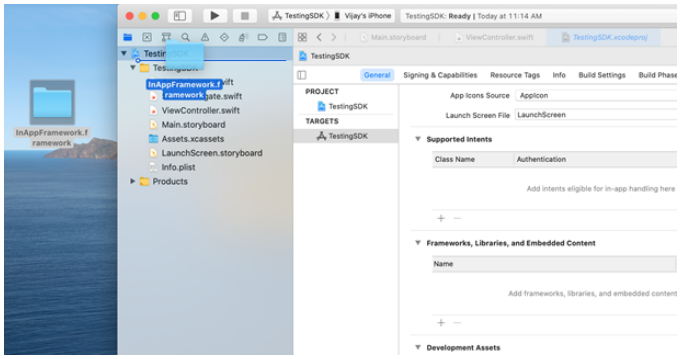
03 - Native App Integration (iOS)

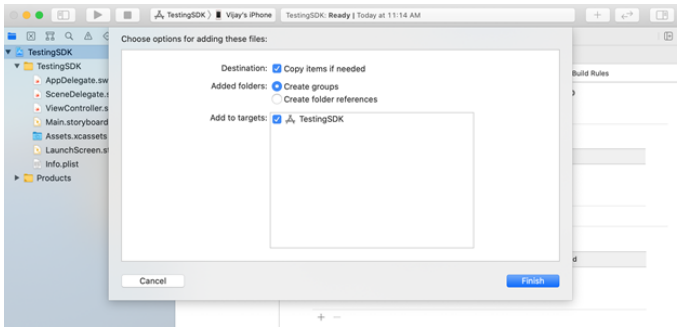
Procedures

Here are the steps necessary to integrate the in-app survey iOS SDK (In-App Survey Module):

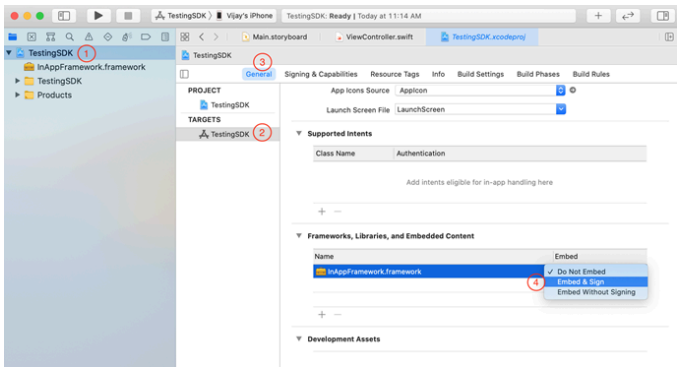
1. With your project open, drag the provided framework file into the project navigator.



2. The options for adding these files will appear. Ensure that “Copy items if needed” is checked and click **Finish**.



3. Select “Embed & Sign” in **Frameworks, Libraries, and Embedded Content** section located by clicking on the target and the **General** tab.



4. To inject a survey, do two things:

a. Add the necessary import:

import InAppFramework

b. Add the call to present the survey to the user. Here are the steps, with code:

Step 1 – Create Rule Request Model Object

var ruleRequestModel = RuleRequestModel(programToken: "programToken_value", eventName: "event_name_value")

Step 2 – (Optional) Add Prepopulated data in rule request if needed

ruleRequestModel.addPrePopulationData(nameField: "FirstName", nameFieldValue: "firstNameValue")
ruleRequestModel.addPrePopulationData(nameField: "LastName", nameFieldValue: "lastNameValue")

Step 3 – (Optional) Provide custom prompt configuration if you need a custom invitation to display

If you want to use the default prompt, then skip this step. For more details, see the table Custom Prompt Parameters and Descriptions.

