19.12.2021

Платформа OMMG 3.0

Описание API

Георгий Хазан
ОММДЖИ ТЕХНОЛОДЖИ

Оглавление

1. Общая идеология OMMG API

OMMG API (далее – API) использует только HTTPS запросы, направляемые на поддомен api того домена, на котором разворачивается платформа OMMG. Например, для домена `yourdomain.com` все запросы надо посылать на поддомен `https://api.yourdomain.com`

В описании запроса всегда указывается относительный путь к нужному URL, начиная от корня сайта `api.yourdomain.com`. Так, если в описании указано, что надо запросить `/message/send`, то запрос надо отправить на `https://api.yourdomain.com/message/send`

Все запросы имеют тип POST, содержат тело запроса в формате json, таким образом в запросе обязательно должно быть указано поле
`>Content-type: application/json`

Все запросы требуют наличия строки авторизации в формате HTTP Basic Auth, если явно не указано обратное. В качестве пароля необходимо передавать md5 hash от склейки строк пароля и идентификатора устройства (`globalId`), использованного при регистрации устройства. Например, для пароля `mypass` и идентификатора устройства `ios4698237469582345` надо в качестве пароля передать

```
>select md5('mypass'+'ios4698237469582345') as hash;
----------------------------------
| hash
----------------------------------
| cfcd208495d565ef66e7dff9f98764da
----------------------------------
```

Ответ присылается сервером также в формате json. В случае успешного завершения возвращается код ошибки 200, а в json появляются поля `"status":"ok"` и `"statusCode":200`.

В случае возникновения ошибки присылается код ошибки (40x, 50x), в поле `status` указывается непосредственная причина ошибки, а в поле `statusCode` – целочисленный расширенный код ошибки, полный список которых указан в Приложении 1.

В том случае, если по итогам выполнения запроса рассылается уведомление на другие устройства того же пользователя, к ним добавляется поле `from_global_id`, куда помещается код устройства, инициировавшего операцию. Таким образом устраняется дублирование, например, удаления сообщений. Соответственно, по наличию или отсутствию этого поля клиент может судить о том, входящее это уведомление или исходящее.

2. Используемые сокращения и определения

INT – поле целого типа, 32 бита, знаковое.
SID – серверный идентификатор, поле целого типа, 64 бита, знаковое.
JID – идентификатор пользователя вида login@domain, в формате utf8
GUID – строка из 32 шестнадцатеричных символа, записанных подряд
STRING – произвольная строка в формате utf8
TIME_STAMP – время в формате Unix time 64 (число миллисекунд с 01.01.1970)
BOOL – целое число, которое может принимать значения 0 или 1
bool – поле в json, может принимать значения `false` или `true`.

STATUS – строковая константа, код статуса пользователя:

- "1": OFFLINE;
- "3": VISIBLE;
- "5": INVISIBLE;
- "7": AWAY;
- "8": OCCUPIED

PLATFORM_TYPE – целое число, тип платформы для клиентского приложения:

- 0: Windows;
- 1: IPHONE;
- 2: Android;
- 3: MAC;
- 4: WinPhone;
- 5: Web.

CONTACT_FLAGS – целое число (32 бита), представляющее собой набор флагов для контакта в списке контактов или группового чата, участником которого вы являетесь. Смысл нижеследующих флагов не может быть изменён, все остальные битовые маски доступны для переопределения в клиентском приложении:

- 0x00000001 – контакт или групчат прикреплён к верху списка контактов.

GROUPCHAT_FLAGS – целое число (32 бита), представляющее собой набор свойств группового чата (применяется для создания, модификации или получения информации).

- 0x00000001 – публичный групповой чат (в него может войти любой посторонний контакт);
- 0x00000002 – модерируемый групповой чат (с разделением участников на роли);
- 0x00000004 – история группового чата не копируется участникам (рекомендуется для каналов).

RESPONSE_TYPE – целое число, тип входящего события

- 101: изменение статуса сессии;
- 102: изменение ника;

- 103: изменение аватара;
- 104: изменение списков видимости;
- 198: вход с другого устройства;
- 199: выход с другого устройства;
- 201: приглашение в групповой чат;
- 202: присоединение к групповому чату;
- 203: выбрасывание из группового чата;
- 204: выход контакта из комнаты;
- 205: вход контакта в комнату;
- 206: изменение параметров группового чата;
- 207: удаление чата;
- 301: приглашение в ростер;
- 302: получение положительного ответа на приглашение в ростер;
- 303: получение отрицательного ответа на приглашение в ростер;
- 304: удаление контакта из списка контактов;
- 306: изменение списка контактов;
- 307: новый зарегистрировавшийся контакт из телефонной книги;
- 308: обновление телефонной книги;
- 309: обновление никнейма контакта;
- 310: изменение команды;
- 401: новое сообщение;
- 402: изменение сообщения;
- 403: удаление сообщения;
- 404: удаление всех сообщений;
- 405: уведомление о доставке сообщения;
- 406: уведомление о прочтении сообщения;
- 407: пользователь начал набор текста;
- 408: пользователь закончил набор текста;
- 409: аудио- или видео- сообщение прослушано.

CHAT_STATUS – целое число, статус участника в групповом чате;

- 0: оффлайн;
- 1: онлайн;
- 2: исключён из чата

3. Авторизация и управление устройствами

3.1. Регистрация нового устройства.

Регистрация нового устройства проходит в два этапа. Сначала серверу выставляется запрос на регистрацию, в котором указывается мобильный телефон. В ответ на это сервер высылает на этот номер телефона смс с кодом, после чего клиент делает тот же самый вызов с указанием присланного кода. Код смс является временным и действует на протяжении 2 часов (если в настройках сервера не задано другое значение), после этого код надо запросить заново.

В параметре `lang` указывается двухбуквенный код языка, например, "en" или "ru".

Вызов не требует авторизации.

Запрос:

```
/register
```

Тело запроса:

```
{
  "login": STRING,           // логин пользователя в виде строки
  "globalId": GUID,          // идентификатор устройства (например, хеш IMEI)
  "name": STRING,            // наименование устройства
  "platform": PLATFORM_TYPE, // тип платформы
  "lang": STRING,            // язык сообщений для смс и голосовых звонков
  "code": STRING             // присланный код регистрации, опционально
}
```

Ответ (если параметр code пропущен):

```
{
  "status": "ok",
  "statusCode": 200,
  "sms_sent": 1
}
```

Ответ (если параметр code указан):

```
{
  "status": "ok",
  "statusCode": 200,
  "password": STRING         // сгенерированный пароль для устройства
}
```

3.2. Создание сессии.

Перед тем, как отправить любой другой запрос, необходимо войти на сервер и выставить свой статус в любое значение, отличное от STATUS.OFFLINE. Без этого все запросы будут отвергаться с кодом ошибки 3001 (неверный логин или пользователь). В дальнейшем, если интервал между запросами превысит 30 минут, сессия может быть автоматически сброшена сервером, тогда, при получении кода ошибки 3001, надо послать этот запрос на сервер, дождаться положительного ответа и послать исходный запрос ещё раз.

Запрос:
```
/logIn
```

Тело запроса:
```
{
  "status": STATUS,          // выставляемый статус сессии, по умолчанию VISIBLE.
  "textStatus": STRING       // статусное сообщение, опционально
}
```

Ответ:
```
{
  "status": "ok",
  "statusCode": 200,
  "lastSid": SID,            // последний прочитанный устройством идентификатор
события
  "user": {
    "firstName": STRING,     // имя пользователя
    "lastName": STRING,      // фамилия пользователя, опционально
    "avatar": STRING,        // хеш аватара
    "status": STATUS,        // статус сессии
    "textStatus": STRING     // статусное сообщение, опционально
  }
}
```

После успешного входа на сервер всем контактам из списка контактов, а также всем другим устройствам этого пользователя, будет разослано следующее уведомление:
```
{
  "type": "101",             // тип уведомления – смена статуса
  "from": JID,               // ваш идентификатор пользователя
  "ts": TIME_STAMP,          // серверная метка времени
  "status": STATUS,          // статус сессии
  "platform": PLATFORM_TYPE  // тип платформы пользователя
}
```

3.3. Регистрация пуш-уведомлений

В том случае, если клиент стартует с платформы, поддерживающей пуш-уведомления (iOS, Mac, Android), необходимо передать на сервер специальный платформо-зависимый ключ (токен), отвечающий за доставку пушей в данное приложение.

