Полный справочник по I-lacker Social API

Вижу, вы взяли себя в руки и дошли до 2 главы наших приключений?
Тогда встерчайте, *справочник*!
P.S. Если вы не читали предидущюю главу, то самое время это сделать, иначе вы не поймёте ничего тут.

Особенности API

Немного напомним о том, что:

- Слеш в конце URL обязателен
- Методы запроса, кроме `GET` выполняются через `POST` и заголовок `X-Http-Method-Override`
- Если нет возможности передать заголовок, то используется параметр `override_method`
  А теперь о том, о чём вы ещё не знаете:
- Все даты привидены в соответствии с RFC 850

ПРО ОТРИЦАТЕЛЬНЫЕ И ПОЛОЖИТЕЛЬНЫЕ ID

В этом прекрасном справочнике вы ещё не раз столкнётесь с этой терминологией.
Так вот, так уж случилось, что некоторые методы работают и с группами, и с пользователями.
Получают они, конечно, данные о пользователе или группе по id.
Поэтому чтобы различать ресурсы пользователя или группы, у групп id отрицателен.
Например: у пользователя с id 30, id - 30
а вот у группы с id 1, id будет -1

Это также касается альбомов или получателей сообщения.
Предостережение: не спешите всегда писать id группы/альбома со знаком минус, делайте это тогда, когда это сказано сделать.

Особенности этой документации

Документация закончилась на прошлой главе, пора расстаться с объяснением работы API, в первую очередь, это справочник.
Здесь будут приведены примеры использования методов и их краткое описание, но не больше.

Статус этого документа

Это - оффициальный справочник по I-lacker Social API, однако он не нормативен относительно стандартов.
Что это значит? А какая вам разница? Читайте дальше.

Знаете, мне в метро занятся нечем, есть ли у вас pdf этого справочника?

Именно её ты сейчас и читаешь, так что лучше не задавай этот вопрос, а то рискешь выглядеть как дебил.

TL;DR А где взять библиотеку под $ЛюбимыйЯзык ?

А зачем вам оно, вообще, надо? Но, так уж и быть, вот вам список **оффициально одобренных l-lackerom** библиотек:

- JavaScript (браузерный)
- PHP5 (не JPHP)

А ГДЕ ЕЩЁ МОЖНО ВЗЯТЬ БИБЛИОТЕКУ ПОД JAVASCRIPT?

- Yarn:

  ```
  yarn add socioko
  ```

  NPM:

  ```
  npm i socioko
  ```

  Unpkg: https://unpkg.com/socioko@1.1.0/index.js

Справочник

Получение сущностей

Что такое сущности? Это группа, ну или пользователь.

ПОЛУЧИТЬ ПОЛЬЗОВАТЕЛЯ

`GET /user/{id}/`
Успешное выполнение:

- Response 202 (application/json)

  ```
  {
        "state": "202:Success",
        "result": {

            "id": 1,
            "name": [
  ```

```
            "Александр",
            "Детилот"
        ],
        "gender": "male",
        "user_group": "ADMINISTRATOR",
        "verified": true,
        "ban": false,
        "dead": 0,
        "rating": 500,
        "have_donated": 0,
        "status": "",
        "last_action": "Sunday, 01-Jul-18 11:48:57 EDT",
        "avatar": {
            "any": "content/avatars/9970080465.jpg"
        }
    }
}
```


ПОЛУЧИТЬ ГРУППУ

`GET /club/{id}/`

Успешное выполнение:

- Response 202 (application/json)


```
{
    "state": "202:Success",
    "result": {
        "id": 2,
        "name": "/etc/コックのアニメ",
        "about": "Первый мемесовый паблик",
        "owner": 30,
        "verified": true,
        "ban": false,
        "dates": null,
        "nsfw": true,
        "avatar": {
            "any": "content/gavatars/199_1.jpg"
        }
    }
}
```


Друзья&Подписки

`id` - id сущности, в этом случае id у всех сущностей положителен.

