Glympse Client API Lite Guide

Contents

Change Log

Date	Version	Description
02/22/2013	0.1	Initial version.

Introduction

The Glympse API Lite provides a way of integrating Glympse functionality into 3rd party applications.

Please contact partners@glympse.com if you need assistance, find a problem, or would like to request a feature.

Supported Platforms

The Glympse API is capable of running under the following mobile platforms:

- iOS 4.0 and above.

Terminology

API / Client API / Library / Platform - class library encapsulating Glympse related functionality.
Client / Client application / Host application / Application - 3rd party application.
Device / Phone - any smartphone supported by Glympse API.
Server / Server API - Glympse back end server.
Glympse / Ticket - any Glympse on the system which was either sent or received by the user.

Servers and API keys

The Glympse API is capable of running into production and sandbox modes. Sandbox mode is intended to be used during development and testing. Production mode should be used when application is released to the market. Major configuration parameter for the mode is a server base URL. Server URL is specified when Glympse API is initialized.

Mode	**Server Base Url**
Production	`api.glympse.com`
Sandbox	`sandbox.glympse.com`

An API key is required to instantiate the Glympse API and get access to Glympse functionality. The API key is usually bound to a specific server (production, sandbox).

Here are some more notes on using the API keys during product development lifecycle.

- You should never use production configuration for development/testing purposes. Only released applications should talk to production servers.
- You should never release applications talking to sandbox server. Once application goes to market, it should talk to production infrastructure only.
- You should never switch servers dynamically. Once an application is installed on a device, it should talk to one and only one server during its the entire life. Server base url should be hardcoded. The Glympse API caches tokens and other information that is specific to the server it initially talked to.

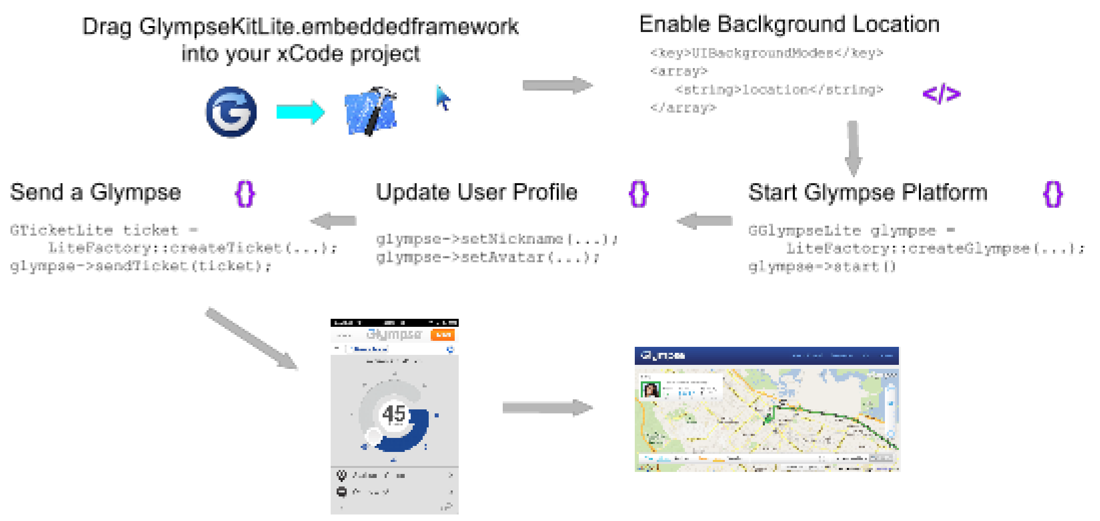
All applications running against sandbox environment are subject to rate limiting. The following limits apply:

- Maximum of 30 requests per 60 seconds are allowed per device;
- Maximum of 20 devices are allowed per API key.

It is required to provide application bundle ID (`CFBundleIdentifier`) during key creation process.

Integration

The following diagram illustrate the process of Glympse API Lite integration into 3rd party application:


Drag GlympseKitLite.embeddedframework into your xCode project
Enable Baclkground Location
</>
Start Glympse Platform
{}
Update User Profile
{}
Send a Glympse
{}
45

```
<key>UIBackgroundModes</key>
<array>
   <string>location</string>
</array>
```

```
GGlympseLite glympse =
    LiteFactory::createGlympse(...);
glympse->start()
```

```
glympse->setNickname(...);
glympse->setAvatar(...);
```

```
GTicketLite ticket =
    LiteFactory::createTicket(...);
glympse->sendTicket(ticket);
```

All phases mentioned above are required. Subsequent sections provide more information on each of these steps.

API Lifecycle

The following snippet of code instantiates Glympse API, points it to specific server and starts the library:

```
Glympse::GGlympseLite glympse =
    Glympse::LiteFactory::createGlympse(SERVER_BASE_URL, API_KEY);
glympse->start();
```
C++

It is required to stop Glympse, when Glympse functionality is no longer required:

```
glympse->stop();
glympse = NULL;
```
C++

It is also required to call `IGlympseLite::setActive(bool)`, when host application transitions between foreground and background.

