ReDoc documentation access

- Create account at https://aiendurance.com
- Request access to API documention via email from markus@aiendurance.com. Specify email address of created AI Endurance account
- Wait for access to be granted
- Log in at https://aiendurance.xyz/admin/

API endpoints in ReDoc

ReDoc Documentation

Authorization:

Authorize all API calls via HTTP header. For example for the <partner_id> account with API key <auth_key> the request header should be:

```
Unset
{
  'PARTNER-ID': <partner_id>,
  'Authorization': 'Bearer <auth_key>',
}
```

Endpoints:

All endpoints have the root url https://aiendurance.xyz/api/

- usercore_partner/

- goal_event_partner/
- prediction_partner/
- availability_week_partner/
- availability_day_partner/
- training_plan_partner/
- future_workout_static_partner/
- recovery_prediction_partner/
- cycling_activity_detail_partner/
- running_activity_detail_partner/
- swimming_activity_detail_partner/
- recovery_model_partner/
- recovery_metric_partner/
- notification_partner/

GET options to filter endpoints

- usercore_partner/?user=123
- goal_event_partner/?user=123
- prediction_partner/?goal_event=123
- availability_week_partner/?user=123
- availability_day_partner/?availability_week=123
- training_plan_partner/?user=123
- future_workout_static_partner/?plan=123
- recovery_prediction_partner/?plan=123
- cycling_activity_detail_partner/?user=123
- running_activity_detail_partner/?user=123
- swimming_activity_detail_partner/?user=123
- recovery_model_partner/?user=123
- recovery_metric_partner/?user=123
- notification_partner/?user=123

Required parameters for training plan creation

Required for usercore_partner/

- user_name
- user_type
- birthyear
- gender
- units
- weight_kg
- hours_week (if user_type Runner or Cyclist)
- hours_week_tri_ride (if user_type Triathlete)
- hours_week_tri_run (if user_type Triathlete)
- hours_week_tri_swim (if user_type Triathlete)
- do_use_running_power

Required for goal_event_partner/

- event_type
- event_date
- user (id)

Required for prediction_partner/

- discipline
- weight
- user (id)
- goal_event (id)

Required for availability_week_partner/

- user (id)

Required for availability_day_partner/

- day_int
- act_type
- hours
- user (id)
- availability_week (id)

Relevant output parameters of usercore_partner/

- zwift_zip: all bike and run (in pace only, no running power) workouts as .zwo files for Zwift
- zwift_dfa_ramp_zip: bike and run DFA alpha 1 ramp test for Zwift as .zwo
- zones
- zones_run_power
- Intensity_distrib_plan
- can_optimize_endurance_thresholds
- can_use_running_power
- critical_swim_speed
- use_good_athletes_<user_type>: good_athletes accumulated model had to be used because of lack of data
- *_<user_type>_cverror_percent: prediction errors for all relevant disciplines *

Optional parameters for training plan creation

These parameters *can be set manually.* If they are not set manually they will be determined automatically.

Optional for usercore_partner/

- zones

These are the cycling (in Watts) and running (in Pace m/s) zones of the user. There are no swimming zones as these are derived from critical swim speed. Has to be submitted in the following object form:

```
Unset
"zones": {
    "Ride": {
        "Endurance": 234.16954073242258,
        "Tempo": 249.3814862188829,
```

```
        "Threshold": 285.0074128215805,
        "VO2Max": 338.4463027256267,
        "Anaerobic": null
    },
    "Run": {
       "Endurance": 3.8197053070831624,
       "Tempo": 4.032934030534835,
       "Threshold": 4.268188515649368,
       "VO2Max": 4.671481918702851,
       "Anaerobic": null
   }
}
```

- zones_run_power

These are the running zones in power (in Watts) of the user. If the user uses running power (`do_use_running_power = true` and `can_use_running_power = true`) these zones are used. If the user does not use running power the above `zones['Run']` pace units are used. Has to be submitted in the form:

```
Unset
"zones_run_power": {
    "Endurance": 234.16954073242258,
    "Tempo": 249.3814862188829,
    "Threshold": 285.0074128215805,
    "VO2Max": 338.4463027256267,
    "Anaerobic": null
}
```

