WHOOP API

Download OpenAPI specification: Download

Authentication

OAuth

Security Scheme Type	OAuth2

authorizationCode OAuth Flow	**Authorization URL:** https://api.prod.whoop.com/oauth/oauth2/auth **Token URL:** https://api.prod.whoop.com/oauth/oauth2/token **Scopes:** • `read:recovery` - Read Recovery data, including score, heart rate variability, and resting heart rate. • `read:cycles` - Read cycles data, including day Strain and average heart rate during a physiological cycle. • `read:workout` - Read workout data, including activity Strain and average heart rate. • `read:sleep` - Read sleep data, including performance % and duration per sleep stage. • `read:profile` - Read profile data, including name and email. • `read:body_measurement` - Read body measurements data, including height, weight, and max heart rate.

Cycle

getCycleById

Get the cycle for the specified ID

AUTHORIZATIONS: OAuth (`read:cycles`)

PATH PARAMETERS

`cycleId` integer <int64>
`required`
ID of the cycle to retrieve

Responses

200 Successful request

— **400** Client error constructing the request

— **401** Invalid authorization

— **404** No resource found

— **429** Request rejected due to rate limiting

— **500** Server error occurred while making request

GET `/v1/cycle/{cycleId}`

Response samples

200

Content type

application/json

Copy Expand all Collapse all


```
{
  "id": 93845,
  "user_id": 10129,
  "created_at": "2022-04-24T11:25:44.774Z",
  "updated_at": "2022-04-24T14:25:44.774Z",
  "start": "2022-04-24T02:25:44.774Z",
  "end": "2022-04-24T10:25:44.774Z",
  "timezone_offset": "-05:00",
  "score_state": "SCORED",
```

```
  - "score": {
        "strain": 5.2951527,
        "kilojoule": 8288.297,
        "average_heart_rate": 68,
        "max_heart_rate": 141
    }
}
```


getCycleCollection

Get all physiological cycles for a user, paginated. Results are sorted by start time in descending order.

AUTHORIZATIONS: OAuth (`read:cycles`)

QUERY PARAMETERS

`limit`
integer <int32> <= 25
Default: `10`
Limit on the number of cycles returned

`start`
string <date-time>
Return cycles that occurred after or during (inclusive) this time. If not specified, the response will not filter cycles by a minimum time.

`end`
string <date-time>
Return cycles that intersect this time or ended before (exclusive) this time. If not specified, `end` will be set to `now`.

`nextToken`
string
Optional next token from the previous response to get the next page. If not provided, the first page in the collection is returned

Responses

200 Successful request

— **400** Client error constructing the request

— **401** Invalid authorization

— **429** Request rejected due to rate limiting

— **500** Server error occurred while making request

GET `/v1/cycle`

Response samples

200

Content type
application/json

Copy Expand all Collapse all


```
{
 - "records": [
    + { … }
   ],
   "next_token": "MTIzOjEyMzEyMw"
}
```

Recovery

getRecoveryCollection

Get all recoveries for a user, paginated. Results are sorted by start time of the related sleep in descending order.

AUTHORIZATIONS: OAuth (`read:recovery`)

QUERY PARAMETERS

`limit`	integer <int32> <= 25 Default: `10` Limit on the number of recoveries returned
`start`	string <date-time> Return recoveries that occurred after or during (inclusive) this time. If not specified, the response will not filter recoveries by a minimum time.
`end`	string <date-time> Return recoveries that intersect this time or ended before (exclusive) this time. If not specified, `end` will be set to `now`.
`nextToken`	string Optional next token from the previous response to get the next page. If not provided, the first page in the collection is returned

Responses

200 Successful request

— **400** Client error constructing the request

— **401** Invalid authorization

— **429** Request rejected due to rate limiting

— **500** Server error occurred while making request

GET `/v1/recovery`

Response samples

200

Content type

application/json

Copy Expand all Collapse all


```
{
  - "records": [
      + { … }
    ],
    "next_token": "MTIzOjEyMzEyMw"
}
```


---

getRecoveryForCycle

Get the recovery for a cycle

AUTHORIZATIONS: OAuth (`read:recovery`)

PATH PARAMETERS

`cycleId` integer <int64>
`required`
ID of the cycle to retrieve

Responses

200 Successful request

— **400** Client error constructing the request

— **401** Invalid authorization

— **404** No resource found

— **429** Request rejected due to rate limiting

— **500** Server error occurred while making request

GET /v1/cycle/{cycleId}/recovery

Response samples

200

Content type

application/json

Copy Expand all Collapse all


```
{
    "cycle_id": 93845,
    "sleep_id": 10235,
    "user_id": 10129,
    "created_at": "2022-04-24T11:25:44.774Z",
    "updated_at": "2022-04-24T14:25:44.774Z",
    "score_state": "SCORED",
  - "score": {
        "user_calibrating": false,
        "recovery_score": 44,
        "resting_heart_rate": 64,
        "hrv_rmssd_milli": 31.813562,
        "spo2_percentage": 95.6875,
        "skin_temp_celsius": 33.7
    }
}
```


Sleep

getSleepById

Get the sleep for the specified ID

AUTHORIZATIONS: OAuth (`read:sleep`)

PATH PARAMETERS

`sleepId` integer <int64>
`required` ID of the sleep to retrieve

Responses

200 Successful request

— **400** Client error constructing the request

— **401** Invalid authorization

— **404** No resource found

— **429** Request rejected due to rate limiting

— **500** Server error occurred while making request

GET `/v1/activity/sleep/{sleepId}`

Response samples

200

Content type
application/json

Copy Expand all Collapse all


```
{
    "id": 93845,
    "user_id": 10129,
    "created_at": "2022-04-24T11:25:44.774Z",
```

```
    "updated_at": "2022-04-24T14:25:44.774Z",
    "start": "2022-04-24T02:25:44.774Z",
    "end": "2022-04-24T10:25:44.774Z",
    "timezone_offset": "-05:00",
    "nap": false,
    "score_state": "SCORED",
  - "score": {
     + "stage_summary": { … },
     + "sleep_needed": { … },
       "respiratory_rate": 16.11328125,
       "sleep_performance_percentage": 98,
       "sleep_consistency_percentage": 90,
       "sleep_efficiency_percentage": 91.69533848
    }
}
```


---

getSleepCollection

Get all sleeps for a user, paginated. Results are sorted by start time in descending order.

AUTHORIZATIONS: OAuth (`read:sleep`)

QUERY PARAMETERS

`limit`	integer <int32> <= 25 Default: `10` Limit on the number of sleeps returned
`start`	string <date-time> Return sleeps that occurred after or during (inclusive) this time. If not specified, the response will not filter sleeps by a minimum time.
`end`	string <date-time> Return sleeps that intersect this time or ended before (exclusive) this time. If not specified, `end` will be set to `now`.
`nextToken`	string Optional next token from the previous response to get the next page. If not provided, the first page in the collection is returned

Responses

200 Successful request

— **400** Client error constructing the request

— **401** Invalid authorization

— **429** Request rejected due to rate limiting

— **500** Server error occurred while making request

GET `/v1/activity/sleep`

Response samples

200

Content type

application/json

Copy Expand all Collapse all


```
{
- "records": [
    + { … }
  ],
  "next_token": "MTIzOjEyMzEyMw"
}
```

User

getBodyMeasurement

Get the user's body measurements

AUTHORIZATIONS: OAuth (`read:body_measurement`)

Responses

200 Successful request

— **400** Client error constructing the request

— **401** Invalid authorization

— **429** Request rejected due to rate limiting

— **500** Server error occurred while making request

GET `/v1/user/measurement/body`

Response samples

200

Content type

application/json

Copy

```
{
    "height_meter": 1.8288,
    "weight_kilogram": 90.7185,
    "max_heart_rate": 200
}
```


---

getProfileBasic

Get the user's Basic Profile

AUTHORIZATIONS: OAuth (`read:profile`)

Responses

200 Successful request

— **400** Client error constructing the request

— **401** Invalid authorization

— **429** Request rejected due to rate limiting

— **500** Server error occurred while making request

GET `/v1/user/profile/basic`

Response samples

200

Content type

application/json

Copy

```
{
    "user_id": 10129,
    "email": "jsmith123@whoop.com",
    "first_name": "John",
    "last_name": "Smith"
}
```


revokeUserOAuthAccess

Revoke the access token granted by the user. If the associated OAuth client is configured to receive webhooks, it will no longer receive them for this user.

AUTHORIZATIONS: OAuth

Responses

— **204** Successful request; no response body

— **400** Client error constructing the request

— **401** Invalid authorization

— **429** Request rejected due to rate limiting

— **500** Server error occurred while making request

DELETE `/v1/user/access`

Workout

getWorkoutById

Get the workout for the specified ID

AUTHORIZATIONS: OAuth (`read:workout`)

PATH PARAMETERS

`workoutId` integer <int64>
`required` ID of the workout to retrieve

Responses

200 Successful request

— **400** Client error constructing the request

— **401** Invalid authorization

— **404** No resource found

— **429** Request rejected due to rate limiting

— **500** Server error occurred while making request

GET `/v1/activity/workout/{workoutId}`

Response samples

200

Content type

application/json

Copy Expand all Collapse all


```
{
    "id": 1043,
    "user_id": 9012,
    "created_at": "2022-04-24T11:25:44.774Z",
    "updated_at": "2022-04-24T14:25:44.774Z",
    "start": "2022-04-24T02:25:44.774Z",
    "end": "2022-04-24T10:25:44.774Z",
    "timezone_offset": "-05:00",
    "sport_id": 1,
    "score_state": "SCORED",
  - "score": {
        "strain": 8.2463,
        "average_heart_rate": 123,
        "max_heart_rate": 146,
        "kilojoule": 1569.34033203125,
        "percent_recorded": 100,
        "distance_meter": 1772.77035916,
        "altitude_gain_meter": 46.64384460449,
        "altitude_change_meter": -0.781372010707855,
      + "zone_duration": { … }
    }
}
```


---

getWorkoutCollection

Get all workouts for a user, paginated. Results are sorted by start time in descending order.

AUTHORIZATIONS: OAuth (`read:workout`)

QUERY PARAMETERS

`limit`	integer <int32> <= 25 Default: 10	

	Limit on the number of workouts returned
`start`	string <date-time> Return workouts that occurred after or during (inclusive) this time. If not specified, the response will not filter workouts by a minimum time.
`end`	string <date-time> Return workouts that intersect this time or ended before (exclusive) this time. If not specified, `end` will be set to `now`.
`nextToken`	string Optional next token from the previous response to get the next page. If not provided, the first page in the collection is returned

Responses

200 Successful request

— **400** Client error constructing the request

— **401** Invalid authorization

— **429** Request rejected due to rate limiting

— **500** Server error occurred while making request

GET `/v1/activity/workout`

Response samples

200

Content type

application/json

Copy Expand all Collapse all

```
{
  - "records": [
      + { … }
    ],
    "next_token": "MTIzOjEyMzEyMw"
}
```

WHOOP API Terms of Use

By accessing or using the WHOOP application programming interface, software development kits, documentation, other developer services, and associated software (collectively, "APIs"), you agree to these API Terms of Use (collectively, with any additional terms, terms within any accompanying documentation, and any applicable policies and guidelines “Terms”).

Under the Terms, (a) "**WHOOP**" means Whoop, Inc., a Delaware corporation with principal offices at 1325 Boylston St. #401, Boston, MA 02215; and (b) “**Company**” means anyone accessing or using the WHOOP APIs. Each of WHOOP and Company may be referred to herein separately as a “**Party**” and collectively as the “**Parties**.”

1. Use of WHOOP APIs

a. Acceptance of the Terms. By entering into the Terms and using the APIs, Company represents and warrants that Company is not a person barred from using or receiving the APIs under the laws of the United States or other countries (including the country in which Company is a resident or from which Company uses the APIs). Company agrees to comply with the Terms and that the Terms control Company’s relationship with WHOOP with respect to the APIs. If Company uses the APIs as an interface to, or in conjunction with other WHOOP products or services, then the terms for those other products or services also apply.

b. Registration. In order to access and/or continue to use certain APIs, Company may be required to provide WHOOP with certain information from time to time (such as identification or contact details). Any such information must always be accurate and up to date, and Company agrees to promptly inform WHOOP of any updates to such information. If required, once approved to access and/or continue to use the APIs, Company is authorized to use the APIs to develop Company Applications subject at all times to the restrictions, conditions, and limitations in the Terms.

c. Compliance with Law, Third Party Rights, and Other WHOOP Terms of Use. Company will (i) comply with all applicable laws, regulations, and third party rights (including without limitation laws regarding the import or export of data or software, privacy, and local laws); (ii) require Company End Users to comply with (and not knowingly enable them to violate) applicable laws, regulations, and the Terms; (iii) not use the APIs to encourage or promote illegal activity or violation of third party rights; and (iv) not violate any applicable terms of use with WHOOP (or its affiliates).

d. Permitted Access. Company will only access (or attempt to access) any API by the means described in the documentation of that API. If WHOOP assigns Company developer credentials (e.g., client IDs), Company must use them with the applicable APIs. Company will not misrepresent or mask either Company identity or Company Applications’ identity when using the APIs or developer accounts. Company will not share Company credentials with any other developer or use them for more than one Application.

e. API Limitations. WHOOP may set and enforce limits on Company's use of the APIs (e.g., limiting the number of API requests that Company may make, or the number of users Company may serve), in WHOOP's sole discretion. Company agrees to abide by, and will not attempt to circumvent, such limitations documented with each API. If Company would like to use any API beyond these limits, Company must obtain the prior written consent of WHOOP (and WHOOP may decline such request or condition acceptance on Company's agreement to additional terms and/or charges for that use), which may be requested by contacting WHOOP at apisupport@whoop.com.

f. Monitoring. COMPANY AGREES THAT (I) WHOOP MAY (A) MONITOR USE OF THE APIS; AND (B) USE INFORMATION COLLECTED FROM SUCH MONITORING FOR ANY BUSINESS PURPOSE, INTERNAL OR EXTERNAL, INCLUDING WITHOUT LIMITATION TO ENSURE QUALITY, IMPROVE WHOOP PRODUCTS AND SERVICES, PROVIDE DEVELOPER SUPPORT, OR OTHERWISE; AND (II) COMPANY WILL NOT INTERFERE WITH SUCH MONITORING. Monitoring may include WHOOP accessing and using Company Applications (e.g., to identify security issues that could affect WHOOP or its members). WHOOP may suspend access to the APIs by Company or Company Applications without notice if WHOOP reasonably believes that Company is in violation of the Terms.

g. Open-Source Software. Certain software required by or included in the APIs may be open-source software governed by an open-source license. Any such open-source software is licensed under the terms of the end-user license that accompanies such open-source software. Nothing in this Agreement limits Company's rights under, or grants Company rights that supersede, the terms and conditions of the applicable end-user license for such open-source software.

h. HIPAA. Unless otherwise agreed in writing by WHOOP, WHOOP makes no representations that the APIs satisfy any obligations or meet any requirements under the Health Insurance Portability and Accountability Act, as amended ("**HIPAA**"), and does not intend use of the APIs to create any obligations under HIPAA. If Company is (or becomes) a "covered entity" or "business associate" (each as defined under HIPAA) Company agrees that it will not use the APIs for any purpose or in any manner involving transmitting Protected Health Information (as defined by HIPAA) unless Company has received prior written consent to such use from WHOOP. Company is solely responsible for any applicable compliance with HIPAA and agrees to hold WHOOP harmless for any uses contrary to this provision.

i. Non-Exclusivity. The APIs are provided on a non-exclusive basis. Company acknowledges that WHOOP and other users of the APIs may develop products or services that may be similar to or may compete with Company Applications, products or services.

j. Ownership. By using the APIs, Company does not acquire ownership of any rights in the APIs or the content that is accessed through the APIs. WHOOP does not acquire ownership in Company Applications.

k. Changes to APIs; Technical Support. WHOOP may modify, remove or add portions of, or otherwise update the APIs from time to time in WHOOP's sole discretion. WHOOP has no obligation to (i) ensure that any update of the APIs continues to be compatible with Company Applications; or (ii) provide any type of technical or other support for the APIs.

l. Pricing. Access to and use of the APIs is currently provided at no charge. In the event WHOOP decides to start charging for use of the APIs in the future, WHOOP will provide Company with prior notice.

m. Suggestions. Company may choose to provide WHOOP with feedback, suggestions or comments regarding the APIs or the WHOOP Platform, or Company's use thereof (collectively, "**Suggestions**"). Company agrees that WHOOP will be free to use, copy, modify, create derivative works, distribute, publicly display, publicly perform, grant sublicenses to, and otherwise exploit in any manner any such Suggestions, for any and all purposes, with no obligation of any kind to Company.

