COROSAPI Reference V2.0.6

*Revision History**

Version	Date	AMD	Participant	Remark
V2.0	2020-06-08	M		Version 2.0 interface supports more workout type data
V2.1	2021-08-26	A		Description of the limit on the number of requests per minute
V2.2	2022-01-07	A		Added the feature of route synchronization
V2.3	2023-06-06	M		Modified Section 6.1: Changed the time limit for syncing workout plans from 3rd party apps from the next 7 days to the next 1 year to only sync 30 workout plans at a time. Support for trail running.
V2.4	2023-10-19	A		Added Section 6.2: Delete the Training Schedule
V2.5	2023-10-25	M		Starting from January 1, 2024, the earliest query date of 3rd-party apps is limited to three months before the current date. For example, if the day is October 30, 2023, then only data after July 30, 2023 can be queried.
V2.6	2024-05-23	M		4.2 Obtaining Workout Records, return "planWorkoutId",5.3 Workout Summary Data Push, push "planWorkoutId"
V2.7	2024-10-29	M		6.1 Receiving the Training Schedule from the third party platform, target type add "EndManually"

(A-add，M-modify，D-delete)

1. INTEGRATION PREPARATION

1.1 Reference Intro

This documentation explains the integration process and technical standards required for COROS API. Test environment domain is opentest.coros.com. Public environment domain is open.coros.com.

1.2 Integration Application

Please complete the application with detailed info to begin the API request reviewing process.

Field	Notes
Company Name	
Company URL	
Application Name	Limit to 50 characters
Application Logo	Png Format with 2 sizes, 144px * 144px and 102px * 102px
Application Description	Limit to 100 characters
Authorized Callback Domain (redirect_uri)	Provide one or two domains starting with http:// or https:// such as https://open.other.com. To ensure security, we will use this to check whether the authorized domain is consistent with your redirect URL.

If you need workout summary data push service from COROS, please provide

the requested information below. Check section 5.3 for details.

Information	Description
Workout data receiving Endpoint URL	Https service interface that accepts workout data and allows duplicate data.
Service Status Check API	To check if partner interface is operating as expected(need a get request,when http status code is 200,will be work)

After the application is approved, COROS will issue an application ID (clientId) and access token (clientSecret). COROS reserves the right to terminate ID and tokens at any time.

1.3 Interface Request

Partner app needs to support HTTP redirect and supports these three status codes: 301, 302, and 303.

2. AUTHORIZATION AGREEMENT

2.1 Authorization Process

COROS open platform is based on OAuth 2.0 protocol. The Authorization process consists of 4 parts.

1. Guide the user to the authorization page to agree to the authorization and obtain the code (valid for 30 minutes, valid once)
2. Use the code to receive an accessToken.
3. The accessToken is valid for 30 day by default. If necessary, the developer can refresh the authorized accessToken to avoid expiration.
4. Use accessToken for available API.
5. All interface call forms use application/x-www-form-urlencoded format.

3. INTERFACE AUTHORIZATION

3.1 User agrees to authorization and receives code

3.1.1 Description

1. Authorization agreed by user

User agrees to the authorization and the page will redirect to redirect_uri/?code=CODE&state=STATE

If the user denies authorization, the code parameter will not be brought after the redirection, only the state parameter will be included redirect_uri?state=STATE

2. Code Description

The code is used in exchange for the accessToken. Each time the user authorizes the code, the code will be different. The code can only be used once, and it will expire automatically if it is not used for 30 minutes.

3.1.2 Request URL Example：

Request method：GET

Example：

```
https://open.coros.com/oauth2/authorize?client_id=f0b4a1659286c1f3555xx&redirect_uri=http://www.coros.com&state=123&response_type=code
```

3.1.3 Request Parameter Description

Parameter	Required?	Type	Length	Description	Notes
client_id	Yes	String	32	client unique identifier	
redirect_uri	Yes	String	200	Redirect url	The redirected callback link address after authorization, which can be followed by parameters. Please use urlencode to process the link, and keep it consistent with the domain name filled in the API application.
state	Yes	String	128	The state parameter will be brought after the redirection. Developers can fill in the parameter value of a-z A-Z 0-9, up to 128 bytes.	We recommend that developers can use this parameter to verify the validity of the request and can also record the position before the user request authorization page. This parameter can be used to prevent cross-site request forgery (CSRF) attacks, please refer to: STATE description
response_type	Yes	String		Authorized return type	Please enter code

3.2 Use code to receive accessToken

3.2.1 Description

1. accessToken is valid for 30 days.

3.2.2 Request URL Example：

Request method:POST

Content-Type：application/x-www-form-urlencoded

Example:

```
curl --location --request POST 'https://open.coros.com/oauth2/accesstoken' \
--data-urlencode 'client_id=e4781d74be2942590210xx' \
--data-urlencode 'redirect_uri=https://www.coros.com' \
--data-urlencode 'code=rg1-17f615f33e54f0bde3bdfxx' \
--data-urlencode 'client_secret=244e293cafcfe2a3bae0b4xx' \
--data-urlencode 'grant_type=authorization_code' \
--data-urlencode 'user_id=123'
```

3.2.3 Request Parameter Description

Parameter	Required?	Type	Length	Description	Notes
client_id	Yes	String	32	client unique identifier	
redirect_uri	Yes	String	200	Redirect url	Please keep it the same as when requesting the code via the previous step
code	Yes	String	64	Access credentials	Code received via the previous step
client_secret	Yes	String	128	Client key	
grant_type	Yes	String		Authorization type	Please enter authorization_code

3.2.4 Return Parameter Description

Parameter	Type	Description	Notes
expiresIn	signed 32 int type	Timeout	Timeout for the accessToken in second.
refreshToken	String	Authorization code	To refresh accessToken
accessToken	String	token	Interface call authorization credentials
openId	String	COROS user unique identifier	

Return result example:

```
{
    "expires_in":2592000,
    "refresh_token":"08a06b7df38d0d2852e5xx5",
    "access_token":"db0214b6006e7570bdxx",
    "openId":"b93ac3b5df6b4db3bexx"
}
```

3.3 Refresh accessToken

3.3.1 Description

After the refresh is successful, the effective time of the current accessToken will be updated (one month added from the current time), and it will be returned in json format. For the parameter description, please refer to the previous step.

RefreshToken nerver expires.

Example：

The validity period of adding a token is 30 days. Partner app will store the token and expiration time locally. When the time is approaching 30-day mark, partner app can call this interface to refresh the token and extend the validity period of the token.

