

Facebook Open Platform Release Notes

Facebook Open Platform (fbOpen) is a snapshot of the infrastructure that runs Facebook Platform. It includes the API infrastructure, the FQL parser, the FBML parser, and FBJS, as well as implementations of many common methods and tags. We've included samples and some dummy data to help you get started fast.

Facebook Open Platform also has extensibility points built in so you can add your own functionality, such as your own FBML tags, API methods, and so forth.

We hope this release is a starting point to help you more easily debug and optimize your applications, build tools that will help other developers and strengthen the ecosystem as a whole, and generally understand Facebook Platform more deeply.

If you find a bug, please submit it to the Facebook Open Platform product category at <http://bugs.developers.facebook.com/>.

We encourage you to give us feedback and to share your thoughts with other developers in the Facebook Platform Developer Forum at <http://forum.developers.facebook.com/>.

Other open source projects that already exist for Facebook Platform are listed at http://wiki.developers.facebook.com/index.php/Open_source_projects.

What's Included

The `fb-open-platform.tar.gz` archive contains the tools necessary to implement Facebook Open Platform -- including the API, FBML (Facebook Markup Language), FBJS (Facebook JavaScript) and FQL (Facebook Query Language) -- in your own environment. The archive contains the following files and top-level directories:

- A `README` file.
- An `html` directory, which contains the framework for the Facebook Open Platform API REST server, an XML schema, and the PHP client libraries for Facebook Open Platform. There are also sample files, for a canvas page and a demonstration of sample FBML tags and API calls in the `html/fbopentest` directory.
- A `lib` directory with all the core Facebook Open Platform PHP libraries for the API, FBML, FBJS, and FQL.
- An `fbopen_data_dump` file, which contains dummy data so you can test using data in a sample database.
- `libfbml-1.2.0.tar.gz`, which contains the essential libraries for parsing and rendering FBML.

The `libfbml-1.2.0.tar.gz` archive contains the following files and top-level directories:

- A `Makefile`, which is used to build the libraries.
- A `default.mk` file, which supports `Makefile`.
- A `rules.mk` file, which also supports `Makefile`.
- A `build-all.py` file, which builds Firefox and its dependencies.
- A `README` file, which contains configuration information.
- A `dependencies` directory, which is empty initially. Place all third-party open source components here. For example, put the Firefox 2.0.0.4 installer (`firefox-2.0.0.4-source.tar.bz2`) here.
- An `ext` directory, which contains the code to build the Apache PHP extension.
- A `src` directory, which contains the source files for the parsers.
- A `test` directory, which contains unit test programs and various parsers (CSS, FBML, JavaScript).

Other Dependencies

In addition to the above, you need the following:

- PHP 5
- Apache HTTP Server 1.3 (or another Web server that supports PHP 5)
- MySQL 4.1 or higher
- Firefox 2.0.0.4 (which has its own dependencies -- see below)
- Facebook Thrift (optional)

PHP 5

PHP 5 is available from <http://www.php.net>. Facebook Open Platform requires a number of PHP 5 extensions, many of which come installed in the standard build or binary. Verify they exist on your system using `php -m`.

The required standard PHP extensions are: `ctype`, `curl`, `date`, `iconv`, `json`, `mysql`, `pcre`, `simplexml`, `standard`, `xmlwriter`.

You also need the Facebook FBML extension for PHP. It is included in the Facebook Open Platform release.

Another standard PHP extension you can choose to use is the APC extension. It is referenced in `lib/thrift` but is required only if you plan on using the Thrift framework (see below).

Apache HTTP Server

You can download Apache HTTP Server 1.3 or higher from <http://www.apache.org/>. You can use another Web server as long as it supports PHP 5 and inserts data similarly into these variables:

- `$_COOKIE`
- `$_GET`
- `$_POST`
- `$_SERVER['argv']`
- `$_SERVER['CONTENT_TYPE']`
- `$_SERVER['HTTPS']`
- `$_SERVER['HTTP_HOST']`
- `$_SERVER['HTTP_USER_AGENT']`
- `$_SERVER['PHP_ROOT']`
- `$_SERVER['REMOTE_ADDR']`
- `$_SERVER['REQUEST_URI']`
- `$_SERVER['REQUEST_METHOD']`

MySQL

You need MySQL 4.1 or higher to use the dummy database. You can download MySQL from <http://dev.mysql.com>.

