Instructions for smartphones:

I'm terribly sorry I did not send this to you guys weeks ago, but as you know with programming things take longer than initially expected. However, the site is online at this point and for the most part bug free. (visit hangchillparty.com)

Here is how you make GET or POST requests of the server. All requests are formatted like so.

www.hangchillparty.com?action=(whatever the action is).

If additional GET data is required, it will be tacked on to the end of the url like so:

www.hangchillparty.com?action=requestFriend&receiverID=1;

If additional POST data is required, it will be included in typical POST fashion (not on the url itself).

All requests come back as JSON objects, for example:

{"AlertCount": [

{

"co": "0"

},

],

"NewsItems": [],

"NewFriendships": [],

"FriendRequests": []

}

If you can't tell, the pattern is OBJECT, then ARRAY, then OBJECT, then ARRAY, etc, with objects being wrapped in { } and arrays being [ ].

If you send a request to the server, and it is successful (it doesn't guarantee it is successful, but most of the time it will be) you will receive a success message:

{"Success": [

{

"ms": "{Request successfully processed!}"

},

]

}

If you send a request to the server, and it is not successful, you will receive an error message (or more than one, but usually just one) back:

{ "Errors": [ { "ms": "User is not logged in.", "co": "0" }, ], }

*ms* is the message and *co* is the code, but all the codes are the same haha. I may fix that at some point.

The first thing you must do is set the environment variable to either "iphone" or "android" like so:

www.hangchillparty.com?action=setEnvironment&environment=android

The environment variable must be set first, or else other requests will not know how to respond.

Alright enough intro, here are some requests to play with:

**tryRememberMe**

action: tryRememberMe;

get Variables: none;

post Variables: none

response: login status

**getInitialData**

action: getInitialData

get Variables: none;

post Variables: none

response:

currentTime Object: of is the current unix server time (# seconds since 1970)

basicUser Object: id is userID

fn is first\_name

ln is last\_name

ignore st;

**cancelStatus**

action: cancelStatus;

get Variables: none;

post Variables: none

response: success message or errors message

**getAlertSetttings**

action: getAlertSettings;

get Variables: none;

post Variables: none

response:

alertSettings Object: tj is option to receive texts when a friend joins you

tg is option to receive text when friend signals you a green

ty is option to receive text when friend signals you a yellow

ej is option to receive email when a friend joins you

efar is option to receive email when you get a friend request

efnr is option to receive email when you get submit a friend request

ewu is option to receive email when you there are site updates

eg is option to receive email when friend signals you a green

ey is option to receive email when friend signals you a yellow

These are all boolean values 1 is true 0 is false;

**getUserSetttings**

action: getUserSettings;

get Variables: none;

post Variables: none

response:

userSettings Object: id is user id

fn is first name

ln is last name

gn is gender

bi is birthday

em is email

bl is blurb

tzr is time zone region

tzz is time zone zone

ci is city ID

cn is city Name

rn is city Region Name

nl is number of logins

ignore fd

**getNewAlerts**

action: getNewAlerts;

get Variables: none;

post Variables: none

response:

alertsCount Object ignore this

newsItems Object

id is news Item ID

li is news item url link

tx is news item text or title

up is news item update time

newFriendships Object

friend requests Object (NOTE both these have the same format of response)

id1 is requesterID

fn1 is requester first name

ln1 is requester last name

id2 is receiverID

fn2 is receiver first name

ln2 is receiver last name

st is status (0 is new, 1 is accepted, 2 is rejected)

up is update time

**getNewUpdates**

action: getNewUpdates;

get Variables: none;

post Variables: none

response:

logID Object ignore this

Updates Object

id is update ID

ts is time start

te is time end

pl is place

ac is activity

li is the leader user ID (with Eric Debusschere)

lfn is leader first name

lln is leader last name

fid is friend first ID (if the leader is not your friend Jim Johnson is with....)

ffn is friend first name

fln is friend last name

fc is friend count

sr is social rating of the person in the update (friend or leader)

ca is canceled boolean 1 if the update is canceled 0 if not

yu is your update, so if this is true you have to be the friend (Eric Debusschere is...)

ut is update time

**getUserProfile**

action: getUserProfile;

get Variables: none;

post Variables: none

response:

Friend Object

id is friend ID

fn is friend first name

ln is friend last name

ph is friend phone

bl is blurb or quickie

fb is facebook link

tn is twitter name (link is twitter.com/name)

sr is social rating

fc is total number of friends

mu is total number of mutual friends

csn is current school name

csc is current school class

fr is friend bool (1 if you are friends with that person, 0 if not)

re is request bool (1 if you have requested this person, 0 if not)

nu is this friends number of signals

Update Object

if you are friends with this person and they have updated you will get their current signal as well. This is formatted the same as a new update so see getNewUpdates.

