Resources

Accessing resources

Resources are available on http://yogomee.com:8080

Each request must be signed as defined in **Signing Requests**, and contains the following parameters:

> *oauth_consumer_key*
>> The User's Login.
>
> *oauth_signature*
>> The signature as defined in **Signing Requests**.
>
> *oauth_timestamp*
>> As defined in **Nonce and Timestamp**.
>
> *oauth_nonce*
>> As defined in **Nonce and Timestamp**.

Response format

Each responce is JSON object, and contains the fillowing fields:

> *status*
>> request staus. «OK» or «FAIL»
>
> *data*
>> request result. JSON object or list.
>
> *time*
>> request time in seconds
>
> *error*
>> error description. Present if status is «FAIL»

Nonce and Timestamp

The timestamp is expressed in the number of seconds since January 1, 1970 00:00:00 GMT. The timestamp value must be a positive integer and must be equal or greater than the timestamp used in previous requests.

The nonce value is unique for all requests with that timestamp. A nonce is a random string, uniquely generated for each request.

Signing Requests

All Resources requests must be signed. The signature process encodes the User Secret into a verifiable value which is included with the request.

Signature Base String

The Signature Base String is a consistent reproducible concatenation of the request elements into a single string. The string is used as an input in HMAC-SHA1 signing algorithm. All the request parameters must be URL-encoded prior to constructing the Signature Base String.

Normalize Request Parameters

The request parameters are collected, sorted and concatenated into a normalized string.

The oauth_signature parameter must be excluded.

The parameters are normalized into a single string as follows:

1. Parameters are sorted by name, using lexicographical byte value ordering.

2. Parameters are concatenated in their sorted order into a single string. For each parameter, the name is separated from the corresponding value by an '=' character, even if the value is empty. Each name-value pair is separated by an '&' character

Construct Request URL

The Signature Base String includes the request absolute URL, tying the signature to a specific endpoint. The URL used in the Signature Base String must include the scheme, authority, and path, and must exclude the query and fragment.

Concatenate Request Elements

The following items must be concatenated in order into a single string. Each item is URL-encoded and separated by an '&' character, even if empty.

1. The HTTP request method used to send the request. Value must be uppercase.
2. The request URL.
3. The normalized request parameters string.

Generating Signature

The signature is generated using the HMAC-SHA1 signature algorithm where the Signature Base String is the *text* and the *key* is the concatenation of the encoded User Secret and '&' character.

oauth_signature is set to the calculated digest octet string, first base64-encoded, then URL-encoded.

Global resources

/me

Methods:

    GET

        Returns current user.

        Params:

        Returns:

            {user} - current user

/users

Methods:

    GET

        Returns users.

        Params:

        Returns:

            [{user}] – list of users

    POST

        Add new user.

        Only allowed for root.

        Params:

            {user} – user to add

        Returns:

            {user} – added user with id

User resources

{user}/profile

Examples:

    /me/profile

    /me/friends/*{id}*/profile

    /user/*{id}*/profile

Methods:

    GET

        Returns user's profile.

        Params:

        Returns:

            {profile} - current user's profile

    PUT

        Save user's profile.

        Only allowed to /me resource.

        Params:

            {profile} - profile to save

        Returns:

            {profile} - saved profile

{user}/messages

Examples:

    /me/messages

    /me/friends/*{id}*/messages

    /users/*{id}*/messages

Methods:

    GET

        Returns user's messgaes.

        Only allowed to /me resource.

        Params:

Returns:

> *[{message}]* - list of messages

POST

> Send new message.
> Not allowed to /me resource.
> **Params:**
>> *{message}* - message to send.
> **Returns:**
>> *{message}* – message

{user}/invitations

Examples:

> /me/invitations
> /me/friends/*{id}*/invitations
> /users/*{id}*/invitations

Methods:

GET

> Returns user's invitations
> Only allowed to /me resource.
> **Params:**
> **Returns:**
>> *[{invitation}]* - list of invitations

POST

> Send invitation to user.
> Not allowed to /me resource.
> **Params:**
>> *{invitation}* - invitation to send.
> **Returns:**
>> *{invitation}* - sent invitation.

