In-House App Development Accelerator Guide

Overview

Turn your in-house app ideas into reality. Here's how.

Your business is unique, and so are your users. There's a world of possibilities to address their needs with innovative mobile apps that can change the way they work. If you're like most internal development teams, you also have constraints. Whether they're financial, organizational, or resource constraints, you need to focus on delivering just those apps that truly meet your business needs.

iOS gives you a great way to deliver those apps. iOS development can be fast and highly rewarding, enabling you to deliver apps that provide immediate benefits to your users. You won't have to apply onerous process and excessive resources to make a difference to your business, though it may mean focusing your efforts differently than you would with a typical in-house development project.

This guide will help you do just that. It explores best practices for making mobile apps great for your users, and it provides tips and resources to help you organize your project for maximum efficiency.

This guide is organized into the following four sections:

Planning—Defining your project, gathering requirements, and planning for the development process.

Design—Leveraging iOS interface design concepts to be sure your in-house apps delight users.

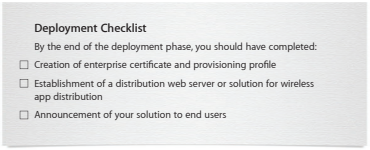
Development—Developing your in-house apps and getting the most out of tools from Apple, including the iOS SDK.

Deployment—Distributing in-house apps within the enterprise and establishing your own over-the-air app distribution service.

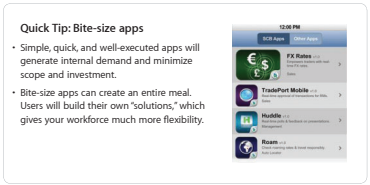
Let's get started.

Using this Guide

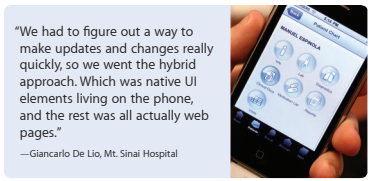
Checklist. Use the checklist in each chapter to review and track the most important steps in the development process.



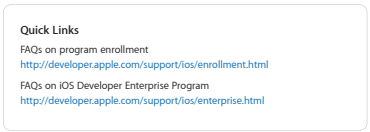
Quick Tips. Get important information essential to the process in an easily accessible resource.



Examples. Explore the customer examples for inspiration and ideas about how other businesses have built in-house apps for iPhone and iPad.



Quick Links. Look for these quick reference links to learn even more about iOS in-house app development.



Planning

Creating a great app requires a great plan. It's important that your internal stakeholders feel connected to the project's objective and that they actively participate when formulating the plan. The more your team understands the balance of work throughout the process and the steps they need to take to execute the plan, the more effectively they can create something remarkable.

As part of that plan, you and your team should explore the following:

- **Discover business and user requirements.** Be sure your solution addresses true business needs. Does it save time and/or money, make the workforce more productive, or otherwise address a requirement for moving the business forward?
- **Evaluate existing resources/infrastructure.** You can maximize valuable resources by repurposing something you've already built and leverage it for the mobile environment.
- **Explore ease of implementation.** Harvest the low-hanging fruit first: Look for projects with rapid time-to-return, where it's faster and easier to demonstrate positive results from your efforts.

Keeping these planning concepts in mind will help develop a focus for your project. This chapter will explore steps you can take to get your project organized and off to a fast start.

Get User Input

Put yourself in your users' shoes. Spend some time in their workspace by attending a few meetings or going out to a job site. You'll get invaluable insight into their work habits, bottlenecks in the workflow, and employee or customer pain points. Invite particularly insightful or passionate users to join your project team to provide ongoing input.

Identify which problem a mobile app could solve that would deliver the most value to your users and your business, in the shortest amount of time.

Don't try to bite off too much when you start developing mobile apps. Narrow down the ideas for what you *could* build to just what you *should* build.

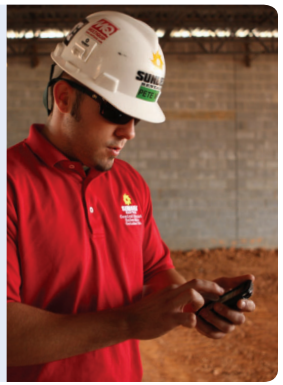
Planning Checklist

Refer to this checklist throughout the planning phase of your project. By the end of the planning phase, you should have a scope of work that includes:

- ☐ Inventory of all potential business needs/requirements
- ☐ Inventory of user needs (to align with business requirements)
- ☐ IT infrastructure requirements to support project effort
- ☐ Application definition statement describing the app and purpose of the solution
- ☐ General project timeline and milestones
- ☐ Identified team of stakeholders with roles and responsibilities defined
- ☐ Enrollment process started for the iOS Developer Enterprise Program

"We had direct communication from developer to sales force. We even went out on sales calls, we put on our steel tips and hard hats, we drove out and went on sales calls to see exactly what they needed and how they were going to use the app."

—Dean Moore, Sunbelt Rentals



Look for a few simple tasks that the majority of your users do frequently and think about how a mobile app could make those tasks easier. Here are a few examples:

- A simple app that allows an employee to approve an expense report or purchase order on the go
- A corporate directory or campus map that everyone can use
- A meeting room finder that employees can easily use while not at their desks
- A simple time-tracking app that might tie into to your back-end time management or billing system

Inventory Your Assets

Often the best way to build an in-house app efficiently is to leverage existing technology. You might want to do an inventory of your employee websites and determine if it makes sense to optimize them for iPhone or iPad. Or you might have back-end systems with data in a form you can easily deliver in a mobile context. Also, don't forget about apps already in the App Store: An app that meets your needs may already be available. Review the Asset Inventory example (to the right) for additional questions that can help you survey your existing environment.

Define Your App

Once you have a solid understanding of what your users need, as well as the potential mobile solutions to meet those needs, you'll want to refine those concepts into a concise project plan to share with your project stakeholders.

The most important element in defining your project plan is the application definition statement—a concise definition of your app's purpose. An application definition statement can help you avoid two common pitfalls:

- You have an existing desktop app that you want to move to the mobile space and therefore a long list of features to bring to the new environment.
- You have a great idea for a new mobile app, but you immediately jump to features before honing in on the core purpose of the app.

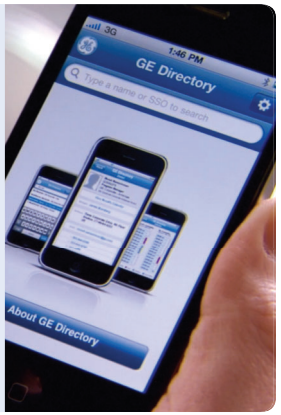
Example: Asset Inventory

Answering these questions can help you determine if you can reuse existing technology in your mobile app:

- What systems do the most mobile part of your workforce use every day?
- What do your mobile workers need to do the most?
- What manual processes could be automated or simplified by mobile apps?
- Do you have existing nonmobile systems that could become useful for mobile workers?
- Which functions within those systems are used most frequently?
- What kind of data access do your enterprise systems provide? Is data easily accessed through web services?
- Do you have internal websites that your employees access every day? Could these easily become mobile apps?

"We just went across the different product lines and said, 'Where does mobile truly make sense? What's the top tier? What are the first hits that we should go after?' And we went from there. We went across our product portfolio and figured out what made most sense."

—James Blomberg, General Electric



Creating an Application Definition Statement

Start by writing an application definition statement that includes the following:

- The purpose behind your app
- Who it's for and how they'll use it
- Its core functionality

Be sure your statement defines a solution and only its core functionality—not a detailed set of features. You should have a strong purpose statement that you use to filter every idea for a feature. Ask yourself if each feature serves the intended purpose. Then choose the fewest, most frequently used, and most appropriate features for a mobile context. You don't want to end up with a long, unfocused list of features that are either difficult to execute or don't solve the problem. Keeping your app focused will give your users the greatest productivity in a simple-to-use package.

Plan for the Development Process

Typical enterprise software development projects absorb huge resources during the development phase. Using the iOS SDK and high-level Cocoa Touch frameworks, your development teams can spend less time coding and more time designing the ideal user experience. Not only does this process enable you to deliver an app for your employees more quickly, but it also helps you provide solutions that exceed your users' expectations.

Whether you use an agile development process or a more traditional waterfall method, make sure you budget time and resources to invest in the design process as a central and ongoing part of your app development effort.

Establish a rough timeline of the process you envision and the roles for each stakeholder at every step along the way. This doesn't need to be etched in stone, but it can provide a common reference point for everyone involved.

