Deployment von iOS Apps in Unternehmen

Von Thomas Kestler, elevato GmbH – 25.08.2012

Einleitung

Die Entwicklung mobiler Apps an sich sich schon eine Herausforderung, Apple macht es einem Unternehmen dazu aber auch nicht gerade leicht, iOS-Apps zu deployen. Der Aufwand, eine App in den iTunes Store ist erheblich und der Weg dorthin steinig.

Dieser Artikel beschreibt die wichtigsten Begriffe auf Abläufe. Natürlich ist das bei Apple auch sehr, sehr ausführlich, aber wenig verständlich bei Apple auf den Entwicklerportalen beschrieben. Ich habe aber festgestellt, dass eine kompaktere und didaktisch strukturierte Abhandlung hier hilfreich ist.

Um es vorweg zu nehmen, man benötigt mindestens einen Apple Computer, z. B. Ein MacBook Pro, um iOS-Apps in den iTunes Store zu bekommen.

Begriffsklärung

Im Folgenden beschreibe ich die wichtigsten Begriffe zu dem Thema.

Developer-Account

Damit man iOS-Apps entwickeln und verteilen kann, benötigt man einen iOS-Developer-Account. Ob Einzelperson oder Unternehmen, das ist hier ziemlich gleich, nur dass die Akkreditierung als Unternehmen zeitaufwändiger ist (von Behörden ganz zu schweigen).

iOS Provisioning Portal

Dieses Web-Portal dient der Veraltung von Zerifikaten, Profilen, etc. Der Login erfolgt mit der Apple-ID des jeweiligen Benutzers, Je nach Rolle (siehe unten) stehen unterschiedliche Funktionen zur Verfügung.

Team Agent

Ein Mitarbeiter des Unternehmens wird als Team Agent bestimmt. Er verwaltet den Developer-Account. Es macht allerdings Sinn, statt hans.huber@meinunternehmen.de ein Funktionspostfach zu verwenden, denn Herr Huber kann das Unternehmen ja auch mal verlassen. Der Team Agent kann weitere Admins bestimmen und ist als einziger berechtigt, Apps in den iTunes Store hoch zu laden. Beim Anlegen des Developer Account wird der Team Agent auch gleichzeitig zum Legal Account, d. h. Rechtlich verbindlicher Ansprechpartner. Die Rolle des Team Agent kann Hr. Huber weitergeben, die Rolle des Legal Account bleibt ihm erhalten.

Team Admin

Ein Team Admin kann neue Mitglieder in das Team aufnehmen oder wieder entfernen, er ist für Bestätigung von Zertifikatsanfragen und Verwaltung von Geräten und Profilen zuständig. Es kann mehrere Team Admins geben. Der Team Agent ist gleichzeitig Team Admin.

Team Member

Ein „normaler“ Entwickler wird als Team Member geführt. Jeder Entwickler kann ein Entwicklungszertifikat anfordern und damit entwickeln.

Zertifikate
Es gibt Entwicklungszertifikate und Distributionszertifikate. Beide dienen dem Signieren von Apps, beide sind von der Apple WWDR CA[1] beglaubigt. Aber nur letztere erlauben ein Hochladen in den iTunes Store.

Provisioning Profiles
Zu jeder App gehört ein Provisioning Profile (PP), welches Information über die signierenden Zertifikate, der für Entwicklung zugelassenen Geräte und weitergehende Konfigurationen (In-App-Purchase, GameCenter, Push) enthält. Keine App ohne PP. Auch hier gibt es zwei Ausprägungen, PPs für die Entwicklung und PPs für die Verteilung, sog. Distribution Provisioning Profiles (DPP).

AppID
Jede App benötigt eine Bundle-ID, z. B. `de.elevato.mobile.superapp`. Diese muss mit der AppID im PP übereinstimmen, wobei es auch Wildcard AppIDs giben darf (`de.elevato.*`). Wildcard-AppIDs haben allerdings Einschränkungen, so ist z.B. Push Notification oder In-App-Purchase damit nicht nöglich.

Es können beliebig viele AppIDs erzeugt werden, aber nie mehr gelöscht werden, daher sollte man hier planvoll vorgehen.

