

# In-App Purchase for iOS and Mac OS X

Session 510

**Max Müller**

Director, iTunes Store, Digital Supply Chain Engineering

These are confidential sessions—please refrain from streaming, blogging, or taking pictures

# Overview

- Welcome Mac OS X Developers
  - In-App Purchase, new for Lion



# In-App Purchase for iOS and Mac OS X

## What we will cover

- What is an In-App Purchase?
- Types of In-App Purchases
- iTunes Connect Metadata Setup
- Testing in Sandbox and submitting to the App Store
- StoreKit on iOS and OS X
- In-App Purchase reporting
- Auto-renewable subscriptions
- Best practices

# Today's Agenda

1. What Is an In-App Purchase?
2. In-App Purchase Types
3. iTunes Connect Setup
4. Testing in Sandbox and Submitting to the App Store
5. StoreKit on iOS and OS X
6. Auto-renewable Subscriptions
7. In-App Purchase Reporting
8. Best Practices

# What Is an In-App Purchase?

- Allows enhanced functionality/content to be sold directly within an app
- Implemented using the StoreKit API, introduced with iOS 3.0 and 10.7
  - StoreKit prompts for payment and securely authorizes transaction
  - In-App purchases aid in fighting piracy
- Developer's server responsible for dynamically unlocking new content

# What Is an In-App Purchase?

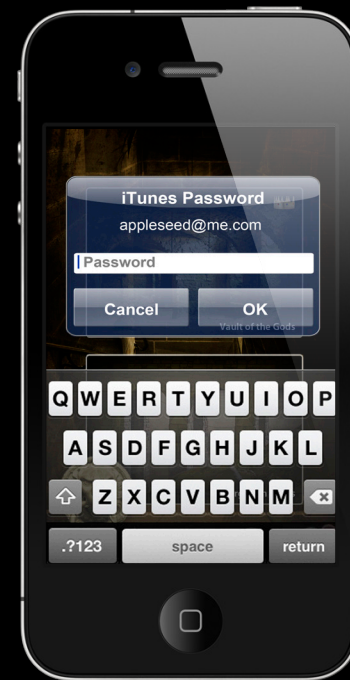
- Example usages
  - Adding for pay features to a free app
  - Adding additional levels to a game
  - Purchasing virtual goods
- The Daily
  - Leverages auto-renewable In-App Purchases

# Today's Agenda

1. What Is an In-App Purchase?
2. In-App Purchase Types
3. iTunes Connect Setup
4. Testing in Sandbox and Submitting to the App Store
5. StoreKit on iOS and OS X
6. Auto-renewable Subscriptions
7. In-App Purchase Reporting
8. Best Practices

# In-App Purchase Types

- Non-consumable
- Consumable
- Non-renewing subscription
- Auto-renewing subscription





# In-App Purchase Types

A blue rectangular badge with rounded corners and a subtle gradient, containing the word "New" in white sans-serif font.

## Non-consumables

- Available on iOS and new for Lion
- Designed to be purchased once and only once
- Consider these purchases “durable”
- Developer is responsible for ensuring purchase is present on all devices using `restoreCompletedTransactions`
- Examples
  - New level in a game
  - Additional feature in an app

# In-App Purchase Types

## Consumables



- Available on iOS and new for Lion
- Designed to be purchased multiple times by the customer
- Examples
  - Pet food or other items consumed as part of game play
  - Virtual currency as a means of advancement

# In-App Purchase Types

## Non-renewing subscriptions

- Only available on iOS
- Deprecated in favor of auto-renewing subscriptions

# Today's Agenda

1. What Is an In-App Purchase?
2. In-App Purchase Types
3. iTunes Connect Setup
4. Testing in Sandbox and Submitting to the App Store
5. StoreKit on iOS and OS X
6. Auto-renewable Subscriptions
7. In-App Purchase Reporting
8. Best Practices

# iTunes Connect Setup

## Create In-App Purchase item

- In-App Purchase type
- Reference name
- Product ID
- Languages: Display name, display description
- Pricing: Cleared for sale, price tier
- Screenshot for review

# iTunes Connect Setup

## Metadata: In-App Purchases link


**Touch Fighter III**

**App Information**

Identifiers		Links	
SKU	com.cyberinteractive.touchfighter.3	<a href="#">View in App Store</a>	<a href="#">Rights and Pricing</a>
Bundle ID	com.cyberinteractive.touchfighter.3		<a href="#">Manage In-App Purchases</a>
Apple ID	440030095		<a href="#">Manage Game Center</a>
Type	IOS App		<a href="#">Set Up iAd Network</a>
			<a href="#">Delete App</a>

**Versions**

**Current Version**

	Version <b>1.0</b>
	Status <b>Prepare for Upload</b>
	Date Created <b>07 June 2011</b>

[View Details](#)

[Done](#)

Manage  
In-App Purchases

# iTunes Connect Setup

## Metadata: In-App Purchase type

Consumable

Consumable

A consumable In-App Purchase must be purchased every time the user downloads it. One-time services, such as fish food in a fishing app, are usually implemented as consumables.

Choose

Non-Consumable

Non-Consumable

A non-consumable In-App Purchase only needs to be purchased once by the user. Services that do not expire or decrease with use, such as a new race track for a game app, are usually implemented as non-consumables.

Choose

Auto-Renewable Subscriptions

Auto-Renewable Subscriptions

An auto-renewable In-App Purchase subscription allows the user to purchase in-app content for a set duration of time. At the end of that duration the subscription will renew itself, unless the user opts out. An example of an auto-renewable subscription would be a magazine or newspaper that takes advantage of the auto-renewing functionality built into iOS.

Auto-renewable subscriptions will be delivered to all devices associated with the user's Apple ID. When you create an auto-renewable subscription in iTunes Connect, you begin by selecting the duration(s) that you will offer. When a duration ends, the App Store will automatically renew the subscription. Note that if the user has opted out of this functionality, the subscription will expire at the end of that duration. You must make sure that your app can determine whether a subscription is currently active and renewable.

Choose

# iTunes Connect Setup

## Metadata: Reference name and product ID

**Details**

Enter a reference name and a product ID for this In-App Purchase. You must also add at least one language, along with a display name and a description in that language.

Reference Name  ?

Product ID  ?

[Add Language](#)

Language	Display Name	Description
Click Add Language to get started.		



# iTunes Connect Setup

## Metadata: Adding languages

**Add Language**

Language	<input type="text" value="English"/>	?
Display Name	<input type="text" value="Rockets"/>	?
Display Description	<input type="text" value="Purchase extra rockets to fight enemy spaceships!"/>	?

# iTunes Connect Setup

## Metadata: Pricing

**Pricing**

Enter pricing details for this In-App Purchase below.

Cleared for Sale **Yes**  **No**

Price Tier **Tier 1**

[View Price Tiers](#)

Price Tier 1												
App Store	U.S.*	Mexico	Canada	U.K.	European Union*	Norway	Sweden	Denmark	Switzerland	Australia	New Zealand	Japan
Customer Price	US\$0.99	\$10.00	CA\$0.99	£0.59	0,79 €	6.00Kr(NO)	7.00Kr(SE)	6.00Kr(DK)	1.10Fr	AUS\$1.19	NZ\$1.29	¥115
Your Proceeds	US\$0.70		CA\$0.70	£0.36	0,48 €					AUS0.76		¥81

# iTunes Connect Setup

## Metadata: Review screenshot

### Screenshot for Review

Before you submit your In-App Purchase for review, you must upload a screenshot. This screenshot will be for review purposes only. It will not be displayed on the App Store.



