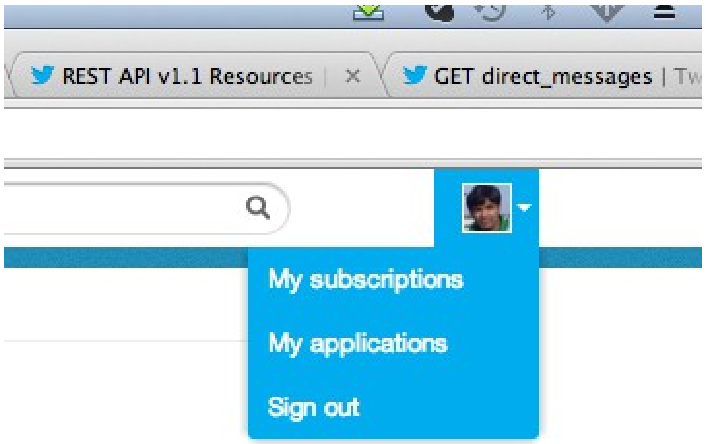
Twitter integration for Titanium iPhone + Android using birdhouse.js
By **Jayesh Joshi**

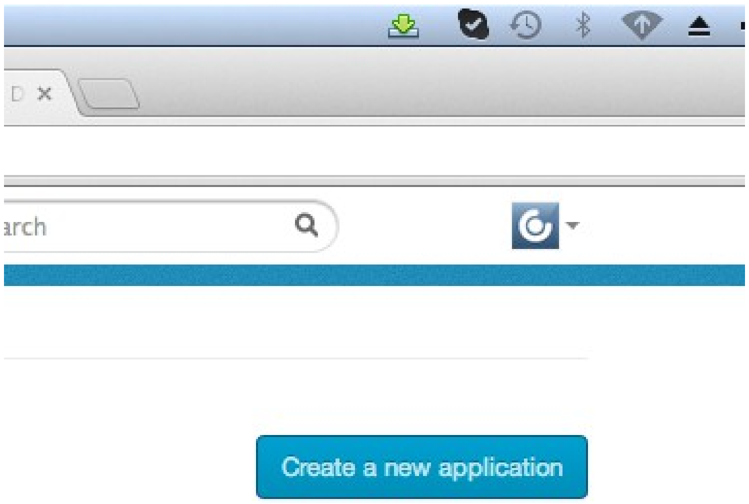
First you will have a valid twitter Account.
Login to you twitter account and
open https://dev.twitter.com

Take mouse on left side your profie image icon



Now click on **My Application** and you will have list of your Applications.

if No application than it will look like below

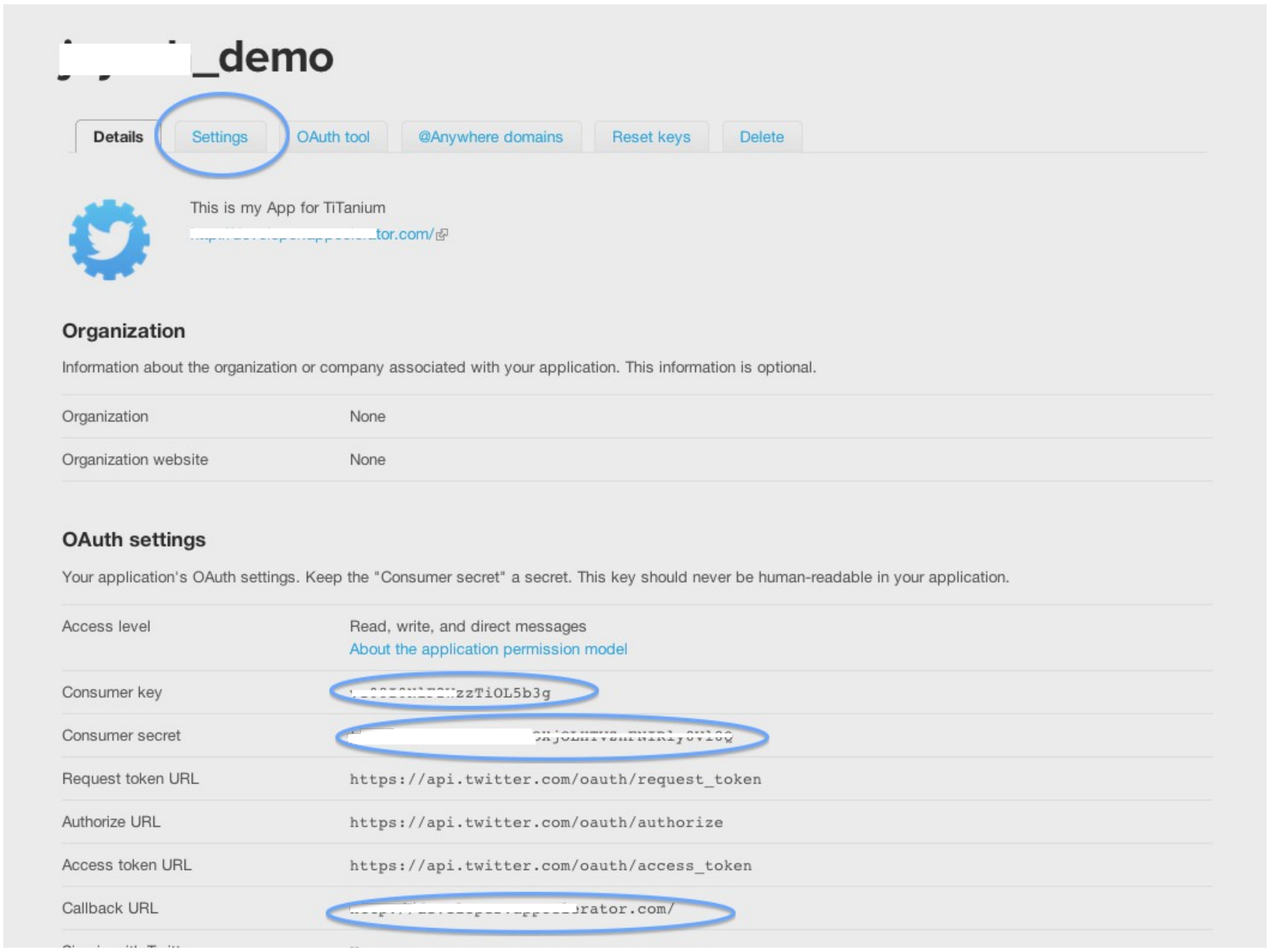


- Create New Application . - **So Create New Application**

While Creating App Provide below details

- Name
- Description
- Website
- Callback – URL (it will be must there)

Now
Select created/selecting existing application you will have detail page of app like below



[redacted]_demo

This is my App for TiTanium
[redacted]tor.com/

Organization

Information about the organization or company associated with your application. This information is optional.

Organization	None
Organization website	None

OAuth settings

Your application's OAuth settings. Keep the "Consumer secret" a secret. This key should never be human-readable in your application.

Access level	Read, write, and direct messages About the application permission model
Consumer key	[redacted]
Consumer secret	[redacted]
Request token URL	`https://api.twitter.com/oauth/request_token`
Authorize URL	`https://api.twitter.com/oauth/authorize`
Access token URL	`https://api.twitter.com/oauth/access_token`
Callback URL	[redacted]rator.com/

Below three from Application details is required. From

- OAuth Setting

1 – Consumer key
2 - Consumer secret
3 - callback url

- Access- Level shoud be Read, Write and Direcct Message if not change from setting Tab

Click on Setting Tab - Look Like below.

Now see Application type – Access

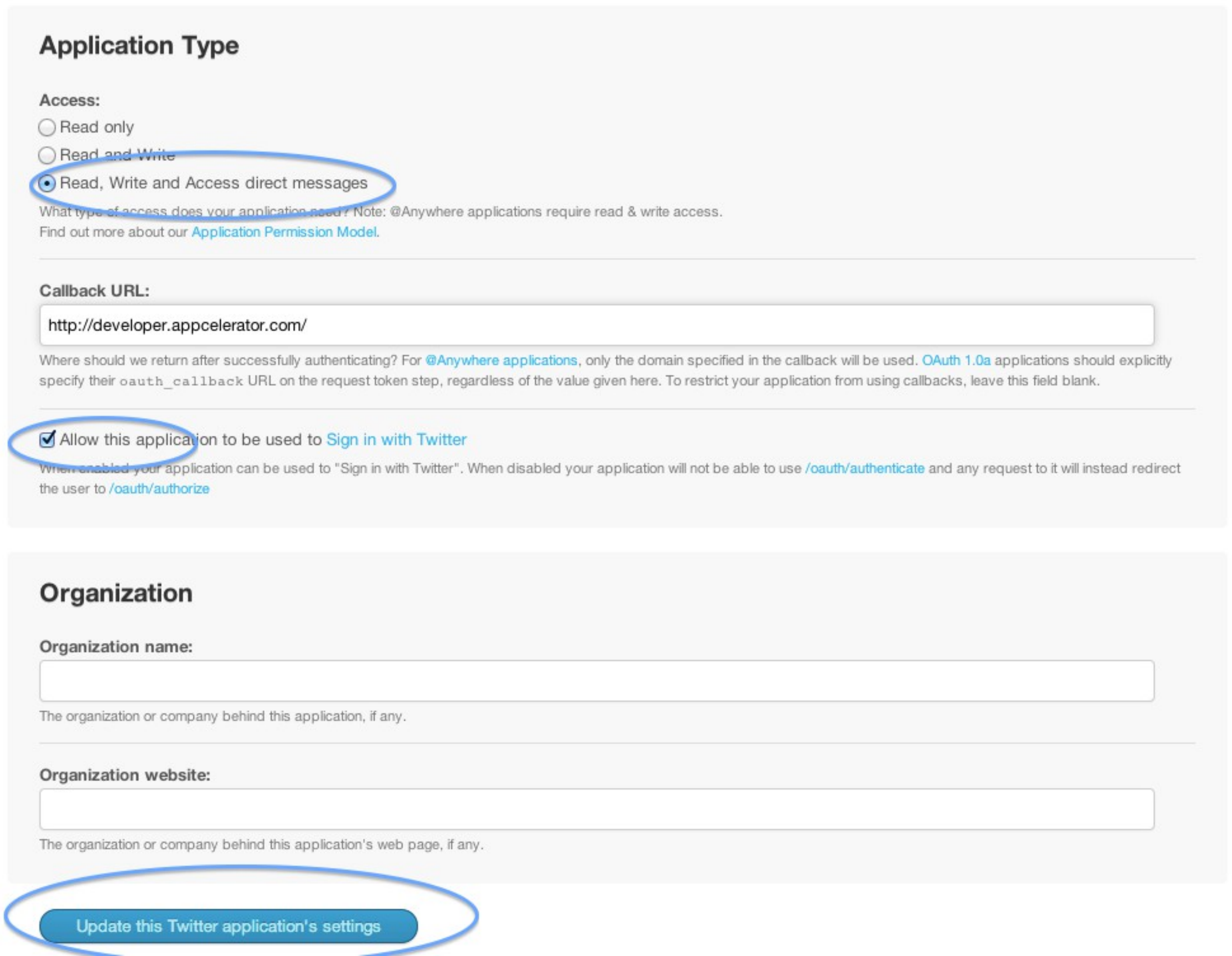
3 radio buttons are there select last button .

- Read, Writer and Access Direct Messages.

- check Allow this application to be used to sign In With Twitter.

Don't Forgot to click last button to update your setting .

Now All things are Done.



Application Type

Access:

( ) Read only
( ) Read and Write
[✓ Read, Write and Access direct messages]

What type of access does your application need? Note: @Anywhere applications require read & write access.
Find out more about our Application Permission Model.

Callback URL:

http://developer.appcelerator.com/

Where should we return after successfully authenticating? For @Anywhere applications, only the domain specified in the callback will be used. OAuth 1.0a applications should explicitly specify their `oauth_callback` URL on the request token step, regardless of the value given here. To restrict your application from using callbacks, leave this field blank.

[✓] Allow this application to be used to Sign in with Twitter

When enabled your application can be used to "Sign in with Twitter". When disabled your application will not be able to use /oauth/authenticate and any request to it will instead redirect the user to /oauth/authorize

Organization

Organization name:

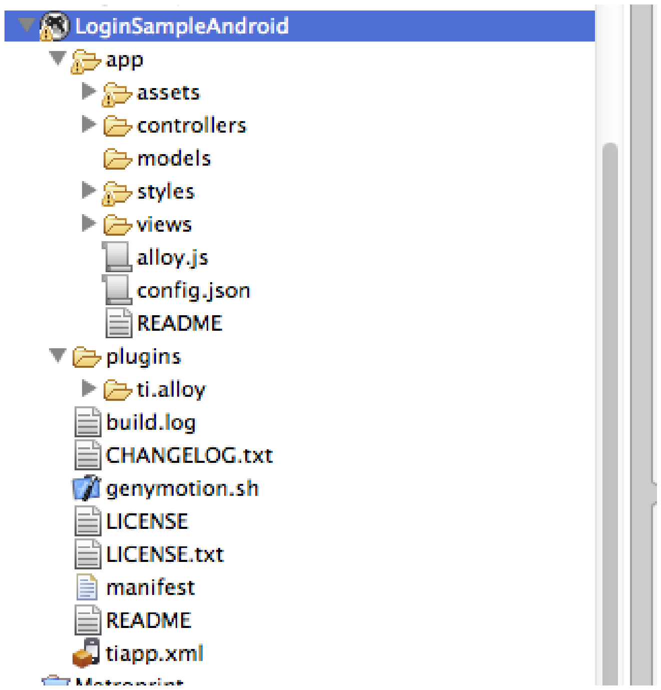
The organization or company behind this application, if any.

