Contents

AmeliaKit – version 3.1.2

Getting started

This section describes two options for installing AmeliaKit into your Xocde project.

Installation using a Dynamic Framework

1. Go to your Xcode project's "General" settings. Drag `AmeliaKit.framework` to the "Embedded Binaries" section. Make sure *Copy items if needed* is selected and click *Finish*.
2. Create a new "Run Script Phase" in your app's target's "Build Phases" and paste the following snippet in the script text field:

```
bash "${BUILT_PRODUCTS_DIR}/${FRAMEWORKS_FOLDER_PATH}/AmeliaKit.framework/strip-frameworks.sh"
```

This is a required step to work around an App Store submission bug when archiving universal binaries.

Quick start

This section describes how to integrate AmeliaKit into a new project. This will create a simple application that performs an anonymous login, selects a domain, presents a simple chat UI. The point is to illustrate the basics concepts.

1. Create a new Xcode Single View Application project.
2. Follow the steps outlined in Installation using a Dynamic Framework.
3. Embed the default view controller in a navigation controller.
4. Add `#import <AmeliaKit/AmeliaKit.h>` to `ViewController.m`.
5. Conform to the `IPSAKChatSessionDelegate` and add a property for the chat instance:

```
@interface ViewController () <IPSAKChatSessionDelegate>
@property (strong) IPSAKChat *chat;
@end
```

6. Create a configuration object and a chat instance. For the purpose of this guide we will connect to an Amelia instance that allows anonymous login and we will use the manual domain selection method. Add the following code to `viewDidLoad`:

```
self.title = @"Amelia";

IPSAKConfiguration *configuration = [IPSAKConfiguration configurationWithURL:
                                             [NSURL URLWithString:@"<AMELIA-URL>"]];
configuration.domainSelectionMode = IPSAKDomainSelectionModeManual;

self.chat = [IPSAKChat chatWithConfiguration:configuration];
self.chat.sessionDelegate = self;
 [self.chat initialize];
[self.chat startNewConversation];
```

7. Add the necessary `IPSAKChatSessionDelegate` methods to support this sample application. The error methods are not necessary for this sample application but are provided to make sure we are not stopping on an error.

```
- (void)chat:(IPSAKChat *)chat domainSelectionRequired:(NSArray<IPSAKDomain *> *)domains {
    NSLog(@"Selecting domain: %@", domains.firstObject.name);
    [chat selectDomain:domains.firstObject];
}

- (void)chatConversationStart:(IPSAKChat *)chat {
    NSLog(@"Present a chat UI.");
}

- (void)chat:(IPSAKChat *)chat sessionFailWithError:(NSError *)error {
    NSLog(@"%s: %@", __PRETTY_FUNCTION__, error.localizedDescription);
}

- (void)chat:(IPSAKChat *)chat domainFailWithError:(NSError *)error {
    NSLog(@"%s: %@", __PRETTY_FUNCTION__, error.localizedDescription);
}

- (void)chat:(IPSAKChat *)chat conversationFailWithError:(NSError *)error {
    NSLog(@"%s: %@", __PRETTY_FUNCTION__, error.localizedDescription);
}
```

8. Running this application should first print Selecting domain: "Domain name" and then "Present a chat UI." to the console.

9. Add a new view controller file called ConversationViewController to the project. Add a new view controller to the storyboard and set its class to "ConversationViewController" and the storyboard ID to "ConversationViewController". Add a `UITextView`, a `UITextField`, and a `UIButton` to the new view controller. It is not important how these controls are placed, it is only for demonstrating the conversation aspect.

10. Add `#import <AmeliaKit/AmeliaKit.h>`, conform to `<IPSAKChatConversationDelegate>`, and connect the controls of the ConversationViewController as outlets:

```
@interface ConversationViewController () <IPSAKChatConversationDelegate>
@property (weak, nonatomic) IBOutlet UITextField *textField;
@property (weak, nonatomic) IBOutlet UITextView *textView;
@property (weak, nonatomic) IBOutlet UIButton *sendButton;
@end
```

11. Add a property for the chat instance (in `ConversationViewController.h`):

```
@class IPSAKChat;

@interface ConversationViewController : UIViewController
@property (weak) IPSAKChat *chat;
@end
```

12. Set the conversation delegate in `viewDidLoad`:

```
- (void)viewDidLoad {
    [super viewDidLoad];
    self.title = self.chat.selectedDomain.name;
    self.chat.conversationDelegate = self;
    self.textView.text = @""; // Make sure the text view is cleared when we start.
}
```

