FathomAI - Plans API (v 4.6.0)

Common provisions

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**.

Protocol

The API supports communication over HTTPS only. Requests sent via HTTP will be redirected to HTTPS connections.

Encoding

The API supports communication using JSON encoding only. The client **must** submit the headers `Content-Type: application/json` and `Accept: application/json` 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` with its response.

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.

Two token sources are accepted:

- Tokens generated by Amazon Cognito and acquired as part of authentication to the Users Service (Fathom Mobile App Only);
- Tokens generated by partners according to the guidelines specified below. 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 AWS API Gateway responses and the specific responses for each endpoint, the serve**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.

Schema

Simple

The following simple types **may** be used in requests and responses:

- `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.

Endpoints

Daily Readiness

Create

This endpoint can be called to register a new daily readiness survey.

Query String

The client **must** submit a request to the endpoint `/plans/{version}/daily_readiness/{User UUID}`. The request method **must** be `POST`.

Request

The client **must** submit a request body containing a JSON object having the following schema:

```
{
    "date_time": Datetime,
    "soreness": [sore_part, sore_part],
    "sessions": [session, session],
    "sessions_planned": boolean,
    "health_sync_date": Datetime,
    "user_age": number
}
```

- `date_time` **should** reflect the local time that survey was taken
- `soreness` **should** reflect a list of body parts(`sore_part`) with symptoms. Length **could** be 0.
- `sessions` is **optional** and **should** be a list of session objects, where each session matches the body of Create Session.
- `sessions_planned` is **optional** and **should** represent whether the user plans to train again that day.
- `health_sync_date` (Fathom Mobile App Only) is **optional** and only provided if one of the sessions is obtained from a third party source
- `user_age` (Fathom Mobile App Only) is **optional** and only needed if one of the sessions is obtained from a third party source and contains heart rate data
- `sore_part` **should** have the following schema:

```
{
    "body_part": number,
    "side": number,
    "tight": number,
    "knots": number,
    "ache": number,
    "sharp": number
}
```

- `body_part` **should** be an integer reflecting BodyPart enumeration as defined in Appendix
- `side` **should** be an integer reflecting Side enumeration as defined in Appendix
- `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(see Appendix) 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`

```
 POST /plans/{version}/daily_readiness/{User UUID} HTTPS/1.1
Host: apis.{env}.fathomai.com
Content-Type: application/json
Authorization: eyJraWQ...ajBc4VQ

{
    "date_time": "2018-12-10T17:45:24Z",
    "soreness":[{
                                    "body_part": 14,
                                    "side": 2
                                    "tight": null,
                                    "knots": null,
                                    "ache": 3,
                                    "sharp": 6,
                                }],
    "sessions": [{"event_date": "2018-12-10T12:30:00Z",
                  "sport_name": 3,
                  "duration": 90,
                  "description": "Evening Practice",
                  "post_session_survey": {
                                          "event_date": "2018-12-10T17:45:24Z",
                                          "RPE": 9,
                                          "soreness": []
                                         }
                  }
                ],
    "user_age": 25,
    "sessions_planned": false
}
```

Response

If the write was successful, the Service**will** respond with HTTP Status `201 Created`, with a body having the following schema:

```
{
    "daily_plans": [daily_plan]
}
```

- `daily_plan` will have the same schema as defined in Get Daily Plan.

Session

Create

This endpoint can be called to log a new session to today's plan and should include the session's information as well as post-session surveys for the session being logged.

Query String

The client **must** submit a request to the endpoint `/plans/{version}/session/{User UUID}`. The request method **must** be `POST`.

Request

The client **must** submit a request body containing a JSON object having the following schema:

```
{
    "event_date": Datetime,
    "sessions": [session, session],
    "health_sync_date": Datetime,
    "sessions_planned": Boolean,
    "user_age": number
}
```

- `event_date` **should** reflect the date and time when the survey is submitted.
- `health_sync_date` (Fathom Mobile App Only) is **optional** and only provided if one of the sessions is obtained from a third party source
- `sessions_planned` **should** represent whether the user plans to train again that day
- `user_age` (Fathom Mobile App Only) is **optional** and only provided if one of the sessions is obtained from a third party source and contains heart rate data