Organization website:

The organization or company behind this application's web page, if any.

Mobile Application Project : -

- Open Titanium Studio and take new project in **Alloy**.
- File - > New Mobile project
- Your new project structure look like below in Alloy.[expect genymotion.sh]



```
LoginSampleAndroid
    app
        assets
        controllers
        models
        styles
        views
        alloy.js
        config.json
        README
    plugins
        ti.alloy
    build.log
    CHANGELOG.txt
    genymotion.sh
    LICENSE
    LICENSE.txt
    manifest
    README
    tiapp.xml
```

- in Assets folder create new folder named **lib and copy oauth.js and sha1.js and Birdhouse.js**

`/lib/oauth.js`

`/lib/sha1.js`

`/lib/birdhouse.js`

Note :

I added both oauth and sha1 in birdhouse.js

```
Ti.include('/lib/oauth.js');
Ti.include('/lib/sha1.js');
```

---0------------------------------------- --------------

index Controller

-index.xml

1 - Login button
2 – Logout button

-inddx.tss

Give Appropriate style for ios and android

-index.js

include birdhouse.js in index.js file

```
Ti.include('/lib/birdhouse.js');
```

crate new object for birdhouse and provide `consumer_key , consumer_secret` and `callback_url`

All are required don't miss single. As like below

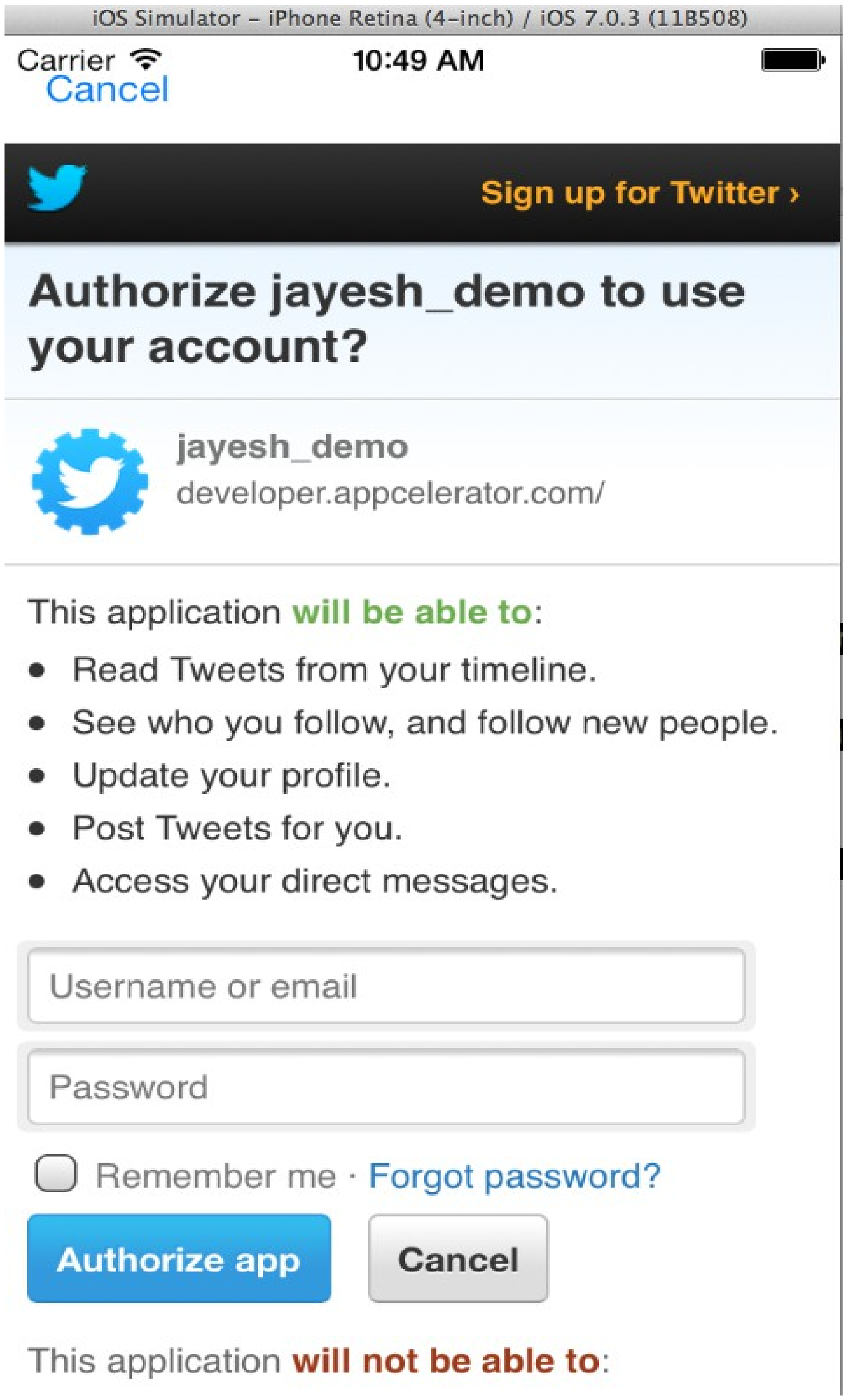
```
var BH = new BirdHouse({
           consumer_key : "xxxxxxxxxxxxxx",
           consumer_secret : "xxxxxxx",
           callback_url : "you callback url.com"
      });
```

-Login button Click

```
$.login.addEventListener('click', function(e) {
if (Titanium.Network.online) {
     BH.authorize(function(e) {
        if (e === true) {
      var TwitterMainWin =Alloy.createController('TwitterMain').getView();
                      TwitterMainWin.open();
                 } else {
     alert('Failed to authorize Twitter \n Please try again.');
                      }
     });
     } else {
                 alert('Please turn on internet connection.');
           }
     });
```

in Above code Bh.authorize is function check for authorization.
You are not loggedIn so it will open login dialog box for authorization.

Like below


iOS Simulator – iPhone Retina (4-inch) / iOS 7.0.3 (11B508)

Carrier 10:49 AM

Cancel

Sign up for Twitter ›

Authorize jayesh_demo to use your account?

jayesh_demo
developer.appcelerator.com/

This application will be able to:

- Read Tweets from your timeline.
- See who you follow, and follow new people.
- Update your profile.
- Post Tweets for you.
- Access your direct messages.

This application will not be able to:

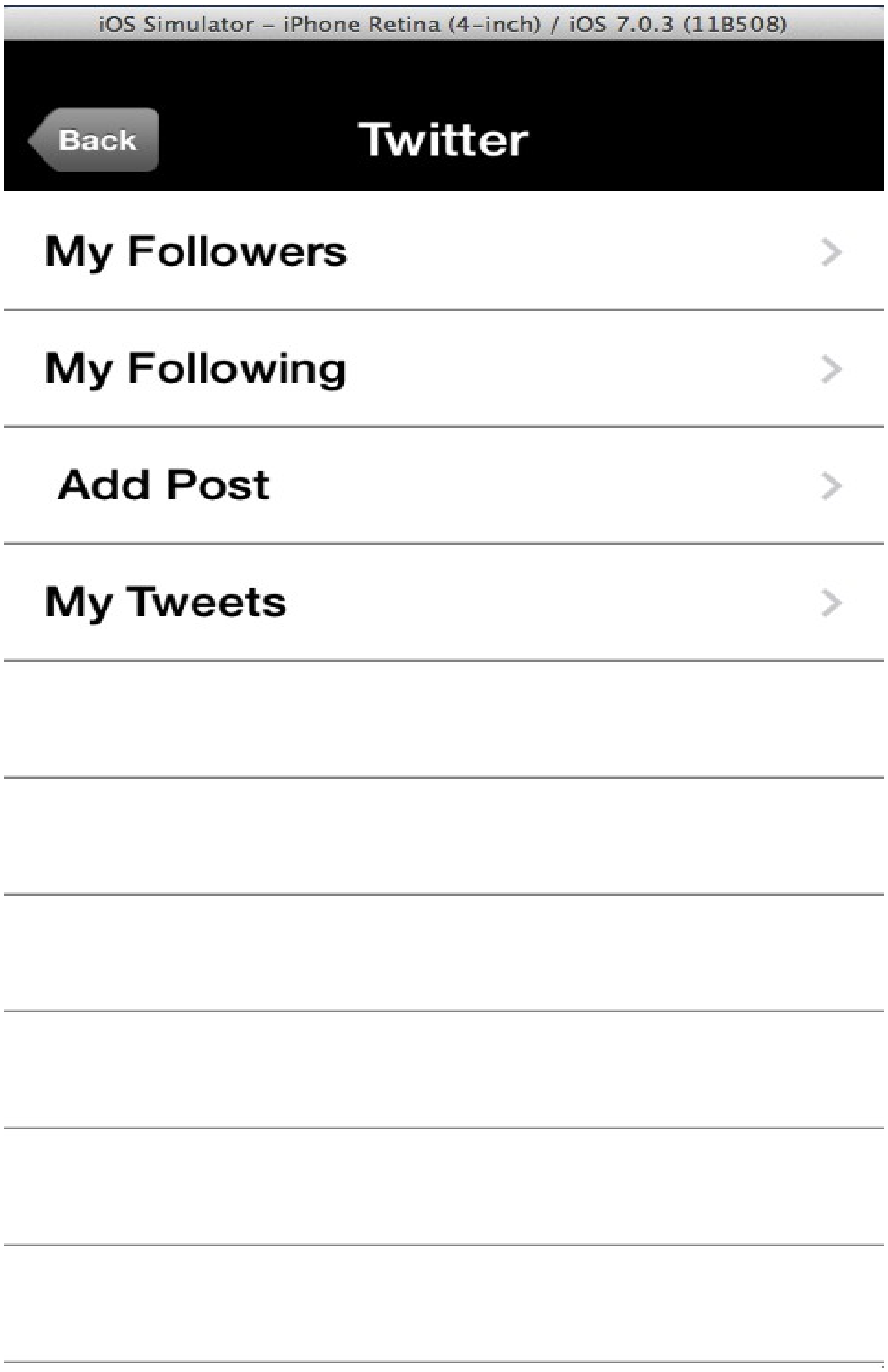
Enter you Login credentials and tap authorization button

---1------------------------------------- --------------

TwitterMain Controller

This can take some time for authorization

On successFull login your TwitterMain screen will open :- like below



-TwitterMain.xml

TableView with rows provide option

-TwitterMain.tss

Appropriate style to tableview

-TwitterMain.js

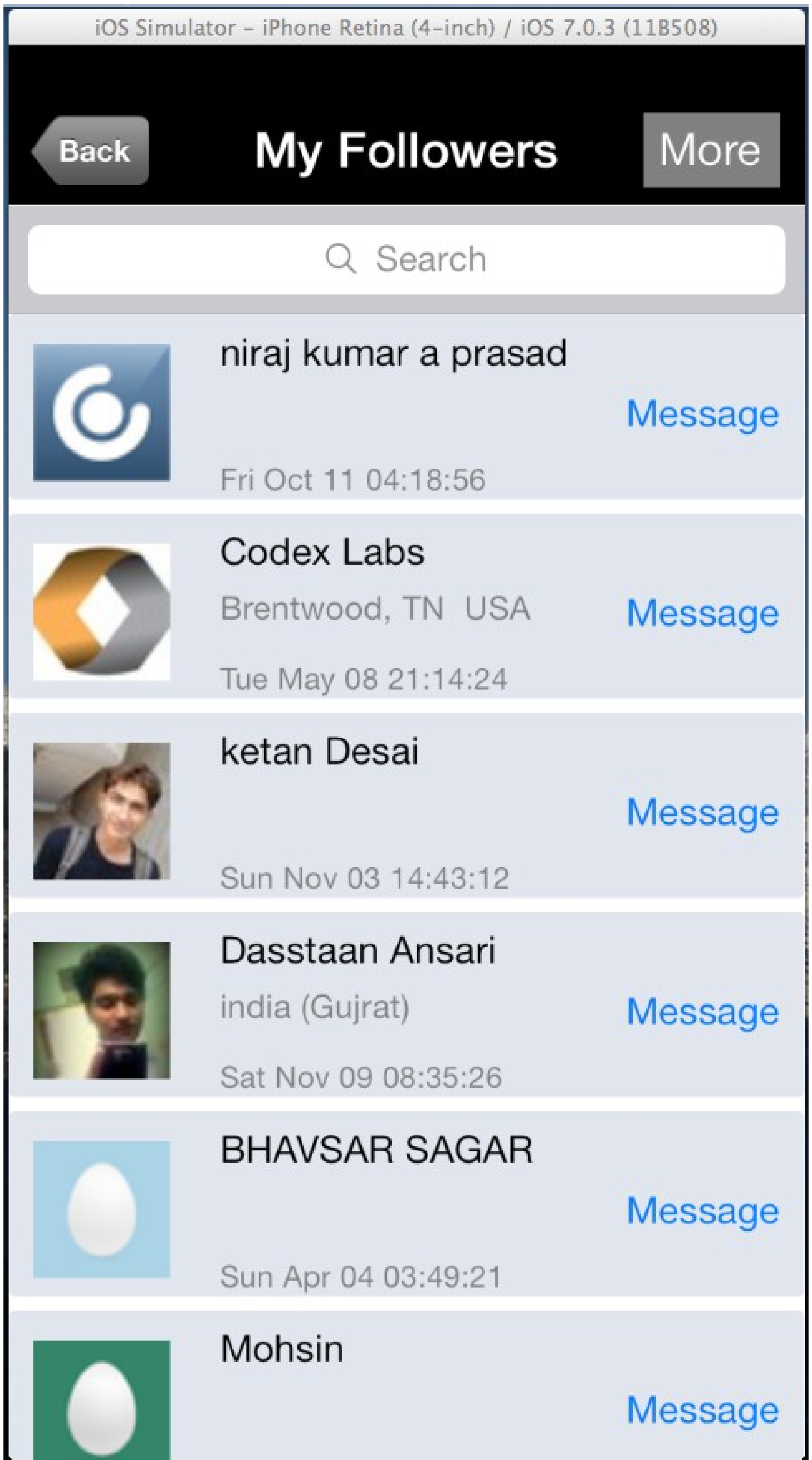
open new Controller based on selection from tableView

```
$.tablebview.addEventListener('click', function(e) {
     if (e.index == 0) {

     var TwitterMyFollowersWin =
Alloy.createController('TwitterMyFollowers').getView();
                TwitterMyFollowersWin.open();
     } else if (e.index == 1) {

                var TwitterMyFollowingWin =
Alloy.createController('TwitterMyFollowing').getView();
                TwitterMyFollowingWin.open();

     } else if (e.index == 2) {
          var TwitterPostWin =
Alloy.createController('TwitterPost').getView();
          TwitterPostWin.open();
     } else if (e.index == 3) {

                var TwitterMyTweetsWin =
Alloy.createController('TwitterMyTweets').getView();
                TwitterMyTweetsWin.open();

     }
});
```

