Namespace StravaAPILibary.API

Classes

Activities
: Provides methods to interact with Strava's Activities API.

Athletes
: Provides methods to interact with Strava's Athletes API.

Clubs
: Provides methods to interact with Strava's Clubs API.

Gears
: Provides methods to interact with Strava's Gear API.

Routes
: Provides methods to interact with Strava's Routes API.

Segments
: Provides methods for interacting with Strava's Segments API.

SegmentsEfforts
: Provides methods for interacting with Strava's Segment Efforts API.

Streams
: Provides methods for retrieving stream data (detailed time-series data) from Strava.

Uploads
: Provides methods for uploading activities and retrieving upload statuses from the Strava API.

Class Activities

Namespace: StravaAPILibary.API

Assembly: StravaAPILibary.dll

Provides methods to interact with Strava's Activities API.

```
public static class Activities
```

Inheritance

object ← Activities

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Examples

Example: Retrieve the authenticated athlete's activities:

```
var activities = await Activities.GetAthletesActivitiesAsync(accessToken, page: 1,
perPage: 30);
```

Example: Upload a new activity:

```
var uploadResponse = await Activities.PostActivityAsync(accessToken, "Morning Ride",
"gpx", @"C:\ride.gpx");
```

Remarks

This static class contains methods for retrieving, creating, and updating athlete activities in Strava. It communicates with Strava's REST API endpoints related to activities and returns JSON responses.

Features:

- Retrieve authenticated athlete's activities.
- Fetch details of a specific activity by ID.
- Get related data like laps, zones, comments, and kudos.
- Upload new activities and update existing ones.

Usage:

All methods require a valid `accessToken` (OAuth token) with appropriate scopes granted by Strava. Many methods also support pagination and optional filtering parameters.

API Documentation:

Refer to the official Strava API documentation: Strava API Reference.

Methods

GetActivityByIdAsync(string, string)

Get a specific activity by its ID.

```
public static Task<JsonObject> GetActivityByIdAsync(string accessToken, string activityId)
```

Parameters

`accessToken` string

The OAuth access token.

`activityId` string

The ID of the activity to retrieve.

Returns

Task<JsonObject>

A JsonObject containing the activity details.

Exceptions

ArgumentException

Thrown when `accessToken` or `activityId` is null or empty.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the API response is empty.

JsonException

Thrown when the JSON response cannot be parsed as a JsonObject.

GetActivityCommentsAsync(string, string)

Get the comments for a specific activity.

```
public static Task<JsonArray> GetActivityCommentsAsync(string accessToken,
string activityId)
```

Parameters

`accessToken` string

The OAuth access token.

`activityId` string

The ID of the activity to retrieve comments for.

Returns

Task<JsonArray>

A JsonArray containing the comments of the activity.

Exceptions

ArgumentException

Thrown when `accessToken` or `activityId` is null or empty.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the API response is empty.

JsonException

Thrown when the JSON response cannot be parsed as a JsonArray.

GetActivityKudosAsync(string, string)

Get the kudos given for a specific activity.

```
public static Task<JsonArray> GetActivityKudosAsync(string accessToken, string activityId)
```

Parameters

`accessToken` string

The OAuth access token.

`activityId` string

The ID of the activity to retrieve kudos for.

Returns

Task<JsonArray>

A JsonArray containing the kudos for the activity.

Exceptions

ArgumentException

Thrown when `accessToken` or `activityId` is null or empty.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the API response is empty.

JsonException

Thrown when the JSON response cannot be parsed as a JsonArray.

GetActivityLapsAsync(string, string)

Get a list of laps for a specific activity.

```
public static Task<JsonArray> GetActivityLapsAsync(string accessToken, string activityId)
```

Parameters

`accessToken` string

The OAuth access token.

`activityId` string

The ID of the activity to retrieve laps for.

Returns

Task <JsonArray>

A JsonArray containing the laps of the specified activity.

Exceptions

ArgumentException

Thrown when `accessToken` or `activityId` is null or empty.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the API response is empty.

JsonException

Thrown when the JSON response cannot be parsed as a JsonArray.

GetActivityZonesAsync(string, string)

Get the zones for a specific activity.

```
public static Task<JsonArray> GetActivityZonesAsync(string accessToken, string activityId)
```

Parameters

`accessToken` string

The OAuth access token.

`activityId` string

The ID of the activity to retrieve zones for.

Returns

Task <JsonArray>

A JsonArray containing the activity's zones.

Exceptions

ArgumentException

Thrown when `accessToken` or `activityId` is null or empty.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the API response is empty.

JsonException

Thrown when the JSON response cannot be parsed as a JsonArray.

GetAthletesActivitiesAsync(string, int, int, int, int)

Retrieves the activities of the authenticated athlete with optional filtering and pagination.

```
public static Task<JsonArray> GetAthletesActivitiesAsync(string accessToken, int before = 0,
int after = 0, int page = 1, int perPage = 30)
```

Parameters

`accessToken` string

The OAuth access token for authentication. Must be valid and not expired.

`before` int

Unix timestamp to filter activities before this date. Use 0 for no filter.

`after` int

Unix timestamp to filter activities after this date. Use 0 for no filter.

`page` int

Page number for pagination. Must be greater than 0.

`perPage` int

Number of activities per page. Must be between 1 and 200.

Returns

Task<JsonArray>

A JsonArray containing the athlete's activities.

Examples

Get recent activities:

```
var activities = await Activities.GetAthletesActivitiesAsync(accessToken, page: 1,
perPage: 10);
foreach (var activity in activities)
{
    Console.WriteLine($"Activity: {activity["name"]} - {activity["distance"]}m");
}
```

Get activities from last 30 days:

```
int after = (int)DateTimeOffset.UtcNow.AddDays(-30).ToUnixTimeSeconds();
var recentActivities = await Activities.GetAthletesActivitiesAsync(accessToken,
after: after);
```

Get activities before a specific date:

```
int before = (int)DateTimeOffset.Parse("2024-01-01").ToUnixTimeSeconds();
var oldActivities = await Activities.GetAthletesActivitiesAsync(accessToken,
before: before);
```

Implement pagination:

```
int page = 1;
int perPage = 50;
bool hasMoreActivities = true;

while (hasMoreActivities)
{
    var activities = await Activities.GetAthletesActivitiesAsync(accessToken, page: page,
perPage: perPage);

    if (activities.Count == 0)
    {
        hasMoreActivities = false;
    }
    else
    {
        // Process activities
        foreach (var activity in activities)
        {
            Console.WriteLine($"Activity: {activity["name"]}");
        }

        page++;
    }
}
```

Remarks

This method retrieves the authenticated athlete's activities from Strava. Activities are returned in reverse chronological order (most recent first).

Filtering:

- Use `after` to get activities from a specific date onwards
- Use `before` to get activities up to a specific date
- Both filters use Unix timestamps (seconds since epoch)
- Use 0 for either parameter to disable that filter

Pagination:

- Activities are paginated with a maximum of 200 per page
- Use `page` to navigate through pages
- Use `perPage` to control the number of activities returned
- Empty array is returned when no more activities are available

Required Scope:

This method requires the `activity:read_all` scope to access all activities, or `activity:read` for public activities only.

Rate Limits:

This endpoint is subject to Strava's API rate limits. Consider implementing exponential backoff for retry logic.

Exceptions

ArgumentException

Thrown when `accessToken` is null or empty.

ArgumentOutOfRangeException

Thrown when `page` or `perPage` is less than or equal to zero.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the API response is empty.

JsonException

Thrown when the JSON response cannot be parsed as a JsonArray.

PostActivityAsync(string, string, string, string, string?, bool, bool, string?, string?)

Posts (uploads) an activity file to Strava.

```
public static Task<JsonObject> PostActivityAsync(string accessToken, string name, string
dataType, string filePath, string? description = null, bool isTrainer = false, bool
isCommute = false, string? externalId = null, string? sportType = null)
```

Parameters

`accessToken` string

The OAuth access token.

`name` string

The name of the activity.

`dataType` string

The file type of the upload (fit, gpx, tcx).

`filePath` string

The path to the activity file.

`description` string

Optional: A description of the activity.

`isTrainer` bool

Optional: Set to true if this is a trainer ride.

`isCommute` bool

Optional: Set to true if this is a commute activity.

`externalId` string

Optional: An external ID to uniquely identify this upload.

`sportType` string

Optional: The sport type (e.g., "Run", "Ride").

Returns

Task<JsonObject>

A JsonObject containing the upload status.

Exceptions

ArgumentException

Thrown when accessToken, name, or dataType are invalid.

FileNotFoundException

Thrown when the specified file does not exist.

HttpRequestException

Thrown when the Strava API request fails.

JsonException

Thrown when the response cannot be parsed as JSON.

UpdateActivityAsync(string, long, string?, string?, bool?, bool?, string?, string?)

Updates an activity by its ID.

```
public static Task<JsonObject> UpdateActivityAsync(string accessToken, long activityId,
string? name = null, string? description = null, bool? isTrainer = null, bool? isCommute =
null, string? sportType = null, string? gearId = null)
```

Parameters

`accessToken` string

The OAuth access token.

`activityId` long

The ID of the activity to update.

`name` string

Optional: New name for the activity.

`description` string

Optional: New description for the activity.

`isTrainer` bool?

Optional: Whether the activity is a trainer ride.

`isCommute` bool?

Optional: Whether the activity is a commute.

`sportType` string

Optional: The sport type (e.g., "Run", "Ride").

`gearId` string

Optional: The gear ID to associate with the activity.

Returns

Task<JsonObject>

A JsonObject containing the updated activity data.

Exceptions

ArgumentException

Thrown when `accessToken` is null or empty or `activityId` is invalid.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

JsonException

Thrown when the JSON response cannot be parsed.

Class Athletes

Namespace: StravaAPILibary.API

Assembly: StravaAPILibary.dll

Provides methods to interact with Strava's Athletes API.

```
public static class Athletes
```

Inheritance

object ← Athletes

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Examples

Example: Retrieve the authenticated athlete's profile:

```
var profile = await Athletes.GetAuthenticatedAthleteProfileAsync(accessToken);
```

Example: Update athlete weight:

```
var updatedProfile = await Athletes.UpdateAuthenticatedAthleteAsync(accessToken, 72.5f);
```

Remarks

This static class contains methods for retrieving and updating information about the authenticated athlete and other related athlete-specific data. It communicates with Strava's REST API and returns JSON responses.

Features:

- Retrieve the authenticated athlete's profile details.
- Fetch athlete statistics (e.g., totals, ride/run stats).
- Retrieve heart rate and power zones.
- Update athlete information such as weight (requires `profile:write` scope).

Usage:

All methods require a valid `accessToken` (OAuth token) with the necessary scopes granted.
API Documentation:

See the official Strava API reference: Strava API Reference.

Methods

GetAuthenticatedAthleteProfileAsync(string)

Get the authenticated Athlete's profile.

```
public static Task<JsonObject> GetAuthenticatedAthleteProfileAsync(string accessToken)
```

Parameters

`accessToken` string

The access token for authentication.

Returns

Task<JsonObject>

A JsonObject containing the athlete's profile information.

Exceptions

ArgumentException

Thrown when `accessToken` is null or empty.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the API response is empty.

JsonException

Thrown when the JSON response cannot be parsed as a JsonObject.

GetAuthenticatedAthleteStatsAsync(string, string)

Get Athlete's stats for the authenticated athlete. This method retrieves the statistics of a specific athlete using their athlete ID.

```
public static Task<JsonObject> GetAuthenticatedAthleteStatsAsync(string accessToken,
string athleteId)
```

Parameters

`accessToken` string

The access token for authentication.

`athleteId` string

The ID of the athlete whose stats are to be retrieved.

Returns

Task <JsonObject>

A JsonObject containing the athlete's statistics.

Exceptions

ArgumentException

Thrown when `accessToken` or `athleteId` is null or empty.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the API response is empty.

JsonException

Thrown when the JSON response cannot be parsed as a JsonObject.

GetAuthenticatedAthleteZonesAsync(string)

Get Athlete's heart rate and power zones for the authenticated athlete.

```
public static Task<JsonObject> GetAuthenticatedAthleteZonesAsync(string accessToken)
```

Parameters

`accessToken` string

The access token for authentication.

Returns

Task <JsonObject>

A JsonObject containing the athlete's heart rate and power zones.

Exceptions

ArgumentException

Thrown when `accessToken` is null or empty.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the API response is empty.

JsonException

Thrown when the JSON response cannot be parsed as a JsonObject.

UpdateAuthenticatedAthleteAsync(string, float)

Updates the authenticated athlete's weight. Requires scope: profile:write.

```
public static Task<JsonObject> UpdateAuthenticatedAthleteAsync(string accessToken,
float weight)
```

Parameters

`accessToken` string

The OAuth access token.

`weight` float

The new weight of the athlete in kilograms.

Returns

Task<JsonObject>

A JsonObject containing the updated athlete profile.

Exceptions

ArgumentException

Thrown when the access token is null or empty.

HttpRequestException

Thrown when the API request fails.

InvalidOperationException

Thrown when the response is empty.

JsonException

Thrown when the response cannot be parsed as JSON.

Class Clubs

Namespace: StravaAPILibary.API

Assembly: StravaAPILibary.dll

Provides methods to interact with Strava's Clubs API.

```
public static class Clubs
```

Inheritance

object ← Clubs

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Remarks

This static class offers functionality for retrieving information related to Strava clubs, including club details, members, activities, and admins. All methods require a valid OAuth `accessToken` with the necessary scopes.

Features:

- Retrieve detailed information about a specific club by its ID.
- Fetch the list of clubs the authenticated athlete is a member of.
- Retrieve members, activities, and admins of a specific club.

Usage:

```
var club = await Clubs.GetClubByIdAsync(accessToken, 12345);
var members = await Clubs.GetClubMembersAsync(accessToken, 12345);
var activities = await Clubs.GetClubActivitiesAsync(accessToken, 12345);
var admins = await Clubs.GetClubAdminsAsync(accessToken, 12345);
```

API Documentation:

See the official Strava API reference: Strava API Reference.

Methods

GetClubActivitiesAsync(string, long, int, int)

Get the activities of a club by its ID.

```
public static Task<JsonArray> GetClubActivitiesAsync(string accessToken, long clubId, int
page = 1, int perPage = 30)
```

Parameters

`accessToken` string

The OAuth access token.

`clubId` long

The ID of the club.

`page` int

Page number for pagination (default: 1).

`perPage` int

Number of items per page (default: 30).

Returns

Task <JsonArray>

A JsonArray containing the activities of the club.

Exceptions

ArgumentException

Thrown when `accessToken` is null or empty or when `clubId` is invalid.

ArgumentOutOfRangeException

Thrown when `page` or `perPage` is less than or equal to zero.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the API response is empty.

JsonException

Thrown when the JSON response cannot be parsed as a JsonArray.

GetClubAdminsAsync(string, long, int, int)

Get the admins of a club by its ID.

```
public static Task<JsonArray> GetClubAdminsAsync(string accessToken, long clubId, int page =
1, int perPage = 30)
```

Parameters

`accessToken` string

The OAuth access token.

`clubId` long

The ID of the club.

`page` int

Page number for pagination (default: 1).

`perPage` int

Number of items per page (default: 30).

Returns

Task<JsonArray>

A JsonArray containing the admins of the club.

Exceptions

ArgumentException

Thrown when `accessToken` is null or empty or when `clubId` is invalid.

ArgumentOutOfRangeException

Thrown when `page` or `perPage` is less than or equal to zero.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the API response is empty.

JsonException

Thrown when the JSON response cannot be parsed as a JsonArray.

GetClubByIdAsync(string, long)

Get a club by its ID.

```
public static Task<JsonObject> GetClubByIdAsync(string accessToken, long clubId)
```

Parameters

`accessToken` string

The access token for the API.

`clubId` long

The ID of the club to retrieve.

Returns

Task<JsonObject>

A JsonObject containing the club data.

Exceptions

ArgumentException

Thrown when the access token or club ID is null or empty.

HttpRequestException

Thrown when the request to the API fails.

InvalidOperationException

Thrown when the response is empty.

JsonException

Thrown when the JSON response cannot be parsed.

GetClubMembersAsync(string, long, int, int)

Get the members of a club by its ID.

```
public static Task<JsonArray> GetClubMembersAsync(string accessToken, long clubId, int page
= 1, int perPage = 30)
```

Parameters

`accessToken` string

The OAuth access token.