ДРУЗЬЯ

GET `/friends/id` Получить друзей

Example URI

GET /friends/id

Response `202` Hide

Headers

```
Content-Type: application/json
```

Body


```
{
  "state": "202:Success",
  "result": {
    "0": 6,
    "1": 15,
    "2": 1,
    "3": 4,
    "4": 16,
    "5": 24,
    "6": 34,
    "7": 9,
    "8": 29,
    "9": 36
  }
}
```


ПОДПИСЧИКИ

GET `/followers/id` Получить подписчиков

Example URI

GET /followers/id

Response `202` Hide

Headers

```
Content-Type: application/json
```

Body


```
{
  "state": "202:Success",
  "result": {
    "0": 7
  }
}
```


ПОДПИСЧИКИ ГРУППЫ

GET `/subscribers/id` Получить подписчиков

Example URI

GET /subscribers/id

Response `202` Hide

Headers


```
Content-Type: application/json
```


Body


```
{
  "state": "202:Success",
  "result": {
    "0": 1,
    "1": 2,
    "2": 7,
    "3": 30,
    "4": 11,
    "5": 5,
    "6": 156,
    "7": 25,
    "8": 19
  }
}
```


Стена

ПОСТЫ

Посты, тут всё ясно. В id нужно передать id сущности. У пользователя id положительный, у группы отрицательный.

GET `/wall/id` Получение постов

ПРЕДУПРЕЖДЕНИЕ: с 1 июля, получение постов требует наличие валидного токена

Example URI

GET /wall/id

Response `202` Hide

Headers

```
Content-Type: application/json
```

Body


```
{
  "state": "202:Success",
  "result": {
    "0": {
      "id": 420,
      "text": "Обновление: с завтрашнего дня wall будет обязательно требов
    },
    "1": {
      "id": 407,
      "text": "Документация: <a href=\"http://l-lsoc.cf/api2/docs.html\">h
    },
    "2": {
      "id": 411,
      "text": "Небольшое обновление: API теперь доступен при помощи AJAX (
    },
    "3": {
      "id": 414,
      "text": "Теперь у нас так же есть API Playground!\n<br><br><a href=\
    },
    "4": {
      "id": 415,
      "text": "хуйня"
    }
  }
}
```


PUT `/wall/id` Создание поста

Example URI

PUT /wall/id

Request Hide

Body

```
{
  "post": {
    "text": "Hello, Write API!"
  }
}
```

Response `202` Show

Комментирование

В `target` передают тип объекта, а в `id` его id.
Список типов объекта: post, group (*посты в группе*), note, photo, video.
Внизу всё есть вроде.

КОММЕТАРИЙ

GET `/comment/target/id` Получить комментарий

Example URI

GET /comment/target/id

Response `202` Hide

Headers

```
Content-Type: application/json
```

Body

```
{
  "state": "202:Success",
  "result": {
    "id": 133,
```

```
    "target": 284,
    "author": 11,
    "created": "Saturday, 12-May-18 13:28:45 EDT",
    "text": "Альберт, хмм</b>"
  }
}
```


КОММЕНТАРИИ ПОД ПОСТАМИ

GET `/comments/post/id` Получение списка комментариев

Example URI

GET /comments/post/id

Response `202` Hide

Headers


```
Content-Type: application/json
```


Body


```
{
  "state": "202:Success",
  "result": {
    "0": {
      "id": 128
    },
    "1": {
      "id": 129
    },
    "2": {
      "id": 130
    },
    "3": {
      "id": 131
    },
    "4": {
      "id": 132
    },
    "5": {
      "id": 133
    }
  }
}
```

PUT /comments/post/id Написание комментария

Example URI

PUT /comments/post/id

Request Hide

Body

```
{
    "comment": {
        "target": 284, //id поста
        "text": "Just Monika."
    }
}
```

