モバイルゲーム配信業務効率化
&
運用体制

業務部
畑 圭輔、石田 祐介

お品書き

- **自己紹介**
- **ないものは作って効率化**～Web APIを自作する話～
- **登録作業を劇的改善**～Game Service系効率化の話～
- **署名を極める**～iOS/Android™の署名マスターへの道～

自己紹介

- **畑 圭輔**

  業務部 マネージャー・テクニカルディレクター

  iOS/Android™ プラットフォーム関連 全社的な技術マネジメント、リサーチ、対外折衝担当。
  業務改善を”技術で解決”をモットーに内製アプリケーション開発のディレクションも行う。

  【代表作】
  「CRYSTAL DEFENDERS」、 「国破れて山河あり」、
  「ソングサマナー 歌われぬ戦士の旋律：完全版 」[配信：SQUARE ENIX Co., Ltd.]
  - 各iOS版ディレクター担当

- **石田 祐介**

  業務部 アプリケーションエンジニア（プラットフォームサポート）

  iOS/Android™ プラットフォーム技術サポート、業務改善ツール作成を担当

  【代表作】
  「黒騎士と白の魔王」[配信：グラニ]
  - 立ち上げからプログラムを担当
  - サーバーサイド、クライアントサイド、管理画面と幅広く携わる

ないものは作って効率化
～Web APIを自作する話～

ないものは作って効率化~Web APIを自作する話~

ソーシャルゲームの流行し、Push通知の利用が増加

弊社におけるPush通知採用タイトルは、

50タイトル以上！！

実は、Push通知の証明書の作成、運用は
意外と大変・・・

これはその運用改善に立ち向かったお話です・・・

相手を知る～iOSとAndroid™ Push通知の理解～

iOSとAndroid™のPush通知の実装についておさらいしましょう

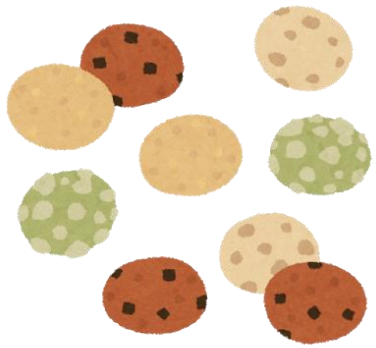
- **iOS：証明書ベースとトークンベース**
  - **証明書ベース** ← 弊社で採用
    - 手順が多く、各種サービス向けにフォーマットが異なる
  - **トークンベース**
    - WWDC 2016でAppleがアナウンス
      - https://developer.apple.com/videos/play/wwdc2016/724/
    - 一度発行すれば**”全て”のタイトル**で利用でき**更新不要**
      - Firebaseでは、この方式を推奨
- **Android™：サーバーキーベース**
  - サーバーキーベース
    - **タイトル別**で発行し、**更新不要**

当社での運用方法と課題

iOS	証明書ベース
Android™	サーバーキーベース

- **iOS側の課題【1】**
  - **タイトル毎にPush通知用のサービスが異なる**
    - ■ P12形式、パスワード指定、PEM形式
    - ■ 採用サービス
      - SQEX BRIDGE
      - mBaaS
      - GrowthPush
      - Amazon
      - Tapjoy
      - Firebase
      - 等・・・

運用上課題あり



当社での運用方法と課題

- **iOS側の課題【2】**
  - **証明書作成手順が地味に面倒**
    - **■ 手順のおさらい**
      - Keychain Accessを使い、.certificateを作成
      - iOS Developerサイトへ.certificateのアップロード
      - 生成されたCertificateのダウンロード(aps.cer)
      - aps.cerをKeychain Accessに登録
      - 登録された証明書と秘密鍵の書きだし
      - P12ファイルを目的のフォーマットに変換

    - **■ 弊社事情**
      - Push通知対応タイトルが50以上。**毎月更新**がある。
      - 手動でももちろんやれるが、**ヒューマンエラーが恐い**

“ツール(自動)化”の前に～手順を整理～

手順	代替方法
Keychain Accessを使い、.certificateを作成	opensslコマンドで対応可能
iOS Developerサイトへ.certificateのアップロード	**Web APIが必要**
生成されたCertificateのダウンロード(aps.cer)	**Web APIが必要**
aps.cerをKeychain Accessに登録	**securityコマンドで対応可能**
登録された証明書と秘密鍵の書きだし	**securityコマンドで対応可能**
P12ファイルを目的のフォーマットに変換	opensslコマンドで対応可能

iOS Developer サイトは、Web APIを公開していないので
自作して頑張る必要があります。
(https://developer.apple.com/account/)