`clubId` long

The ID of the club.

`page` int

Page number for pagination (default: 1).

`perPage` int

Number of items per page (default: 30).

Returns

Task<JsonArray>

A JsonArray containing the members of the club.

Exceptions

ArgumentException

Thrown when `accessToken` is null or empty or when `clubId` is invalid.

ArgumentOutOfRangeException

Thrown when `page` or `perPage` is less than or equal to zero.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the API response is empty.

JsonException

Thrown when the JSON response cannot be parsed as a JsonArray.

GetClubsAsync(string, int, int)

Get clubs for the authenticated athlete.

```
public static Task<JsonArray> GetClubsAsync(string accessToken, int page = 1, int perPage
= 30)
```

Parameters

`accessToken` string

The OAuth access token.

`page` int

Page number for pagination (default: 1).

`perPage` int

Number of items per page (default: 30).

Returns

Task<JsonArray>

A JsonArray containing the clubs of the authenticated athlete.

Exceptions

ArgumentException

Thrown when `accessToken` is null or empty.

ArgumentOutOfRangeException

Thrown when `page` or `perPage` is less than or equal to zero.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the API response is empty.

JsonException

Thrown when the JSON response cannot be parsed as a JsonArray.

Class Gears

Namespace: StravaAPILibary.API

Assembly: StravaAPILibary.dll

Provides methods to interact with Strava's Gear API.

```
public static class Gears
```

Inheritance

object ← Gears

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Remarks

This static class offers functionality for retrieving information about gear associated with an athlete's Strava account.

Features:

- Retrieve detailed information about a specific gear item (bike or shoes) by its ID.

Usage:

```
var gear = await Gears.GetGearByIdAsync(accessToken, "b12345678987655098 76");
Console.WriteLine(gear["name"]);
```

API Documentation:

See the official Strava API reference: Strava API Reference.

Methods

GetGearByIdAsync(string, string)

Get a gear by its ID.

```
public static Task<JsonObject> GetGearByIdAsync(string accessToken, string gearId)
```

Parameters

`accessToken` string

The OAuth access token.

`gearId` string

The ID of the gear.

Returns

Task<JsonObject>

A JsonObject containing the gear details.

Exceptions

ArgumentException

Thrown when `accessToken` or `gearId` is null or empty.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the API response is empty.

JsonException

Thrown when the JSON response cannot be parsed as a JsonObject.

Class Routes

Namespace: StravaAPILibary.API

Assembly: StravaAPILibary.dll

Provides methods to interact with Strava's Routes API.

```
public static class Routes
```

Inheritance

object ← Routes

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Remarks

This static class contains methods for retrieving and exporting route data associated with Strava athletes.

Features:

- Retrieve detailed information about a specific route by its ID.
- Retrieve a list of routes associated with a specific athlete.
- Export a route in GPX or TCX format for external usage.

Usage:

```
var route = await Routes.GetRouteByIdAsync(accessToken, 123456);
Console.WriteLine(route["name"]);

var gpxData = await Routes.GetRouteGpxExportAsync(accessToken, 123456);
File.WriteAllText("route.gpx", gpxData);
```

API Documentation:

See the official Strava API reference: Strava Routes API Reference.

Methods

GetRouteByIdAsync(string, long)

Get a route by its ID.

```
public static Task<JsonObject> GetRouteByIdAsync(string accessToken, long routeId)
```

Parameters

`accessToken` string

The OAuth access token.

`routeId` long

The ID of the route to retrieve.

Returns

Task<JsonObject>

A JsonObject containing the route data.

Exceptions

ArgumentException

Thrown when `accessToken` is null or empty, or when `routeId` is invalid.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the API response is empty.

JsonException

Thrown when the JSON response cannot be parsed as a JsonObject.

GetRouteGpxExportAsync(string, long)

Get a GPX export of a route by its ID.

```
public static Task<string> GetRouteGpxExportAsync(string accessToken, long routeId)
```

Parameters

`accessToken` string

The OAuth access token.

`routeId` long

The ID of the route to export.

Returns

Task <string>

A string containing the GPX data of the route.

Exceptions

ArgumentException

Thrown when `accessToken` is null or empty, or when `routeId` is invalid.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the GPX response is empty.

GetRouteTcxExportAsync(string, long)

Get a TCX export of a route by its ID.

```
public static Task<string> GetRouteTcxExportAsync(string accessToken, long routeId)
```

Parameters

`accessToken` string

The OAuth access token.

`routeId` long

The ID of the route to export.

Returns

Task <string>

A string containing the TCX data of the route.

Exceptions

ArgumentException

Thrown when `accessToken` is null or empty, or when `routeId` is invalid.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the TCX response is empty.

GetRoutesByAthleteIdAsync(string, long, int, int)

Get a list of routes from a specific athlete.

```
public static Task<JsonArray> GetRoutesByAthleteIdAsync(string accessToken, long athleteId,
int page = 1, int perPage = 30)
```

Parameters

`accessToken` string

The OAuth access token.

`athleteId` long

The ID of the athlete.

`page` int

Page number for pagination (default: 1).

`perPage` int

Number of items per page (default: 30).

Returns

Task<JsonArray>

A JsonArray containing the athlete's routes.

Exceptions

ArgumentException

Thrown when `accessToken` is null or empty or `athleteId` is invalid.

ArgumentOutOfRangeException

Thrown when `page` or `perPage` is less than or equal to zero.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the API response is empty.

JsonException

Thrown when the JSON response cannot be parsed as a JsonArray.

Class Segments

Namespace: StravaAPILibary.API

Assembly: StravaAPILibary.dll

Provides methods for interacting with Strava's Segments API.

```
public static class Segments
```

Inheritance

object ← Segments

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Remarks

This static class includes methods to retrieve, explore, and manage segments for the authenticated athlete.

Features:

- Retrieve detailed information about a specific segment by its ID.
- Get a list of starred segments for the authenticated athlete.
- Explore top 10 segments within a defined geographical bounding box.
- Star or unstar a segment for the authenticated athlete.

Usage Example:

```
// Retrieve a segment by ID:
var segment = await Segments.GetSegmentByIdAsync(accessToken, "12345");
Console.WriteLine(segment["name"]);

// Star a segment:
var updatedSegment = await Segments.PutStarredSegmentAsync(accessToken, 12345, true);
```

API Documentation:

Refer to the official Strava API documentation for segments: Strava Segments API Reference.

Methods

GetExploreSegmentsAsync(string, float[], string, int, int)

Get a list of top 10 segments for the specified query.

```
public static Task<JsonArray> GetExploreSegmentsAsync(string accessToken, float[] bounds,
string activityType = "", int minCat = -1, int maxCat = -1)
```

Parameters

`accessToken` string

Strava access token.

`bounds` float[]

Array of four float values [SW lat, SW lon, NE lat, NE lon].

`activityType` string

Activity type: "running" or "riding".

`minCat` int

Minimum climbing category (-1 for any).

`maxCat` int

Maximum climbing category (-1 for any).

Returns

Task<JsonArray>

A JsonArray containing the top 10 segments matching the query.

Exceptions

ArgumentException

Thrown when `accessToken` is null or empty, or when `bounds` is null or does not contain exactly 4 elements.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the API response is empty.

JsonException

Thrown when the JSON response cannot be parsed or does not contain a 'segments' array.

GetSegmentByIdAsync(string, string)

Get a segment by its ID.

```
public static Task<JsonObject> GetSegmentByIdAsync(string accessToken, string segmentId)
```

Parameters

`accessToken` string

The access token for the API.

`segmentId` string

The ID of the segment to retrieve.

Returns

Task<JsonObject>

A JsonObject containing the segment data.

Exceptions

ArgumentException

Thrown when the access token or segment ID is null or empty.

HttpRequestException

Thrown when the request to the API fails.

InvalidOperationException

Thrown when the response is empty.

JsonException

Thrown when the JSON response cannot be parsed.

GetStarredSegmentsAsync(string, int, int)

Get starred segments for the authenticated athlete.

```
public static Task<JsonArray> GetStarredSegmentsAsync(string accessToken, int page = 1, int
perPage = 30)
```

Parameters

`accessToken` string

The OAuth access token.

`page` int

Page number (default: 1).

`perPage` int

Number of items per page (default: 30).

Returns

Task <JsonArray>

A JsonArray containing the starred segments.

Exceptions

ArgumentException

Thrown when `accessToken` is null or empty.

ArgumentOutOfRangeException

Thrown when `page` or `perPage` is less than or equal to zero.

HttpRequestException

Thrown when the API request fails or returns a non-success status code.

InvalidOperationException

Thrown when the API response is empty.

JsonException

Thrown when the response JSON cannot be parsed as a JsonArray.

PutStarredSegmentAsync(string, long, bool)

Stars or unstars a segment for the authenticated athlete.

```
public static Task<JsonObject> PutStarredSegmentAsync(string accessToken, long segmentId,
bool starred)
```

Parameters

`accessToken` string

The OAuth access token.

`segmentId` long

The ID of the segment to star or unstar.

`starred` bool

True to star the segment; false to unstar.

Returns

Task<JsonObject>

A JsonObject containing the updated segment data.

Exceptions

ArgumentException

Thrown when `accessToken` is null or empty, or when `segmentId` is invalid.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the API response is empty.

JsonException

Thrown when the API response cannot be parsed.

Class SegmentsEfforts

Namespace: StravaAPILibary.API

Assembly: StravaAPILibary.dll

Provides methods for interacting with Strava's Segment Efforts API.

```
public static class SegmentsEfforts
```

Inheritance

object ← SegmentsEfforts

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Remarks

This static class includes methods to retrieve segment efforts for the authenticated athlete. Some endpoints require a Strava subscription and the `activity:read` OAuth scope.

Features:

- Retrieve a list of segment efforts for the authenticated athlete and a given segment.
- Retrieve details of a specific segment effort by its ID.

Usage Example:

```
// Get all segment efforts for a segment (requires subscription):
var efforts = await SegmentsEfforts.GetSegmentEffortsAsync(accessToken, 123456,
DateTime.Parse("2024-01-01"), DateTime.Parse("2024-02-01"));

// Get details of a specific segment effort:
var effort = await SegmentsEfforts.GetSegmentEffortByIdAsync(accessToken, 987654);
Console.WriteLine(effort["elapsed_time"]);
```

API Documentation:

Refer to the official Strava API documentation for segment efforts: Strava Segment Efforts API Reference .

Methods

GetSegmentEffortByIdAsync(string, long)

Get a specific segment effort by its ID.

```
public static Task<JsonObject> GetSegmentEffortByIdAsync(string accessToken,
long segmentEffortId)
```

Parameters

`accessToken` string

The OAuth access token.

`segmentEffortId` long

The ID of the segment effort.

Returns

Task<JsonObject>

A JsonObject containing the segment effort details.

Exceptions

ArgumentException

Thrown when `accessToken` is null or empty, or when `segmentEffortId` is invalid.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the API response is empty.

JsonException

Thrown when the API response cannot be parsed as JSON.

GetSegmentEffortsAsync(string, long, DateTime?, DateTime?, int)

Returns a set of the authenticated athlete's segment efforts for a given segment. Requires subscription and scope: activity:read.

```
public static Task<JsonArray> GetSegmentEffortsAsync(string accessToken, long segmentId,
DateTime? startDateLocal = null, DateTime? endDateLocal = null, int perPage = 30)
```

Parameters

`accessToken` string

The OAuth access token.

`segmentId` long

The ID of the segment.

`startDateLocal` DateTime?

Optional: ISO 8601 formatted start date (e.g. "2024-01-01T00:00:00Z").

`endDateLocal` DateTime?

Optional: ISO 8601 formatted end date (e.g. "2024-02-01T00:00:00Z").

`perPage` int

Number of items per page. Default: 30.

Returns

Task<JsonArray>

A JsonArray containing the list of segment efforts.

Exceptions

ArgumentException

Thrown if `accessToken` is null or empty, or `segmentId` is invalid.

HttpRequestException

Thrown when the API request fails.

InvalidOperationException

Thrown when the response is empty.

JsonException

Thrown when the response cannot be parsed.

Class Streams

Namespace: StravaAPILibary.API

Assembly: StravaAPILibary.dll

Provides methods for retrieving stream data (detailed time-series data) from Strava.

```
public static class Streams
```

Inheritance

object ← Streams

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Remarks

Streams represent detailed data points over time or distance, such as GPS coordinates, altitude, speed, heart rate, cadence, power, and more.

Features:

- Retrieve streams for a specific activity.
- Retrieve streams for a specific segment.
- Retrieve streams for a specific route.
- Retrieve streams for a specific segment effort.

Usage Example:

```
// Example: Get lat/lng and altitude streams for an activity
var streams = await Streams.GetActivityStreamsAsync(accessToken, 123456789,
"latlng,altitude", true);
Console.WriteLine(streams["latlng"]);
```

API Documentation:

Refer to the official Strava API documentation for streams: Strava Streams API Reference.

Methods

GetActivityStreamsAsync(string, long, string?, bool)

Get streams for a specific activity.

```
public static Task<JsonObject> GetActivityStreamsAsync(string accessToken, long activityId,
string? keys = null, bool keyByType = false)
```

Parameters

`accessToken` string

The OAuth access token.

`activityId` long

The ID of the activity.

`keys` string

Optional: Comma-separated list of stream types to retrieve (e.g., "latlng,altitude").

`keyByType` bool

Optional: If true, streams will be returned as a map keyed by type.

Returns

Task<JsonObject>

A JsonObject containing the streams data.

Exceptions

ArgumentException

Thrown when `accessToken` is null or empty or `activityId` is invalid.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the API response is empty.

JsonException

Thrown when the JSON response cannot be parsed as a JsonObject.

GetRouteStreamsAsync(string, long, string?, bool)

Get streams for a specific route.

```
public static Task<JsonObject> GetRouteStreamsAsync(string accessToken, long routeId,
string? keys = null, bool keyByType = false)
```

Parameters

`accessToken` string

The OAuth access token.

`routeId` long

The ID of the route.

`keys` string

Optional: Comma-separated list of stream keys (e.g., "distance,altitude,latlng").

`keyByType` bool

Optional: If true, streams will be returned as a map keyed by type.

Returns

Task<JsonObject>

A JsonObject containing the route streams.

Exceptions

ArgumentException

Thrown when `accessToken` is null or empty or `routeId` is invalid.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the API response is empty.

JsonException

Thrown when the JSON response cannot be parsed as a JsonObject.

GetSegmentEffortStreamsAsync(string, long, string?, bool)

Get streams for a specific segment effort.

```
public static Task<JsonObject> GetSegmentEffortStreamsAsync(string accessToken, long
segmentEffortId, string? keys = null, bool keyByType = false)
```

Parameters

`accessToken` string

The OAuth access token.

`segmentEffortId` long

The ID of the segment effort.

`keys` string

Optional: Comma-separated list of stream keys (e.g., "distance,time,latlng").

`keyByType` bool

Optional: If true, streams will be returned as a map keyed by type.

Returns

Task<JsonObject>

A JsonObject containing the segment effort streams.

Exceptions

ArgumentException

Thrown when `accessToken` is null or empty or `segmentEffortId` is invalid.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the API response is empty.

JsonException

Thrown when the JSON response cannot be parsed as a JsonObject.

GetSegmentStreamsAsync(string, long, string?, bool)

Get streams for a specific segment.

```
public static Task<JsonObject> GetSegmentStreamsAsync(string accessToken, long segmentId,
string? keys = null, bool keyByType = false)
```

Parameters

`accessToken` string

The OAuth access token.

`segmentId` long

The ID of the segment.

`keys` string

Optional: Comma-separated list of stream keys (e.g., "distance,altitude,latlng").

`keyByType` bool

Optional: If true, streams will be returned as a map keyed by type.

Returns

Task<JsonObject>

A JsonObject containing the segment streams.

Exceptions

ArgumentException

Thrown when `accessToken` is null or empty or `segmentId` is invalid.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the API response is empty.

JsonException

Thrown when the JSON response cannot be parsed as a JsonObject.

Class Uploads

Namespace: StravaAPILibary.API

Assembly: StravaAPILibary.dll

Provides methods for uploading activities and retrieving upload statuses from the Strava API.

```
public static class Uploads
```

Inheritance

object ← Uploads

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Remarks

The `Uploads` class supports two main operations:

- Uploading activity files (FIT, GPX, TCX) to Strava.
- Checking the processing status of previously uploaded activity files.