- `session` **should** be of the following schema

```
{
    "event_date": Datetime,
    "end_date": Datetime,
    "session_type": integer,
    "sport_name": integer,
    "duration": integer,
    "description": string,
    "calories": integer,
    "distance": integer,
    "source": integer,
    "deleted": boolean,
    "ignored": boolean,
    "post_session_survey": {
                            "event_date": Datetime
                            "RPE": integer,
                            "soreness": [sore_part, sore_part],
                            "clear_candidates": [sore_part]}
                        },
    "hr_data": [hr, hr, hr],
}
```

- `event_date` **should** reflect the start time of the session (third party/FathomPRO data) or when the session is logged (manually logged session)
- `end_date` is **optional** parameter that reflects the end time of the session for third party/FathomPRO data
- `session_type` **should** reflect SessionType enumeration. NOTE: We'll only use 6 moving forward
- `sport_name` **should** reflect SportName enumeration as defined in Appendix.
- `duration` **should** be in minutes and reflect the duration which the user confirmed (third party data) or entered (manually logged session).
- `calories` **if present**, **should** be calorie information obtained from third party workout*(only needed for third party workout)*
- `distance` **if present**, **should** be distance information obtained from third party workout*(only needed for third party workout)*
- `source` **if present**, **should** be SessionSource enumeration as defined in Appendix
- `deleted` **if present**, **should** be true if the user deletes the workout detected from the third party source
- `ignored` **if present**, **should** be true for short walking workouts.
- `hr_data` **if present**, **should** be the heart rate data associated with the third party workout. Each hr will have `startDate`, `endDate` and `value` *(only needed for third party workout)*
- `description` is **optional** parameter to provide short description of the session they're adding

```
POST /plans/{version}/session/{User UUID} HTTP/1.1
Host: apis.{env}.fathomai.com
Content-Type: application/json
Authorization: eyJ0eX...xA8
Cache-Control: no-cache
{
    "event_date": "2019-01-12T16:54:57Z",
    "sessions":[
                {
                    "event_date": "2019-01-12T10:41:57Z",
                    "session_type": 6,
                    "sport_name": 1,
                    "duration": 14,
                    "description": "Evening Practice",
                    "calories": 100,
                    "distance": 200,
                    "end_date": "2019-02-12T10:54:57Z",
                    "source": 1,
                    "deleted": false,
                    "ignored": false,
                    "hr_data":  [
                                {"value": 153,
                                 "startDate": "2019-01-12T10:43:08.490-0500",
                                 "endDate": "2019-01-12T10:43:08.490-0500"
                                 },
                            ],
                    "post_session_survey": {
                                            "event_date": "2019-02-08T16:54:57Z",
                                            "RPE": 5,
                                            "soreness": [],
                                            "clear_candidates": []
                                            }
                    }

                ],
    "sessions_planned": true,
    "health_sync_date": "2019-02-08T16:54:57Z"
}
```

Response

If the write was successful, the Service**will** respond with HTTP Status `201 Created`, with a body having the following schema:

```
{
    "daily_plans": [daily_plan]
}
```

- `daily_plan` will have the same schema as defined in Get Daily Plan.

Mark no sessions planned

This endpoint can be called to notify that no sessions are planned for the day.

Query String

The client **must** submit a request to the endpoint `/plans/{version}/session/{User UUID}/no_sessions`. The request method **must** be `POST`.

Request

The client **must** submit a request body containing a JSON object having the following schema:

```
{
    "event_date": Datetime
}
```

- `event_date` **should** reflect the date and time when the call is made

```
POST /plans/{version}/session/{User UUID}/no_sessions HTTP/1.1
Host: apis.{env}.fathomai.com
Content-Type: application/json
Authorization: eyJ0eX...xA8
Cache-Control: no-cache

{
    "event_date": "2018-09-14T19:54:48Z"

}
```

Response

If the request was successful, the Service**will** respond with HTTP Status `200 OK`, with a body having the following schema:

```
{
    "daily_plans": [daily_plan]
}
```

- `daily_plan` will have the same schema as defined in Get Daily Plan.

Delete

This endpoint can be called to delete existing externally regimented session from today or yesterday's plan.

Query String

The client **must** submit a request to the endpoint `/plans/{version}/session/{User UUID}/{session_id}`. The request method **must** be `DELETE`.

