Engineering Subscriptions

Implementation Best Practices

Session 705

Pete Hare, Engineering Manager, App Store
Michael Gargas, Technical Advocate, App Store Operations
Device and Server Architecture
In-App Experience
Reducing Subscriber Churn
Analytics and Reporting
Device and Server Architecture
What is a subscription?
1. Receive Transaction
2. Verify Authenticity
3. Update Subscription Status

Subscription Period
1. Receive Transaction
2. Verify Authenticity
3. Update Subscription Status

Subscription Period
1. Receive Transaction
2. Verify Authenticity
3. Update Subscription Status

Subscription Period
1. Receive Transaction
2. Verify Authenticity
3. Update Subscription Status

Subscription Period
Application

StoreKit

SKPaymentTransactionObserver

Application
// Start Observing the Payment Queue

import UIKit
import StoreKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate, SKPaymentTransactionObserver {

    func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
        SKPaymentQueue.default().add(self)
        return true
    }

}
// Start Observing the Payment Queue

import UIKit
import StoreKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate, SKPaymentTransactionObserver {

    func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
        SKPaymentQueue.default().add(self)
        return true
    }

}
// Start Observing the Payment Queue

import UIKit
import StoreKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate, SKPaymentTransactionObserver {

    func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
        SKPaymentQueue.default().add(self)
        return true
    }

    }

}
/ Start Observing the Payment Queue

import UIKit
import StoreKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate, SKPaymentTransactionObserver {

    func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
        SKPaymentQueue.default().add(self)
        return true
    }

}
class AppDelegate: UIResponder, UIApplicationDelegate, SKPaymentTransactionObserver {
  // MARK: - SKPaymentTransactionObserver

  func paymentQueue(_ queue: SKPaymentQueue, updatedTransactions transactions: [SKPaymentTransaction]) {
    for transaction in transactions {
      switch transaction.transactionState {
      case .purchased:
        // Validate the purchase
      }
    }
  }
}
class AppDelegate: UIResponder, UIApplicationDelegate, SKPaymentTransactionObserver {

    // MARK: - SKPaymentTransactionObserver

    func paymentQueue(_ queue: SKPaymentQueue, updatedTransactions transactions: [SKPaymentTransaction]) {
        for transaction in transactions {
            switch transaction.transactionState {
            case .purchased:
                // Validate the purchase
            }
        }
    }
}
class AppDelegate: UIResponder, UIApplicationDelegate, SKPaymentTransactionObserver {

    // MARK: - SKPaymentTransactionObserver

    func paymentQueue(_ queue: SKPaymentQueue, updatedTransactions transactions: [SKPaymentTransaction]) {
        for transaction in transactions {
            switch transaction.transactionState {
                case .purchased:
                    // Validate the purchase
            }
        }
    }
}
class AppDelegate: UIResponder, UIApplicationDelegate, SKPaymentTransactionObserver {

    // MARK: - SKPaymentTransactionObserver

    func paymentQueue(_ queue: SKPaymentQueue, updatedTransactions transactions: [SKPaymentTransaction]) {
        for transaction in transactions {
            switch transaction.transactionState {
            case .purchased:
                // Validate the purchase
            }
        }
    }
}
App Store Receipt

Trusted record of App and In-App Purchases
App Store Receipt

Trusted record of App and In-App Purchases

Stored on device
App Store Receipt

Trusted record of App and In-App Purchases

Stored on device

Issued by the App Store
App Store Receipt

Trusted record of App and In-App Purchases

Stored on device

Issued by the App Store

Signed and verifiable
App Store Receipt

Trusted record of App and In-App Purchases

Stored on device

Issued by the App Store

Signed and verifiable

For your app, on that device only
Receipt Validation
Don’t let your app be fooled!
Receipt Validation
Don’t let your app be fooled!

On-device validation
• Update subscription state within the app
Receipt Validation
Don’t let your app be fooled!

On-device validation
• Update subscription state within the app

Server-to-server validation
• Online validation through a request to the App Store
• Update subscription state on your server
Receipt Validation
Don’t let your app be fooled!

On-device validation
• Update subscription state within the app

Server-to-server validation
• Online validation through a request to the App Store
• Update subscription state on your server

Do not use online validation directly from the device!
Comparing Validation Approaches
Comparing Validation Approaches

<table>
<thead>
<tr>
<th>On-Device Receipt Validation</th>
<th>Server-to-Server Receipt Validation</th>
</tr>
</thead>
</table>
## Comparing Validation Approaches

<table>
<thead>
<tr>
<th></th>
<th>On-Device Receipt Validation</th>
<th>Server-to-Server Receipt Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate authenticity of receipt</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
## Comparing Validation Approaches

<table>
<thead>
<tr>
<th></th>
<th>On-Device Receipt Validation</th>
<th>Server-to-Server Receipt Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate authenticity of receipt</td>
<td>![Yes]</td>
<td>![Yes]</td>
</tr>
<tr>
<td>Receipt includes renewal transactions</td>
<td>![Yes]</td>
<td>![Yes]</td>
</tr>
</tbody>
</table>
## Comparing Validation Approaches

<table>
<thead>
<tr>
<th></th>
<th>On-Device Receipt Validation</th>
<th>Server-to-Server Receipt Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate authenticity of receipt</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Receipt includes renewal transactions</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Additional user subscription information</td>
<td>✗</td>
<td>✔</td>
</tr>
</tbody>
</table>
## Comparing Validation Approaches

<table>
<thead>
<tr>
<th>Feature</th>
<th>On-Device Receipt Validation</th>
<th>Server-to-Server Receipt Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate authenticity of receipt</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Receipt includes renewal transactions</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Additional user subscription information</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Always “on” to handle renewals</td>
<td>✗</td>
<td>✔</td>
</tr>
</tbody>
</table>
## Comparing Validation Approaches

<table>
<thead>
<tr>
<th>Feature</th>
<th>On-Device Receipt Validation</th>
<th>Server-to-Server Receipt Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate authenticity of receipt</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Receipt includes renewal transactions</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Additional user subscription information</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Always “on” to handle renewals</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Not susceptible to device “clock change”</td>
<td>✗</td>
<td>✔</td>
</tr>
</tbody>
</table>
## Comparing Validation Approaches

<table>
<thead>
<tr>
<th></th>
<th>On-Device Receipt Validation</th>
<th>Server-to-Server Receipt Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate authenticity of receipt</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Receipt includes renewal transactions</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Additional user subscription information</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Always “on” to handle renewals</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Not susceptible to device “clock change”</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>No cryptography needed (OpenSSL, ASN.1)</td>
<td>×</td>
<td>✓</td>
</tr>
</tbody>
</table>
## Comparing Validation Approaches

<table>
<thead>
<tr>
<th>Feature</th>
<th>On-Device Receipt Validation</th>
<th>Server-to-Server Receipt Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate authenticity of receipt</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Receipt includes renewal transactions</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Additional user subscription information</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Always “on” to handle renewals</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Not susceptible to device “clock change”</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>No cryptography needed (OpenSSL, ASN.1)</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>
### Comparing Validation Approaches

<table>
<thead>
<tr>
<th></th>
<th>On-Device Receipt Validation</th>
<th>Server-to-Server Receipt Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate authenticity of receipt</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Receipt includes renewal transactions</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Additional user subscription information</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Always “on” to handle renewals</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Not susceptible to device “clock change”</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>No cryptography needed (OpenSSL, ASN.1)</td>
<td>✗</td>
<td>✔</td>
</tr>
</tbody>
</table>

**Advanced StoreKit**


WWDC 2017
switch transaction.transactionState {
    case .purchased:

        if let appStoreReceiptURL = Bundle.main.appStoreReceiptURL,
            FileManager.default.fileExists(atPath: appStoreReceiptURL.path) {

            let rawReceiptData = Data(contentsOf: appStoreReceiptURL)
            let receiptData = rawReceiptData.base64EncodedString(options: ...)

