<SUP>

LiveJournal XML-RPC Client/Server Protocol Reference

LiveJournal XML-RPC Specification

Contents

Introduction

The document covers functionality required for using LiveJournal XML-RPC protocol. The document is targeted at application developers.

1. XML-RPC Protocol

2.1.1. XML-RPC Definition

XML-RPC (abbreviation for *Extensible Markup Language Remote Procedure Call*) is a protocol for calling remote procedures (functions) through XML-messages. Specifications of XML-RPC protocol are available at http://www.xmlrpc.com/spec.

Remote Procedure Call is a technology that allows the requesting of functions or procedures from a remote machine.

2.1.2. Operation Principle

XML-RPC is a *Client-Server* interface that performs in the following way:

1. The *Client* encodes the name of a called function (mode) and its arguments in compliance with XML-RPC.
2. The *Client* sends a POST-request with an XML message to the *server* via HTTP protocol.
3. The *Server* obtains a request and exports data to the XML-RPC processor.
4. The *Server* XML-RPC processor decodes the received data to obtain the function name and its arguments.
5. The *Server* performs the function.
6. The function result is encoded in compliance with XML-RPC.
7. The *Server* sends the function result to the *Client*.

2.1.3. XML-RPC Support

Various platforms and programming languages support XML-RPC, in particular:

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

2.1.4. Data Types

XML-RPC protocol supports all data types specified in Table 1.

Table 1

Type	Description	Example
Integer	Round number	`<i4>23</i4>` Or

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

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

Type	Description	Example
		`<int>43</int>`
double	Floating point number	`<double>2.0</double>`
boolean	Logical value (0 or 1)	`<boolean>1</boolean>`
string	Character row	`<string>Hello, World!</string>`
Date/time	Date and time	`<dateTime.iso8601>19980717T14:08:55</dateTime.iso8601>`
base64	Encoded data in Base64 format	`<base64>SGVsbG8sIFdvcmxkIQ==</base64>`

array	Data array	`<array>`
		` <data>`
		`  <value> XML-RPC value </value>`
		`  ...`
		`  <value> XML-RPC value </value>`
		` </data>`
		`</array>`

Struct	Structure	`<struct>`
		`  <member>`
		`  <name>Field-1</name>`
		`  <value>Value-1</value>`
		`  </member>`
		`  ...`
		`  <member>`
		`  <name>Field-n</name>`
		`  <value>Value-n</value>`
		`  </member>`
		`</struct>`

2. LiveJournal XML-RPC

LiveJournal XML-RPC is based on the XML-RPC protocol. Figure 1 shows the operation principle. The client (user) sends an XML-RPC request to the server at http://www.livejournal.com/interface/xmlrpc. A request contains an XML message. The server receives the message, processes it, and performs the required function. Then the function result is returned to the client (XML-RPC response).

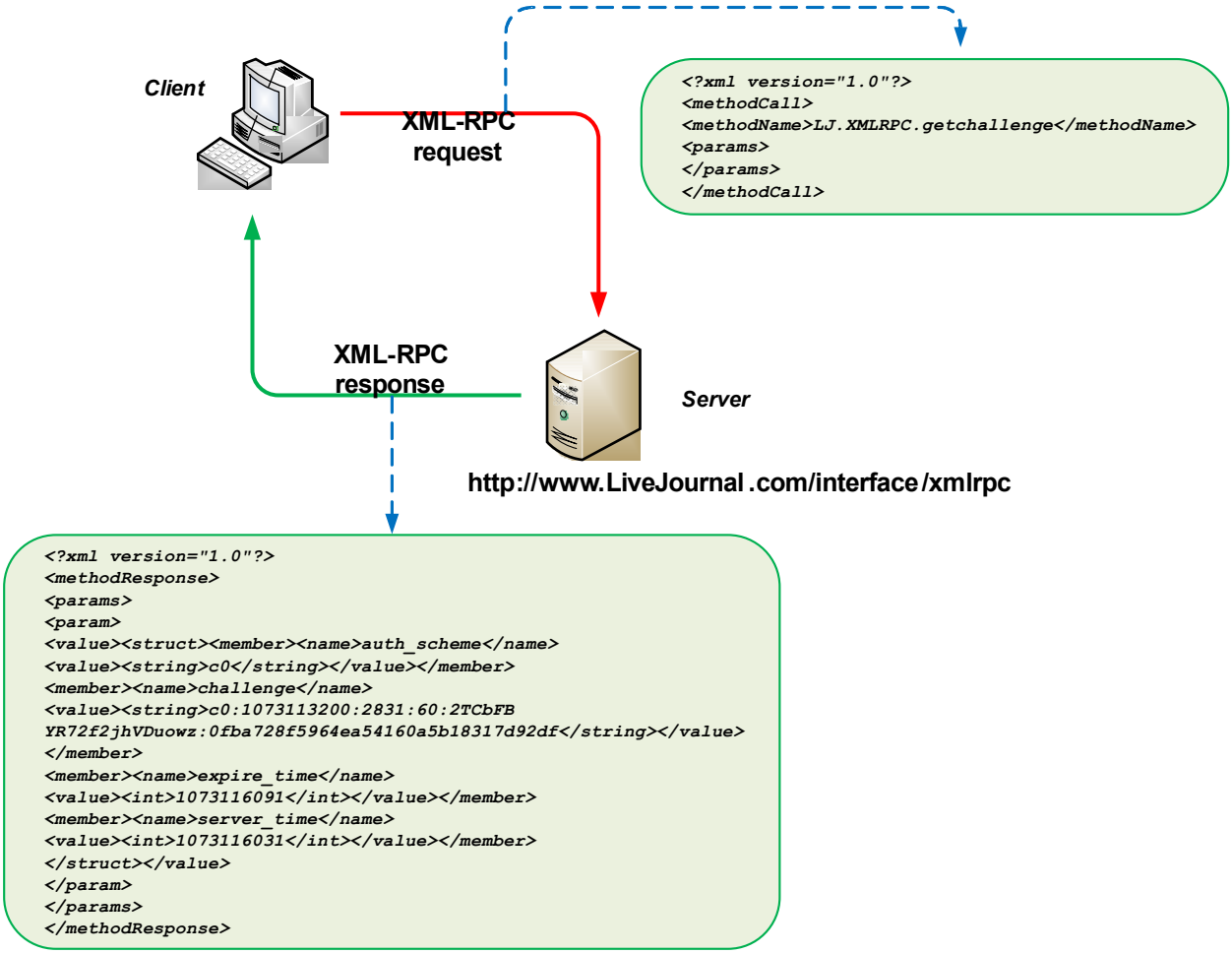

Fig.1 – Operation principle

Names of LiveJournal XML-RPC functions have the following format: ***LJ.XMLRPC.****function_name* (e.g. LJ.XMLRPC.postevent) where ***LJ.XMLRPC.*** is a constant element. The *function_name* variable contains one or several words (written continuously without breaks) and gives a short description of the purpose of the function. Only variable parts of function names (*function_name*) will be used in this document.

2.2.1. Request Structure

A request consists of two parts:

- HTTP headers;
- XML message that specifies the called function and its arguments.

HTTP headers contain:

- **User-Agent** – application name and version;

- **Host** – host domain name of the requested resource
- **Content-Type** – format and presentation model
- **Content-Length** – byte length of XML content

XML message is enclosed between *<methodCall>* and *</methodCall>* tags. The method call element contains the function name and its parameters. The former is enclosed between *<methodName>* and *</methodName>* tags. Parameters of LiveJournal XML-RPC functions are enclosed in *<params><param><value>…</params></param></value>* tags in a structured format. Parameter values comply with the structured parameters.

Sample XML-RPC request:

```
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>
```

2.2.2. Response Structure

Upon the receipt of an XML-RPC request, the server responds to a client. A response contains:

- HTTP headers
- XML message with the function result

There are two types of XML-messages:

- Function result
- Error message

The function result contains returned values and is enclosed in *<methodResponse>… </methodResponse>* tags. The returned values are enclosed in *<params><param><value>..</params></param></value>* tags in a structured form. Names of the structure parameters indicate names of the returned values.

Sample response:

```
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>
```

In case of an error, the server returns an error message enclosed in `<methodResponse>…</methodResponse>` tags. Error parameters are enclosed in `<fault><value>..</value></fault>` tags in a structured form containing the following fields:

- **faultString** (string)[3] – error description
- **faultCode** (integer) – error code

Refer to Appendix A for the list of errors returned at request of LiveJournal XML-RPC functions.