Choose File

# Today's Agenda

1. What Is an In-App Purchase?
2. In-App Purchase Types
3. iTunes Connect Setup
4. Testing in Sandbox and Submitting to the App Store
5. StoreKit on iOS and OS X
6. Auto-renewable Subscriptions
7. In-App Purchase Reporting
8. Best Practices

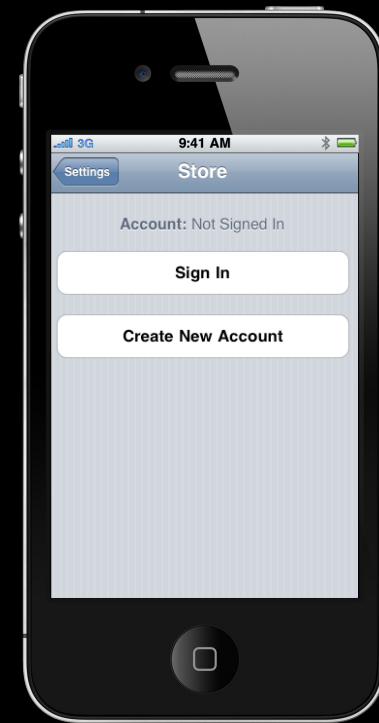
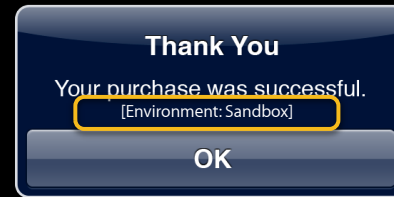
# Testing in Sandbox and Submission

## Testing in Sandbox with StoreKit

- Development-signed builds automatically hit the sandbox environment
- Payments are not processed in sandbox, but transactions are returned
- You must leverage a sandbox test user, not a regular production store account for sandbox testing

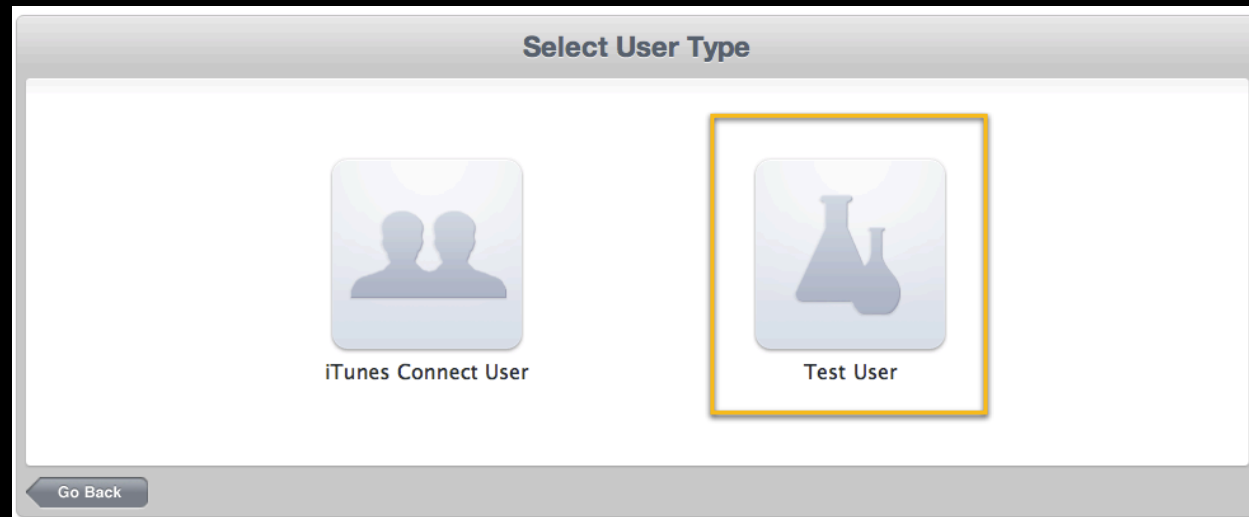
# Testing in Sandbox and Submission

- Create sandbox test users
  - Same sandbox test users work across iOS and Lion
  - Same goes with receipt test accounts on OS X
- Test in sandbox with StoreKit on your device
  - You must sign out production iTunes account first



# Testing in Sandbox and Submission

## Create Sandbox users



# Testing in Sandbox and Submission

## Create Sandbox users

**Add New User**

Fill out the information and click Save.

**First Name :**

**Last Name :**

**Email Address :**

**Password :**

**Confirm Password :**

**Secret Question :**

**Secret Answer :**

**Date of Birth :**

**Select iTunes Store :**



# Testing in Sandbox and Submission

## Submission

- First submission of In-App purchases must be with a binary
- Subsequent submissions can be ad hoc or with a binary

# Testing in Sandbox and Submission

## Submission


The screenshot shows the 'Touch Fighter III - In-App Purchases' submission page. At the top, there is a 'Delete' button on the left and a 'Submit for Review' button on the right, which is highlighted with a yellow border. Below the header, the product ID is 'com.cyberinteractive.touchfighter.missiles.twenty' and the status is 'Ready to Submit' with a yellow dot icon. The type is 'Consumable'. Under the 'Details' section, there is a note: 'The details for this In-App Purchase are shown below. You must maintain at least one language at all times.' The reference name is '20 Missiles' with an 'Edit' button. There is an 'Add Language' button. Below that is a table with one row for 'English'.

Language	Display Name	Description	
English	Twenty Missiles	Buy twenty extra missiles to help you on your way.	Delete

# Testing in Sandbox and Submission

## Submission

Create New **Touch Fighter III – In-App Purchases**

 **Touch Fighter III**  
Apple ID : 411446758 Bundle ID : com.cyberinteractive.touchfighter.pbs

3 In-App Purchases

	Reference Name		Product ID	Type	Status	
<input type="checkbox"/>	20 Missiles		om.cyberin...r.missiles.twenty	Consumable	🟡 Ready to Submit	<input type="button" value="Delete"/>
<input type="checkbox"/>	Touch Fighter ...Strategy	7 Days	com.cyberi...meStrategy.7days	Auto-Renewable	🟡 Ready to Submit	<input type="button" value="Delete"/>
<input type="checkbox"/>	Touch Fighter ...Strategy	1 Month	com.cyberi...eStrategy.30days	Auto-Renewable	🟡 Ready to Submit	<input type="button" value="Delete"/>

[View or generate a shared secret](#)

# Today's Agenda

1. What Is an In-App Purchase?
2. In-App Purchase Types
3. iTunes Connect Setup
4. Testing in Sandbox and Submitting to the App Store
5. StoreKit on iOS and OS X
6. Auto-renewable Subscriptions
7. In-App Purchase Reporting
8. Best Practices

