Citrix MAM iOS SDKs - Local Authentication SDK - Developer guide

Last updated on: 2020-02-03 19:00 EST

Overview

Local Auth SDK is designed for apps that need local authentication. It implements App Passcode policy and Max Offline Period policy. Customer apps that use the Local Auth SDK are called third party apps. These apps created using the Local Auth SDK (and possibly other CTXMAM SDKs) should be able to use the same Local Auth SDK and will have their own inactivity timer record and it will be shared among the third party apps. However, this inactivity timer record will not be shared with first party apps since they are signed with a different provisioning profile. Thus the Inactivity timer and Max Offline Period policies are not applied across app silos.

App Passcode

When the AppPasscode app policy is set to true, the Local Auth SDK will consider the value of Inactivity Timer policy value which is set in minutes (default 60 minutes). The Local Auth SDK sets up a timer to check for User Inactivity and when the Inactivity Timer expires it will display Touch ID/ Face ID/ Device passcode Auth prompt to the user to authenticate. If the user successfully authenticates then the SDK will reset the Inactivity Timer and the user will be able to use the app. The Inactivity countdown will restart and it will reset when the user interacts with the app. If the user cannot successfully authenticate then the auth prompt will continue and the app will be unusable to the user until the user successfully authenticates.

Max Offline Period

Max Offline period is a period during which the app can be used when the device is offline and is specified as an app policy in hours (default 0). When the offline period expires the user will need to enable the Internet connection and login to the CEM server using Secure Hub app. The Local Auth SDK will try to refresh the policies using the Core SDK and if needed the app will flip to SecureHub. However, if the user is offline or Secure Hub returns any other other failure, then the user will need to make sure that the login to Secure Hub succeeds so that the policy refresh takes place and the app entitlement check succeeds.

When the offline period expires the app will become temporarily unusable until the user does the login in SecureHub. There will be offline period warning alert messages as the offline period approaches. The warnings will be 30 minutes, 15 minutes and 5 minutes before the Max Offline period expires.

Functionality provided and differences with the legacy MDX toolkit functionality

App Passcode

The Max Number Of Retries policy will not be present in Local Auth SDK. Thus the device lock or wipe policies will not apply when the user fails the authentication challenge. When the device Passcode authentication fails for 5 times then iOS disables the passcode validation screen for incrementing times starting from 1 minute to 5 minutes, to 15 minutes to 60 minutes and finally it disables the iPhone and when this happens the user cannot use the device. This is standard iOS behavior and so the user experience will be different than legacy MDX.

For the Local Auth SDK the TouchID Enabled policy is not supported. We will rely on iOS behavior wherein iOS will first use Touch ID/ Face ID if present and enabled and if it's not present or locked out due to incorrect attempts, then it will use the device passcode.

The SecureHub PIN related policies like PIN type, PIN strength and PIN length requirement will not apply to the App Passcode policy since we are using the device passcode to authenticate. Thus these policies are deprecated for Local Auth SDK. They will continue to function for Secure Hub.

The App lock and wipe policy will not be handled as part of this policy and the Max Number Of Retries policy will not be considered. This is because we will rely on iOS to lock or wipe the device as necessary after authentication failures. Currently the max number of authentication failures before iOS locks the device is 10.

The following is the list of policies related to App Passcode that will not apply in Local Auth SDK.

- Touch ID Enabled
- Max Number Of Retries
- PIN type
- PIN strength
- PIN length
- Wipe device on Lock