Sample error report:

```
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)

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

[3] The field data type is specified in parenthesis both here and later in this document.

```
<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>
```

2.2.3. Authentication

Each LiveJournal XML-RPC function (except for ***getchallenge***) must contain authentication settings. Alongside the rest of function parameters, authentication settings are transferred as structure parameters.

Authentication is the process of checking the user ID for authenticity.

2.2.3.1. Authentication Methods

There are three user authentication methods:

- **Clear method (i.e., login and password are sent as plain text, a/k/a "in the clear")**
- **Challenge-response method**
- **Through cookies**

Below each method is covered separately.

2.2.3.2. Clear Authentication Method

When this method is used, the client exports user credentials (login and password or its MD5 hash) to the server without encryption. This method is the simplest, though the least secure. This is the default option.

Warning! This method is **NOT** recommended!

2.3.2.3.2.1. Settings

Structure parameters:

- **username**[4] (string) – username
- **auth_method** (string) – authentication method («*clear*»)
- **password** (string) – user password; exported in plain text. Note that the ***hpassword*** field is not used for configuration of this setting
- **hpassword** (string) – MD5 hash of the user password. The ***password*** field is left empty if this one is filled
- **ver** (integer) – protocol version (always 1)

[4] Underlined field names indicate mandatory fields. Field names that are not underlined are optional fields for function calls.

2.2.3.3. Challenge-Response Method

When the **challenge-response** method is used, the password is not transferred through a communication channel.

Operation principle of the challenge-response authentication method:

- To perform authentication, a user creates a request. In response, the server returns a random value (challenge)
- The user then adds their password to the returned value (challenge) and the client computes the MD5 hash that is then sent to the server
- In the meantime, the server performs the same operations with the challenge and compares the result with the MD5 hash provided by the client. If hash values match, authentication is considered successful

2.3.2.3.3.1. Settings

Structure parameters:

- **username** (string) – username
- **auth_method** (string) – authentication method (“*challenge*”)
- **auth_challenge** (string) – challenge value returned by the server
- **auth_response** (string) – client response value
- **ver** (integer) – protocol version used (always set to “*1*”)

Value of the *auth_response* field is calculated according to the following formula:

$$MD5\_hex(challenge + MD5\_hex(password))$$

Where:

”+” – string merging operation; the server returns the *challenge* value in response to a ***getchallenge*** function request *(*refer to item ***3.13.1****)*

2.3.2.3.3.2. getchallenge Function

Description

The function requests the server to generate a value to be used in the challenge-response authentication method.

Arguments

No settings required for this function.

Sample ***getchallenge*** request:

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

Return Values

The structure contains the following keys:

- **auth_scheme** (string) – authentication scheme identifier

- **challenge** (string) – a opaque cookie used to generate a hashed response (challenge)
- **expire_time** (integer) – expiration Unix-time of the *challenge* value
- **server_time** (integer) – Unix-time of a *challenge* value generation at the server

Note! The validity of a generated challenge value is limited to 60 seconds!

Sample response to a ***getchallenge*** request:

```
<?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>
```

2.2.3.4. Cookie Authentication Method

Cookie is a small data chunk generated at the server. The client sends it to the server at each session in an HTTP-header.

Cookies returned by the LiveJournal server are used for authentication of further requests to XML-RPC.

2.3.2.3.4.1. Settings

Structure keys:

- **username** (string) – username
- **auth_method** (string) – authentication method (“*cookie*”)
- **ver** (integer) – protocol version (always set to “1”).

Each HTTP request to LiveJournal XML-RPC functions must contain the *ljsession* cookie equal to *ljmastersession* provided by the LiveJournal server (in case of web-authorization) or to the value in the *ljsession* key returned by the ***sessiongenerate*** function (refer to item ***2.3.4.2***). In addition, the request must send the *X-LJ-Auth: cookie* HTTP-header.

Sample cookie authentication:

```
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
```

2.3.2.3.4.2. sessiongenerate Function

Description

The function generates a session and returns its ID (cookies).

Arguments

Structure with the following components:

- **Authentication settings**[5] (also refer to item ***2.3.1***)
- **Expiration** (string) – session validity period. Available options:
  - *short* – 24 hours
  - *long* – 720 hours (30 days)
- **bindtoip** (boolean) – when this is true, the server generates a session only for the IP-address that requested the ***sessiongenerate*** function. By default, the session is available to any IP-address

[5]The clear or challenge-response authentication method is used to call this function

Sample **sessiongenerate** request:

```
<?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>
```

Return values

Structure with the following components:

- **ljsession** (string) – ID of a generated session

Sample ***sessiongenerate*** response:

```
<?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>
```

Returned errors

308 – Account is locked and cannot be used

2.3.2.3.4.3. sessionexpire Function

Description

Expires all the sessions generated by the ***sessiongenerate*** function of LiveJournal web-interface.

Arguments

Structure with the following components:

- **authentication settings** (refer to item ***2.3.1***)
- **expireall** (integer) – expiration of user sessions. When the key is set to 1, all user sessions are expired
- **expire** (array) – array of session ID's to be expired

Sample ***sessionexpire*** request:

```
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>
```

Return Values

Blank structure

Sample response to a ***sessionexpire*** request:

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

Return errors

502 –Database is temporarily unavailable

2.2.4. User Profile

Apart from registration data, each registered LiveJournal user owning an account can enter their contact information, personal data (i.e., gender, age, list of interests, list of schools they attended, etc.). All this data is stored in a **user profile**.

To access to a user profile, the ***login*** function is required (refer to item ***2.4.1***).

2.2.4.1. login Function

Description

The function gets information from a user profile: user name, list of friend groups, other data.

Arguments

Structure with the following components:

- **Authentication settings** (refer to item ***2.3.1***)
- **clientversion** (string) – allows identifying the version of a client that requests data. Allows the LiveJournal server to collect user statistics. The following string formats are supported `Platform-ProductName/ClientVersionMajor.Minor.Rev`, e.g., `Win32-MFC/1.2.7` or `GTK2-LogJam: 4.5.3`
- **getmoods** (integer) – to obtain the whole mood list, set the value to 0. When the value is different (above 0), the server returns the list of moods having ID's above the defined value
- **getmenus** (integer) – when true (*1*), the server returns a list (tree) of navigations to the web-menu
- **getpickws** (integer) – to obtain the list of userpic keywords, set the value to *1*
- **getpickwurls** (integer) – to obtain the list of links to userpics (URL) as well as userpic keywords, set the key value to *1*
- **getcaps** (integer) – to get all capability classes of the account, set the value to 1

Sample ***login*** request:

```
<?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>
```

Return Values

Structure with the following components:

- **fullname** (string) – username
- **message** (string) – message that notifies a user about software updates, problems with their account, etc.
- **friendgroups** (array) – array defining a user group
- **usejournals** (array) – array of journals/communities available to a user for posting
- **moods** (array) – mood array
- **pickws** (array) – array containing userpic keywords. The array is returned when ***getpickws*** = 1
- **pickwurls** (array) – array containing links (URL) to usepics
- **defaultpicurl** (string) – link (URL) to a default userpic. The value is returned when ***getpickwurls*** = 1

- **fastserver** (boolean) – when true, a client can enter "`Cookie: ljfastserver=1`" in HTTP-header for priority request processing
- **userid** (integer) – user identifier
- **menus** (array) – array containing menu items in the order of display in the corresponding web-menu of the LiveJournal user application. The array is returned when the ***getmenus*** structure component is set to 1
- **caps** (integer) – user account capability class defined by bit-mapped fields within a 2-byte integer. The key is returned when the ***getcaps*** structure component is set to 1

User Group Description

Structure with the following components:

- **public** (boolean) – flag indicating whether a group is public or private
- **name** (string) – friends group name
- **id** (integer) – bit number for this friends group (from 1 to 30)
- **sortorder** (integer) – number for ordering groups (from 1 to 255)

Mood Definition

Structure with the following components:

- **parent** (integer) – identifies the "parent" mood (i.e., you want to create a mood tree containing various themes for happiness, for example)
- **name** (string) – mood name
- **id** (integer) – mood identifier

Menu Item Definition

Structure with the following components:

- **text** (string) – menu item text or "-" for separator
- **url** (string) – link (URL) to a menu item. Defined for all menu items except for separators and sub-menus
- **sub** (array) – structure array containing sub-menu items. The key is returned when an item is a submenu. The submenu structure has the same format as that of the top-level menu

Account capability class Combination of bit-mapped keys:

- **0x01** – new user
- **0x02** – normal user
- **0x04** – early adopter (registered before September, 2000)
- **0x08** – paid user
- **0x10** – permanent account
- **0x20** – many friends class (can add up to 10,000 friends)

- **0x40** – move in progress (user account locked for maintenance, available as read-only)
- **0x80** – sponsored accounts
- **0x100** – beta-testing
- **0x200** – extra userpics
- **0x400** – Russian-speaking SUP-flagged users who receive some services from SUP
- **0x800** – SUP opt-out (Russian speakers who asked 6A to opt out from SUP)
- **0x1000** – temporary unlimited phone posts (available in some countries)
- **0x2000** – undergrade class (user age below 14 years)
- **0x4000** – ad supporter (testing); ads added to the user's journal though not visible to them
- **0x8000** – Plus (enhanced account)

Sample ***login*** response:

```
<?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 jeopardize the security of your journal. Please see
http://www.livejournal.com/support/faqbrowse.bml?faqid=71 (this may change)
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+0LvRjNC30L7QstCw0YLQtdC7
0Yw=</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>
```

Return Errors

207 – Protocol version mismatch

208 – Invalid text encoding

308 – Account is locked and cannot be used

502 – Database is temporarily unavailable

2.2.5. Journal

User journal is the user's personal journal or blog.

2.2.5.1. Journal Type

There several available journal types (journal properties depend on the **journal type**):

- User journal (**P**) – a journal of a separate user ("personal"). Only the journal owner can post in it
- Community (**C**). Unlike a user journal, a community journal ("community") allows multiple community members to post in it. Apart from the community creator, a community has moderators forming a particular user group that monitors the journal content. Moderators can edit and remove posts from other users and can modify member rights
- News wire (**N** – "news") — a news journal. There is a single instance of this type and it is designed to mail news to users
- Shared journal (**S** – "shared") — original version of communities. An outdated type; currently not used

- Syndicated journals (RSS) (**Y** – “syndicated”) — journal for collecting RSS-feeds. This type does not allow commenting or user-generated posts
- Renamed journal (**R**) — journal of a user who changed their login. Login is not a unique user account identifier in LiveJournal, so the user can change it while keeping their account groups and settings
- OpenID journal (**I** – “identity”) — users having an account at a different OpenID server, FaceBook or Twitter and using it to access the system

2.2.5.2. Entry

Each LiveJournal journal represents a set of **entries** (also called **posts**) created by the journal owner or other users.

Entry-management functions:

- ***postevent*** (refer to item ***2.5.3.1***) – used to create entries
- ***getevents*** (refer to item **2.5.3.2**) – used to get entries
- ***editevent*** (refer to item ***2.5.3.3***) – used to edit entries
- ***syncitems*** *(*refer to item ***2.5.3.4****)* – used to obtain the list of all entries created or edited in the journal after the specified time
- ***getdaycounts*** (refer to item ***2.5.3.5***) – used to obtain the number of entries created each day

2.2.5.3. Entry Settings

An entry has the following settings:

Date — the date of entry publication in the journal

Subject — Entry subject. A user can enter any character string, including HTML-tags and lj-tags (e.g., `<lj-user>`). Also, the line can be left empty

Text — A user can enter any text with HTML-tags and special LiveJournal tags like `<lj-user>`, `<lj-cut>`, `<lj-embed>`, etc.

Tags — entry tags

Mood — user mood is displayed in their journal beside the entry

Location — current user location; free-text input

Music — this displays the music the user is listening to at the moment; free-text input

Two types of identifiers – internal (***itemid***) and external (***ditemid***) are used to catalogue journal entries. The internal entry identifier (***itemid***) is not used to access an entry externally. The external entry identifier (***ditemid***) has the following structure:

```
ditemid = itemid * 256 + anum
```

where ***anum*** — is a random integer in the range between 0 and 255, inclusively

Tags

Tags are words or word combinations designed to build journal taxonomy. Tags simplify search and navigation in the journal for its owner and guests. Thus, a user can tag entries covering the same event with a relevant event name.

To get the list of tags, the ***getusertags*** function is used (refer to item ***2.5.3.6***).

2.3.2.5.3.1. postevent Function

Description

This function allows creating a new journal entry.

Arguments

Structure with the following components:

- **authentication settings** (refer to item ***2.3.1***)
- **event** (string) – entry text
- **lineendings** (string) – symbols used to separate lines within an entry:
  - *unix* - 0x0A (\n) for Unix
  - *pc* - 0x0D0A (\r\n) for PC (default)
  - *mac* - 0x0D (\r) for MAC
- **subject** (string) – entry subject
- **security** (string) – entry security level. Available options are:
  - *public* – the default option; allows public access to an entry
  - *private* – available to the journal owner only
  - *usemask*– entry available to friends of a journal owner. Availability is defined by the value of the ***allowmask*** structure component.
- **allowmask** (integer) – bit mask defines user groups that will have access to a published entry. A 32-bit unsigned integer that defines the post visibility to user friends and groups, as follows:
  - 0 bit stands for all user friends (also used for member only post in community)
  - bits in the range from 1 to 30 stand for user groups
  - 31st bit is reserved

  The **allowmask** structure component is used when ***security*****=** *«usemask*».
- **year** (integer) – current year (according to user time zone)
- **mon** (integer) – current month (from 1 to 12; according to user time zone)
- **day** (integer) – current day (from 1 to 31; according to user time zone)
- **hour** (integer) – current hour (from 0 to 23, according user time zone)
- **min** (integer) – current minute (from 0 to 59; according to user time zone)
- **tz** (string) – current user time zone. When ***year, mon, day, hour, min*** are not defined, the current server time is used with the relevant time zone added. When the "*guess*" option is selected, the user time zone is defined according to previous posts
- **props** (struct) – entry properties. Refer to Appendix B
- **usejournal** (string) – journal (user, community) name for posting

Sample ***postevent*** request:

```
<?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>
```

Return Values

Structure with the following components:

- **itemid** (integer) – internal entry identifier
- **anum** (integer) – random number from 0 to 255, generated when the entry is created; used as the lower byte in the external entry identifier
- **url** (string) – link to the entry (URL)
- **warnings** (string) – system notification/warning about errors at entry publication. Errors are caused by missed or unclosed tags

Sample response to a ***postevent*** request:

```
<?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>
```

Return Errors

102 –Can't use custom/private security in communities

103 –Poll error

150 –Can't post as a non-user

151 –Banned from journal

152 – Can't post back-dated entries in a community

153 –Incorrect time value

155 –Non-validated email address

200 –Missing required argument(s)

203 – Invalid argument(s)

301 – Access to restricted feature

305 – Action forbidden; account is suspended

306 –This journal is temporarily in read-only mode. Try again in a couple minutes

307 –Selected journal no longer exists

308 –Account is locked and cannot be used

309 –Account is marked as a memorial (the journal is locked and cannot be edited)

310 –Account user must be be age-verified before use

312 –Not allowed to add tags to entries in this journal

316 –Poster is read-only and cannot post entries

317 –Journal is read-only and entries cannot be posted to it

404 –Cannot post

405 –Post frequency limit

407 –Moderation queue full

408 – Maximum queued posts for this <community+poster> combination reached

409 – Post is too large

410 – Your trial account has expired. Posting is disabled

501 –Database error

503 –Error obtaining necessary database lock

2.3.2.5.3.2. getevents Function

Description

The function returns entries from the user journal.

Arguments

Structure with the following components:

- **authentication settings** (refer to item ***2.3.1***)
- **truncate** (integer) – defines the length of return values. Return values are truncated to the specified number of characters after decryption
- **prefersubject** (boolean) – when set to true, the ***event*** key returns an entry subject instead of its text, and the ***subject*** key remains empty
- **noprops** (boolean) – when set to true, entry properties are not returned
- **notags** (boolean) – when set to true, entry tags are not returned
- **selecttype** (string) – allows choosing the entry retrieval mode for the selected journal. Available options are as follows:
  - **one** – to retrieve a particular entry
  - **day** – to retrieve entries for the indicated year, month, or day. The number of returned entries is limited to 200
  - **lastn** – to retrieve the indicated number of recent entries; the number is specified in the ***howmany*** key)
  - **syncitems** – to retrieve entries created/changed after the date specified in the ***lastsync*** key

    Note: It is the server that determines the number of entries returned in this entry retrieval mode. So, make sure you use the *Syncitems protocol mode* to avoid incomplete retrieval.
  - **multiple** – to retrieve required entries with ID's specified in the ***itemids*** array (see below); the number of retrieved entries is limited to 100
  - **before** – to retrieve entries created before the date specified in the ***before*** key (see below)
- **lastsync** (string) – entries created after the specified time (in "*yyyy-mm-dd hh:mm:ss*" format, livejournal.com server event creation time, GMT) . Used when ***selecttype*** = «*syncitems*» (see above)
- **year** (integer) – year; used when ***selecttype*** = «*day*»
- **month** (integer) – month (from 1 to 12); used when ***selecttype*** = «*day*»
- **day** (integer) – day (from 1 to 31); used when ***selecttype*** = «*day*»

- **howmany/itemshow** (integer) – number of returned entries (from 0 to 50). The default is set to *20*. The key is used when ***selecttype*** = «*lastn*»
- **skip** (integer) – number of skipped entries (you can define a value in the range from *0* to *500*)
- **before** (string) –date specified in "*yyyy-mm-dd hh:mm:ss*" format; entries created before the specified date (livejournal.com server event creation time, GMT) are retrieved; it must explicit specify ***howmany*** parameter value; the key is used when ***selecttype*** = «*before*»
- **beforedate** (string) – date in "*yyyy-mm-dd hh:mm:ss*" format; indicates the last date of a period for which LJ entries are retrieved. Used when ***selecttype*** = «*lastn*»
- **itemid** (integer) – internal entry identifier; -1 value used to retrieve the last post in the journal; used when ***selecttype*** =*one*
- **itemids** (string) –internal identifiers of entries splitted by commas that must be retrieved; used when ***selecttype*** = «*multiple*»
- **lineendings** (string) – symbols used to separate lines within an entry. Available options are:
    - *unix* - 0x0A (\n) for Unix
    - *pc* - 0x0D0A (\r\n) for PC (default)
    - *mac* - 0x0D (\r) for MAC
        - *space* – space used as a line break
        - *dots* – dots used as a line break
- **usejournal** (string) – name of an available journal from which entries need to be retrieved
- **trim_widgets** (integer) – number of characters by which each entry is reduced. Reduction unit is a word (including tags and pics)
- **widgets_img_length** (integer) – number of characters an image within in an entry equals. The default is set to *50*, i.e., an entry containing no text and two images is considered to contain 100 characters; the system then reduces the entry length to one link to one image; used when ***trim_widgets*** parameter specified.
- **parseljtags** (boolean) – converts LJ-tags to plain HTML. By default set to false (*0*); when set to true (*1*), the system converts LJ-tags to plain HTML

LJ-tag conversion

- **lj-poll** tag is replaced by a link with the following description:

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

- **lj-embed** tag is replaced by a link of the following type:

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

- **lj-user** tag is replaced by a link to the user journal:

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

Sample ***getevents*** request:

```
<?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>
```

Return Values

Structure with the following components:

- **skip** (integer) – number of skipped entries. Corresponds to the value in the ***skip*** input key
- **events** (array) – array containing structures with the following keys:
    - **itemid** (integer) – entry internal identifier
    - **subject** (string) – entry subject
    - **event** (string) – entry body
    - **eventtime** (string) – entry publication time (in "*yyyy-mm-dd hh:mm:ss"* format);
    - **props** (struct) – entry properties. Refer to Appendix B
    - **url** (string) – link (URL) to an entry
    - **anum** (integer) – a random number from 0 to 255 generated at entry creation; used as the lower byte in the entry external identifier

- **event_timestamp** (string) –user-defined entry in Unix-time
- **reply_count** (integer) – number of replies to an entry
- **security** (string) – entry security type. When "*usemask"* is selected, the security type is defined by value specified in the ***allowmask*** key
    - **allowmask** (integer) – bit mask defining user groups that will have access to a posted entry. A 32-bit unsigned integer that defines the post visibility to user friends and groups. 0 bit stands for all user friends, bits ranging from 1 to 30 stand for user groups, the 31st bit is reserved. This structure member is used when ***security*** = *«usemask*».

Sample ***getevents*** response:

```
<?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 it's true it's ...</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>
```

Return errors

200 – Missing required argument(s)

203 – Invalid argument(s)

207 – Protocol version mismatch

208 – Invalid text encoding

209 – Parameter out of range

307 – Selected journal no longer exists

311 – Access temporarily disabled

406 – Client is making repeated requests. Perhaps it's broken?

501 – Database error

502 – Database is temporarily unavailable

506 – Journal sync is temporarily unavailable

2.3.2.5.3.3. editevent Function

Description

The function allows the editing of a previously created entry.

Arguments

Structure with the following components:

- **authentication settings** (refer to item ***2.3.1***)
- **itemid** (integer) – internal entry identifier
- **event** (string) – entry body. To delete en entry, just leave the key empty. Carriage returns are okay (`0x0A`, `0x0A0D`, or `0x0D0A`), although `0x0D` are removed internally to make line-endings comply with Unix-style (just *\n*'s) (see below). Entries can also contain HTML tags, though the LiveJournal server converts line breaks into
 tags, so make sure your client does not enter them
- **lineendings** (string) – symbols used as line breaks within an entry:
  - *unix* - 0x0A (\n) – for Unix
  - *pc* - 0x0D0A (\r\n) – for PC (default)
  - *mac* - 0x0D (\r) – for Mac
- **subject** (string) – entry subject
- **security** (string) – entry security level. Available options are:
  - *public* – the default option; allows public access to an entry
  - *private* – entry is available to the journal owner only
  - *usemask*– entry visible to friends or to particular friend groups. Access is defined by the ***allowmask*** value

- **allowmask** (integer) – bit mask defining user groups that will have access to a posted entry. A 32-bit unsigned integer that defines the post visibility to user friends and groups. 0 bit stands for all user friends, bits ranging from 1 to 30 stand for user groups, the 31st bit is reserved. The parameter is used when ***security*****=** *«usemask»*
- **year** (integer) – publication year
- **mon** (integer) – publication month (from 1 to 12)
- **day** (integer) – publication day (from 1 to 31)
- **hour** (integer) – publication time, hours (from 0 to 23)
- **min** (integer) – publication time, minutes (from 0 to 59)
- **props** (string) – entry properties. Refer to Appendix B
- **usejournal** (string) – journal (user) name for entry publication

Sample ***editevent*** request:

```
<?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>
```

Return Values

Structure with the following components:

- **itemid** (integer) – internal entry identifier
- **anum** (integer) – random number from 0 to 255, generated when the entry is created; used as the lower byte in the external entry identifier
- **url** (string) – link to the entry (URL)
- **warnings** (string) – system notification/warning about errors at entry posting. Errors are caused by missed or unclosed tags

Sample ***editevent*** response:

```
<?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>
```

Return errors

102 –Can't use custom/private security in shared communities

152 –Can't post back-dated entries in non-personal journal (define non-personal)

203 – Invalid argument(s)

210 – Client tried to edit with corrupt data. Preventing

302 –Can't edit post from requested journal

303 –Can't edit post in this community

304 –Can't delete post in this community

306 –This journal is temporarily in read-only mode. Try again in a couple minutes

307 –Selected journal no longer exists

310 – Account user needs to be age-verified before use

318 –Poster is read-only and cannot edit entries

319 –Journal is read-only and its entries cannot be edited

409 –Post is too large

501 –Database error

2.3.2.5.3.4. syncitems Function

Description

The function returns a list (or partial list) of elements (journal entries, replies) created or updated within a user journal after the specified date. The function does not return the elements themselves, but their types and ID's.

Arguments

Structure with the following components:

- **authentication settings** (see item ***2.3.1***)
- **lastsync** (string) – date and time in *yyyy-mm-dd hh:mm:ss* (GMT, livejournal.com server time) format that indicates the starting point of the period for which changes must be retrieved. *The default is set to January 1, 1970 00:00:00 GMT*. When the date is set to the last request, it returns a list of objects updated since the last request call (GMT).

Sample ***syncintems*** request:

```
<?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>
```

Return Values

Structure with the following components:

- **syncitems** (array) – array of structures with the following members:
  - **item** (string) – represented as *"type-identifier."* The following types exist:

- *L* – for entries
- *C* – for comments

Internal entry and reply identifiers are used (e.g.: *C-241738* or *L-2537*)

- **action** (string) – element status; available options are:
  - *create* – newly created element
  - *update* – update element
- **time** (string) – server time in "*yyyy-mm-dd hh:mm:ss*" format of element creation/update (GMT)

- **count** (integer) – number of changed (updated) elements (starts with 1)
- **total** (integer) – total number of elements updated after the specified time

Sample ***syncintems*** request:

```
<?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>
```

Return errors

203 –Invalid argument(s)

502 –Database is temporarily unavailable

506 –Journal sync is temporarily unavailable

2.3.2.5.3.5. getdaycounts Function

Description

This function returns the number of LJ entries per day

Arguments

Structure with the following componpents:

- **authentication settings** (refer to item ***2.3.1***)
- **usejournal** (string) – Name of the journal for which counts are being requested and retrieved. *By default, it returns values for the current user*

Sample ***getdaycounts*** request:

```
<?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>
```

Return values

Structure with the following components:

- **daycounts** (array) – array containing structures with the following parameters:
  - **date** (string) – date in "*yyyy-mm-dd*" format
  - **count** (integer) – number of LJ-entries for the date specified in the ***date*** key

Sample response to a ***getdaycounts*** request:

```
<?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>
```

Return errors

203 –Invalid argument(s)

204 –Invalid metadata datatype

205 –Unknown metadata

207 –Protocol version mismatch

208 –Invalid text encoding

211 –Invalid or malformed tag list

2.3.2.5.3.6. getusertags Function

Description

This function returns a list of user-defined tags.

Arguments

Structure with the following components:

- **authentication settings** (refer to item ***2.3.1***)
- **usejournal** (string) – journal/username tags for which the function has to retrieve tags

Sample ***getusertags*** request:

```
<?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>
```

Return Values

Structure with the following components:

- **tags** (array) – array containing structures with the following:
    - **name** (string) – tag name
    - **security_level** (string) – tag security/visibility level. Supported options:
        - *public*
        - *private*
        - *friends*
        - *group*

      List of group ID's can be retrieved from the ***security*** parameter
    - **uses** (integer) – number of times the tag is used
    - **display** (string) – when set to *on,* the tag is visible in the **S2**[6] style system. When set to *off,* tags are available, but not visible, in the **S2** style system
    - **security** (struct) – structure that shows tag use statistics across security categories:

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

- **public** (integer) – number of tag instances in public entries
- **private** (integer) – number of tag instances in private entries
- **friends** (integer) – number of tag instances in entries visible to user's friends only
- **groups** (struct) – structure that has user group identifiers as keys; values stand for the number of times the tag was used in entries within the relevant user group

Sample ***getusertags*** response:

```
<?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>
<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>
```

Return Errors

100 –Invalid username

2.2.5.4. Friends

A major LiveJournal function is creating user-defined **Friends** lists. Users' friends have certain privileges compared to other users. In particular, they can read the user's secure entries and make comments. To retrieve the Friends list, call the ***getfriends*** function (refer to item ***2.5.4.1***). To check whether or not the Friends list has been changed since the specified date, use the ***checkfriends*** function (refer to item ***2.5.4.3***).

A user can edit the Friends list by using the ***editfriends*** function (refer to item ***2.5.4.4***). To retrieve the list of users who friended the user, call the ***friendof*** function (refer to item ***2.5.4.2***).

There are cases when one group of friends is not enough, e.g., when a user has too many friends and cannot read all entries on the Friends Page or when a user wants to make a certain entry visible to only a subset of their friends. In such cases, LiveJournal allows the creation of custom **friends groups**, which are managed similarly to general friends group. A user can read entries from particular users by filtering the Friends Page, restricting access to their entries to a particular friends group, etc. By default, the user can divide their friends into coworkers (Work), classmates (School), offline friends (Local Friends), online friends (Online Friends), and family members (Family). A user can rename groups, create new ones, and delete redundant groups. To obtain the list of the user's friends groups, use the ***getfriendgroups*** function (refer to item ***2.5.4.5***). To edit groups, call the ***editfriendgroups*** function (refer to item ***2.5.4.6***).

2.3.2.5.4.1. getfriends Function

Description

This function returns the user's Friends list.

Arguments

Structure with the following components:

- **Authentication settings** (refer to item ***2.3.1***)
- **includefriendof** (boolean) – when set to true, returns information on users who friended the current user
- **includegroups** (boolean) – when set to true, returns information on friends groups
- **includebdays** (boolean) – when set to true, returns birthday of each user
- **friendlimit** (integer) – number of friend descriptions required
- **friendoflimit** (integer) – number of friend descriptions to be retrieved; used when the ***includefriendof*** parameter is set to *true*

Sample ***getfriends*** request:

```
<?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>
```

Return Values

Structure with the following members:

- **friendgroups** (array) – array of descriptions for each user group
- **friendofs** (array) – array of descriptions for users who friended the current user
- **friends** (array) – array of descriptions for the user's Friends list

Friend group description. Structure with the following components:

- **id** (integer) – bit number assigned to a friend group (1 to 30)

- **name** (string) – friend group name
- **sortorder** (integer) – group number for sorting (from 0 to 255)
- **public** (boolean) – indicates whether a group is public (*1*) or private (*0*)

User description. Structure with the following members:

- **username** (string) – username
- **fullname** (string) – full username displayed on the user's Profile page or in search results
- **type** (string) – user type
- **identity_type** (string) – alternative identification type; used when ***type*** = "*identity*";
- **identity_value** (string) – URL identifier; used when ***type*** = "*identity*"
- **identity_display** (string) – Displayed username; used when ***type*** = "*identity*"
- **fgcolor** (string) – font color used for this user on the Friends page (HTML color codes format, #RRGGBB), default value #000000
- **bgcolor** (string) – background color used for this user on the Friends page (HTML color codes format, #RRGGBB), default value #ffffff
- **groupmask** (integer) – user group mask
- **birthday** (string) – user birth date in "*yyyy-mm-dd hh:mm:ss*" format if specified
- **defaultpicurl** (string) – link to a user default userpic (URL). The key is available when a user has selected a default userpic

Sample ***getfriends*** response:

```
<?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>
```

Return Errors

502 –Database is temporarily unavailable

2.3.2.5.4.2. friendof Function

Description

This function returns a list of users who friended the current user.

Arguments

Structure with the following components:

- **Authentication settings** (refer to item ***2.3.1***)
- **friendoflimit** (integer) – if the value is above *0*, the number of users returned cannot be above the defined value

Sample ***friendof*** request:

```
<?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>
```

Return Values

Structure with the following components:

- **friendofs** (array) – array describing each user

User description. Structure with the following components:

- **username** (string) – username
- **fullname** (string) – full username displayed on your Profile page or in search results
- **type** (string) – user type
- **identity_type** (string) – alternative identification type; used when ***type*** = "*identity*"
- **identity_value** (string) – URL identifier; used when ***type*** = "*identity*"
- **identity_display** (string) – Displayed username; used when ***type*** = "*identity*"
- **fgcolor** (string) – font color used for this user on the Friends page (HTML color codes format, #RRGGBB), default value #000000
- **bgcolor** (string) – background color used for this user on the Friends page (HTML color codes format, #RRGGBB), default value #ffffff
- **groupmask** (integer) – user group mask
- **birthday** (string) – user birth date in "*yyyy-mm-dd hh:mm:ss*" format if specified
- **defaultpicurl** (string) – link to a user default userpic (URL). The key is available when a user does have a default userpic

Sample ***friendof*** response:

```
<?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>
```

Return Errors

502 –Database is temporarily unavailable

2.3.2.5.4.3. checkfriends Function

Description

This function checks whether the Friends list has changed since the specified time.

Arguments

Structure with the following components:

- **Authentication settings** (refer to item ***2.3.1***)
- **lastupdate** (string) – date of the last function call in "*yyyy-mm-dd hh:mm:ss*" format (GMT)
- **mask** (integer) – user groups checked for new elements. Setting any combination of bits 1-30 detects new friends within corresponding groups. Setting bit *0* or leaving the key empty launches a general check across all groups

Sample ***checkfriends*** request:

```
<?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>
```

Return Values

Structure with the following components:

- **new** (boolean) – new elements. When request returns *1*, new elements are detected, when it returns *0,* new elements are not found
- **interval** (integer) – time in seconds that you must wait before you can make the next request to the server. An early request returns an error
- **count** (integer) – number of elements in the response
- **total** (integer) – number of elements updated after the date defined in the request (***lastupdate***)

Sample ***checkfriends*** response:

```
<?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>
```

Return Errors

203 –Invalid argument(s)

2.3.2.5.4.4. editfriends Function

Description

This function allows adding, editing, and deleting friends from the Friends list.

Arguments

Structure with the following components:

- **authentication settings** (refer to item ***2.3.1***)
- **delete** (array) – array of friends (names) to be removed from the Friends list
- **add** (array) – array of descriptions of users to be added to the Friends list

Description of a user account added to the Friends' list. Structure with the following members:

- **username** (string) – username
- **fgcolor** (string) – font color used for this user on the Friends page (HTML color codes format, #RRGGBB), default value #000000
- **bgcolor** (string) – background color used for this user on the Friends page (HTML color codes format, #RRGGBB), default value #ffffff
- **groupmask** (integer) – user group mask

Sample ***editfriends*** request:

```
<?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>
```

Return Values

Structure with the following components:

- **added** (array) – list of added friends. Array containing structures with the following keys:
  - **fullname** (string) – user's full name, if defined
  - **username** (string) – friend's username
  - **journaltype** (string) – journal type (refer to item ***2.5.1***)
  - **defaultpicurl** (string) – link to a user's default userpic (URL). The key is available when a user does have a default userpic

Sample ***editfriends*** response:

```
<?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>
```

Return Errors

104 –Error adding one or more friends

154 –Can't add a redirected account as a friend

203 –Invalid argument(s)

305 – Action forbidden; account is suspended

306 – This journal is temporarily in read-only mode. Try again in a couple minutes

308 – Account is locked and cannot be used

501 – Database error

2.3.2.5.4.5. getfriendgroups Function

Description

This function returns the list of user-defined Friends groups.

Arguments

Structure with the following components:

- **Authentication settings** (refer to item ***2.3.1***)

Sample ***getfriendgroups*** request:

```
<?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>
```

Return Values

Structure with the following components:

- **friendgroups** (array) – array of user group descriptions.

User group description. Structure with the following components:

- **public** (boolean) – indicates whether a group is public (*1*) or private (*0*)
- **name** (string) – friend group name
- **id** (integer) – bit number assigned to a friend group (1 to 30)
- **sortorder** (integer) – group number for sorting (from 0 to 255)

Sample ***getfriendgroups*** response:

```
<?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>
```

Return Errors

502 – Database is temporarily unavailable

2.3.2.5.4.6. editfriendgroups Function

Description

The function edits user-defined groups.

Arguments

Structure with the following components:

- **Authentication settings** (refer to item ***2.3.1***)
- **groupmasks** (struct) – structure contains user identifiers. It is an unsigned 32-bit integer with bit 0 set (otherwise the server forces it), bits 1-30 are defined for each group to which the friend belongs; 31 is reserved for future use
- **set** (struct) – user group description
- **delete** (array) – array of identification numbers of user groups to be deleted (from 1 to 30)

User group description. Structure with the following components:

- **public** (boolean) – indicates whether a group is public (*1*) or private (*0*)
- **name** (string) – friend group name
- **id** (integer) – bit number assigned to a friend group (1 to 30)
- **sortorder** (integer) – group number for sorting (from 0 to 255)

Sample ***editfriendgroups*** request:

```
<?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>
```

Return Values

This function returns no values.

Sample ***editfriendgroups*** response:

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

Return Errors

207 – Protocol version mismatch

208 – Invalid text encoding

306 – This journal is temporarily in read-only mode. Try again in a couple minutes

308 – Account is locked and cannot be used

2.2.5.5. Friends Page

The **Friends Page** is the list of last entries created by the user's friends. Entries on this page are displayed in reverse chronological order, from most recent first.

2.3.2.5.5.1. getfriendspage Function

Description

The function returns a part of the Friends page.

Arguments

Structure with the following components:

- **Authentication settings** (refer to item ***2.3.1***)
- **itemshow** (integer) – maximum number of returned entries (from 0 to 100). By default set to *"100"*
- **skip** (integer) – number of an entry (in the current order, from 0 to 100) that starts the Friends page. By default set to *"0"*
- **before** (string) – Unix-time when an entry was created on the server. When defined, the function returns only entries created before the specified time
- **trim_widgets** (integer) – number of characters by which an entry must be reduced (taking into account tag compressing and images)
- **lastsync** (string) – Unix-time when an entry was created on the server. When defined, this function return entries created after the specified time
- **parseljtags** (boolean) – converts embedded LJ-tags into plain HTML. By default set to false (*0*); when set to true (*1*), the system converts LJ-tags to plain HTML
- **widgets_img_length** (integer) – number of characters an image within in an entry equals. The default is set to *50*, i.e., an entry containing no text and two images is considered to contain 100 characters; the system then reduces the entry length to one link to one image; used when ***trim_widgets*** parameter specified.
- **journaltype** (string) – string with capital letters corresponded to journal types to filter entries from specified journal types (refer to item ***2.5.1***), ex. "CP" for personal journals and communities
- **groupmask** (integer) – group mask to return entries from specified groups only;

LJ-tag conversion

- **lj-poll** tag is replaced by a link with the following description:

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

- **lj-embed** tag is replaced by a link of the following type:

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

- **lj-user** tag is replaced by a link to the user journal:

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

Sample **getfriendspage** request

```
<?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:a82d42c39db2ac0147
bbd21896eb919c</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>
```

Return Values

Structure with the following components:

- **skip** (integer) – number of entries skipped from the start of the Friends page. Corresponds to the value of the ***skip*** input
- **entries** (array) – array of structures with the following parameters:
  - **userpic** (struct) – userpic description. Structure with the following members:
    - **picid** (integer) – picture identifier
    - **userid** (integer) – user identifier
  - **journaltype** (string) – journal type (refer to item ***2.5.1***)
  - **do_captcha** (boolean) – whether Captcha[7] must be entered to make a reply ("*1*" – yes, "*0*" - no)
  - **journalname** (string) – journal name
  - **ditemid** (integer) – entry external identifier
  - **postertype** (string) – type of user/journal (refer to item ***2.5.1***) that created an entry
  - **postername** (string) – name of a user who created an entry
  - **subject_raw** (string) – entry subject
  - **event_raw** (string) – entry body
  - **security** (string) – access type ("*public*" – public entry visible to all, "*private*" – private entry, visible to the journal owner only)
  - **logtime** (string) – actual Unix-time of the last entry change on the server (server time)
  - **poster_userpic_url** (string) – link (URL) to the userpic
  - **journalurl** (string) – link (URL) to the journal

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

- **posterurl** (string) – link (URL) to the journal of an entry author
- **reply_count** (integer) – number of replies to an entry
- **props** (struct) – entry properties. Refer to Appendix B

Sample ***getfriendspage*** response:

```
<?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>
```

Return Errors

209 – Parameter is out of range

2.2.5.6. Replies

A small user forum dedicated to a specific subject that generates automatic replies, which tracks user entries, comments, and replies to comments To add a reply, use the ***addcomment*** function (refer to item ***2.5.6.2***). To retrieve replies to an entry, use the ***getrecentcomments*** function (refer to item ***2.5.6.3***).

2.3.2.5.6.1. Reply Settings

A reply contains the following settings.

Reply author — user who added the reply

Date — reply post time

Subject — reply subject. A user can enter any sequence of characters, including certain HTML-tags and LJ-tags (e.g. `<lj-user>`). Also, the subject line can be left empty. The subject length is limited to 100 symbols

Note: Since Cyrillic symbols are stored as Unicode, maximum length of a Cyrillic subject is less than 100 symbols

Subject icon — a small icon selected by a reply author to represent the user's mood. This is similar to the way mood icons are added using associated keywords when an entry is created. User cannot change the set of icons

Text — reply text containing a user-defined text with HTML-tags and special tags supported by LiveJournal. Reply length is limited to 4,300 characters

Each reply has two identifiers: ***jtalkid*** (internal) and ***dtalkid*** (external). The latter is used for non-sequential numeric identifiers of replies. It is generated from the internal ID by multiplying by 256 and adding a random number from 0 to 255, generated at the moment the reply is added

2.3.2.5.6.2. addcomment Function

Description

This function allows the adding of a reply to an entry or another reply.

Arguments

Structure with the following components:

- **Authentication settings** (refer to item ***2.3.1***)
- **body** (string) – reply text (cannot be left blank). The length is limited by server settings
- **ditemid** (string) – external identifier of a commented entry
- **parenttalkid** (string) – internal identifier of a parent reply
- **parent** (string) – external identifier of a parent reply
- **subject** (string) – reply subject. By default it is an empty line.
- **prop_picture_keyword** (string) – a keyword defining a userpic is used for reply. The default is set to select the user's default userpic.
- **journal** (string) – name of the user making an API request (username and password specified in the call authentication settings). The default is set to the journal of a current user

Sample ***addcomment*** request:

```
<?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:7f3016eb8fb51c82d
91299283b0a8c17</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>
```

Return Values

Structure with the following components:

- **status** (string) – This displays function performance status. When the function is performed successfully, it returns "*OK*"
- **commentlink** (string) – Link to a reply (URL)
- **dtalkid** (integer) – External identifier of an added reply

Sample ***addcomment*** response:

```
<?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>
```

Return Errors

204 – Invalid metadata datatype

214 – Message looks like spam

314 – Only paid users are allowed to make this request

2.3.2.5.6.3. getrecentcomments Function

Description

This function returns replies in the user's journal in reverse chronological order, most recent first.

Arguments

Structure with the following components:

- **authentication settings** (refer to item ***2.3.1***)
- **itemshow** (integer) – number of retrieved replies. By default set to *10*
- **skip** (integer) – number of skipped replies. The parameter is used to define the number of replies displayed on each page. For example, when ***skip*** = *10*, 10 most recent replies are displayed on the first page, replies 11-20 are displayed on the second page, etc. By default, the parameter is set to *0*
- **trim_widgets** (integer) – number of symbols by which a reply is compressed (taking into account tag compression and images). The default is set to *50*
- **widgets_img_length** (integer) – number of symbols an image utilizes in a reply. If **trim_widgets** parameter specified. The default is set to *50*
- **parseljtags** (boolean) – converts embedded LJ-tags into simple HTML. The default is set to false (*0*); when set to true (*1*), the system converts LJ-tags to plain HTML

LJ-tag conversion

- **lj-poll** tag is replaced by a link with the following description:

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

- **lj-embed** tag is replaced by a link of the following type:

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

- **lj-user** tag is replaced by a link to the user journal:

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

Sample ***getrecentcomments*** request:

```
<?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>
```

Return Values

Structure with the following members:

- **status** (string) – This displays function performance status. Set to "*OK,"* as long as the function is performed successfully
- **comments** (array) – array containing the following keys:
  - **nodeid** (integer) – internal identifier of a commented entry. A reply belongs to a reply node of a particular entry
  - **subject** (string) – reply subject
  - **posterid** (integer) – reply author ID
  - **state** (string) – reply status. Available options are:
    - *F* for frozen
    - *S* for secure
    - *A* for active (not frozen, secure or deleted)
    - *D* for deleted
  - **jtalkid** (integer) – reply internal identifier
  - **parenttalkid** (integer) – internal identifier of a parent reply
  - **postername** (string) – name of a reply author
  - **text** (string) – reply text
  - **nodetype** (string) – reserved for future use; currently set to "*L"*
  - **datepostunix** (string) – Unix-time of reply posting

Sample ***getrecentcomments*** response:

```
<?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>
```

Return Errors

This function returns no errors

2.2.5.7. Messages

Message Center is similar to an email inbox. Mainly, the functionality is designed to store notifications received by a user. Owners of permanent, paid, and plus accounts can send and receive text messages by using the message center.

All messages are arranged into separate folders:

- **All** – general message list
- **Messages** – incoming messages from journal users
- **Friend Updates** – messages related to changes in Friends groups (e.g., when someone has friended/unfriended the current user)
    - **Birthdays** – birthday notifications
    - **New Friends** – notifications to the current user that someone friended them
- **Entries & Comments** – notifications on replies to comments made by the current user and on replies to entries of the current user
- **Flagged** – messages flagged by the user
- **Sent** – messages sent by the user

To receive messages from the Inbox, call the ***getinbox*** function (refer to item ***2.5.7.1***). An incoming message is marked as read by the ***setmessageread*** function (refer to item ***2.5.7.2***). To forward a message to another user, the ***sendmessage*** function is called (refer to item ***2.5.7.3***).

2.3.2.5.7.1. getinbox Function

Description

This function returns incoming messages.

Arguments

Structure with the following components:

- **Authentication settings** (refer to item ***2.3.1***)
- **itemshow** (integer) – number of returned entries (from 0 to 100). The default is set to *100*
- **skip** (integer) – order number (from 0 to 100), starting with the first message in Inbox (from 0 to 100). By default, set to *0*
- **lastsync** (string) – Unix-time after which messages are retrieved (inclusively)
- **before** (string) – Unix-time till which messages are retrieved (inclusively)
- **extended** (boolean) – when set to true, extended message information is displayed
- **gettype** (array) – allows the retrieving of messages for selected types

Message types:

1. Friended
2. Birthday
3. CommunityInvite
4. CommunityJoinApprove
5. CommunityJoinReject
6. CommunityJoinRequest
7. Defriended
8. InvitedFriendJoins
9. JournalNewComment
10. JournalNewEntry
11. NewUserpic
12. NewVGift (new virtual gift)
13. OfficialPost (notification from a news community)
14. PermSale (notification on permanent account sale) (probably won't happen)
15. PollVote
16. SupOfficialPost (ru_news notification)
17. UserExpunged
18. UserMessageRecvd
19. UserMessageSent
20. UserNewComment
21. UserNewEntry

Sample ***getinbox*** request:

```
<?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>
```

Return Values

Structure with the following components:

- **skip** (integer) – number of skipped messages. Corresponds to the ***skip*** input
- **journaltype** (string) – journal type (refer to item ***2.5.1***)
- **login** (string) – username
- **items** (array) – structure array with the following keys:
  - **qid** (integer) – message identifier
  - **when** (string) – message Unix-time
  - **state** (string) – message status. Available options:
    - "*N*" for not read
    - "*R*" for read
  - **type** (integer) – message type
  - **typename** (string) – message type as a string (when ***type*** = «*0*»);

Additional parameters for particular message types:

- When ***type*** = «*9*» (JournalNewComment):
  - **journal** (string) – name of the journal owner
  - **action** (string) – actions applied to the reply; options: «*deleted*», «*comment_deleted*», «*edited*», «*new*»
  - **entry** (string) – link (URL) to the commented entry
  - **comment** (string) – link to a reply (URL)
  - **poster** (string) – reply author's username
  - **subject** (string) – reply subject

When the ***extended*** key is set to "true" in the request, the following is added to the ***items*** array:

- **subject_raw** (string) – reply subject
- **body** (string) – reply text

        - **dtalkid** (integer) – reply external ID
- When ***type*** = «*18*» (UserMessageRecvd):
    - **from** (string) – name of the user who sent a message
    - **picture** (string) – link to a userpic (URL)
    - **subject** (string) – message subject
    - **body** (string) – message text
    - **msgid** (integer) – message identifier
    - **parent** (integer) – identifier of a parent message, if any
- When ***type*** = 19 (UserMessageSent):
    - **to** (string) – name of a user the message is sent to
    - **picture** (string) – link to a userpic (URL)
    - **subject** (string) – message subject
    - **body** (string) – message text

Sample ***getinbox*** response:

```
<?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>
```

Return Errors

203 – Invalid argument(s)

209 – Parameter out of range

500 – Internal server error

2.3.2.5.7.2. setmessageread Function

Description

This function marks incoming messages as read.

Arguments

Structure with the following components:

- **authentication settings** (refer to item ***2.3.1***)
- **qid** (array) – array of message identifiers to be marked as read

Sample ***setmeddageread*** request:

```
<?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>
```

Return Values

Structure with the following components:

- **result** (array) – array containing structures with the following keys:
  - **qid** (integer) – message identifiers
  - **result** (string) – message status (“*set read*”)

Sample ***setmeddageread*** response:

```
<?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>
```

Return Errors

This function returns no errors

2.3.2.5.7.3. sendmessage Function

Description

This function allows the sending of messages to other users.

Arguments

Structure with the following components:

- **authentication settings** (refer to item ***2.3.1***)
- **subject** (string) – message subject (the input field must not be empty)
- **body** (string) – message text (the input field must not be empty)
- **to** (array) – array of recipients
- **parent** (integer) – parent message identifier
- **usepic** (integer) – userpic identifier

Sample ***sendmessage*** request:

```
<?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>
```

Return Values

Structure with the following components:

- **msgid** (array) – array of identifiers of sent messages
- **sent_count** (integer) – number of sent messages

Sample ***sendmessage*** response:

```
<?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>
```

Return Errors

200 – Missing required argument(s)

203 – Invalid argument(s)

212 – Message body is too long

213 – Message body is empty

208 – Invalid text encoding

305 – Action forbidden; account is suspended

2.2.5.8. Entering Console Commands

The LiveJournal Server has a text-based administrative console for command entry[8]. The console interface is available at http://www.livejournal.com/admin/console/. To access the console via XML-RPC, use the ***consolecommand*** function (refer to item ***2.5.8.1***).

2.3.2.5.8.1. consolecommand Function

Description

This function launches administrative commands.

Arguments

Structure with the following components:

- **authentication settings** (refer to item ***2.3.1***)
- **commands** (array) – array containing commands to be sent

Sample ***consolecommand*** request:

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

Return Values

Structure with the following components:

- **results** (array) – array containing command results. Contains the structure with the following components:
    - **success** (integer) – command result (*0* or *1*)

[8] See the list of commands at http://www.livejournal.com/admin/console/reference.bml

- **output** (array) – array containing:
  - type of received message string (“*error*”, “*info*”, «» – general message)
  - text string

Sample ***consolecommand*** response:

```
<?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. Examples of LiveJournal XML-RPC Use

3.1. Perl

Sample ***getevents*** request:

```
#!/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;
```

Sample cookie authentication, retrieval of incoming messages from an email inbox and expiration of an obtained session:

```
#!/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();
```

Appendix A. List of Return Errors

User Errors:

- **100** – Invalid username
- **101** – Invalid password
- **102** – Can't use custom/private security in communities
- **103** – Poll error
- **104** – Error adding one or more friends
- **105** – Challenge expired
- **150** – Can't post as non-user
- **151** – Banned from journal
- **152** – Can't post back-dated entries in a non-personal journal (??)
- **153** – Incorrect time value
- **154** – Can't add a redirected account as a friend
- **155** – Non-validated email address
- **156** – Protocol authentication denied due to user’s failure to accept TOS.
- **157** – Tags error

Client Errors:

- **200** – Missing required argument(s)
- **201** – Unknown method
- **202** – Too many arguments
- **203** – Invalid argument(s)
- **204** – Invalid metadata datatype
- **205** – Unknown metadata
- **206** – Invalid destination journal username
- **207** – Protocol version mismatch
- **208** – Invalid text encoding
- **209** – Parameter out of range
- **210** – Client tried to edit with corrupt data. Preventing
- **211** – Invalid or malformed tag list
- **212** – Message body is too long
- **213** – Message body is empty
- **214** – Message looks like spam

Access Errors:

- **300** – Don't have access to requested journal
- **301** – Access of restricted feature
- **302** – Can't edit post from requested journal
- **303** – Can't edit post in this community
- **304** – Can't delete post in this community
- **305** – Action forbidden; account is suspended
- **306** – This journal is temporarily in read-only mode. Try again in a couple minutes
- **307** – Selected journal no longer exists
- **308** – Account is locked and cannot be used
- **309** – Account is marked as a memorial (journal is locked and does not accept comments)
- **310** – Account user needs to be age-verified before use
- **311** – Access temporarily disabled
- **312** – Not allowed to add tags to entries in this journal
- **313** – Must use existing tags for entries in this journal (can't create new ones)
- **314** – Only paid users are allowed to use this request
- **315** – User messaging is currently disabled
- **316** – Poster is read-only and cannot post entries
- **317** – Journal is read-only and entries cannot be posted to it
- **318** – Poster is read-only and cannot edit entries
- **319** – Journal is read-only and its entries cannot be edited
- **320** – Sorry, there was a problem with entry content
- **321** – Sorry, deleting is temporary disabled. Entry is 'private' now

Limit Errors:

- **402** – Your IP address has been temporarily banned for exceeding the login failure rate
- **404** – Cannot post
- **405** – Post frequency limit
- **406** – Client is making repeated requests. Perhaps it's broken?
- **407** – Moderation queue full
- **408** – Maximum queued posts for this <community+poster> combination reached
- **409** – Post is too large
- **410** – Your trial account has expired. Posting is now disabled
- **411** – Action frequency limit

Server Error:

- **500** – Internal server error
- **501** – Database error
- **502** – Database is temporarily unavailable
- **503** – Error obtaining necessary database lock
- **504** – Protocol mode no longer supported
- **505** – Account data format on server is old and needs to be upgraded
- **506** – Journal sync is temporarily unavailable

Appendix B. List of Entry Properties

Entry properties are stored within a structure with keys described in Table 2.

Table 2.

Key	Name	Description	Date Type
admin_content_flag	Admin Content Flag	Internal flag describing entry properties. Set by an administrator	string
adult_content	Adult Content Flag	Adult content flag (no adult content; not recommended for users under the age of 14; explicit adult content)	string
commentalter	Comments altered	Unix-time of last update to replies to the current entry	string
current_coords	Current Coordinates	Author geographic coordinates (in '45.2935N 123.3452W' format)	string
current_location	Current Location	Current location of an entry when the entry is posted (free text)	string
current_mood	Current Mood	Current user mood defined at posting	string
current_moodid	Current Mood ID#	Current mood identifier	integer
current_music	Current Music	Music the user is listening to when an entry is posted	string
hasscreened	Has screened replies	Set to true, when an entry has screened replies	boolean
interface	Update interface	Interface used for last update	string
opt_backdated	Back-dated	Set to true when an entry cannot be displayed in the Friends page	boolean
opt_nocomments	Don't Allow Comments	Set to true, when readers cannot comment on the entry	boolean
opt_noemail	Don't email comments	Set to true, when an entry author does not want to receive replies to their entry by email	boolean
opt_preformatted	Don't Auto-Format	Set to true when an entry contains HTML-tags and is not subject to auto-format	boolean
opt_screening	Custom Screening Level	Defines screening level for new replies to a current entry. By default, the general level set for the whole journal. Available options are: • N = screen no replies; • R = screen replies from anonymous users • F = show only friends' replies • L = screen replies from users outside the friends list, if contain links • A = screen all replies	string
personifi_lang	Personifi language	Automatic language definition by Personifi system	string

Key	Name	Description	Date Type
personifi_tags	Personifi tags	Personifi categories/tags of an entry	string
personifi_word_count	Personifi word count	Entry length defined by Personifi	integer
picture_keyword	Picture Keyword	Keyword of a picture chosen by user as an entry avatar	string
qotdid	Writer's block ID	Identifier of the Question of the Day (answered by the retrieved entry)	integer
revnum	Revision number	Number of entry revisions	integer
revtime	Revision time	Unix-time of the last revision	string
sms_msgid	SMS Message ID	Identifier of an SMS message that was converted into a journal entry	integer
statusvis	Visibility Status of an Entry	*"V'"* or "*undef*" for visible entries, "*S*" for invisible	string
syn_id	Syndicated item id	Unique identifier of a syndicated item	string
syn_link	Syndication item link URL	Original link to a syndication item	string
taglist	Tag List	List of tags separated by comma	string
unknown8bit	Unknown 8-bit text	True, when the text contains8-bit data not encoded in UTF-8	boolean
unsuspend_supportid	Support Request ID for Unsuspension Request	Identifier of a request to the support service/abuse team for entry unsuspension. Set to "*Undef*" or "*0"* when no such request is pending at the moment	integer
used_rte	Composed in RTE	True, when an entry was composed in the rich text editor	boolean
useragent	User Agent	Client type (web/mobile/sip/etc) used to create an entry	String
verticals_list	Verticals List	List of verticals (topics) to which an entry belongs, separated by comma	String