<SUP>

LiveJournal XML-RPC Client/Server Protocol Reference

LiveJournal XML-RPC Specification

SUP Fabrik
Москва

Содержание

Введение

В документе представлено описание функций для работы по протоколу LiveJournal XML-RPC. Документ рассчитан на разработчиков приложений.

1. Протокол XML-RPC

1.1. Определение XML-RPC

XML-RPC (сокращение от английского *Extensible Markup Language Remote Procedure Call*) — протокол вызова удаленных процедур (функций) в сети Интернет с помощью XML сообщений. Со спецификацией к протоколу XML-RPC можно ознакомиться на сайте http://www.xmlrpc.com/spec.

Вызов удаленных процедур – технология, позволяющая вызывать функции или процедуры на удаленном компьютере.

1.2. Принцип работы

Вызов XML-RPC осуществляется между двумя сторонами: *клиентом* и *сервером*. Принцип работы XML-RPC состоит в следующем:

1. *Клиент* кодирует название вызываемой функции и параметры в соответствии с XML-RPC.
2. *Клиент* отправляет *серверу* по протоколу HTTP запрос POST, содержащий XML сообщение.
3. *Сервер* получает запрос и передает данные обработчику XML-RPC.
4. Обработчик XML-RPC *сервера* декодирует полученные данные, получает название функции и параметры.
5. *Сервер* выполняет соответствующую функцию.
6. Результат выполнения функции кодируется в соответствии с XML-RPC.
7. *Сервер* отправляет *клиенту* результат выполнения функции.

1.3. Поддержка XML-RPC

Протокол XML-RPC поддерживается множеством платформ и языков программирования, среди которых:

- Java[1];
- Perl;
- Ruby;
- PHP[2];
- Python;
- JavaScript;
- ASP.

1.4. Типы данных

Протокол XML-RPC поддерживает типы данных, представленные в таблице 1.

Таблица 1

[1] http://helma.at/hannes/xmlrpc

[2] http://phpxmlrpc.sourceforge.net/

Имя типа	Описание	Пример
Integer	Целое число	*<i4>23</i4>* или *<int>43</int>*
double	Число с плавающей точкой	*<double>2.0</double>*
boolean	Логическая величина (0 или 1)	*<boolean>1</boolean>*
string	Строка символов	*<string>Hello, World!</string>*
date/time	Дата и время	*<dateTime.iso8601>19980717T14:08:55</dateTime.iso8601>*
base64	Кодированные данные в формате Base64	*<base64>SGVsbG8sIFdvcmxkIQ==</base64>*
array	Массив	*<array> <data> <value> XML-RPC значение </value> ... <value> XML-RPC значение </value> </data> </array>*
struct	Структура	*<struct> <member> <name>Поле-1</name> <value>Значение-1</value> </member> ... <member> <name>Поле-n</name> <value>Значение-n</value> </member> </struct>*

Все текстовые переменные должны быть представлены в кодировке UTF-8. При этом текст может кодироваться в формате base64 с использованием тэга <base64>.

Для строк, включающих не только символы латиницы, API XML-RPC кодирует возвращаемые строки с помощью тэгов base64 (это является особенностью реализации API LiveJournal в данный момент).

2. LiveJournal XML-RPC

В основе LiveJournal XML-RPC лежит протокол XML-RPC. Принцип работы изображен на рисунке 1. Клиент (пользователь) посылает XML-RPC запрос серверу по адресу **http://www.livejournal.com/interface/xmlrpc**. Запрос содержит сообщение в формате XML. Сервер получает сообщение, обрабатывает его и выполняет требуемую функцию. Результат выполнения функции сервер отправляет клиенту (XML-RPC ответ).

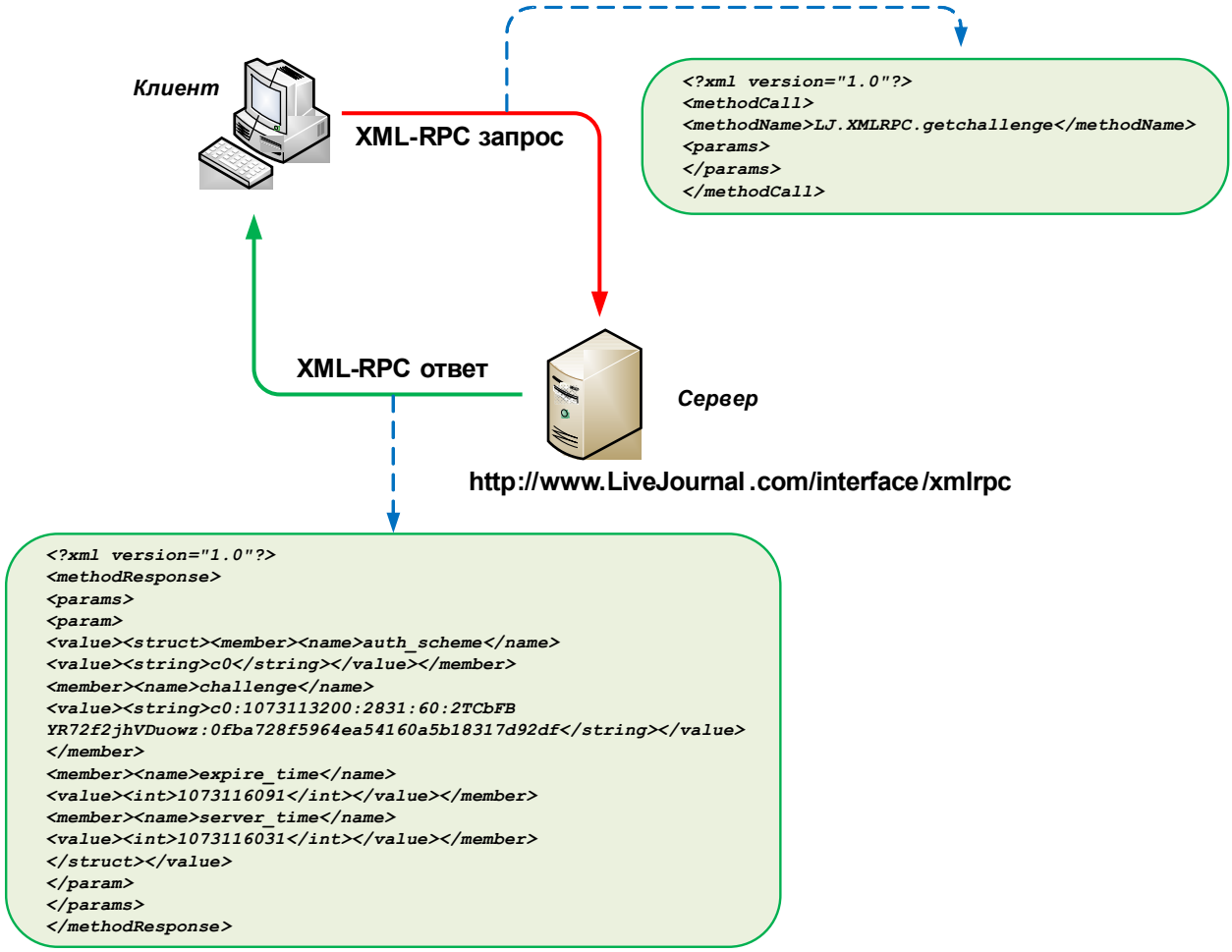

Рис.1 – Принцип работы.

Названия функций LiveJournal XML-RPC имеют следующий формат: ***LJ.XMLRPC.****название_функции* (например LJ.XMLRPC.postevent), где ***LJ.XMLRPC.*** постоянная неизменяемая часть. Изменяемая часть *название_функции* представляет собой одно, либо несколько слов (написанных слитно), кратко описывающих принцип действия функции. Далее при описании функций в тексте документа будет использоваться только изменяемая часть названия (*название_функции).*

2.1. Структура запроса

Запрос состоит из двух частей:

- HTTP заголовков;
- XML сообщения с вызываемой функцией и параметрами.

HTTP заголовки содержат:

- **User-Agent** – название и версия приложения;
- **Host** – доменное имя хоста запрашиваемого ресурса;
- **Content-Type** – формат и способ представления;
- **Content-Length** – размер содержимого XML сообщения в байтах.

XML сообщение заключено в тегах *<methodCall> </methodCall>*. Элемент methodCall содержит название функции и параметры. Название функции расположено в тегах *<methodName>..</methodName>*. Параметры функций LiveJournal XML-RPC заключены в тегах *<params><param><value>..</params></param></value>* и передаются в структуре. Значения параметров соответствуют значению полей структуры.

Пример запроса XML-RPC:

```
POST /interface/xmlrpc HTTP/1.0
User-Agent: XMLRPC Client 1.0
Host: www.livejournal.com
Content-Type: text/xml
Content-Length: 396

<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
<methodName>LJ.XMLRPC.sessiongenerate</methodName>
<params><param>
<value><struct>
<member><name>username</name>
<value><string>test</string></value></member>
<member><name>password</name>
<value><string>test</string></value></member>
<member><name>ver</name>
<value><int>1</int></value></member>
</struct></value>
</param></params>
</methodCall>
```

2.2. Структура ответа

После получения XML-RPC запроса, сервер отправляет ответ клиенту. Ответ содержит:

- HTTP заголовки;
- XML сообщение с результатом выполнения функции.

XML сообщение может принимать одну из двух форм:

- результат выполнения функции;
- сообщение об ошибке.

Результат выполнения функции содержит возвращаемые значения и заключен в тегах *<methodResponse>…</methodResponse>*. Возвращаемые значения расположены в

тегах *<params><param><value>..</params></param></value>* и содержатся в структуре. Названия возвращаемых параметров передаются в полях структуры, а значения параметров передаются в значениях полей структуры.

Пример ответа:

```
HTTP/1.0 200 OK
Date: Fri, 20 Aug 2010 19:03:59 GMT
Server: Apache/2.2.3 (CentOS)
Content-Length: 2221
Content-Type: text/xml
SOAPServer: SOAP::Lite/Perl/0.710.08
Keep-Alive: timeout=30, max=100
Connection: keep-alive

<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
<params><param>
<value><struct>
<member><name>ljsession</name>
<value><string>ws:test:124:zfFG136kSz</string>
</value></member>
</struct></value>
</param></params>
</methodResponse>
```

В случае ошибки сервер возвращает сообщение об ошибке, заключенное в тегах *<methodResponse>…</methodResponse>*. Параметры ошибки передаются в тегах *<fault><value>..</value></fault>* и содержатся в структуре, содержащей следующие поля:

- **faultString** (string)[3] – описание ошибки;
- **faultCode** (integer) – код ошибки.

Список ошибок, возвращаемых при запросе функций LiveJournal XML-RPC, приведен в Приложении А.

Пример отчета об ошибке:

```
HTTP/1.1 200 OK
Connection: close
Content-length: 228
Content-Type: text/xml
Date: Fri, 26 Mar 2004 18:14:17 GMT
Server: Apache/1.3.4 (Unix)
```

[3] Здесь и далее в скобках указан тип данных значения поля.

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
<fault>
<value><struct>
<member><name>faultString</name>
<value><string>Client error: Unknown method</string></value></member>
<member><name>faultCode</name>
<value><int>201</int></value></member>
</struct></value>
</fault>
</methodResponse>
```

2.3. Аутентификация

Каждая функция LiveJournal XML-RPC должна содержать параметры аутентификации (кроме функции ***getchallenge***). Параметры аутентификации передаются в виде полей в структуре вместе с остальными параметрами функции.

Аутентификация – проверка подлинности предъявленного пользователем идентификатора.

2.3.1. Методы аутентификации

Аутентификация пользователя может проходить тремя способами:

- **передачей логина и пароля в явном виде;**
- **методом «вызов-ответ» («challenge-response»);**
- **с помощью «cookie».**

Рассмотрим далее каждый из методов по отдельности.

2.3.2. Передача логина и пароля в явном виде

При данном методе аутентификации имя пользователя (логин) и пароль (или его хэш MD5) передаются на сервер в незашифрованном виде. Этот метод является самым простым и в то же время самым незащищенным. Данный метод аутентификации используется по умолчанию.

Примечание! Использование данного метода не рекомендуется.

2.3.2.1. Параметры

Поля структуры:

- **username**[4] (string) – имя пользователя;
- **auth_method** (string) – метод аутентификации («*clear*»);
- **password** (string) – пароль пользователя. Пароль передается в явном виде. При установке данного параметра поле ***hpassword*** не используется.

[4] Подчеркиванием обозначаются обязательные поля. Если подчеркивание отсутствует – наличие поля при вызове функции необязательно.

- **hpassword** (string) – хэш MD5 от пароля пользователя. При установке данного параметра поле ***password*** не используется.
- **ver** (integer) – версия используемого протокола (всегда «*1*»).

2.3.3. Метод аутентификации «вызов-ответ»

Вызов-ответ («challenge-response») — способ аутентификации, при котором пароль не передается по каналу связи.

Принцип работы метода аутентификации «вызов-ответ»:

- Пользователь, желающий пройти аутентификацию, делает запрос. В ответ на запрос сервер посылает произвольное, но всякий раз разное значение (challenge).
- Пользователь дописывает к полученному значению (challenge) MD5 хэш от пароля и от этой строки вычисляет MD5 хэш и отправляет его серверу.
- Сервер проделывает со значением (challenge) аналогичные действия и сравнивает результат. Если значения хэшей совпадают, то аутентификация считается успешной.

2.3.3.1. Параметры

Поля структуры:

- **username** (string) – имя пользователя;
- **auth_method** (string) – метод аутентификации («*challenge*»);
- **auth_challenge** (string) – значение challenge, полученное от сервера;
- **auth_response** (string) – значение ответа для сервера;
- **ver** (integer) – версия используемого протокола (всегда «*1*»).

Значение поля *auth_response* рассчитывается по формуле: *MD5_hex(challenge* + *MD5_hex(*`password`*)),* где «+» это операция объединения строк, значение *challenge* передается сервером при вызове функции ***getchallenge*** *(*см. *п. **2.3.3.2**)*.

2.3.3.2. Функция getchallenge

Описание

Функция запрашивает у сервера значение для использования в методе аутентификации «вызов-ответ» (challenge-response).

Параметры

Функция не требует параметров.

Пример запроса ***getchallenge***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
<methodName>LJ.XMLRPC.getchallenge</methodName>
</methodCall>
```

Возвращаемые значения

Структура, содержащая следующие поля:

- **auth_scheme** (string) – идентификатор схемы аутентификации;
- **challenge** (string) – значение для хеширования пароля (challenge);
- **expire_time** (integer) – предельное Unix-время функционирования полученного значения *challenge*;
- **server_time** (integer) – Unix-время генерации значения *challenge* на сервере;

Примечание! Максимальное время функционирования полученного значения challenge составляет 60 секунд.

Пример ответа на запрос ***getchallenge***:

```
<?xml version="1.0" encoding="UTF-8" ?>
<methodResponse>
<params><param>
<value><struct>
<member><name>auth_scheme</name>
<value><string>c0</string></value></member>
<member><name>challenge</name>
<value><string>c0:1073113200:2831:60:2TCbFBYR72f2jhVDuowz:0fba728f5964ea541
60a5b18317d92df</string></value></member>
<member><name>expire_time</name>
<value><int>1073116091</int></value></member>
<member><name>server_time</name>
<value><int>1073116031</int></value></member>
</struct></value>
</param></params>
</methodResponse>
```

2.3.4. Метод аутентификации с помощью «cookie»

Cookie – небольшой фрагмент данных, созданный сервером, который клиент каждый раз пересылает серверу в HTTP-заголовке.

Полученные от сервера LiveJournal cookies используются для аутентификации последующих вызовов к XML-RPC.

2.3.4.1. Параметры

Поля структуры:

- **username** (string) – имя пользователя;
- **auth_method** (string) – метод аутентификации («*cookie*»);
- **ver** (integer) – версия используемого протокола (всегда «*1*»).

В каждом HTTP запросе с вызовом функций LiveJournal XML-RPC должна передаваться cookie *ljsession*, равная *ljmastersession*, выставленной сервером LiveJournal (в случае веб-авторизации), либо равная значению поля *ljsession*, полученному с помощью функции ***sessiongenerate*** (см. *п. 2.3.4.2*). Дополнительно необходимо передавать HTTP заголовок *X-LJ-Auth: cookie*.