Max Offline Period

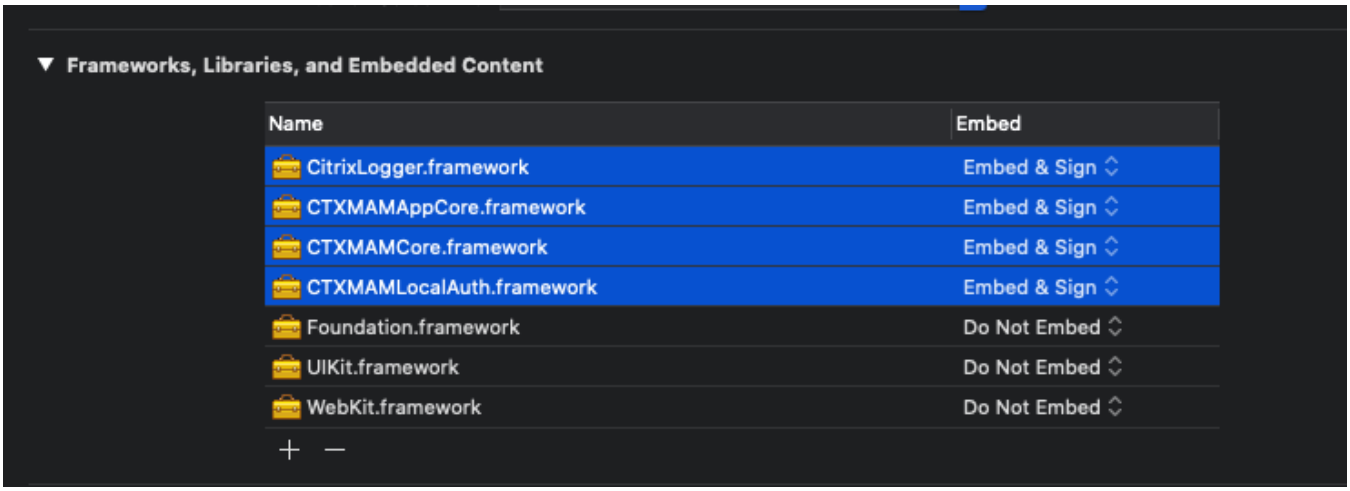
The functionality for Max Offline Period policy will be disabled when the value for the Max Offline Period policy is 0. The Local Auth SDK will consider the policy refresh time and if the policy is not refreshed for the specified period then the Local Auth SDK will send a delegate callback to the app. Also, there will be delegate callbacks, when the Max Offline period is approaching, and it will be 30 minutes, 15 minutes and 5 minutes before the Max Offline period expires. If the app didn't set the delegate then the Local Auth SDK will display alert window with a "Quit" button that will exit the app and a "Logon" button, when the Max Offline period expires. Also if the app didn't set the delegate, then another alert window with "Ok" button and a "Logon" button, will be displayed when the max offline period is approaching, and similar to above delegate callbacks, it will be 30 minutes, 15 minutes and 5 minutes before the Max Offline period expires.

When the app developer implements the delegate pop-up, if the "Logon" button is clicked, the app should call a new API in the Core SDK that flips to SH and does the login. In the future when we add AML support, we will check if AML can be used for in-app login and if so the flip to SH will be skipped and auth will occur in-app. If the delegate is not implemented the Local Auth SDK will put up a similar popup.

Steps to integrate the Local Auth SDK

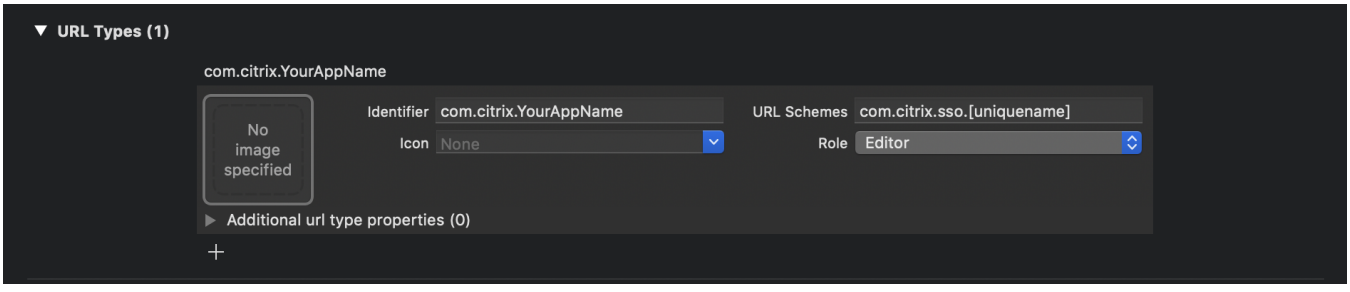
1. Add **CTXMAMLocalAuth.framework** to your project. If not previously added, add the **CTXMAMCore.framework, CTXMAMAppCore.framework** and **CitrixLogger.framework** as well.