# StoreKit for iOS and Mac OS X

**Jean-Pierre Ciudad**

Engineering Manager, Mac OS X, App Store

# StoreKit for iOS and Mac OS X

## Overview

StoreKit API overview

Getting product information

Purchasing

Restoring completed transactions

Receipts

# StoreKit for iOS and Mac OS X

## Overview

Same in iOS as Mac OS



StoreKit API overview

Getting product information

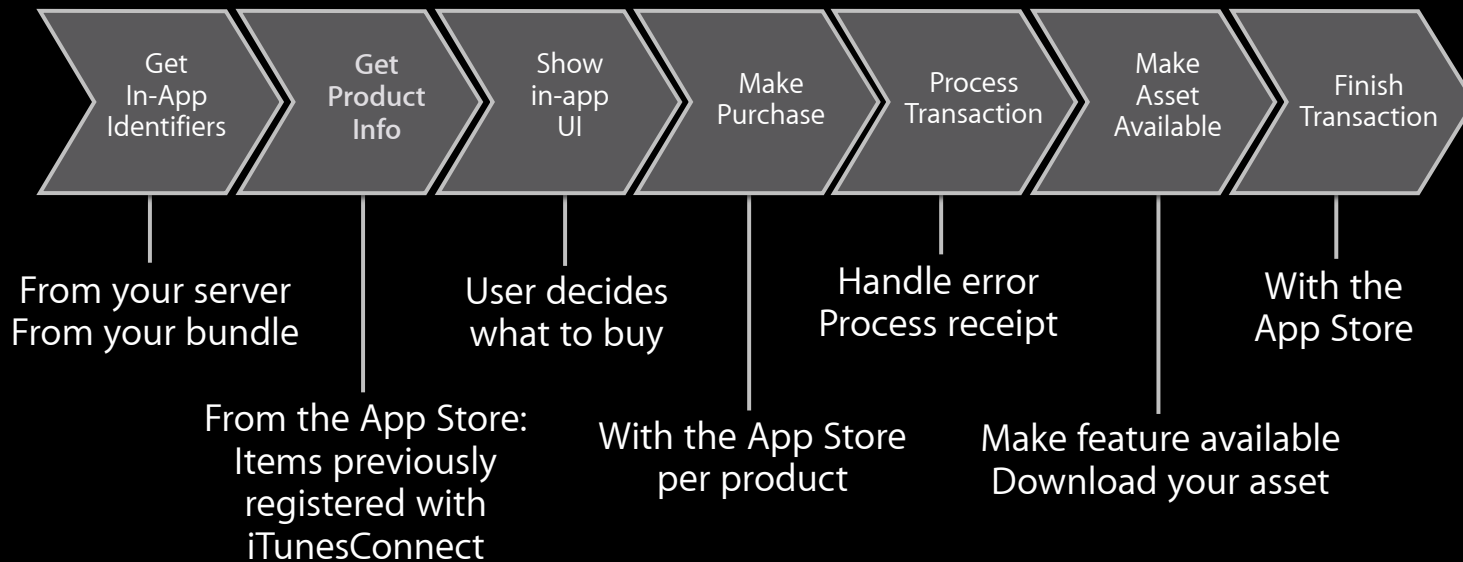
Purchasing

Restoring completed transactions

Receipts

# StoreKit API Overview

## Step by step

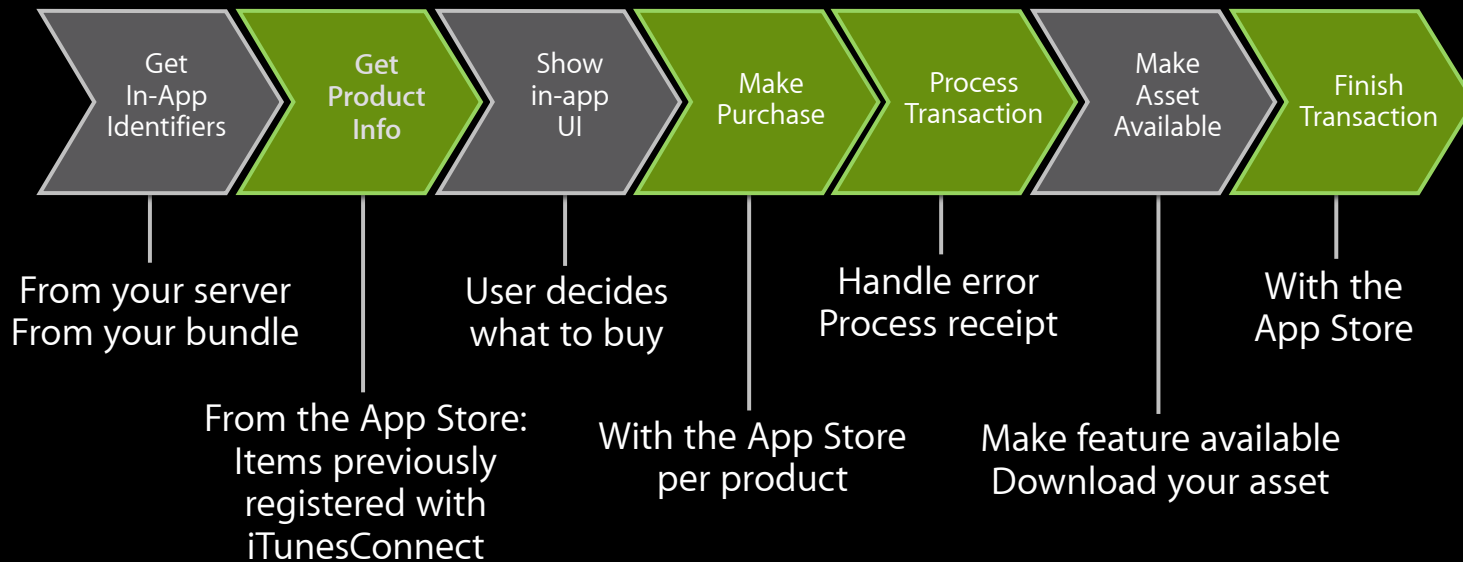




# StoreKit API Overview

## Step by step

### APIs Provided by StoreKit



# Getting Product Information

## Making the request

- Get In-App product identifiers
- Use SKProductRequest to create the request

```
SKProductsRequest* request = [[SKProductsRequest alloc]
initWithProductIdentifiers:identifiers];
request.delegate = self;
[request start];
```

# Getting Product Information

## Handling the response

- Implement SKProductsRequestDelegate protocol

```
- (void)productsRequest:(SKProductsRequest *)request didReceiveResponse:  
(SKProductsResponse *)response
```

```
response.products: description, name, price  
response.invalidProductIdentifiers
```

```
- (void)request:(SKRequest *)request didFailWithError:(NSError *)error
```

# Purchasing

## Preparing the purchase

- Add an observer

```
[[SKPaymentQueue defaultQueue] addTransactionObserver: self];
```

- Implement SKPaymentTransactionObserver protocol

```
- (void)paymentQueue:(SKPaymentQueue *)queue updatedTransactions:(NSArray *)  
transactions
```

# Purchasing

## 1. Making the purchase

```
SKPayment *payment = [SKPayment paymentWithProduct:product];  
[[SKPaymentQueue defaultQueue] addPayment:payment];
```

## 2. Observer gets called

```
SKPaymentTransactionStatePurchased  
    receipt data is available  
SKPaymentTransactionStateFailed
```

## 3. Make content available to user

## 4. Complete the transaction

```
[[SKPaymentQueue defaultQueue] finishTransaction: transaction];
```

# Restoring

1. Initiate the restore

```
[[SKPaymentQueue defaultQueue] restoreCompletedTransactions];
```

2. Observer gets called

```
SKPaymentTransactionStateRestored
```

3. Make content available to user

4. Complete the transaction

```
[[SKPaymentQueue defaultQueue] finishTransaction: transaction];
```

5. Optionally

```
- (void)paymentQueueRestoreCompletedTransactionsFinished:(SKPaymentQueue *)queue
```

# Receipts

## iOS and Mac OS

	iOS	Mac OS
<b>Format</b>	opaque	PKCS7
<b>Verification</b>	Your server with Apple	local or server
<b>Scope</b>	per In-App Purchase	per application
<b>Obtained from</b>	SKTransaction	app bundle
<b>Saved by</b>	you	App Store