Example: Application Definition Statement

Here's what an application definition statement might look like, using a time tracking app for lawyers as an example:

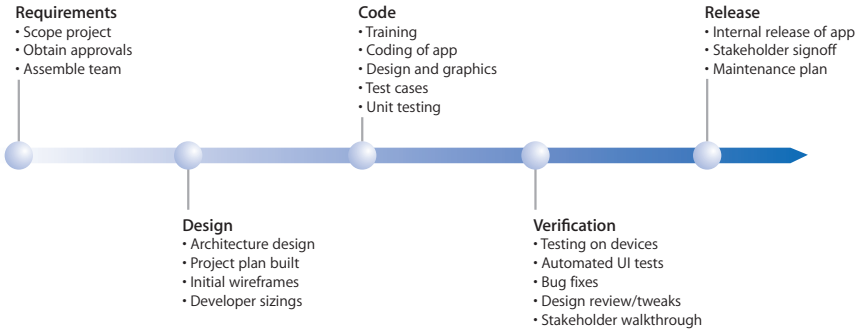
- Purpose: Track time spent and billable hours for each client case
- Who it's for: Lawyers in the firm who need to track billable hours
- How they'll use it: At every client meeting to start and stop billable time
- Core functionality: Track and report time spent to the CRM system
- Consolidated definition statement: App for lawyers to track time and billable hours for each client
- Features that fit the definition: A start/stop watch; background tracking/processing; server integration with CRM system; client record lookup for associating time tracked with client/case; online/offline syncing based on network connectivity
- Example of features that don't fit the definition or exceed the project scope: Alerts for new cases in trial; document lookup for legal reference; map of client locations; patent lookup interface

Quick Tip: Crowdsourcing

Genentech knew that great app ideas can come from anywhere and anyone, so they created a crowdsourcing model that takes employee suggestions for apps they'd like to see developed internally. They have since created the top five requested apps to extraordinary user satisfaction and adoption.

Creating a Scope of Work

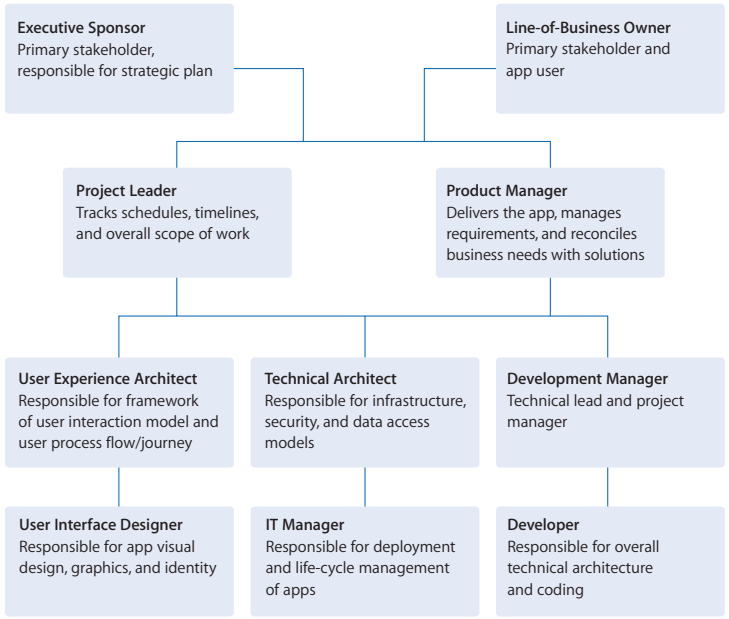
To help you stay focused and communicate the process, your plan should include a central scope-of-work document that includes all project resources, goals, objectives, timelines, and key deliverables. This is an important guide to the project for all stakeholders. The scope of work can incorporate preliminary technical requirements for your app, as well as flowcharts or visual diagrams to help communicate the intent of the app concept during the development phase.



Assemble Your Team

As with any project, you'll want to assemble a team of contributors who each share a stake in the success and outcome of your app project. Some participants may be your internal customers (a line-of-business owner or user group), and others will be tasked with owning specific parts of the development process itself (designers, developers, technical architects, and so on). Ultimately, you want to align the team roles and responsibilities with the project timeline and milestones discussed in the prior step. For example, because design is a central element of any iOS development project, you'll want to make sure you have a design team (or resources to match). Different groups may have different points of participation and interest in the outcome, so it's good to document those roles so that everyone can stay abreast of responsibilities along the way.

Example: Typical Project Team

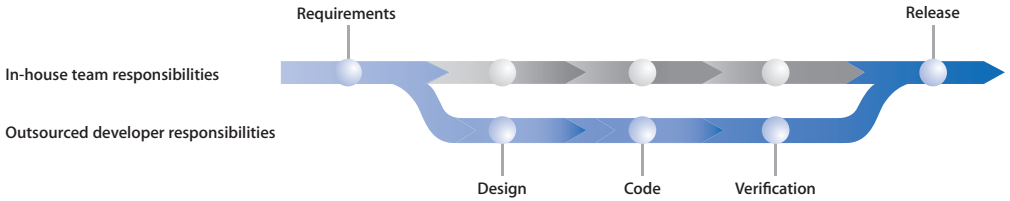


Outsourcing Development

If you don't have sufficient in-house resources, consider outsourcing all or part of the development work. Outsourced developers can also present you with a portfolio of their work that could spark new ideas.

Of course, to be successful, the outsourced team needs a thorough understanding of your project—everything you've determined during the planning process—and regular interaction with you and your in-house team. Discuss your needs and make sure they understand what your objectives are. Review the application definition statement and carefully review your project details. And be sure right from the start that you've established clear, two-way communication and a process for keeping in touch.

You need to define the role your outsourced provider will play, just as you would a member of your internal team. Align their roles and responsibilities to the project plan and timeline so that you can communicate clearly about which aspects of your project they'll deliver.



Some outsourced partners can help you through all elements of the project, from initial requirements to final deployment. Others may focus only on writing code. It's good to explore these capabilities and services with your outsourced provider, whether or not you ultimately contract with them. It can help you evaluate their strengths and also inform how you shape the relationship.

Quick Tip: Selecting an Outside Vendor

- Meet multiple vendors.
- Review existing work, including apps on the App Store; note app ranking and user comments.
- Evaluate skills and capabilities, for example, is all coding done in house?
- Ask for references.
- Disregard one-size-fits-all ethic or generic multiplatform approach.
- Focus on UI design, high-quality art, and the app "journey."
- Discuss maintenance and life cycle of app beyond version 1.0.
- Ask about IT infrastructure experience.

"We use outside contractors for several different pieces. And it depends really on the need of the app. If there's a very specific look and feel, we'll go talk with contractors who have made things that are similar because we know that they have expertise in that already. In turn, we can give a higher-quality product to our staff or to our customers."

—Todd Schofield, Standard Chartered Bank

Getting Started with the iOS Developer Enterprise Program

Once you've gathered requirements, obtained input from your users, and defined your application and project plan, the final step before proceeding to the design phase is to enroll in the iOS Developer Enterprise Program. This program offers a complete and integrated process for developing, testing, and distributing iOS apps to employees within your organization. Once you're enrolled in the program, you'll be able to access the tools and resources noted in the list to the right.

Here is an overview to help you understand the enrollment process and then get started. For more details, visit http://developer.apple.com/programs/ios/enterprise.

Enrollment Requirements

Before applying to the program, ensure the following:

- You plan to distribute iOS apps only within your company or organization. The iOS Developer Enterprise Program is intended for developers who wish to develop and distribute their iOS apps to employees within their company or organization.
- Your company has a Dun & Bradstreet Number (D-U-N-S). You'll need to provide it to Apple during the enrollment process. You'll also need to know the legal name of your company or organization. To request or obtain a D-U-N-S number, visit https://eupdate.dnb.com/requestoptions.asp.
- You have authority to bind your company to the legal agreements. During enrollment you will need to provide a legal contact who can verify that you have the authority to bind your company to the iOS Developer Program Enterprise License Agreement.
- You have the technical capability to sign applications in Xcode. As the enrollee you will be your team's "Agent," which makes you responsible for app provisioning and technical account administration tasks.

Enrollment Process Overview

1. Register as an Apple Developer. To start your enrollment, you'll need to register with Apple by creating a new, dedicated Apple ID for this program. It's helpful to set up an email address specifically for this account so that your organization can assign it to different individuals if necessary.

2. Enter company, contact, and legal information. This information is required to validate your status as a business entity. The key requirement is a valid D-U-N-S number. Make sure your company name and address information matches the information listed in the Dun & Bradstreet database. As part of the identity verification process, you may need to provide Apple with business documents, such as articles of incorporation, an operating agreement, and a business license.

