扩展 Asterisk1.8.7 的 CLI 接口

大部分情况下，配置 asterisk 的拨号方案，用 CLI、AMI 之类的就可以满足我们的需求。可有些情况下涉及到业务的东东，需要数据库的参与（比如用 sqlserve 存储 asterisk 的录音记录等等），拨号方案那种静态的做法完全不用考虑，而原始的 CLI、AMI 已经不能满足需求。这时就需要考虑从源码入手，扩展 asterisk 了。

asterisk 是基于插件的，很容易扩展。手动编译过 asterisk 源码的朋友应该知道，在 asterisk 源码目录里有一个 addons 的目录，里面就是 asterisk 的插件（其实 apps 下也可以看做是插件）。

这里有个小例子，主要演示怎么从源码扩展 asterisk 的 CLI 接口。

一、建立目录结构，配置 Makefile

1、为了方便代码的管理，我决定新建立一个叫 addons_test 的文件夹；

2、将 apps 下的 Makefile 复制到该目录；

3、打开 asterisk 主目录下的 Makefile 文件，在 MOD_SUBDIRS 变量中加入 addons_test（我的 Makefile 是在 266 行）。

```
263
264 _ASTCFLAGS+=$(OPTIONS)
265
266 MOD_SUBDIRS:=channels pbx apps codecs formats cdr cel bridges funcs tests main res addons addons_test $(LOCAL_MOI
267 OTHER_SUBDIRS:=utils agi
268 SUBDIRS:=$(OTHER_SUBDIRS) $(MOD_SUBDIRS)
269 SUBDIRS_INSTALL:=$(SUBDIRS:%=%-install)
270 SUBDIRS_CLEAN:=$(SUBDIRS:%=%-clean)
271 SUBDIRS_DIST_CLEAN:=$(SUBDIRS:%=%-dist-clean)
```

二、编写 CLI 插件代码

1、在 addons_test 目录添加文件 app_testApp20120605.c 和文件 app_testApp20120605.exports

说明：

app_testApp20120605.c 为程序代码

app_testApp20120605.exports 为动态库导出配置

2、编写文件内容

app_testApp20120605.exports 文件简单，可以将 apps 目录下的任一”.exports”文件 copy 至本目录改名即可，这里主要介绍 app_testApp20120605.c 的书写。

2.1 首先需要添加头文件：

```
#include "asterisk.h"
#include "asterisk/module.h"
#include "asterisk/cli.h"
```

2.2 定义 Application 名称：

```
static char *app_testApp = "testApp20120605";
```

2.3 写模块加载函数：

```
static int testApp_exec(struct ast_channel *chan, const char *data)
{
    ast_verb(2,"testApp_exec : %s\r\n",data);
    return 0;
}
```

说明：这个要用此格式，尽管 chan 变量没有用到，但加载模块的函数指针是这种格式。

2.4 编写 CLI 接口函数：

```
static char *handle_cli_testApp(struct ast_cli_entry *e, int cmd, struct
ast_cli_args *a)
{
    struct ast_channel *chan=NULL;

    if(CLI_INIT == cmd) {
        e->command = "testApp20120605 {print}";
        e->usage =
            "Usage: testApp20120605 <print> <something2print>\n"
            "       Print something to test application\n"
            "       application when the 'print' command is used.\n";
        return NULL;
    }

    if (a->argc < 2)
        return CLI_SHOWUSAGE;

    if (!strcasecmp(a->argv[1], "print")) {
        testApp_exec(chan, a->argv[2]);
    }else{
        return CLI_SHOWUSAGE;
    }

    return CLI_SUCCESS;
}
```

2.5 编写模块加载函数：

```
static int load_module(void)
{
    int res;
    ast_cli_register_multiple(cli_testApp, ARRAY_LEN(cli_testApp));
    res = ast_register_application_xml(app_testApp,testApp_exec);
    return res;
}
```

2.6 编写模块卸载函数：

```
static int unload_module(void)
{
    int res;
    ast_cli_unregister_multiple(cli_testApp, ARRAY_LEN(cli_testApp));
    res = ast_unregister_application(app_testApp);
    return res;
}
```

三、测试 CLI 插件

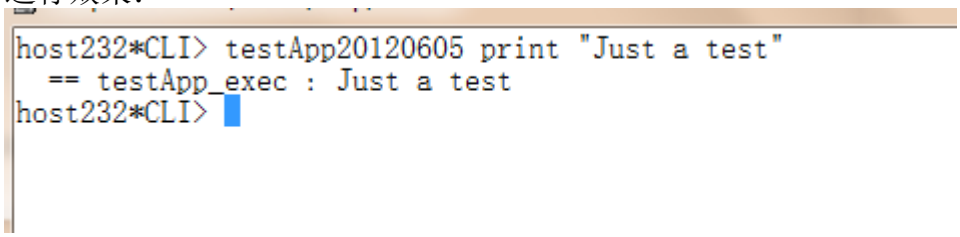
1、编译运行

执行如下命令：

make && make install && asterisk && asterisk -rvvvvvvvv

2、测试

启动后，执行如下命令：
testApp20120605 print "Just a test"
运行效果：



app_testApp20120605.c 的完整代码：

```
/*
 * Asterisk -- An open source telephony toolkit.
 *
 * Copyright (c) 2004 - 2005 Tilghman Lesher.  All rights reserved.
 *
 * Tilghman Lesher <app_verbose_v001@the-tilghman.com>
 *
 * This code is released by the author with no restrictions on usage.
 *
 * See http://www.asterisk.org for more information about
 * the Asterisk project. Please do not directly contact
 * any of the maintainers of this project for assistance;
 * the project provides a web site, mailing lists and IRC
 * channels for your use.
 *
 */

#include "asterisk.h"
#include "asterisk/module.h"
#include "asterisk/cli.h"

static char *app_testApp = "testApp20120605";

// Must in this format
static int testApp_exec(struct ast_channel *chan, const char *data)
{
    ast_verb(2,"testApp_exec : %s\r\n",data);
    return 0;
}

static char *handle_cli_testApp(struct ast_cli_entry *e, int cmd, struct
ast_cli_args *a)
{
    struct ast_channel *chan=NULL;

    if(CLI_INIT == cmd) {
        e->command = "testApp20120605 {print}";
        e->usage =
            "Usage: testApp20120605 <print> <something2print>\n"
```

```
            "       Print something to test application\n"
            "       application when the 'print' command is used.\n";
        return NULL;
    }

    if (a->argc < 2)
        return CLI_SHOWUSAGE;

    if (!strcasecmp(a->argv[1], "print")) {
        testApp_exec(chan, a->argv[2]);
    }else{
        return CLI_SHOWUSAGE;
    }

    return CLI_SUCCESS;
}

static struct ast_cli_entry cli_testApp[] = {
    AST_CLI_DEFINE(handle_cli_testApp, "Execute a testApp20120605 command")
};

static int unload_module(void)
{
    int res;
    ast_cli_unregister_multiple(cli_testApp, ARRAY_LEN(cli_testApp));
    res = ast_unregister_application(app_testApp);
    return res;
}

static int load_module(void)
{
    int res;
    ast_cli_register_multiple(cli_testApp, ARRAY_LEN(cli_testApp));
    res = ast_register_application_xml(app_testApp,testApp_exec);
    return res;
}

AST_MODULE_INFO_STANDARD(ASTERISK_GPL_KEY, "testApp20120605 by
Mike_Zhang@live.com");
```

app_testApp20120605.exports 的完整代码：

```
{
        local:
                *;
};
```