- On the project target properties under the "General" tab,
  - (In Xcode 11+), add these framework(s) in the “Frameworks, Libraries, and Embedded Content” section and select “Embed & Sign” option from the Embed dropdown.
  - (Prior to Xcode 11), add these framework(s) listed above to the "Embedded Binaries" section.

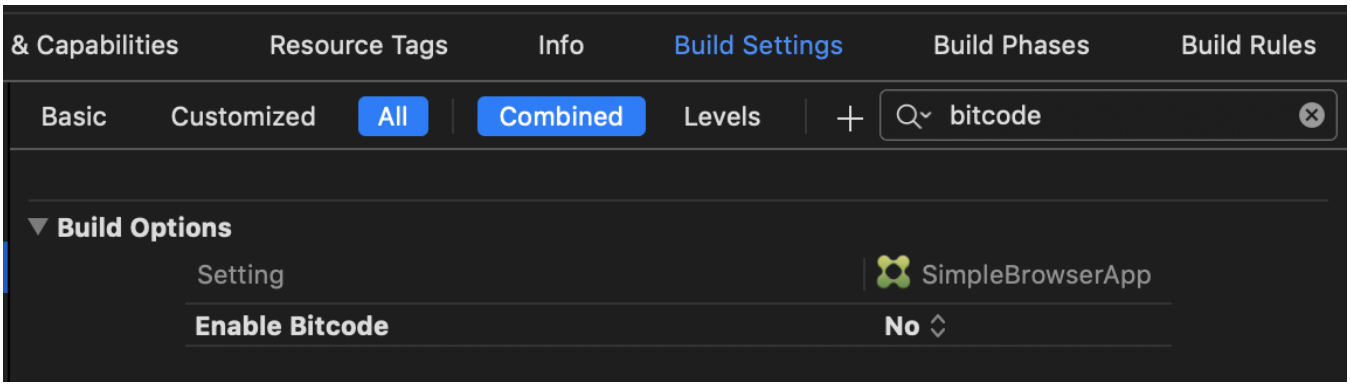


2. On the project target properties under the "Info" tab, add a new URL Type with URL Scheme which starts with com.citrix.sso.[PACKAGEID] (e.g. com.citrix.sso.22A955C3-237B-4952-A271-567BDFFA28BA). PACKAGEID can be generated by running "uuidgen" command in your MacOS Terminal. You may skip this step is an URL Scheme with “com.citrix.sso.” prefix was previously added.

Note: Use this exact PACKAGEID in the Run Script mentioned in the subsequent step.