Пример аутентификации с помощью cookie:

```
POST /interface/xmlrpc HTTP/1.1
TE: deflate,gzip;q=0.3
Connection: TE, close
Host: www.livejournal.com
User-Agent: XML-RPC/0.9
Content-Length: 433
Content-Type: text/xml
Cookie: ljsession=v1:u2:s437:aJHBATCvaz1//GEN;
ljuniq=lmqDEGGWoIAQqBc:1285070754:pgstats0:m0
Cookie2: $Version="1"
X-LJ-Auth: cookie

<?xml version="1.0" encoding="UTF-8" ?>
<methodCall>
<methodName>LJ.XMLRPC.login</methodName>
<params><param>
<value><struct>
<member><name>ver</name>
<value><i4>1</i4></value></member>
<member><name>auth_method</name>
<value><string>cookie</string></value></member>
<member><name>username</name>
<value><string>test</string></value></member>
</struct></value>
</param></params>
</methodCall
```

2.3.4.2. *Функция sessiongenerate*

Описание

Функция строит сессию и возвращает ее идентификатор (cookie).

Параметры

Структура, содержащая следующие поля:

- **параметры аутентификации**[5] (см. *п. 2.3.1*);
- **expiration** (string) – период времени, в течение которого сессия будет функционировать. Значение «*short*» – 24 часа, значение «*long*» – 720 часов (30 дней);

[5] при вызове данной функции используются методы аутентификации «вызов-ответ» или передача логина и пароля в явном виде

- **bindtoip** (string) – при истинном значении поля сервер строит сессию, доступную только для данного IP адреса, с которого происходит вызов функции ***sessiongenerate***. *По умолчанию сессия доступна для любого IP адреса.*

Пример запроса **sessiongenerate**:

```
<?xml version="1.0" encoding="UTF-8" ?>
<methodCall>
<methodName>LJ.XMLRPC.sessiongenerate</methodName>
<params><param>
<value><struct>
<member><name>password</name>
<value><string>test</string></value></member>
<member><name>ver</name>
<value><i4>1</i4></value></member>
<member>
<name>username</name>
<value><string>test</string></value></member>
</struct></value>
</param></params>
</methodCall>
```

Возвращаемые значения

Структура, содержащая следующие поля:

- **ljsession** (string) – идентификатор построенной сессии.

Пример ответа на запрос ***sessiongenerate***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
<params><param>
<value><struct>
<member><name>ljsession</name>
<value><string>v1:u2:s437:aJHBATCvaz1//GEN</string></value></member>
</struct></value>
</param></params>
</methodResponse>
```

Возвращаемые ошибки

308 – аккаунт заблокирован и не может быть использован ("Account is locked and cannot be used").

2.3.4.3. Функция sessionexpire

Описание

Функция сбрасывает все указанные сессии, полученные через ***sessiongenerate*** или через веб-интерфейс LiveJournal.

Параметры

Структура, содержащая следующие поля:

- **параметры аутентификации** (см. *п. 2.3.1*);
- **expireall** (integer) – сброс пользовательских сессий. При значении поля равном «*1*» все пользовательские сессии будут сброшены.
- **expire** (array) – массив идентификаторов сессий для сбрасывания.

Пример запроса ***sessionexpire***:

```
POST /interface/xmlrpc HTTP/1.1
Connection: TE, close
Host: www.livejournal.com
User-Agent: XML-RPC/0.9
Content-Length: 441
Content-Type: text/xml
Cookie: ljmastersession=v1:u2:s406:avkLi0UWgXx//GEN;
langpref=en_LJ/1284558970; BMLschemepref=horizon; ljloggedin=u2:s406;
ljsession=v1:u2:s406:t1284555600:gcdaf4b82c80c3cec8091d17d72f25387cedbbabe/
/GEN;
ljuniq=CbAVRoe5s5R4B7T:1284558970:pgstats0:m0
X-LJ-Auth: cookie

<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
<methodName>LJ.XMLRPC.sessionexpire</methodName>
<params><param>
<value><struct>
<member><name>ver</name>
<value><i4>1</i4></value></member>
<member><name>auth_method</name>
<value><string>cookie</string></value></member>
<member><name>username</name>
<value><string>test</string></value></member>
</struct></value>
</param></params>
```

```
</methodCall>
```

Возвращаемые значения

Пустая структура.

Пример ответа на запрос ***sessionexpire***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
<params><param>
<value><struct /></value>
</param></params>
</methodResponse>
```

Возвращаемые ошибки

502 – база данных временно недоступна ("Database temporarily unavailable").

2.4. Профайл пользователя

Любой зарегистрированный пользователь LiveJournal, имеющий свою учетную запись (далее аккаунт), помимо регистрационных данных, может оставить свою контактную информацию, личные данные, такие как пол и возраст, список своих предпочтений, список учебных заведений, в которых он учился, и т.д. Вся эта информация хранится в **профайле пользователя**.

Для запроса профайла пользователя используется функция ***login*** (см. *п.* ***2.4.1***).

2.4.1. Функция login

Описание

Функция позволяет получить информацию из профиля пользователя: имя пользователя, список групп друзей, и другие данные.

Параметры

Структура, содержащая следующие поля:

- **параметры аутентификации** (см. *п.* ***2.3.1***);
- **clientversion** (string) – строка для идентификации клиента, который запрашивает данные. Это позволяет серверу LiveJournal собирать статистику.
- **getmoods** (integer) – если необходимо получить список всех настроений, следует установить значение поля равное «*0*». При установке другого значения (больше *0*), сервер вернет список настроений, идентификаторы которых больше установленного значения.
- **getmenus** (integer) – при значении поля равном «*1*» возвращается список (дерево) переходов в веб-меню;
- **getpickws** (integer) – если необходимо получить список ключевых слов картинок, следует установить значение поля равное «*1*»;

- **getpickwurls** (integer) – если необходимо получить список ссылок картинок (URL) и ключевые слова картинок следует установить значение поля равное «*1*»;
- **getcaps** (integer) – если необходимо получить уровни пользовательского аккаунта, следует установить значение поля равное «*1*».

Пример запроса ***login***:

```
<?xml version="1.0" encoding="UTF-8" ?>
<methodCall>
<methodName>LJ.XMLRPC.login</methodName>
<params><param>
<value><struct>
<member><name>getpickwurl</name>
<value><i4>1</i4></value></member>
<member><name>getpickws</name>
<value><i4>1</i4></value></member>
<member><name>ver</name>
<value><i4>1</i4></value></member>
<member><name>auth_response</name>
<value><string>bb3971ea01b65a5587b892c67436429f</string></value>
</member>
<member><name>auth_method</name>
<value><string>challenge</string></value></member>
<member><name>username</name>
<value><string>test></string></value></member>
<member><name>auth_challenge</name>
<value><string>c0:1284562800:2882:60:DjTOrFuYOQerHKCbG736:7b0bf9d97cfb3c79d
cd27c92e505257e</string></value></member>
<member><name>getmenus</name>
<value><i4>1</i4></value></member>
<member><name>getcaps</name>
<value><i4>1</i4></value></member>
</struct></value>
</param></params>
</methodCall>
```

Возвращаемые значения

Структура, содержащая следующие поля:

- **fullname** (string) – имя пользователя;
- **message** (string) – сообщение, которое уведомляет пользователя о запрошенных обновлениях в программном обеспечении, проблемах с его аккаунтом, и т.д.

- **friendgroups** (array) – массив, описывающий группу пользователей;
- **usejournals** (array) – массив журналов (сообществ), доступных пользователю для размещения записей;
- **moods** (array) – массив настроений;
- **pickws** (array) – массив, содержащий ключевые слова картинок. Поле возвращается, если при запросе функции значение поля ***getpickws*** было равно «*1*».
- **pickwurls** (array) – массив, содержащий ссылки на пользовательские картинки (URL);
- **defaultpicurl** (string) – ссылка на картинку по умолчанию (URL). Поле возвращается, если при запросе функции значение поля ***getpickwurls*** было равно «*1*».
- **fastserver** (boolean) – при истинном значении этого поля клиент может устанавливать «Cookie: ljfastserver=1» в HTTP заголовке для приоритетной обработки запроса;
- **userid** (integer) – идентификатор пользователя;
- **menus** (array) – массив, содержащий элементы меню в том порядке, который должен быть в соответствующем веб-меню пользовательского приложения LiveJournal. Поле возвращается, если при запросе функции значение поля ***getmenus*** было равно «*1*».
- **caps** (integer) – уровень пользовательского аккаунта, заданный битовыми полями в 2-х байтовом целом числе. Поле возвращается, если при запросе функции значение поля ***getcaps*** было равно «*1*».

Описание группы пользователей. Структура, содержащая следующие поля:

- **public** (boolean) – флаг публичности группы;
- **name** (string) – имя группы друзей;
- **id** (integer) – номер бита для этой группы друзей (от 1 до 30);
- **sortorder** (integer) – число, для упорядочивания групп (от 1 до 255).

Описание настроения. Структура, содержащая следующие поля:

- **parent** (integer) – идентификатор родительского настроения;
- **name** (string) – название настроения;
- **id** (integer) – идентификатор настроения.

Описание элемента меню. Структура, содержащая следующие поля:

- **text** (string) – текст элемента меню, или «-» для разделителя.
- **url** (string) – ссылка на пункт меню (URL). Заданы для всех элементов меню за исключением разделителей и подменю.
- **sub** (array) – массив структур, содержащих элементы подменю. Поле возвращается, если данный элемент меню является подменю. Структура

подменю имеет тот же самый формат, как и структура меню верхнего уровня.

Уровень аккаунта. Комбинация битовых полей:

- **0x01** – новый пользователь (new user);
- **0x02** – обычный пользователь (normal user);
- **0x04** – пользователь, который зарегистрировался ранее сентября 2000 года (early adopter);
- **0x08** – платный пользователь (paid user);
- **0x10** – постоянный аккаунт (permanent account);
- **0x20** – пользователь может добавить в список друзей до 10000 друзей (the "many friends" class);
- **0x40** – аккаунт пользователя заблокирован и ему присвоен статус «только чтение» на время выполнения технических операций (move in progress);
- **0x80** – спонсорские аккаунты (Sponsored accounts);
- **0x100** – пользователь-бетатестер (betatesting);
- **0x200** – дополнительные картинки пользователя (extra userpics);
- **0x400** – кириллические пользователи, получающие сервис от SUP (SUP-flagged users who receive some services from SUP);
- **0x800** – кириллические пользователи, которые отказались от сервисов SUP через службу поддержки пользователей (SUP optout (has asked 6A to opt out from SUP));
- **0x1000** – в некоторых странах пользователь может позвонить по определенному номеру телефона и продиктовать запись (temporary unlimited phone posts);
- **0x2000** – детский аккаунт, если пользователь указал свой возраст меньше 14 лет (underage class);
- **0x4000** – пользователь, которому реклама была включена в журнале, несмотря на то, что он не должен ее видеть (ad supporter (testing));
- **0x8000** – улучшенный аккаунт (Plus).

Пример ответа на запрос ***login***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
<params><param>
<value><struct>
<member><name>defaultpicurl</name>
<value/></member>
<member><name>userid</name>
<value><int>2</int></value></member>
<member><name>caps</name>
```

<value><int>33794</int></value></member>

<member><name>menus</name>

<value><array><data>

<value><struct>

<member><name>text</name>

<value><string>Recent Entries</string></value></member>

<member><name>url</name>

<value><string>http://www.livejournal.com/users/test/</string></value>

</member>

</struct></value>

<value><struct>

<member><name>text</name>

<value><string>Calendar View</string></value></member>

<member><name>url</name>

<value><string>http://www.livejournal.com/users/test/calendar</string></val
ue></member>

</struct></value>

<value><struct>

<member><name>text</name>

<value><string>Friends View</string></value></member>

<member><name>url</name>

<value><string>http://www.livejournal.com/users/test/friends</string></valu
e></member>

</struct></value>

<value><struct>

<member><name>text</name>

<value><string>-</string></value></member>

<member><name>url</name><value /></member>

</struct></value>

<value><struct>

<member><name>text</name>

<value><string>Your Profile</string></value>

</member>

<member><name>url</name>

<value><string>http://www.livejournal.com/userinfo.bml?
user=test</string></value></member>

</struct></value>

<value><struct><member><name>text</name>

<value><string>Your To-Do List</string></value></member>

<member><name>url</name>

<value><string>http://www.livejournal.com/todo/?user=test</string></value>