Important Notes:

- Uploads may take a while to process. You should poll the upload status using GetUploadAsync(string, long).
- Uploading activities requires an authenticated athlete's access token with write permissions.

Example Usage:

```
// Upload an activity file
var uploadResponse = await Uploads.UploadActivityAsync(accessToken, "activity.fit", "fit",
"Morning Ride", "Great ride", false, false);
Console.WriteLine(uploadResponse["id"]);

// Check upload status
var uploadStatus = await Uploads.GetUploadAsync(accessToken, (long)uploadResponse["id"]);
Console.WriteLine(uploadStatus["status"]);
```

API Reference:

See: Strava Uploads API Documentation.

Methods

GetUploadAsync(string, long)

Get an upload status by its ID.

```
public static Task<JsonObject> GetUploadAsync(string accessToken, long uploadId)
```

Parameters

`accessToken` string

The OAuth access token.

`uploadId` long

The ID of the upload to retrieve.

Returns

Task<JsonObject>

A JsonObject containing the upload status and details.

Exceptions

ArgumentException

Thrown when `accessToken` is null or empty or when `uploadId` is invalid.

HttpRequestException

Thrown when the request to the Strava API fails or returns a non-success status code.

InvalidOperationException

Thrown when the API response is empty.

JsonException

Thrown when the JSON response cannot be parsed as a JsonObject.

UploadActivityAsync(string, string, string, string?, string?, bool, bool)

Uploads an activity file to Strava.

```
public static Task<JsonObject> UploadActivityAsync(string accessToken, string filePath,
string dataType, string? name = null, string? description = null, bool isTrainer = false,
bool isCommute = false)
```

Parameters

`accessToken` string

The OAuth access token.

`filePath` string

The path to the activity file (FIT, GPX, TCX).

`dataType` string

The file type: "fit", "gpx", or "tcx".

`name` string

Optional: The name of the activity.

`description` string

Optional: A description of the activity.

`isTrainer` bool

Optional: Set to true if this is a trainer ride.

`isCommute` bool

Optional: Set to true if this is a commute.

Returns

Task<JsonObject>

A JsonObject containing the upload status.

Exceptions

ArgumentException

Thrown when `accessToken` or `filePath` is invalid.

FileNotFoundException

Thrown when the specified file does not exist.

HttpRequestException

Thrown when the upload request fails.

Namespace StravaAPILibary.Authentication

Classes

Credentials

Represents the credentials required for authenticating with the Strava API.

UserAuthentication

Handles the OAuth 2.0 authentication flow for Strava users, including authorization, token exchange, and token refresh.

Class Credentials

Namespace: StravaAPILibary.Authentication

Assembly: StravaAPILibary.dll

Represents the credentials required for authenticating with the Strava API.

```
public class Credentials
```

Inheritance

object ← Credentials

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Examples

Complete authentication flow with credentials:

```
public class StravaClient
{
    private readonly Credentials _credentials;

    public StravaClient(string clientId, string clientSecret, string scope)
    {
        _credentials = new Credentials(clientId, clientSecret, scope);
    }

    public async Task<string> AuthenticateAsync()
    {
        var userAuth = new UserAuthentication(_credentials,
"http://localhost:8080/callback", _credentials.Scope);

        userAuth.StartAuthorization();
        string authCode = await userAuth.WaitForAuthCodeAsync();

        bool success = await userAuth.ExchangeCodeForTokenAsync(authCode);
        if (success)
        {
            return _credentials.AccessToken;
        }
```

```
            throw new InvalidOperationException("Authentication failed.");
        }

        public bool IsTokenValid()
        {
            return !string.IsNullOrEmpty(_credentials.AccessToken) &&
                   _credentials.TokenExpiration > DateTime.UtcNow.AddMinutes(5);
        }
    }
```

Remarks

The `Credentials` class encapsulates all the necessary information for OAuth 2.0 authentication with the Strava API. It stores both the application credentials (client ID and secret) and the user's OAuth tokens (access token and refresh token).

Application Credentials:

- `ClientId` - Unique identifier for your Strava application
- `ClientSecret` - Secret key for your Strava application (keep secure)
- `Scope` - Requested permissions for the application

OAuth Tokens:

- `AccessToken` - Used for API requests (expires in 6 hours)
- `RefreshToken` - Used to get new access tokens (no expiration)
- `TokenExpiration` - When the access token expires

Security Considerations:

- Never expose the client secret in client-side code
- Store tokens securely (environment variables, secure storage)
- Request minimal scopes needed for your application
- Implement proper token refresh logic

Usage Example:

```
// Create credentials for a new application
var credentials = new Credentials("your_client_id", "your_client_secret",
"read,activity:read_all");

// After OAuth flow, tokens will be populated
Console.WriteLine($"Access Token: {credentials.AccessToken}");
Console.WriteLine($"Refresh Token: {credentials.RefreshToken}");
```

```
Console.WriteLine($"Expires: {credentials.TokenExpiration}");

// Check if token is expired
bool isExpired = credentials.TokenExpiration <= DateTime.UtcNow;
```

Available Scopes:

- read - Basic profile access
- activity:read_all - Read all activities
- activity:write - Upload activities
- profile:read_all - Detailed profile access
- profile:write - Update profile information

Constructors

Credentials(string, string, string)

Initializes a new instance of the Credentials class with the specified client ID, client secret, and scope.

```
public Credentials(string clientId, string clientSecret, string scope)
```

Parameters

`clientId` string

The client ID provided by Strava for the application. Must not be null or empty.

`clientSecret` string

The client secret associated with the Strava application. Must not be null or empty.

`scope` string

The scope of access required for API interactions. Multiple scopes can be separated by commas (e.g., read,activity:read_all).

Examples

```
// Basic credentials with minimal scope
var credentials = new Credentials("12345", "your_secret", "read");
```

```
// Credentials with full activity access
var credentials = new Credentials("12345", "your_secret",
"read,activity:read_all,activity:write");

// Load from environment variables (recommended for production)
string clientId = Environment.GetEnvironmentVariable("STRAVA_CLIENT_ID")
    ?? throw new InvalidOperationException("STRAVA_CLIENT_ID not set");
string clientSecret = Environment.GetEnvironmentVariable("STRAVA_CLIENT_SECRET")
    ?? throw new InvalidOperationException("STRAVA_CLIENT_SECRET not set");
var credentials = new Credentials(clientId, clientSecret, "read,activity:read_all");
```

Remarks

This constructor initializes a new credentials object with the application's client credentials and requested scope. The OAuth tokens (access token, refresh token, expiration) will be populated after a successful OAuth flow.

Parameter Validation:

- Client ID must be a valid Strava application ID
- Client secret must match the client ID
- Scope must be valid Strava OAuth scopes

Common Scopes:

- read - Basic profile access (recommended minimum)
- read,activity:read_all - Read profile and activities
- read,activity:read_all,activity:write - Full activity access

Exceptions

ArgumentNullException

Thrown when `clientId`, `clientSecret`, or `scope` is null.

ArgumentException

Thrown when `clientId`, `clientSecret`, or `scope` is empty or whitespace.

Properties

AccessToken

Gets or sets the OAuth access token used for API requests.

```
public string AccessToken { get; set; }
```

Property Value

string

Examples

```
// Use access token for API requests
using var client = new HttpClient();
client.DefaultRequestHeaders.Authorization =
    new AuthenticationHeaderValue("Bearer", credentials.AccessToken);

var response = await client.GetAsync("https://www.strava.com/api/v3/athlete");
```

Remarks

The access token is obtained through the OAuth 2.0 authorization flow and is used to authenticate API requests to Strava. It expires after 6 hours and must be refreshed using the refresh token.

Token Properties:

- Expires after 6 hours
- Used in Authorization header: Bearer {access_token}
- Can be refreshed using RefreshToken
- Should be stored securely

Validation:

Check TokenExpiration to determine if the token is still valid.

ClientId

Gets or sets the client ID provided by Strava for the application.

```
public string ClientId { get; set; }
```

Property Value

string

Examples

```
var credentials = new Credentials("12345", "secret", "read");
Console.WriteLine(credentials.ClientId); // Output: 12345
```

Remarks

The client ID is a unique identifier for your Strava application. It is provided when you create a new application in the Strava API settings.

This value is used in:

- OAuth authorization URL generation
- Token exchange requests
- Token refresh requests

Security:

The client ID is not sensitive and can be safely included in client-side code.

ClientSecret

Gets or sets the client secret associated with the Strava application.

```
public string ClientSecret { get; set; }
```

Property Value

string

Examples

```
// Load from environment variable (recommended)
string clientSecret = Environment.GetEnvironmentVariable("STRAVA_CLIENT_SECRET");
var credentials = new Credentials("12345", clientSecret, "read");
```

Remarks

The client secret is a sensitive credential that should be kept secure. It is used to authenticate your application with Strava during the OAuth token exchange process.

Security Requirements:

- Never expose in client-side code
- Store securely (environment variables, secure storage)
- Don't commit to version control
- Rotate if compromised

Usage:

The client secret is only used server-side during OAuth token exchange and refresh operations.

RefreshToken

Gets or sets the OAuth refresh token used to obtain a new access token when it expires.

```
public string RefreshToken { get; set; }
```

Property Value

string

Examples

```
// Refresh access token when expired
if (credentials.TokenExpiration <= DateTime.UtcNow.AddMinutes(5))
{
    var userAuth = new UserAuthentication(credentials, redirectUri, scope);
    bool success = await userAuth.RefreshAccessTokenAsync();

    if (success)
    {
        // New access token is now available
        Console.WriteLine("Token refreshed successfully");
    }
}
```

Remarks

The refresh token is obtained during the initial OAuth authorization and can be used to get new access tokens without requiring user re-authorization. Refresh tokens do not expire unless revoked.

Token Properties:

- No expiration (until revoked by user)
- Used only for token refresh operations
- Should be stored securely alongside access token
- Can be revoked by user in Strava settings

Usage:

Use this token with `UserAuthentication.RefreshAccessTokenAsync()` to get a new access token.

Scope

Gets or sets the scope of access for the token.

```
public string Scope { get; set; }
```

Property Value

string

Examples

```
// Minimal scope for basic functionality
var credentials = new Credentials("client_id", "secret", "read");

// Full scope for complete functionality
var credentials = new Credentials("client_id", "secret",
"read,activity:read_all,activity:write");

// Check if specific scope is granted
bool hasActivityRead = credentials.Scope.Contains("activity:read_all");
```

Remarks

The scope defines what permissions your application has been granted by the user. Multiple scopes can be combined using commas.

Available Scopes:

- read - Basic profile access (always included)
- activity:read_all - Read all activities
- activity:write - Upload activities
- profile:read_all - Detailed profile access
- profile:write - Update profile information

Best Practices:

- Request only the scopes you need
- Explain to users why you need each scope
- Handle cases where users deny certain scopes

TokenExpiration

Gets or sets the expiration date and time of the current access token.

```
public DateTime TokenExpiration { get; set; }
```

Property Value

DateTime

Examples

```
// Check if token is expired
bool isExpired = credentials.TokenExpiration <= DateTime.UtcNow;

// Check if token expires soon (within 5 minutes)
bool expiresSoon = credentials.TokenExpiration <= DateTime.UtcNow.AddMinutes(5);

// Calculate time until expiration
TimeSpan timeUntilExpiration = credentials.TokenExpiration - DateTime.UtcNow;
Console.WriteLine($"Token expires in {timeUntilExpiration.TotalMinutes:F0} minutes");
```

Remarks

This property indicates when the current access token will expire. Access tokens expire after 6 hours from the time they were issued. After expiration, you must use the refresh token to get a new access token.

Token Lifecycle:

1. Token is issued (expires in 6 hours)
2. Use token for API requests
3. Check expiration before making requests
4. Refresh token when close to expiration

Validation:

Check if token is expired: TokenExpiration <= DateTime.UtcNow Check if token expires soon: TokenExpiration <= DateTime.UtcNow.AddMinutes(5)

Class UserAuthentication

Namespace: StravaAPILibary.Authentication

Assembly: StravaAPILibary.dll

Handles the OAuth 2.0 authentication flow for Strava users, including authorization, token exchange, and token refresh.

```
public class UserAuthentication
```

Inheritance

object ← UserAuthentication

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Examples

Complete authentication flow with error handling:

```
public async Task<string> AuthenticateAsync()
{
    var credentials = new Credentials("client_id", "client_secret",
"read,activity:read_all");
    var userAuth = new UserAuthentication(credentials, "http://localhost:8080/callback",
"read,activity:read_all");

    try
    {
        userAuth.StartAuthorization();
        string authCode = await userAuth.WaitForAuthCodeAsync();

        bool success = await userAuth.ExchangeCodeForTokenAsync(authCode);
        if (success)
        {
            return credentials.AccessToken;
        }
        else
        {
            throw new InvalidOperationException("Failed to exchange authorization code
for tokens.");
```

```
        }
    }
    catch (TimeoutException)
    {
        throw new InvalidOperationException("Authorization timed out. Please try again.");
    }
    catch (Exception ex)
    {
        throw new InvalidOperationException($"Authentication failed: {ex.Message}");
    }
}
```

Remarks

The `UserAuthentication` class implements the complete OAuth 2.0 authorization code flow for Strava API integration.

OAuth 2.0 Flow:

1. User is redirected to Strava's authorization page
2. User authorizes the application and grants requested scopes
3. Strava redirects back with an authorization code
4. Application exchanges the code for access and refresh tokens
5. Access token is used for API requests (expires in 6 hours)
6. Refresh token is used to get new access tokens when needed

Usage Example:

```
// Create credentials
var credentials = new Credentials("your_client_id", "your_client_secret",
"read,activity:read_all");

// Initialize authentication
var userAuth = new UserAuthentication(credentials, "http://localhost:8080/callback",
"read,activity:read_all");

// Start authorization process
userAuth.StartAuthorization();

// Wait for user to complete authorization
string authCode = await userAuth.WaitForAuthCodeAsync();

// Exchange code for tokens
bool success = await userAuth.ExchangeCodeForTokenAsync(authCode);
```

```
if (success)
{
    string accessToken = credentials.AccessToken;
    // Use accessToken for API calls
}
```

Token Management:

- Access tokens expire after 6 hours
- Refresh tokens have no expiration (until revoked)
- Use RefreshAccessTokenAsync() to get new access tokens
- Store tokens securely (environment variables, secure storage)

Security Considerations:

- Never expose client secret in client-side code
- Use HTTPS for redirect URIs in production
- Request minimal scopes needed for your application
- Implement proper error handling for token refresh failures

API Documentation:

Refer to the official Strava API documentation for authentication: Strava Authentication Guide.

Constructors

UserAuthentication(Credentials, string, string)

Initializes a new instance of the UserAuthentication class with the specified credentials and configuration.

```
public UserAuthentication(Credentials creds, string redirectUri, string scope)
```

Parameters

`creds` Credentials

The Strava API credentials containing client ID and secret. Must not be null.

`redirectUri` string

The redirect URI registered for the Strava application. Must match the URI configured in your Strava app settings.

`scope` string

The requested OAuth scope(s). Multiple scopes can be separated by commas (e.g., `read,activity:read_all`).

Remarks

The redirect URI must exactly match what is configured in your Strava application settings.

Common redirect URIs for development:

- `http://localhost:8080/callback`
- `http://localhost:3000/callback`
- `http://localhost:5000/callback`

Exceptions

ArgumentNullException

Thrown when `creds`, `redirectUri`, or `scope` is null.

Methods

BuildAuthorizationUrl()

Builds the Strava OAuth authorization URL with the configured parameters.

```
public string BuildAuthorizationUrl()
```

Returns

string

The complete authorization URL for redirecting the user to Strava's authorization page.

Examples

```
var userAuth = new UserAuthentication(credentials, "http://localhost:8080/callback",
"read,activity:read_all");
string authUrl = userAuth.BuildAuthorizationUrl();
Console.WriteLine(authUrl);
// Output: https://www.strava.com/oauth/authorize?
```

```
client_id=12345&redirect_uri=http%3A%2F%2Flocalhost%3A8080%2Fcallback&response_type=code&sco
pe=read%2Cactivity%3Aread_all&approval_prompt=force
```

Remarks

The generated URL includes:

- Client ID from credentials
- Redirect URI (URL-encoded)
- Response type set to "code"
- Requested scope (URL-encoded)
- Approval prompt set to "force" to always show authorization page

ExchangeCodeForTokenAsync(string)

Exchanges an authorization code for an access token and refresh token.

```
public Task<bool> ExchangeCodeForTokenAsync(string authCode)
```

Parameters

`authCode` string

The authorization code received from the OAuth redirect.