            // Send up receiptString to server
            currentUser.processTransaction(receiptData) { isValid in

            }

        }

    }
}
switch transaction.transactionState {
    case .purchased:

        if let appStoreReceiptURL = Bundle.main.appStoreReceiptURL,
            FileManager.default.fileExists(atPath: appStoreReceiptURL.path) {

            let rawReceiptData = Data(contentsOf: appStoreReceiptURL)
            let receiptData = rawReceiptData.base64EncodedString(options: …)

            // Send up receiptString to server
            currentUser.processTransaction(receiptData) { isValid in

            }
        }
    }
}
switch transaction.transactionState {
    case .purchased:

        if let appStoreReceiptURL = Bundle.main.appStoreReceiptURL,
        FileManager.default.fileExists(atPath: appStoreReceiptURL.path) {

            let rawReceiptData = Data(contentsOf: appStoreReceiptURL)
            let receiptData = rawReceiptData.base64EncodedString(options: …)

            // Send up receiptString to server
            currentUser.processTransaction(receiptData) { isValid in

            }
switch transaction.transactionState {
    case .purchased:

        if let appStoreReceiptURL = Bundle.main.appStoreReceiptURL, FileManager.default.fileExists(atPath: appStoreReceiptURL.path) {

            let rawReceiptData = Data(contentsOf: appStoreReceiptURL)
            let receiptData = rawReceiptData.base64EncodedString(options: …)

            // Send up receiptString to server
            currentUser.processTransaction(receiptData) { isValid in

            }

        }

    }
}
switch transaction.transactionState {
    case .purchased:

        if let appStoreReceiptURL = Bundle.main.appStoreReceiptURL,
            FileManager.default.fileExists(atPath: appStoreReceiptURL.path) {

            let rawReceiptData = Data(contentsOf: appStoreReceiptURL)
            let receiptData = rawReceiptData.base64EncodedString(options: …)

            // Send up receiptString to server
            currentUser.processTransaction(receiptData) { isValid in

            }

        }

    }
}
switch transaction.transactionState {
    case .purchased:

        if let appStoreReceiptURL = Bundle.main.appStoreReceiptURL,
            FileManager.default.fileExists(atPath: appStoreReceiptURL.path) {

            let rawReceiptData = Data(contentsOf: appStoreReceiptURL)
            let receiptData = rawReceiptData.base64EncodedString(options: …)

            // Send up receiptString to server
            currentUser.processTransaction(receiptData) { isValid in

            }

        }

    }
}
Verifying a Transaction

/processTransaction?userId=90000001
Verifying a Transaction

Your Server

/processTransaction?userId=90000001

{ receiptData: "e2FzZ...jg3OTgwN30=" }
Verifying a Transaction

Server-to-server receipt validation

App Store
/verifyReceipt

Your Server
/processTransaction?userId=90000001
Verifying a Transaction
Server-to-server receipt validation

{ receipt-data: "e2FzZ…jg3OTgwN30=", password: "416e7469204865726f" }

/verifyReceipt

/App Store

/processTransaction?userId=90000001

/Your Server
Verifying a Transaction

Server-to-server receipt validation

App Store

/verifyReceipt

Your Server

/processTransaction?userId=90000001

{ receipt-data: "e2FzZ…jg3OTgwN30=", password: "416e7469204865726f" }
Verifying a Transaction

App Store
/verifyReceipt

Your Server
/processTransaction?userId=90000001
Verifying a Transaction

```json
{
    status: 0,
    receipt: {
        bundle_id: "com.your.app",
        in_app: [{
            transaction_id: "1234567890",
            product_id: "com.your.product.id",
            original_transaction_id: "1133557799",
            expires_date: "2018-07-08...",
            ...
        }]
    }
}
```

Your Server:
/processTransaction?userId=90000001

App Store:
/verifyReceipt
Verifying a Transaction

```json
{  status: 0,
  receipt: {
    bundle_id: "com.your.app",
    in_app: [{
      transaction_id: "1234567890",
      product_id: "com.your.product.id",
      original_transaction_id: "1133557799",
      expires_date: "2018-07-08...",
      ...
    }]
  }
}
```

![Diagram](https://via.placeholder.com/150)

Your Server
/processTransaction?userId=90000001
Verifying a Transaction

{ status: 0,
  receipt: {
    bundle_id: "com.your.app",
    in_app: [{
      transaction_id: "1234567890",
      product_id: "com.your.product.id",
      original_transaction_id: "1133557799",
      expires_date: "2018-07-08...",
      ...
    }]
  }
}
Verifying a Transaction

```json
{
  status: 0,
  receipt: {
    bundle_id: "com.your.app",
    in_app: [{
      transaction_id: "1234567890",
      product_id: "com.your.product.id",
      original_transaction_id: "113357799",
      expires_date: "2018-07-08...",
      ...
    }]
  }
}
```

/verifyReceipt

App Store

/processTransaction?userId=90000001

Your Server
Verifying a Transaction

```
{
  status: 0,
  receipt: {
    bundle_id: "com.your.app",
    in_app: [
      {
        transaction_id: "1234567890",
        product_id: "com.your.product.id",
        original_transaction_id: "1133557799",
        expires_date: "2018-07-08...",
        ...
      }
    ]
  }
}
```

Your Server
/processTransaction?userId=90000001

App Store
/verifyReceipt
Verifying a Transaction

```json
{
  status: 0,
  receipt: {
    bundle_id: "com.your.app",
    in_app: [{
      transaction_id: "1234567890",
      product_id: "com.your.product.id",
      original_transaction_id: "1133557799",
      expires_date: "2018-07-08...",
...
    }]
  }
}
```

App Store

/verifyReceipt

Your Server

/processTransaction?userId=90000001
Verifying a Transaction

```
{ status: 0,
  receipt: {
    bundle_id: "com.your.app",
    in_app: [{
      transaction_id: "1234567890",
      product_id: "com.your.product.id",
      original_transaction_id: "1133557799",
      expires_date: "2018-07-08...",
      ...
    }]
  }
}
```

Your Server

/processTransaction?userId=90000001

App Store

/verifyReceipt
1. Receive Transaction
2. Verify Authenticity
3. Update Subscription Status

Subscription Period
Update Subscription State

```
{ receipt: {
    bundle_id: "com.yourcompany.app",
    in_app: [{
        transaction_id: "1234567890",
        product_id: "com.your.product.id",
        original_transaction_id: "1133557799",
        expires_date: "2018-07-08...",
        ...
    }]
}
```
Update Subscription State

Your Server

/processTransaction?userId=90000001

```json
{ receipt: {
    bundle_id: "com.yourcompany.app",
    in_app: [{
        transaction_id: "1234567890",
        product_id: "com.your.product.id",
        original_transaction_id: "1133557799",
        expires_date: "2018-07-08...",
        ...
    }]
}
```
Update Subscription State

{ receipt: {
    bundle_id: "com.yourcompany.app",
    in_app: [{
        transaction_id: "1234567890",
        product_id: "com.your.product.id",
        original_transaction_id: "1133557799",
        expires_date: "2018-07-08...",
        ...}
    }
}
Update Subscription State

```
{ receipt: {
    bundle_id: "com.yourcompany.app",
    in_app: [{
        transaction_id: "1234567890",
        product_id: "com.your.product.id",
        original_transaction_id: "1133557799",
        expires_date: "2018-07-08...",
        ...
    }]
}
```
Update Subscription State

```
{
  receipt: {
    bundle_id: "com.yourcompany.app",
    in_app: [{
      transaction_id: "1234567890",
      product_id: "com.your.product.id",
      original_transaction_id: "1133557799",
      expires_date: "2018-07-08...",
      ...
    }]
  }
}
```
Update Subscription State

```json
{ receipt: {
    bundle_id: "com.yourcompany.app",
    in_app: [{
        transaction_id: "1234567890",
        product_id: "com.your.product.id",
        original_transaction_id: "113357799",
        expires_date: "2018-07-08...",
        ...
    }]
}
}```
Update Subscription State

```json
{ receipt: {
    bundle_id: "com.yourcompany.app",
    in_app: [{
        transaction_id: "1234567890",
        product_id: "com.your.product.id",
        original_transaction_id: "1133557799",
        expires_date: "2018-07-08...",
        ...
    }]
}
}
```

Your Server

`/processTransaction?userId=90000001`

<table>
<thead>
<tr>
<th>userId</th>
<th>originalTransactionId</th>
<th>latestExpiresDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>90000001</td>
<td>2018-07-08...</td>
<td></td>
</tr>
</tbody>
</table>

isUserSubscribed()
Update Subscription State

Your Server

/processTransaction?userId=90000001

<table>
<thead>
<tr>
<th>userId</th>
<th>originalTransactionId</th>
<th>latestExpiresDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>90000001</td>
<td>1133557799</td>
<td>2018-07-08...</td>
</tr>
</tbody>
</table>

isUserSubscribed() ✅

{ receipt: {
    bundle_id: "com.yourcompany.app",
    in_app: [{
        transaction_id: "1234567890",
        product_id: "com.your.product.id",
        original_transaction_id: "1133557799",
        expires_date: "2018-07-08...",
    }, ...
}}
Update Subscription State

