**Server API Design To Support External Devices**

Feb 23, 2013

## General Design

* All API calls to the server are REST based, and all parameters are JSON.
* Beyond HTTP return codes, (eg 404 and 500), all HTTP successful calls return an application status code and, if necessary, a translated and displayable description of status code
  + status code == 0 means no error
  + status code <> 0 means an error, plus an optional translated description
* Server copy of data is always master, superseding what may be on device.
* User must be authenticated (log in) before using app, although there may be a way to view tutorials/help screens without logging in.
* If a translation is ever missing, the English translation is always present and used instead.
* Clients identify their version number in each API call. We only change the version number for a breaking API change. Versions numbers can be compared numerically.
* As much as possible, non-presentation elements are controlled and defined by the server; presentation is defined by the device

## Community Screen APIs

* Get later events for user=me\_id since ARRAY(my group\_id + later\_event\_id)  
  HTTP GET api/v1/events?token=<token>?aev1=1234,2345&aev2=1235,2346
  + Get events after later\_event\_id
  + Event stream is a sequence of different event types, but each is sequenced by an group id + event\_id
    - Community event
      * plus a flag for prayer requested events, indicating whether or not the user has already pledged to pray
    - Prayer pledge count update event
      * Event\_id
      * New count (integer)
    - New Group Friend / Update Group Friend event (needed for cache purposes only)
      * Group\_id
      * Friend\_id
      * Name
      * Photo
    - Delete Group Friend event (needed for cache purposes only)
      * Group id
      * Friend id
  + *Need way to handle too many events from this call*. *Perhaps do count before sending off to client, and only send the most recent N events.*
  + *Need way to combine separate event streams into one stream ordered by time. Perhaps order result set by event timestamp – for multiple servers this requires keeping the clock between servers mostly in sync.*
* Pledge to pray by=me\_id for event=group\_id,event\_id  
  HTTP PUT api/v1/pledge/1234,3456?token=<token>
* Get earlier events for user=me\_id before ARRAY(my group\_id + before\_event\_id) max\_number=n  
  HTTP GET api/v1/events?token=token?bev1=1234,2345&bev2=1235,2346&max=30
  + Get events before the given before\_event\_id. This is needed to support the user scrolling to the earliest event and then wanting to see even earlier events. *B feature, not necessary for first version.*

## Friends Screen APIs

* Get friend list for user=me  
  HTTP GET api/v1/friends?token=<token>
  + *Returns all my groups and my friends in each*
  + *Called when coming to this screen and (no data or it has been a while since last sync)*
  + *Note that a friend of a fellow group member is not also my friend.*
* // Get latest events for user=me\_id and friend=friend\_id and max\_number=n
  + Not necessary, instead this can be filtered from the current event list, which is added to the front of the per user event list cache.
* Create event by user=me\_id with friend=friend\_id and activity=activity\_id  
  HTTP POST api/v1/event  
  token=<token>  
  group=group\_id  
  friend=friend\_id  
  activity=activity\_id  
  prayer\_requested=true|false
* Add friend in group for=me first\_name=fn last\_name=ln photo=ph email=em mobile phone=mp  
  HTTP POST api/v1/friend/group\_id  
  token=<token>  
  group=group\_id  
  first\_name=fn  
  last\_name=ln  
  photo=ph (encoded)  
  email=em  
  mobile=mp
* Delete friend in group by=me\_id group=group\_id friend=friend\_id  
  HTTP DELETE api/v1/friend/group\_id/friend\_id?token=<token>
* Get activities by me=me\_id for threshold=threshold\_enum  
  HTTP GET api/v1/activities?token=<token>&threshold=threshold\_enum
* Get earlier events for user=me\_id and friend=friend\_id and group=group\_id and before\_event\_id) max\_number=n  
  HTTP GET api/v1/events?token=<token>&group=grp\_id&friend=friend\_id&bev=1234,2345&max=n
  + Get events before the given before\_event\_id. This is needed to support the user scrolling to the earliest event and then wanting to see even earlier events. *B feature, not necessary for first version*

## Groups Screen APIs

* Get groups for=me\_id *(including users for each group)*HTTP GET api/v1/groups?token=<token>
  + *Called when coming to this screen and (no data or it has been a while since last sync).*
* Get users in group=group\_id and me=me\_id  
  HTTP GET api/v1/group\_users/group\_id?token=<token>
* Remove user=user\_id from group=group\_id and me=me\_id  
  HTTP DELETE api/v1/user/group\_id/user\_id?token=<token>
  + me\_id must be the group leader
  + will notify users
  + *Note that a group leader cannot delete another user’s friend.*
* Create group me=me\_id and name=name

HTTP POST api/v1/group?token=<token>&name=name

* + Returns group id
  + Not already a part of max number of groups
* Delete group me =me\_id group=group\_id  
  HTTP DELETE api/v1/group/group\_id?token=<token>
  + Me\_id must be group leader
  + Will notify any remaining users in group
* Invite to group=group\_id me=me\_id and email=em  
  HTTP POST api/v1/invite/group\_id?token=<token>&email=em
  + Me\_id must be group leader
  + Email not already in group’s users
  + Not already max number of users already in group or invited to group
* Invite to group=group\_id me=me\_id and phone=ph  
  HTTP POST api/v1/invite/group\_id?token=<token>&phone=ph
  + Me\_id must be group leader
  + Phone number not already in group’s users
  + Not already max number of users already in group or invited to group

## Activity Notes

* *Note: activities are created on the server, with translation*s. *They can be very broad or very specific. It is expected that activities have the following fields*
  + *Title (shown on the relevant activity list after selecting What’s Next)*
  + *Full description (a paragraph)*
  + *Summary (one line, shown in the event feed on the Community screen)*
  + *Category (possibly)*

*Each of these can use a templating language, for example*

* + *$u = user first name*
  + *$f = friend’s first name*

## Security Notes

* Server is HTTPS.
* Use ProGuard for Android app.
* Authentication is a concern: Even if this is not a financial app, how to know that the client is who they say they are? At start of session, user must authenticate themselves with username (known) / password (secret). The server then replies with a token that the client provides with every subsequent API call. The token uniquely and securely identifies the user. At some point the token expires, and the user must re-authenticate. To make this usable, the token will be stored in the saved state when an app goes into the background, which is less secure, but still necessary. See 3 legged OAuth or Facebook examples.
  + *How long should a session last? Would it be a day, a week, a month or a year? We could experiment, and the time would be set by the server.*
  + Server API: authenticate username=u password=p  
    HTTP POST authenticate?username=u&password=p
    - Returns authentication token
  + Server API: each call, except authenticate, can fail with an authentication failure, meaning that the user must reenter their password
  + Admin: need way to suspend a user account (e.g., user reports stolen phone)