Returns

Task<bool>

`true` if the token exchange was successful; otherwise, `false`.

Examples

```
string authCode = "received_authorization_code";
bool success = await userAuth.ExchangeCodeForTokenAsync(authCode);

if (success)
{
    Console.WriteLine($"Access Token: {credentials.AccessToken}");
    Console.WriteLine($"Refresh Token: {credentials.RefreshToken}");
    Console.WriteLine($"Expires: {credentials.TokenExpiration}");
```

```
}
else
{
    Console.WriteLine("Token exchange failed.");
}
```

Remarks

This method performs the OAuth 2.0 token exchange by sending a POST request to Strava's token endpoint. If successful, the access token, refresh token, and expiration time are stored in the credentials object.

Token Exchange Process:

1. Send POST request to `https://www.strava.com/oauth/token`
2. Include client ID, client secret, authorization code, and grant type
3. Parse response to extract tokens and expiration
4. Store tokens in the credentials object

Response Fields:

- `access_token` - Used for API requests (expires in 6 hours)
- `refresh_token` - Used to get new access tokens
- `expires_at` - Unix timestamp when access token expires
- `expires_in` - Seconds until access token expires

Exceptions

ArgumentException

Thrown when `authCode` is null or empty.

HttpRequestException

Thrown when the token exchange request fails or returns a non-success status code.

JsonException

Thrown when the response JSON cannot be parsed.

OpenBrowser(string)

Opens the specified URL in the default browser.

```
public void OpenBrowser(string url)
```

Parameters

`url` string

The URL to open in the browser.

Remarks

This method uses the system's default browser to open the provided URL. On Windows, it uses `Process.Start` with `UseShellExecute = true`.

RefreshAccessTokenAsync()

Refreshes the access token using the refresh token.

```
public Task<bool> RefreshAccessTokenAsync()
```

Returns

Task<bool>

`true` if the token refresh was successful; otherwise, `false`.

Examples

```
// Check if token needs refresh
if (credentials.TokenExpiration <= DateTime.UtcNow.AddMinutes(5))
{
    bool refreshSuccess = await userAuth.RefreshAccessTokenAsync();
    if (refreshSuccess)
    {
        Console.WriteLine("Token refreshed successfully.");
    }
    else
    {
        Console.WriteLine("Token refresh failed. User needs to re-authenticate.");
    }
}
```

Remarks

This method uses the refresh token to obtain a new access token without requiring user re-authorization. The new access token and expiration time are stored in the credentials object.

When to Use:

- Access token has expired (6 hours)
- Access token will expire soon (within 5 minutes)
- API calls return 401 Unauthorized errors

Error Handling:

If the refresh token is invalid or has been revoked, this method will return false. In such cases, the user will need to re-authenticate using the full OAuth flow.

Exceptions

InvalidOperationException

Thrown when no refresh token is available.

HttpRequestException

Thrown when the refresh request fails or returns a non-success status code.

JsonException

Thrown when the response JSON cannot be parsed.

StartAuthorization()

Starts the authorization process by opening the browser with the Strava OAuth URL.

```
public void StartAuthorization()
```

Examples

```
var userAuth = new UserAuthentication(credentials, "http://localhost:8080/callback",
"read,activity:read_all");
userAuth.StartAuthorization();

// Option 1: Wait for automatic code capture
```

```
string authCode = await userAuth.WaitForAuthCodeAsync();

// Option 2: Handle redirect manually
// User will be redirected to http://localhost:8080/callback?code=YOUR_AUTH_CODE
// Extract the code parameter from the URL
```

Remarks

This method opens the user's default browser and navigates to Strava's authorization page. The user will be prompted to log in to Strava (if not already logged in) and authorize your application.

After authorization, Strava will redirect the user back to your redirect URI with an authorization code.

Use WaitForAuthCodeAsync() to automatically capture the authorization code, or handle the redirect manually.

WaitForAuthCodeAsync()

Waits for the user to authorize the application and returns the authorization code.

```
public Task<string> WaitForAuthCodeAsync()
```

Returns

Task<string>

The authorization code from the OAuth redirect.

Examples

```
try
{
    userAuth.StartAuthorization();
    string authCode = await userAuth.WaitForAuthCodeAsync();
    Console.WriteLine($"Authorization code: {authCode}");
}
catch (TimeoutException)
{
    Console.WriteLine("Authorization timed out. Please try again.");
}
```

Remarks

This method starts an HTTP listener on the redirect URI port and waits for the OAuth callback. It automatically handles the redirect response and extracts the authorization code.

Important:

- The redirect URI must be accessible (localhost for development)
- Port must be available and not blocked by firewall
- Method times out after 5 minutes
- Returns a success page to the user's browser

Alternative:

For production applications, consider handling the redirect manually instead of using this method.

Exceptions

InvalidOperationException

Thrown when the redirect URI is not set or the authorization code is missing in the response.

TimeoutException

Thrown if no authorization is received within 5 minutes.

Namespace StravaAPILibary.Models

Classes

ActivityStats

Represents rolled-up statistics and totals for an athlete, including recent, year-to-date, and all-time metrics for rides, runs, and swims.

ActivityTotal

A roll-up of metrics pertaining to a set of activities. Values are in seconds and meters.

ActivityZone

Represents an activity zone, such as heart rate or power, including its score, distribution, and related data.

Comment

Represents a comment made on an activity.

Error

Represents an error returned from the Strava API.

Fault

Represents a fault response from the Strava API, including a message and associated errors.

TimedZoneDistribution

Represents a collection of exclusive ranges (zones) and the time spent in each.

TimedZoneRange

Represents the time spent in a specific zone range.

Class ActivityStats

Namespace: StravaAPILibary.Models

Assembly: StravaAPILibary.dll

Represents rolled-up statistics and totals for an athlete, including recent, year-to-date, and all-time metrics for rides, runs, and swims.

```
public class ActivityStats
```

Inheritance

object ← ActivityStats

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

ActivityStats()

Initializes a new instance of the ActivityStats class with default values for all activity totals.

```
public ActivityStats()
```

Properties

AllRideTotals

The all-time ride statistics for the athlete.

```
[JsonPropertyName("all_ride_totals")]
public ActivityTotal AllRideTotals { get; set; }
```

Property Value

ActivityTotal

AllRunTotals

The all-time run statistics for the athlete.

```
[JsonPropertyName("all_run_totals")]
public ActivityTotal AllRunTotals { get; set; }
```

Property Value

ActivityTotal

AllSwimTotals

The all-time swim statistics for the athlete.

```
[JsonPropertyName("all_swim_totals")]
public ActivityTotal AllSwimTotals { get; set; }
```

Property Value

ActivityTotal

BiggestClimbElevationGain

The highest climb elevation gain achieved by the athlete, in meters.

```
[JsonPropertyName("biggest_climb_elevation_gain")]
public double BiggestClimbElevationGain { get; set; }
```

Property Value

double

BiggestRideDistance

The longest distance ridden by the athlete, in meters.

```
[JsonPropertyName("biggest_ride_distance")]
public double BiggestRideDistance { get; set; }
```

Property Value

double

RecentRideTotals

The recent (last 4 weeks) ride statistics for the athlete.

```
[JsonPropertyName("recent_ride_totals")]
public ActivityTotal RecentRideTotals { get; set; }
```

Property Value

ActivityTotal

RecentRunTotals

The recent (last 4 weeks) run statistics for the athlete.

```
[JsonPropertyName("recent_run_totals")]
public ActivityTotal RecentRunTotals { get; set; }
```

Property Value

ActivityTotal

RecentSwimTotals

The recent (last 4 weeks) swim statistics for the athlete.

```
[JsonPropertyName("recent_swim_totals")]
public ActivityTotal RecentSwimTotals { get; set; }
```

Property Value

ActivityTotal

YtdRideTotals

The year-to-date ride statistics for the athlete.

```
[JsonPropertyName("ytd_ride_totals")]
public ActivityTotal YtdRideTotals { get; set; }
```

Property Value

ActivityTotal

YtdRunTotals

The year-to-date run statistics for the athlete.

```
[JsonPropertyName("ytd_run_totals")]
public ActivityTotal YtdRunTotals { get; set; }
```

Property Value

ActivityTotal

YtdSwimTotals

The year-to-date swim statistics for the athlete.

```
[JsonPropertyName("ytd_swim_totals")]
public ActivityTotal YtdSwimTotals { get; set; }
```

Property Value

ActivityTotal

Class ActivityTotal

Namespace: StravaAPILibary.Models

Assembly: StravaAPILibary.dll

A roll-up of metrics pertaining to a set of activities. Values are in seconds and meters.

```
public class ActivityTotal
```

Inheritance

object ← ActivityTotal

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

AchievementCount

The total number of achievements in the considered activities.

```
[JsonPropertyName("achievement_count")]
public int AchievementCount { get; set; }
```

Property Value

int

Count

The number of activities considered in this total.

```
[JsonPropertyName("count")]
public int Count { get; set; }
```

Property Value

int

Distance

The total distance covered by the considered activities (in meters).

```
[JsonPropertyName("distance")]
public float Distance { get; set; }
```

Property Value

float

ElapsedTime

The total elapsed time of the considered activities (in seconds).

```
[JsonPropertyName("elapsed_time")]
public int ElapsedTime { get; set; }
```

Property Value

int

ElevationGain

The total elevation gain of the considered activities (in meters).

```
[JsonPropertyName("elevation_gain")]
public float ElevationGain { get; set; }
```

Property Value

float

MovingTime

The total moving time of the considered activities (in seconds).

```
[JsonPropertyName("moving_time")]
public int MovingTime { get; set; }
```

Property Value

int

Class ActivityZone

Namespace: StravaAPILibary.Models

Assembly: StravaAPILibary.dll

Represents an activity zone, such as heart rate or power, including its score, distribution, and related data.

```
public class ActivityZone
```

Inheritance

object ← ActivityZone

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

CustomZones

Indicates if custom zones were used.

```
public bool CustomZones { get; set; }
```

Property Value

bool

DistributionBuckets

The time distribution buckets for this zone. Each bucket represents the time spent in a specific range.

```
public List<TimedZoneDistribution> DistributionBuckets { get; set; }
```

Property Value

List<TimedZoneDistribution>

Max

The maximum value recorded in this zone.

```
public int Max { get; set; }
```

Property Value

int

Points

The number of points in this zone.

```
public int Points { get; set; }
```

Property Value

int

Score

The score achieved in this zone.

```
public int Score { get; set; }
```

Property Value

int

SensorBased

Indicates whether the zone data is sensor-based.

```
public bool SensorBased { get; set; }
```

Property Value

bool

Type

The type of the activity zone. Valid values: "heartrate" or "power".

```
public string Type { get; set; }
```

Property Value

string

Class Comment

Namespace: StravaAPILibary.Models

Assembly: StravaAPILibary.dll

Represents a comment made on an activity.

```
public class Comment
```

Inheritance

object ← Comment

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

ActivityId

The identifier of the activity this comment is related to.

```
public long ActivityId { get; set; }
```

Property Value

long

Athlete

The athlete who made the comment.

```
public SummaryAthlete Athlete { get; set; }
```

Property Value

SummaryAthlete

CreatedAt

The time at which this comment was created.

```
public DateTime CreatedAt { get; set; }
```

Property Value

DateTime

Id

The unique identifier of this comment.

```
public long Id { get; set; }
```

Property Value

long

Text

The content of the comment.

```
public string Text { get; set; }
```

Property Value

string

Class Error

Namespace: StravaAPILibary.Models

Assembly: StravaAPILibary.dll

Represents an error returned from the Strava API.

```
public class Error
```

Inheritance

object ← Error

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Code

The code associated with this error.

```
public string Code { get; set; }
```

Property Value

string

Field

The specific field or aspect of the resource associated with this error.

```
public string Field { get; set; }
```

Property Value

string

Resource

The type of resource associated with this error.

```
public string Resource { get; set; }
```

Property Value

string

Class Fault

Namespace: StravaAPILibary.Models

Assembly: StravaAPILibary.dll

Represents a fault response from the Strava API, including a message and associated errors.

```
public class Fault
```

Inheritance

object ← Fault

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Errors

The set of specific errors associated with this fault, if any.

```
public List<Error> Errors { get; set; }
```

Property Value

List<Error>

Message

The message of the fault.

```
public string Message { get; set; }
```

Property Value

string

Class TimedZoneDistribution

Namespace: StravaAPILibary.Models

Assembly: StravaAPILibary.dll

Represents a collection of exclusive ranges (zones) and the time spent in each.

```
public class TimedZoneDistribution
```

Inheritance

object ← TimedZoneDistribution

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Zones

The list of timed zone ranges.

```
public List<TimedZoneRange> Zones { get; set; }
```

Property Value

List<TimedZoneRange>

Class TimedZoneRange

Namespace: StravaAPILibary.Models

Assembly: StravaAPILibary.dll

Represents the time spent in a specific zone range.

```
public class TimedZoneRange
```

Inheritance

object ← TimedZoneRange

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Max

The maximum value in the range.

```
public int Max { get; set; }
```

Property Value

int

Min

The minimum value in the range.

```
public int Min { get; set; }
```

Property Value

int

Time

The number of seconds spent in this zone.

```
public int Time { get; set; }
```

Property Value

int

Namespace StravaAPILibary.Models.Activities

Classes

DetailedActivity

Represents a detailed Strava activity with extensive metadata, statistics, and related objects.

Lap

Represents a single lap within an activity.

MetaActivity

Represents a minimal reference to an activity.

Split

Represents a split of an activity, typically used for running activities.

SummaryActivity

Represents a summarized Strava activity with key performance and metadata fields.

UpdatableActivity

Represents an updatable activity in Strava.

Class DetailedActivity

Namespace: StravaAPILibary.Models.Activities

Assembly: StravaAPILibary.dll

Represents a detailed Strava activity with extensive metadata, statistics, and related objects.

```
public class DetailedActivity
```

Inheritance

object ← DetailedActivity

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

AchievementCount

The number of achievements gained during this activity.

```
public int AchievementCount { get; set; }
```

Property Value

int

Athlete

The athlete associated with this activity.

```
public MetaAthlete Athlete { get; set; }
```

Property Value

MetaAthlete

AthleteCount

The number of athletes participating in this activity.

```
public int AthleteCount { get; set; }
```

Property Value

int

AverageSpeed

The average speed in meters per second.

```
public float AverageSpeed { get; set; }
```

Property Value

float

AverageWatts

Average power output in watts (rides only).

```
public float? AverageWatts { get; set; }
```

Property Value

float?

BestEfforts

Best efforts during this activity.

```
public List<DetailedSegmentEffort>? BestEfforts { get; set; }
```

Property Value

List <DetailedSegmentEffort>

Calories

Calories burned during the activity.

```
public float? Calories { get; set; }
```

Property Value

float?

CommentCount

The number of comments for this activity.

```
public int CommentCount { get; set; }
```

Property Value

int

Commute

Whether this activity is a commute.

```
public bool Commute { get; set; }
```

Property Value

bool

Description

Description of the activity.

```
public string? Description { get; set; }
```

Property Value

string

DeviceName

The device used to record this activity.

```
public string? DeviceName { get; set; }
```

Property Value

string

DeviceWatts

Whether the watts are from a power meter (false if estimated).

```
public bool? DeviceWatts { get; set; }
```

Property Value

bool?