Overview: iOS Developer Enterprise Program Resources

With membership in the iOS Developer Enterprise Program, you receive the following benefits:

- Access to the iOS Dev Center
- Access to the iOS SDK
- Select prerelease software and tools
- Ability to set up your development team in the Team section of the Member Center
- Access to Apple Developer Forums
- Technical support incidents (two per membership program year)
- Ability to test applications directly on iPad, iPhone, and iPod touch
- Ad hoc distribution of your app on up to 100 registered devices
- Enterprise in-house distribution to an unlimited number of employee devices

Quick Tip: Assemble Your Development Toolkit

The basic requirement for the iOS SDK is an Intel-based Mac. Developers typically choose a MacBook Air or a MacBook Pro for the portability and freedom they provide. But an iMac or a Mac mini are equally good choices, particularly if you have an in-house development lab. Also, make sure you have test devices available. If you want to ensure complete compatibility, be sure to have prior-generation hardware, such as an iPhone 3GS or first-generation iPad.

3. Submit to Apple. Once you've submitted your enrollment, you can check the status by logging in to the developer Member Center http://developer.apple.com/membercenter. Authenticate with the Apple ID you created in step 1. As part of this step, Apple will review the app and contact you or your legal team as necessary.

4. Agree to the Enterprise Program License Agreement. To proceed with your enrollment, you'll need to agree to the terms of the program license. You can review these terms and share them with your legal team at this time.

5. Purchase the program. Once you've agreed to the terms, you'll receive instructions on how to purchase the membership through the Apple Online Store. If you'd like to use a purchase order, contact your local Apple Store to see if institutional procurement options are available.

6. Activate your membership. You'll receive an order acknowledgment after you've purchased the program. Within 24 hours, you should receive an activation email from Apple that includes a code for activating your membership. Once you've activated, you can access all the program resources.

Setting Up Your Team

Once your company is enrolled in the iOS Developer Enterprise Program, you need to set up your development team in the Team section of the Member Center.

Team Roles and Responsibilities

A development team consists of individuals with the following roles:

Agent. The primary contact for the development team responsible for accepting all iOS Developer Program agreements; also the primary user who enrolled in the program. Responsible for managing the enterprise distribution certificate used to provision apps for broad-based deployment to employees.

Admin(s). Admins manage their own development teams and development certificates. Require the Agent's involvement to manage enterprise distribution.

Members. Primary developers within the organization. Members receive approval from Admins to provision apps and devices for testing and development purposes. Require the Agent's involvement to manage enterprise distribution.

Overview: Team Setup

Agents and Admins can add new Members, who can have either an Admin or Member role, by navigating to the People tab in the Apple Developer Member Center. Navigate to the Invitations section and click the Invite Person button to invite new Members to join your team.

Quick Tip: Registering Devices for Development

Admins can enter multiple device IDs at once by uploading a .deviceids file generated by the iPhone Configuration Utility. Within the iPhone Configuration Utility, select the devices you wish to upload and click the Export button. This will create a .deviceids file.

Visit www.apple.com/support/iphone/enterprise to download the iPhone Configuration Utility.

Learning Resources

Once you've set up your team, visit the iOS Dev Center at http://developer.apple.com/devcenter/ios where you'll find a wealth of resources. You can bookmark them or even make them your home page for all things development. Here are just some of the great resources available.

Forums
Connect with other enterprise developers and share ideas and best practices. It's always helpful to have a community of like-minded developers at your fingertips.

Reference Library
An encyclopedia, text book, and syllabus all wrapped into one. It's searchable, categorized, and gives you all the direct information you need to build apps.

Sample Code
Use the samples to inspire development of your own great apps. You can even copy and paste the sample code right into your project.

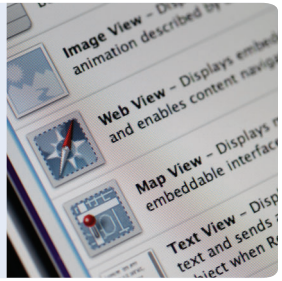
Getting Started Guides
If you're new to iOS development, these guides provide your team with fundamental concepts and best practices for iOS development.

Looking Ahead

When your planning process is complete, refer back to the planning checklist at the beginning of this chapter. If you've completed each step, chances are you'll have executive support, commitments from your team, and a clear project plan that everyone can follow. Your team will be positioned for the next phase of process: Exploring design and development best practices and establishing a basic understanding of iOS development concepts.

"The samples on Apple.com really do make it simpler to see a specific feature, like drilldown or maps, or integrating with a local SQL database."

— Keith Debickes, JM Family Enterprises, Inc.



Quick Links

FAQs on program enrollment
http://developer.apple.com/support/ios/enrollment.html

FAQs on iOS Developer Enterprise Program
http://developer.apple.com/support/ios/enterprise.html

Design

Design matters. Creating a basic iPhone or iPad app is easy. However, highly successful apps take a little more effort. What makes the most successful apps appealing? They have an attractive design and make excellent use of colors and audio, they're simple to use and work as expected, they keep the user involved, and they keep the user coming back again and again. By paying close attention to design when you build a new app or enhance an existing app, you can increase its appeal, create a more engaging user experience, and make your app delightful to use.

This chapter describes some strategies you can use to refine your idea, review your design options, and determine an app design that will make your users more productive.

Design for Touch

Designing a user interface for interaction with the mouse is very different than designing for touch. As you begin designing an app, you'll want to understand what makes iOS devices unique. Spend some time with an iPhone or an iPad and get familiar with the user interaction and interface design conventions.

At a basic level, for touch interaction you need more pixels to represent a selectable button for a finger than you would for a mouse in the desktop environment. For example, the comfortable minimum size of tappable UI elements is 44 x 44 points. Elements like pull-down menus or scroll bars that are common on the desktop don't work well on a mobile device that's designed for touch.

Read the Human Interface Guidelines

The iOS Human Interface Guidelines describe the principles that help you design a superlative user interface and user experience for your iOS app. These principles are just as important for enterprise in-house apps as they are for apps built for the App Store.

Design Checklist

By the end of the design phase, you should have accomplished the following:

- ☐ Read the iOS Human Interface Guidelines from Apple
- ☐ Established a concise feature list that's directly aligned to your core application definition statement
- ☐ Prioritized a list of objects, tasks, and concepts and how they relate to one another
- ☐ Created a baseline set of wireframes and rough compositions to visualize the app journey

"One of the ways we ensure consistency in our apps is that we follow the Apple HIG, the Human Interface Guidelines. It really helps make sure that we have consistency app to app. There's still lots of different design styles we can choose from, and we also make sure that those have a consistent theme running through them. But following the HIG is very important for us."

—Todd Schofield, Standard Chartered Bank

Simplify

Many times your enterprise in-house apps will be derived from an existing desktop application environment or based on line-of-business systems that your users depend on. It's easy to fall into the trap of trying to bring every feature and function from the desktop application down to the mobile device. This approach usually fails to deliver the type of experience expected on a mobile device. Remember that users accomplish tasks differently on mobile devices and that certain tasks might not be at all practical for a mobile device. Small, bite-size tasks are better suited to mobile development, which is why it's important to continually filter features through the application definition statement as you refine your app.

Here are a few questions you might ask yourself about the user interface elements in your app to help simplify the design:

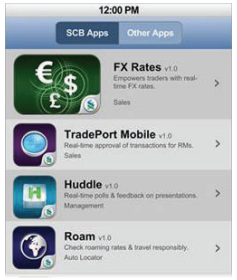
- Does it make sense for the element to be onscreen?
- Does the element provide access to critical functionality?
- Is it frequently used? Almost always?
- Does the user need the element each time a selection is made?
- Given the flow of the app, is it important to display the element now?

If the answer to any of those questions is no, perhaps you can do without the element. Or you may want to consider combining the functionality with something else.

iPhone and iPad users are accustomed to the appearance and behavior of the built-in apps that ship with those devices. You don't want to mimic every detail of the built-in apps, but it's helpful to understand the design patterns they follow and consider how to apply those patterns to your own apps in a simple, functional, and easy to use design. Investigate these apps for common controls, touch events (such as pinch and zoom), and animations, and start to think about how you might apply those concepts to your own app consistently.

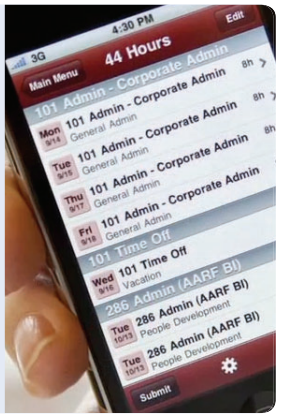
Quick Tip: Bite-size apps

- Simple, quick, and well-executed apps will generate internal demand and minimize scope and investment.
- Bite-size apps can create an entire meal. Users will build their own "solutions," which gives your workforce much more flexibility.