Firefox 2.0.0.4

The `libfbml` package depends on Firefox 2.0.0.4. The `firefox-2.0.0.4-source.tar.bz2` installer package is included with this release and resides in the `/dependencies` subdirectory. Firefox 2.0.0.4 also requires the following open source products. You can download the installer packages from the URLs associated with each component:

- `glib-2.14.6.tar.gz` (<http://ftp.acc.umu.se/pub/GNOME/sources/glib/2.14/glib-2.14.6.tar.gz>)
- `pkg-config-0.20.tar.gz` (<ftp://ftp.gtk.org/pub/gtk/v2.10/dependencies/pkg-config-0.20.tar.gz>)
- `atk-1.9.1.tar.bz2` (<ftp://ftp.gtk.org/pub/gtk/v2.10/dependencies/atk-1.9.1.tar.bz2>)
- `freetype-2.3.4.tar.gz` (<http://download.savannah.gnu.org/releases/freetype/freetype-2.3.4.tar.gz>)
- `fontconfig-2.3.97.tar.gz` (<http://fontconfig.org/release/fontconfig-2.3.97.tar.gz>)
- `libpng-1.2.25.tar.gz` (<http://internap.dl.sourceforge.net/sourceforge/libpng/libpng-1.2.25.tar.gz>)
- `cairo-1.2.6.tar.gz` (<ftp://ftp.gtk.org/pub/gtk/v2.10/dependencies/cairo-1.2.6.tar.gz>)

- `tiff-3.7.4.tar.gz` (<ftp://ftp.gtk.org/pub/gtk/v2.10/dependencies/tiff-3.7.4.tar.gz>)
- `pango-1.18.4.tar.bz2` (<http://ftp.gnome.org/pub/gnome/sources/pango/1.18/pango-1.18.4.tar.bz2>)
- `gtk+-2.10.13.tar.bz2` (<ftp://ftp.gtk.org/pub/gtk/v2.10/gtk+-2.10.13.tar.bz2>)
- `libIDL-0.8.8.tar.gz` (<http://ftp.gnome.org/pub/gnome/sources/libIDL/0.8/libIDL-0.8.8.tar.gz>)
- `libXft-2.1.12.tar.gz` (<http://xorg.freedesktop.org/releases/individual/lib/libXft-2.1.12.tar.gz>)
- `xproto-7.0.7.tar.gz` (<http://xorg.freedesktop.org/releases/individual/proto/xproto-7.0.7.tar.gz>)
- `xrender-0.8.3.tar.bz2` (<http://freeware.sgi.com/source/xrender/xrender-0.8.3.tar.bz2>)

Note: We recommend placing these installation packages (without expanding them) in the `/dependencies` subdirectory alongside `firefox-2.0.0.4-source.tar.bz2`. This way, you can use the `build.all.py` script to install all of the packages in the correct order with one command. For more information, see [Configuring Your Platform](#).

Facebook Thrift

As an option, you can install Facebook Thrift, an open source IPC implementation developed by Facebook that provides for scalable cross-language Web services development. Thrift allows you to easily set up new API methods, generating much of the API request-handling code automatically from an interface definition file. If you only wish to use the methods currently implemented, sample Thrift-generated PHP files are included under `lib/thrift`.

You can read all the documentation and download the binary from <http://developers.facebook.com/thrift>.

Facebook Open Platform License Agreement

This version of Facebook Open Platform (except for the FBML parser) is licensed under a Common Public Attribution License, or CPAL, which follows the Mozilla Public License (MPL) with two additional provisions:

1. That you include attribution to Facebook on any modifications.
2. That network deployment, or making modifications available over the network, counts as “distribution,” which makes the license appropriate for Web services.

You can find the CPAL at <http://developers.facebook.com/fbopen/cpal.html>.

