C2call

SDK Quick Start Guide

Index

Requirements

C2Call SDK
-SocialCommunication.ressources
-SocialCommunication.framework

Required Frameworks

- AVFoundation.framework
- Accounts.framework
- AdSupport.framework
- AddressBook.framework
- AddressBookUI.framework
- AssetsLibrary.framework
- AudioToolbox.framework
- CFNetwork.framework
- CoreAudio.framework
- CoreData.framework
- CoreFoundation.framework
- CoreLocation.framework
- CoreMedia.framework
- CoreTelephony.framework
- CoreText.framework
- CoreVideo.framework
- MapKit.framework
- MediaPlayer.framework
- MessageUI.framework
- MobileCoreServices.framework
- OpenGLES.framework
- QuartzCore.framework
- QuickLook.framework
- Security.framework
- StoreKit.framework
- SystemConfiguration.framework
- iAd.framework
- libsqlite3.dylib
- libz.dylib

Project Target Build Settings

HEADER_SEARCH_PATHS = /usr/include/libxml2
OTHER_LDFLAGS = -lxml2 -lstdc++
ARCHS = armv7
VALID_ARCHS = armv7

AppDelegate sub-classing

Application Delegate base class for an automatic initialization of C2Call services.

As a VoIP and Messaging Service the C2Call Framework requires a complex initialization procedure and as well a detailed monitoring of all application states like active (app is in foreground), inactive (screensaver is active), background (app is in background) or termined. During the different states, the C2Call Framework has to maintain the connection to the C2Call backend systems, monitor network changes, automatically retrieve missed messages or call records and nevertheless save battery power as good as possible. In order to make it easy for the developer, all this complex application management has been capsuled into the C2CallAppDelegate base class. This base class should now be the UIApplicationDelegate base class for apps using the C2Call Framework.

The C2CallAppDelegate class implements the following methods from UIApplicationDelegate:

```
- (BOOL)application:(UIApplication *)application
      didFinishLaunchingWithOptions:(NSDictionary *)launchOptions

- (void)applicationDidBecomeActive:(UIApplication *)application;
- (void)applicationWillResignActive:(UIApplication *)application;

- (BOOL)application:(UIApplication *)application
      handleOpenURL:(NSURL *)url;
- (BOOL)application:(UIApplication *)application
      openURL:(NSURL *)url
      sourceApplication:(NSString *)sourceApplication
      annotation:(id)annotation

- (void)applicationWillTerminate:(UIApplication *)application;
- (void)application:(UIApplication *)application
      didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken
- (void)application:(UIApplication *)application
      didFailToRegisterForRemoteNotificationsWithError:(NSError *)error
- (void)application:(UIApplication *)application
      didReceiveRemoteNotification:(NSDictionary *)userInfo
- (void)application:(UIApplication *)application
      didReceiveLocalNotification:(UILocalNotification*)notification
- (void)applicationDidEnterBackground:(UIApplication *)application
- (void)applicationWillEnterForeground:(UIApplication *)application
```

On all methods above the corresponding super method has to be called when overwriting in your own C2CallAppDelegate subclass.

Example:

```
// MyAppDelegate.h
#import <SocialCommunication/SocialCommunication.h>
@interface MyAppDelegate : C2CallAppDelegate

// MyAppDelegate.m
- (BOOL)application:(UIApplication *)application
     didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
     self.affiliateid = @"<My AffiliateId>";
     self.secret = @"<MyApplication Secret>";

#ifdef __DEBUG
     self.useSandboxMode = YES;
#endif

     return [super application:application
     didFinishLaunchingWithOptions:launchOptions];
}

- (void)applicationWillResignActive:(UIApplication *)application
{
    [super applicationWillResignActive:application];
}

- (void)applicationDidEnterBackground:(UIApplication *)application
{
    [super applicationDidEnterBackground:application];
}

- (void)applicationWillEnterForeground:(UIApplication *)application
{
    [super applicationWillEnterForeground:application];
}

- (void)applicationDidBecomeActive:(UIApplication *)application
{
    [super applicationDidBecomeActive:application];
}

- (void)applicationWillTerminate:(UIApplication *)application
{
    [super applicationWillTerminate:application];
}
```

Using the Low Level API

C2CallPhone – Base class for low-level C2Call phone and messaging API.