API endpoints not listed in ReDoc documentation

fit_partner_ping/ (POST)

```
Unset
{
  'activity_files': [
    'file_type': 'FIT',
    'activity_name': 'string'/null,
    'priority': 0,1,2 (0 is lowest priority=bulk import,
otherwise 1),
    'user_id': 'string',
    'source': 'garmin',
    'overwrite_mode': false (true gives option to
overwrite what is in db already),
    'start_time_in_seconds': 'string' (as send by
Garmin),
    'callback_url': 'string' (where AI Endurance can
download fit file),
    'summary_id': 'string' (as send by Garmin)
  ]
}
```

- Onboarding flow: send all fit files with priority = 0, then send bulk_fit_transfer_complete to task_partner (see below)
- Single activity flow (user just finished an activity, already has an account): priority = 1

task_partner/ (POST)

Sending of fit files complete

```
Unset
{
  'bulk_fit_transfer_complete': true,
  'source': 'garmin',
  'is_initial_import': true for user signing up for app,
false for connecting later,
  'user_id': 'string'
}
```

Automatically triggers Request new training plan if `is_initial_import = true`, **do not trigger** Request new training plan as well if a new plan should be created.

Request new training plan

```
Unset
{
  'do_find_plan': true,
  'is_update': true if 'difficulty' of plan should stay
the same, false if restart,
  'user_id': 'string'
}
```

Re-train the user’s ML model/digital twin

```
Unset
{
  'do_train_model': true,
  'user_id': 'string'
}
```

Skip a future workout

Handled like missed workout plan adaptation via notification workflow.

```
Unset
{
  'do_skip_workout': true,
  'user_id': 'string',
  'workout_pk': 'string',
}
```

Make adaptive training plan adjustments, i.e. answer to received Notification from AI Endurance API

```
Unset
{
  'do_adapt_plan': true,
  'user_id': 'string',
  ..data_json (included in related Notification),
}
```

Send daily check-in result

```
Unset
{
  'do_receive_recovery_data': true,
  'calendar_date': 'string' (yyyy-mm-dd when the result
would be displayed to user),
  'utc_offset': 'string' (number of seconds offset of
local time where measurement was taken to UTC),
  'rmssd': 'string' (rmssd hrv in milliseconds),
  'hr_rest': 'string' (resting heart rate in bpm),
```

```
  'arm_shoulder_soreness': 'string' (0-10),
  'core_soreness': 'string' (0-10),
  'leg_soreness': 'string' (0-10),
  'user_id': 'string',
}
```

Events/webhooks sent from AI Endurance API to partner

Sent to <event_url> at partner using event_url_key as authentication via header

```
Unset
{
  'X-Api-Key': <event_url_key>
}
```

Plan finding complete

```
Unset
{
  'eventType': 'find_plan_complete',
  'userId': 'string',
  'eventId': 'string',
}
```

Bulk fit import complete

```
Unset
{
  'eventType': 'bulk_fit_import_complete',
  'userId': 'string',
  'eventId': 'string',
}
```

Individual fit file import complete

```
Unset
{
  'eventType': 'fit_import_complete',
  'userId': 'string',
  'summaryId': 'string' (identifying the fit file, see
Garmin Ping summaryId),
  'actType': 'string' ('Ride' or 'Run'),
  'actId': 'string' (id of either
CyclingActivityPartnerDetail or RunningActivityPartner
Detail),
  'eventId': 'string',
}
```

Re-training of ML model/digital twin complete

```
Unset
{
  'eventType': 'train_model_complete',
  'userId': 'string',
  'did_create_new_plan': false (default), true (if model
training resulted in a situation where a new plan needed
to be created, e.g. user now has enough data available to
use running power as they originally requested),
```

```
  'eventId': 'string',
}
```

