Proxomo iPhone SDK

**January**

12

**Proxomo**

Proxomo iPhone SDK

Fred Crable

Version 0.9

08

**Fall**

Table of Contents

API Authentication 3

Example: 3

User Authentication 3

Example 3

Proxomo Objects 4

Examples 4

Application Delegates 5

Proxomo List 6

Get All 6

Example 6

Application Data 7

Example 7

Location 8

Asynchronous Location Searches 8

Example 8

GeoCode 9

Example 9

Social Network Friend 9

Example 9

Notification 10

Friend 10

Event 10

Category 10

# API Authentication

Application wide operations such as querying for geo-location coordinates or application global data can be added using a Proxomo API identifier. Upon first use of the application context an application level authorization will be done.

### Example:

apiContext = [[ProxomoApi alloc]

initWithKey:@"ProxomoKey"

appID:@"AppID" delegate:self];

## User Authentication

The Person class implements the social network logins that authenticate a person. After a successful login to a social network the person identifier is used as the context for all User related operations.

### Example

userContext = [[Person alloc] init];

[\_userContext loginToSocialNetwork:FACEBOOK

forApplication:apiContext];

# Proxomo Objects

All Proxomo Definition classes are extended from the base class Proxomo Object. This class provides the basic CRUD (Create, Update, Delete) operations. The API identifier of a Proxomo API is the context for Application definitions such as Locations, Person, or Application Data.

Asynchronously adds the object using the API context

-(void) Add:(id)context;

Updates or creates a single instance from object asynchronously

-(void) Update:(id)context;

Gets an instance by ID and updates and overwrites current properties

-(void) Get:(id)context;

Deletes an instance by ID

-(void) Delete:(id)context;

This value is available on some object classes with embedded application data.

-(NSArray\*)appData;

### Examples

[location GetSynchronous:context];

[appData AddSynchronous:location];

[appData AddSynchronous:context];

appData.Value = @"NewValue";

[appData UpdateSynchronous:context];

As shown, AppData instances can use the identifier of another Proxomo Object in context. Both synchronous and asynchronous versions of these methods are provided. Synchronous versions return a Boolean status of true on success and asynchronous methods are managed using the ProxomoAppDelegate protocol.

## Proxomo Application Delegates

Application delegates can be used to receive a ProxomoAppDelegate protocol message on status of asynchronous REST operations with the Proxomo Server. The Proxomo Object delegate will use this selector to signal success or failure of a Proxomo Object’s REST transaction. On failure, the apiContext’s lastError property can be used along with the application logs to diagnose any application errors. Applications can use isAsyncPending to determine if there are additional asynchronous operations pending within an application context.

-(void)asyncObjectComplete:(BOOL)success

proxomoObject:(id)proxomoObject;

# Proxomo List

The ProxomoList class is a container and API delegate for list based operations. Searching for Locations, Application Data, and Friends are just some examples that use ProxomoList. The ProxomoList will aggregate a list of results and respond to to a single Application Delegate for the entire list. The arrayValue property of the ProxomoList is used to obtain the NSArray of Proxomo Objects processed by the Proxomo SDK.

## Get All

The ProxomoList has two methods used to obtain a collection of Proxomo AppData Objects. These bulk get operations discriminate the type of Proxomo Object using the enumeration type enumObjectType. These methods can be used to get all AppData; all App Data for Locations, Persons or Events; Locations for Persons; Social Network Friends for a Person, etc.

Gets all of the instances for given type uses the ProxomoList as the delegate calling object.

-(void) GetAll:(id)context

getType:(enumObjectType)type;

### Example ProxomoList

[list GetAll:apiContext getType:APPDATA\_TYPE];

[list GetAll:person getType:SOCIALNETFRIEND\_TYPE];

Combined context and object type products:

|  |  |  |
| --- | --- | --- |
| **Context** | **Get Type** | **Produces List** |
| API | APPDATA\_TYPE | Application’s AppData |
| Person | SOCIALNETFRIEND\_TYPE | Person’s SocialNetworkFriend |
| Person | APPDATA\_TYPE | Person’s AppData |
| Person | LOCATION\_TYPE | Person’s Location |
| Location | APPDATA\_TYPE | Location’s AppData |
| Event | APPDATA\_TYPE | Event’s AppData |
| Event | EVENTCOMMENT\_TYPE | Event’s Comments |

# Application Data

Search for application data about a Person, Location, or from the Application using the searchInContext method. Application Data can be added to Persons, Events, and Locations by substituting the Proxomo Object identifier when Adding, Updating, and Deleting an AppData instance. Search in context is a Class function on the AppData class.

