATH	/jobTemplates
HEADER	Realm: ***{realm_of_the_account}*** Authorization: Bearer ***{access_token}***

BODY	<pre>{
	  "type": "Multi",
	  "name": "My job template",
	  "urls": [
	  "http://my.url.com",
	  "http://your.url.com"
	  ],
	  "browsers": [
	  "browsers/ch"
	  ]
	}</pre>

Example showing how to create a new crawl job in Chrome:

METHOD	POST
PATH	/jobTemplates
HEADER	Realm: ***{realm_of_the_account}*** Authorization: Bearer ***{access_token}***

BODY	<pre>{
	  "type": "Crawl",
	  "name": "My crawl template",
	  "urls": [</pre>

	<pre>"http://my.url.com"
	],
	"browsers": [
	"browsers/ch"
	]
	}</pre>

Example showing how to create a new crawl job in Firefox with a particular device profile:

METHOD	POST
PATH	/jobTemplates
HEADER	Realm: ***{realm_of_the_account}*** Authorization: Bearer ***{access_token}***

BODY	<pre>{
	"type": "Crawl",
	"name": "My crawl template",
	"urls": [
	"http://my.url.com"
	],
	"browsers": [
	"browsers/ff"
	],
	"deviceProfile": "deviceProfiles/1"
	}</pre>

Example showing how to create a new single job in Chrome with a particular device profile and predefined network speed:

METHOD	POST
PATH	/jobTemplates
HEADER	Realm: ***{realm_of_the_account}*** Authorization: Bearer ***{access_token}***

BODY	<pre>{
	"type": "Single",
	"name": "My single template",
	"urls": [
	"http://my.url.com"
	],
	"browsers": [
	"browsers/ch"
	],
	"deviceProfile": "deviceProfiles/1",
	"description": "Use predefined network speed and device profile",
	"networkSpeed": {
	"networkSpeedUri": "networkSpeeds/1"
	}
	}</pre>

Example showing how to create a new single job in Chrome with a particular device profile and custom network speed:

METHOD	`POST`
PATH	`/jobTemplates`
HEADER	`Realm: {realm_of_the_account}` `Authorization: Bearer {access_token}`

BODY	`{`
	`  "type": "Single",`
	`  "name": "My single template",`
	`  "urls": [`
	`  "http://my.url.com"`
	`  ],`
	`  "browsers": [`
	`  "browsers/ch"`
	`  ],`
	`  "deviceProfile": "deviceProfiles/1",`
	`  "description": "Set a custom network speed and a device profile",`
	`  "networkSpeed": {`
	`  "networkSpeedUri": null,`
	`  "meta": {`
	`  "downloadKbps": 1500`
	`  "uploadKbps": 250,`
	`  "latency": 0.05,`
	`  "packetLossRate": 10`
	`  }`
	`  }`
	`}`

Note we have set `"networkSpeedUri": null` in this example as we're defining a custom network speed. This is optional, but must either be `null` or omitted completely if `meta` is being used to define the custom network speed.

RESPONSE

The response will include the self-reference **`sref`** of the job template that has been created:

BODY	`"results": {`
	`  "sref": "jobTemplates/123"`
	`}`

You can use the **`sref`** of the newly created job template to run a new job from it, as documented above.