So You Want To Be An Asterisk Developer?

Kevin P. Fleming
Senior Software Engineer
Digium

June 15, 2005

Overview

- How to obtain, build and work with CVS
- Contributions and Licensing
- General module structure overview
- Types of modules
- Coding guidelines and Documentation
- IRC, Mailing Lists and Bug Tracker
- The Future

How to Obtain, Build and CVS

- Source modules and branches
- Operating system platforms
- Required and optional tools
- Required and optional libraries
- Debugging
- Use 'make update' to update a working copy

Source Modules and Branches

- asterisk
- asterisk-addons
- zaptel
- libpri

- v1-0
- HEAD
- v1-2 (soon)

```
$ cvs -d:pserver:anoncvs@cvs.digium.com:/usr/cvsroot login
$ cvs co asterisk
$ cvs co -rv1-0 -d asterisk-1-0 asterisk
...
$ make update
```

Operating System Platforms

- GNU/Linux (primary development)
- FreeBSD 5.x
- Solaris
- Mac OS X (Darwin)

Required Tools

- GNU C/C++ compilers, v3.x or higher
- GNU binary utilities (assembler and linker)
- GNU make, v3.80 or higher
- GNU bison

Optional Tools

- doxygen and graphviz
- ccache
- GNU debugger (gdb)
- valgrind
- GNU flex
- ctags and/or etags

Required Libraries

- OpenSSL (res_crypto)
- zlib (pbx_dundi)
- ncurses (editline)

Optional Libraries

- newt (zttool, astman)
- popt (smsq)
- curl (app_curl)
- nbs (chan_nbs)
- Voicetronix drivers (chan_vpb)
- OSP toolkit (res_osp and chan_sip)
- speex (codec_speex)
- Open H.323 (chan_h323)
- UnixODBC, Postgres, MySQL, Sqlite (v2), FreeTDS (config drivers, CDR modules)

Debugging

- Asterisk is heavily threaded
- Debugging is easiest post-mortem
- Threading models differ by platform

Building for Debugging

- use 'make dont-optimize' to build and install
- use 'make OPTIMIZE= K6OPT=' to build only
- various defines in the Asterisk Makefile:
  - DEBUG_THREADS
  - DO_CRASH
  - DETECT_DEADLOCKS
  - TRACE_FRAMES
  - MALLOC_DEBUG
- various defines in source modules
- ensure that 'debug' is listed for the console output channel in logger.conf

Core dumps and backtraces

- **always** include the -g parameter
- preserve binaries and sources
- use 'thread apply all bt full'
- use 'ast_grab_core'

Contributions and Licensing

- Patch creation and support
- Maintaining out-of-tree patches
- Licensing
- Short form disclaimer
- Long form disclaimer
- Copyright assignment (future)

Patch Creation

- always run 'make update'
- use 'cvs diff -u' (or add 'diff -u' to your .cvsrc file)
- **read** the patch before posting it
- watch for new/removed files

Out-of-tree patches

- You may choose to maintain your changes outside of the Digium-maintained Asterisk tree, which allows you to distribute them under any GPL-compatible license you wish to use
- Warning: keeping an out-of-tree patch up to date with the CVS HEAD branch can be hard work!
- Users of your patch may have limited support options

Licensing

- Asterisk, Zaptel and libpri are dual-licensed
  - they are distributed under the version 2 GNU General Public License
  - they are also distributed by Digium under a commercial license that does not require source code distribution
- Contributions **must** be disclaimed
- All code contributed to the tree **must** be original code, or **must** include a disclaimer from the original author, or **must** be licensed in a way that does not restrict Digium's distribution of the code in any way
- Changes contributed to Asterisk **cannot** be subject to any encumbrances, such as patents, exclusivity agreements or non-competition agreements

Licensing of your changes

- If you make changes to Asterisk, you are **not** required to distribute the source code for them (unless your distribute binaries containing the changed code, as the GPL mandates)
- Likewise, you are **not** required to contribute your changes to the Asterisk source tree, although we would prefer that you do so
- If your changes are accepted into Asterisk, they will be maintained and kept current with the rest of the Asterisk source by the community and/or Digium's developers

Asterisk GPL is permanent

- Contrary to occasional discussions on the Internet, the Asterisk source code, since it has been released under the GPL, will **always** be available under the GPL. That is a requirement of the GPL itself.
- Anyone can distribute the Asterisk source code as they see fit, although if they distribute a modified version Digium may require them to not use the 'Asterisk' name for it, since 'Asterisk' is a registered trademark owned by Digium.

Short Form Disclaimer

- http://www.digium.com/disclaim.changes
- This form disclaims all copyright to the submitted code, thereby placing it into the public domain and making it available for anyone to use for any purpose

Long Form Disclaimer

- http://www.digium.com/disclaimer.txt
- This form retains all copyrights by the original owner, but grants Digium an irrevocable and royalty-free license to use the code as it sees fit.
- The owner of the code can continue to distribute the code as they wish, under any license they choose to use, as long as it does not conflict with the license granted to Digium (i.e. they cannot grant an exclusive license to the code to any other party).

Copyright Assignment (future)

- This form will be similar to that used by the Free Software Foundation, which allows the contributor to assign their copyright interest in the contributed code directly to Digium.
- Code contributed under this disclaimer is automatically licensed back to the contributor, allowing them to distribute the code as long as that distribution does not conflict with Digium's ownership.
- Copyright assignment allows Digium to enforce the GPL licensing of your code on your behalf.