Geräte-ID / UDID
Jedes mobile Apple-Geräte hat eine eindeutige ID, die UDID[2] (40-stellige Hex-Zahl). Apps. Die mit einen Enwicklungs-PP ausgeliefert werden, laufen nur auf den in diesem PP gelisteten UDIDs. Daher muss der Team Admin die Liste der (bis zu 100) UDIDs im iOS Provisioning Portal verwalten.

iTunes Connect
Dieses Portal dient dem Team Agent zum Verwalten der Apps im iTunes Store und zum Anlegen neuer Apps (oder Versionen) und der Vorbereitung zum Hochladen.

Berechtigungen und Privatsphäre

Gerade Unternehmen werden von außen oft kritisch betrachtet und sollten daher sehr sorgfältig mit den von der App geforderten Berechtigungen und der Privatsphäre der Benutzer umgehen. Es sollten keine Berechtigungen gefordert werden, die nicht zwingend erforderlich sind. Ein Beispiel: wer seine App kompatibel zu Android 1.5 machen will, fordert damit implizit Internet-Verbindung an, selbst wenn dies gar nicht nötig ist. Eigentlich kann man Android 1.5 mittlerweile aufgrund der geringen Verbreitung ignorieren. Im Rahmen der QS sollte sichergestellt werden, dass keine unnötigen Berechtigungen angefordert werden.

Auch die Informationen und Vereinbarungen zur Privatsphäre sollten sehr sorgfältig überlegt und deren Umsetzung und Einhaltung laufend kontrolliert werden.

Das Verpackungsformat IPA

Eine App wird als .ipa-Datei bereitgestellt. Hierbei handelt es sich um ein ZIP-Archiv mit dem Unterordner `Payload` und darunter ein Ordner für die eigentliche App, z. B. `superapp.app`. In diesem sind die ausführbaren Dateien enthalten, das PP

1 World Wide Developer Relations Certification Authority
2 Universal Device Identifier

(`embedded.mobileprovision`) und die Code-Signatur. Die Code-Signatur besteht aus SHA1-Hashes, die mit dem privaten Schlüssel des Signierers erzeugt wurden und gegen den Public-Key des Zertifikates im PP verifiziert werden können.

Üblicherweise erzeugt und signiert ein Entwickler die IPA-Datei mit seinem Entwicklungswerkzeug. D. h. Die IPA-Datei enthält sein Entwickler-PP und ist damit auf keinen Fall für das Hochladen in den iTunes-Store geeignet. Vor der Einlieferung muss die IPA also resigniert werden.

Resignieren der App

Der Team Agent kann ein Distributionszertifikat und ein Distributions-PP erstellen und damit die App signieren. Nur in dieser Form ist ein Hochladen in den iTunes-Store möglich. Das DPP muss speziell für iTunes-Store markiert sein (nicht AdHoc).

Man könnte nun auf die Idee kommen, die App als Team Agent neu zu bauen und dann entsprechend zu signieren. Das würde aber eine Vielzahl von Fehlerquellen eröffnen und man lädt eine App hoch, die in dieser Form nie getestet wurde! Daher ist es besser, die bereits getestete App zu resignieren. Hierfür gibt es Tools wie iResign[3]. Dieses Programm macht nichts anderes als die IPA-Datei zu entpacken, das PP auszutauschen, die Code-Signatur zu ersetzen und alles wieder einzupacken[4].

Entwicklungsumgebung

Häufig hat man in größeren Unternehmen eine kontrollierte Umgebung (Controlled Environment), d.h. Fehlende Administratorrechte, zentrale Software-Versorgung, Internet-Zugang über Proxy, Policies, USB-Blocker, etc.. Das alles erschwert die Einrichtung einer Entwicklungsumgebung für mobile Apps, da die Hersteller eher von den kleineren Teams oder Einzelentwicklern ausgegangen sind.