В поле флагов можно указывать следующие комбинации битов:

```
FLAG_PRIVATE_MESSAGE   = 0x0001; // уведомлять о приходе приватных сообщений
FLAG_HIGHLIGHT_MESSAGE = 0x0002; // уведомлять о подсветке имени в сообщении чата
FLAG_CHAT_MESSAGE      = 0x0004; // уведомлять обо всех сообщения в чатах
FLAG_CUT_PRIVATE       = 0x0008; // обрезать текст приватных сообщений
FLAG_CUT_CHAT              = 0x0010; // обрезать текст сообщений в групчатах
FLAG_SOUND_PRIVATE     = 0x0020; // звуковое уведомление о приватных сообщениях
FLAG_SOUND_CHAT        = 0x0040; // звуковое уведомление о сообщениях в групчатах
FLAG_SOUND_ACTIVATE    = 0x0080; // звуковое уведомление об активации другого
устройства
FLAG_ONLINE_MESSAGE    = 0x0100; // уведомлять о выходе контактов в онлайн
FLAG_INVITE            = 0x0200; // уведомлять о приходе приглашений на дружбу
FLAG_JOINED            = 0x0400; // уведомлять о регистрации контакта из телефонной
книги
FLAG_INTERACTIVE_PUSH  = 0x0800; // флаг поддержки интерактивных пушей (Андроид)
```

Запрос:

```
/setToken
```

Тело запроса:

```
{
  "token": STRING,            // токен устройства
  "platform": PLATFORM_TYPE,  // код платформы, передаётся как строка
  "flags": INT,               // комбинация флагов для управления пушами
}
```

Ответ:

```
{
  "status": "ok",
  "statusCode": 200
}
```

3.4. Чтение входящих сообщений в сессии

Для чтения входящих сообщений необходимо послать HTTP запрос, который ожидает ответа сервера не свыше 30 секунд, после чего сервер принудительно разрывает сокет. Если данные от сервера приходят раньше, то сервер посылает данные в сокет и разрывает соединение сразу же, после чего клиент должен послать ещё один такой же запрос. В блоке history возвращаются все сообщения

Запрос:
/pollEvents

Тело запроса:
```
{
  "lastSid": SID,  // идентификатор последнего прочитанного сообщения или 0
  "nowait": BOOL   // опционально, сбрасывать соединение сразу же в любом случае
}
```

Ответ:
```
{
  "status": "ok",
  "statusCode": 200,
  "data": {
    "history": [
    {
      "type": RESPONSE_TYPE, // тип события
       …                     // данные, относящиеся к данному событию
      "ts":TIME_STAMP,       // серверное время сообщения
    }],
    "lastSid":SID            // идентификатор последнего присланного сообщения
  }
}
```

3.5. Окончание сессии

Сервер автоматически переводит устройство в оффлайн, если не получает от него очередной запрос на pollEvents в течение заданного на сервере интервала времени. Однако рекомендуется по возможности насильственно завершать сессию, не дожидаясь таймаута. Также с помощью этого запроса можно отключить устройство или разрушить учетную запись пользователя целиком.

Запрос:
/logOut

Тело запроса:
```
{
  "removeAccount": BOOL,      // надо ли убить учётную запись, опционально
  "killUser": BOOL,           // надо ли убить устройство, опционально
  "device": GUID              // идентификатор другого устройства, опционально
}
```

Ответ:
```
{
  "status": "ok",
```

```
  "statusCode": 200
}
```

После окончания сессии всем контактам из списка контактов, а также всем другим устройствам этого пользователя, будет разослано следующее уведомление:

```
{
  "type": "101",              // тип уведомления – смена статуса
  "from": JID,                // ваш идентификатор пользователя
  "ts": TIME_STAMP,           // серверная метка времени
  "status": STATUS,           // статус сессии
  "platform": PLATFORM_TYPE   // тип платформы пользователя
}
```

3.6. Изменение имени пользователя

Считается, что имя задается в формате Имя + Фамилия, однако поле фамилии может не заполняться.

Запрос:

```
/changeName
```

Тело запроса:

```
{
  "firstName": STRING,        // имя пользователя
  "lastName": STRING          // фамилия пользователя, опционально
}
```

Ответ:

```
{
  "status": "ok",
  "statusCode": 200
}
```

После успешного изменения имени и/или фамилии всем контактам из списка контактов, а также всем другим устройствам этого пользователя, будет разослано следующее уведомление:

```
{
  "type": "102",              // тип события – изменение имени
  "firstName": STRING,        // имя пользователя
  "lastName": STRING,         // фамилия пользователя
  "ts": TIME_STAMP,           // серверное время события
}
```

3.7. Изменение статуса сессии

Данный запрос применяется для того, чтобы изменить текстовый статус сессии устройства или изменить режим сессии на любой, кроме STATUS.OFFLINE.

Запрос:
```
/changeStatus
```

Тело запроса:
```
{
  "status": STATUS,          // статус сессии
  "textStatus": STRING       // статусное сообщение, опционально
}
```

Ответ:
```
{
  "status": "ok",
  "statusCode": 200
}
```

После успешного изменения имени и/или фамилии всем контактам из списка контактов, а также всем другим устройствам этого пользователя, будет разослано следующее уведомление:
```
{
  "type": "101",             // тип уведомления – смена статуса
  "from": JID,               // ваш идентификатор пользователя
  "ts": TIME_STAMP,          // серверная метка времени
  "status": STATUS,          // новый статус сессии
  "textStatus": STRING,      // текстовый статус
  "platform": PLATFORM_TYPE  // тип платформы устройства
}
```

3.8. Получение списка устройств

Данный запрос возвращает список всех устройств, присоединенных к данной учетной записи, вместе с их статусами.

Запрос:
```
/devicesInfo
```

Тело запроса:
```
{}
```

Ответ:
```
{
  "status": "ok",
  "statusCode": 200,
  "data": [
  {
    "global_id": GUID,       // идентификатор устройства
    "status": STATUS         // статус устройства
```

```
    },
    {…}
    ]
}
```

4. Работа с сообщениями

4.1. Написание нового сообщения

Отсылает массив сообщений на сервер (одновременно можно передать несколько сообщений). Клиент передаёт для каждого сообщения локальный идентификатор, уникальный для данного устройства. Этот идентификатор потом используется, чтобы соотнести сообщению серверный идентификатор сообщения после успешной доставки сообщения на сервер.

Если каждое сообщение в пакете надо вставить в определённом порядке, надо внутри сообщения указать целочисленный номер в поле `order`, в противном случае сообщения могут быть вставлены в произвольном порядке.