3.3.2 Request URL Example：

Request method: POST

Content-Type： application/x-www-form-urlencoded

Example:

```
curl --location --request POST 'https://open.coros.com/oauth2/refresh-token' \
--data-urlencode 'client_id=55c52a3ba2c54759a8bc55exx' \
--data-urlencode 'refresh_token=rg2-ff531d2a9a64f77a9dcxx' \
--data-urlencode 'client_secret=a7a78f5906a748d7a9d8xx' \
--data-urlencode 'grant_type=refresh_token'
```

3.3.3 Request Parameter Description

Parameter	Required?	Type	Length	Description	Notes
client_id	Yes	String	32	Client unique identifier	
refresh_token	Yes	String	64	Authorization code	To refresh accessToken
client_secret	Yes	String	32	Client key	
grant_type	Yes	String		Authorization type	Please input refresh_token

3.3.4 Return Parameter Description

Parameter	Type	Description	Notes
result	String	Return code	
message	String	Return message	

Return result example:

```
{
    "result": "0000",
    "message": "OK"
}
```

3.4 Deauthorization

3.4.1 Description

After deauthorize, both the refresh token and the token will be set as invalid. If you need to resume use, you must re-authorize.

3.4.2 Request URL Example：

Request method: POST

Example:

`https://open.coros.com/oauth2/deauthorize?token=xxxxxxxx`

3.4.3 Request Parameter Description

Parameter	Required?	Type	Length	Description	Notes
token	Yes	String	32	accessToken	Parameters are placed in the header field of the http request

3.4.4 Return Parameter Description

Parameter	Type	Description	Notes
result	String	Return code	
message	String	Return message	

Return result example:

```
{
    "result": "0000",
    "message": "OK"
}
```

4. DATA INTERFACE

4.1 Obtaining User Info

4.1.1 Description

4.1.2 Request URL Example：

Request method: GET

Example:

```
https://open.coros.com/coros/userinfosim?token=token&openId=openId
```

4.1.3 Request Parameter Description

Parameter	Required?	Type	Length	Description	Notes
token	Yes	String	64	accessToken	
openId	Yes	String	32	COROS user unique identifier	

4.1.4 Return Parameter Description

Parameter	Type	Description	Notes
result	String	Return code	
message	String	Return message	
data	Return results		
- nick	String	Nickname	COROS user nickname
- openId	String	User unique identifier	
- profilePhoto	String	User profile photo	
- runCalorie	String	Total calories for running	Unit: kcal
- runDistance	String	Total distance for running	Unit: meter
- runTotalTime	String	Total time for running	Unit: second
- isGray	Boolean	whether the user is on the whitelist	true: the user is on the whitelist false: the user is not on the whitelist

Return result example:


```
{
    "data": {
        "nick": "infi测试1",
        "openId": "48d1aec7ef564937bf40d5bec57a3ffe",
        "profilePhoto": "https://test-coros.oss-cn-
qingdao.aliyuncs.com/avatar/201801/1516788666Z3XEGODBG38RD74TCK3B",
        "runCalorie": "296",
        "runDistance": "33602",
        "runTotalTime": "9096"
        "isGray": true
    },
    "message": "OK",
    "result": "0000"
}
```


4.2 Obtaining Workout Records (by specific date range)

4.2.1 Description

1. Interface description: Inquire the workout record within the specified date range. The maximum date range for one query is 30 days, and the query date is not earlier than three months before the day (for example, if the day is October 30, 2023, you can only query the data after July 30, 2023)

4.2.2 Request URL Example

Request method: GET

Example:

`https://open.coros.com/v2/coros/sport/list?token=2f03c4fefa75474097dd4d31e189364e&openId=OPENID&startDate=20170101&endDate=20170110`

4.2.3 Request Parameter Description

Parameter	Required?	Type	Length	Description	Notes
token	Yes	String	64	accessToken	
openId	Yes	String	32	COROS User Unique Identifier	
startDate	Yes	signed 32 int type	32	Workout start date for record query	Start date (the first day of the month), int type, such as 20170101
endDate	Yes	signed 32 int type	32	Workout end date for record query	End date (the first day of the month), int type, such as 20170101

4.2.4 Return Parameter Description

Parameter	Type	Description	Notes
result	String	Return code	
message	String	Return message	
data	Return result		
- planWorkoutId	Signed 64 int	WorkoutId for this activity (Same as the ID submitted in Chapter 6.1)	
- labelId	String	Workout ID	
- mode	signed 32 int type	Parent workout type	Refer to workout type details.
- subMode	signed 32 int type	Child workout type	Refer to workout type details.
- deviceName	String	Device name	
- distance	Floating point	Distance	Unit: meter
- calorie	Floating point	Calories	Unit: calorie
- avgSpeed	signed 32 int type	Average pace	Unit: second/km
- avgFrequency	signed 32 int type	Average cadence	Unit: step/min
- step	signed 32 int type	Total steps	
- startTime	signed 32 int type	Workout start timestamp	Count to second

Parameter	Type	Description	Notes
- endTime	signed 32 int type	Workout end timestamp	Count to second
- startTimezone	signed 32 int type	Start time time zone	15-minute time zone system, 32 means UTC+08:00
- endTimezone	signed 32 int type	End time time zone	15-minute time zone system, 32 means UTC+08:00
- fitUrl	String	.fit workout data download URL	Multisport .fit file download is included in the child workout type list
- triathlonItemList	List	Triathlon data	
- - mode	signed 32 int type	Parent workout type for triathlon	Refer to workout type details.
- - subMode	signed 32 int type	Child workout type for triathlon	Refer to workout type details.
- - distance	Floating point	Triathlon parent workout distance	Unit: meter
- - calorie	Floating point	Triathlon parent workout calories	Unit: calorie
- - duration	signed 32 int type	Triathlon parent workout time	Unit: second
- - step	signed 32 int type	steps	

Workout type details

Parent Workout Type	Child Workout Type	Workout Type Details
8	1	Outdoor Run
8	2	Indoor Run
9	1	Outdoor Bike
9	2	Indoor Bike
9	3	E-Bike
9	4	Mountain Bike
9	5	E-Mountain Bike
9	6	Gravel Bike
10	1	Open Water
10	2	Pool Swim
13	1	Triathlon
13	2	Multisport
13	3	Ski Touring
13	4	Outdoor Climb
14	1	Mountain Climb
15	1	Trail Run
16	1	Hike
18	1	GPS Cardio
18	2	Gym Cardio
19	1	XC Ski
20	1	Track Run
21	1	Ski
21	2	Snowboard
22	1	Pilot
23	2	Strength

Parent Workout Type	Child Workout Type	Workout Type Details
24	1	Rowing
24	2	Indoor Rower
25	1	Whitewater
26	1	Flatwater
27	1	Windsurfing
28	1	Speedsurfing
29	1	Ski Touring
31	1	Walk
33	2	Single-Pitch
33	3	Bouldering
34	2	Jump Rope
98	1	custom sport -outdoor
99	2	custom sport -indoor

Return result example

```
{
    "data":[
        {
            "mode":8,
            "avgFrequency":8,
            "avgSpeed":163,
            "calorie":9553,
            "deviceName":"COROS PACE",
            "distance":3014,
            "duration":491,
            "endTime":1516097362,
            "labelId":"406974289395351552",
            "subMode":1,
            "startTime":1516096869,
            "step":52,
            "startTimezone":32,
            "endTimezone":32,

"fitUrl":"https://oss.coros.com/fit/407419767966679040/418173292602490880.fit"
        },
        {
            "mode":13,
            "avgSpeed":414,
            "calorie":34058,
            "deviceName":"COROS PACE",
            "distance":3620,
            "duration":1497,
            "endTime":1516195402,
            "labelId":"407000611102425088",
            "startTime":1516193903,
            "startTimezone":32,
            "subMode":1,
            "endTimezone":32,
            "triathlonItemList":[
                {
                    "mode":8,
                    "subMode":1,
                    "calorie":1000,
                    "distance":1000,
                    "duration":500,

"fitUrl":"https://oss.coros.com/fit/407419767966679040/418173292602490881.fit"
                },
                {
                    "mode":9,
                    "subMode":1,
                    "calorie":1000,
                    "distance":1000,
                    "duration":500,
```

```
"fitUrl":"https://oss.coros.com/fit/407419767966679040/418173292602490882.fit"
                },
                {
                    "mode":10,
                    "subMode":1,
                    "calorie":1000,
                    "distance":1000,
                    "duration":500,
                    "step":200,

"fitUrl":"https://oss.coros.com/fit/407419767966679040/418173292602490883.fit"
                }
            ]
        }
    ],
    "message":"OK",
    "result":"0000"
}
```


4.3 Obtaining Daily Data(by specific date range)

4.3.1 Description

Interface description: Inquire the daily data(sleep,resting heart rate etc.) within the specified date range. The maximum date range for one query is 30 days, and the query date is not earlier than three months before the day (for example, if the day is October 30, 2023, you can only query the data after July 30, 2023)

4.3.2 Request URL Example：

Request method: GET

Example:

`https://open.coros.com/coros/daily/query?token=2f03c4fefa75474097dd4d31e189364e&openId=OPENID&startDate=20170101&endDate=20170110`

4.3.3 Request Parameter Description

Parameter	Required?	Type	Length	Description	Notes
token	Yes	String	64	accessToken	
openId	Yes	String	32	COROS user unique identifier	
startDate	Yes	signed 32 int type	32	Workout start date for record query	Start date (the first day of the month), int type, such as 20170101
endDate	Yes	signed 32 int type	32	Workout end date for record query	End date (the first day of the month), int type, such as 20170101

4.3.4 Return Parameter Description

Parameter	Type	Description	Notes
result	String	Return code	
message	String	Return message	
data			
- dailyList	List		
- - happenDay	32 int type	date for record query	yyyyMMdd
-- sleepStartTime	String	Sleep data	yyyy-MM-ddHH:mm:ss
- - sleepEndTime	String	Sleep data	yyyy-MM-ddHH:mm:ss
-- calorie	float	Total calories	Unit: calorie
-- step	32 int type	Total steps	
-- rhr	32 int type	Resting heart rate	If the user does not have a resting heart rate, this data is not returned
-- hrvList	List	HRV data list	
--- hrv	32 int type	HRV value	
--- timestamp	32 int type	HRV timestamp	
--- hr	32 int type	Mean heart rate	
-- ppgHrv	32 int type	Overnight HRV	
-- sleepAvgHr	32 int type	Mean sleep heart rate	

Return result example:

```
{
    "data":{
        "dailyList":[
            {
                "happenDay":20200615,
                "sleepStartTime":"2020-06-14 22:00:01",
                "sleepEndTime":"2020-06-15 08:00:01",
                "calorie":9553,
                "step":52,
                "rhr":56,
                "hrvList":[
                    {
                        "hrv":25,
                        "hr":60,
                        "timestamp":1592098222
                    },
                    {
                        "hrv":30,
                        "timestamp":1592101822
                    }
                ],
                  "ppgHrv": 50,
                  "sleepAvgHr": 70
            },
            {
                "happenDay":20200616,
                "sleepStartTime":"2020-06-15 22:00:01",
                "sleepEndTime":"2020-06-18 08:00:01",
                "calorie":9553,
                "step":52,
                "rhr":56,
                "hrvList":[
                    {
                        "hrv":21,
                        "hr":68,
                        "timestamp":1592247630
                    }
                ],
                  "ppgHrv": 51,
                  "sleepAvgHr": 76
            }
        ]
    },
    "message":"OK",
    "result":"0000"
}
```


5. WORKOUT DATA PUSH

5.1 Description

1. COROS pushes the workout summary data to partners. Partner platform then obtains detailed workout data according to the workout ID.
2. COROS supports detailed workout file in .fit format and detailed workout data in json format.
3. COROS prefers partners to support detailed workout file in .fit format. .fit format is easier for future data expansion than json format.

5.2 Information Needed From Partners

1. Information needed from partners:
    1. Https service interface that accepts workout data
    2. After receiving the workout data, partner verifies the client and secret of the push request header information before accepting workout data.
    3. Partner platform needs to be compatible with duplicate workout data pushed by the COROS service. COROS may push the same workout data again if push timeout occurs since COROS can't verify if partner has received the data.

Information	Description
Workout data receiving Endpoint URL	Https service interface that accepts workout data and allows duplicate data.
Service Status Check API	To check if partner interface is operating as expected(need a get request,when http status code is 200,will be work)

2. Push Process:
    1. COROS server checks newly added and previously failed workout data of each user that links with partners every 5 minutes (frequency can be adjusted).
    2. Pushes the workout summary data to partners and requests https interface service provided by partners to complete the data push.
    3. COROS receives the return result from partner platform. If the push fails, it will retry push for two more times.
    4. If successful, mark the workout data push status as successful.
    5. If still fails after retry, mark the push status as failure.

6. The failed workout data will be pushed to the partner along with newly added workout data during the next push. If push fails for more than 24 hours, the affected data will no longer be pushed. Partner will be notified to check the service status.

5.3 Workout Summary Data Push

5.3.1 Description

1. COROS pushes workout summary data to partner platform.
2. COROS includes the client and secret info in the push request (https) header field.
3. After receiving the workout data, partner verifies the client and secret of the push request header information before accepting workout data.

5.3.2 Information needed from partners

Information	Notes
Workout data receiving URL	POST service interface that uses https
client	Third-party identification for verifying valid requests provided by COROS.
secret	Third-party key for verifying valid requests provided by COROS.