Installation auf dem mobilen Gerät

Als Entwickler will man möglichst einfach die App auf das Gerät bringen. Hierzu bietet sich das USB-Kabel an. Auf dem Comupter sind dazu Xcode oder iTunes oder das iPCU[5] nötig. In Unternehmen mit kontrollierter Umgebung kann der USB-Port gesperrt sein. Sofern der USB nicht freigegeben werden kann, muss man auf OTA ausweichen-

OTA

OTA[6] steht für Over the Air[7] und ist nichts anders als der Download einer App (im .ipa-Format). Dazu muss auf dem Web-Server eine Seite mit einem Link auf eine Beschreibungsdatei (.plist) und die IPA-Datei und Icons bereitgestellt werden. Der Link sieht z. B. so aus:

```
<a href="itms-services://?action=download-manifest&url=http://meinunternehmen.de/
manifest.plist">Installieren</a>
```

3 http://code.google.com/p/iresign/
4 http://stackoverflow.com/questions/6896029/re-sign-ipa-iphone
5 Iphone Configuration Utility, auch für iPad geeignet
6 Siehe auch: http://developer.apple.com/library/ios/#featuredarticles/FA_Wireless_Enterprise_App_Distribution/Introduction/Introduction.html
7 Vorsicht vor der deutschen übersetzung bei Apple: „Drahtloses Installieren“ ;-)

Die Beschreibungsdatei kann man in Xcode erzeugen oder manuell anlegen und sie sieht z. B. so aus:

```
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-
1.0.dtd">
<plist version="1.0">
 <dict>
  <key>items</key>
  <array>
   <dict>
    <key>assets</key>
    <array>
     <dict>
      <key>kind</key>
      <string>software-package</string>
      <key>url</key>
      <string>http://meinunternehmen.de/test/superapp.ipa</string>
     </dict>
     <dict>
      <key>kind</key>
      <string>display-image</string>
      <key>needs-shine</key>
      <true />
      <key>url</key>
      <string>http://meinunternehmen.de/images/icon1.png</string>
     </dict>
     <dict>
      <key>kind</key>
      <string>full-size-image</string>
      <key>needs-shine</key>
      <true />
      <key>url</key>
      <string>http://meinunternehmen.de/images/icon2.png</string>
     </dict>
    </array>
    <key>metadata</key>
    <dict>
     <key>bundle-identifier</key>
     <string>de.meinunternehmen.superapp</string>
     <key>bundle-version</key>
     <string>1.0</string>
     <key>kind</key>
     <string>software</string>
     <key>title</key>
     <string>SuperApp</string>
     <key>subtitle</key>
     <string>Best App in the World</string>
    </dict>
   </dict>
  </array>
 </dict>
</plist>
```

Es kann nötig sein, dass man auf dem Web-Server noch entsprechende MIME-Types definieren muss, damit die Dateien richtig ausgeliefert werden:

```
application/octet-stream ipa
text/xml plist
```

Netzwerk-Infrastruktur

In Unternehmen ist das Netzwerk meist abgeschottet. WLANs sind eher ungern gesehen. Für die Entwicklung mobiler Apps ist ein WLAN aber fast unersetzlich. Daher muss frühzeitig mit den Abteilungen für Netzwerk und IT-Sicherheit der Einsatz von WLANs (ggf. abgeschottet wie ein Gastnetz) geklärt werden.

CI/Build
Viele Entwicklungsteams nutzen CI-Systeme wie Hudson/Jenkins. Auch hierfür gibt es bereits Unterstützung (sofern man Hudson auf einem Mac-Server installiert). Einen guten Überblick gibt ein Artikel von Mike Nachbaur:

http://nachbaur.com/blog/how-to-automate-your-iphone-app-builds-with-hudson

Testumgebung
Neben einer Entwicklungsumgebung wird natürlich auch eine Testumgebung benötigt. In Unternehmen wird diese meist von einer eigenen Abteilung betrieben. Da viele Apps gegen ein Backend laufen, muss das zur App-Version passende Backend ebenfalls in der Testumgebung verfügbar sein. Der Zugriff der App sollte dann realitätsnah über UMTS erfolgen, somit muss die Testumgebung (mindestens das Backend) auch über eine DMZ von außen erreichbar sein.

Verteilung für Beta-Test
Bevor man die App in den iTunes Store freigibt ist ein erweiterter Beta-Test mit einem erweiterten Testerkreis sinnvoll. Zunächst sollte die App im Entwickler-TEam soweit getestet sein, dass sie reif ist für den Beta-Test. Der Beta-Test mobiler Anwendungen ist wegen der Vielfalt der Geräte, der Mobilfunkzugänge und Konfigurationen im Feld durch nichts zu ersetzen.