Запрос:

```
/message/send
```

Тело запроса:

```
{
  "deviceType": STRING,       // имя устройства, с которого отсылается сообщение
  "messages": [
  {
    "to": JID,                // идентификатор пользователя-адресата
    "body": STRING,           // текст сообщения
    "localId": GUID,          // локальный идентификатор сообщения
    "order": INT              // порядковый номер сообщения, опционально
  },
  {…}
  ]
}
```

Ответ:

```
{
  "status": "ok",
  "statusCode": 200,
  "data": [
  {
    "sid": SID,               // серверный идентификатор сообщения
    "utc": TIME_STAMP         // серверное время приёма сообщения
    "localId": GUID           // локальный идентификатор сообщения
  },
  {…}
  ]
}
```

После приема сообщения сервером адресату доставляется следующее уведомление:

```
{
   "type": "401",              // тип события – новое сообщение
   "body": STRING,             // текст сообщения
   "ts": TIME_STAMP,           // серверное время события
   "sid": SID,                 // серверный идентификатор события
   "with": JID,                // идентификатор автора сообщения
   "localId": GUID,            // локальный идентификатор сообщения
   "deviceType": STRING        // имя устройства, с которого пришло сообщение
}
```

На остальные устройства автора сообщения также рассылается аналогичное уведомление, с той только разницей, что в поле with указывается не собственный идентификатор пользователя, а идентификатор адресата.

После того, как сервер заканчивает обработку нового сообщения и архивирует его в историю сообщений пользователя, сервер подтверждает автору сообщения доставку на сервер, высылая следующее уведомление:

```
{
   "type":"405",               // тип события – доставка на сервер
   "with": JID,                // идентификатор собеседника
   "sid": SID,                 // серверный идентификатор сообщения
   "ts": TIME_STAMP            // серверное время события
}
```

4.2. Редактирование сообщения

Любое сообщение может быть отредактировано его автором. На сервере в истории сообщений оно помечается особым образом.

Запрос:

```
/message/edit
```

Тело запроса:

```
{
   "sid": STRING,              // серверный идентификатор сообщения
   "jid": JID,                 // идентификатор пользователя-адресата
   "body": STRING              // текст сообщения
}
```

Ответ:

```
{
   "status": "ok",
   "statusCode": 200
}
```

После редактирования и автор, и адресат сообщения получают уведомление о редактировании:

```
{
  "type": "402",              // тип события – редактирование сообщения
  "body": STRING,             // текст сообщения
  "ts": TIME_STAMP,           // серверное время события
  "sid": SID,                 // серверный идентификатор события
  "from": JID,                // идентификатор автора сообщения
  "localId": GUID,            // локальный идентификатор сообщения
  "deviceType": GUID          // имя устройства, с которого пришло сообщение
}
```

4.3. Удаление сообщения

Для выбранного собеседника может быть удалено как одно-единственное сообщение, так и вся история целиком. В случае полного удаления истории поле `exact-sid` не передаётся.

Сообщение может удаляться как для обеих сторон, так и только для отправителя, это контролируется параметром `forMe` (по умолчанию сообщение удаляется для обеих сторон). В случае группового удаления истории история всегда удаляется только для отправителя.

Запрос:

```
/message/remove
```

Тело запроса:

```
{
  "exact-sid": STRING,        // серверный идентификатор сообщения, опционально
  "jid": JID,                 // идентификатор пользователя-адресата, опционально
  "forMe": BOOL,              // удалять сообщения только для меня, по умолчанию 0
  "removeAll": BOOL           // удаление всей истории сообщений, по умолчанию 0
}
```

Ответ:

```
{
  "status": "ok",
  "statusCode": 200
}
```

После успешного удаления сообщения и автор, и адресат получают следующее уведомление:

```
{
  "type":"403",               // тип события – удаление сообщения
  "ts": TIME_STAMP,           // серверное время события
  "sid": SID,                 // идентификатор события об удалении
  "with": JID,                // идентификатор собеседника
  "deleted": SID              // серверный идентификатор сообщения
}
```

Если производится массовое удаление, то в поле `deleted` сервер возвращает диапазон идентификаторов сообщений в формате "SID1, SID2".

4.4. Групповое удаление сообщений

Если возникает нужна удалить сразу несколько сообщений, например, отмеченных пользователем, то можно воспользоваться функцией группового удаления. Для этого нужно передать массив идентификаторов сообщений.

Запрос:
```
/message/removeGroup
```

Тело запроса:
```
{
  "forMe": BOOL,              // удалять сообщения только для меня, по умолчанию 0
  "messages" : [
  {
    "sid": SID,               // серверный идентификатор сообщения
    "localId": GUID,          // локальный идентификатор сообщения
    "jid": JID                // идентификатор пользователя-адресата
  },
  …
  ]
}
```

Ответ:
```
{
  "status": "ok",
  "statusCode": 200,
  "data":[
    {
      "sid": SID,             // идентификатор сообщения об удалении
      "deleted": SID,         // идентификатор удалённого сообщения
      "localId": GUID         // локальный идентификатор удалённого сообщения
    }
  ]
}
```

После успешного удаления сообщения и автор, и адресат получают следующее уведомление:
```
{
  "type":"403",               // тип события – удаление сообщения
  "ts": TIME_STAMP,           // серверное время события
  "sid": SID,                 // идентификатор события об удалении
  "with": JID,                // идентификатор собеседника
  "deleted": SID              // серверный идентификатор сообщения
```

```
}
```

4.5. Уведомление о наборе текста

Запрос:
```
/message/notifyTyping
```

Тело запроса:
```
{
  "to": JID,                  // идентификатор пользователя-адресата
  "finished": BOOL            // закончен ли набор текста
}
```

Ответ:
```
{
  "status": "ok",
  "statusCode": 200
}
```

Пользователь, указанный в поле to, получает следующее уведомление о начале набора текста:
```
{
  "type":"407",               // тип события – начало набора текста
  "from": JID,                // идентификатор собеседника
  "ts": TIME_STAMP            // серверное время события
}
```

После окончания набора тип уведомления меняется:
```
{
  "type":"408",               // тип события – окончание набора текста
  "from": JID,                // идентификатор собеседника
  "ts": TIME_STAMP            // серверное время события
}
```

4.6. Уведомление о доставке

После получения сообщения клиент может подтвердить факт доставки сообщения на клиент, это нужно для того, чтобы отославшая сообщение сторона могла корректно отобразить факт доставки в интерфейсе (т.н. "вторая галочка").

Обратите внимание на то, что аргументом вызова является массив пар JID/SID, который позволяет известить о доставке сообщений сразу нескольким адресатам.

Запрос:
```
/message/notifyDelivered
```

Тело запроса:

```
{
  "messages":
  {
    JID1: SID1,              // JIDn - идентификатор пользователя-адресата
    JIDn: SIDn               // SIDn - идентификатор доставленного сообщения
  }
}
```

Ответ:

```
{
  "status": "ok",
  "statusCode": 200
}
```

Собеседнику на все устройства доставляется следующее уведомление:

```
{
  "type":"405",              // тип события – уведомление о доставке
  "with": JID,               // идентификатор собеседника
  "sid": SID                 // серверный идентификатор сообщения
  "ts": TIME_STAMP           // серверное время события
}
```

4.7. Уведомление о прочтении

После того, как клиент показывает сообщение на экране, он обязан уведомить сервер о его прочтении, используя серверный идентификатор сообщения. Все сообщения с меньшим идентификатором также считаются прочтенными.

