Life Mobile Application

APPLICATION PROGRAM INTERFACE: Research

Colin Man, Malina Jiang

**Version 1.0**

**Last Update:** 2 February 2015

Table of Contents

1 Document Control 3

1.1 Change Record 3

1.2 Definitions 3

2 Introduction 4

2.1 Scope 4

2.2 Functional Overview 4

2.2.1 Discovery 4

2.2.2 Maps 4

2.2.3 Lists 4

2.2.4 Safety 5

2.2.5 Scheduler 5

2.2.6 Entertainment 5

2.2.7 Tools 5

2.3 Assumptions/Constraints 5

2.4 Requirements 6

3 Component: Discovery 7

3.1 Navigation 7

3.2 Overview 7

3.3 Description 8

3.4 Feature: Goods and Services 9

3.5 Feature: Places 13

3.6 Feature: Travel 17

3.7 Feature: Food 19

3.8 Combination Features 22

3.9 Use Cases 23

Figure 3.9.1 – John 23

Figure 3.9.2 23

# Document Control

## Change Record

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Changes |
| 1.0 | 01-25-15 | Colin Man | Draft following overview + description |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Definitions

|  |  |
| --- | --- |
| Term | Definition |
| LMA | Life Mobile Application |
| LBS | Location Based Services |
| API | Application Program Interface |
|  |  |
|  |  |

Note: “LMA” is consistently used throughout all documents as the name of the mobile application, although it is understood that another name will be chosen for branding purposes.

# Introduction

## Scope and Purpose

This document contains a description of third party API’s that may potentially be incorporated into LMA, in terms of contribution to the business model as well as technical functionality and constraints.

The API’s outlined allow interaction with a variety of third-party products that do not define LMA individually, but come together to form a unique platform that encompasses the totality of the mobile user’s needs.

The existence of this document is imperative to the success of the product as it clearly defines the constraints of each interface and the extent to which we can leverage the product to create a novel platform. It is crucial to the development of a business plan with a clearly defined direction as well as a viable and efficient implementation. Without such definitions, it is easy to design a product that is impossible to develop on the technical side or assumes third-party functionality that does not exist.

A description of the API’s will also help in defining and outlining a minimal viable product, a subset of LMA’s features that form the core functionality and can stand alone if necessary.

As such, each third-party API description contains the following components:

* **Overview** – Summary of the features and purpose. This includes a description of the functionality encompassed in the API as well as its contribution to the system.
* **Structure and API Calls** – A technical description of the API and the structure of its interface. Since the way information is obtained varies widely depending on the design of each API, this is vital both in extracting constraints for the business model as well as development of the actual application in the implementation stage.
* **Technical Constraints –** Detailed summary of constraints in terms of information that can be obtained through the application user interface. Includes information such as maximum volume of requests that is critical to making important design decisions.

# API: Google Maps

## Summary

The Google Maps API is divided into four main types of interfaces, depending on the technology used to implement the product as well as to access the data.

The four API interfaces are:

* **Embed API** – The simplest of the four, which simply uses HTML and iframes to embed content into any website.
* **Web API** – Provides a set of functionality that can be accessed using standard RESTful calls. These calls can be placed from web applications (AJAX) or native applications.
* **JavaScript API –** Provides extended maps functionality in visualization and map content. Can be integrated into platforms that run mainly JavaScript.
* **iOS API** – Native integration of functionality that interfaces with the operating system directly. Can only be used by iPhone applications, but has similar functionality to the JavaScript API.

## Embed API

### Overview

The Embed API is a simple way to embed maps into an application. It allows developers to embed iframe elements into the web application and specify various features using parameters in the query portion of the src URL. Thus, information is obtained and displayed to the user through purely HTTP requests and no interaction with the API in between the requests is needed. Technical details are outlined in “Structure and API Calls”.

The API allows for four different ways to display maps content:

* **Place Mode** – displays a map pin at a predefined location (landmark, business, geographic feature, or custom pin)
* **Directions Mode** – displays the path between two different points and shows the time and distance estimated.
* **Search Mode** – displays a map that contains the results to a search query with a term specified in the attributes of the iframe tag.
* **View Mode** – Provides a barebones map that contains no visual content apart from the basic map functionality. This can be either a map view or a street view.