"We have an overall philosophy that internal apps should be just as elegant and beautiful as the best commercial app. So when we started looking at designing the UI, we didn't want to just solve the problem functionally, we wanted to solve it…in a really clean way."

—Mark McWilliams, Razorfish

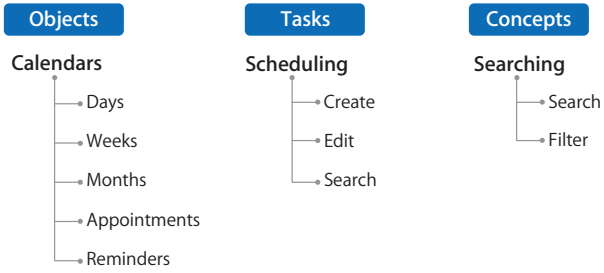


Prioritize

When an iOS app establishes and maintains focus on its primary task, it's satisfying and enjoyable to use. Each part of your app should be fine-tuned for its purpose. Creating a list of objects, tasks, and concepts—and then sorting them based on their relevance to your apps primary purpose or task—will help you deliver an organized and focused user interface. This step will also help you think about the workflow or process of your app interaction, which will inform your user interface design decisions.

Inventory Objects, Tasks, and Concepts

- **Objects.** These are the primary functional elements of your app. For example in a calendar app, they would be things like days, months, appointments, and reminders.
- **Tasks.** These are actions that are typically performed on objects, for example, filtering, scheduling, editing, and creating.
- **Concepts.** These are workflows or in some cases, a series of related tasks that form a larger concept. Using our calendar example, a concept might be searching, which would involve multiple tasks.



Once you've created these lists, you'll start to notice some relationships between the items within each category. This will help you group related objects, tasks, and concepts in a hierarchy that should simplify how they present to the user.

Quick Tip: Retina Display

The Retina display on iPhone 4 allows you to display high-resolution versions of your art and icons. If you scale up your existing artwork, you miss out on the opportunity to provide the beautiful, captivating images users expect. Instead, rework your existing image resources to create large, higher quality versions that are richer in texture, more detailed, and more realistic.

Think Top Down

Put the most frequently used (usually higher-level) elements near the top of the screen, where they're most visible and easy to reach. As the user scans the screen from top to bottom, the elements should display progressively according to the following criteria:

- **Frequency of use:** Most frequently used elements should be higher; less frequently used, lower.
- **Importance to the user:** More important elements should be higher.
- **Visual emphasis:** Elements you want to appear more prominently in your design should be higher.

The same approach holds true for the information in your app. It should progress down the screen from more general material at the top to more specific at the bottom.

Optimize

Good design is an iterative process. The more you exercise your interface design concepts early on in the process (before you write any code), the better the end results will be.

It's also important to optimize your design for your target audience and the target device. Great apps compensate for the user interaction concepts that will vary with the device's unique characteristics. Optimizing your app is all about refining and iterating on these concepts so that the end result will delight your users.

Iterate

Before you can begin to successfully build an app, you need a solid set of blueprints. You might start off with some rough sketches and then refine your ideas over time. With each turn you'll discover more about how your user might interact with the app and new ideas that you could incorporate—all without spending any time, money, or resources on actual development. Consider drawing or sketching out your entire app flow, beginning to end, to get a complete feel of the user experience as well as the functionality your design will create.

Quick Tip: Sketch Your App

Iterate on paper. Often the best way to articulate your design vision is to create rough sketches early in the design process that help you shape and refine your design without the cost of code development. You can buy handy templates online to help you crank out rapid sketches with some polish.

Iterate with an app. There are also apps in the App Store, such as iMockup and App Layout, that help you create user interface mockups for iOS using standard controls and views.

iPad vs. iPhone

If you're planning to develop an app that runs on both iPhone, and iPad, you need to adapt your design to each device. While most individual UI elements are available on all devices, the overall layout usually differs dramatically. For example, users tend to expect more high-fidelity artwork in iPad apps than they do in iPhone apps. Merely scaling up an iPhone app to fill the iPad screen is not recommended. Instead, you need to make your iPad app engage the user in ways that take full advantage of its larger screen and capabilities. Also keep in mind that iPhone 4 supports higher-resolution graphics via the Retina display, which requires doubling the artwork resolution. There are also differences in the available gestures and the ways that rotation is handled. The devices also support different UI elements. For example, popover controllers or split view controllers are unique to iPad.

Universal Apps

The iOS SDK supports the development of Universal applications. A Universal app is optimized to run on all iOS devices. It's essentially an iPhone app and an iPad app built as a single binary.

A Universal app can determine which device it's running on and provide the best experience for that device. Well-designed Universal apps leverage the device's unique hardware features, provide the right choice of user interface elements, and use only the functionality that's supported by that device.

When designing a Universal app for iOS, it's important to think about how to separate user interaction from the underlying application code. The iOS SDK's classes and APIs leverage a model-view-controller (MVC) paradigm that encourages a clean separation of your app data and logic from the views that are used to present that data. For example, building your UI using Interface Builder gives your project this type of flexibility (see next chapter).

The first step in making a Universal app is to create user interface designs for each of the form factors—one design for iPad devices and another for iPhone/iPod touch devices. Much of your design will be affected by the features you want to expose in each of the different form factors. Think about how users might use orientation or gestures differently. Consider each device's hardware capabilities, such as the camera. Differences in how your users use the device should inform how you approach a consistent design for each and where you might need conditional coding.

> "iPad definitely gave us more real estate, which we wanted to take advantage of. That was key for us. It was not, let's just make everything three times as big, have that many more pixels, but let's really make sure we're properly using that space…If we're going to translate one from an iPhone to an iPad, we re-think it. Probably 60 percent of the core functionality remains, but what else can we do? How can we make it more usable, how can we have less clicks, or less screens to get to everything?"
>
> —James Blomberg, General Electric

Accessibility

It's important in a business environment to provide equal access to mobile tools and technology to all users. iOS includes several features out of the box that make a device accessible and easy for everyone to use. However, its also important to optimize your in-house apps for accessibility so that users with visual, auditory, and physical disabilities can use and enjoy your app.

iOS includes the UI Accessibility programming interface, a lightweight API that helps an app provide all the information VoiceOver needs to describe the interface so that visually impaired people can use the app. The UI Accessibility programming interface allows you to add a thin layer of functionality that doesn't alter your app's appearance or interfere with its main logic. This means that when you use standard controls and views, much of the work of making your app accessible is done for you. Depending on the level of customization in your app, making it accessible can be as simple as providing accurate and helpful descriptions of your accessible user interface elements.

The iOS SDK also provides these tools to help you make your app accessible:

- An Interface Builder inspector window that makes it easy to furnish descriptive accessibility information while designing nib files
- Accessibility Inspector, which displays the accessibility information embedded in your app's user interface and allows you to verify this information when you run your app in iOS Simulator

In addition, you can use VoiceOver itself to test the your app's accessibility.

Looking Ahead

With a foundation of UI design best practices, you'll be ready to move into the development phase of your project. However, as you might recall from the planning chapter, design is an iterative process that continues throughout the app development life cycle. Executing good design with a focus on user experience should be a conscious strategy even when your development team starts cranking out code. The toolset and concepts discussed in the next chapter will help you do just that—develop rock-solid code, and at the same time enable you to deliver well-designed apps for your users.

Quick Tip: Building in VoiceOver Support

Making your iOS app accessible to VoiceOver users is the right thing to do. It might also help you address accessibility guidelines created by various governing bodies.

To make sure VoiceOver users can use your app, you don't need to change the visual design of your interface in any way. When you use standard elements, you have little (if any) additional work to do.

However, you might need to supply some descriptive information about the views and controls in your user interface. The more custom your user interface is, the more custom information you need to provide so that VoiceOver can accurately describe your app.

Development

With iOS you can deliver content and information in simple, yet powerful new ways to help your employees be more productive. Leveraging the iOS SDK, your development team will be building apps using the same tools Apple engineers use to build the OS and the apps that ship with every iPhone and iPad. This enables you to create apps that look, feel, and respond to your users elegantly and with maximum efficiency.

Using iOS tools you can leverage high-level frameworks that help you take full advantage of the platform. We'll explore the essential APIs for in-house development to give you ideas for integrating these capabilities into your apps. We'll also discuss how you can leverage web technologies by using HTML5, CSS, and JavaScript. With the right security features built into your solution, these technologies work together to create a powerful, secure foundation that supports your business needs.