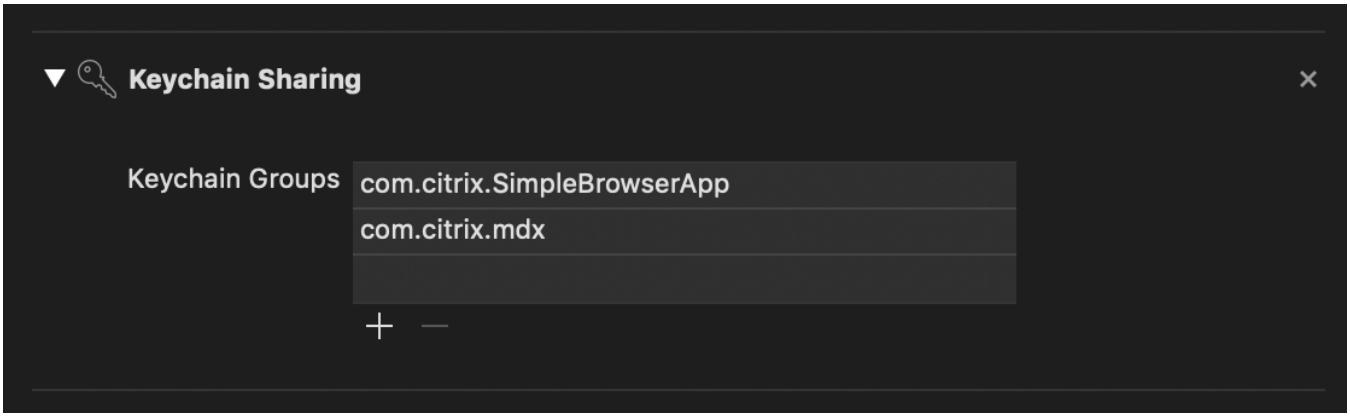


3. On the project target properties under the "Build Settings" tab, set "Enable BitCode" to "No".



4. On the project target properties under "Capabilities" tab, enable "Keychain Sharing".

1. Add "com.citrix.mdx" access group to the list of "Keychain Groups", if it was not previously added.



5. Import <CTXMAMCore/CTXMAMCore.h> into your AppDelegate.m file, if it was not previously imported.

Import headers code block

```
//
//  AppDelegate.m
//  Your App Name here
//
//  Created by Your Name Here on ??/??/??.
//  Copyright © 2020 Your Company Name Here, LLC. All rights reserved.
//

#import "AppDelegate.h"
#import <CTXMAMCore/CTXMAMCore.h>
```

6. In the AppDelegate.m in your application didFinishLaunchingWithOptions, add a call to initialize the SDKs, [CTXMAMCore initializeSDKs], if it was not previously added.

Application Did Finish Launching override

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    // Override point for customization after application launch.
    [CTXMAMCore initializeSDKs];

    return YES;
}
```

7. Integrate Local Authentication SDK in AppDelegate (or in another class instance)

    Local Auth SDK provides a set of delegate methods, that are defined in the protocol CitrixLocalAuthSdkDelegate, for the application to handle. It is recommended that the app developers handle each of these callbacks as per the suggested behavior in the documentation. If the app developers choose not to implement some of these delegate methods, then the Local Auth SDK will handle those callbacks. These callbacks will be sent asynchronously to the app on the main queue.

    More information on handling these callbacks can be found in the Quick Help of these delegate methods. For example, for the devicePasscodeRequired delegate method we see the following in Quick Help:

To integrate Local Auth SDK,

1. Import <CTXMAMLocalAuth/CTXMAMLocalAuth.h>.
2. Conform your AppDelegate class to Local Authentication SDK delegate.
3. Add all the delegate methods.

Application Did Finish Launching override

```
#import "AppDelegate.h"
#import <CTXMAMCore/CTXMAMCore.h>

#import <CTXMAMLocalAuth/CTXMAMLocalAuth.h>


@interface AppDelegate () <CitrixLocalAuthSdkDelegate>

@end


@implementation AppDelegate

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    // Override point for customization after application launch.
```

```
    [CTXMAMCore initializeSDKs];

    [CTXMAMLocalAuth setDelegate:self];

    return YES;
}



#pragma mark - Local Authentication SDK delegate methods
- (void) maxOfflinePeriodWillExceedWarning:(NSTimeInterval) secondsToExpire
{
    NSLog(@"Received maxOfflinePeriodWillExceedWarning");

    NSString * alertTitle = @"Warning message from App:";
    NSString * alertMsg = [NSString stringWithFormat:@"Offline lease will expire in %f seconds. Please go
online and login.", secondsToExpire];
    UIAlertController * alert = [UIAlertController
        alertControllerWithTitle:alertTitle
                         message:alertMsg
                  preferredStyle:UIAlertControllerStyleAlert];
    UIAlertAction *actionOk = [UIAlertAction actionWithTitle:@"Ok"
                                                       style:UIAlertActionStyleDefault
                                                     handler:nil];

    [alert addAction:actionOk];

    dispatch_async(dispatch_get_main_queue(), ^{
        UIViewController *topController = [UIApplication sharedApplication].keyWindow.rootViewController;
        [topController presentViewController:alert animated:YES completion:nil];
    });
}

- (void) maxOfflinePeriodExceeded
{
    NSLog(@"Received maxOfflinePeriodExceeded");

    NSString * alertTitle = @"Error message from App:";
    NSString * alertMsg = @"Offline lease has expired. Please login again.";
    UIAlertController * alert = [UIAlertController
        alertControllerWithTitle:alertTitle
                         message:alertMsg
                  preferredStyle:UIAlertControllerStyleAlert];

    UIAlertAction * quitButton = [UIAlertAction actionWithTitle:@"Quit"
                                                       style:UIAlertActionStyleDestructive
                                                     handler:^(UIAlertAction * action) {
                                    NSLog(@"Application will terminate");
                                    id appdelegate =   [[UIApplication sharedApplication] delegate];
                                    [appdelegate applicationWillTerminate:[UIApplication
sharedApplication]];
                                    exit(0); // terminate the app
                                }];