# iOS Receipts

- NSData, property of transaction object
  - `SKPaymentTransactionStatePurchased`
  - `SKPaymentTransactionStateRestored`
- You are responsible for saving the receipt
- Your server can send the receipt to Apple for verification
  - JSON



# Mac OS Receipts

- Placed in the app bundle by the store
- One receipt per application contains all In-App Purchases
- Used to validate In-App Purchases on app launch
  - Can be verified locally
- PKCS7 archive, ASN/1 format
  - Signed by Apple
  - Tied to machine
- Your server can also send the receipt to Apple for verification
  - Prior to delivering additional content for example

# In-App Purchase Tips

- Always call `finishTransaction` on all transactions
  - Cause of most In-App Purchase errors
- Register as `SKPaymentQueue` observer **on launch**
  - To collect pending transactions
- Only restore In-Apps when user needs access to In-App Purchases
- On Mac OS, check receipt validity **on launch**
  - Exit 173 if invalid
  - Receipt required to use the sandbox
  - The store will give a new receipt after authentication

# More Information

## Development resource documents

- Validating App Store receipts
- Submitting your app to the Mac App Store
- In-App Purchase Programming Guide
- StoreKit Framework available on the ADC website

# StoreKit Demo

**Jean-Pierre Ciudad**

Engineering Manager, Mac OS X, App Store

# Today's Agenda

1. What Is an In-App Purchase?
2. In-App Purchase Types
3. iTunes Connect Setup
4. Testing in Sandbox and Submitting to the App Store
5. StoreKit on iOS and OS X
6. Auto-renewable Subscriptions
7. In-App Purchase Reporting
8. Best Practices

# Auto-renewable Subscriptions

**Ricardo Cortes**

Engineering Manager, iTunes Store, Provider Services

# Auto-renewable Subscriptions

1. Overview

2. iTunes Connect Setup

3. Using Auto-renewables

4. Auto-renewable Receipts

5. Auto-renew Policies

6. Auto-renew Cancellation Flow

7. Auto-renew and Sandbox

8. Auto-renew and the App Review Flow

# Auto-renewable Subscriptions

Only on  
iOS

## Overview

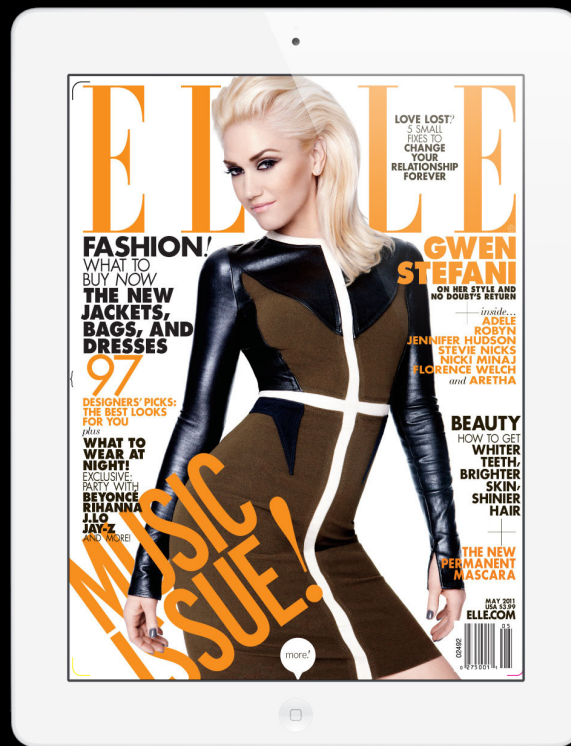
- Only available on iOS
- Enables true subscription support within your app
- Subscription renewed at the end of the subscription period
  - Customer given option to opt out in iTunes
- New content delivered to all devices with same customer Apple ID
- Duration options
  - 7 days, 1 month, 2 months, 3 months, 6 months, 1 year
- The most complex In-App purchase type to implement



# Auto-renew Examples



# Auto-renew Examples



# Auto-renew Examples



# Auto-renewable Subscriptions

## Overview

- Developer responsible for checking receipts to validate subscriptions
  - Shared Secret required to validate auto-renewables
- Marketing opt-in incentive
  - Customers receive free subscription extension for opting in
  - Duration chosen by developer/publisher in iTunes Connect
  - Optional to supply a duration, store dialog will always show
- Publisher responsible for displaying duration bonus in their UI
- Publisher's privacy URL required
  - Shown on customer's invoice

# Auto-renewable Subscriptions

1. Overview

2. iTunes Connect Setup

3. Using Auto-renewables

4. Auto-renewable Receipts

5. Auto-renew Policies

6. Auto-renew Cancellation Flow

7. Auto-renew and Sandbox

8. Auto-renew and the App Review Flow

# iTunes Connect Setup

## Choosing In-App Purchase type

Select Type

**Consumable**  
A consumable In-App Purchase must be purchased every time the user downloads it. One-time services, such as fish food in a fishing app, are usually implemented as consumables.

[Choose](#)

**Non-Consumable**  
A non-consumable In-App Purchase only needs to be purchased once by the user. Services that do not expire or decrease with use, such as a new race track for a game app, are usually implemented as non-consumables.

[Choose](#)

**Auto-Renewable Subscriptions**  
An auto-renewable In-App Purchase subscription allows the user to purchase in-app content for a set duration of time. At the end of that duration the subscription will renew itself, unless the user opts out. An example of an auto-renewable subscription would be a magazine or newspaper that takes advantage of the auto-renewing functionality built into iOS.

Auto-renewable subscriptions will be delivered to all devices associated with the user's Apple ID. When you create an auto-renewable subscription in iTunes Connect, you begin by selecting the duration(s) that you will offer. When a duration ends, the App Store will automatically renew the subscription. Note that if the user has opted out of this functionality, the subscription will expire at the end of that duration. You must make sure that your app can determine whether a subscription is currently active and renewable.

[Choose](#)

# iTunes Connect Setup

Add Duration and Pricing

Duration: 1 Month ?

Product ID: com.cyberinteractive.touchfighter.gameStrategy.30days ?

Offer a marketing opt-in incentive? Yes  No  ?

Incentive Duration: 7 Days

Cleared for Sale: Yes  No

Price Tier: Tier 3 ?  
[View Price Tiers](#)

Price Tier 3												
App Store	U.S.*	Mexico	Canada	U.K.	European Union*	Norway	Sweden	Denmark	Switzerland	Australia	New Zealand	Japan
Customer Price	US\$2.99	\$30.00	CA\$2.99	£1.79	2,39 €	17.00Kr(ND)	22.00Kr(SE)	18.00Kr(DK)	3.30Fr	AUS\$3.99	NZ\$4.19	¥350
Your Proceeds	US\$2.10		CA\$2.10	£1.09	1,45 €					AUS2.54		¥245

Cancel Save

# iTunes Connect Setup

### Touch Fighter III – In-App Purchases

#### Reference Name and Languages

Enter a reference name for this family of auto-renewable In-App Purchase subscriptions. You must also add at least one language, along with a display name and a description in that language. Note that the localized display name(s) and description(s) will be used for every subscription duration that you offer for this family.