This class provides the low-level API methods for major C2Call communications features:

```
- Phone calls to PSTN and mobile phone networks,
- VoIP and video calls,
- Group calls (VoIP and Video)
- Instant Messaging,
- SMS / Text Messages
- Rich Media Messages like photo, video, voicemail, location, etc.
```

This class also provides the core functionality to start and stop the C2CallPhone and handle the background behavior. The C2CallAppDelegate uses this Class in order to initialize the connection to the C2Call service and to handle the background status of the app.

Example call functionality:

```
// Call the C2Call TestCall
[[C2CallPhone currentPhone] callVoIP:@"9bc2858f1194dc1c107"];
```

Example messaging functionality:

```
// Send an SMS / Text message
[[C2CallPhone currentPhone] submitMessage:@"Hi, this is an SMS"
toNumber:@"+1408234123456"];
```

Example Rich Messaging Functionality:

```
#import <SocialCommunication/UIViewController+SCCustomViewController.h>
...skip...
[self captureImageFromCameraWithQuality:UIImagePickerControllerQualityTypeMedium
      andCompleteAction:^(NSString *key) {
            [[C2CallPhone currentPhone]
                  submitRichMessage:key message:@"Hi, see my photo..."
                  toTarget:@"max@gmail.com"];
}];
```

Using C2Call SDK Feature in your UIViewController sub-class

The C2Call SDK comes with an UIViewController category, which provides access to the C2Call SDK Feature with only a single line of code:

This UIViewController category provides convenience methods for instantiation and launch of C2Call SDK GUI component.

The C2Call SDK provides feature rich components for rich media content handling and social media communication features. The basic concept of this SDK is, to combine convenience and ease of use with great flexibility for all provided components. The UIViewController+SCCustomViewController category provides easy access to those components, whether they will be used as standard components from SCStoryboard or as customized components from the MainStoryboard of the developed application.

In iOS Storyboard an UIViewController represents one screen of content and the transitions between two UIViewControllers will be defined by an UIStoryboardSegue. In C2Call SDK the UIViewController based GUI components will be also presented via UIStoryboardSegue, so that the developer can define the transition in his MainStoryboard. Several components require one or more parameters, which typically have to be set in method prepareForSegue:sender. The UIViewController+SCCustomViewController provides convenience methods to present C2Call SDK GUI components with a one or two lines of code and ensures that required parameter will be correctly set in the presented ViewController.

The following GUI components are supported:

```
Presenting Rich Media Content
- SCComposeMessageController (Composes and submit a new Instant Message,
  Rich Media Message or SMS/Text Message to a user or phone number)
- SCPhotoViewerController (Presents photos)
- SCVideoPlayerController (Presents a Video Player)
- SCLocationViewerController (Presents a location in a map view)
- SCAudioPlayerController (Presents an Audio Player for VoiceMails)

Capture Rich Media Content:
- UIImagePickerController (Captures an Image or Video from Camera or Photo Album
   with Quality)
- SCAudioRecorderController (Records and submits a VoiceMail)
- SCLocationSubmitController (Picks and submits the current location or nearby
   Google places)

Show Friends, Groups and Chat History:
- SCChatController (Presents a Chat History with a specific contact)
- SCFriendDetailController (Presents the details of a connected friend)
- SCGroupDetailController (Presents the details of a connected group)

Others:
- SCBrowserViewController (Opens an URL in a Browser View)
```

