Glympse Scheme Tutorial

Contents

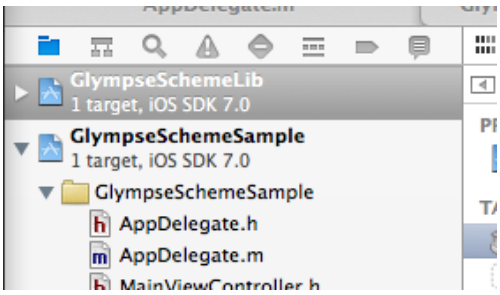
Please contact partners@glympse.com if you need assistance, find a problem, or would like to request a feature.

The `GlympseSchemeLib` project is provided as a convenience library when using iOS URL Schemes to interact with the Glympse application. Successfully using this library requires an understanding of Apple's iOS Custom URL Scheme system.

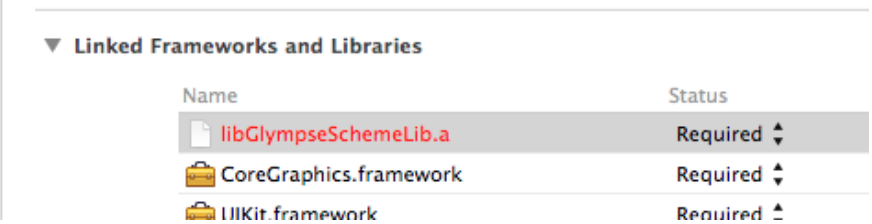
Adding Glympse Scheme support to a New or Existing Project

The following assumes you are using the latest iOS SDK with Xcode. There are two recommended options:

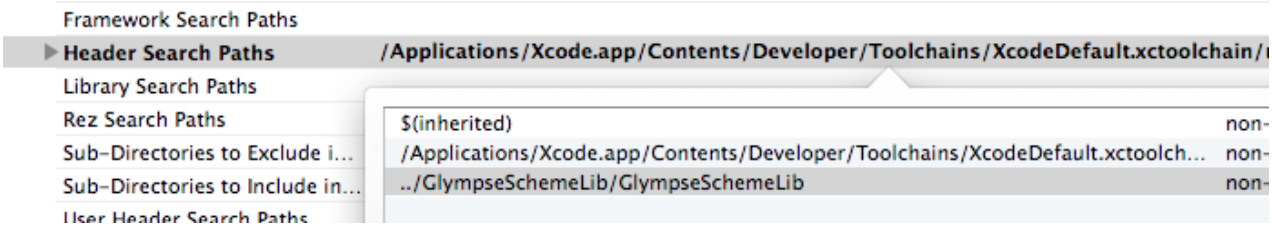
1. Either create a new project or open your existing Xcode project.
2. Copy the top-level GlympseSchemeLib folder to your preferred location for external code sources. (The GlympseSchemeLib can be found under the "iOS" folder in the Glympse Application SDK root folder.)
3. Include GlympseSchemeLib.xcodeproj as an external dependency by dragging the file from its new location directly into Xcode alongside your project in the workspace, similar to the screenshot on the right:
    a. Note: You may also choose to nest it inside of your project, but complete instructions for that approach are not in this tutorial.



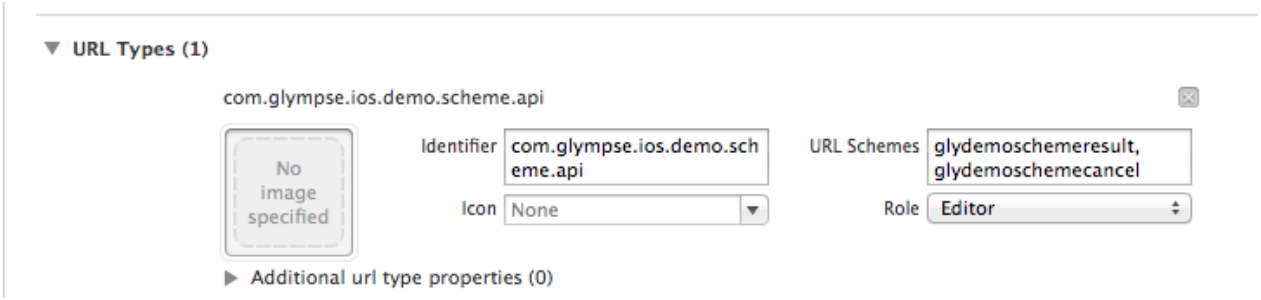
4. Under your project's Target's "General" settings, add libGlympseSchemeLib.a as a required Linked Library:



5. Set up appropriate "Header Search Paths" in your project's Target's "Build Settings" to help it locate GlympseSchemeLib header files. The correct search path for your environment will depend on where you placed the GlympseSchemeLib in Step 2.