securityコマンドはハマリどころが
あるのでいずれどこかで纏めます（汗

自作する前に・・・～車輪の再発明を避けたい～

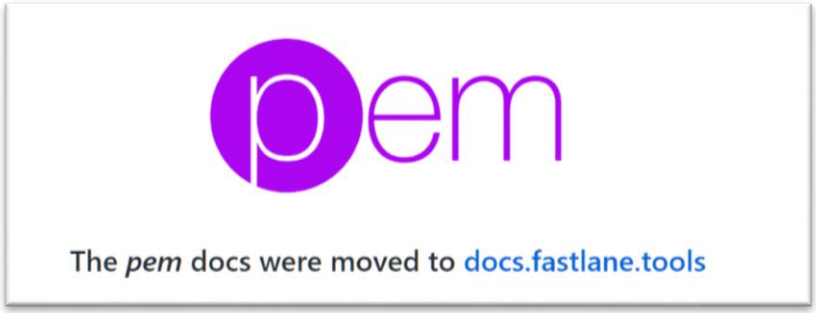
何かを作る前には、誰か作っていないかを調べる。これ鉄則です。

調べると・・・**ありました！！**

fastlane https://github.com/fastlane

fastlaneは、iOS/Android™関連の自動化ツールがまとまっている
人気のOSSプロダクトです。

今回の目的を満たすツールは、証明書作成を担う”pem”



早くも目的達成？

fastlaneを使うと…

- 確かに便利な機能が含まれており、様々な目的を達成可能
- 人気OSSプロダクトであり、メンテナンス性は高い

開発コストを割かずに
導入したい人には、
とてもオススメです！

<u>メリットも多く、採用可能ではありましたが、一方で</u>

- プログラムがモジュール化されておりソースコードも比較的量が多い
- 証明書作成に特化したシンプルなプログラムが欲しい
- （Pythonで作りたかった）

※全て個人の見解です。

再発明ではなく、
自社運用に最適化する

Web API作成 ～最初に考えること～

Web APIを自作する上で考慮した点は以下3点です。

開発メリット

- 自作Web APIの自動化により、私達は**何を得る**のか？

開発コスト

- 開発にかかる工数や当てるリソースに**妥当性**があるか？
- 継続的に**メンテナンス**できるようなプログラムに出来るか？

運用イメージ

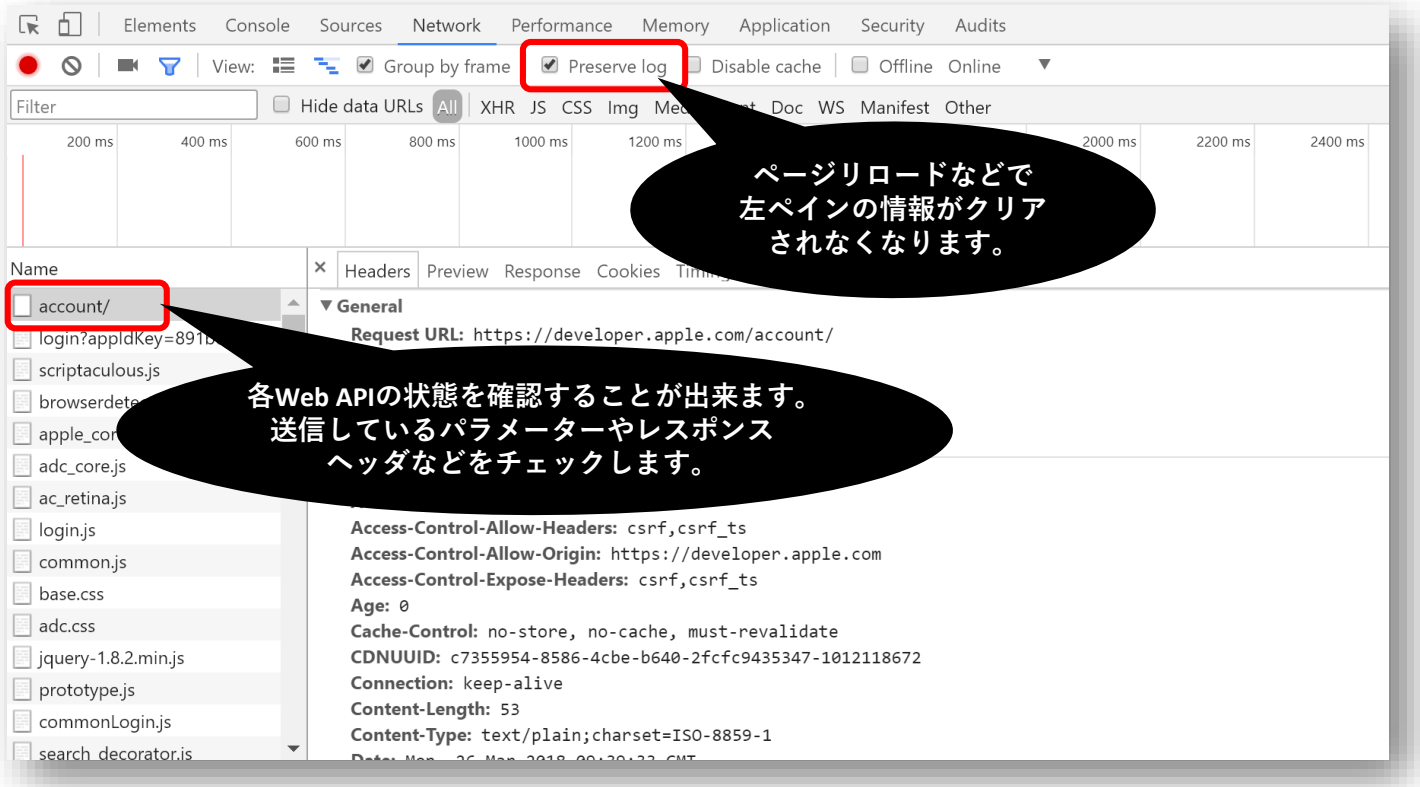
- Web APIを利用することでどれぐらい時間が**短縮**できるか？
- 手順が**簡略化**され、利用者の**スキルに依存しない**？
- 実行環境は**シンプル**にできるか？