Response 202 Hide

Headers

```
Content-Type: application/json
```

Body

```
{
  "state": "202:Success",
  "result": {}
}
```

КОММЕНТАРИИ ПОД ПОСТАМИ ГРУППЫ

GET /comments/group/id Получение списка комментариев

Example URI

GET /comments/group/id

Response 202 Hide

Headers

```
Content-Type: application/json
```

Body


```
{
  "state": "202:Success",
  "result": {
    "0": {
      "id": 128
    },
    "1": {
      "id": 129
    },
    "2": {
      "id": 130
    },
    "3": {
      "id": 131
    },
    "4": {
      "id": 132
    },
    "5": {
      "id": 133
    }
  }
}
```


PUT `/comments/group/id` Написание комментария

Example URI

PUT /comments/group/id

Request Hide

Body

```
{
    "comment": {
        "target": 284, //id поста
        "text": "Just Monika."
    }
}
```

Response `202` Hide

Headers

```
Content-Type: application/json
```

Body


```
{
  "state": "202:Success",
  "result": {}
}
```


КОММЕНТАРИИ ПОД ЗАМЕТКАМИ

GET `/comments/note/id` Получение списка комментариев

Example URI

GET /comments/note/id

Response `202` Hide

Headers

```
Content-Type: application/json
```

Body


```
{
  "state": "202:Success",
  "result": {
    "0": {
      "id": 128
    },
    "1": {
      "id": 129
    },
    "2": {
      "id": 130
    },
    "3": {
      "id": 131
    },
    "4": {
      "id": 132
    },
```

```
    "5": {
      "id": 133
    }
  }
}
```


PUT `/comments/note/id` Написание комментария

Example URI

PUT /comments/note/id

Request Hide

Body


```
{
    "comment": {
        "target": 284, //id поста
        "text": "Just Monika."
    }
}
```


Response `202` Hide

Headers


```
Content-Type: application/json
```


Body


```
{
  "state": "202:Success",
  "result": {}
}
```


КОММЕНТАРИИ ПОД ВИДЕО

GET `/comments/video/id` Получение списка комментариев

Example URI

GET /comments/video/id

Response `202` Hide

Headers

```
Content-Type: application/json
```

Body


```
{
  "state": "202:Success",
  "result": {
    "0": {
      "id": 128
    },
    "1": {
      "id": 129
    },
    "2": {
      "id": 130
    },
    "3": {
      "id": 131
    },
    "4": {
      "id": 132
    },
    "5": {
      "id": 133
    }
  }
}
```


PUT `/comments/video/id` Написание комментария

Example URI

PUT /comments/video/id

Request Hide

Body

```
{
    "comment": {
        "target": 284, //id поста
```

```
        "text": "Just Monika."
    }
}
```

Response `202` Hide

Headers

```
Content-Type: application/json
```

Body

```
{
  "state": "202:Success",
  "result": {}
}
```

КОММЕНТАРИИ ПОД ФОТОГРАФИЯМИ

GET `/comments/photo/id` Получение списка комментариев

Example URI

GET /comments/photo/id

Response `202` Hide

Headers

```
Content-Type: application/json
```

Body

```
{
  "state": "202:Success",
  "result": {
    "0": {
      "id": 128
    },
    "1": {
      "id": 129
    },
    "2": {
      "id": 130
```

```
    },
    "3": {
      "id": 131
    },
    "4": {
      "id": 132
    },
    "5": {
      "id": 133
    }
  }
}
```


PUT `/comments/photo/id` Написание комментария

Example URI

PUT /comments/photo/id

Request Hide

Body

```
{
    "comment": {
        "target": 284, //id поста
        "text": "Just Monika."
    }
}
```

Response `202` Hide

Headers

```
Content-Type: application/json
```

Body


```
{
  "state": "202:Success",
  "result": {}
}
```

Заметки

`id` - id заметки

ПОЛУЧИТЬ ЗАМЕТКУ ПО ID

GET `/note/id` Получить заметку по id

Example URI

GET /note/id

Response `202` Hide

Headers

```
Content-Type: application/json
```

Body


```
{
  "state": "202:Success",
  "result": {
    "id": 46,
    "name": "test.php",
    "author": 7,
    "edited": false,
    "created": "Monday, 07-May-18 15:51:07 EDT",
    "text": "&lt;br&gt;"
  }
}
```


