Using Store Kit for In-App Purchases

Session 305

Thomas Alsina
Manager, iOS Media Apps and Stores

96%

of 25 top grossing iPhone apps use in-app purchases

Agenda

Agenda

- What’s new in Store Kit
- Implementing in-app purchases
- Using the test environment
- Tips for passing app review

What’s New in Store Kit

Store Kit Is a Payment System

- Manages transactions for in-app purchases
- Provides a transaction receipt
- Security comes from the receipt

Security Is in the Receipt

Related Sessions

Using Receipts to Protect Your Digital Sales	Presidio Thursday 2:00PM

Evolution of the Receipt

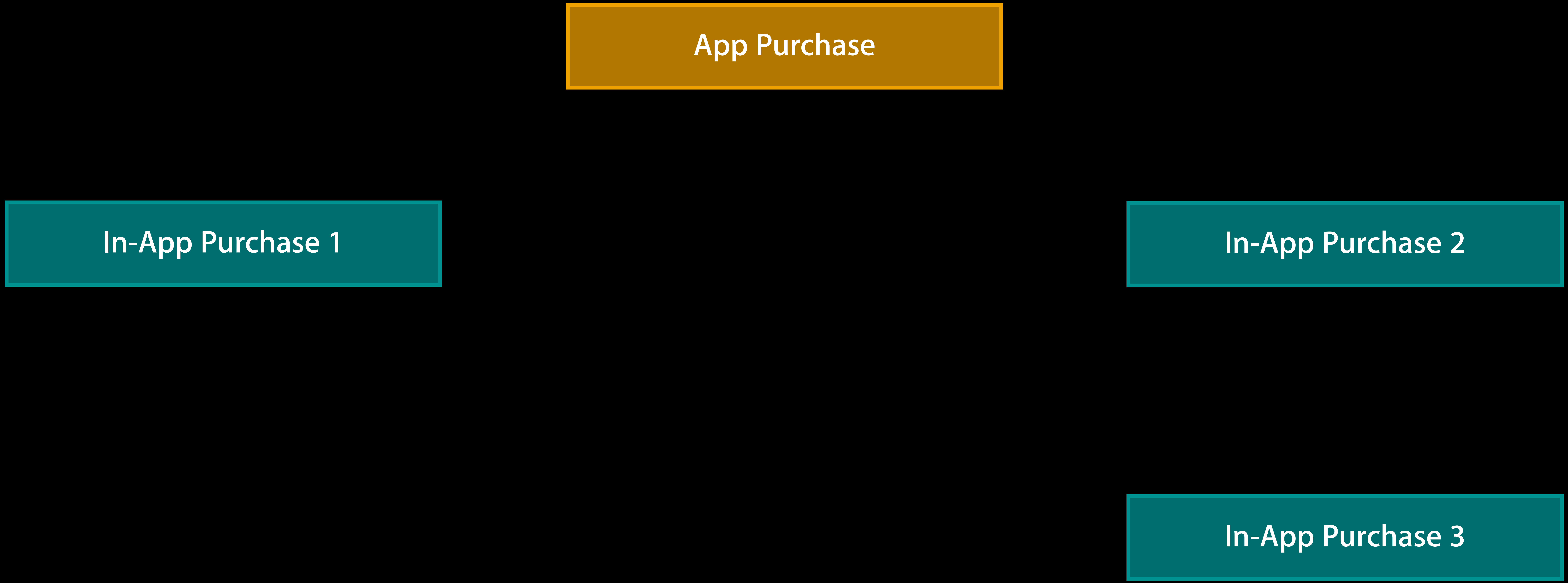
Evolution of the Receipt

In-App Purchase 1

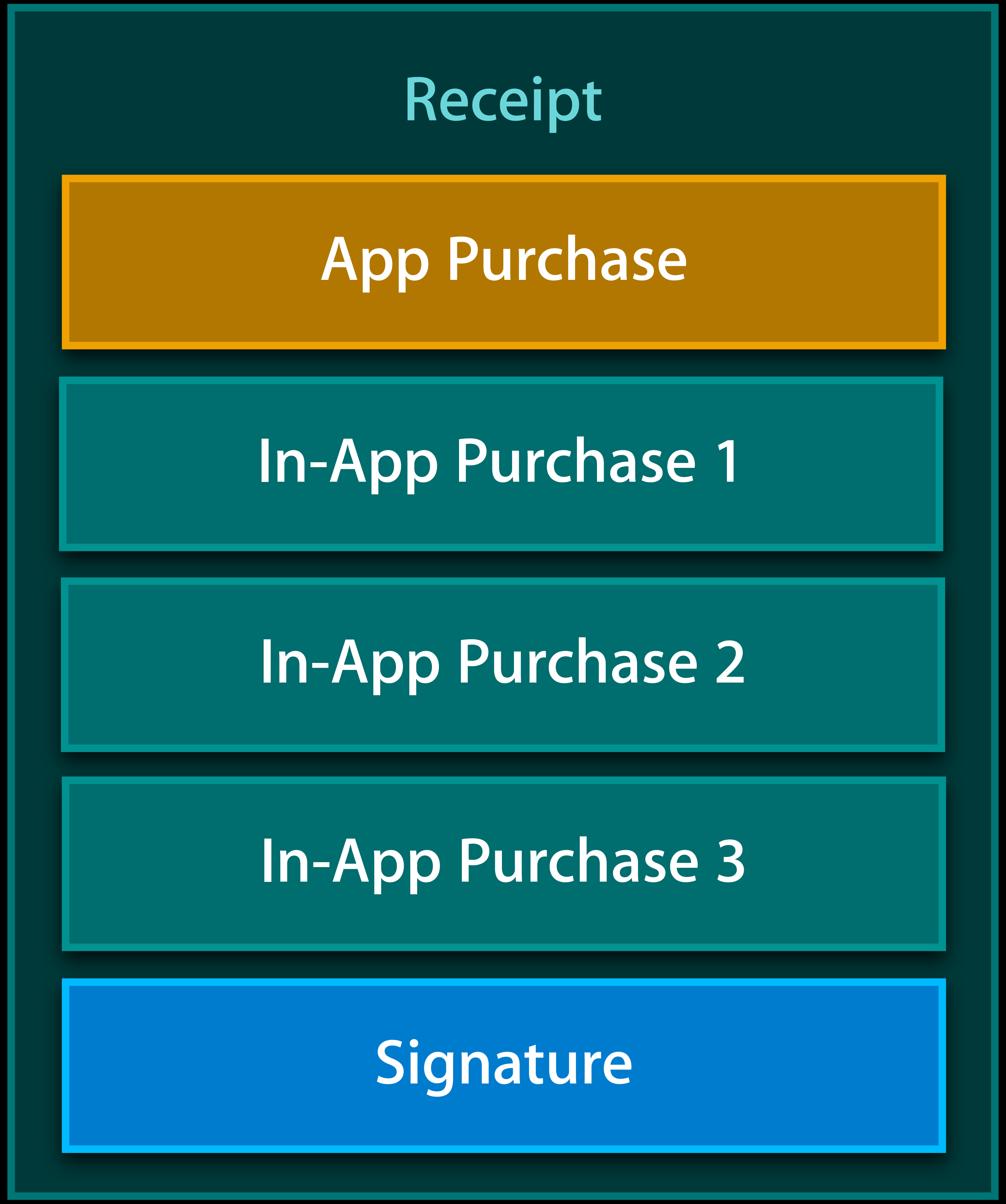
In-App Purchase 2

In-App Purchase 3

Evolution of the Receipt

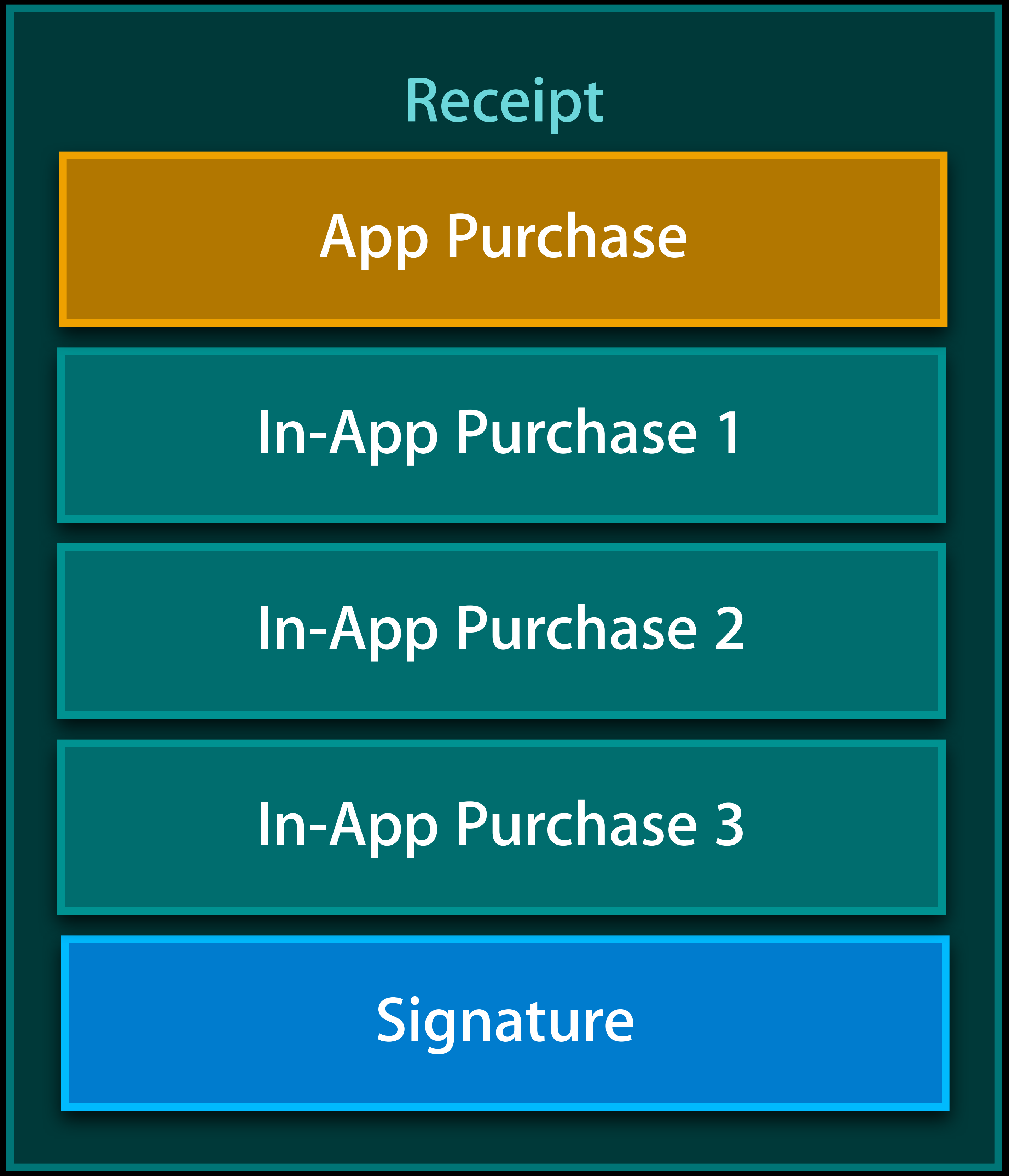


Evolution of the Receipt

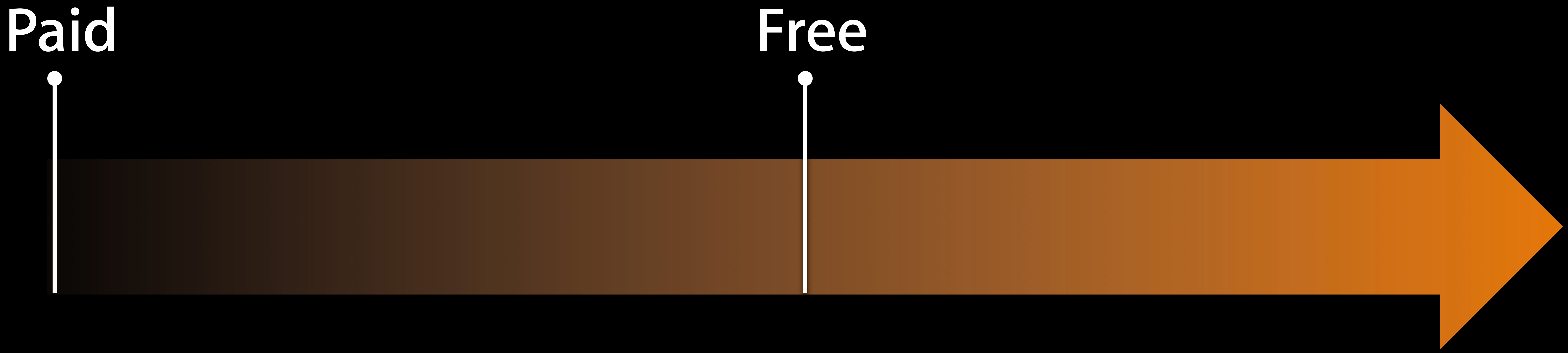


Grand Unified Receipt

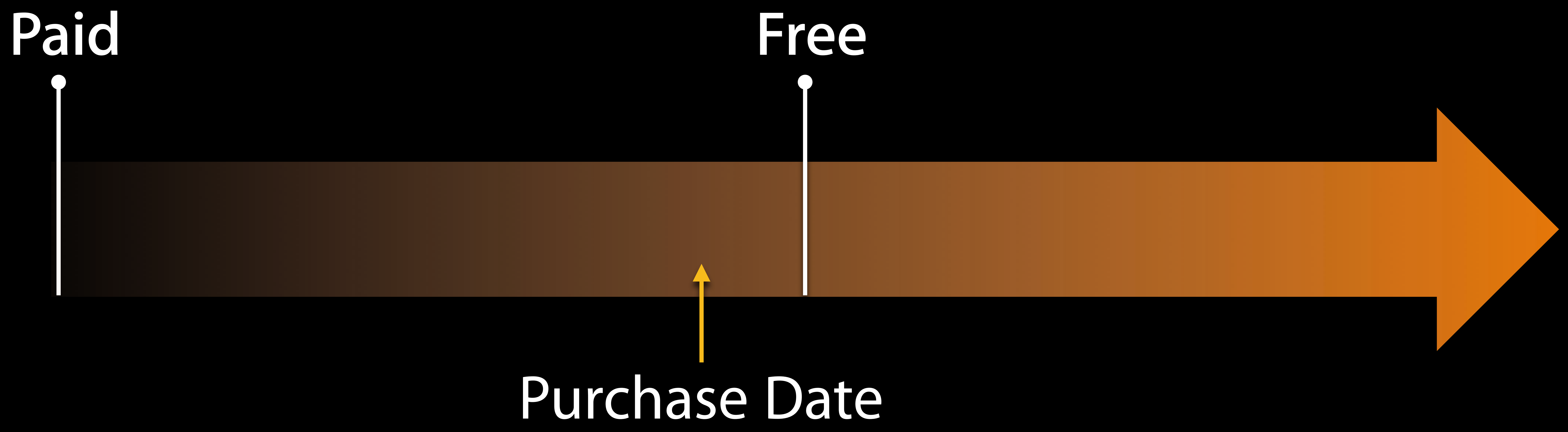
- Same receipt format for iOS 7 and OS X
- Managed for you
- You can validate on device
- Includes app purchase receipt
    - Extra level of security on iOS
    - Helps apps switch to “freemium” model