Zunächst müssen die UDIDs dieser Beta-Tester bekannt sein, so dass ein entsprechendes Beta-Test-DPP erstellt werden kann. Anschließen kann die App in einem Download-Bereich des Web-Servers zur Verfügung gestellt werden (Siehe OTA weiter oben). Ob man den Zugang zum Download noch extra schützt, kann man diskutieren. Gerade wenn es sich um eine Marktneuheit handelt, sollte man darüber nachdenken.

Werkzeuge
Für den Entwickler sind effiziente Werkzeuge unverzichtbar. Hier eine unvollständige Auswahl:

Xcode
Das von Apple bevorzugte Werkzeug ist Xcode. Wer native iOS-Apps entwickeln will, kommt daran kaum vorbei. Xcode ist auch mit dem iOS Provisioning Portal verzahnt und kümmert sich somit um Zertifikate und PPs (mit persönlich zu viel Magic). Xcode ist extrem mächtig, läuft aber nur auf Mac. Für Eclipse-Liebhaber wie mich, anfangs reichlich ungewohnt, aber das gibt sich.

iPCU
Das iPhone Configuration Utility ist hilfreich, wenn man mit einem Windows-Computer Apps auf Geräte per USB installieren will. Man kann auch PPs verwalten und Device-Logs auswerten.

iTunes
Wer ein iPhone sein eigen nennt, der kennt iTunes längst. Man verwaltet damit Medien und Apps. Ich bin kein Fan von iTunes und versuche es zu meiden. In Unternehmen dürfte iTunes ebenfalls nicht gerne gesehen sein.

FlashBuilder
FlashBuilder dient dem Erstellen mobiler Apps auf Basis Flex/AIR. Der Mehrwert liegt im Versprechen, dass die App, einmal entwickelt, auf mehrere Plattformen läuft: iOS, Android, Blackberry Tablet OS. In der Tat sind solche Apps ziemlich verbreitet. Leider ist auch hier nicht immer alles Gold, was glänzt: zwar ist Flex/Actionscript eine ausgereifte Programmierumgebung, die kaum etwas vermissen lässt, die Umsetzung, gerade unter iOS, hat aber ihre Tücken.

FB basiert auf Eclipse und ist für Eclipse-Entwickler leicht zu beherrschen.

Hochladen in iTunes-Store

Wenn die App umfänglich getestet wurde, kann sie in den iTunes-Store hochgeladen werden. Dies erfolgt über das Web-Portal iTunes-Connect. Der Team Agent definiert eine neue App und hinterlegt Metadaten wie Beschreibung, Schlüsselworte, Bildschirmkopien, etc. Im Portal kann das (gewünschte) Erscheinungsdatum und die Märkte definiert werden. Letztlich unterliegt aber jede App einer Prüfung durch Apple.

Zusammenfassung

Die Begriffe im Umfeld der iOS-Apps sind anfangs verwirrend, ich hoffe, ich konnte zu Klärung beitragen. Ich habe hier bewusst auf Screenshots verzichtet und manche Details weggelassen, um das Dokument kurz zu halten.

Gerade für größere Unternehmen stellen sich Fragen der Infrastruktur, die sich bei kleinen Teams so nicht stellen.

Letztlich handelt es sich aber auch hier um SW-Entwicklung, welche die nötige Sorgfalt erfordert, zumal die Außenwirkung einer vermurksten App für ein Unternehmen fatal sein kann.

Weblinks

http://developer.apple.com/library/ios/#featuredarticles/FA_Wireless_Enterprise_App_Distribution/Introduction/Introduction.html - Information zu OTA

https://developer.apple.com/ios/manage/overview/index.action - iOS Provisioning Portal

https://itunesconnect.apple.com/WebObjects/iTunesConnect.woa – iTunes Connect

https://developer.apple.com/devcenter/ios/index.action – iOS Developer Center

---

Thomas Kestler ist Geschäftsführer der elevato GmbH. http://www.elevato.de