EXHIBIT 17

UNREDACTED VERSION OF DOCUMENT SOUGHT TO BE LODGED UNDER SEAL

From: Mike Vernal </O=THEFACEBOOK/OU=FIRST ADMINISTRATIVE GROUP/CN=RECIPIENTS/CN=MVERNAL>
Sent: Saturday, June 29, 2013 12:41 PM
To: Douglas Purdy
Subject: Re: Thoughts On Platform & Community

100%

Two small comments:

1. Don't think we're building recommendations this year

2. Should we move to 1 year breaking change?

On Jun 28, 2013, at 9:05 PM, "Douglas Purdy" <dmp@fb.com> wrote:

Would be interesting for you to read this thread and see if we are thinking about things roughly the same in this area?

From: Douglas Purdy <dmp@fb.com>
Date: Friday, June 28, 2013 7:36 PM
To: Justin Osofsky <josofsky@fb.com>, David Weekly <dew@fb.com>
Subject: Re: Thoughts On Platform & Community

Thanks for sharing.

I'll attempt to address some of the product issues here.

1. I think it is important to understand that the social part of platform is not solely a developer platform, unlikely some of the comps below. Unlike Parse, etc., the social part of platform (identity and OG) are both user and developer products. This creates a dynamic where we are have to effectively create friction in the developer experience in order to address user trust/perception of Platform. If there is one audience that dislikes Platform more than developers, it is users. And that is actually a more meaningful audience because if we lose user trust, we die and there is no developer ecosystem. Many of the things we have put in place like reviewing OG actions, etc. are to address this user trust issue. BTW: I disagree that unified review is going to "brutalize" NPS. I think it will raise it. Why? Because it combines our disparate review processes and most importantly will serve as a gate on policy enforcement. Today, a developer builds an app, releases and then we may shut them off if they violate our policies. This unified review process will front load that and ensure that developers (and most importantly) users are not impacted by apps violating our policies.

2. Breaking Stuff. I won't bore you you with how long it took to get a breaking change process in place, much less publishing those changes to developers. We have seen the platform stability number in our survey's go up dramatically as a result of those steps. There is more we can do here (more on that in a minute), but the primary issue here isn't breaking changes, it is actually how we change the user side of the product. This goes back to the dual nature of the social part of Platform. As we change how we surface developer content, that will lead to changes in distribution to developers. As free distribution is the primary reason that developers use this part of Platform, they get super frustrated when that changes unexpectedly, particularly when we give no notice and or guidance on how they claw back to the previous state. There are steps we can take to help here, but I don't expect this to ever change as a cause for frustration, as the primary driver of user features is user engagement/happiness, not developer stability in terms of distribution. Because of this tension, we are spending much more time focused on building and selling paid developer products that have stable distribution/value as the sole goal. In terms of

API changes, we are moving to quarterly API releases, including both new features and breaking changes. Further, if you build with a native SDK, we **are** versioning our API in lockstep with the SDK and these versions are supported for 18-24 months.

3. Platform 3.0. I don't think we are calibrated on what Platform 3.0 is. We are not going to remove APIs when we launch this, we are actually adding two new APIs: social context and recommendations. Once we have those APIs nailed down, we will move to drain down the use of the friends permissions. The best way to think about this is the REST API -> Graph API transition. We still support the REST API 3 years after the Graph was announced, but we no longer, after a long, long drawdown time, allow new apps to use the REST API. Net, I don't think this is going to brutalize NPS. If we do this right, this could be viewed as a positive, like the Graph API was.

4. Supporting developers. I haven't been happy at our progress here. One of the key issue here is "who owns it". There is your own team, David, that owned StackOverflow. I personally negotiated the deal with Joel and I have not seen it come close to the potential it should have had. There is Zhen's team, which handles bugs (BTW: I think you being a little unfair in your characterization of how we handle bugs. There is literally no one triaging bugs a few years ago and we are now light years ahead), but we have to get better at actually fixing the bugs. Which brings me to the last team, developer support, who supports the major partners. So there are three different team supporting developers and all failing in some way. One of the ways to address this is to nominate a single leader to be responsible for this entire area and then give them the team and room to forward. That is one of the main goals of this most recent reorg. We are moving all developer support functions into Justin's team. If you are really passionate about this area, we have a leadership role that will manage the global developer support team that is completely and absolutely focused on this problem in a scalable way. I don't know if you and Justin discussed this at all, but it is something to consider. Side note: when I got here, no one was triaging bugs. We had 5000 open and untouched bugs. I started triaging them with one other person at 5pm everyday for months. I think it is a hell of a way to add value to even more than talking to them, leading me to the next point.

5. Talking to developers. We talk to developers a lot. We have a huge partnership team that talks to developers, large and small. We have a huge sales team that talks to developers. All of the leaders on Platform, Mike included, spends as much time as we can talking with developers. The recurring theme that we hear is "the value is too little for the investment I am putting in" (for the social part of Platform). That and the user trust/value are the primary things that we are working on in Platform product in H2. That is why we built Neko (predictable, low friction distribution), it is the reason we are building Pay3.0, to increase transaction conversion, it is the reason we bought Parse (building x-platform games/app is hard), it is the reason that we are building reengagement ads, etc. In terms of being the "voice of developers" (in and out) in the company, my expectation is that this is the role of partnerships team (partner engineering and developer support/ops).

