EXHIBIT 168

UNREDACTED VERSION OF DOCUMENT SOUGHT TO BE LODGED UNDER SEAL

From: Charles Jolley </O=THEFACEBOOK/OU=EXCHANGE ADMINISTRATIVE GROUP (FYDIBOHF23SPDLT)/CN=RECIPIENTS/CN=CHARLESJ>
Sent: Thursday, November 22, 2012 12:49 AM
To: Mike Vernal
Cc: Rose Yao; Douglas Purdy; Vladimir Fedorov; Rohit Wad; Alex Himel; Ash Wahi
Subject: Re: Action Importer Spec

+1. Can we also get an endpoint I can hit on Facebook to ask FB to repull my feed? This would allow a developer to potentially just publish a feed and ping FB when it needs to update. That is a much more reliable way to publish if you aren't ready to do the asynchronous tier.

Also, one other benefit of this is that it could make configuring your open graph implementation and even reviewing the implementation easier for us since we can pull in historical data.

Imagine for example if the on boarding process for building an og integration involved supplying us with a Json file or feed URL with historical content that you want to publish. Ten you can design your collections and news stories using real data from this feed. Much more tangible.

Sent from my iPad

On Nov 21, 2012, at 12:53 PM, "Mike Vernal" <vernal@fb.com> wrote:

> I thought about this more (requiring partners to implement a feed endpoint vs. calling an API).
>
> *tl;dr - I think we need to require apps to implement an API endpoint for us to hit for historical data, but still strongly encourage apps to use the OG API for real-time publishing.*
>
> **Background**
>
> Why did we start investigating Action Importers in the first place? We found that there was this leverage imbalance between us and our partners. We exposed our value via an API that partners could call at will, with great flexibility. We gave them access to our crown jewels, and they had complete control over how to integrate it. On the flip side, if we wanted something in return -- some value from the partner -- we basically had to cajole them into giving it to us. We had to create a contract, get them to promise to integrate with us, and then send our BD + Pops team after them to integrate it. It was easy for partners to promise to do something, but then fall back on schedule priorities, bugs, etc. as excuses for why something wasn't done.
>
> Case and point -- Spotify happily agreed to integrate our subscriptions at a 10% rev share last year when we signed the deal. We still don't have them integrated, and they're (subtly) dragging their heels about integrating it. We can yell at them about this, and we can threaten to pull the plug, but they just keep tell us they're working on it and they need more time. For the sake of the relationship, we give them more time.
>
> Basically, we're in a position of low leverage.

The theory behind Action Importers was that we needed to balance the leverage. You can call our APIs and access our data, as long as we can call your APIs (if you have them) or crawl your web site (if not) and access your data.

It's one thing to drag your heels, but if we're the ones doing the work then we force you to make a decision -- either you allow us access to your data, or you block us. If you block us, then it's really easy/straightforward for us to decide to block you.

That's the basic theory/strategy behind action importers - to require partners to give us an API, so we're both exposing APIs to each other and can access each others' data.

What's Changed?

When we first starting discussing this, we were talking about doing this only for top partners. I think a lot of folks interpreted this as just a negotiation tactic -- we'd just threaten to do this if they didn't cooperate. I think this is part of the reason we haven't made much progress here - every time we talk about this, someone says "oh, well the partner is just going to shut us down, so why are we wasting our time on this." (To be clear, I never thought this was "just a negotiation tactic," but that perception seemed pretty widespread.)

What's changed between then and now is that this is now very clearly not a negotiation tactic -- this is _literally_ the strategy for the read-side of our platform. Our strategy is basically -- you can access our social graph, as long as we can access your social actions (both with user consent).

Pull (Feeds) vs. Push (API) for Historical Data

I think we all agree there should be a way to import all your activity from an app form Timeline.

There are two ways to implement this -- you can either require the app to implement some endpoint we can hit (pull-based approach, or a feed), or you can ping the app when the user turns this on, and require the app publish all the back actions to you via our API (the push-based approach).

I'm pretty strongly convinced that, for historical data, this has to be a feed-based approach. Reasons:

- Technical Reliability -- when a user turns this on, we may be importing hundreds or thousands of actions. Transferring all this data might take minutes (maybe even hours, worst case). For partners, they'll need to build a separate service (or something like the async tier) to be able to reliably do this -- they won't just be able to do this in callback handler when we ping them. This is a bunch of complexity. On our side, if they try to write to us too quickly, it's possible that they'll overwhelm one of our UDBs and cause us to start erroring out. Figuring out this failed and making sure the partner retries this is going to be hugely complex for each partner, etc.