---2------------------------------------- --------------

TwitterMyFollwers Controller

it can only load 20 Follwers first and on click of more next 20 will added preform upto all.


iOS Simulator – iPhone Retina (4-inch) / iOS 7.0.3 (11B508)

niraj kumar a prasad
Message
Fri Oct 11 04:18:56

Codex Labs
Brentwood, TN USA
Message
Tue May 08 21:14:24

ketan Desai
Message
Sun Nov 03 14:43:12

Dasstaan Ansari
india (Gujrat)
Message
Sat Nov 09 08:35:26

BHAVSAR SAGAR
Message
Sun Apr 04 03:49:21

Mohsin
Message

-TwitterMyFollwers.xml

Tableview

-TwitterMyFollwers.tss

style to tableView

TwitterMyFollwers.js

GetFollowers function load followers

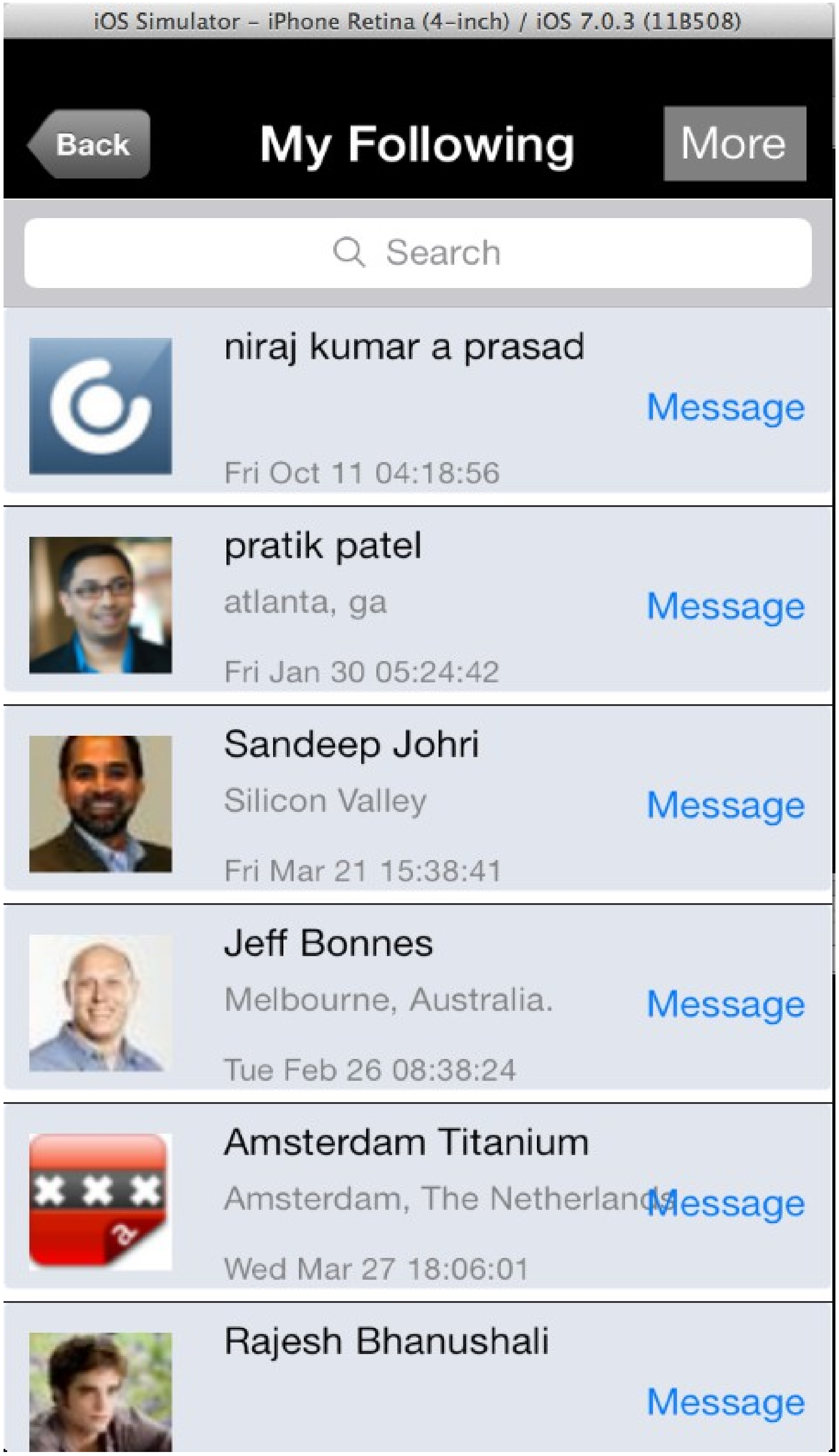
Data post format

```
var cursor1 = -1;
function GetFollowers(cursorData) {
	Titanium.API.info('------Cursor:------:' + cursorData);
	Ti.App.showIndicator();
	//BH.get_followers('cursor=' + cursorData + 'screen_name=' +
Ti.App.Properties.getString('ttname') + 'skip_status=' + true +
'include_user_entities=' + false, function(resp) {
	BH.get_Myfollowers('cursor=' + cursorData + '&screen_name=' +
Ti.App.Properties.getString('ttname') + '&skip_status=' + true +
'&include_user_entities=' + false, function(resp) {
		Titanium.API.info('----Resoponse user Followers------' +
JSON.parse(resp));
//your data from twitterAPI
}
```

---3-------------------------------------- --------------

TwitterMyFollowing Controller

it can only load 20 Following first and on click of more next 20 will added preform upto all.



iOS Simulator – iPhone Retina (4-inch) / iOS 7.0.3 (11B508)

Back
My Following
More

Search

niraj kumar a prasad
Message
Fri Oct 11 04:18:56

pratik patel
atlanta, ga
Message
Fri Jan 30 05:24:42

Sandeep Johri
Silicon Valley
Message
Fri Mar 21 15:38:41

Jeff Bonnes
Melbourne, Australia.
Message
Tue Feb 26 08:38:24

Amsterdam Titanium
Amsterdam, The Netherlands
Message
Wed Mar 27 18:06:01

Rajesh Bhanushali
Message

-TwitterMyFollowing.xml

Tableview

-TwitterMyFollowing .tss

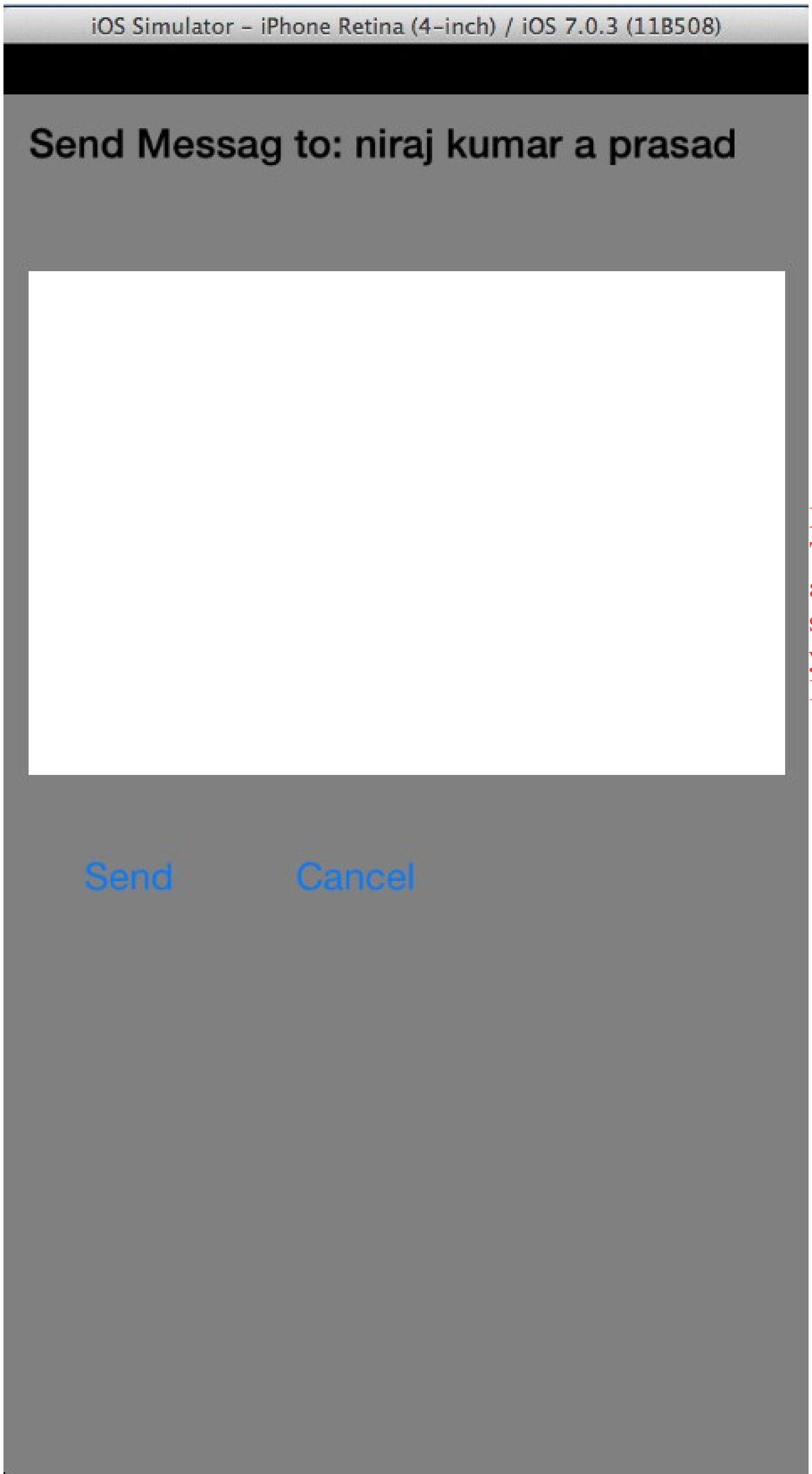
Style to tableView

TwitterMyFollowing.js

GetFriends function load Followings

Data send Fromat

```
function GetFriends(cursorData) {
     Titanium.API.info('------Cursor:------:' + cursorData);
     Ti.App.showIndicator();
     //BH.get_Friends('cursor=' + cursorData + 'screen_name=' +
Ti.App.Properties.getString('ttname') + 'skip_status=' + true +
'include_user_entities=' + false, function(resp) {
     BH.get_Myfollowing('cursor=' + cursorData + '&screen_name=' +
Ti.App.Properties.getString('ttname') + '&skip_status=' + true +
'&include_user_entities=' + false, function(resp) {
          Titanium.API.info('----Resoponse user------' +
JSON.parse(resp));
//data from Twitter API
}
}
```



iOS Simulator - iPhone Retina (4-inch) / iOS 7.0.3 (11B508)

