DOWNLOAD PDF PROVISIONING AND DISTRIBUTING YOUR APPS

Chapter 1 : ios - What is a provisioning profile used for when developing iPhone applications? - Stack Over

With auto-provisioning, App Center helps you streamline the release and distribution process and makes it easier for testers and team members to install and test your app. We encourage you to give it a spin so you can spend more time coding and less time managing devices!

Apple provides three ways to distribute your application based on which developer program you are a member of: App Store or iTunes Store: Publish your application to the iTunes Store. Publish your application as a package that can be distributed on a limited number of devices for testing. Both work flows are similar except ad hoc distribution requires you to specify a list of devices when creating your provisioning profile and you need to specify two extra options file format and location when packaging your application with Studio. Contents Publishing your iOS application involves a few, somewhat involved steps. Fortunately, you have done two of these steps already if you have deployed your app to a device for testing. If you plan to sell your app, rather than distribute it for free, you need to register on the iTunes Connect site and supply legal and banking information to Apple. They must verify your legal status and banking info. That process can take as much as a week or more. An App ID is a unique identifier for your app. It is composed of two parts: It is unique to you and your developer account. You can also create a wildcard Bundle Identifier. Wildcard Bundle IDs are great for quick development â€" you do not have to create a new ID for each and every app you test. However, you cannot use some iOS features with a wildcard ID. Create and install the distribution certificate The distribution certificate attests to the identity of the entity publishing the app. While each developer on your team could have their own developer certificate, you or your company will have a single distribution certificate. This arrangement enables you to add or remove people from your development team, while not giving those folks the means to publish your final app. That will be controlled by you. The Team Agent will be the user account associated with creation and first-logon to your provisioning portal account. To create a distribution certificate: You will be returned to the Certificates page with the status listed as Pending. Wait a moment then refresh the page in your browser. Even though you are logged in as the Team Agent or Admin, you will need to approve your certificate. Download the distribution certificate. Double-click the file to install it into your keychain. Icon Back Up Your Private Key It is critical that you save your private key somewhere safe in the event that you need to develop on multiple computers or decide to reinstall your system OS. Without your private key, you will be unable to sign binaries and test your application on any Apple device. To export your private key, open up the Keychain Access Application and select login under Keychains and Keys under Category. Highlight the private key associated with your iOS Distribution Certificate. Save your key in the Personal Information Exchange. You will be prompted to create a password which will be used when you attempt to import this key on another computer. You can now transfer this. You will be prompted for the password you entered above. Create and install the distribution provisioning profile As with deploying an app for testing, you need to create a provisioning profile file. The distribution provisioning profile gathers together your distribution certificate and App ID. If this is an Ad Hoc provision profile, you also need to specify your devices. Enter a name for your provisioning profile. You should use a word like "distribution" or "ad hoc" in the name so that it is clear later that this profile is for final distribution or ad hoc distribution, respectively. You have two options to install the provisioning profile file onto your development computer. You can drag the file and drop it on the Xcode icon, or you can install it from Studio by following the steps in the following section. Either way, installing the provisioning profile is a one-time operation on each computer, until it expires. Once you are done with the preceding steps, use Studio to build your app for final distribution. If the Launch Automatically option is enabled under the Target drop-down list, the application will be automatically launched after the target is selected. If not, you need to click the Launch button to start the build process. If you have not previously distributed your app using Studio, you will see the following wizard dialog box. The wizard walks you through the necessary steps to build and distribute your app. On the Certificates page, select your distribution certificate and keychain. Click Next to proceed. Provisioning profiles are specified on the last page. To install the provisioning profile, click the Browse If you have installed more than

