User Authorization (OAuth)

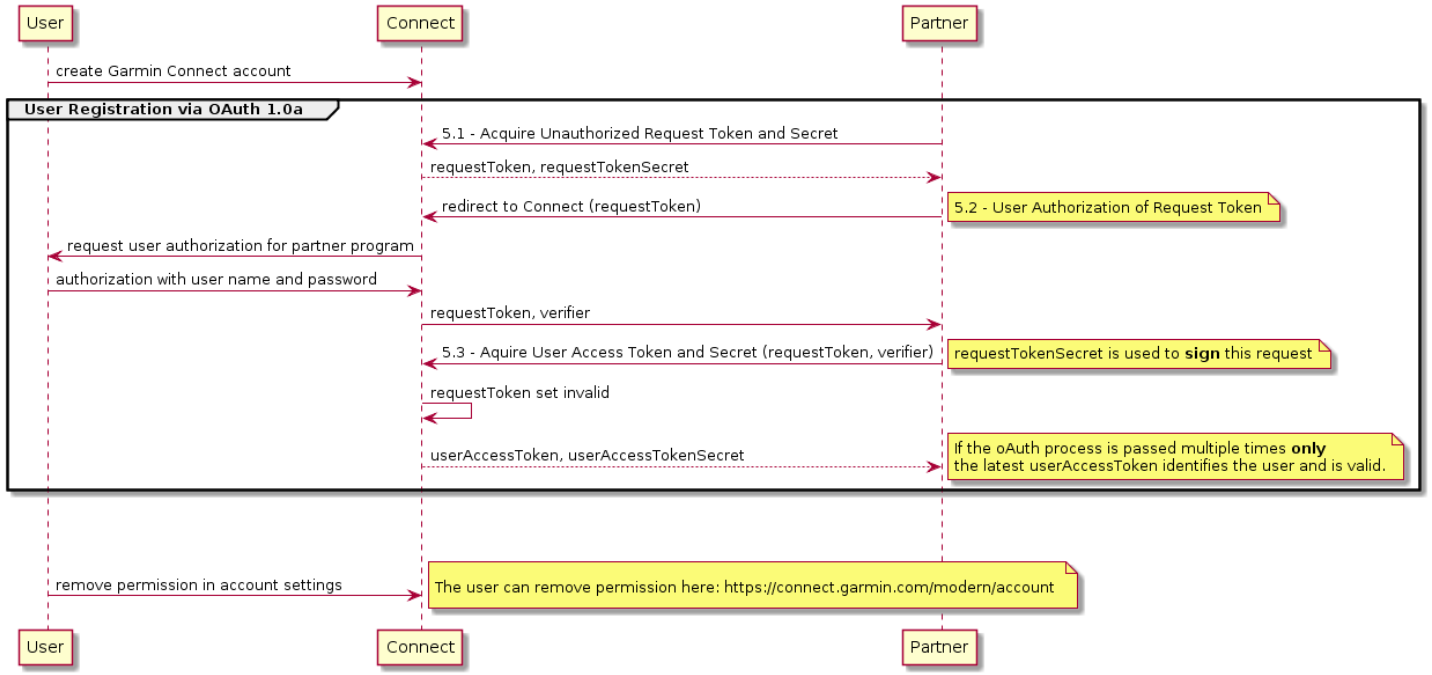
This section details the process of requesting access to wellness and activity data of Garmin Connect accounts. Accessing data of a Garmin Connect account requires a User Access Token for each Garmin Connect account. Garmin follows OAuth v1.0a.

Note:
Implementing OAuth signing manually can be quite complex. It is recommended that all partner integrations leverage an existing OAuth client library. A list of OAuth libraries that supposed OAuth v1.0 by language is maintained at https://oauth.net/1/.