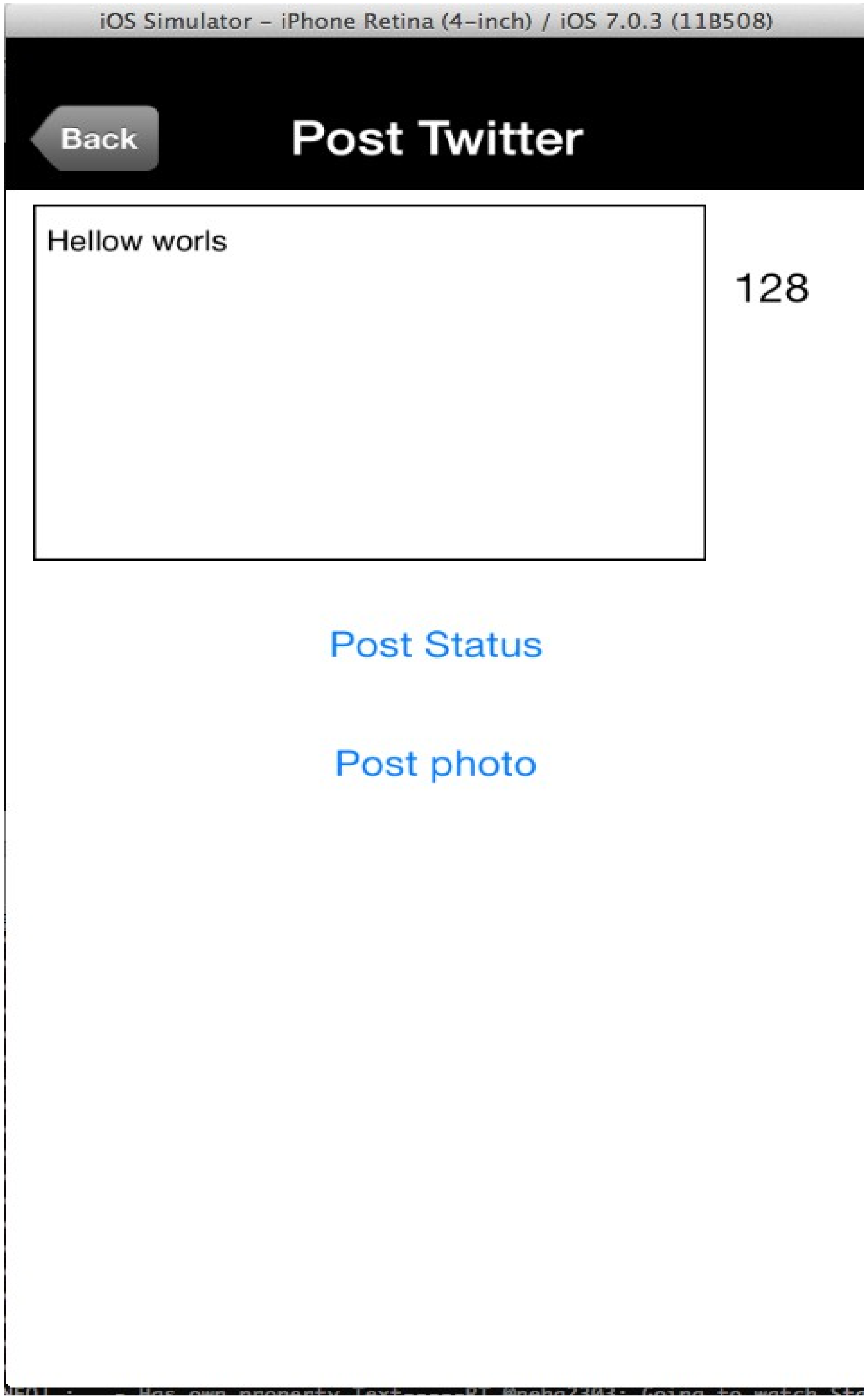
Send Messag to: niraj kumar a prasad

Message Button on TableView Row for iOS and android Allow you to send direct message to your Followers/Followings.

---4------------------------------------- --------------

Twitterpost Controller

-enter text and tap post status
-post photo allow you to share photo from gallery or camera directly.

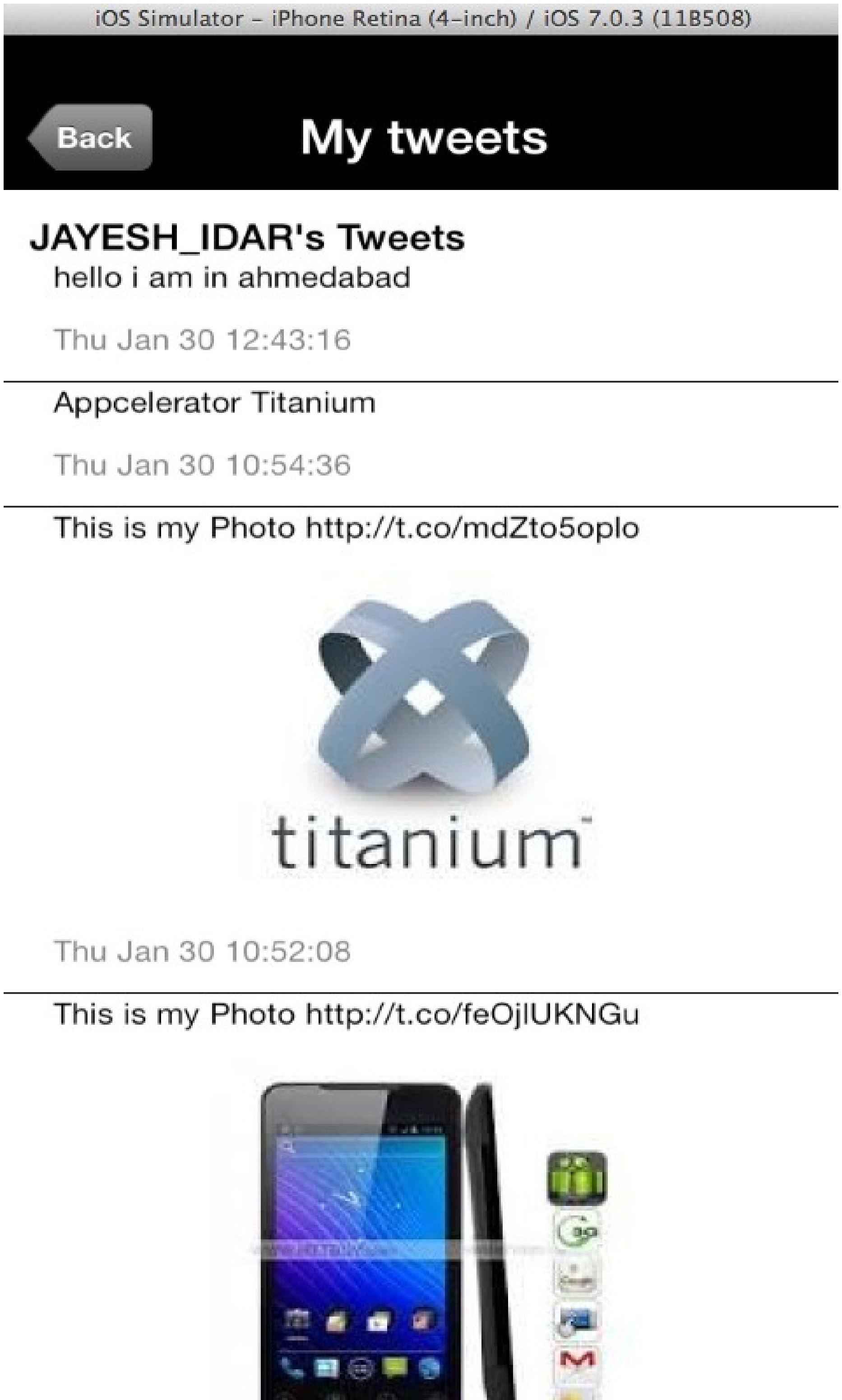


Data Fromat for send tweet

```
BH.send_tweet('status=' + $.textshare.value, function(resp) {

}
```

---5------------------------------------- --------------

TwitterMyTweets controller



JAYESH_IDAR's Tweets

hello i am in ahmedabad

Thu Jan 30 12:43:16

Appcelerator Titanium

Thu Jan 30 10:54:36

This is my Photo http://t.co/mdZto5oplo

Thu Jan 30 10:52:08

This is my Photo http://t.co/feOjlUKNGu

-TwitterMytweets.xml

Tableview

-TwitterMytweets .tss

Style to tableView

TwitterMytweets.js

Get_My_tweets function load All my tweets API limit is 3200

Data format for MyTweets

```
var Get_MyTweets = function() {
     Ti.App.showIndicator();
     BH.get_Mytweets('screen_name=' + Ti.App.Properties.getString('ttname')
+ '&count=' + 5000, function(resp) {
}
}
```