6. In order for the Glympse app to call your app back after the Glympse is sent or cancelled, you need to register two return URL schemes for your app. This can be accomplished in your project's Target's "Info" settings, under the "URL Types" subheading. The picture below is an example of this, however, you should create a new, unique URL identifier and URL schemes for your app.



Tutorial: Sending a Glympse using GlympseSchemeLib

1. Import the required GlympseSchemeLib headers:

```objc
#import "GLYGlympseApp.h"
#import "GLYRecipient.h"
```

2. Create a CreateGlympseParams object and populate it with pre-configured data as desired. Many fields are optional, but a typical scenario where the invite is delivered by the app invoking the scheme is as follows:

```objc
// Allocate a CreateGlympseParams object.
GLYCreateGlympseParams *glympseCreateParams = [[GLYCreateGlympseParams alloc]
init];

// Since we are just creating a "link" recipient, we hide the recipient chooser.
glympseCreateParams.flags = GLYFlagRecipientsHidden;

glympseCreateParams.returnUrl = @"your_app_return_result_scheme://";
glympseCreateParams.returnCancelUrl = @"your_app_return_cancel_scheme://";

NSString *subtype = @"your_app_name";
NSString *brand = @"your_app_brandname";

/ Specify that we want a single "app" recipient.
[glympseCreateParams.recipients addObject:[[GLYRecipient alloc]
    initWithType:GLYRecipientTypeApp subtype:subtype brand:brand name:NULL
```

```
address:NULL]];


// Invoke the Glympse-app to create this Glympse, if possible, with these
parameters
if ( ![GLYGlympseApp createGlympse:glympseCreateParams] )
{
    // This error case should occur when the installed Glympse app version is too
old.
}
```

3. When the invite is created, or cancelled by the user, your app will receive a callback through the iOS URL Scheme handling system. The code below should be added to your UIApplicationDelegate implementation:

```
- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url
  sourceApplication:(NSString *)sourceApplication annotation:(id)annotation
{
    if ( [[url scheme] isEqualToString:@"your_app_return_result_scheme"] )
    {
        GLYCreateGlympseResult *reply = [[GLYCreateGlympseResult alloc]
            initWithUriString:url.absoluteString];
        // Handle successful reply here.
        return YES;
    }
    else if ( [[url scheme] isEqualToString:@"your_app_return_cancel_scheme"] )
    {
        // Handle cancel here.
        return YES;
    }
    return NO;
}
```

ObjC

Tutorial: Viewing a Glympse using GlympseSchemeLib

1. Import the required GlympseSchemeLib headers:

```
#import "GLYGlympseApp.h"
#import "GLYRecipient.h"
```

ObjC

2. `GlympseSchemeLib` has convenience methods for parsing Glympse invite codes and group names out of NSStrings. To launch an activity to view a Glympse, populate a `ViewGlympseParams` object with the parsed results and call GlympseApp.viewGlympse().

```objc
// Grab the received text containing a Glympse invite URL. An example would be:
NSString *buffer = @"Sample text message that contains one or more Glympse URLs,
such as http://glympse.com/BOT-002";

// Parse the string and check to see if it has any Glympse URLs that
// contain glympse codes or group names.
GLYUriParser *parseBufferResult = [[GLYUriParser alloc] initWithBuffer:buffer];
if ([parseBufferResult hasGlympseOrGroup])
{
    // Allocate a ViewGlympseParams object and tell it what we want to view.
    GLYViewGlympseParams *glympseViewParams = [[GLYViewGlympseParams alloc] init];
    [glympseViewParams addAllGlympsesAndGroups:parseBufferResult];


    // Generate a "view a glympse" URL and start the activity for it.
    if ( ![GLYGlympseApp viewGlympse:YES params:glympseViewParams] )
    {
        // This error case should occur when the installed Glympse app version is
too old.
    }
}
```

More Information

See the **Glympse_URI_Schema** document for more details about using Glympse schema handlers.

If your question cannot be answered by these documents, please contact partners@glympse.com for assistance.