Switching to Freemium



Switching to Freemium



Pre-iOS 7 Receipts

Deprecated

Pre-iOS 7 Receipts

- Existing apps continue to work
- Transition to new receipt
- Use weak linking to support both iOS 6 and iOS 7 receipts

```
[[NSBundle mainBundle] respondsToSelector:@selector(appStoreReceiptURL)];
```

Volume Purchase Program

New licensing model

NEW

- Opt in using iTunes Connect
- Organizations can buy licenses in bulk
- Licenses can be assigned to members
- Licenses can be taken back
- Grace period before expiration

Volume Purchase Program

Handling the expiration

- iOS prevents your app from launching if expired
- OS X does not enforce expiration
- In both cases, app enforces expiration
- New fields in the receipt

Related Sessions

Extending Your Apps for Enterprise and Education Use	Nob Hill Tuesday 3:15PM
Using Receipts to Protect Your Digital Sales	Presidio Thursday 2:00PM

Free Hosted In-App Purchases

NEW

- Use them to provide optional content packages
- We will host them, serve them, and deliver them in the background
- You don't need servers anymore
- They are still regular hosted in-app purchases
    - User needs to authenticate to "buy" them
    - Go through review process

Consumable Products

- Provided to the app once
- Can be purchased multiple times
- Not restored
- Up to the app to manage
  - VoIP credit
  - Gold coins

Non-Consumable Products

- Persistent
- Restored across multiple devices
- Managed by Store Kit
    - Game levels
    - Books, magazines

Auto-Renewable Subscriptions

- Commonly used for periodic content
- Renew automatically
- One transaction per renewal
- Incentive for users to provide email address

Non-Renewing Subscriptions

- For all other subscriptions
- Provided once to the app
- Not restored
- No duration information
- Up to the app to manage the subscription
    - Flight charts
    - Access to financial services
    - Professional apps

Types of In-App Purchases

NEW

	iOS	OS X
Consumable	√	√
Non-consumable	√	√
Auto-renewable subscription	√	
Non-renewing subscription	√	

Types of In-App Purchases

NEW

	iOS	OS X
Consumable	√	√
Non-consumable	√	√
Auto-renewable subscription	√	√
Non-renewing subscription	√	√

Cross-Platform Subscriptions

- Separate product identifiers on OS X and iOS
- You can still use your own account tracking

Agenda

- What’s new in Store Kit
- **Implementing in-app purchases**
- Using the test environment
- Tips for passing app review

Implementing In-App Purchases

Transitioning to the new receipt format

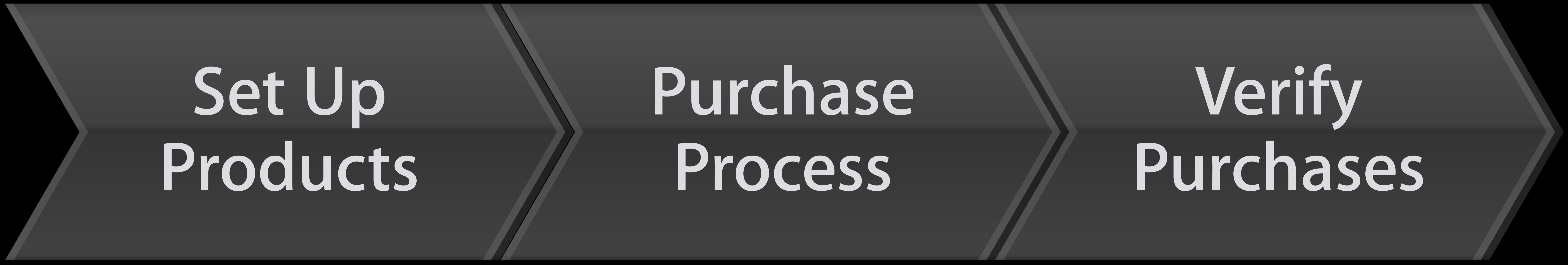
Sessions from Last Year

Documentation
In-App Purchase Programming Guide
Receipt Validation Programming Guide
http://developer.apple.com

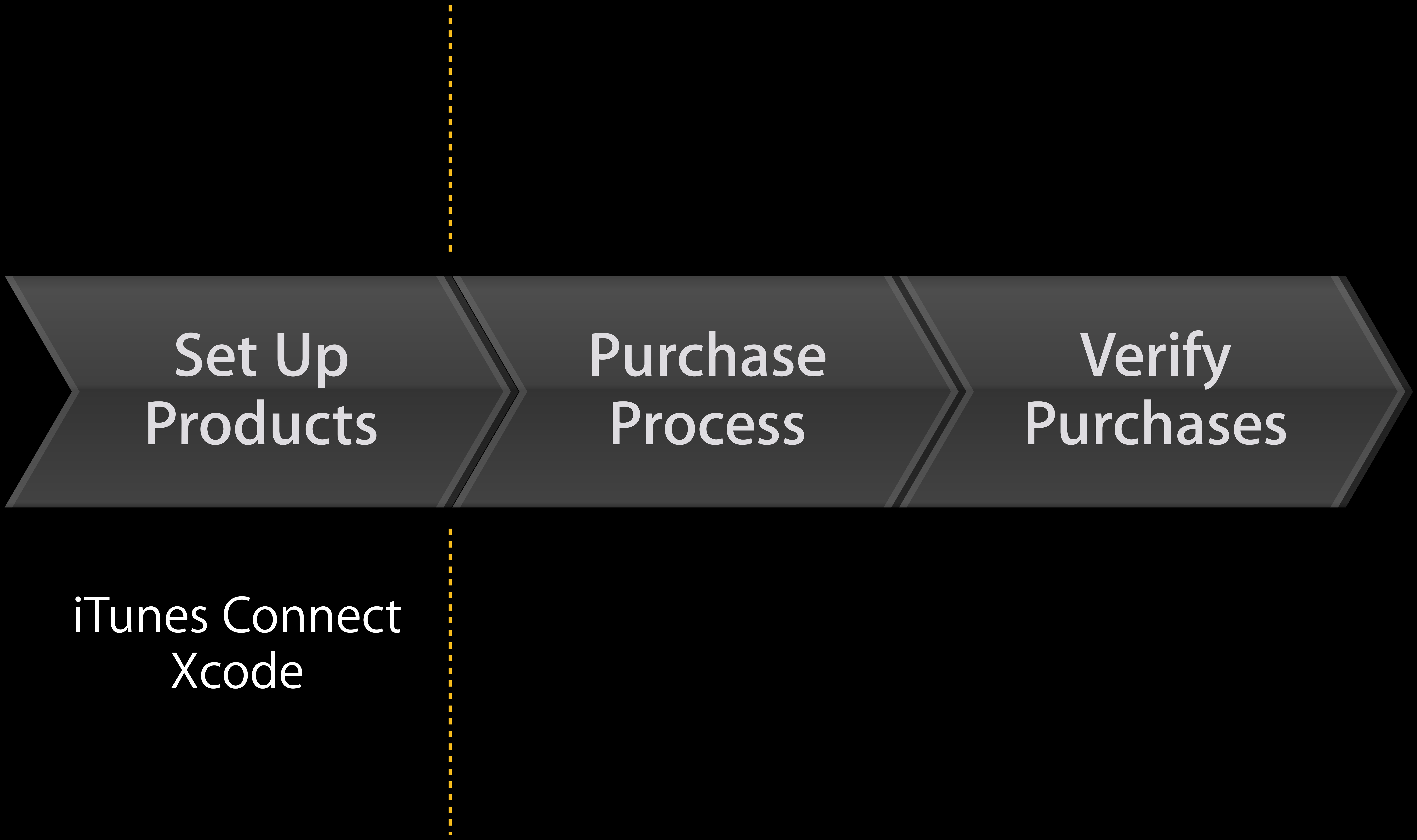
WWDC session
Selling Products with Store Kit
https://developer.apple.com/videos/wwdc/2012/