```
</member>
</struct></value>
<value><struct>
<member><name>text</name>
<value><string>-</string></value></member>
<member><name>url</name><value />
</member>
</struct></value>
<value><struct>
<member><name>sub</name>
<value><array><data>
<value><struct>
<member><name>text</name>
<value><string>Personal Info</string></value></member>
<member><name>url</name>
<value><string>http://www.livejournal.com/manage/profile/</string></value>
</member>
</struct></value>
<value><struct>
<member><name>text</name>
<value><string>Customize Journal</string></value></member>
<member><name>url</name>
<value><string>http://www.livejournal.com/customize/</string></value>
</member>
</struct></value>
</data></array></value></member>
<member><name>text</name>
<value><string>Change Settings</string></value></member>
<member><name>url</name>
<value /></member>
</struct></value>
<value><struct>
<member><name>text</name>
<value><string>-</string></value></member>
<member><name>url</name><value /></member>
</struct></value>
<value><struct>
<member><name>text</name>
<value><string>Support</string></value></member>
<member><name>url</name>
```

<value><string>http://www.livejournal.com/support/</string></value>
</member>
</struct></value>
<value><struct>
<member><name>text</name>
<value><string>Upgrade your account</string></value></member>
<member><name>url</name>
<value><string>http://www.livejournal.com/paidaccounts/</string></value>
</member>
</struct></value>
</data></array></value></member>
<member><name>friendgroups</name>
<value><array><data>
<value><struct>
<member><name>public</name>
<value><int>0</int></value></member>
<member><name>name</name>
<value><string>Family</string></value></member>
<member><name>id</name>
<value><int>1</int></value></member>
<member><name>sortorder</name>
<value><int>5</int></value></member>
</struct></value>
<value><struct>
<member><name>public</name>
<value><int>0</int></value></member>
<member><name>name</name>
<value><string>Local Friends</string></value></member>
<member><name>id</name>
<value><int>2</int></value></member>
<member><name>sortorder</name>
<value><int>10</int></value></member>
</struct></value>
<value><struct>
<member><name>public</name>
<value><int>0</int></value></member>
<member><name>name</name>
<value><string>Online Friends</string></value></member>
<member><name>id</name>
<value><int>3</int></value></member>

<member><name>sortorder</name>
<value><int>15</int></value></member>
</struct></value>
<value><struct>
<member><name>public</name>
<value><int>0</int></value></member>
<member><name>name</name>
<value><string>School</string></value></member>
<member><name>id</name>
<value><int>4</int></value></member>
<member><name>sortorder</name>
<value><int>20</int></value></member>
</struct></value>
<value><struct>
<member><name>public</name>
<value><int>0</int></value></member>
<member><name>name</name>
<value><string>Work</string></value></member>
<member><name>id</name>
<value><int>5</int></value></member>
<member><name>sortorder</name>
<value><int>25</int></value></member>
</struct></value>
<value><struct>
<member><name>public</name>
<value><int>0</int></value></member>
<member><name>name</name>
<value><string>Mobile View</string></value></member>
<member><name>id</name>
<value><int>6</int></value></member>
<member><name>sortorder</name>
<value><int>30</int></value></member>
</struct></value>
<value><struct>
<member><name>public</name>
<value><int>0</int></value></member>
<member><name>name</name>
<value><string>TEST13</string></value></member>
<member><name>id</name>
<value><int>7</int></value></member>

<member><name>sortorder</name>

<value><int>35</int></value></member>

</struct></value>

</data></array></value></member>

<member><name>usejournals</name>

<value><array><data>

<value><string>fakingfashion</string></value>

<value><string>game_comm</string></value>

<value><string>hotelaftershow</string></value>

<value><string>veg_food_porn</string></value>

</data></array></value></member>

<member><name>message</name>

<value><string>Your password must be at least six characters long. Your password is too easy to guess. It's recommended that you change it, otherwise you risk having your journal hijacked. Please see http://www.livejournal.com/support/faqbrowse.bml?faqid=71 for LiveJournal.com's password rules, and visit http://www.livejournal.com/changepassword.bml to change your password.</string></value></member>

<member><name>fullname</name>

<value><base64>VEVTVDog0YLQtdGB0YLQvtCy0YvQuSDQv9C+0LvRjNC30L7QstCw0YLQtdC70Yw=</base64></value></member>

<member><name>pickws</name>

<value><array><data>

<value><string>bull</string></value>

<value><string>championat</string></value>

</data></array></value></member>

<member><name>pickwurls</name>

<value><array>

<data />

</array></value></member>

</struct></value>

</param></params>

</methodResponse>

Возвращаемые ошибки

207 – ошибочная версия протокола ("Protocol version mismatch").

208 – неправильная кодировка текста ("Invalid text encoding").

308 – аккаунт заблокирован и не может быть использован ("Account is locked and cannot be used").

502 – база данных временно недоступна ("Database temporarily unavailable").

2.5. Журнал

Журнал пользователя – это персональный дневник (блог) пользователя.

2.5.1. Тип журнала

Журналы бывают разных типов. **Тип журнала** определяет свойства журнала. Существуют следующие типы журнала:

- Пользовательский журнал (**P**) — журнал отдельного пользователя. Писать в такой журнал может лишь владелец журнала.
- Сообщество (**C**). Журнал сообщества отличается от пользовательского журнала тем, что писать в него могут несколько пользователей-членов сообщества. У журнала-сообщества кроме владельца есть модераторы — особая группа пользователей, следящая за содержимым журнала. Модераторы могут редактировать и удалять сообщения других пользователей, а также изменять права членов сообщества.
- Новостная лента (**N**) — новостной журнал. Существует в единственном экземпляре и служит для рассылки новостей пользователям.
- Многопользовательский журнал (**S**) — первоначальный вариант сообществ. Устаревший вид журнала, на данный момент не используется.
- Транслируемый журнал (RSS-ленты) (**Y**) — журнал для сбора RSS потоков с других сайтов. В таком журнале нельзя комментировать записи.
- Переименованный журнал (**R**) — журнал пользователя, сменившего свой логин. В LiveJournal логин не является уникальным идентификатором аккаунта пользователя, поэтому его можно поменять, сохранив свой аккаунт со всеми его группами и настройками.
- OpenID-журнал (**I**) — пользователи, имеющие учетную запись на некотором другом OpenID сервере и воспользовавшиеся этой учетной записью при входе в систему.

2.5.2. Запись

Любой журнал в LiveJournal представляет собой набор **записей** (также известных под названием **посты**), сделанных владельцем дневника, или другими пользователями.

Функции для работы с записями:

- ***postevent*** (см. *п.* ***2.5.3.1***) – для создания записей;
- ***getevents*** (см. *п.* **2.5.3.2**) – для получения записей;
- ***editevent*** (см. *п.* ***2.5.3.3***) – для редактирования записей;
- ***syncitems*** *(*см. *п.* ***2.5.3.4****)* – для получения списка всех созданных или измененных элементов журнала с указанного времени;
- ***getdaycounts*** (см. *п.* ***2.5.3.5***) – для получения количества записей по дням.

2.5.3. Параметры записи

Запись состоит из следующих параметров:

Дата (*Date*) — дата, под которой запись будет опубликована в журнале.

Тема (*Subject*) — тема записи. В качестве темы записи пользователь может указать произвольную последовательность символов, включая HTML-теги, а также использовать здесь некоторые из LJ-тегов, например, `<lj-user>`. Тема записи может быть пустой.

Текст (*Text*) — текст записи. В тексте записи пользователь может ввести произвольный текст, включая HTML-теги и специальные теги разметки записей в LiveJournal, такие как **<lj-user>**, **<lj-cut>**, **<lj-embed>** и другие.

Теги (*Tags*) — теги записи.

Настроение (*Mood*) — настроение пользователя, которое будет указано в его журнале вместе с записью.

Местоположение (*Location*) — текущее местоположение пользователя в виде произвольной текстовой строки.

Музыка (*Music*) — музыка, которую пользователь слушает в данный момент. Произвольная строка текста.

Для каталогизации записей в журнале пользователя существует два идентификатора, внутренний (***itemid***) и внешний (***ditemid***). Первый из них является порядковым номером записи в журнале. Внутренний идентификатор записи ***itemid*** не используется для доступа к записи извне. Внешний идентификатор записи (***ditemid***) равен:

ditemid = itemid * 256 + anum,

где ***anum*** — случайное целое число в диапазоне от 0 до 255 включительно.

Теги

Теги (или **метки**) — это слова или короткие фразы, предназначенные для систематизации журнала. Теги упрощают поиск по журналу и облегчают навигацию по нему автору и посетителям. Например, пользователь может пометить записи, посвященные одному и тому же событию, тегом с названием этого события.

Для получения списка тегов используется функция ***getusertags*** (см. *п.* ***2.5.3.6***).

2.5.3.1. Функция postevent

Описание

Функция позволяет создать новую запись в журнале.

Параметры

Структура, содержащая следующие поля:

- **параметры аутентификации** (см. *п.* ***2.3.1***);
- **event** (string) – текст записи;
- **lineendings** (string) – символы, используемые для разделения строк в записи («*unix*» - 0x0A (\n), «*pc*» - 0x0D0A (\r\n), «*mac*» - 0x0D (\r));
- **subject** (string) – тема записи;
- **security** (string) – уровень доступа к записи. Значения поля: «*public*» – публичный (*значение по умолчанию*), «*private*» – приватный, «*usemask*» – по группам пользователей. При значении «*usemask*» уровень доступа задается значением поля ***allowmask***.
- **allowmask** (integer) – битовая маска, задающая группы пользователей, которым будет доступна публикуемая запись. Беззнаковое 32-х битное целое, где 0-й бит – все друзья пользователя, биты с 1 по 30 – соответствуют группам пользователей, 31-й бит зарезервирован. Поле используется при ***security***= «*usemask*».
- **year** (integer) – текущее значение года;

- **mon** (integer) – текущее значение месяца (от 1 до 12);
- **day** (integer) – текущее значение дня (от 1 до 31);
- **hour** (integer) – текущее значение часа (от 0 до 23);
- **min** (integer) – текущее значение минуты (от 0 до 59);
- **tz** (string) – текущая временная зона пользователя. Если не заданы поля ***year, mon, day, hour, min***, то используется текущее время сервера и добавляется указанная временная зона. При установке значения «*guess*», строится предположение о временной зоне пользователя исходя из предыдущих записей.
- **props** (struct) – свойства записи. *См.* Приложение Б.
- **usejournal** (string) – имя журнала (пользователя) для размещения записи.

Пример запроса ***postevent***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
<methodName>LJ.XMLRPC.postevent</methodName>
<params><param>
<value><struct>
<member><name>username</name>
<value><string>test</string></value></member>
<member><name>password</name>
<value><string>test</string></value></member>
<member><name>event</name>
<value><string>This is a test post.
</string></value></member>
<member><name>subject</name>
<value><string>Test</string></value></member>
<member><name>lineendings</name>
<value><string>pc</string></value></member>
<member><name>year</name>
<value><int>2002</int></value></member>
<member><name>mon</name>
<value><int>7</int></value></member>
<member><name>day</name>
<value><int>13</int></value></member>
<member><name>hour</name>
<value><int>20</int></value></member>
<member><name>min</name>
<value><int>35</int></value></member>
</struct></value>
```

```
</param></params>
</methodCall>
```

Возвращаемые значения

Структура, содержащая следующие поля:

- **itemid** (integer) – внутренний идентификатор записи;
- **anum** (integer) – случайное число от 0 до 255, построенное при создании записи. Используется в качестве младшего байта во внешнем идентификаторе записи.
- **url** (string) – ссылка на запись (URL);
- **warnings** (string) – системное уведомление (предупреждение) об ошибках при публикации записи. Ошибки связаны с пропущенными и не закрытыми тегами.

Пример ответа на запрос ***postevent***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
<params><param>
<value><struct>
<member><name>itemid</name>
<value><int>5</int></value></member>
<member><name>url</name>
<value><string>http://testtw.livejournal.com/1477.html</string></value>
</member>
<member><name>anum</name>
<value><int>197</int></value></member>
</struct></value>
</param></params>
</methodResponse>
```

Возвращаемые ошибки

102 – невозможно использовать приватные (подзамочные) записи в сообществах/новостных журналах ("Can't use custom/private security on shared/community journals").

103 – ошибка при создании голосования ("Poll error").

150 – невозможно делать записи в чужом журнале и/или невозможно делать записи незарегистрированному пользователю ("Can't post as non-user").

151 – пользователь заблокирован в данном журнале (сообществе) ("Banned from journal").

152 – невозможно делать записи с параметром «внеочередная дата» в сообществе ("Can't make back-dated entries in non-personal journal").

153 – неправильное значение времени ("Incorrect time value").

155 – неподтвержденный email адрес ("Non-authenticated email address").

200 – отсутствует необходимый параметр ("Missing required argument(s)").

203 – недопустимый параметр ("Invalid argument(s)").

301 – доступ к заблокированной функции ("Access of restricted feature").

305 – действие запрещено, аккаунт заморожен ("Action forbidden; account is suspended").

306 – журнал временно находится в режиме только для чтения. Попробуйте еще раз через пару минут ("This journal is temporarily in read-only mode. Try again in a couple minutes").

307 – выбранный журнал больше не существует ("Selected journal no longer exists").

308 – аккаунт заблокирован и не может быть использован ("Account is locked and cannot be used").

309 – статус журнала «memorial». Журнал опубликован, но писать в него нельзя ("Account is marked as a memorial").

310 – возраст владельца аккаунта нужно проверить, чтобы убедиться, что он не младше 18 лет ("Account needs to be age verified before use").

312 – не разрешается добавлять теги к записям в этом журнале ("Not allowed to add tags to entries in this journal").

316 – пользователь находится в режиме «только чтение» и не может писать записи ("Poster is read-only and cannot post entries").

317 – журнал в режиме «только чтение» и записи не могут быть в нем опубликованы ("Journal is read-only and entries cannot be posted to it").

404 – невозможно сделать запись ("Cannot post").

405 – достигнут предел частоты записей ("Post frequency limit").

407 – очередь на модерацию переполнена ("Moderation queue full").

408 – достигнуто максимальное число поставленных в очередь записей для данного сочетания «сообщество-автор» ("Maximum queued posts for this community+poster> combination reached").

409 – запись слишком большая ("Post too large").

410 – пробный период использования эккаунта истек. Невозможно опубликовать запись ("Your trial account has expired. Posting now disabled").

501 – ошибка базы данных ("Database error").

503 – ошибка при получении необходимой блокировки ("Error obtaining necessary database lock").

2.5.3.2. Функция getevents

Описание

Функция возвращает записи из журнала пользователя.

Параметры

Структура, содержащая следующие поля:

- **параметры аутентификации** (см. *п. 2.3.1*);
- **truncate** (integer) – число символов, до которого сокращается возвращаемая запись после декодирования;

- **prefersubject** (boolean) – при истинном значении поля вместо текста записи (значения поля ***event***) возвращается тема записи, а значение поля ***subject*** возвращается пустым.
- **noprops** (boolean) – при истинном значении поля для записей не возвращаются свойства;
- **notags** (boolean) – при истинном значении поля для записей не возвращаются теги;
- **selecttype** (string) – тип выборки событий из журнала. Возможные варианты:
    - **one** – получить одну указанную запись;
    - **day** – получить записи за указанный день (максимальное число - 200);
    - **lastn** – получить указанное число последних записей;
    - **syncitems** – получить записи, созданные позже указанного времени;
    - **multiple** – получить указанные записи по идентификаторам (максимальное количество – 100 записей);
    - **before** – получить записи, созданные до указанной даты;
- **lastsync** (string) – записи, созданные со времени, задаваемого в формате «*yyyy-mm-dd hh:mm:ss*», где *yyyy* – год, *mm* – месяц, *dd* – день, *hh* – час, *mm* – минута, *ss* – секунда. Поле используется при ***selecttype*=** «*syncitems*»*.*
- **year** (integer) – год. Поле используется при ***selecttype*=** «*day*»*.*
- **month** (integer) – месяц (от 1 до 12). Поле используется при ***selecttype*=** «*month».*
- **day** (integer) – день (от 1 до 31). Поле используется при ***selecttype=*** *«day».*
- **howmany/itemshow** (integer) – число возвращаемых записей (от 0 до 50). *Значение по умолчанию – 20*. Поле используется при ***selecttype*=** «*lastn*»*.*
- **skip** (integer) – число пропускаемых записей (от 0 до 500).
- **before** (string) – записи, созданные до указанного времени (в формате «*yyyy-mm-dd hh:mm:ss*», где *yyyy* – год, *mm* – месяц, *dd* – день, *hh* – час, *mm* – минута, *ss* – секунда). Поле используется при ***selecttype*=** «*before*»*.*
- **beforedate** (string) – дата (в формате «*yyyy-mm-dd hh:mm:ss*», где *yyyy* – год, *mm* – месяц, *dd* – день, *hh* – час, *mm* – минута, *ss* – секунда), до которой выводится список созданных записей. Поле используется при ***selecttype*=** «*lastn*»*.*
- **itemid** (integer) – внутренний идентификатор записи. Поле используется при ***selecttype****=one.*
- **itemids** (array) – массив внутренних идентификаторов записей, которые нужно получить. Поле используется при ***selecttype*=** «*multiple*»*.*
- **lineendings** (string) – символы, используемые для разделения строк в записи («*space*» – перевод строк заменяется пробелом, «*dots*» – перевод строк заменяется многоточием);
- **usejournal** (string) – имя журнала, доступного пользователю, из которого необходимо получить записи;
- **trim_widgets** (integer) – число символов, до которого сокращается каждая запись. Сокращение производится по словам (с учетом тегов и картинок).

- **widgets_img_length** (integer) – число символов, которое занимает картинка в записи. *Значение по умолчанию – 50.*
- **parseljtags** (boolean) – раскрывать (преобразовывать) LJ-теги в простой HTML вид. По умолчанию установлено значение *0* (ложь), при котором преобразование не осуществляется. При истинном значении (*1*) осуществляется преобразование тегов.

Преобразование **LJ-тегов**:

- тег **lj-poll** заменяется на ссылку страницы с этим описанием:

```
<a href="http://livejournal.com/poll/?id=<pollid>" target="_blank">View Poll: <pll name>.</a>
```

- тег **lj-embed** заменяется на ссылку вида:

```
<a href="<embed url>">View movie.</a>
```

- тег **lj-user** заменяется на ссылку журнала пользователя:

```
<a href="<user_profile_url>" target="_blank"><img src="<userinfo.gif>" lt=""></a><a href="<journalbase_url>"><username></a>
```

Пример запроса ***getevents***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
<methodName>LJ.XMLRPC.getevents</methodName>
<params><param>
<value><struct>
<member><name>username</name>
<value><string>test</string></value></member>
<member><name>password</name>
<value><string>test</string></value></member>
<member><name>ver</name>
<value><int>1</int></value></member>
<member><name>truncate</name>
<value><int>20</int></value></member>
<member><name>selecttype</name>
<value><string>lastn</string></value></member>
<member><name>howmany</name>
<value><int>2</int></value>
</member>
<member><name>noprops</name>
<value><boolean>1</boolean></value></member>
<member><name>lineendings</name>
<value><string>unix</string></value></member>
</struct></value>
```

```
</param></params>
</methodCall>
```

Возвращаемые значения

Структура, содержащая следующие поля:

- **skip** (integer) – число пропущенных записей. Соответствует значению входного поля ***skip***.
- **events** (array) – массив, содержащий структуры со следующими полями:
  - **itemid** (integer) – внутренний идентификатор записи;
  - **subject** (string) – тема записи;
  - **event** (string) – текст записи;
  - **eventtime** (string) – время публикации записи, заданное пользователем (в формате «*yyyy-mm-dd hh:mm:ss*», где *yyyy* – год, *mm* – месяц, *dd* – день, *hh* – час, *mm* – минута, *ss* – секунда);
  - **props** (struct) – свойства записи. *См.* Приложение Б.
  - **url** (string) – ссылка на запись (URL);
  - **anum** (integer) – случайное число от 0 до 255, построенное при создании записи. Используется в качестве младшего байта во внешнем идентификаторе записи.
  - **event_timestamp** (string) – Unix-время записи, заданное пользователем;
  - **reply_count** (integer) – число комментариев к записи;
  - **security** (string) – уровень доступа к записи. При значении поля «*usemask*» уровень доступа определяется значением поля ***allowmask***.
  - **allowmask** (integer) – битовая маска, задающая группы пользователей, которым будет доступна публикуемая запись. Беззнаковое 32-х битное целое, где 0-й бит - все друзья пользователя, биты с 1 по 30 - соответствуют группам пользователей, 31-й бит зарезервирован. Поле используется при ***security*** = *«usemask*».