one provisioning profile, make sure to choose the one that corresponds to your app and your distribution certificate. Select a Provisioning Profile and click Finish. Studio packages your iOS application. For Ad Hoc distributions, the file is copied to the specified path. If you get errors, most likely source of trouble is that the app ID you entered in the tiapp. The next time you execute the Distribute action, you will see a more simplified wizard where you specify the certificate, provisioning profile, and iOS SDK. Their device needs to be included in the provisioning profile for them to use it. Click on the device you will install the app on. Your app will be installed on your device. See Deploying to iOS devices: If you have not done so yet, make sure to set up your iTunes Connect profile and supply the legal and banking information required by Apple. They verify all the documents you supply, a process that can take more than a week in some cases. You will not be able to publish your app until that process is complete. You need to supply: You can use letters, numbers, hyphens, periods, and underscores. The SKU cannot start with a hyphen, period, or underscore. Through a series of pages, you then define the following information about your app: The date on which it should be available defaults to the current date. You do not set a specific price, but instead set a tier. Each tier specifies a selling price in a selection of countries around the world. Optionally, you select specific locales in which your app will be sold.

Chapter 2 : Announcing Auto-Provisioning: Build and Distribute Apps Faster â€“ Visual Studio App Center

Distributing your App In-House. With the Apple Developer Enterprise Program, the licensee is the person responsible for distributing the application, and for adhering to the guidelines set by Apple.

Free provisioning for Xamarin. While simulator testing is valuable and convenient, it is also essential to test apps on physical iOS devices to verify that they function properly under real-world memory, storage, and network connectivity constraints. To use free provisioning to deploy an app to a device: Use the signing identity and provisioning profile created by Xcode in Visual Studio for Mac or Visual Studio to deploy your Xamarin. Important Automatic provisioning allows Visual Studio for Mac or Visual Studio to automatically set up a device for developer testing. However, automatic provisioning is not compatible with free provisioning. In order to use automatic provisioning, you must have a paid Apple Developer Program account. Requirements To deploy your Xamarin. The bundle identifier used in your Xamarin. Any bundle identifier used with free provisioning cannot be re-used. If you have already distributed an app, you cannot deploy that app with free provisioning. If your app uses App Services, you will need to create a provisioning profile as detailed in the device provisioning guide. Take a look at the Limitations section of this document for more information about limitations associated with free provisioning, and refer to the App distribution guides for more information about distributing iOS applications. Testing on device with free provisioning Follow these steps below to test your Xamarin. Use Xcode to create a signing identity and provisioning profile If you do not have an Apple ID, create one. It should look similar to the screenshot below: In Xcode, create a new project. In the new project dialog, set Team to the Apple ID that you just added. In the drop-down list, it should look similar to Your Name Personal Team: Once the new project has been created, choose an Xcode build scheme that targets your iOS device rather than a simulator. If they differ, you will not be able to use free provisioning to deploy your Xamarin. Under Deployment Info, ensure that the deployment target matches or is lower than the version of iOS installed on your connected iOS device. Under Signing, select Automatically manage signing and select your team from the drop-down list: Xcode will automatically generate a provisioning profile and signing identity for you. You can view this by clicking on the information icon next to provisioning profile: To test in Xcode, deploy the blank application to your device by clicking the run button.

Chapter 3 : IPhone provisioning profile?

A distribution certificate identifies your team/organization within a distribution provisioning profile and allows you to submit your app to the Apple App Store. A P12 file contains the certificates Apple needs in order to build and publish apps.