```
{ receipt: {
    bundle_id: "com.yourcompany.app",
    in_app: [{
        transaction_id: "1234567890",
        product_id: "com.your.product.id",
        original_transaction_id: "1133557799",
        expires_date: "2018-07-08...",
        ...
    }]
}
```

Your Server

/processTransaction?userId=90000001

<table>
<thead>
<tr>
<th>userId</th>
<th>originalTransactionId</th>
<th>latestExpiresDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>90000001</td>
<td>1133557799</td>
<td>2018-07-08...</td>
</tr>
</tbody>
</table>

isUserSubscribed() ✓
Verifying Transactions
Finishing the transaction

Your Server
/processTransaction?userId=90000001
Verifying Transactions
Finishing the transaction

Your Server
/processTransaction?userId=90000001

{ valid: true }
switch transaction.transactionState {
    case .purchased:

        if let appStoreReceiptURL = Bundle.main.appStoreReceiptURL,
            FileManager.default.fileExists(atPath: appStoreReceiptURL.path) {

            let rawReceiptData = Data(contentsOf: appStoreReceiptURL)
            let receiptData = rawReceiptData.base64EncodedString(options: …)

            // Send up receiptString to server
            currentUser.processTransaction(receiptData) { isValid in
                if isValid {
                    // Finish the transaction
                    queue.finishTransaction(transaction)
                }
            }
        }
}
switch transaction.transactionState {
    case .purchased:
        if let appStoreReceiptURL = Bundle.main.appStoreReceiptURL,
            FileManager.default.fileExists(atPath: appStoreReceiptURL.path) {
            let rawReceiptData = Data(contentsOf: appStoreReceiptURL)
            let receiptData = rawReceiptData.base64EncodedString(options: …)

            // Send up receiptString to server
            currentUser.processTransaction(receiptData) { isValid in
                if isValid {
                    // Finish the transaction
                    queue.finishTransaction(transaction)
                }
            }
        }
switch transaction.transactionState {
    case .purchased:

        if let appStoreReceiptURL = Bundle.main.appStoreReceiptURL,
            FileManager.default.fileExists(atPath: appStoreReceiptURL.path) {

            let rawReceiptData = Data(contentsOf: appStoreReceiptURL)
            let receiptData = rawReceiptData.base64EncodedString(options: …)

            // Send up receiptString to server
            currentUser.processTransaction(receiptData) { isValid in
                if isValid {
                    // Finish the transaction
                    queue.finishTransaction(transaction)
                }
            }
        }
    }
}
Receive Transaction → Verify Authenticity → Update Subscription Status → Subscription Period
1. Receive Transaction
2. Verify Authenticity
3. Update Subscription Status

Subscription Period
<table>
<thead>
<tr>
<th>userId</th>
<th>originalTransactionId</th>
<th>latestExpiresDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>90000001</td>
<td>1133557799</td>
<td>2018-07-08...</td>
</tr>
<tr>
<td>90000002</td>
<td>4820018625</td>
<td>2018-06-02...</td>
</tr>
<tr>
<td>90000003</td>
<td>6090522426</td>
<td>2018-06-01...</td>
</tr>
<tr>
<td>90000004</td>
<td>3092890314</td>
<td>2018-06-20...</td>
</tr>
<tr>
<td>90000005</td>
<td>8426576390</td>
<td>2018-06-21...</td>
</tr>
<tr>
<td>90000006</td>
<td>4286343643</td>
<td>2018-06-03...</td>
</tr>
<tr>
<td>userId</td>
<td>originalTransactionId</td>
<td>latestExpiresDate</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>90000001</td>
<td>1133557799</td>
<td>2018-07-08...</td>
</tr>
<tr>
<td>90000002</td>
<td>4820018625</td>
<td>2018-06-02...</td>
</tr>
<tr>
<td>90000003</td>
<td>6090522426</td>
<td>2018-06-01...</td>
</tr>
<tr>
<td>90000004</td>
<td>3092890314</td>
<td>2018-06-20...</td>
</tr>
<tr>
<td>90000005</td>
<td>8426576390</td>
<td>2018-06-21...</td>
</tr>
<tr>
<td>90000006</td>
<td>4286343643</td>
<td>2018-06-03...</td>
</tr>
</tbody>
</table>
Verifying a Transaction

```
{
  status: 0,
  receipt: {
    bundle_id: "com.your.app",
    in_app: [{
      transaction_id: "1234567890",
      product_id: "com.your.product.id",
      original_transaction_id: "1133557799",
      expires_date: "2018-07-08...",
      ...
    }]
  }
}
```
Verifying a Transaction

```
{ status: 0,
  receipt: {
    bundle_id: "com.your.app",
    in_app: [{
      transaction_id: "1234567890",
      product_id: "com.your.product.id",
      original_transaction_id: "1133557799",
      expires_date: "2018-07-08...",

      ...
    },
    { transaction_id: "2233445566",
      original_transaction_id: "1133557799",
      expires_date: "2018-08-08...",

      ...
    }]
  }
}
```
Update Subscription State

```
{ receipt: {
    in_app: [{
        transaction_id: "1234567890",
        original_transaction_id: "1133557799",
        expires_date: "2018-07-08...", ...
    }, {
        transaction_id: "2233445566",
        original_transaction_id: "1133557799",
        expires_date: "2018-08-08...", ...
    }]
}
```

Your Server

/processTransaction?userId=90000001

<table>
<thead>
<tr>
<th>userId</th>
<th>originalTransactionId</th>
<th>latestExpiresDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>90000001</td>
<td>1133557799</td>
<td>2018-07-08...</td>
</tr>
</tbody>
</table>
Update Subscription State

```
{ receipt: {
   in_app: [{
         transaction_id: "1234567890",
         original_transaction_id: "1133557799",
         expires_date: "2018-07-08...", ...
     }, {
         transaction_id: "2233445566",
         original_transaction_id: "1133557799",
         expires_date: "2018-08-08...", ...
     }]
}
```

Your Server

/processTransaction?userId=90000001

<table>
<thead>
<tr>
<th>userId</th>
<th>originalTransactionId</th>
<th>latestExpiresDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>90000001</td>
<td>1133557799</td>
<td>2018-07-08...</td>
</tr>
</tbody>
</table>

isUserSubscribed()
Does the User Have an Active Subscription?
Does the User Have an Active Subscription?

Filter transactions by `original_transaction_id`
Does the User Have an Active Subscription?

Filter transactions by `original_transaction_id`

Find transaction with the latest `expires_date`

Today
Does the User Have an Active Subscription?

Filter transactions by `original_transaction_id`

Find transaction with the latest `expires_date`

Date in past shows user is not subscribed
Does the User Have an Active Subscription?

Filter transactions by `original_transaction_id`

Find transaction with the latest `expires_date`

Date in past shows user is not subscribed

Date in future shows user is subscribed
Update Subscription State

```json
{ receipt: {
  in_app: [{
    transaction_id: "1234567890",
    original_transaction_id: "1133557799",
    expires_date: "2018-07-08...",
  }, {
    transaction_id: "2233445566",
    original_transaction_id: "1133557799",
    expires_date: "2018-08-08...",
  }]
}
```
Update Subscription State

```json
{ receipt: {
    in_app: [{
        transaction_id: "1234567890",
        original_transaction_id: "1133557799",
        expires_date: "2018-07-08...
    }, {
        transaction_id: "2233445566",
        original_transaction_id: "1133557799",
        expires_date: "2018-08-08...
    }]
} }
```
Update Subscription State

```
{ receipt: {
    in_app: [{
        transaction_id: "1234567890",
        original_transaction_id: "1133557799",
        expires_date: "2018-07-08...",
    }, {
        transaction_id: "2233445566",
        original_transaction_id: "1133557799",
        expires_date: "2018-08-08...",
    }]
}
```
Update Subscription State

```
{
  receipt: {
    in_app: [{
      transaction_id: "1234567890",
      original_transaction_id: "1133557799",
      expires_date: "2018-07-08...", ...
    }, {
      transaction_id: "2233445566",
      original_transaction_id: "1133557799",
      expires_date: "2018-08-08...", ...
    }]
  }
}
```
Update Subscription State

{ receipt: {
    in_app: [{
        transaction_id: "1234567890",
        original_transaction_id: "1133557799",
        expires_date: "2018-07-08...", ...
    }, {
        transaction_id: "2233445566",
        original_transaction_id: "1133557799",
        expires_date: "2018-08-08...", ...
    }]
}}

Your Server
/processTransaction?userId=90000001

<table>
<thead>
<tr>
<th>userId</th>
<th>originalTransactionId</th>
<th>latestExpiresDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>90000001</td>
<td>1133557799</td>
<td>2018-07-08...</td>
</tr>
</tbody>
</table>

isUserSubscribed()
Update Subscription State

```
{ receipt: {
    in_app: [{
        transaction_id: "1234567890",
        original_transaction_id: "1133557799",
        expires_date: "2018-07-08...", ...
    }, {
        transaction_id: "2233445566",
        original_transaction_id: "1133557799",
        expires_date: "2018-08-08...", ...
    }]
}
```
Update Subscription State

Your Server

/processTransaction?userId=90000001

<table>
<thead>
<tr>
<th>userId</th>
<th>originalTransactionId</th>
<th>latestExpiresDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>90000001</td>
<td>1133557799</td>
<td>2018-08-08...</td>
</tr>
</tbody>
</table>