**loginUser**

action: loginUser;

get Variables: none;

post Variables: email, password, rememberMe (0 or 1)

response: success or errors

**logoutUser**

action: logoutUser;

get Variables: none;

post Variables: none;

response: success or errors

**setProfileSettings**

action: setProfileSettings;

get Variables: none;

post Variables: firstName, lastName, gender, birthday, blurb, cityID, timeZoneID, currentSchool, currentSchoolClass, pastSchoolsAR(1,2,3...), pastSchoolsClassAR, currentPassword

response: success message or errors message

**setContactSettings**

action: setContactSettings;

get Variables: none;

post Variables: phone, email, currentPassword

response: success message or errors message

**setPasswordSettings**

action: setPasswordSettings;

get Variables: none;

post Variables: password, passwordVerified,currentPassword

response: success message or errors message

**setTextSettings**

action: setTextSettings;

get Variables: none;

post Variables: text\_friend\_joins, text\_green, text\_yellow, currentPassword

response: success message or errors message

**setEmailSettings**

action: setEmailSettings;

get Variables: none;

post Variables: email\_friend\_joins, email\_fri\_accept\_request, email\_new\_fri\_request, email\_weutt\_updates, email\_green, email\_yellow

response: success message or errors message

**setStatus**

action: setStatus;

get Variables: none;

post Variables: timeEnd, place, activity, light

response: success message or errors message

**setJoinedStatus**

action: setJoinedStatus

getVariables: none;

post Variables: timeEnd, joinedWithID

response: success message or errors message

**registerUser**

action: registerUser

getVariables: none;

post Variables: firstName, lastName, gender, birthday, phone, email, blurb, cityID, timeZoneID, displayFriendDist, password, passwordVerified, currentSchool, currentSchoolClass, pastSchoolsAR, pastSchoolsClassAR

response: success message or errors message

just enter null for values that a user does not enter or dont have any meaning (like displayFriendDist)

**requestFriend**

action: requestFriend

getVariables: receiverID;

post Variables: none

response: success message or errors message

**respondToFriendRequest**

action: respondToFriendRequest

getVariables: none

post Variables: requesterID, status (1 for accepted, 2 for rejected)

response: success message or errors message

**findFriends**

action: findFriends

getVariables: none

post Variables: name, schoolID, classYear, cityID, distance, pageNumber, orderByName (0 or 1), newSearch (0 or 1)

response:

searchCount Object (only on new Searches)

co is number of total results

searchResults Object

id is user ID

fn is user first name

ln is user last name

sid is current school ID

sna is current school name

ignore sty

cl is current school class

mf is mutual friends count

frc is friend request check (1 if you have requested this user, 0 if not)

**getTimeZones**

action: getTimeZones

getVariables: region (either Africa, America, Asia, Atlantic, Australia, Europe, Indian, Pacific)

post Variables: none

response:

TimeZones Object

id is timeZoneID

zo is zone name

**getImage - this is how you get all user images**

action: getImage

getVariables: userID, width, height (image will come back as width and height you specify

post Variables: none

response: this is just a standard link, no response

**uploadImage**

action: uploadImage

standard upload image input for html (not sure how this works on phones, probably wont do it at first)

**getFriendsOf**

action: getFriendsOf

getVariables: none

post Variables: userID, type (either "generalFriends" or "mutualFriends"), pageNumber, newSearch (0 or 1)

response:

same response as getFindFriends so see that

**getTermsOfService**

action: getTermsOfService

getVariables: none

post Variables: none

response:

terms of service in HTML not JSON

**sendForgottenPassword**

action: sendForgottenPassword

getVariables: email

post Variables: none

response: success or errors

**checkIfEmailExists**

action: checkIfEmailExists

getVariables: email

post Variables: none

response: success or errors

**getAllFriends**

action: getAllFriends

getVariables: none

post Variables: none

response:

Friends Object

id is user ID

fn is user first name

ln is user last name

ignore st

**getAutoComplete**

action: getAutoComplete

getVariables: query (what you are searching for), type (either "schools", "cities", or "friends")

post Variables: none

response:

**if schools**

Suggestions Object

na is school name

id is school ID

cn is region name

co is region code

cnt is count

Ho w I do it on the website is if count is greater than 1 (so two schools have the same name) then you add (region name, country code) in parenthesis after each of the duplicate schools to differentiate them for example:

searching *olympia high school*  brings up:

Olympia High school (Washington, US)

Olympia High school (Florida, US)

Olympia High school (Illinois, US)

Olympia Institute

**if cities**

Suggestions Object

na is city name

id is city ID

cn is region name

co is region code

For cities I always have the region name, region code in parenthesis

**if friends**

Suggestions Object

fn is first name

ln is last name

id is friend ID

if you need any special requests or are confused by anything let me know at ANYTIME, 24 hours a day. These requests are not thoroughly tested for the smart phones environments so there may be some issues. If you install firebug you can see the requests made on the website and this may help if you have any issues, but once again I want to make clear that I am available anytime to give you guys help on this stuff. I know we didn't get you guys a ton of time with actual data, but if you can get anything done by the end of the summer (even if it's just a simple app that lets you login, signal, view updates, and join updates that would be fantastic.

And no hard feelings if you can't get anything done, I know it's a lot of work.

Good luck

Eric D.