Reference Name  ?

[Add Language](#)

Language	Display Name	Description	
English	Touch Fighter Game Strategy	Touch Fighter Game Strategy is designed with you the ga...	<a href="#">Delete</a>

#### Subscription Durations and Pricing

A subscription duration is the length of time between autorenewals. You must add at least one. Note that each duration can only be used once per family.

[Add Duration](#)

Duration	Product ID	Price Tier	Status	
7 Days	com.cyberinteractive.touchfighter.game...	Tier 1	● Waiting for Screenshot	<a href="#">Delete</a>
1 Month	com.cyberinteractive.touchfighter.game...	Tier 3	● Waiting for Screenshot	<a href="#">Delete</a>

#### Screenshot for Review



# iTunes Connect Setup

## Privacy URL

**Add Privacy Policy URL**

To save this family of auto-renewable In-App Purchase subscriptions, you must add a URL that links to your company's privacy policy. Privacy policy URLs are required for all apps that offer auto-renewable subscriptions. Customers will see this URL on their invoice and on the subscription confirmation email they receive. Enter a privacy policy URL for each language listed below. You will be able to view or edit these localized URLs on the Version Details page for your app.

English	<input type="text" value="http://touchfighter.com/Privacy"/>	?
---------	--	---

# iTunes Connect Setup

## Shared secret

### In-App Purchase Shared Secret

#### Generate Shared Secret

A shared secret is a unique code that you should use when you make the call to our servers for your In-App Purchase receipts. Without a shared secret, you will not be able to test auto-renewable In-App Purchase subscriptions in the sandbox mode. Also note that you will not be able to make them available on the App Store.

Note: Regardless of what app they are associated with, all of your auto-renewable subscriptions will use this same shared secret.

**5ecf8af6a88b434da63ef54b178be2e9**

Generate New

# Auto-renewable Subscriptions

1. Overview

2. iTunes Connect Setup

3. Using Auto-renewables

4. Auto-renewable Receipts

5. Auto-renew Policies

6. Auto-renew Cancellation Flow

7. Auto-renew and Sandbox

8. Auto-renew and the App Review Flow

# Using Auto-renewables

**David Neumann**

Engineering Manager, iTunes Store, Commerce

# Using Auto-renewables

## Subscription title vs. offer title

- SKProduct's `localizedTitle` is the title of your subscription:  
For example: "The Oregon Times"
  - Each SKProduct in the same family has the **same title**
  - Used by Apple in dialogs, management UI, emails
  - Used by you?
- Offer titles are **not** on SKProduct:  
For example: "Weekly Plan," "Monthly Plan," "Yearly Plan"
  - Get by product ID: `com.ortimes.sub.1month`
    - Encode duration and combine with `localizedTitle` or
    - Fetch from your dev server

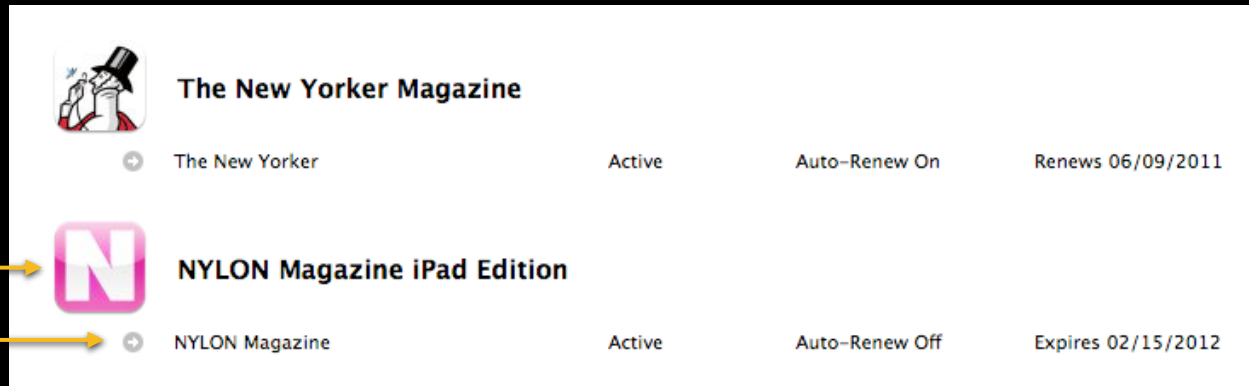
# iTunes Connect Metadata Setup



## Metadata setup vs. device screens

Desktop Management

App Name

Subscription Name  
(SKProduct title)



	<b>The New Yorker Magazine</b>			
⊕	The New Yorker	Active	Auto-Renew On	Renews 06/09/2011
	<b>NYLON Magazine iPad Edition</b>			
⊕	NYLON Magazine	Active	Auto-Renew Off	Expires 02/15/2012

# iTunes Connect Metadata Setup

## Metadata setup vs device screens

**The New Yorker**  
The New Yorker Magazine

Current Subscription: 1 month  
Price: \$5.99  
Renews: Jun 09, 2011

Turning off auto-renewal will cancel this subscription at the end of the current subscription period.

Auto-Renewal  On  Off

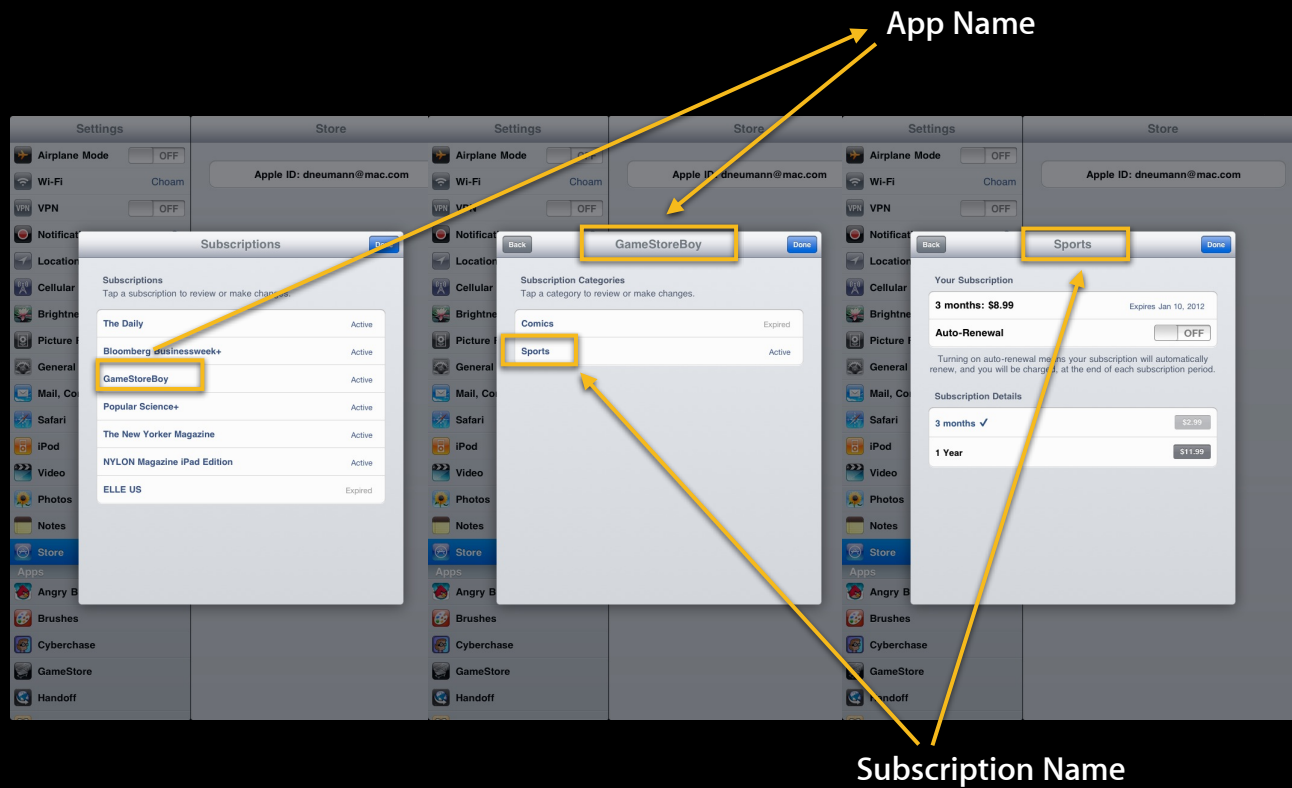
**Subscription Details**  
Your current subscription is for 1 month. If you want a different subscription period to take effect when your current one ends, choose an option below.

