# FathomAI - Plans API (v 4.6.0)

## Technical Summary

### Overview

This technical summary provides prospective third party partners ('providers') of **Fathom's Plans API** with a brief summary of the type of information returned from the service along with a summary of the minimum technical and data requirements. This document is not a substitute for the full API specification.

### Daily Plan

The plans service generates a **Daily Plan** for an athlete based on one or both of the following data elements:

- **Daily Readiness** - information about the athlete's training plan and pain/soreness for a given day
- **Post-Workout** - information about an athlete's workout session along with their Rating of Perceived Exertion (RPE) and pain/soreness following the workout

A **Daily Plan** can be created with as little as one of the above data elements. As this information is gathered over time, Fathom's analytics also use historical patterns in pain/soreness and workouts to identify underlying imbalances unique to the athlete which influence the creation of their **Daily Plan**.

A **Daily Plan** provides a personalized, research-driven prep, recovery, and corrective exercise plan for an athlete. A plan may consist of one or more modalities, targeting one or more recovery goals Fathom analytics identifies for the athlete.

The **Daily Plan** includes modalities such as **foam rolling**, **static stretching**, **active stretching**, **dynamic stretching**, **targeted muscle activation**, and **integrated movement** exercises personalized for the athlete for that day. These exercises are provided in a sequence consistent with sports science research to expedite tissue recovery, reduce pain, and prevent injury.

Other modalities do not include exercises but are assigned to a plan based on athlete needs. These modalities include **heat**, **ice** and **cold water immersion**.

Recommended dosages are also provided for each exercise and modality. These dosages are associated with three different active times which correspond with minimal, optimal, and comprehensive sequences of activities. These sequences are designed to achieve each of the athlete's unique combination of goals. Additionally, dosages are also provided by goal, allowing the athlete to further customize their recovery.

## Technical and Data Requirements

### Terminology

The terminology of RFC 2119 (specifically **must**, **should**, **may** and their negatives) applies. The word **will**, when applied to the Plans API ("the API"), has the same meaning as **must**.

Each third-party partner will be recognised as a "provider" and will be assigned a unique 'Provider Code'. This

**will** be a string matching the regular expression `^[a-z][a-z0-9\-]{3,31}$`.

### Protocol

The API supports communication over HTTPS only. The client **must** recognise the Amazon Trust Services LLC certificate root.

### Encoding

The API supports communication using JSON encoding only. The client **must** submit the headers `Content-Type: application/json` and `Accept: application/json` (or a subtype `application/{subtype}+json`, if appropriate) for all requests. Failure to do so **will** result in a `415 Unsupported Media Type` response. The API **will** include the header `Content-Type: application/json` (or a subtype if appropriate) with its response.

### Endpoints

Each provider will also be assigned a unique set of test and production endpoints to access the plans service.

### Authentication

Unless otherwise specified, the endpoints in the API are authenticated by a JWT bearer token. The client **must** submit the header `Authorization: <JWT>` with all requests. Failure to do so, or submitting an invalid or expired JWT, **will** result in a `401 Unauthorized` response.

It is expected that partners will normally generate and sign their own JWTs for their clients, providing appropriate authorization for each athlete in accordance with their business and compliance requirements.

#### Signing keys

Prior to integrating with the API, each partner **must** supply a set of one or more public keys with which they will sign clients' JWT credentials. This **must** take the form of an RFC 7517 JSON Web Key Set document, for example:

```
{
    "keys": [
        {
            "kid": "fathom_001",
            "alg": "RS256",
            "kty": "RSA",
            "use": "sig",
            "e": "AQAB",
            "n": "snrCqqc2tC......Z29H9DBLIQ",
            "_env": ["dev", "test"]
        },
        {
            "kid": "fathom_002",
            "alg": "RS256",
            "kty": "RSA",
            "use": "sig",
            "e": "AQAB",
            "n": "yuHDihazrP......UuEPOofbVQ",
            "_env": "production"
        }
    ]
}
```

Each key within the key set **must** have a `kid` field matching the regular expression `^([a-z][a-z0-9\-]{3,31})_([a-z0-9\-]+)$`, where the first group of the expression is the partner's Provider Code.