Received corrupt fit file

Unset

```
{
  'eventType': 'fit_file_corrupt',
  'userId': 'string',
  'summaryId': 'string' (identifying the fit file, see
Garmin Ping summaryId),
  'eventId': 'string',
}
```

Notification for training plan adaptation was created

To be confirmed by user (see Notification flow below)

Unset

```
{
  'eventType': 'notification_created',
  'userId': 'string',
  'notificationId': 'string',
  'eventId': 'string',
}
```

Recovery model is ready

The updated recovery value for cardiovascular + orthopedic readiness in each sport can be fetched from recovery_model_partner/ endpoint.

```
Unset
{
  'eventType': 'recovery_model_ready',
  'userId': 'string',
  'recoveryModelId': 'string',
  'eventId': 'string',
}
```

Example Prediction Templates

These are the disciplines to be optimized for the user's goal event. To be submitted to prediction_partner/

Cyclist

Any combination of cycling disciplines, for example

```
Unset
  "Gran Fondo": [
    { discipline: "Ultra Endurance Power", weight: 0.8 },
    { discipline: "Critical Cycling Power", weight: 0.2 },
  ],
  "Hilly Gran Fondo": [
    { discipline: "Ultra Endurance Power", weight: 0.7 },
    { discipline: "Critical Cycling Power", weight: 0.3 },
  ],
```

Runner

Any combination of running disciplines, for example

User using running power

```
Unset
    "5k/10k": [{ discipline: "Critical Running Power",
weight: 1.0 }],
    "Half-Marathon": [
      { discipline: "Ultra Endurance Power", weight: 0.7 },
      { discipline: "Critical Running Power", weight: 0.3 },
    ],
    Marathon: [
      { discipline: "Ultra Endurance Power", weight: 0.8 },
      { discipline: "Critical Running Power", weight: 0.2 },
    ],
    Ultra: [{ discipline: "Ultra Endurance Power", weight:
1.0 }],
    "Hilly Ultra": [
      { discipline: "Ultra Endurance Power", weight: 0.8 },
      { discipline: "Critical Running Power", weight: 0.2 },
    ]
```

User NOT using running power

```
Unset
  Ultra: [{ discipline: "Ultra Endurance Pace", weight: 1.0
}],
  "Hilly Ultra": [
    { discipline: "Ultra Endurance Pace", weight: 0.8 },
    { discipline: "10k", weight: 0.2 },
  ]
```

Triathlete

User using running power

Unset

```
    'Not Hilly Sprint Distance': [
        { discipline: 'FTP', weight: 1.0 },
        { discipline: 'Critical Running Power', weight:
1.0 },
    ],
    'Not Hilly Olympic Distance': [
        { discipline: 'FTP', weight: 0.9 },
        { discipline: 'Endurance Power Cycling', weight:
0.1 },
        { discipline: 'Critical Running Power', weight:
0.9 },
        { discipline: 'Endurance Power Running', weight:
0.1 },
    ],
    'Not Hilly Half-IM': [
        { discipline: 'Endurance Power Cycling', weight:
0.8 },
        { discipline: 'FTP', weight: 0.2 },
        { discipline: 'Endurance Power Running', weight:
0.8 },
        { discipline: 'Critical Running Power', weight:
0.2 },
    ],
    'Not Hilly IM': [
        { discipline: 'Endurance Power Cycling', weight:
1.0 },
        { discipline: 'Endurance Power Running', weight:
1.0 },
    ],
    'Hilly Sprint Distance': [
        { discipline: 'FTP', weight: 1.0 },
        { discipline: 'Critical Running Power', weight:
1.0 },
    ],
    'Hilly Olympic Distance': [
        { discipline: 'FTP', weight: 1.0 },
```

```
        { discipline: 'Critical Running Power', weight:
1.0 },
    ],
    'Hilly Half-IM': [
        { discipline: 'Endurance Power Cycling', weight:
0.7 },
        { discipline: 'FTP', weight: 0.3 },
        { discipline: 'Endurance Power Running', weight:
0.7 },
        { discipline: 'Critical Running Power', weight:
0.3 },
    ],
    'Hilly IM': [
        { discipline: 'Endurance Power Cycling', weight:
0.8 },
        { discipline: 'FTP', weight: 0.2 },
        { discipline: 'Endurance Power Running', weight:
0.8 },
        { discipline: 'Critical Running Power', weight:
0.2 },
    ]
```