Запрос:

```
/message/notifyRead
```

Тело запроса:

```
{
  "jid": JID,                // идентификатор пользователя-адресата
  "sid": SID                 // идентификатор прочтённого сообщения
}
```

Ответ:

```
{
  "status": "ok",
  "statusCode": 200
}
```

Собеседнику на все устройства доставляется следующее уведомление:

```
{
  "type":"406",              // тип события – уведомление о прочтении
```

```
  "with": JID,                // идентификатор собеседника
  "sid": SID                  // серверный идентификатор сообщения
  "ts": TIME_STAMP            // серверное время события
}
```

4.8. Получение истории сообщений

При запросе истории параметр `talker` может принимать следующие значения:

- `"all": STRING` – получить историю контакта полностью;
- `"chats": STRING` – получить только историю групповых чатов, полностью;
- `"contacts": STRING` – получить только историю контактов, полностью;
- `[JID, JID]` – массив из одного или нескольких идентификаторов пользователей, для которых надо считать историю сообщений

Запрос:

```
/message/history
```

Тело запроса:

```
{
  "talker": JID / STRING,     // идентификатор пользователя-собеседника
  "exact-id": SID,            // идентификатор сообщения для получения, опционально
  "since-id": SID,            // стартовый идентификатор сообщения, опционально
  "to-id": SID,               // конечный идентификатор сообщения, опционально
  "since-time": TIME_STAMP,   // начальное время сообщения для поиска, опционально
  "to-time": TIME_STAMP,      // конечное время сообщения для поиска, опционально
  "limit": INT,               // максимальное количество сообщений в списке, по
                              // умолчанию 50
  "latest": BOOL,             // сортировка сообщений в сторону уменьшения SID,
                              // опционально
  "countOnly": BOOL,          // вести только подсчёт сообщений, опционально
  "team": STRING              // идентификатор команды, опционально
}
```

Ответ:

```
{
  "status": "ok",
  "statusCode": 200,
  "data": {
  "history": [
  {
    "type": STRING,           // тип сообщения, см. таблицу ниже
    "body": STRING,           // тело сообщения
    "stime": TIME_STAMP,      // серверное время сообщения
    "sid": SID,               // серверный идентификатор сообщения
    "talker": JID,            // идентификатор собеседника
    "deviceType": STRING      // имя устройства, с которого пришло сообщение
```

```
  },
  {…}
  ]
}
```

Если в запросе указано поле `countOnly:1`, то ответ будет иметь следующий формат:

```
{
  "status": "ok",
  "statusCode": 200,
  "data": {
  "history": [
  {
    "talker": JID,           // идентификатор собеседника
    "count": INT,            // число сообщений
    "lastMsg": {             // последнее сообщение, аналогично предыдущему
    примеру
    …
    }
  },
  {…}
  ]
}
```

В зависимости от типа в описание события может включаться добавочное поле:

Тип сообщения	**Описание**	**Добавочное поле**	**Значение**
"message"	Входящее сообщение	"direction"	"to"
"message"	Исходящее сообщение	"direction"	"from"
"muc-message"	Сообщение из группового чата		
"sms"	СМС-сообщение		
"deleted "	Сообщение было удалено		
"deleted-range"	Несколько сообщений было удалено		
"edited"	Сообщение было изменено		
"edited"	Измененное сообщение было прочитано	"listened"	1
"joined"	Контакт из телефонной книги присоединился к системе		

5. Работа со списком пользователей

5.1. Получение списка пользователей

Запрос:
/roster/getList

Тело запроса:
{}

Ответ:
```
{
  "status": "ok",
  "statusCode": 200,
  "data": [
  {
    "jid": STRING,                // идентификатор пользователя
    "flags": CONTACT_FLAGS,       // набор флагов пользователя
    "delivered": SID,             // номер последнего доставленного события
    "markRead": SID,              // номер последнего прочитанного события
    "remoteMarkRead": SID,        // то же, но для того пользователя
    "remoteMarkReadTime": TIME_STAMP,
    "logoutTime": TIME_STAMP,     // время последнего выхода
    "firstName": STRING,          // имя пользователя
    "lastName": STRING,           // фамилия пользователя
    "group": STRING,              // группа в списке, если установлена
    "status": STATUS,             // статус
    "textStatus": STRING,         // текстовый статус
    "platform": PLATFORM_TYPE     // тип устройства
  },
  {…}
  ],
}
```

5.2. Изменение информации о пользователе в списке

Для любого контакта в списке доступно хранение трех собственных атрибутов: имя и фамилия, как их видит клиент, а не сам пользователь, а также набор флагов для данного контакта.

Запрос:
/roster/edit

Тело запроса:
{

```
  "jid": STRING,              // идентификатор пользователя
  "firstName": STRING,        // имя пользователя, опционально
  "lastName": STRING,         // фамилия пользователя, опционально
  "group": STRING,            // группа в списке, опционально
  "flags": CONTACT_FLAGS      // набор флагов контакта
}
```

Ответ:

```
{
  "status": "ok",
  "statusCode": 200,
  "data": {
    "jid": STRING,            // идентификатор пользователя
    "firstName": STRING,      // имя пользователя
    "lastName": STRING,       // фамилия пользователя
    "flags": CONTACT_FLAGS    // набор флагов контакта
  }
}
```

После успешного изменения информации все остальные устройства этого пользователя получают следующее уведомление:

```
{
  "type":"306",               // тип события – изменение контакта
  "from": JID,                // ваш собственный идентификатор
  "jid": JID,                 // идентификатор контакта в списке
  "updated": {                // список произведённых изменений
    "firstName": STRING,      // имя контакта
    "lastName": STRING,       // фамилия контакта
    "group": STRING,          // группа в списке
    "flags": CONTACT_FLAGS    // набор флагов контакта
  },
  "ts": TIME_STAMP,           // серверное время события
}
```

5.3. Удаление пользователя из списка контактов

После удаления пользователя из списка контактов клиент теряет возможность посылать ему сообщения, а также отслеживать изменение статуса.

Запрос:

```
/roster/remove
```

Тело запроса:

```
{
  "jid": STRING               // идентификатор пользователя
}
```

Ответ:
```
{
  "status": "ok",
  "statusCode": 200
}
```

После успешного удаления на все устройства пользователя, а также удалённому пользователю направляется следующее уведомление:
```
{
  "type":"304",              // тип события – удаление пользователя из списка
  "jid": JID,                // идентификатор собеседника
  "ts": TIME_STAMP           // серверное время события
}
```

5.4. Направление запроса о дружбе

Чтобы добавить произвольный контакт в свой список, клиент должен отослать запрос на дружбу. Впоследствии он может быть подтверждён или отклонён.

Данное действие регулируется настройкой `enableFriends` в файле `/var/www/ommg3_nodejs/config.json` на фронтенд-сервере. Если эта настройка установлена в `false`, то подтверждения запроса на дружбу не требуется, а контакт автоматически добавляется в список контактов. Если же настройка выставлена в `true`, то требуется подтвердить или отклонить запрос.

Запрос:
```
/roster/invite
```

Тело запроса:
```
{
  "jid": STRING,             // идентификатор пользователя
  "firstName": STRING,       // имя пользователя
  "lastName": STRING,        // фамилия пользователя, опционально
  "team": STRING             // команда, в которую пригласить, опционально
}
```

Ответ:
```
{
  "status": "ok",
  "statusCode": 200
}
```

После успешной обработки запроса сервер передаёт указанному пользователю уведомление о том, что ему прислан запрос на дружбу:
```
{
```

```
  "type":"301",              // тип уведомления – запрос на дружбу
  "ts": TIME_STAMP,          // серверное время события
  "user":{                   // контакт, приславший запрос на дружбу
    "firstName": STRING,     // имя контакта
    "lastName": STRING,      // фамилия контакта
    "jid": JID,              // идентификатор контакта
    "avatar": STRING         // хеш аватара
  }
}
```

Отправитель запроса получает точно такое же сообщение, но в нем в полях `firstName` и `lastName` указываются имя и фамилия того, кому прислан запрос на дружбу.

5.5. Принятие запроса о дружбе

Клиент получил уведомление о запросе на дружбу, показал его пользователю и получил от него подтверждение того, что он хочет принять его в список контактов.