Each key within the key set **must** have a `use` field set to `sig` if the key is to be used for signing JWTs. Partners **should not** include keys with other values in the key set.

At the present time the only algorithm from the RFC 7518 list supported is RSA-256, so the value of the `alg` field for each key in the key set **must** be `RS256`. We hope to support at least `ES256` in the near future.

Partners **may** include the non-standardized fields `_nbf` and `_exp` in key definitions; if these fields are provided, they **must** follow the semantics of the corresponding JWT claim fields in RFC7519, and the API **will** interpret them similarly (that is to say, a JWT with an `iat` value falling before the corresponding key's `nbf` value or after its `exp` value, will not be considered valid). This allows partners to perform key rotation in an orderly fashion.

Partners **may** include the non-standardized field `_env` in key definitions; if this field is provided the value **must** be a String matching the regular expression `^[a-z0-9]+$` or an array of such Strings, and the API **will** interpret this as a list of the environments where the key should be accepted. This allows partners to use different signing keys for production and non-production environments.

#### JWT claims

The JWTs provided by clients **must** contain the following claims:

- `iss`, which **must** be a String matching the regular expression `^([a-z][a-z0-9\-]{3,31})_([a-z0-9\-]+)$`, where the first group of the expression is the partner's Provider Code.
- `aud`, which **must** be a String matching the regular expression `^fathom(_[a-z0-9]+)?$` (or an array containing such a String). If the group is provided (eg `fathom_production`), the API **will** treat the second part as an environment specifier, and **will** only accept as valid JWTs targeted at its own environment (for instance, the production API will only accept tokens with an `aud` value of `fathom` and/or `fathom_production`).
- `iat` **must** be specified.
- `exp` **must** be specified. The total period of validity of the JWT (ie the time range between the lesser of `iat` and `nfb`, and `exp`) **must not** be greater than 86400 seconds.
- `sub`, which **must** be a Uuid identifying the athlete on whose behalf the client is acting. In general the API will only allow requests which correspond to actions affecting this user.
- `scope`, which **must** be a String containing a space-separated list of Scopes, where each Scope is a String matching the regular expression `^[a-z][a-z0-9\.:]*$`. The following scopes are recognised by the API:
    - `fathom.plans:read`: provides access to read-only functionality for the athlete identified by the `sub` claim
    - `fathom.plans:write`: provides access to write functionality for the athlete identified by the `sub` claim. This is a superset of `fathom.plans:read`.
    - `fathom.plans:service`: provides access to all functionality for all users. JWTs with this scope are subject to additional validation conditions described below.

#### Service tokens

Partners **may** interact with the API on a business-to-business basis instead of, or in addition to, building clients which allow users to interact with the API directly. Partners' private servers **may** authenticate such requests using a JWT carrying the `fathom.plans:service` scope. Such tokens must meet the following additional validation conditions:

- The value of the `sub` field **must** be the String `00000000-0000-4000-8000-000000000000`.
- The total period of validity of the JWT **must not** be greater than 600 seconds.

### General responses

In addition to the API responses and the specific responses for each endpoint, the server **may** respond with one of the following HTTP responses:

- `400 Bad Request` with `Status` header equal to `InvalidSchema`, if the JSON body of the request does not match the requirements of the endpoint.
- `403 Forbidden` with `Status` header equal to `Forbidden`, if the user is not allowed to perform the requested action.
- `404 Unknown` with `Status` header equal to `UnknownEndpoint`, if an invalid endpoint was requested.

# Data Requirements

## Types

Required data elements are based on the following simple types :

- `string`, `number`, `integer`, `boolean` : as defined in the JSON Schema standard.
- `Uuid` : a `string` matching the regular expression `^[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}$`, that is, the string representation of an RFC 4122 UUID.
- `Datetime` : a `string` matching the regular expression `/\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}(Z|+\d{2}:\d{2})/` and representing a date and time in full ISO 8601 format.