Distance

The activity's distance in meters.

```
public float Distance { get; set; }
```

Property Value

float

ElapsedTime

The activity's elapsed time in seconds.

```
public int ElapsedTime { get; set; }
```

Property Value

int

ElevHigh

The activity's highest elevation in meters.

```
public float ElevHigh { get; set; }
```

Property Value

float

ElevLow

The activity's lowest elevation in meters.

```
public float ElevLow { get; set; }
```

Property Value

float

EmbedToken

The token used to embed this activity.

```
public string? EmbedToken { get; set; }
```

Property Value

string

EndLatLng

The end latitude/longitude of the activity.

```
public LatLng? EndLatLng { get; set; }
```

Property Value

LatLng

ExternalId

The identifier provided at upload time.

```
public string? ExternalId { get; set; }
```

Property Value

string

Flagged

Whether this activity is flagged.

```
public bool Flagged { get; set; }
```

Property Value

bool

Gear

Gear summary used during the activity.

```
public SummaryGear? Gear { get; set; }
```

Property Value

SummaryGear

GearId

The gear ID used for this activity.

```
public string? GearId { get; set; }
```

Property Value

string

HasKudoed

Whether the logged-in athlete has kudoed this activity.

```
public bool HasKudoed { get; set; }
```

Property Value

bool

HideFromHome

Whether this activity is muted in the home feed.

```
public bool HideFromHome { get; set; }
```

Property Value

bool

Id

The unique identifier of the activity.

```
public long Id { get; set; }
```

Property Value

long

Kilojoules

The total work done in kilojoules during this activity (rides only).

```
public float? Kilojoules { get; set; }
```

Property Value

float?

KudosCount

The number of kudos given for this activity.

```
public int KudosCount { get; set; }
```

Property Value

int

Laps

Laps of the activity.

```
public List<Lap>? Laps { get; set; }
```

Property Value

List<Lap>

Manual

Whether this activity was created manually.

```
public bool Manual { get; set; }
```

Property Value

bool

Map

The map associated with this activity.

```
public PolylineMap Map { get; set; }
```

Property Value

PolylineMap

MaxSpeed

The maximum speed in meters per second.

```
public float MaxSpeed { get; set; }
```

Property Value

float

MaxWatts

Maximum power in watts (rides with power meter only).

```
public int? MaxWatts { get; set; }
```

Property Value

int?

MovingTime

The activity's moving time in seconds.

```
public int MovingTime { get; set; }
```

Property Value

int

Name

The name of the activity.

```
public string Name { get; set; }
```

Property Value

string

PhotoCount

The number of Instagram photos for this activity.

```
public int PhotoCount { get; set; }
```

Property Value

int

Photos

Photos associated with the activity.

```
public PhotosSummary? Photos { get; set; }
```

Property Value

PhotosSummary

Private

Whether this activity is private.

```
public bool Private { get; set; }
```

Property Value

bool

SegmentEfforts

Segment efforts during this activity.

```
public List<DetailedSegmentEffort> SegmentEfforts { get; set; }
```

Property Value

List<DetailedSegmentEffort>

SplitsMetric

Splits of the activity in metric units (for runs).

```
public List<Split>? SplitsMetric { get; set; }
```

Property Value

List<Split>

SplitsStandard

Splits of the activity in imperial units (for runs).

```
public List<Split>? SplitsStandard { get; set; }
```

Property Value

List<Split>

SportType

The sport type of the activity.

```
public SportType SportType { get; set; }
```

Property Value

SportType

StartDate

The UTC time when the activity was started.

```
public DateTime StartDate { get; set; }
```

Property Value

DateTime

StartDateLocal

The local time when the activity was started.

```
public DateTime StartDateLocal { get; set; }
```

Property Value

DateTime

StartLatLng

The start latitude/longitude of the activity.

```
public LatLng? StartLatLng { get; set; }
```

Property Value

LatLng

Timezone

The timezone of the activity.

```
public string Timezone { get; set; }
```

Property Value

string

TotalElevationGain

The activity's total elevation gain in meters.

```
public float TotalElevationGain { get; set; }
```

Property Value

float

TotalPhotoCount

The total number of photos (Instagram + Strava) for this activity.

```
public int TotalPhotoCount { get; set; }
```

Property Value

int

Trainer

Whether this activity was recorded on a training machine.

```
public bool Trainer { get; set; }
```

Property Value

bool

Type

Deprecated. Prefer to use SportType.

```
public ActivityType Type { get; set; }
```

Property Value

ActivityType

UploadId

The identifier of the upload that resulted in this activity.

```
public long UploadId { get; set; }
```

Property Value

long

UploadIdStr

The upload identifier in string format.

```
public string? UploadIdStr { get; set; }
```

Property Value

string

WeightedAverageWatts

Weighted average watts (Normalized Power) for rides with power data.

```
public int? WeightedAverageWatts { get; set; }
```

Property Value

int?

WorkoutType

The workout type of the activity.

```
public int? WorkoutType { get; set; }
```

Property Value

int?

Class Lap

Namespace: StravaAPILibary.Models.Activities

Assembly: StravaAPILibary.dll

Represents a single lap within an activity.

```
public class Lap
```

Inheritance

object ← Lap

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Activity

The activity associated with this lap.

```
public MetaActivity Activity { get; set; }
```

Property Value

MetaActivity

Athlete

The athlete associated with this lap.

```
public MetaAthlete Athlete { get; set; }
```

Property Value

MetaAthlete

AverageCadence

The lap's average cadence.

```
public float AverageCadence { get; set; }
```

Property Value

float

AverageSpeed

The lap's average speed.

```
public float AverageSpeed { get; set; }
```

Property Value

float

Distance

The lap's distance, in meters.

```
public float Distance { get; set; }
```

Property Value

float

ElapsedTime

The lap's elapsed time, in seconds.

```
public int ElapsedTime { get; set; }
```

Property Value

int

EndIndex

The end index of this lap in the activity's stream.

```
public int EndIndex { get; set; }
```

Property Value

int

Id

The unique identifier of this lap.

```
public long Id { get; set; }
```

Property Value

long

LapIndex

The index of this lap in the activity it belongs to.

```
public int LapIndex { get; set; }
```

Property Value

int

MaxSpeed

The maximum speed during this lap, in meters per second.

```
public float MaxSpeed { get; set; }
```

Property Value

float

MovingTime

The lap's moving time, in seconds.

```
public int MovingTime { get; set; }
```

Property Value

int

Name

The name of the lap.

```
public string Name { get; set; }
```

Property Value

string

PaceZone

The athlete's pace zone during this lap.

```
public int PaceZone { get; set; }
```

Property Value

int

Split

The split number for this lap.

```
public int Split { get; set; }
```

Property Value

int

StartDate

The time at which the lap was started.

```
public DateTime StartDate { get; set; }
```

Property Value

DateTime

StartDateLocal

The time at which the lap was started in the local timezone.

```
public DateTime StartDateLocal { get; set; }
```

Property Value

DateTime

StartIndex

The start index of this lap in the activity's stream.

```
public int StartIndex { get; set; }
```

Property Value

int

TotalElevationGain

The elevation gain of this lap, in meters.

```
public float TotalElevationGain { get; set; }
```

Property Value

float

Class MetaActivity

Namespace: StravaAPILibary.Models.Activities

Assembly: StravaAPILibary.dll

Represents a minimal reference to an activity.

```
public class MetaActivity
```

Inheritance

object ← MetaActivity

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Id

The unique identifier of the activity.

```
public long Id { get; set; }
```

Property Value

long

Class Split

Namespace: StravaAPILibary.Models.Activities

Assembly: StravaAPILibary.dll

Represents a split of an activity, typically used for running activities.

```
public class Split
```

Inheritance

object ← Split

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

AverageSpeed

The average speed of this split, in meters per second.

```
public float AverageSpeed { get; set; }
```

Property Value

float

Distance

The distance of this split, in meters.

```
public float Distance { get; set; }
```

Property Value

float

ElapsedTime

The elapsed time of this split, in seconds.

```
public int ElapsedTime { get; set; }
```

Property Value

int

ElevationDifference

The elevation difference of this split, in meters.

```
public float ElevationDifference { get; set; }
```

Property Value

float

MovingTime

The moving time of this split, in seconds.

```
public int MovingTime { get; set; }
```

Property Value

int

PaceZone

The pacing zone of this split.

```
public int PaceZone { get; set; }
```

Property Value

int

SplitIndex

The index number of this split.

```
public int SplitIndex { get; set; }
```

Property Value

int

Class SummaryActivity

Namespace: StravaAPILibary.Models.Activities

Assembly: StravaAPILibary.dll

Represents a summarized Strava activity with key performance and metadata fields.

```
public class SummaryActivity
```

Inheritance

object ← SummaryActivity

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

AchievementCount

The number of achievements gained during this activity.

```
public int AchievementCount { get; set; }
```

Property Value

int

Athlete

The athlete who performed this activity.

```
public MetaAthlete? Athlete { get; set; }
```

Property Value

MetaAthlete

AthleteCount

The number of athletes participating in this activity (group activities).

```
public int AthleteCount { get; set; }
```

Property Value

int

AverageSpeed

The average speed of this activity, in meters per second.

```
public float AverageSpeed { get; set; }
```

Property Value

float

AverageWatts

The average power output in watts (rides only).

```
public float? AverageWatts { get; set; }
```

Property Value

float?

CommentCount

The number of comments on this activity.

```
public int CommentCount { get; set; }
```

Property Value

int

Commute

Indicates whether this activity is a commute.

```
public bool Commute { get; set; }
```

Property Value

bool

DeviceWatts

Indicates whether the power data is from a power meter (true) or estimated (false).

```
public bool? DeviceWatts { get; set; }
```

Property Value

bool?

Distance

The activity's distance, in meters.

```
public float Distance { get; set; }
```

Property Value

float

ElapsedTime

The activity's elapsed time, in seconds.

```
public int ElapsedTime { get; set; }
```

Property Value

int

ElevHigh

The highest elevation reached during the activity, in meters.

```
public float ElevHigh { get; set; }
```

Property Value

float

ElevLow

The lowest elevation reached during the activity, in meters.

```
public float ElevLow { get; set; }
```

Property Value

float

EndLatLng

The end latitude/longitude of the activity.

```
public LatLng? EndLatLng { get; set; }
```

Property Value

LatLng

ExternalId

The identifier provided at upload time.

```
public string? ExternalId { get; set; }
```

Property Value

string

Flagged

Indicates whether this activity was flagged.

```
public bool Flagged { get; set; }
```

Property Value

bool

GearId

The gear ID associated with this activity.

```
public string? GearId { get; set; }
```

Property Value

string

HasKudoed

Indicates whether the authenticated athlete has given kudos to this activity.

```
public bool HasKudoed { get; set; }
```

Property Value

bool

HideFromHome

Indicates whether the activity is muted (hidden from home feed).

```
public bool HideFromHome { get; set; }
```

Property Value

bool

Id

The unique identifier of the activity.

```
public long Id { get; set; }
```

Property Value

long

Kilojoules

The total work done in kilojoules during this activity (rides only).

```
public float? Kilojoules { get; set; }
```

Property Value

float?

KudosCount

The number of kudos given for this activity.

```
public int KudosCount { get; set; }
```

Property Value

int

Manual

Indicates whether this activity was created manually.

```
public bool Manual { get; set; }
```

Property Value

bool

Map

The map information associated with this activity.

```
public PolylineMap? Map { get; set; }
```

Property Value

PolylineMap

MaxSpeed

The maximum speed of this activity, in meters per second.

```
public float MaxSpeed { get; set; }
```

Property Value

float

MaxWatts

The maximum power output during this activity (rides with power meter only).

```
public int? MaxWatts { get; set; }
```

Property Value

int?

MovingTime

The activity's moving time, in seconds.

```
public int MovingTime { get; set; }
```

Property Value

int

Name

The name of the activity.

```
public string Name { get; set; }
```

Property Value

string

PhotoCount

The number of Instagram photos for this activity.

```
public int PhotoCount { get; set; }
```

Property Value

int

Private

Indicates whether this activity is private.

```
public bool Private { get; set; }
```

Property Value

bool

SportType

The sport type of the activity.

```
public SportType SportType { get; set; }
```

Property Value

SportType

StartDate

The time at which the activity was started (UTC).

```
public DateTime StartDate { get; set; }
```

Property Value

DateTime

StartDateLocal

The local time at which the activity was started.

```
public DateTime StartDateLocal { get; set; }
```

Property Value

DateTime

StartLatLng

The start latitude/longitude of the activity.

```
public LatLng? StartLatLng { get; set; }
```

Property Value

LatLng

Timezone

The timezone of the activity.

```
public string Timezone { get; set; }
```

Property Value

string

TotalElevationGain

The total elevation gain of the activity, in meters.

```
public float TotalElevationGain { get; set; }
```

Property Value

float

TotalPhotoCount

The total number of Instagram and Strava photos for this activity.

```
public int TotalPhotoCount { get; set; }
```

Property Value

int

Trainer

Indicates whether this activity was recorded on a training machine.

```
public bool Trainer { get; set; }
```

Property Value

bool

Type

The deprecated activity type. Prefer SportType.

```
public ActivityType? Type { get; set; }
```

Property Value

ActivityType?

UploadId

The identifier of the upload that resulted in this activity.

```
public long? UploadId { get; set; }
```

Property Value

long?

UploadIdStr

The unique identifier of the upload in string format.

```
public string? UploadIdStr { get; set; }
```

Property Value

string

WeightedAverageWatts

Similar to Normalized Power, rides with power meter data only.

```
public int? WeightedAverageWatts { get; set; }
```

Property Value

int?

WorkoutType

The workout type of this activity.

```
public int? WorkoutType { get; set; }
```

Property Value

int?

Class UpdatableActivity

Namespace: StravaAPILibary.Models.Activities

Assembly: StravaAPILibary.dll

Represents an updatable activity in Strava.

```
public class UpdatableActivity
```

Inheritance

object ← UpdatableActivity

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Commute

Indicates whether this activity is a commute.

```
public bool Commute { get; set; }
```

Property Value

bool

Description

The description of the activity.

```
public string Description { get; set; }
```

Property Value

string

GearId

Identifier for the gear associated with the activity. Use 'none' to clear gear from activity.

```
public string GearId { get; set; }
```

Property Value

string

HideFromHome

Indicates whether this activity is muted (hidden from home feed).

```
public bool HideFromHome { get; set; }
```

Property Value

bool

Name

The name of the activity.

```
public string Name { get; set; }
```

Property Value

string

SportType

The sport type of the activity.

```
public SportType SportType { get; set; }
```

Property Value

SportType

Trainer

Indicates whether this activity was recorded on a training machine.

```
public bool Trainer { get; set; }
```

Property Value

bool

Type

The deprecated activity type. Prefer to use SportType.

```
public ActivityType? Type { get; set; }
```

Property Value

ActivityType?

Namespace StravaAPILibary.Models.Athletes

Classes

DetailedAthlete

Represents a detailed athlete profile with extensive personal, equipment, and club data.

MetaAthlete

Represents a minimal athlete object containing only the athlete's ID.

SummaryAthlete

Represents a summary of an athlete with profile and location details.

Class DetailedAthlete

Namespace: StravaAPILibary.Models.Athletes

Assembly: StravaAPILibary.dll

Represents a detailed athlete profile with extensive personal, equipment, and club data.

```
public class DetailedAthlete
```

Inheritance

object ← DetailedAthlete

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Bikes

The bikes owned by the athlete.

```
public List<SummaryGear> Bikes { get; set; }
```

Property Value

List<SummaryGear>

City

The athlete's city.

```
public string? City { get; set; }
```

Property Value

string

Clubs

The clubs the athlete is a member of.

```
public List<SummaryClub> Clubs { get; set; }
```

Property Value

List <SummaryClub>

Country

The athlete's country.

```
public string? Country { get; set; }
```

Property Value

string

CreatedAt

The time at which the athlete was created.

```
public DateTime CreatedAt { get; set; }
```

Property Value

DateTime

FirstName

The athlete's first name.

```
public string FirstName { get; set; }
```

Property Value

string

FollowerCount

The athlete's follower count.

```
public int FollowerCount { get; set; }
```

Property Value

int

FriendCount

The athlete's friend count.

```
public int FriendCount { get; set; }
```

Property Value

int

Ftp

The athlete's Functional Threshold Power (FTP).

```
public int? Ftp { get; set; }
```

Property Value

int?