2. Company Applications

a. End User Authorization and Consent. Company Applications must (i) be expressly authorized by each end user of the Application (an "**End User**") prior to Company accessing any of such End User's data from the WHOOP Platform; (ii) allow the End User to access such End User's data that Company has collected via the APIs at the request of such End User; and (iii) provide easily accessible End User support contact information. Company represents, warrants and covenants that it prior to Company accessing any of such End User's data from the WHOOP Platform (a) it has obtained all necessary and appropriate consents required by all applicable laws, regulations or rules, including without limitation all federal, state, local, and international privacy and data security related laws and regulations that are, or which may in the future be, applicable to information identifying or relating to a particular individual, including information referred to as "personally identifiable information" or "personal information" and other data (such laws, regulations or rules, collectively, "**Data Privacy Laws**"), to allow Whoop to provide such End User data to Company; and (b) it will use such End User data in compliance with all applicable laws, regulations or rules, including without limitation all Data Protection Laws.

b. Application Support. Company is responsible for providing all customer and technical support and maintenance for Company Applications.

c. End User Privacy. Company will comply with all Data Privacy Laws. Company will provide and adhere to a privacy policy for Company Applications that clearly and accurately describes to End Users the information, including any End User personal information, Company collects and how Company uses and shares such information. Company agrees not to transfer or disclose user information to any third parties, except as expressly permitted by the Terms, End Users, Company's then-current privacy policy, and in compliance with all applicable laws, including Data Privacy Laws.

d. Application Security. Company agrees (i) that Company is fully responsible for the security of End User information in connection with Company Applications; (i) to use commercially reasonable and appropriate administrative, technical, and physical measures to protect End User information collected by Company Applications, including End User personal information, from unauthorized access or use; (ii) to ensure that any WHOOP data or information is encrypted (in transit and at rest) and transmitted over a secure, encrypted channel (e.g., HTTPS); and (iii) to promptly report to End Users any unauthorized access or use of such information (a "**Security Incident**") to the extent required by applicable law. Company must notify WHOOP of any Security Incident as soon

as possible, but in any event without undue delay after the discovery of such Security Incident by contacting WHOOP at apisupport@whoop.com.

e. Advertisements. Company must not use WHOOP data in any advertisements without the prior written consent of WHOOP. Company advertisements may not be displayed in any manner that suggests approval or endorsement by WHOOP.

3. Restrictions; Confidentiality

a. API Prohibitions. When using the APIs, Company is responsible for all use that occurs under Company's credentials, and Company may not (or allow those acting on Company's behalf to):
   i. Impose any terms on End Users that are inconsistent with the Terms.
   ii. Sell, rent, lease, redistribute, or syndicate access to any services WHOOP makes available through the APIs; provided that Company is not prohibited from charging a fee for the provision of functionality not provided by the WHOOP Platform in Company Applications.
   iii. Market, sell, license or lease data transferred or accessed through the API, directly or indirectly, to any third party even if an End User consents to such use.
   iv. Sublicense any API for use by a third party.
   v. Create any Applications that function substantially the same as the APIs.
   vi. Introduce or distribute any viruses, worms, defects, Trojan horses, spyware, malware, or any items of a malicious or destructive nature.
   vii. Distribute unsolicited advertising or promotions, send messages, make comments, or initiate any other unsolicited direct communication with anyone.
   viii. Use the APIs in, or in connection with, any application, website or other product or service that includes content that is (A) defamatory, libelous, hateful, violent, obscene, pornographic, unlawful, or otherwise offensive; or (B) promotes or facilitates online gambling, disruptive commercial messages or advertisements.
   ix. Interfere with, overburden, disrupt or otherwise impair the APIs or the servers or networks providing the APIs.
   x. Reverse engineer or attempt to extract any WHOOP algorithm or the source code from any API or any related software, except to the extent that this restriction is expressly prohibited by applicable law.
   xi. Use the APIs to process or store any data that is subject to the International Traffic in Arms Regulations maintained by the U.S. Department of State or U.S. Export Administration Regulations maintained by the U.S. Department of Commerce.
   xii. Remove, obscure, or alter any WHOOP terms of service or any links to or notices of those terms.
   xiii. Compete, directly or indirectly, with WHOOP or its products and services (including the WHOOP Platform) in any manner.
   xiv. Disparage WHOOP or participate in activities perceived as detrimental or harmful to WHOOP, its business or reputation.

b. Confidentiality.
   i. Developer credentials (such as passwords, keys, and client IDs) are intended to be used by Company and identify Company Applications. Company will keep Company credentials

confidential and shall not allow other Applications to use Company credentials. Developer credentials may not be embedded in open-source projects. If Company believes an unauthorized person has gained access to Company credentials, Company must notify WHOOP as soon as possible at apisupport@whoop.com.

ii. WHOOP communications to Company and the APIs may contain Confidential Information. Company agrees not to disclose any Confidential Information to any third party without the prior written consent of WHOOP. "**Confidential Information**" includes, without limitation, all financial, business, legal and technical information of WHOOP or any of its affiliates, suppliers, customers and employees (including information about research, development, operations, marketing, transactions, regulatory affairs, discoveries, inventions, methods, processes, articles, materials, algorithms, software, specifications, designs, drawings, data, strategies, plans, prospects, know-how and ideas, whether tangible or intangible), that is marked or otherwise identified as proprietary or confidential at the time of disclosure, or that by its nature would be understood by a reasonable person to be proprietary or confidential, including all copies, abstracts, notes, summaries, analyses and other derivatives thereof. Confidential Information does not include information that Company independently developed, that was rightfully given to Company by a third party without confidentiality obligation, or that becomes public through no fault of Company. Company may disclose Confidential Information when compelled to do so by law if Company provides WHOOP reasonable prior notice (to the extent legally permissible).

4. WHOOP Data

a. WHOOP Data Accessible Through the APIs. The APIs contain user information, including End User personal information, WHOOP calculations and analysis, as well as some third-party content such as text, images, videos, audio, or software ("**WHOOP Data**"). WHOOP Data is the sole responsibility of the person that makes it available. In WHOOP's sole discretion, WHOOP may sometimes review WHOOP Data to determine whether it is illegal or violates any WHOOP policies or the Terms, and WHOOP may remove or refuse to display WHOOP Data. Finally, WHOOP Data accessible through the APIs may be subject to intellectual property rights, and, if so, Company may not use it unless Company is licensed to do so by the owner of that WHOOP Data or are otherwise permitted by law. Company's access to the WHOOP Data provided by the APIs may be restricted, limited, or filtered in accordance with applicable law, regulation, and policy.

b. Prohibitions on WHOOP Data. Unless expressly permitted by the WHOOP Data owner or by applicable law, Company will not, and will not permit Company's End Users or others acting on Company's behalf to, do the following with WHOOP Data returned from the APIs:

i. Expose WHOOP Data to other users or to third parties without explicit opt-in consent from the user that provided the WHOOP Data and in accordance with the Terms;

ii. Scrape, build databases, or otherwise create permanent copies of WHOOP Data, or keep cached copies longer than permitted by the cache header;

iii. Copy, translate, modify, create a derivative work of, sell, lease, lend, convey, distribute, publicly display, or sublicense to any third party;

iv. Misrepresent the source or ownership; or

v. Remove, obscure, or alter any copyright, trademark, or other proprietary rights notices; or falsify or delete any author attributions, legal notices, or other labels of the origin or source of material.

5. WHOOP Brand

a. Use Restrictions. Subject to the Terms, WHOOP hereby grants to Company a nontransferable, nonsublicenseable, nonexclusive license while the Terms are in effect to display the trade names, trademarks, service marks, logos, domain names, and other distinctive brand features of WHOOP (collectively, the "**Brand Elements**") for the sole purpose of promoting or advertising that Company uses the APIs. Company agrees to (i) display any attribution(s) required by WHOOP as described in the documentation for the API; (ii) only use the WHOOP Brand Elements in accordance with the Terms and for the purpose of fulfilling Company's obligations under this Section; (iii) abide by any additional Brand Elements usage guidelines. Company understands and agrees that WHOOP has the sole discretion to determine whether Company's attribution(s) and use of the Brand Elements are in accordance with the above requirements and guidelines. Except where expressly stated, the Terms do not grant Company any right, title, or interest in or to the Brand Elements.

b. Publicity. Company will not make any statement, press release, or other announcement that references WHOOP or the APIs without the prior written approval of WHOOP.

c. Promotions and Marketing. In the course of promoting, marketing, or demonstrating the APIs Company is using and the associated WHOOP products, WHOOP may produce and distribute incidental depictions, including screenshots, video, or other content from Company Applications, and may use Company or product name. Company grants WHOOP all necessary rights for the above purposes.

6. WHOOP Privacy and Security

a. WHOOP Privacy Policies. By using the APIs, Company represents and warrants that Company has reviewed the WHOOP Terms of Use and the WHOOP Privacy Policy and understands how WHOOP collects, uses and safeguards the information, including any End User personal information, Company may provide to WHOOP.

b. WHOOP Security. WHOOP is responsible for protecting the security of End User personal information in its possession and will maintain commercially reasonable and appropriate administrative, technical, and physical procedures to protect all the End User personal information that is stored on WHOOP servers from unauthorized access and accidental loss or modification. If Company believes it has identified a security vulnerability, Company must notify the WHOOP security team as soon as possible at apisupport@whoop.com.

7. Termination

a. Termination. Company may stop using the APIs at any time. To terminate the Terms, Company must provide WHOOP with written notice. WHOOP reserves the right to terminate the Terms with

Company or discontinue the APIs or any portion or feature or Company's access thereto for any reason and at any time without liability or other obligation to Company. Upon any termination of the Terms or if Company's access to an API is discontinued, Company must immediately stop using the applicable APIs, cease all use of the Brand Elements, and delete any cached or stored content that was permitted under the Terms. WHOOP may independently communicate with any account owner whose account(s) are associated with Company Applications and developer credentials to provide notice of the termination of Company's right to use an API.

8. Liability

a. WARRANTIES. WHOOP PROVIDES THE APIS "AS IS" AND ON AN "AS AVAILABLE" BASIS, WITHOUT WARRANTY OF ANY KIND AND DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTABILITY, NON-INFRINGEMENT OF THIRD PARTIES' RIGHTS, AND FITNESS FOR A PARTICULAR PURPOSE.
WHOOP, ITS AFFILIATES AND PARTNERS, AND ITS AND THEIR RESPECTIVE OFFICERS, DIRECTORS, EMPLOYEES, AGENTS, SUPPLIERS, OR LICENSORS, MAKE NO WARRANTIES OR REPRESENTATIONS (I) THAT THE APIS WILL MEET COMPANY OR COMPANY'S END USERS' REQUIREMENTS, OR THOSE OF COMPANY APPLICATIONS, (ii) THAT COMPANY'S USE OF THE APIS WILL BE UNINTERRUPTED, TIMELY, SECURE, OR ERROR-FREE, (iii) THAT ANY ERRORS IN THE APIS WILL BE CORRECTED AND/OR (IV) ABOUT THE WHOOP DATA (INCLUDING THE END USER WHOOP DATA OR OTHER CONTENT) OR SERVICES, INCLUDING BUT NOT LIMITED TO ACCURACY, RELIABILITY, COMPLETENESS, TIMELINESS, OR RELIABILITY. TO MAXIMUM THE EXTENT PERMITTED BY LAW, WHOOP EXCLUDES AND DISCLAIMS ALL EXPRESS AND IMPLIED WARRANTIES, GUARANTEES, CONDITIONS, REPRESENTATIONS, AND UNDERTAKINGS.
b. WAIVER OF DAMAGES; LIMITATION OF LIABILITY. UNDER NO CIRCUMSTANCE WILL WHOOP, ITS AFFILIATES OR PARTNERS, OR ITS AND THEIR RESPECTIVE OFFICERS, DIRECTORS, EMPLOYEES, AGENTS, SUPPLIERS, OR LICENSORS BE LIABLE OR RESPONSIBLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, EXEMPLARY, PUNITIVE OR CONSEQUENTIAL DAMAGES (INCLUDING, WITHOUT LIMITATION, LOSS OF USE, DATA, BUSINESS OR PROFITS) ARISING OUT OF OR IN CONNECTION WITH THIS AGREEMENT OR COMPANY'S USE OF THE APIS, THE WHOOP PLATFORM OR THE BRAND ELEMENTS, WHETHER SUCH LIABILITY ARISES FROM ANY CLAIM BASED UPON CONTRACT, WARRANTY, TORT (INCLUDING, WITHOUT LIMITATION, NEGLIGENCE), STRICT LIABILITY OR OTHERWISE, AND WHETHER OR NOT COMPANY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE. THE FOREGOING LIMITATIONS WILL SURVIVE AND APPLY EVEN IF ANY LIMITED REMEDY SPECIFIED IN THE TERMS IS FOUND TO HAVE FAILED ITS ESSENTIAL PURPOSE. IN ANY EVENT, THE AGGREGATE LIABILITY OF WHOOP UNDER THE AGREEMENT WILL NOT EXCEED THE GREATER OF (I) THE FEES COMPANY PAID TO WHOOP FOR USE OF THE APIS IN THE LAST YEAR OR (II) ONE HUNDRED DOLLARS ($100).
c. Indemnification. To the maximum extent allowed by law, Company will defend, indemnify and hold harmless WHOOP, WHOOP, its affiliates and its and their directors, officers, employees, agents, users, licensors and partners, against all liabilities, damages, losses, costs, fees (including

legal fees), and expenses relating to any allegation or third-party legal proceeding arising out of or in connection with (i) Company's use of the APIS, the WHOOP Platform, or the Brand Elements other than as expressly allowed by this Agreement; (ii) Company's breach or alleged breach of any of the terms, conditions, representations, warranties or covenants under the Terms; (iii) Company's applications or business; (iv) Company's negligence, willful misconduct or fraud; or (v) any content or data, including WHOOP Data, routed into or used with the APIs by Company, those acting on Company's behalf, or End Users.

d. Allocation of Risk. The parties acknowledge that the terms of this Section reflect the allocation of risk set forth in the Terms and that the parties would not have entered into the Terms without these waivers, limitations of liability and indemnities.

9. General Provisions

a. Modification. Upon notice to Company, WHOOP may modify the Terms or any portion to, for example, reflect changes to the law or changes to the APIs. If Company does not agree to the modified Terms for an API, Company should discontinue its use of that API. Company's continued use of the API constitutes acceptance of the modified Terms.

b. U.S. Federal Agency Entities. The APIs were developed solely at private expense and are commercial computer software and related documentation within the meaning of the applicable U.S. Federal Acquisition Regulation and agency supplements thereto.

c. Miscellaneous. The Terms do not create any third-party beneficiary rights or any agency, partnership, or joint venture. WHOOP is not liable for failure or delay in performance to the extent caused by circumstances beyond WHOOP's reasonable control. No waiver or failure to exercise any option, right, or privilege under the Terms on any occasion or occasions shall be construed to be a waiver of the same or any other option, right or privilege on any other occasion. Nothing in the Terms will limit either Party's ability to seek injunctive relief.

d. Assignment. Neither Party may assign the Terms without the prior written consent of the other Party; provided that WHOOP may, without Company's consent, assign the Terms (i) to an affiliate or (ii) in connection with a merger, acquisition, corporate reorganization, or sale of all, or substantially all of WHOOP's assets. In the case of any permitted assignment or transfer of or under the Terms, the Terms or the relevant provisions shall inure to the benefit of and be binding upon the Parties' respective executors, heirs, representatives, administrators and assigns.

e. Entire Agreement; Severability. The Terms constitute the entire agreement of the Parties relating to the subject matter of the Terms, represents the exclusive understanding of the Parties, either oral or written, and supersedes any prior written or oral agreement, representations and discussions between the Parties regarding its subject matter. If for any reason a court of competent jurisdiction finds any provision of the Terms to be unenforceable, that provision will be enforced to the maximum extent permissible so as to effect the intent of the Parties, and the remainder the Terms will continue in full force and effect.

f. Choice of Law/Venue. The Terms will be governed by and construed in accordance with the laws of the Commonwealth of Massachusetts and the Parties agree to submit to the exclusive jurisdiction of the courts located in Suffolk county in Massachusetts.

g. Surviving Provisions. Those terms that by their nature are intended to continue indefinitely will continue to apply following any termination of these Terms.

WHOOP API

Download OpenAPI specification: Download

Authentication

OAuth

Security Scheme Type	OAuth2

authorizationCode OAuth Flow	**Authorization URL:** https://api.prod.whoop.com/oauth/oauth2/auth **Token URL:** https://api.prod.whoop.com/oauth/oauth2/token **Scopes:** • `read:recovery` - Read Recovery data, including score, heart rate variability, and resting heart rate. • `read:cycles` - Read cycles data, including day Strain and average heart rate during a physiological cycle. • `read:workout` - Read workout data, including activity Strain and average heart rate. • `read:sleep` - Read sleep data, including performance % and duration per sleep stage. • `read:profile` - Read profile data, including name and email. • `read:body_measurement` - Read body measurements data, including height, weight, and max heart rate.

Cycle

getCycleById

Get the cycle for the specified ID

AUTHORIZATIONS: OAuth (`read:cycles`)

PATH PARAMETERS

`cycleId` integer <int64>
`required`
ID of the cycle to retrieve

Responses

200 Successful request

— **400** Client error constructing the request

— **401** Invalid authorization

— **404** No resource found

— **429** Request rejected due to rate limiting

— **500** Server error occurred while making request

GET `/v1/cycle/{cycleId}`

Response samples

200

Content type

application/json

Copy Expand all Collapse all


```
{
  "id": 93845,
  "user_id": 10129,
  "created_at": "2022-04-24T11:25:44.774Z",
  "updated_at": "2022-04-24T14:25:44.774Z",
  "start": "2022-04-24T02:25:44.774Z",
  "end": "2022-04-24T10:25:44.774Z",
  "timezone_offset": "-05:00",
  "score_state": "SCORED",
```

```
  - "score": {
        "strain": 5.2951527,
        "kilojoule": 8288.297,
        "average_heart_rate": 68,
        "max_heart_rate": 141
    }
}
```


getCycleCollection

Get all physiological cycles for a user, paginated. Results are sorted by start time in descending order.

AUTHORIZATIONS: OAuth (`read:cycles`)

QUERY PARAMETERS

`limit`
integer <int32> <= 25
Default: `10`
Limit on the number of cycles returned

`start`
string <date-time>
Return cycles that occurred after or during (inclusive) this time. If not specified, the response will not filter cycles by a minimum time.

`end`
string <date-time>
Return cycles that intersect this time or ended before (exclusive) this time. If not specified, `end` will be set to `now`.

`nextToken`
string
Optional next token from the previous response to get the next page. If not provided, the first page in the collection is returned

Responses

200 Successful request

— **400** Client error constructing the request

— **401** Invalid authorization

— **429** Request rejected due to rate limiting

— **500** Server error occurred while making request

GET `/v1/cycle`

Response samples

200

Content type
application/json

Copy Expand all Collapse all


```
{
 - "records": [
    + { … }
   ],
   "next_token": "MTIzOjEyMzEyMw"
}
```

Recovery

getRecoveryCollection

Get all recoveries for a user, paginated. Results are sorted by start time of the related sleep in descending order.

AUTHORIZATIONS: OAuth (`read:recovery`)

QUERY PARAMETERS

`limit`	integer <int32> <= 25 Default: `10` Limit on the number of recoveries returned
`start`	string <date-time> Return recoveries that occurred after or during (inclusive) this time. If not specified, the response will not filter recoveries by a minimum time.
`end`	string <date-time> Return recoveries that intersect this time or ended before (exclusive) this time. If not specified, `end` will be set to `now`.
`nextToken`	string Optional next token from the previous response to get the next page. If not provided, the first page in the collection is returned

Responses

200 Successful request

— **400** Client error constructing the request

— **401** Invalid authorization

— **429** Request rejected due to rate limiting

— **500** Server error occurred while making request

GET `/v1/recovery`

Response samples

200

Content type

application/json

Copy Expand all Collapse all


```
{
  - "records": [
      + { … }
    ],
    "next_token": "MTIzOjEyMzEyMw"
}
```


---

getRecoveryForCycle

Get the recovery for a cycle

AUTHORIZATIONS: OAuth (`read:recovery`)

PATH PARAMETERS

`cycleId` integer <int64>
`required`
ID of the cycle to retrieve

Responses

200 Successful request

— **400** Client error constructing the request

— **401** Invalid authorization

— **404** No resource found

— **429** Request rejected due to rate limiting

— **500** Server error occurred while making request

GET /v1/cycle/{cycleId}/recovery

Response samples

200

Content type

application/json

Copy Expand all Collapse all


```
{
    "cycle_id": 93845,
    "sleep_id": 10235,
    "user_id": 10129,
    "created_at": "2022-04-24T11:25:44.774Z",
    "updated_at": "2022-04-24T14:25:44.774Z",
    "score_state": "SCORED",
  - "score": {
        "user_calibrating": false,
        "recovery_score": 44,
        "resting_heart_rate": 64,
        "hrv_rmssd_milli": 31.813562,
        "spo2_percentage": 95.6875,
        "skin_temp_celsius": 33.7
    }
}
```


Sleep

getSleepById

Get the sleep for the specified ID

AUTHORIZATIONS: OAuth (`read:sleep`)

PATH PARAMETERS

`sleepId` integer <int64>
`required` ID of the sleep to retrieve

Responses

200 Successful request

— **400** Client error constructing the request

— **401** Invalid authorization

— **404** No resource found

— **429** Request rejected due to rate limiting

— **500** Server error occurred while making request

GET `/v1/activity/sleep/{sleepId}`

Response samples

200

Content type
application/json

Copy Expand all Collapse all


```
{
    "id": 93845,
    "user_id": 10129,
    "created_at": "2022-04-24T11:25:44.774Z",
```

```
    "updated_at": "2022-04-24T14:25:44.774Z",
    "start": "2022-04-24T02:25:44.774Z",
    "end": "2022-04-24T10:25:44.774Z",
    "timezone_offset": "-05:00",
    "nap": false,
    "score_state": "SCORED",
  - "score": {
     + "stage_summary": { … },
     + "sleep_needed": { … },
       "respiratory_rate": 16.11328125,
       "sleep_performance_percentage": 98,
       "sleep_consistency_percentage": 90,
       "sleep_efficiency_percentage": 91.69533848
    }
}
```


---

getSleepCollection

Get all sleeps for a user, paginated. Results are sorted by start time in descending order.

AUTHORIZATIONS: OAuth (`read:sleep`)

QUERY PARAMETERS

`limit`	integer <int32> <= 25 Default: `10` Limit on the number of sleeps returned
`start`	string <date-time> Return sleeps that occurred after or during (inclusive) this time. If not specified, the response will not filter sleeps by a minimum time.
`end`	string <date-time> Return sleeps that intersect this time or ended before (exclusive) this time. If not specified, `end` will be set to `now`.
`nextToken`	string Optional next token from the previous response to get the next page. If not provided, the first page in the collection is returned

Responses

200 Successful request

— **400** Client error constructing the request

— **401** Invalid authorization

— **429** Request rejected due to rate limiting

— **500** Server error occurred while making request

GET `/v1/activity/sleep`

Response samples

200

Content type

application/json

Copy Expand all Collapse all


```
{
- "records": [
    + { … }
  ],
  "next_token": "MTIzOjEyMzEyMw"
}
```

User

getBodyMeasurement

Get the user's body measurements

AUTHORIZATIONS: OAuth (`read:body_measurement`)

Responses

200 Successful request

— **400** Client error constructing the request

— **401** Invalid authorization

— **429** Request rejected due to rate limiting

— **500** Server error occurred while making request

GET `/v1/user/measurement/body`

Response samples

200

Content type

application/json

Copy

```
{
    "height_meter": 1.8288,
    "weight_kilogram": 90.7185,
    "max_heart_rate": 200
}
```


---

getProfileBasic

Get the user's Basic Profile

AUTHORIZATIONS: OAuth (`read:profile`)

Responses

200 Successful request

— **400** Client error constructing the request

— **401** Invalid authorization

— **429** Request rejected due to rate limiting

— **500** Server error occurred while making request

GET `/v1/user/profile/basic`

Response samples

200

Content type

application/json

Copy

```
{
    "user_id": 10129,
    "email": "jsmith123@whoop.com",
    "first_name": "John",
    "last_name": "Smith"
}
```


revokeUserOAuthAccess

Revoke the access token granted by the user. If the associated OAuth client is configured to receive webhooks, it will no longer receive them for this user.

AUTHORIZATIONS: OAuth

Responses

— **204** Successful request; no response body

— **400** Client error constructing the request

— **401** Invalid authorization

— **429** Request rejected due to rate limiting

— **500** Server error occurred while making request

DELETE `/v1/user/access`

Workout

getWorkoutById

Get the workout for the specified ID

AUTHORIZATIONS: OAuth (`read:workout`)

PATH PARAMETERS

`workoutId` integer <int64>
`required` ID of the workout to retrieve

Responses

200 Successful request

— **400** Client error constructing the request

— **401** Invalid authorization

— **404** No resource found

— **429** Request rejected due to rate limiting

— **500** Server error occurred while making request

GET `/v1/activity/workout/{workoutId}`

Response samples

200

Content type

application/json

Copy Expand all Collapse all


```
{
    "id": 1043,
    "user_id": 9012,
    "created_at": "2022-04-24T11:25:44.774Z",
    "updated_at": "2022-04-24T14:25:44.774Z",
    "start": "2022-04-24T02:25:44.774Z",
    "end": "2022-04-24T10:25:44.774Z",
    "timezone_offset": "-05:00",
    "sport_id": 1,
    "score_state": "SCORED",
  - "score": {
        "strain": 8.2463,
        "average_heart_rate": 123,
        "max_heart_rate": 146,
        "kilojoule": 1569.34033203125,
        "percent_recorded": 100,
        "distance_meter": 1772.77035916,
        "altitude_gain_meter": 46.64384460449,
        "altitude_change_meter": -0.781372010707855,
      + "zone_duration": { … }
    }
}
```


---

getWorkoutCollection

Get all workouts for a user, paginated. Results are sorted by start time in descending order.

AUTHORIZATIONS: OAuth (`read:workout`)

QUERY PARAMETERS

`limit`	integer <int32> <= 25 Default: 10	

	Limit on the number of workouts returned
`start`	string <date-time> Return workouts that occurred after or during (inclusive) this time. If not specified, the response will not filter workouts by a minimum time.
`end`	string <date-time> Return workouts that intersect this time or ended before (exclusive) this time. If not specified, `end` will be set to `now`.
`nextToken`	string Optional next token from the previous response to get the next page. If not provided, the first page in the collection is returned

Responses

200 Successful request

— **400** Client error constructing the request

— **401** Invalid authorization

— **429** Request rejected due to rate limiting

— **500** Server error occurred while making request

GET `/v1/activity/workout`

Response samples

200

Content type

application/json

Copy Expand all Collapse all

```
{
  - "records": [
      + { … }
    ],
    "next_token": "MTIzOjEyMzEyMw"
}
```

API Changelog

This changelog highlights notable changes to the WHOOP Developer Platform, such as new API endpoints or webhook event types.

2024-05

Strength Trainer activities

Strength Trainer activities are available via the `/workout` endpoint.

2023-02

De-authorization endpoint

revokeUserOauthAccess – if a user wants to disable your integration, you can revoke their access token from your application in order to respect their privacy. This will ensure you no longer receive webhooks for this user.

2022-09

Developer Platform Launch 🚀

WHOOP releases their Developer Platform! Read more about the launch here.

App Approval

Apps can be used for development immediately with a limit of 10 WHOOP members. To launch your app to all WHOOP members, you must first submit your app for approval.

Approval Process

1. Ensure that you have read, are familiar with, and are abiding by the WHOOP API Terms of Use. As a reminder, in order to use the WHOOP API you must adhere to these terms.
2. Ensure that you have tested your app with at least one WHOOP member.
3. Verify your app information is correct in the Developer Dashboard. In particular, make sure your **App Name**, **Contact Email(s)**, and **Privacy Policy URL** are filled in and accurate to accelerate the approval process.
4. Verify your app adheres to the WHOOP Design and Brand Guidelines.
5. Once ready, fill out an App Submission request that includes designs and other helpful context via this link. Once approved, your app will be able to have more than 10 WHOOP members.

Design Guidelines

By integrating with the WHOOP API, you are helping our members accelerate their journey towards unlocking their performance by bringing their WHOOP data into your contextually relevant experience. We've established the following design and brand guidelines to ensure that the WHOOP brand and accompanying data is consistent for our members.

Design Guidelines

The WHOOP Design Guidelines is a valuable resource for designers and developers who wish to integrate WHOOP into their products. You'll find guidance for logo use, typefaces, colors, recommendations, and more in this comprehensive guide.

View Design Guidelines

Digital Assets

Below you can easily download acceptable versions of the WHOOP logo. Files are provided in SVG, PNG, and PDF formats.

Download Logo & Icon

By downloading any files from this page, you agree to the WHOOP Privacy Principles & Terms of Use.

Getting Started

To start developing with WHOOP, you first need to create an `App` in the WHOOP Developer Dashboard. If you're already registered in the Developer Dashboard, you can skip this section and learn how to authenticate with WHOOP using your app's `Client ID` and `Client Secret`.

The Developer Dashboard site manages Client Secrets and access to WHOOP OAuth API endpoints. The log-in for the Developer Dashboard uses the same credentials as your WHOOP account.

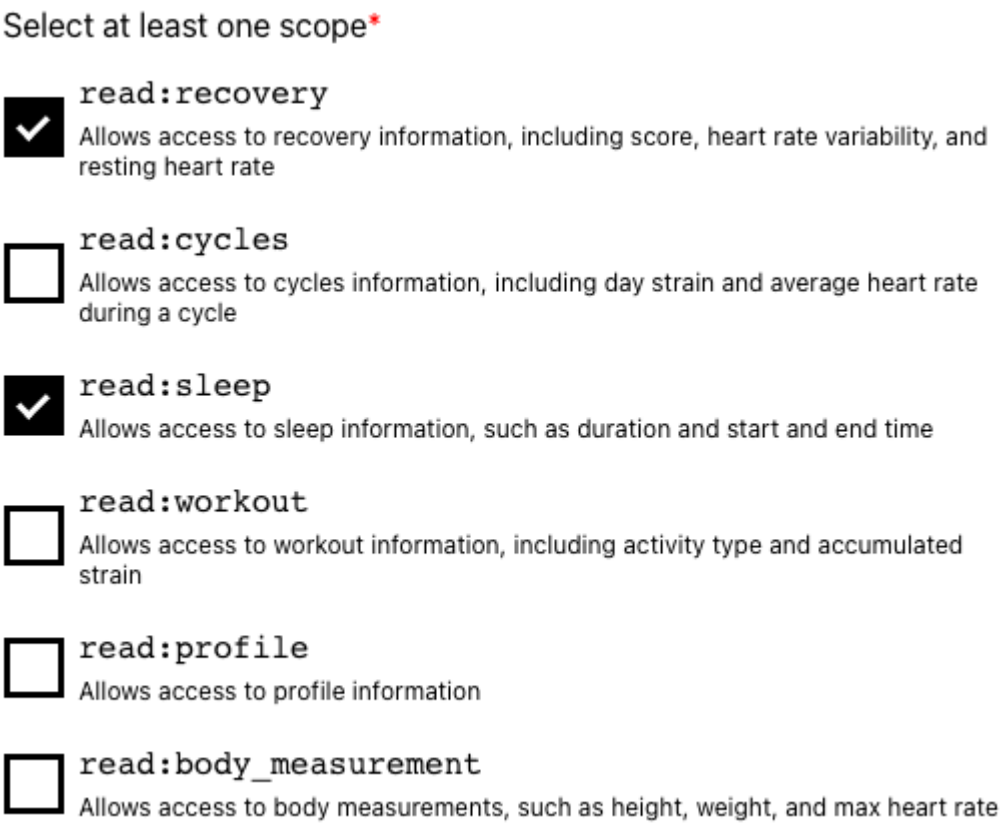
Creating your first App

To create your first App, head to the App creation flow in the Developer Dashboard. You will be prompted to create a Team if you have not yet done so.

You can create up to 5 Apps. If you need more, please submit a request for more apps via this link.

Scopes

Scopes define the limits of access your App will have to an individual's account data. At least one scope *must* be specified to create an App. When performing the OAuth flow, you can request one or more of these scopes from a WHOOP member.

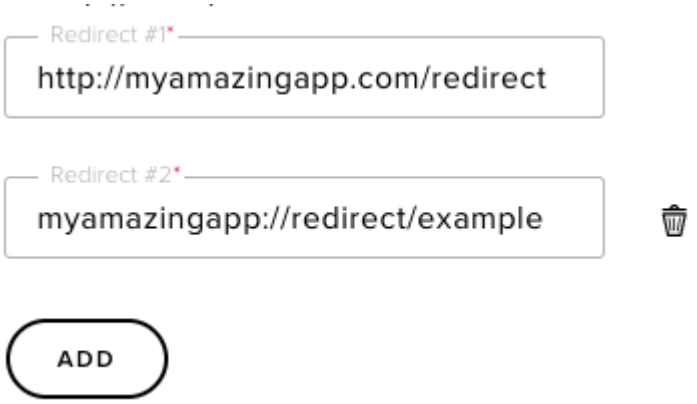


Please note that you should limit the scopes requested to *only* the ones your application will use.

You can learn more about the available scopes and what they represent from the API Docs.

Redirect URIs

You must specify at least one redirect URL for each App. The redirect URI on your OAuth authorization request must match the value in the Developer Dashboard. You may provide multiple Redirect URIs that your client needs.



Completing Client Creation

After filling in all the fields, click the **Create** button at the bottom to complete the process.

Accessing Client Credentials

After you have created your App, you will be provided with your `Client ID` and `Client Secret`. These values are required to complete the OAuth flow.

Note: your `Client Secret` should *never* be logged or shared with anyone. It should only be used server side and should never be exposed in a client, web, or mobile application.

After creating the App, you will be able to see these values later by clicking into the app from the Apps section, where all your current Apps are listed.

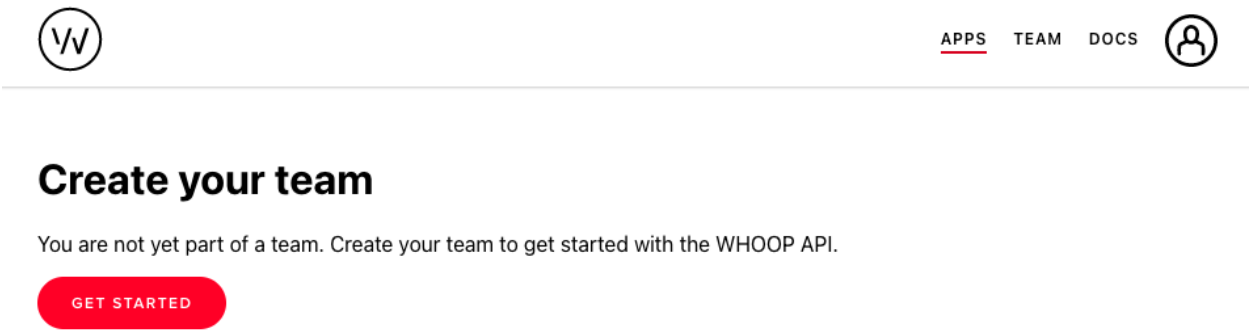
Editing or Deleting an App

To change any details about your app, navigate to the App, click the pencil icon, edit any values, and save your app.

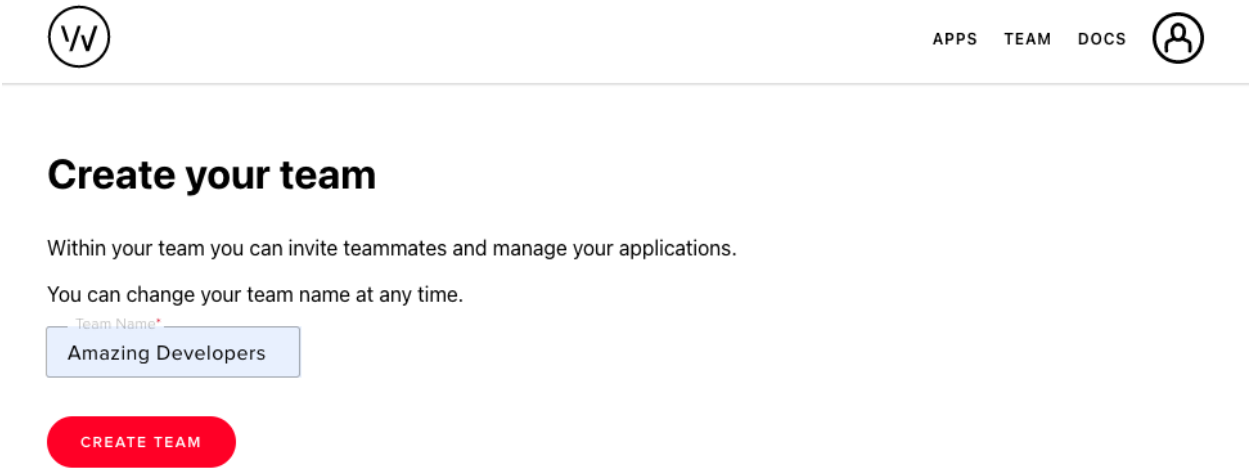
> **CAUTION!**
>
> Clicking the red trash can button will delete the app. Exercise caution when deleting apps, as this action cannot be undone.

Creating your team

Before creating your first app, you'll be prompted to create a Team.



After clicking **Get Started**, you'll be taken to the team creation form.

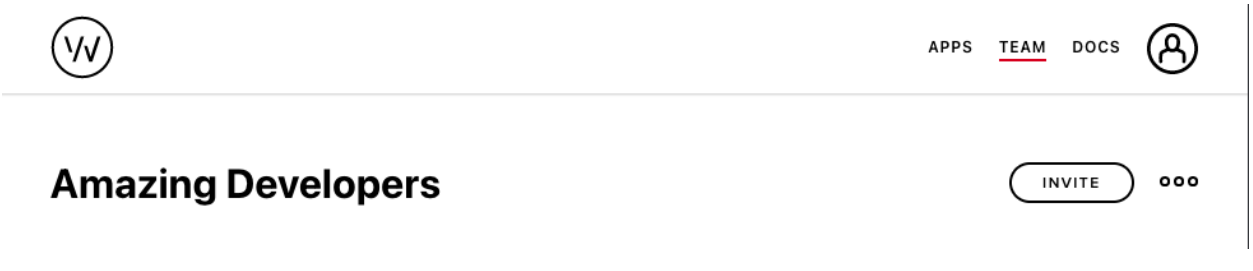


Choose a name for your team and click **Create Team**.

Adding Users To A Team

You can add other developers to your team to share access to app information and management.

To invite someone to your team, you can head to the Team section of the Developer Dashboard and click the **Invite** button in the top right corner.



From there, you can add another developer to your team. To add the developer to your team, you need the email address associated with their WHOOP account.

Fill in the WHOOP email of the other developer, and click **Invite**.

OAuth 2.0

All endpoints for retrieving a WHOOP user's data require successful authorization via the OAuth 2.0 protocol. IETF RFC-6749 documents the full standard. While we will not explain OAuth 2.0 generally in this section, we will demonstrate how to use it to authorize an app's access to a WHOOP user's data after the user grants consent.

Any subsequent references to *OAuth* in this documentation will be referring to *OAuth 2.0*.

We recommend you use a library to manage the details of the OAuth flow. You can find recommended libraries in a variety of programming languages here.

Pre-requisite

Before starting, you must create an App in the WHOOP Developer Dashboard. You can follow the getting started to learn how if you have not already.

Required Information for OAuth

Client ID

The `Client ID` can be found after creating a client in the WHOOP Developer Dashboard.

Client Secret

You can find the `Client Secret` after creating a Client in the WHOOP Developer Dashboard.

Authorization URL

This WHOOP URL prompts a user to authorize your app to access the WHOOP user's data. After the user grants authorization, WHOOP will respond with an authorization code.

```
https://api.prod.whoop.com/oauth/oauth2/auth
```

Token URL

This WHOOP URL provides your app with the authorization code from the authorization URL after the WHOOP user authorizes your app to access their WHOOP data. This endpoint provides an access

token in exchange for the authorization code.

```
https://api.prod.whoop.com/oauth/oauth2/token
```

Redirect URL

WHOOP will redirect the user to the Redirect URL after they authorize your App to access the request scopes. You must register the Redirect URL in the WHOOP Developer Dashboard.

A valid URL will take the form of `https://whoop.com/example/redirect` or `whoop://example/redirect`.

Note: Your OAuth library may reference `Redirect URL` as a `Callback URL`.

State

The state parameter is a security measure to prevent against CSRF attacks. It's a generated random string that you provide to WHOOP's server. Upon successful authorization by the user, WHOOP will respond with this same state value.

Your OAuth 2.0 library may handle this functionality for you; however, you may be responsible for explicitly using it by setting the option to true.

The state parameter must be eight characters long if you need to generate it yourself.

Scope

The scope parameter is how your application declares which data and what level of access is being requested.

When the OAuth flow prompts a WHOOP user to authorize access to your app, the user will see the list of scopes your app requested. The user must authorize access to them before WHOOP grants your app a token.

You can learn more about what scopes are available in the API Docs.

Request An Access Token

We highly recommend using a library to manage the flow. There are available libraries in many programming languages.

The steps that the library will follow are:

1. Your application will send a request to the Authorization URL for an authorization code.
2. WHOOP prompts the user to log in with their WHOOP credentials.
3. The user will confirm they authorize access to your app.
4. WHOOP will respond to the Redirect URL with an authorization code.
5. Your application will send a request to the Token URL, exchanging the authorization code for an access token (and optionally a refresh token if your app requested the `offline` scope).

Access Token Request Examples

- Javascript and Passport

Using An Access Token

After receiving an access token, all requests must include the access token. Your app should pass the access token as a `Bearer` token in the `Authorization` header.

WHOOP will return the requested data if the access token is successfully verified. However, if the token is invalid, WHOOP returns a `401 - Unauthorized` response.

Access Token Expiration

Access tokens are only valid for a short time. WHOOP provides the token expiry in the `expires_in` parameter (in seconds). After the token expires, your app must refresh the token.

Refreshing an Access Token

Access tokens are short-lived, depending on the `expires_in` value. WHOOP will respond with a (401 - Unauthorized) HTTP status code if your app sends a request with an expired access token.

When the access token is expired, you can no longer use it and must refresh the token. If you do not want to wait for the token to expire before refreshing, you can refresh the complete set of access tokens regularly. One strategy is to execute refreshing access tokens using a background cron job.

Existing access tokens are invalidated once your app uses the refresh token to generate a new access token. The access token from the refresh response is now the valid access token, and your app can use the new access token for future requests. Similarly, the refresh token from the refresh response is now the valid refresh token, and your app must use the new refresh token on the subsequent refresh request.

Another advantage to refreshing tokens in a recurring job is that it limits the likelihood of issues occurring with multiple concurrent requests attempting to refresh an expired token. When your app

makes simultaneous refresh token requests, the first refresh request that reaches WHOOP will succeed. The second request would fail because the first request invalidates the refresh token.

Receiving a Refresh Token

WHOOP provides your app with a refresh token after completing the OAuth 2.0 flow *if* the `offline` scope is included in the authorization request. You *must* request the `offline` scope to receive a refresh token.

Required Information To Refresh

Refresh Token Endpoint

The WHOOP URL where your app sends a POST request including the refresh token in exchange for a new access token.

```
https://api.prod.whoop.com/oauth/oauth2/token
```

Refresh Token

If your app requests the `offline` scope during the OAuth authorization flow, WHOOP returns a refresh token that your app uses to refresh expired access tokens.

Client ID

The Client ID is a unique identifier for your Client app. Your app uses this ID to authenticate with WHOOP. You can create and manage this ID in the WHOOP Developer Dashboard.

Client Secret

The Client Secret is a secret value that accompanies a Client ID. You can view the Client Secret in the WHOOP Developer Dashboard.

Sample POST Request Payload

```
{
  "grant_type": "refresh_token",
  "refresh_token": "{{RefreshToken}}",
  "client_id": "{{ClientID}}",
  "client_secret": "{{ClientSecret}}",
  "scope": "offline"
}
```

The payload above uses Postman variables, rather than actual data. The variables represent:

- RefreshToken - the value of the refresh token you receive along with an access token.
- ClientID - the client identifier created in the WHOOP Developer Dashboard.
- ClientSecret - the secret that accompanies the Client ID in the WHOOP Developer Dashboard.

Sample POST Response Payload

```
{
  "access_token": "the-value-of-the-new-access-token",
  "expires_in": 3600,
  "refresh_token": "the-value-of-the-new-refresh-token",
  "scope": "offline other-scopes-requested",
  "token_type": "bearer"
}
```

Token Refresh Examples

- Postman
- Javascript

Revoking an Access Token

When a user disables your integration, you should revoke their access token from your application to respect their privacy. If you have adopted webhooks, revoking an access token will ensure that you no longer receive webhooks for this user.

To revoke an access token, you can use the revokeUserOauthAccess API endpoint.

Overview

Before diving into specifics, let's review the process for launching an app on the WHOOP Developer Platform.

1. Sign up for WHOOP

You must have a WHOOP membership to develop an app on the Developer Platform. Your WHOOP account is also your WHOOP developer login. You can join WHOOP here.

2. Create an App in the WHOOP Developer Dashboard

The WHOOP Developer Dashboard is where you register your app with WHOOP. You can create a new app, access your Client Secret, and invite other developers to view and maintain the app. Getting started with the Developer Dashboard will walk you through registering your first app.

You can create up to 5 apps. If you need more, please submit a request for more apps via this link.

3. Authenticate with WHOOP using the OAuth 2.0 protocol

Once you have a `Client ID` and `Client Secret` in the Developer Dashboard, you can use the OAuth standard to request consent from WHOOP members to access data on their behalf. Once a WHOOP member grants your app permission, you will receive a per-user Access and Refresh Token that your app uses on requests to the WHOOP API.

4. Make requests to the WHOOP API

WHOOP exposes RESTful endpoints that you can access using the Access and Refresh Token from **Step #3: Authenticate with WHOOP**. You can view the documentation on the API endpoints here.

5. Launch your app

When you are ready to launch your app, submit it for approval.

Also, consider how request limits may impact your app.

Pagination

Collections such as the Cycle , Recovery, Sleep, and Workout Collections can include a large number of records. Responses to queries on the collection may be truncated, but you can paginate through the results to read the full collection.

The Collection APIs return a token to represent the current page on each response. You send that token on the subsequent request to retrieve the next page. You iterated through all pages if the `next_token` field is empty on the response.

Using the' limit' header, you can also control how many records the API endpoint returns per page. However, the API endpoint will not return more than the max allowed value (API Docs specify the max value on each endpoint).