Запрос:
```
/roster/inviteAffirmative
```

Тело запроса:
```
{
  "jid": STRING,             // идентификатор пользователя
  "team": STRING             // команда, в которую пригласить, опционально
}
```

Ответ:
```
{
  "status": "ok",
  "statusCode": 200
}
```

После этого обе стороны получают уведомление о добавлении в список контактов:
```
{
  "type":"302",              // тип уведомления – добавление в ростерs
  "ts": TIME_STAMP,          // серверное время события
  "user": {                  // описание нового контакта в ростере
    "jid": JID,              // идентификатор контакта
    "flags": CONTACT_FLAGS,  // набор флагов контакта
    "delivered": SID,        // последнее доставленное на сервер сообщение
    "markRead": SID,         // последнее прочитанное мной сообщение
    "remoteMarkRead": SID,   // последнее прочитанное контактом сообщение
    "remoteMarkReadTime": TIME_STAMP, // время, когда контакт прочитал мое
сообщение
    "firstName": STRING,           // имя контакта
```

```
    "lastName": STRING,      // фамилия контакта
    "logoutTime": TIME_STAMP,// время последнего выхода контакта в оффлайн
    "avatar": STRING,        // хеш аватара контакта
    "status": STATUS,        // статус контакта
    "textStatus": STRING,    // строковый статус
    "platform": PLATFORM_TYPE// код платформы активной сессии
  }
}
```

5.6. Отклонение запроса о дружбе

Если пользователь отвергает запрос о дружбе, клиент обязан уведомить об этом сервер.

Запрос:
```
/roster/inviteNegative
```

Тело запроса:
```
{
  "jid": STRING,             // идентификатор пользователя
  "team": STRING             // команда, в которую пригласить, опционально
}
```

Ответ:
```
{
  "status": "ok",
  "statusCode": 200
}
```

После успешной обработки запроса обе стороны получают следующее уведомление:
```
{
  "type":"305",              // тип события – отклонённый запрос о дружбе
  "with": JID,               // идентификатор контакта
  "ts": TIME_STAMP           // серверное время события
}
```

5.7. Получение списка запросов на дружбу

Запрос:
```
/roster/requests
```

Тело запроса:
```
{
}
```

Ответ:
```
{
```

```
  "status": "ok",
  "statusCode": 200,
  "data": {
    [
      "jid": JID,            // идентификатор пользователя
      "firstName": STRING,   // имя пользователя
      "lastName": STRING,    // фамилия пользователя
      "avatar": STRING,      // хеш аватара контакта
      "type": STRING,        // тип запроса на дружбу, см. ниже
      "created": TIME_STAMP  // время создания запроса
    ]
    …
    []
  }
}
```

Код запроса на дружбу	Описание
1 - TYPE_IN	Пользователь хочет подружиться со мной
2 - TYPE_OUT	Я послал пользователю запрос на дружбу
3 - TYPE_OUT_DECLINED	Пользователь отклонил запрос на дружбу
4 - TYPE_IN_IGNORED	Пользователь добавил мою заявку в игнор
5 - TYPE_IN_POSTPONED	Отложенный входящий запрос на дружбу
6 - TYPE_OUT_POSTPONED	Отложенный исходящий запрос на дружбу
7 - TYPE_IN_DECLINED	Я отклонил запрос на дружбу

6. Списки видимости

Списки видимости применяются для того, чтобы изменять свою видимость для других контактов из своего списка контактов. Возможны два варианта:

- включение контакта в список "всегда видимых", тогда этот контакт будет видеть ваш статус даже в состоянии INVISIBLE;
- включение контакта в список "никогда не видимых", тогда этот контакт будет видеть вас в состоянии OFFLINE, даже если вы сейчас находитесь в сети.

6.1. Получение списков видимости

Запрос:
```
/privacy/getList
```

Тело запроса:
```
{
}
```

Ответ:
```
{
  "status": "ok",
  "statusCode": 200,
  "data": {
    [
      {
        "jid": JID,          // идентификатор пользователя
        "privacy": "visible",// контакт "всегда видимый"
        "firstName": STRING, // имя пользователя
        "lastName": STRING,  // фамилия пользователя
        "avatar": STRING,    // хеш аватара контакта
      },
      {
        "jid": JID,          // идентификатор пользователя
        "privacy": "ignore"  // контакт "никогда не видимый"
        "firstName": STRING, // имя пользователя
        "lastName": STRING,  // фамилия пользователя
        "avatar": STRING,    // хеш аватара контакта
      }
    ]
  }
}
```

6.2. Изменение списков видимости

Запрос:
```
/privacy/manage
```

Тело запроса:
```
{
  "jid": JID,                // идентификатор пользователя
  "privacy": STRING          // строка, см. примечание ниже
}
```

Строковый параметр "privacy" может приобретать три значения:

- `"visible"` для всегда видимых
- `"ignore"` для всегда не видимых
- `"none"` для исключения контакта из списков видимости.

Ответ:
```
{
  "status": "ok",
  "statusCode": 200,
  "data": {}
}
```

В ответ на успешно изменённый список видимости сервер рассылает всем устройствам следующее уведомление:

```
{
  "type": "104",
  "ts": TIME_STAMP,          // время совершения операции
  "user": {
    "jid": JID,              // идентификатор пользователя
    "firstName": STRING,     // имя пользователя
    "lastName": STRING,      // фамилия пользователя
    "avatar": STRING,        // хеш аватара контакта
    "privacy": STRING,       // изначальное значение параметра privacy
    "status": STATUS,        // статус
    "textStatus": STRING,          // текстовый статус
    "platform": PLATFORM_TYPE// тип устройства
    "logoutTime": TIME_STAMP,// время последнего выхода
  }
}
```

7. Работа с личной карточкой

7.1. Получение личной карточки пользователя

Запрос:
```
/vcard/get
```

Тело запроса:
```
{
  "jid": JID                  // идентификатор пользователя
}
```

Ответ:
```
{
  "status": "ok",
  "statusCode": 200,
  "data": {
    "jid": STRING,            // идентификатор пользователя
    "vcard": {
      структура данных в формате JSON, определяется клиентом
    }
  }
}
```

7.2. Запись собственной личной карточки

Запрос:
```
/vcard/set
```

Тело запроса:
```
{
  "vcard": STRING       // структура данных в формате JSON, определяется клиентом
}
```

Ответ:
```
{
  "status": "ok",
  "statusCode": 200
}
```

8. Групповые чаты

Групповой чат – это объект, разделяющий общую историю между своим списком участников.

Для реализации группового чата создаётся фейковый пользователь с идентификатором следующего вида: {GUID}@conference.yourdomain.com . По наличию слова @conference можно судить о том, что контакт является групповым чатом. Соответственно, история группового чата является историей этого фейкового пользователя.

Список участников группового чата задаётся при создании и впоследствии может быть изменён. Каждому участнику группового чата ставится в соответствие так называемая роль, определяющая его возможности в этом чате. Ролей предусмотрено пять:

- `owner` (владелец) – пользователь, создавший групповой чат, имеет права делать любую операцию. Впоследствии может передать роль owner другому пользователю;
- `admin` (администратор) – может делать любую операцию с чатом, кроме назначения владельца;
- `moderator` (модератор) – может менять информацию о чате и писать в модерируемые групповые чаты;
- `participant` (участник) – может писать в модерируемый групповой чат;
- `visitor` (визитёр) – может только читать информацию в групповом чате.

Для каждого чата предусмотрено несколько свойств, которые определяют режим использования чата:

- модерируемый: изменение свойств чата и добавление новых сообщений в чат доступно не всем участникам, а только пользователям с ролью не ниже модератора. По умолчанию отключено. В немодерируемом чате все участники являются владельцами, т.е. могут совершать любые операции.
- приватный: чат доступен только по приглашению в него другим пользователем или добавлением пользователем с правами администратора или владельца. По умолчанию все чаты создаются публичными. Если чат является модерируемым, то зашедший в публичный модерируемый чат новый пользователь по умолчанию получает права визитёра, т.е. может лишь просматривать сообщения.
- отдельная история: история чата не копируется в истории его участников (по умолчанию выключена). Если для небольших чатов копирование истории участникам помогает значительно оптимизировать работу с историей, гарантируя полное её получение одним-единственным запросом, то для каналов или чатов с большим количеством участников (более 500) рекомендуется ведение истории только для фейкового контакта чата.

