Methods

What are methods?

Methods are the basic constructs that allow you to perform actions or fetch information from Flock. For example, Flock has methods for each of the sample use cases listed below, among others.

- Sending a message to a user or a group;
- Fetching the list of a user's contacts;
- Fetching information about a user or a group.

Similar to methods and function calls in any programming language, Flock methods too receive a list of named parameters, and may optionally return a value.

List of methods available in FlockOS

- channels.addMembers
- channels.create
- channels.getInfo
- channels.list
- channels.listMembers
- channels.removeMembers
- chat.sendMessage
- chat.fetchMessages
- roster.listContacts
- users.getInfo
- users.getPublicProfile

Calling a method

Flock provides an HTTPS endpoint for each API method.

```
https://api.flock.co/v1/<method-name>
```

To call a method, send an HTTP `GET` or `POST` request to the method endpoint, alongwith its parameters (can be either `GET` or `POST`).

For example, to call chat.sendMessage, you can either send a `GET` request to following URL:

Method call using a GET request

```
https://api.flock.co/v1/chat.sendMessage?to=u:
513a9f4ae409&text=hello&token=632542db-1a23-4e06-8b39-01f8b0868d57
```

(See Authentication for more on the `token` parameter.)

Or you can send a `POST` request to `https://api.flock.co/v1/chat.sendMessage`:

Method call using a POST request

```
POST /v1/chat.sendMessage HTTP/1.1
Host: api.flock.co
Content-Type: application/x-www-form-urlencoded
Content-Length: 70

to=u:513a9f4ae409&text=hello&token=632542db-1a23-4e06-8b39-01f8b0868d57
```

If a method call is successful, the status code 200 is returned in the HTTP response.

Authentication

To call a method, an app will need to pass an authentication token on behalf of the user (either a normal user or a bot).

For normal users, you receive this token in the app.install event.

For a bot, you can see the token in the developer dashboard after you enable it.

This token is passed as the `token` parameter in the HTTP request that calls the method.

Return values

Yes, methods may optionally return a value. A response value is returned as a JSON document in the response body of the HTTP request.

HTTP response to a method call

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 51

{
    "uid": "d9e054e0-a679-4a7a-9cfa-7299669aa083"
}
```

Errors

If an error occurs while processing your request, an HTTP status code other than 200 is returned. Further information about the error is provided by a JSON in the HTTP response body.

Error response to a method call

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "error": "MissingParameter",
    "description": "One or more parameters required for this method call
are missing",
    "parameter": "text"
}
```

Two fields will always be present when an error is returned:

- `error` – an error code
- `description` – a human readable description of the error

Depending on the error code, additional fields may also be present in the JSON if they provide more context. For example, `parameter` is an additional field in the response above.

Below is a list of generic error codes that can be returned in response to a method call. Note that some methods can return other errors in addition to these. See individual method pages for more information.

Error Code	HTTP Status	Description	Additional Attributes
`InvalidRequestData`	400	The request was malformed	
`InvalidParameter`	400	A parameter for the method call is missing or invalid	• `parameter` – name of the missing paramenter
`InvalidToken`	403	An invalid authentication token was sent with the request	
`AppDisabled`	403	App has been disabled for this user	• `disabledBy` – one of `user` or `team`
`NoSuchMethod`	404	The method named in the request URL does not exist	
`HTTPMethodNotAllowed`	405	An HTTP method other than `POST` was used	
`RequestTimeout`	408	A timeout occured before the server could receive the complete request from the app	
`TooManyRequests`	429	Rate limit reached	
`InternalServerError`	500	An unexpected condition was encountered	
`ServiceUnavailable`	503	The API service is unavailable at this time	