Three Phases

Three Phases

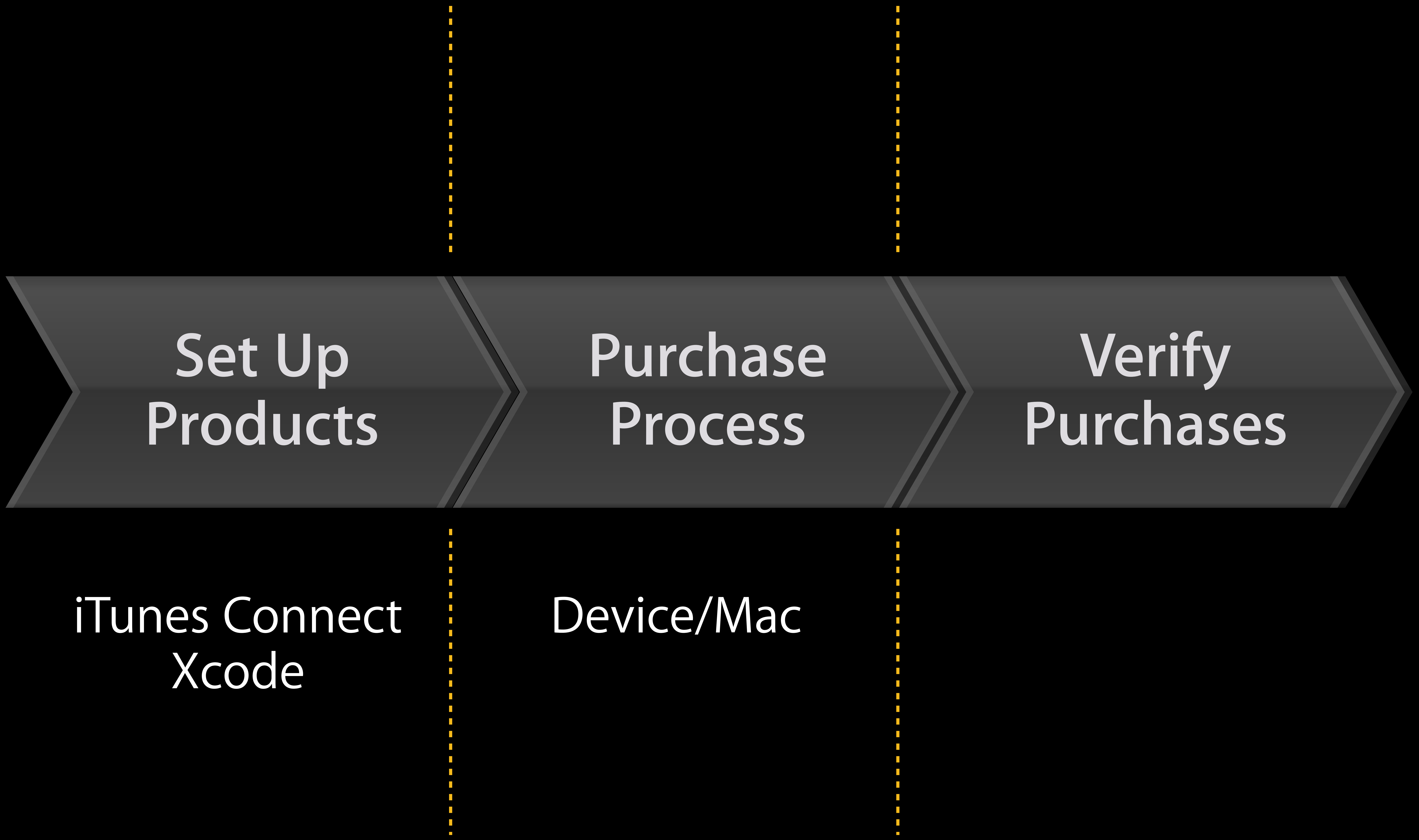


Three Phases

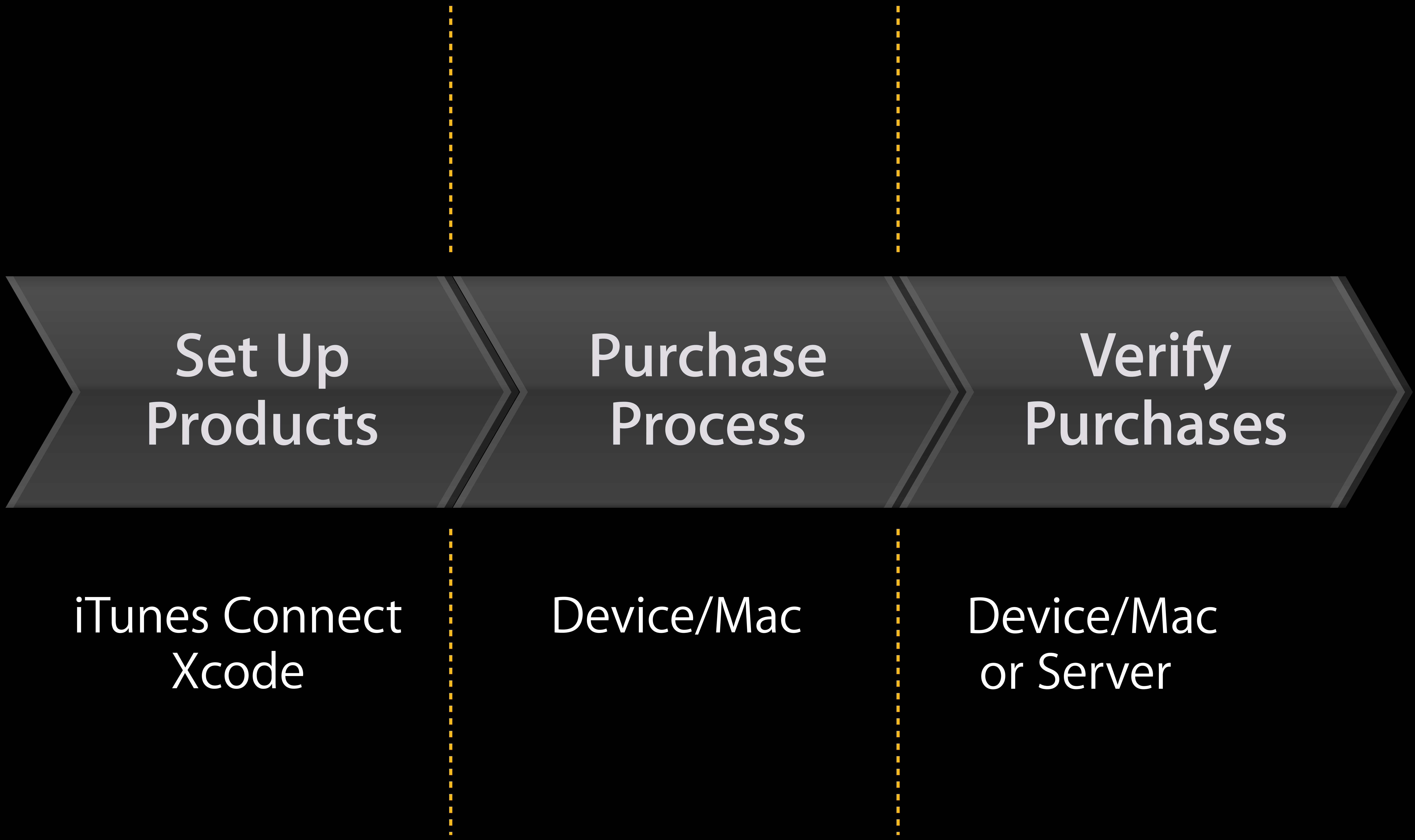


iTunes Connect
Xcode

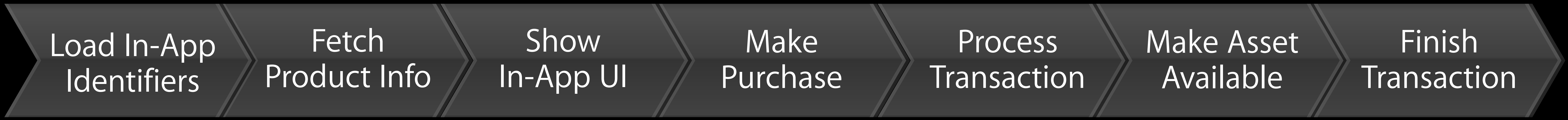
Three Phases



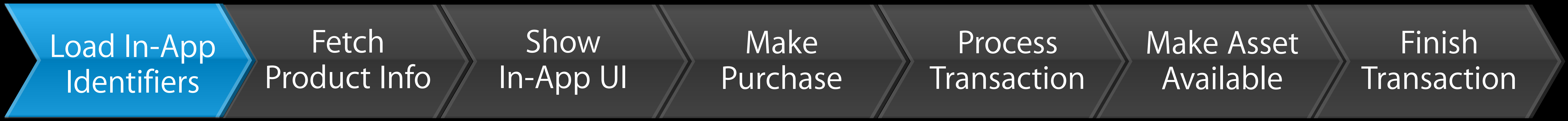
Three Phases



In-App Process Overview



In-App Process Overview



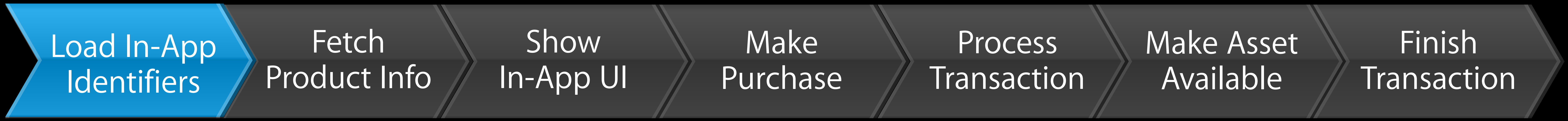
Product Identifiers

- Stored inside your app

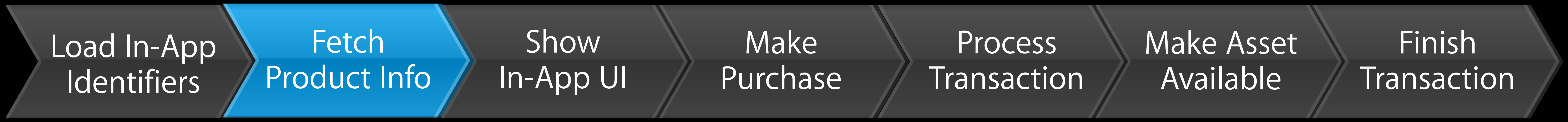
```
NSArray* productIdentifiers = @[@"com.myCompany.myApp.product1",
                                 @"com.myCompany.myApp.product2",
                                 @"com.myCompany.myApp.product3"];
```

- Fetched from your server
  - Good for dynamic catalog

In-App Process Overview



In-App Process Overview



Making a Product Request

Making a Product Request

```
NSSet* identifierSet = [NSSet setWithArray:productIdentifiers];
```

Making a Product Request

```
NSSet* identifierSet = [NSSet setWithArray:productIdentifiers];

SKProductsRequest* request = [[SKProductsRequest alloc]
                initWithProductIdentifiers: identifierSet];
```

Making a Product Request

```
NSSet* identifierSet = [NSSet setWithArray:productIdentifiers];

SKProductsRequest* request = [[SKProductsRequest alloc]
               initWithProductIdentifiers: identifierSet];

request.delegate = self;
[request start];
```

Product Response

`SKRequestDelegate` protocol

Product Response

SKRequestDelegate protocol

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

Product Response

SKRequestDelegate protocol

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

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

Product Response

SKRequestDelegate protocol

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

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

response.invalidProductIdentifiers
```

Product Response

SKRequestDelegate protocol

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

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

response.invalidProductIdentifiers
response.products
```

SKProduct Properties

```
response.products
```

- Localized title
- Localized description
- Price
- Price locale

SKProduct Properties

```
response.products
```

- Localized title
- Localized description
- Price
- Price locale
- Hosted
    - Content size
    - Content version

Formatting the Product Price

Formatting the Product Price

```
NSNumberFormatter *numberFormatter = [[NSNumberFormatter alloc] init];
```

Formatting the Product Price

```
NSNumberFormatter *numberFormatter = [[NSNumberFormatter alloc] init];

[numberFormatter setNumberStyle:NSNumberFormatterCurrencyStyle];
[numberFormatter setLocale:product.priceLocale];
```

Formatting the Product Price

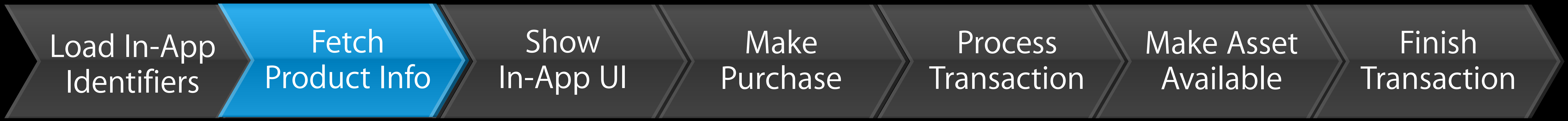
```
NSNumberFormatter *numberFormatter = [[NSNumberFormatter alloc] init];

[numberFormatter setNumberStyle:NSNumberFormatterCurrencyStyle];
[numberFormatter setLocale:product.priceLocale];

NSString *formattedString = [numberFormatter
          stringFromNumber:product.price];
```

Formatting the Product Price

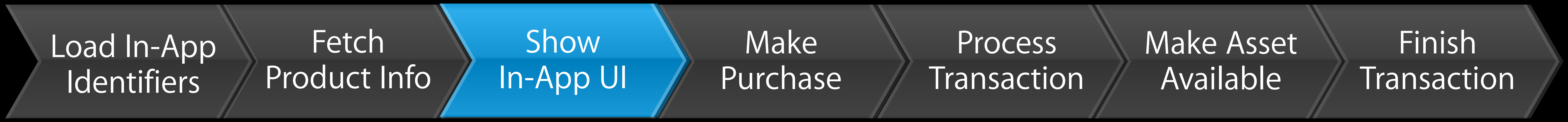
```
NSNumberFormatter *numberFormatter = [[NSNumberFormatter alloc] init];
[numberFormatter setNumberStyle:NSNumberFormatterCurrencyStyle];
[numberFormatter setLocale:product.priceLocale];
NSString *formattedString = [numberFormatter
          stringFromNumber:product.price];
```