Пример ответа на запрос ***getevents***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
<params><param>
<value><struct>
<member><name>events</name>
<value><array><data>
<value><struct>
<member><name>eventtime</name>
<value><string>2020-02-20 02:20:00</string></value></member>
<member><name>event</name>
<value><string>yes its true its ...</string></value></member>
<member><name>anum</name>
<value><int>108</int></value></member>
```

```
<member><name>itemid</name>
<value><int>1965</int></value></member>
</struct></value>
<value><struct>
<member><name>eventtime</name>
<value><string>2002-07-14 11:17:00</string></value></member>
<member><name>event</name>
<value><string>Yes, Yes, YES!</string></value></member>
<member><name>anum</name>
<value><int>66</int></value></member>
<member><name>subject</name>
<value><string>Is this private?</string></value></member>
<member><name>itemid</name>
<value><int>1964</int></value></member>
</struct></value>
</data></array></value></member>
</struct></value>
</param></params>
</methodResponse>
```

Возвращаемые ошибки

200 – отсутствует необходимый параметр ("Missing required argument(s)").

203 – недопустимый параметр ("Invalid argument(s)").

207 – ошибочная версия протокола ("Protocol version mismatch").

208 – недопустимая кодировка текста ("Invalid text encoding").

209 – параметр вне диапазона ("Parameter out of range").

307 – выбранный журнал больше не существует ("Selected journal no longer exists").

311 – доступ временно отключен ("Access temporarily disabled").

406 – клиент посылает повторяющиеся запросы. Возможно он сломан? ("Client is making repeated requests. Perhaps it's broken?").

501 – ошибка базы данных ("Database error").

502 – база данных временно недоступна ("Database temporarily unavailable").

506 – синхронизация журнала временно недоступна (“Journal sync temporarily unavailable").

2.5.3.3. Функция editevent

Описание

Функция редактирует запись, созданную ранее.

Параметры

Структура, содержащая следующие поля:

- **параметры аутентификации** (см. *п. 2.3.1*);
- **itemid** (integer) – внутренний идентификатор записи.
- **event** (string) – текст записи. Для удаления записи в этом поле нужно передать пустую строку. Записи также могут содержать HTML теги, но сервер LiveJournal конвертирует переводы строк в HTML-теги
 при отображении.
- **lineendings** (string) – символы, используемые для разделения строк в записи («*unix*» - 0x0A (\n); «*pc*» - 0x0D0A (\r\n); «*mac*» - 0x0D (\r));
- **subject** (string) – тема записи;
- **security** (string) – уровень доступа к записи. Значения поля: «*public*» – публичный (*значение по умолчанию*), «*private*» – приватный, «*usemask*» – по группам пользователей. При значении поля «*usemask*» уровень доступа задается полем ***allowmask***.
- **allowmask** (integer) – маска, задающая пользовательские группы, которым доступна эта запись. Беззнаковое 32-х битное целое, где 0-й бит – все друзья пользователя, биты с 1 по 30 – соответствуют группам пользователей, 31-й бит зарезервирован. Поле используется при ***security***= *«usemask*».
- **year** (integer) – год публикации записи;
- **mon** (integer) – месяц публикации записи (от 1 до 12);
- **day** (integer) – день публикации записи (от 1 до 31);
- **hour** (integer) – время публикации записи, час (от 0 до 24)*;*
- **min** (integer) – время публикации записи, минуты (от 0 до 59)*;*
- **props** (string) – свойства записи. *См.* Приложение Б.
- **usejournal** (string) – имя журнала (пользователя) для размещения записи.

Пример запроса ***editevent***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
<methodName>LJ.XMLRPC.editevent</methodName>
<params><param>
<value><struct>
<member><name>username</name>
<value><string>test</string></value></member>
<member><name>password</name>
<value><string>test</string></value></member>
<member><name>itemid</name>
<value><int>1959</int></value></member>
<member><name>event</name>
<value><string>This &lt;strike&gt;is&lt;/strike&gt; was a test post.
</string></value></member>
<member><name>subject</name>
```

```
<value><string>Test</string></value></member>
<member><name>lineendings</name>
<value><string>pc</string></value></member>
<member><name>year</name>
<value><int>2002</int></value></member>
<member><name>mon</name>
<value><int>7</int></value></member>
<member><name>day</name>
<value><int>13</int></value></member>
<member><name>hour</name>
<value><int>20</int></value></member>
<member><name>min</name>
<value><int>35</int></value></member>
</struct></value>
</param></params>
</methodCall>
```

Возвращаемые значения

Структура, содержащая следующие поля:

- **itemid** (integer) – внутренний идентификатор записи;
- **anum** (integer) – случайное число от 0 до 255, построенное при создании записи. Используется в качестве младшего байта во внешнем идентификаторе записи.
- **url** (string) – ссылка на запись (URL);
- **warnings** (string) – системное уведомление (предупреждение) об ошибках при публикации записи. Ошибки связаны с пропущенными и не закрытыми тегами.

Пример ответа на запрос ***editevent***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
<params><param>
<value><struct>
<member><name>anum</name>
<value><int>141</int></value></member>
<member><name>itemid</name>
<value><int>1959</int></value></member>
</struct></value>
</param></params>
</methodResponse>
```

Возвращаемые ошибки

102 – невозможно использовать приватные (подзамочные) записи в сообществах/новостных журналах ("Can't use custom/private security on shared/community journals").

152 – невозможно делать записи с параметром «внеочередная дата» в сообществе ("Can't make back-dated entries in non-personal journal").

203 – недопустимый параметр ("Invalid argument(s)").

210 – Предупреждение. Пользователь пытается редактировать запись, содержащую поврежденные данные ("Client tried to edit with corrupt data. Preventing").

302 – невозможно редактировать запись из запрашиваемого журнала ("Can't edit post from requested journal").

303 – невозможно редактировать запись в сообществе ("Can't edit post in community journal").

304 – невозможно удалить запись в сообществе ("Can't delete post in this community journal").

306 – журнал временно находится в режиме только для чтения. Попробуйте еще раз через пару минут ("This journal is temporarily in read-only mode. Try again in a couple minutes").

307 – выбранный журнал больше не существует ("Selected journal no longer exists").

310 – возраст владельца аккаунта нужно проверить, чтобы убедиться, что он не младше 18 лет ("Account needs to be age verified before use").

318 – пользователь находится в режиме «только чтение» и не может редактировать записи ("Poster is read-only and cannot edit entries").

319 – журнал в режиме «только чтение» и его записи не могут быть отредактированы ("Journal is read-only and its entries cannot be edited").

409 – запись слишком большая ("Post too large").

501 – ошибка базы данных ("Database error").

2.5.3.4. Функция syncitems

Описание

Функция возвращает список (или часть списка) элементов (записей журнала, комментариев), которые были созданы или обновлены в журнале пользователя, начиная с указанного времени. Возвращаются не сами элементы, а только тип элемента и его идентификационный номер.

Параметры

Структура, содержащая следующие поля:

- **параметры аутентификации** (см. *п. 2.3.1*);
- **lastsync** (string) – время, начиная с которого необходимо получить изменения (в формате «*yyyy-mm-dd hh:mm:ss*», где *yyyy* – год, *mm* – месяц, *dd* – день, *hh* – час, *mm* – минута, *ss* – секунда). *Значение по умолчанию – 1 января 1970 года 00:00:00 GMT*. При установке даты последнего вызова этой функции, вернется список обновленных элементов.

Пример запроса ***syncintems***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
<methodName>LJ.XMLRPC.syncitems</methodName>
<params><param>
<value><struct>
<member><name>username</name>
<value><string>test</string></value></member>
<member><name>password</name>
<value><string>test</string></value></member>
<member><name>ver</name>
<value><int>1</int></value></member>
<member><name>lastsync</name>
<value><string>2002-07-13 00:00:00</string></value></member>
</struct></value>
</param></params>
</methodCall>
```

Возвращаемые значения

Структура, содержащая следующие поля:

- **syncitems** (array) – массив структур, содержащих следующие поля:
  - **item** (string) – элемент в формате *«тип»-«идентификатор»*. Возможны следующие типы: «*L*» - для записей, «*C*» - для комментариев. Используются внутренние идентификаторы для записей и комментариев.
  - **action** (string) – статус элемента: либо «*create»* - элемент вновь создан, либо «*update»* - элемент обновлен.
  - **time** (string) – серверное время (в формате «*yyyy-mm-dd hh:mm:ss*», где *yyyy* – год, *mm* – месяц, *dd* – день, *hh* – час, *mm* – минута, *ss* – секунда) создания (обновления) элемента;
- **count** (integer) – количество измененных (обновленных) элементов (нумерация начинается с 1);
- **total** (integer) – общее количество элементов, которые были обновлены с указанного времени.

Пример ответа на запрос ***syncintems***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
<params><param>
<value><struct>
<member><name>total</name>
<value><int>11</int></value></member>
<member><name>count</name>
```

<value><int>11</int></value></member>
<member><name>syncitems</name>
<value><array><data>
<value><struct>
<member><name>item</name>
<value><string>L-1947</string></value></member>
<member><name>time</name>
<value><string>2002-07-13 00:06:26</string></value></member>
<member><name>action</name>
<value><string>del</string></value></member>
</struct></value>
<value><struct>
<member><name>item</name>
<value><string>L-1954</string></value></member>
<member><name>time</name>
<value><string>2002-07-13 00:09:05</string></value></member>
<member><name>action</name>
<value><string>del</string></value></member>
</struct></value>
<value><struct>
<member><name>item</name>
<value><string>L-1958</string></value></member>
<member><name>time</name>
<value><string>2002-07-13 02:01:07</string></value></member>
<member><name>action</name>
<value><string>create</string></value></member>
</struct></value>
<value><struct>
<member><name>item</name>
<value><string>L-1948</string></value></member>
<member><name>time</name>
<value><string>2002-07-13 08:27:56</string></value></member>
<member><name>action</name>
<value><string>update</string></value></member>
</struct></value>
<value><struct>
<member><name>item</name>
<value><string>L-1960</string></value></member>
<member><name>time</name>
<value><string>2002-07-14 02:52:18</string></value></member>

<member><name>action</name>
<value><string>create</string></value></member>
</struct></value>
<value><struct>
<member><name>item</name>
<value><string>L-1961</string></value></member>
<member><name>time</name>
<value><string>2002-07-14 03:07:55</string></value></member>
<member><name>action</name>
<value><string>create</string></value></member>
</struct></value>
<value><struct>
<member><name>item</name>
<value><string>L-1962</string></value></member>
<member><name>time</name>
<value><string>2002-07-14 03:08:14</string></value></member>
<member><name>action</name>
<value><string>create</string></value>
</member>
</struct></value>
<value><struct>
<member><name>item</name>
<value><string>L-1963</string></value></member>
<member><name>time</name>
<value><string>2002-07-14 03:13:26</string></value></member>
<member><name>action</name>
<value><string>create</string></value></member>
</struct></value>
<value><struct>
<member><name>item</name>
<value><string>L-1964</string></value></member>
<member><name>time</name>
<value><string>2002-07-14 03:17:03</string></value></member>
<member><name>action</name>
<value><string>create</string></value></member>
</struct></value>
<value><struct>
<member><name>item</name>
<value><string>L-1959</string></value></member>
<member><name>time</name>

```
<value><string>2002-07-14 14:25:07</string></value></member>
<member><name>action</name>
<value><string>update</string></value></member>
</struct></value>
<value><struct>
<member><name>item</name>
<value><string>L-1965</string></value></member>
<member><name>time</name>
<value><string>2002-07-16 04:36:15</string></value></member>
<member><name>action</name>
<value><string>update</string></value></member>
</struct></value>
</data></array></value></member>
</struct></value>
</param></params>
</methodResponse>
```

Возвращаемые ошибки

203 – недопустимый параметр ("Invalid argument(s)").

502 – база данных временно недоступна ("Database temporarily unavailable").

506 – синхронизация журнала временно недоступна (“Journal sync temporarily unavailable").

2.5.3.5. Функция getdaycounts

Описание

Функция возвращает количество записей в журнале по дням.

Параметры

Структура, содержащая следующие поля:

- **параметры аутентификации** (см. *п. 2.3.1*);
- **usejournal** (string) – имя журнала пользователя из которого необходимо получить количество записей. *Значение по умолчанию – журнал текущего пользователя*.

Пример запроса ***getdaycounts***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
<methodName>LJ.XMLRPC.getdaycounts</methodName>
<params><param>
<value><struct>
<member><name>username</name>
<value><string>test</string></value></member>
```

```
<member><name>password</name>
<value><string>test</string></value></member>
<member><name>ver</name>
<value><int>1</int></value></member>
</struct></value>
</param></params>
```

Возвращаемые значения

Структура, содержащая следующие поля:

- **daycounts** (array) – массив, содержащий структуры со следующими полями:
  - **date** (string) – дата (в формате *«yyyy-mm-dd»*, где *yyyy* – год, *mm* – месяц, *dd* - день);
  - **count** (integer) – число записей в журнале за указанную в поле ***date*** дату;

Пример ответа на запрос ***getdaycounts***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
<params><param>
<value><struct>
<member><name>daycounts</name>
<value><array>
<data><value><struct>
<member><name>count</name>
<value><int>1</int></value></member>
<member><name>date</name>
<value><string>2010-08-01</string></value></member>
</struct></value>
<value><struct>
<member><name>count</name><value>
<int>1</int></value></member>
<member><name>date</name>
<value><string>2010-08-02</string></value></member>
</struct></value>
<value><struct>
<member><name>count</name>
<value><int>2</int></value></member>
<member><name>date</name>
<value><string>2010-08-03</string></value></member>
</struct></value>
<value><struct>
```

```
<member><name>count</name>
<value><int>1</int></value></member>
<member><name>date</name>
<value><string>2010-09-13</string></value></member>
</struct></value>
</data></array></value>
</member></struct></value>
</param></params>
</methodResponse>
```

Возвращаемые ошибки

203 – недопустимый параметр ("Invalid argument(s)").

204 – недопустимый тип метаданных ("Invalid metadata datatype").

205 – неизвестные метаданные ("Unknown metadata").

207 – ошибочная версия протокола ("Protocol version mismatch").

208 – недопустимая кодировка текста ("Invalid text encoding").

211 – недопустимый или искаженный список тегов ("Invalid or malformed tag list").

2.5.3.6. Функция getusertags

Описание

Функция возвращает список определенных пользователем тегов.

Параметры

Структура, содержащая следующие поля:

- **параметры аутентификации** (см. *п. 2.3.1*);
- **usejournal** (string) – имя журнала (пользователя) для которого необходимо получить список тегов.

Пример запроса ***getusertags***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
<methodName>LJ.XMLRPC.getusertags</methodName>
<params><param>
<value><struct>
<member><name>username</name>
<value><string>test</string></value></member>
<member><name>password</name>
<value><string>test</string></value></member>
<member><name>ver</name>
<value><int>1</int></value></member>
```

```
</struct></value>
</param></params>
</methodCall>
```

Возвращаемые значения

Структура, содержащая следующие поля:

- **tags** (array) – массив, содержащий структуры со следующими полями:
  - **name** (string) – имя тега;
  - **security_level** (string) – уровень безопасности (видимости) тега. Возможные значения: «*public*», «*private*», «*friends*» или «*group*». Перечень идентификаторов для групп («*group*») можно получить из параметра поля ***security***.
  - **uses** (integer) – количество использований тега;
  - **display** (string) – при значении поля «*on*» тег виден для системы стилей S2[6]. При значении поля «*off*» теги доступны, но не отображаются в системе стилей S2.
  - **security** (struct) – структура, описывающая статистику использования тега по категориям безопасности:
    - **public** (integer) – количество использований тега в публичных записях;
    - **private** (integer) – количество использований тега в приватных записях;
    - **friends** (integer) – количество использований тега в записях только для друзей;
    - **groups** (struct) – структура, содержащая в качестве полей идентификаторы групп пользователей, в качестве значений этих полей соответствующие количество использований этого тега в записях, принадлежащих этой группе пользователей.