In case the developer connects the above components in his MainStoryboard via UIStoryboardSegue the segues have to be named with a specific name convention in order to handle the parameters setup in prepareForSegue:sender correctly. Each connected UIStoryboardSegue has to be named like the target ViewControllers original class name, for example:

```
Target UIViewController Class           Segue Name
SCComposeMessageController      ->      SCComposeMessageControllerSegue
SCPhotoViewerController         ->      SCPhotoViewerControllerSegue
SCVideoPlayerController         ->      SCVideoPlayerControllerSegue
SCLocationViewerController      ->      SCLocationViewerControllerSegue
etc.
```

In addition to this naming convention, the developer needs to call customPrepareForSegue:sender in his prepareForSegue:sender method:

```
-(void) prepareForSegue:(UIStoryboardSegue*) segue sender:(id) sender
{
    [self customPrepareForSegue:segue sendersender];

    // Do your own setup
}
```

All ViewControllers covered by this UIViewController category can be also presented without UIStoryboardSegue connections. In this case the component will be instantiated from MainStoryboard and if not found there, from SCStoryboard. It'll push the component if the current ViewController is embedded in a UINavigationController and present modal else. We recommend to use UIStoryboardSegue if the developers wants to have full control on this behavior.

Access to C2Call CoreData Objects using SCDataManager

SCDataManager provides the low-level API access to the CoreData Database handled by C2Call SDK.

The Database Schema is available under:
SocialCommunication.ressources/C2CallDataModel.xcdatamodeld

All relevant Entities in this Data Model are subclasses from NSManagedObject for convenient access to the Entity Attributes. The following NSManagedObject subclasses are defined by C2Call SDK.

```
- MOChatHistory
- MOC2CallUser
- MOCallHistory
- MOC2CallEvent
- MOOpenId
- MOAddress
- MOPhoneNumber
- MOGroupMember
- MOC2CallGroup
```

C2Call SDK GUI Components are based on CoreData. All access to CoreData Objects and all queries to the database have to be done in main Thread. SCDataManager capsules the access to the CoreData database layer and ensures that all request are done in main Thread.

Accessing the User Profile using SCUserProfile class

SCUserProfile class provides access to the current users profile with the following properties

```
 displayname property
  userid property
  email property
  didnumber property
  callerid property
  isCalleridVerified property
  credit property
  firstname property
  lastname property
  country property
  phoneWork property
  phoneHome property
  phoneMobile property
  phoneOther property
– saveUserProfile
+ currentUser
```

Using the C2Call SDK GUI Components

All C2Call SDK components are provided in the SCStoryboard.storyboard file as part of the SocialCommunication.ressources.
The C2Call SDK wants to provide read made components which are easy to use, feature complete but also highly customizable for the developer.
Customization of a GUI component can be achieved by implementing the following steps:

1. Copy the required GUI component + all connected components from SCStoryboard into your Main Storyboard.
2. Change the design of the GUI component as required, remove unwanted buttons, images or labels and re-arrange the remaining according to your requirements.
3. Additional Functionality can be added by sub-classing the GUI component ViewController class. Add the specific new functions or methods, change the class name of the GUI component to your sub-class and add all required connections.

Several components will be presented by the C2Call sub-system on event using instantiateViewControllerWithIdentifier:

In order to ensure that always the customized version of a specific GUI component will be presented, the C2Call sub-system will first seek for the name of the GUI Component in the Main Storyboard and if not available there, it will be instantiated from SCStoryboard.
As those components are found by name, it's important to keep the Storyboard name, even if the component will be sub-classed. A typical example of an event trigger component is SCCallStatusController and SCInboundCallController.

