Embedding FreeSWITCH

About

Just as FreeSWITCH™ can run embedded languages you can also embed FreeSWITCH™ into other programs in a variety of languages. FreeSWITCH™ is designed so you can interact with it through modules, embedding freeswitch in other applications, or through the Event Socket.

Benefits of using Embedded FreeSWITCH in your application

- FreeSWITCH is billed as a multi-protocol softswitch for data. If you have an application that requires even a segment of what freeswitch requires, it can be the easiest way to add that functionality in.
- simple
- large code base
- maintained by others
- stable (and when you are only using voice as part of your application you don't want a bad voice library crashing your application or requiring a restart)
- free
- Rapidly evolving
- Runs in separate threads so not really effected by your other code (no audio issues generally due to the application doing something else)
- Nat handling
- Documentation
- Cross protocol/network communication

Downsides of Embedded FreeSWITCH

- You have to include the core of freeswitch which is several megabytes (4-8mb or so)
- Load time is not great if you need embedded SQL/UPNP NAT detection. Most likely optimization here could be made but you are looking at 10-15 seconds on startup by default (although this can be done in the background but must be done before you can make calls with FS).
- No g729 support in windows yet

Using Embedded FreeSWITCH

Environment Setup

The first thing you need to do once you have freeswitch compiling (or the binary installed) is setup your working environment. For any platform you will want to make a mod folder and copy and modules you want from freeswitch's mod folder to yours. You will also want to move your languages runtime from the mod folder to your working environment (for .net this is mod_managed.dll). Generally when you are using the bindings for your language it will try to load the dll/lib from the working directory rather than mod/. Finally there are a few language specific changes:

Linux

For Linux this means you want the copy of the contents of the lib folder in to your working directory

Windows

In windows this means lib*.dll, ssleay*.dll, Freeswitch.dll from the build folder to your working directory

FreeSWITCH Config

FreeSWITCH must have a conf folder and that must contain a freeswitch.xml file in it that tells freeswitch its initial config. You may be able to embed this (or provide it internally) but it does need to have a config of some sort. You can start with all the configs in the freeswitch build conf folder, but it may be easier to start with the simple example under the resources part of this page.

Starting Up FreeSWITCH

Note some of the functions mentioned here may be a little bit different depending on the language (the swig binding call should be similar though). Theoretically all you would have to do is call switch_core_init_and_modload it would do everything needed. switch_core_init_and_modload takes 3 params, flags which specifies things like SQL or NAT support, if a console is present (only use this if using c++ generally and even still you probably don't want the console spew) and a reference to an error string for an error if one happens. Of course it isn't actually quite that easy. The proper way to spin it up is:

- call switch_core_set_globals
- call switch_core_init
- finally spin freeswitch up with: switch_core_init_and_modload

Now the switch_core_init call is not needed unless you want to tie into the initial loading. The most likely reason to be involved in the initial loading is for tying into the xml_search binding to be able to return custom XML during load from code. Why must the call pattern follow that? Technically switch_core_init calls switch_core_set_globals, however switch_core_set_globals must be called independently as switch_core_set_globals is not called high enough in the function for it to work properly.

Interacting with freeswitch

Now you could tie your own app into also handling sessions (much like traditional dialplan calls) from your app. Depending on the application however it may be better to do as a traditional module/script just for code cleanness / separation. You of course do want to have some interaction with the core directly from your application most likely. The event module you can still use and then communicate over the event socket like external applications. You can also rely on the swig calls and interact with freeswitch through the native functions themselves. Swig allows you to call just about any native code, code with callbacks can be a bit more complex but in general you can call most native methods without issue. A best practice may be to instead use the API interface for dealing with freeswitch. Using the API most tasks can be accomplished and has the added safety of making it much harder to decrease the stability of freeswitch itself. There is also a large amount of documentation for the public API for freeswitch vs all the internal functions themselves. There are a few situations however where direct swig calls can be useful, most of the time it has to do with complex callbacks or function ties. There are two examples that frequently come up. The first is the switch_xml_bind_search function, this allows you to return XML configuration during loading. There is a .net binding that the managed dll has to make it easier to add a callback (other bindings may or may not provide this). This allows you to return the config for any module directly. The second example is for events. Now you can tie into events several ways, for example through the event socket, but you can also tie into the main event loop (switch_event_bind) . Now doing this is risky, as its part of the main event loops events are not fully processed until you are done with them, so its suggested you duplicate events if you aren't handling them very quickly. The managed dll binding for this has a parameter for that.

Language Examples

Note the below examples use the SQL and AUTO_NAT flags. If you do not need these (and many softphones using embedded FS do not) do not turn them on (instead pass 0). Turning them on delays FS loading by 5-15 seconds each.

.NET / C#

You must add a reference to the FreeSWITCH.Managed.dll to your .net project. The two lines that start with `search_bind` and `event_bind` are not needed, they are only used if you want to bind to the XML search path or the internal event loop; otherwise they can be completely skipped.

.Net C# Embedded Example

```
String err = "";
const uint flags = (uint)(switch_core_flag_enum_t.SCF_USE_SQL | switch_core_flag_enum_t.SCF_USE_AUTO_NAT);
freeswitch.switch_core_set_globals();
/*Next 3 lines only needed if you want to bind to the initial event or xml config search loops */
       freeswitch.switch_core_init(flags, switch_bool_t.SWITCH_FALSE, ref err);
       search_bind = FreeSWITCH.SwitchXmlSearchBinding.Bind(your_xml_search_func, switch_xml_section_enum_t.
SWITCH_XML_SECTION_CONFIG);
       event_bind = FreeSWITCH.EventBinding.Bind("SampleClient", switch_event_types_t.SWITCH_EVENT_ALL, null,
your_event_handler_func, true);
/* End Optional Lines */
freeswitch.switch_core_init_and_modload(flags, switch_bool_t.SWITCH_FALSE, ref err);
```

C/C++

Embedded C Example

```
#include <switch.h>
int main(int argc, char** argv)
{
        switch_core_flag_t flags = SCF_USE_SQL;
        bool console = true;
        const char *err = NULL;
        switch_core_set_globals();
        switch_core_init_and_modload(flags, console ? SWITCH_TRUE : SWITCH_FALSE, &err);
        switch_core_runtime_loop(!console);
        return 0;
}
```

PHP

Embedded PHP Example

```
<?php
 include("freeswitch.php");
 fs_core_set_globals();
 fs_core_init("");
 fs_loadable_module_init();
 fs_console_loop();
 fs_core_destroy();
?>
```

Projects that use Embedded FreeSWITCH

- Softphones - There are SIP Clients/Softphones that use Embedded FreeSWITCH, see that page for which

References

- .NET Example project that starts up and performs a simple phone call - http://fisheye.freeswitch.org/browse/~raw,r=HEAD/freeswitch-contrib/mitchcapper/SampleClient/FSSampleEmbeddedClient.zip
- Check out some of the projects using embedded freeswitch many are open source

See Also

- Developer Documentation