Example

Let's query a user's sleep for the first three months of 2022. We make the first request specifying the query parameters for `start` and `end`.

```
GET https://api.prod.whoop.com/developer/v1/activity/sleep?start=2022-01-
01T00:00:00.000Z&end=2022-04-01T00:00:00.000Z
```

The request returns each sleep in the `records` field and a `next_token`, which we can use on the request to get the next page.

```
{
  "records": [
    {
      ...
    }
  ],
  "next_token": "eyIkIjoib0AxIiwibyI6MTB9"
}
```

The `next_token` can be placed on the subsequent request to fetch the next page of Sleeps using the same query parameters.

```
GET https://api.prod.whoop.com/developer/v1/activity/sleep?
nextToken=eyIkIjoib0AxIiwibyI6MTB9&start=2022-01-01T00:00:00.000Z&end=2022-04-
```

```
01T00:00:00.000Z
```

You can continue paginating in this manner to retrieve all pages until the `next_token` is empty.

API Rate Limiting

All clients are subject to two sets of rate limiting for requests. By default, clients may only send:

- 100 requests per minute.
- 10,000 requests per day.

> **RATE LIMIT INCREASES**
>
> Upon request, WHOOP can increase your rate limits. Please submit a request for increased rate limits via this link and be sure to include details surrounding your request.

Rate Limit Information

Information about current rate limits is provided in response headers, following this specification.

You may see response headers that look like the following:

```
X-RateLimit-Limit = "100, 100;window=60, 10000;window=86400"
X-RateLimit-Remaining = "98"
X-RateLimit-Reset = "3"
```

X-RateLimit-Limit

This value expresses the current values for the rate-limiting and over what time window.

The example here is using the default rate limits. The client has not reached any limit yet and is closest to hitting the `minute` window (versus the `day` window). The first number (`100`) states the limit the client is closest to hitting. The time window is in seconds. `100;window=60` describes the first rate limit of 100 requests per minute (60 seconds). `10000;window=86400` represents a rate limit of 10,000 requests over a 24-hour (86,400 seconds) period.

X-RateLimit-Remaining

`X-RateLimit-Remaining` is the number of requests that your app may execute within the time window before WHOOP will reject subsequent requests. In this case, the client is closest to hitting the minute limit of 100, so `X-RateLimit-Remaining: 98` means the client can make 98 more requests in that minute. In other words, the client only made two requests in the current minute.

X-RateLimit-Reset

`X-RateLimit-Reset` is the number of seconds that need to elapse before the `X-RateLimit-Remaining` header resets. In this case, the client is closest to hitting the minute limit of 100 so the `X-RateLimit-Reset` header corresponds to that limit precisely.

Rate Limited Response

After an API key's rate limit has been reached or exceeded, WHOOP's servers will respond with a `429 - Too Many Requests` HTTP status code.

Support

FAQ

How do I get started with the WHOOP Developer Platform?

To get started, you need to sign up for an account on whoop.com and obtain a WHOOP device. Once you have an account, you can create an application by navigating to Dashboard on developer.whoop.com and logging in with your account credentials. You can reference our getting started guide to learn how to integrate WHOOP data into your application.

What data is available through the WHOOP API?

The WHOOP API provides access to various types of data, including activity data, sleep data, recovery data, and more. For a detailed list of available endpoints and the data they return, please refer to our API documentation.

Are there any costs associated with using the WHOOP API?

Access to the WHOOP Developer Platform and API is currently free. However, you must have a WHOOP device, which requires a membership. Check join.whoop.com for more details.

Do you offer a sandbox environment for developers who do not have a WHOOP?

We require all developers on the Developer Platform to have a WHOOP device. Using WHOOP and understanding the user flow dramatically helps with understanding and integrating with our API.

What are the future improvement plans for the WHOOP API?

We strive to continuously improve our API offering and make it straightforward to work with. We will regularly update the API as new functionality and features are released, and note the update in our changelog. For ideas on enhancing the API, please share them with us.

What is your rate limiting policy?

Check out our rate limiting policy here.

What is a rough estimate of how much a day's worth of data for an individual would be in terms of KB?

A day's worth of member data (with one workout, one sleep, and one recovery) would be about 4 KB. For each additional workout or nap, add 1 KB.

How often should we refresh tokens in the OAuth flow?

You should refresh tokens every hour. See an example flow here and documentation for refreshing an access token here.

Do I, as the developer, need to show that my app has collected a member's authorization in my UI?

The WHOOP member should always be able to see that they've granted you access to their data. You should offer an easy and intuitive way for the member to disable the integration as well. The member can also revoke access to an integration via the WHOOP app.

Generally, what is the lift to use the API?

The WHOOP API utilizes OAuth 2.0 authentication, which is well-documented and industry standard, and webhooks to notify when new data is available. Integrating with the WHOOP API is very straightforward. Typically, the hardest part of integrating with the WHOOP API is understanding the data model (*e.g., what is a physiological cycle, and how does that differ from a calendar day?*). We've heard from countless developers that wearing a WHOOP makes understanding the API and data model significantly easier.

Does the API offer access to continuous heart rate data?

Each WHOOP device can broadcast heart rate over Bluetooth Low Energy (BLE) to other devices. Many applications have implemented a Bluetooth listener to let members connect their WHOOP as a heart rate monitor. Continuous heart rate data is not available via the WHOOP API.

Submitting feedback

If you are encountering issues with or have feedback about the WHOOP API, you can submit a request. Please note that responses to this form may take some time.

Still have questions? Share your questions with us.

Cycle

The human body undergoes a biological rhythm of asleep, awake, asleep, awake. We often think of one repetition within a calendar day, such as Tuesday, April 19, 2022. However, calendar days and sleep patterns don't always align. Some people work when other people are sleeping. Some people don't go to sleep for at least 24 hours.

We always reference a member's activity in the context of a Physiological Cycle (known as Cycle for short) rather than calendar days. When you request a member's latest Cycle, the member's current Cycle will only have a `Start Time`. Cycles in the past have both a start and end time.

Data Model

`id` required	integer <int64> Unique identifier for the physiological cycle
`user_id` required	integer <int64> The WHOOP User for the physiological cycle
`created_at` required	string <date-time> The time the cycle was recorded in WHOOP
`updated_at` required	string <date-time> The time the cycle was last updated in WHOOP
`start` required	string <date-time> Start time bound of the cycle
`end`	string <date-time> End time bound of the cycle. If not present, the user is currently in this cycle
`timezone_offset` required	string The user's timezone offset at the time the cycle was recorded. Follows format for Time Zone Designator (TZD) - '+hh:mm', '-hh:mm', or 'Z'.

Learn more about the Time Zone Designator from the W3C Standard

---

`score_state` `required`

string

Enum: `"SCORED"` `"PENDING_SCORE"` `"UNSCORABLE"`

`SCORED` means the cycle was scored and the measurement values will be present. `PENDING_SCORE` means WHOOP is currently evaluating the cycle. `UNSCORABLE` means this activity could not be scored for some reason - commonly because there is not enough user metric data for the time range.

`score`

object (CycleScore)

WHOOP's measurements and evaluation of the cycle. Only present if the score state is `SCORED`

Copy Expand all Collapse all


```
{
    "id": 93845,
    "user_id": 10129,
    "created_at": "2022-04-24T11:25:44.774Z",
    "updated_at": "2022-04-24T14:25:44.774Z",
    "start": "2022-04-24T02:25:44.774Z",
    "end": "2022-04-24T10:25:44.774Z",
    "timezone_offset": "-05:00",
    "score_state": "SCORED",
  - "score": {
        "strain": 5.2951527,
        "kilojoule": 8288.297,
        "average_heart_rate": 68,
        "max_heart_rate": 141
    }
}
```

Recovery

WHOOP Recovery is a daily measure of how prepared your body is to perform. When you wake up in the morning, WHOOP calculates a Recovery score as a percentage between 0 - 100%. The higher the score, the more primed your body is to take on Strain that day.

In addition to the WHOOP Recovery score, the Recovery object has objective measurements that factored into the score such as the resting heart rate (RHR), heart rate variability (HRV), and for 4.0 members, blood oxygen (SpO2) and skin temperature.

Data Model

`cycle_id` required	integer <int64> The Recovery represents how recovered the user is for this physiological cycle
`sleep_id` required	integer <int64> ID of the Sleep associated with the Recovery
`user_id` required	integer <int64> The WHOOP User for the recovery
`created_at` required	string <date-time> The time the recovery was recorded in WHOOP
`updated_at` required	string <date-time> The time the recovery was last updated in WHOOP
`score_state` required	string Enum: `"SCORED"` `"PENDING_SCORE"` `"UNSCORABLE"` `SCORED` means the recovery was scored and the measurement values will be present. `PENDING_SCORE` means WHOOP is currently evaluating the cycle. `UNSCORABLE` means this activity could not be scored for some reason - commonly because there is not enough user metric data for the time range.
`score`	object (RecoveryScore)

WHOOP's measurements and evaluation of the recovery. Only present if the Recovery State is `SCORED`

Copy Expand all Collapse all


```
{
    "cycle_id": 93845,
    "sleep_id": 10235,
    "user_id": 10129,
    "created_at": "2022-04-24T11:25:44.774Z",
    "updated_at": "2022-04-24T14:25:44.774Z",
    "score_state": "SCORED",
  - "score": {
        "user_calibrating": false,
        "recovery_score": 44,
        "resting_heart_rate": 64,
        "hrv_rmssd_milli": 31.813562,
        "spo2_percentage": 95.6875,
        "skin_temp_celsius": 33.7
    }
}
```

Sleep

WHOOP tracks a member's sleep performance. You can view details and measurements about sleep.

Note: In addition to the sleep activity that starts a Cycle, a member can also take *naps* throughout the Cycle. WHOOP sets the field `nap` to true on a Sleep that is a Nap. Naps reduce the amount of sleep needed by a member for the next Cycle.

Data Model

`id` required	integer <int64> Unique identifier for the sleep activity
`user_id` required	integer <int64> The WHOOP User who performed the sleep activity
`created_at` required	string <date-time> The time the sleep activity was recorded in WHOOP
`updated_at` required	string <date-time> The time the sleep activity was last updated in WHOOP
`start` required	string <date-time> Start time bound of the sleep
`end` required	string <date-time> End time bound of the sleep
`timezone_offset` required	string The user's timezone offset at the time the sleep was recorded. Follows format for Time Zone Designator (TZD) - '+hh:mm', '-hh:mm', or 'Z'. Learn more about the Time Zone Designator from the W3C Standard
`nap` required	boolean If true, this sleep activity was a nap for the user

`score_state` `required`	string Enum: `"SCORED"` `"PENDING_SCORE"` `"UNSCORABLE"` `SCORED` means the sleep activity was scored and the measurement values will be present. `PENDING_SCORE` means WHOOP is currently evaluating the sleep activity. `UNSCORABLE` means this activity could not be scored for some reason - commonly because there is not enough user metric data for the time range.
`score`	object (SleepScore) WHOOP's measurements and evaluation of the sleep activity. Only present if the Sleep State is `SCORED`

Copy Expand all Collapse all


```
{
    "id": 93845,
    "user_id": 10129,
    "created_at": "2022-04-24T11:25:44.774Z",
    "updated_at": "2022-04-24T14:25:44.774Z",
    "start": "2022-04-24T02:25:44.774Z",
    "end": "2022-04-24T10:25:44.774Z",
    "timezone_offset": "-05:00",
    "nap": false,
    "score_state": "SCORED",
  - "score": {
     + "stage_summary": { … },
     + "sleep_needed": { … },
       "respiratory_rate": 16.11328125,
       "sleep_performance_percentage": 98,
       "sleep_consistency_percentage": 90,
       "sleep_efficiency_percentage": 91.69533848
    }
}
```

User

Basic Profile

Profile information about the user, such as their email and name.

Data Model

`user_id` required	integer <int64> The WHOOP User
`email` required	string User's Email
`first_name` required	string User's First Name
`last_name` required	string User's Last Name

Copy

```
{
    "user_id": 10129,
    "email": "jsmith123@whoop.com",
    "first_name": "John",
    "last_name": "Smith"
}
```

Body Measurements

Body measurements about the user, such as their weight and height.

Data Model

`height_meter` required	number <float> User's height in meters
`weight_kilogram` required	number <float> User's weight in kilograms
`max_heart_rate` required	integer <int32> The max heart rate WHOOP calculated for the user WHOOP Locker: Understanding Max Heart Rate and Why It Matters for Training

Copy


```
{
    "height_meter": 1.8288,
    "weight_kilogram": 90.7185,
    "max_heart_rate": 200
}
```

Workout

WHOOP keeps track of what type of activity you performed (e.g., Running, Cycling), Strain score, and other physiological measurements such as duration in Heart Rate Zones.

Data Model

`id` required	integer <int64> Unique identifier for the workout activity
`user_id` required	integer <int64> The WHOOP User who performed the workout
`created_at` required	string <date-time> The time the workout activity was recorded in WHOOP
`updated_at` required	string <date-time> The time the workout activity was last updated in WHOOP
`start` required	string <date-time> Start time bound of the workout
`end` required	string <date-time> End time bound of the workout
`timezone_offset` required	string The user's timezone offset at the time the workout was recorded. Follows format for Time Zone Designator (TZD) - '+hh:mm', '-hh:mm', or 'Z'. Learn more about the Time Zone Designator from the W3C Standard
`sport_id` required	integer <int32> ID of the WHOOP Sport performed during the workout
`score_state` required	string Enum: `"SCORED"` `"PENDING_SCORE"` `"UNSCORABLE"`

	`SCORED` means the workout activity was scored and the measurement values will be present. `PENDING_SCORE` means WHOOP is currently evaluating the workout activity. `UNSCORABLE` means this activity could not be scored for some reason - commonly because there is not enough user metric data for the time range.
score	object (WorkoutScore) WHOOP's measurements and evaluation of the workout activity. Only present if the Workout State is `SCORED`

Copy Expand all Collapse all

```
{
    "id": 1043,
    "user_id": 9012,
    "created_at": "2022-04-24T11:25:44.774Z",
    "updated_at": "2022-04-24T14:25:44.774Z",
    "start": "2022-04-24T02:25:44.774Z",
    "end": "2022-04-24T10:25:44.774Z",
    "timezone_offset": "-05:00",
    "sport_id": 1,
    "score_state": "SCORED",
  - "score": {
        "strain": 8.2463,
        "average_heart_rate": 123,
        "max_heart_rate": 146,
        "kilojoule": 1569.34033203125,
        "percent_recorded": 100,
        "distance_meter": 1772.77035916,
        "altitude_gain_meter": 46.64384460449,
        "altitude_change_meter": -0.781372010707855,
      + "zone_duration": { … }
    }
}
```


WHOOP Sports

Sport ID	Sport
-1	Activity
0	Running
1	Cycling
16	Baseball
17	Basketball

Sport ID	Sport
18	Rowing
19	Fencing
20	Field Hockey
21	Football
22	Golf
24	Ice Hockey
25	Lacrosse
27	Rugby
28	Sailing
29	Skiing
30	Soccer
31	Softball
32	Squash
33	Swimming
34	Tennis
35	Track & Field
36	Volleyball
37	Water Polo
38	Wrestling
39	Boxing

Sport ID	Sport
42	Dance
43	Pilates
44	Yoga
45	Weightlifting
47	Cross Country Skiing
48	Functional Fitness
49	Duathlon
51	Gymnastics
52	Hiking/Rucking
53	Horseback Riding
55	Kayaking
56	Martial Arts
57	Mountain Biking
59	Powerlifting
60	Rock Climbing
61	Paddleboarding
62	Triathlon
63	Walking
64	Surfing
65	Elliptical

Sport ID	Sport
66	Stairmaster
70	Meditation
71	Other
73	Diving
74	Operations - Tactical
75	Operations - Medical
76	Operations - Flying
77	Operations - Water
82	Ultimate
83	Climber
84	Jumping Rope
85	Australian Football
86	Skateboarding
87	Coaching
88	Ice Bath
89	Commuting
90	Gaming
91	Snowboarding
92	Motocross
93	Caddying

Sport ID	Sport
94	Obstacle Course Racing
95	Motor Racing
96	HIIT
97	Spin
98	Jiu Jitsu
99	Manual Labor
100	Cricket
101	Pickleball
102	Inline Skating
103	Box Fitness
104	Spikeball
105	Wheelchair Pushing
106	Paddle Tennis
107	Barre
108	Stage Performance
109	High Stress Work
110	Parkour
111	Gaelic Football
112	Hurling/Camogie
113	Circus Arts

Sport ID	Sport
121	Massage Therapy
123	Strength Trainer
125	Watching Sports
126	Assault Bike
127	Kickboxing
128	Stretching
230	Table Tennis
231	Badminton
232	Netball
233	Sauna
234	Disc Golf
235	Yard Work
236	Air Compression
237	Percussive Massage
238	Paintball
239	Ice Skating
240	Handball
248	F45 Training
249	Padel
250	Barry's

Sport ID	Sport
251	Dedicated Parenting
252	Stroller Walking
253	Stroller Jogging
254	Toddlerwearing
255	Babywearing
258	Barre3
259	Hot Yoga
261	Stadium Steps
262	Polo
263	Musical Performance
264	Kite Boarding
266	Dog Walking
267	Water Skiing
268	Wakeboarding
269	Cooking
270	Cleaning
272	Public Speaking

Webhooks

WHOOP webhooks allow you to get notified when an event takes place for users that have authenticated with your app. By subscribing to webhooks, you will be alerted when a user's data was updated rather than needing to constantly make API requests to check for updates.