1 month	\$5.99	✓
1 Year	\$59.99	<input type="button" value="SUBSCRIBE"/>

Back Done

App Name  
Subscription Name

# iTunes Connect Metadata Setup





# Auto-renewable Subscriptions

1. Overview

2. iTunes Connect Setup

3. Using Auto-renewables

4. Auto-renewable Receipts

5. Auto-renew Policies

6. Auto-renew Cancellation Flow

7. Auto-renew and Sandbox

8. Auto-renew and the App Review Flow

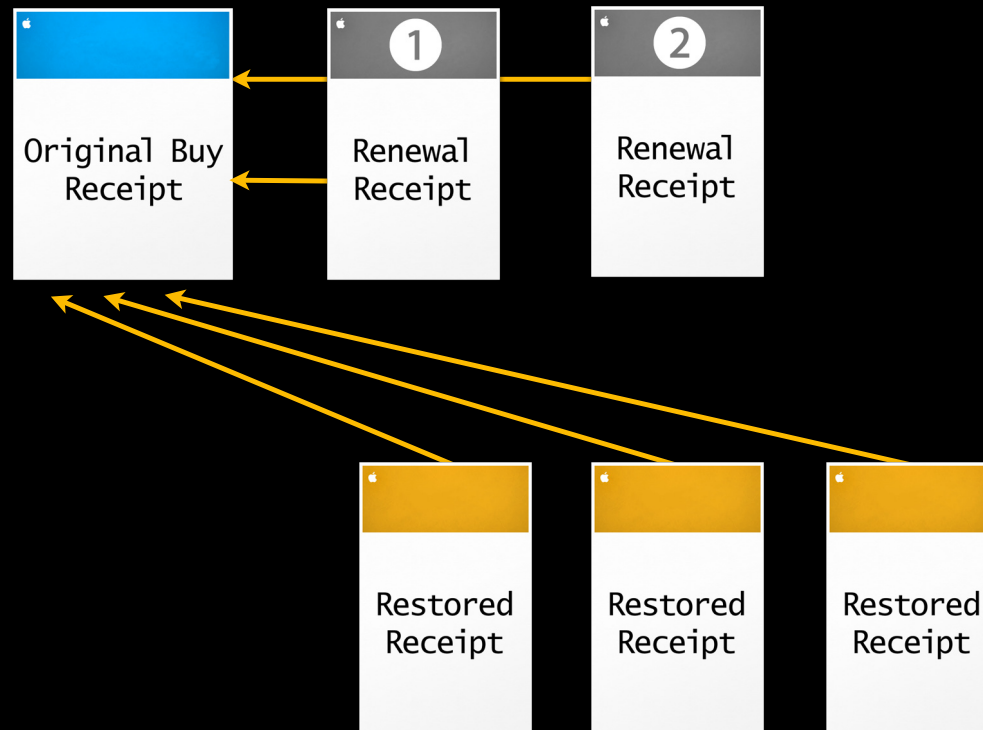
# Auto-renewable Receipts

## Fields

- Meaning of some fields slightly different
  - **Purchase date**
    - On nonconsumables this is the date of download or redownload
    - For auto-renewable this is the date of orig buy or subsequent renewal
  - **Original transaction ID**
    - txn id of that very first subscription buy
    - All receipts for that user subscription use the same value

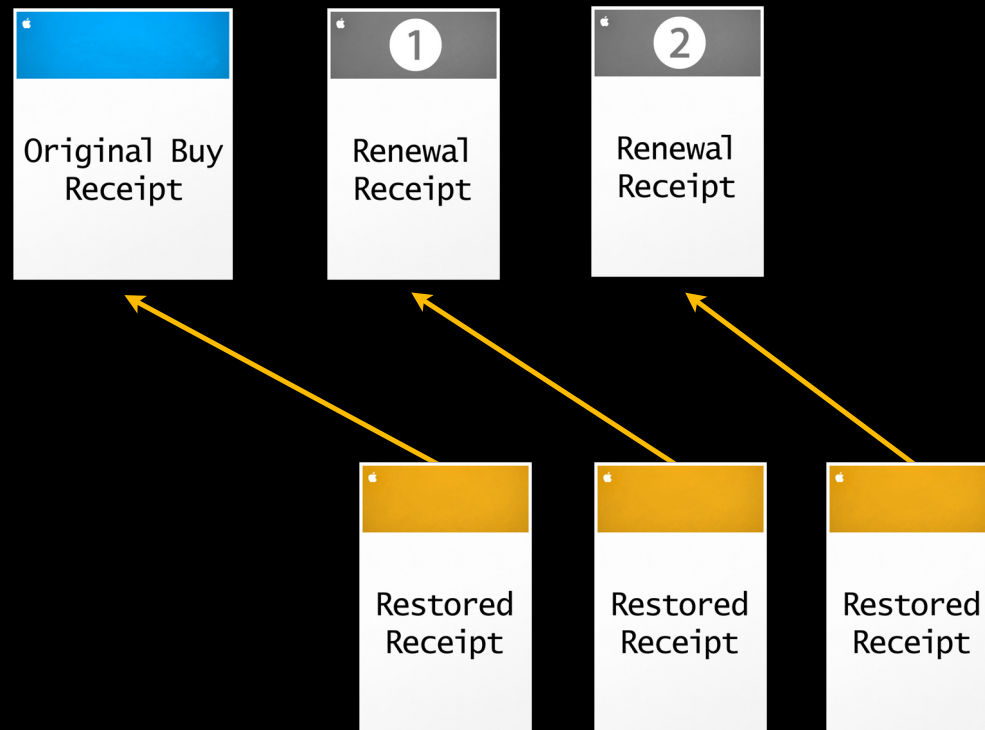
# Auto-renewable Receipts

Original transaction ID



# Auto-renewable Receipts

Purchase date



# Auto-renewable Receipts

## Fields

- One new field
  - **Expire date**
  - The date the receipt expired
  - Sub not necessarily inactive though if in past
  - Use expire date less purchase date to know active range
  - Useful when restoring a user's content history