Note that the Embed API creates a map that links to Google Maps and provides an interface to work with options that may not be in the code injected through the iframe.

If the user is logged in to their Google account, they will also be able to save the locations from the embedded map into their user account, with custom attribution information that can b specified by the application.

### Structure and API Calls

The API call is performed through an HTTP request sent via the “src” attribute of an iframe. The call is structured as follows:

*https://www.google.com/maps/embed/v1/MODE?key=API\_KEY&parameters*

The mode of operation (MODE) is one of place, directions, search, view, or streetview

In order to use the Embed API to obtain maps, the application must first register with Google to obtain an API key (API\_KEY) that is included in the url query used to embed the map into the page. This key can be restricted to certain domains to ensure that others do not steal it.

Since the HTML to retrieve the maps is retrieved using a simple HTTP call to the web service, all parameters that have to do with obtaining data must be sent through the query string. Other properties such as width, height, border, etc. can be customized in the iframe and slightly change the rendering of the map.

The possible query parameters to the Embed API (parameters) are as follows:

* attribution\_source – string that describes the site or app; allows users to see what source is attributed to saving the place on their maps
* attribution\_web\_url – the url of the site or app (as attribution information)
* attribution\_ios\_deep\_link\_id – a URL scheme that links to the iOS application (using iOS internal links)
* center – the center of the map view (latitude and longitude values)
* zoom – (zoom level from 0 to 21)
* maptype – either “roadmap” or “satellite” depending on the view that should be loaded
* language – the language that should be used in the embedded map
* region – the borders and labels that should be displayed

The mode-specific query parameters are as follows:

* **Place**
  + q – the place to show on the map. This value can either be a location (escaped string) or an address (does not take latitude and longitude since it must be a name location to be highlighted).
* **Direction**
  + origin – the starting point location (as before, in either escaped address format or place name)
  + destination – the end point location (as before, in either escaped address format or place name)
  + waypoints – a set of intermediary places (as before, in either escaped address format or place name) separated by the pipe character
  + mode – method of travel (choose from driving, walking, bicycling, transit, or flying). Options will be shown on map if none is chosen.
  + avoid – any obstacles that the map should avoid (tolls, ferries, highways). Different items can be separated using the pipe character.
  + units – either metric or imperial depending on the units that should be displayed
* **Search**
  + q – the search term that should be shown on the map (in escaped URL format – can include restrictions such as “in Danville near 94526”)
* **Street View**
  + pano – the panorama id that should be shown (if not specified, one will be found from the location)
  + heading – the direction that the camera is facing in degrees from due north
  + pitch – the angle of the camera up or down in degrees
  + fov – the horizontal field of view in degrees

### Technical Constraints

Using the map modes in the Embed API is not restrictive in terms of the user interaction – the user can move the map around in the iframe and they can switch view modes (Satellite, Maps, etc.). The constraints are mainly on the ease of incorporating dynamic content and on showing or controlling the maps interaction.

It is not possible to dynamically load content into the map since the map is hosted on Google’s own servers and we are merely using an iframe to load it. Any attempt to change the “src” of the iframe in order to load the new parameters will cause the entire map to reload, which does not give a good user experience (we want the search to be done within the map interface, not by refreshing and reloading the entire map, which also has a higher latency).

This interface can only be used in a website since there are many limitations with iframes on mobile devices (Safari has compatibility issues, as does the android browser). The iframe served by Google Embed API is also not optimized for mobile, so may not be as easy to use on a phone.

There are no constraints on the volume of API calls that can be made, so the use of this API is essentially unlimited.

### Summary

When compatibility and dynamic flexibility is not an issue, using the Google Embed API is the best solution to provide map functionality. It is the easiest to use out of the four API’s and provides a very fast integration of maps functionality into a website.

In terms of LMA, the Embed API can be used for various maps services in which the location of a place should remain static for the duration of the webpage load. Examples of this may include: loading directions to a certain place, keeping track of saved routes and places, etc. For additional functionality of Google Maps, we may have to resort to the Web or native API’s. See below for a full description.

## Web API

### Overview

### Structure and API Calls

### Technical Constraints

### Summary

## JavaScript API

### Overview

### Structure and API Calls

### Technical Constraints

### Summary

## iOS API

### Overview

### Structure and API Calls

### Technical Constraints

### Summary