Finally, before you can consider your app ready for users, you'll want to thoroughly test and debug it. You'll also want to validate its performance on different devices. We'll discuss how iOS tools help you perform those tasks and suggest best practices for making sure your app works the way you intend it to.

Native Development

When it comes to in-house development, it's all about executing the vision of your app design, taking full advantage of the device's capabilities, and doing so in a organized and efficient environment. The iOS SDK includes Xcode, the IDE for coding, building, and debugging your app; Interface Builder for creating the UI; Instruments to analyze behavior and performance; and dozens of additional tools.

Xcode is the hub of your development experience. Xcode provides code completion, real-time static analysis, and instant on-device debugging.

Interface Builder makes it simple to prototype your app. Drag elements to create a full user interface without writing any code. With Xcode 4, Interface Builder is built right into the Xcode IDE.

Instruments collects and displays data such as disk, memory, or CPU usage in real time, making it easy to pinpoint problem areas.

The Simulator runs your app in much the same way an iOS device would, so you can verify and test your code right from your desktop environment.

Development Checklist

By the end of the development phase, you should have a basic understanding of:

- ☐ iOS SDK tools, including Xcode, Interface Builder, Instruments, and Simulator
- ☐ The key APIs and frameworks for enterprise in-house development
- ☐ Web app development
- ☐ iOS architecture for accessing data in back-office systems
- ☐ Best practices for securing your in-house apps
- ☐ Testing, debugging, and performance validation of your app

"It is my team's experience that SDK and Xcode are fantastic programming tools and very easy to use, even for developers not coming from a Mac background."

—Hans-Christian Pahlig, Axel Springer

Essential APIs for In-House Developers

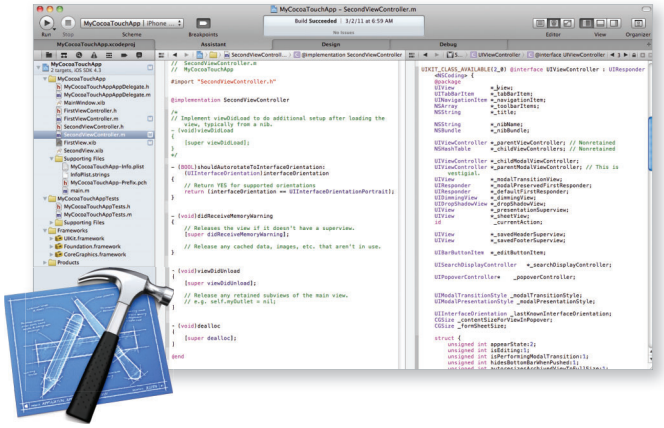
The iOS SDK provides tools that help you write almost any app functionality you can imagine. Many of these tools also include sample code and resources to help you get started quickly. Let's review a few of the thousands of APIs available in the iOS SDK. Just a quick look may fuel your imagination when you see the amazing range of capabilities you can easily build into your in-house apps.

Multitasking. Developers have access to seven multitasking services that enable tasks to be performed in the background while preserving performance and battery life. These include functions such as Voice over IP, background audio, background location services, push and local notifications, task finishing, and fast app switching.

Push notification. The Apple Push Notification Service provides a way to alert your users of new information, even when your app isn't running. Send text notifications, trigger audible alerts, or add a numbered badge to your app icon.

Accessories. Applications can communicate with accessories either through the 30-pin dock connector or wirelessly using Bluetooth. Build an app that retrieves data from external sensors or even control accessories with a sophisticated Multi-Touch interface. Create an inventory app for your barcode reader. Or build an app that logs and tracks the readings from an attached heart rate monitor. You can also create your own custom protocols to exchange data and commands with your app. To find out how you can add support for iOS apps in your accessory, learn about the MFi licensing program at http://developer.apple.com/programs/mfi.

Location-based services. Use the Core Location framework to determine the current latitude and longitude of a device and to configure and schedule the delivery of location-related events. The framework uses available hardware to triangulate a user's position based on nearby signal information. iOS 4 brings enhancements to mapping via the MapKit API. MapKit provides support for panning and zooming, custom annotations, showing current location, and even geocoding to highlight regions of the map and display additional information.

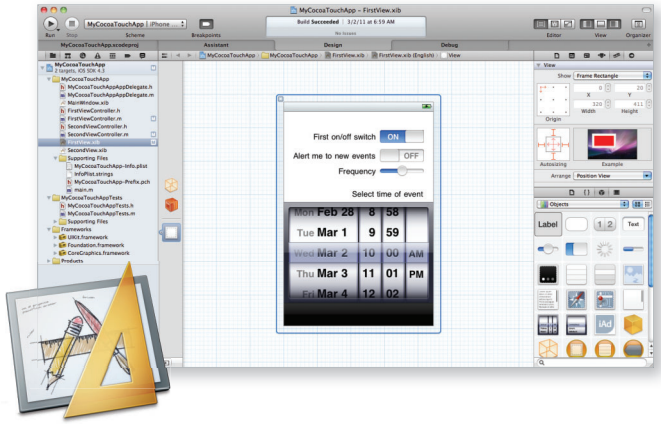


Xcode is the hub of your development experience, providing code completion, real-time static analysis, and instant on-device debugging.

Integrating shared data. iOS provides powerful connectivity options for sharing information between apps. Using a URL-based syntax, you can access data from the web, as well as initiate actions in other installed apps, such as Mail, Calendar, Contacts, and more. Your own app can also declare a unique URL scheme, allowing any application to launch your app.

- **Mail.** iOS lets you present a standard Mail or SMS composition interface from within your app. In both cases, you can programmatically preconfigure the message with recipients and content, which the user can then edit before sending. Outgoing mail and SMS messages are automatically handled by the system's Mail and Messages queues.
- **Contacts.** With Address Book APIs for shared data, your app can create a new contact or get existing contact info. By accessing the built-in contact list, your app can enable a user to associate a contact or business address with an application task or process.
- **Calendars.** Event Kit allows iOS apps to access event information from a user's Calendar database. Fetch events based on a date range or a unique identifier, receive notifications when event records change, and allow users to create and edit events for any of their calendars. Changes made to events in a user's Calendar database with Event Kit are automatically synced with the appropriate calendar, including business calendars hosted on CalDAV and Exchange servers.
- **Photos.** UIKit provides access to the user's photo library. The photo picker interface provides controls for navigating the user's photo library and selecting an image to return to your app. You also have the option of enabling user editing controls, which let the user pan and crop the returned image. It can also be used to provide an interface to the camera, so photos taken can be loaded directly into your app.

Audio and video. Multimedia technologies in the iOS SDK let you implement sophisticated audio and video capabilities within your app. The Media Player framework supports full-screen playback of video files, and built-in support for HTTP live streaming makes it easy to use standard web servers to stream high-quality audio and video content over-the-air. Your app can also take advantage of Core Audio to generate, record, mix, process, and play audio in your app. Use Core Animation to add smooth motion and dynamic feedback to the user interface. Or leverage OpenGL ES for high-performance 2D and 3D graphics.



Interface Builder, which is integrated directly into the Xcode 4 IDE, makes it simple to prototype your app. Drag elements to create a full user interface without writing any code.

Web Development

Web apps—an entirely new category of mobile applications—are opening up a world of possibilities for enterprise. Web apps are custom-designed web pages that take advantage of advanced HTML, CSS, and JavaScript to deliver an immersive app experience to iOS users. And because you build apps using HTML, you can develop web apps in any web development environment. You just need to host a web page to deploy a web app, and you can manage changes or updates from the server where the page resides. Let's take a quick look at the technologies you use to build web apps for iOS devices.

HTML5 is the latest specification of HTML, the primary standard that determines how web content interacts with modern browsers. HTML5 allows developers to integrate rich media directly into standard web pages, reducing development time and providing rich interactivity for the creation of web apps.

CSS3 animations and visual effects allow you to create sophisticated graphical user interfaces for web apps. The visual effects available range from gradients, masks, and reflections, to more complex 2D and 3D effects. When you combine these visual effects with touch events, you can create web apps that interact much like native apps on iPhone and iPad.

WebKit is an open source web browser engine. Powering Safari on iPhone and iPad, WebKit simplifies web development and accelerates innovation. An open source toolset, WebKit is free for anyone to use, and provides the HTML engine for web apps on iPhone and iPad.

Dashcode is included in the iOS SDK. Its integrated environment allows you to lay out, code, and even test web apps. Dashcode also provides handy templates to help you bring your web app to life.

Safari 5 for Mac and Windows includes a powerful set of tools that make it easy to debug, tweak, and optimize web apps for peak performance and compatibility. To access them, turn on the Develop menu in Safari preferences on your Mac or PC.

The Simulator runs your app in much the same way as an actual iOS device so that you can verify and test your code right from your desktop environment. And because the Simulator includes the Safari browser for iOS, you can also test and verify your web apps prior to deployment.

"One of the things that we're able to do in the SDK, is to go in and actually make quick changes. It's something that can be done very quickly onscreen, and then using the Simulator, you can see it almost instantly. And for us, that is able to provide direct feedback."

—Todd Schofield, Standard Chartered Bank

Integrating Web Content into Native Apps

With the iOS SDK, you can also provide access to web content within an iOS app using an element called Web View. This allows your web applications to access iOS features such as push notifications, the built-in camera, 3D motion awareness, and more. As a benefit, you can deploy new enhancements to your app with adjustments to your web server without having to redeploy the app itself. Leveraging web content in your native app is also a great way to ramp up on native development without scaling back or discarding your existing web development investments.

Accessing Back-Office Systems

In many cases, your enterprise app will need to tap into existing back-office systems and data warehouses. While delivering a great user experience on the client is a top priority for any successful mobile app, the same attention to detail and architecture is required to integrate the client experience with data from back-end servers. The iOS SDK has a powerful collection of tools and frameworks for storing, accessing, and sharing data resident on corporate data servers.

Web Services

With the iOS SDK, you can work with XML data to communicate between your client application and the server. XML files provide a lightweight, structured format that your app can easily read and write, and they readily fit into the iOS file system. If you're using SOAP, you can build and parse your own data transactions or use third-party libraries such as gSOAP or Axis2. If you're implementing REST, you can integrate XML directly into your app to provide increased performance. Also, many iOS apps utilize JSON for lightweight data interchange and third-party libraries such as JSON framework.

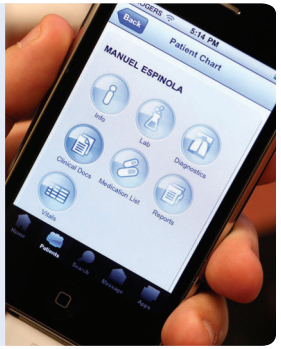
Networking

iOS offers a range of modern, sophisticated, and easy-to-use networking technologies. BSD Sockets is the fundamental iOS network programming interface; all of the higher-level frameworks are based on it. It's a good choice for maximum performance and flexibility. Because BSD is the de facto standard for UNIX network programming, it's often easy to port networking code over from other platforms.

Bonjour is the powerful protocol from Apple that makes it easy to find systems and services on a local network automatically, without tedious configuration. Your app has access to these features through high-level frameworks that make it easy to connect to, render, and interact with information anywhere in the world.

"We had to figure out a way to make updates and changes really quickly, so we went the hybrid approach. Which was native UI elements living on the phone, and the rest was all actually web pages."

—Giancarlo De Lio, Mt. Sinai Hospital



Quick Tip: Web View

To integrate web content, you simply include a UIWebView object within your native app, attach it to a window, and send it a request to load web content. You can also use this class to move back and forward in the history of web pages, and you can even set some web content properties programmatically.

Local Storage

iOS provides Core Data and SQLite to help your app manage and interact with data stored on the device itself.

- **Core Data.** The Core Data framework includes generalized and automated solutions to common tasks associated with object life cycle and object graph management, including persistence. Core Data provides a general-purpose data management solution developed to handle the data model needs of every kind of application, large or small. You can quickly define your apps data model graphically and easily access it from your code. It provides an infrastructure to deal with common functionality, such as save, restore, undo, and redo, allowing you to get on with the task of building innovation into your app. Because Core Data uses the built-in SQLite data library, you don't need to install a separate database system.
- **SQLite.** iOS includes the popular SQLite library, a lightweight yet powerful relational database engine that's easily embedded into an app. Used in countless apps across many platforms, SQLite is considered the de facto industry standard for lightweight embedded SQL database programming. Unlike the object-oriented Core Data framework, SQLite uses a procedural, SQL-focused API to manipulate the data tables directly. You can even use SQLite in a web app using JavaScript.

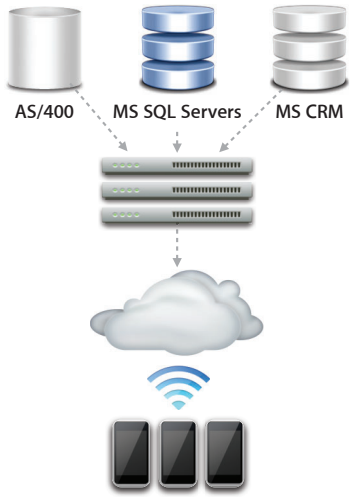
Securing Your App

Chances are your in-house app uses sensitive corporate data that needs to be secured and protected. Many of the basic device-level safeguards, such as passcode policies and remote wipe, are available to your IT managers to administer. But regardless of these security features, it's always a good idea to have a strategy for securing data residing within your in-house apps.

To support the process of securing data in your app, iOS provides a "sandboxed" approach and requires that apps be signed so they can't be tampered with. iOS also has a framework that facilitates secure storage of app credentials in an encrypted keychain. And it provides high-level frameworks that can be used to encrypt app data and provide secure networking. You can leverage all these capabilities within your own development process to provide a secure foundation without impacting the user experience.

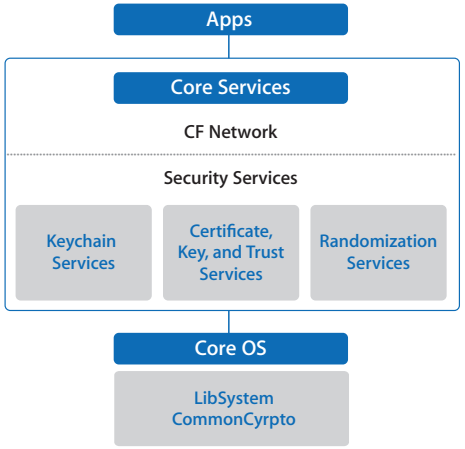
Example: Centralized Web Services

A great way to simplify your back-end development is to centralize web services, particularly if your app needs to talk to multiple back-office systems. For example, Sunbelt Rentals used .Net application servers to write scripts that could invoke stored procedures on legacy AS/400 systems as well as Microsoft CRM databases. They were then able to expose the data retrieved by .Net via XML that could be easily consumed by the mobile app.



"Security was a big factor in our decision to do application development for the iPhone. The iPhone offered us HTTPS security, keychain for keeping certain things private, and we were able to encrypt data."

—Keith DeBickes, JM Family



Architecture

The iOS security APIs are located in the Core Services layer of the operating system and are based on services in the underlying Core OS (kernel) layer of the operating system. Apps call the security services APIs directly rather than going through the Cocoa Touch or Media layers. Networking apps can also access secure networking functions through the CFNetwork API, which is also located in the Core Services layer. CFNetwork is a high-level C API that makes it easy to create, send, and receive serialized HTTP messages. Because CFNetwork is built on top of Secure Transport, you can encrypt the data stream using any of a variety of SSL or TLS protocol versions.

Network Security

Without the need for you to do any coding, iOS supports VPN services, enabling in-house apps to communicate with corporate networks securely. Enterprise IT organizations can configure the built-in VPN settings for IPSec, L2TP, or PPTP, or they can instruct users to download the Juniper, Cisco, or F5 SSL VPN client apps from the App Store.

For configurations using certificate-based authentication, iOS provides VPN On Demand. VPN On Demand establishes a connection automatically when an app accesses predefined domains, providing seamless connectivity for in-house apps. For apps that need Wi-Fi access, iOS supports WPA2 Enterprise Wi-Fi with 802.1X authentication. iOS also supports standard authentication methods such as digital certificates, security tokens such as a Secure ID or CRYPTOCard, and password authentication.

Quick Tip: Shared Keychain

You can share keychain items between multiple apps. Sharing items makes it easier for apps in the same suite to interoperate more smoothly. For example, you could use this feature to share user passwords or other elements that might otherwise require a separate prompt to the user from each application. Accessing shared items at runtime involves using the Keychain Services programming interface with the access groups you set up during development. For information about how to access the keychain, see the *Keychain Services Programming Guide* in the iOS Developer Portal.

Data Security

In-house apps can protect sensitive data by taking advantage of the built-in encryption features available in the latest generations of Apple devices. Data Protection leverages each user's unique device passcode in concert with the hardware encryption on the device to generate a strong encryption key.

When your in-house app designates a particular file as protected, the system stores that file on the device in an encrypted format. While the device is locked, the contents of the file are inaccessible to both your app and to any potential intruders. However, when the user unlocks the device, iOS creates a decryption key that gives your app access to the file. You'll need to design your app to secure the data as it's created and to be prepared for changes in accessing that data when the user locks and unlocks the device.

Secure Authentication

iOS provides a secure, encrypted keychain for storing digital identities, user names, and passwords. The operating system partitions keychain data so that credentials stored by third-party apps cannot be accessed by apps with a different identity. This enables iOS to secure authentication credentials across a range of apps and services within the enterprise. In iOS, Keychain Services checks an app's signature before giving it access to a keychain, handling all keychain access without user interaction. Your in-house apps can interact with the keychain through the Keychain Services API.

Testing and Validation

Validating testing for performance, UI optimization, network testing, and real-world usability should be an integral part of your ongoing development process. In fact, the motto "test early and often" is key to a successful iOS app development project. You can ensure that your app design and code are on track with early testing and validation using a number of methods. The following is a summary of the iOS testing tools you can use for analysis and debugging.

Static analysis. Find bugs in your code before the application is even run by letting the Xcode built-in static analyzer try out thousands of possible code paths in a few seconds, reporting potential bugs that could have remained hidden or nearly impossible to replicate.

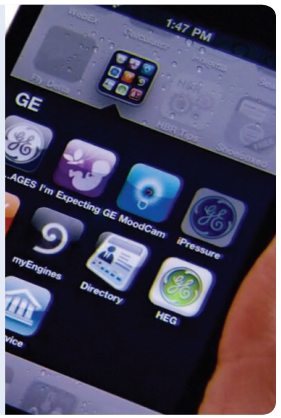
On-device real-time debugging. Plug in your device to use the Xcode graphical debugger, or collect real-time performance data in Instruments' timeline view. These powerful optimization tools allow you to quickly identify and address any performance issues. You'll be able to see variable values with a mouse hover.

Quick Tip: Authentication Library

An elegant way to implement security in iOS apps is to create a shared authentication library you can use across all in-house apps. You can integrate this authentication library with your existing directory services (LDAP or Active Directory) so that each time you create a new app, you don't need to write new authentication code. By storing a credential in the shared keychain, your user experience is further enhanced because users don't have to log in to each app they use. Your library could also define time-out periods per your internal IT requirements. Sharing this type of code across internal apps creates consistency in your policies and consistency in the user experience—a win for both users and IT.

"We have a single sign-on on all apps, and we have access-control lists, on the private ones. So even if you download it, for single sign-on, it'll pass a perimeter over to our application where we'll do a check to see if you have access. And if you do, great, and if not, you won't be able to get into it."

—James Blomberg, General Electric



Instruments. The Instruments application is a powerful performance measurement tool that lets you peer into your code as it's running and gather important metrics about what it's doing. You can view and analyze the data Instruments collects in real time, or you can save that data and analyze it later.

Data recording. Tell Instruments which app to analyze and which instruments to use. Click the big red button to start the recording process. Data is collected and stored for further analysis.

Visual comparison. As data is recorded and displayed over time, it's easy to see relationships between different types of collected data as well as the same data collected over multiple runs.

Drill down. Inspect data spikes on the graph to see what code is executing when the spike occurs, then easily jump into Xcode to fix the problem.

Play back. Create an ad hoc test harness by recording a user interacting with your app, then play back the recording to see how code changes affect the performance.

Automated UI testing. Built-in automation instrument works from scripts (written in JavaScript) that you provide to drive the simulation of events in your app. These synthetic events are generated with the help of the accessibility interfaces built into iOS. You can use this instrument to improve your testing process and exercise the user interface elements of your app while it's running on a connected device.

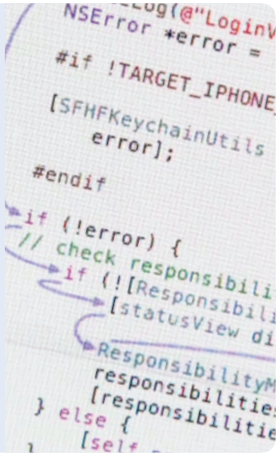
Looking Ahead

As discussed throughout this chapter, the tools and resources provided in the iOS SDK allow you to rapidly develop breakthrough apps that take full advantage of the capabilities of both iOS and the device. Additionally, with the iOS SDK, your apps are ready for deployment at a moments notice. In fact, many in-house development teams take an iterative approach to the entire development process, building and deploying apps frequently throughout the project life cycle. With integrated validation and testing and a security model that protects your enterprise data, you can deploy your apps to users and groups of any size. The next chapter will walk you through the app distribution process step by step and help you craft a strategy that fits your own unique business environment.

Instruments collects and displays data in real time, such as disk, memory, or CPU usage, making it easy to pinpoint problem areas.

"One tool that was invaluable for us was the Static Analyzer, which was able to look for problems without having to actually run the app. Something that could've taken us weeks or months of debugging, or things that we may have never found, the Static Analysis tool was able to find it immediately."

—Mark McWilliams, Razorfish



Deployment

Once development and testing of your app code is complete, there are a few important tasks to undertake before it's ready for users. To prepare your app for distribution, you need to obtain an enterprise distribution certificate from Apple and sign your code in Xcode. After your Xcode project is prepared for deployment, you can host your in-house app securely on your own web server and distribute it directly to users over Wi-Fi and 3G. This chapter outlines the processes for deployment and life-cycle management of your app.

Prepare for Launch

To begin the deployment process, you'll need to certify and provision your app within the iOS Developer Enterprise Program and sign and build your project in Xcode. A simple three-step process will have you ready to distribute your app straight from Xcode.

1. Create and download a distribution certificate. To distribute your iOS app, the designated Agent in your Developer Program membership will need to create a distribution certificate. Only the Agent for your team will be able to create this certificate and only this certificate will enable enterprise app distribution. Find information and step-by-step instructions on how to download and create an enterprise distribution certificate in the iOS Provisioning Portal at http://developer.apple.com/ios/manage/overview/.

2. Create and download a provisioning profile. When you're ready to deploy your app in production, you'll need to create an enterprise provisioning profile. These profiles can be installed on any device, so you'll want to use this method for broad-based app distribution within your company.

Distribution provisioning profiles are matched to your distribution certificate, allowing you to create apps that users can run on their iOS devices. You create a provisioning profile for a specific app, or multiple apps, by specifying the AppID that's authorized by the profile. If a user has an app, but doesn't have a profile that authorizes its use, the user won't be able to use the app. Because these profiles are tied to your certificate, when you revoke your certificate or when it expires, the app will no longer run.

There are two kinds of provisioning profiles: Ad Hoc and Enterprise. Ad Hoc provisioning profiles are restricted to specific device IDs, so they can run only on a specific phone that has been identified (via device ID) and uploaded to the Developer Program Portal. Ad Hoc profiles are best used for internal testing or limited beta programs because they aren't scalable beyond 100 devices and they require a significant administrative overhead (that is, adding device IDs to the program portal).

Deployment Checklist

By the end of the deployment phase, you should have completed:

- ☐ Creation of enterprise certificate and provisioning profile
- ☐ Establishment of a distribution web server or solution for wireless app distribution
- ☐ Announcement of your solution to end users

Overview: Developer Provisioning Portal

The iOS Provisioning Portal takes you through the steps required to test your apps on iOS devices and prepare them for distribution. You'll use the iOS Provisioning Portal for many of the steps described in this chapter, such as creating certificates and provisioning profiles. Visit the Member Center within the iOS Dev Center to locate the portal, where you'll also find additional helpful documentation.

It's important to note that a provisioning profile is not a security mechanism. While it does provide basic authorization for an app to run, it doesn't provide user authentication or additional protection of data used or accessed within your app. It's always a best practice to secure the app itself through internal means. As mentioned in the "Development" chapter of this guide, you can leverage a wealth of iOS security features and frameworks for your in-house app. For example, one of the best ways to secure your in-house app is to create a standard library for user authentication.

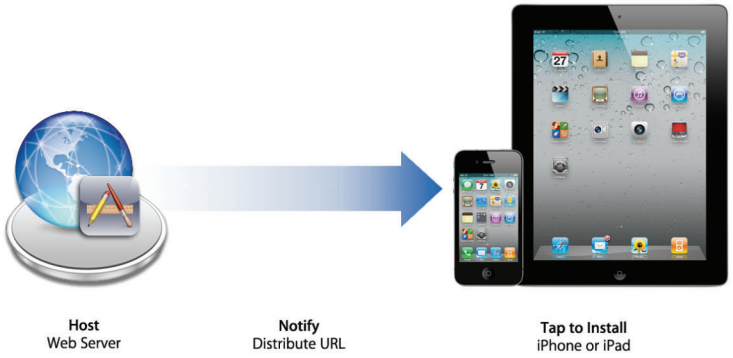
3. Sign and build in Xcode. Once your distribution certificate and provisioning profile are installed, you'll need to sign your code in Xcode. For more details on the code signing process, follow the step-by-step instructions provided in the Developer Provisioning Portal.

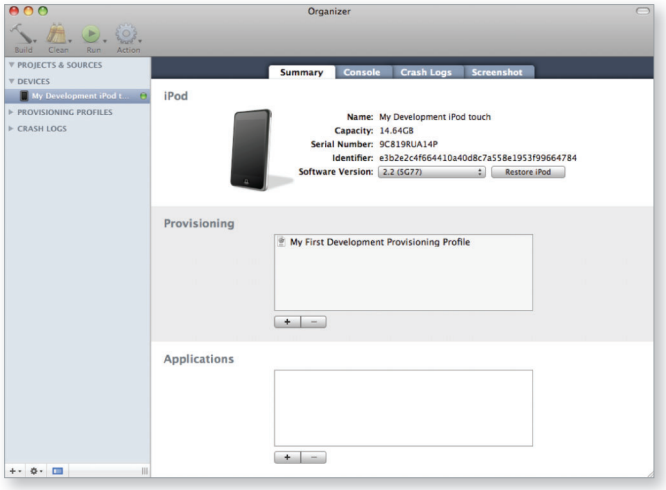
When your app is signed, Xcode packages it for enterprise distribution with a simple export process. Use the Xcode Organizer to share a project that's been added to your archive and select the options for enterprise distribution. This process automatically packages the app, the provisioning profile, and other elements needed for wireless distribution.

Distribution

Once you've finished building your app, distributing in-house apps can be done either by hosting your app on a simple web server you create internally, setting up your own in-house app catalog, or using a third-party mobile device management solution.

The solution that's best for you depends on your specific requirements, your infrastructure, and the level of app management you need.





Overview: Xcode Organizer

The Organizer is a single window for managing Xcode projects, SCM repositories, app archives, and devices—including one-click setup of new iOS devices for development. In the context of app distribution, the organizer is the central library from which apps can be shared (exported) for enterprise distribution. The Organizer can also be used to install in-house apps and provisioning profiles on connected devices.

Wireless App Distribution Process

The simplest way to distribute your app is to host it on a web server. Just follow these steps:

1. Host your app on a web server that your employees can access.

2. Notify your users that the app is available via email, SMS, push notification, or other methods users can receive on an iOS device; be sure to include the app's URL.

3. Tap the URL to install the app. A dialog will ask if they want to proceed with installation.

For more details on how to establish your own wireless app distribution service visit http://help.apple.com/iosdeployment-apps/.

In-House App Catalog

Your team can also build an internal app catalog that provides a portal for over-the-air distribution of your iOS apps. This self-service model requires minimal download and installation effort for employees. The catalog can deliver app download URLs directly, allowing multiple apps to install and update concurrently and enabling fast deployment and setup. A website or native app—optimized for iPhone or iPad—is an even easier method for serving URLs in an organized and familiar way. For an informative example of an in-house app catalog, see "Case Study: The GE Internal App Store" to the right.

Managing Updates

In-house apps that are distributed internally aren't automatically updated. You'll need to notify employees of the update and instruct them to install the app. If the application identifier assigned to the app in Xcode is unaltered, it will recognize the app as an existing app and install the update while retaining locally stored app data or preferences. For greater convenience, consider developing a function within the app that contacts the server for updates at runtime.

With wireless app distribution, you can provide a link to the updated app right within your app. If you create a native app catalog application, consider using Push Notification Services with an alert or icon badge that lets users know updates are available.

Mobile Device Management

Many third-party mobile device management solutions provide wireless app distribution capabilities right out of the box. The benefits of managing in-house apps within a managed environment include the ability to do version control and track which users are running which version of your app. Many device management solutions also provide Push Notification services that let users know when new and updated apps are available. And because mobile device management solutions can establish

Case Study: The GE Internal App Store

GE's in-house mobile task force, the Mobile Center for Excellence, doesn't just develop cutting-edge apps. They've also built an internal web portal, the GE Internal App Store, to simplify downloads of company-specific apps for GE's 300,000+ employees.

"We needed a great way to distribute mobile applications internally," says James Blomberg, GE's director of New Media and Emerging Technology. "We also have apps on Apple's App Store, but we needed something private as well, for GE applications that shouldn't be shared with the world."

Since its launch in 2009, the GE App Store has logged tens of thousands of internal visitors and more than 100,000 app downloads. When new apps are available, the group promotes them on a companywide intranet portal and through postings and word of mouth among GE's 200-member Mobile Center of Excellence, which includes participants from all of GE's major business sectors.

The store's success is due in part to its simple, effective design. "It's a rich interface, but very easy to use," says Dayan Anandapa, director of Digital Technologies and Collaboration at GE. "Once you register, you click on a URL to help you through the download process. Since the devices themselves are seamless, we want installation to be seamless as well."

At a company as large and diverse as GE, not all in-house apps are appropriate for all users. To control access and make apps available only to qualified users, the company has instituted a two-tiered access system. "We have a single sign-on for all apps, and additional access control lists for the private ones," says Blomberg.

The GE Internal App Store helps drive overall awareness of GE's mobile resources in addition to easing the app download process. But it has other benefits as well.

"It serves as a repository, a central knowledge-sharing hub for our different businesses," Blomberg says. "And it's really opened up new relationships. People who didn't know each other now work together. Across GE, there's a tremendous amount of collaboration and communication through mobile."

network configurations and security policies, it's a great way to deliver settings directly to the device at the time an app is installed (for example VPN or Wi-Fi certificates). For more information about mobile device management solutions, visit www.apple.com/iphone/business/integration/mdm.

Announcing Your Solution

Congratulations! You've designed, developed, and deployed an iOS app for your employees. The only thing left to do is shout it from the rooftops within your organization. Some of the best, and most innovative in-house apps can fail to achieve the business adoption or return on investment if users don't know about them. You can communicate the solutions to your users in many ways. Here are a few ideas to consider when assembling your app announcement and communication package:

- Consider promoting your latest and greatest in-house apps on your company intranet.
- Create a dedicated site on your intranet just for iOS apps, and allow users to post comments, participate in forums, and so on.
- Provide a video demo of your app in action that can help users understand the power of the solution.
- Send email and newsletters to raise awareness.
- Put up posters and other graphics at key locations so employees will discover the app as they move about your office or corporate campus.
- When possible, send your users push notification messages of new apps as they become available or offer major feature enhancements.
- Supplement your internal app catalog with screenshots and video demos of your app so your employees can learn more about its purpose.

Looking Ahead

Deploying and announcing your app is not the end of the process—it's really just the beginning. With each successful mobile app, your users will be clamoring for more. This guide is merely a starting point for your development team. Beyond this guide, a wealth of learning resources, best practices, tips, and techniques are available as part of the iOS Developer Program. Connect with others through the developer forums, or download developer videos to explore and discover more advanced capabilities of the platform. The possibilities are limitless.

Example: Internal Communications

In announcing new in-house apps for their users, Genentech made communicating the features and benefits of each individual solution a top priority. Just like a commercial developer might create a product launch campaign, Genentech created unique internal marketing materials for each new in-house app. These efforts had an immediate impact and raised overall awareness and adoption of the new apps.

Additional Education Resources

Want to take your in-house development to the next level? The advanced learning resources listed below go deeper and provide detailed technical information on the most relevant in-house app development topics.

WWDC Videos

To watch Apple engineers and experts discuss how to innovate with the latest Apple technologies, visit: http://developer.apple.com/videos/wwdc/2010/

Stanford University Podcast

Learn the tools and APIs necessary to build apps for iPad, iPhone, and iPod touch. "Developing Apps for iOS" from Stanford University covers user interface design for mobile devices, unique user interactions using Multi-Touch technologies, Core Animation, and more. Find this series on iTunes or visit:
http://itunes.apple.com/us/itunes-u/developing-apps-for-ios-hd/id395605774

Big Nerd Ranch

Get a comprehensive kickstart to iOS development at this seven-day course designed to provide you with the basics of Objective-C and the foundations of the iOS SDK. Big Nerd Ranch can also come to you and provide workshops for your development team onsite. To learn about Big Nerd Ranch offerings, visit: www.bignerdranch.com

Pragmatic Studio

Learn how to create full-featured iOS apps from scratch in this four-day, hands-on training course. To learn more, visit: http://pragmaticstudio.com/iphone/

About Objects

Get up to speed on iOS app development with smaller class sizes and more individualized instruction. To see the complete list of training courses, visit: www.aboutobjects.com