Garmin Connect User Authorization

The Garmin Connect user authorization process must be completed once for each Connect user to give that user the opportunity to consent to sharing their Garmin data. The steps to perform Garmin Connect user authorization are as follows:

1. Acquire Unauthorized Request Token and Token Secret
2. User Authorization of Request Token
3. Acquire User Access Token and Token Secret



Acquire Unauthorized Request Token and Token Secret

The first step in getting user consent to share Garmin data is acquiring a request token and token secret. This request token does not have the ability to access data, nor is it user-specific yet.

Specification: https://oauth.net/core/1.0a/#auth_step1

URL (POST): https://connectapi.garmin.com/oauth-service/oauth/request_token

Authorization header:

To get a Request Token it is required to send the request with a valid Authorization header according to OAuth 1.0a
Common OAuth libraries handle the following steps automatically:

- **Generate a Signature Base String**

  The Signature Base String is a consistent reproducible concatenation of the request elements into a single string and needs to be encoded as defined in Parameter Encoding.

  The Signature Base String contains the HTTP method (POST), the URL and the normalized parameters.
  The request parameters are sorted alphabetically and concatenated into a normalized string:

  - oauth_consumer_key
    The consumer key provided by Garmin.

  - oauth_signature_method
    The signature method which is used to sign the request (HMAC-SHA1).

  - oauth_nonce
    A random string, uniquely generated for each request.

  - oauth_timestamp
    The current timestamp (number of seconds since January 1, 1970 00:00:00 GMT).

  - oauth_version
    This must bet set to 1.0

    Example:

    Normalized parameters:

```
oauth_consumer_key=cb60d7f5-4173-7bcd-ae02-
e5a52a6940ac&oauth_nonce=kbki9sCGRwU&oauth_signature_method=HMAC-
SHA1&oauth_timestamp=1484837456&oauth_version=1.0
```

    Signature Base String (percent-encoded):

```
POST&https%3A%2F%2Fconnectapi.garmin.com%2Foauth-
service%2Foauth%2Frequest_token&oauth_consumer_key%3Dcb60d7f5-4173-7bcd-ae02-
e5a52a6940ac%26oauth_nonce%3Dkbki9sCGRwU%26oauth_signature_method%3DHMAC-
SHA1%26oauth_timestamp%3D1484837456%26oauth_version%3D1.0
```

- **Calculate HMAC-SHA1 signature**

  The Signature Base String and the Consumer Secret are used to calculate the HMAC-SHA1 as defined in RFC 2104 where the Signature Base String is the text and the key is the Consumer Secret, followed by an '&' character (ASCII code 38). The resulting octet string needs to be base64-encoded as defined in RFC 2045 section 6.8, then encoded like defined in Parameter Encoding.

  Example:

  Consumer Secret, including trailing '&' character:

```
3LFNjTLbGk5QqWVoypl8S2wAYcSL586E285&
```

  Base64-encoded and parameter-encoded signature:

```
%2BHlCpVX8Qgdw5Djfw0W30s7pfrY%3D
```

- **Create HTTP Authorization header**

  The OAuth Protocol Parameters are sent in the `Authorization` header in the following way:
  - Parameter names and values must be encoded like defined in Parameter Encoding.
  - Parameter values must be enclosed with double quote characters: " (ASCII code 34)
  - Parameters are separated by a comma character (ASCII code 44) and optional whitespace

  Example:

```
Authorization: OAuth oauth_version="1.0", oauth_consumer_key="cb60d7f5-4173-
7bcd-ae02-e5a52a6940ac", oauth_timestamp="1484837456",
oauth_nonce="kbki9sCGRwU", oauth_signature_method="HMAC-SHA1",
oauth_signature="%2BHlCpVX8Qgdw5Djfw0W30s7pfrY%3D"
```

  A complete request sent with the command line tool 'cURL' looks like that:

```
curl -X POST -v -header 'Authorization: OAuth oauth_version="1.0",
oauth_consumer_key="cb60d7f5-4173-7bcd-ae02-e5a52a6940ac",
oauth_timestamp="1484837456", oauth_nonce="kbki9sCGRwU",
oauth_signature_method="HMAC-SHA1",
```

```
oauth_signature="%2BHlCpVX8Qgdw5Djfw0W30s7pfrY%3D"'
https://connectapi.garmin.com/oauth-service/oauth/request_token
```

Response

oauth_token=<request token>&oauth_token_secret=<request token secret>

User Authorization of Request Token

Once a request token has been generated, user consent is required to allow the token to be exchanged for an access token. This process occurs on the Garmin Connect oauthConfirm page as users only enter their login credentials on the Connect servers and never on third-party sites. Once the authentication process is complete, the user is redirected back to the partner's site. At this point in time the request token is authorized with the user's account.