- Do not perform currency conversion

In-App Process Overview

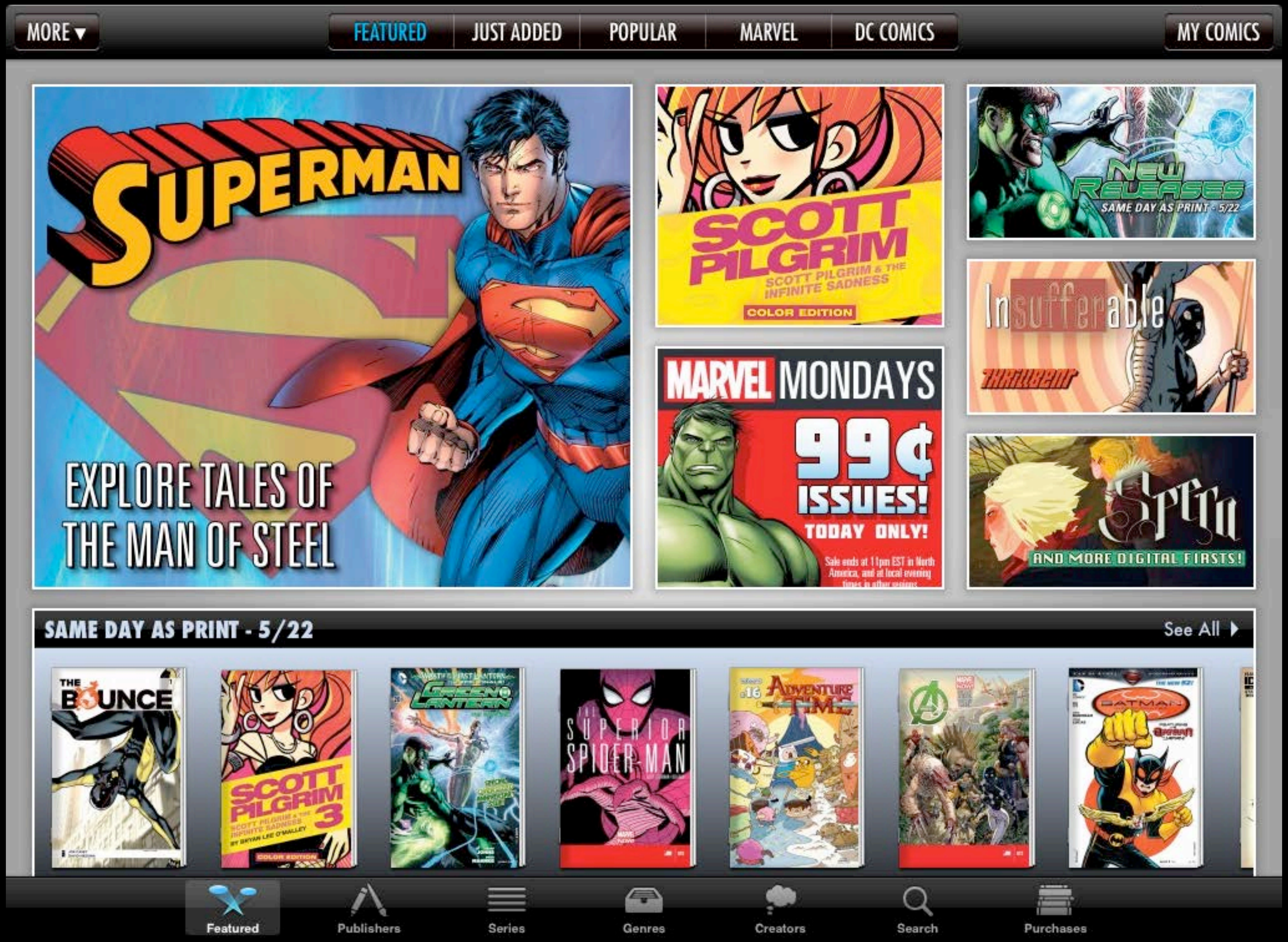


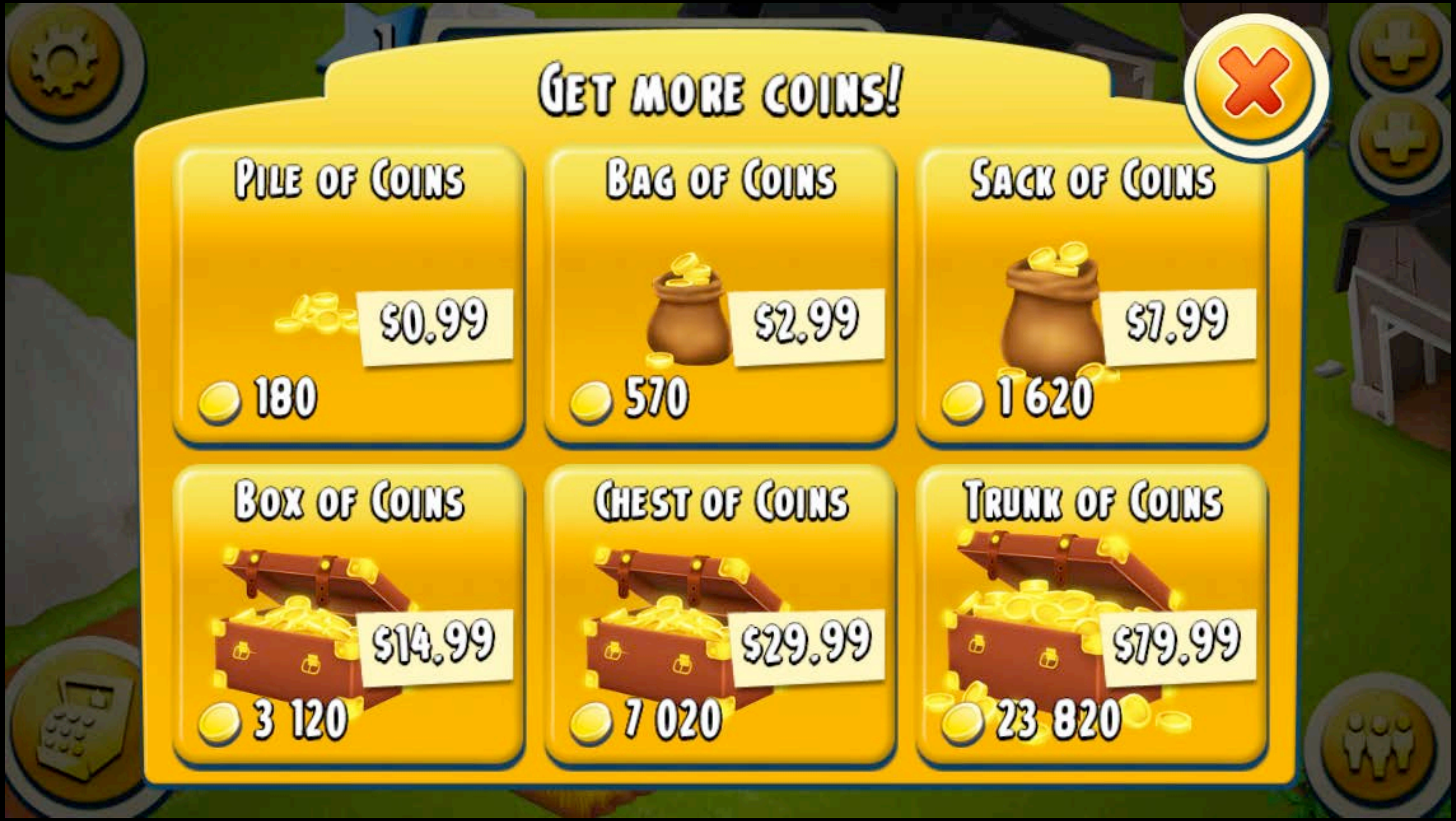
In-App Process Overview



In-App Purchase UI

- Up to the application
- Major effect on sales

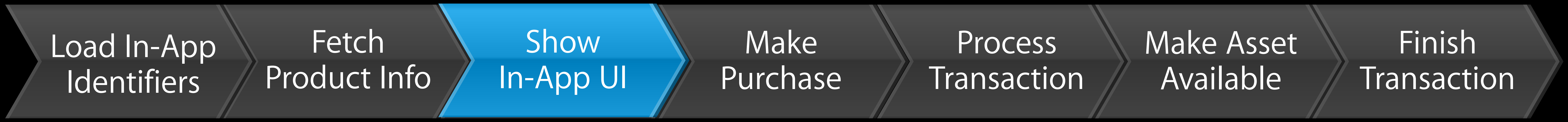

MORE ▾
FEATURED
JUST ADDED
POPULAR
MARVEL
DC COMICS
MY COMICS
SUPERMAN
EXPLORE TALES OF THE MAN OF STEEL
SCOTT PILGRIM
SCOTT PILGRIM & THE INFINITE SADNESS
COLOR EDITION
MARVEL MONDAYS
99¢ ISSUES!
TODAY ONLY!
NEW RELEASES
SAME DAY AS PRINT - 5/22
Insufferable
AND MORE DIGITAL FIRSTS!
SAME DAY AS PRINT - 5/22
See All
THE BOUNCE
SUPERIOR SPIDER-MAN
ADVENTURE TIME
Featured
Publishers
Series
Genres
Creators
Search
Purchases



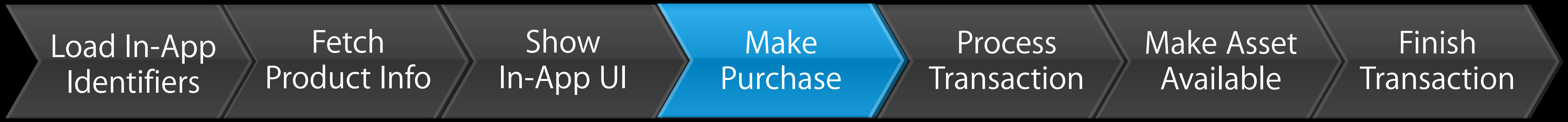
Get more coins!

Pile of Coins	Bag of Coins	Sack of Coins
$0.99	$2.99	$7.99
180	570	1 620
Box of Coins	Chest of Coins	Trunk of Coins
$14.99	$29.99	$79.99
3 120	7 020	23 820

In-App Process Overview



In-App Process Overview



Requesting a Payment

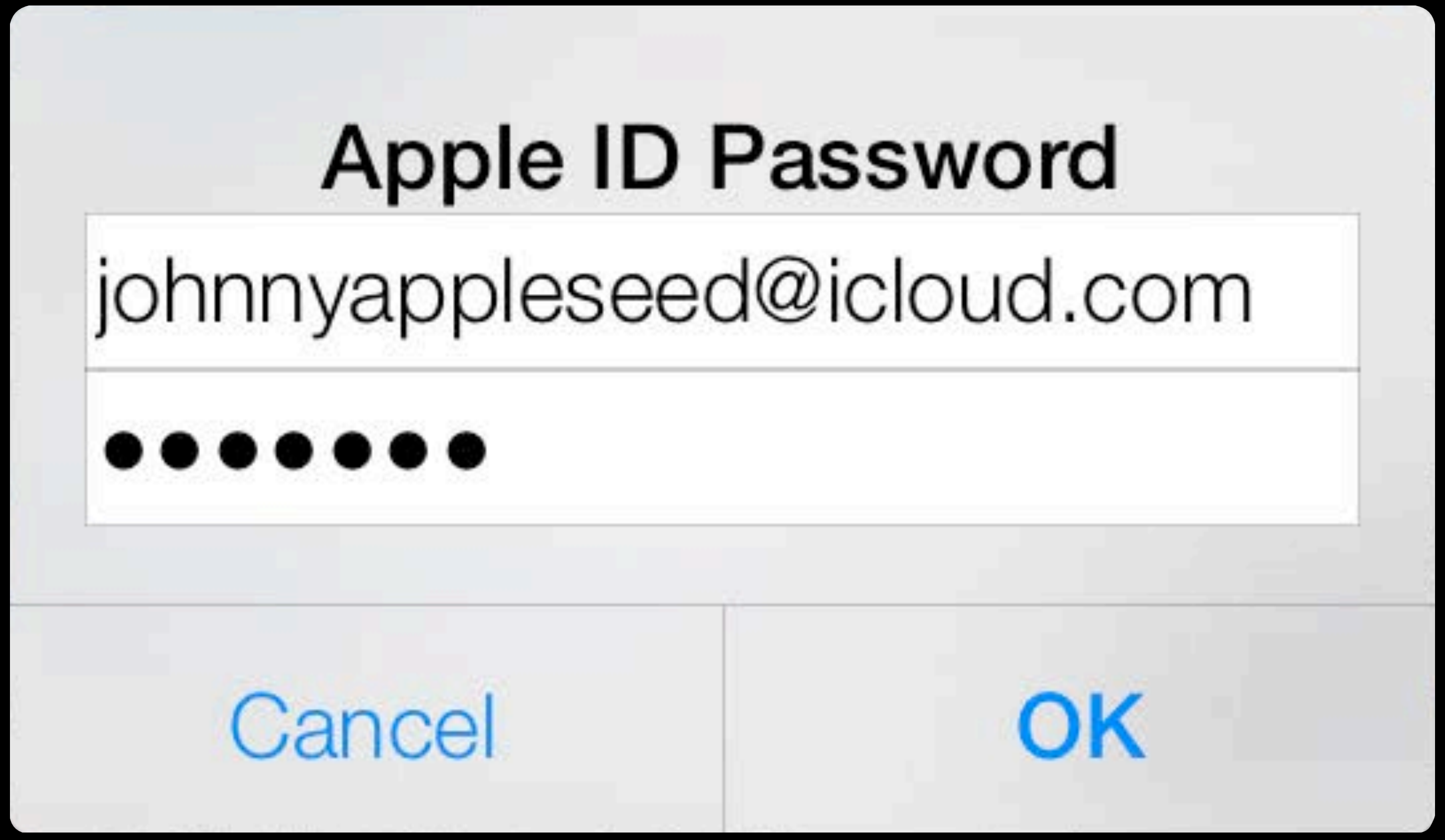
Requesting a Payment

```
SKPayment* payment = [SKPayment paymentWithProduct:product];
```