5.3.3 Interface Call Description

Call Description

http request method: POST

Input Parameter Description

Parameter	Required?	Type	Length	Description
client	Yes	Yes	Character	Third-party identification, placed in the request header field, used for request verification
secret	Yes	Yes	Character	Third-party key, placed in the request header field, used for request verification
sportDataList	Yes	Yes	List	Workout summary data
- planWorkoutId	No	Signed 64 int	WorkoutId for this activity (Same as the ID submitted in Chapter 6.1)	
- openId	Yes	String	COROS user unique identifier	
- labelId	Yes	String	Workout ID	
- mode	Yes	32 int type	Workout type	Refer to workout type details.
- subMode	Yes	32 int type	Detailed workout type	Refer to workout type details.
- deviceName	No	String	Device name	
- distance	Yes	Floating point	Distance	Unit: meter
- calorie	Yes	Floating point	Calories	Unit: calorie
- avgSpeed	Yes	32 int type	Avg pace	Unit: second/km
- avgFrequency	Yes	32 int type	Avg cadence	Unit: step/min

Parameter	Required?	Type	Length	Description
- step	No	32 int type	Total steps	
- startTime	Yes	32 int type	Workout start timestamp	Count to second
- endTime	Yes	32 int type	Workout end timestamp	Count second
- startTimezone	Yes	32 int type	Start time time zone	15-minute time zone system, 32 means UTC+08:00
- endTimezone	Yes	32 int type	End time time zone	15-minute time zone system, 32 means UTC+08:00
- fitUrl	Yes	String	.fit workout data download URL	Triathlon .fit file download is included in the Child workout type list
- triathlonItemList	No	List	Triathlon data	
- - mode	Yes	32 int type	Triathlon parent workout type	Refer to workout type details
- - subMode	Yes	32 int type	Triathlon child workout type	Refer to workout type details
- - distance	Yes	Floating point	Triathlon parent workout distance	Unit: meter
- - calorie	Yes	Floating point	Triathlon parent workout calories	Unit: calorie
- - duration	Yes	32 int type	Triathlon parent workout time	Unit: second
- - step	No	32 int type	Triathlon parent workout steps	

Parameter	Required?	Type	Length	Description
- - fitUrl	Yes	String	Triathlon parent workout .fit download URL	

Workout type details

Parent Workout Type	Child Workout Type	Workout Type Details
8	1	Outdoor Run
8	2	Indoor Run
9	1	Outdoor Bike
9	2	Indoor Bike
9	3	E-Bike
9	4	Mountain Bike
9	5	E-Mountain Bike
9	6	Gravel Bike
10	1	Open Water
10	2	Pool Swim
13	1	Triathlon
13	2	Multisport
13	3	Ski Touring
13	4	Outdoor Climb
14	1	Mountain Climb
15	1	Trail Run
16	1	Hike
18	1	GPS Cardio
18	2	Gym Cardio
19	1	XC Ski
20	1	Track Run
21	1	Ski
21	2	Snowboard
22	1	Pilot
23	2	Strength

Parent Workout Type	Child Workout Type	Workout Type Details
24	1	Rowing
24	2	Indoor Rower
25	1	Whitewater
26	1	Flatwater
27	1	Windsurfing
28	1	Speedsurfing
29	1	Ski Touring
31	1	Walk
33	2	Single-Pitch
33	3	Bouldering
34	2	Jump Rope
98	1	custom sport -outdoor
99	2	custom sport -indoor

Input parameter format:

1. Client and secret parameter transfer by https header field
2. See json parameter below (included in the body for transfer)

```
{
    "sportDataList":[
        {
            "avgFrequency":0,
            "avgSpeed":0,
            "calorie":220000,
            "deviceName":"COROS PACE",
            "distance":22000,
            "duration":2200,
            "endTime":1532576178,
            "endTimezone":32,
            "labelId":"418173292602490880",
                       "openId": "42dbb958c5a146f29ce9f89e05e5195a",
            "mode":13,
            "subMode":1,
            "startTime":1522576178,
            "startTimezone":32,
            "step":2200,
            "triathlonItemList":[
                {
                    "calorie":117738,
                    "distance":748.07,
                    "duration":901,

"fitUrl":"https://oss.coros.com/fit/407419767966679040/418173292602490880_0.fit",
                    "mode":10,
                    "subMode":1
                },
                {
                    "calorie":391220,
                    "distance":19911.3,
                    "duration":2761,

"fitUrl":"https://oss.coros.com/fit/407419767966679040/418173292602490880_1.fit",
                    "mode":9,
                    "subMode":1
                },
                {
                    "calorie":248338,
                    "distance":5142.43,
                    "duration":1592,

"fitUrl":"https://oss.coros.com/fit/407419767966679040/418173292602490880_2.fit",
                    "mode":8,
                    "subMode":1
                }
            ]
        },
        {
            "avgFrequency":0,
```

```
            "avgSpeed":368,
            "calorie":30000,
            "deviceName":"COROS PACE",
            "distance":40000,
            "duration":300,
            "endTime":1532576159,
            "endTimezone":32,

"fitUrl":"https://oss.coros.com/fit/407419767966679040/418173315956375553.fit",
            "labelId":"418173315956375553",
                        "openId": "42dbb958c5a146f29ce9f89e05e5195a",
            "mode":8,
            "startTime":1522576159,
            "startTimezone":32,
            "step":300,
            "subMode":1
        },
        {
            "avgFrequency":0,
            "avgSpeed":0,
            "calorie":20000,
            "deviceName":"COROS PACE",
            "distance":20000,
            "duration":200,
            "endTime":1532576158,
            "endTimezone":32,

"fitUrl":"https://oss.coros.com/fit/407419767966679040/418173315956375552.fit",
            "labelId":"418173315956375552",
                        "openId": "42dbb958c5a146f29ce9f89e05e5195a",
            "mode":9,
            "startTime":1522576158,
            "startTimezone":32,
            "step":200,
            "subMode":1
        },
        {
            "avgFrequency":199,
            "avgSpeed":368,
            "calorie":10000,
            "deviceName":"COROS PACE",
            "distance":10000,
            "duration":100,
            "endTime":1532576157,
            "endTimezone":32,

"fitUrl":"https://oss.coros.com/fit/407419767966679040/418173292602490881.fit",
            "labelId":"418173292602490881",
            "mode":8,
            "startTime":1522576157,
            "startTimezone":32,
```

```
            "step":30166,
            "subMode":2
        }
    ]
}
```

5.3.4 3rd Party Return Result Description

Parameter	Type	Description	Notes
result	String	Result code	0000 - success All other codes indicate failure.It will retry push.
message	String	Message Description	Message provides different results by failure type

See return result for a proper call:

```
{
    "message":"ok",
    "result":"0000"
}
```

5.4 Obtaining Detailed Workout Data In .fit Format

5.4.1 Description

1. Partner platform uses the workout data ID from the workout summary data to request detailed workout data.
2. COROS returns detailed workout data .fit format file download URL.

5.4.2 Instructions

1. Use this interface to inquire about the fitUrl info if it is missing from the workout data which is accessed from the interface listed in chapter 4.2.
2. Use this interface to inquire about the fitUrl info if it is missing from the workout data which is pushed from COROS listed in chapter 5.3.