Web API作成 ～心強いツール達～

Web APIを自作する上で便利なツールを紹介します。

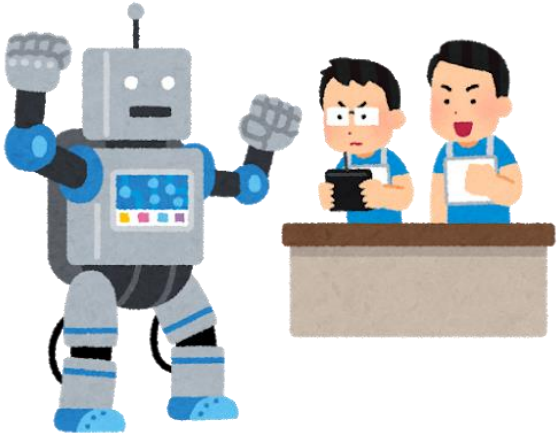
Google Chrome

- 開発者ツールの圧倒的な充実度で各種通信情報を詳細に把握可能



作成したWeb API

- **login**
  - 開発サイトへのログイン(2要素認証対応)
- **get_team**
  - Team情報の取得(弊社の場合は、SEAとSEJの２つのアカウントがあるため)
- **find_appid_id**
  - 指定したバンドルIDから目的のApp ID情報を取得
- **get_appid_detail**
  - App IDに設定されている情報の取得
- **update_push_service**
  - App IDの情報をアップデートする
    - Push通知であれば、設定をオンにする
- **create_push_certificate**
  - Push通知用のCertificateを作成する
- **download_certificate**
  - 作成したcertificateをダウンロード



Web APIを使った流れ

Python

```python
def main():
    # iOSクライアント作成
    ios_client = iOSPortalClient()
    # ログイン
    login = ios_client.login(
        account_name=apple_id,
        account_pw=apple_id_passwd)
    # Team IDの取得
    team_id = ios_client.get_team(apple_id_team)
    # App IDの検索
    finditem = ios_client.find_appid_id(team_id, bundle_identifier)
    _, appid_id = finditem[0]
    # App IDの情報取得
    app_detail = ios_client.get_appid_detail( team_id, appid_id)
    # Push通知の利用をON
    ios_client.update_push_service(team_id, appid_id)
    # Push通知のCertificateをアップロード
    certificateId = ios_client.create_push_certificate(
        teamId=team_id,
        appId_Id=appid_id,
        certSigningRequestFilePath='aps.certSigningRequest'
    )
    # aps.cerのダウンロード
    ios_client.download_certificate(teamId=team_id,
                                    certificateId=certificateId)
```

流れを重視するために
エラーチェックなどは
あえて省いて掲載しています

Web API作成 ～成果～

Web APIを自作することで、得られたことをまとめてみました

開発メリット

- **自作APIの理解**が深まり、**社内の事情にあわせた最適化**が可能に。

開発コスト

- 開発は1名
  - 開発環境
    - **Google Chrome + Visual Studio Code + Python**
      - 無償且つ手軽に構築できる環境
  - 開発言語
    - **Python**
      - **外部ライブラリにほぼ依存しない作り**を目指した