Request

The client **must** submit a request body containing a JSON object having the following schema:

```
{
    "event_date": Datetime,
    "session_type": integer
}
```

- `event_date` **should** reflect the date and time the session was scheduled for.
- `session_type` **should** reflect SessionType enumeration.

```
DELETE /plans/{version}/session/{User UUID}/{Session ID} HTTPS/1.1
Host: apis.{env}.fathomai.com
Content-Type: application/json
Authorization: eyJraWQ...ajBc4VQ

{
    "event_date": "2018-08-10T16:30:00Z",
    "session_type": 6
}
```

Response

If the delete was successful, the Service**will** respond with HTTP Status `200 OK`, with a body having the following schema:

```
{
    "message": "success"
}
```

Update

This endpoint can be called to update and combine a manually logged session with a third party workout session

Query String

The client **must** submit a request to the endpoint `/plans/{version}/session/{User UUID}/{session_id}`. The request method **must** be `PATCH`.

Request

The client **must** submit a request body containing a JSON object having the following schema:

```
{
    "event_date": Datetime,
    "sessions": [session],
    "health_sync_date": Datetime
}
```

- `event_date` **should** reflect the local date and time when the call is being made
- `session` **should** follow the same schema as defined in Create Session
- `health_sync_date` (Fathom Mobile App only) **should** reflect the time when the data was obtained from the third party source

```
PATCH /plans/{version}/session/{User UUID}/{Session ID} HTTPS/1.1
Host: apis.{env}.fathomai.com
Content-Type: application/json
Authorization: eyJraWQ...ajBc4VQ

{
    "event_date": "2019-02-08T16:30:00Z",
    "sessions": [
                {
                   "event_date": "2019-02-08T16:45:55Z",
                   "end_date": "2019-02-08T16:45:55Z",
                   "session_type": 6,
                   "sport_name": 1,
                   "duration": 90.5,
                   "calories": 50,
                   "distance": 100,
                   "source": 1,
                   "hr_data": [],
                   "post_session_survey": {
                                           "event_date": "2019-02-08T16:30:00Z",
                                           "RPE": null,
                                           "soreness": [],
                                           "clear_candidates": []
                                           }

                }
                ],
    "health_sync_date": "2019-02-08T16:30:00Z"
}
```

Response

If the update was successful, the Service**will** respond with HTTP Status `200 OK`, with a body having the following schema:

```
{
    "message": "success"
}
```

Typical Sessions history

This endpoint can be called to get typical sessions that the user logs

Query String

The client **must** submit a request to the endpoint `/plans/{version}/session/{User UUID}/typical`. The request method **must** be `POST`.

Request

For `POST` method, the client **must** submit a request body containing a JSON object having the following schema:

```
{
    "event_date": Datetime
}
```

```
POST /plans/{version}/session/{User UUID}/typical HTTPS/1.1
Host: apis.{env}.fathomai.com
Content-Type: application/json
Authorization: eyJraWQ...ajBc4VQ

{
    "event_date": "2019-02-08T16:30:00Z"
}
```

Authentication is required for this endpoint

Response

The Service **will** respond with HTTP Status `200 OK`, with a body having the following schema:

```
{
    "typical_sessions": [sesson, session]
}
```

- `typical sessions` will be a list of sessions that the user has logged in the last 14 days

- `session` object will be of the following schema

```
{
    "count": integer,
    "duration": integer,
    "event_date": datetime,
    "session_type": integer,
    "sport_name": integer,
    "strength_and_conditioning_type": integer
}
```

Symptoms

Submit

This endpoint can be called to submit symptoms to receive a new plan without submitting a session.

Query String

The client **must** submit a request to the endpoint `/plans/version/symptoms/{User UUID}`. The request method **must** be `POST`.

Request

The client **must** submit a request body containing a JSON object having the following schema:

```
{
    "event_date": Datetime,
    "soreness": [sore_part, sore_part]
}
```

- `event_date` **should** be a Datetime and reflect the local time that survey was taken
- `soreness` **should** reflect a list of body parts(`sore_part`) as defined in Readiness Create

```
POST /plans/version/symptoms/{User UUID} HTTPS/1.1
Host: apis.{env}.fathomai.com
Content-Type: application/json
Authorization: eyJraWQ...ajBc4VQ

{
    "event_date": "2019-10-29T17:45:24Z",
    "soreness":[
        {
            "body_part": 18,
            "side": 0,
            "tight": 4,
            "knots": null,
            "sharp": null,
            "ache": null
        }
    ]
}
```

Response

If the write was successful, the Service**will** respond with HTTP Status `201 Created`, with a body having the following schema:

```
{
    "daily_plans": [daily_plan]
}
```

- `daily_plan` will have the same schema as defined in Get Daily Plan.

Active Recovery

Mark Started (Exercise Modalities)

This endpoint can be called to mark the start of exercise-based modalities.