After the user authenticates with Garmin Connect and grants permission for partner access the partner will be notified that the Request Token has been authorized and ready to be exchanged for an Access Token. If the user denies access the Partner will not get the required 'verifier'.

To make sure that the user granting access is the same user returning back to the partner to complete the process Garmin Connect will generate a verification code. This will be a complex value passed to the partner via the user and is required to complete the authorization process.

By default, Garmin Connect will honor the oauth callback URL configured when creating the consumer key through the Developer Portal. This pre-configured callback URL may be overridden using the URL parameter 'oauth_callback' if desired.

- oauth_token
  - The Request Token the user authorized or denied.
- oauth_verifier
  - The verification code or 'NULL' if the user denied access permissions

The callback URL may include partner provided query parameters. Garmin Connect will retain them unmodified and append the OAuth parameters to the existing query.

Specification: https://oauth.net/core/1.0a/#auth_step2

URL (GET): https://connect.garmin.com/oauthConfirm

Request Parameters:

- oauth_token (required): The Request token from previous step
- oauth_callback (optional)

Response

<Redirect URL>?oauth_token=<request token>&oauth_verifier=<alphanumeric value>

Acquire User Access Token and Token Secret

Request tokens are valid to link to a user's account, but not to acquire their data. To get protected resources a user access token is required. In this step, a user-authorized request token and verifier are exchanged for a user access token and token secret. This user access token can be used to acquire the authenticated user's protected resources using the Garmin APIs. This access token should be stored and will be used to request and/or associate summary data with a specific user. The user access token is valid until the user removes access permissions on Garmin Connect or a new User Access Token is created for the same Consumer Key and Garmin Connect user by performing the OAuth process again.

Specification: https://oauth.net/core/1.0a/#auth_step3

URL (POST): https://connectapi.garmin.com/oauth-service/oauth/access_token

Requesting the User Access Token is very similar to the steps to get the Request Token. The difference is that the signature is calculated using the Request Token and Secret as well as the verifier from the previous step.

Authorization header:

To get a UserAccessToken it is required to send the request with a valid Authorization header according to OAuth 1.0a
Please see chapter 0 for details.

- **Generate a Signature Base String from the following parameters:**
    - oauth_consumer_key
      The consumer key provided by Garmin.
    - oauth_token
      The request token obtained in first authorization step
    - oauth_signature_method
      The signature method which is used to sign the request (HMAC-SHA1).
    - oauth_nonce
      A random string, uniquely generated for each request.

- oauth_timestamp
  The current timestamp (number of seconds since January 1, 1970 00:00:00 GMT).

- oauth_version
  This must bet set to 1.0

- oauth_verifier
  The verification code received from Garmin Connect in second authorization step

  Example:

  Normalized parameters:

```
oauth_consumer_key=cb60d7f5-4173-7bcd-ae02-
e5a52a6940ac&oauth_nonce=2lRbgVyTAgh&oauth_signature_method=HMAC-
SHA1&oauth_timestamp=1484913680&oauth_token=760d85bd-b86e-4da6-b58b-
ba57a542b23b&oauth_verifier=wvDJQmLSwY&oauth_version=1.0
```

  Signature Base String (percent-encoded):

```
POST&http%3A%2F%2Fconnectapi.garmin.com%2Foauth-
service%2Foauth%2Faccess_token&oauth_consumer_key%3Dcb60d7f5-4173-7bcd-ae02-
e5a52a6940ac%26oauth_nonce%3D2lRbgVyTAgh%26oauth_signature_method%3DHMAC-
SHA1%26oauth_timestamp%3D1484913680%26oauth_token%3D760d85bd-b86e-4da6-b58b-
ba57a542b23b%26oauth_verifier%3DwvDJQmLSwY%26oauth_version%3D1.0
```

- **Calculate HMAC-SHA1 signature**

  Please see first authorization step for details.