運用イメージ

- 証明書発行までの時間は**約1分に短縮**
- スクリプトを**ダブルクリック**で証明書が発行

登録作業を劇的改善
～Game Service系効率化の話～

登録作業を激的改善～Game Service関連 効率化の話～

iOS/Android™ゲームにおけるエンゲージメントを
高めてくれるサービスや機能といえば、

Game Center / Google Play Game Service です。

開発をする上では今やなくてはならないものですが、
その**登録作業は意外と面倒**。

その作業をいかに効率化したか？
についてお話します。

おさらい ～ゲームサービスの機能～

Game Center, Google Play Game Serviceには、
様々な便利な機能が用意されています。

その中でも共通して用意されているものについておさらいしてみましょう。

サービス名	内容
アチーブメント（Achievement）	**ゲーム内の実績**
リーダーボード（Leaderboard）	ゲーム内のスコアボード
マルチプレイ（Multiplay）	マルチプレイに関する機能
クラウドセーブ（Cloud Save）	ゲームの進捗などをクラウドに保存

最も実装率が高い！！

これらゲームサービスの中でもアチーブメントは、手軽に実装できるということもあり、弊社では多くのアプリケーションで実装されています。

ですが、**実装は、簡単。しかし登録が実は面倒**・・・
そんな課題を弊社では抱え続けてきました。

何が面倒か？

登録作業の**何が面倒**なのでしょうか？

登録作業は、実は各種情報を入力し、ポチッと登録するだけですが、

その**件数が非常に多い場合に問題**になります。

1件当たりの登録時間：平均約1分。

実際にあった例。登録件数100件近く且つ12言語。

1分 x 100件 x 12言語＝約1200分＝**20時間!?**

正直これを手作業でやるのは非現実的です。

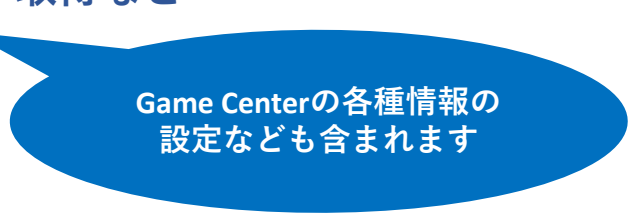
公式配布のツールを使い自動化することでこの問題を解決しました。

公式配布ツール

ゲームサービスなどを登録するためにプラットフォーマーから公式のツールが配布されています。弊社ではこれらツールを使い、ラップすることで使い易さを向上させ業務改善にチャレンジしました。

【iOS】：iTMSTransporter

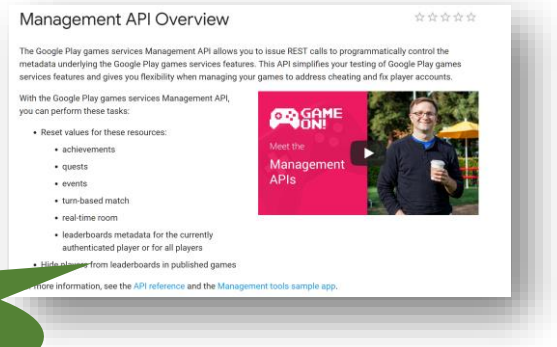
- **入手先**
  - **iTunes Connect内のページから**
    - https://itunespartner.apple.com/jp/apps/guides
  - **マニュアル**
    - https://help.apple.com/itc/transporteruserguide/
  - **対応プラットフォーム**
    - Microsoft Windows, macOS®, Red Hat Enterprise Linux
- **出来る事**
  - **iTunes Connect用のメタデータの設定や取得など**
    - ブック、ビデオ、ミュージック、App



公式配布ツール

【Android™】：game-config.py, Game Service API

- 入手先
  - Google Play Game Serviceページから
    - https://developers.google.com/games/services/management/
  - マニュアル
    - https://github.com/playgameservices/management-tools/tree/master/publishing-sample
      - アチーブメント関連の追加、変更など(game-config.py)
    - https://developers.google.com/games/services/management/api/
      - アチーブメントのリセットなど(WebAPI)
  - 対応プラットフォーム
    - Microsoft Windows, macOS®
- 出来る事
  - Google Play Game Serviceの各種項目の設定
    - アチーブメント、クエスト、イベントなど



game-config.py ツールは
弊社からの要望もあり、
実現しました

https://Android™-developers.googleblog.com/2014/12/google-play-game-services-ends-year.html