Пример ответа на запрос ***getusertags***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
<params><param>
<value><struct>
<member><name>tags</name>
<value><array><data>
<value><struct>
<member><name>security_level</name>
<value><string>private</string>
</value></member>
<member><name>uses</name>
```

[6] http://www.livejournal.com/doc/s2

<value><int>0</int></value></member>
<member><name>security</name><value><struct>
<member><name>friends</name>
<value><int>0</int></value></member>
<member><name>groups</name>
<value><struct /></value></member>
<member><name>private</name>
<value><int>0</int></value></member>
<member><name>public</name>
<value><int>0</int></value></member>
</struct></value></member>
<member><name>name</name><value>
<base64>0YLQtdGB0YIy</base64></value></member>
<member><name>display</name>
<value><int>1</int></value></member>
</struct></value>
<value><struct>
<member><name>security_level</name>
<value><string>private</string></value></member>
<member><name>uses</name>
<value><int>0</int></value></member>
<member><name>security</name>
<value><struct>
<member><name>friends</name>
<value><int>0</int></value></member>
<member><name>groups</name>
<value><struct /></value></member>
<member><name>private</name>
<value><int>0</int></value></member>
<member><name>public</name>
<value><int>0</int></value></member>
</struct></value></member>
<member><name>name</name>
<value><string>tag1</string></value></member>
<member><name>display</name>
<value><int>1</int></value></member>
</struct></value>
</data></array></value></member>
</struct></value>
</param></params>

```
</methodResponse>
```

Возвращаемые ошибки

100 – недопустимое имя пользователя ("Invalid username").

2.5.4. Друзья пользователя

Одной из основных возможностей LiveJournal является создание собственных списков **друзей**. Пользователи-друзья обладают определенными привилегиями по сравнению с другими: они могут читать защищенные записи пользователя, комментировать записи. Для получения списка друзей пользователя следует вызвать функцию ***getfriends*** (см. *п.* ***2.5.4.1***). Для того чтобы проверить, поменялся ли список пользователей с указанной даты используется функция ***checkfriends*** (см. *п.* ***2.5.4.3***).

Пользователь может отредактировать список друзей функцией ***editfriends*** (см. *п.* ***2.5.4.4***). Список пользователей, добавивших пользователя в свои друзья, можно получить, вызвав функцию ***friendof*** (см. *п.* ***2.5.4.2***).

В некоторых случаях одной группы друзей начинает не хватать. Это происходит, например, когда количество друзей пользователя становится настолько большим, что читать все записи в ленте друзей не представляется возможным; либо когда нужно открыть доступ к какой-то записи не всем своим друзьям, а лишь определенной части. В таком случае LiveJournal предлагает пользователю возможность выделить в своих друзьях определенные **группы пользователей**, с которыми можно обращаться точно так же, как с основной группой друзей — читать записи в ленте друзей только от пользователей определенной группы, ограничивать область видимости своих записей для пользователей определенной группы и т.п. Пользователь может распределить список друзей на тех, с кем он работает, учится, на друзей, которых он знает лично, или в сети, членов семьи. Пользователь может переименовывать группы, создавать новые, удалять ненужные. Для получения списка групп пользователей используется функция ***getfriendgroups*** (см. *п.* ***2.5.4.5***). Для редактирования групп пользователей следует вызвать функцию ***editfriendgroups*** (см. *п.* ***2.5.4.6***).

2.5.4.1. Функция getfriends

Описание

Функция возвращает список друзей пользователя.

Параметры

Структура, содержащая следующие поля:

- **параметры аутентификации** (см. *п.* ***2.3.1***);
- **includefriendof** (boolean) – при истинном значении поля, выводится информация о пользователях, добавивших текущего пользователя в друзья;
- **includegroups** (boolean) – при истинном значении поля, выводится информация о группах друзей;
- **includebdays** (boolean) – при истинном значении поля, выводится информация о дне рождения каждого пользователя;
- **friendlimit** (integer) – количество описаний друзей, которое необходимо получить;
- **friendoflimit** (integer) – количество описаний друзей друзей, которое необходимо получить. Поле используется при истинном значении поля ***includefriendof***.

Пример запроса ***getfriends***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
<methodName>LJ.XMLRPC.getfriends</methodName>
<params><param>
<value><struct>
<member><name>username</name>
<value><string>test</string></value></member>
<member><name>password</name>
<value><string>test</string></value></member>
<member><name>ver</name>
<value><int>1</int></value></member>
</struct></value>
</param></params>
</methodCall>
```

Возвращаемые значения

Структура, содержащая следующие поля:

- **friendgroups** (array) – массив описаний каждой пользовательской группы.
- **friendofs** (array) – массив описаний пользователей, включивших данного пользователя в друзья.
- **friends** (array) – массив описаний пользователей - друзей пользователя.

Описание группы друзей. Структура, содержащая следующие поля:

- **id** (integer) – номер бита для этой группы друзей (от 1 до 30);
- **name** (string) – имя группы друзей;
- **sortorder** (integer) – число для упорядочивания групп (от 0 до 255);
- **public** (boolean) – флаг публичности группы.

Описание пользователя. Структура, содержащая следующие поля:

- **username** (string) – имя пользователя;
- **fullname** (string) – полное имя пользователя;
- **type** (string) – тип пользователя;
- **identity_type** (string) – альтернативный тип идентификации. Поле используется при ***type*=** «*identity*»;
- **identity_value** (string) – идентификатор (URL). Поле используется при ***type*=** «*identity*»;

- **identity_display** (string) – отображаемое имя пользователя. Поле используется при ***type*=** «*identity*»;
- **fgcolor** (string) – цвет символов при отображении записей этого пользователя в ленте друзей;
- **bgcolor** (string) – цвет фона при отображении записей этого пользователя в ленте друзей;
- **groupmask** (integer) – маска групп пользователей;
- **birthday** (string) – день рождения пользователя (в формате «*yyyy-mm-dd hh:mm:ss*», где *yyyy* – год, *mm* – месяц, *dd* – день, *hh* – час, *mm* – минута, *ss* – секунда);
- **defaultpicurl** (string) – ссылка на картинку пользователя по умолчанию (URL). Поле присутствует, если у пользователя задана картинка по умолчанию.

Пример ответа на запрос ***getfriends***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
<params><param>
<value><struct>
<member><name>friends</name>
<value><array><data>
<value><struct>
<member><name>fgcolor</name>
<value><string>#000000</string></value></member>
<member><name>username</name>
<value><string>bradfitz</string></value></member>
<member><name>fullname</name>
<value><string>Brad Fitzpatrick</string></value></member>
<member><name>bgcolor</name>
<value><string>#ffffff</string></value></member>
</struct></value>
<value><struct>
<member><name>fgcolor</name>
<value><string>#efcfff</string></value></member>
<member><name>username</name>
<value><string>ljfresno</string></value>
</member>
<member><name>fullname</name>
<value><string>Fresno LJ users</string></value></member>
<member><name>type</name>
<value><string>community</string></value></member>
```

```
<member><name>bgcolor</name>
<value><string>#000000</string></value></member>
</struct></value>
<value><struct>
<member><name>fgcolor</name>
<value><string>#520155</string></value></member>
<member><name>username</name>
<value><string>webkin</string></value></member>
<member><name>fullname</name>
<value><string>Ellen Stafford</string></value></member>
<member><name>bgcolor</name>
<value><string>#fcddff</string></value></member>
</struct></value>
</data></array></value></member>
</struct></value>
</param></params>
</methodResponse>
```

Возвращаемые ошибки

502 – база данных временно недоступна ("Database temporarily unavailable").

2.5.4.2. Функция friendof

Описание

Функция возвращает список пользователей, которые пометили пользователя своим другом.

Параметры

Структура, содержащая следующие поля:

- **параметры аутентификации** (см. *п. 2.3.1*);
- **friendoflimit** (integer) – если значение поля больше *0*, то возвращается число пользователей, не больше указанного.

Пример запроса ***friendof***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
<methodName>LJ.XMLRPC.friendof</methodName>
<params><param>
<value><struct>
<member><name>username</name>
<value><string>test</string></value></member>
```

```
<member><name>password</name>
<value><string>test</string></value></member>
<member><name>ver</name>
<value><int>1</int></value></member>
<member><name>friendoflimit</name>
<value><int>2</int></value></member>
</struct></value>
</param></params>
</methodCall>
```

Возвращаемые значения

Структура, содержащая следующие поля:

- **friendofs** (array) – массив, описывающий каждого пользователя.

Описание пользователя. Структура, содержащая следующие поля:

- **username** (string) – имя пользователя;
- **fullname** (string) – полное имя пользователя;
- **type** (string) – тип пользователя;
- **identity_type** (string) – альтернативный тип идентификации. Поле используется при ***type***=*identity*;
- **identity_value** (string) – идентификатор (URL). Поле используется при ***type***=*identity*;
- **identity_display** (string) – отображаемое имя пользователя. Поле используется при ***type***=*identity*;
- **fgcolor** (string) – цвет символов при отображении записей этого пользователя в ленте друзей;
- **bgcolor** (string) – цвет фона при отображении записей этого пользователя в ленте друзей;
- **groupmask** (integer) – маска групп пользователей;
- **birthday** (string) – день рождения пользователя (в формате «*yyyy-mm-dd hh:mm:ss*», где *yyyy* – год, *mm* – месяц, *dd* – день, *hh* – час, *mm* – минута, *ss* – секунда);
- **defaultpicurl** (string) – ссылка на картинку пользователя по умолчанию (URL). Поле присутствует, если у пользователя задана картинка по умолчанию.

Пример ответа на запрос ***friendof***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
<params><param>
<value><struct>
```

```
<member><name>friendofs</name>
<value><array><data>
<value><struct>
<member><name>fgcolor</name>
<value><string>#000000</string></value></member>
<member><name>username</name>
<value><string>aisha_plushie</string></value></member>
<member><name>fullname</name>
<value><string>Stripe</string></value></member>
<member><name>bgcolor</name>
<value><string>#ffffff</string></value></member>
</struct></value>
<value><struct>
<member><name>fgcolor</name>
<value><string>#000000</string></value></member>
<member><name>username</name>
<value><string>badcharlotte</string></value></member>
<member><name>fullname</name>
<value><string>Charlotte</string></value></member>
<member><name>bgcolor</name>
<value><string>#ffffff</string></value></member>
</struct></value>
</data></array></value></member>
</struct></value>
</param></params>
</methodResponse>
```

Возвращаемые ошибки

502 – база данных временно недоступна ("Database temporarily unavailable")

2.5.4.3. Функция checkfriends

Описание

Функция проверяет, поменялся ли список друзей с указанного времени.

Параметры

Структура, содержащая следующие поля:

- **параметры аутентификации** (см. *п. 2.3.1*);
- **lastupdate** (string) – время предыдущего запроса функции (в формате «*yyyy-mm-dd hh:mm:ss*», где *yyyy* – год, *mm* – месяц, *dd* – день, *hh* – час, *mm* – минута, *ss* – секунда).

- **mask** (integer) – группы друзей, в которых пользователь проверяет наличие новых элементов. Установка любой комбинации битов 1-30 позволит проверить наличие новых друзей в соответствующих группах. Установка бита 0 или сброс всех битов позволит проверить элементы для всех групп.

Пример запроса ***checkfriends***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
<methodName>LJ.XMLRPC.checkfriends</methodName>
<params><param>
<value><struct>
<member><name>username</name>
<value><string>test</string></value></member>
<member><name>password</name>
<value><string>test</string></value></member>
<member><name>ver</name>
<value><int>1</int></value></member>
<member><name>lastupdate</name>
<value><string></string></value></member>
</struct></value>
</param></params>
</methodCall>
```

Возвращаемые значения

Структура, содержащая следующие поля:

- **new** (boolean) – наличие новых элементов. При значении «*1*» - новые элементы присутствуют, при значении «*0*» - новых элементов нет;
- **interval** (integer) – число секунд, которое необходимо подождать перед следующим обращением к серверу. Если обратиться к серверу раньше указанного времени, то сервер вернет ошибку;
- **count** (integer) – число элементов, содержащихся в данном ответе;
- **total** (integer) – число элементов, которые были обновлены с указанной в запросе даты (поле ***lastupdate***).

Пример ответа на запрос ***checkfriends***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
<params><param>
<value><struct>
<member><name>lastupdate</name>
```

```
<value><string>2002-07-16 14:22:16</string></value></member>
<member><name>new</name>
<value><int>0</int></value></member>
<member><name>interval</name>
<value><int>90</int></value></member>
</struct></value>
</param></params>
</methodResponse>
```

Возвращаемые ошибки

203 – недопустимый параметр ("Invalid argument(s)").

2.5.4.4. Функция editfriends

Описание

Функция позволяет добавлять, редактировать или удалять друзей из списка друзей.

Параметры

Структура, содержащая следующие поля:

- **параметры аутентификации** (см. *п. 2.3.1*);
- **delete** (array) – массив друзей (имен друзей) для удаления из списка друзей;
- **add** (array) – массив описаний пользователей для добавления в друзья.

Описание пользователя для добавления в друзья. Структура, содержащая следующие поля:

- **username** (string) – имя пользователя;
- **fgcolor** (string) – цвет символов при отображении записей этого пользователя в ленте друзей;
- **bgcolor** (string) – цвет фона при отображении записей этого пользователя в ленте друзей;
- **groupmask** (integer) – маска групп пользователей.

Пример запроса ***editfriends***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
<methodName>LJ.XMLRPC.editfriends</methodName>
<params><param>
<value><struct>
<member><name>username</name>
<value><string>test</string></value></member>
```

```
<member><name>password</name>
<value><string>test</string></value></member>
<member><name>ver</name>
<value><int>1</int></value></member>
<member><name>add</name>
<value><array><data>
<value><struct>
<member><name>username</name>
<value><string>bradfitz</string></value></member>
<member><name>fgcolor</name>
<value><string>#000000</string></value></member>
<member><name>bgcolor</name>
<value><string>#ffffff</string></value></member>
</struct></value></data>
</array></value></member>
</struct></value>
</param></params>
</methodCall>
```

Возвращаемые значения

Структура, содержащая следующие поля:

- **added** (array) – список добавленных друзей. Массив, содержащий структуры со следующими полями:
  - **fullname** (string) – полное имя пользователя, если оно задано;
  - **username** (string) – имя пользователя – друга;
  - **journaltype** (string) – тип журнала (см. *п. **2.5.1***);
  - **defaultpicurl** (string) – ссылка на картинку пользователя по умолчанию (URL), если она установлена.

Пример ответа на запрос ***editfriends***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
<params><param>
<value><struct>
<member><name>added</name>
<value><array><data>
<value><struct>
<member><name>username</name>
<value><string>bradfitz</string></value></member>
```

```
<member><name>fullname</name>
<value><string>Brad Fitzpatrick</string></value></member>
<member><name>defaultpicurl</name>
<value><string>http://l-userpic.livejournal.com/1891759/512232</string></value></member>
</struct></value>
</data></array></value></member>
</struct></value>
</param></params>
</methodResponse>
```

Возвращаемые ошибки

104 – ошибка добавления одного или нескольких друзей ("Error adding one or more friends").

154 – невозможно добавить в друзья пользователя с данным именем, так как он переименовал свой журнал ("Can't add a redirected account as a friend").

203 – недопустимый параметр ("Invalid argument(s)").

305 – действие запрещено, аккаунт заморожен ("Action forbidden; account is suspended").

306 – журнал временно находится в режиме только для чтения. Попробуйте еще раз через пару минут ("This journal is temporarily in read-only mode. Try again in a couple minutes").

308 – аккаунт заблокирован и не может быть использован ("Account is locked and cannot be used").

501 – ошибка базы данных ("Database error").

2.5.4.5. Функция getfriendgroups

Описание

Функция возвращает список определенных пользователем групп пользователей.

Параметры

Структура, содержащая следующие поля:

- **параметры аутентификации** (см. *п.* ***2.3.1***);

Пример запроса ***getfriendgroups***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
<methodName>LJ.XMLRPC.getfriendgroups</methodName>
<params><param>
<value><struct>
<member><name>username</name>
<value><string>test</string></value></member>
<member><name>password</name>
```

```
<value><string>test</string></value></member>
<member><name>ver</name>
<value><int>1</int></value></member>
</struct></value>
</param></params>
</methodCall>
```

Возвращаемые значения

Структура, содержащая следующие поля:

- **friendgroups** (array) – массив описаний групп пользователей.

Описание группы пользователей. Структура, содержащая следующие поля:

- **public** (boolean) – флаг публичности группы;
- **name** (string) – имя группы друзей;
- **id** (integer) – номер бита для этой группы друзей (от 1 до 30);
- **sortorder** (integer) – число, для упорядочивания групп (от 0 до 255).

Пример ответа на запрос ***getfriendgroups***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
<params><param>
<value><struct>
<member><name>friendgroups</name>
<value><array><data>
<value><struct>
<member><name>sortorder</name>
<value><int>25</int></value></member>
<member><name>id</name>
<value><int>1</int></value></member>
<member><name>public</name>
<value><int>1</int></value></member>
<member><name>name</name>
<value><string>Good Friends</string></value></member>
</struct></value>
</data></array></value></member>
</struct></value>
</param></params>
</methodResponse>
```

Возвращаемые ошибки

502 – база данных временно недоступна ("Database temporarily unavailable").

2.5.4.6. Функция editfriendgroups

Описание

Функция редактирует определенные пользователем группы пользователей.

Параметры

Структура, содержащая следующие поля:

- **параметры аутентификации** (см. *п. 2.3.1*);
- **groupmasks** (struct) – структура, содержащая пользовательские идентификаторы. Значением для каждого идентификатора является строка, представляющая беззнаковое 32-х битное целое, с 0-м установленным битом, биты 1-30 установлены для каждой группы, которой принадлежит этот пользователь, 31 бит зарезервирован.
- **set** (struct) – описание группы пользователей;
- **delete** (array) – массив номеров групп пользователей для удаления (от 1 до 30).

Описание группы пользователей. Структура, содержащая следующие поля:

- **public** (boolean) – флаг публичности группы;
- **name** (string) – имя группы друзей;
- **id** (integer) – номер бита для этой группы друзей (от 1 до 30);
- **sortorder** (integer) – число, для упорядочивания групп (от 0 до 255).

Пример запроса ***editfriendgroups***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
<methodName>LJ.XMLRPC.editfriendgroups</methodName>
<params><param>
<value><struct>
<member><name>username</name>
<value><string>test</string></value></member>
<member><name>password</name>
<value><string>test</string></value></member>
<member><name>ver</name>
<value><int>1</int></value></member>
<member><name>set</name>
<value><struct>
<member><name>1</name>
<value><struct>
```

```
<member><name>name</name>
<value><string>Good Friends</string></value></member>
<member><name>sort</name>
<value><int>25</int></value></member>
<member><name>public</name>
<value><boolean>1</boolean></value></member>
</struct></value></member>
</struct></value></member>
</struct></value>
</param></params>
</methodCall>
```

Возвращаемые значения

Функция не возвращает значения.