5.4.3 Request URL Example

Request Method: GET

Example:

```
https://open.coros.com /v2/coros/sport/detail/fit?
mode=8&subMode=1&labelId=labelId&openId=openId&token=token
```

5.4.4 Request Parameter Description

Parameter	Required?	Type	Length	Description	Parameter
token	Yes	String	64	accessToken	Parameters are placed in the url
openId	Yes	String	32	COROS User Unique Identifier	Parameters are placed in the url
labelId	Yes	Signed 64 int type		Workout ID	Parameters are placed in the url
mode	Yes	signed 32 int type		Parent Workout Type	Parameters are placed in the url
subMode	Yes	signed 32 int type		Child Workout Type	Refer to workout type details

Workout type details

Parent Workout Type	Child Workout Type	Workout Type Details
8	1	Outdoor Run
8	2	Indoor Run
9	1	Outdoor Bike
9	2	Indoor Bike
9	3	E-Bike
9	4	Mountain Bike
9	5	E-Mountain Bike
9	6	Gravel Bike
10	1	Open Water
10	2	Pool Swim
13	1	Triathlon
13	2	Multisport
13	3	Ski Touring
13	4	Outdoor Climb
14	1	Mountain Climb
15	1	Trail Run
16	1	Hike
18	1	GPS Cardio
18	2	Gym Cardio
19	1	XC Ski
20	1	Track Run
21	1	Ski
21	2	Snowboard
22	1	Pilot
23	2	Strength

Parent Workout Type	Child Workout Type	Workout Type Details
24	1	Rowing
24	2	Indoor Rower
25	1	Whitewater
26	1	Flatwater
27	1	Windsurfing
28	1	Speedsurfing
29	1	Ski Touring
31	1	Walk
33	2	Single-Pitch
33	3	Bouldering
34	2	Jump Rope
35	2	Stair Climb
98	1	custom sport -outdoor
99	2	custom sport -indoor

5.4.5 Return Parameter Description

Parameter	Type	Description	Notes
result	String	Return code	
message	String	Return message	
data	data		
- labelId	Signed 64 int type	Workout ID	
- mode	32 int type	Parent Workout type	
- subMode	32 int type	Child workout type	
- deviceName	String	Device name	
- distance	Floating point	Distance	Unit: meter
- calorie	Floating point	Calories	Unit: calorie
- avgSpeed	32 int type	Avg pace	Unit: second/km
- avgFrequency	32 int type	Avg cadence	Unit: step/min
- step	32 int type	Total steps	
- startTime	32 int type	Workout start timestamp	Count to second
- endTime	32 int type	Workout end timestamp	Count second
- startTimezone	32 int type	Start time time zone	15-minute time zone system, 32 means UTC+08:00
- endTimezone	32 int type	End time time zone	15-minute time zone system, 32 means UTC+08:00

Parameter	Type	Description	Notes
- fitUrl	String	.fit workout data download URL	Triathlon .fit file download is included in the child workout type list
- triathlonItemList	List	Triathlon data	
- - mode	32 int type	Triathlon parent workout type	Refer to workout type details
- - subMode	32 int type	Triathlon child workout type	Refer to workout type details
- - distance	Floating point	Triathlon parent workout distance	Unit: meter
- - calorie	Floating point	Triathlon parent workout calories	Unit: calorie
- - duration	32 int type	Triathlon parent workout time	Unit: second
- - step	32 int type	Triathlon parent workout steps	Only available in running
- - fitUrl	String	Triathlon parent workout .fit download URL	Only available in running

Return result from a proper call – non-triathlon:

```
{
    "data":{
        "avgFrequency":199,
        "avgSpeed":368,
        "calorie":310000,
        "deviceName":"COROS PACE",
        "distance":31000,
        "duration":3100,
        "endTime":1546394402,
        "endTimezone":32,

"fitUrl":"https://oss.coros.com/fit/407419767966679040/418408855184113664.fit",
        "labelId":"418408855184113664",
        "mode":8,
        "startTime":1546394401,
        "startTimezone":32,
        "step":30166,
        "subMode":2
    },
    "message":"OK",
    "result":"0000"
}
```


Return result from a proper call – triathlon:

```
{
    "data":{
        "avgFrequency":0,
        "avgSpeed":0,
        "calorie":80000,
        "deviceName":"COROS PACE",
        "distance":100000,
        "duration":800,
        "endTime":1532576164,
        "endTimezone":32,
        "labelId":"418407322954530816",
        "mode":13,
        "subMode":1,
        "startTime":1522576164,
        "startTimezone":32,
        "step":800,
        "triathlonItemList":[
            {
                "calorie":117738,
                "distance":748.07,
                "duration":901,

"fitUrl":"https://oss.coros.com/fit/407419767966679040/407419767966679040_0.fit",
                "mode":10,
                "step":0,
                "subMode":1
            },
            {
                "calorie":391220,
                "distance":19911.29,
                "duration":2761,

"fitUrl":"https://oss.coros.com/fit/407419767966679040/407419767966679040_1.fit",
                "mode":9,
                "step":0,
                "subMode":1
            },
            {
                "calorie":248338,
                "distance":5142.43,
                "duration":1592,

"fitUrl":"https://oss.coros.com/fit/407419767966679040/407419767966679040_2.fit",
                "mode":8,
                "step":4627,
                "subMode":1
            }
        ]
    },
    "message":"OK",
```

```
    "result":"0000"
}
```

Possible return result when in error:

```
{
  "result":"5016",
  "message":"No data found"
}
```

6. Training Plan's API

6.1 Receiving the Training Schedule from the third party platform

Partner Platform

6.1.1 Description

The partner platform pushes to COROS the training schedule created by the authorized user whose account has been linked to the COROS platform. COROS receives the training schedule of the user covering the next 1 year including today. After this, it will return the data—StartDate, EndDate—to its partner platform and inform it that the data within the date range [StartDate, EndDate] has been successfully received.

6.1.2 Example of request URL

Request method: POST

Example: `https://open.coros.com/coros/tp/list/push`

```
curl --location --request POST 'https://open.coros.com/coros/tp/list/push' \
--data-urlencode 'token=rg1-6b4cca6d8a8171e77f123d**********' \
--data-urlencode 'openId=25b5ab7357aa4c7791a022**********' \
--data-urlencode 'data={"AthleteId":"132265","StartDate":"2023-06-16","EndDate":"2024-
06-09","Workouts":[{"Description":"Description","LastModifiedDate":"2022-07-
26T07:26:33","Title":"10KW09-
2E10+5ST","TotalTime":"2660","Id":"100019","WorkoutDay":"2023-06-
16","WorkoutType":"run","StartTime":"2023-06-16T06:00:00","Structure":
[{"IntensityClass":"Active","Name":"name1","Ftp":236,"ThresholdHr":170,"ThresholdSpeed
":2.5,"Length":{"Unit":"Meter","Value":1200},"IntensityTarget":
{"Unit":"PercentOfThresholdSpeed","MinValue":90,"MaxValue":100},"Description":"","Type
":"Step"}]}]}'
```