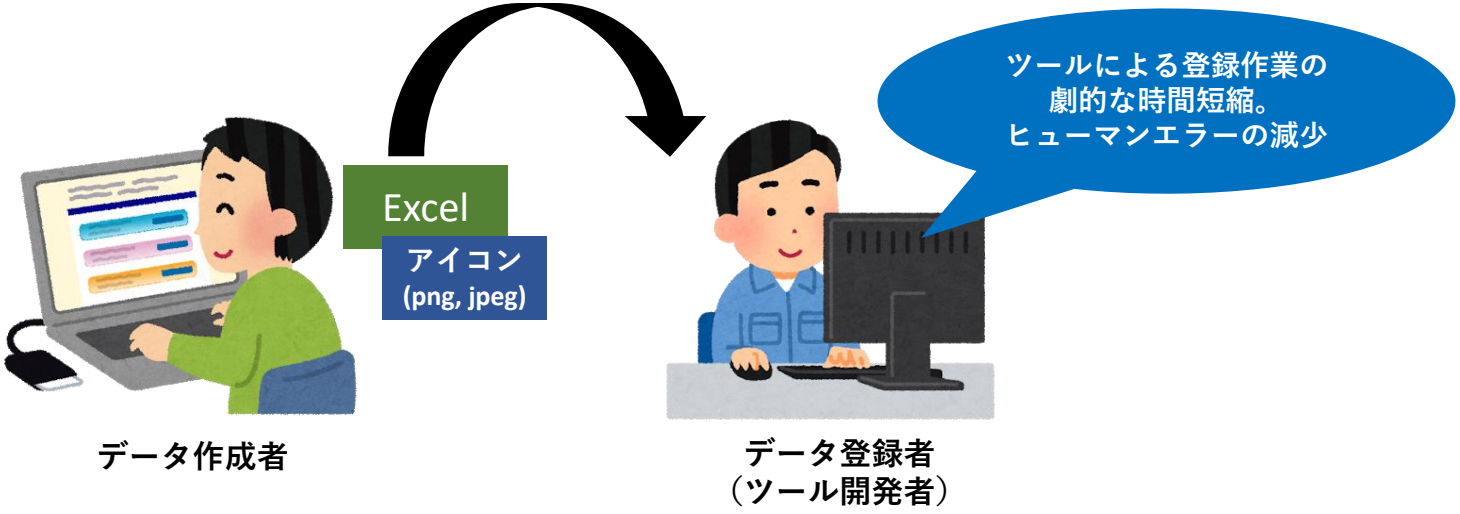
弊社での運用体制～現在～

iOS/Android™公式ツールのラッパーツールを作成。

データ作成者とデータ登録者を分けて運用。

作業フロー

1. Excelでデータを作成、検査
2. アイコン画像とともに登録者へ連絡
3. 受け取ったデータの出力
4. 出力されたデータをラッパーツールで加工してアップロード



弊社での運用体制～現在：Pros/Cons～

Pros

- **データ作成がしやすい**
  - **Excelを使ったデータ作成**
    - 使い慣れたツールであり、利用者側の学習コストが低い
- **データ検査の確実性**
  - **Excel VBAマクロ ＞ データ入力規則**
    - データ入力規則はセルに設定するため、情報が上書きされることも。

Cons

- **VBAの言語仕様と開発**
  - **正規表現が扱いにくい**
  - **開発がしづらい（効率が悪い）**
    - VBAのエディタの品質が良くない。
    - ソースコードの検索は、検索ツールで関数を探すなど、その検索方法が貧弱。
- **VBAのバージョン管理**
  - **VBAコードの管理**
    - 基本的に難しい。差分チェック管理がしづらい。
- **Windowsのバージョンによる互換性**
  - **コンポーネントの互換性**
    - Windows 7で使えていたプログレスバーがWindows 10では使えないなど・・・
- **データ作成用Excel資料同期**
  - **資料更新時の利用者への配付**
    - 資料に更新があった場合に古いExcelファイルを更新する術が利用者任せになりがち
    - 結果、古い資料で納品されることがしばしば

弊社での運用体制～今後～

iOS/Android™公式ツールのラッパーツールを作成。 ここは変わらず

データ作成者とデータ登録者を分けない＝同じ人が行う。

作業フロー

1. Excelでデータを作成
2. アイコン画像とともに登録ツールでアップロード
3. 登録サーバー側でデータの検査、登録処理

弊社での運用体制～今後：運用詳細～

データ作成ツール

- **Excel**
  - データ作成ツールとしてExcelの運用は変更せず、**データ作成に特化**

アーキテクチャ

- **サーバー**
  - **処理アチーブメントデータの検査、登録作業を担う**
  - **C# + .NET Core**
    - **運用サーバーがmac miniで稼働しているため**
    - **Excelを処理する＝.NET Coreライブラリが便利**
      - **ライブラリ：EPPlus（LGPLライセンス）**
      - **https://github.com/JanKallman/EPPlus**