8.1. Создание группового чата

При создании группового чата сервер создаёт новый GUID и нового фейкового пользователя со случайным идентификатором вида {GUID}@conference.yourdomain.com. Владельцем этого группового чата становится создавший его пользователь. При необходимости создания приватного или модерируемого группового чата необходимо указать поле `flags`.

При создании немодерируемого чата все участники, указанные в параметре `userJids`, также становятся владельцами этого чата, в противном случае они получают статус участников (`participant`).

Аналогично созданию сообщения (`/message/send`), при создании группового чата может быть указан локальный идентификатор события, чтобы остальные устройства пользователя могли отличать групповые чаты, созданные данным устройством. Если клиент рассчитан на работу с одним-единственным устройством, данный параметр может быть пропущен.

Запрос:
```
/chat/create
```

Тело запроса:
```
{
  "topic": STRING,                  // тема чата
  "userJids": [JID1, JID2, …],      // идентификаторы участников чата, опционально
  "flags": GROUPCHAT_FLAGS,         // набор свойств чата, опционально
  "localId": GUID,                  // локальный идентификатор события, опционально
  "team": STRING                    // идентификатор команды, опционально
}
```

Ответ:
```
{
  "status": "ok",
  "statusCode": 200,
  "data": {
    "chatJid": JID                  // идентификатор созданного чата
  }
}
```

После создания чата всем участникам рассылается уведомление следующего формата:
```
{
  "type": "201",                    // тип события – создание чата
  "from": JID,                      // идентификатор пользователя-создателя чата
  "to": JID,                        // идентификатор участника чата
  "ts": TIME_STAMP,                 // серверное время события
  "chat": {
    "jid": JID,                     // идентификатор чата
    "topic": STRING,                // тема чата
```

```
    "flags": GROUPCHAT_FLAGS,      // набор флагов группового чата
    "chatStatus": CHAT_STATUS,     // статус участника в чате
    "markRead": SID,               // последний прочитанный ID сообщения
    "created": TIME_STAMP,         // дата создания чата
    "members": [JID, JID, JID],    // список участников
    "localId": GUID                // опционально, если был указан в запросе на
создание
  }
}
```

8.2. Уничтожение группового чата

Удаление модерируемого группового чата может производиться только владельцем, в то время как немодерируемого – любым участником.

Запрос:
```
/chat/destroy
```

Тело запроса:
```
{
  "chatJid": JID             // идентификатор чата
}
```

Ответ:
```
{
  "status": "ok",
  "statusCode": 200
}
```

После успешного уничтожения группового чата всем участникам рассылается сообщение следующего формата:
```
{
  "type": "207",             // тип события – уничтожение чата
  "from": SYSTEM_JID,
  "chatJid": JID,            // идентификатор уничтоженного чата
  "to": JID,                 // идентификатор участника чата
  "ts": TIME_STAMP           // время события
}
```

8.3. Редактирование группового чата

Редактирование темы модерируемого группового чата может осуществляться только участником с правами не ниже модератора. Изменение флага pinned и других произвольно назначаемых клиентом флагов может производиться любым участником группового чата.

Запрос:

```
/chat/edit
```

Тело запроса:

```
{
  "chatJid": JID,             // идентификатор чата
  "topic": STRING,            // новая тема чата, опционально
  "flags": CONTACT_FLAGS      // набор опций участника чата, опционально
}
```

Ответ:

```
{
  "status": "ok",
  "statusCode": 200
}
```

Если одно или оба поля были изменены тема чата, то всем участникам чата направляется уведомление следующего формата:

```
{
  "type": "206",
  "from": JID,                // идентификатор пользователя, произвёдшего изменение
  "chatJid": JID,             // идентификатор чата
  "updated": {
    "topic": STRING,          // новая тема чата
    "flags": GROUPCHAT_FLAGS  // новый набор флагов чата
  },
  "ts": TIME_STAMP            // серверное время события
}
```

8.4. Получение списка групповых чатов для пользователя

Запрос:

```
/chat/getList
```

Тело запроса:

```
{}
```

Ответ:

```
{
  "status": "ok",
  "statusCode": 200,
  "data": { [
  {
    "jid": JID,                    // идентификатор чата
    "topic": STRING,               // тема чата
    "flags": GROUPCHAT_FLAGS,      // набор флагов для чата
    "chatStatus": CHAT_STATUS,     // статус пользователя в чате
```

```
    "created": TIME_STAMP,         // время создания чата
    "markRead": SID,               // последнее прочтённое сообщение из чата
    "members": [JID, JID, JID]     // список участников
    "team": STRING,                // идентификатор команды, опционально
    "type": INT                    // тип командного чата, опционально
  },
  {…}
  ]
}
```

8.5. Получение списка участников группового чата

Запрос:
```
/chat/members
```

Тело запроса:
```
{
  "chatJid": JID             // идентификатор чата
}
```

Ответ:
```
{
  "status": "ok",
  "statusCode": 200,
  "data": [
  {
    "jid": JID,              // идентификатор участника чата
    "avatar": STRING,        // хеш аватара
    "logoutTime": TIME_STAMP,// время последнего нахождения в сети
    "firstName": STRING,     // имя
    "lastName": STRING,      // фамилия
    "status": TYPE_STATUS,   // статус пользователя, если он в сети
    "textStatus": STRING,    // текстовый статус
    "platform": PLATFORM_TYPE// текущая платформа
  },
  {…}
  ]
}
```

8.6. Изменение списка участников группового чата

Этот вызов требует от участника статуса администратора или создателя чата для модерируемых групповых чатов.

Если необходимо изменить роль уже существующих участников, то их надо перечислить в списке `addJids`, система автоматически поймёт, что добавлять их заново не нужно.

Запрос:
/chat/modify

Тело запроса:
```
{
   "chatJid": JID,                   // идентификатор чата
   "addJids": [JID, JID, JID],       // добавляемые участники чата, опционально
   "role": STRING,                   // роль добавляемых участников, опционально
   "removeJids": [JID, JID, JID]     // удаляемые участники чата, опционально
}
```

Ответ:
```
{
   "status": "ok",
   "statusCode": 200
}
```

Если в запросе был указан параметр `addJids`, и это привело к добавлению нового пользователя, то всем пользователям групчата рассылается уведомление о присоединении к групчату:
```
{
  “type": "202",
  "from": JID,                // пользователь, произведший изменение
  "chatJid": JID,       // идентификатор чата
  "role": STRING,       // роль пользователя
  "flags": CONTACT_FLAGS      // набор опций участника чата, опционально
  "team_id": STRING,          // идентификатор команды, опционально
  "members": [
    {
      "jid": JID,                   // идентификатор нового пользователя
      "avatar": STRING,             // хеш аватара
      "logoutTime": 0,              // время последнего захода
      "firstName": STRING,          // имя
      "lastName": STRING,           // фамилия
      "status": CHAT_STATUS,        // статус пользователя в чате
      "textStatus": STRING,         // строковый статус
      "platform": PLATFORM_TYPE     // текущая платформа
    }
  ],
  "ts": 1624524204415
}
```

Если же был указан параметр `removeJids`, и в результате были удалены один или несколько участников, то им и оставшимся участникам групчата рассылается уведомление об исключении пользователя из групчата:

```
{
    "type": "203",
    "from": JID,              // пользователь, произведший изменение
    "chatJid": JID,           // идентификатор чата
    "userJids": [
       JID, JID, JID          // список идентификаторов удаленных пользователей
    ],
    "ts": 1624524204431
}
```