Birdhouse.js

```
// ---------------------------------------------------------------
// birdhouse.js
//
// BirdHouse is a Titanium Developer plugin for
// authenticating and sending API calls to Twitter.
//
// Copyright 2011 (c) iEntry, Inc
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
//     http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
// See the License for the specific language governing permissions and
// limitations under the License.
//
// Author: Joseph D. Purcell, iEntry Inc
// Version: 0.9
// Modified: May 2011
// ---------------------------------------------------------------
// INCLUDES
// iphone requires complete path
Ti.include('/lib/oauth.js');
Ti.include('/lib/sha1.js');
// THE CLASS
function BirdHouse(params) {
	// ----------------------------------------------------------------
	// ==================== PRIVATE ===========================
	// ----------------------------------------------------------------
	// VARIABLES
	var cfg = {
		// user config
		oauth_consumer_key : "",
		consumer_secret : "",
		show_login_toolbar : false,
		// system config
		oauth_version : "1.0",
		oauth_token : "",
		oauth_signature_method : "HMAC-SHA1",
		request_token : "",
		request_token_secret : "",
		request_verifier : "",
		access_token : "",
		access_token_secret : "",
		callback_url : ""
	};
	var accessor = {
		consumerSecret : cfg.consumer_secret,
		tokenSecret : cfg.access_token_secret
	};
	var authorized = false;
	// ----------------------------------------------------------------
	// set_message
	//
	// Creates a message to send to the Twitter service with
	// the given parameters, and adds the consumer key,
	// signature method, timestamp, and nonce.
	//
	// In Parameters:
	//	url (String) - the url to send the message to
	//	method (String) - 'POST' or 'GET'
	//	params (String) - parameters to add to the
	//	  message in URL form, i.e. var1=2&var2=3
	//
	// Returns:
	//	message (Array) - the message parameters to send
```

```
    //      to Twitter
    // ----------------------------------------------------
    function set_message(url, method, params) {
        var message = {
            action : url,
            method : (method == 'GET') ? method : 'POST',
            parameters : (params != null) ? OAuth.decodeForm(params) : 
[]
        };
        message.parameters.push(['oauth_consumer_key', 
cfg.oauth_consumer_key]);
        message.parameters.push(['oauth_signature_method', 
cfg.oauth_signature_method]);
        message.parameters.push(["oauth_timestamp", 
OAuth.timestamp().toFixed(0)]);
        message.parameters.push(["oauth_nonce", OAuth.nonce(42)]);
        message.parameters.push(["oauth_version", "1.0"]);
        return message;
    }

    // ----------------------------------------------------
    // get_request_token
    //
    // Sets the request token and token secret.
    //
    // In Parameters:
    //    callback (Function) - a function to call after
    //      the user has been authorized; note that it won't
    //      be executed until get_access_token()
    // ----------------------------------------------------
    function get_request_token(callback) {
        Ti.API.info('========================== + get reqquest Tokewn + 
==========================');
        var url = 'https://api.twitter.com/oauth/request_token';
        var params = (cfg.callback_url != "") ? 'oauth_callback=' + 
escape(cfg.callback_url) : '';
        api(url, 'POST', params, function(resp) {
            if (resp != false) {
                Ti.API.info('========================== + get 
reqquest Tokewn SUCCESS+ ==========================');
                var responseParams = OAuth.getParameterMap(resp);
                cfg.request_token = responseParams['oauth_token'];
                cfg.request_token_secret = 
responseParams['oauth_token_secret'];
                get_request_verifier(callback);
            } else {
                Ti.API.info('========================== + get 
reqquest Tokewn FALSE+ ==========================');
            }
        }, false, true, false);
    }
```

```
        // ----------------------------------------------------------
        // get_request_verifier
        //
        // Sets the request verifier. There is no reason to call
        // this unless you have the request token and token secret.
        // In fact, it should only be called from get_request_token()
        // for that very reason.
        //
        // In Parameters:
        //    callback (Function) - a function to call after
        //      the user has been authorized; note that it won't
        //      be executed until get_access_token()
        // ----------------------------------------------------------
        function get_request_verifier(callback) {
            try {
                var url_1 = "https://api.twitter.com/oauth/authorize?
oauth_token=" + cfg.request_token;
                Ti.API.info('========================== + get reqquest 
Verifier + =========================' + url_1);
```

```
                if (OS_IOS) {
                    var win = Ti.UI.createWindow({
                        top : 0,
                        modal : true,
                    });
                    var leftnavBtn = Ti.UI.createButton({
                        left : 10,
                        height : 29,
                        width : 62,
                        title : 'Cancel'
                    });
                    leftnavBtn.addEventListener('click', function(e) {
                        Ti.App.hideIndicator();
                        win.close();
                    });
                    var view = Titanium.UI.createView({
                        top : 0,
                        left : 0,
                        right : 0,
                        height : 45,
					backgroundColor : 'white'
				});
				view.add(leftnavBtn);
				win.add(view);
```

```
				var webView = Ti.UI.createWebView({
					top : 45,
					left : 0,
					url : url_1,
					scalesPageToFit : true,
					touchEnabled : true
				});
			} else {
				var win = Ti.UI.createWindow({
					statusBarHidden : true,
					navBarHidden : true,
					exitOnClose : false,
					orientationModes : [Ti.UI.PORTRAIT]
				});
				var leftnavBtn = Ti.UI.createButton({
					title : 'Cancel',
					backgroundColor : 'gray',
					top : 10,
					left : 10,
					height : 40,
					width : 100
				});
				leftnavBtn.addEventListener('click', function(e) {
					Ti.App.hideIndicator();
					win.close();
				});
				var view = Titanium.UI.createView({
					top : 0,
					left : 0,
					right : 0,
					height : 70,
					backgroundColor : 'white'
				});
				view.add(leftnavBtn);
				win.add(view);
```

```
				var webView = Ti.UI.createWebView({
					top : 70,
					left : 0,
					url : url_1,
					scalesPageToFit : true,
                        touchEnabled : true
                    });
                }
                var request_token = "";
                var url_base = "";
                var params = "";
                var loading = false;
                // since the 'loading' property on webView is broke, use
this
                var loads = 0;
                // number of times webView has loaded a URl
                var doinOurThing = false;
                // whether or not we are checking for oauth tokens
                // add the webview to the window and open the window
                win.add(webView);
                win.open();
                Titanium.API.info('---00-----');
                // since there is no difference between the 'success' or
'denied' page apart from content,
                // we need to wait and see if Twitter redirects to the
callback to determine success
                function checkStatus() {
                    Titanium.API.info('---123-----');
                    if (!doinOurThing) {
                        // access denied or something else was clicked
                        if (!loading) {
                            webView.stopLoading();
                            win.remove(webView);
                            win.close();
                            if ( typeof (callback) == 'function') {
                                callback(false);
                            }
                            return false;
                        }
                    } else {
                    }
                }


                Titanium.API.info('--456-----');
                webView.addEventListener('beforeload', function() {
                    Titanium.API.info('---789-----' + webView.url);
                    loading = true;
                });
                webView.addEventListener('load', function(e) {
                    loads++;
                    Titanium.API.info('---10-----' + loads);
                    Titanium.API.info('---10 and URL-----:' +
webView.url);
                    // the first time load, ignore, because it is the
initial 'allow' page
                    // set timeout to check for something other than
'allow', if 'allow' was clicked
                    // then loads==3 will cancel this
                    if (loads == 2) {
                        // something else was clicked
                        Titanium.API.info('---11-----');
                        if (e.url !=
'https://api.twitter.com/oauth/authorize') {
                            Titanium.API.info('---12-----');
                            webView.stopLoading();
                            win.remove(webView);
                            win.close();
                            if ( typeof (callback) == 'function') {
                                callback(false);
                            }
                            return false;
                        }
                        // wait a bit to see if Twitter will redirect
                        else {
                            setTimeout(checkStatus, 1000);
                        }
                    }
                    // Twitter has redirected the page to our callback
URL (most likely)
                    else if (loads == 3) {
                        try {
                            Titanium.API.info('---13------------' +
e.url);
                            doinOurThing = true;
                            // kill the timeout b/c we are doin our
thing
                            // success!
                            params = "";
                            Titanium.API.info('---14------------' +
e.url);

                            var parts = (e.url).replace(/[?&]+([^=&]
+)=([^&]*)/gi, function(m, key, value) {
                                params = params + m;
                                Titanium.API.info('========
PARAM:=========' + params);
```

```
                            if (key == 'oauth_verifier') {
                                cfg.request_verifier = value;
                            }
                        });
                        Titanium.API.info('---15:------------' +
cfg.request_verifie);
                        if (cfg.request_verifier != "") {
                            Titanium.API.info('========GET TOKEN
qirh WEbVIEW=========');
                            // my attempt at making sure the
stupid webview dies
                            webView.stopLoading();
                            win.remove(webView);
                            win.close();
                            get_access_token(callback);
                            return true;
                            // we are done here
                        }
                        Titanium.API.info('---16------------' +
e.url);
                    } catch(ex) {
                        alert('ex' + ex)
                    }
                }
                // we are done loading the page
                loading = false;
            });
        } catch(ex) {
            alert('excep' + ex);
            Ti.API.info('=============================== ex
exe=========' + ex);
        }
    }

    // ----------------------------------------------------------
    // get_access_token
    //
    // Trades the request token, token secret, and verifier
    // for a user's access token.
    //
    // In Parameters:
    //    callback (Function) - a function to call after
    //      the user has been authorized; this is where
    //      it will get executed after being authorized
    // ----------------------------------------------------------
    function get_access_token(callback) {
            Ti.API.info('========================== + get_access_token + 
==========================');
            var url = 'https://api.twitter.com/oauth/access_token';
            api(url, 'POST', 'oauth_token=' + cfg.request_token + 
'&oauth_verifier=' + cfg.request_verifier, function(resp) {
            if (resp != false) {
                    var responseParams = OAuth.getParameterMap(resp);
                    cfg.access_token = responseParams['oauth_token'];
                    cfg.access_token_secret = 
responseParams['oauth_token_secret'];
                    cfg.user_id = responseParams['user_id'];
                    cfg.screen_name = responseParams['screen_name'];
                    accessor.tokenSecret = cfg.access_token_secret;
                    save_access_token();
                    authorized = load_access_token();
                    // execute the callback function
                    if ( typeof (callback) == 'function') {
                        callback(true);
                    }
            } else {
                    // execute the callback function
                    if ( typeof (callback) == 'function') {
                        callback(false);
                    }
            }
            }, false, true, false);
    }

    // ----------------------------------------------------------------
    // load_access_token
    //
    // Loads the access token and token secret from
    // 'twitter.config' to the class configuration.
    // ----------------------------------------------------------------
    function load_access_token() {
            Ti.API.info('============================load ACCESS 
TOEKN=======================');
            // try to find file
            var file = 
Ti.Filesystem.getFile(Ti.Filesystem.applicationDataDirectory, 
'twitter.config');
         if (!file.exists()) {
               return false;
         }
         // try to read file
         var contents = file.read();
```

```
		if (contents == null) {
			return false;
		}
		// try to parse file into json
		try {
			var config = JSON.parse(contents.text);
		} catch(e) {
			return false;
		}
		// set config
		if (config.access_token) {
			cfg.access_token = config.access_token;
		}
		if (config.access_token_secret) {
			cfg.access_token_secret = config.access_token_secret;
			accessor.tokenSecret = cfg.access_token_secret;
		}
		return true;
	}
```

```
	// ---------------------------------------------------------
	// save_access_token
	//
	// Writes the access token and token secret to
	// 'twitter.config'. Saving the config in a file instead
	// of using Ti.App.Property jazz allows the config to
	// stay around even if the app has been recompiled.
	// ---------------------------------------------------------
	function save_access_token() {
		Ti.API.info('============================SAVE 
TOEKN======================');
		// get file if it exists
		var file = 
Ti.Filesystem.getFile(Ti.Filesystem.applicationDataDirectory, 
'twitter.config');
		// create file if it doesn't exist
		if (file == null) {
			file = 
Ti.Filesystem.createFile(Ti.Filesystem.applicationDataDirectory, 
'twitter.config');
		}
		Ti.App.Properties.setString('ttname', cfg.screen_name);
		Ti.App.Properties.setString('ttid', cfg.user_id);
		Ti.App.Properties.setString('ttimg', 
'https://api.twitter.com/1/users/profile_image/' + cfg.screen_name);
		// 296732259
```

```
            //JAESH_IDAR
            Ti.API.info(cfg.user_id);
            Ti.API.info(cfg.screen_name);
            //Ti.App.info('https://api.twitter.com/1/users/profile_image/' +
cfg.screen_name);
            // write config
            var config = {
                  access_token : cfg.access_token,
                  access_token_secret : cfg.access_token_secret,
                  user_id : cfg.user_id,
                  screen_name : cfg.screen_name
            };
            file.write(JSON.stringify(config));
      }

      // ----------------------------------------------------------------
      // api
      //
      // Makes a Twitter API call to the given URL by the
      // specified method with the given parameters.
      //
      // In Parameters:
      //    url (String) - the url to send the XHR to
      //    method (String) - POST or GET
      //    params (String) - the parameters to send in URL
      //      form
      //    callback (Function) - after execution, call
      //      this function and send the XHR data to it
      //    auth (Bool) - whether or not to force auth
      //    setUrlParams (Bool) - set the params in the URL
      //    setHeader (Bool) - set "Authorization" HTML header
      //
      // Notes:
      //    - the setUrlParams and setHeader should only need
      //      to be set whenever getting request tokens; values
      //      should be 'true' and 'false' respectively
      //    - take advantage of the callback function, if you
      //      want to tweet a message and then display an alert:
      //          BH.tweet("some text",function(){
      //              alertDialog = Ti.UI.createAlertDialog({
      //                  message:'Tweet posted!'
      //              });
      //              alertDialog.show();
      //          });
      //
      // Returns: false on failure and the responseText on
```

```
    //   success.
    // --------------------------------------------------
    function api(url, method, params, callback, auth, setUrlParams, 
setHeader) {
        try {
            var finalUrl = '';
            // authorize user if not authorized, and call this in the 
callback
            if (!authorized && ( typeof (auth) == 'undefined' || auth 
=== true)) {
                authorize(function(retval) {
                    if (!retval) {
                        // execute the callback function
                        if ( typeof (callback) == 'function') {
                            callback(false);
                        }
                        return false;
                    } else {
                        api(url, method, params, callback, auth);
                    }
                });
            }
            // user is authorized so execute API
            else {
                // VALIDATE INPUT
                if (method != "POST" && method != "GET") {
                    return false;
                }
                if (params == null || typeof (params) == "undefined") 
{
                    params = "";
                }
                // VARIABLES
                var initparams = params;
                if (params != null) {
                    params = params + "&";
                }
                if (cfg.access_token != '') {
                    params = params + "oauth_token=" + 
cfg.access_token;
                }

                var message = set_message(url, method, params);

                OAuth.SignatureMethod.sign(message, accessor);
```

```
                        // if we are getting request tokens, all params have 
to be set in URL
                        if ( typeof (setUrlParams) != 'undefined' && 
setUrlParams == true) {

                              finalUrl = OAuth.addToURL(message.action, 
message.parameters);

                        }
                        // for all other requests only custom params need set 
in the URL
                        else {

                              finalUrl = OAuth.addToURL(message.action, 
initparams);

                        }
                        var XHR = Ti.Network.createHTTPClient();
                        // on success, grab the request token
                        XHR.onload = function() {
                              // execute the callback function
                              if ( typeof (callback) == 'function') {
                                    callback(XHR.responseText);
                              }
                              return XHR.responseText;
                        };
                        // o error, show message
                        XHR.onerror = function(e) {
                              //alert('Can not open Twiitter dialog....try 
again');
                              // execute the callback function
                              Titanium.API.info('-XHR ERROR--' + 
JSON.stringify(e));
                              if ( typeof (callback) == 'function') {
                                    callback(false);
                              }
                              return false;
                        };
                        Ti.API.info('---the URL------' + finalUrl);
                        XHR.open(method, finalUrl, false);

                        // if we are getting request tokens do not set the 
HTML header
                        if ( typeof (setHeader) == 'undefined' || setHeader 
== true) {
                              var init = true;
```

```
                                        var header = "OAuth ";
                                        for (var i = 0; i < message.parameters.length;
i++) {
                                                if (init) {
                                                        init = false;
                                                } else {
                                                        header = header + ",";
                                                }