Signing your apps can be complicated. Signing identifies you to users. Signing also prevents spoofing your apps. Signing also allows access to system services like iCloud internet payments. What do you need in order to sign your apps? Today, you have to do things like sync private keys. Or revoke and recreate certificates. Now when Xcode builds and signs your app, behind the scenes it uses a tool called codesign. Xcode does this for you automatically. How does it work? Well, we use this codesign tool. And this allows the system to detect changes that have been made after the app has been signed. Codesign also applies a code signature to the application. So it can be identified. And people know where the app came from. There are three things that you need in order to sign your apps. The first thing is a signing certificate. Signing certificates establish your identity as a developer. Provisioning profiles are all about your apps, and they grant permissions. And entitlements declare support for capabilities. Like iCloud or Wallet. These are issued by Apple. And this is important because the device needs a trust chain back to Apple so it can install the app. And certificates come in two forms. You have development certificates, which are for build and run. And you have distribution certificates. Which are used for authentication with iTunes Connect. And all signing certificates require a private key. This private key has to be in the keychain if you want to sign your apps. And the private key for a certificate is created when the certificate gets generated. Provisioning profiles are also issued by Apple. And in contrast to certificates which are all about your identity as a developer, the profiles are all about your apps. Profiles allow access to things like running on devices. And at install time, the device runs down those list of identifiers to check if yours is included. And if it is, you can install the app. Profiles also allow access to entitlements. And if you want to use an entitlement, it has to be in that list. Xcode handles this for you. Speaking of entitlements, entitlements declare support for capabilities. And fundamentally an entitlement is just a string with an associated value. Like a string or a bowl. And these are defined per target. And Xcode has a dedicated editor for working with capabilities, called the Capabilities tab. You can do it all from within Xcode. So those are the things you need in order to sign your apps. Now what happens when Xcode goes to sign your app? It needs to know which certificate to use, and which provision and profile. The way that works is, Xcode will build your app. And then it looks at the list of certificates on your machine. And it picks the newest one that matches your development team. Xcode then looks at the list of profiles. And it picks the newest one that matches your bundle ID. And these two things are brought together. The profile is embedded in the app. And the certificate is used with Code Sign to apply the code seal. Your account already has a certificate, and Xcode offers to reset it. So what happens when you click that button? Yeah, so what happens is it gets rid of your current cert and makes a new one. And why is this? Well, your development account has a limit of one signing certificate. And if you have a single machine, this works great. The cert is created. Now what happens if you have a second Mac? Well, to build and sign on that second Mac, you need a signing cert and a private key. And you can either sync that over using keychain access or Xcode and import the private key on the second Mac. Or you can create a new signing certificate. But that first requires getting rid of the old one. And this is really frustrating and cumbersome. Well, the good news is in Xcode 8, you can now have multiple development certificates. So -- yeah [applause]. Each of your machines can have their own dedicated certificate and private key. These changes apply just for development certificates. Your distribution certs still have a limit of one that you need to manage. These are created both through Xcode or the developer websites. And these are created in Xcode 8 and later. And if you have an old copy of Xcode on your machine, and you create one of these new signing certificates, it works. But many of you also have app extensions. Or embedded frameworks in your app. And each of those brings its own signing requirements. All this can add up to a complicated picture, where signing becomes a bit of a puzzle. And fitting all the pieces together can be pretty difficult sometimes. So Xcode for many releases has provided assistance to help make this easier. In Xcode 7,

the Fix Issue button would appear whenever Xcode detected that something was wrong with your signing setup. Now sometimes clicking on Fix Issue would fix the problem, and you can return to developing your app. Or you had to click it multiple times. What it did when you clicked Fix Issue, and why. And really, if Xcode knew how to solve the problem, why did it need to interrupt your development workflow and make you click a button? Interacting with signing in Xcode 7 often felt like you were talking to a black box. We fully redesigned how signing works in Xcode 8. On top of that, we built new workflows, and a new user interface in the General tab. So you can see and manage your signing settings all in one place. Then we took a look at how Xcode communicates with you about signing issues. A lot of the error messages were not very actionable or communicative. So that you see much more helpful and actionable information when something goes wrong. And for those of you who are very curious about what Xcode does on your behalf when it fixes a signing problem. So you can always see exactly what Xcode did and why. So as Joshua mentioned at the beginning, there are two new ways of working with signing in Xcode 8. The first one is automatic signing. Now automatic signing has been fully redesigned, and it takes care of all of your signing needs for you.

Chapter 4 : Appdome | Creating an iOS Distribution Certificate and P12 File for Signing iOS Apps

With App Center, your testers can get up and running with your app in a few clicks. Our Android Native Tester App aims to to get all types of testers using your apps and make the process of distributing a beta build familiar, helping you reduce churn.