## Daily Readiness

### Required Data Elements

The following data elements are required when following the **Daily Readiness** pathway to **Daily Plan** generation.

- `date_time` **should** be a Datetime and reflect the local time that survey was taken
- `soreness` **should** reflect a list of body parts( `sore_part` ) with symptoms. Length **could** be 0.

`sore_part` **should** include the following:

- `body_part` **should** be an integer reflecting BodyPart enum of the body part with symptom
- `side` **should** be an integer, either 0 (both sides/non-bilateral), 1 (left) or 2 (right)
- `tight` **should** be an integer (1-10) indicating the severity of tightness felt. If not reported, it should be `null`
- `knots` **should** be reported for muscles only and **should** be an integer (1-10) indicating the severity of discomfort caused by knots, tigger points, and musclular adhesions felt. If not reported, it should be `null`
- `ache` **should** be an integer (1-10) indicating the severity of discomfort felt described as an ache, dull, or sore, indicating inflammation and muscle spasms are likely present. If not reported, it should be `null`
- `sharp` **should** be an integer (1-10) indicating the severity of discomfort felt described as sharp, acute, shooting, indicating that inflammation and muscle spasms are likely present. If not reported, it should be `null`

Note: Fathom can customize the processing of symptoms data upon request to accommodate third-party systems that only report a subset of measures.

### Optional Data Elements

The following data elements are not required to generate a plan using the **Daily Readiness** pathway, but enhance the customization of the plan for the athlete.

- `sessions` **should** be a list of workout sessions completed but not yet submitted to Fathom.
- `sessions_planned` **should** be a boolean representing whether the athlete plans to train again that day.

`session` **if present**, **should** follow the requirements outlined in the **Post-Workout** pathway

## Post-Workout

### Required Data Elements

The following data elements are required when following the **Post-Workout** pathway to **Daily Plan** generation. Sessions can either be logged manually be an athlete or transferred from a third party source such as Apple's HealthKit app.

- `session` **should** include the data elements as specified below
- `sessions_planned` **should** be a boolean representing whether the athlete plans to train again that day.

`session` data elements

- `event_date` **should** be a Datetime and reflect the start time of the session
- `end_date` is **optional** Datetime parameter that reflects the end time of the session from third party source
- `sport_name` **should** be an integer reflecting SportName enumeration.
- `duration` **should** be an integer and reflect the minutes duration which the athlete confirmed (third party source) or entered (manually logged session).
- `calories` **if present**, **should** be an integer and represent the calorie information obtained from a third party source workout *(only needed for third party source workouts)*
- `distance` **if present**, **should** be an integer and represent the distance information obtained from a third party source workout *(only needed for third party source workouts)*
- `source` **if present**, **should** be 0 for manually logged session and 1 for a third party source workout
- `deleted` **if present**, **should** be a boolean and true to delete the workout transferred from a third party source
- `ignored` **if present**, **should** be a boolean and true for short walking workouts. This is typically only used for sessions created by third-party apps that should be excluded from Fathom processing.
- `hr_data` **if present**, **should** be the heart rate data associated with a third party source workout. Each hr will have `startDate` (Datetime), `endDate` (Datetime) and `value` (integer) *(only needed for third party source workouts)*
- `description` is **optional** string parameter to provide a short description of the session they're adding
- `post-session-survey` **should** follow requirements below

`post-session-survey` data elements

- `event_date` **should** be a Datetime and reflect the local date and time when the survey (associated with the workout) was completed
- `RPE` **should** be an integer between 1 and 10 indicating the *Rating of Perceived Exertion* of the athlete during the session
- `soreness` **should** follow the same definition as in *Daily Readiness*

## Symptom-Reporting

### Required Data Elements

The following data elements are required when following the **Symptom-Reporing** pathway to **Daily Plan** generation.

- `event_date` **should** be a Datetime and reflect the local time that survey was taken
- `soreness` **should** follow the same definition as in *Daily Readiness*