8.7. Изменение статуса в чате

Запрос:

```
/chat/online – для подключения к чату
/chat/offline – для отключения от чата
```

Тело запроса:

```
{
  "chatJid": JID              // идентификатор чата
}
```

Ответ:

```
{
  "status": "ok",
  "statusCode": 200
}
```

После смены статуса в чате на остальные устройства этого же пользователя рассылается следующее уведомление:

```
{
  "type": "204",              // для оффлайн или 205 для онлайн
  "from": JID,                // собственный идентификатор пользователя
  "chatJid": JID,             // идентификатор чата
  "status": CHAT_STATUS,      // статус пользователя в чате
  "textStatus": STRING,       // текстовый статус пользователя
  "platform": PLATFORM_TYPE,  // текущая платформа пользователя
  "ts": TIME_STAMP            // серверное время события
}
```

Остальные участники чата получают другое уведомление;

```
{
  "type": "204",              // для оффлайн или 205 для онлайн
  "from": JID,                // собственный идентификатор пользователя
  "to": JID,                  // собственный идентификатор пользователя
  "chatJid": JID,             // идентификатор чата
  "ts": TIME_STAMP,           // серверное время события
```

```
  "chat": {
    "topic": STRING,          // тема чата
    "members": [JID, …]       // список участников чата в онлайне
  }
}
```

9. Работа с командами.

9.1. Создание команды.

Запрос:
```
/team/create
```

Тело запроса:
```
{
  "name": STRING,             // имя команды
  "id": STRING,               // идентификатор команды, только английские буквы и _
  "date": SID,                // время создания команды в формате SID
  "description": STRING,      // описание команды
  "photo": URL_STRING,        // URL закачанной картинки, опционально
  "members": [JID, …]         // идентификаторы участников группы
}
```

Ответ:
```
{
  "status": "ok",
  "statusCode": 200,
  "data": {
    "id": STRING,
    "name": STRING,
    "date": SID,
    "photo": URL_STRING,
    "members": [JID, …]
  }
}
```

9.2. Уничтожение команды

Запрос:
```
/team/destroy
```

Тело запроса:
```
{
  "id": STRING                // идентификатор команды
}
```

Ответ:
```
{
  "status": "ok",
  "statusCode": 200
```

```
}
```

9.3. Получение информации о команде

Запрос:
```
/team/info
```

Тело запроса:
```
{
   "id": STRING                // идентификатор команды
}
```

Ответ:
```
{
   "status": "ok",
   "statusCode": 200,
   "data": {
      "id": STRING,
      "name": STRING,
      "description": STRING,
      "photo": URL_STRING,
      "members": [JID, …]
   }
}
```

9.4. Присоединение себя к команде

Запрос:
```
/team/join
```

Тело запроса:
```
{
   "id": STRING                // идентификатор команды
}
```

Ответ:
```
{
   "status": "ok",
   "statusCode": 200
}
```

9.5. Получение списка своих команд

Запрос:
```
/team/list
```

Тело запроса:
Игнорируется

Ответ:
```
{
   "status": "ok",
   "statusCode": 200,
   "list": [idstr: STRING, …]
}
```

9.6. Получение списка участников команды

Запрос:
/team/members

Тело запроса:
```
{
   "id": STRING                // идентификатор команды
}
```

Ответ:
```
{
   "status": "ok",
   "statusCode": 200,
   "members": [JID, …]
}
```

9.7. Изменение команды

Запрос:
/team/modify

Тело запроса:
```
{
   "id": STRING,               // идентификатор команды
   "name": STRING,
   "description": STRING,
   "remove": [JID, …],         // список для удаления из команды
   "admin": [JID, …],          // список администраторов команды
   "members": [JID, …]         // список простых участников команды
}
```

Все параметры запроса, кроме id, являются необязательными.

Ответ:
```
{
```

```
  "status": "ok",
  "statusCode": 200
}
```

Приложение 1. Коды ошибок, используемые в API

Код ошибки	Значение
1001	Модуль не найден (невозможно загрузить код на сервере)
1002	Пользователь не найден
1003	Страница не найдена
1004	Данные о пользователе не найдены
1005	Указанный групповой чат не существует
1006	Указанное устройство не найдено
1007	Сообщение не найдено
1008	Стикер не найден
2001	Неверный тип данных (для списков приватности)
2002	Неверный тип акции (для списков приватности)
2003	Неверный статус пользователя
2004	Требуемый параметр отсутствует или пропущен
2005	Неверный тип платформы устройства
2006	Неверная маска флагов
2007	Неверный тип параметра
3001	Неверный логин или пароль
4001	Попытка добавить в друзья самого себя
4002	Попытка добавить в друзья несуществующий контакт
4003	Неверный или несуществующий пользователь группового чата
4004	Ошибка приглашения в групповой чат
4005	Указанный пользователь отсутствует в списке контактов
4006	Запрошенное действие запрещено
4007	Платформа не поддерживает рассылку пуш-нотификаций
4008	Пользователь не авторизован

Приложение 2. Расширенный формат записи событий в истории.

Простые текстовые сообщения записываются в историю как обычный текст. Для того, чтобы вставить в текст сообщения файл или другой объект, используется расширенный формат записи. Расширенный формат состоит из открывающего тега `[extra ver="1" type="xxxx" len="NNN"]`, где len – длина последующего блока данных в байтах, собственно блока данных в формате JSON и закрывающего тега [/extra]. Содержимое блока данных варьируется в зависимости от поля type. Ниже перечислены все возможные варианты расширенного формата:

- стикер (`gallery_id` - номер коллекции стикеров, `sticker_id` - номер стикера в коллекции):
  ```
  [extra ver="1" type="file" len="124"]
  {"data":[{"name":"sticker.png", "filesize":49430,
  "link":"stickers\/s\/1\/6\/256", "gallery_id":1, "sticker_id":6}]}
  [/extra]
  ```

- ссылка на файл с картинкой:
  ```
  [extra ver="1" type="file" len="217"] {"data":[{"name":"i1.jpg",
  "filesize":37111,
  "domain":"files.ommg.org","link":"00000000000000001000010d40016001637303030313
  2343831363140666c6f6469756d2e72753739313635353138343838406 6c6f6469756d2e7275
  \/i1.jpg"}]}
  [/extra]
  ```

- ссылка на аудио сообщение:
  ```
  [extra ver="1" type="audio" len="408"]
  {"data":[{"name":"filename.mp3", "filesize":1695956, "duration":1220,
  "listener":[JID, …], "domain":"files.ommg.org",
  "link":"00005af3107b23c700000d640018003e373932303833343232363140737065616b636
  861742e696f7b61363264663635632d396561302d343237652d393831662d3837653363313166
  306534667d40636f6e666572656e63652e737065616b636861742e696f\/filename.mp3"}]}
  [/extra]
  ```

- ссылка на видео сообщение:
  ```
  [extra ver="1" type="video" len="270"]
  {"data":[{"filesize":1754040, "name":"filename.mp4",
  "domain":"files.ommg.org",
  "link":"00044364c5bc17df0000056a00180018373030303030303033303640737065616b636
  861742e696f373936303535363133336334073706561 6b636861742e696f/filename.mp4"}]}
  ```

```
[/extra]
```

- пересылка контакта
  ```
  [extra ver="1" type="contact" len="206"]
  {"firstName":"xxxx","lastName":"xxxxxx","avatar":"https:\/\/files.ommg.org\/
  avatars\/42845af4e84e662e04adee51dc8973cc756dae28a182a4d3","mobile":
  ["xxxxxxxxxx"],"home":[],"work":[],"other":[],"email":[]}
  [/extra]
  ```

- географическое местоположение
  ```
  [extra ver="1" type="location" len="75"]
  {"lat":"54.639845159887","long":"36.13243536095706","name":"","address":""}
  [/extra]
  ```

- для подсветки пользователя в чате используется следующий формат:
  ```
  [user="7xxxxxxxxxx@ommg.org"]Xxxxxx Xxxxx[/user]
  ```

- событие "Пользователь только что зарегистрировался в системе", которое начинает историю любого зарегистрировавшегося контакта из вашей телефонной книги, записывается в историю самим сервером в формате
  ```
  [just-registered]
  ```

Приложение 3. Файловые операции

В состав платформы OMMG 3.0 входит специальный файловый сервер, который позволяет хранить файлы, ссылаться на них, асинхронно конвертировать аудио- и видео- сообщения в стандартный формат, а также вести права на просмотр любого файла.