# Auto-renewable Receipts

## Verifying auto-renewable receipts

- Auto-renewable status codes
  - 0: Active subscription
  - 21000: App store could not read the JSON you provided
  - 21002: the data in the receipt-data property was malformed
  - 21003: the receipt could not be authenticated
  - 21004: the shared secret you provided does not match
  - 21005: the receipt server is currently unavailable
  - 21006: the receipt is valid but the subscription is expired
  - 21007: Sandbox receipt sent to production
  - 21008: production receipt sent to sandbox

# Auto-renewable Receipts

## Info from verifying receipts

- Pass receipt
  - We return its contents under `receipt`
- If subscription is active
  - We return latest receipt available under `latest_receipt`
  - We show you its contents under `latest_receipt_info`
  - Works regardless of old or new receipt being sent
- If subscription is inactive
  - We return info about last good receipt under `latest_expired_receipt_info`

# Auto-renewable Receipts

## Reconstructing user's purchase history with receipts

- Restore all really restores all
  - We return a receipt for each renewal **and** orig buy
  - Once you have got all of the receipts, send them to your server for verification
  - Orig txn id, purchase date, expire date let you know what user qualified for
  - Some apps will not care, just whether active
  - But others will want to restore the bookshelf of content the user got before... **"Physical magazines don't vanish on expiry; digital ones do not have to either."**



# Auto-renewable Receipts

## Making client app aware of renewal

- After renewal, new txn will appear in a user's txn queue
- App checks queue
  - On launch
  - On rotation out of the background
- Auto-renewed txn will appear as restored
  - App should finish these restored txns after processing them
  - App can then send the receipt to dev server for verification
  - Based on verification, app can get qualified content from dev server

# Auto-renewable Receipts

## Renewing in the app

- User clicks buy subscription in the app and...
  - **If sub still active, user presented with dialog**
    - Dialog explains user is already active
    - Has link to manage subscription
  - **If sub is expired, user allowed to renew right in the app**
    - Unlike original buy, user is not presented with data sharing opt-in dialog

# Auto-renewable Subscriptions

1. Overview

2. iTunes Connect Setup

3. Using Auto-renewables

4. Auto-renewable Receipts

5. Auto-renew Policies

6. Auto-renew Cancellation Flow

7. Auto-renew and Sandbox

8. Auto-renew and the App Review Flow

# Auto-renew Policies

- We renew a few hours before expiry to guarantee continuity
- Auto-renew is disabled if price goes up, email sent
- Auto-renew is disabled if item no longer available, email sent
- Email sent before renew if it looks like the user can not pay for the renewal
- Email sent if renew fails
- Email sent once per month for subs that will expire or renew in next month

# Auto-renewable Subscriptions

1. Overview
2. iTunes Connect Setup
3. Using Auto-renewables
4. Auto-renewable Receipts
5. Auto-renew Policies
6. Auto-renew Cancellation Flow
7. Auto-renew and Sandbox
8. Auto-renew and the App Review Flow

# Auto-renew Cancellation Flow

- No user cancellation per se;
  - But users can choose to **disable** auto-renew
- Support may cancel
  - Such cancellation impacts status
  - Receipt that has not expired will be indicated as 21006 if cancelled

# Auto-renewable Subscriptions

1. Overview

2. iTunes Connect Setup

3. Using Auto-renewables

4. Auto-renewable Receipts

5. Auto-renew Policies

6. Auto-renew Cancellation Flow

7. Auto-renew and Sandbox

8. Auto-renew and the App Review Flow

# Auto-renew and Sandbox

## What is different from production?

- We do not keep renewing over and over
  - Stop at 6
- Stuff renews fast; you might want to test with “Years”
- Stuff renews so fast, sometimes renewal happens after expiry
- Management interface for users not available



# Auto-renew and Sandbox

## What is different from production?

- Renewal schedule

Face Value	Actual Duration
7 Days	3 minutes
1 Month	5 minutes
2 Months	10 minutes
3 Months	15 minutes
6 Months	30 minutes
1 Year	60 minutes

# Auto-renew and Sandbox

## Why cap renewals at six?

- Test how your app handles end of subscription
- Interactive renewal post expiry allows
  - Test dealing with non-contiguous history
  - Test changing between durations in same user's history

# Auto-renewable Subscriptions

1. Overview
2. iTunes Connect Setup
3. Using Auto-renewables
4. Auto-renewable Receipts
5. Auto-renew Policies
6. Auto-renew Cancellation Flow
7. Auto-renew and Sandbox
8. Auto-renew and the App Review Flow

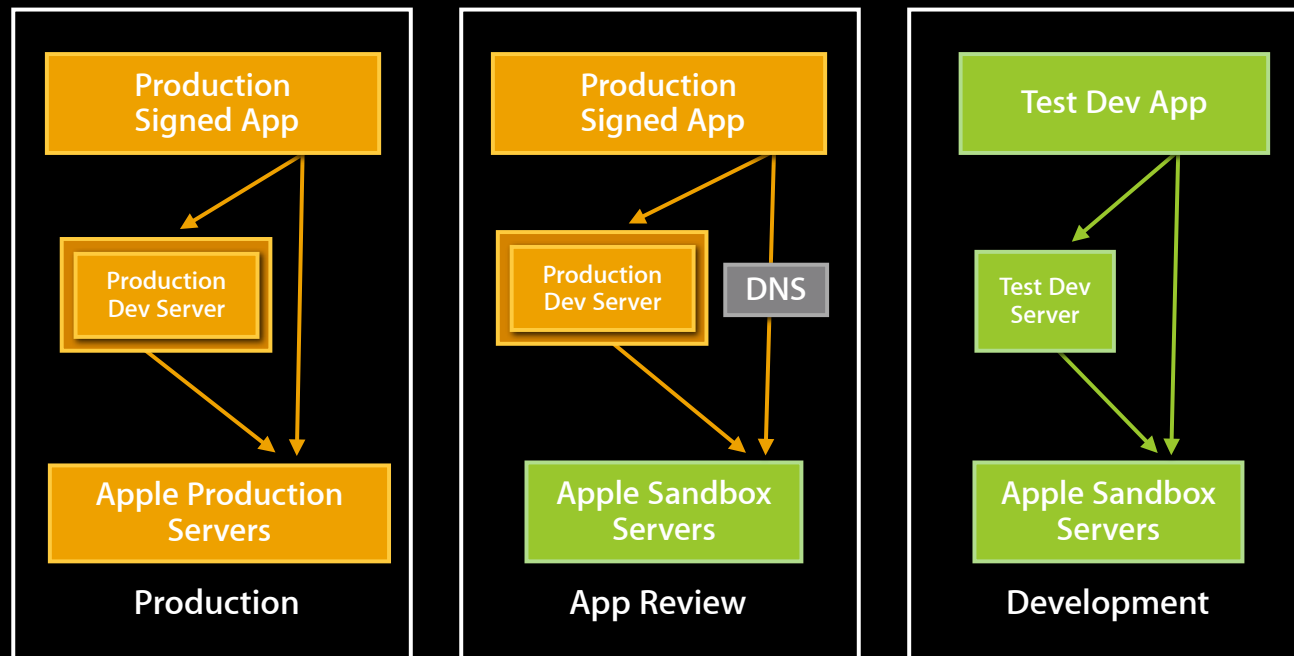
# Auto-renew and App Review

## App review

- Same Sandbox, different app
- Production signed. Why?
  - Need to review what customers will receive
  - Some bugs only affect a production signed app
- Half production, half development
  - To elaborate...