Worse, what if their background syncing service crashes for a few hours, or a day. We'll need them to be make it robust to these kinds of failures by replaying all those actions once it actually comes back up. They'll need to build a message queue or a transaction log. This adds more and more complexity.

A feed-based approach is technically much simpler. They have to write an endpoint that reads their actions out of their DB and returns them as a JSON blob. We can build the retry logic if things fail. We can implement rate-limiting/throttling once and have it work for all partners.

Net, I think the feed approach will clearly be way more reliable.

- **Auditability** - it needs to be really easy for us to verify that we're actually getting the data we're supposed to be getting. With a feed approach, I think it's easy -- you look at the data in the endpoint, you look at the site, and if something is missing you flag it. You can always go back to the feed as the authoritative source of truth, and it would be really easy for us to build a tool for POPS to help them verify/audit this stuff.

With the push API, I think it's just unnecessarily more complicated. We'd need to do something where we poke them, get them to re-publish all their historical actions, and then look at those. What we'd probably do is build a tool that has a test account, drops all the actions from the app, pokes it to re-push them, and then looks at the apps activity log. We could do that, but it's just so much more complex than being able to look at an URL in a browser and see the list of all your actions.

Net, both are auditable but the feed approach is clearly easier to audit (especially for real users).

- Leverage - the feed model clearly has more leverage. Either the feed is there and has all the data, or it doesn't. If it doesn't, it's really black-and-white and is easy for us to turn off our endpoint until its fixed. We can integrate it as we see fit, etc.

The push model is just much worse here. If they're not pushing data to us, first we need to detect it. then we need to escalate to them, but it's then on them to fix and publish the actions they missed. It would be easy for a partner to selectively break this, cause a bunch of actions to get dropped, and make this feel like a busted experience to users. (If you don't think partners would do this, look at what we did with the Contacts API for Apple). They could do this with the feed approach as well, but it's much harder to justify/defend.

Net, net -- it's pretty clear to me that the feed-based approach is way, way better for importing historical actions.

Pull (Feeds) vs. Push (API) for Current/Future Data

On the flip side, it's pretty clear that the API wins for future data. Any pull model will have some latency, and that creates a crappy user experience (you take a photo on instragram, but it doesn't show up on facebook for 10 hours). You can fix this by having some ping/fat-ping model (like we do with our own push service), but that adds complexity.

If we had never launched an API, I think you could make an argument for only doing feeds + fat-pings. Since we already have the API, I think it would be insane to deprecate it and replace it with a more complex system (fat pings).

So this suggests to me that we should definitely keep the API for actions that are published as you do them.

Integrating The Two

Anyway, here's probably what I would do:
- Use the feed for historical data (pre-Connecting)
- Use an API for actions after Connecting
- Have some background task that probably re-pulls the feed once a month (in case any of the API publishes fail)

Sorry for the long note, but I feel really strongly that we should have a feed-based approach for historical data, and so I want to make sure everyone understands why. If you disagree w/ my logic, please raise your concerns/disagreements.

-mike

From: Mike Vernal <vernal@fb.com>
Date: Wed, 21 Nov 2012 10:37:09 -0800
To: Rose Yao <rose.yao@fb.com>
Cc: Charles Jolley <charlesj@fb.com>, Rohit Wad <rohitw@fb.com>, Alex Himel <ahimel@fb.com>, Ash Wahi <ashwahi@fb.com>
Subject: Re: Action Importer Spec

I'm not sure I agree (that's not why we just used feeds). I think this is part of the confusion about this stuff. I very explicitly wanted a feed architecture for inspect ability reasons. This is also why I've been asking for a spec / conversation about this.

-mike

On Nov 21, 2012, at 10:33 AM, "Rose Yao" <rose.yao@fb.com> wrote:

The only reason we used feeds was for developers with existing apis. Also the original premise for the project was that we will do this using the feed apis even if the developer didn't implement OG. It is significantly easier if the developer just sent us OG actions.

From: Mike Vernal <vernal@fb.com>
Date: Wednesday, November 21, 2012 10:20 AM
To: Charles Jolley <charlesj@fb.com>
Cc: Rose Yao <rose.yao@fb.com>, Rohit Wad <rohitw@fb.com>, Alex Himel <ahimel@fb.com>, Ash Wahi <ashwahi@fb.com>
Subject: Re: Action Importer Spec

I don't think we want to replace the way Open Graph. I just want a mechanism for historical importing. Feeds have all kinds of drawbacks (latency, etc.).