For new testers, the process of getting ready to test an app can be a confusing and foreign experience. In addition to welcoming them to App Center, we guide them step-by-step through the device registration process for iOS apps. If your testers were confused, fear no more. The app invites and new release notifications for testers are now more contextual, with emails to clearly set expectations when an action, such as signing up for App Center, is required in order to start testing, or when no further action is required at the time. We hope that with these updates, your testers can get started much faster, and they will be more engaged when testing your apps. Updates to the Developer Portal Improved Insight into Tester Engagement With the new distribution statistics feature, developers will have more nuanced data right within the distribution dashboard to provide better visibility into how testers are interacting with your apps. View insights for your latest release over the last seven days, or see trends for all your releases over the last 90 days using version and time period filters. Mandatory App Updates We get it, issues happen and buggy apps get shipped. During the new release distribution flow, developers can mark the release as mandatory for testers to update to the latest version. When a release for an app has been marked as mandatory, the next time a tester opens their app, they will be prompted to update to the latest version before they can continue to use the app. Automatically Manage Devices with Auto-Provisioning In June, we released auto-provisioning capabilities in App Center, enabling iOS developers to spend less time managing device provisioning, and more time creating and shipping great apps. Now you can resend app invites and new release notifications to increase tester engagement. New Distribution Groups Page Sometimes small updates can make a huge difference. Build Information in the Install Portal Better decipher between your releases in the install portal with newly-included build information. For releases distributed using the App Center Build service, you now have access to the branch name, commit hash, and commit message in addition to the release notes. Our Android Native Tester App aims to to get all types of testers using your apps and make the process of distributing a beta build familiar, helping you reduce churn. You can get started with the HockeyApp for Android native tester app by downloading it directly from the Google Play Store today. Our native tester app for iOS is currently in private beta, and we look forward to sharing it with our customers soon! More Coming Soon Stay tuned in the coming weeks for additional new features. The team is currently hard at work bringing Azure Active Directory integration and support for sharing distribution groups across multiple apps in your organization. Try out all of the new Distribution features now!

Chapter 5 : Create In-House Distribution Provisioning Profile | Code | Telerik Platform

Distributing an app is arguably the Achilles's heel of the iOS app development process - and Apple makes us feel dumb at times. Provisioning profiles, certificates, device registrations, app ids, oh my! Apple enforces certain requirements to make sure they know who created which apps and control.