```javascript
{ receipt: {
    in_app: [{
        transaction_id: "1234567890",
        original_transaction_id: "1133557799",
        expires_date: "2018-07-08...",
    }, {
        transaction_id: "2233445566",
        original_transaction_id: "1133557799",
        expires_date: "2018-08-08...",
    }]
}
```
Finish All Transactions on Device
Including all renewal transactions

/processTransaction?userId=90000001

Your Server
Finish All Transactions on Device
Including all renewal transactions

Your Server
/processTransaction?userId=90000001

{ valid: true }
Finish All Transactions on Device
Including all renewal transactions

Your Server
/processTransaction?userId=90000001

{ valid: true }
queue.finishTransaction()
Renewal Transactions

Your Server

userId | latestExpiresDate
---|---
90000001 | 2018-07-08...

App Store

1111 2222 3333 4444

isUserSubscribed()
Renewal Transactions

- Card Number: 1111 2222 3333 4444
- App Store
- Your Server
  - userId: 90000001
  - latestExpiresDate: 2018-07-08...
- isUserSubscribed(): ✗
Renewal Transactions

Your Server

App Store

userId

latestExpiresDate

90000001

2018-07-08...

isUserSubscribed()
Renewal Transactions

userId | latestExpiresDate
-------|------------------
90000001 | 2018-08-08...

isUserSubscribed()
Multiple Platforms

Your Server

App Store

userId: 90000001
latestExpiresDate: 2018-08-08...

isUserSubscribed()
Multiple Platforms

userId: 90000001
latestExpiresDate: 2018-08-08...

App Store

Your Server

isUserSubscribed() ☑

Website
Multiple Platforms

userId: 90000001
latestExpiresDate: 2018-08-08

App Store

Your Server

userId: 90000001
latestExpiresDate: 2018-08-08

Website

isUserSubscribed()
Status Polling
Discover new transactions directly from your server
Status Polling
Discover new transactions directly from your server

Save latest version of encoded receipt data on your server
Status Polling
Discover new transactions directly from your server

Save latest version of encoded receipt data on your server

Treat receipt data like a token
Status Polling
Discover new transactions directly from your server

Save latest version of encoded receipt data on your server

Treat receipt data like a token

/verifyReceipt response also includes new transactions
Status Polling

Discover new transactions directly from your server

Save latest version of encoded receipt data on your server

Treat receipt data like a token

/verifyReceipt response also includes new transactions

• Located in the latest_receipt_info field
Status Polling

Discover new transactions directly from your server

Save latest version of encoded receipt data on your server

Treat receipt data like a token

/verifyReceipt response also includes new transactions
• Located in the latest_receipt_info field
• Unlock new subscription periods without waiting for app to launch
Verifying Renewal Transactions

/processTransaction?userId=90000001
Verifying Renewal Transactions

Your Server

/processTransaction?userId=90000001

{ receiptData: "d24Fs...kJ87dDGe3=" }
Update Subscription State
Store receipt data with user

```json
{ receipt: {
    in_app: [{
        transaction_id: "1234567890",
        original_transaction_id: "1133557799",
        expires_date: "2018-07-08...",
    }, {
        transaction_id: "2233445566",
        original_transaction_id: "1133557799",
        expires_date: "2018-08-08...",
    }]
}
```

Your Server

```
/processTransaction?userId=90000001

<table>
<thead>
<tr>
<th>userId</th>
<th>latestReceiptData</th>
<th>originalTransactionId</th>
<th>latestExpiresDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>90000001</td>
<td>1133557799</td>
<td>2018-07-08...</td>
<td></td>
</tr>
</tbody>
</table>
```
Update Subscription State

Store receipt data with user

```json
{
  receipt: {
    in_app: [{
      transaction_id: "1234567890",
      original_transaction_id: "1133557799",
      expires_date: "2018-07-08...",
    }, {
      transaction_id: "2233445566",
      original_transaction_id: "1133557799",
      expires_date: "2018-08-08...",
    }]
  }
}
```

```
{ receiptData: "d24Fs...kJ87dDGe3=" }
```

Your Server

```plaintext
/processTransaction?userId=90000001
```

<table>
<thead>
<tr>
<th>userId</th>
<th>latestReceiptData</th>
<th>originalTransactionId</th>
<th>latestExpiresDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>90000001</td>
<td></td>
<td>1133557799</td>
<td>2018-07-08...</td>
</tr>
</tbody>
</table>
Update Subscription State

Store receipt data with user

{ receipt: {
  in_app: [{
    transaction_id: "1234567890",
    original_transaction_id: "1133557799",
    expires_date: "2018-07-08...", ...
  }, {
    transaction_id: "2233445566",
    original_transaction_id: "1133557799",
    expires_date: "2018-08-08...", ...
  }]
}}

Your Server

/processTransaction?userId=90000001

userId | latestReceiptData | originalTransactionId | latestExpiresDate
-------|-------------------|-----------------------|---------------------
90000001 | d24Fs...kJ87dDGe3= | 1133557799 | 2018-07-08...
Status Polling

App Store

/verifyReceipt

Your Server

<table>
<thead>
<tr>
<th>latestReceiptData</th>
<th>latestExpiresDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>d24Fs...kJ87dDGe3=</td>
<td>2018-08-08...</td>
</tr>
</tbody>
</table>

isUserSubscribed()
Status Polling

App Store
/verifyReceipt

Your Server

<table>
<thead>
<tr>
<th>latestReceiptData</th>
<th>latestExpiresDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>d24Fs...kJ87dDGe3=</td>
<td>2018-08-08...</td>
</tr>
</tbody>
</table>

isUserSubscribed()
Status Polling

App Store

```
{ receipt-data: "d24Fs...kJ87dDGe3=",
  exclude-old-transactions: true,
  password: "416e7469204865726f" }
```

Your Server

```javascript
isUserSubscribed()
```
Status Polling

App Store

/verifyReceipt

Your Server

{ receipt-data: "d24Fs...kJ87dDGe3=",
exclude-old-transactions: true,
password: "416e7469204865726f" }

latestReceiptData
latestExpiresDate

d24Fs...kJ87dDGe3= 2018-08-08...

isUserSubscribed()
Status Polling

App Store
/verifyReceipt

Your Server
latestReceiptData
latestExpiresDate
d24Fs...kJ87dDGe3= 2018-08-08...

isUserSubscribed()
Status Polling

App Store
/verifyReceipt

Your Server

{ latest_receipt_info: [{
  expires_date: "2018-09-08...", ...
} ] }

latestReceiptData
latestExpiresDate
d24Fs...kJ87dDGe3= 2018-08-08...

isUserSubscribed()
Status Polling

{ latest_receipt_info: [{
  expires_date: "2018-09-08...", ...
} ] }

Your Server

/verifyReceipt

App Store

isUserSubscribed()
Status Polling

Your Server

```javascript
1

{ latest_receipt_info: [{
  expires_date: "2018-09-08...", ...
}]
}

isUserSubscribed() ✗
```

App Store

/verifyReceipt
Status Polling

App Store
/verifyReceipt

{ latest_receipt_info: [{
  expires_date: "2018-09-08...", ...
}] }

Your Server
latestReceiptData  latestExpiresDate
d24Fs...kJ87dDGe3=  2018-09-08...

isUserSubscribed() ✓
Multiple Platforms

- **Your Server**
  - latestReceiptData: d24Fs...kJ87dGe3=
  - latestExpiresDate: 2018-09-08...

- **App Store**
  - 1111 2222 3333 4444

- **Website**
  - `isUserSubscribed()` successful
Status Polling
Discover new transactions directly from your server
Status Polling
Discover new transactions directly from your server

New transactions **will** still appear in StoreKit on next app launch
Status Polling
Discover new transactions directly from your server

New transactions will still appear in StoreKit on next app launch
• Must verify and finish these transactions
Status Polling
Discover new transactions directly from your server

New transactions **will** still appear in StoreKit on next app launch
• Must verify and finish these transactions
• Even if your server already knows about them
Status Polling
Discover new transactions directly from your server

New transactions will still appear in StoreKit on next app launch
• Must verify and finish these transactions
• Even if your server already knows about them
• Opportunity to update latest receipt data on server
Reacting to Billing Issues
Reacting to Billing Issues

1. Observe no renewal transaction appears
Reacting to Billing Issues

1. Observe no renewal transaction appears
2. Direct user to amend their billing details
Reacting to Billing Issues

1. Observe no renewal transaction appears
2. Direct user to amend their billing details
3. Unblock user immediately when transaction occurs
Server-to-Server Notifications

originalTransactionId: 1133557799
latestExpiresDate: 2018-09-08...

1111 2222 3333 4444
App Store
Your Server
Website

isUserSubscribed()
Server-to-Server Notifications

originalTransactionId: 1133557799
latestExpiresDate: 2018-09-08...

1111 2222 3333 4444

App Store

Your Server

Website

isUserSubscribed()
Server-to-Server Notifications

Your Server

originalTransactionId: 1133557799
latestExpiresDate: 2018-09-08

App Store

1111 2222 3333 4444

Website

isUserSubscribed()
Sorry! Your subscription has expired.
Server-to-Server Notifications

Your Server

originalTransactionId  latestExpiresDate
1133557799  2018-09-08...

isUserSubscribed()  ×

Sorry! Your subscription has expired

Website

Close
Server-to-Server Notifications

Original Transaction Id: 1133557799
Latest Expires Date: 2018-09-08...

isUserSubscribed()
Server-to-Server Notifications

App Store

Your Server

<table>
<thead>
<tr>
<th>originalTransactionId</th>
<th>latestExpiresDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1133557799</td>
<td>2018-09-08...</td>
</tr>
</tbody>
</table>

isUserSubscribed() ✗
Server-to-Server Notifications