C# .NET Core

- **クライアント**
  - **アップロードするデータの設定、アップロードを担う**
  - **Google Chrome + JQuery(JavaScript)**

こだわりポイント

- **クライアントソフトウェアの追加インストールが不要**
  - ブラウザは、業務上ほぼインストールされている
- **使い易いUI**
  - SKU IDやApplication IDといったシステム用語をなくし、操作マニュアル不要

壱 署名を極める
～iOS/Android™の署名マスターへの道～

署名を極める～iOS/Android™の署名マスターへの道～ 壱

弊社では多くのモバイルタイトルを配信、運用しています。

アプリケーションを配信するには、当然ながら製品版としてビルドされたものが必要であり、
且つ**署名を適用することが絶対条件**です。

ソースコードからビルドすることも解決方法の1つですが、
全てのタイトルに対して、ツールセットを用意し、
ビルドすることは、人、お金、時間が必要になります。

ここから先は、この問題を解決するために弊社の事情や
実践している内容についてお話します。

納品事情と課題

弊社の場合、モバイルタイトルのほとんどが外部の開発会社で行われます。

そのため、最終的にソースコード一式が弊社に納品され、
最終ビルドを作成し、配信を行います。

そこで発生する大きな問題は、**2点**です。

- ソースコードを**ビルドするための環境が**タイトル毎に異なる
  - SDKのバージョン、ツールチェインのバージョン、ビルド手順、独自のツールなど
    ビルドを成功させるための要件が非常に多い
- **ビルド時間が**非常に長くなる傾向にある
  - 昨今のモバイルタイトルのソースコード規模は、アセットを含め、数GBにも及び
    そのビルド時間は、12時間かかることもあった

**結果：納品、検収を行うテクニカルQA部門に
非常に負担がかかってしまいます**

署名プロセスの活用～配信業務効率化～

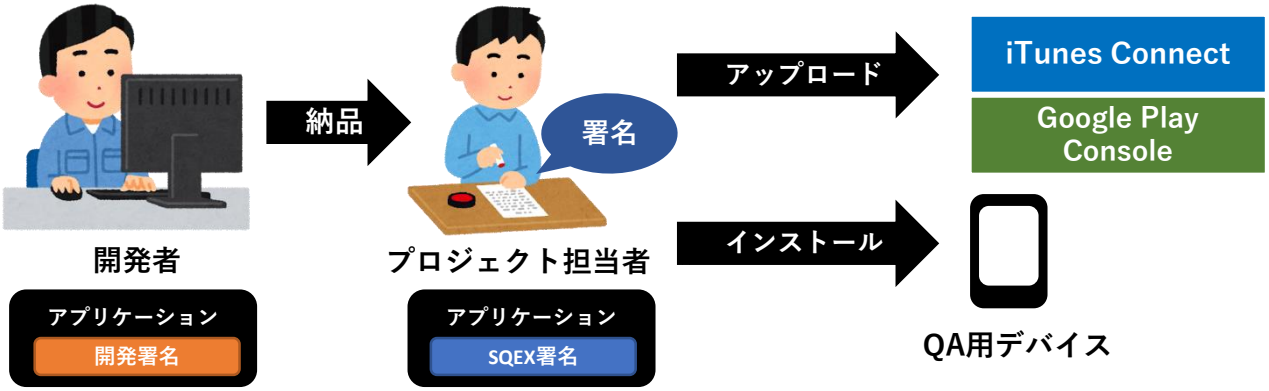
この問題を解決するために弊社では署名プロセスを活用した
独自署名ツールを作成し、既存のアプリケーションに再署名プロセスを導入することでソースコードビルドは不要となり、
アプリケーションの配信業務の大幅な効率化が実現出来ました。

