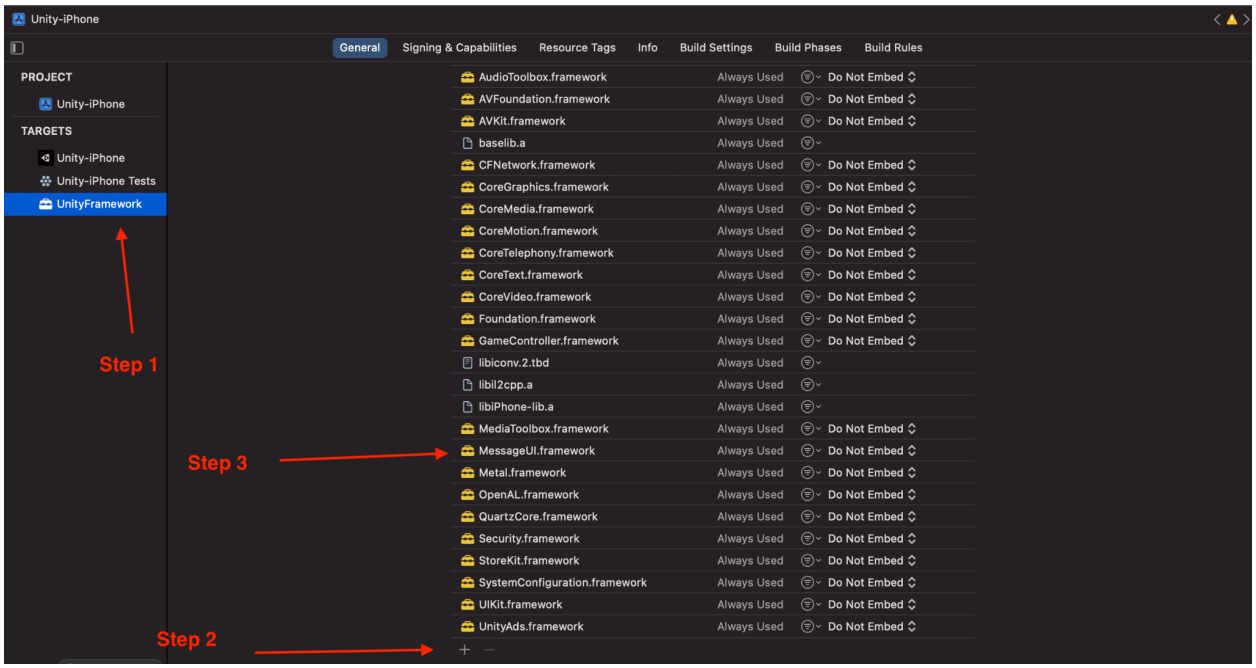
Email Composer For iOS

Plugin will allow developers to send emails from Unity Games in iOS by using native Mail Composer of iOS. Plugin provides many features to customize the Email according to requirements. Developers can set recipients, ccRecipients, bcc-recipients, subject and body. Developers can even send an email with a screenshot of the game. Read this document carefully. Take a look at the DemoScene in Scenes folder.

SetUp

Import the asset package in the assets folder. Ensure that both **Email.h** and **Email.mm** exist in Plugins/iOS/ folder.

Open the project in Xcode. Add **MessageUI.framework** in the project. To add the same click on '+' button at bottom under the **Linked Frameworks and Libraries** section in the **General** tab of the **Unity-Framework** Target. Please see screenshot below :



API for Email Composer

➔ To show the EmailComposer use following method :

EmailComposer.ShowMailComposer();
This Api will open the blank native mail composer.