Файловая подсистема реализована на языке php и не является непременной частью платформы OMMG, т.к. расширенный формат сообщений требует только включение ссылок на файлы, которые могут быть физически размещены в любом другом месте.

1. Загрузка файла одним запросом

Для загрузки файла в один приём, без возможности отображения прогресса закачки, применяется следующий POST запрос на `https://files.ommg.org/upload.php` со следующими параметрами:

- `userfile` – строковое имя файла, например, sample1.txt;
- `local_id` – строка, локальный идентификатор файла, впоследствии используемый файловой системой для доставки оповещения об окончании конвертации файла, буде такая потребуется;
- `to_jid` – получатель файла, контакт или групповой чат, для которого устанавливаются права видимости (всем остальным этот файл будет недоступен);
- `device_type` – строковое имя текущего устройства;
- `pic=1` – опционально, используется для загрузки картинок и генерации preview;
- `audio=1` – опционально, указывает на то, что файл является аудио-сообщением
- `video=1` – опционально, указывает на то, что файл является видео-сообщением

Результатом выполнения такого запроса является json вида
`{"status":"ok","domain":"files.ommg.org","public_link":Url},`
где `public_link` – путь на загруженный файл относительно домена, указанного в `domain`.

2. Загрузка больших файлов по частям

Для того, чтобы загружать большие файлы по частям, используется POST запрос на `https://files.ommg.org/upload_chunk.php`. Формат и параметры этого запроса полностью идентичны предыдущему, однако он позволяет собирать файл по кусочкам, используя поле заголовка X-Content-Range. Первым запросом передаётся начало файла и идентификатор сессии закачки:

```
POST /upload_chunk.php HTTP/1.1
Host: ommg.org
Content-Length: 50000
Content-Type: application/octet-stream
```

```
Content-Disposition: attachment; filename="myvideo.mp4"
X-Content-Range: bytes 0-50000/8723987623
Session-ID: 78379476837465
```

Вторым и последующими запросами в уже созданную сессию дописываются другие куски файла:

```
POST /upload_chunk.php HTTP/1.1
Host: ommg.org
Content-Length: 50000
Content-Type: application/octet-stream
Content-Disposition: attachment; filename="myvideo.mp4"
X-Content-Range: bytes 50001-100000/8723987623
Session-ID: 78379476837465
```

В тот момент, когда файл будет полностью передан, сервер скомпонует итоговый файл, разместит его в хранилище и выдаст ответ, описанный в разделе 1 данного приложения.

3. Работа с аватарами

Аватар – это небольшое изображение пользователя, размером до 320x320. Чтобы загрузить аватар, надо послать POST запрос на `https://files.ommg.org/upload_avatar.php` со следующими параметрами:

- `userfile` – поле HTTP-формы, отвечающее за файл;
- gc – опциональный идентификатор группового чата, если аватар выставляется для группового чата;
- `team` – опциональное имя команды, если загружается локальный аватар в команде.

По завершении операции в теле ответа приходит json следующего содержания:
`{"status": "ok", "url": STRING}`

Чтобы задать в качестве аватара одну из предустановленных картинок, надо выполнить запрос
https://files.ommg.org/set_def_avatar.php?avatar_id=NNN
где NNN – номер предустановленной картинки от 1 до 40.

Для того, чтобы считать аватар другого пользователя, надо запомнить хеш аватара, который присылается в описании ростера или запроса на дружбу, и отправить запрос на
https://files.ommg.org/avatars/<hash>

Для считывания аватаров групчатов зарезервирован специальный URL:
https://files.ommg.org/avachat/{GUID}, где GUID – часть идентификатора групчата до @. Обращаем ваше внимание, что символы { и } в URL должны быть закодированы как %7B и %7D соответственно.

Для локальных аватаров команды предусмотрен другой адрес:
https://files.ommg.org/avateam/<teamid>/<hash>

4. Конвертация файлов.

Иногда для размещения файлов их приходится конвертировать в единый формат, понимаемый всеми клиентами на всех платформах. Так, например, аудио-сообщения переводятся в формат MP3, а видео – MP4 с использованием кодека h264.

Для таких сообщений после загрузки файла не указывается ссылка на результат, а лишь возвращается подтверждение того, что файл принят для обработки. Впоследствии, после завершения обработки файла, файловый сервер передаёт событие завершения через сервер приложений в формате сообщения:

```
{
  "type":"401",              // тип события - сообщение
  "body": STRING,            // сообщение в произвольном формате
  "ts": TIME_STAMP,          // серверное время сообщения
  "sid": SID,                // серверный идентификатор сообщения
  "with": STRING,            // получатель файла
  "localId": GUID,           // локальный идентификатор файла
  "deviceType": STRING,      // имя устройства
  "from_global_id": GUID     // идентификатор устройства, инициировавшего закачку
}
```

5. Предварительный просмотр.

Для работы на мобильных клиентах важное значение имеет загрузка изображений небольшого размера для предварительного просмотра больших изображений и видеороликов. Загрузка таких изображений делается с помощью вспомогательных виртуальных URL:

- обычное изображение для предварительного просмотра имеет наибольший размер 320 точек и загружается через добавочный виртуальный URL `/preview`:
  https://files.ommg.org/linktovideo.mp4/preview
- изображение с уменьшенным максимальным размером (меньше 320 точек) загружается через добавочный виртуальный URL `/preview/NNN`, где NNN – максимальный размер в точках от 1 до 320:
  https://files.ommg.org/linktovideo.mp4/preview/200
- квадратное изображение с размером 320 точек, вырезанное по центру, загружается через добавочный виртуальный URL `/crop`:
  https://files.ommg.org/bigpicture.jpg/crop
- квадратное изображение уменьшенного размера, вырезанное по центру, загружается через добавочный виртуальный URL `/crop/NNN`, где NNN – размер уменьшенного изображения в точках от 1 до 320:
  https://files.ommg.org/bigpicture.jpg/crop/300

6. Получение ссылки на файл.

Иногда, чтобы передать загруженный файл другому пользователю или в другой групповой чат, необходимо получить копию файла с другими правами на чтение. Платформа OMMG позволяет делать ссылки на файл, избегая повторной загрузки файла, с помощью запроса на `https://files.ommg.org/make_link.php` со следующими параметрами:

- `filename` – оригинальная ссылка на файл;
- `to_jid` – идентификатор пользователя или группового чата, которому выдаются права на доступ к файлу.

В результате возвращается json следующего формата:
`{"status":"ok", "public_link": STRING}`

Допускается одновременная обработка нескольких файлов. Для этого их имена передаются не в параметре filename, а в теле POST-запроса, разделенные возвратом каретки (\n). В таком случае в ответе придёт json со списком URL в том же порядке, в каком присланы исходные имена файлов.

7. Удаление файлов.

Если файл больше не нужен клиенту, или удаляется сообщение, содержащее имя файла, его необходимо стереть с сервера. Делается это с помощью POST запроса на
`https://files.ommg.org/remove.php`
в теле которого указывается список файлов (в виде URL на скачивание), разделённых возвратом каретки (\n).