Пример ответа на запрос ***editfriendgroups***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
<params><param>
<value><struct>
</struct></value>
</param></params>
</methodResponse>
```

Возвращаемые ошибки

207 – ошибочная версия протокола ("Protocol version mismatch").

208 – недопустимая кодировка текста ("Invalid text encoding").

306 – журнал временно находится в режиме только для чтения. Попробуйте еще раз через пару минут ("This journal is temporarily in read-only mode. Try again in a couple minutes").

308 – аккаунт заблокирован и не может быть использован ("Account is locked and cannot be used").

2.5.5. Лента друзей

Лента друзей – это список последних записей друзей пользователя. Записи в ленте друзей идут в обратном хронологическом порядке.

2.5.5.1. Функция getfriendspage

Описание

Функция возвращает часть ленты друзей.

Параметры

Структура, содержащая следующие поля:

- **параметры аутентификации** (см. *п. 2.3.1*);
- **itemshow** (integer) – максимальное число возвращаемых записей (от 0 до 100). *Значение по умолчанию «100».*
- **skip** (integer) – номер записи (в текущем порядке), с которой выводится лента друзей (от 0 до 100). *Значение по умолчанию «0»*.
- **before** (string) – Unix-время. Необходимо выдать записи, созданные до указанного времени (время создания записи на сервере);
- **trim_widgets** (integer) – число символов, до которого нужно сократить запись (с учетом сжатия тегов и учета картинок);
- **lastsync** (string) – Unix-время. Необходимо выдать записи, созданные после указанного времени (время создания записи на сервере).
- **parseljtags** (boolean) – раскрывать (преобразовывать) встроенные LJ-теги в простую HTML форму. По умолчанию установлено значение *0* (ложь), при котором преобразование не осуществляется. При истинном значении (*1*) осуществляется преобразование тегов.
- **widgets_img_length** (integer) – число символов, занимаемых картинкой при сокращении записи.

Преобразование **LJ-тегов**:

- тег **lj-poll** заменяется на ссылку страницы с этим описанием:

```
<a  href="http://livejournal.com/poll/?id=<pollid>"  target="_blank">View Poll: <pll name>.</a>
```

- тег **lj-embed** заменяется на ссылку вида:

```
<a href="<embed url>">View movie.</a>
```

- тег **lj-user** заменяется на ссылку журнала пользователя:

```
<a href="<user_profile_url>" target="_blank"><img src="<userinfo.gif>" lt=""></a><a href="<journalbase_url>"><username></a>
```

Пример запроса **getfriendspage**:

```
<?xml version="1.0" encoding="UTF-8"?>

<methodCall>

<methodName>LJ.XMLRPC.getfriendspage</methodName>

<params><param>

<value><struct>

<member><name>auth_challenge</name>

<value><string>c0:1284566400:498:60:6VzIzyZnPTAnxSG7u1Po:a82d42c39db2ac0147bbd21896eb919c</string></value></member>

<member><name>ver</name>

<value><int>1</int></value></member>
```

```
<member><name>auth_response</name>
<value><string>73e3cb49b1e778d67fb14f80e53f5bd0</string></value></member>
<member><name>auth_method</name>
<value><string>challenge</string></value></member>
<member><name>itemshow</name>
<value><int>3</int></value></member>
<member><name>username</name>
<value><string>ljwebt40</string>
</value></member>
</struct></value>
</param></params>
</methodCall>
```

Возвращаемые значения

Структура, содержащая следующие поля:

- **skip** (integer) – число записей, пропущенных от начала ленты друзей. Соответствует значению входного поля ***skip***.
- **entries** (array) – массив структур, содержащих следующие поля:
  - **userpic** (struct) – описание картинки пользователя. Структура, содержащая следующие поля:
    - **picid** (integer) – идентификатор картинки;
    - **userid** (integer) – идентификатор пользователя;
  - **journaltype** (string) – тип журнала (см. *п.* ***2.5.1***);
  - **do_captcha** (boolean) – необходимость пройти Captcha[7] при создании комментариев («*1*» – необходимо пройти, «*0*» – не проходить);
  - **journalname** (string) – имя журнала;
  - **ditemid** (integer) – внешний идентификатор записи;
  - **postertype** (string) – тип пользователя (журнала) (см. *п.* ***2.5.1***), автора записи;
  - **postername** (string) – имя пользователя, автора записи;
  - **subject_raw** (string) – заголовок записи;
  - **event_raw** (string) – текст записи;
  - **security** (string) – тип доступа («*public*» – публичный, «*private*» – приватный, только для владельца журнала);
  - **logtime** (string) – фактическое Unix-время последнего изменения записи на сервере (серверное время);
  - **poster_userpic_url** (string) – ссылка на картинку пользователя (URL);
  - **journalurl** (string) – ссылка на журнал (URL);

---

[7] http://ru.wikipedia.org/wiki/CAPTCHA

- **posterurl** (string) – ссылка на журнал автора (URL);
- **reply_count** (integer) – число комментариев к записи;
- **props** (struct) – свойства записи. *См.* Приложение Б.

Пример ответа на запрос ***getfriendspage***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
<params><param>
<value><struct>
<member><name>skip</name>
<value><int>0</int></value></member>
<member><name>entries</name>
<value><array><data>
<value><struct>
<member><name>userpic</name>
<value><struct>
<member><name>picid</name>
<value><int>9277267</int></value></member>
<member><name>userid</name>
<value><int>484155</int></value></member>
</struct></value></member>
<member><name>journaltype</name>
<value><string>P</string></value></member>
<member><name>journalname</name>
<value><string>drugoi</string></value></member>
<member><name>ditemid</name>
<value><int>3362664</int></value></member>
<member><name>posterurl</name>
<value><string>http://drugoi.livejournal.com</string></value></member>
<member><name>subject_raw</name><value>
<string>Single photo</string></value></member>
<member><name>poster_userpic_url</name>
<value><string>http://l-
userpic.livejournal.com/9277267/484155</string></value></member>
<member><name>postername</name>
<value><string>drugoi</string></value></member>
<member><name>journalurl</name>
<value><string>http://drugoi.livejournal.com</string></value></member>
<member><name>logtime</name>
```

<value><int>1284562023</int></value></member>
<member><name>reply_count</name>
<value><int>109</int></value></member>
<member><name>do_captcha</name>
<value><int>0</int></value></member>
<member><name>postertype</name>
<value><string>P</string></value></member>
<member><name>props</name>
<value><struct>
<member><name>picture_keyword</name>
<value><string>troll</string></value></member>
<member><name>personifi_word_count</name>
<value><int>4</int></value></member>
<member><name>replycount</name>
<value><int>109</int></value></member>
<member><name>commentalter</name>
<value><int>1284566832</int></value></member>
<member><name>interface</name>
<value><string>web</string></value></member>
<member><name>personifi_lang</name>
<value><string>nil</string></value></member>
</struct></value></member>
<member><name>event_raw</name>
<value><base64>PGxqLWN1dCB0ZX0YDQvdC+0LPQviI+..L3RgdC60LUu</base64></value>
</member>
<member><name>security</name>
<value><string>public</string></value></member>
</struct></value>
<value><struct>
<member><name>userpic</name>
<value><struct>
<member><name>picid</name>
<value><int>94907468</int></value></member>
<member><name>userid</name>
<value><int>12548111</int></value></member>
</struct></value></member>
<member><name>journaltype</name>
<value><string>P</string></value></member>
<member><name>journalname</name>
<value><string>atereshko</string></value></member>

<member><name>ditemid</name>
<value><int>94672</int></value></member>
<member><name>posterurl</name>
<value><string>http://atereshko.livejournal.com</string></value></member>
<member><name>subject_raw</name>
<value><string /></value></member>
<member><name>poster_userpic_url</name>
<value><string>http://l-userpic.livejournal.com/94907468/12548111</string>
</value></member>
<member><name>postername</name>
<value><string>atereshko</string></value></member>
<member><name>journalurl</name>
<value><string>http://atereshko.livejournal.com</string></value></member>
<member><name>logtime</name>
<value><int>1284558659</int></value></member>
<member><name>reply_count</name>
<value><int>0</int></value></member>
<member><name>do_captcha</name>
<value><int>0</int></value></member>
<member><name>postertype</name>
<value><string>P</string></value></member>
<member><name>props</name>
<value><struct>
<member><name>personifi_word_count</name>
<value><int>3</int></value></member>
<member><name>replycount</name>
<value><int>0</int></value></member>
<member><name>taglist</name>
<value><base64>0YHRg9C/LCDRgNCw0LHQvtGH0LXQtQ==</base64></value></member>
<member><name>opt_preformatted</name>
<value><int>1</int></value></member>
<member><name>interface</name>
<value><string>web</string></value></member>
<member><name>personifi_lang</name>
<value><string>nil</string></value></member>
<member><name>used_rte</name>
<value><int>1</int></value></member>
</struct></value></member>
<member><name>event_raw</name>

<value><base64>0LLRi9C/0YPRgdGC0LjQu9C4INC9..stG0dHA6Ly9tLmxpdmVqb3VybmFsLmNvbSI+bS5saXZlam91cm5hbC5jb208L2E+</base64></value></member>

<member><name>security</name>

<value><string>public</string></value></member>

</struct></value>

<value><struct>

<member><name>userpic</name>

<value><struct>

<member><name>picid</name>

<value><int>68612100</int></value></member>

<member><name>userid</name>

<value><int>484155</int></value></member>

</struct></value></member>

<member><name>journaltype</name>

<value><string>P</string></value></member>

<member><name>journalname</name>

<value><string>drugoi</string></value></member>

<member><name>ditemid</name>

<value><int>3362481</int></value></member>

<member><name>posterurl</name>

<value><string>http://drugoi.livejournal.com</string></value></member>

<member><name>subject_raw</name>

<value><base64>0JjRgdGC0L7RgNC40Y8g0YHdC90LPQsNCz..DQv9C+0LTRgNC+0LHQvdC+0YHRgtC4</base64></value></member>

<member><name>poster_userpic_url</name>

<value><string>http://l-userpic.livejournal.com/68612100/484155</string>

</value></member>

<member><name>postername</name>

<value><string>drugoi</string></value></member>

<member><name>journalurl</name>

<value><string>http://drugoi.livejournal.com</string></value></member>

<member><name>logtime</name><value><int>1284558228</int></value></member>

<member><name>reply_count</name>

<value><int>108</int></value></member>

<member><name>do_captcha</name>

<value><int>0</int></value></member>

<member><name>postertype</name>

<value><string>P</string></value></member>

<member><name>props</name>

<value><struct>

```
<member><name>replycount</name>
<value><int>108</int></value></member>
<member><name>commentalter</name>
<value><int>1284566655</int></value></member>
<member><name>personifi_word_count</name>
<value><int>28</int></value></member>
<member><name>revnum</name>
<value><int>2</int></value></member>
<member><name>interface</name>
<value><string>web</string></value></member>
<member><name>personifi_lang</name>
<value><string>nil</string></value></member>
<member><name>hasscreened</name>
<value><int>1</int></value></member>
<member><name>revtime</name>
<value><int>1284558841</int></value></member>
</struct></value></member>
<member><name>event_raw</name>
<value><base64>PGltZwZyI+QpN..C/0LDQv0LrQLPQvi4=</base64></value></member>
<member><name>security</name>
<value><string>public</string></value></member>
</struct></value></data>
</array></value></member>
</struct></value>
</param></params>
</methodResponse>
```


Возвращаемые ошибки

209 – параметр вне диапазона ("Parameter out of range").

2.5.6. Комментарии

С помощью комментирования любая запись в журнале пользователя автоматически превращается в небольшой форум по обсуждению какой-то темы, в котором другие пользователи могут комментировать саму запись, отвечать на комментарии других пользователей, следить за появлением новых ответов к записи и к своим комментариям, и т.д. Для добавления комментария используется функция ***addcomment*** (см. *п.* ***2.5.6.2***). Получить комментарии можно вызвав функцию ***getrecentcomments*** (см. *п.* ***2.5.6.3***).

2.5.6.1. Параметры комментария

Комментарий состоит из следующих параметров.

Автор комментария — пользователь, добавивший комментарий.

Дата добавления комментария (*Date*) — время публикации комментария.

Тема комментария (*Subject*) — тема комментария. Пользователь может указать произвольную последовательность символов, включая некоторые HTML-теги, а также использовать здесь некоторые из LJ-тегов, например, **<lj-user>**. Тема комментария может быть пустой. Максимальная длина темы комментария составляет 100 символов. Поскольку символы кириллицы хранятся в кодировке Unicode, то максимальная длина темы, написанной кириллицей, меньше 100 символов.

Иконка темы комментария (*Subject icon*) — небольшая иконка, выражающая отношение автора комментария к записи, или его настроение. Имеет такой же смысл, как настроение пользователя для записи. Набор иконок не может быть изменен пользователем.

Сообщение (*Text*) — текст комментария. Может содержать произвольную последовательность символов, в том числе HTML-теги, разрешенные в LiveJournal и специальные теги. Максимальная длина сообщения составляет 4300 символов.

У каждого комментария имеются два идентификатора: ***jtalkid*** (внутренний) и ***dtalkid*** (внешний). Внешний идентификатор используется для формирования не последовательных числовых идентификаторов комментариев и получается из внутреннего, путем умножения его на 256 (сдвиг влево на 8 бит) и добавлением случайного числа от 0 до 255, созданного в момент добавления комментария.

2.5.6.2. Функция addcomment

Описание

Функция позволяет добавить комментарий к записи или другому комментарию.

Параметры

Структура, содержащая следующие поля:

- **параметры аутентификации** (см. *п. 2.3.1*);
- **body** (string) – текст комментария (не пустое значение). Допустимая длина ограничена настройками сервиса.
- **ditemid** (string) – внешний идентификатор записи, к которой добавляется комментарий;
- **parenttalkid** (string) – внутренний идентификатор родительского комментария;
- **parent** (string) – внешний идентификатор родительского комментария;
- **subject** (string) – заголовок комментария. *Значение по умолчанию – пустая строка*.
- **prop_picture_keyword** (string) – ключевое слово, по которому определяется картинка к комментарию. *Значение по умолчанию – картинка пользователя по умолчанию*.
- **journal** (string) – Пользователь, от чьего имени выполняется запрос к API (его имя и пароль указываются в параметрах аутентификации/авторизации вызова). Значение по умолчанию – журнал пользователя.

Пример запроса ***addcomment***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
<methodName>LJ.XMLRPC.addcomment</methodName>
<params>
```

```
<param>
<value><struct>
<member><name>body</name><value>
<base64>0JTQvtCx0LDQstC70LXQvdC40LUg0LrQvtC80LzQtdC90YLQsNGA0LjQtdCyINGH0LXRgNC10LcgWE1MLVJQQw==</base64></value></member>
<member><name>auth_challenge</name>
<value><string>c0:1284566400:1080:60:HCu8lP9SuFWPDtMcFNhy:7f3016eb8fb51c82d91299283b0a8c17</string></value></member>
<member><name>ver</name>
<value><int>1</int></value></member>
<member><name>auth_response</name>
<value><string>4c972dd3b5045247c02b6764544f97fa</string></value></member>
<member><name>auth_method</name>
<value><string>challenge</string></value></member>
<member><name>ditemid</name>
<value><int>297</int></value></member>
<member><name>username</name>
<value><string>test</string></value></member>
</struct></value>
</param></params>
</methodCall>
```

Возвращаемые значения

Структура, содержащая следующие поля:

- **status** (string) – «*OK*», всегда такой статус;
- **commentlink** (string) – ссылка на комментарий (URL);
- **dtalkid** (integer) – внешний идентификатор добавленного комментария;

Пример ответа на запрос ***addcomment***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
<params><param>
<value><struct>
<member><name>dtalkid</name>
<value><int>553</int></value></member>
<member><name>commentlink</name>
<value><string>http://test.livejournal.com/297.html?thread=553#t553</string></value></member>
<member><name>status</name>
<value><string>OK</string></value></member>
</struct></value>
```

```
</param></params>
</methodResponse>
```

Возвращаемые ошибки

204 – недопустимый тип метаданных ("Invalid metadata datatype").