iOS

- **開発会社のアカウントでビルド、エクスポートされたipaファイルを弊社独自の署名ツールで署名、提出**

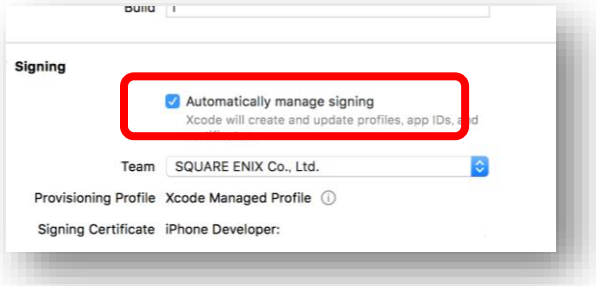
Android™

- **開発会社でビルド、無署名もしくはデバッグ署名されたapkを弊社独自の署名ツールで署名、提出**

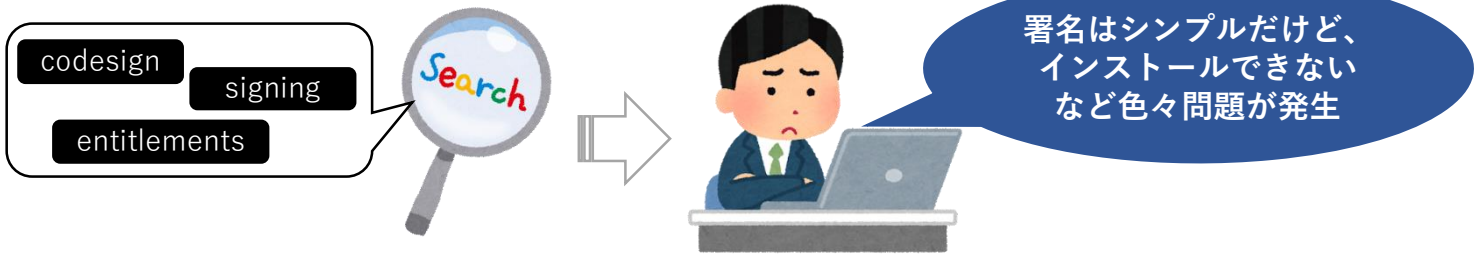


署名の基本的な知識と理解

- 【iOS】の署名
  - **自動署名：Xcodeを用いたビルドと署名**
    - 基本的には、Xcode上でCertificate, Provisioning を設定すれば良い感じに署名
    - Xcode 8からAutomatically manage signing が導入＝さらに楽になった？

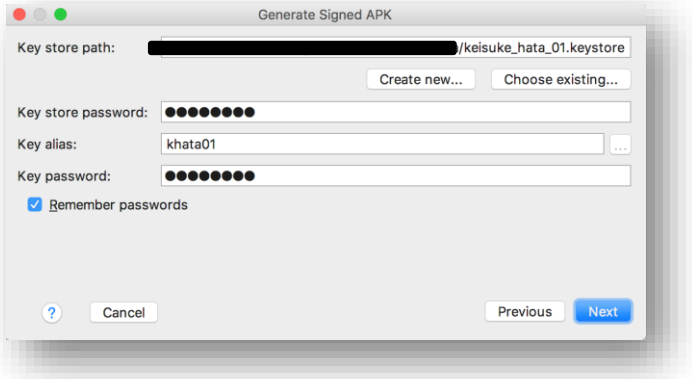


  - **手動署名：codesignコマンド**
    - コマンドラインで用意
    - 署名に関する様々なオプションが用意＝**ただし詳しい使い方がない**
      - WWDCで現地エンジニアに質問するも、裏コマンド？的なことは教えてくれる
      - ドキュメントは存在しないから**自力で頑張れ**と言われる



署名の基本的な知識と理解

- 【Android™】の署名
  - **自動署名：Android Studioを用いたビルドと署名**
    - 基本的には、Android Studio上でKeystore等を設定すれば署名



  - **手動署名：apksignerコマンド**
    - コマンドラインで用意(旧：jarsigner→apksigner)
    - 署名に関する様々なオプションが用意＝**使い方が公式に用意されている！！**
      - https://developer.Android™.com/studio/command-line/apksigner.html

署名に関するセキュリティ

iOSとAndroid™の署名を理解し、運用する上でもう一つ重要なことは、
セキュリティです。運用を間違うと意図しない情報漏洩などにも繋がります。

【 iOS 】

- **“Automatic Signing(自動署名)”利用時には、Apple IDが必要**
  - Automatic Signingを利用したビルドには、 iOS Developerサイトにログイン可能な権限を持つApple IDを用いてXcode上でログインする必要があります。
- **セキュリティ的な気になるポイント**
  - **iOS Developer サイトにログイン可能なApple ID = App ID情報が全て閲覧**
    - iOS Developer サイトにログイン可能なApple IDは、ロールが Member と Adminのみであるため、iTunes Connectのようなタイトル別権限が設定できず、App ID（開発中のBundle ID）情報などが閲覧出来てしまう
- **手動署名ビルド or 納品後にパブリッシャー側で署名対応する必要あり**
  - **開発側に必要な証明書(p12ファイル)とProvisioning Profileを配付**
    - ビルドに必要な証明書などを他企業向けに配付することでビルドが可能になりますが、Apple Developer Program License Agreement における 2.1 Permitted Uses and Restrictions; Program services (c)項 に記載されている内容を守り開発用途でのみ運用すること。Developer Certificateまでに止めるのが運用上許容範囲。