Id

The unique identifier of the athlete.

```
public long Id { get; set; }
```

Property Value

long

LastName

The athlete's last name.

```
public string LastName { get; set; }
```

Property Value

string

MeasurementPreference

The athlete's preferred unit system. May take one of the following values: "feet", "meters".

```
public string MeasurementPreference { get; set; }
```

Property Value

string

Premium

Deprecated. Use Summit instead. Whether the athlete has any Summit subscription.

```
public bool Premium { get; set; }
```

Property Value

bool

Profile

URL to a 124x124 pixel profile picture.

```
public string Profile { get; set; }
```

Property Value

string

ProfileMedium

URL to a 62x62 pixel profile picture.

```
public string ProfileMedium { get; set; }
```

Property Value

string

ResourceState

Resource state, indicates level of detail. Possible values: 1 -> "meta", 2 -> "summary", 3 -> "detail".

```
public int ResourceState { get; set; }
```

Property Value

int

Sex

The athlete's sex. May take one of the following values: "M", "F".

```
public string? Sex { get; set; }
```

Property Value

string

Shoes

The shoes owned by the athlete.

```
public List<SummaryGear> Shoes { get; set; }
```

Property Value

List <SummaryGear>

State

The athlete's state or geographical region.

```
public string? State { get; set; }
```

Property Value

string

Summit

Whether the athlete has any Summit subscription.

```
public bool Summit { get; set; }
```

Property Value

bool

UpdatedAt

The time at which the athlete was last updated.

```
public DateTime UpdatedAt { get; set; }
```

Property Value

DateTime

Weight

The athlete's weight.

```
public float? Weight { get; set; }
```

Property Value

float?

Class MetaAthlete

Namespace: StravaAPILibary.Models.Athletes

Assembly: StravaAPILibary.dll

Represents a minimal athlete object containing only the athlete's ID.

```
public class MetaAthlete
```

Inheritance

object ← MetaAthlete

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Id

The unique identifier of the athlete.

```
public long Id { get; set; }
```

Property Value

long

Class SummaryAthlete

Namespace: StravaAPILibary.Models.Athletes

Assembly: StravaAPILibary.dll

Represents a summary of an athlete with profile and location details.

```
public class SummaryAthlete
```

Inheritance

object ← SummaryAthlete

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

City

The athlete's city.

```
public string City { get; set; }
```

Property Value

string

Country

The athlete's country.

```
public string Country { get; set; }
```

Property Value

string

CreatedAt

The time at which the athlete was created.

```
public DateTime CreatedAt { get; set; }
```

Property Value

DateTime

Firstname

The athlete's first name.

```
public string Firstname { get; set; }
```

Property Value

string

Id

The unique identifier of the athlete.

```
public long Id { get; set; }
```

Property Value

long

Lastname

The athlete's last name.

```
public string Lastname { get; set; }
```

Property Value

string

Premium

Deprecated. Use Summit field instead. Whether the athlete has any Summit subscription.

```
public bool Premium { get; set; }
```

Property Value

bool

Profile

URL to a 124x124 pixel profile picture.

```
public string Profile { get; set; }
```

Property Value

string

ProfileMedium

URL to a 62x62 pixel profile picture.

```
public string ProfileMedium { get; set; }
```

Property Value

string

ResourceState

Resource state, indicates level of detail. Possible values: 1 (meta), 2 (summary), 3 (detail).

```
public int ResourceState { get; set; }
```

Property Value

int

Sex

The athlete's sex. M = Male, F = Female.

```
public string Sex { get; set; }
```

Property Value

string

State

The athlete's state or geographical region.

```
public string State { get; set; }
```

Property Value

string

Summit

Whether the athlete has any Summit subscription.

```
public bool Summit { get; set; }
```

Property Value

bool

UpdatedAt

The time at which the athlete was last updated.

```
public DateTime UpdatedAt { get; set; }
```

Property Value

DateTime

Namespace StravaAPILibary.Models.Clubs

Classes

ClubActivity

Represents an activity within a club.

ClubAthlete

Represents an athlete within a club.

DetailedClub

Represents detailed information about a club.

MetaClub

Represents a meta-level summary of a club in Strava.

SummaryClub

Represents a summary of a Strava club with basic metadata and membership information.

Class ClubActivity

Namespace: StravaAPILibary.Models.Clubs

Assembly: StravaAPILibary.dll

Represents an activity within a club.

```
public class ClubActivity
```

Inheritance

object ← ClubActivity

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Athlete

The athlete who performed the activity.

```
public MetaAthlete Athlete { get; set; }
```

Property Value

MetaAthlete

Distance

The activity's distance in meters.

```
public float Distance { get; set; }
```

Property Value

float

ElapsedTime

The activity's elapsed time in seconds.

```
public int ElapsedTime { get; set; }
```

Property Value

int

MovingTime

The activity's moving time in seconds.

```
public int MovingTime { get; set; }
```

Property Value

int

Name

The name of the activity.

```
public string Name { get; set; }
```

Property Value

string

SportType

The sport type of the activity.

```
public SportType SportType { get; set; }
```

Property Value

SportType

TotalElevationGain

The activity's total elevation gain in meters.

```
public float TotalElevationGain { get; set; }
```

Property Value

float

Type

The type of the activity. (Deprecated - prefer SportType)

```
public ActivityType Type { get; set; }
```

Property Value

ActivityType

WorkoutType

The activity's workout type.

```
public int WorkoutType { get; set; }
```

Property Value

int

Class ClubAthlete

Namespace: StravaAPILibary.Models.Clubs

Assembly: StravaAPILibary.dll

Represents an athlete within a club.

```
public class ClubAthlete
```

Inheritance

object ← ClubAthlete

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Admin

Whether the athlete is a club admin.

```
public bool Admin { get; set; }
```

Property Value

bool

FirstName

The athlete's first name.

```
public string FirstName { get; set; }
```

Property Value

string

LastName

The athlete's last initial.

```
public string LastName { get; set; }
```

Property Value

string

Member

The athlete's member status.

```
public string Member { get; set; }
```

Property Value

string

Owner

Whether the athlete is the club owner.

```
public bool Owner { get; set; }
```

Property Value

bool

ResourceState

Resource state, indicates level of detail. 1 -> meta, 2 -> summary, 3 -> detail.

```
public int ResourceState { get; set; }
```

Property Value

int

Class DetailedClub

Namespace: StravaAPILibary.Models.Clubs

Assembly: StravaAPILibary.dll

Represents detailed information about a club.

```
public class DetailedClub
```

Inheritance

object ← DetailedClub

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

ActivityTypes

The activity types that count for a club. This takes precedence over SportType.

```
public List<ActivityType> ActivityTypes { get; set; }
```

Property Value

List<ActivityType>

Admin

Whether the currently logged-in athlete is an administrator of this club.

```
public bool Admin { get; set; }
```

Property Value

bool

City

The club's city.

```
public string? City { get; set; }
```

Property Value

string

Country

The club's country.

```
public string? Country { get; set; }
```

Property Value

string

CoverPhoto

URL to a ~1185x580 pixel cover photo.

```
public string CoverPhoto { get; set; }
```

Property Value

string

CoverPhotoSmall

URL to a ~360x176 pixel cover photo.

```
public string CoverPhotoSmall { get; set; }
```

Property Value

string

Featured

Whether the club is featured or not.

```
public bool Featured { get; set; }
```

Property Value

bool

FollowingCount

The number of athletes in the club that the logged-in athlete follows.

```
public int FollowingCount { get; set; }
```

Property Value

int

Id

The club's unique identifier.

```
public long Id { get; set; }
```

Property Value

long

MemberCount

The club's member count.

```
public int MemberCount { get; set; }
```

Property Value

int

Membership

The membership status of the logged-in athlete. May take values: "member", "pending".

```
public string? Membership { get; set; }
```

Property Value

string

Name

The club's name.

```
public string Name { get; set; }
```

Property Value

string

Owner

Whether the currently logged-in athlete is the owner of this club.

```
public bool Owner { get; set; }
```

Property Value

bool

Private

Whether the club is private.

```
public bool Private { get; set; }
```

Property Value

bool

ProfileMedium

URL to a 60x60 pixel profile picture.

```
public string ProfileMedium { get; set; }
```

Property Value

string

ResourceState

Resource state, indicates level of detail. Possible values: 1 -> "meta", 2 -> "summary", 3 -> "detail".

```
public int ResourceState { get; set; }
```

Property Value

int

SportType

Deprecated. Prefer to use ActivityTypes. May take values: cycling, running, triathlon, other.

```
public string? SportType { get; set; }
```

Property Value

string

State

The club's state or geographical region.

```
public string? State { get; set; }
```

Property Value

string

Url

The club's vanity URL.

```
public string Url { get; set; }
```

Property Value

string

Verified

Whether the club is verified or not.

```
public bool Verified { get; set; }
```

Property Value

bool

Class MetaClub

Namespace: StravaAPILibary.Models.Clubs

Assembly: StravaAPILibary.dll

Represents a meta-level summary of a club in Strava.

```
public class MetaClub
```

Inheritance

object ← MetaClub

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Id

The club's unique identifier.

```
public long Id { get; set; }
```

Property Value

long

Name

The club's name.

```
public string Name { get; set; }
```

Property Value

string

ResourceState

Resource state, indicates level of detail. Possible values: 1 -> "meta", 2 -> "summary", 3 -> "detail".

```
public int ResourceState { get; set; }
```

Property Value

int

Class SummaryClub

Namespace: StravaAPILibary.Models.Clubs

Assembly: StravaAPILibary.dll

Represents a summary of a Strava club with basic metadata and membership information.

```
public class SummaryClub
```

Inheritance

object ← SummaryClub

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

ActivityTypes

The activity types that count for a club. This takes precedence over SportType.

```
public List<ActivityType>? ActivityTypes { get; set; }
```

Property Value

List<ActivityType>

City

The club's city.

```
public string? City { get; set; }
```

Property Value

string

Country

The club's country.

```
public string? Country { get; set; }
```

Property Value

string

CoverPhoto

URL to a ~1185x580 pixel cover photo.

```
public string? CoverPhoto { get; set; }
```

Property Value

string

CoverPhotoSmall

URL to a ~360x176 pixel cover photo.

```
public string? CoverPhotoSmall { get; set; }
```

Property Value

string

Featured

Indicates whether the club is featured.

```
public bool Featured { get; set; }
```

Property Value

bool

Id

The club's unique identifier.

```
public long Id { get; set; }
```

Property Value

long

MemberCount

The club's member count.

```
public int MemberCount { get; set; }
```

Property Value

int

Name

The club's name.

```
public string Name { get; set; }
```

Property Value

string

Private

Indicates whether the club is private.

```
public bool Private { get; set; }
```

Property Value

bool

ProfileMedium

URL to a 60x60 pixel profile picture.

```
public string? ProfileMedium { get; set; }
```

Property Value

string

ResourceState

Resource state, indicates level of detail. Possible values: 1 -> meta, 2 -> summary, 3 -> detail.

```
public int ResourceState { get; set; }
```

Property Value

int

SportType

Deprecated. Prefer to use ActivityTypes. May take one of the following values: cycling, running, triathlon, other.

```
public string? SportType { get; set; }
```

Property Value

string

State

The club's state or geographical region.

```
public string? State { get; set; }
```

Property Value

string

Url

The club's vanity URL.

```
public string Url { get; set; }
```

Property Value

string

Verified

Indicates whether the club is verified.

```
public bool Verified { get; set; }
```

Property Value

bool

Namespace StravaAPILibary.Models.Common

Classes

LatLng

Represents a geographical coordinate as a pair of latitude and longitude.

Class LatLng

Namespace: StravaAPILibary.Models.Common

Assembly: StravaAPILibary.dll

Represents a geographical coordinate as a pair of latitude and longitude.

```
public class LatLng
```

Inheritance

object ← LatLng

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

LatLng()

Initializes a new instance of LatLng.

```
public LatLng()
```

LatLng(float, float)

Initializes a new instance of LatLng with specified latitude and longitude.

```
public LatLng(float latitude, float longitude)
```

Parameters

`latitude` float

The latitude component.

`longitude` float

The longitude component.

Properties

Latitude

Latitude component of the coordinate.

```
public float Latitude { get; set; }
```

Property Value

float

Longitude

Longitude component of the coordinate.

```
public float Longitude { get; set; }
```

Property Value

float

Methods

ToArray()

Returns the coordinate as an array of floats [latitude, longitude].

```
public float[] ToArray()
```

Returns

float[]

Namespace StravaAPILibary.Models.Enums

Enums

ActivityType

Enumeration of the types an activity may have. Note: Does not include new sport types (e.g. MountainBikeRide).

SportType

Enumeration of the sport types an activity may have. Unlike ActivityType, this includes newer activity types such as MountainBikeRide.

Enum ActivityType

Namespace: StravaAPILibary.Models.Enums

Assembly: StravaAPILibary.dll

Enumeration of the types an activity may have. Note: Does not include new sport types (e.g. MountainBikeRide).

```
public enum ActivityType
```

Fields

`AlpineSki = 0`

Alpine skiing activity.

`BackcountrySki = 1`

Backcountry skiing activity.

`Canoeing = 2`

Canoeing activity.

`Crossfit = 3`

Crossfit training session.

`EBikeRide = 4`

E-bike ride activity.

`Elliptical = 5`

Elliptical trainer workout.

`Golf = 6`

Golf session.

`Handcycle = 7`

Handcycle ride.

`Hike = 8`

Hiking activity.

`IceSkate = 9`

Ice skating activity.

`InlineSkate = 10`

Inline skating activity.

`Kayaking = 11`

Kayaking activity.

`Kitesurf = 12`

Kitesurfing session.

`NordicSki = 13`

Nordic skiing activity.

`Ride = 14`

Standard bike ride.

`RockClimbing = 15`

Rock climbing activity.

`RollerSki = 16`

Roller skiing activity.

`Rowing = 17`

Rowing session.

`Run = 18`

Running activity.

`Sail = 19`

Sailing session.

`Skateboard = 20`

Skateboarding session.

`Snowboard = 21`

Snowboarding activity.

`Snowshoe = 22`

Snowshoeing activity.

`Soccer = 23`

Soccer game or practice.

`StairStepper = 24`

Stair stepper workout.

`StandUpPaddling = 25`

Stand-up paddling session.

`Surfing = 26`

Surfing session.

`Swim = 27`

Swimming activity.

`Velomobile = 28`

Velomobile ride.

`VirtualRide = 29`

Virtual ride on trainer or simulator.

`VirtualRun = 30`

Virtual run on treadmill or simulator.

`Walk = 31`

Walking activity.

`WeightTraining = 32`

Weight training session.

`Wheelchair = 33`

Wheelchair workout or ride.

`Windsurf = 34`

Windsurfing session.

`Workout = 35`

General workout session.

`Yoga = 36`

Yoga session.

Enum SportType

Namespace: StravaAPILibary.Models.Enums

Assembly: StravaAPILibary.dll

Enumeration of the sport types an activity may have. Unlike ActivityType, this includes newer activity types such as MountainBikeRide.

```
public enum SportType
```

Fields

`AlpineSki = 0`

Alpine skiing activity.

`BackcountrySki = 1`

Backcountry skiing activity.

`Badminton = 2`

Badminton activity.

`Canoeing = 3`

Canoeing activity.

`Crossfit = 4`

Crossfit workout.

`EBikeRide = 5`

Electric bike ride.

`EMountainBikeRide = 7`

Electric mountain bike ride.

`Elliptical = 6`

Elliptical workout.

`Golf = 8`

Golf session.

`GravelRide = 9`

Gravel bike ride.

`Handcycle = 10`

Handcycling activity.

`HighIntensityIntervalTraining = 11`

High-intensity interval training.

`Hike = 12`

Hiking activity.

`IceSkate = 13`

Ice skating activity.

`InlineSkate = 14`

Inline skating activity.

`Kayaking = 15`

Kayaking activity.

`Kitesurf = 16`

Kitesurfing activity.

`MountainBikeRide = 17`

Mountain biking activity.

`NordicSki = 18`

Nordic skiing activity.

`Pickleball = 19`

Pickleball activity.

`Pilates = 20`

Pilates workout.

`Racquetball = 21`

Racquetball activity.

`Ride = 22`

Standard cycling activity.

`RockClimbing = 23`

Rock climbing activity.

`RollerSki = 24`

Roller skiing activity.

`Rowing = 25`

Rowing activity.

`Run = 26`

Running activity.

`Sail = 27`

Sailing activity.

`Skateboard = 28`

Skateboarding activity.

`Snowboard = 29`

Snowboarding activity.

`Snowshoe = 30`

Snowshoeing activity.

`Soccer = 31`

Soccer game.

`Squash = 32`

Squash activity.

`StairStepper = 33`

Stair stepper workout.

`StandUpPaddling = 34`

Stand-up paddling activity.

`Surfing = 35`

Surfing activity.

`Swim = 36`

Swimming activity.

`TableTennis = 37`

Table tennis activity.

`Tennis = 38`

Tennis activity.

`TrailRun = 39`

Trail running activity.

`Velomobile = 40`

Velomobile ride.