+(void)searchInContext:(id)context

forObjectType:(NSString\*)objectType

intoList:(ProxomoList\*)proxomoList

useAsync:(BOOL)useAsync;

### Example

AppData \*appData = [[AppData alloc]

initWithValue:@”Value”

forKey:@"Key"];

ProxomoList \*list = [[ProxomoList alloc] init];

[AppData searchInContext:context

forObjectType:@"PROXOMO"

intoList:list useAsync:YES];

# Location

These three functions search and resolve an array of locations by the named strings.

-(NSArray\*)byAddress:(NSString\*)address

-(NSArray\*)byIP:(NSString\*)ip

-(NSArray \*)byLatitude:(NSNumber\*)latitude

byLogitude:(NSNumber\*)longitude

All three methods have the following properties to identify the API context and define synchronous or asynchronous operation.

apiContext:(id)context useAsync:(BOOL)useAsync

## Asynchronous Location Searches

The location class has a method for obtaining the locations resolved asynchronously. This method may be used after the application has successful searched for a location.

-(NSArray\*)locations;

### Example

Location.Name = @"JWB";

Location.City = @"Dallas";

Location.Address1 = @"100 main";

Location.LocationSecurity = PRIVATE\_LOCATION;

[location Add:apiContext];

NSArray \*pList = [location byAddress:kTestAddress

apiContext:apiContext

useAsync:NO];

# GeoCode

The GeoCode class has the following functions defined to search and identify locations using geographic coordinates. The GeoCode instance is updated as a result of locating the address or IP.

-(void) byAddress:(NSString\*)address

apiContext:(ProxomoApi\*)context

useAsync:(BOOL)useAsync;

-(void) byIPAddress:(NSString\*)ipAddress

apiContext:(ProxomoApi\*)context

useAsync:(BOOL)useAsync;

Synchronous lookup of location by latitude and longitude.

-(Location \*) byLatitude:(NSNumber\*)latitude

byLogitude:(NSNumber\*)longitude

apiContext:(ProxomoApi\*)context;

Asynchronous lookup using the location as the API delegate

-(void) byLatitude:(NSNumber\*)latitude

byLogitude:(NSNumber\*)longitude

locationDelegate:(Location\*)location

apiContext:(ProxomoApi\*)context;

### Example

GeoCode \*gcode = [[GeoCode alloc] init];

Location \*location = [[Location alloc] init];

[gcode byAddress:kTestAddress

apiContext:apiContext

useAsync:NO];

location = [gcode byLatitude:kLatitude

byLogitude:kLongitude

apiContext:apiContext];

# Social Network Friend

Social network friends may be retrieved using the ProxomoList GetAll method and the Person’s identifier as context.

### Example

ProxomoList \*pList = [[ProxomoList alloc] init];

[pList GetAll:objectContext

getType:SOCIALNETFRIEND\_TYPE];

SocialNetworkFriend \*firstFriend;

firstFriend = [[pList arrayValue] objectAtIndex:0];

# 

# Event

There are two event methods for searching for events within a distance from a geo-coordinate and by person.

-(NSArray\*) searchByDistance:(double)miles

fromLatitude:(double)latitude

fromLongitude:(double)longitude

startTime:(NSDate\*)start

endTime:(NSDate\*)end

apiContext:(id)context useAsync:(BOOL)useAsync;

-(NSArray\*) searchByPerson:(Person\*)person

startTime:(NSDate\*)start

endTime:(NSDate\*)end

apiContext:(id)context useAsync:(BOOL)useAsync;

# Event Comment

Event comments are added or updated using the Event identifier as context.

### Example

comment.PersonID = \_userContext.ID;

comment.PersonName = @"Person’s Name";

comment.Comment = @"Event Comment";

[comment AddSynchronous:event]

# Category

# Notification

# Friend