User NOT using running power

Unset

```
    'Not Hilly Sprint Distance': [
        { discipline: 'FTP', weight: 1.0 },
        { discipline: '5k Pace', weight: 1.0 },
    ],
    'Not Hilly Olympic Distance': [
        { discipline: 'FTP', weight: 0.9 },
        { discipline: 'Endurance Power Cycling', weight:
0.1 },
        { discipline: '10k Pace', weight: 1.0 },
```

```
    ],
    'Not Hilly Half-IM': [
        { discipline: 'Endurance Power Cycling', weight:
0.8 },
        { discipline: 'FTP', weight: 0.2 },
        { discipline: 'Endurance Pace', weight: 0.8 },
        { discipline: '10k Pace': weight: 0.2},
    ],
    'Not Hilly IM': [
        { discipline: 'Endurance Power Cycling', weight:
1.0 },
        { discipline: 'Endurance Pace', weight: 1.0 },
    ],
    'Hilly Sprint Distance': [
        { discipline: 'FTP', weight: 1.0 },
        { discipline: '5k Pace', weight: 1.0 },
    ],
    'Hilly Olympic Distance': [
        { discipline: 'FTP', weight: 1.0 },
        { discipline: '10k Pace', weight: 1.0 },
    ],
    'Hilly Half-IM': [
        { discipline: 'Endurance Power Cycling', weight:
0.7 },
        { discipline: 'FTP', weight: 0.3 },
        { discipline: 'Endurance Pace', weight: 0.7 },
        { discipline: '10k Pace', weight: 0.3 },
    ],
    'Hilly IM': [
        { discipline: 'Endurance Power Cycling', weight:
0.8 },
        { discipline: 'FTP', weight: 0.2 },
        { discipline: 'Endurance Pace', weight: 0.8 },
        { discipline: '10k Pace', weight: 0.2 },
    ],
```

User Notifications

AI Endurance API reacts to certain circumstances in the user's training (e.g. missed session) with user notifications that have to be accepted or declined by the user. These include modifications to their training plan or zone updates.

They follow the following sequence of calls:

1. AI Endurance API calls <event_url> with eventType `notification_created` to tell the partner about the existence of a new notification
2. Partner fetches this notification from the notification_partner/ endpoint
   It includes title and text to display to the user and a field called `data_json`
3. Partner sends a notification to their frontend and waits for a confirmation or decline by the user
4. If the user confirms partner sends a POST request to task_partner/ with eventType `do_adapt_plan = true`

Identifier

- **update_zones_alpha_Ride**, **update_zones_alpha_Run**
  - Background: recommend user to update selection of training zones user_core.zones or user_core.zones_run_power based on DFA a1 measurements.
  - Example data_json:

Unset

```
{Endurance: 200, Tempo: 220, Threshold: 250, VO2Max: 270, Anaerobic: null}
```

  - Rules for submission back to AI Endurance API: currently requires both PATCH requests to update user_core.zones and/or

user_core.zones_run_power and submit data_json to task_partner/ as explained in API documentation v2.3

- **not_on_track_new_plan**
  - Background: user has not followed training plan and is urged to request a new training plan
  - data_json to API documentation v2.3 :

```
Unset
{}
```

- **not_recovered**
  - Background: user is not ready for the workout due to measured cardio or orthopedic readiness
  - Example data_json to API documentation v2.3 :

```
Unset
{
  'to_delete': {
    'to_delete_pk': 'string',
  }
}
```

- **missed_workout**
  - Background: user did not perform the workout planned for yesterday
  - Example data_json to API documentation v2.3 :

```
Unset
{
  'to_switch': {
    'to_switch_pk': 'string',
    'to_be_switched_against_pk': 'string',
```

```
  }
}
```

Workout generators

Get workouts in json format either from specifying time at intensity and/or workout stress or by prompting GPT4.