{user}/friends

Examples:

> /me/friends
> /me/friends/*{id}*/friends
> /users/*{id}*/friends

Methods:

GET

> Returns users's friends.
> **Params:**
> **Returns:**
>> *[{user}]* - list of friends

{user}/gomees

Examples:

> /me/gomees
> /me/friends/*{id}*/gomees
> /users/*{id}*/gomees

Methods:

    GET

        Returns user's gomees.

        Params:

        Returns:

            [{gomee}] - list of gomees.

    POST

        Create new gomee.

        Only allowed to /me resource.

        Params:

            {gomee} - new gomee

        Returns:

            {gomee} - created gomee

{user}/targets

Examples:

    /me/targets

    /me/friends/*{id}*/targets

    /users/*{id}*/targets

Methods:

    GET

        Return current user's targets.

        Params:

        Returns:

            [{target}] - list of targets.

    POST

        Create new target.

        Only allowed to /me resource.

        Params:

            {target} - new target.

        Returns:

            {target} - created target.

Users list resources

[{user}]/{id}

Examples:

    /users/*{id}*

    /me/friends/*{id}*

    /me/friends/*{id}*/friends/*{id}*

    /users/*{id}*/friends/*{id}*

Methods:

    GET

        Returns user with id.

        Params:

        Returns:

            {user} - user

DELETE
> Remove user from list.
> Only allowed to /me/friends resource and to *{user}*/friends resource where *{id}* is mine id.
> **Params:**
> **Returns:**

Messages list resources

[{message}]/sent

Examples:
> /me/messages/sent

Methods:
GET
> Returns sent messages.
> Only allowed to /me resource.
> **Params:**
> **Returns:**
>> *[{message}]* - list of messages

[{message}]/received

Examples:
> /me/messages/received

Methods:
GET
> Returns received messages.
> Only allowed to /me resource.
> **Params:**
> **Returns:**
>> *[{message}]* - list of messages

[{message}]/*{id}*

Examples:
> /me/messages/*{id}*
> /me/messages/sent/*{id}*
> /me/messages/received/*{id}*

Methods:
GET
> Returns message with id.
> Only allowed to /me resource.
> **Params:**
> **Returns:**
>> *{message}* - message

PUT
> Update message.
> Only allowed to /me resource.
> **Params:**

> *{message}* - message to update.
>> **Returns:**
>>> *{message}* - updated message
>
> DELETE
>> Remove message.
>> Only allowed to /me resource.
>> Params:
>> Returns:

Invitation resources

{invitation}/accept

Examples:
> /me/invitations/*{id}*/accept
> /me/invitations/friends/*{id}*/accept
> /me/invitations/targets/received/*{id}*/accept
> /me/invitations/targets/received/waiting/*{id}*/accept

Methods:
> POST
>> Accept invitation.
>> Only allowed to /me resource.
>> Only allowed to received invitations.
>> Only allowed to waiting invitations.
>> **Params:**
>> **Returns:**

{invitation}/reject

Examples:
> /me/invitations/*{id}*/reject
> /me/invitations/friends/*{id}*/reject
> /me/invitations/targets/received/*{id}*/reject
> /me/invitations/targets/received/waiting/*{id}*/reject

Methods:
> POST
>> Reject invitation.
>> Only allowed to /me resource.
>> Only allowed to received invitations.
>> Only allowed to waiting invitations.
>> **Params:**
>> **Returns:**

Invitations list resources

[{invitation}]/sent

Examples:
> /me/invitations/sent
> /me/invitations/friends/sent

/me/invitations/targets/rejectd/sent

Methods:

GET

Returns sent invitations.
Only allowed to /me resource.
Params:
Returns:
[{invitation}] - list of invitations

[{invitation}]/received

Examples:

/me/invitations/received
/me/invitations/friends/received
/me/invitations/targets/rejectd/received

Methods:

GET

Retuns received invitations.
Only allowed to /me resource.
Params:
Returns:
[{invitation}] - list of invitations