214 – сообщение похоже на спам ("Message looks like spam").

314 – только платным пользователям разрешено использовать этот запрос ("Only paid users allowed to use this request").

2.5.6.3. Функция getrecentcomments

Описание

Функция возвращает комментарии из журнала пользователя в обратном хронологическом порядке.

Параметры

Структура, содержащая следующие поля:

- **параметры аутентификации** (см. *п. 2.3.1*);
- **itemshow** (integer) – количество возвращаемых комментариев. *Значение по умолчанию 10*.
- **skip** (integer) – число пропущенных комментариев. Параметр позволяет разбивать возвращаемые значения на страницы. Например, при значении равном «10», на первой странице отбражаются последние 10 комментариев к текущей записи, на второй – комментарии с 11-го по 20-й, и так далее. *Значение по умолчанию 0.*
- **trim_widgets** (integer) – число символов, до которого сокращается комментарий (с учетом сжатия тегов и учета картинок). *Значение по умолчанию 50.*
- **widgets_img_length** (integer) – число символов, которое занимает картинка в комментарии. *Значение по умолчанию 50.*
- **parseljtags** (boolean) – раскрывать (преобразовывать) встроенные LJ-теги в простую HTML форму. По умолчанию установлено значение *0* (ложь), при котором преобразование не осуществляется. При истинном значении (*1*) осуществляется преобразование тегов.

Преобразование **LJ-тегов**:

- тег **lj-poll** заменяется на ссылку страницы с этим описанием:

```
<a   href="http://livejournal.com/poll/?id=<pollid>"   target="_blank">View
Poll: <pll name>.</a>
```

- тег **lj-embed** заменяется на ссылку вида:

```
<a href="<embed url>">View movie.</a>
```

- тег **lj-user** заменяется на ссылку журнала пользователя:

```
<a href="<user_profile_url>" target="_blank"><img src="<userinfo.gif>"
lt=""></a><a href="<journalbase_url>"><username></a>
```

Пример запроса ***getrecentcomments***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
<methodName>LJ.XMLRPC.getrecentcomments</methodName>
<params><param>
<value><struct>
<member><name>auth_challenge</name>
<value><string>c0:1284566400:1271:60:wljHCEpyjOqWYLST5s4I:b922c78a774be93e3
800e86699115daf</string></value></member>
<member><name>ver</name>
<value><int>1</int></value></member>
<member><name>auth_response</name>
<value><string>793be3ebc8756914a42cecb25707f6b7</string></value></member>
<member><name>auth_method</name>
<value><string>challenge</string></value></member>
<member><name>username</name>
<value><string>test</string></value></member>
</struct></value>
</param></params>
</methodCall>
```

Возвращаемые значения

Структура, содержащая следующие поля:

- **status** (string) – поле статуса. Значение поля всегда «*OK*».
- **comments** (array) – массив, содержащий структуры со следующими полями:
  - **nodeid** (integer) – внутренний идентификатор записи, с которым связан комментарий;
  - **subject** (string) – тема комментария;
  - **posterid** (integer) – идентификатор автора комментария;
  - **state** (string) – состояние комментария («*F*» - заморожен, «*S*» - скрыт, «*A*» – опубликован (не заморожен, не скрыт, не удален), «*D*» - удален);
  - **jtalkid** (integer) – внутренний идентификатор комментария;
  - **parenttalkid** (integer) – внутренний идентификатор родительского комментария;
  - **postername** (string) – имя пользователя, автора комментария;
  - **text** (string) – текст комментария;
  - **nodetype** (string) – поле зарезервировано для будущего использования, в данный момент значение поля всегда равно «*L*»;
  - **datepostunix** (string) – Unix-время публикации комментария.

Пример ответа на запрос ***getrecentcomments***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
<params><param>
<value><struct>
<member><name>status</name>
<value><string>OK</string></value></member>
<member><name>comments</name>
<value><array><data>
<value><struct>
<member><name>subject</name>
<value><string>some subject</string></value></member>
<member><name>nodeid</name>
<value><int>1</int></value></member>
<member><name>posterid</name>
<value><int>2</int></value></member>
<member><name>state</name>
<value><string>A</string></value></member>
<member><name>jtalkid</name>
<value><int>1</int></value></member>
<member><name>parenttalkid</name>
<value><int>0</int></value></member>
<member><name>postername</name>
<value><string>test</string></value></member>
<member><name>text</name>
<value><string>some text</string></value></member>
<member><name>nodetype</name>
<value><string>L</string></value></member>
<member><name>datepostunix</name>
<value><int>1266408839</int></value></member>
</struct></value>
<value><struct>
<member><name>subject</name>
<value><string /></value></member>
<member><name>nodeid</name>
<value><int>1</int></value></member>
<member><name>posterid</name>
<value><int>2</int></value></member>
<member><name>state</name>
```

```
<value><string>A</string></value></member>
<member><name>jtalkid</name>
<value><int>2</int></value></member>
<member><name>parenttalkid</name>
<value><int>0</int></value></member>
<member><name>postername</name>
<value><string>test</string></value></member>
<member><name>text</name>
<value><base64>0JTQvtCx0LDQstC70LXQvdC40LUg0LrQvtC80LzQtdC90YLQsNGA0LjQtdCy
INGH0LXRgNC10LcgWE1MLVJQQw==</base64></value></member>
<member><name>nodetype</name>
<value><string>L</string></value></member>
<member><name>datepostunix</name>
<value><int>1284567481</int></value></member>
</struct></value>
</data></array></value></member>
</struct></value>
</param></params>
</methodResponse>
```

Возвращаемые ошибки

Функция не возвращает ошибок.

2.5.7. Сообщения

Центр сообщений – аналог электронного почтового ящика. Основное назначение которого – хранить уведомления, полученные пользователем. Владельцы постоянных, платных и улучшенных аккаунтов могут отправлять и получать текстовые сообщения, используя центр сообщений.

Все сообщения организованы по папкам:

- **All** (все) – все сообщения одним списком;
- **Messages** (сообщения) – только входящие сообщения от пользователей;
- **Friend Updates** (обновления списка друзей) – сообщения, связанные с изменениями в группе друзей. Когда кто-то удалил пользователя или добавил к себе в друзья.
    - **Birthdays** (дни рождения) – сообщения о днях рождениях;
    - **New Friends** (новые друзья) – сообщения о том, что кто-то добавил пользователя в свой список друзей;
- **Entries & Comments** (записи к комментариям) – сообщения об ответах на оставленные пользователем комментарии и сообщения о комментариях на записи в журнале пользователя.
- **Flagged** (закладки) – только те сообщения, которые отмечены пользователем флажком.

- **Sent** (отправленные) – отправленные пользователем сообщения.

Для получения сообщений из папки входящие используется функция ***getinbox*** (см. *п. 2.5.7.1*). Установка входящему сообщению статуса «прочитанное» осуществляется функцией ***setmessageread*** (см. *п. 2.5.7.2*). Для отправки сообщения другому пользователю используется функция ***sendmessage*** (см. *п. 2.5.7.3*).

2.5.7.1. Функция getinbox

Описание

Функция возвращает сообщения пользователя из папки входящие (messages).

Параметры

Структура, содержащая следующие поля:

- **параметры аутентификации** (см. *п. 2.3.1*)**;**
- **itemshow** (integer) – количество возвращаемых сообщений (от 0 до 100). *Значение по умолчанию 100.*
- **skip** (integer) – порядковый номер сообщения (от 0 до 100) в папке входящие, с которого начинается вывод (от 0 до 100). *Значение по умолчанию 0.*
- **lastsync** (string) – Unix-время, начиная с которого (включительно) необходимо получить сообщения;
- **before** (string) – Unix-время, до которого (включительно) необходимо получить сообщения;
- **extended** (boolean) – при истинном значении поля отображать расширенную информацию о сообщениях;
- **gettype** (array) – выбрать сообщения только указанных типов.

Типы сообщений:

1 – добавления в друзья (Befriended);

2 – день рождения (Birthday);

3 – приглашение в сообщество (CommunityInvite);

4 – вступление в сообщество одобрено (CommunityJoinApprove);

5 – вступление в сообщество отклонено (CommunityJoinReject);

6 – запрос на присоединение к сообществу (CommunityJoinRequest);

7 – удаление из друзей (Defriended);

8 – приглашенный друг присоединился (InvitedFriendJoins);

9 – новый комментарий в журнале (JournalNewComment);

10 – новая запись в журнале (JournalNewEntry);

11 – новая картинка пользователя (NewUserpic);

12 – новый виртуальный подарок (NewVGift);

13 – уведомление от сообщества news (OfficialPost);

14 – нотификационное сообщение о продаже постоянных аккаунтов (PermSale);

15 – голосование в опросе (PollVote);

16 – уведомление от сообщества ru_news (SupOfficialPost);

17 – пользователь удален (UserExpunged);

18 – получено пользовательское сообщение (UserMessageRecvd);

19 – отправлено пользовательское сообщение (UserMessageSent);

20 – пользователь отправил новый комментарий (UserNewComment);

21 – новая запись пользователя (UserNewEntry).

Пример запроса ***getinbox***:

```
<?xml version="1.0" encoding="UTF-8" ?>
<methodCall>
<methodName>LJ.XMLRPC.getinbox</methodName>
<params><param>
<value><struct>
<member><name>ver</name>
<value><i4>1</i4></value></member>
<member><name>auth_method</name>
<value><string><cookie></string></value></member>
<member><name>username</name>
<value><string>test</string></value></member>
</struct></value>
</param></params>
</methodCall>
```

Возвращаемые значения

Структура, содержащая следующие поля:

- **skip** (integer) – число пропущенных сообщений. Соответствует значению входного поля ***skip***.
- **journaltype** (string) – тип журнала (см. *п. **2.5.1***);
- **login** (string) – имя пользователя;
- **items** (array) – массив структур, содержащих следующие поля:
  - **qid** (integer) – идентификатор сообщения;
  - **when** (string) – Unix-время сообщения;
  - **state** (string) – состояние сообщения: «*N*» – не прочитано, «*R*» – прочитано;
  - **type** (integer) – тип сообщения;
  - **typename** (string) – тип сообщения в виде строки (если значение поля ***type***= «*0*»);

Дополнительные поля для отдельных типов сообщений.

При значении поля ***type*** = «*9*» (новый комментарий в журнале):

- **journal** (string) – имя пользователя, в журнале которого оставлен комментарий;
- **action** (string) – действия с комментарием (возможные значения): «*deleted*», «*comment_deleted*», «*edited*», «*new*»;
- **entry** (string) – ссылка на запись, к которой добавлен комментарий (URL);
- **comment** (string) – ссылка на комментарий (URL);
- **poster** (string) – имя пользователя, автора комментария;
- **subject** (string) – тема комментария;

При истинном значении поля ***extended*** при запросе, к структуре массива ***items*** добавляется структура, содержащая следующие поля:

- **subject_raw** (string) – заголовок комментария;
- **body** (string) – текст комментария;
- **dtalkid** (integer) – внешний идентификатор комментария;

При значении поля ***type*** = «*18*» (получено пользовательское сообщение):

- **from** (string) – имя пользователя, от которого получено сообщение;
- **picture** (string) – ссылка на картинку пользователя (URL);
- **subject** (string) – тема сообщения;
- **body** (string) – текст сообщения;
- **msgid** (integer) – идентификатор сообщения;
- **parent** (integer) – идентификатор родительского сообщения (если есть);

При значении поля ***type*** = 19 (отправлено пользовательское сообщение):

- **to** (string) – имя пользователя кому адресовано сообщение;
- **picture** (string) – ссылка на картинку пользователя (URL);
- **subject** (string) – тема сообщения;
- **body** (string) – текст сообщения;

Пример ответа на запрос ***getinbox***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
<params><param>
<value><struct>
<member><name>skip</name>
<value><int>0</int></value></member>
<member><name>journaltype</name>
<value><string>P</string></value></member>
<member><name>login</name>
<value><string>test</string></value></member>
<member><name>items</name>
```

```
<value><array><data>
<value><struct>
<member><name>qid</name>
<value><int>322</int></value></member>
<member><name>when</name>
<value><int>1283855875</int></value></member>
<member><name>type</name>
<value><int>1</int></value></member>
<member><name>state</name>
<value><string>N</string></value></member>
</struct></value>
<value><struct>
<member><name>qid</name>
<value><int>318</int></value></member>
<member><name>when</name>
<value><int>1283845692</int></value></member>
<member><name>type</name>
<value><int>11</int></value></member>
<member><name>state</name>
<value><string>N</string></value></member>
</struct></value>
<value><struct>
<member><name>qid</name>
<value><int>269</int></value></member>
<member><name>when</name>
<value><int>1282143014</int></value></member>
<member><name>type</name>
<value><int>1</int></value></member>
<member><name>state</name>
<value><string>N</string></value></member>
</struct></value>
</data></array></value></member>
</struct></value>
</param></params>
</methodResponse>
```

Возвращаемые ошибки

203 – недопустимый параметр ("Invalid argument(s)").

209 – параметр вне диапазона ("Parameter out of range").

500 – внутренняя ошибка сервера ("Internal server error");

2.5.7.2. Функция setmessageread

Описание

Функция устанавливает указанным входящим сообщениям статус «прочитанное».

Параметры

Структура, содержащая следующие поля:

- **параметры аутентификации** (см. *п. 2.3.1*);
- **qid** (array) – массив идентификаторов сообщений, которым будет установлен статус о прочтении.

Пример запроса ***setmeddageread***:

```
<?xml version="1.0" encoding="UTF-8" ?>
<methodCall>
<methodName>LJ.XMLRPC.setmessageread</methodName>
<params><param>
<value><struct>
<member><name>qid</name>
<value><array><data>
<value><i4>322</i4></value>
</data></array></value></member>
<member><name>ver</name>
<value><i4>1</i4></value></member>
<member><name>auth_method</name>
<value><string>cookie</string></value></member>
<member><name>username</name>
<value><string>test></string></value></member>
</struct></value>
</param></params>
</methodCall>
```

Возвращаемые значения

Структура, содержащая следующие поля:

- **result** (array) – массив, содержащий структуры со следующими полями:
  - **qid** (integer) – идентификатор сообщения;
  - **result** (string) – статус сообщения («*set read*»).

Пример ответа на запрос ***setmeddageread***:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<methodResponse>
<params><param>
<value>
<struct>
<member><name>result</name>
<value><array><data>
<value><struct>
<member><name>qid</name>
<value><int>322</int></value></member>
<member><name>result</name>
<value><string>set read</string></value></member>
</struct></value>
</data></array></value></member>
</struct></value>
</param></params>
</methodResponse>
```

Возвращаемые ошибки

Функция не возвращает ошибок.

2.5.7.3. Функция sendmessage

Описание

Функция для отправки сообщения другому пользователю.

Параметры

Структура, содержащая следующие поля:

- **параметры аутентификации** (см. *п. 2.3.1*);
- **subject** (string) – тема сообщения (значение поля не должно быть пустым);
- **body** (string) – текст сообщения (значение поля не должно быть пустым);
- **to** (array) – массив имен пользователей-адресатов;
- **parent** (integer) – идентификатор родительского сообщения;
- **usepic** (integer) – идентификатор картинки пользователя.

Пример запроса ***sendmessage***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
<methodName>LJ.XMLRPC.sendmessage</methodName>
<params><param>
<value><struct>
```

```
<member><name>parent</name>
<value><int>14535325</int></value></member>
<member><name>ver</name>
<value><int>1</int></value></member>
<member><name>auth_response</name>
<value><string>0c291157bc8da5cad9a705f73faa5d17</string></value></member>
<member><name>auth_method</name>
<value><string>challenge</string></value></member>
<member><name>username</name>
<value><string>ljwebt40</string></value></member>
<member><name>body</name>
<value><base64>0J3QvtCy0L7QtSDQvtGC0LLQtdGC0L3QvtC1INGB0L7QvtCx0YnQtdC90LjQ
tSE=</base64></value></member>
<member><name&g t;to</name>
<value><string>ljwebt17</string></value></member>
<member><name>auth_challenge</name><value>
<string>c0:1285066800:2164:60:nNeT1QYcGSoa228G46bV:4e13220eef6767389afe490e
89186db6</string></value></member>
</struct></value>
</param></params>
</methodCall>
```

Возвращаемые значения

Структура, содержащая следующие поля:

- **msgid** (array) – массив идентификаторов отправленных сообщений;
- **sent_count** (integer) – количество отправленных сообщений.

Пример ответа на запрос ***sendmessage***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
<params><param>
<value><struct>
<member><name>msgid</name>
<value><array><data>
<value><int>19570690</int></value>
</data>
</array></value></member>
<member><name>sent_count</name>
<value><int>1</int></value></member>
</struct></value>
```

```
</param></params>
</methodResponse>
```

Возвращаемые ошибки

200 – отсутствует необходимый параметр ("Missing required argument(s)").

