leafChat Software Development Kit

Introduction

Welcome to the leafChat SDK. I hope you find this SDK easy to use and sufficient for the plug-ins you want to code.

This is version 1.0 (final) of the SDK, and is slightly upgraded from the beta version. Plug-ins that worked in the beta (system version 1) are compatible with this version (system version 2), but not vice-versa.

Legal

Although the author owns copyright in all the files that make up the leafChat SDK (contained in this ZIP file leafChatSDK10.zip), you may use these as part of a leafChat plug-in you are coding, without payment.

You may not redistribute the individual files themselves. This entire ZIP file may be redistributed only as part of the leafChat10.exe file that forms the leafChat 1.0 package.

Intended audience & requirements

The intended audience for this SDK is programmers with experience of programming Windows in C or C++. If you don't have the necessary skills (perhaps you only program in Visual Basic rather than a "real" language), you should learn them before you try to use this SDK. It isn't possible at present to develop leafChat plug-ins in any language other than C or C++, and there are no plans to provide an SDK for other languages.

Before you can develop leafChat plug-ins, you also need:
- This *SDK*
- A 32-bit Windows *C++ compiler* - Microsoft VC++ 4.0 or above is recommended. You may need to tweak the header files etc. if you have another type of compiler (see below).
- You will need to be familiar with the "basic" *Win32 API* or with *MFC* if you want to create your own windows etc. If all you want to do is use text or bitmapped windows and run IRC commands, you can get away without any Windows programming knowledge.
- A *good understanding of leafChat* and its reaction/event system (in particular, the "On" and "OnTimer" commands are likely to be useful)

Types of plug-in

With this SDK, you can develop two kinds of plug-in:
- *Command plug-ins* - these are launched when the user types a given command, or can launch automatically on a timer or received IRC data. Command plug-ins can create windows of their own type within the leafChat system, as well as creating leafChat text or graphics windows.
- *DCC plug-ins* - these are launched when the user receives a file with a particular extension. Typically, these are useful to display graphics files as they are received, but they could also work for sound files or other data.

How to begin

When you know which type of plug-in you need to create, read the instructions in the appropriate .doc file supplied as part of this zip.

Generally, you will create a new project in your C++ environment, which should be a standard Windows 32-bit DLL. You'll #include either ccSCPlugIn.h or ccDCCPlugIn.h, and then fill in some more required functions. There's no need to write a DllMain function (or similar Windows overhead) because the .h files are designed to make things as easy as

possible. Once the DLL is finished and working, you rename it to either ".leafChatPlugIn" or ".leafChatDCCPlugIn" so that it shows up on the install dialogs, and it is then ready for release.

Using MFC

In this release of the SDK, you can use MFC to develop plug-ins as well as the standard Windows API.

If you do this, you need to make a “regular DLL” (not an “extension DLL”). leafChat may not necessarily ship with the MFC libraries, so you should probably use static libraries (shipping large shared libraries with a tiny plug-in would not be sensible).

Using MFC 4.2 in the debug version causes some problems when combined with the release version of leafChat, so build your code in release version. I am not sure what compatibility issues exist with other versions of MFC.

The only “problem issue” that came up in my use of MFC for plug-ins was resource handles. MFC tries by default to load your resources from leafChat.exe, which obviously doesn’t work. Before you load any resources (such as a dialog box) in your program, you need to use AfxGetResourceHandle and store the handle. Then AfxSetResourceHandle to the handle for your DLL, which you can get with GetModuleHandle(“whatever.leafChatPlugIn”). [This is why you will often need to alter the code to make it a .leafChatPlugIn, rather than just renaming from .DLL.] Then create the dialog box or load the resource; finally, AfxSetResourceHandle back to the previous value.

Because MFC takes the “DllMain” function, which normally calls Shutdown, Shutdown will not be called in an MFC-built DLL. (You are warned of this by a #pragma message.) You should call Shutdown yourself from CYourApp::ExitInstance().

Using non-Microsoft compilers

The key thing you will need to change if you don’t use a Microsoft compiler is the #definition of DllExport. Microsoft’s syntax is not the same as that used by other compiler vendors. The purpose of “DllExport” is to mark the function as one that is callable from outside the DLL, the procedure which (back in the days) would have been accomplished using a .def file. Some compilers use a variant of __export.

Technical support

Please try to find people who might help on IRC or other appropriate locations; you could try the channel #leafChat, for example.

If you have a problem you can't solve you can email me at leafChat@poboxes.com. Please don't email me with problems until you've spent at least a couple of days trying to figure out what's wrong.

Also, don't email me if the problem is with the .h files not working under your compiler; the files are only tested under MS VC++, and are not likely to work unchanged with any other compiler. If they don't work, you'll have to fix them yourself. (By the way, we strongly recommend either MS VC++ or Symantec C++ for programming general Windows applications, and possibly Watcom C++ for games. We don't recommend Borland's C++ products under any situation.)

Feel free to use the above address for bug reports or suggestions for improvements to this SDK. Note that I might not always be able to reply quickly, or even at all, depending on my email load.

Do *not* email the above address for any problems related to leafChat in general, rather than this SDK. I don't provide any kind of technical support for leafChat. (You can still send bug reports, but that's it.) You might be able to get help with leafChat from other users online, in the channel #leafChat, or in appropriate newsgroups. See leafChat documentation for more details.