5. Developer NPS. This remains a focus for us in Platform product in H2, but not as high of a priority as user trust (which is an existential threat to FB and Platform) or developer value (new products that deliver on consistent developer value -- which I can argue, this underlying cause of poor NPS -- even if you look at the developer we talk to daily -- the NPS is negative — -4 actually). Concretely, we are going to focus on four things: a "Parse quality" doc set (Zhen is leading), a "Parse quality" sample story (I am personally leading this effort), simple and transparent policies and app review (Monika & Constantin driving) and developer support for bugs and direct support (led by Ellen in Justin's team).

Hope that helps.

On Jun 28, 2013, at 3:46 PM, "Justin Osofsky" <josofsky@fb.com> wrote:

> + Doug)I want to take the time to digest this fully, but it looks like there are a lot of thoughts below which are relevant for Doug as well)
>
> ---
>
> **From:** David Weekly <dew@fb.com>
> **Date:** Friday, June 28, 2013 3:40 PM
> **To:** Justin Osofsky <josofsky@fb.com>
> **Subject:** Thoughts On Platform & Community
>
> Justin,

I really appreciate your spending the time this afternoon to talk about developers, community, and how I could help best move the needle for Facebook in the coming half. I loved seeing your concern for developers and the Parse acquisition, as well as the candid acknowledgement of the holes in our strategy.

So in looking at Platform's H1 review, I'd ask: What have we learned from the last few years of awful developer NPS? Given that Bret has moved on, how can we champion developer interest at Facebook?

While there's much we can learn from Parse, the biggest meta is right in our face: go talk to developers, like Ilya does, and ask them what they don't like. In that light, here are the top reasons I've seen for dissatisfaction in the past few months:

#1 "You break my stuff."
We give API deprecations with 90 day notices. We don't version our APIs and we don't even prep developers new to our platform to the fact that developing against Facebook is an *ongoing commitment*. This is not the same as almost any other "API" or "Platform" on the planet. (e.g. iOS 5 APIs are iOS 5 APIs forever.) Consequently, emails with the subject line "Operation: Developer Love" are guaranteed to give every FB developer the chills because they know the body of the message will communicate an unexpected new pile of work to throw a schedule.

#2 "You don't care."
Developers lack access to community & resources to share best practices, coupled with evident favoritism. How should I actually be using Open Graph (this week)? Or Notifications? We are obvious in showering undue love upon our strategic partners, visibly giving them private access to APIs that are mysteriously gated (n.b. Ads API, FBX, Buy Now) and secret roadmap guidance. But for an unmanaged developer, there is no effective forum to ask a question, no phone number to call, no IRC channel, no place to go. You can file a bug but it will get auto-triaged to closed because you're a 1MAU app when you're just getting started. You have to "get lucky" and happen to know someone who works at Facebook or gone to a rare event where you've actually gotten a chance to meet a Developer Advocate in person and gotten their contact info. (I was vigorously discouraged from carrying business cards when I joined - "people might actually email you!")

#3 "It isn't fun."
Getting started with the APIs is a huge chore. The iOS SDK isn't even signed. It takes days to get up and running with a half-decent integration and Open Graph custom action approvals add even more friction. Unified Review is only going to increase this friction while Platform 3.0 reduces the utility of the APIs - the combination of which mean that Facebook integrations are going to be even less fun by the end of the year. Why do people love Parse? (Or Twilio, node.js, Rails, or MySQL?) Because you can be up and running super fast and it gives you this exciting tingle of having made something useful, quickly. Eddie has a passion to deliver this, we need to give him the resources. Using Facebook is rapidly becoming Enterprise JavaBeans: an effective but ultimately dull way to address technical issues but not a joy, not sexy, and not the future.

Our H2 goals for Platform don't address any of these concerns, or the one area where we whiffed hard on our H1 goals: NPS (and growing our Quality developer base). Our 10mau developer base has been flat at ~375k for the last year as has our NPS. Unified Review & Platform 3.0 are going to brutalize both numbers and yet we are magically expecting to grow penetration in the top 400 app charts, despite the fact that these charts have high churn and those surprise-hit titles often come from the very sort of small developers we're failing to connect with.

You're right that we'll need a Product answer to the above in conjunction with outreach. And you're right that it's a very cross-functional role. In the few days alone I've had conversations with Eugene, Monika, Amanda, Allison, Constantin, and more to ensure we're building a developer-friendly platform. I honestly think a small team tasked with building a scaled global developer community and the products needed in-house to deliver it would be awesome. It'd be bigger than one person and candidly I think a team with myself, Bear, and Ravi could make this happen.

I'd love your thoughts.

Yours,
David