To: Ward Mundy
From: Jared Smith
Organization: Digium, Inc.
Date: Thu, 03 Apr 2008 10:26:11 -0400
Re: The Next Dinosaur

On Wed, 2008-04-02 at 09:09 -0400, Ward Mundy wrote:
I hope you'll pass along the attached article to those that might still be able to do something about the direction of Asterisk 1.6.

I've forwarded your comments onto several people inside of Digium, including those responsible for Asterisk development. I'm sure they'll have responses of their own, but let me take a minute to address some of the points in your article.

I think we can both agree that the feature set is an important part of any PBX system. Or, as you put it, ³It¹s the Feature Set, Stupid!²
There are two major reasons for moving from Asterisk 1.4 to the upcoming Asterisk 1.6 release at all, and the first one is features. Asterisk 1.6 brings a lot of new features to the table over what was available in Asterisk 1.4 and Asterisk 1.2. (The other big change in Asterisk 1.6 is that a lot of its internal plumbing got re-worked, so that it should be more efficient, more stable, and better able to handle larger call volumes.)

Unfortunately, your article doesn't differentiate between features of Asterisk and features that third-parties (yourself included) have bolted on to Asterisk. To use the same analogy that I gave when I met you in Charleston, we here at Digium want Asterisk to be the best engine in the world.
Whether you make that engine into a Formula One race car or a big brown delivery truck is up to you -- we're simply building the best engine we can. Now, we've gone and built a newer version of the engine ("More horsepower! Higher torque! Faster zero-to-sixty speed!"), and suddenly everyone complains that the starter motor doesn't fit in the same place that it used to. I know it's not a perfect analogy, but hopefully you get my point... While we believe Asterisk 1.6 is just about ready to be released, that doesn't mean that all the modules that PBX-in-a-Flash adds to Asterisk are ready for 1.6.

Asterisk is a living breathing animal, and as such, it's not simply a drop-in replacement for Asterisk 1.4. I realize you're not a software developer, so let me point out a few things that you might not be aware of. APIs change when major versions of the software are released. (APIs are Application Programming Interfaces -- think of them as building blocks inside of the Asterisk code that both Asterisk and third-party programs can use to do various things.) The problem is, when we make Asterisk better, we often have to change those APIs to do so. While we try very hard not to change those APIs between minor releases (such as 1.4.18 to 1.4.19), it's almost impossible not to have them change between major releases (1.4.x to 1.6.x, for example). In fact, I'd challenge you to find any major project that provides source-level API compatibility as a *guarantee* between major release versions. (Look at Apache 2.0 - 2.2, PHP 4 - 5, MySQL 4 - 5, PostgreSQL 7 - 8. They all have the same thing -- Major changes almost always require API changes.) When the Asterisk APIs stop changing from major release to major release, then Asterisk *WILL* be as dead as the dinosaurs are. As for now, Asterisk is not a dinosaur simply due to the fact that *it is still evolving*.

Luckily, Asterisk is an open-source project, which means that when Asterisk does evolve, that the changes aren't made in secret. Any third-party developer who wants to make sure his code remains compatible with the latest version of Asterisk can do so at any time. He doesn't have to wait until 1.6.0 is released to find out that his code will have to be changed to fit the new APIs. The Asterisk code is always available to test, play with, qualify against, etc. so that the developer can update their code to be compatible, so that when the time comes that real users want to use it, their applications will be ready.

We also have an Asterisk development mailing list and IRC channel where the Asterisk developers are always happy and eager to help third-party application developers keep up with the changes. (For various reasons I'll specify later, they often *can't* make the changes themselves.) The Asterisk developers are also happy and willing to take feedback from the community regarding changes. Again, Asterisk development doesn't happen in a vacuum -- all of this is open and available, if people will just avail themselves of those resources.

The next point I'd like to address is that of responsibility. Your article somehow assumes that it's the responsibility of the Asterisk developers to somehow know about all these third-party apps, and make sure they never break due to API changes. I can see three flaws with that argument -- first of all, there's no way the Asterisk developers could possibly know of every third-party application, it's state of affairs, and so forth. Even if they had a master list of all those apps, where the code was, and who to contact, there just aren't enough resources (both from a man-power and time standpoint) to even begin to tackle that sort of endeavor.

The second flaw is this -- even assuming for a moment that we could keep track of all the third-party apps and try to keep them up to date (which we both know isn't possible), licensing concerns would keep the Digium-paid Asterisk developers from doing so. We take copyright issues very seriously, which is why we ask the Digium-paid Asterisk developers not to look at any third-party code that hasn't been properly licensed to Digium (or is available under an OSI-approved license). I'm sure your experience with the court system of this country proves that we live in a litigious society, so I hope you can understand that Digium is going to do all it can to avoid any potential problems related to copyrights. In a nutshell, Asterisk developers *can't* be responsible for third-party code, as it creates too much potential for litigation down the road.

The third flaw to that argument is the point I made earlier... if Asterisk *were* to guarantee source-code API compatibility between major releases, there's no way possible that Asterisk could continue to grow, evolve, and adapt to the changing telephony market. I've been a part of the Asterisk community for almost six years now, and I'll be the first to admit that some of the growing pains we're all experiencing are due to some short-sightedness about how the world of telephony was going to change. But the only way to be able to compete is to be agile and flexible.

To sum things up so far, I guess what I'm trying to say is this: Digium (and its Asterisk developers) can't be responsible for making sure third-party apps stay current with the latest version of Asterisk. We're more than willing to help teach third-party developers how to keep their own code up to speed, but we can't be responsible for it.