13. Add the necessary `IPSAKChatConversationDelegate` methods:

```
- (void)chat:(IPSAKChat *)chat didReceiveMessage:(IPSAKMessage *)message {
    // This is the quick way to get something showing. In a real application it is important
    //to check the messageType of the message.
    if (message.messageText != nil) {
        NSString *messageText = [NSString stringWithFormat:@"%@: %@\n",
                                          message.fromUserDisplayName, message.messageText];
        [self.textView insertText:messageText];
    }
}

- (void)chat:(IPSAKChat *)chat inputEnabled:(BOOL)enabled {
    // It is now allowed to send new questions to Amelia until she is ready.
    self.sendButton.enabled = enabled;
}

- (void)chat:(IPSAKChat *)chat error:(NSError *)error {
    NSLog(@"%s: %@", __PRETTY_FUNCTION__, error.localizedDescription);
}
```

14. Add an action for the sendButton:

```
- (IBAction)sendButtonTap:(id)sender {
    [self.chat ask:self.textField.text];
    self.textField.text = @"";
}
```

15. Present the simple conversation UI. Remember to add `#import "ConversationViewController.h"` to `ViewController.m`. Update `chatConversationStart:` to:

```
- (void)chatConversationStart:(IPSAKChat *)chat {
    ConversationViewController *conversationViewController = [self.storyboard
    conversationViewController.chat = self.chat;
    [self.navigationController pushViewController:conversationViewController animated:YES];
}
```

16. Running the application should now log on anonymously to the Amelia instance provided by the url, select the first available domain, and present a simple chat UI.

Reference

AppConfig

Initialize Amelia chat with

```
[self.chat initialize]
```

will allow the sdk to return the appConfig asynchronous, when the below IPSAKChatSessionDelegate is implemented:

`-(void)chatInitialized:(IPSAKChat*)chat withAppConfig:(IPSAKAppConfig *)appConfig;`

AuthSystem

AuthSystem indicates the type of authentication method the system supports. It could either be internal or external. Usually internal login uses LDAP and external login uses SAML. The AuthSystem is returned after calling `[self.chat startNewConversation]` when login is required and the IPSAKChatConversationDelegate method is implemented:

```
- (void)chat:(IPSAKChat *)chat loginRequiredWithAuthSystems:(NSArray<IPSAKAuthSystem *> *)authSystems;
```

Each auth system contains an auth code. One could also check if `[authSystem redirect]` is true to see if it is an SSO auth system. A SSO auth system requires a third party to authorize the user, and the user needs to be redirected to the site indicated by the url returned by [authSystem loginPath]. To know more details about SSO login, please see the section below. As an example, let's say there are two auth systems in the NSArray, one is internal, and code is i_LDAP. Other one is external, and code is e_saml. In this case, we can give the user an option to login with the username/password, and another option to login with SAML. On the other hand, if for some reason the Amelia backend had the SSO auth system removed in the configuration, and only i_LDAP is returned, then we should hide the SSO login button and not let the user to use single sign on.

Login methods

Anonymous

To login anonymously, simply call:

```
[self.chat startNewConversation];
```

On Conversation start success, chatConversationStart will be called back when domain selection mode is automatic and there is only one domain. Otherwise a list of domains will be returned with the callback method "domainSelectionRequired". If anonymous user is not allowed on the Amelia server side, then the IPSAKChatConversationDelegate's loginRequiredWithAuthSystems will be called (see internal login section below)with a list of auth systems supported on the server side.

Internal login

Stepup call will trigger the internal login.

```
[self.chat stepUp];
```

After stepup is called, below method is called back from the sdk.

```
- (void)chat:(IPSAKChat *)chat loginRequiredWithAuthSystems:(NSArray<IPSAKAuthSystem *> *)authSystems;
```

In the implementation of this delegate method, you should make another call to login the user with username/password:

```
IPSAKLoginOptions *options = [[IPSAKLoginOptions alloc] initWithUsername:username password:password];
[self.chat login:options];
```

SAML (A type of single sign on)

Once the auth systems are returned in the IPSAKChatConversationDelegate's loginRequiredWithAuthSystems method, and there is an external auth system, we can allow to user to use single sign on. Generally speaking, to use SAML as single sign on, we need to retrieve the cookie from third party, then pass that cookie to the IPSAKLoginOptions when logging in. To login with SSO, load the third party login page with SFSafariWebController Once login is successful, the backend server should redirect to a URL which the IOS app can decipher. Provided the Amelia backend is setup to do so, and URL scheme is set up in xcode under Targets-> Your-app -> info -> Url types. The url will be in the form of urlScheme://login?cookie=[amelia-session-cookie] Once this url is captured by the app, extract the cookie and pass it into login options:

```
IPSAKLoginOptions *options = [[IPSAKLoginOptions alloc] initWithAuthSystem:authSystem
                                                              sessionCookie:cookie];
[self.chat login:options];
```

Domain

Domains are returned automatically when session is established successfully. The list of domains will be returned in the IPSAKChatConversationDelegate's method:

```
- (void)chat:(IPSAKChat *)chat domainSelectionRequired:(NSArray<IPSAKDomain *> *)domains {
    [chat selectDomain:domains.firstObject];
}
```

After domain is selected, the conversation is will start.