This may come with some difficulties as apps provided by 3rd party vendor rarely come with a detailed manual. This Knowledge Base article will guide you through all the steps in order to create the necessary credentials for signing iOS apps. We hope you find it useful and enjoy using Appdome! Signing iOS apps can be done using Appdome built-in signing capabilities or using your own mechanism outside of Appdome. This knowledge base article will provide an example using two different types of accounts: With this plan, there are two distribution methods: Devices do not need to be registered. Apple Developer Enterprise Program Account For companies and educational institutions that intend to distribute apps, they develop to employees within their organization. The Team ID is supplied by Apple and is unique to a specific development team, while the bundle ID search string is supplied by you to match either the bundle ID of a single app or a set of bundle IDs for a group of your apps. There are two types of App IDs: A P12 file contains the certificates Apple needs in order to build and publish apps. The certificate created in this example will work for an app that will available on the App Store or for an Ad Hoc deployment that will also work with an EMM Enterprise Mobility Management solution. Provisioning Profile When distributing different iOS apps, they are usually signed with the same distribution certificate. The entity that changes when signing different iOS app is the provisioning profile. A provisioning profile is a collection of digital entities that uniquely ties developers and devices to an authorized iPhone Development Team and enables a device to be used for testing. There are four types of provisioning profiles you can create for iOS devices. Developmentâ€” This type of provisioning profile must be used with a development certificate installed on each device on which you wish to run your application. It is used in the development cycle and allows developers to debug the application. It can only be installed on a set of pre-registered development devices and is not meant for any distribution scenario. App Storeâ€” This type of provisioning profile is matched to a specific distribution certificate. It is used to sign before submitting the application to the official iOS app store. After signing, the app will not install on any device and can only be used to upload to the app store. You can use this provisioning profile to distribute apps in a small organization where all devices are registered on the Apple site and are assigned to this provisioning profile, or for testing as part of the development cycle. Apps can be installed on any iOS device. In-House provisioning profiles are matched to specific distribution certificates. Follow these step-by-step instructions to create iOS signing credentials above: If you have an account, you can verify the account Entity Type from the Membership Details as shown in the following screenshot. Under App ID description, put your company name or other text to uniquely identify the app. Wildcard App ID is also possible; however, it comes with certain limitations on which App services will be available for the app. Under App Services, check the boxes next to the services the app uses. If you are using a third-party app or an app provided to you by a developer, you may not know which App Services entitlements to select. If you are not publishing on the Apple App Store, and you do not know the App services used, it is OK to select more options than what is actually being used by the app. When signing on the Appdome platform, the signing process will remove entitlements from the app if the provisioning profile does not have them included. The platform will provide a warning message for this. If the provisioning profile has entitlements that the app does not need, the signing process will continue without making any changes to the app. You will be presented with a preview of the App ID to be created. You now have an App ID. Next, we will configure your App ID to support push notifications, create a production distribution certificate and a provisioning profile. Push Notifications If your app uses push notifications, you must edit the App ID to enable push notifications. This will also require the creation of Push Notification SSL certificates as shown in the picture below. These steps will require a Mac. When done, click Continue To generate your certificate, you will need to import the certificate signing request. Email Address and Certificate Common Name. Back in your developer account on developer. Give the certificate a name with a. This needs

to be done so you can create a P12 file from Keychain Access. After expanding, you should see a private key under the distribution certificate. This P12 certificate will be used to sign apps on the Appdome platform. After saving, you will be prompted to enter a password to protect the P12 certificate file. Click OK after entering and verifying your password. Do not lose this password, it is required for future iOS signing. You now have a proper P12 certificate file. Next, we will create a provisioning profile to complete the set of signing credentials. Under Distribution choose the distribution type: Individual Developer account can create an Ad Hoc Distribution certificate for proof of concept for small organizations. Registering devices can be done one at a time or by importing a list of devices. Repeat for each the devices you wish to test with Register Multiple Device â€" In order to register multiple devices, you can retrieve the format for a multiple register upload file by clicking on the Download sample files. The following screenshot shows the file format for iPhones. If you have multiple distribution certificates, ensure you note the one you select here. You will need the P12 file containing the selected certificate as well as the private key to sign the app. In-House provisioning profile will not prompt you to select a device list. Enter a name for the profile: You now have a provisioning profile and are ready to sign your iOS apps. For information please read the knowledge base articles on:

Chapter 6 : Distributing iOS apps

The distribution provisioning profile consists of a name, a distribution certificate, and an app ID. The name is used only so that you can identify a provisioning profile. A provisioning profile is valid for one year.

Certificates and Provisioning Profiles. Apple will then verify the information, and create a certificate for you. If you are already done with this, you can skip to step 6. At the top of the window select Accounts. A dialog will appear. Add your Apple ID and your password, then select Sign in. A dialog will appear where you will see your code signing identities and the provisioning profiles. If you have not created them you will see a Create button next to them. Simply select it and Xcode will issue and download your code signing identities for you. Warning If you already have Code Signing Identities issued, you will see a Reset button next to them. Click on Continue and save the generated certSigningRequest file locally. On the next page you see the instructions for creating the certSigningRequest file. Upload the created certSigningRequest to the form and click continue. It will generate your code signing certificate for you. Download the certificate and double click to install it. Once installed it will be added to your Keychain Access app. Distribution Provisioning Profiles can include App Store profiles, that lets you distribute your app to the App Store and Ad-hoc profiles are good for distributing to your testers. An App ID is used to identify one or more of your apps. It can be an explicit App ID that only matches one unique bundle identifier or a wildcard App ID that can match multiple ones. Xcode will also create a Team Provisioning Profile for your project automatically, so you can start deploying to your device automatically. Also make sure that your app is connected to the correct Team. Note To setup a distribution Provisioning Profile, go ahead with the manual setup. Select any additional App Services that you need. For development select the correct project type under Development, or for distribution select the correct one under Distribution and click continue. Select the App ID you would like to use. Select the certificates you wish to include in the Provisioning Profile. These certificates will be able to build with this profile; click continue. Select all the devices you would like to use with this profile and click continue. Name your Provisioning Profile and click continue. Your profile is generated. You can download it to your device and double click to install it on your Mac. Click on the selected Provisioning Profile, this will expand the details. If its status is invalid, you can click on the Edit button and save again. Click on the Download button to download it and double click to install it on your Mac.

