

Developer 101: API Guide

Tilghman Leshner
Senior Software Developer
September 23, 2008

Module API

- #include “asterisk/module.h”
- load_module/unload_module/reload
- AST_MODULE_INFO(ASTERISK_GPL_KEY,
AST_MODFLAG_DEFAULT, “Some module description”,
.load = load_module,
.unload = unload_module,
.reload = reload,
);

Module API: load_module()

- All startup routines for the module:
 - Read configuration
 - Allocate structures
 - Setup network ports
 - Start monitor thread
 - Register CLI, AMI, AGI, CDR, Application, and Function pointers (among others)

Module API: `unload_module()`

- Unregister pointers registered in `load_module()`
- Shutdown monitor threads
- Close all connections
- Shutdown network port
- Deallocate global structures

Configuration parsing

- `#include "asterisk/config.h"`
- `struct ast_config *ast_config_load("file.conf", flags)`
where flags contains `CONFIG_FLAG_`
`{COMMENTS,FILEUNCHANGED,NOCACHE}`
- `ast_config_destroy(cfg)`

Configuration parsing

- `char *ast_category_browse(cfg, const char *prev)`
- `struct ast_variable *ast_variable_browse(cfg, category)`
- `var->name, var->value`
- `const char *ast_variable_retrieve(cfg, category, varname)`

Useful for configuration parsing

- `ast_true/ast_false`
- `ast_strip`
- `ast_strlen_zero`
- `ast_copy_string`

Linked lists

- `#include "asterisk/linkedlists.h"`
- mutex-based vs rwlock-based
- `AST_{RW}LIST_HEAD_STATIC(name, type)`
- `AST_{RW}LIST_ENTRY(type) elm_name`

Linked lists

- `AST_{RW}LIST_TRAVERSE(head, current, list_elm_name)`
- `AST_{RW}LIST_INSERT_{HEAD, TAIL}(head, list_elm_name)`
- `AST_{RW}LIST_REMOVE_HEAD(head, list_elm_name)`
- `AST_{RW}LIST_TRAVERSE_SAFE(head, current, list_elm_name)`
- `AST_{RW}LIST_REMOVE_CURRENT(list_elm_name)`

Linked lists, locking

- `AST_LIST_{LOCK,UNLOCK,TRYLOCK}(head)`
- `AST_RWLIST_{RD,WR,UN}LOCK(head)`
- `AST_RWLIST_TRY{RD,WR}LOCK(head)`

AstObj2

- #include “asterisk/astobj2.h”
- Advanced, hash-based indexes (but no collision avoidance)
- Reference-counting
- Close to $O(1)$ search time

AstObj2

- `o = ao2_alloc(sizeof(struct foo), my_destructor_function)`
- `ao2_{lock,unlock,trylock}(o)`
- `ao2_ref(o, {+,-}1)`
- `c = ao2_container_alloc(BUCKETS, my_hash_fn, my_cmp_fn)`
- `ao2_link(container, object)`
- `ao2_unlink(container, object)`

AstObj2

- ```
struct ao2_iterator ao2i;
struct some_structure *obj;
ao2i = ao2_iterator_init(container, flags)
while ((obj = ao2_iterator_next(&ao2i))) {
 ...
 ao2_ref(obj, -1);
}
```
- `ao2_callback(container, flags, callback_function, argument)`

# Memory management

- `ast_malloc()`, `ast_calloc()`, `ast_realloc()`, `ast_free()`
- Use `ast_free_ptr()` when needing a pointer to a free function
- Usually wrappers, but useful when debugging is needed.

# API registration

- No global handles, all pointers are registered
- `ast_cdr_register(name, description, function_ptr)`
- `ast_register_application(name, function_ptr, synopsis, description)`
- `ast_custom_function_register(&structure)`
- `ast_manager_register(name, permission, function_ptr, short_descr)`

# Module dependencies

- Ensuring another module is loaded as a prerequisite
- `ast_module_check("some_module.so")`
- `ast_load_resource("some_module.so")`
- `ast_module_helper("", "some_module.so", 0, 0, 0, 0)`

# Common API: CDRs

- Easiest of all modules to write
- Pre-parsed data
- Common use is to format an SQL string
- Another common use is to output directly to a file
- #include “asterisk/cdr.h”
- `ast_cdr_getvar(struct ast_cdr *, fieldname, int *indicator, char *buffer, buffer_size, recursive_flag, rawflag)`

# Common API: String building

- `#include "asterisk/strings.h"`
- `ast_str_create(initial_size)`
- `ast_str_alloca(maximum_size)`
- `AST_THREADSTORAGE(foo)`  
`ast_str_thread_get(&foo, initial_size)`

# Common API: String building

- `ast_str_set(&str, max_len_or_zero, "format", ...)`
- `ast_str_append(&str, max_len_or_zero, "format", ...)`
- `ast_str_reset(&str)`
- `ast_free(str)`

# CDR Challenge: cdr\_curl

- Create a working cdr\_curl by the end of Astricon
- Not a speed contest
- Challenge is to create the most fully-featured version
- Use res\_curl for loading resources
- May use func\_curl for posting URLs

# APIs: Applications

- Slightly more complex than CDRs
- Need to parse arguments
- Definition of an application: “doing” something

# Argument parsing

- #include “asterisk/app.h”
- ```
AST_DECLARE_APP_ARGS(structure_name,  
    AST_APP_ARG(field1);  
    AST_APP_ARG(field2);  
);
```
- ```
AST_STANDARD_APP_ARGS(structure_name, destructable_string)
```
- `structure_name.field1, .field2, .argc`

# Options parsing

- Our standard option syntax
- `ab(b_arg)c(c_arg)def`
- See `app_skel.c` for a good example

# APIs: Dialplan Functions

- `#include "asterisk/pbx.h"`
- When to use Dialplan Functions as opposed to Applications
- Both read and write functions

# APIs: Media File Playback

- `#include "asterisk/file.h"`
- `ast_streamfile(channel_ptr, filename, pref_language)`  
`ast_waitstream(channel_ptr, "")`
- `ast_stream_and_wait(channel_ptr, filename, "")`

# APIs: other media file operations

- `#include "asterisk/file.h"`
- `ast_fileexists(filename, "fmt1|fmt2|...", pref_language)`
- `ast_filerefname(old_filename, new_filename, "fmt1|fmt2|...")`
- `ast_filedelete(filename, "fmt1|fmt2|...")`
- `ast_filecopy(old_filename, new_filename, "fmt1|fmt2|...")`

# APIs: checking voicemail

- `#include "asterisk/app.h"`
- `ast_app_has_voicemail("mailbox@context", "Old")`
- `ast_app_inboxcount("mailbox@context", int *new, int *old)`
- `ast_app_messagecount(context, mailbox, folder)`

# APIs: external programs

- `#include "asterisk/app.h"`
- `ast_safe_system("/sbin/reboot")`
- `ast_safe_fork(int stop_reaper)`  
`ast_safe_fork_cleanup()`

# APIs

- Questions?

# Asterisk Debugging

# Inline debugging

- `#include "asterisk/logger.h"`
- `ast_verb(verbose_level, "format", ...)`
- `ast_debug(debug_level, "format", ...)`
- `ast_log(LOG_LEVEL, "format", ...)`  
LOG\_ERROR, LOG\_WARNING, LOG\_NOTICE
- `ast_backtrace()`

# DEBUG\_THREADS

- “core show locks”
- Getting better with regard to deadlocks, but some may remain

# MALLOC\_DEBUG

- “memory show allocations”
- Memory usage doesn’t always follow the intended path; look for edge cases.
- If code uses utility functions for allocations, may want to use `__ast_calloc(a, b, filename, lineno, function)`

# External Debugging

- Common tools
- When to use GDB
- When to use Valgrind... “This isn’t possible.”

# GDB

- First things first... DONT\_OPTIMIZE
- Use -g flag or 'ulimit -c unlimited' to get core dump
- # gdb /usr/sbin/asterisk /tmp/core.12345

# GDB: Issue #12453: British Telecom line testing causes segfault on

- (gdb) bt  
#0 ast\_frame\_free (fr=0x0, cache=1) at frame.c:322  
#1 0x001f561e in ss\_thread (data=0x8f54ff8) at chan\_zap.c:6262  
#2 0x080fb08b in dummy\_start (data=0x8f566e8) at utils.c:865  
#3 0x00a3b2da in start\_thread () from /lib/libpthread.so.0

# GDB: Issue #12453: British Telecom line testing causes segfault on

```
for (;;) {
 struct ast_frame *f;
 res = ast_waitfor(chan, res);
 if (res <= 0) {
 ast_log(LOG_WARNING, "CID timed out waiting for ring. "
 "Exiting simple switch\n");
 ast_hangup(chan);
 return NULL;
 }
 f = ast_read(chan);
 ast_frfree(f);
 if (chan->_state == AST_STATE_RING ||
 chan->_state == AST_STATE_RINGING)
 break; /* Got ring */
}
```

# GDB: Issue #12453: British Telecom line testing causes segfault on

```
for (;;) {
 struct ast_frame *f;
 res = ast_waitfor(chan, res);
 if (res <= 0) {
 ast_log(LOG_WARNING, "CID timed out waiting for ring. "
 "Exiting simple switch\n");
 ast_hangup(chan);
 return NULL;
 }
 if (!(f = ast_read(chan))) {
 ast_hangup(chan);
 return NULL;
 }
 ast_frfree(f);
 if (chan->_state == AST_STATE_RING ||
 chan->_state == AST_STATE_RINGING)
 break; /* Got ring */
}
```

# Valgrind

- Again, DONT\_OPTIMIZE
- `bash# valgrind --log-file=asterisk /usr/sbin/asterisk -vvvvvc`  
...lots of screen output...  
`*CLI>`  
...Reproduce the action that caused the crash...  
`*CLI> quit`  
`bash# ls asterisk.*`  
`asterisk.12345`  
`bash# vim asterisk.12345`

# Valgrind: Issue #13462: “file convert file.ogg file.gsm” segfaults

```

==10049== Invalid free() / delete / delete[]
==10049== at 0x402237F: free (vg_replace_malloc.c:233)
==10049== by 0x80EC268: ast_closestream (file.c:826)
==10049== by 0x4D0D1CC: ??? (res_convert.c:137)
==10049== by 0x80B22A7: ast_cli_command (cli.c:1889)
==10049== by 0x8079CF2: consolehandler (asterisk.c:1515)
==10049== by 0x80805B8: main (asterisk.c:3436)
==10049== Address 0x6266400 is 0 bytes inside a block of size 536 free'd
==10049== at 0x402237F: free (vg_replace_malloc.c:233)
==10049== by 0x80EC8B4: ast_readfile (file.c:918)
==10049== by 0x4D0CF7A: ??? (res_convert.c:102)
==10049== by 0x80B22A7: ast_cli_command (cli.c:1889)
==10049== by 0x8079CF2: consolehandler (asterisk.c:1515)
==10049== by 0x80805B8: main (asterisk.c:3436)

```

# Valgrind: Issue #13462: “file convert file.ogg file.gsm” segfaults

```
if (!bfile || (fs = get_filestream(f, bfile)) == NULL || open_wrapper(fs)) {
 ast_log(LOG_WARNING, "Unable to open %s\n", fn);
 if (fs)
 ast_free(fs);
 if (bfile)
 fclose(bfile);
 ast_free(fn);
 break;
}
```

...

```
return fs;
```

# Valgrind: Issue #13462: “file convert file.ogg file.gsm” segfaults

```
if (!bfile || (fs = get_filestream(f, bfile)) == NULL || open_wrapper(fs)) {
 ast_log(LOG_WARNING, "Unable to open %s\n", fn);
 if (fs)
 ast_free(fs);
 fs = NULL;
 if (bfile)
 fclose(bfile);
 ast_free(fn);
 break;
}
```

...

```
return fs;
```

# Debugging

- Questions?