6.1.3 Request parameters

Description of the request body

Parameter name	Must?	Type	Unit	Description
token	Yes	String	64	Proof of authorization: accessToken
openId	Yes	String	32	The unique identifier of the COROS platform's user
data	Yes			
- AthleteId	Yes	Int		The user id at the partner platform
- StartDate	Yes	Date	Date, eg: 2020-09-20	**The first date of the training plan. The date is not earlier than today.**
- EndDate	Yes	Date	Date, eg: 2020-09-20	**The last day of the training plan. The time span between the End Date and Start Date is at most 365 days. **
- **Workouts**	Yes	**array**	**Refer to the following description of the Workout's structure.**	**Training plan set.The size of set cannot exceed 30 at a time**

Description of the Workout's structure

Parameter name	Must?	Type	Unit	Description
Description	No	string		Description
LastModifiedDate	Yes	datetime		The time when the last edit was made such as "2020-09-14T21:05:00.307"
Title	Yes	string		Title
TotalTime	No	float	Unit: second	Total duration
Id	Yes	int		The unique id of the training schedule
WorkoutDay	Yes	datetime	Date	The date of the workout such as"2020-09-10"
WorkoutType	Yes	string		Workout type: Swim, Bike, Run, Strength, trailRun
StartTime	No	datetime	local timezone	Start time, such as"2020-09-16T00:00:00"
Structure	Yes	**array**	**Refer to the following description of Structure**	**Training movement set**

Description of the Structure (Example of json)

```
{
    "Structure": [
        {
            "IntensityClass": "WarmUp", //movement type
            "Name": "Warm up", //movement name
            "Description": "Description",  //movement description or brief intro
            "Ftp": 0, //power threshold, a value must be assigned when IntensityTarget
is set as PercentOfFtp
            "ThresholdHr": 0, //heart rate threshold, a value must be assigned when
IntensityTarget is set as PercentOfThresholdHr
            "ThresholdSpeed": 0.83, //pace and speed threshold, X meters/sec, a value
must be assigned when IntensityTarget is set as PercentOfThresholdSpeed
            "Length": {
                "Unit": "Second",
                "Value": 600
            },
            "Rest": {
                "Unit": "Second",
                "Value": 600
            },
            "Type": "Step",
            "IntensityTarget": {
                "Unit": "PercentOfThresholdHr",
                "MinValue": 70,
                "MaxValue": 80
            }
        },
        {
            "IntensityClass": "Active",
            "Name": "Active",
            "Length": {
                "Unit": "Second",
                "Value": 2095
            },
            "Type": "Step",
            "IntensityTarget": {
                "Unit": "PercentOfThresholdHr",
                "MinValue": 80,
                "MaxValue": 100
            }
        },
        {
            "Type": "Repetition",
            "Length": {
                "Unit": "Repetition",
                "Value": 3
            },
            "Steps": [
                {
                    "IntensityClass": "Active",
```

```
                    "Name": "Hard",
                    "Length": {
                        "Unit": "Second",
                        "Value": 60
                    },
                    "Type": "Step",
                    "IntensityTarget": {
                        "Unit": "PercentOfThresholdHr",
                        "Value": 115
                    }
                },
                {
                    "IntensityClass": "Rest",
                    "Name": "Easy",
                    "Length": {
                        "Unit": "Second",
                        "Value": 60
                    },
                    "Type": "Step",
                    "IntensityTarget": {
                        "Unit": "ValueOfEquipmentWeight",
                        "Value": 70
                    }
                }
            ]
        },
        {
            "IntensityClass": "CoolDown",
            "Name": "Cool Down",
            "Length": {
                "Unit": "Second",
                "Value": 600
            },
            "Type": "Step",
            "IntensityTarget": {
                "Unit": "ValueOfEquipmentWeight",
                "Value": 75
            }
        }
    ]
}
```


Notes about the **Structure's format: json**

1. Enumeration values:
    - **IntensityClass:**
        - WarmUp

        - CoolDown
        - Active
        - Rest
    - **Length/Unit: target type**
        - Meter: distance, unit: meter, WorkoutType in ["swim", "bike", "run", "trailRun"]
        - Second: time, unit: second, WorkoutType in ["swim", "bike", "run", "strength", "trailRun"]
        - Reps: number of times, unit: number of times, WorkoutType in ["strength"]
        - EndManually: manually end
    - **Rest/Unit: reset settings** (Value assignment is not a must. When it is absent, it means that there is no rest)
        - Second: time, unit: second, WorkoutType in ["swim", "bike", "run", "strength", "trailRun"]
        - Bpm: heart rate. When the heart rate recovers to below xx bpm, the rest stops. WorkoutType in ["run", "trailRun"]
        - EndManually: manually end the rest. WorkoutType in ["swim", "bike", "run", "strength", "trailRun"]. When the rest type is “End Manually”, it is alright that the value is not assigned.
    - **Type:**
        - Repetition: supports the repetition of a group of multiple movements
        - Step: a single movement
    - **IntensityTarget/Unit** (the absence of value assignment indicates that the Intensity Target has not been set)
        - PercentOfFtp, WorkoutType in ["bike", "run", "trailRun"], supports a fixed value and interval
        - PercentOfThresholdHr, WorkoutType in ["bike", "run", "trailRun"], supports a fixed value and interval
        - PercentOfThresholdSpeed, WorkoutType in ["bike", "run", "trailRun"], supports a fixed value and interval
        - RangeOfFtp, WorkoutType in ["bike", "run", "trailRun"], supports an interval
        - RangeOfCandence, WorkoutType in ["run", "trailRun"], supports an interval

        - RangeOfThresholdHr, WorkoutType in ["bike", "run", "trailRun"], supports an interval
        - RangeOfThresholdSpeed, WorkoutType in ["bike", "run", "trailRun"], supports an interval
        - ValueOfEquipmentWeight, WorkoutType in ["strength"], only supports a fixed value
        - ValueOfWeight (the user's weight), WorkoutType in ["strength"], only supports a fixed value
        - ValueOfStroke. When IntensityTarget/Unit=ValueOfStroke, the Value represents the specific type of the swimming stroke. 0=undefined, 1=free style, 2=breaststroke, 3=backstroke, 4=butterfly stroke, 255=mixed type

2. Units

    - In Steps, the units of the Length are: meter, second, number of times
    - The unit of heart rate is number of times/minute
    - The unit of cadence is step/minute
    - The unit of power is watt
    - The unit of speed is meter/second
    - The unit of weight is KG

3. Notes

    - Length: an integer
    - Steps: represents a combination of multiple sets of movements
    - Percentages: an integer
    - IntensityTarget: can be a fixed value or an interval

When the target intensity needs to be described as an interval, then it needs to be set as:

```
"IntensityTarget": {
  "Unit": "RangeOfThresholdSpeed",
  "MinValue":70, //left endpoint of the interval
  "MaxValue":80 //right endpoint of the interval
}
```

When the target intensity needs to be described as a fixed value, then it needs to be set as:

```
"IntensityTarget": {
    "Unit": "ValueOfWeight",
    "Value":80
}
```

Example of the pushed data:

```
{
    "AthleteId": "8",
    "StartDate": "2020-11-10",
    "EndDate": "2020-11-10",
    "Workouts": [
        {
            "Description": "Description",
            "LastModifiedDate": "2020-11-10T07:26:33",
            "Title": "10KW09-2E10+5ST",
            "TotalTime": "2660",
            "Id": "277248",
            "WorkoutDay": "2020-11-10",
            "WorkoutType": "run",
            "StartTime": "2020-11-10T06:00:00",
            "Structure": [
                {
                    "IntensityClass": "Active",
                    "Name": "name1",
                    "Ftp": 0,
                    "ThresholdHr": 0,
                    "ThresholdSpeed": 0,
                    "Length": {
                        "Unit": "Second",
                        "Value": 120
                    },
                    "IntensityTarget": [],
                    "Description": "",
                    "Type": "Step"
                },
                {
                    "Type": "Repetition",
                    "Length": {
                        "Unit": "Repetition",
                        "Value": "2"
                    },
                    "Steps": [
                        {
                            "IntensityClass": "CoolDown",
                            "Name": "name2",
                            "Ftp": 0,
                            "ThresholdHr": 0,
                            "ThresholdSpeed": 0,
                            "Length": {
                                "Unit": "Second",
                                "Value": 600
                            },
                            "Type": "Step",
                            "IntensityTarget": {
                                "Unit": "RangeOfThresholdHr",
                                "MinValue": 135,
```

```
                        "MaxValue": 151
                    },
                    "Description": "heartrate 135 ~ 151 bpm"
                },
                {
                    "IntensityClass": "Active",
                    "Name": "name3",
                    "Ftp": 0,
                    "ThresholdHr": 0,
                    "ThresholdSpeed": 0,
                    "Length": {
                        "Unit": "Second",
                        "Value": 300
                    },
                    "Type": "Step",
                    "IntensityTarget": [],
                    "Description": ""
                }
            ]
        }
    ]
}
```


Return Parameter Description

Parameter	Type	Description	Notes
result	String	Return code	
message	String	Return message	
data	object	Data	
- startDate	32 int type	The start date for coros to accept data	
- endDate	32 int type	The end date for coros to accept data	

Example of the returned result:

```
{
    "message": "OK",
    "result": "0000",
    "data": {
        "StartDate": 20200917,//the start date when the Coros platform starts
receiving data
        "EndDate": 20200922,   //the end data when the Coros platform stops receiving
data
  }
}
```

6.2 Delete the Training Schedule

Partner Platform

6.2.1 Description

Training workouts created through the endpoint of Chapter 6.1 can be deleted through this endpoint. Only training workouts that have not been executed in today's and future dates can be deleted.

6.2.2 Example of request URL

Request method: POST

Example: `https://open.coros.com/coros/tp/workout/deleteById`

```
curl --location 'https://open.coros.com/coros/tp/workout/deleteById' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'token=rg1-*****f123ddaa07fa72d*****' \
--data-urlencode 'openId=25b5ab***************27da0c7f' \
--data-urlencode 'workoutIds=[20002,100000,100002,100001]'
```

6.2.3 Request parameters

Description of the request body

Parameter name	Must?	Type	Unit	Description
token	Yes	String	64	Proof of authorization: accessToken
openId	Yes	String	32	The unique identifier of the COROS platform's user
workoutIds	Yes	String		The list of unique identifier for the training schedule(Workout[i].id submitted in Chapter 6.1),such as [100013,100014]

6.2.4 Description of the returned result

Name	Type	Description	Note
result	String	Result code	0000 indicates success; 5001 indicates parameter error; 5006 indicates invalid authorization; 30009 indicates no access Others indicate failure
message	String	Message description	Message description. Different messages about the errors are returned according to different errors.
data	List	Returned data	
- successIdList	list	The list of id for success	
- failIdList	list	The list of id for failure	

Example of the returned result:

```
{
    "data": {
        "failIdList": [
            100013
        ],
        "successIdList": [
            100017
        ]
    },
    "message": "OK",
    "result": "0000"
}
```


7 Route feature's API

7.1 Receiving the route data from the partner platform

7.1.1 Description

The partner platform pushes to COROS the route data. To begin with, it has to be linked to a COROS user account. Then it pushes the route data of the user to COROS.

7.1.2 Description of the API call

Calling method

HTTP request method: POST

Content-Type: multipart/form-data;

URL: https://open.coros.com/coros/route/push

Test environment address: https://opentest.coros.com/coros/route/push

HTTP Header's parameter

Name	Must?	Type	Description	Note
token	Yes	String	Authentication credential: accessToken	The parameter is placed in the header field of the http request.

HTTP Body's parameter

Name	Must?	Type	Description	Note
openId	Yes	String	The unique identifier of the COROS user	
openUserId	Yes	String	The partner platform's user ID	
routeId	Yes	64-bit int	Route ID	
routeFile	Yes	File	Route file	The size of the file cannot exceed 50MB.
routeFileType	Yes	32-bit int	Type of the route file	0: GPX 1: KML
type	Yes	32-bit int	Workout type	1: bike riding 2: running
name	Yes	String	Route name	The name is 100 characters long.
distance	Yes	Float	Route distance	Unit: meter, keeping 2 decimal places
timestamp	Yes	64-bit int	Time stamp of route creation	Accurate to seconds
language	Yes	String	The route's language type	Eg: zh-CN、en-US
duration	No	32-bit int	The route's duration	
elevationGain	No	Float	Accumulated elevation gain	Unit: meter, keeping 2 decimal places
imageUrl	No	String	Route thumbnail	

Description of input parameters

The format of input parameters is as follows:

Content-Type:multipart/form-data;

```
curl --location --request POST 'https://opentest.coros.com/coros/route/push' \
--header 'token: token123' \
--form 'openId="123"' \
--form 'openUserId="123"' \
--form 'routeId="123"' \
--form 'routeFile=@"/route/route.gpx"' \
--form 'routeFileType="1"' \
--form 'type="2"' \
--form 'name="Route name"' \
--form 'distance="1000.56"' \
--form 'timestamp="1641439666"' \
--form 'language="zh-CN"' \
--form 'duration="600"' \
--form 'elevationGain="10.80"' \
--form 'imageUrl="https://abc.abc.com/route.jpg"'
```

7.1.3 Description of the returned result

Name	Type	Description	Note
result	String	Result code	0000 indicates success; 1031 indicates parameter error; 1008 indicates that the size of the route file exceeds the limit; 5006 indicates invalid authorization; 5010 indicates invalid openid; 13001 indicates the repeated upload of the route file 30009 indicates no access Others indicate failure
message	String	Message description	Message description. Different messages about the errors are returned according to different errors.

After a call is completed as normal, the result is returned as follows as an example:

```
{
  "message":"ok",
  "result":"0000"
}
```

8 The partner platform uploads the workout data

8.1 The partner platform synchronously uploads the workout data

8.1.1 Description

The partner platform uploads the workout data to COROS. To begin with, it has to be linked to a COROS user account. Then it uploads the workout data of the user to COROS.

So far, only the fit file format can be uploaded.

8.1.2 Description of the API call

Calling method

HTTP request method: POST

Content-Type: multipart/form-data;

URL：https://open.coros.com/coros/file/synchronous

Test environment address: https://opentest.coros.com/coros/file/synchronous

HTTP Header's parameter

Name	Must?	Type	Description	Note
token	Yes	String	Authentication credential: accessToken	The parameter is placed in the header field of the http request.

HTTP Body's parameter

Name	Must?	Type	Description	Note
openId	Yes	String	The unique identifier of the COROS user	
fileType	Yes	32-bit int	File type. Only the fit format is supported for now.	4: fit file
sportFile	Yes	File	Workout data file	
name	No	String	Workout name	name.length<=200
language	No	String	language tag	BCP 47 Code: en-US,de-De,zh-CN, de-DE,fr-FR,ja-JP etc.

Description of input parameters

The format of input parameters is as follows:

Content-Type:multipart/form-data;

```
curl --location --request POST 'https://opentest.coros.com/coros/file/synchronous' \
--header 'token: abc' \
--form 'sportFile=@"/fitFile/abc.fit"' \
--form 'fileType="4"' \
--form 'openId="abc"'
```

8.1.3 Description of the returned result

Name	Type	Description	Note
result	String	Result code	0000 indicates success; 1031 indicates parameter error; 1008 indicates that the size of the route file exceeds the limit; 5006 indicates invalid authorization; 5010 indicates invalid openid; 5082 indicates that the workout data has been uploaded to COROS and that it cannot be repeatedly uploaded<b 5096 indicates that this workout data is not supported by COROS 5098 indicates uploaded failure; 30009 indicates no access Others indicate failure
message	String	Message description	Message description. Different messages about the errors are returned according to different errors.
data	List	Returned data	
- labelId	64-bit int	The workout data ID saved in the COROS platform	

After a call is completed as normal, the result is returned as follows as an example:

```
{
    "data": [
        {
            "labelId": 443847671331979261
        }
    ],
    "message": "OK",
    "result": "0000"
}
```

8.2 The partner platform asynchronously uploads the workout data

8.2.1 Description

The partner platform uploads the workout data to COROS. To begin with, it has to be linked to a COROS user account. Then it uploads the workout data of the user to COROS.

So far, only the fit file format can be uploaded.

8.2.2 Description of the API call

Calling method

HTTP request method: POST

Content-Type: multipart/form-data;

URL：https://open.coros.com/coros/file/upload

Test environment address: https://opentest.coros.com/coros/file/upload

HTTP Header's parameter

Name	Must?	Type	Description	Note
token	Yes	String	Authentication credential: accessToken	The parameter is placed in the header field of the http request.

HTTP Body's parameter

Name	Must?	Type	Description	Note
openId	Yes	String	The unique identifier of the COROS user	
fileType	Yes	32-bit int	File type. Only the fit format is supported for now.	4: fit format. File name extension: .fit
sportFile	Yes	File	Workout data file	
name	No	String	Workout name	name.length<=200
language	No	String	language tag	BCP 47 Code: en-US,de-De,zh-CN,etc.

Description of input parameters

The format of input parameters is as follows:

Content-Type:multipart/form-data;

```
curl --location --request POST 'https://opentest.coros.com/coros/file/upload' \
--header 'token: abc' \
--form 'sportFile=@"/fitFile/abc.fit"' \
--form 'fileType="4"' \
--form 'openId="abc"'
```

8.2.3 Description of the returned result

Name	Type	Description	Note
result	String	Result code	0000 indicates success; 1031 indicates parameter error; 1008 indicates that the size of the route file exceeds the limit; 5006 indicates invalid authorization; 5010 indicates invalid openid; 5082 indicates that the workout data has been uploaded to COROS and that it cannot be repeatedly uploaded 5096 indicates that this workout data is not supported by COROS 5098 indicates uploaded failure; 30009 indicates no access Others indicate failure
message	String	Message description	Message description. Different messages about the errors are returned according to different errors.
data	List	Returned data	
- uploadId	64-bit int	Upload ID saved in the COROS platform	

After a call is completed as normal, the result is returned as follows as an example:

```
{
    "data": [
        {
            "uploadId": 443847671331979264
        }
    ],
    "message": "OK",
    "result": "0000"
}
```


8.3 Query the status of the partner platform's asynchronous upload of workout data

8.3.1 Description

The partner platform calls the API that asynchronously uploads the workout data （https://open.coros.com/coros/file/upload） and COROS performs the upload tasks in sequence in the back-end. The partner platform can query the status of the asynchronous upload through this API.

8.3.2 Description of the API call

Calling method

HTTP request method: GET

Content-Type: multipart/form-data;

URL：https://open.coros.com/coros/file/upload/get

Test environment address: https://opentest.coros.com/coros/file/upload/get

HTTP Header's parameter

Name	Must?	Type	Description	Note
token	Yes	String	Authentication credential: accessToken	The parameter is placed in the header field of the http request.

HTTP Body's parameter

Name	Must?	Type	Description	Note
openId	Yes	String	The unique identifier of the COROS user	
uploadId	Yes	64-bit int	The uploadId returned from the asynchronous upload	

Description of input parameters

The format of input parameters is as follows:

Content-Type:multipart/form-data;

```
curl --location --request GET 'https://open.coros.com/coros/file/upload/get?
openId=123&uploadId=122' \
--header 'token: 123'
```

8.3.3 Description of the returned result

Name	Type	Description	Note
result	String	Result code	0000 indicates success; 1031 indicates parameter error; 1008 indicates that the size of the route file exceeds the limit; 5006 indicates invalid authorization; 5010 indicates invalid openid; 5082 indicates that the workout data has been uploaded to COROS and that it cannot be repeatedly uploaded<b 5096 indicates that this workout data is not supported by COROS 30009 indicates no access Others indicate failure
message	String	Message description	Message description. Different messages about the errors are returned according to different errors.
data	List	Returned data	
- uploadId	64-bit int	Upload ID saved in the COROS platform	
- status	32-bit int	Upload status	-1: Execution failed 1: Execution underway 2: Execution completed

After a call is completed as normal, the result is returned as follows as an example:


```
{
    "data": {
        "status": 2,
        "uploadId": 443866905538035714
    },
    "message": "OK",
    "result": "0000"
}
```

Addendum

1. Quota and rate limit

- 1000 maximum calls/minute.
- Please contact us if you would like to raise the cap.

2. HTTP Response Code

- 429 Too Many Requests - it means that you have exceeded the rate limit.