  Example:

  Consumer Secret and Request Token Secret, separated with '&' character:

```
3LFNjTLbGk5QqWVoypl8S2wAYcSL586E285&VP2ZGuciICb7Lu769KWOP0wNMxxoLUZdAbq
```

  Base64-encoded and parameter-encoded signature:

```
zSAEERG2NNoQaVjVthJ5xP4XcCM%3D
```

- **Create HTTP Authorization header**

  Please see first authorization step for details.

  Example:

```
Authorization:  OAuth oauth_verifier="wvDJQmLSwY", oauth_version="1.0",
oauth_consumer_key="cb60d7f5-4173-7bcd-ae02-e5a52a6940ac",
oauth_token="760d85bd-b86e-4da6-b58b-ba57a542b23b",
oauth_timestamp="1484913680", oauth_nonce="2lRbgVyTAgh",
oauth_signature_method="HMAC-SHA1",
oauth_signature="zSAEERG2NNoQaVjVthJ5xP4XcCM%3D"
```

A complete request sent with the command line tool 'cURL' looks like that:

```
curl -X POST -v -header 'Authorization:  OAuth oauth_verifier="wvDJQmLSwY",
oauth_version="1.0", oauth_consumer_key="cb60d7f5-4173-7bcd-ae02-
e5a52a6940ac", oauth_token="760d85bd-b86e-4da6-b58b-ba57a542b23b",
oauth_timestamp="1484913680", oauth_nonce="2lRbgVyTAgh",
oauth_signature_method="HMAC-SHA1",
oauth_signature="zSAEERG2NNoQaVjVthJ5xP4XcCM%3D"'
https://connectapi.garmin.com/oauth-service/oauth/access_token
```

Response

oauth_token=<access token> &oauth_token_secret=<access token secret>

Signing requests

Requests to Garmin APIs must be signed using both the provided consumer key/secret and the User Access
Token/Secret acquired during the authorization phase described above. The User Access Token also identifies the Garmin Connect Account the request refers to.

Specification: https://oauth.net/core/1.0a/#signing_process

Request Parameters (placed in the Authorization header):

- oauth_consumer_key
  The consumer key provided by Garmin.

- oauth_token
  The User Access Token acquired during the authorization phase described in third authorization step
- oauth_signature_method
  The signature method the partner used to sign the request. This must be **HMAC-SHA1**.
- oauth_signature
  The signature as defined in Signing Requests (https://oauth.net/core/1.0a/#signing_process)
  Signature Base string example:

  ```
  GET&https%3A%2F%2Fhealthapi.garmin.com%2Fwellness-api%2Frest%2Fepochs&oauth_consumer_key%3Deb60d6a5-0172-4bbd-ae02-d5a5ea2140fa%26oauth_nonce%3D2464567464%26oauth_signature_method%3DHMAC-SHA1%26oauth_timestamp%3D1473668857%26oauth_token%3D07c6dd26-a57f-4c39-8fd3-6ac81d10fde6%26oauth_version%3D1.0%26uploadEndTimeInSeconds%3D1473668824%26uploadStartTimeInSeconds%3D1473582424
  ```
- oauth_timestamp
  As defined in Nonce and Timestamp: https://oauth.net/core/1.0a/#nonce
  Requests where the timestamp differs more than 10 minutes from the current UTC time will fail.
- oauth_nonce
  As defined in Nonce and Timestamp: https://oauth.net/core/1.0a/#nonce
- oauth_version
  This parameter is optional. If present, value MUST be “1.0”. Garmin assumes the protocol version is 1.0 if this parameter is not present.

An OAuth realm should **not** be specified.