Transition from one ViewController to another ViewController in iOS Storyboard will be typically done via UIStoryboardSegue. In SCStoryboard, all ViewController which are providing transitions to other ViewController are connected via UIStoryboardSegues. Therefore it's important when copying a component from SCStoryboard to the Main Storyboard, that all connected components will be copied as well.
Several of the connected components are taking parameters, which are automatically transmitted via prepareForSegue:sender: method. The correct segue here will be identified also by its name. In order to make it easy to name the UIStoryboardSegue correctly, we have created a name convention for all segues we are using in C2Call SDK. The easy rule is, that the segue name has to be the name of the target ViewController class (always the original C2Call SDK ViewController class name) plus the postfix "Segue".
So, for example, the SCFriendListController will transition to an SCFriendDetailController on tap on a friend item. Therefore the SCFriendListController will call the method [self performSegueWithIdentifier:@"SCFriendDetailControllerSegue"].
In method [self prepareForSegue:(UIStoryboardSegue*) segue sender:(id) sender] the userid of the friend will be set in the SCFriendDetailController, before it will be presented.

So, if after customization, something is not working as expected. Please carefully check Segue Names, Class-Names of Sub-classed ViewController and Storyboard Ids of the components.

Code Snippets

Here are some useful code snippets for typical functions in C2Call SDK

Initiate a call to a number phone number

```
[[C2CallPhone currentPhone] callNumber:@"+1408234123456"];
```

Initiate a call to a Userid

```
[[C2CallPhone currentPhone] callVoIP:@"9bc2858f1194dc1c107"];
```

Initate a video call to an EMail Address

```
[[C2CallPhone currentPhone] callVideo:@"max.muster@gmail.com"];
```

Sending an SMs/Text Message

```
[[C2CallPhone currentPhone] submitMessage:@"Hi, this is an SMS"
      toNumber:@"+1408234123456"];
```

Sending a photo taken from camera

```
#import <SocialCommunication/UIViewController+SCCustomViewController.h>
...skip...
[self captureImageFromCameraWithQuality:UIImagePickerControllerQualityTypeMedium
      andCompleteAction:^(NSString *key) {
            [[C2CallPhone currentPhone]
                  submitRichMessage:key message:@"Hi, see my photo..."
                  toTarget:@"max@gmail.com"];
}];
```

Receiving a message

```
// Overwrite your AppDelegate Method
-(void) message:(DDXMLElement *)msg
{
    NSString *text = [[msg elementForName:@"Description"] stringValue];
    NSString *userid = [[msg elementForName:@"Contact"] stringValue];
    NSString *remoteParty = [[C2CallPhone currentPhone] nameForUserid:userid];

     // Handle the message here
}

// Identify the message type (text, video, photo, etc

    SCRichMediaType mt = [[C2CallPhone currentPhone] mediaTypeForKey:message];
    if (mt == SCMEDIATYPE_TEXT) {
      // Handle the text message here
    }

    if (mt == SCMEDIATYPE_IMAGE) {
      // Handle the photo here
      // --> See Download an attachment
    }

// Download an attachment
        // Check if attachment is already downloaded
        if (![[C2CallPhone currentPhone] hasObjectForKey:message])  {

            // Start download
            [[C2CallPhone currentPhone] retrieveObjectForKey:message
                  completion:^(BOOL finished) {

                        // Do this on Main-Thread
                        dispatch_async(dispatch_get_main_queue(), ^{
                              UIImage *thumb = [[C2CallPhone currentPhone]
                              thumbnailForKey:message];

                        if (thumb) {
                              // Present the thumbnail
                        }

                        UIImage *photo = [[C2CallPhone currentPhone]
                              imageForKey:message];
                        if (photo) {
                              // Present the photo here
                        }
                });
            }];
        }
```

Customizing Chat Bubbles

```
[[SCBubbleViewOut appearance] setBaseColor:[UIColor
  colorWithRed:70./255.
  green:188./255.
  blue:255./255.
  alpha:0.69]];
[[SCBubbleViewIn appearance] setBaseColor:[UIColor
  colorWithRed:115./255.
  green:223./255.
  blue:81./255.
  alpha:0.69]];
[[SCBubbleViewIn appearance] setBubbleTypeIn:SC_BUBBLE_IN_WHAZZUPP];
[[SCBubbleViewOut appearance] setBubbleTypeOut:SC_BUBBLE_OUT_WHAZZUPP];
```

Please visit our forum for support or any feedback.