DOWNLOAD PDF PROVISIONING AND DISTRIBUTING YOUR APPS

Chapter 7 : In-House Distribution for racedaydvl.com Apps - Xamarin | Microsoft Docs

During development, you choose which devices can run your app and which app services your app can access. A provisioning profile is downloaded from your developer account and embedded in the app.

In-House Distribution for Xamarin. Proprietary apps can be distributed In-House previously called Enterprise through the Apple Developer Enterprise Program, which offers the following benefits: Your application does not need to be submitted for review by Apple. There are no limits to the amount of devices onto which you can deploy an application It is important to note that Apple makes it very clear that In-House applications are for internal use only. It is also important to note that the Enterprise Program: Does not provide access to iTunes Connect for distribution or testing including TestFlight. All apps still need to be signed by Apple. Testing your Application Testing your application is carried out by using Ad Hoc distribution. For more information about testing, follow the steps in the Ad-Hoc Distribution guide. Be aware that you can only test on up to a maximum of devices. Apple Developer Enterprise Program certificates will last for three years, and provisioning profiles will expire after one year. It is important to note that expired certificates cannot be renewed, and instead, you will have to replace the expired certificate with a new one, as detailed below. Under Certificates, select Production. Click Generate to create your certificate. Download the completed certificate and double-click on the file to install it. At this point, your certificate should be installed on the machine, but you may need to refresh your profiles, to ensure that they are visible in Xcode. Alternatively, it is possible to request a Certificate via the Preferences dialog in Xcode. To do this, follow the steps below: Select your team, and click View Details: Select App IDs under Identifiers. The App prefix should be already set as your Team ID, and cannot be changed. Click the Continue button and following the on screen instructions to create the new App ID. Once you have the required components needed for creating a Distribution Profile, follow the steps below to create it: Click the Continue button and select App ID from the dropdown list that you want to create a Distribution Profile for: Click the Continue button and select distribution certificate required to sign the application: Click the Continue button and enter a Name for the new Distribution Profile: Click the Generate button to create the new profile and finalize the process. Distributing your App In-House With the Apple Developer Enterprise Program, the licensee is the person responsible for distributing the application, and for adhering to the guidelines set by Apple. Your app can be distributed securely using a variety of different means, such as: Locally through iTunes An internal, secure web server Email To distribute your app in any of these ways you must first create an IPA file, as explained in the next section. This is a zip file that contains the application, along with additional metadata and icons. The IPA is used to add an application locally into iTunes so that it can be synced directly to a device that is included in the provisioning profile. Summary This article gave a brief overview of distributing Xamarin.

Chapter 8 : Renew Distribution Provisioning Profile and Dis |Apple Developer Forums

If your app uses App Services, you will need to create a provisioning profile as detailed in the device provisioning guide. Take a look at the Limitations section of this document for more information about limitations associated with free provisioning, and refer to the App distribution guides for more information about distributing iOS.

Chapter 9 : More Information on Generating and Distributing Mobile Apps - OutSystems

A Distribution Provisioning Profile combines your App ID and Distribution Certificate. It authorizes your app to use particular services (like Push Notifications) and ensures that your app is submitted by you.