    [alert addAction:quitButton];

    dispatch_async(dispatch_get_main_queue(), ^{
        UIViewController *topController = [UIApplication sharedApplication].keyWindow.rootViewController;
        [topController presentViewController:alert animated:YES completion:nil];
    });
}

- (void) devicePasscodeRequired
{
    NSLog(@"Received devicePasscodeRequired");

    NSString * alertTitle = @"Error message from App:";
    NSString * alertMsg = @"Please set the device passcode and Touch ID/FaceID since it is required when
Inactivity Timer expires.";
    UIAlertController * alert = [UIAlertController
        alertControllerWithTitle:alertTitle
```

```
                        message:alertMsg
                  preferredStyle:UIAlertControllerStyleAlert];

    UIAlertAction * quitButton = [UIAlertAction actionWithTitle:@"Quit"
                                                          style:UIAlertActionStyleDestructive
                                                        handler:^(UIAlertAction * action) {
                                        NSLog(@"Application will terminate");
                                        id appdelegate =   [[UIApplication sharedApplication] delegate];
                                        [appdelegate applicationWillTerminate:[UIApplication
sharedApplication]];
                                        exit(0); // terminate the app
                                    }];

    [alert addAction:quitButton];

    dispatch_async(dispatch_get_main_queue(), ^{
        UIViewController *topController = [UIApplication sharedApplication].keyWindow.rootViewController;
        [topController presentViewController:alert animated:YES completion:nil];
    });
}


@end
```

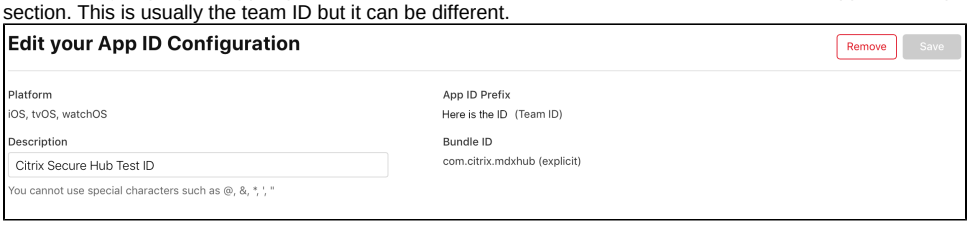
8. Back on the project target properties under the "Build Phases" tab add a new "Run Script" phase with the following script.

MDX File Creation Script

```
# Type a script or drag a script file from your workspace to insert its path.
export STOREURL="http://yourstore.yourdomain.com"
export APPTYPE="sdkapp"
export PACKAGEID="" #Please run uuidgen at the command line and paste the output value in the PACKAGEID
variable, This has be the same UUID which was generated in step 2.
export APPIDPREFIX=""#Your apps prefix id which can be found in your apple's developer account page.
export TOOLKIT_DIR="$PROJECT_DIR/../Tools"

if [ -z "${PACKAGEID}" ]
then
    echo "PACKAGEID variable was not found or was empty, please run uuidgen at the command line and paste the
output value in the PACKAGEID variable in your post build script."
    exit 1
fi

if [ -z "${APPIDPREFIX}" ]
then
    echo "APPIDPREFIX variable was not found or was empty, please refer to the \"how to\" document located in
the documentation folder of the SDK package on where to find your Apple's application prefix ID."
exit 1
fi
if [! -d $TOOLKIT_DIR/logs ]
then
    mkdir $TOOLKIT_DIR/logs
fi