/ess_workout/ (POST)

Input:

```
Unset
{
  "user_id": null,
  "act_type": "Ride",
  "units": "Metric",
  "zones": {
    "Endurance": 231,
    "Tempo": 247,
    "Threshold": 291,
    "VO2Max": 356,
    "Anaerobic": null
  },
  "intensity_type": "Tempo",
  "ess": 180.0,
  "intensity_time": 1800,
  "repeats": 2
}
```

- “user_id”: null or user_id of associated AI Endurance user (required)
- "act_type": "Ride" or "Run" (required)
- "units": "Metric" or "Imperial" (required)

- "zones": {"Endurance": 231, "Tempo": 247, "Threshold": 291, "VO2Max": 356, "Anaerobic": null} (required)
- "intensity_type": “Endurance”, "Tempo", “Threshold”, “VO2Max”, or “Anaerobic” (required)
- "ess": 180.0 (optional)
- "intensity_time": 1800 (in seconds, optional)
- "repeats": 2 (optional)

- You need to specify **at least either intensity_time or ess**
- Zones are the **upper bounds of each zone** in Watts for “Ride” and in meters/second for “Run”. The “Anaerobic” upper bound can be null.

Output:

Unset


```
{
  "act_type": "Ride",
  "date": "2024-01-17T15:04:49.908588Z",
  "user_id": null,
  "future_workout_static_id": 1965565,
  "units": "Metric",
  "zones": {
    "Endurance": 231.8969506163821,
    "Tempo": 247.10142691631026,
    "Threshold": 291.0305694792098,
    "VO2Max": 356.6273750348052,
    "Anaerobic": 392.29011253828577
  },
  "steps": {
    "act_type": "Ride",
    "type": "Tempo",
    "duration_warmup": 600.0,
    "target_warmup_high": 231.8969506163821,
    "target_warmup_low": null,
    "duration_cooldown": 180.0,
    "target_cooldown_high": 231.8969506163821,
    "target_cooldown_low": null,
```

```
    "duration_add_endurance": 0,
    "target_add_endurance_high": 231.8969506163821,
    "target_add_endurance_low": null,
    "target_on_high": 247.10142691631026,
    "target_on_low": 231.8969506163821,
    "target_off_high": 185.5175604931057,
    "target_off_low": null,
    "duration_off": 300,
    "repeats": 2,
    "duration_on": 3360.0
  },
  "zwift_zip": "UEsDBBQAAAAIAJh4...AA",
  "ess": 179.0
}
```


- "future_workout_static_id": 1965565 (associated id of the workout in AI Endurance)
- "steps": see explanation on https://aiendurance.com/partnerapi
- "zwift_zip": "UEsDBBQAAAAIAJh4...AA" base64 encoded zip file of .zwo of the workout for Zwift.

/gpt_workout/ (POST)

Input:


```
Unset
{
  "user_id": null,
  "act_type": "Ride",
  "units": "Metric",
  "zones": {
    "Endurance": 231,
    "Tempo": 247,
    "Threshold": 291,
    "VO2Max": 356,
```

```
    "Anaerobic": null
  },
  "text_array": ["hard tempo workout"],
  "gptw_history_pk": null,
}
```

- “user_id”: null or user_id of associated AI Endurance user (required)
- "act_type": "Ride" or "Run" (required)
- "units": "Metric" or "Imperial" (required)
- "zones": {"Endurance": 231, "Tempo": 247, "Threshold": 291, "VO2Max": 356, "Anaerobic": null} (required)
- "text_array": [“hard tempo workout”] list with prompt instructions in this thread (required)
- "gptw_history_pk”: null, or gptw_history_pk from previous call (optional)

Output:

Unset


```
{
  "act_type": "Ride",
  "date": "2024-01-17T15:04:49.908588Z",
  "user_id": null,
  "future_workout_static_id": 1965565,
  "units": "Metric",
  "zones": {
    "Endurance": 231.8969506163821,
    "Tempo": 247.10142691631026,
    "Threshold": 291.0305694792098,
    "VO2Max": 356.6273750348052,
    "Anaerobic": 392.29011253828577
  },
  "steps": {
    "act_type": "Ride",
    "type": "Tempo",
    "duration_warmup": 600.0,
    "target_warmup_high": 231.8969506163821,
    "target_warmup_low": null,
```

```
    "duration_cooldown": 180.0,
    "target_cooldown_high": 231.8969506163821,
    "target_cooldown_low": null,
    "duration_add_endurance": 0,
    "target_add_endurance_high": 231.8969506163821,
    "target_add_endurance_low": null,
    "target_on_high": 247.10142691631026,
    "target_on_low": 231.8969506163821,
    "target_off_high": 185.5175604931057,
    "target_off_low": null,
    "duration_off": 300,
    "repeats": 2,
    "duration_on": 3360.0
  },
  "zwift_zip": "UEsDBBQAAAAIAJh4...AA",
  "ess": 179.0,
   "text_array": ["hard tempo workout"],
   "gptw_history_pk": "1234",
}
```


- "future_workout_static_id": 1965565 (associated id of the workout in AI Endurance)
- "steps": see explanation on https://aiendurance.com/partnerapi
- "zwift_zip": "UEsDBBQAAAAIAJh4...AA" base64 encoded zip file of .zwo of the workout for Zwift.

Threads:

If you want to iterate over a prompt you can do this by appending prompts to the “text_array” list and referencing the “gptw_history_pk” from a previous call. Here’s an example that shortens the interval durations of the first returned workout in a second prompt:

1. Input:

```
Unset
{
  "user_id": null,
  "act_type": "Ride",
  "units": "Metric",
  "zones": {
    "Endurance": 231,
    "Tempo": 247,
    "Threshold": 291,
    "VO2Max": 356,
    "Anaerobic": null
  },
  "text_array": ["hard tempo workout"],
  "gptw_history_pk": null,
}
```

1. Output:

```
Unset
{
  "act_type": "Ride",
  "date": "2024-01-17T15:04:49.908588Z",
  "user_id": null,
  "future_workout_static_id": 1965565,
  "units": "Metric",
  "zones": {
    "Endurance": 231.8969506163821,
    "Tempo": 247.10142691631026,
    "Threshold": 291.0305694792098,
    "VO2Max": 356.6273750348052,
    "Anaerobic": 392.29011253828577
  },
  "steps": {
    "act_type": "Ride",
    ...
```

```
    },
  "zwift_zip": "UEsDBBQAAAAIAJh4...AA",
  "ess": 179.0,
   "text_array": ["hard tempo workout"],
   "gptw_history_pk": "1234",
}
```

Now lets make the the intervals shorter

2. Input:

Unset

```
{
  "user_id": null,
  "act_type": "Ride",
  "units": "Metric",
  "zones": {
    "Endurance": 231,
    "Tempo": 247,
    "Threshold": 291,
    "VO2Max": 356,
    "Anaerobic": null
  },
  "text_array": ["hard tempo workout", "with shorter
intervals"],
  "gptw_history_pk": "1234",
}
```

1. Output:

Unset

```
{
  "act_type": "Ride",
  "date": "2024-01-17T15:04:49.908588Z",
  "user_id": null,
```

```
  "future_workout_static_id": 1965565,
  "units": "Metric",
  "zones": {
    "Endurance": 231.8969506163821,
    "Tempo": 247.10142691631026,
    "Threshold": 291.0305694792098,
    "VO2Max": 356.6273750348052,
    "Anaerobic": 392.29011253828577
  },
  "steps": {
    "act_type": "Ride",
    ...
    },
  "zwift_zip": "UEsDBBQAAAAIAJh4...AA",
  "ess": 179.0,
   "text_array": ["hard tempo workout", "with shorter
intervals"],
   "gptw_history_pk": "1235",
}
```