                                                header = header + message.parameters[i][0]
+ '="' + escape(message.parameters[i][1]) + '"';
                                        }
                                        XHR.setRequestHeader("Authorization", header);
                                }
                                Titanium.API.info('------' +
JSON.stringify(message));
                                XHR.send();
                        }
                } catch(ex) {
                        Titanium.API.info('-----Exception in API ' + ex);
                }
        }
```

```
        // ---------------------------------------------------------
        //
        //
        //
        //
        //                      API 3 start
        //
        //                      This will take arguments from function and Return the
response
        //
        // ---------------------------------------------------------
```

```
        var api3 = function(url, method, params, callback) {
                try {
                        var finalUrl = '';
                        if ( typeof (params) == 'function' && typeof (callback) ==
'undefined') {
                                callback = params;
                                params = '';
                        }
                        var initparams = params;
                        if (params != null) {
                                params = params + "&";
                        }
```

```
                    var message = set_message(url, method, params);
                    message.parameters.push(['oauth_token', cfg.access_token]);
                    OAuth.SignatureMethod.sign(message, accessor);
                    finalUrl = OAuth.addToURL(message.action, initparams);
                    Ti.API.info("My Final URL-:" + finalUrl);
                    var XHR = Ti.Network.createHTTPClient();
                    XHR.open(method, finalUrl, false);
                    XHR.setRequestHeader("Content-Type", "application/json; 
charset=utf-8");
                    XHR.setTimeout(1000);
                    XHR.onload = function() {
                        Ti.API.info("My twitts-:" + XHR.responseText);
                        if (callback) {
                            callback(XHR.responseText);
                        }
                    };
                    XHR.onerror = function(e) {
                        Ti.API.info("XHR.onerror get twitts : " + 
JSON.stringify(e));
                        if (callback) {
                            callback(false);
                        }
                    };
                    var init = true;
                    var header = "OAuth ";
                    for (var i = 0; i < message.parameters.length; i++) {
                        if (init) {
                            init = false;
                        } else {
                            header = header + ",";
                        }
                        header = header + message.parameters[i][0] + '="' + 
escape(message.parameters[i][1]) + '"';
                    }
                    header = OAuth.getAuthorizationHeader("", 
message.parameters);
                    XHR.setRequestHeader("Authorization", header);
                    if (method == "POST") {
                        XHR.send(params);
                    } else {
                        XHR.send();
                    }

                } catch(ex) {
                    Titanium.API.info('----Exception in API 3-------' + ex);
                }
```

```
        };

        function get_Myfollowers(params, callback) {
                try {
                        api3('https://api.twitter.com/1.1/followers/list.json',
"GET", params, function(resp) {
                                Titanium.API.info('-------------GET Tweets Response:'
+ resp);
                                if (resp != "" && resp != 0 && resp != undefined &&
resp != false) {
                                        Ti.API.info("fn-get_Myfollowers: response was "
+ resp + '--------------');
                                        if ( typeof (callback) == 'function') {
                                                callback(resp);
                                        }
                                        return resp;
                                } else {

                                        Ti.API.info("Failed to send tweet." +
'------------------' + JSON.stringify(resp));

                                        if ( typeof (callback) == 'function') {
                                                callback(resp);
                                        }
                                        return resp;
                                }
                        });
                } catch(ex) {
                        Ti.API.info('---------exception in get_Myfollowers-------'
+ ex);
                }
        }

        function get_Myfollowing(params, callback) {
                try {
                        api3('https://api.twitter.com/1.1/friends/list.json',
"GET", params, function(resp) {
                                Titanium.API.info('-------------GET Tweets Response:'
+ resp);
                                if (resp != "" && resp != 0 && resp != undefined &&
resp != false) {
                                        Ti.API.info("fn-get_Myfollowing: response was "
+ resp + '--------------');
                                        if ( typeof (callback) == 'function') {
                                                callback(resp);
                                        }
```

```
                            return resp;
                        } else {

                            Ti.API.info("Failed to send tweet." +
'-----------------' + JSON.stringify(resp));

                            if ( typeof (callback) == 'function') {
                                callback(resp);
                            }
                            return resp;
                        }
                    });
                } catch(ex) {

                    Titanium.API.info('----Exception in get_Myfollowing-------'
+ ex);
                }

        }

        function get_Mytweets(params, callback) {
                try {

        api3('https://api.twitter.com/1.1/statuses/user_timeline.json', "GET",
params, function(resp) {
                        Titanium.API.info('-------------GET Tweets Response:'
+ resp);
                        if (resp != "" && resp != 0 && resp != undefined &&
resp != false) {
                            Ti.API.info("fn-Get_tweet: response was " + resp
+ '-------------');
                            if ( typeof (callback) == 'function') {
                                callback(resp);
                            }
                            return resp;
                        } else {

                            Ti.API.info("Failed to send tweet." +
'-----------------' + JSON.stringify(resp));

                            if ( typeof (callback) == 'function') {
                                callback(resp);
                            }
                            return resp;
                        }
                    });
                } catch(ex) {
                    Ti.API.info('-------exception in get_Mytweets------' + ex);
                }
        }

        // ---------------------------------------------------------
        //
        //
        //
        //
        //               API 3 Ends
        //
        //
        //
        // ---------------------------------------------------------

        /*
        // ---------------------------------------------------------
        //
        //
        //
        //
        //              it self function for My Tweets and My folowers
        //
        //
        //
        // ---------------------------------------------------------

        function get_Myfollowers(params, callback) {
        var finalUrl = '';
        if ( typeof (params) == 'function' && typeof (callback) == 
'undefined') {
        callback = params;
        params = '';
        }
        var url = 'https://api.twitter.com/1.1/followers/list.json';
        var initparams = params;
        if (params != null) {
        params = params + "&";
        }
        var message = set_message(url, "GET", params);
        message.parameters.push(['oauth_token', cfg.access_token]);
        OAuth.SignatureMethod.sign(message, accessor);
        finalUrl = OAuth.addToURL(message.action, initparams);
        Ti.API.info("My Final URL-:" + finalUrl);
        var XHR = Ti.Network.createHTTPClient();
```

```
	XHR.open("GET", finalUrl);
	XHR.setRequestHeader("Content-Type", "application/json; charset=utf-
8");
	XHR.onload = function() {
	Ti.API.info("My twitts-:" + XHR.responseText);
	if (callback) {
	callback(XHR.responseText);
	}
	};

	XHR.onerror = function(e) {

	Ti.API.info("XHR.onerror get twitts : " + JSON.stringify(e));
	alert('Erro Occurs Please Try Again....');
	if (callback) {
	callback(e);
	}
	};
	if ( typeof (setHeader) == 'undefined' || setHeader == true) {
	var init = true;
	var header = "OAuth ";
	for (var i = 0; i < message.parameters.length; i++) {
	if (init) {
	init = false;
	} else {
	header = header + ",";
	}
	header = header + message.parameters[i][0] + '="' +
escape(message.parameters[i][1]) + '"';
	}
	header = OAuth.getAuthorizationHeader("", message.parameters);
	Titanium.API.info('op');
	Titanium.API.info(header);
	XHR.setRequestHeader("Authorization", header);
	Titanium.API.info('op1');
	}
	XHR.send();
	};

	// ---------------------------------------------------
	// get_tweets
	//

	function get_Mytweets(params, callback) {
	var finalUrl = '';
	if ( typeof (params) == 'function' && typeof (callback) ==
'undefined') {
     callback = params;
     params = '';
     }

     var url = 'https://api.twitter.com/1.1/statuses/user_timeline.json';
     var initparams = params;
     if (params != null) {
     params = params + "&";
     }
     var message = set_message(url, "GET", params);
     message.parameters.push(['oauth_token', cfg.access_token]);
     OAuth.SignatureMethod.sign(message, accessor);
     finalUrl = OAuth.addToURL(message.action, initparams);
     Ti.API.info("My Final URL-:" + finalUrl);
     var XHR = Ti.Network.createHTTPClient();
     XHR.open("GET", finalUrl, false);
     XHR.setRequestHeader("Content-Type", "application/json; charset=utf-
8");
     XHR.setTimeout(1000);
     XHR.onload = function() {
     Ti.API.info("My twitts-:" + XHR.responseText);
     if (callback) {
     callback(XHR.responseText);
     }
     };

     XHR.onerror = function(e) {
     Ti.API.info("XHR.onerror get twitts : " + JSON.stringify(e));
     alert('Erro Occurs Please Try Again....');
     if (callback) {
     callback(e);
     }
     };
     if ( typeof (setHeader) == 'undefined' || setHeader == true) {
     var init = true;
     var header = "OAuth ";
     for (var i = 0; i < message.parameters.length; i++) {
     if (init) {
     init = false;
     } else {
     header = header + ",";
     }
     header = header + message.parameters[i][0] + '="' +
escape(message.parameters[i][1]) + '"';
     }
```

```
    header = OAuth.getAuthorizationHeader("", message.parameters);
    Titanium.API.info('op');
    Titanium.API.info(header);
    XHR.setRequestHeader("Authorization", header);
    Titanium.API.info('op1');
    }
    XHR.send();
    };

    // ---------------------------------------------------
    //
    //
    //
    //
    //             Ends  it self functiosns for My Tweets and My folowers
    //
    //
    //
    // ---------------------------------------------------
    */
    // ---------------------------------------------------
    // send_tweet
    //
    // Makes an API call to Twitter to post a tweet.
    //
    // In Parameters:
    //   params (String) - the string of optional and
    //     required parameters in url form
    //   callback (Function) - function to call on completion

    // ---------------------------------------------------
    function send_tweet(params, callback) {
        try {
            api3('https://api.twitter.com/1.1/statuses/update.json', 
"POST", params, function(resp) {
                if (resp != false) {
                    Ti.API.debug("fn-send_tweet: response was " + 
resp + '-------------');
                    if ( typeof (callback) == 'function') {
                        callback(true);
                    }
                    return true;
                } else {
                    Ti.API.info("Failed to send tweet." + 
'-----------------');
                    if ( typeof (callback) == 'function') {
```

```
                                callback(false);
                           }
                           return false;
                      }
                 });
            } catch(ex) {
                 Titanium.API.info('------exception in send_tweet-----' +
ex);
            }
       }

       function send_message(params, callback) {
            try {

       api3('https://api.twitter.com/1.1/direct_messages/new.json', "POST",
params, function(resp) {
                      Titanium.API.info('-------------GET Tweets Response:'
+ resp);
                      if (resp != "" && resp != 0 && resp != undefined &&
resp != false) {
                           Ti.API.info("fn-send_message: response was " +
resp + '-------------');
                           if ( typeof (callback) == 'function') {
                                callback(resp);
                           }
                           return resp;
                      } else {

                           Ti.API.info("Failed to send tweet." +
'-----------------' + JSON.stringify(resp));

                           if ( typeof (callback) == 'function') {
                                callback(false);
                           }
                           return false;
                      }
                 });
            } catch(ex) {
                 Ti.API.info('-------exception in get_Mytweets------' + ex);
            }
       }

       var sendTwitterImage = function(params, pSuccessCallback,
pErrorCallback) {
            var finalUrl = '';
            if (!authorized && ( typeof (auth) == 'undefined' || auth ===
true)) {
                 authorize(function(retval) {
                      if (!retval) {
                           if ( typeof (callback) == 'function') {
                                callback(false);
                           }
                           return false;
                      } else {
                           sendTwitterImage(postParams, pSuccessCallback, 
pErrorCallback);
                      }
                 });
            } else {
                 var url = 
'https://api.twitter.com/1.1/statuses/update_with_media.json';
                 var initparams = params;
                 if (params != null) {
                      params = params + "&";
                 }
                 var message = set_message(url, "POST");
                 message.parameters.push(['oauth_token', cfg.access_token]);
                 OAuth.SignatureMethod.sign(message, accessor);
                 var XHR = Ti.Network.createHTTPClient();
                 XHR.open("POST", url);
                 XHR.setTimeout(1000);
                 XHR.onload = function() {
                      Ti.API.info("--------Successfully imafge share---- " 
+ XHR.responseText);
                      if (pSuccessCallback) {
                           pSuccessCallback(true);
                      }
                 };
                 if (OS_IOS) {
                      XHR.setRequestHeader('Content-Type', 'multipart/form-
data');
                 } else {
                      XHR.setRequestHeader('enctype', 'multipart/form-
data');
                 }
                 XHR.onerror = function(e) {
                      Ti.App.hideIndicator();
                      Ti.API.info("XHR.onerror Twitter Image share : " + 
JSON.stringify(e));
                      if (pErrorCallback) {
                           pErrorCallback(false);
                      }
```

```
            };
            if ( typeof (setHeader) == 'undefined' || setHeader == 
true) {
                var init = true;
                var header = "OAuth ";
                for (var i = 0; i < message.parameters.length; i++) {
                    if (init) {
                        init = false;
                    } else {
                        header = header + ",";
                    }
                    header = header + message.parameters[i][0] + 
'="' + escape(message.parameters[i][1]) + '"';
                }
                header = OAuth.getAuthorizationHeader("", 
message.parameters);
                XHR.setRequestHeader("Authorization", header);
            }
            XHR.send(postParams);
        }
    };
    // -----------------------------------------------------
    // shorten_url
    //
    // Shortens a URL using twe.ly.
    //
    // In Parameters:
    //   url (String) - the url to shorten
    //
    // Returns:
    //   shorturl (String) - the shortened URL, else false
    //   callback (Function) - function to call on completion
    // -----------------------------------------------------
    function shorten_url(url, callback) {
        var XHR = Titanium.Network.createHTTPClient();
        XHR.open("GET", "https://www.twe.ly/short.php?url=" + url + 
"&json=1");
        XHR.onload = function() {
            try {
                shorturl = JSON.parse(XHR.responseText);
            } catch(e) {
                shorturl = false;
            }
            if (shorturl != false && shorturl.substr(0, 5) == 'Sorry') 
{
                shorturl = false;
                }
                if ( typeof (callback) == 'function') {
                     callback(shorturl, url);
                }
                return shorturl;
           };
           XHR.onerror = function(e) {

                if ( typeof (callback) == 'function') {
                     callback(false);
                }
                return false;
           };
           XHR.send();
      }

      // authorize
      //
      // The whole authorization sequence begins with
      // get_request_token(), which calls get_request_verifier()
      // which finally calls get_access_token() which then
      // saves the token in a file.
      //
      // In Parameters:
      //    callback (Function) - a function to call after
      //      the user has been authorized; note that it won't
      //      be executed until get_access_token(), unless we
      //      are already authorized.
      //
      // Returns: true if the user is authorized
      // ----------------------------------------------------
      function authorize(callback) {
           if (!authorized) {
                get_request_token(callback);
                // get_request_token or a function it calls will call 
callback
           } else {
                // execute the callback function
                if ( typeof (callback) == 'function') {
                     callback(authorized);
                }
           }
           return authorized;
      }

      // ----------------------------------------------------
```

```
	// deauthorize
	//
	// Delete the stored access token file, delete the tokens
	// from the config and accessor, and set authorized to
	// load_access_token() which should return false since
	// we deleted the file, thus resulting in a deauthroized
	// state.
	//
	// In Parameters:
	//    callback (Function) - function to call after
	//      user is deauthorized
	//
	// Returns: true if the user is deauthorized
	// ------------------------------------------------------------
	function deauthorize(callback) {
		if (authorized) {
			var file = 
Ti.Filesystem.getFile(Ti.Filesystem.applicationDataDirectory, 
'twitter.config');
			file.deleteFile();
			authorized = load_access_token();
			accessor.tokenSecret = "";
			cfg.access_token = "";
			cfg.access_token_secret = "";
			cfg.request_verifier = "";
			// execute the callback function
			if ( typeof (callback) == 'function') {
				callback(!authorized);
			}
			var client = Titanium.Network.createHTTPClient();
			client.clearCookies('https://twitter.com/login/');
		} else {
			// execute the callback function
			if ( typeof (callback) == 'function') {
				callback(!authorized);
			}
		}
		return !authorized;
	}


	this.get_Myfollowers = get_Myfollowers;
	this.sendTwitterImage = sendTwitterImage;
	this.get_Myfollowing = get_Myfollowing;
	this.send_message = send_message;
	this.get_Mytweets = get_Mytweets;
```

```
    this.authorize = authorize;
    this.deauthorize = deauthorize;
    this.api = api;
    this.screen_name = cfg.screen_name;
    this.user_id = cfg.user_id;
    this.send_tweet = send_tweet;
    this.authorized = function() {
        return authorized;
    };

    // ---------------------------------------------------------
    // =================== INITIALIZE ==========================
    // ---------------------------------------------------------
    if ( typeof params == 'object') {
        if (params.consumer_key != undefined) {
            cfg.oauth_consumer_key = params.consumer_key;
        }
        if (params.consumer_secret != undefined) {
            cfg.consumer_secret = params.consumer_secret;
            accessor.consumerSecret = cfg.consumer_secret;
        }
        if (params.callback_url != undefined) {
            cfg.callback_url = params.callback_url;
        }
        if (params.show_login_toolbar != undefined) {
            cfg.show_login_toolbar = params.show_login_toolbar;
        }
    }
    authorized = load_access_token();
    // load the token on startup to see if authorized
};
```