```json
{ latest_transaction_info: [{
    original_transaction_id: "1133557799",
    expires_date: "2018-10-08...
] }
}
```
Server-to-Server Notifications

{ latest_transaction_info: [{
    original_transaction_id: "1133557799",
    expires_date: "2018-10-08…"
} ] }

isUserSubscribed()
Server-to-Server Notifications

{ latest_transaction_info: [{
    original_transaction_id: "1133557799",
    expires_date: "2018-10-08..."
} ] }
## Server-to-Server Notifications

<table>
<thead>
<tr>
<th>originalTransactionId</th>
<th>latestExpiresDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1133557799</td>
<td>2018-09-08...</td>
</tr>
</tbody>
</table>

Your Server

```json
{ latest_transaction_info: [{
    original_transaction_id: "1133557799",
    expires_date: "2018-10-08..."
}] }
```
Server-to-Server Notifications

App Store

\{ latest_transaction_info: [{
   original_transaction_id: "1133557799",
   expires_date: "2018-10-08..."
}]
\}

Your Server

<table>
<thead>
<tr>
<th>originalTransactionId</th>
<th>latestExpiresDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1133557799</td>
<td>2018-09-08...</td>
</tr>
</tbody>
</table>

isUserSubscribed()
Server-to-Server Notifications

{ latest_transaction_info: [{
  original_transaction_id: "1133557799",
  expires_date: "2018-10-08..." ...
} ]

isUserSubscribed() ✗
Server-to-Server Notifications

```
{ latest_transaction_info: [{
  original_transaction_id: "1133557799",
  expires_date: "2018-10-08..." ...
}]
```

App Store

Your Server

<table>
<thead>
<tr>
<th>originalTransactionId</th>
<th>latestExpiresDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1133557799</td>
<td>2018-10-08...</td>
</tr>
</tbody>
</table>

isUserSubscribed()
Server-to-Server Notifications

Your Server

originalTransactionId: 1133557799
latestExpiresDate: 2018-10-08

isUserSubscribed():

Yes

App Store

Sorry! Your subscription has expired

Website

Close
Server-to-Server Notifications

Your Server

originalTransactionId: 1133557799
latestExpiresDate: 2018-10-08

App Store

Website

isUserSubscribed()
Server-to-Server Notifications
Server-to-Server Notifications

Status URL in App Store Connect
Server-to-Server Notifications

Status URL in App Store Connect

HTTPS POST to your server for status changes
Server-to-Server Notifications

Status URL in App Store Connect

HTTPS POST to your server for status changes

Includes `latest_transaction_info` for the transaction in question
In-App Experience
<table>
<thead>
<tr>
<th>userId</th>
<th>originalTransactionId</th>
<th>latestExpiresDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>90000001</td>
<td>1133557799</td>
<td>2018-07-08...</td>
</tr>
<tr>
<td>90000002</td>
<td>4820018625</td>
<td>2018-06-02...</td>
</tr>
<tr>
<td>90000003</td>
<td>6090522426</td>
<td>2018-06-01...</td>
</tr>
<tr>
<td>90000004</td>
<td>3092890314</td>
<td>2018-06-20...</td>
</tr>
<tr>
<td>90000005</td>
<td>8426576390</td>
<td>2018-06-21...</td>
</tr>
<tr>
<td>90000006</td>
<td>4286343643</td>
<td>2018-06-03...</td>
</tr>
</tbody>
</table>
Creating User Accounts
Creating User Accounts

Offer in-app purchase before account creation
Creating User Accounts

Offer in-app purchase before account creation
• Better user experience
Creating User Accounts

Offer in-app purchase before account creation

• Better user experience
• Higher conversion
Offer in-app purchase before account creation

• Better user experience
• Higher conversion

Use anonymous account

<table>
<thead>
<tr>
<th>userId</th>
<th>email</th>
<th>originalTransactionId</th>
<th>latestReceiptData</th>
<th>latestExpiresDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>90000001</td>
<td>NULL</td>
<td>1133557799</td>
<td>d24Fs...kJ87dDGe3=</td>
<td>2018-10-08...</td>
</tr>
</tbody>
</table>
Offer in-app purchase before account creation

• Better user experience
• Higher conversion

Use anonymous account

• Rely on original_transaction_id to associate multiple devices
Creating User Accounts

Offer in-app purchase before account creation
• Better user experience
• Higher conversion

Use anonymous account
• Rely on `original_transaction_id` to associate multiple devices
• Optionally take user through account “creation” later

<table>
<thead>
<tr>
<th>userId</th>
<th>email</th>
<th>originalTransactionId</th>
<th>latestReceiptData</th>
<th>latestExpiresDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>90000001</td>
<td>NULL</td>
<td>1133557799</td>
<td>d24F...kJ87dDGy3=</td>
<td>2018-10-08...</td>
</tr>
</tbody>
</table>
Offer in-app purchase before account creation
• Better user experience
• Higher conversion

Use anonymous account
• Rely on `original_transaction_id` to associate multiple devices
• Optionally take user through account “creation” later

Creating User Accounts

<table>
<thead>
<tr>
<th>userId</th>
<th>email</th>
<th>originalTransactionId</th>
<th>latestReceiptData</th>
<th>latestExpiresDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>90000001</td>
<td>johnappleseed@...</td>
<td>1133557799</td>
<td>d24Fs...kJ87dDGe3=</td>
<td>2018-10-08...</td>
</tr>
</tbody>
</table>
Pro Subscription for Forest Explorer
Detailed satellite imagery

1 year for $19.99, then $39.99/year after trial.

INSTALL & SUBSCRIBE
In-App Purchases
Introductory Pricing
Determining user eligibility
Introductory Pricing
Determining user eligibility

Need to show correct price to your user
Introductory Pricing
Determining user eligibility

Need to show correct price to your user

Check transactions to see if discount has been used
Introductory Pricing
Determining user eligibility

Need to show correct price to your user

Check transactions to see if discount has been used

{ …
  transaction_id: "9988776655",
  product_id: "com.your.product.id",
  is_trial_period: "false",
  is_in_intro_offer_period: "true" },

Introductory Pricing
Determining user eligibility

Need to show correct price to your user

Check transactions to see if discount has been used

```json
{...
  transaction_id: "9988776655",
  product_id: "com.your.product.id",
  is_trial_period: "false",
  is_in_intro_offer_period: "true"
}
```
Introductory Pricing

Determining user eligibility

Need to show correct price to your user

Check transactions to see if discount has been used

```
{ ... transaction_id: "9988776655",
  product_id: "com.your.product.id",
  is_trial_period: "false",
  is_in_intro_offer_period: "true"
},
```
// Displaying the correct product price for “Product A”

func fetchConsumedGroupDiscounts() {
    let consumedProducts = currentUser.consumedProductDiscounts
    let productsRequest = SKProductsRequest(productIdentifiers: Set(consumedProducts))
    productsRequest.delegate = self
    productsRequest.start()
}

// MARK: - SKProductsRequestDelegate

func productsRequest(_ request: SKProductsRequest, didReceive response: SKProductsResponse) {
    for product in response.products {
        currentUser.consumedGroupDiscounts.append(product.subscriptionGroupIdentifier)
    }
}
func fetchConsumedGroupDiscounts() {
    let consumedProducts = currentUser.consumedProductDiscounts
    let productsRequest = SKProductsRequest(productIdentifiers: Set(consumedProducts))
    productsRequest.delegate = self
    productsRequest.start()
}

// MARK: - SKProductsRequestDelegate

func productsRequest(_ request: SKProductsRequest, didReceive response: SKProductsResponse) {
    for product in response.products {
        currentUser.consumedGroupDiscounts.append(product.subscriptionGroupIdentifier)
    }
}
/ Displaying the correct product price for “Product A”

```swift
func fetchConsumedGroupDiscounts() {
    let consumedProducts = currentUser.consumedProductDiscounts
    let productsRequest = SKProductsRequest(productIdentifiers: Set(consumedProducts))
    productsRequest.delegate = self
    productsRequest.start()
}