РАБОТА С ЗАМЕТКАМИ

Здесь `id` - это id пользователя, чьи заметки мы собираемся получить (или изменить, правда изменять можно только свои)

GET `/notes/id` Получить список заметок пользователя

Example URI

GET /notes/id

Response `202` Hide

Headers

```
Content-Type: application/json
```

Body


```
{
  "state": "202:Success",
  "result": {
    "0": {
      "id": 47
    },
    "1": {
      "id": 48
    }
  }
}
```


Видео

ПОЛУЧЕНИЕ ОДНОГО ВИДЕО

GET `/video/id` Получить видео

Example URI

GET /video/id

Response `202` Hide

Headers

```
Content-Type: application/json
```

Body


```
{
  "state": "202:Success",
  "result": {
    "id": 2557,
    "name": "Kasane Teto (All VB's)「Love is War (Mwk Remix)」UTAUカバー",
    "author": 30,
```

```
    "url": "https://youtu.be/er-Sw2b-XVM",
    "description": "Write API test",
    "category": "GENERAL",
    "banned": false,
    "created": "Saturday, 30-Jun-18 05:30:59 EDT"
  }
}
```


ПОЛУЧЕНИЕ СПИСКА ВИДЕО

Тут `id` - это id пользователя

GET `/videos/id` Получить список видео

Example URI

GET /videos/id

Response `202` Hide

Headers


```
Content-Type: application/json
```


Body


```
{
  "state": "202:Success",
  "result": {
    "0": {
      "id": 2525
    }
  }
}
```


PUT `/videos/id` Добавить видео

Example URI

PUT /videos/id

Request Hide

Body

```
{
  "video": {
    "title": "Kasane Teto (All VB's)「Love is War (Mwk Remix)」UTAUカバー"
    "url": "https://www.youtube.com/watch?v=er-Sw2b-XVM",
    "description": "Write API test"
  }
}
```


Response `202` Hide

Headers


```
Content-Type: application/json
```


Body


```
{
  "state": "202:Success",
  "result": {}
}
```


Сообщения$^{\beta}$

Тут `id` - это id сообщения,
`from` - отправитель,
`to` - получатель

ПОЛУЧЕНИЕ ОДНОГО СООБЩЕНИЯ

GET `/message/id` Получить сообщение

Example URI

GET /message/id

Response `202` Hide

Headers


```
Content-Type: application/json
```

Body


```
{
  "state": "202:Success",
  "result": {
    "id": 466,
    "from": 30,
    "to": 1,
    "sent": "Friday, 29-Jun-18 15:10:14 EDT",
    "red": true,
    "content": {
      "subject": "ъ",
      "text": "ъ"
    }
  }
}
```


РАБОТА С ПЕРЕПИСКАМИ

Да, мы сами понимаем какая система неудобная, но, к сожалению, вам придётся с этим смириться.
Но мы уже работаем над исправлением этой проблемы!

GET `/messages/from/to` Получить переписку

Example URI

GET /messages/from/to

Response `202` Hide

Headers


```
Content-Type: application/json
```


Body


```
{
  "state": "202:Success",
  "result": {
    "0": {
      "id": 466
    },
    "1": {
      "id": 464
    },
    "2": {
```

```
      "id": 462
    },
    "3": {
      "id": 458
    },
    "4": {
      "id": 457
    }
  }
}
```


PUT `/messages/from/to` Отправить сообщение

Example URI

PUT /messages/from/to

Request Hide

Body


```
{
  "message": {
    "subject": "Восстание против l-lackera",
    "text": "Сегодня в 23:00 в группе /testpool, присоединяйтесь к нам."
  }
}
```


Response `202` Hide

Headers


```
Content-Type: application/json
```


Body


```
{
  "state": "202:Success",
  "result": {}
}
```

Фотографии

Загрузка фотографий очень сложный процесс, поэтому объяснения о том, как работает метод `PUT photo` будут в следующем параграфе.

СПИСОК АЛЬБОМОВ

Здесь `id` - id сущности, чьи альбомы мы хотим получить (или изменить)
Здесь дейвствует стандартное правило про id сущностей. Напомним, у пользователя id положителен, а у группы нет.