oauth.js

```
/*jslint maxerr:1000 */
/*
* Copyright 2008 Netflix, Inc.
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

/* Here's some JavaScript software for implementing OAuth.

This isn't as useful as you might hope.  OAuth is based around
allowing tools and websites to talk to each other.  However,
JavaScript running in web browsers is hampered by security
restrictions that prevent code running on one website from
accessing data stored or served on another.
```

```
Before you start hacking, make sure you understand the limitations
posed by cross-domain XMLHttpRequest.

On the bright side, some platforms use JavaScript as their
language, but enable the programmer to access other web sites.
Examples include Google Gadgets, and Microsoft Vista Sidebar.
For those platforms, this library should come in handy.
*/

// The HMAC-SHA1 signature method calls b64_hmac_sha1, defined by
// http://pajhome.org.uk/crypt/md5/sha1.js

/* An OAuth message is represented as an object like this:
 {method: "GET", action: "http://server.com/path", parameters: ...}

 The parameters may be either a map {name: value, name2: value2}
 or an Array of name-value pairs [[name, value], [name2, value2]].
 The latter representation is more powerful: it supports parameters
 in a specific sequence, or several parameters with the same name;
 for example [["a", 1], ["b", 2], ["a", 3]].

 Parameter names and values are NOT percent-encoded in an object.
 They must be encoded before transmission and decoded after reception.
 For example, this message object:
 {method: "GET", action: "http://server/path", parameters: {p: "x y"}}
 ... can be transmitted as an HTTP request that begins:
 GET /path?p=x%20y HTTP/1.0
 (This isn't a valid OAuth request, since it lacks a signature etc.)
 Note that the object "x y" is transmitted as x%20y.  To encode
 parameters, you can call OAuth.addToURL, OAuth.formEncode or
 OAuth.getAuthorization.

 This message object model harmonizes with the browser object model for
 input elements of an form, whose value property isn't percent encoded.
 The browser encodes each value before transmitting it. For example,
 see consumer.setInputs in example/consumer.js.
 */

/* This script needs to know what time it is. By default, it uses the local
 clock (new Date), which is apt to be inaccurate in browsers. To do
 better, you can load this script from a URL whose query string contains
 an oauth_timestamp parameter, whose value is a current Unix timestamp.
 For example, when generating the enclosing document using PHP:

 <script src="oauth.js?oauth_timestamp=<?=time()?>" ...
 Another option is to call OAuth.correctTimestamp with a Unix timestamp.
 */

var OAuth;
if (OAuth == null)
    OAuth = {};

OAuth.setProperties = function setProperties(into, from) {
    if (into != null && from != null) {
        for (var key in from) {
            into[key] = from[key];
        }
    }
    return into;
};

OAuth.setProperties(OAuth, // utility functions
{
    percentEncode : function percentEncode(s) {
        if (s == null) {
            return "";
        }
        if ( s instanceof Array) {
            var e = "";
            for (var i = 0; i < s.length; ++s) {
                if (e != "")
                    e += '&';
                e += OAuth.percentEncode(s[i]);
            }
            return e;
        }
        s = encodeURIComponent(s);
        // Now replace the values which encodeURIComponent doesn't do
        // encodeURIComponent ignores: - _ . ! ~ * ' ( )
        // OAuth dictates the only ones you can ignore are: - _ . ~
        // Source: 
http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Reference:Global_F
unctions:encodeURIComponent
        s = s.replace(/\!/g, "%21");
        s = s.replace(/\*/g, "%2A");
        s = s.replace(/\'/g, "%27");
        s = s.replace(/\(/g, "%28");
        s = s.replace(/\)/g, "%29");
        return s;
    },
	decodePercent : function decodePercent(s) {
		if (s != null) {
			// Handle application/x-www-form-urlencoded, which is
defined by
			// http://www.w3.org/TR/html4/interact/forms.html#h-
17.13.4.1
			s = s.replace(/\+/g, " ");
		}
		Titanium.API.debug(s);
		return decodeURIComponent(s);
	},
	/** Convert the given parameters to an Array of name-value pairs. */
	getParameterList : function getParameterList(parameters) {
		if (parameters == null) {
			return [];
		}
		if ( typeof parameters != "object") {
			return OAuth.decodeForm(parameters + "");
		}
		if ( parameters instanceof Array) {
			return parameters;
		}
		var list = [];
		for (var p in parameters) {
			list.push([p, parameters[p]]);
		}
		return list;
	},
	/** Convert the given parameters to a map from name to value. */
	getParameterMap : function getParameterMap(parameters) {
		if (parameters == null) {
			return {};
		}
		if ( typeof parameters != "object") {
			return OAuth.getParameterMap(OAuth.decodeForm(parameters +
""));
		}
		if ( parameters instanceof Array) {
			var map = {};
			for (var p = 0; p < parameters.length; ++p) {
				var key = parameters[p][0];
				if (map[key] === undefined) {// first value wins
					map[key] = parameters[p][1];
				}
			}
			return map;
```

```
        }
        return parameters;
    },
    getParameter : function getParameter(parameters, name) {
        if ( parameters instanceof Array) {
            for (var p = 0; p < parameters.length; ++p) {
                if (parameters[p][0] == name) {
                    return parameters[p][1];
                    // first value wins
                }
            }
        } else {
            return OAuth.getParameterMap(parameters)[name];
        }
        return null;
    },
    formEncode : function formEncode(parameters) {
        var form = "";
        var list = OAuth.getParameterList(parameters);
        for (var p = 0; p < list.length; ++p) {
            var value = list[p][1];
            if (value == null)
                value = "";
            if (form != "")
                form += '&';
            form += OAuth.percentEncode(list[p][0]) + '=' + 
OAuth.percentEncode(value);
        }
        return form;
    },
    decodeForm : function decodeForm(form) {
        var list = [];
        var nvps = form.split('&');
        for (var n = 0; n < nvps.length; ++n) {
            var nvp = nvps[n];
            if (nvp == "") {
                continue;
            }
            var equals = nvp.indexOf('=');
            var name;
            var value;
            if (equals < 0) {
                name = OAuth.decodePercent(nvp);
                value = null;
            } else {
                name = OAuth.decodePercent(nvp.substring(0, equals));
```

```
                        value = OAuth.decodePercent(nvp.substring(equals +
1));
                  }
                  list.push([name, value]);
            }
            return list;
      },
      setParameter : function setParameter(message, name, value) {
            var parameters = message.parameters;
            if ( parameters instanceof Array) {
                  for (var p = 0; p < parameters.length; ++p) {
                        if (parameters[p][0] == name) {
                              if (value === undefined) {
                                    parameters.splice(p, 1);
                              } else {
                                    parameters[p][1] = value;
                                    value = undefined;
                              }
                        }
                  }
                  if (value !== undefined) {
                        parameters.push([name, value]);
                  }
            } else {
                  parameters = OAuth.getParameterMap(parameters);
                  parameters[name] = value;
                  message.parameters = parameters;
            }
      },
      setParameters : function setParameters(message, parameters) {
            var list = OAuth.getParameterList(parameters);
            for (var i = 0; i < list.length; ++i) {
                  OAuth.setParameter(message, list[i][0], list[i][1]);
            }
      },

      /** Fill in parameters to help construct a request message.
       This function doesn't fill in every parameter.
       The accessor object should be like:
       {consumerKey:'foo', consumerSecret:'bar', accessorSecret:'nurn',
token:'krelm', tokenSecret:'blah'}
       The accessorSecret property is optional.
       */
      completeRequest : function completeRequest(message, accessor) {
            if (message.method == null) {
                  message.method = "GET";
```

```
        }
        var map = OAuth.getParameterMap(message.parameters);

        if (map.oauth_consumer_key == null) {
            OAuth.setParameter(message, "oauth_consumer_key", 
accessor.consumerKey || "");
        }
        if (map.oauth_token == null && accessor.token != null) {
            OAuth.setParameter(message, "oauth_token", accessor.token);
        }
        if (map.oauth_version == null) {
            OAuth.setParameter(message, "oauth_version", "1.0");
        }
        if (map.oauth_timestamp == null) {
            OAuth.setParameter(message, "oauth_timestamp", 
OAuth.timestamp());
        }
        if (map.oauth_nonce == null) {
            OAuth.setParameter(message, "oauth_nonce", OAuth.nonce(6));
        }
        OAuth.SignatureMethod.sign(message, accessor);
    },
    setTimestampAndNonce : function setTimestampAndNonce(message) {


Titanium.API.info('-------------------------------------------------------
----------setTimestampAndNonce---------------------------------');
        OAuth.setParameter(message, "oauth_timestamp", 
OAuth.timestamp());
        OAuth.setParameter(message, "oauth_nonce", OAuth.nonce(6));
    },
    addToURL : function addToURL(url, parameters) {
        newURL = url;
        if (parameters != null) {
            var toAdd = OAuth.formEncode(parameters);
            if (toAdd.length > 0) {
                var q = url.indexOf('?');
                if (q < 0)
                    newURL += '?';
                else
                    newURL += '&';
                newURL += toAdd;
            }
        }
        Titanium.API.info('THE NEWUSRL----------' + newURL);
        return newURL;
```

```
	},
	/** Construct the value of the Authorization header for an HTTP 
request. */
	getAuthorizationHeader : function getAuthorizationHeader(realm, 
parameters) {
		var header = 'OAuth realm="' + OAuth.percentEncode(realm) + '"';
		var list = OAuth.getParameterList(parameters);
		for (var p = 0; p < list.length; ++p) {
			var parameter = list[p];
			var name = parameter[0];
			if (name.indexOf("oauth_") == 0) {
				header += ',' + OAuth.percentEncode(name) + '="' + 
OAuth.percentEncode(parameter[1]) + '"';
			}
		}
		return header;
	},
	/** Correct the time using a parameter from the URL from which the 
last script was loaded. */
	correctTimestampFromSrc : function 
correctTimestampFromSrc(parameterName) {
		parameterName = parameterName || "oauth_timestamp";
		if (document === undefined) {
			return;
		}
		var scripts = document.getElementsByTagName('script');
		if (scripts == null || !scripts.length)
			return;
		var src = scripts[scripts.length - 1].src;
		if (!src)
			return;
		var q = src.indexOf("?");
		if (q < 0)
			return;
		parameters = 
OAuth.getParameterMap(OAuth.decodeForm(src.substring(q + 1)));
		var t = parameters[parameterName];
		if (t == null)
			return;
		OAuth.correctTimestamp(t);
	},
	/** Generate timestamps starting with the given value. */
	correctTimestamp : function correctTimestamp(timestamp) {
		OAuth.timeCorrectionMsec = (timestamp * 1000) - (new 
Date()).getTime();
	},
    /** The difference between the correct time and my clock. */
    timeCorrectionMsec : 0,
    timestamp : function timestamp() {
        var t = (new Date()).getTime() + OAuth.timeCorrectionMsec;
        return Math.floor(t / 1000);
    },
    nonce : function nonce(length) {
        var chars = OAuth.nonce.CHARS;
        var result = "";
        for (var i = 0; i < length; ++i) {
            var rnum = Math.floor(Math.random() * chars.length);
            result += chars.substring(rnum, rnum + 1);
        }
        return result;
    }
});

OAuth.nonce.CHARS =
"0123456789ABCDEFGHIJKLMNOPQRSTUVWXTZabcdefghiklmnopqrstuvwxyz";

/** Define a constructor function,
 without causing trouble to anyone who was using it as a namespace.
 That is, if parent[name] already existed and had properties,
 copy those properties into the new constructor.
 */
OAuth.declareClass = function declareClass(parent, name, newConstructor) {
    var previous = parent[name];
    parent[name] = newConstructor;
    if (newConstructor != null && previous != null) {
        for (var key in previous) {
            if (key != "prototype") {
                newConstructor[key] = previous[key];
            }
        }
    }
    return newConstructor;
};

/** An abstract algorithm for signing messages. */
OAuth.declareClass(OAuth, "SignatureMethod", function
OAuthSignatureMethod() {
});

OAuth.setProperties(OAuth.SignatureMethod.prototype, // instance members
{
    /** Add a signature to the message. */
	sign : function sign(message) {
		var baseString = OAuth.SignatureMethod.getBaseString(message);
		var signature = this.getSignature(baseString);
		OAuth.setParameter(message, "oauth_signature", signature);
		return signature;
		// just in case someone's interested
	},
	/** Set the key string for signing. */
	initialize : function initialize(name, accessor) {
		var consumerSecret;
		if (accessor.accessorSecret != null && name.length > 9 && 
name.substring(name.length - 9) == "-Accessor") {
			consumerSecret = accessor.accessorSecret;
		} else {
			consumerSecret = accessor.consumerSecret;
		}
		this.key = OAuth.percentEncode(consumerSecret) + "&" + 
OAuth.percentEncode(accessor.tokenSecret);
	}
});

/* SignatureMethod expects an accessor object to be like this:
{tokenSecret: "lakjsdflkj...", consumerSecret: "QOUEWRI..", accessorSecret: 
"xcmvzc..."}
The accessorSecret property is optional.
*/
// Class members:
OAuth.setProperties(OAuth.SignatureMethod, // class members
{
	sign : function sign(message, accessor) {
		var name = 
OAuth.getParameterMap(message.parameters).oauth_signature_method;
		if (name == null || name == "") {
			name = "HMAC-SHA1";
			OAuth.setParameter(message, "oauth_signature_method", 
name);
		}
		OAuth.SignatureMethod.newMethod(name, accessor).sign(message);
	},
	/** Instantiate a SignatureMethod for the given method name. */
	newMethod : function newMethod(name, accessor) {
		var impl = OAuth.SignatureMethod.REGISTERED[name];
		if (impl != null) {
			var method = new impl();
			method.initialize(name, accessor);
			return method;
```

```
        }
        var err = new Error("signature_method_rejected");
        var acceptable = "";
        for (var r in OAuth.SignatureMethod.REGISTERED) {
            if (acceptable != "")
                acceptable += '&';
            acceptable += OAuth.percentEncode(r);
        }
        err.oauth_acceptable_signature_methods = acceptable;
        throw err;
    },
    /** A map from signature method name to constructor. */
    REGISTERED : {},
    /** Subsequently, the given constructor will be used for the named 
methods.
     The constructor will be called with no parameters.
     The resulting object should usually implement 
getSignature(baseString).
     You can easily define such a constructor by calling makeSubclass, 
below.
     */
    registerMethodClass : function registerMethodClass(names, 
classConstructor) {
        for (var n = 0; n < names.length; ++n) {
            OAuth.SignatureMethod.REGISTERED[names[n]] = 
classConstructor;
        }
    },
    /** Create a subclass of OAuth.SignatureMethod, with the given 
getSignature function. */
    makeSubclass : function makeSubclass(getSignatureFunction) {
        var superClass = OAuth.SignatureMethod;
        var subClass = function() {
            superClass.call(this);
        };
        subClass.prototype = new superClass();
        // Delete instance variables from prototype:
        // delete subclass.prototype... There aren't any.
        subClass.prototype.getSignature = getSignatureFunction;
        subClass.prototype.constructor = subClass;
        return subClass;
    },
    getBaseString : function getBaseString(message) {
        var URL = message.action;
        var q = URL.indexOf('?');
        var parameters;
        if (q < 0) {
            parameters = message.parameters;
        } else {
            // Combine the URL query string with the other parameters:
            parameters = OAuth.decodeForm(URL.substring(q + 1));
            var toAdd = OAuth.getParameterList(message.parameters);
            for (var a = 0; a < toAdd.length; ++a) {
                parameters.push(toAdd[a]);
            }
        }
        return OAuth.percentEncode(message.method.toUpperCase()) + '&' +
OAuth.percentEncode(OAuth.SignatureMethod.normalizeUrl(URL)) + '&' +
OAuth.percentEncode(OAuth.SignatureMethod.normalizeParameters(parameters));
    },
    normalizeUrl : function normalizeUrl(url) {
        var uri = OAuth.SignatureMethod.parseUri(url);
        var scheme = uri.protocol.toLowerCase();
        var authority = uri.authority.toLowerCase();
        var dropPort = (scheme == "http" && uri.port == 80) || (scheme
== "https" && uri.port == 443);
        if (dropPort) {
            // find the last : in the authority
            var index = authority.lastIndexOf(":");
            if (index >= 0) {
                authority = authority.substring(0, index);
            }
        }
        var path = uri.path;
        if (!path) {
            path = "/";
            // conforms to RFC 2616 section 3.2.2
        }
        // we know that there is no query and no fragment here.
        return scheme + "://" + authority + path;
    },
    /*parseUri : function parseUri(str) {
     // This function was adapted from parseUri 1.2.1
     // http://stevenlevithan.com/demo/parseuri/js/assets/parseuri.js

     var o = {
     key : ["source", "protocol", "authority", "userInfo", "user",
"password", "host", "port", "relative", "path", "directory", "file",
"query", "anchor"],
     parser : {
     strict : /^(?:([^:\/?#]+):)?(?:\/\/((?:(([^:@\/]*):?([^:@\/]*))?@)?
([^:\/?#]*)(?::(\d*))?))?((((?:[^?#\/]*\/)*)([^?#]*))(?:\?([^#]*))?
(?:#(.*))?)/
     }
     };
     var m = o.parser.strict.exec(str);
     var uri = {};
     var i = 14;
     while (i--)
     uri[o.key[i]] = m[i] || "";
     return uri;
     },*/

    parseUri : function parseUri(str) {

        options = {
            strictMode : false,
            key : ["source", "protocol", "authority", "userInfo", 
"user", "password", "host", "port", "relative", "path", "directory", 
"file", "query", "anchor"],
            q : {
                name : "queryKey",
                parser : /(?:^|&)([^&=]*)=?([^&]*)/g
            },
            parser : {
                strict : /^(?:([^:\/?#]+):)?(?:\/\/((?:(([^:@]*)(?::
([^:@]*))?)?@)?([^:\/?#]*)(?::(\d*))?))?((((?:[^?#\/]*\/)*)([^?#]*))(?:\?
([^#]*))?(?:#(.*))?)/,
                loose : /^(?:(?![^:@]+:[^:@\/]*@)([^:\/?#.]+):)?
(?:\/\/)?((?:(([^:@]*)(?::([^:@]*))?)?@)?([^:\/?#]*)(?::(\d*))?)(((\/(?:[^?
#](?![^?#\/]*\.[^?#\/.]+(?:[?#]|$)))*\/?)?([^?#\/]*))(?:\?([^#]*))?
(?:#(.*))?)/
            }
        };
        var o = options, m = o.parser[o.strictMode ? "strict" : 
"loose"].exec(str), uri = {}, i = 14;

        while (i--)
        uri[o.key[i]] = m[i] || "";

        uri[o.q.name] = {};
        uri[o.key[12]].replace(o.q.parser, function($0, $1, $2) {
            if ($1)
                uri[o.q.name][$1] = $2;
        });

        return uri;
    },
    normalizeParameters : function normalizeParameters(parameters) {
        if (parameters == null) {
            return "";
        }
        var list = OAuth.getParameterList(parameters);
        var sortable = [];
        for (var p = 0; p < list.length; ++p) {
            var nvp = list[p];
            if (nvp[0] != "oauth_signature") {
                sortable.push([OAuth.percentEncode(nvp[0]) + " "// 
because it comes before any character that can appear in a percentEncoded 
string.
                             + OAuth.percentEncode(nvp[1]), nvp]);
            }
        }
        sortable.sort(function(a, b) {
            if (a[0] < b[0])
                return -1;
            if (a[0] > b[0])
                return 1;
            return 0;
        });
        var sorted = [];
        for (var s = 0; s < sortable.length; ++s) {
            sorted.push(sortable[s][1]);
        }
        return OAuth.formEncode(sorted);
    }
});

OAuth.SignatureMethod.registerMethodClass(["PLAINTEXT", "PLAINTEXT-
Accessor"], OAuth.SignatureMethod.makeSubclass(function 
getSignature(baseString) {
    return this.key;
}));

OAuth.SignatureMethod.registerMethodClass(["HMAC-SHA1", "HMAC-SHA1-
Accessor"], OAuth.SignatureMethod.makeSubclass(function 
getSignature(baseString) {
    b64pad = '=';
    var signature = b64_hmac_sha1(this.key, baseString);
    return signature;
}));

try {
     // OAuth.correctTimestampFromSrc();
} catch(e) {
}
```