Requesting a Payment

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

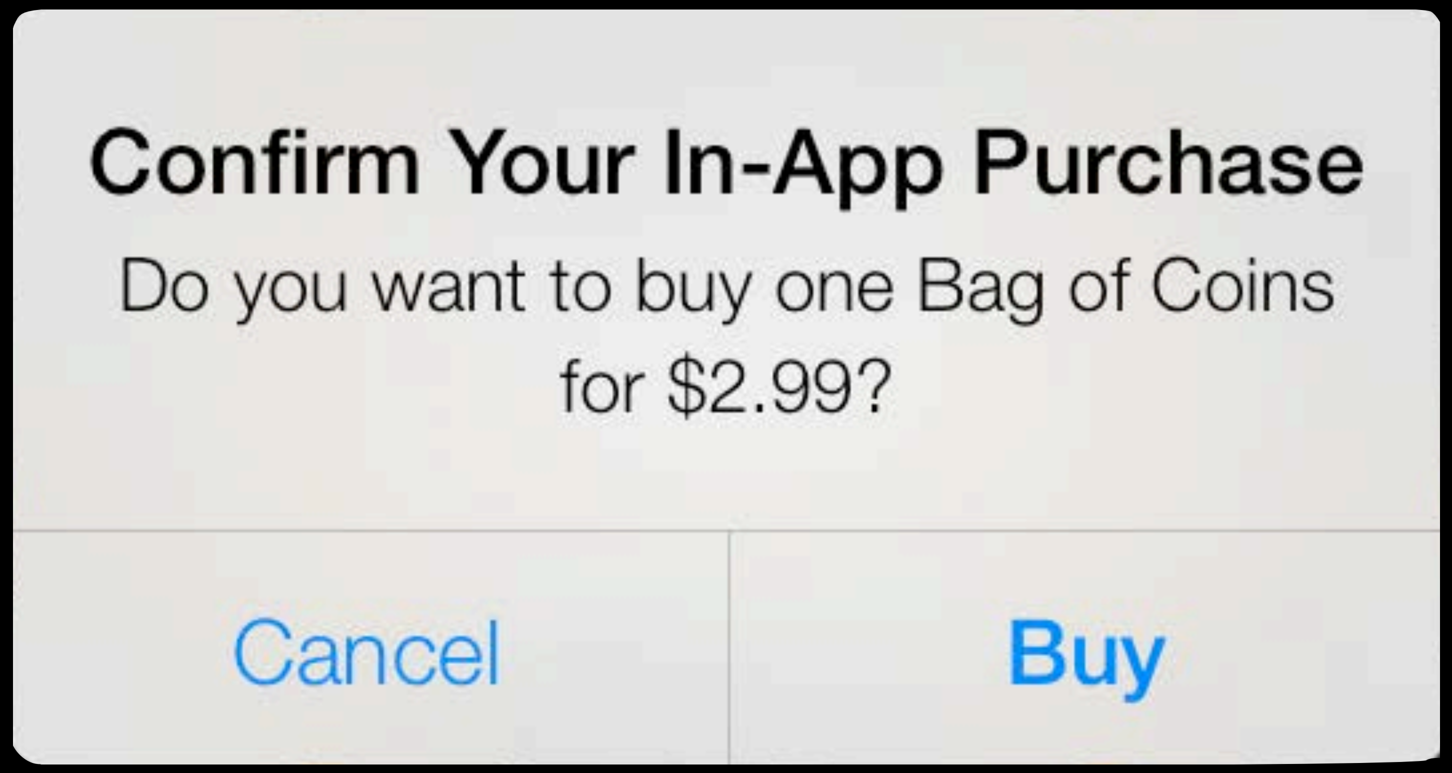
Requesting a Payment


Apple ID Password

johnnyappleseed@icloud.com

●●●●●●●

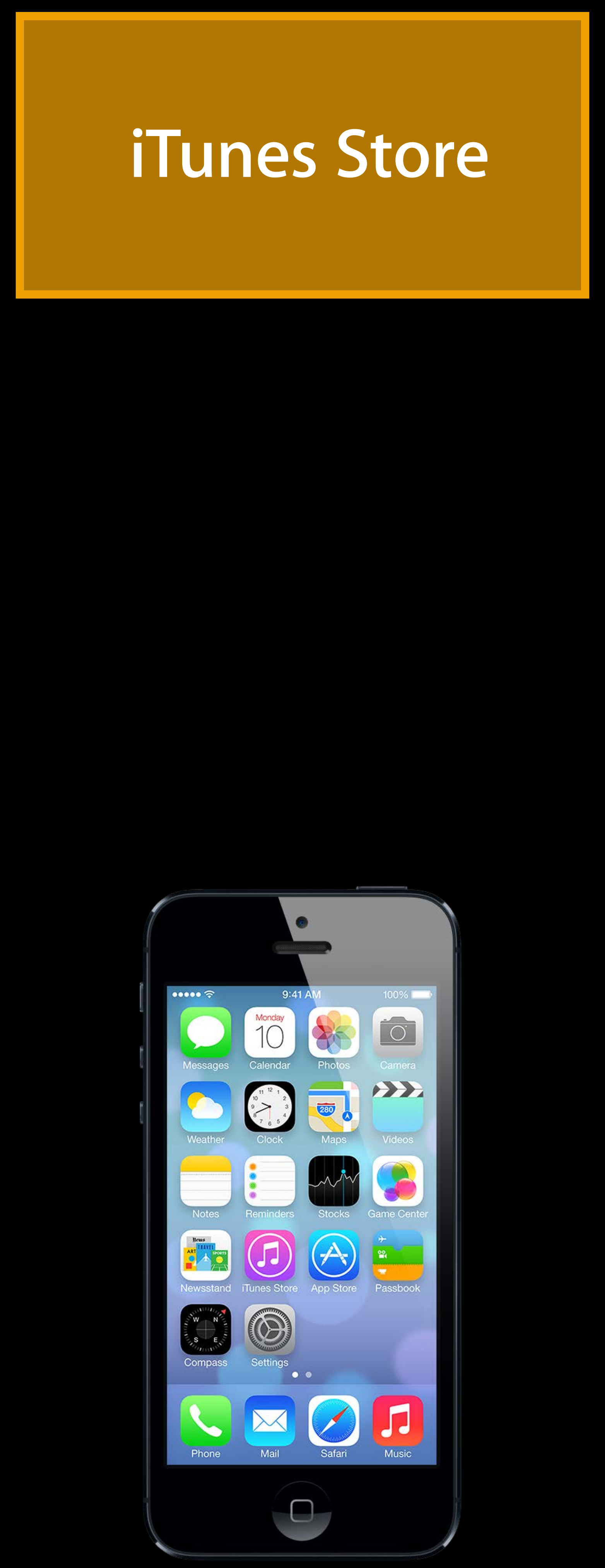
Requesting a Payment



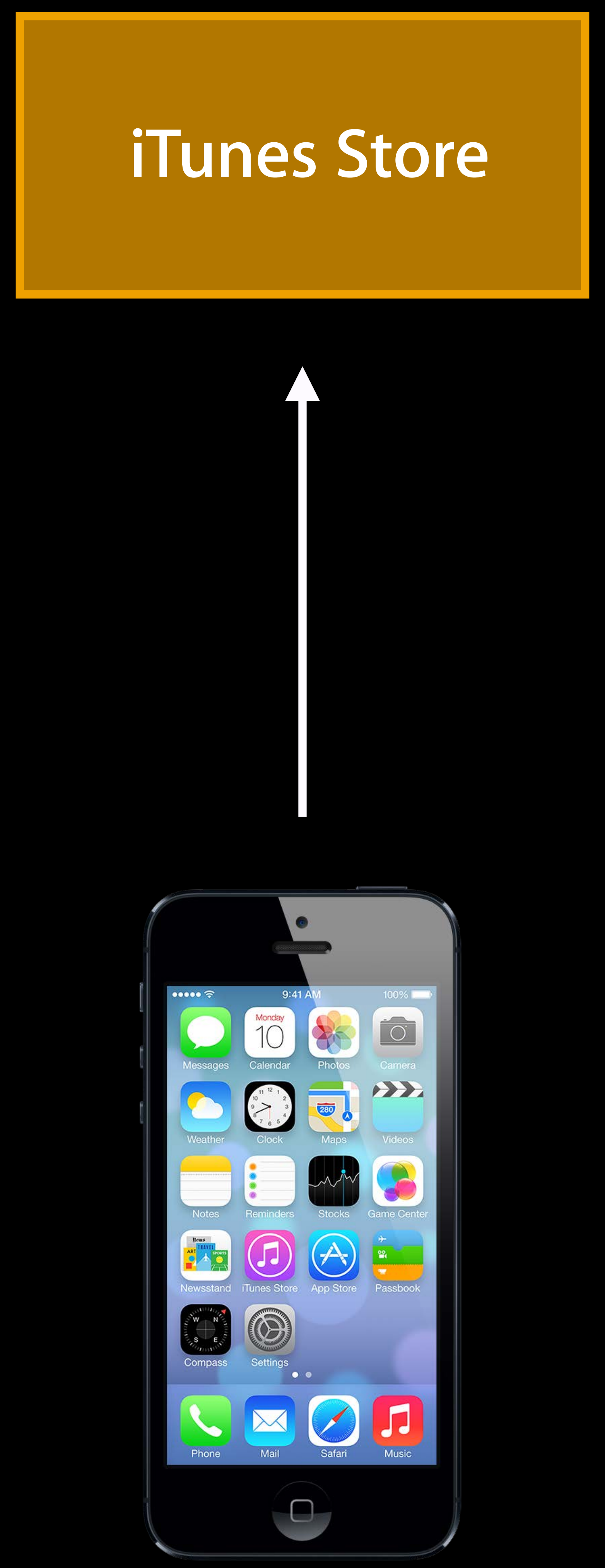
Confirm Your In-App Purchase

Do you want to buy one Bag of Coins for $2.99?

Requesting a Payment

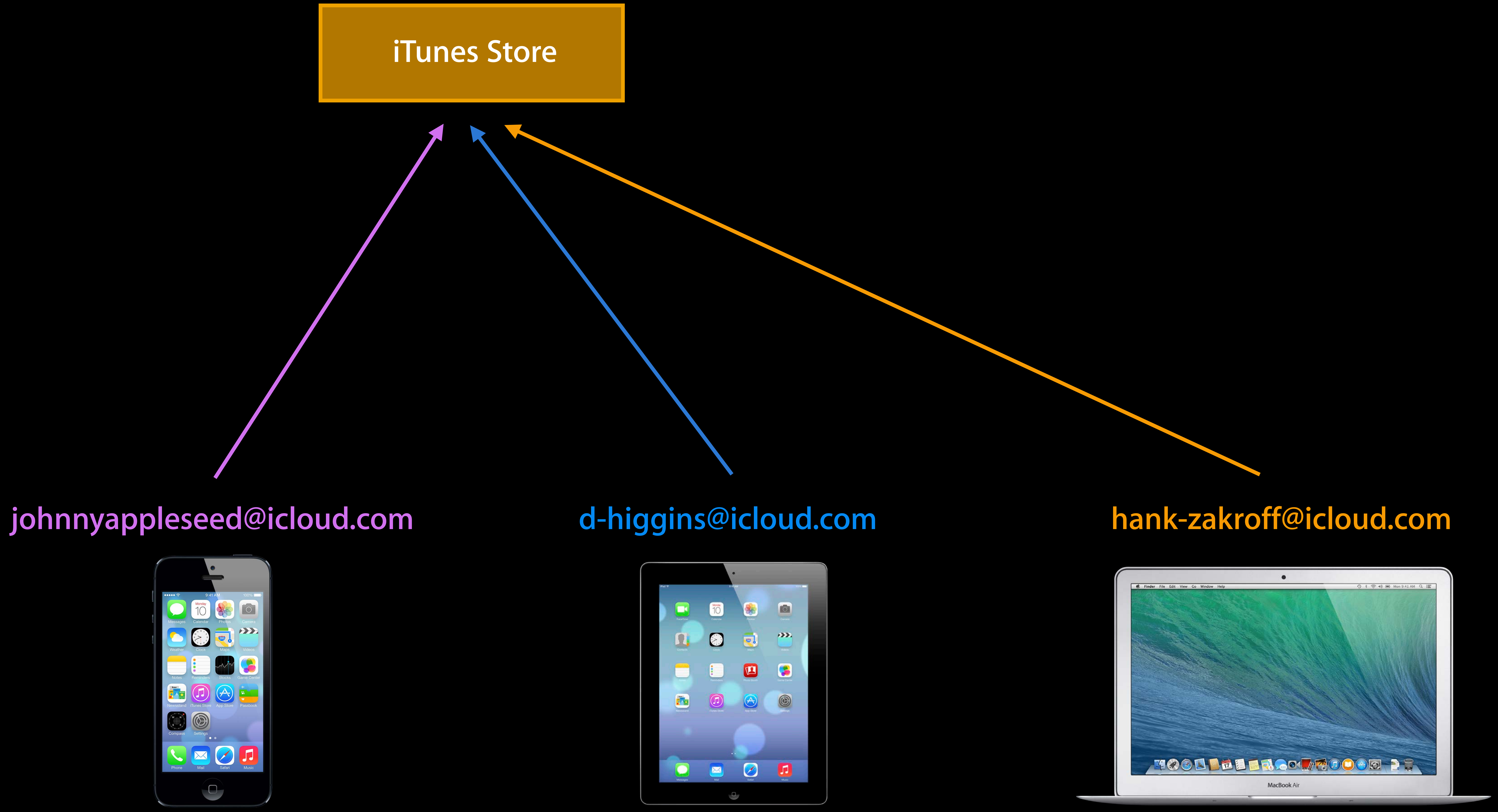


Requesting a Payment

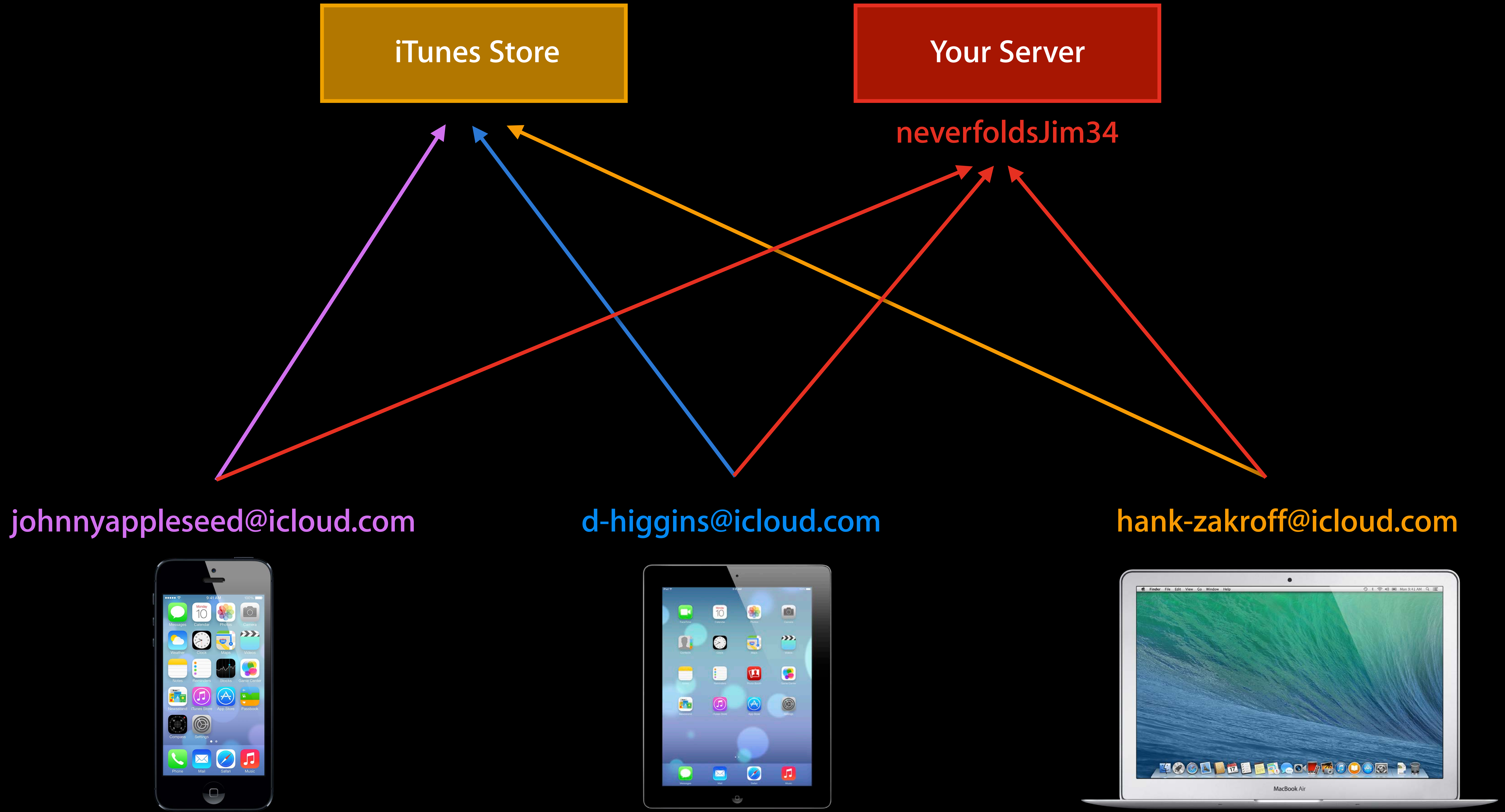


Detecting Irregular Activity

Detecting Irregular Activity



Detecting Irregular Activity



Detecting Irregular Activity

Provide account identifier

NEW

- For applications with their own account management
- Provide an opaque identifier for your user's account
  - Don't send us the Apple ID!
  - Don't provide the account name!
  - Don't provide the password either!
  - We suggest a hash of the account name

Detecting Irregular Activity

Provide account identifier

- For applications with their own account management
- Provide an opaque identifier for your user's account
    - Don't send us the Apple ID!
    - Don't provide the account name!
    - Don't provide the password either!
    - We suggest a hash of the account name