You can create a CustomPromptModel instance in 2 ways:

1. Create an instance using an empty initializer and provide a value for each property:

```
var customPrompt = CustomPromptModel()
customPrompt.headerImageURL = "URL here"
customPrompt.bodyText = "Your feedback is incredibly valuable and will allow us to focus on those aspects that are most important to you. <br /><br /> Please take our 1 minute survey."
customPrompt.bodyBackgroundColor = "#ffffff"
customPrompt.mainBackgroundColor = "#ffffff"
customPrompt.noButtonBackgroundColor = "#FF6900"
customPrompt.noButtonText = "No Thanks"
customPrompt.noButtonTextColor = "#ffffff"
customPrompt.yesButtonBackgroundColor = "#FF6900"
customPrompt.yesButtonText = "Start Survey"
customPrompt.yesButtonTextColor = "#ffffff"
customPrompt.footerBackgroundColor = "#003366"
customPrompt.footerText = "Please contact if need more details."
customPrompt.footerTextColor = "#ffffff"
customPrompt.footerHeaderText = "Thank you very much."
customPrompt.optOutButtonText = "Opt-Out"
customPrompt.optOutButtonTextColor = "#FF6900"
```

customPrompt.disclosureText = "If you decline this invitation, we will not offer this survey again while using this platform for a period of time."

customPrompt.disclosureTextColor = "#000000"

2. Create an instance by providing the dictionary as an argument to the initializer:

var json = [String: Any]()

json["headerImageURL"] = "https://images.pexels.com/photos/676...oms-67636.jpeg"

json["bodyText"] = "Your feedback is incredibly valuable and will allow us to focus on those aspects that are most important to you.

 Please take our 1 minute survey."

json["bodyBackgroundColor"] = "#ffffff"

json["mainBackgroundColor"] = "#ffffff"

json["noButtonBackgroundColor"] = "#FF6900"

json["noButtonText"] = "No Thanks"

json["noButtonTextColor"] = "#ffffff"

json["yesButtonBackgroundColor"] = "#FF6900"

json["yesButtonText"] = "Start Survey"

json["yesButtonTextColor"] = "#ffffff"

json["footerBackgroundColor"] = "#003366"

json["footerText"] = "Please contact if need more details."

json["footerTextColor"] = "#ffffff"

json["footerHeaderText"] = "Thank you very much."

json["optOutButtonText"] = "Opt-Out"

json["optOutButtonTextColor"] = "#FF6900"

json["disclosureText"] = "If you decline this invitation, we will not offer this survey again while using this platform for a period of time."

json["disclosureTextColor"] = "#000000"

let customPrompt = CustomPromptModel(json: json)

Note: By providing the parameter values above you can configure the custom prompt invitation. See the table Custom Prompt Parameters and Descriptions for more details.

Step 4 – (Optional) If you want a callback after the survey is finished or canceled

MCXInAppSurvey.shared.surveyDelegate = self

Delegate is used as a callback function. Once a survey is completed or cancelled then you will get a callback in the SurveyResultDelegate method. To get a callback we need to set the Delegate in the calling class.

```
extension ViewController: SurveyResultDelegate {
func surveyDidFinished(_ result: Bool, errorMessage: String?) {
```

```
    }
}
```

If the survey is finished successfully then the result will be true and contain no error message; otherwise the result will be false and provide the error message why the survey was not completed.

You can also perform another action like push or present next controller in the Delegate function, which can replace Step 5.

Step 5 – (Optional) If you want to push a controller after the survey is finished or canceled

To do this, create an instance of the next controller and pass it to the initiateSurvey method:

```
let nextController = self.storyboard?.instantiateViewController(withIdentifier: "SecondViewController")
```

Step 6 – Present survey with alternate approaches

1. Present survey with the default prompt (no custom invitation prompt):

```
MCXInAppSurvey.shared.initiateSurvey(controller: self, ruleRequestModel: ruleRequestModel)
```

2. Present survey with a custom prompt:

```
MCXInAppSurvey.shared.initiateSurvey(controller: self, ruleRequestModel: ruleRequestModel, customPromptModel: customPrompt)
```

3. Present survey with a custom prompt and the next controller which needs to be pushed after the survey is completed or fails.

```
MCXInAppSurvey.shared.initiateSurvey(controller: self, ruleRequestModel: ruleRequestModel, customPromptModel: customPrompt, nextController: nextController)
```

Not Passing in the Locale, Resuming to Subsequent Controller

```
var ruleRequestModel = RuleRequestModel(programToken: "program_token_here", eventName: "event_name_here")
MCXInAppSurvey.shared.surveyDelegate = self
MCXInAppSurvey.shared.loggingEnabled = true
MCXInAppSurvey.shared.initiateSurvey(controller: self, ruleRequestModel: ruleRequestModel, customPromptModel: customPrompt, nextController: nextController)
```

Here “customPrompt” is an instance of “CustomPromptModel” configured in Step 3 above. Pass “nil” if you don’t want custom invitation. SDK will display default alert dialogue.