Query String

The client **must** submit a request to the endpoint `/plans/{version}/active_recovery/{User UUID}/exercise_modalities`. The request method **must** be `POST`.

Request

The client **must** submit a request body containing a JSON object having the following schema:

```
{
    "event_date": Datetime,
    "recovery_type": string
}
```

- `event_date` **should** be the time when user checks the first exercise for the session.
- `recovery_type` **should** be one of `pre_active_rest` or `post_active_rest`

```
POST /plans/{version}/active_recovery/{User UUID}/exercise_modalities HTTPS/1.1
Host: apis.{env}.fathomai.com
Content-Type: application/json
Authorization: eyJraWQ...ajBc4VQ
{
    "event_date": "2018-09-21T17:53:39Z",
    "recovery_type": "pre_active_rest"
}
```

Response

If the write was successful, the Service **will** respond with HTTP Status `200 OK`, with a body having the following schema:

```
{
    "message": "success"
}
```

Mark Completed (Exercise Modalities)

This endpoint can be called to mark the completion of exercise-based modalities.

Query String

The client **must** submit a request to the endpoint `/plans/{version}/active_recovery/{User UUID}/exercise_modalities`. The request method **must** be `PATCH`.

Request

The client **must** submit a request body containing a JSON object having the following schema:

```
{
    "event_date": Datetime,
    "recovery_type": string
    "completed_exercises": [string]
}
```

- `event_date` **should** be the time when user completes the session.
- `recovery_type` **should** be one of `pre_active_rest` or `post_active_rest`
- `completed_exercises` **should** be a list representing the exercises that the user checked off

```
PATCH /plans/{version}/active_recovery/{User UUID}/exercise_modalities HTTPS/1.1
Host: apis.{env}.fathomai.com
Content-Type: application/json
Authorization: eyJraWQ...ajBc4VQ
{
    "event_date": "2018-09-21T17:53:39Z",
    "recovery_type": "post_active_rest",
    "completed_exercises": ["3", "5", "20", "142"]
}
```

Response

If the write was successful, the Service **will** respond with HTTP Status `202 Accepted`, and return the daily_plan in the body with following schema.

```
{
    "daily_plans": [daily_plan]
}
```

- `daily_plan` will have the same schema as defined in Get Daily Plan.

Daily Plan

Get Daily Plan

Query String

The client **must** submit a request to the endpoint `/plans/{version}/daily_plan/{User UUID}`. The request method **must** be `POST`.

Request

The client **must** submit a request body containing a JSON object having the following schema:

```
{
    "event_date": Datetime
    "start_date": string,
    "end_date": string
}
```

- `event_date` **should** reflect the time (in local timezone) when the api call is made.
- `start_date` **should** be of format `yyyy-mm-dd`
- `end_date` **should** be of format `yyyy-mm-dd` but is **optional**

```
POST /plans/{version}/daily_plan/{User UUID} HTTPS/1.1
Host: apis.{env}.fathomai.com
Content-Type: application/json
Authorization: eyJraWQ...ajBc4VQ

{
    "event_date": "2018-07-31T02:50:02Z",
    "start_date": "2018-07-31"
}
```

Response

The Service **will** respond with HTTP Status `200 OK`, with a body having the following schema:

```
{
    "daily_plans": [daily_plan1, daily_plan2],
    "readiness": readiness,
    "typical_sessions": [session, session]
}
```

- `readiness` is only returned if readiness survey hasn't been completed for the day and will follow the schema defined in Daily Readiness
- `typical_sessions` is only returned if readiness survey hasn't been completed for the day and will follow the schema defined in Session
- `daily_plans` **could** be an empty list
- each `daily_plan*` will be of following schema:

```
{
    "date": Date,
    "day_of_week": integer,
    "pre_active_rest": [PreActiveRest],
    "completed_pre_active_rest": [PreActiveRest],
    "heat": null,
    "completed_heat": [],
    "warm_up": [],
    "completed_warm_up": [],
    "training_sessions": [Session],
    "cool_down": [],
    "completed_cool_down": [],
    "post_active_rest": [PostActiveRest],
    "completed_post_active_rest": [PostActiveRest],
    "ice": null,
    "completed_ice": [],
    "cold_water_immersion": null,
    "completed_cold_water_immersion": [],
    "cross_training_sessions": [],
    "daily_readiness_survey_completed": Boolean,
    "landing_screen": integer,
    "last_sensor_sync": Datetime,
    "last_updated": Datetime,
    "nav_bar_indicator": null,
    "post_active_rest_completed": Boolean,
    "pre_active_rest_completed": Boolean,
    "sessions_planned": Boolean,
    "train_later": Bolean
}
```

