# The Future of Asterisk

Kevin P. Fleming
Director of Software Technologies
Digium, Inc.
kpfleming@digium.com

## Summary

- Background
- History
- Recent Events
- Current Development
- Future Plans

## Where did we come from?

Created by Mark Spencer in 1999 to 'fill a need', as all open source projects begin.

“I was so excited the first time I got a phone call delivered through my PC using my own software.” - Mark Spencer

## Background

## Community Contribution

- ✗Over 400 contributors from around the globe
- ✗More than 2/3rds of new major features in each release contributed by community

## Digium Contribution

- ✗Development team has grown from 2 full-time developers in mid-2005 to 10 (and still growing!)
- ✗Asterisk core design and architecture improvements
- ✗Integration of third-party commercial products

## History

## Asterisk Release History

Version 0.1.0 – First Release
December 5th, 1999
Mark Spencer, Author
Released under GPL
Version 0.2.0
September 12th, 2002
Added Queues, Agents, MeetMe, Speex

## Asterisk Release History

Version 0.3.0

February 7th, 2003

Major SIP Improvements, G.729a, MGCP Support

Version 0.4.0

April 11th, 2003

IAX2, IAX2 Trunking, and Macros added

Version 0.5.0

September 9th, 2003

iLBC, SIP and IAX Video Support

## Asterisk Release History

Version 0.7.0

January 13th, 2004

SCCP, AES Encryption, ODBC Support for CDRs

Version 0.9.0

April 12th, 2004

CVS Branched as “Stable,” ADPCM, DSP Fixes

Version 1.0.0

September 23rd, 2004

More CDR Backends, GR-303, OSP, NFAS, G.726

## Asterisk Release History

Version 1.2.0

November 16th, 2005

Distributed Universal Number Discovery (DUNDi)

Asterisk Extension Language (AEL)

Realtime for SIP, IAX users/peers

Native / Internal Music on Hold

IAX2 Encryption

Q.SIG

FastAGI – AGI across TCP, like FastCGI

ODBC Voicemail Storage

## Asterisk Release History

Version 1.4.0

December 25th, 2006

Variable Length DTMF

T.38 FAX passthrough

Shared Line Appearance

Multithreaded IAX2

IMAP Voicemail Storage

Generic jitter buffer

Asterisk Extension Language (AEL2)

Jabber/Jingle/XMPP/Googletalk

## Development History

- ✗Bug/issue reports now average 8-10 per day
- ✗Code commits range from 60-150 per update release
- ✗Security issues handled by core team in 24-48 hours from initial report

## Recent Events

## AsteriskNOW

- ✗“Asterisk Software Appliance”
- ✗Based on rPath Linux and rBuilder
- ✗Tightly-focused distribution designed for building Asterisk servers

## Asterisk GUI Project

- ✗Simple, HTTP/AJAX based framework
- ✗Does not require any software outside of Asterisk and a web browser
- ✗Manages existing Asterisk configuration files

# Asterisk Project Security Advisories

- ✗Formal reporting of vulnerabilities and subsequent advisories
- ✗Coordination with other advisory reporting organizations and common advisory tracking numbers
- ✗Fully transparent reporting to enable end users to quickly understand vulnerability
- ✗Advisories posted and archived at http://www.asterisk.org/security

## Current Development

## Call Bridging

- ✗Flexible multi-channel bridging
- ✗Channels can be added and dropped, so calls instantly convert from two-party to multi-party
- ✗Will simplify and stabilize features like spying, whispering, in-call announcements, etc.

## RTP media streams

- ✗Performance, performance, performance!
- ✗Initial tests have shown 100-200% improvement in number of RTP media streams that a given server can handle
- ✗Will also reduce thread/memory footprint for handling large numbers of media streams

## Dialing

- Simplified but extensible internal dialing API
- Will make it easier to ensure that all applications that can dial channels have the same features/functionality
- For the first time, will expose more complex dialing features (acknowledgment, call forwarding) to AMI and spool files

## Codec (format) negotiation

- ✗Support for end-to-end negotiation of media stream formats
- ✗In-call media negotiation for stream format changes (mu-/A-law to T.38 for example)
- ✗Support for media stream 'attributes' required for video streams and complex voice codecs

## Asynchronous Events

- ✗New core infrastructure to handle events between Asterisk modules
- ✗Will eliminate 'polling' for applications like voicemail MWI
- ✗Can be extended across a cluster of Asterisk servers using DUNDi-like mechanism

## Call event logging

- Will allow complete tracking of 'events' that take place during a call
- Can support far more functionality than CDRs
- Will eventually support custom events from within the dialplan, allowing for audit trails to be created

## SS7 Support

- ✗SS7 support for trunking only (not applications)
- ✗Will support both ANSI and ETSI variants
- ✗Limited to a single Asterisk server servicing a point code

# IPv6 Support

- Developed by community members
- Supported across all Asterisk channel protocols and interfaces
- When the network supports IPv6, NAT problems will disappear (we hope)

## Future Plans

# Clustering and Failover

- ✗Research continues on infrastructure required
- ✗Goal is to be able to support active-active failover between a pair of Asterisk servers with minimal call disruption
- ✗Will require some extensive rework of internal data structures to allow synchronization between servers

## Separate signaling and media

- ✗Extend NFAS and SS7 support to allow for signaling links to live on servers without bearer channels, or servers to have bearer channels without signaling links
- ✗Allow Asterisk to be used as a media gateway platform from other softswitches

## Release Management

## The problem

- ✗Users want a stable product with new features frequently
- ✗Developers need new functionality to get adequate testing
- ✗Long release cycles mean no 'regular' users test new features, so bugs don't get found before release
- ✗Developers don't work with the released code as often

# Current Release Process

- ✗When developers decide, the development branch ('trunk') is copied (branched) into a release branch
- ✗An indeterminate period of time (at least a few months) passes during which attempts are made to test and stabilize the release branch
- ✗During this time, the development branch is 'frozen' except for minor fixes, thus halting further development
- ✗Once the release branch has been released, no new features are added to it

## New Release Process

- Development branch will always be in 'release candidate' mode; all changes made to this branch **should** be release-worthy and bug free
- Periodically, a release branch will be made from the development branch and quickly brought to a releasable state
- No changes will be made to this release branch except for security vulnerabilities and regressions found during testing of the branch
- The development branch continues to receive changes during this time

## An example release

- 2007-11-01: trunk branch copied to branch 1.6.0, and first 1.6.0 release candidate tagged and released as 1.6.0-rc1
- 2007-11-05: after a few days of testing and bug fixes being applied, 1.6.0-rc2 is released
- 2007-11-10: more testing/fixing time, and if no regressions are left to resolve, 1.6.0 is released
- 2007-11-11: trunk branch copied to branch 1.6.1, and process repeats
- (for illustration purposes only; these dates and release numbers are not real!)

## Benefits to Users

- New features become available within weeks or months of being developed, instead of one year or longer
- Developers and users are deploying and testing the same code base, thus enabling developers to more quickly find and fix bugs
- Support for new devices, interoperability changes and other interactions arrives more quickly

Thank You!