// MARK: - SKProductsRequestDelegate

func productsRequest(_ request: SKProductsRequest, didReceive response: SKProductsResponse) {
    for product in response.products {
        currentUser.consumedGroupDiscounts.append(product.subscriptionGroupIdentifier)
    }
}
```
// Displaying the correct product price for “Product A”

func fetchConsumedGroupDiscounts() {
    let consumedProducts = currentUser.consumedProductDiscounts
    let productsRequest = SKProductsRequest(productIdentifiers: Set(consumedProducts))
    productsRequest.delegate = self
    productsRequest.start()
}

// MARK: - SKProductsRequestDelegate

func productsRequest(_: SKProductsRequest, didReceive response: SKProductsResponse) {
    for product in response.products {
        currentUser.consumedGroupDiscounts.append(product.subscriptionGroupIdentifier)
    }
}
func fetchConsumedGroupDiscounts() {
    let consumedProducts = currentUser.consumedProductDiscounts
    let productsRequest = SKProductsRequest(productIdentifiers: Set(consumedProducts))
    productsRequest.delegate = self
    productsRequest.start()
}

// MARK: - SKProductsRequestDelegate

func productsRequest(_ request: SKProductsRequest, didReceive response: SKProductsResponse) {
    for product in response.products {
        currentUser.consumedGroupDiscounts.append(product.subscriptionGroupIdentifier)
    }
}
// Displaying the correct product price for “Product A”

func fetchConsumedGroupDiscounts() {
    let consumedProducts = currentUser.consumedProductDiscounts
    let productsRequest = SKProductsRequest(productIdentifiers: Set(consumedProducts))
    productsRequest.delegate = self
    productsRequest.start()
}

// MARK: - SKProductsRequestDelegate

func productsRequest(_ request: SKProductsRequest, didReceive response: SKProductsResponse) {
    for product in response.products {
        currentUser.consumedGroupDiscounts.append(product.subscriptionGroupIdentifier)
    }
}
// Displaying the correct product price for “Product A”

let price: NSDecimalNumber
let priceLocale: Locale
if currentUser.consumedGroupDiscounts.contains(productA.subscriptionGroupIdentifier) {
    price = productA.price
    priceLocale = productA.priceLocale
} else {
    price = productA.introductoryPrice.price
    priceLocale = productA.introductoryPrice.priceLocale
}

let formatter = NumberFormatter()
formatter.numberStyle = .currency
formatter.locale = priceLocale
let formattedString = formatter.string(from: price)
let price: NSDecimalNumber
let priceLocale: Locale
if currentUser.consumedGroupDiscounts.contains(productA.subscriptionGroupIdentifier) {
    price = productA.price
    priceLocale = productA.priceLocale
} else {
    price = productA.introductionPrice.price
    priceLocale = productA.introductionPrice.priceLocale
}

let formatter = NumberFormatter()
formatter.numberStyle = .currency
formatter.locale = priceLocale
let formattedString = formatter.string(from: price)
let price: NSDecimalNumber
let priceLocale: Locale

if currentUser.consumedGroupDiscounts.contains(productA.subscriptionGroupIdentifier) {
    price = productA.price
    priceLocale = productA.priceLocale

} else {
    price = productA.introductoryPrice.price
    priceLocale = productA.introductoryPrice.priceLocale

}

let formatter = NumberFormatter()
formatter.numberStyle = .currency
formatter.locale = priceLocale
let formattedString = formatter.string(from: price)
// Displaying the correct product price for “Product A”

let price: NSDecimalNumber
let priceLocale: Locale
if currentUser.consumedGroupDiscounts.contains(productA.subscriptionGroupIdentifier) {
    price = productA.price
    priceLocale = productA.priceLocale
} else {
    price = productA.introductoryPrice.price
    priceLocale = productA.introductoryPrice.priceLocale
}

let formatter = NumberFormatter()
formatter.numberStyle = .currency
formatter.locale = priceLocale
let formattedString = formatter.string(from: price)
// Displaying the correct product price for “Product A”

let price: NSDecimalNumber
let priceLocale: Locale
if currentUser.consumedGroupDiscounts.contains(productA.subscriptionGroupIdentifier) {
    price = productA.price
    priceLocale = productA.priceLocale
} else {
    price = productA.introductoryPrice.price
    priceLocale = productA.introductoryPrice.priceLocale
}

let formatter = NumberFormatter()
formatter.numberStyle = .currency
formatter.locale = priceLocale
let formattedString = formatter.string(from: price)
<table>
<thead>
<tr>
<th>Event Description</th>
<th>Hall</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best Practices and What’s New with In-App Purchases</td>
<td>1</td>
<td>Tuesday</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2:00PM</td>
</tr>
<tr>
<td>What’s New in App Store Connect</td>
<td>3</td>
<td>Wednesday</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5:00PM</td>
</tr>
</tbody>
</table>
Upgrade/Downgrade Subscriptions
Inside your app
Upgrade/Downgrade Subscriptions

Inside your app

Just like selling an initial subscription
Upgrade/Downgrade Subscriptions
Inside your app

Just like selling an initial subscription

Must be a subscription in the same subscription group
Upgrade/Downgrade Subscriptions
Inside your app

Just like selling an initial subscription

Must be a subscription in the same subscription group

```swift
let payment = SKPayment(product: product)
SKPaymentQueue.default().add(payment)
```
Upgrade/Downgrade Subscriptions
Inside your app

Just like selling an initial subscription

Must be a subscription in the same subscription group

```swift
let payment = SKPayment(product: product)
SKPaymentQueue.default().add(payment)
```

StoreKit handles the upgrade/downgrade
Subscription Management
In App Store account settings
Subscription Management
In App Store account settings

Provide a link to “Manage Your Subscription”
• Upgrades/downgrades
• Cancellation
Subscription Management
In App Store account settings

Provide a link to “Manage Your Subscription”
• Upgrades/downgrades
• Cancellation

Link on the In-App Purchase Programming Guide
Subscription Management
In App Store account settings

Provide a link to “Manage Your Subscription”
• Upgrades/downgrades
• Cancellation

Link on the In-App Purchase Programming Guide

https://apps.apple.com/account/subscriptions
Reducing Subscriber Churn

Michael Gargas, Technical Advocate, App Store Operations
Reducing Subscriber Churn

Involuntary churn

Voluntary churn

Winback
Involuntary Churn
Involuntary Churn

- Subscription Period
- Subscription Period
- Subscription Period

- Transaction
- Transaction
- Payment Issue

(time)
Involuntary Churn

Billing retry

Updated failed payment logic
• Expanded retry duration
• New retry strategies
• Tuning
July 13, 2017

After Billing Retry
Performance

65%

Pre Launch Post Launch

Recovery Rate Involuntary Churn
Pre Launch

Recovery Rate: 25.95%

Involuntary Churn: 60.57%

Post Launch

Recovery Rate: 60.57%

Involuntary Churn: +34.62%
Performance

Pre Launch Post Launch

Recovery Rate

25.95% 25.95% 60.57%

Involuntary Churn

6.46% 6.46% 4.23%

Performance

-2.23% +34.62%

Performance Graph:

- Recovery Rate: Pre Launch 25.95% Post Launch 60.57%
- Involuntary Churn: Pre Launch 6.46% Post Launch 4.23%

Overall change:

+34.62%
Incremental Subscriptions Recovered

<table>
<thead>
<tr>
<th></th>
<th>FY17 Q4</th>
<th>FY18 Q1</th>
<th>FY18 Q2</th>
</tr>
</thead>
<tbody>
<tr>
<td>5M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3M</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Incremental Subscriptions Recovered

FY17 Q4
FY18 Q1
FY18 Q2
12 Million
Subscriptions Recovered
Minimizing Involuntary Churn

Leverage subscription receipt fields

Implement a grace period

Customer messaging
Example Subscription