Module Structure Overview

- Mandatory headers and functions
- Use count tracking
- Registration/deregistration of commands and handlers

Mandatory Headers

- Modules should include system headers, followed by Asterisk headers. Asterisk headers should be included using double-quotes, rather than angle-brackets.
- Include “asterisk.h” first, and then use the ASTERISK_FILE_VERSION macro to embed a version tag into your source file that can be displayed by the 'show version files' CLI command
- Include “asterisk/module.h”
- Include other needed Asterisk headers

Mandatory Functions

- Prototypes are in module.h:
  - char *description(void);
  - char *key(void);
  - int load_module(void);
  - int unload_module(void);
  - int reload(void);
  - int usecount(void);
- description is used in the 'show modules' CLI command
- key is used by the loader; GPL versions of Asterisk will refuse to load non-GPL-licensed binary modules
- load_module and unload_module perform the named task, and can return a failure code if it could not be completed
- reload is called when the user issues a 'reload {module}.so' CLI command

Use Count Tracking

- Needed for dynamically loadable modules
- usecount function is called by the loader
- These macros implement basic usecount management:
  - STANDARD_LOCAL_USER
  - LOCAL_USER_DECL
  - LOCAL_USER_ADD
  - LOCAL_USER_REMOVE
  - STANDARD_HANGUP_LOCALUSERS
  - STANDARD_USECOUNT
- You are free to implement your own usecount management

Registration and Deregistration of Handlers

- Types of handlers that must be registered
  - CLI commands
  - Manager events
  - Dialplan applications
  - Dialplan functions
  - Dialplan switches
  - Codec translators
  - AGI commands
  - Call Detail Record (CDR) backends
  - Configuration engines (Realtime)
  - File formats (audio/video/image)
  - Indications/Countries
  - Verbose loggers
  - RTP-using protocols
- Register your handlers during load_module() and deregister them during unload_module()

Types of Modules

- Core
- Resource
- Application
- Channel
- Codec
- Format
- Function
- CDR

Core Modules

- These modules provide functionality required by every Asterisk instance.
- They are combined into the Asterisk binary, not built as loadable modules.

Resource Modules

- Provide functionality used by many modules but that are optional unless those modules are loaded.
- Typically provide external symbols that are directly accessed by other modules using run-time linking, but can also provide CLI commands and other related handlers.

Application Modules

- These modules provide dialplan applications and related CLI commands.
- These modules must keep use counts as channels enter and exit the application.

Channel Modules

- These modules provide 'channel technologies', methods to connect Asterisk with TDM, VOIP and other interfaces.
- Each module keeps a use count of channels currently alive using that technology.
- Each module provides one or more 'technology' structures that provide a set of method pointers used by the Asterisk core to dispatch frames and out-of-band operations to the channel module as required.

Codec Modules

- These modules provide translation services to convert audio frames to and from SLINEAR (raw 16-bit) format, which Asterisk uses internally.
- Many codecs keep internal state from frame to frame

Format Modules

- These modules provide methods to read, write and update disk files containing audio frames.

Function Modules

- These modules provide dialplan functions, used in the Asterisk expression parser.
- Most of these are built into a single loadable module, pbx_functions.so

CDR Modules

- These modules provide CDR backends which are engines to post CDRs out into storage systems (databases, text files or other systems).

Coding Guidelines and Documentation

- Formatting
- Structure
- Naming
- Compatibility, Portability and Obscurity
- Writing Documentation
- Building/Using Documentation

Coding Guidelines Basics

- doc/CODING-GUIDELINES
- watch the asterisk-cvs mailing list for changes
- meant to keep the source code base as consistently formatted and readable as possible
- send a patch first that changes **only** the formatting, then send a patch with your code changes

Code Formatting

- code generally follows the 'Linux kernel' style
- there are efforts under way to add the appropriate formatting instructions to each file for the major editors (vi and Emacs)
- do not add comments that provide no direct value (initials, dates, 'I did this', etc.)

Code Structure

- we prefer that functions be kept as short and succint as possible
- Make liberal use of the operators, statements and block types that C offers
  - don't use a tree of 'if' statements when a 'switch/case' block would be more appropriate
  - don't put operations on separate lines that can be combined into a single step, for example:
  - *ch = '\0';
    ch++;
  - *ch++ = '\0';

Naming of Functions

- Functions that are internal to a module can use any name you wish, except for names that start with an 'ast_' prefix
- Functions that are intended to become part of the API should be named starting with an 'ast_' prefix and the prototype **must** be added to the relevant header file in include/asterisk

Naming of Variables

- Global variables in modules should be named with the 'global_' prefix, and should have descriptive names (not single characters)
- Variables for internal use by functions should be descriptively named
- Variables exposed outside the module (which should be kept to a minimum) must be named with the 'ast_' prefix

Compatibility and Portability

- Keep in mind that Asterisk supports many operating systems and CPU architectures
- Changes should not use platform-specific or CPU-specific features unless they are mandatory

Obscurity

- Do not use obscure language, library or system features just for the sake of using them
- Use of obscure or unusual features is acceptable when a performance or maintainability benefit can be demonstrated

Writing Documentation

- Functions and structures in modules should be documented using Doxygen syntax
- New configuration options must be documented in the relevant sample configuration file
- Modules that use new configuration files must include a sample configuration file

Building/Using Documentation

- Ensure that your build system has doxygen and graphviz is installed; graphviz is used to generate call and reference graphs in the documentation
- 'make progdocs' in the Asterisk source directory
- Documentation will be generated in the doc/api directory

IRC, Mailing Lists and Bug Tracker

- IRC channel #asterisk-dev on Freenode
- asterisk-dev mailing list on lists.digium.com
- asterisk-cvs mailing list on lists.digium.com
- Bug Tracker at bugs.digium.com

The Future

- Asterisk Object Model
- Asynchronous Event System
- ???