//----------

➔ To set the EmailComposer with recipients , ccRecipitents , bccRecipitents , subject and body content use following method :

StartCoroutine(
EmailComposer.ShowMailComposerWithSubjectBodyCCReceiptentsBCCReceiptentsAndScreenShot(string subject, string content ,string [] receiptent, string [] ccReceiptents, string []bccReceiptents, bool isScreenshot));

In above code

1. **subject** represents the Subject which developers want to set for the email and if not required developers can pass empty string.
2. **content** represents the body of email which developers want to set and developers can leave it blank is not required.
3. **receiptent** represents the recipients developers want to add by default in the mail compose and developers can leave it blank if not required. Developers should pass a string array in this parameter.
4. **ccReceiptents** represents the cc recipients developers want to set and developers can leave it blank if now required. Developers should pass a string array in this parameter.

5. b**ccReceiptents** represents the bcc recipients developers want to set and developers can leave it blank. Developers should pass a string array in this parameter.
6. In the last parameter if developers pass **true** then it will take a screenshot of the device and attach it to the email itself and if you pass **false** then it will open mail composer without screenshot.

//----------

➔ To receive callback from the plugin use following method:

EmailComposer.SetCallbackMethod(string gameObject, string methodName);

If developers want to receive the call back from the plugin about whether the user has sent the image, saved in draft or canceled it then developers should use this api before opening mail app. Refer **EmailScript.cs** for its usage. Messages are :

1. Mail Cancelled
2. Mail Saved
3. Mail Sent
4. Mail sent failure: // Developers will also get error message

//----------

➔ To send email with Custom Image use following method:

StartCoroutine(EmailComposer.ShowMailComposerWithSubjectBodyCCReceiptentsBCCReceiptentsAndImageName(string subject, string content, string [] receiptent, string [] ccReceiptent, string [] bccReceiptent, string imageName))

If developers want to send an email along with a custom image, developers can use the above API. For this api to work, developers need to convert the image to bytes and save it in the Application persistent path so that plugin

can use it from there. In demo we have converted a 2D texture into bytes and saved it in application persistent path in method **SaveTextutreToApplicationPathAndSendEmail** and then invoked method **ShowMailComposerWithSubjectBodyCCReceiptentsBCCReceiptentsAndImageName** with the image name which has been used to save the image in application persistent path. If developers want to convert any other image type except Texture2D then developers can just save image to persistent path and pass the image name in **ShowMailComposerWithSubjectBodyCCReceiptentsBCCReceiptentsAndImageName** method.

//----------

➔ **To send email with CSV file use following method:**

StartCoroutine(EmailComposer.ShowMailComposerWithSubjectBodyCCReceiptentsBCCReceiptentsWithCsvFile(string subject, string content, string [] receiptent, string [] ccReceiptent, string [] bccReceiptent, string fileName));

If developers want to send an email along with a custom csv file, developers can use the above API. For this api to work, developers need to save the csv file in the Application persistent path so that the plugin can use it from there. In demo we have created a csv file and saved it in application persistent path and then invoked method **ShowMailComposerWithSubjectBodyCCReceiptentsBCCReceiptentsWithCsvFile** with the file name which has been used to save the csv file in application persistent path.

//----------

➔ **To send email with text file use following method:**

StartCoroutine(EmailComposer.ShowMailComposerWithSubjectBodyCCReceiptentsBCCReceiptentsWithTextFile(string subject, string content, string [] receiptent, string [] ccReceiptent, string [] bccReceiptent, string fileName));

If developers want to send an email along with a custom text file with format txt, developers can use the above API. For this api to work, developers need to save the text file in the Application persistent path so that the plugin can use it from there. In demo we have created a text file and saved it in application persistent path and then invoked method **ShowMailComposerWithSubjectBodyCCReceiptentsBCCReceiptentsWithTextFile** with the file name which has been used to save the text file in application persistent path.

Note : In case a user has not logged in its native mail app on an iOS device then email can't be sent. For that reason the plugin will return the following error message : **Can not send mail on this device. Either account is not logged in Mail app or can not access it.**

Please contact us at **guptamayank516@gmail.com** in case of any query or clarifications.