Client and Developer Interfaces

About

Many languages can directly build FreeSWITCH modules or be run directly from the dialplan or from the fs_cli.

There are a few ways that you can connect your own applications with FreeSWITCH:

mod_xml_curl

The curl module is used to provide FreeSWITCH with information such as configuration, dialplans and users.

mod_xml_curl

mod_xml_rpc

Connect to FreeSWITCH from your own application RPC client and issue commands.

mod_xml_rpc

Script languages

The first and simplest one is to use one of the scripting languages.

Script Language Choice

Event socket

The event socket is the most advanced interface to use. It's also the most powerful interfance. It let's you control almost everything in FreeSWITCH.

Event Socket Library

Current directly supported languages

- C/C++
- Java
- C# / .NET
- Javascript
- Lua
- Python
- Perl

Any language not directly supported can still interact with and control FreeSWITCH via these several options:

- Event Socket Library - a generic socket to watch for events and issue commands to FreeSWITCH - bindings are available in many languages.
- mod_xml_curl - to serve the dialplan, directory, ACL, configuration.
- mod_xml_cdr - to process CDRs upon call completion.

Previously supported languages

- PHP
- Ruby

Distilled Wisdom

It is generally best to do most telephony processing in the dialplan, not in your script. The FS team has put a great deal of effort into making sofia handle many edge cases of SIP processing, so trying to do all that in your script would prove hopeless.

From a core developer:

Bridge inside script

From: Anthony Minessale <anthony.minessale@gmail.com>
Date: Mon, 22 Sep 2014 13:20:29 -0500

uuid_bridge is a "transfer to" type of bridge so if you want to use it and your script is running on one of the legs involved, you need to allow your script to exit so the bridge can begin.

If you have the uuid of another leg and you want to bridge this leg to it you can use:

```
session:execute("intercept", "<other uuid>")
```

If you are manually creating legs inside Lua scripts just to bridge them, it's generally overkill. You can just do

```
session:execute("bridge", "<dial string>")
```

which is a blocking call that will dial and bridge and return to your script when it's done.

If you don't need to remain in your script you should set a custom variable to the destination you want to dial, exit the script, and use the bridge app from the xml dialplan.

Calling out to somewhere is a bridge, transfer is to take your inbound call and go run some different script for example

From an add-on developer:

Script threads

crienzo: any dialplan application that executes runs inside the same session thread that handles RTP media
crienzo: so a Lua script can stall RTP processing if you do silly things
crienzo: however, luarun is an API that runs in its own thread