"$TOOLKIT_DIR/CGAppCLPrepTool" SdkPrep -in "$CONFIGURATION_BUILD_DIR/$EXECUTABLE_FOLDER_PATH" -out
"$CONFIGURATION_BUILD_DIR/$EXECUTABLE_NAME.mdx" -storeURL "${STOREURL}" -appType "${APPTYPE}" -packageId
"${PACKAGEID}" -entitlements "$SRCROOT/$PROJECT/$PROJECT.entitlements" -appIdPrefix "${APPIDPREFIX}" -
minPlatform "9.0"
```

a. Ensure that you paste the same PACKAGEID used in the above step to add the new unique URL Scheme with the prefix "com.citrix.sso.", in the PACKAGEID field of this script.

b. Ensure that you put your app id prefix in the APPIDPREFIX field. This ID can be found in the Apple Developer portal under the AppIDs section. This is usually the team ID but it can be different.



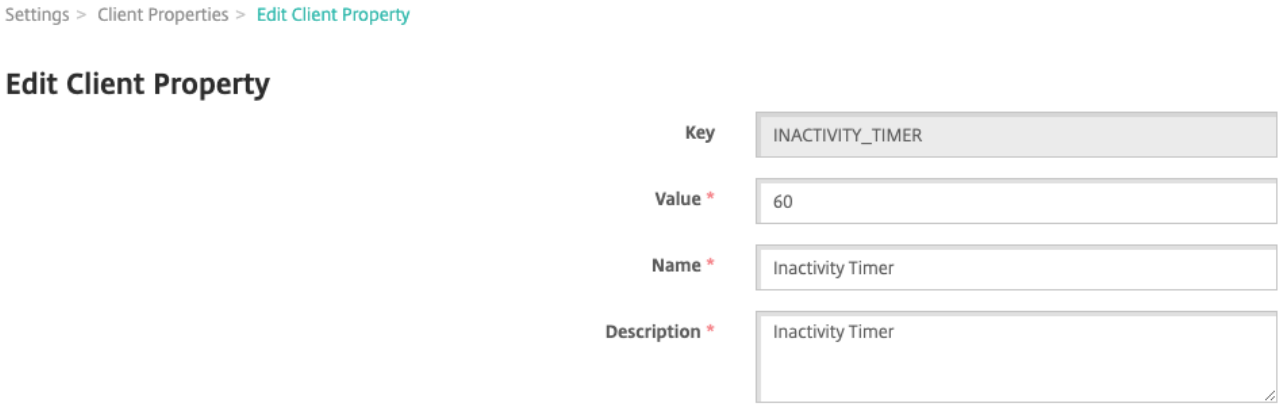
c. Ensure that the TOOLKIT_DIR property points to the Tools folder in the SDK package.

9. The above step will generate a .mdx package that can be uploaded to Citrix Endpoint Management (CEM). This package will have the following app policies that can be set in the CEM Server for the proper functioning of the Local Auth SDK.

App Passcode

The CEM administrator needs to enable the App passcode policy in the app policies section.

Also the CEM administrator needs to set the Inactivity Timer policy value in minutes in the Client properties section.



When the App Passcode policy is enabled then the Local Auth SDK will detect the user inactivity. If there is no user activity for the duration specified in the Inactivity Timer policy value, then the Local Auth SDK will prompt for Auth using Touch ID / Face ID if enabled. If Touch ID / Face ID is not enabled or it is locked out due to incorrect attempts then it will prompt for authentication using the Device Passcode. The Local Auth SDK will always make sure that before the app is being used the device passcode is present if the App Passcode policy is enabled.

MaxOffline Period

To enable the Max Offline period the administrator needs to set a non zero value for the App policy as shown below:

If the value is set to 0 then the policy will be disabled. The default value displayed in the CEM is 168.

Embedding your IPA file into the MDX file for publication on the CEM Server

1. Create you IPA by going thru the archiving process in Xcode.
2. Run the following command:

Embedding IPA File

```
export TOOLKIT_DIR="$PROJECT_DIR/../Tools"
export IPA_FILE_PATH="Provide IPA File Path"
export EXECUTABLE_NAME="Provide the name for your application's app folder."
#CONFIGURATION_BUILD_DIR comes from Xcode.


"$TOOLKIT_DIR/CGAppCLPrepTool" SetInfo -in "$CONFIGURATION_BUILD_DIR/$EXECUTABLE_NAME.mdx" -out
"$CONFIGURATION_BUILD_DIR/$EXECUTABLE_NAME.mdx"  -embedBundle "${IPA_FILE_PATH}"
```

Sample App

A sample app called "Simple Browser app" is included in the downloads folder of the iOS MAM SDKs. All the iOS MAM SDKs are included and used in this sample app. Developers should be able to directly launch and verify the functionality of the SDKs and take similar steps for handling the delegate callbacks as shown in the app.

Extra Notes

You are required to have icons set up for your app. In particular, your 60x60 icon (the 20x20@3x icon) must be in PNG format with the name "AppIcon60x60.png"