Setting Up Webhooks

You will first need to create an HTTPS URL endpoint to receive webhooks. This endpoint should be capable of accepting POST requests with a body consisting of the webhook data format and be able to perform webhook signature validation.

With your webhook endpoint set up, you can create or update an app to start accepting webhooks at that URL. Simply add your HTTPS URL to the **Webhook URL** section at the bottom of the page and save the app.

That's it! The next time a WHOOP user that has authenticated with your app triggers one of the supported webhook events, you will receive a webhook alerting you of the change.

Webhook Specifications

Webhooks are sent over HTTPS as a POST request to your configured URL. The body of the webhook request includes the following fields:

`user_id`	int64 The WHOOP User for the event
`id`	int64 Identifier of the object that triggered this webhook
`type`	string Enum: `"workout.updated"` `"workout.deleted"` `"sleep.updated"` `"sleep.deleted"` `"recovery.updated"` `"recovery.deleted"` The type of event that triggered this webhook
`trace_id`	string

Trace ID for the event that triggered this webhook

Copy

```
{
    "user_id": 10129,
    "id": 10235,
    "type": "workout.updated",
    "trace_id": "d3709ee7-104e-4f70-a928-2932964b017b"
}
```

Webhook Event Types

There are several events that are published as webhooks:

Event Type	ID Type	Explanation
`recovery.updated`	The id of the cycle for the recovery	Occurs when a recovery is created or updated.
`recovery.deleted`	The id of the cycle for the recovery	Occurs when a recovery is deleted. Note: a recovery is deleted when its associated sleep is deleted.
`workout.updated`	The id of the workout	Occurs when a workout is created or updated.
`workout.deleted`	The id of the workout	Occurs when a workout is deleted.
`sleep.updated`	The id of the sleep	Occurs when a sleep is created or updated.
`sleep.deleted`	The id of the sleep	Occurs when a sleep is deleted.

NOTE

> All webhook event types will be sent to the Webhook URL configured against your app. As such, if you do not want to process a certain type of event, you can simply respond with a `2XX` and skip any other processing you would normally do.

Webhooks Security

In order to validate that the webhooks you are receiving are originating from WHOOP, you will want to implement signature validation. This can be done by making use of two of the headers sent with each webhook request:

- `X-WHOOP-Signature` - the actual signature
- `X-WHOOP-Signature-Timestamp` - the milliseconds since epoch timestamp used to verify the signature

To verify the signature, first prepend the timestamp header value to the raw http request body. Then, generate a SHA256 HMAC signature of that string using the secret key for your app, which can be found in the WHOOP Developer Dashboard. Finally, base64 encode the result, and compare it to the signature header. If they do not match then the request is invalid and should be dropped.

See below for example signature validation pseudocode:

```
calculated_signature_string = base64Encode(HMACSHA256(timestamp_header +
raw_http_request_body, client_secret))
```

Example Request Flow

To illustrate a webhook use case, check out the below example request flow that utilizes webhooks.

1. WHOOP user `456` goes through the OAuth consent flow with your app, allowing you to read their sleep data.
2. WHOOP user `456` records a sleep. WHOOP assigns id `1234` to this sleep.
3. WHOOP makes a post request to your webhook endpoint with the body of:

```
{
  "user_id": 456,
  "id": 1234,
  "type": "sleep.updated",
  "trace_id": "e369c784-5100-49e8-8098-75d35c47b31b"
}
```

4. Based on seeing `sleep.updated` in the `type` field, your app makes a GET request to the `/activity/sleep/1234` endpoint using the access token for WHOOP user `456` in order to retrieve the sleep data.

As you begin implementing webhooks, you should have logic for interpreting each of these webhook event types. As shown in the example above, when you receive a `sleep.updated` webhook for user `456`, you may want to make a request to the `v1/activity/sleep/{sleepId}` endpoint using the sleepId of `1234` and the token for user of `456` if you want to get the data of the sleep the webhook was sent for.

Webhooks Testing

To generate a webhook, first authenticate with your app to ensure your data is retrievable via the API. Then, open the WHOOP app and do one of the following:

1. Log an activity in the past (e.g. navigate to yesterday and create a 5 minute cycling activity) – once this activity is processed, your webhook endpoint will receive a `workout.updated` webhook.
2. Edit a previous sleep by changing the start or end time by 1 minute – once the sleep is processed, your webhook will receive both a `sleep.updated` webhook and a `recovery.updated` webhook.

You can then edit or delete the workout you created, and you can revert the sleep duration change. Each of these actions will also result in webhooks being sent – `workout.updated`, `workout.deleted`, and `sleep.updated`, respectively.

Delivery & Retries

WHOOP will retry webhook delivery for failed webhook requests **five times over the course of about one hour**. A webhook delivery is considered failed if it receives any response other than a successful `2XX`, or if it receives no response before timing out.

In order to reduce the amount of delivery failures you encounter, you should consider processing the requests asynchronously after returning a successful response, e.g. with a queue worker.

Best Practices

While working with webhooks, it is important to follow these best practices in order to ensure that you can get the most out of the webhook system.

Respond quickly

As laid out in the Delivery & Retries section, WHOOP will not indefinitely retry if your webhook endpoint is determined to be unavailable. We recommend designing your webhook API to return a successful `2XX` status code within a second and for high availability. If your webhook implementation has other dependencies or needs to do expensive work, one strategy is to have your API enqueue the event on a queue for asynchronous processing.

Validate signatures

Be sure to validate the message signature to ensure webhooks you receive were actually sent by WHOOP. Check out the Webhooks Security section to see more details on how to implement this.

Implement a reconciliation job

Since webhook delivery can fail webhooks should not be the sole source of truth for your application. Therefore, we recommend implementing a reconciliation job to occasionally fetch data from WHOOP.

Limitations

There are a few limitations to be aware of when implementing webhooks:

- It is possible you will receive multiple webhook invocations for the same triggering event. You can make use of the `trace_id` field in the webhook to detect duplicate webhooks.
- These are event based webhooks, meaning they are notifications of changes and not actually the changes themselves. You will need to call the API in order to retrieve the most up-to-date data around this event. There is an example of what this flow would look like in the Example Request Flow section.
- It is possible that a webhook may be missed. You should implement a reconciliation job that can occasionally reach out to the WHOOP API for the data types you care about to fetch data that may have been missed.

Webhooks FAQs

Why are there only update and delete events? Do we get notified of create events?

Yes! When workouts, sleeps, or recoveries are created those events are published as an "update" event.

Are there webhooks for Day Strain, cycles, or body measurements?

Not at the moment; these data points should be retrieved by calling their respective APIs. All of the current webhook types are listed in the Event Types section.

How can I stop receiving webhooks for users who have disabled my integration?

It is best practice to revoke access tokens for users who have disabled your integration. Once you do revoke their access token, no webhooks will be sent for the user.

WHOOP Developer Platform

WHOOP is on a mission to unlock human performance. We want companies and developers that share our mission to join us on the journey.

The goal of the WHOOP Developer Platform is to empower developers and companies to create robust apps and integrations with WHOOP. We believe empowering WHOOP members to share their insights and data with other apps can help them achieve their personal goals - whether that is sleeping better, reaching a new fitness goal, or living a healthier lifestyle.

The WHOOP Developer Platform documentation includes the information you need to design, develop, launch, and support your app.

WHOOP 101

WHOOP 101 is where you learn about the core concepts and science behind WHOOP, including Recovery, Sleep, Strain, and Workouts.

> **ARE YOU NEW TO WHOOP?**
>
> Start with WHOOP 101. We believe developing a successful app requires understanding the core concepts and science behind WHOOP.

Developing Your App

Check out our guide to help you get started with developing and launching your first app. You will learn how to register your app, make requests to WHOOP on behalf of a member, what data is available, and considerations before launching to your users.

- Learn about what data is available to you are available.
- Learn how to create an app and authenticate with WHOOP.
- Review design and brand guidelines.

Tutorials

Tutorials provide sample code and guides for common patterns such as authenticating with WHOOP or requesting a member's current Recovery Score.

API Reference Docs

API Reference Docs include what API endpoints are available and how to make requests to them including the URL, required parameters, and response body.

Tutorials

This section walks you through common scenarios you might face while developing with the WHOOP API.

- Authentication (Javascript)
  - Authenticating with WHOOP (with Passport)
  - Refreshing Access Tokens
- Authentication (Postman)
  - Authenticating with WHOOP
  - Refreshing Access Tokens
- Request Patterns
  - Get Current Recovery Score for User

Authenticating with WHOOP (with Passport)

Though Javascript is a primary language for client-side application development, all requests to WHOOP must be made server-side. This tutorial does not propose or require a given framework for building Javascript server-side applications. Some examples of Javascript server-side frameworks are express.js and next.js.

You can choose from one of the many OAuth 2.0 libraries. In this tutorial, we will use Passport for authentication. Specifically, this will use the passport-oauth2 package.

Install Dependencies

```
npm install passport passport-oauth2
```

Additionally, if you're using Typescript, you'll want to include the type definitions as well:

```
npm install @types/passport @types/passport-oauth2
```

Configure The Strategy

We first need to define the data `Passport` will need to complete the OAuth 2.0 authorization flow, stored in a configuration.

```
const whoopOAuthConfig = {
    authorizationURL: `${process.env.WHOOP_API_HOSTNAME}/oauth/oauth2/auth`,
    tokenURL: `${process.env.WHOOP_API_HOSTNAME}/oauth/oauth2/token`,
    clientID: process.env.CLIENT_ID,
    clientSecret: process.env.CLIENT_SECRET,
    callbackURL: process.env.CALLBACK_URL,
    state: true,
    scope: [
        'offline',
        'read:profile'
    ],
}
```

The data elements are:

- `authorizationURL`: The WHOOP URL prompts a user to sign in to WHOOP and authorize your application. Learn more
- `tokenURL`: The WHOOP URL to exchange an authorization code for an access token. Learn more
- `clientID`: A unique identifier for your client. Learn more
- `clientSecret`: A secret value that accompanies your client identifier. Learn more
- `callbackURL`: WHOOP will redirect the user to this location after the user authorizes your application. Learn more
- `state`: Set this to `true` to have the strategy define the state value that WHOOP will pass back to your app. Learn more
- `scope`: A list of the data types and operations your app can access. We are requesting the `offline` scope to retrieve a refresh token and `read:profile` to access User Profile information from the WHOOP API in this example. You need to add the relevant scopes your app needs access. Learn more

Determine What To Do After Authentication Succeeds

`Passport` needs a function to execute once the OAuth flow is complete. What occurs in that function is specific to your application.

In this example, we will use the received information from the OAuth flow to save the user to our database using Prisma. But, of course, what your application does with this may differ.

```
const getUser = async (
    accessToken,
    refreshToken,
    {expires_in},
    profile,
    done,
) => {
    const {first_name, last_name, user_id} = profile

    const createUserParams = {
        accessToken,
        expiresAt: Date.now() + expires_in * 1000,
        firstName: first_name,
        lastName: last_name,
        refreshToken,
        userId: user_id,
    }
```

```
    const user = await prisma.user.upsert({
        where: {userId: user_id},
        create: createUserParams,
        update: createUserParams,
    })

    done(null, user)
}
```

Passport will pass in several parameters that we can use in our application-specific logic:

- `accessToken`: the access token to retrieve user information from the WHOOP API. For future requests, your app must pass the Access Token as a Bearer token in the Authorization header. We will use this as an example below to retrieve profile data.
- `refreshToken`: Your app can use the Refresh Token to retrieve a new access token after expiration. Learn more
- `results`: this is an object that contains fields for `access_token`, `expires_in`, `scope`, and `token_type`.
- `profile`: the normalized representation of a user's profile data. The Passport library will populate the `profile` because we later tell Passport how to get user profile information.
- `done`: a callback function accepting errors and the user as the parameters.

Given all that information, we are extracting the user's first name, last name, and WHOOP user id from their profile and storing that in our database, along with their active tokens.

Note that Passport's documentation shows a function that takes four parameters. Using this won't have the information about when the token expires, so we're instead, we're relying on a less-common variant for this data.

Get User Profile Information

WHOOP includes an API endpoint to access user profile details, which your app will have access to if the app requests (and the user authorizes) the `read:profile` scope.

```
const fetchProfile = async (
    accessToken,
    done,
) => {
    const profileResponse = await fetch(
        `${process.env.WHOOP_API_HOSTNAME}/developer/v1/user/profile/basic`,
        {
            headers: {
                Authorization: `Bearer ${accessToken}`,
            },
```

```
        },
    )

    const profile = await profileResponse.json()

    done(null, profile)
}
```

This async function uses Javascript's Fetch API to make a request to the WHOOP profile endpoint, passing in the access token we receive from a successful OAuth 2.0 flow as a bearer token in the Authorization header.

The function returns the user's profile information. Passport normalizes received profile information.

Use The WHOOP Strategy

Now that we have all of the individual pieces built, it's time to put them together.

```
const whoopAuthorizationStrategy = new OAuth2Strategy(whoopOAuthConfig, getUser)
whoopAuthorizationStrategy.userProfile = fetchProfile

passport.use('withWhoop', whoopAuthorizationStrategy)
```

Here we are building a strategy, passing it our WHOOP configuration and telling it what to do when a user is authenticated - which is to save or get the user from our database.

We're also telling the strategy to use our ability to fetch profile information.

Lastly, we're telling Passport itself to use our newly-created strategy.

Authenticate With WHOOP

We need to authenticate requests using `passport.authenticate()`. Where this gets invoked will depend on your framework of choice. Let's assume an Express application.

```
app.get('/auth/example',
    passport.authenticate('withWhoop'));

app.get('/auth/example/callback',
    passport.authenticate('withWhoop', {failureRedirect: '/login'}),
    function (req, res) {
```

```
        res.redirect('/welcome');
    });
```

How this gets invoked in your middleware stack or routing layer will depend on the framework you're using.

Congratulations

After going through this tutorial, you have learned:

- How to build a custom Passport OAuth 2.0 strategy to complete the OAuth flow with WHOOP.
- Where to add your custom logic to handle user management and persistence.
- How to access a user's profile information and tell Passport about it to automatically retrieve user information from WHOOP.
- How to connect those individual pieces with Passports OAuth 2.0 package.

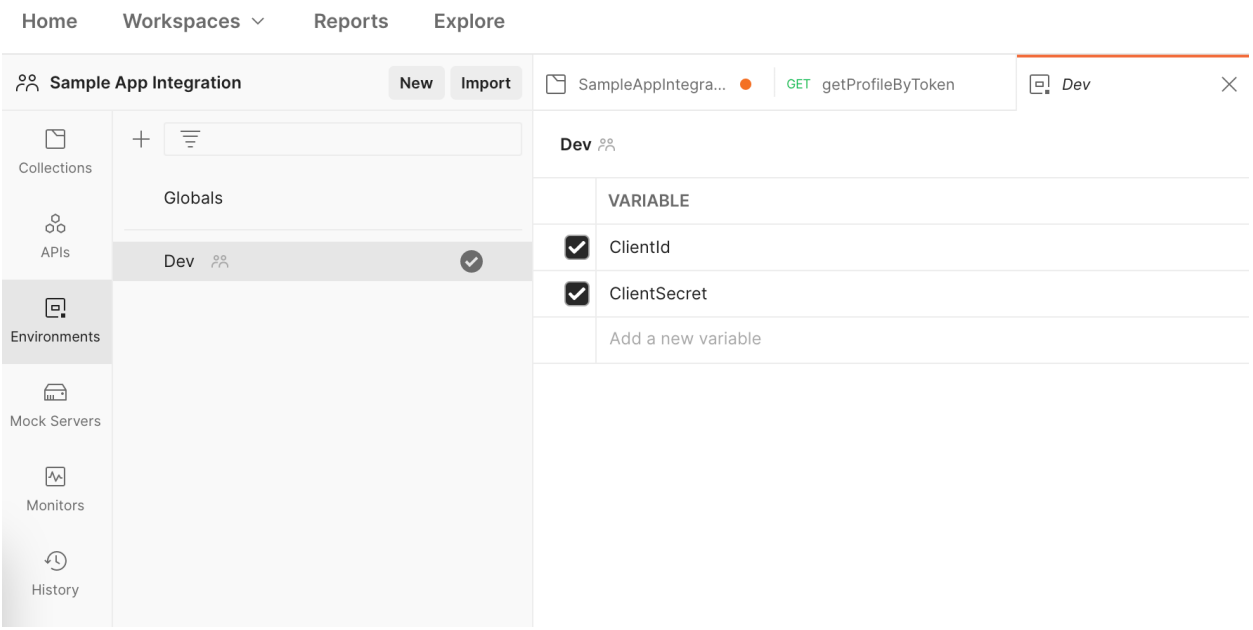
The next step for you is to plug this into whatever framework or middleware you're using to trigger Passport to authenticate.

Authenticating with WHOOP

Postman is an application you can use to make, save, and share API requests. It includes the functionality to complete an OAuth 2.0 flow for user data. Being programming language-agnostic, it may be a great starting point to validate your credentials are functioning.

Environment Setup

The Postman collection used here requires that you set up the Client Secret information from the WHOOP Developer Dashboard, and you saved the data as Postman variables.



In the Postman app, navigate to the "Environments" section on the far left. Then, add or modify two variables in the main content window:

1. `ClientId` - this is the unique client Id from the WHOOP Developer Dashboard.
2. `ClientSecret` - this is the secret for the client Id from the WHOOP Developer Dashboard.

In addition to naming these variables, fill in the values with the credentials from the Developer Dashboard.

Note: *The screenshot does not show that information to avoid sharing credentials.*

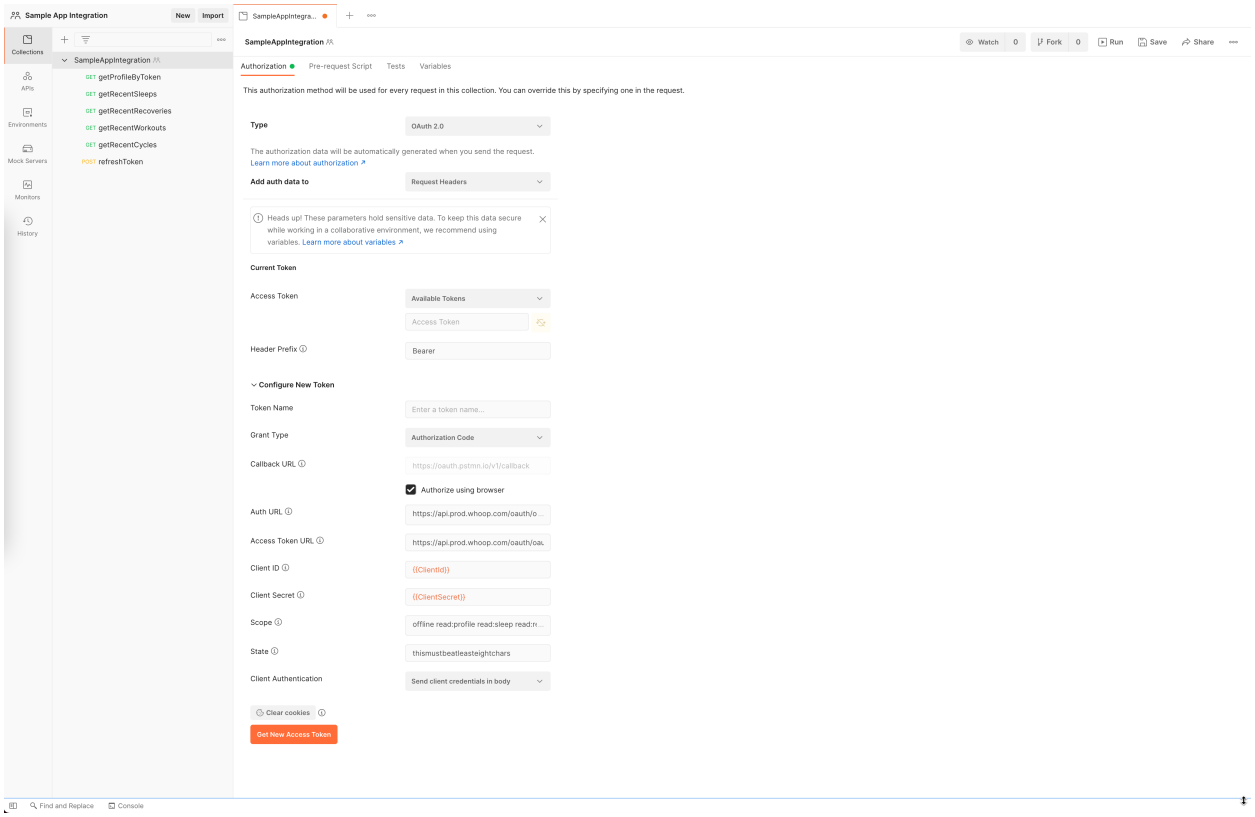
Now you're ready to use those values as variables for the rest of the flow, and don't need to share the actual secrets in the saved requests.

Starting the Authorization Flow

Postman includes support for completing the OAuth 2.0 flow. This section will show you how to give Postman access to your WHOOP credentials for future requests.

Navigate to the "Collections" section on the far left, and select the collection name for the WHOOP API in the pane to the right of that.

Next, select the "Authorization" heading in the main content window. It should be underlined in orange.



Fill in the fields as follows:

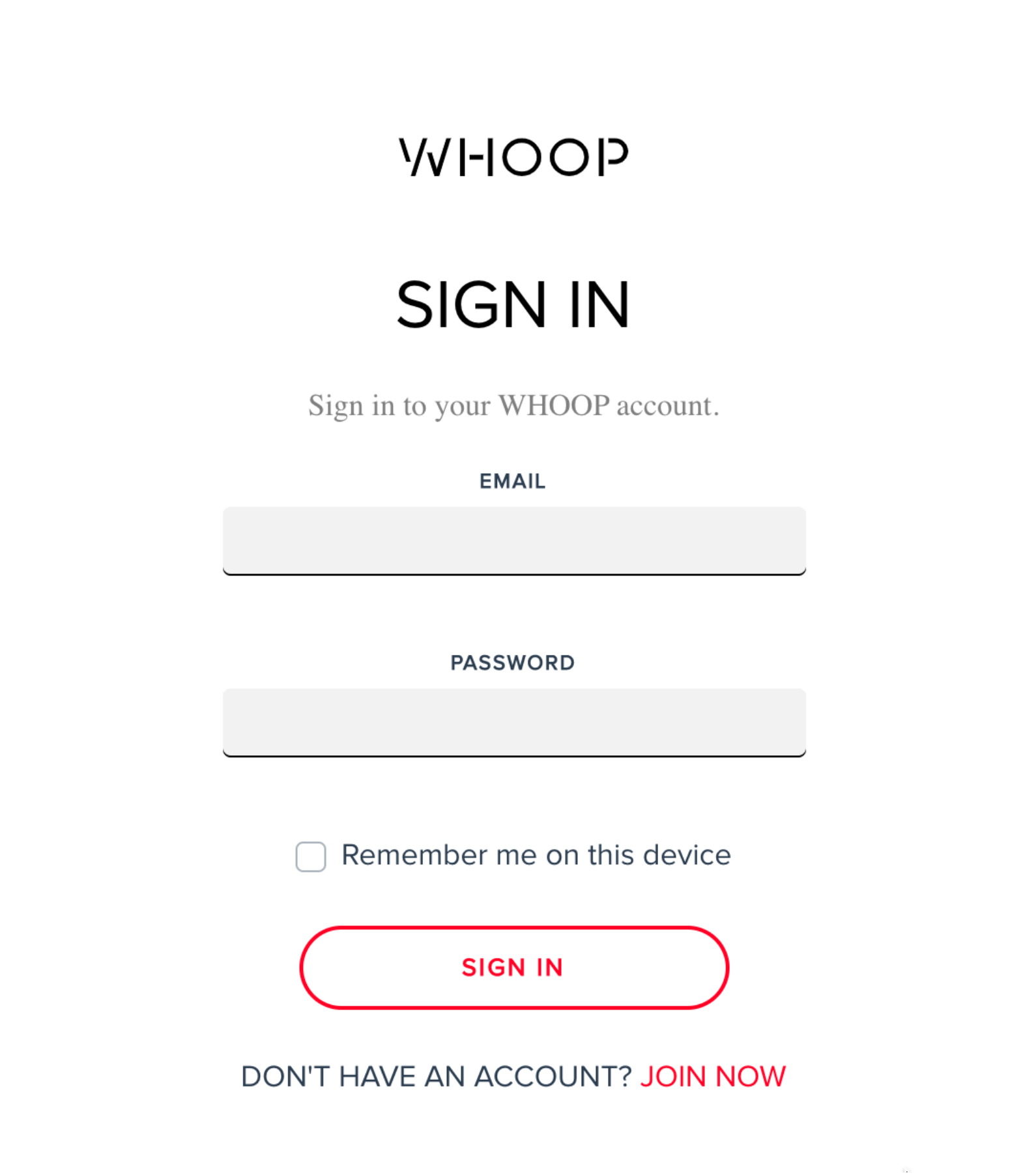
- **Type**: OAuth 2.0
- **Add auth data to**: Request Headers
- **Access Token**: Available tokens
- **Header Prefix**: Bearer
- **Grant Type**: Authorization Code
- **Callback URL**: check "Authorize using browser"
- **Auth URL**: `https://api.prod.whoop.com/oauth/oauth2/auth` - Learn more.
- **Access Token URL**: `https://api.prod.whoop.com/oauth/oauth2/token` - Learn more.
- **Client Id**: `{{ClientId}}` - this is using the variable updated earlier. Learn more.
- **Client Secret**: `{{ClientSecret}}` - this is using the variable updated earlier. Learn more.

- **Scope**: A space-delimited list of scopes to request data access to. Learn more.
- **State**: A string that must be at least eight characters long to be sent to the WHOOP server and used for verification. Learn more.
- **Client Authentication**: Send client credentials in body

Lastly, click the "Get New Access Token" button, and Postman will make the request to WHOOP's Auth URL specified above.

Sign in to WHOOP

Postman should then redirect to a browser window and present WHOOP's sign-in form.



Fill in this information with the account information that Postman will use in future requests to access data.

Click the "SIGN IN" button after providing the account information.

Authorize Access to WHOOP Data

After providing valid WHOOP user account credentials, Postman will redirect you to a WHOOP log-in asking you to authorize your app and displays the requested scopes as part of the OAuth flow. These are all the scopes that were requested in Postman earlier.

To complete the flow, you must grant the application access by clicking the "AUTHORIZE" button.

Depending on your browser, you may receive an alert to redirect you back to the Postman app. Please do so to complete the round-trip.

Manage Access Tokens

After completing the redirect, Postman will next show a modal that displays all of the token information returned from WHOOP.

Once again, this screenshot purposely censors the actual *values* of these tokens to not share personal information.

If you want to use the `refresh_token` in Postman to request a refreshed token, you will need to copy this value at this point and save it somewhere safe. Postman does not retain access to this data.

You will use the Access Token in future requests, and you can save those credentials in Postman.

Use Access Token

Click the "Use Token" button on the right in the Manage Access Tokens modal.

For this to persist on future requests, that change must be saved by clicking the "Save" button on the far right of the collection after leaving the modal.

Congratulations

You have completed the OAuth 2.0 flow using Postman, and you can make additional requests for WHOOP data using this token in Postman.

Get Current Recovery Score

This example will use the Fetch API to make an HTTP request to WHOOP's server.

Prerequisites

- Access Token: The token received after completing the OAuth 2.0 flow. Your app must pass the Access Token as a Bearer token in the `Authorization` header.
- The Access Token is granted permission to `read:cycles` and `read:recovery` OAuth scopes.

Overview

The process of getting a user's current recovery score requires two steps:

- Get the user's current Cycle. (API Docs)
- Get the user's Recovery for the current Cycle if one exists. (API Docs)

These two steps are required because every Cycle is not guaranteed to have a Recovery score. For example, the user may not have worn their WHOOP the previous day. In this scenario, the Recovery score will be missing for the Cycle.

Get the User's Current Cycle

Let's start by getting the current Cycle. The Cycle Collection is sorted by time in descending order, so making a request with `limit=1` will give us the latest Cycle.

```
const accessToken = "__ACCESS_TOKEN_FOR_USER__";

query = new URLSearchParams({
  limit: "1",
});
```

We will append those parameters as query param values to our GET request.

```
const getCurrentCycle = async (accessToken, query) => {
  const uri = `https://api.prod.whoop.com/developer/v1/cycle?${query}`;

  const cycleResponse = await fetch(uri, {
```

```
    headers: {
      Authorization: `Bearer ${accessToken}`,
    },
  });

  if (cycleResponse.status === 200) {
    return cycleResponse.json();
  } else {
    throw new Error(`Received ${cycleResponse.status} status from Whoop`);
  }
};
```

Response

The response will include an array of Cycles, which should have precisely one if the user has at least one Cycle (new users may not have a Cycle yet, so check there is at least one in the array). You will need the `cycle_id` from the latest cycle for the next step.

records	Array of objects (Cycle) [items] The collection of records in this page.
next_token	string A token that can be used on the next request to access the next page of records. If the token is not present, there are no more records in the collection.

Copy Expand all Collapse all


```
{
  - "records": [
      + { … }
    ],
    "next_token": "MTIzOjEyMzEyMw"
}
```

Get Recovery for Cycle

With the latest Cycle, we can now get the Recovery score for the Cycle if one exists. Before using the Recovery score, there are a few things you should consider:

- Not every Cycle has a Recovery - the user may not have worn their strap the previous day.
- New users may be calibrating. New users require calibration before the Recovery score is relevant to them. The calibration period lasts a few days for new users. You can use the field `user_calibrating` to check if the user is still in the calibration phase.
- WHOOP may not be able to score every Recovery - you should check if the `score_state` is `SCORED`. If the `score_state` is `PENDING_SCORE`, you will need to check back later. If the `score_state` is `UNSCORABLE`, a Recovery Score cannot be calculated by WHOOP for this user's Cycle.

```
const getRecoveryForCycle: = async (accessToken, cycleId) => {
    const uri =
`https://api.prod.whoop.com/developer/v1/cycle/${cycleId}/recovery`

    const recoveryResponse = await fetch(uri, {
        headers: {
            Authorization: `Bearer ${accessToken}`,
        },
    })

    if (recoveryResponse.status === 200) {
        return recoveryResponse.json()
    } else if (recoveryResponse.status === 404) {
        return null
    } else {
        throw new Error(`Received ${recoveryResponse.status} status from Whoop`)
    }
}
```

Response

Upon success, the API should return a 200 HTTP status or 404 if a Recovery does not exist for the current cycle. Remember not all Cycles will have a Recovery.