The FBML parser is licensed under the MPL. You can find the MPL at <http://developers.facebook.com/fbopen/mpl.html>.

Configuring Your Host Server

After installing and configuring your server, you should have:

- A working API server under `YOUR_PLATFORM_SERVER_URL/api/restserver.php`, which responds to all API calls in the release.
- A working application under `YOUR_APP_SERVER_URL/fbopentest`, which tests each API call in the release, as well as a great deal of included FBML and FBJS.
- A set of data in a dummy database which backs the return values of each API call in the documentation.

A list of recommended operating systems and compilers for this installation process is on the Facebook Developers Wiki at http://wiki.developers.facebook.com/index.php/Facebook_Open_Platform_System_Notes. We encourage you to contribute your findings there.

Installing the Facebook Open Platform Source Code Files

You need to copy the contents of both archives to the root directory of the server where you plan to run Facebook Open Platform.

Configuring Your Facebook Open Platform Environment

To configure your Facebook Open Platform environment correctly, you need to modify a few files. The three files that need the most attention are:

- `lib/core/init.php`: Within this file, you'll need to specify:
 - Your server URL
 - The IP address, username, and password for the dummy database containing the results of the `MySQLdump` provided
 - Your domain name, suffix, and prefix for the XML namespaces in the output
 - The URL path for `restserver.php`
- `html/fbopentest/demo_libs/server_url.php`: Within this file, you'll need to specify the server URL
- `fbopen_data_dump`: the URL rooted at the `html` directory in the source distribution, to enable the `fbopentest` application included

The easiest way to do the configuration is to grep the code base for every instance of **FBOPEN:SETUP**. Instructions are provided at each instance.

Insert your dummy database content. At a command prompt in the root directory, run this command:

```
mysql -u root -h YOUR_DB_IP -p < fbopen_data_dump
```

Once you make these edits, you can run the code.

Installing Firefox and Its Dependencies

You can install Firefox and all its dependencies at once by using the `build-all.py` script, which sits at the root of the unzipped, untarred `libfbml-1.2.0` directory. Make sure you copy all the dependent packages you downloaded into the `dependencies` directory alongside the Firefox installer package. Do not unzip or untar these packages. Then, from the command line, type:

```
./build-all.py
```

The collection of open source packages will be configured, made and make installed automatically, in the order outlined above.

Then the script builds Firefox version 2.0.0.4. Once Firefox has been made, the installer creates soft links to a subset of `.o` and `.a` files that are relevant to the `libfbml` make system. An overwhelming majority of the header and source files that come with the Firefox project are already included within the `src` subdirectory. The current version of `libfbml` relies on the Mozilla HTML parser, which was modified to recognize FBML tags and construct FBML parse trees.

Once the requisite set of archive files have been generated and the soft links to those files have been laid down, `build-all.py` compiles, builds, and installs `libfbml` as a PHP extension to Apache. Right before it exits, the script should print:

```
Build complete. You'll need to restart Apache before the new PHP extension can be used.
```

Note: On some systems, we've noticed that the installation of Pango fails to copy a specific file -- `pangocairo.pc`. If the build fails because of complaints that Pango wasn't configured to use Cairo, then the quick fix is to copy `pangocairo.pc` (which resides in the `pango-1.18.4` directory) to `/usr/local/lib/pkgconfig` and run the build script again.

Changing the Source

The `build-all.py` scripts need only be executed once, as its primary purpose is to build the Mozilla parser and to lay down soft links to the Mozilla libraries within the `libfbml`-specified source tree. As you change the `libfbml` source, you only need to interact with the Makefile at the root of the entire project tree, and configure within the `/ext` directory.

Testing Your Platform

Once you have configured your environment, you should run the test applications installed in your html directory. The file `apitest.php` runs all available API methods, and the file `fbml.php` outputs some FBML to be rendered by your system. Both of these are run using the same data set as test user 1240077.

Testing the API

To test the API, load the following URL from a browser.

`http://apps.example.com/canvas.php?fb_app_name=fbopentest&fb_user_id=1240077&fb_url_suffix=apitest.php%3Fparam1%3Dfoo%26param2%26param3%3Dbar`

This URL represents the expanded form of an URL like

`http://apps.example.com/SOME_APP/SOME_PATH?SOME_GET_STRING`.

If you want to support the shorter form of an application URL (e.g. `apps.example.com/SOME_APP/SOME_PATH?SOME_GET_STRING`), Apache rewriting rules (or similar mechanisms) can be used to redirect this to the `canvas.php` URL with the following arguments:

- `fb_app_name`: A unique, human-readable name of the application. It's equivalent to `SOME_APP` in the example above.
- `fb_user_id`: The user ID of a verified user on your site, authenticated through some external mechanisms (for example, with cookies).
- `fb_url_suffix`: A URL-encoded version of `SOME_PATH?SOME_GET_STRING` in the above example. In the sample above, the string maps to query string `?param1=foo¶m2¶m3=bar`.

Testing FBML

To test FBML, load the following URL from a browser:

`http://www.example.com/canvas.php?fb_app_name=fbopentest&fb_user_id=1240077&fb_url_suffix=fbml.php`

Testing the API, FBML, and FBJS Altogether

To test the API, FBML, and FBJS all at once, load `fbopentest/testall.php` (`http://apps.example.com/fbopentest/testall.php`).

This opens iframes to your canvas pages for the API (`html/fbopentest/apitest.php`), pages featuring FBML (`html/fbopentest/fbml.php`), and many pages featuring FBJS (the rest of `html/fbopentest/`, including `fbmlplusfbjs.php`, a simple FBML page augmented by some FBJS).

Scope of this Release

The Facebook Open Platform release provides a great deal of functionality required to test and optimize Facebook Platform applications. However, for the sake of simplicity, this release leaves some implementation details to the developer.

Implementation Differences from Facebook Platform

Facebook Open Platform differs from the standard Facebook Platform in a few notable ways. With the exception of session management, the Facebook Open Platform API Web service remains essentially the same as the Facebook Platform version.

Throughout the code you'll see instances of **FBOPEN:NOTE**. These notes indicate decisions relevant to the open source release. There are a number of omissions of functionality and assumptions **not** made about the uses to which the open source API, FQL, FBML, and FBJS implementation can be put. For instance:

Read-Only Datastore

Any components that require a writable datastore are not available or significantly stubbed in this release. Most of the dummy database wrappers provided use **read** operations only.

No Caching of Static Content

Caching of static content (for example, via a content delivery network) is not part of this release. You can't include external JavaScript and CSS references, as these need to be pre-parsed and stored in your database if you wish to use Facebook Platform's functionality. This can, of course, be added again.

Only inline JavaScript (that is, wrapped in `<script></script>` tags) and inline CSS (`<style></style>`) are supported for canvas page FBML parsing. Similarly, image caching is not included.

Authentication Up to Host

This release provides no authentication mechanism for logging into canvas pages. You need to provide your own authentication system.

No Facebook CSS Included

When Facebook Platform renders FBML to the viewer, it applies CSS to give it the look and feel of Facebook. However, you must determine the look and feel for your platform by including your own CSS, as none is provided in this release.

Establishing Sessions

Because much of the authorization flow depends on proprietary site login (implemented largely in Facebook's case by cookies), you need to develop your own methods for establishing user sessions. The `$DEMO_SESSION_KEY` is used in `lib/core/init.php` as a hard-coded session that is guaranteed to work. Much of the flow surrounding session checking is left intact.

Cookies

Cookies are not available on the Facebook Open Platform display site, as they would be absent or altered in, say, a mobile platform.

Cookies are used on Facebook Platform within `html/fbml/fbjs_ajax_proxy.php`. This is omitted in Facebook Open Platform. You can re-enable your own implementation, or insert authentication information into another part of the AJAX request. Logged in users have a verified (hashed) cookie set, so it's useful to verify these cookies' hashes in order to prevent requests that masquerade as an incorrect user.

Facebook Platform API Methods in Facebook Open Platform

The Facebook Platform API is a Web service that transforms authenticated GET and POST requests to `html/api/restserver.php` to internal method calls, and returns the output as XML or JSON. This release uses a subset of standard Facebook Platform API methods (<http://wiki.developers.facebook.com/index.php/API>) such as `users.getInfo` and `friends.get`. These methods are implemented in `html/api/1.0/implementation.php`. All requests are authenticated using the application key, session key (for some methods), and signatures before being transformed into an internal stack call to `implementation.php`. Note that a “dummy” application key, secret, and session key are provided in this release.

Full documentation of the API methods is available on the Facebook Developer Wiki -- <http://wiki.developers.facebook.com/index.php/API>.

Some namespaces **not** included in Facebook Open Platform are:

- The data namespace (`data.getCookies` `data.setCookie`)
- The fbml namespace

In addition, the following specific API calls are **not** included in Facebook Open Platform:

- `admin.getAllocation`
- `admin.getDailyMetrics`
- `feed.publishTemplatizedAction`
- `users.hasAppPermission`
- `users.setStatus`

FBML Tags in Facebook Open Platform

FBML tags are essentially internal method invocations initiated during the FBML parsing stage. Some tags may wrap data from your database (like `fb:profile-pic`), some of them may be control structures (as in `fb:if`), and some of the may be display-oriented (like `fb:visible-to-friends`). `libfbml` itself normalizes and parses the input FBML tree, resulting in callbacks to methods in the PHP implementation. Reading the PHP code provides the best understanding of how FBML works on the implementation side.

The FBML tags included in the Facebook Open Platform release are in `lib/fbml/implementation/mini_facebook_implementation.php` and `lib/fbml/implementation/fbjs_facebook_implementation.php`.

Full documentation of FBML is available on the Facebook Developer Wiki -- <http://wiki.developers.facebook.com/index.php/FBML>.

About FBJS

The parser for FBJS, the Facebook JavaScript implementation, is available in this version of Facebook Open Platform. Documentation for FBJS is available on the Facebook Developer Wiki -- <http://wiki.developers.facebook.com/index.php/FBJS>.

Getting Help, Providing Feedback, and Contributing to the Ecosystem

You can find all the documentation on the Platform API, FBML, FBJS, and FQL on the Facebook Developer Wiki -- <http://wiki.developers.facebook.com/index.php>.

Should you find a bug in the Facebook Open Platform release, please report it in the Facebook Open Platform product category at <http://bugs.developers.facebook.com/>.

We encourage you to give us feedback and to share your thoughts with other developers in the Facebook Platform Developer Forum at <http://forum.developers.facebook.com/>.

You can also use the Forum to exchange code snippets – just insert `[code]` `[/code]` tags around the code in your post, and the code will be rendered in a code block.

If you'd like to contribute to Facebook Open Platform, please sign and return the Contribution Agreement (<http://developers.facebook.com/fbopen/contribution.pdf>). We'll evaluate any submitted patches or features to decide whether they'd be strong inclusions into the overall Facebook Open Platform. If we incorporate your changes, we'll send you a t-shirt!

Adding Your Own Functionality

You can also use the extensibility points built into Facebook Open Platform to create your own API methods and FBML tags.

Adding a Method to the API Interface

Before you add a method to the API interface, you need to install the Thrift binaries (see above) so you can define the types and interfaces specific to the API methods you want to create. The interface your Web service implements on the server side, `FacebookApi10If`, is defined in `lib/thrift/packages/api_10/FacebookApi10.php`, and is automatically generated from a WSDL-like file, `lib/api/1.0/api_10.thrift`. Creating types and method calls in `api_10.thrift` create the PHP types used by `FacebookApi10If`.

You can use a predefined API method like `users.getInfo` as a model for guidance for modifying existing methods, deleting methods, and adding your own.

In order to add a new method and test it, complete the following steps.

1. Add a method to `lib/api/1.0/api_10.thrift`. Within the `"service FacebookApi10 { }` scope, add:

```
string addOne(1:i32 input_num)
    throws (1:FacebookApiException error_response),
```

This example uses simple, predefined Thrift types, but you may add your own types as method inputs or outputs, which are rendered in a logical way into JSON and/or XML. A good example to walk through is `users.getInfo`. This returns a list template of member user types, which is itself a complex structural type defined in `api_10.thrift`.