SHA1.js

```
/*
 * A JavaScript implementation of the Secure Hash Algorithm, SHA-1, as
defined
 * in FIPS PUB 180-1
 * Version 2.1a Copyright Paul Johnston 2000 - 2002.
 * Other contributors: Greg Holt, Andrew Kepert, Ydnar, Lostinet
 * Distributed under the BSD License
 * See http://pajhome.org.uk/crypt/md5 for details.
 */

/*
 * Configurable variables. You may need to tweak these to be compatible
with
 * the server-side, but the defaults work in most cases.
 */
var hexcase = 0;  /* hex output format. 0 - lowercase; 1 - uppercase
*/
var b64pad  = ""; /* base-64 pad character. "=" for strict RFC compliance
*/
var chrsz   = 8;  /* bits per input character. 8 - ASCII; 16 - Unicode
*/

/*
 * These are the functions you'll usually want to call
 * They take string arguments and return either hex or base-64 encoded
strings
```

```
 */
function hex_sha1(s){return binb2hex(core_sha1(str2binb(s),s.length * 
chrsz));}
function b64_sha1(s){return binb2b64(core_sha1(str2binb(s),s.length * 
chrsz));}
function str_sha1(s){return binb2str(core_sha1(str2binb(s),s.length * 
chrsz));}
function hex_hmac_sha1(key, data){ return binb2hex(core_hmac_sha1(key, 
data));}
function b64_hmac_sha1(key, data){ return binb2b64(core_hmac_sha1(key, 
data));}
function str_hmac_sha1(key, data){ return binb2str(core_hmac_sha1(key, 
data));}
/*
 * Perform a simple self-test to see if the VM is working
 */
function sha1_vm_test()
{
  return hex_sha1("abc") == "a9993e364706816aba3e25717850c26c9cd0d89d";
}
/*
 * Calculate the SHA-1 of an array of big-endian words, and a bit length
 */
function core_sha1(x, len)
{
  /* append padding */
  x[len >> 5] |= 0x80 << (24 - len % 32);
  x[((len + 64 >> 9) << 4) + 15] = len;

  var w = Array(80);
  var a =  1732584193;
  var b = -271733879;
  var c = -1732584194;
  var d =  271733878;
  var e = -1009589776;

  for(var i = 0; i < x.length; i += 16)
  {
    var olda = a;
    var oldb = b;
    var oldc = c;
    var oldd = d;
    var olde = e;

    for(var j = 0; j < 80; j++)
    {
      if(j < 16) w[j] = x[i + j];
      else w[j] = rol(w[j-3] ^ w[j-8] ^ w[j-14] ^ w[j-16], 1);
      var t = safe_add(safe_add(rol(a, 5), sha1_ft(j, b, c, d)),
                       safe_add(safe_add(e, w[j]), sha1_kt(j)));
      e = d;
      d = c;
      c = rol(b, 30);
      b = a;
      a = t;
    }

    a = safe_add(a, olda);
    b = safe_add(b, oldb);
    c = safe_add(c, oldc);
    d = safe_add(d, oldd);
    e = safe_add(e, olde);
  }
  return Array(a, b, c, d, e);

}

/*
 * Perform the appropriate triplet combination function for the current
 * iteration
 */
function sha1_ft(t, b, c, d)
{
  if(t < 20) return (b & c) | ((~b) & d);
  if(t < 40) return b ^ c ^ d;
  if(t < 60) return (b & c) | (b & d) | (c & d);
  return b ^ c ^ d;
}

/*
 * Determine the appropriate additive constant for the current iteration
 */
function sha1_kt(t)
{
  return (t < 20) ?  1518500249 : (t < 40) ?  1859775393 :
         (t < 60) ? -1894007588 : -899497514;
}

/*
 * Calculate the HMAC-SHA1 of a key and some data
 */
function core_hmac_sha1(key, data)
```

```
{
  var bkey = str2binb(key);
  if(bkey.length > 16) bkey = core_sha1(bkey, key.length * chrsz);

  var ipad = Array(16), opad = Array(16);
  for(var i = 0; i < 16; i++)
  {
    ipad[i] = bkey[i] ^ 0x36363636;
    opad[i] = bkey[i] ^ 0x5C5C5C5C;
  }

  var hash = core_sha1(ipad.concat(str2binb(data)), 512 + data.length *
chrsz);
  return core_sha1(opad.concat(hash), 512 + 160);
}

/*
 * Add integers, wrapping at 2^32. This uses 16-bit operations internally
 * to work around bugs in some JS interpreters.
 */
function safe_add(x, y)
{
  var lsw = (x & 0xFFFF) + (y & 0xFFFF);
  var msw = (x >> 16) + (y >> 16) + (lsw >> 16);
  return (msw << 16) | (lsw & 0xFFFF);
}

/*
 * Bitwise rotate a 32-bit number to the left.
 */
function rol(num, cnt)
{
  return (num << cnt) | (num >>> (32 - cnt));
}

/*
 * Convert an 8-bit or 16-bit string to an array of big-endian words
 * In 8-bit function, characters >255 have their hi-byte silently ignored.
 */
function str2binb(str)
{
  var bin = Array();
  var mask = (1 << chrsz) - 1;
  for(var i = 0; i < str.length * chrsz; i += chrsz)
    bin[i>>5] |= (str.charCodeAt(i / chrsz) & mask) << (32 - chrsz - i%32);
  return bin;
```

```
}

/*
 * Convert an array of big-endian words to a string
 */
function binb2str(bin)
{
  var str = "";
  var mask = (1 << chrsz) - 1;
  for(var i = 0; i < bin.length * 32; i += chrsz)
    str += String.fromCharCode((bin[i>>5] >>> (32 - chrsz - i%32)) & mask);
  return str;
}

/*
 * Convert an array of big-endian words to a hex string.
 */
function binb2hex(binarray)
{
  var hex_tab = hexcase ? "0123456789ABCDEF" : "0123456789abcdef";
  var str = "";
  for(var i = 0; i < binarray.length * 4; i++)
  {
    str += hex_tab.charAt((binarray[i>>2] >> ((3 - i%4)*8+4)) & 0xF) +
           hex_tab.charAt((binarray[i>>2] >> ((3 - i%4)*8  )) & 0xF);
  }
  return str;
}

/*
 * Convert an array of big-endian words to a base-64 string
 */
function binb2b64(binarray)
{
  var tab = 
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";
  var str = "";
  for(var i = 0; i < binarray.length * 4; i += 3)
  {
    var triplet = (((binarray[i   >> 2] >> 8 * (3 -  i   %4)) & 0xFF) << 
16)
                | (((binarray[i+1 >> 2] >> 8 * (3 - (i+1)%4)) & 0xFF) << 8 
)
                |  ((binarray[i+2 >> 2] >> 8 * (3 - (i+2)%4)) & 0xFF);
    for(var j = 0; j < 4; j++)
    {
      if(i * 8 + j * 6 > binarray.length * 32) str += b64pad;
      else str += tab.charAt((triplet >> 6*(3-j)) & 0x3F);
    }
  }
  return str;
}
```