`cycle_id` `required`	integer <int64> The Recovery represents how recovered the user is for this physiological cycle

`sleep_id` required	integer <int64> ID of the Sleep associated with the Recovery
`user_id` required	integer <int64> The WHOOP User for the recovery
`created_at` required	string <date-time> The time the recovery was recorded in WHOOP
`updated_at` required	string <date-time> The time the recovery was last updated in WHOOP
`score_state` required	string Enum: `"SCORED"` `"PENDING_SCORE"` `"UNSCORABLE"` `SCORED` means the recovery was scored and the measurement values will be present. `PENDING_SCORE` means WHOOP is currently evaluating the cycle. `UNSCORABLE` means this activity could not be scored for some reason - commonly because there is not enough user metric data for the time range.
`score`	object (RecoveryScore) WHOOP's measurements and evaluation of the recovery. Only present if the Recovery State is `SCORED`

Copy Expand all Collapse all


```
{
    "cycle_id": 93845,
    "sleep_id": 10235,
    "user_id": 10129,
    "created_at": "2022-04-24T11:25:44.774Z",
    "updated_at": "2022-04-24T14:25:44.774Z",
    "score_state": "SCORED",
```

```
  - "score": {
        "user_calibrating": false,
        "recovery_score": 44,
        "resting_heart_rate": 64,
        "hrv_rmssd_milli": 31.813562,
        "spo2_percentage": 95.6875,
        "skin_temp_celsius": 33.7
    }
}
```


Congratulations

You have now received Recovery data for the current Cycle.

Refreshing Access Tokens

This example will use the Fetch API to make an HTTP request to WHOOP's server.

Prerequisites

- Refresh Token: A refresh token is received along with an access token when completing the initial OAuth 2.0 flow, when the auth request includes the `offline` scope. Learn more
- Client Id: A unique identifier for your client. Learn more
- Client Secret: A secret value that accompanies your client identifier. Learn more
- Refresh Token Endpoint: `https://api.prod.whoop.com/oauth/oauth2/token` . Learn more

Making the Request

We first need to assemble the parameters to provide to WHOOP's server in order to retrieve new access and refresh tokens.

```
const refreshParams = {
    grant_type: 'refresh_token',
    client_id: process.env.CLIENT_ID,
    client_secret: process.env.CLIENT_SECRET,
    scope: 'offline',
    refresh_token: refresh_token,
}
```

These fields represent:

- **grant_type**: `refresh_token`. This grant type explicitly tells an OAuth provider you're asking for a refresh token.
- **client_id**: A unique identifier for your client.
- **client_secret**: A secret value that accompanies your client identifier.
- **scope**: The `offline` scope allows your app the receive a refresh token, along with the new access token.
- **refresh_token**: The value of the refresh token received along with an access token.

Now that we have the parameters to send to the API endpoint, we can construct the entire API call:

```
const getFreshTokens = async (refreshParams) => {
    const body = new URLSearchParams(refreshParams)
    const headers = {
        'Content-Type': 'application/x-www-form-urlencoded',
    }

    const refreshTokenResponse = await fetch(
        `https://api.prod.whoop.com/oauth/oauth2/token`,
        {
            body,
            headers,
            method: 'POST',
        },
    )

    return refreshTokenResponse.json()
}
```

Response Type

The response object received from making the API request will look as follows:

```
interface AuthResult {
    access_token: string
    refresh_token: string
    expires_in: number
    scope: string
    token_type: 'bearer'
}
```

Congratulations

Your app can now use the new access token for subsequent API requests for this user's data. Your app can also use the refresh token to complete this flow once the access token expires.

Refreshing Access Tokens

Postman is an application you can use to make, save, and share API requests. We're going to use it to demonstrate using a refresh token to receive a new access token from WHOOP.

Prerequisites

- Refresh Token: A refresh token is received along with an access token when completing the initial OAuth 2.0 flow, when the auth request includes the `offline` scope. Learn more
- Client Id: A unique identifier for your client. Learn more
- Client Secret: A secret value that accompanies your client identifier. Learn more
- Refresh Token Endpoint: `https://api.prod.whoop.com/oauth/oauth2/token` . Learn more

Making the Request

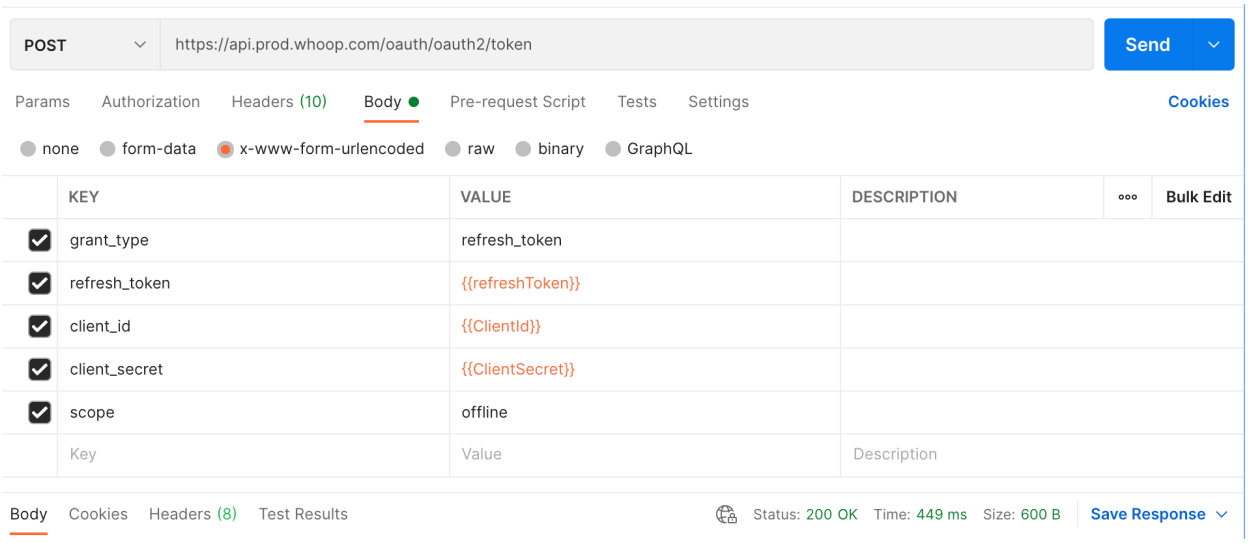
We're going to issue a POST request to the refresh token endpoint to receive a new access token.

Fill in the fields as follows:

- **HTTP Request Type/Verb**: POST
- **URL**: `https://api.prod.whoop.com/oauth/oauth2/token`

In the Body section, select the "x-www-form-urlencoded" radio button. Fill in the following keys and values:

- **grant_type**: `refresh_token`. This grant type explicitly tells an OAuth provider you're asking for a refresh token.
- **refresh_token**: The value of the refresh token received along with an access token.
- **client_id**: A unique identifier for your client.
- **client_secret**: A secret value that accompanies your client identifier.
- **scope**: The `offline` scope allows you to get a new refresh token and an access token.



In this image, Postman variables take the place of the refresh token, client id, and client secret. Postman should prepopulate those variables with the configured values. Alternatively, place the values directly in the value of the POST data rather than using variables at all.

Click "Send" to make your request.

Receiving the Response

Under the Request Body section, a Response Body should be visible as a JSON payload. It will have the following form:

```
{
  "access_token": "the-value-of-the-new-access-token",
  "expires_in": 3600,
  "refresh_token": "the-value-of-the-new-refresh-token",
  "scope": "offline other-scopes-requested",
  "token_type": "bearer"
}
```

Congratulations

You can use the new access token to make additional API requests for this user's data. You can also use the new refresh token to complete this flow once the access token expires.

WHOOP 101

WHOOP is a 24/7 wearable that helps members understand how well their bodies function and how they change over time. Through continuous monitoring of physiological signals, WHOOP helps members understand how well their body performs and recovers. As a developer, you have the opportunity to build integrations that leverage WHOOP insights across three pillars - **Strain, Recovery, and Sleep**.

This guide will provide you with an overview of the core concepts powering WHOOP to get you ready to build your app.

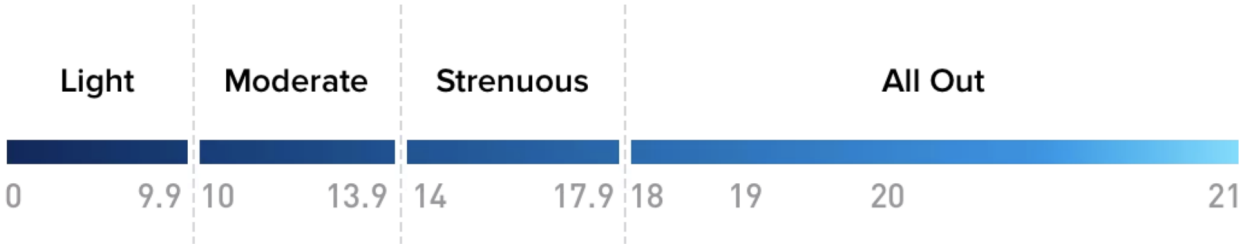
Strain

WHOOP Strain is a measurement of the amount of stress on your body. The Strain score is a number on a 0 to 21 scale, based on the Borg Scale of Perceived Exertion. WHOOP scores Strain continuously for members throughout the day and every workout.

Note: *Strain is not on a linear scale. It takes more stress to move from a score of **16 to 17** than **4 to 5**.*

Your body is constantly changing, and each day is different. If you do the same workout two days in a row, you will likely report different Strain scores based on fluctuations in how your body recovered and changed. Strain is also specific to your baseline. Two people making the same motions will likely generate different Strain scores.

WHOOP categorizes Strain score values into Light, Moderate, High, and All Out.



- **Light Strain (0-9)** - This strain category indicates room for active recovery with minimal stress on the body.
- **Moderate Strain (10-13)** - This category indicates moderate stress on the body, which helps maintain fitness.
- **High Strain (14-17)** - This category indicates increased stress which helps build fitness gains in your training.

- **All Out (18-21)** - This category indicates all-out training or a packed activity day that puts significant stress on the body and may be difficult to recover from the day after.

> **WANT TO LEARN MORE?**
>
> - WHOOP Strain
> - How Does WHOOP Strain Work?
> - Why Doesn't WHOOP Count Steps?
> - Podcast No. 26: Understanding Strain

Workout Activity

WHOOP tracks workouts for you and how much Strain accumulated over the workout. You can create workout activities manually, import them from other sources like Apple Health, or WHOOP may automatically detect and classify workouts .

WHOOP keeps track of what type of activity you performed (e.g., Running, Cycle), Strain score, and other physiological measurements such as duration in Heart Rate Zones.

> **A NOTE ON RECOVERY ACTIVITIES**
>
> We recently separated activities that primarily promote Recovery ("Recovery Activities") from those that primarily promote Strain ("Workout Activities"). As such, Recovery activities (examples below) are not accessible via the Workouts endpoint:
>
> - Ice Bath
> - Massage Therapy
> - Meditation
> - Other - Recovery
> - Stretching
>
> We will add Recovery activities to the API in the near future.

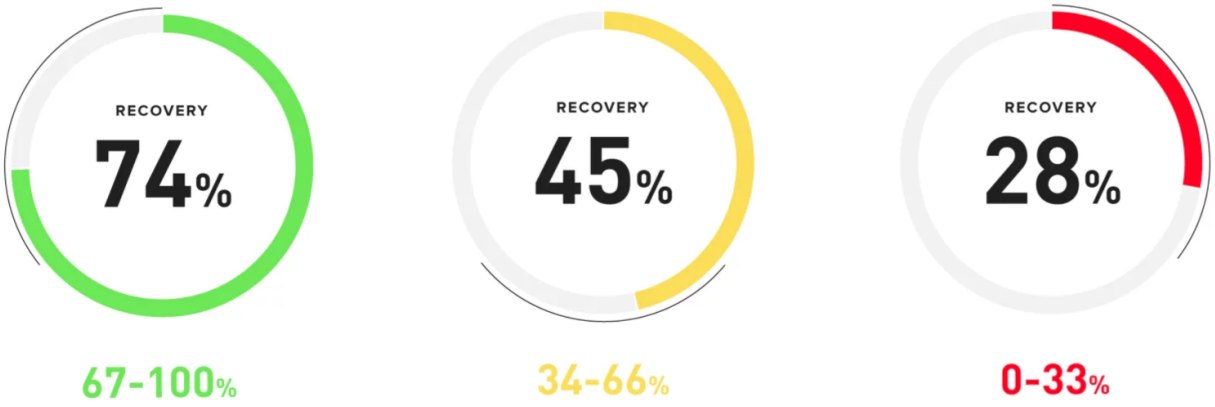
Recovery

WHOOP Recovery is a daily measure of how prepared your body is to perform. When you wake up in the morning, WHOOP calculates a Recovery score as a percentage between 0 - 100%. The higher the score, the more primed your body is to take on Strain that day.

WHOOP calculates Recovery scores using measurements from the previous day and your sleep, such as resting heart rate (RHR), heart rate variability (HRV), respiratory rate, sleep duration/quality, skin

temperature, and blood oxygen. Unlike Strain, a Recovery score does not change over the day (unless you edit your sleep which triggers a Recovery score re-calculation).

WHOOP categorizes Recovery scores into one of three colors:



- **GREEN (67-100%)**: You are well recovered and primed to perform. Whether at home, at work, or in the gym, your body is signaling that it can handle a strenuous day.
- **YELLOW (34-66%)**: Your body is maintaining and ready to take on moderate amounts of strain.
- **RED (0-33%)**: Rest is likely what your body needs. Your body is working hard to recover. Some reasons could include overtraining, sickness, stress, lack of sleep, or other lifestyle factors.

> **WANT TO LEARN MORE?**
>
> - WHOOP Recovery
> - How Does WHOOP Recovery Work?
> - Everything You Need to Know About Heart Rate Variability (HRV)
> - Podcast No. 29: Heart Rate Variability (HRV)

Sleep

WHOOP tracks your sleep, including how long you slept and the stages of your sleep - Light, REM, and Slow Wave Sleep (Deep). WHOOP also calculates how much sleep you need based on your Sleep Debt and your previous day's activity.

> **WANT TO LEARN MORE?**
>
> - WHOOP Sleep
> - How Does WHOOP Measure Sleep, and How Accurate is it?
> - What is Light Sleep? | Deep Sleep? | REM Sleep?
> - What are the differences between Deep and REM Sleep?

Build Better Health & Wellness Experiences with the WHOOP Developer Platform

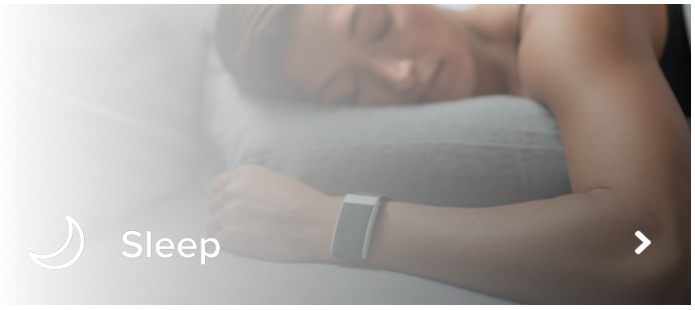
GET STARTED

WHOOP uses advanced wearable technology and precise data collection to empower our members, and now so can you. The WHOOP Developer Platform gives you access to powerful infrastructure to build better health and wellness experiences. Design,

develop, support, and launch your app or integration with WHOOP to begin collecting valuable insights.

Discover WHOOP

WHOOP is a wearable health & fitness coach that tracks your sleep, recovery, and strain metrics with top scientific accuracy. We use this data to offer personalized recommendations on how to reach your goal — whether you want to improve your health, get more sleep, feel more energized, or perform at your best.



Health & Fitness Infrastructure of the Future

OAuth 2.0 authentication, webhooks, and world class documentation bring innovation to your fingertips.

OAuth 2.0
Build high-quality, powerful solutions quickly and reliably utilizing the industry-standard OAuth 2.0 protocol.

Webhooks
Get alerted of events immediately, keeping your integration up to date without constant API polling.

VIEW OUR API DOCS

Join WHOOP

Get started for less than $1/day. Purchase includes membership, 4.0 hardware, onyx SuperKnit band, wearable + waterproof* battery pack, and more.

*WHOOP 4.0 is IP68 dustproof and water-resistant up to 10 meters for 2 hours. WHOOP 4.0 Battery Pack is IP68 dustproof and water-resistant up to 1 meter for 2 hours.

GET STARTED