Text to speech

By default, Amelia messages will be automatically played after they have been received. To completely mute it, you could pass into configuration the SpeechParams:

```
IPSAKConfiguration *configuration = [IPSAKConfiguration configurationWithURL:serverURL];
configuration.speechParams.enabled = NO;
```

Speech to text

Speech to text function is not provided by the SDK. IOS 10+ has built in speech recognition library. There are also numerous 3rd party libraries that provide STT function.

MMO download

To receive download messages, adaopt the IPSAKChatConversationDelegate protocol and implement 'didReceiveDownload-Message' method:

```
- (void)chat:(IPSAKChat *)chat didReceiveDownloadMessage:(IPSAKDownloadMessage *)message {
    [message download:^(IPSAKDownloadedMMO *downloadedMMO, NSError *error) {

        if (error != nil) {
            dispatch_async(dispatch_get_main_queue(), ^{
                [self addErrorMessage:error.localizedDescription];
            });
        } else if (downloadedMMO != nil) {
            [self addMediaMessage:downloadedMMO];
        } else {
            dispatch_async(dispatch_get_main_queue(), ^{
                [self addErrorMessage:@"Can't download unknown content."];
            });
        }
    }];
}
```

In order to download the file, an additional method call is required. However, the downloading part is already built in the SDK. So all you need to do is calling method 'download' on the IPSAKDownloadMessage and then it will return the downloadedMMO on completion. The downloadedMMO contains file data and other information such as mime type.

MMO upload

If the IPSAKChatConversationDelegate's protocol method is implemented, once there is an upload request message, it will be called:

```
- (void)chat:(IPSAKChat *)chat uploadRequested:(IPSAKUploadMessage *)uploadRequest {
    if (uploadRequest.fileType == IPSAKUploadFileTypeImage) {
        [self presentImagePicker];
    } else {
        [self addInfoMessage:[NSString stringWithFo
        rmat:@"Client doesn't know how to handle uploads of %@.", uploadRequest.fileTypeString]];
    }
}
```

Once file data is received, call AmeliaChat's upload method and pass in the file's data and type: `[self.chat uploadFileType:@"IMAGE" data:imageData filename:filename];`

If we provide the below method implementation for protocol IPSAKChatConversationDelegate, then we could receive feedback on our upload progress:

```
- (void)chatUploadSuccess:(IPSAKChat *)chat;
- (void)chat:(IPSAKChat *)chat uploadFailWithError:(NSError *)error;
```

Forms

When IPSAKChatConversationDelegate protocol is adopted and an didReceiveFormInputMessage method is impelemented, a form message can be received:

```
-(void)chat:(IPSAKChat *)chat didReceiveFormInputMessage:(IPSAKMessage *)message;
     if (message.formInputData != nil) {//form message
     }
}
```

You can contruct a custom UI with the form data, and when form response data is received from user, send the form input with response data with the AmeliaChat API:

```
IPSAKFormInputData *formInputData = form.content;
[self.chat submitForm:formInputData message:nil];
```

For a complete example, please see the sample App code.

Integration message

Integration messages can be received when IPSAKChatConversationDelegate protocol is adopted and method 'didReceiveIntegrationMessage' is implemented:

```
- (void)chat:(IPSAKChat *)chat didReceiveIntegrationMessage:(IPSAKMessage *)message {
    if (message.messageType == IPSAKMessageTypeOutboundIntegration) {
        NSString *sectionType = message.integrationMessageData[@"sectionType"];
        if ([sectionType isKindOfClass:[NSString class]] && [sectionT
        ype isEqualToString:@"AccordionSection"]) {
            [self showIntegrationView:message.integrationMessageData];
        } else {
            [self addInfoMessage:@"Integration message not supported."];
        }
    }
}
```

Integration message is entirely freeform and it is up to the app developer to find out what the integration message standard is and to decide on how to present the data. The integration data is stored in property 'integrationMessageData' as a NSDictionary in the IPSAKMessage class.

Data masking

When Amelia asks a question that requires responses to be secured, e.g. password, credit card number, the user input needs to be masked. The SDK will fire a change event to notify the developer that the secure input state has changed. Here is an example implementing the event listener method of IPSAKChatConversationDelegate protocol:

```
- (void) setSecureUserInputFlag :(NSDictionary *)attributes {
    NSString * secureInput = attributes[@"secureUserInput"];
    if(secureInput != nil && [secureInput isEqualToString:@"yes"]){
        _toSecureResponse = true;
    }else{
        _toSecureResponse = false;
    }
}
```

Only when the secure input enabled state is changed, this event will be fired. Otherwise it indicates the secure input state stays the same. The initial state at new conversation is always false.

Third Party Dependencies

SocketRocket is used for the websocket implementation.