`VirtualRide = 41`

Virtual cycling ride.

`VirtualRow = 42`

Virtual rowing session.

`VirtualRun = 43`

Virtual running session.

`Walk = 44`

Walking activity.

`WeightTraining = 45`

Weight training session.

`Wheelchair = 46`

Wheelchair activity.

`Windsurf = 47`

Windsurfing activity.

`Workout = 48`

Generic workout activity.

`Yoga = 49`

Yoga session.

Namespace StravaAPILibary.Models.Explorer

Classes

ExplorerResponse

Represents the response from the Explorer API containing a set of segments.

ExplorerSegment

Represents a segment returned from the Explorer API.

Class ExplorerResponse

Namespace: StravaAPILibary.Models.Explorer

Assembly: StravaAPILibary.dll

Represents the response from the Explorer API containing a set of segments.

```
public class ExplorerResponse
```

Inheritance

object ← ExplorerResponse

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Segments

The set of segments matching an explorer request.

```
public List<ExplorerSegment> Segments { get; set; }
```

Property Value

List<ExplorerSegment>

Class ExplorerSegment

Namespace: StravaAPILibary.Models.Explorer

Assembly: StravaAPILibary.dll

Represents a segment returned from the Explorer API.

```
public class ExplorerSegment
```

Inheritance

object ← ExplorerSegment

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

AvgGrade

The segment's average grade in percents.

```
public float AvgGrade { get; set; }
```

Property Value

float

ClimbCategory

The category of the climb [0, 5]. Higher is harder (e.g. 5 = Hors catégorie, 0 = uncategorized).

```
public int ClimbCategory { get; set; }
```

Property Value

int

ClimbCategoryDesc

The description for the category of the climb (NC, 4, 3, 2, 1, HC).

```
public string ClimbCategoryDesc { get; set; }
```

Property Value

string

Distance

The segment's distance in meters.

```
public float Distance { get; set; }
```

Property Value

float

ElevDifference

The segment's elevation difference in meters.

```
public float ElevDifference { get; set; }
```

Property Value

float

EndLatLng

The ending latitude/longitude of the segment.

```
public LatLng EndLatLng { get; set; }
```

Property Value

LatLng

Id

The unique identifier of this segment.

```
public long Id { get; set; }
```

Property Value

long

Name

The name of this segment.

```
public string Name { get; set; }
```

Property Value

string

Points

The polyline of the segment.

```
public string Points { get; set; }
```

Property Value

string

StartLatLng

The starting latitude/longitude of the segment.

```
public LatLng StartLatLng { get; set; }
```

Property Value

LatLng

Namespace StravaAPILibary.Models.Gears

Classes

DetailedGear

Represents detailed information about a piece of gear in Strava.

SummaryGear

Represents a summary of gear associated with an athlete or activity.

Class DetailedGear

Namespace: StravaAPILibary.Models.Gears

Assembly: StravaAPILibary.dll

Represents detailed information about a piece of gear in Strava.

```
public class DetailedGear
```

Inheritance

object ← DetailedGear

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

BrandName

The gear's brand name.

```
public string BrandName { get; set; }
```

Property Value

string

Description

The description of the gear.

```
public string Description { get; set; }
```

Property Value

string

Distance

The distance logged with this gear.

```
public float Distance { get; set; }
```

Property Value

float

FrameType

The gear's frame type (bike only).

```
public int FrameType { get; set; }
```

Property Value

int

Id

The gear's unique identifier.

```
public string Id { get; set; }
```

Property Value

string

ModelName

The gear's model name.

```
public string ModelName { get; set; }
```

Property Value

string

Name

The gear's name.

```
public string Name { get; set; }
```

Property Value

string

Primary

Whether this gear is the owner's default gear.

```
public bool Primary { get; set; }
```

Property Value

bool

ResourceState

Resource state, indicates level of detail. Possible values: 2 -> "summary", 3 -> "detail".

```
public int ResourceState { get; set; }
```

Property Value

int

Class SummaryGear

Namespace: StravaAPILibary.Models.Gears

Assembly: StravaAPILibary.dll

Represents a summary of gear associated with an athlete or activity.

```
public class SummaryGear
```

Inheritance

object ← SummaryGear

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Distance

The distance logged with this gear, in meters.

```
public float Distance { get; set; }
```

Property Value

float

Id

The gear's unique identifier.

```
public string Id { get; set; }
```

Property Value

string

Name

The gear's display name.

```
public string Name { get; set; }
```

Property Value

string

Primary

Indicates whether this gear is the athlete's primary/default gear.

```
public bool Primary { get; set; }
```

Property Value

bool

ResourceState

Resource state, indicates level of detail. Possible values: 2 = "summary", 3 = "detail".

```
public int ResourceState { get; set; }
```

Property Value

int

Namespace StravaAPILibary.Models.Maps

Classes

PolylineMap

Represents the polyline map data of an activity, route, or segment.

Class PolylineMap

Namespace: StravaAPILibary.Models.Maps

Assembly: StravaAPILibary.dll

Represents the polyline map data of an activity, route, or segment.

```
public class PolylineMap
```

Inheritance

object ← PolylineMap

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Id

The identifier of the map.

```
public string Id { get; set; }
```

Property Value

string

Polyline

The polyline of the map, only returned in detailed representations.

```
public string? Polyline { get; set; }
```

Property Value

string

SummaryPolyline

The summary polyline of the map.

```
public string? SummaryPolyline { get; set; }
```

Property Value

string

Namespace StravaAPILibary.Models.Photos

Classes

PhotosSummary

Represents a summary of photos related to an activity.

PhotosSummaryPrimary

Represents the primary photo associated with an activity.

Class PhotosSummary

Namespace: StravaAPILibary.Models.Photos

Assembly: StravaAPILibary.dll

Represents a summary of photos related to an activity.

```
public class PhotosSummary
```

Inheritance

object ← PhotosSummary

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Count

The number of photos.

```
public int Count { get; set; }
```

Property Value

int

Primary

The primary photo associated with this activity.

```
public PhotosSummaryPrimary? Primary { get; set; }
```

Property Value

PhotosSummaryPrimary

Class PhotosSummaryPrimary

Namespace: StravaAPILibary.Models.Photos

Assembly: StravaAPILibary.dll

Represents the primary photo associated with an activity.

```
public class PhotosSummaryPrimary
```

Inheritance

object ← PhotosSummaryPrimary

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Id

The ID of the primary photo.

```
public long Id { get; set; }
```

Property Value

long

Source

The source of the primary photo.

```
public int Source { get; set; }
```

Property Value

int

UniqueId

The unique identifier of the primary photo.

```
public string UniqueId { get; set; }
```

Property Value

string

Urls

The URLs associated with the primary photo.

```
public string Urls { get; set; }
```

Property Value

string

Namespace StravaAPILibary.Models.Routes

Classes

Route

Represents a Strava route, including its metadata and associated details.

Waypoint

Represents a waypoint along a route.

Class Route

Namespace: StravaAPILibary.Models.Routes

Assembly: StravaAPILibary.dll

Represents a Strava route, including its metadata and associated details.

```
public class Route
```

Inheritance

object ← Route

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Athlete

The athlete who created the route.

```
public SummaryAthlete Athlete { get; set; }
```

Property Value

SummaryAthlete

CreatedAt

The time at which the route was created.

```
public DateTime CreatedAt { get; set; }
```

Property Value

DateTime

Description

The description of the route.

```
public string Description { get; set; }
```

Property Value

string

Distance

The distance of the route, in meters.

```
public float Distance { get; set; }
```

Property Value

float

ElevationGain

The elevation gain of the route, in meters.

```
public float ElevationGain { get; set; }
```

Property Value

float

EstimatedMovingTime

Estimated time in seconds for the authenticated athlete to complete the route.

```
public int EstimatedMovingTime { get; set; }
```

Property Value

int

Id

The unique identifier of this route.

```
public long Id { get; set; }
```

Property Value

long

IdStr

The unique identifier of this route in string format.

```
public string IdStr { get; set; }
```

Property Value

string

Map

The polyline map of the route.

```
public PolylineMap Map { get; set; }
```

Property Value

PolylineMap

Name

The name of the route.

```
public string Name { get; set; }
```

Property Value

string

Private

Whether this route is private.

```
public bool Private { get; set; }
```

Property Value

bool

Segments

The segments traversed by this route.

```
public List<SummarySegment> Segments { get; set; }
```

Property Value

List <SummarySegment>

Starred

Whether this route is starred by the logged-in athlete.

```
public bool Starred { get; set; }
```

Property Value

bool

SubType

This route's sub-type (1 for road, 2 for mountain bike, 3 for cross, 4 for trail, 5 for mixed).

```
public int SubType { get; set; }
```

Property Value

int

Timestamp

An epoch timestamp of when the route was created.

```
public int Timestamp { get; set; }
```

Property Value

int

Type

This route's type (1 for ride, 2 for run).

```
public int Type { get; set; }
```

Property Value

int

UpdatedAt

The time at which the route was last updated.

```
public DateTime UpdatedAt { get; set; }
```

Property Value

DateTime

Waypoints

The custom waypoints along this route.

```
public List<Waypoint> Waypoints { get; set; }
```

Property Value

List <Waypoint>

Class Waypoint

Namespace: StravaAPILibary.Models.Routes

Assembly: StravaAPILibary.dll

Represents a waypoint along a route.

```
public class Waypoint
```

Inheritance

object ← Waypoint

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Categories

Categories that the waypoint belongs to.

```
public string Categories { get; set; }
```

Property Value

string

Description

A description of the waypoint (optional).

```
public string? Description { get; set; }
```

Property Value

string

DistanceIntoRoute

The number of meters along the route where the waypoint is located.

```
public int DistanceIntoRoute { get; set; }
```

Property Value

int

LatLng

The location along the route that the waypoint is closest to.

```
public LatLng LatLng { get; set; }
```

Property Value

LatLng

TargetLatLng

A location off of the route that the waypoint is (optional).

```
public LatLng? TargetLatLng { get; set; }
```

Property Value

LatLng

Title

A title for the waypoint.

```
public string Title { get; set; }
```

Property Value

string

Namespace StravaAPILibary.Models.Segments

Classes

DetailedSegment

Represents detailed information about a Strava segment.

DetailedSegmentEffort

Represents a detailed segment effort, including performance metrics and related activity/athlete information.

SummaryPRSegmentEffort

Represents the personal record (PR) effort for a segment.

SummarySegment

Represents a summary of a Strava segment.

SummarySegmentEffort

Represents a summary of an athlete's effort on a segment.

Class DetailedSegment

Namespace: StravaAPILibary.Models.Segments

Assembly: StravaAPILibary.dll

Represents detailed information about a Strava segment.

```
public class DetailedSegment
```

Inheritance

object ← DetailedSegment

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

ActivityType

The activity type of this segment (e.g., Ride, Run).

```
public string ActivityType { get; set; }
```

Property Value

string

AthleteCount

The number of unique athletes who have an effort on this segment.

```
public int AthleteCount { get; set; }
```

Property Value

int

AthletePrEffort

The athlete's personal record effort on this segment.

```
public SummaryPRSegmentEffort? AthletePrEffort { get; set; }
```

Property Value

SummaryPRSegmentEffort

AthleteSegmentStats

The athlete's segment statistics.

```
public SummarySegmentEffort? AthleteSegmentStats { get; set; }
```

Property Value

SummarySegmentEffort

AverageGrade

The segment's average grade, in percents.

```
public float AverageGrade { get; set; }
```

Property Value

float

City

The city where the segment is located.

```
public string City { get; set; }
```

Property Value

string

ClimbCategory

The category of the climb [0, 5]. Higher is harder (e.g., 5 = HC, 0 = uncategorized).

```
public int ClimbCategory { get; set; }
```

Property Value

int

Country

The country of the segment.

```
public string Country { get; set; }
```

Property Value

string

CreatedAt

The date and time when the segment was created.

```
public DateTime CreatedAt { get; set; }
```

Property Value

DateTime

Distance

The segment's distance, in meters.

```
public float Distance { get; set; }
```

Property Value

float

EffortCount

The total number of efforts recorded for this segment.

```
public int EffortCount { get; set; }
```

Property Value

int

ElevationHigh

The segment's highest elevation, in meters.

```
public float ElevationHigh { get; set; }
```

Property Value

float

ElevationLow

The segment's lowest elevation, in meters.

```
public float ElevationLow { get; set; }
```

Property Value

float

EndLatLng

The segment's end coordinates (latitude/longitude).

```
public LatLng? EndLatLng { get; set; }
```

Property Value

LatLng

Hazardous

Indicates whether this segment is considered hazardous.

```
public bool Hazardous { get; set; }
```

Property Value

bool

Id

The unique identifier of this segment.

```
public long Id { get; set; }
```

Property Value

long

Map

The polyline map of the segment.

```
public PolylineMap? Map { get; set; }
```

Property Value

PolylineMap

MaximumGrade

The segment's maximum grade, in percents.

```
public float MaximumGrade { get; set; }
```

Property Value

float

Name

The name of this segment.

```
public string Name { get; set; }
```

Property Value

string

Private

Whether this segment is private.

```
public bool Private { get; set; }
```

Property Value

bool

StarCount

The number of stars for this segment.

```
public int StarCount { get; set; }
```

Property Value

int

StartLatLng

The segment's start coordinates (latitude/longitude).

```
public LatLng? StartLatLng { get; set; }
```

Property Value

LatLng

State

The state or geographical region of the segment.

```
public string State { get; set; }
```

Property Value

string

TotalElevationGain

The total elevation gain of the segment, in meters.

```
public float TotalElevationGain { get; set; }
```

Property Value

float

UpdatedAt

The date and time when the segment was last updated.

```
public DateTime UpdatedAt { get; set; }
```

Property Value

DateTime

Class DetailedSegmentEffort

Namespace: StravaAPILibary.Models.Segments

Assembly: StravaAPILibary.dll

Represents a detailed segment effort, including performance metrics and related activity/athlete information.

```
public class DetailedSegmentEffort
```

Inheritance

object ← DetailedSegmentEffort

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Activity

The activity associated with this effort.

```
public MetaActivity? Activity { get; set; }
```

Property Value

MetaActivity

ActivityId

The unique identifier of the activity related to this effort.

```
public long ActivityId { get; set; }
```

Property Value

long

Athlete

The athlete who performed this effort.

```
public MetaAthlete? Athlete { get; set; }
```

Property Value

MetaAthlete

AverageCadence

The average cadence during this effort.

```
public float AverageCadence { get; set; }
```

Property Value

float

AverageHeartrate

The average heart rate of the athlete during this effort.

```
public float AverageHeartrate { get; set; }
```

Property Value

float

AverageWatts

The average power output during this effort, in watts.

```
public float AverageWatts { get; set; }
```

Property Value

float

DeviceWatts

Indicates whether the wattage was measured by a power meter (true) or estimated (false).

```
public bool DeviceWatts { get; set; }
```

Property Value

bool

Distance

The distance of the effort, in meters.

```
public float Distance { get; set; }
```

Property Value

float

ElapsedTime

The elapsed time of the effort, in seconds.

```
public int ElapsedTime { get; set; }
```

Property Value

int

EndIndex

The end index of this effort in its activity's stream.

```
public int EndIndex { get; set; }
```

Property Value

int

Hidden

Indicates whether this effort should be hidden when viewed within an activity.

```
public bool Hidden { get; set; }
```

Property Value

bool

Id

The unique identifier of this effort.

```
public long Id { get; set; }
```

Property Value

long

IsKom

Indicates whether this effort is the current best on the leaderboard (KOM/QOM).

```
public bool IsKom { get; set; }
```

Property Value

bool

KomRank

The athlete's rank on the global leaderboard (if in the top 10 at the time of upload).

```
public int? KomRank { get; set; }
```

Property Value

int?

MaxHeartrate

The maximum heart rate of the athlete during this effort.

```
public float MaxHeartrate { get; set; }
```

Property Value

float

MovingTime

The moving time of the effort, in seconds.

```
public int MovingTime { get; set; }
```

Property Value

int

Name

The name of the segment on which this effort was performed.

```
public string Name { get; set; }
```

Property Value

string

PrRank

The athlete's rank on their personal leaderboard (if in the top 3 at the time of upload).

```
public int? PrRank { get; set; }
```

Property Value

int?