(Besides, in the case of app_swift, we're not talking rocket science here. As I understand it, the program is only a few hundred lines of code, and shouldn't take much work to update to the 1.6 APIs.)

Anyway, that's the gist of message. The rest of the email below is just meant to address some of the other points raised by your article. I don't do this to try to start an argument... I simply feel that I've gotta stand up and defend the developer community when it's unfairly criticized.

"We defy you to find a link to any document that explains the transition from Asterisk 1.2 verbs to Asterisk 1.4 produced by the developers of the product."

You don't need to go any further than UPGRADE.txt and CHANGES files in the source code of Asterisk itself. They list every item that changed between the two releases, including any dialplan applications that have changed, any configuration file syntax changes, any changes to the Manager interface, etc. Besides that, many of the Asterisk developers

blogged about the transition from Asterisk 1.2 to Asterisk 1.4 at http://www.asterisk.org/blog/. There was also quite a bit of discussion on the asterisk-dev mailing list, and last but not least a number of presentations at events such as AstriCon and VON.

In fact, in Asterisk 1.6, we still provide instructions on moving from 1.0 to 1.2, from 1.2 to 1.4, and from 1.4 to 1.6.

"Asterisk 1.6 continues the programming carnage while adding some bells and whistles of its own: for example, an entirely new and different Asterisk Manager."

I wouldn't characterize the changes to the Asterisk Manager Interface as really new or radically different. While there was a call from various voices in the community for a completely new AMI interface, in the end it was decided to simply fix some of the most egregious problems in the current interface for now, and to make more radical changes after 1.6 has been released. We simply bumped the version number of AMI from 1.0 to 1.1 (so that application developers could programatically know which version of AMI they were talking to), and fixed a few of the commands to return their data in a more logical fashion. Each of the changes was documented (including the old behavior and the new behavior), along with examples. You'll find all this documented in doc/manager_1_1.txt in the Asterisk source as well. But no, it's not an entirely new and different Asterisk Manager... it's simply an update to the same old Manager interface.

For what it's worth, these changes were made due to input from the community, not from some sort of mandate within Digium itself. If you use and care about the Asterisk Manager Interface as much as I do, I highly recommend that you join the asterisk-dev mailing list and actively take part in shaping the future of AMI. After all, once Asterisk 1.6 is released, it's really too late to go back and say "Gee, that's not how I would have liked it." Any significant changes after that point will probably have to wait until the next major release.

You see, we're in an interesting position at Digium. When people come to us with new features or enhancements, we've got to walk a fine line. We can choose to ignore them or incorporate them (and make sure the changes are documented along the way). Either way, some people will complain... and there's no possible way I can keep all the people happy all of the time.

Last but not least, let's talk directly about your bug report. In it, you claim that "Lack of native support for either Flite or Cepstral TTS breaks thousands of existing text-to-speech Asterisk applications". Asterisk has never had native support for either Cepstral or Flite for text-to-speech, so I'm not sure how not having it in Asterisk 1.6 breaks anything. I'm afraid that if I were to follow your logic to its logical conclusion, it would be better to write that as "Since the developer that wrote app_swift won't update the code for Asterisk 1.6, it's Digium's responsibility to do so." Again, I've got to point out that Flite and app_swift are totally outside the control of the Asterisk development team.

You also questioned the attitude of the Asterisk developer who closed the bug. Let me respond to this by throwing out a hypothetical situation. Let's say that someone filed a bug report against PBX-in-a-Flash saying "My softphone doesn't work correctly with PBX-in-a-Flash, because it doesn't handle SIP signaling properly." I'm quite sure your response would be "I'm sorry, but that's not a problem with PBX-in-a-Flash, that's a problem with your soft phone. You'll need to contact the people who wrote your soft phone and try to get them to fix the problem.

The developer who closed the bug that you opened on the Digium bug tracker responded appropriately... to quote him, he said "This is clearly code that is not in Asterisk. Many of us cannot even look at the code, unless it has been disclaimed." He also explained that the bug

tracker shouldn't be used for feature requests, and that if the original authors aren't willing to update their code, then it's up to you to find someone else who will.

Since my job is community relations, let me talk for a minute about how feature requests *should* be handled in the Asterisk community. Ideally, there would be some initial discussion on one of the many Asterisk mailing lists. This gives everyone who is interested a chance to share their opinion, discuss the architecture of the solution, and so forth. At that time, you've got three options... scratch the itch yourself (if you're a developer), convince one of the other Asterisk developers that it's really their itch as well, or pay someone else to scratch the itch for you.

Speaking personally for a second (and not speaking on behalf of Digium), I'd *love* to see a well-written TTS interface for Asterisk, so that you could plug in a variety of TTS engines. The problem is, I'm not a good enough coder to scratch that itch myself. I hope you can appreciate the fact that we can't simply distribute code that hasn't been licensed to us. So there's no easy solution, at least in the short term. In the long term I'm sure we'll come up with some sort of solution, but I personally don't know what that is.

I know this email has been long and rambling, but I hope you understand where I'm coming from, and that we can continue to communicate better before things escalate to this level again. I still believe that we can work together to bridge some of the divide that exists between the Asterisk community and the PBX-in-a-Flash and FreePBX communities, but you guys have to show that you're willing to meet me halfway and actually communicate your concerns, questions, and fears to us before they escalate to this level. As always, I'm more than willing to make myself available over the phone, over email, or in person if necessary to answer any questions or concerns you have. I respect you for the time and effort you donate to the community, and honest hope that we can all work together towards a common goal.

--
Jared Smith
Community Relations Manager
Digium, Inc.