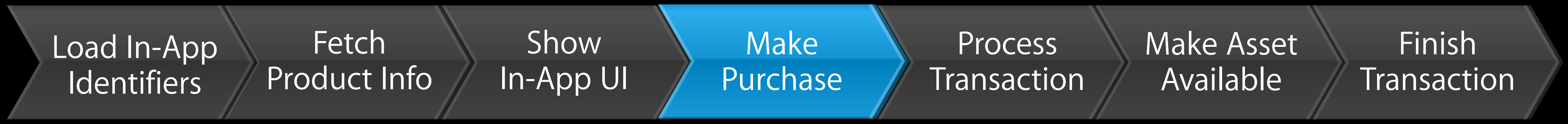
```
SKPayment *payment = [SKPayment paymentWithProduct:product];
```

Detecting Irregular Activity
Provide account identifier

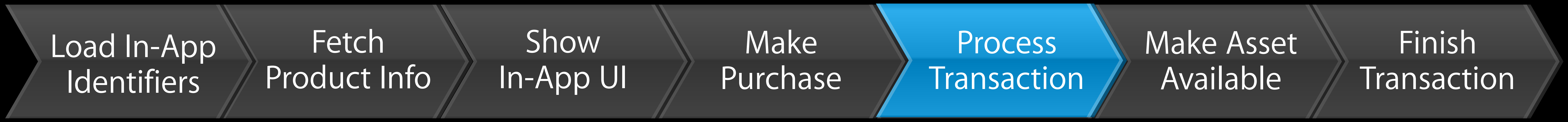
- For applications with their own account management
- Provide an opaque identifier for your user's account
    - Don't send us the Apple ID!
    - Don't provide the account name!
    - Don't provide the password either!
    - We suggest a hash of the account name

```
SKPayment *payment = [SKPayment paymentWithProduct:product];
payment.applicationUsername = hash(customerAccountName);
```

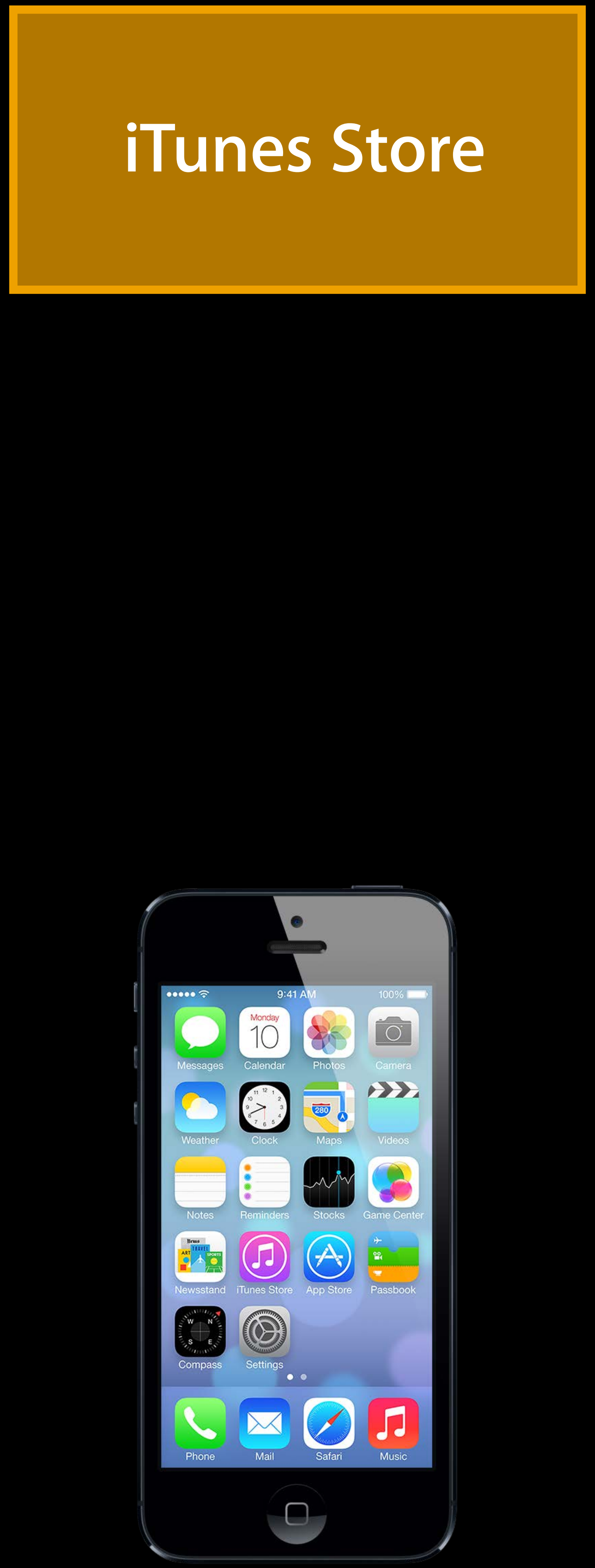
In-App Process Overview



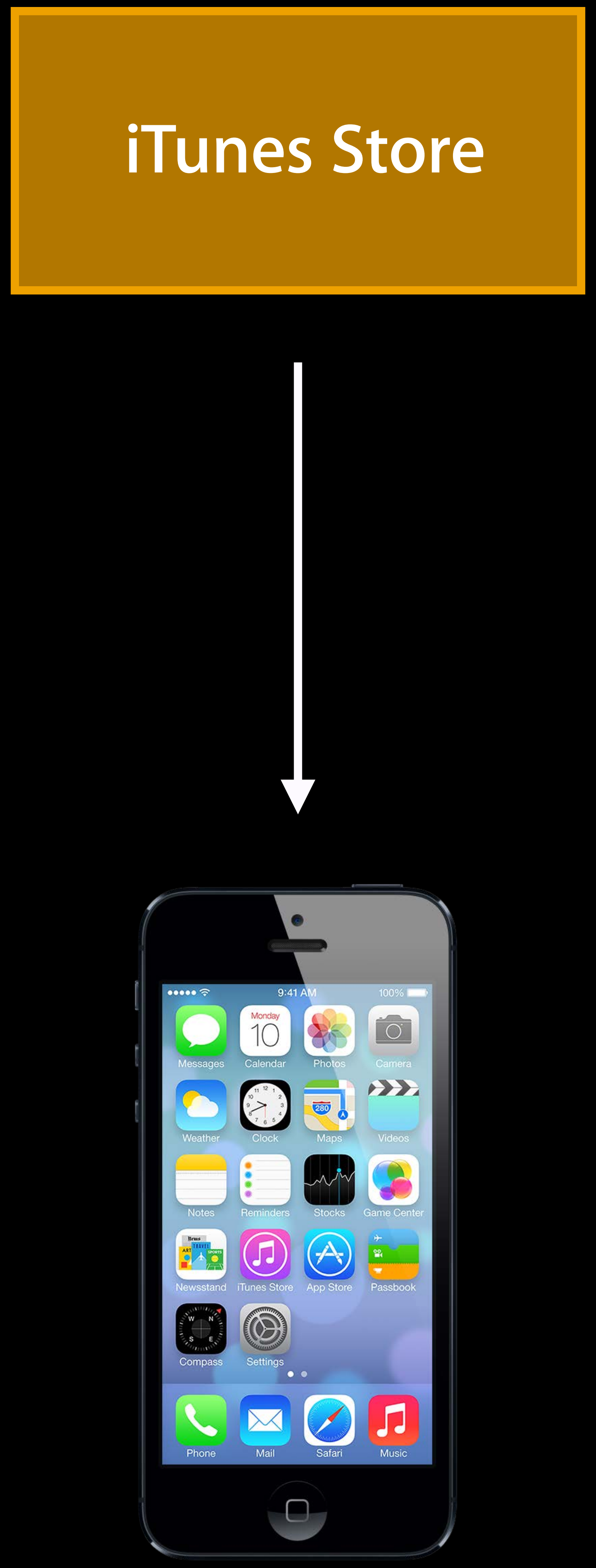
In-App Process Overview



Processing a Transaction



Processing a Transaction



Observing the Payment Queue

SKPaymentTransactionObserver protocol

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

Observing the Payment Queue

SKPaymentTransactionObserver protocol

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

   for (SKPaymentTransaction* transaction in transactions)
   {








   }
```

Observing the Payment Queue

SKPaymentTransactionObserver protocol

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

   for (SKPaymentTransaction* transaction in transactions)
   {
       switch(transaction.transactionState) {







       }

   }
```

Observing the Payment Queue

SKPaymentTransactionObserver protocol

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

   for (SKPaymentTransaction* transaction in transactions)
   {
       switch(transaction.transactionState) {

          case SKPaymentTransactionStatePurchased:




       }

   }
```

Observing the Payment Queue

SKPaymentTransactionObserver protocol

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

   for (SKPaymentTransaction* transaction in transactions)
   {
       switch(transaction.transactionState) {

          case SKPaymentTransactionStatePurchased:
                   NSData* receipt = transaction.receipt;



       }

   }
```

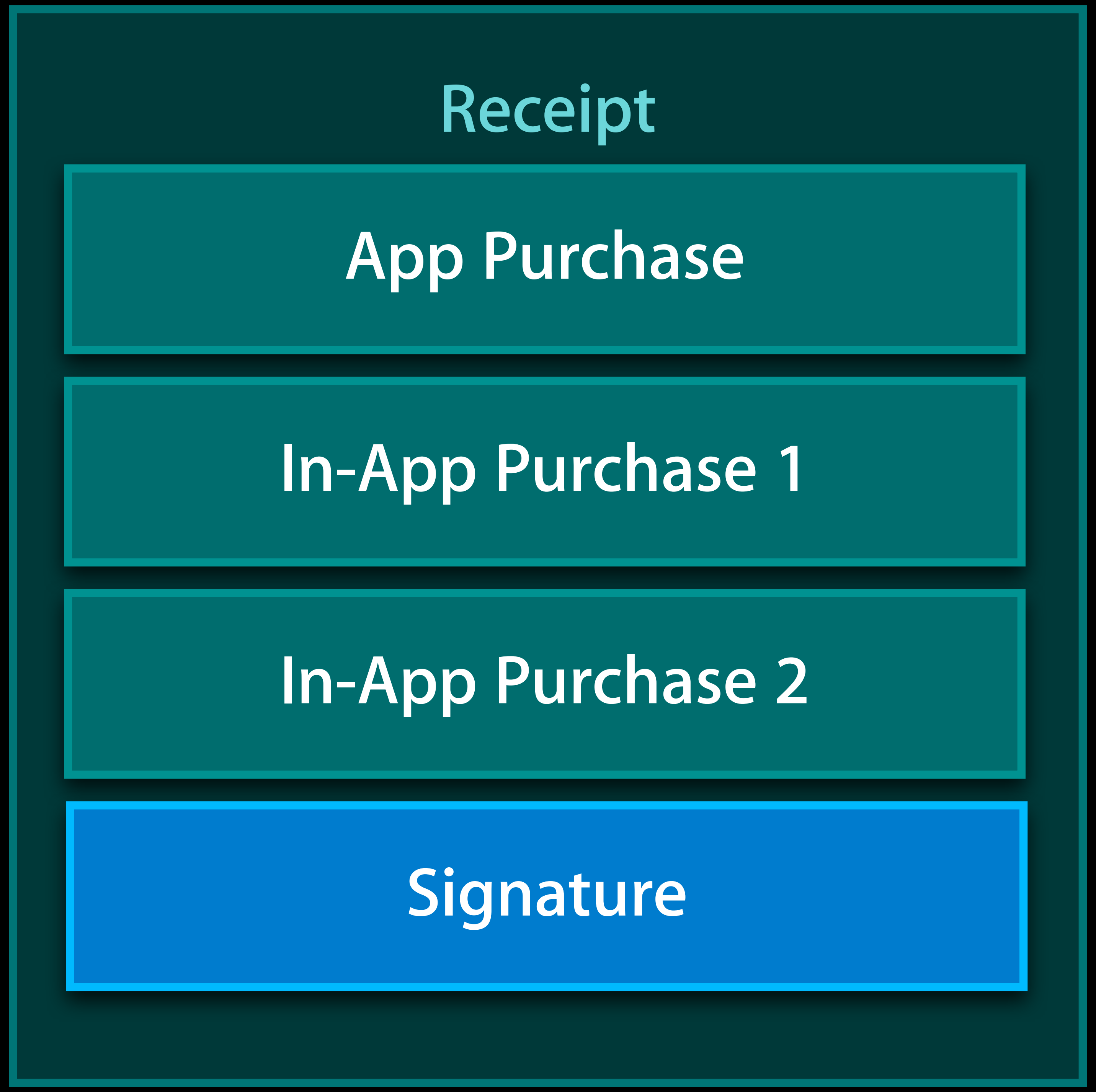
Observing the Payment Queue

SKPaymentTransactionObserver protocol

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

   for (SKPaymentTransaction* transaction in transactions)
   {
       switch(transaction.transactionState) {

          case SKPaymentTransactionStatePurchased:
                  NSURL* receiptURL = [[NSBundle mainBundle] appStoreReceiptURL];
                  NSData* receipt = [NSData dataWithContentsOfURL:receiptURL];