GET `/albums/id` Получить список

Example URI

GET /albums/id

Response `202` Hide

Headers

```
Content-Type: application/json
```

Body


```
{
  "state": "202:Success",
  "result": {
    "0": {
      "id": 14,
      "name": "Тестовый альбом",
      "description": "Не заходить. Или заходить, я всего лишь надпись, не
      "created": "Saturday, 30-Jun-18 06:01:22 EDT"
    }
  }
}
```


PUT `/albums/id` Создать альбом

Example URI

PUT /albums/id

Request Hide

Body


```
{
  "album": {
    "title": "Тестовый альбом",
    "description": "Не заходить. Или заходить, я всего лишь надпись, не ко
  }
}
```


Response `202` Hide

Headers


```
Content-Type: application/json
```


Body


```
{
  "state": "202:Success",
  "result": {}
}
```


СПИСОК ФОТОГРАФИЙ В АЛЬБОМЕ

Здесь `id` - id альбома.
Стоит отметить, что id альбомов у групп отрицателен.

GET `/album/id` Получить список

Example URI

GET /album/id

Response `202` Hide

Headers


```
Content-Type: application/json
```


Body


```
{
  "state": "202:Success",
```

```
  "result": {
    "0": {
      "id": 89
    }
  }
}
```


ФОТОГРАФИЯ

Здесь `id` - id фотографии

GET `/photo/id` Получить фотографию

Example URI

GET /photo/id

Response `202` Hide

Headers


```
Content-Type: application/json
```


Body


```
{
  "state": "202:Success",
  "result": {
    "id": 72,
    "author": 30,
    "path": "content/img-albums/8269339925.jpg",
    "description": "",
    "album": 89968,
    "created": "Friday, 01-Jun-18 16:48:22 EDT"
  }
}
```


PUT `/photo/id` Добавление фотографии

Здесь `id` - id альбома.
Стоит отметить, что id альбомов у групп отрицателен.

Example URI

PUT /photo/id

Request Hide

Body


```
{
  "photo": {
    "server": "nfs://username@l-lsoc.cf",
    "description": "Моя первая картинка",
    "photo": "FFD8FFE000104A4649460001010000010001000FFDB0043000503040404
  }
}
```


Response `202` Hide

Headers

```
Content-Type: application/json
```

Body


```
{
  "state": "202:Success",
  "result": {}
}
```


Заполучение сервера загрузки

И это последний и самый сложный раздел справочника. И именно поэтому он не будет таким строгим и лаконичным.
Сейчас мы с вами попытаемся разобраться, как же загрузить фотографию в l-lacker social?

На самом деле, всё чуток сложней чем вы думаете, но всё же, вот список действий которые вам необходимо предпринять:

- создать сервер загрузки
- перекодировать фотографию в hex
- создать альбом если его ещё нет
- загрузить фотографию в альбом (см. предидущий параграф)
  Сложности, конечно возникнут именно на 2 этапе.
  Если вы используете PHP4+, то вам повезло, вы сможете перекодировать фотографию такой штукой: `bin2hex(file_get_contents("фото.png"))`

Для других языков, чётких инструкций нет, так что гуглите, решение уже есть наверняка, ведь мы не первые кто собрались кодировать файлы в hex.

Но допустим, мы с этим разобрались, что дальше-то?
Читайте справочник к методам, рано или поздно вы всё поймёте.
Подсказка: hex-строка идёт в поле `photo`, а в поле сервер кладём `nfs://user@l-lsoc.cf`, где `user` - ваш id сервера, который вы получили.

ПОЛУЧЕНИЕ СЕРВЕРА

GET `/get_upload_server` Получить сервер

Example URI

GET /get_upload_server

Response `202` Hide

Headers

```
Content-Type: application/json
```

Body

```
{
  "state": "202:Success",
  "result": {
    "upload_server": {
      "consumes": [
        "*"
      ],
      "for": "<ВАШ_ТОКЕН>",
      "id": "<ID_СЕРВЕРА>"
    }
  }
}
```