# Auto-renew and App Review

## Environments compared



# Auto-renew and App Review

## Choosing Apple Receipt Server

- **Smart production-signed App**

- App knows live version, selects Dev's QA or production server

- **Smart production server**

- Client passes version
- Server knows live version, selects Apple's Sandbox or production server

- **Reactive production server**

- Server always tries Apple production first
- If gets 21007, tries Apple sandbox

# Today's Agenda

1. What Is an In-App Purchase?
2. In-App Purchase Types
3. iTunes Connect Setup
4. Testing in Sandbox and Submitting to the App Store
5. StoreKit on iOS and OS X
6. Auto-renewable Subscriptions
7. In-App Purchase Reporting
8. Best Practices

# In-App Purchase Reporting

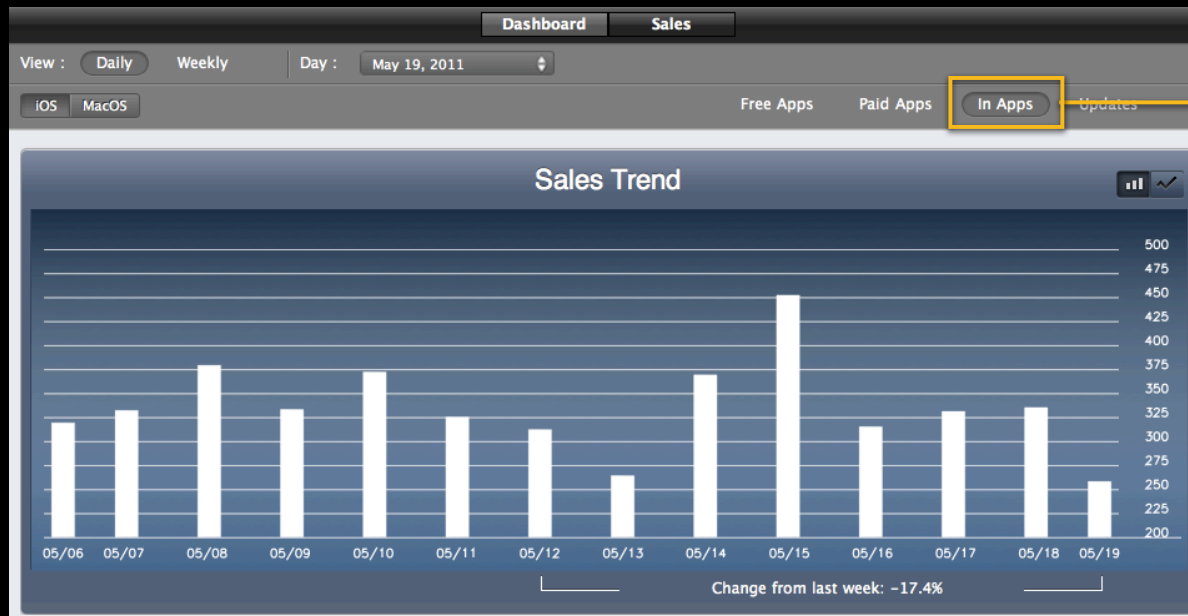
**Ricardo Cortes**

Engineering Manager, iTunes Store, Provider Services



# In-App Purchase Reporting

## iTunes Connect: Sales and trends

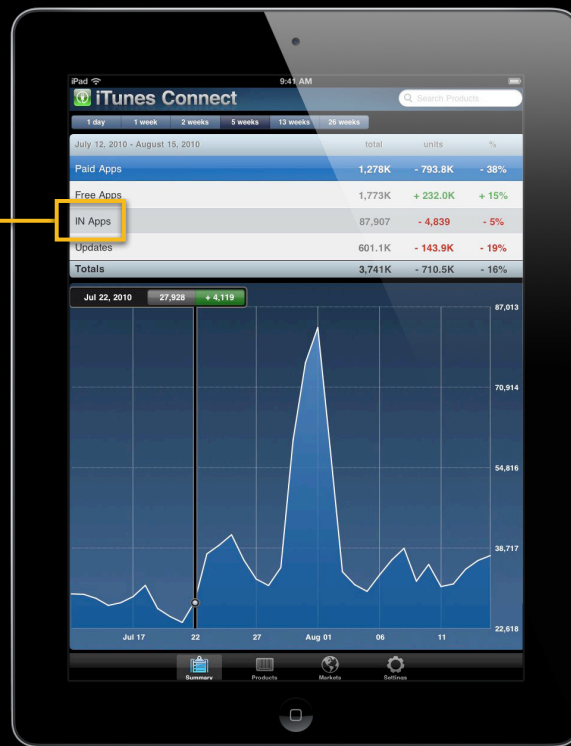


In Apps

# In-App Purchase Reporting

## iTunes Connect Mobile: Sales and trends

IN Apps



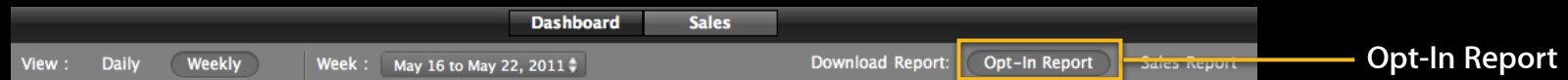
# In-App Purchase Reporting

## iTunes Connect Mobile: Sales and trends



# Auto-renewable Reporting

- Personal information reports
- Available in the Sales/Trends module of iTunes Connect
- Click sales tab, then Opt-in Report
- Ensure you unlock the zip archive with your Opt-In Passkey



# Today's Agenda

1. What Is an In-App Purchase?
2. In-App Purchase Types
3. iTunes Connect Setup
4. Testing in Sandbox and Submitting to the App Store
5. StoreKit on iOS and OS X
6. Auto-renewable Subscriptions
7. In-App Purchase Reporting
8. Best Practices

# Best Practices

**Max Müller**

Director, iTunes Store, Digital Supply Chain Engineering

# Best Practices

- In-App Purchases are not cross-platform
- In-App Purchases are not cross-app
- Consumable vs. non-consumable types matter
- Virtual currency allowed when used within an application, non-transferable
- App Review reviews a production signed application pointed at the sandbox

# Summary

- Auto-renewable subscriptions are complicated
- Restoration of In-App Purchases is required
- Use auto-renewables for true subscription support
- OS X developers, start using Sandbox



# More Information

## **Bill Dudney**

Application Frameworks Evangelist  
[dudney@apple.com](mailto:dudney@apple.com)

## **Documentation**

iTunes Connect Developer User Guide  
[http://itunesconnect.apple.com/docs/iTunesConnect\\_DeveloperGuide.pdf](http://itunesconnect.apple.com/docs/iTunesConnect_DeveloperGuide.pdf)

## **Apple Developer Forums**

<http://devforums.apple.com>

# Related Sessions

What is New with App Publishing in iTunes Connect

Presidio  
Thursday 10:15AM

# Labs

iTunes Connect Lab	Internet and Web Lab A Thursday 2pm - 4:15pm
In-App Purchases Lab	Internet and Web Lab B Thursday 11:30am - 1:30pm
iOS App Review Lab	3rd Floor All Week by reservation
Mac App Review Lab	3rd Floor All Week by reservation
Developer Program Services Lab	3rd Floor All Week by reservation
iTunes Connect Developer Lab	3rd Floor All Week by reservation

Q&A