- Any of the `completed_*` attributes could be empty list
- Any of the exercise modalities (`pre_active_rest`, `post_active_rest`, `warm_up` and `cool_down`) could be empty list
- `warm_up` and `cool_down` will always be empty list
- All of the body part modalities (`heat`, `ice`, `cold_water_immersion`) will be null
- `PreActiveRest` has the following example schema

```
{
    "active": true,
    "completed": false,
    "completed_date_time": null,
    "default_plan": "Complete",
    "event_date_time": "2019-05-07T00:00:00Z",
    "high_relative_intensity_logged": false,
    "high_relative_load_session": true,
    "active_stretch_exercises": [AssignedExercise, AssignedExercise]
    "inhibit_exercises": [AssignedExercise, AssignedExercise],
    "isolated_activate_exercises": [AssignedExercise, AssignedExercise],
    "static_integrate_exercises": [AssignedExercise, AssignedExercise],
    "static_stretch_exercises": [AssignedExercise, AssignedExercise]
    "muscular_strain_increasing": false,
    "start_date_time": null,
}
```

- `PostActiveRest` has following example schema

```
{
    "active": true,
    "completed": false,
    "completed_date_time": null,
    "default_plan": "Complete",
    "event_date_time": "2019-05-07T00:00:00Z",
    "high_relative_intensity_logged": false,
    "high_relative_load_session": true,
    "inhibit_exercises": [AssignedExercise, AssignedExercise],
    "isolated_activate_exercises": [AssignedExercise, AssignedExercise],
    "muscular_strain_increasing": false,
    "start_date_time": null,
    "static_integrate_exercises": [AssignedExercise, AssignedExercise],
    "static_stretch_exercises": [AssignedExercise, AssignedExercise]
}
```

- `AssignedExercise` has the following example schema

```
{
    "name" : "Foam Roller - Hamstrings",
    "display_name" : "Foam Roll - Hamstrings",
    "library_id" : "3",
    "description" : "Place foam roller just before the glute & roll the length of
                     your hamstring. If you find a tender spot, hold for 30 seconds.
                     Don’t have a foam roller?
                     You can use a tennis ball or water bottle.",
    "youtube_id" : null,
    "bilateral" : true,
    "seconds_per_rep" : null,
    "seconds_per_set" : 30,
    "unit_of_measure" : "seconds",
    "position_order" : 0,
    "duration_efficient" : 60,
    "duration_complete" : 60,
    "duration_comprehensive" : 60,
    "goal_text" : "",
    "equipment_required" : [
        "Foam Roller"
    ],
    "dosages" : [
        {
            "goal" : {
                "text" : "Soreness",
                "priority" : 1,
                "goal_type" : 1
            },
            "priority" : "1",
            "efficient_reps_assigned" : 30,
            "efficient_sets_assigned" : 1,
            "complete_reps_assigned" : 30,
            "complete_sets_assigned" : 1,
            "comprehensive_reps_assigned" : 30,
            "comprehensive_sets_assigned" : 1,
            "default_efficient_reps_assigned" : 30,
            "default_efficient_sets_assigned" : 1,
            "default_complete_reps_assigned" : 30,
            "default_complete_sets_assigned" : 1,
            "default_comprehensive_reps_assigned" : 30,
            "default_comprehensive_sets_assigned" : 1,
            "ranking" : 0
        }
    ]
}
```

Appendix

Enumerations

BodyPart

```
chest = 2
abdominals = 3
groin = 5
quads = 6
knee = 7
shin = 8
ankle = 9
foot = 10
it_band = 11
lower_back = 12
glutes = 14
hamstrings = 15
calves = 16
achilles = 17
upper_back_neck = 18
elbow = 19
wrist = 20
lats = 21
biceps = 22
triceps = 23
forearm = 24
it_band_lateral_knee = 27
hip_flexor = 28
deltoid = 29
```