Passing in the Locale, Resuming to Subsequent Controller

```
var ruleRequestModel = RuleRequestModel(programToken: "program_token_here", eventName: "event_name_here")
ruleRequestModel.setLocale(Locale.current)
```

OR

```
let locale = Locale(identifier: "en_US")
ruleRequestModel.setLocale(locale)
MCXInAppSurvey.shared.surveyDelegate = self
MCXInAppSurvey.shared.loggingEnabled = true
MCXInAppSurvey.shared.initiateSurvey(controller: self, ruleRequestModel: ruleRequestModel, customPromptModel: customPrompt, nextController: nextController)
```

Note: If no locale is provided, then the SDK will use the current one.

Passing in the Locale, Staying on the Same Controller

```
var ruleRequestModel = RuleRequestModel(programToken: "program_token_here", eventName: "event_name_here")
ruleRequestModel.setLocale(Locale.current)
```

OR

```
let locale = Locale(identifier: "en_US")
ruleRequestModel.setLocale(locale)
MCXInAppSurvey.shared.surveyDelegate = self
MCXInAppSurvey.shared.loggingEnabled = true
MCXInAppSurvey.shared.initiateSurvey(controller: self, ruleRequestModel: ruleRequestModel, customPromptModel: customPrompt)
```

Note: If no locale is provided, then the SDK will use the current locale.

Debugging

You can enable logging by adding the line below before the initiateSurvey method.

```
MCXInAppSurvey.shared.loggingEnabled = true
```

So, what is the call doing? You create a model and provide the In-App Survey SDK some details. They are:

- The optional locale.
- The program token: provided to you by your business owner or manager.
- The event name: this is a value agreed upon between you and your business owner/manager. It's the way you indicate the context of the user (just hit a button, landed on a screen, etc.).
- Custom prompt: you will need to configure the properties provided in Step 3. See the table Custom Prompt Parameters and Descriptions for more details.

From there you're passing off control to the In-App Survey SDK. You may or may not pass in the controller that you would for the user to land on once the survey is completed. You can use SurveyResultDelegate to give control back to your app

and push a particular controller or display message, etc.

Do you need to pass in agreed-upon [the MaritzCX implementation team and client app implementation team must agree upon naming convention] survey prefill values? If so, do this to the RuleRequestModel() instance before calling the initiateSurvey() method. This is explained in Step 2.

ruleRequestModel.addPrePopulationData(nameField: "FirstName", nameFieldValue: "iOS")

Execute this method for each name/value that you wish to prefill.

Custom Prompt Parameters and Descriptions

Parameter Name	Description	Required	Default Value
headerImageURL	Header Image URL to display image on header of prompt	Required	
bodyText	Body text displayed on prompt	Required	
bodyBackgroundColor	Body background color in HEX format		white color
mainBackgroundColor	Custom Prompt background color in HEX format		white color
noButtonText	No button text/title	Required	
noButtonTextColor	No button text color in HEX format		black color
noButtonBackgroundColor	No button background color in HEX format		white color
yesButtonText	Yes button text/title	Required	
yesButtonTextColor	Yes button text color in HEX format		black color
yesButtonBackgroundColor	Yes button background color in HEX format		white color
footerBackgroundColor	Footer background color in HEX format		white

Parameter Name	Description	Required	Default Value
			color
footerText	Footer text displayed in custom prompt footer section, if not passed then no footer text present		
footerTextColor	Footer text color in HEX format		black color
footerHeaderText	Footer header text displayed above footer text	Required	
optOutButtonText	Opt-out button text/title, if not passed then opt-out will be hidden.		
optOutButtonTextColor	Opt-out button text color in HEX format		black color
disclosureText	Disclosure text, displayed below the Yes and No buttons and replacing the opt-out button. If “optOutButtonText” is provided then this value will be ignored.		
disclosureTextColor	Disclosure text color in HEX format		black color

The image below shows the custom prompt parameter and UI element mapping: