Frequently Asked Questions

Frequently Asked Questions about FireBreath

Troubleshooting

Q: I get an error when I trying to run the prep script: cmake is not a recognized command

You'll get an error like this or similar to this if you either a) haven't installed cmake or b) didn't have the cmake directory added to the system path, or on Mac OS X, don't have the symlinks installed. You can test this by just trying to run "cmake" from the command line; if that doesn't work, neither will the prep scripts.

Fix it and try again.

On Mac OS X, if CMake complains that your CMake version is tool old, update by downloading the latest version as explained in Building on Mac OS X, and then install as explained in the Q/A below.

Q: I changed something in my CMake script and it isn't working as I expect

CMake is generally a pretty good tool, but occasionally your build directory gets in a weird state. The first thing to try with bizarre inexplicable errors is to delete the build directory and rerun the prep script.

Q: Compilation failed because CoreServices/CoreServices.r could not be found

You need Xcode command line tools. It's a downloadable component (Xcode Options => Downloads) in 4.3. It can be downloaded separately for 4.2 at: https://developer.apple.com/downloads/index.action

Q: CMake failed via segmentation fault

prepmac.sh fails with a segmentation fault for FireBreath 1.6 on Lion (10.7.3) and CMake 2.8.7 (installed via brew) . CMake 2.8.8 works.

Q: Visual Studio 2012 crashes or freezes with a FireBreath Project

This is most likely because of IntelliSense failing to build its index. This setting is under Text Editor >> C/C++ >> Advanced >> IntelliSense >> Disable IntelliSense.

General

Q: How can I build a firebreath plugin on windows using *<insert your favorite free or non-microsoft compiler here>*?

Short answer: You can't. You can build with the free Visual Studio Express, however.

Longer answer: FireBreath depends on ATL, which is a proprietary Microsoft technology. It does not come with the free versions of Visual C++ available from Microsoft, and so likely you won't be able to use those either. If you are serious about developing professional applications (and plugins) on windows, you probably want to use Visual Studio anyway.

If, however, you are really serious about developing on windows and you **don't** want to use Visual Studio, that's great; feel free to rewrite the portions of FireBreath that depend on ATL so that they no longer depend on ATL, and submit the patch. We'd love to do it and support all of the free compilers, but we just don't have time.

You could also try adapting the method mentioned here for using ATL with the express (free) edition of Visual C++. If you get this working, please send us instructions and we'll update this FAQ.

Q: How do I install or update CMake on Mac OS X?

Double-click the downloaded .pkg. When asked, click *Install Command Line Links* (symlinks). If you are installing over an old version of CMake, the installer may complain that it cannot install these symlinks. If that happens, delete the old symlinks in /usr/bin/ (ccmake, cmake, cmake-gui, cmakebuild, cpack, ctest) and relaunch the installer.

Q: How should I submit a patch to FireBreath?

The preferred way to submit a clone to FireBreath is to create a clone of the repository, make your changes, and then email us on the dev list to tell us where it is and why the patch should be accepted. We are a little choosy about what code we accept, but we try not to reject patches without explaining the reasons, and often after resolving the issues we can later merge the code into the main repository.

Please note that this does not mean that we won't accept a patch in diff format; it'll just take us longer to process and it's more likely to get lost. If you choose to contribute in this way, please at least create an issue and attach the patch there.

For more information, see the post on the dev list on the subject.

Q: Can I develop a plugin that will automatically affect all webpages with FireBreath?

No, Firebreath is designed to write plugins that can only run in a webpage that uses them. Similar to how Flash only runs in webpages that use it, not across all webpages that the browser renders.

What you're looking for is an extension. See e.g.:

- MDC on extensions
- Wikipedia on Browser Helper Objects

Note that some users have had great success with bundling a FireBreath plugin as *part* of an extension.

Q: Can I develop a browser toolbar with FireBreath?

No, Firebreath is designed to write plugins, and a browser toolbar is an extension. See Browser Plugins vs. Extensions - the Difference.

Q: Can I easily extract just what i need to build my plugin from the source tree?

Q: Firebreath is very difficult to use for what I need to do because the project I create using fbgen.py gets strewn throughout the firebreath source tree. Is there a way to cleanly extract just the bits that are needed to build the plugin and separate it completely from other stuff?

See also **FireBreath Tips: Working with Source Control**

The only thing in the source tree generated by fbgen – and the only thing you need – is your project directory in `projects/<project name>`.

So if you create a plugin called `MyCoolPlugin`, it will be placed in `projects/MyCoolPlugin` and that is the only directory that you need to save.

The other files you see are in `build/` and those are generated each time you run the prep script, which runs cmake. The vcproj files should never be modified directly; instead, you should use the cmake project files (`CMakeLists.txt`, `PluginConfig.cmake`, and `projectDef.cmake`).

We realize that this is a change for most developers – particularly windows developers. Please just trust us that we have really looked into every other way that we could find of doing it, and this was the way that was simplest for everyone and allowed the most flexibility. If you take the time to learn to use the cmake project files properly, it is really not difficult at all to keep things going, and making things cross platform is hugely simplified. If you don't care about cross platform – well, sorry, we do, and we have to make the system support it. =]

Q: Why is FireBreath not available as a shared library?

FireBreath doesn't only provide methods you use from your plugin, but also the entry points, ActiveX registration code and more. Also some FireBreath components may be built differently depending on the plugins requirements (e.g. for supporting specific drawing models on Mac OS X).

Nevertheless we are considering how we could make integration in existing build systems easier but don't have much time at our hands. If you want to help out in that regard please let us know.

Q: Why is the prefix "np" prepended to the plug-in's filename on Windows?

The "np" prefix in Windows DLL's (e.g. "npFBTestPlugin.dll") is required for browsers that implement NPAPI on Windows (i.e. all browsers other than Internet Explorer) to be able to recognize the file as a plugin. If you remove this prefix browsers like Firefox, Safari and Chrome will not load your plugin.

Q: I've built the project in Visual Studio, where is my ActiveX/NPAPI plug-in?

You can find your plug-in DLL in `"build/bin/PluginName/Debug/npPluginName.dll"`. Both the Active-X control (for IE) & NPAPI plug-in (for all other browsers) are contained in this in one DLL. Note, the "np" prefix on the DLL filename is required for browsers other than Internet Explorer to be able to load the plug-in.

Q: This cmake thing is lame. I converted the project to relative by hand and everything works without cmake now! I just update the vcproj projects in Visual Studio.

In our experience, you will regret the day you decided to do this. The cmake dependency is a little bit of a pain; however, it's a whole lot better than the alternatives, which would require maintaining 7 or 8 different build files.

The point, however, is that while you can strip the cmake stuff out of the visual studio project, it is almost always a really bad idea. If you ever need to change a plugin setting in PluginConfig.cmake, you have to regenerate your project. If you decide you want to start using logging, you need to regenerate your project. If you upgrade FireBreath, you'll need to regenerate your project. In short, generating your project (with the prep scripts) is something that should be done to set up your project and often.

Trust us on this: it is much easier to just learn a little cmake than it is to fight with the alternatives. Look at FBTestPlugin for examples.

Q: What is the threading model of FireBreath?

Firebreath itself does not have a thread model. Browser Plugins operate entirely on one thread in response to calls from the browser. You are not allowed to make calls to the browser from other threads. To assist with using threads in your plugin, FireBreath 1.3.0 and later have checks to make sure that calls that need to be on the main thread are. When calling into JavaScript, all calls to a FB::JSObject will be automatically forced to the main thread, and InvokeAsync can be used to make a call asynchronously. Additionally, CallOnMainThread and ScheduleOnMainThread can be used with boost::bind to make any call on the main thread.

Q: Is unicode supported - i.e. converting between wstring (utf-16) and string (utf-8)?

Yes, see FB::wstring_to_utf8 and FB::utf8_to_wstring. There is some suspicion that wstring on windows is not actually utf-16, but this will convert to whatever platform APIs expect on the platform it is compiled to.

Q: Will my FireBreath based plugin work in Internet Explorer 9?

Yes. There is an oddity in IE9 (a difference in the ActiveX behavior) that can cause problems with FireBreath 1.4.x and earlier that is fixed in FireBreath 1.5.0. See **FIREBREATH-11** - Getting issue details... STATUS

It is worth noting that even though IE9 has support for addEventListener instead of attachEvent, you can't use it with ActiveX Controls (and thus FireBreath plugins). If you want to use events in IE (all versions) you must use attachEvent.

Q: Why is drawing in my plugin window being rendered upside down when I'm using the Core Graphics drawing model (mac specific)?

NPAPI reference states that the CGContext given is flipped. In order to draw conventionally with Core Graphics (origin in the bottom left), use the following core graphics functions before you draw to the context given you by Firebreath:

```
CGContextTranslateCTM(context, 0.0, height);

CGContextScaleCTM(context, 1.0, -1.0);
```

where height is the height of the plugin window

Q: The project files are all absolute paths; how do I move them around / put them in source control / share them with others?

You don't! The build directory (the location of the XCode project, Visual Studio project (.sln), or makefiles) is not shared! It's disposable! You can delete it at any time and recreate it by re-running the prep script.

See also **FireBreath Tips: Working with Source Control**

Q: How do I draw?

First, see FireBreath Tips: Drawing on windows. Drawing on Mac or Linux is similar, except instead of getting a PluginWindowWin or PluginWindowlessWin object you will get an object deriving from PluginWindowX11 or PluginWindowMac. For Mac drawing you need to understand the Mac Drawing Models.

Q: Is it possible to overload a function or an operator in FB plugin?

You can't easily; If you want to do that, you'll have to use FB::CatchAll. If you use FB::CatchAll, then it will accept any number of parameters to the function and you'd have to decide what to do based on which parameter does what. This isn't a direct interface from the browser, it's a case of the browser saying "call the method named <<name>> with these parameters". so I'd have to go through each possible candidate and try converting the arguments, then fail over if it didn't work. It's recommended not to overload the methods in the plugin.

You can't overload operators; there is no support for that in NPAPI or ActiveX.

Q: Why is my plugin SO big?

Because a large amount of code is needed to make your code run in the browser. You will end up with about 2 MB for a plugin that seems to do "nothing". Things you can do to reduce the size:

- Make sure you build in Release mode instead of Debug mode.
- Compress your plugin, when shipping. Your installer should do this for you.

This should get you down to around 600 KB. If you really need to squeeze out every single byte, build with MinSizeRel.