But I'll think about how these two relate, as it is a little crazy to have double implement push and pull. Anyone else thought about this?

On Nov 21, 2012, at 9:52 AM, "Charles Jolley" <charlesj@fb.com> wrote:

OK. so one more question - is the idea that this model might actually replace the current open graph API for how a developer should publish content to Facebook?

Frankly, as a developer just writing a feed endpoint for Facebook seems like a much simpler task than having to modify a bunch of different call sites in my app to proactively post content to Facebook.

On Nov 21, 2012, at 9:45 AM, Rose Yao <rose.yao@fb.com> wrote:

This doc is old and doesn't reflect the latest thinking on this. Reading through the latest stuff from Mark, the partner also need to feature to ability to sync with FB prominently on their side.

From: Charles Jolley <charlesj@fb.com>
Date: Wednesday, November 21, 2012 9:41 AM
To: Rose Yao <rose.yao@fb.com>
Cc: Rohit Wad <rohitw@fb.com>, Mike Vernal <vernal@fb.com>, Alex Himel <ahimel@fb.com>, Ash Wahi <ashwahi@fb.com>
Subject: Re: Action Importer Spec

Rose & Mike:

Rose's one pager has this to say about reciprocity:

> If a partner use the Facebook API to get data about a Facebook user, they must allow Facebook to also use their APIs to get data for their users who are on Facebook (with user consent). If the partner has an API, we will use that mechanism. We also reserve the right to crawl the partner website for the user's data. Partners cannot blacklist or block Facebook from crawling your site or using the API. If they do, Facebook reserves the right to block the partner from using our APIs.
>
> (If there are load balancing or technical issues, Facebook will work with partner to resolve them

--

Does this mean that for a developer to meet our reciprocity requirement they need to simply be willing to let us access their data if we ask for it? Or do they need to actively publish their content to Facebook to quality?

i.e. This paragraph implies that as long as the partner is willing to expose their content to Facebook, it is our problem to actually pull that data in. vs making it a requirement that they actively publish it?

On Nov 21, 2012, at 9:10 AM, Rose Yao <rose.yao@fb.com> wrote:

Attached are the walkthrough for how action importers would work with a partner we used instagram at the time.

Tech spec here for ideal scenario:
https://docs.fb.com/writer/index.do?docId=18589000000031005

Original draft of reciprocity one pager that I did for the BD team

Rose

On 11/21/12 8:26 AM, "Charles Jolley" <charlesj@fb.com> wrote:

Screenshots would be great! Anything you have would help. Thank you.

On Nov 21, 2012, at 6:25 AM, "Rohit Wad" <rohitw@fb.com> wrote:

Yes.

As background, we want to alleviate the following cases:
1. Partner has implemented, but isn't selling sharing / OG on their site well. We'd like the ability to upsell this from Facebook, and have the ability to bulk import.
2. In addition to the above, we'd like the ability to do this on an ongoing basis. (We need to specify how this will work, push/pull/etc.)

Separately from the above, we also need to figure out how to get partners
to provide good metadata. Many partners (e.g. Kindle, Pinterest) provide
only enough data to generate a story. We'd like the data to be more detailed. I don't know if this falls within the concepts of data reciprocity though. Mike?

Re. specs/PPT, unfortunately we don't have these. However, perhaps screenshots of the (already implemented) flow would help?

On 11/21/12 6:03 AM, "Mike Vernal" <vernal@fb.com> wrote:

Yes

On Nov 21, 2012, at 12:11 AM, "Charles Jolley" <charlesj@fb.com> wrote:

Ok we'll just to confirm my understanding then: the principle here is that the app would provide us with some end points following some protocol we define that would allow us to effectively import in bulk the
open graph objects and actions for a particular user. Is that right?

Sent from my iPad

On Nov 20, 2012, at 10:39 PM, "Mike Vernal" <vernal@fb.com> wrote:

+Rohit

I'm not sure it exists yet (though I have a model in my head, if no one
else has written anything down).

-----Original Message-----
From: Charles Jolley <charlesj@fb.com>
Date: Tue, 20 Nov 2012 20:40:02 -0800
To: Rose Yao <rose.yao@fb.com>, Alex Himel <ahimel@fb.com>
Cc: Mike Vernal <vernal@fb.com>
Subject: Action Importer Spec

action importer spec. can I have it?

Any slides you can send me explaining it would be helpful too. I am prepping for a Mark review of Platform v3.0 on Monday.

Thanks!
-Charles

<instagramtimeline.pdf><reciprocity.pdf>