       }

   }
```

Verifying the Receipt

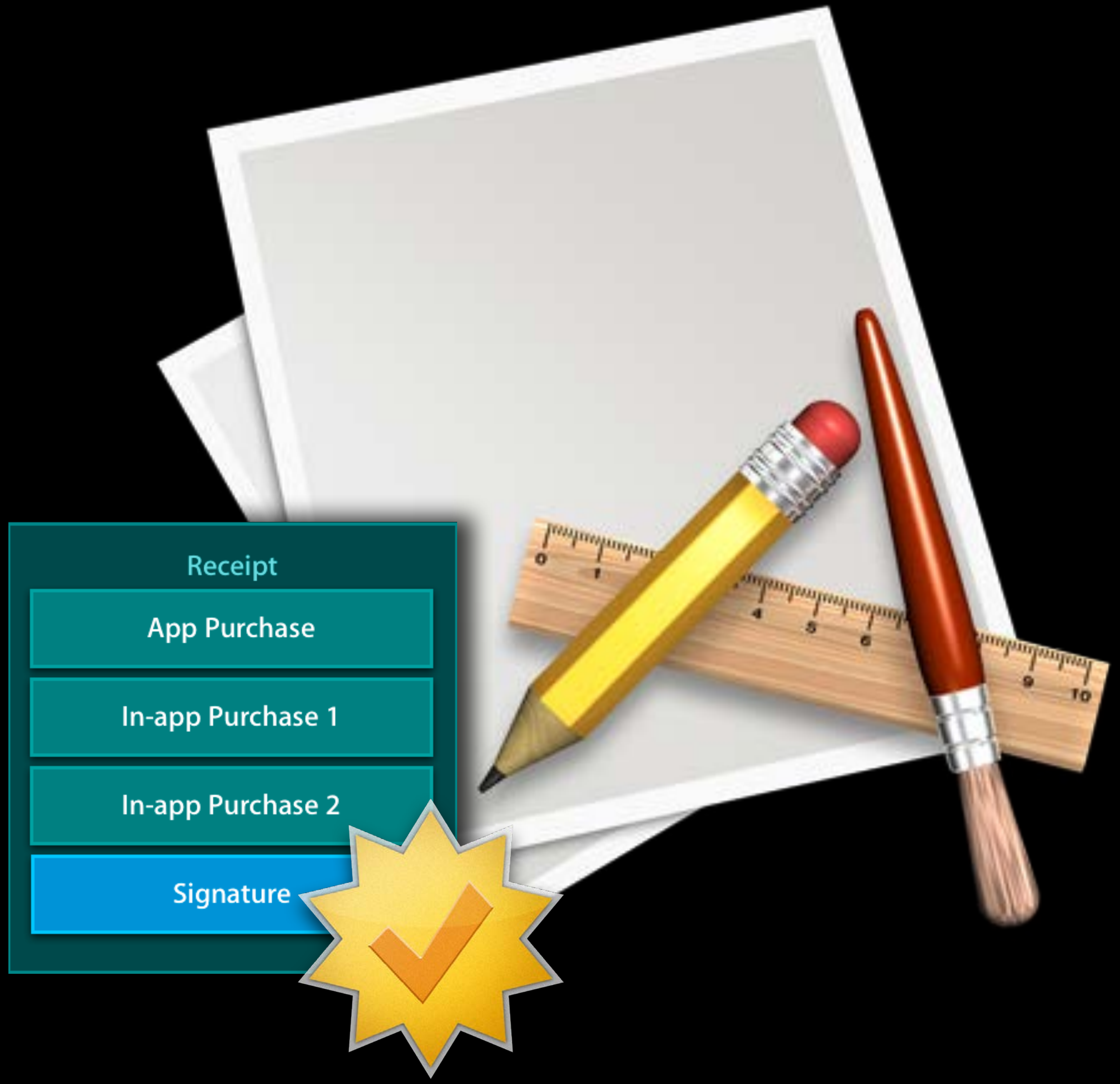
- Make it as strong as you deem necessary
- Verify on device
- Verify on server
- Or both



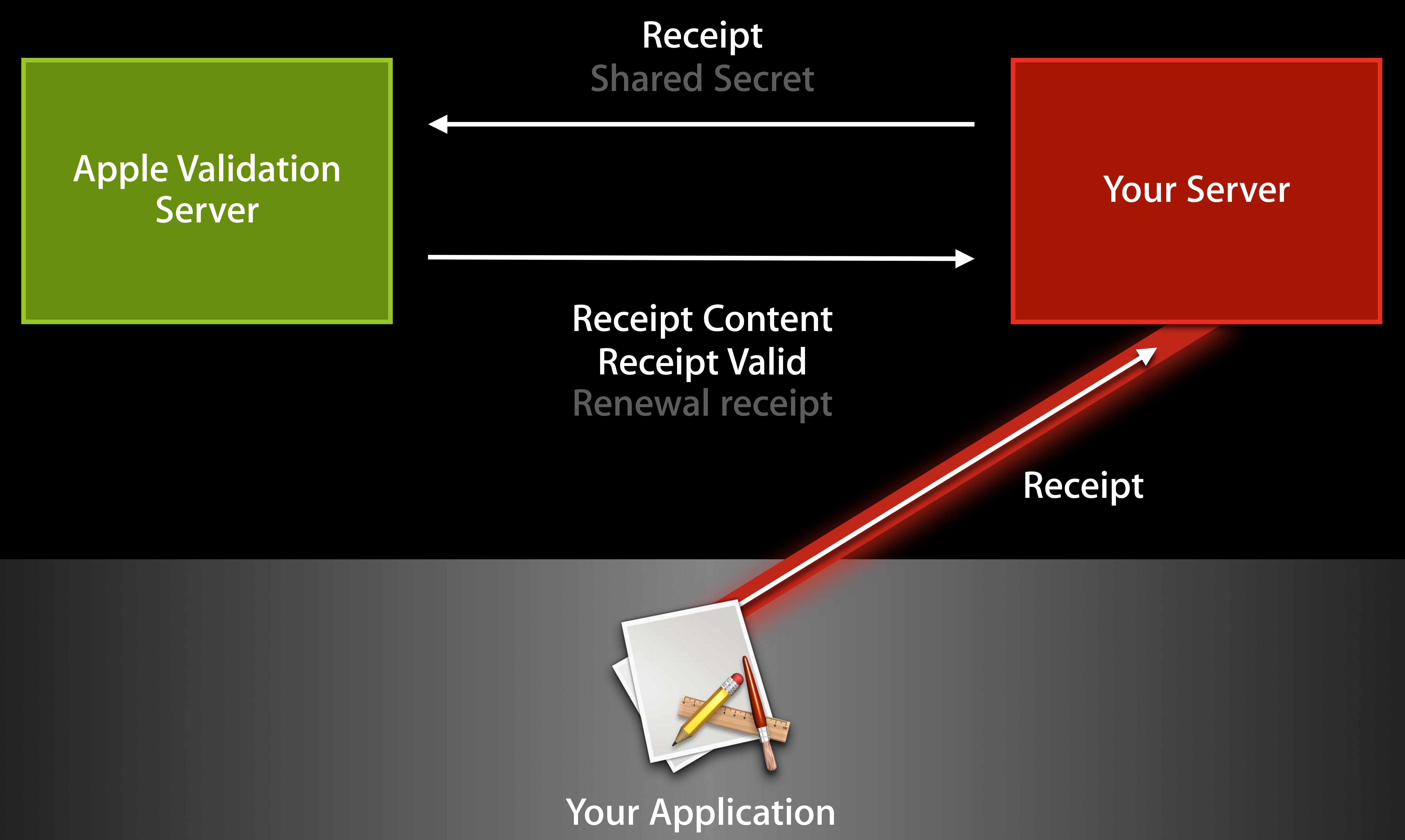
Local Receipt Validation

NEW

- Preferred method
- As secure as OS X
- Works anytime
- Works offline



Server-Based Receipt Validation

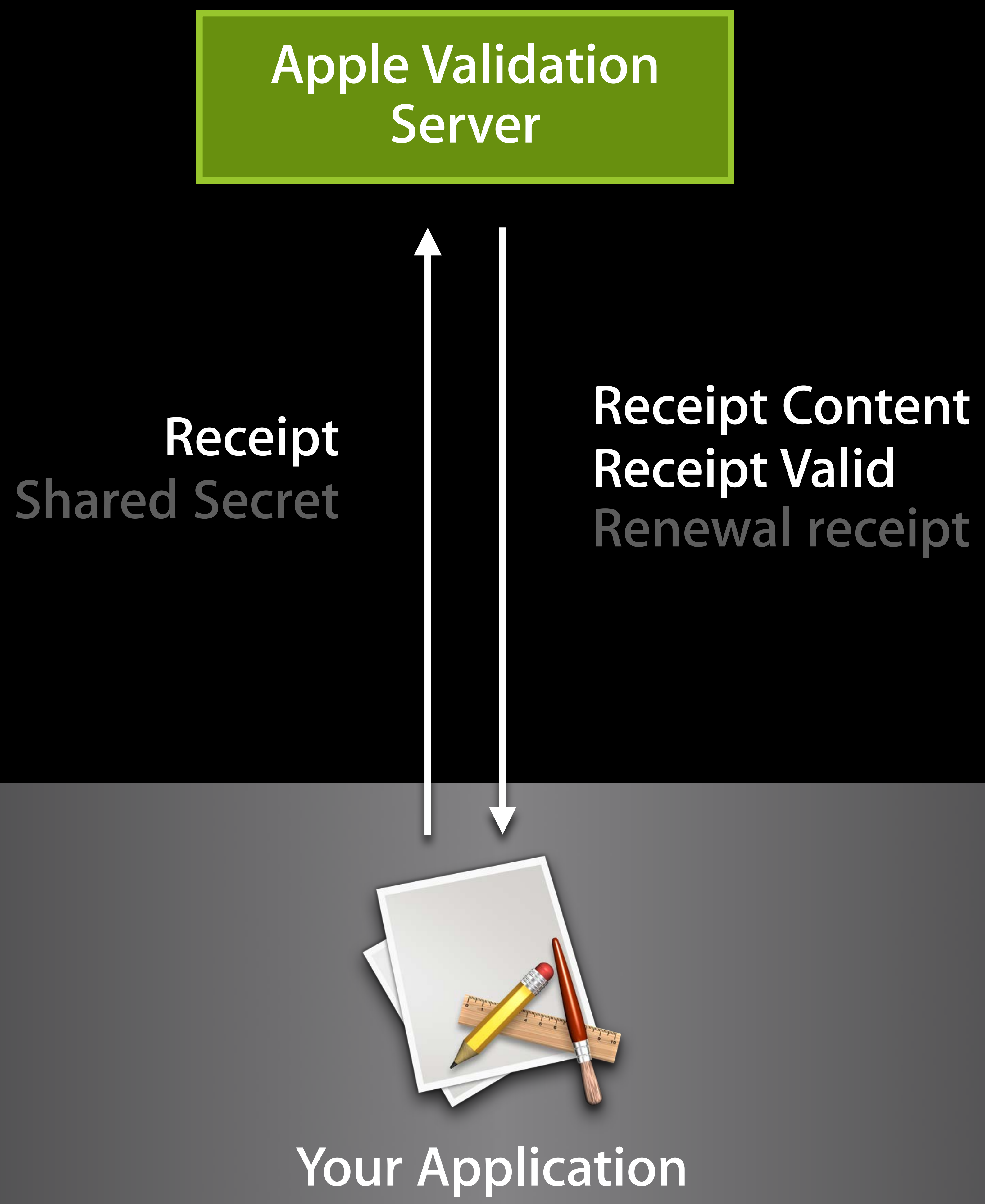


Server-Based Receipt Validation

- Your server validates the receipt against Apple servers
- You own the secure connection to your server
    - SSL, EV cert validation, cert pinning
- Useful for auto-renewable subscriptions
- Requires an internet connection

The Wrong Way

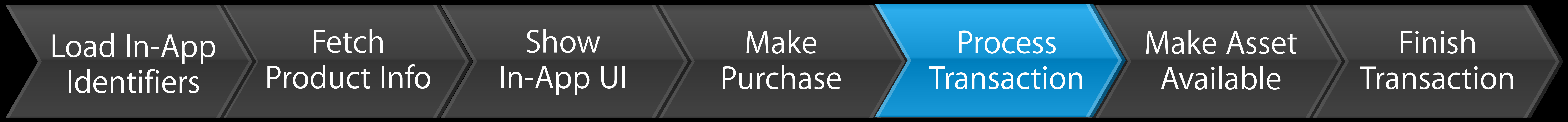
Validating against Apple servers from your app



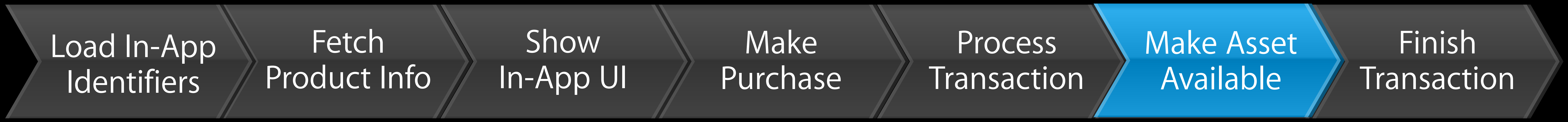
Related Sessions

Using Receipts to Protect Your Digital Sales	Presidio Thursday 2:00PM

In-App Process Overview



In-App Process Overview



Four Download APIs

- Unlock functionality in your app
- Download additional content
    - Hosted in-app purchases
    - Newsstand Kit downloads
    - iOS Background download API
    - Classic download

Hosted In-App Purchases

- From Apple servers
- Scalable and reliable
- Download in the background
- Go through review
- 2GB limit per product identifier

Hosted In-App Purchases

Initiating the download

Hosted In-App Purchases

Initiating the download

```
- (void)paymentQueue:(SKPaymentQueue *)queue
        updatedTransactions:(NSArray *)transactions
    for(SKPaymentTransaction* transaction in transactions)
    {




    }
```

Hosted In-App Purchases

Initiating the download

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

    for(SKPaymentTransaction* transaction in transactions)
    {

       if(transaction.downloads)




    }
```

Hosted In-App Purchases

Initiating the download

```
- (void)paymentQueue:(SKPaymentQueue *)queue
        updatedTransactions:(NSArray *)transactions
    for(SKPaymentTransaction* transaction in transactions)
    {
       if(transaction.downloads)
            [[SKPaymentQueue defaultQueue] startDownloads:
                                       transaction.downloads];
    }
```

Hosted In-App Purchases

Showing progress

Hosted In-App Purchases

Showing progress

```
- (void)paymentQueue:(SKPaymentQueue *)queue
      updatedDownloads:(NSArray *)downloads;
```

Hosted In-App Purchases

Showing progress

```
- (void)paymentQueue:(SKPaymentQueue *)queue
      updatedDownloads:(NSArray *)downloads;

     download.progress

     download.timeRemaining
```

Hosted In-App Purchases

Showing progress

```
- (void)paymentQueue:(SKPaymentQueue *)queue
      updatedDownloads:(NSArray *)downloads;
     download.progress
     download.timeRemaining
     download.state
     download.error
```

Hosted In-App Purchases

Showing progress

```
- (void)paymentQueue:(SKPaymentQueue *)queue
      updatedDownloads:(NSArray *)downloads;

     download.progress

     download.timeRemaining

     download.state

     download.error

   When download.state is SKDownloadStateFinished

     download.contentURL
```