```
in_app: [{
    "purchase_date": "2018-03-26...",
    "expires_date": "2018-04-26...",
}]
```
Example Subscription

in_app: [{
  "purchase_date": "2018-03-26...",
  "expires_date": "2018-04-26...",
}]

Mar 26

Apr 26
Billing Retry
Optimization strategies

pending_renewal_info: [{
   "is_in_billing_retry_period": "1",
   "original_transaction_id": "1133557799",
   "product_id": "plus_subscription_999_6m",
   "auto_renew_status": "1", ...
}]

/过程Transaction?userId=90000001

<table>
<thead>
<tr>
<th>userId</th>
<th>originalTransactionId</th>
<th>inBillingRetry</th>
</tr>
</thead>
<tbody>
<tr>
<td>90000001</td>
<td>1133557799</td>
<td>1</td>
</tr>
</tbody>
</table>
Billing Retry
Optimization strategies

pending_renewal_info: [{
  "is_in_billing_retry_period": "1",
  "original_transaction_id": "113357799",
  "product_id": "plus_subscription_999_6m",
  "auto_renew_status": "1", ...
}]

/processTransaction?userId=90000001

<table>
<thead>
<tr>
<th>userId</th>
<th>originalTransactionId</th>
<th>inBillingRetry</th>
</tr>
</thead>
<tbody>
<tr>
<td>90000001</td>
<td>1133557799</td>
<td>1</td>
</tr>
</tbody>
</table>
Example Subscription

pending_renewal_info: [{
   "original_transaction_id": "113357799",
   "product_id": "plus_subscription_999_6m",
   "auto_renew_status": "1",
   "is_in_billing_retry_period": "1",...
}]
Grace Period

Free subscription access
• While in a billing retry state
• Before a subscriber has churned

Improve recovery
Example Grace Period

pending_renewal_info: [{
  "original_transaction_id": "113357799",
  "product_id": "plus_subscription_999_6m",
  "auto_renew_status": "1",
  "is_in_billing_retry_period": "1",
}]

Mar 26

Apr 26
Example Grace Period

pending_renewal_info: [
  {
    "original_transaction_id": "1133557799",
    "product_id": "plus_subscription_999_6m",
    "auto_renew_status": "1",
    "is_in_billing_retry_period": "1",
    ...
  }
]
Example Grace Period

Grace period = Apr 26 + 3

pending_renewal_info: [{
  "original_transaction_id": "113357799",
  "product_id": "plus_subscription_999_6m",
  "auto_renew_status": "1",
  "is_in_billing_retry_period": "1",
}]}
Customer Messaging

Messaging
• Update payment methods
• Restate the value proposition

Offer limited service
Oops! Something is wrong.

Hey John - there seems to be a billing problem with your account. If we can't resolve it, this would mean that you lose access to Peak Pro.

To fix this issue, please make sure your credit card details are up to date and that you are using a valid credit card.

Update Details
Customer Messaging

Edit billing information
• https://apps.apple.com/account/billing

Manage subscriptions
• https://apps.apple.com/account/subscriptions
<table>
<thead>
<tr>
<th>Grace Period</th>
<th>Customer recovery</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>16%</td>
<td></td>
</tr>
<tr>
<td>12%</td>
<td></td>
</tr>
<tr>
<td>8%</td>
<td></td>
</tr>
<tr>
<td>4%</td>
<td></td>
</tr>
<tr>
<td>0%</td>
<td></td>
</tr>
</tbody>
</table>
Grace Period
Customer recovery

Billing Retry Recovery Timing

<table>
<thead>
<tr>
<th>Days</th>
<th>0%</th>
<th>4%</th>
<th>8%</th>
<th>12%</th>
<th>16%</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example Grace Period

Grace period = Apr 26 + 3

pending_renewal_info: [{
  "original_transaction_id": "113357799",
  "product_id": "plus_subscription_999_6m",
  "auto_renew_status": "1",
  "is_in_billing_retry_period": "1",
}]}
Example Subscription

latest_receipt_info: [{
  "purchase_date": "2018-05-15...",
  "expires_date": "2018-06-15...",...
}]

Retry attempt successful
Example Subscription

latest_receipt_info: [{
    "purchase_date": "2018-05-15...",
    "expires_date": "2018-06-15...",...
}]

Retry attempt successful
Server-to-Server Notifications

{ notification_type: "RENEWAL", ... }
Server-to-Server Notifications

{ notification_type: "RENEWAL", ... }

You're all set!
Voluntary Churn
Voluntary Churn

- Subscription Period
- Subscription Period
- Subscription Period

Transaction
Transaction
Cancellation
Minimizing Voluntary Churn

Implement status polling

Offer attractive alternative subscriptions
Status Polling

Will my subscriber churn?

Has my subscriber renewed?
When to Status Poll

- Subscription Period
- Subscription Period

expires_date
When to Status Poll

Subscription Period

.expires_date

Subscription Period
Additional Receipt Fields
Storing subscriber status fields
Additional Receipt Fields

Storing subscriber status fields

Save decoded JSON from /verifyReceipt
### Additional Receipt Fields

Storing subscriber status fields

Save decoded JSON from `/verifyReceipt`

<table>
<thead>
<tr>
<th>userId</th>
<th>originalTransactionIdentifier</th>
<th>latestReceiptData</th>
<th>latestExpiresDate</th>
<th>latestDecodedReceipt</th>
</tr>
</thead>
<tbody>
<tr>
<td>90000001</td>
<td>1234567890</td>
<td>d24Fs...kJ87dGe3=</td>
<td>2018-08-08...</td>
<td><code>{ ... auto_renew_status: 1 ...}</code></td>
</tr>
</tbody>
</table>
Additional Receipt Fields

Storing subscriber status fields

Save decoded JSON from `/verifyReceipt`

Or parse out specific fields

<table>
<thead>
<tr>
<th>userId</th>
<th>originalTransactionIdentifier</th>
<th>latestReceiptData</th>
<th>latestExpiresDate</th>
<th>inBillingRetry</th>
</tr>
</thead>
<tbody>
<tr>
<td>90000001</td>
<td>1234567890</td>
<td>d24Fs...kJ87dDGe3=</td>
<td>2018-08-08...</td>
<td>1</td>
</tr>
</tbody>
</table>
Additional Receipt Fields
Storing subscriber status fields

Save decoded JSON from /verifyReceipt

Or parse out specific fields

<table>
<thead>
<tr>
<th>userId</th>
<th>originalTransactionIdentifier</th>
<th>latestReceiptData</th>
<th>latestExpiresDate</th>
<th>inBillingRetry</th>
</tr>
</thead>
</table>
Auto Renew Status

Voluntary churn

```
pending_renewal_info: [{
    "is_in_billing_retry_period": "1",
    "original_transaction_id": "1133557799",
    "product_id": "plus_subscription_999_6m",
    "auto_renew_status": "1", ...
}]
```

Your Server

/processTransaction?userId=90000001

<table>
<thead>
<tr>
<th>userId</th>
<th>originalTransactionId</th>
<th>autoRenewStatus</th>
</tr>
</thead>
<tbody>
<tr>
<td>90000001</td>
<td>1133557799</td>
<td>1</td>
</tr>
</tbody>
</table>
Auto Renew Status
Voluntary churn

pending_renewal_info: [{
    "is_in_billing_retry_period": "1",
    "original_transaction_id": "1133557799",
    "product_id": "plus_subscription_999_6m",
    "auto_renew_status": "1", ...
}]

/processTransaction?userId=90000001

<table>
<thead>
<tr>
<th>userId</th>
<th>originalTransactionId</th>
<th>autoRenewStatus</th>
</tr>
</thead>
<tbody>
<tr>
<td>90000001</td>
<td>1133557799</td>
<td>1</td>
</tr>
</tbody>
</table>
Example Subscription

Mar 26 to Apr 26
Example Subscription

Mar 26 - Apr 26

in_app: [{
    "purchase_date": "2018-03-26...",
    "expires_date": "2018-04-26...",...
}]

Example Subscription

Mar 26 - Apr 26

### in_app:

```json
{
  "purchase_date": "2018-03-26",
  "expires_date": "2018-04-26"
}
```
Status Polling
Update auto renew status from server

App Store
/verifyReceipt

Your Server
latestReceiptData  autoRenew
   d24Fs...kJ87dDGe3=  1
Status Polling
Update auto renew status from server

```
{ pending_renewal_info: [{  
"auto_renew_status": "0", ...
} ] }
```

```
latestReceiptData
```

```
autoRenew
```

```
d24Fs...kJ87dDGe3= 1
```
Status Polling
Update auto renew status from server

Your Server
latestReceiptData
autoRenew

d24Fs...kJ87dDGe3= 1

App Store
/verifyReceipt

{ pending_renewal_info: [{
  "auto_renew_status": "0", ... 
}] }
Status Polling
Update auto renew status from server

App Store
/verifyReceipt

Your Server

```
{ pending_renewal_info: [{
  "auto_renew_status": "0",
  ...
} ] }
```

```
<table>
<thead>
<tr>
<th>latestReceiptData</th>
<th>autoRenew</th>
</tr>
</thead>
<tbody>
<tr>
<td>d24Fs...kJ87dDGe3=</td>
<td>0</td>
</tr>
</tbody>
</table>
```
Example Subscription

- Status poll

Mar 26 - Apr 26

pending_renewal_info: [{
  "original_transaction_id": "113357799",
  "product_id": "plus_subscription_999_6m",
  "auto_renew_status": "0",
  ...
}]

Example Subscription

Status poll

Mar 26 - Apr 26

pending_renewal_info: [{
  "original_transaction_id": "113357799",
  "product_id": "plus_subscription_999_6m",
  "auto_renew_status": "0",
}]

Here's a reason to come back!

Go Pro again for 60% less
Here's a reason to come back!

Go Pro again for 60% less
Auto Renew Preference

```json
pending_renewal_info: [{
    "original_transaction_id": "1133557799",
    "product_id": "plus_subscription_999_6m",
    "auto_renew_status": "1",
    "auto_renew_product_id": "com.your.product.id",
}]
```
Auto Renew Preference

pending_renewal_info: [{
    "original_transaction_id": "1133557799",
    "product_id": "plus_subscription_999_6m",
    "auto_renew_status": "1",
    "auto_renew_product_id": "com.your.product.id",
}]

/processTransaction?userId=90000001

<table>
<thead>
<tr>
<th>userId</th>
<th>originalTransactionId</th>
<th>autoRenewProductID</th>
</tr>
</thead>
<tbody>
<tr>
<td>90000001</td>
<td>1133557799</td>
<td>com.your.product.id</td>
</tr>
</tbody>
</table>
Example Subscription

[Diagram showing a timeline from Mar 26 to Apr 26 with a diamond marking Apr 1 for Downgrade subscription level]

```
pending_renewal_info: [{
  "original_transaction_id": "113357799",
  "product_id": "plus_subscription_999_6m",
  "auto_renew_status": "1",
  "auto_renew_product_id": "com.your.product.id", ...
}]
```
Example Subscription

Downgrade subscription level

Mar 26

Apr 1

Apr 26

pending_renewal_info: [{
    "original_transaction_id": "113357799",
    "product_id": "plus_subscription_999_6m",
    "auto_renew_status": "1",
    "auto_renew_product_id": "com.your.product.id",
}]

Server-to-Server Notifications

{ notification_type: "DID_CHANGE_RENEWAL_PREFERENCE", ... }
Server-to-Server Notifications

We’ve noticed you selected another product