Events Mechanism

Glympse API Lite provides single entry point for subscribing on platform events using `IGlympseLite::addListener()`. Subscribers have to implement `IListenerLite` interface to be notified on events coming from the platform. Glympse API provides convenience wrapper for subscribing Objective-C classes on events spread by the platform. Here is an example on how to leverage the technique:

```
@interface ViewController : UIViewController< GLYListenerLite > {
```
ObjC

```
    Glympse::GGlympseLite _glympse;
}
@end

@implementation ViewController
- (void)glympseEvent:(const Glympse::GGlympseLite&)glympse
                code:(int)code
              param1:(const Glympse::GCommon&)param1
              param2:(const Glympse::GCommon&)param2
{
    // See Glympse::LC::EVENT_* for the list of all supported events.
}
- (void)subscribe
{
    [GLYGlympseLite subscribe:self onPlatform:_glympse];
}
- (void)unsubscribe
{
    [GLYGlympseLite unsubscribe:self onPlatform:_glympse];
}
@end
```

User Profile

It is possible to configure nickname and avatar. These profile details are exposed to viewers of user's tickets.

```
glympse->setNickname(Glympse::LiteFactory::createString(“Sylvia”));
glympse->setAvatar(Glympse::LiteFactory::createString(
    “http://glympse.com/images/avatars/sylvia.jpg”), 0);
```

C++

NOTE Glympse API persists nickname and avatar locally (in addition to associating it with user account on the server). It only makes sense to modify any of these properties, when the change is initiated by user.

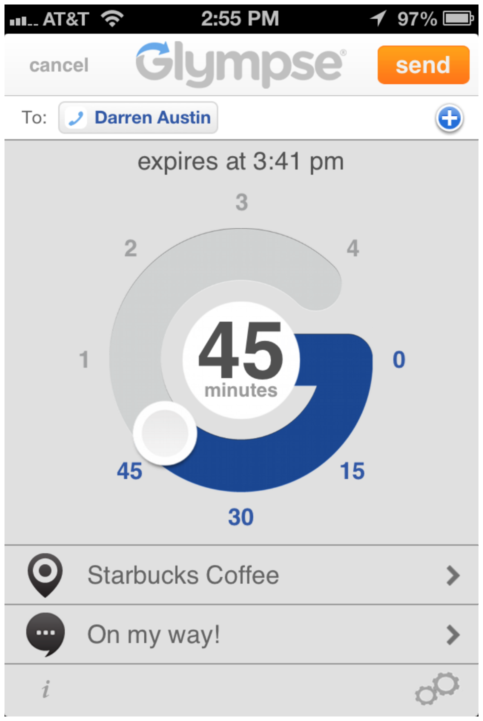
Sending a Ticket

Glympse API Lite is designed to provide host applications with an ability to send a ticket with minimal amount of efforts. The following code snippet creates a ticket with specified duration message and empty destination.

```
Glympse::GTicketLite ticket = Glympse::LiteFactory::createTicket(3600000,
    Glympse::LiteFactory::createString(”On my way”), NULL);
glympse->sendTicket(ticket, 0);
```

C++

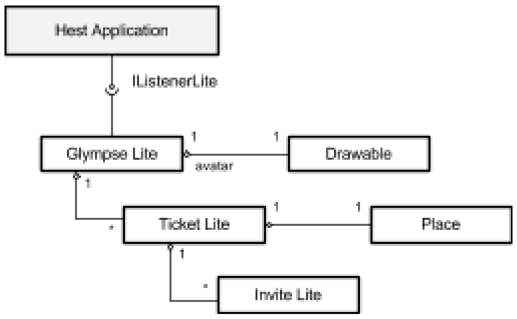
This will present *Send a Glympse* wizard to the user. The wizard walks user through all configurable properties and allows to send a ticket in the end. Wizard appearance is configurable through the set of flags passed to `sendTicket()` method. See `LC::SEND_WIZARD_*` for the list of all available options.



Once ticket is populated and sent, it will become accessible through `IGlympseLite::getTickets()`.

History and Viewing Status

`IGlympseLite` provides access to the list of sent tickets. The list is sorted based on expiration time, which means that active tickets always go first. Each ticket on the list (`ITicketLite`) provides access to some ticket properties and basic modification tools. It also maintains an array of (`IInviteLite`) objects. Information about who and when is viewing a ticket is accessible through `IInviteLite` properties.



Glympse API provides `IListenerLite` interface that host application needs to implement to be notified on all changes happening with outgoing tickets and the library in general.

Glympse account can be shared across multiple Glympse-enabled applications. Some applications have access to all tickets sent by the user (belonging to the same account). Ticket ownership can be transferred between applications. Glympse Lite-enabled applications do not see sibling tickets (belonging to the same user account but owned by another application). In case if ticket is originated by Glympse Lite-enabled application and then transferred away, it disappears from local history. `Glympse::LC::EVENT_TICKET_REMOVED` is sent.