2. Run `./api.build` from its directory. This creates default argument and result types in the `lib/thrift/packages/api_10` directory, in addition to any complex types you may have specified:

```
lib/thrift/packages/api_10/api_10.FacebookApi10_addOne_args.php
lib/thrift/packages/api_10/api_10.FacebookApi10_addOne_result.php
```

3. Add the method to `lib/api/1.0/implementation.php`.

```
public function addOne($input_num) {
    return $input_num + 1;
}
```

4. Add this method to your client in `html/fbopentest/demo_libs/facebook-platform/client/facebookapi_php5_restlib.php`. This is a copy of the PHP5 client Facebook distributes at <http://developers.facebook.com/resources.php>.

```
public function &addOne($input_num) {
    return $this->call_method('facebook.addOne', array('input_num' => $input_num));
}
```

5. Add this method to your `apitest.php` application to try it out.

```
print "<hr>METHOD: addOne<br>";
try {
    $result = $facebook->api_client->addOne(123);
    var_dump($result);
} catch (Exception $e) {
    print "GOT EXCEPTION " . var_dump($e);
}
```

Adding an FBML Tag

Adding a new tag is quite simple. The `addOne` method described above could just as easily be rendered as an FBML tag in the form `<fb:addOne input_num="123"/>`. If you want to treat this tag as a known entity, complete the following steps.

1. Add the new tag to the FBML implementation file. Go to `implementation/mini_facebook_implementation.php`, and add the following:

```
public function fb_addOne($node)
{
    $input_num = $node->attr_int('input_num', 0, true); // required
    return (string) ($input_num + 1);
}
```

2. Test the tag on a demo canvas page. For example, in `html/fbopentest/fbml.php`, add the tag within the `<body>` of the canvas page.

```
<fb:addOne input_num="123"/>
```

Then run `fbml.php` according to the normal canvas instructions. This should render "124" in your output page.

3. [Optional] Define the contexts in which this tag can be rendered. For example, you may want math tags to be allowed only in certain contexts (as in, they're allowed in Feed, but not when rendering canvas pages). To define the contexts for rendering this tag, attach these rules to the `FBMLFlavor` you're using to render it.

- a. First, define all the math tags you have in `lib/fbml/fbml_schema.php`.
- b. To the `$schema_tree` array, add the element:

```
'math' => array('fb:addOne'),
```

- c. If `html/canvas.php` is using, say, the `FBMLCanvasPageFlavor`, add the `allows_math` function to `FBMLCanvasPageFlavor` in `lib/fbml/flavors//canvas_page_flavor.php`:

```
public function allows_math() { return false; }
```

- d. Add this named check to `fb_addOne` (from step 1):

```
public function fb_addOne($node)
{
    $this->check('math'); // ensure flavor allows math
    $input_num = $node->attr_int('input_num', 0, true); // required
    return (string) ($input_num + 1);
}
```

Now, running the same canvas page will render the `<fb:addOne>` tag as empty. Switching `allows_math` to return `true` will display "124" again.

FBML is complex and has many pieces of distinct functionality. To learn about a tag `FOO`, look for methods `open_FOO`, `tag_FOO`, `fb_FOO`, and the `FOO` string itself.

Other Code

The Facebook Open Platform release also includes existing open-source code.

The FQL parser is generated by the `PHP_ParserGenerator`, found in `ParserGenerator.cpp`. That code is licensed under the BSD license, which you can find at <http://www.opensource.org/licenses/bsd-license.php>, and is © 2006, Gregory Beaver. Thanks, Gregory!

The FBML parser is based on the parser used in Firefox, which is licensed under the MPL. The Initial Developer of the Original Code is Netscape Communications Corporation, and portions created by the Initial Developer are Copyright (C) 1998 by the Initial Developer. You can find the MPL at <http://developers.facebook.com/fbopen/mpl.html>.