```
{ notification_type: "DID_CHANGE_RENEWAL_PREFERENCE", ... }
```
Winback

Reengage after subscribers churn

• Resubscription offers
• Customer surveys
Example Subscription

Mar 26 - Apr 26

```javascript
in_app: [{
    "purchase_date": "2018-03-26...",
    "expires_date": "2018-04-26...",
}]
```
Example Subscription

Cancelled via AppleCare

Mar 26 to Apr 26

in_app: [{
    "purchase_date": "2018-03-26...",
    "expires_date": "2018-04-26...",
    "cancellation_date": "2018-04-04...",
    ...
}]}
Example Subscription

Cancelled via AppleCare

Mar 26       Apr 4       Apr 26

in_app: [{
  "purchase_date": "2018-03-26...",
  "expires_date": "2018-04-26...",
  "cancellation_date": "2018-04-04...",
  ...
}]
Server-to-Server Notifications

{ notification_type: "CANCEL", ... }
Server-to-Server Notifications

{ notification_type: "CANCEL", ... }

App Store

Your Server

Please see alternative subscription offers

Close
Expiration Intent

```json
pending_renewal_info: [{
  "original_transaction_id": "1133557799",
  "product_id": "plus_subscription_999_6m",
  "auto_renew_status": "0",
  "expiration_intent": "1",
}]
```
Expiration Intent

```javascript
pending_renewal_info: [{
  "original_transaction_id": "1133557799",
  "product_id": "plus_subscription_999_6m",
  "auto_renew_status": "0",
  "expiration_intent": "1",
}]
```

Your Server

```
/processTransaction?userId=90000001
```

<table>
<thead>
<tr>
<th></th>
<th>originalTransactionId</th>
<th>expirationIntent</th>
</tr>
</thead>
<tbody>
<tr>
<td>90000001</td>
<td>1133557799</td>
<td>1</td>
</tr>
</tbody>
</table>
Example Subscription

Cancelled via AppleCare

Mar 26

Apr 4

Apr 26

in_app: [{
  "purchase_date": "2018-03-26...",
  "expires_date": "2018-04-26...",
  "cancellation_date": "2018-04-04...",
  "expiration_intent": "1", ...
}]

Example Subscription

Cancelled via AppleCare

Mar 26  Apr 4  Apr 26

in_app: [{
  "purchase_date": "2018-03-26...",
  "expires_date": "2018-04-26...",
  "cancellation_date": "2018-04-04...",
  "expiration_intent": "1",
}]

Winback Lapsed Subscribers
Voluntary churn

Survey lapsed subscribers
Show alternative subscription products
It's been too long
Get 60% off Pro
Winback Lapsed Subscribers

Involuntary churn

Show alternative subscription products

Persistent messaging in-app

Limited subscription experience
Upgrade to Tinder Gold

See Who Likes You
Match with them instantly

Upgrade to Tinder Gold for an extra

$6.33/mo
Total $18.83/mo

CONTINUE

NO THANKS
Reducing Subscriber Churn

Summary
Reducing Subscriber Churn

Summary

Leverage subscription specific receipt fields
Reducing Subscriber Churn

Summary

Leverage subscription specific receipt fields

Implement status polling
Reducing Subscriber Churn
Summary

Leverage subscription specific receipt fields
Implement status polling
Implement customer messaging
Reducing Subscriber Churn

Summary

Leverage subscription specific receipt fields
Implement status polling
Implement customer messaging
Present contextual subscription offers
Analytics and Reporting

Pete Hare, Engineering Manager, App Store
Active Paid Subscriptions: 221K (0.48% increase)

Daily Subscription Units: 8.60K (1.1% increase)

Daily Subscription Sales: $87.5 (1.3% increase)

Subscription Activity chart showing trends from July 12 to August 6.

Retention and Conversion Rates chart showing trends with line graphs.
Retention Rate: 63% (▲ 1.1%)
Introductory Price Conversion Rate: 56% (▲ 2.3%)
Subscriptions with a Rate After 1 Year: 35% (▲ 1.6%)

Retention by Start Month

Retention Trends
Retention Rate: 63% (↑ 1.1%), Introductory Price Conversion Rate: 56% (↑ 2.3%), Subscriptions with a Rate After 1 Year: 35% (↑ 1.6%).

Retention by Start Month:

Retention Trends:

Subscription Start Month: April 2017.
<table>
<thead>
<tr>
<th>Retention Trends</th>
<th>Consecutive Renewal Periods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscription</td>
<td>Start Month</td>
</tr>
<tr>
<td>Skinner Started</td>
<td>1</td>
</tr>
<tr>
<td>Rolling 3 Year Average</td>
<td>61</td>
</tr>
<tr>
<td>Apr 2017</td>
<td>26.9K</td>
</tr>
<tr>
<td>May 2017</td>
<td>19.3K</td>
</tr>
<tr>
<td>Jun 2017</td>
<td>20.1K</td>
</tr>
<tr>
<td>Jul 2017</td>
<td>21.3K</td>
</tr>
<tr>
<td>Aug 2017</td>
<td>20.2K</td>
</tr>
<tr>
<td>Sep 2017</td>
<td>15.4K</td>
</tr>
<tr>
<td>Oct 2017</td>
<td>19.2K</td>
</tr>
<tr>
<td>Nov 2017</td>
<td>17.3K</td>
</tr>
<tr>
<td>Dec 2017</td>
<td>20.3K</td>
</tr>
<tr>
<td>Jan 2018</td>
<td>35.3K</td>
</tr>
<tr>
<td>Feb 2018</td>
<td>26.1K</td>
</tr>
<tr>
<td>Mar 2018</td>
<td>23.5K</td>
</tr>
</tbody>
</table>
## Retention Trends

<table>
<thead>
<tr>
<th>Subscription Start Month</th>
<th>Subscriptions Started</th>
<th>Consecutive Renewal Periods</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Rolling 3 Year Average</td>
<td>61%</td>
<td>49%</td>
</tr>
<tr>
<td>Apr 2017</td>
<td>63%</td>
<td>51%</td>
</tr>
<tr>
<td>May 2017</td>
<td>67%</td>
<td>54%</td>
</tr>
<tr>
<td>Jun 2017</td>
<td>57%</td>
<td>49%</td>
</tr>
<tr>
<td>Jul 2017</td>
<td>59%</td>
<td>52%</td>
</tr>
<tr>
<td>Aug 2017</td>
<td>57%</td>
<td>51%</td>
</tr>
<tr>
<td>Sep 2017</td>
<td>59%</td>
<td>43%</td>
</tr>
<tr>
<td>Oct 2017</td>
<td>61%</td>
<td>45%</td>
</tr>
<tr>
<td>Nov 2017</td>
<td>51%</td>
<td>34%</td>
</tr>
<tr>
<td>Dec 2017</td>
<td>58%</td>
<td>41%</td>
</tr>
<tr>
<td>Jan 2018</td>
<td>56%</td>
<td>49%</td>
</tr>
<tr>
<td>Feb 2018</td>
<td>54%</td>
<td>49%</td>
</tr>
<tr>
<td>Mar 2018</td>
<td>58%</td>
<td>51%</td>
</tr>
</tbody>
</table>

44% of subscriptions that started on August 2018 are active after 6 renewal periods.
App Store Connect Reports
App Store Connect Reports

Available through the new App Store Connect API
App Store Connect Reports

Available through the new App Store Connect API

Report data available daily
App Store Connect Reports

Available through the new App Store Connect API

Report data available daily

Can be imported for further analysis
App Store Connect Reports

Available through the new App Store Connect API

Report data available daily

Can be imported for further analysis

Automating App Store Connect  Hall 3  Thursday 3:00PM
Receipts vs Reports
Receipts vs Reports

App Store Receipts

• Validating StoreKit transactions
• Updating user subscription state
• Understand individual subscriber behavior
Receipts vs Reports

App Store Receipts
• Validating StoreKit transactions
• Updating user subscription state
• Understand individual subscriber behavior

App Store Connect Reports
• Aggregated sales activity
• Subscriptions activity
• Financial payments
Summary

Server side state management offers more flexibility
Summary

Server side state management offers more flexibility

Use notifications from the App Store
Summary

Server side state management offers more flexibility

Use notifications from the App Store

Offer introductory pricing
Server side state management offers more flexibility
Use notifications from the App Store
Offer introductory pricing
Reduce churn with simple messaging
Summary

Server side state management offers more flexibility

Use notifications from the App Store

Offer introductory pricing

Reduce churn with simple messaging

Win back users with alternative subscription options
Summary

Server side state management offers more flexibility

Use notifications from the App Store

Offer introductory pricing

Reduce churn with simple messaging

Win back users with alternative subscription options

New reporting tools in App Store Connect
More Information

https://developer.apple.com/wwdc18/705

<table>
<thead>
<tr>
<th>StoreKit / Subscriptions Lab</th>
<th>Technology Lab 12</th>
<th>Tuesday 4:00PM</th>
</tr>
</thead>
<tbody>
<tr>
<td>StoreKit / Subscriptions Lab</td>
<td>Technology Lab 12</td>
<td>Thursday 9:00AM</td>
</tr>
</tbody>
</table>