[{invitation}]/friends

Examples:

/me/invitations/friends
/me/invitations/received/friends
/me/invitations/sent/accepted/friends

Methods:

GET

Returns invitations to be a friend.
Only allowed to /me resource.
Params:
Returns:
[{invitation}] - list of invitations

[{invitation}]/targets

Examples:

/me/invitations/targets
/me/invitations/received/targets
/me/invitations/sent/accepted/targets

Methods:

GET

Returns invitations to be a target.
Only allowed to /me resource.
Params:
Returns:
[{invitation}] - list of invitations

[{invitation}]/waiting

Examples:

/me/invitations/waiting
/me/invitations/friends/waiting
/me/invitations/targets/sent/waiting

Methods:

GET

Returns waiting invitations.
Only allowed to /me resource.
Params:
Returns:

[{invitation}] - list of invitations, that waiting to be accepted

[{invitation}]/accepted

Examples:

/me/invitations/accepted
/me/invitations/friends/accepted
/me/invitations/targets/sent/accepted

Methods:

GET

Returns accepted invitations.
Only allowed to /me resource.
Params:
Returns:

[{invitation}] - list of accepted invitation

[{invitation}]/rejected

Examples:

/me/invitations/rejected
/me/invitations/friends/rejected
/me/invitations/targets/sent/rejected

Methods:

GET

Returns rejected invitations.
Params:
Returns:

[{invitation}] - list of rejected invitations

[{invitation}]/{id}

Examples:

/me/invitations/*{id}*
/me/invitations/friends/*{id}*
/me/invitations/targets/sent/*{id}*
/me/invitations/friends/sent/waiting/*{id}*

Methods:

GET

Returns invitation with id.

Only allowed to /me resource.
Params:
Returns:
>> *{invitation}* – invitation

Gomees list resources

[{gomee}]/{id}

Examples:
> /me/gomees/*{id}*
> /users/*{id}*/gomees/*{id}*
Methods:
> GET
>> Returns gomee with id.
>> **Params:**
>> **Returns:**
>>> *{gomee}* - gomee
> PUT
>> Update gomee.
>> Only allowed to /me resource.
>> **Params:**
>>> *{gomee}* - gomee to update
>> **Returns:**
>>> *{gomee}* - updated gomee
> DELETE
>> Remove gomee.
>> Only allowed to /me resource.
>> **Params:**
>> **Returns:**

Targets list resources

[{target}]/{id}

Examples:
> /me/targets/*{id}*
> /users/*{id}*/targets/*{id}*
Methods:
> GET
>> Return target with id.
>> **Params:**
>> **Returns:**
>>> *{target}* - target.
> PUT
>> Update target.
>> Only allowed to /me resource.
>> **Params:**
>>> *{target}* - target to update

Returns:

{target} - updated target

DELETE

Remove target.

Only allowed to /me resource.

Params:

Returns:

Structures

{user}

Represents user

Example:

Fields:

id

type: *string*

login

type: *string*

secret

type: string

{profile}

Represents user profile

Example:

Fields:

first_name

type: *string*

last_name

type: *string*

email

type: *string*

{message}

Represents message

Example:

Fields:

id

type: *string*

from

       type: *{user}*

to

       type: *{user}*

msg

       type: *strin*g

status

       type: {message_status}

{invitation}

Represents ivitation

Example:

Fields:

id

       type: *string*

from

       type: *{user}*

to

       type: *{user}*

target_id

       type: *string*

type

       type: *{invitation_type}*

status

       type: *{invitation_status}*

{gomee}

Represents gomee

Example:

Fields:

id

       type: *string*

owner

       type: *{user}*

lat

       type: *double*

lng

        type: *double*

address

        type: *string*

description

        type: *string*

{target}

Represnts target

Example:

Fields:

id

        type: *string*

owner

        type: *{user}*

gomee

        type: *{gomee}*

creator

        type: *{user}*

type

        type: *{target_type}*

Enums

{message_status}

{invitation_type}

{invitation_status}

{target_type}