Segment

The segment associated with this effort.

```
public SummarySegment? Segment { get; set; }
```

Property Value

SummarySegment

StartDate

The time at which the effort was started.

```
public DateTime StartDate { get; set; }
```

Property Value

DateTime

StartDateLocal

The local time at which the effort was started.

```
public DateTime StartDateLocal { get; set; }
```

Property Value

DateTime

StartIndex

The start index of this effort in its activity's stream.

```
public int StartIndex { get; set; }
```

Property Value

int

Class SummaryPRSegmentEffort

Namespace: StravaAPILibary.Models.Segments

Assembly: StravaAPILibary.dll

Represents the personal record (PR) effort for a segment.

```
public class SummaryPRSegmentEffort
```

Inheritance

object ← SummaryPRSegmentEffort

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

EffortCount

The number of efforts by the authenticated athlete on this segment.

```
public int EffortCount { get; set; }
```

Property Value

int

PrActivityId

The unique identifier of the activity related to the PR effort.

```
public long PrActivityId { get; set; }
```

Property Value

long

PrDate

The date and time when the PR effort was started.

```
public DateTime PrDate { get; set; }
```

Property Value

DateTime

PrElapsedTime

The elapsed time of the PR effort in seconds.

```
public int PrElapsedTime { get; set; }
```

Property Value

int

Class SummarySegment

Namespace: StravaAPILibary.Models.Segments

Assembly: StravaAPILibary.dll

Represents a summary of a Strava segment.

```
public class SummarySegment
```

Inheritance

object ← SummarySegment

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

ActivityType

The activity type of the segment (e.g. Ride, Run).

```
public string ActivityType { get; set; }
```

Property Value

string

AthletePrEffort

The personal record (PR) effort for this segment.

```
public SummaryPRSegmentEffort AthletePrEffort { get; set; }
```

Property Value

SummaryPRSegmentEffort

AthleteSegmentStats

The athlete's segment statistics.

```
public SummarySegmentEffort AthleteSegmentStats { get; set; }
```

Property Value

SummarySegmentEffort

AverageGrade

The segment's average grade, in percent.

```
public float AverageGrade { get; set; }
```

Property Value

float

City

The city where the segment is located.

```
public string City { get; set; }
```

Property Value

string

ClimbCategory

The climb category of the segment (0-5).

```
public int ClimbCategory { get; set; }
```

Property Value

int

Country

The country where the segment is located.

```
public string Country { get; set; }
```

Property Value

string

Distance

The distance of the segment, in meters.

```
public float Distance { get; set; }
```

Property Value

float

ElevationHigh

The segment's highest elevation, in meters.

```
public float ElevationHigh { get; set; }
```

Property Value

float

ElevationLow

The segment's lowest elevation, in meters.

```
public float ElevationLow { get; set; }
```

Property Value

float

EndLatLng

The ending latitude/longitude of the segment.

```
public LatLng EndLatLng { get; set; }
```

Property Value

LatLng

Id

The unique identifier of this segment.

```
public long Id { get; set; }
```

Property Value

long

MaximumGrade

The segment's maximum grade, in percent.

```
public float MaximumGrade { get; set; }
```

Property Value

float

Name

The name of this segment.

```
public string Name { get; set; }
```

Property Value

string

Private

Indicates whether the segment is private.

```
public bool Private { get; set; }
```

Property Value

bool

StartLatLng

The starting latitude/longitude of the segment.

```
public LatLng StartLatLng { get; set; }
```

Property Value

LatLng

State

The state or geographical region where the segment is located.

```
public string State { get; set; }
```

Property Value

string

Class SummarySegmentEffort

Namespace: StravaAPILibary.Models.Segments

Assembly: StravaAPILibary.dll

Represents a summary of an athlete's effort on a segment.

```
public class SummarySegmentEffort
```

Inheritance

object ← SummarySegmentEffort

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

ActivityId

The unique identifier of the activity related to this effort.

```
public long ActivityId { get; set; }
```

Property Value

long

Distance

The distance covered during this effort, in meters.

```
public float Distance { get; set; }
```

Property Value

float

ElapsedTime

The elapsed time of this effort in seconds.

```
public int ElapsedTime { get; set; }
```

Property Value

int

Id

The unique identifier of this effort.

```
public long Id { get; set; }
```

Property Value

long

IsKom

Indicates whether this effort is the current best on the leaderboard.

```
public bool IsKom { get; set; }
```

Property Value

bool

StartDate

The date and time when this effort was started.

```
public DateTime StartDate { get; set; }
```

Property Value

DateTime

StartDateLocal

The local date and time when this effort was started.

```
public DateTime StartDateLocal { get; set; }
```

Property Value

DateTime

Namespace StravaAPILibary.Models.Streams

Classes

AltitudeStream
: Represents a stream of altitude data points.

BaseStream
: Represents the base properties of a stream returned by the Strava API.

CadenceStream
: Represents a stream of cadence data points.

DistanceStream
: Represents a stream of distance data points.

HeartrateStream
: Represents a stream of heart rate data points.

LatLngStream
: Represents a stream of GPS coordinates (latitude/longitude).

MovingStream
: Represents a stream indicating whether the athlete was moving.

PowerStream
: Represents a stream of power data points.

SmoothGradeStream
: Represents a stream of smoothed gradient data points.

SmoothVelocityStream
: Represents a stream of smoothed velocity (speed) data points.

StreamSet
: Represents a collection of different data streams associated with an activity or segment.

TemperatureStream
: Represents a stream of temperature data points.

TimeStream
: Represents a stream of time data points.

Class AltitudeStream

Namespace: StravaAPILibary.Models.Streams

Assembly: StravaAPILibary.dll

Represents a stream of altitude data points.

```
public class AltitudeStream : BaseStream
```

Inheritance

object ← BaseStream ← AltitudeStream

Inherited Members

BaseStream.OriginalSize , BaseStream.Resolution , BaseStream.SeriesType , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Data

The sequence of altitude values, in meters.

```
public List<float> Data { get; set; }
```

Property Value

List <float>

Class BaseStream

Namespace: StravaAPILibary.Models.Streams

Assembly: StravaAPILibary.dll

Represents the base properties of a stream returned by the Strava API.

```
public class BaseStream
```

Inheritance

object ← BaseStream

Derived

AltitudeStream, CadenceStream, DistanceStream, HeartrateStream, LatLngStream, MovingStream, PowerStream, SmoothGradeStream, SmoothVelocityStream, TemperatureStream, TimeStream

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

OriginalSize

The number of data points in this stream.

```
public int OriginalSize { get; set; }
```

Property Value

int

Resolution

The level of detail (sampling) in which this stream was returned. Possible values: low, medium, high.

```
public string Resolution { get; set; }
```

Property Value

string

SeriesType

The base series used in the case the stream was downsampled. Possible values: distance, time.

```
public string SeriesType { get; set; }
```

Property Value

string

Class CadenceStream

Namespace: StravaAPILibary.Models.Streams

Assembly: StravaAPILibary.dll

Represents a stream of cadence data points.

```
public class CadenceStream : BaseStream
```

Inheritance

object ← BaseStream ← CadenceStream

Inherited Members

BaseStream.OriginalSize , BaseStream.Resolution , BaseStream.SeriesType , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Data

The sequence of cadence values, in rotations per minute.

```
public List<int> Data { get; set; }
```

Property Value

List<int>

Class DistanceStream

Namespace: StravaAPILibary.Models.Streams

Assembly: StravaAPILibary.dll

Represents a stream of distance data points.

```
public class DistanceStream : BaseStream
```

Inheritance

object ← BaseStream ← DistanceStream

Inherited Members

BaseStream.OriginalSize , BaseStream.Resolution , BaseStream.SeriesType , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Data

The sequence of distance values, in meters.

```
public List<float> Data { get; set; }
```

Property Value

List <float>

Class HeartrateStream

Namespace: StravaAPILibary.Models.Streams

Assembly: StravaAPILibary.dll

Represents a stream of heart rate data points.

```
public class HeartrateStream : BaseStream
```

Inheritance

object ← BaseStream ← HeartrateStream

Inherited Members

BaseStream.OriginalSize , BaseStream.Resolution , BaseStream.SeriesType , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Data

The sequence of heart rate values, in beats per minute.

```
public List<int> Data { get; set; }
```

Property Value

List<int>

Class LatLngStream

Namespace: StravaAPILibary.Models.Streams

Assembly: StravaAPILibary.dll

Represents a stream of GPS coordinates (latitude/longitude).

```
public class LatLngStream : BaseStream
```

Inheritance

object ← BaseStream ← LatLngStream

Inherited Members

BaseStream.OriginalSize , BaseStream.Resolution , BaseStream.SeriesType , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Data

The sequence of latitude/longitude coordinate pairs.

```
public List<LatLng> Data { get; set; }
```

Property Value

List<LatLng>

Class MovingStream

Namespace: StravaAPILibary.Models.Streams

Assembly: StravaAPILibary.dll

Represents a stream indicating whether the athlete was moving.

```
public class MovingStream : BaseStream
```

Inheritance

object ← BaseStream ← MovingStream

Inherited Members

BaseStream.OriginalSize , BaseStream.Resolution , BaseStream.SeriesType , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Data

The sequence of movement flags, where true indicates movement.

```
public List<bool> Data { get; set; }
```

Property Value

List<bool>

Class PowerStream

Namespace: StravaAPILibary.Models.Streams

Assembly: StravaAPILibary.dll

Represents a stream of power data points.

```
public class PowerStream : BaseStream
```

Inheritance

object ← BaseStream ← PowerStream

Inherited Members

BaseStream.OriginalSize , BaseStream.Resolution , BaseStream.SeriesType , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Data

The sequence of power values, in watts.

```
public List<int> Data { get; set; }
```

Property Value

List<int>

Class SmoothGradeStream

Namespace: StravaAPILibary.Models.Streams

Assembly: StravaAPILibary.dll

Represents a stream of smoothed gradient data points.

```
public class SmoothGradeStream : BaseStream
```

Inheritance

object ← BaseStream ← SmoothGradeStream

Inherited Members

BaseStream.OriginalSize , BaseStream.Resolution , BaseStream.SeriesType , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Data

The sequence of gradient values, as percentages.

```
public List<float> Data { get; set; }
```

Property Value

List<float>

Class SmoothVelocityStream

Namespace: StravaAPILibary.Models.Streams

Assembly: StravaAPILibary.dll

Represents a stream of smoothed velocity (speed) data points.

```
public class SmoothVelocityStream : BaseStream
```

Inheritance

object ← BaseStream ← SmoothVelocityStream

Inherited Members

BaseStream.OriginalSize , BaseStream.Resolution , BaseStream.SeriesType , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Data

The sequence of velocity values, in meters per second.

```
public List<float> Data { get; set; }
```

Property Value

List <float>

Class StreamSet

Namespace: StravaAPILibary.Models.Streams

Assembly: StravaAPILibary.dll

Represents a collection of different data streams associated with an activity or segment.

```
public class StreamSet
```

Inheritance

object ← StreamSet

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Altitude

Stream of altitude values (meters).

```
public AltitudeStream? Altitude { get; set; }
```

Property Value

AltitudeStream

Cadence

Stream of cadence values (rotations per minute).

```
public CadenceStream? Cadence { get; set; }
```

Property Value

CadenceStream

Distance

Stream of distance values (meters).

```
public DistanceStream? Distance { get; set; }
```

Property Value

DistanceStream

GradeSmooth

Stream of smoothed gradient values (percent grade).

```
public SmoothGradeStream? GradeSmooth { get; set; }
```

Property Value

SmoothGradeStream

Heartrate

Stream of heart rate values (bpm).

```
public HeartrateStream? Heartrate { get; set; }
```

Property Value

HeartrateStream

LatLng

Stream of GPS coordinates (latitude/longitude pairs).

```
public LatLngStream? LatLng { get; set; }
```

Property Value

LatLngStream

Moving

Stream indicating movement state (true = moving).

```
public MovingStream? Moving { get; set; }
```

Property Value

MovingStream

Temp

Stream of temperature values (°C).

```
public TemperatureStream? Temp { get; set; }
```

Property Value

TemperatureStream

Time

Stream of time values (seconds).

```
public TimeStream? Time { get; set; }
```

Property Value

TimeStream

VelocitySmooth

Stream of smoothed velocity values (m/s).

```
public SmoothVelocityStream? VelocitySmooth { get; set; }
```

Property Value

SmoothVelocityStream

Watts

Stream of power values (watts).

```
public PowerStream? Watts { get; set; }
```

Property Value

PowerStream

Class TemperatureStream

Namespace: StravaAPILibary.Models.Streams

Assembly: StravaAPILibary.dll

Represents a stream of temperature data points.

```
public class TemperatureStream : BaseStream
```

Inheritance

object ← BaseStream ← TemperatureStream

Inherited Members

BaseStream.OriginalSize , BaseStream.Resolution , BaseStream.SeriesType , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Data

The sequence of temperature values, in degrees Celsius.

```
public List<int> Data { get; set; }
```

Property Value

List<int>

Class TimeStream

Namespace: StravaAPILibary.Models.Streams

Assembly: StravaAPILibary.dll

Represents a stream of time data points.

```
public class TimeStream : BaseStream
```

Inheritance

object ← BaseStream ← TimeStream

Inherited Members

BaseStream.OriginalSize , BaseStream.Resolution , BaseStream.SeriesType , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Data

The sequence of time values, in seconds.

```
public List<int> Data { get; set; }
```

Property Value

List<int>

Namespace StravaAPILibary.Models.Uploads

Classes

Upload

Represents the status and details of an upload in Strava.

Class Upload

Namespace: StravaAPILibary.Models.Uploads

Assembly: StravaAPILibary.dll

Represents the status and details of an upload in Strava.

```
public class Upload
```

Inheritance

object ← Upload

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

ActivityId

The identifier of the activity this upload resulted in.

```
public long? ActivityId { get; set; }
```

Property Value

long?

Error

The error associated with this upload, if any.

```
public string? Error { get; set; }
```

Property Value

string

ExternalId

The external identifier of the upload (e.g., original file name or external system reference).

```
public string ExternalId { get; set; }
```

Property Value

string

Id

The unique identifier of the upload.

```
public long Id { get; set; }
```

Property Value

long

IdStr

The unique identifier of the upload in string format.

```
public string IdStr { get; set; }
```

Property Value

string

Status

The current status of the upload.

```
public string Status { get; set; }
```

Property Value

string

Namespace StravaAPILibary.Models.Zones

Classes

HeartRateZoneRanges

Represents heart rate zone ranges for an athlete.

PowerZoneRanges

Represents the power zone ranges for an athlete.

ZoneRange

Represents a single heart rate or power zone range.

ZoneRanges

A collection of zone ranges (e.g., heart rate or power zones).

Class HeartRateZoneRanges

Namespace: StravaAPILibary.Models.Zones

Assembly: StravaAPILibary.dll

Represents heart rate zone ranges for an athlete.

```
public class HeartRateZoneRanges
```

Inheritance

object ← HeartRateZoneRanges

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

CustomZones

Indicates whether the athlete has set their own custom heart rate zones.

```
public bool CustomZones { get; set; }
```

Property Value

bool

Zones

The heart rate zones for the athlete.

```
public ZoneRanges Zones { get; set; }
```

Property Value

ZoneRanges

Class PowerZoneRanges

Namespace: StravaAPILibary.Models.Zones

Assembly: StravaAPILibary.dll

Represents the power zone ranges for an athlete.

```
public class PowerZoneRanges
```

Inheritance

object ← PowerZoneRanges

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Zones

The collection of power zone ranges.

```
public ZoneRanges Zones { get; set; }
```

Property Value

ZoneRanges

Class ZoneRange

Namespace: StravaAPILibary.Models.Zones

Assembly: StravaAPILibary.dll

Represents a single heart rate or power zone range.

```
public class ZoneRange
```

Inheritance

object ← ZoneRange

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Max

The maximum value in the range.

```
public int Max { get; set; }
```

Property Value

int

Min

The minimum value in the range.

```
public int Min { get; set; }
```

Property Value

int

Class ZoneRanges

Namespace: StravaAPILibary.Models.Zones

Assembly: StravaAPILibary.dll

A collection of zone ranges (e.g., heart rate or power zones).

```
public class ZoneRanges
```

Inheritance

object ← ZoneRanges

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Zones

A list of individual zone ranges.

```
public List<ZoneRange> Zones { get; set; }
```

Property Value

List<ZoneRange>