署名に関するセキュリティ

【 Android™ 】

- **Keystore（秘密鍵）の提供**
  - ビルドされたapkを署名するための秘密鍵の提供が必要であり、パスワードも提供する必要がある。
- **セキュリティ的な気になるポイント**
  - **“秘密鍵であるということ”＝つまり配付しちゃダメ**
    - 公式ドキュメント「秘密鍵のセキュリティ設定」でも触れられている通り、その企業のアプリケーションであることを証明するための署名で利用する鍵であるため、外部への配付は、悪用される可能性が高くNGとされている。
      - https://developer.Android™.com/guide/publishing/app-signing.html
- **apk納品後にパブリッシャー側で署名する必要あり**
  - **開発は無署名ビルドで納品し、配信企業側で署名が鉄則**
    - 開発会社には、無署名もしくは、デバッグ署名で納品を義務づける。受け取った企業側で配信するための秘密鍵で最終署名を行う運用が必要。

Google App Signing～統合される署名プロセス～

Google Play Consoleは、前ページの問題などを意識してか、
署名プロセスが統合され「**Google App Signing**」が用意されました。

Google App Signingとは？

- Google Play Consoleにアップロードするための署名と最終署名の2段階で署名を行う方式。2017年5月公開。**全てのDeveloperが利用可能**。

引用：https://play.google.com/apps/publish/
アプリの署名ページより

Google App Signing～規約と運用フロー～

規約の同意

- Google App Signing利用には、**規約の同意が必要**です。
  - https://play.google.com/about/play-app-signing-terms.html
- **規約内容のポイント**
  - Google側に秘密鍵をアップロード＆登録を行う
  - Googleが単独の裁量でアプリの最適化を行うための変更を許諾する
  - 秘密鍵の管理は、Googleは行わない。管理責任はデベロッパー側。

運用イメージ

- **開発会社**
  1. Google Play Consoleアップロード用鍵を配付
  2. 配付された鍵で署名を行い、apkを納品
- **パブリッシャー**
  1. アップロード用鍵で署名されたapkをGoogle Play Consoleにアップロード
  2. 正しいアップロード用鍵であるか検証され、最終署名（秘密鍵）で署名

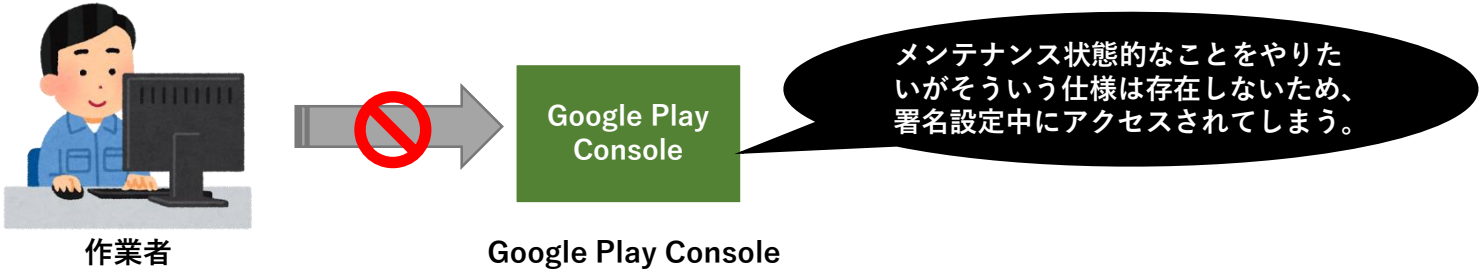
Google App Signing～運用で気になるポイント～

弊社におけるGoogle Play Console運用ルール

- アプリ別に担当者のアクセス権を設定→他のアプリの編集などが出来ない運用
  - 署名も担当者が担当アプリケーションapkをアップロード→署名
    - 署名のために必要な設定（秘密鍵など）はオーナー権限が予め行っておきたい

署名設定の仕様

- オーナー権限以外も設定出来てしまう
  - 全てのアプリケーションの署名設定を予め行う必要があるが、その作業中にGoogle Play Consoleにアクセスされてしまう可能性がある。



署名設定の要望→”署名設定権限はオーナー権限のみ”が理想

Googleにフィードバック済み

- 設定が完了するまでアプリ別権限者の権限を一時的に変更する　という手もあるが、弊社の場合、管理対象ユーザー数が非常に多く設定作業困難。

壱を語り終えて～本日話せなかったこと～

弐 運用してわかった署名プロセスの苦労話など

- 独自の署名ツールとは一体どういう環境か？
- 運用上のトラブルやノウハウ
- 特に難しい“iOSの署名”で知って欲しいこと

参 現在開発してること。今後の展望

- 署名ツールのセキュリティ
- 開発と運用、利用者のあるべき姿

使用している商標について

本資料のイラストは、フリー素材「いらすとや」を利用しております。

https://www.irasutoya.com/