203 – недопустимый параметр ("Invalid argument(s)").

212 – сообщение слишком длинное ("Message body is too long").

213 – пустое сообщение ("Message body is empty").

208 – недопустимая кодировка текста ("Invalid text encoding").

305 – действие запрещено, аккаунт заморожен ("Action forbidden; account is suspended").

2.5.8. Запуск команд

Сервер LiveJournal имеет текстовую административную консоль для ввода команд[8]. Интерфейс к административной консоли доступен по адресу http://www.livejournal.com/admin/console/. Для доступа к консоли с помощью XML-RPC используется функция ***consolecommand*** (см. *п.* ***2.5.8.1***).

2.5.8.1. Функция consolecommand

Описание

Функция выполняет запуск административных команд.

Параметры

Структура, содержащая следующие поля:

- **параметры аутентификации** (см. *п.* ***2.3.1***);
- **commands** (array) – массив, содержащий команды для отправки.

Пример запроса ***consolecommand***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
<methodName>LJ.XMLRPC.consolecommand</methodName>
<params><param>
<value><struct>
<member><name>username</name>
<value><string>test</string></value></member>
<member><name>password</name>
<value><string>test</string></value></member>
<member><name>ver</name>
<value><int>1</int></value></member>
<member><name>commands</name>
```

[8] список команд находится по адресу http://www.livejournal.com/admin/console/reference.bml

```
<value><array><data>
<value><string>help print</string></value>
</data></array></value></member>
</struct></value>
</param></params>
</methodCall>
```

Возвращаемые значения

Структура, содержащая следующие поля:

- **results** (array) – массив с результатами выполнения команды. Содержит структуру со следующими полями:
    - **success** (integer) – результат выполнения команды (0 или 1).
    - **output** (array) – массив, содержащий:
        - тип полученной строки сообщения («*error*» – ошибка, «*info*» – полезная информация, «» – обычное сообщение);
        - строка текста.

Пример ответа на запрос ***consolecommand***:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
<params><param>
<value><struct>
<member><name>results</name>
<value><array>
<data><value><struct>
<member><name>success</name>
<value><int>1</int></value></member>
<member><name>output</name>
<value><array><data>
<value><array><data>
<value><string></string></value>
<value><string>print ...</string></value>
</data></array></value>
<value><array><data>
<value><string></string></value>
<value><string>  This is a debugging function.  Given an arbitrary number
of</string></value>
</data></array></value>
<value><array><data>
<value><string></string></value>
```

<value><string> meaningless arguments, it'll print each one back to you. If an</string></value>

</data></array></value>

<value><array><data>

<value><string></string></value>

<value><string> argument begins with a bang (!) then it'll be printed to the error</string></value>

</data></array></value>

<value><array><data>

<value><string></string></value>

<value><string> stream instead.</string></value>

</data></array></value>

</data></array></value></member>

</struct></value>

</data></array></value></member>

</struct></value>

</param></params>

</methodResponse>

3. Примеры использования LiveJournal XML-RPC

3.1. Perl

Пример демонстрирует вызов функции ***getevents***:

```
#!/usr/bin/perl -w

use strict;
use XMLRPC::Lite;
use Digest::MD5;
use Data::Dumper;

my $proxy = "http://www.livejournal.com/interface/xmlrpc";
my $login = 'gariev';
my $password = 'test4test';

##
## get auth challenge
##
my $result = XMLRPC::Lite
    -> proxy($proxy)
    -> call('LJ.XMLRPC.getchallenge');
die $result->faultstring if $result->fault;

my $challenge = $result->result->{challenge};
my $digest_password = Digest::MD5::md5_hex(
    $challenge .
    Digest::MD5::md5_hex($password)
);

##
## get last posts (events)
##
$result = XMLRPC::Lite
    -> proxy($proxy)
    -> call('LJ.XMLRPC.getevents',
        {
            username        => $login,
            auth_method     => 'challenge',
            auth_challenge  => $challenge,
```

```
            auth_response   => $digest_password,
            ver             => 1,
            selecttype      => 'lastn',
            howmany         => 10,
        }
    );
die $result->faultstring if $result->fault;
print Dumper $result->result;
```

Пример демонстрирует аутентификацию с помощью cookie, получение входящих сообщений из почтового ящика и сброс полученной сессии:

```
#!/usr/bin/perl -w
use strict;
use Data::Dumper;
use XML::RPC;
use LWP::UserAgent;
use HTTP::Cookies;
use HTTP::Headers;

my $cookie_jar = HTTP::Cookies->new();

my $ua = LWP::UserAgent->new();
$ua->timeout(60);
$ua->env_proxy;
$ua->max_redirect(1);
$ua->cookie_jar($cookie_jar);

my $xmlrpc =
XML::RPC->new('http://www.livejournal.com/interface/xmlrpc',
lwp_useragent => $ua );

my $result = $xmlrpc->call('LJ.XMLRPC.sessiongenerate', { username =>
'test', password => 'test', ver => 1 });

$cookie_jar->set_cookie(0,'ljsession', $result->{ljsession}, '/',
'www.livejournal.com', undef, 1, '', 200000, '');

my $h = HTTP::Headers->new;
$h->header('X-LJ-Auth' => 'cookie');
$ua->default_headers($h);

$result = $xmlrpc->call('LJ.XMLRPC.getinbox', { username => 'test',
auth_method => 'cookie', ver => 1});
warn Dumper $result;

$result = $xmlrpc->call( 'LJ.XMLRPC.sessionexpire', {username => 'test',
auth_method => 'cookie', ver => 1});
```

```
$cookie_jar->save();
```

Приложение А. Перечень возвращаемых ошибок

Ошибки пользователя:

- **100** – недопустимое имя пользователя ("Invalid username");
- **101** – недопустимый пароль ("Invalid password");
- **102** – невозможно использовать приватные (подзамочные) записи в сообществах/новостных журналах ("Can't use custom/private security on shared/community journals");
- **103** – ошибка при создании голосования ("Poll error");
- **104** – ошибка добавления одного или нескольких друзей ("Error adding one or more friends");
- **105** – период действия значения challenge истек ("Challenge expired");
- **150** – невозможно делать записи в чужом журнале и/или невозможно делать записи незарегистрированному пользователю ("Can't post as non-user");
- **151** – пользователь заблокирован в данном журнале (сообществе) ("Banned from journal");
- **152** – невозможно делать записи с параметром «внеочередная дата» в сообществе ("Can't make back-dated entries in non-personal journal");
- **153** – неправильное значение времени ("Incorrect time value");
- **154** – невозможно добавить в друзья пользователя с данным именем, так как он переименовал свой журнал ("Can't add a redirected account as a friend");
- **155** – неподтвержденный email адрес ("Non-authenticated email address");
- **156** – пользователь не может авторизоваться в протоколе из-за того, что он не согласился с TOS.
- **157** – ошибка тегов ("Tags error").

Ошибки клиента:

- **200** – отсутствует необходимый параметр ("Missing required argument(s)");
- **201** – неизвестная функция ("Unknown method");
- **202** – слишком много параметров ("Too many arguments");
- **203** – недопустимый параметр ("Invalid argument(s)");
- **204** – недопустимый тип метаданных ("Invalid metadata datatype");
- **205** – неизвестные метаданные ("Unknown metadata");
- **206** – неправильное имя пользователя ("Invalid destination journal username");
- **207** – ошибочная версия протокола ("Protocol version mismatch");
- **208** – недопустимая кодировка текста ("Invalid text encoding");
- **209** – параметр вне диапазона ("Parameter out of range");

- **210** – Предупреждение. Пользователь пытается редактировать запись, содержащую поврежденные данные. ("Client tried to edit with corrupt data. Preventing");
- **211** – недопустимый или искаженный список тегов ("Invalid or malformed tag list");
- **212** – сообщение слишком длинное ("Message body is too long");
- **213** – пустое сообщение ("Message body is empty");
- **214** – сообщение похоже на спам ("Message looks like spam").

Ошибки доступа:

- **300** – отсутствует доступ к запрашиваемому журналу ("Don't have access to requested journal");
- **301** – доступ к заблокированной функции ("Access of restricted feature");
- **302** – невозможно редактировать запись из запрашиваемого журнала ("Can't edit post from requested journal");
- **303** – невозможно редактировать запись в сообществе ("Can't edit post in community journal");
- **304** – невозможно удалить запись в сообществе ("Can't delete post in this community journal");
- **305** – действие запрещено, аккаунт заморожен ("Action forbidden; account is suspended");
- **306** – журнал временно находится в режиме только для чтения. Попробуйте еще раз через пару минут ("This journal is temporarily in read-only mode. Try again in a couple minutes");
- **307** – выбранный журнал больше не существует ("Selected journal no longer exists");
- **308** – аккаунт заблокирован и не может быть использован ("Account is locked and cannot be used");
- **309** – статус журнала «memorial». Журнал опубликован, но писать в него нельзя ("Account is marked as a memorial");
- **310** – возраст владельца аккаунта нужно проверить, чтобы убедиться, что он не младше 18 лет ("Account needs to be age verified before use");
- **311** – доступ временно отключен ("Access temporarily disabled");
- **312** – не разрешается добавлять теги к записям в этом журнале ("Not allowed to add tags to entries in this journal");
- **313** – необходимо использовать существующие теги для записей в этом журнале (невозможно создать новые) ("Must use existing tags for entries in this journal (can't create new ones)");
- **314** – только платным пользователям разрешено использовать этот запрос ("Only paid users allowed to use this request");
- **315** – пользовательские сообщения отключены ("User messaging is currently disabled");
- **316** – пользователь находится в режиме «только чтение» и не может писать записи ("Poster is read-only and cannot post entries");

- **317** – журнал в режиме «только чтение» и записи не могут быть в нем опубликованы ("Journal is read-only and entries cannot be posted to it");
- **318** – пользователь находится в режиме «только чтение» и не может редактировать записи ("Poster is read-only and cannot edit entries");
- **319** – журнал в режиме «только чтение» и его записи не могут быть отредактированы ("Journal is read-only and its entries cannot be edited");
- **320** – приносим извинения. Была проблема с содержанием записи ("Sorry, there was a problem with content of the entry");
- **321** – приносим извинения. Удаление временно отключено. Запись сейчас в режиме «private» ("Sorry, deleting is temporary disabled. Entry is 'private' now").

Ошибки пределов:

- **402** – ваш IP адрес временно заблокирован за чрезмерное число попыток входа ("Your IP address is temporarily banned for exceeding the login failure rate");
- **404** – невозможно сделать запись ("Cannot post");
- **405** – достигнут предел частоты записей ("Post frequency limit");
- **406** – клиент посылает повторяющиеся запросы. Возможно, он сломан? ("Client is making repeated requests. Perhaps it's broken?");
- **407** – очередь на модерацию переполнена ("Moderation queue full");
- **408** – достигнуто максимальное число поставленных в очередь записей для данного сочетания «сообщество-автор» ("Maximum queued posts for this community+poster> combination reached");
- **409** – запись слишком большая ("Post too large");
- **410** – срок действия платного аккаунта истек. Невозможно опубликовать запись ("Your trial account has expired. Posting now disabled");
- **411** – достигнут предел частоты операций ("Action frequency limit").

Ошибки сервера:

- **500** – внутренняя ошибка сервера ("Internal server error");
- **501** – ошибка базы данных ("Database error");
- **502** – база данных временно недоступна ("Database temporarily unavailable");
- **503** – ошибка при получении необходимой блокировки ("Error obtaining necessary database lock");
- **504** – режим работы протокола больше не поддерживается ("Protocol mode no longer supported");
- **505** – формат данных аккаунта на сервере устарел и нуждается в обновлении ("Account data format on server is old and needs to be upgraded");
- **506** – синхронизация журнала временно недоступна (“Journal sync temporarily unavailable").

Приложение Б. Список свойств записей

Список свойств записей содержится в структуре с полями, представленными в таблице 2.

Таблица 2.

Поле	Название	Описание	Тип данных
admin_content_flag	Флаг содержания, установленный администратором (Admin Content Flag)	Внутренний флаг, описывающий свойство записи. Устанавливается администратором.	string
adult_content	Флаг непристойного содержания (Adult Content Flag)	Характеристики непристойного содержания (не содержит, не рекомендуется до 14 лет, явно содержит).	string
commentalter	Изменение комментариев (Comments altered)	Unix-время последнего изменения количества комментариев к этой записи.	string
current_coords	Текущие координаты (Current Coordinates)	Географические координаты автора при публикации (в форме '45.2935N 123.3452W').	string
current_location	Текущие местонахождение (Current Location)	Текущее местонахождение автора при публикации (текст в свободной форме).	string
current_mood	Текущее настроение (Current Mood)	Текущее настроение при публикации.	string
current_moodid	Идентификатор текущего настроения (Current Mood ID#)	Идентификатор текущего настроения	integer
current_music	Текущая музыка (Current Music)	Прослушиваемая музыка при публикации.	string
hasscreened	Скрытые комментарии (Has screened replies)	Истина, если у записи есть скрытые комментарии.	boolean
interface	Интерфейса обновления (Update interface)	Интерфейс, через который выполнялось последнее редактирование.	string
opt_backdated	«Внеочередная дата» (Back-dated)	Истинное значение, если эта запись не может быть отображена в ленте друзей.	boolean
opt_nocomments	Комментарии запрещены (Don't Allow Comments)	Истина, если читатели не могут написать комментарии к записи.	boolean
opt_noemail	Не отправлять комментарии по почте (Don't email comments)	Включить, если автор записи не заинтересован в получении комментариев на запись по электронной почте.	boolean
opt_preformatted	Автоматически не форматировать (Don't Auto-Format)	Включить, если запись содержит HTML теги и не должна быть отформатирована.	boolean
opt_screening	Настраиваемый уровень скрытия комментариев записи (Custom Screening Level)	Позволяет настроить уровень скрытия новых комментариев для данной записи. По умолчанию используется общий уровень, настроенный для всего журнала пользователя. Возможные значения	string

Поле	Название	Описание	Тип данных
		(скрывать комментарии для): N = никого, R = анонимных пользователей, F = не друзей, L = не друзей и если в комментарии есть ссылки, A = для всех пользователей.	
personifi_lang	Язык, определенный системой Personifi (Personifi language)	Автоматически определяется системой Personifi	string
personifi_tags	Теги категорий, определенные системой Personifi (Personifi tags)	Категории / тэги записи, которые вернула система Personifi Для данной записи.	string
personifi_word_count	Количество слов в записи по данным системы Personifi (Personifi word count)	Количество слов в записи, определенное системой Personifi	integer
picture_keyword	Ключевое слово картинки (Picture Keyword)	Ключевое слово картинки, которую ставит пользователь в качестве аватара при создании записи.	string
qotdid	Идентификатор вопроса дня (Writer's block ID)	Идентификатор вопроса дня (ответом на который является полученная запись).	integer
revnum	Число исправлений (Revision number)	Количество редакций записи.	integer
revtime	Время исправления (Revision time)	Unix-время последнего редактирования.	string
sms_msgid	Идентификатор SMS сообщения (SMS Message ID)	Идентификатор SMS сообщения, которое стало записью в журнале.	integer
statusvis	Статус видимости записи (Visibility Status of an Entry)	'V' или undef для отображаемых записей, 'S' для скрытых.	string
syn_id	Идентификатор транслируемого элемента (Syndicated item id)	Уникальный идентификатор транслируемого элемента.	string
syn_link	Ссылка на элемент трансляции (Syndication item link URL)	Исходная ссылка на запись трансляции.	string
taglist	Список тегов (Tag List)	Список тегов записи через запятую.	string
unknown8bit	Неизвестный 8-ми битный текст (Unknown 8-bit text)	Истина, если текст имеет 8-ми битные данные не в формате UTF-8.	boolean
unsuspend_supportid	Идентификатор запроса в службу поддержки/конфликтную комиссию по разблокированию записи (Support Request ID for Unsuspension Request)	Идентификатор запроса в службу поддержки / конфликтную комиссию по разблокированию данной записи, при ее блокировке. Undef или 0 если в данный момент нет открытого запроса.	integer
used_rte	Составлено в RTE (Composed in RTE)	Истина, если запись была составлена в rich text editor.	boolean
useragent	Пользовательский агент (User Agent)	Тип клиента (web/mobile/sip/etc), используемого для создания записи.	string

Поле	Название	Описание	Тип данных
verticals_list	Список тем (Verticals List)	Список (через запятую) вертикалей (тем), к которым относится данная запись.	string