Side

```
none_unilateral = 0
left = 1
right = 2
```

SessionSource

```
user = 0
health = 1
user_health = 2
three_sensor = 3
```

All body parts

```
 shoulder = 1
chest = 2
abdominals = 3
hip = 4
groin = 5
quads = 6
knee = 7
shin = 8
ankle = 9
foot = 10
it_band = 11
lower_back = 12
general = 13
glutes = 14
hamstrings = 15
calves = 16
achilles = 17
upper_back_neck = 18
elbow = 19
wrist = 20
lats = 21
biceps = 22
triceps = 23
forearm = 24
core_stabilizers = 25
erector_spinae = 26
it_band_lateral_knee = 27
hip_flexor = 28
deltoid = 29
deep_rotators_hip = 30
obliques = 31
anterior_tibialis = 40
peroneals_longus = 41
posterior_tibialis = 42
soleus = 43
gastrocnemius_medial = 44
bicep_femoris_long_head = 45
bicep_femoris_short_head = 46
semimembranosus = 47
semitendinosus = 48
adductor_longus = 49
adductor_magnus_anterior_fibers = 50
adductor_magnus_posterior_fibers = 51
adductor_brevis = 52
gracilis = 53
pectineus = 54
```

```
 vastus_lateralis = 55
vastus_medialis = 56
vastus_intermedius = 57
rectus_femoris = 58
tensor_fascia_latae = 59 # hips
piriformis = 60 # deep rotator of hip
gastrocnemius_lateral = 61
sartorius = 62
gluteus_medius_anterior_fibers = 63
gluteus_medius_posterior_fibers = 64
gluteus_minimus = 65
gluteus_maximus = 66
quadratus_femoris = 67
popliteus = 68
external_obliques = 69
quadratus_lumorum = 70
psoas = 71
iliacus = 72
transverse_abdominis = 73
internal_obliques = 74
rectus_abdominis = 75
upper_trapezius = 76
levator_scapulae = 77
middle_trapezius = 78
lower_trapezius = 79
rhomboids = 80
pectoralis_minor = 81
pectoralis_major = 82
anterior_deltoid = 83
medial_deltoid = 84
posterior_deltoid = 85
upper_body = 91
lower_body = 92
full_body = 93
semimembranosus_semitendinosus = 100
anterior_adductors = 101
rectus_femoris_vastus_intermedius = 102
glute_med = 103
upper_traps_levator_scapulae = 105
middle_traps_rhomboids = 106
pec_major_minor = 107
hip_flexor_merge = 108
```

SportName

```
basketball = 0
baseball = 1
softball = 2
cycling = 3
field_hockey = 4
football = 5
general_fitness = 6
golf = 7
gymnastics = 8
skate_sports = 9
lacrosse = 10
rowing = 11
rugby = 12
diving = 13
soccer = 14
pool_sports = 15
tennis = 16
distance_running = 17
sprints = 18
jumps = 19
throws = 20
volleyball = 21
wrestling = 22
weightlifting = 23
track_field = 24
archery = 25
australian_football = 26
badminton = 27
bowling = 28
boxing = 29
cricket = 30
curling = 31
dance = 32
equestrian_sports = 33
fencing = 34
fishing = 35
handball = 36
hockey = 37
martial_arts = 38
paddle_sports = 39
racquetball = 40
sailing = 41
snow_sports = 42
squash = 43
surfing_sports = 44
```

```
 swimming = 45
table_tennis = 46
water_polo = 47
cross_country_skiing = 48
downhill_skiing = 49
kick_boxing = 50
snowboarding = 51
endurance = 52
power = 53
speed_agility = 54
strength = 55
cross_training = 56
elliptical = 57
functional_strength_training = 58
hiking = 59
hunting = 60
mind_and_body = 61
play = 62
preparation_and_recovery = 63
stair_climbing = 64
traditional_strength_training = 65
walking = 66
water_fitness = 67
yoga = 68
barre = 69
core_training = 70
flexibility = 71
high_intensity_interval_training = 72
jump_rope = 73
pilates = 74
stairs = 75
step_training = 76
wheelchair_walk_pace = 77
wheelchair_run_pace = 78
taichi = 79
mixed_cardio = 80
hand_cycling = 81
climbing = 82
other = 83
```

Body Part Types

Joints

The following reportable body parts are considered joints

```
 elbow = 19
wrist = 20
knee = 7
ankle = 9
foot = 10
```

Ligaments

The following reportable body parts are considered ligaments

```
 it_band = 11
it_band_lateral_knee = 27
achilles = 17
```

Muscles

The following reportable body parts are considered muscles

```
 chest = 2
abdominals = 3
groin = 5
quads = 6
shin = 8
lower_back = 12
glutes = 14
hamstrings = 15
calves = 16
upper_back_neck = 18
lats = 21
biceps = 22
triceps = 23
forearm = 24
hip_flexor = 28
deltoid = 29
```

Last Modified: November 22, 2019