Newsstand Kit Downloads

- iOS only
- Only for periodicals
- From your own server
- Download in the background
- App can update icon upon completion

Newsstand Kit Downloads

Initiating the download

Newsstand Kit Downloads

Initiating the download

```
NKLibrary library = [NKLibrary sharedLibrary];
```

Newsstand Kit Downloads

Initiating the download

```
NKLibrary library = [NKLibrary sharedLibrary];
NKIssue* issue = [library addIssueWithName:product.localizedTitle
                                        date:date];
```

Newsstand Kit Downloads

Initiating the download

```
NKLibrary library = [NKLibrary sharedLibrary];
NKIssue* issue = [library addIssueWithName:product.localizedTitle
                                       date:date];
NSURLRequest* request = [NSURLRequest requestWithURL:url];
NKAssetDownload* download = [issue addAssetWithRequest:request];
```

Newsstand Kit Downloads

Initiating the download

```
NKLibrary library = [NKLibrary sharedLibrary];
NKIssue* issue = [library addIssueWithName:product.localizedTitle
                                           date:date];
NSURLRequest* request = [NSURLRequest requestWithURL:url];
NKAssetDownload* download = [issue addAssetWithRequest:request];
[download downloadWithDelegate:self];
```

Newsstand Kit Downloads

Reconnecting to the downloads

Newsstand Kit Downloads

Reconnecting to the downloads

```
NKLibrary library = [NKLibrary sharedLibrary];
```

Newsstand Kit Downloads

Reconnecting to the downloads

```
NKLibrary library = [NKLibrary sharedLibrary];
for (NKAssetDownload* download in [library downloadingAssets]) {
```

Newsstand Kit Downloads

Reconnecting to the downloads

```
NKLibrary library = [NKLibrary sharedLibrary];
for (NKAssetDownload* download in [library downloadingAssets]) {
     [download downloadWithDelegate:self];
  }
```

Background Downloads

NEW

- iOS only
- From your own servers
- Download in the background
- Fine-grained cookie and credentials controls
- Power efficient

Background Downloads

Initiating the download

Background Downloads

Initiating the download

```
NSURLSessionConfiguration *config = [NSURLSessionConfiguration
             backgroundSessionConfiguration:@"MyBackgroundSession"];
```

Background Downloads

Initiating the download

```
NSURLSessionConfiguration *config = [NSURLSessionConfiguration
             backgroundSessionConfiguration:@"MyBackgroundSession"];
NSURLSession *session = [NSURLSession sessionWithConfiguration:config
                                 delegate:self delegateQueue:queue];
```

Background Downloads

Initiating the download

```
NSURLSessionConfiguration *config = [NSURLSessionConfiguration
            backgroundSessionConfiguration:@"MyBackgroundSession"];
NSURLSession *session = [NSURLSession sessionWithConfiguration:config
                             delegate:self delegateQueue:queue];
NSURLRequest *request = [NSURLRequest requestWithURL:myURL];
NSURLSessionDownloadTask *downloadTask = [session
                             downloadTaskWithRequest:request];
```

Background Downloads

NSURLSessionDownloadDelegate

```
- (void)URLSession:(NSURLSession *)session
      downloadTask:(NSURLSessionDownloadTask *)downloadTask
      didWriteData:(int64_t)bytesWritten
      totalBytesWritten:(int64_t)totalBytesWritten
      totalBytesExpectedToWrite:(int64_t)totalBytesExpectedToWrite
{
   // do something with progress
}
```

Background Downloads

NSURLSessionDownloadDelegate

```
- (void)URLSession:(NSURLSession *)session
      downloadTask:(NSURLSessionDownloadTask *)downloadTask
      didFinishDownloadingToURL:(NSURL *)location
{
   // copy the file to a safe location
   NSURL *newLocation = ...
   NSError *error = nil;
   [[NSFileManager defaultManager] copyItemAtURL:location
                                          toURL:newLocation error:&error];
}
```

Background Downloads

Reconnecting to the downloads

Background Downloads

Reconnecting to the downloads

```
- (void)application:(UIApplication *)application
        handleEventsForBackgroundURLSession:(NSString *)identifier
        completionHandler:(void (^)())completionHandler
```

Background Downloads

Reconnecting to the downloads

```
- (void)application:(UIApplication *)application
        handleEventsForBackgroundURLSession:(NSString *)identifier
        completionHandler:(void (^)())completionHandler
{
     NSURLSessionConfiguration *config = [NSURLSessionConfiguration
                 backgroundSessionConfiguration:identifier];
     NSURLSession *session = [NSURLSession sessionWithConfiguration:config
                                  delegate:self delegateQueue:queue];


}
```

Background Downloads

Reconnecting to the downloads

```
- (void)application:(UIApplication *)application
        handleEventsForBackgroundURLSession:(NSString *)identifier
        completionHandler:(void (^)())completionHandler
{
     NSURLSessionConfiguration *config = [NSURLSessionConfiguration
                backgroundSessionConfiguration:identifier];
     NSURLSession *session = [NSURLSession sessionWithConfiguration:config
                                   delegate:self delegateQueue:queue];
     self.completionHandler = completionHandler; // call when done
}
```

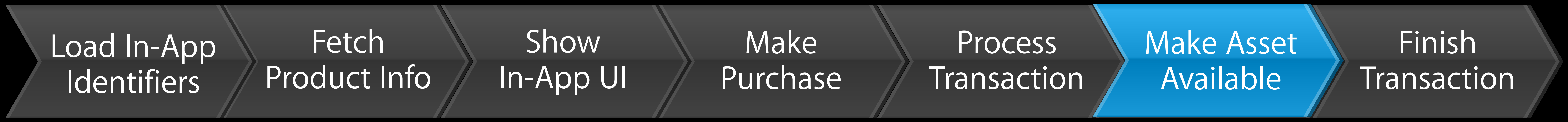
Related Sessions

What's New in Foundation Networking	Mission Wednesday 9:00AM

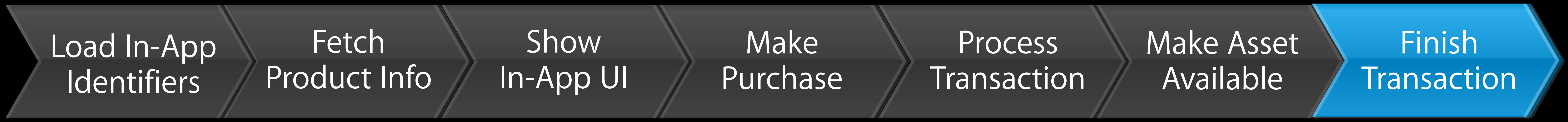
Classic Downloads

- Stops when apps get backgrounded
- User has to wait in the app
- Background task completions have changed
  - No longer prevent sleep
  - Not guaranteed to be immediately executed

In-App Process Overview



In-App Process Overview



Finish the Transaction

Finish the Transaction

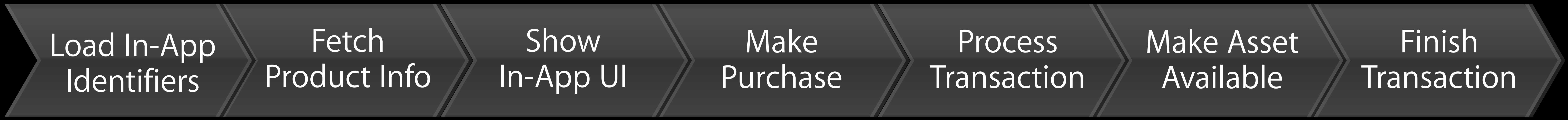
```
[[SKPaymentQueue defaultQueue] finishTransaction:transaction];
```

Finish the Transaction

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

- Improves launch times
- Reduces cellular data consumption

In-App Process Overview



Install Payment Queue Observer

At application launch time

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

- Transactions can happen any time
  - Network losses
  - User redeems a gift code
  - Subscription renewals
- In `appDidFinishLaunching`

Restoring In-App Purchases

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

- Nothing changes except the receipt

Agenda

- What's new in Store Kit
- Implementing in-app purchases
- **Using the test environment**
- Tips for passing app review

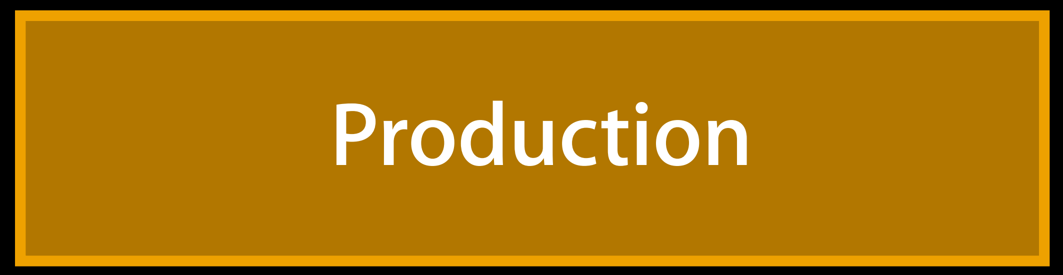
Using the Test Environment

The Test Environment

a.k.a. Sandbox

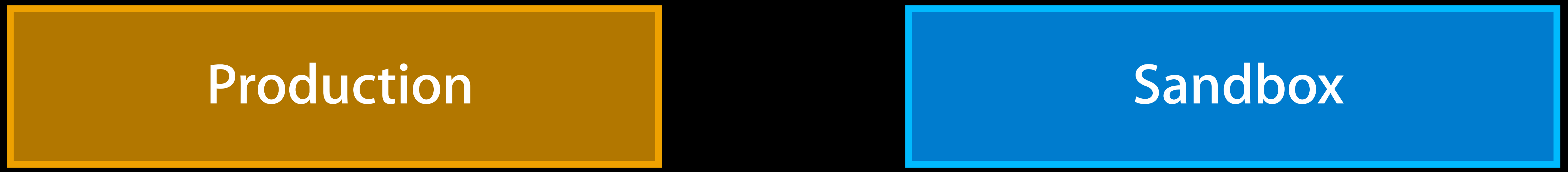
The Test Environment

a.k.a. Sandbox



The Test Environment

a.k.a. Sandbox



The Test Environment

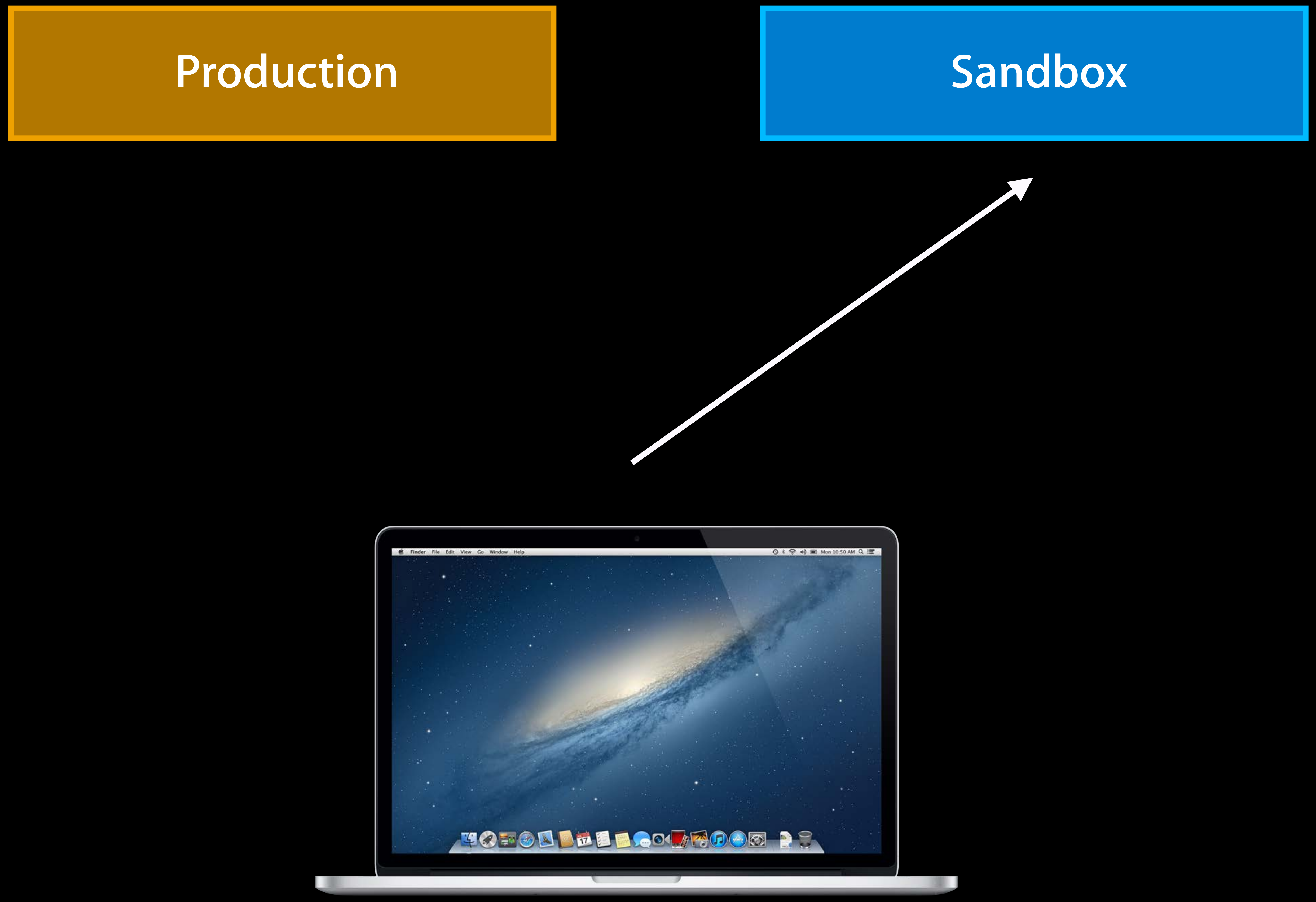
a.k.a. Sandbox

Production

Sandbox

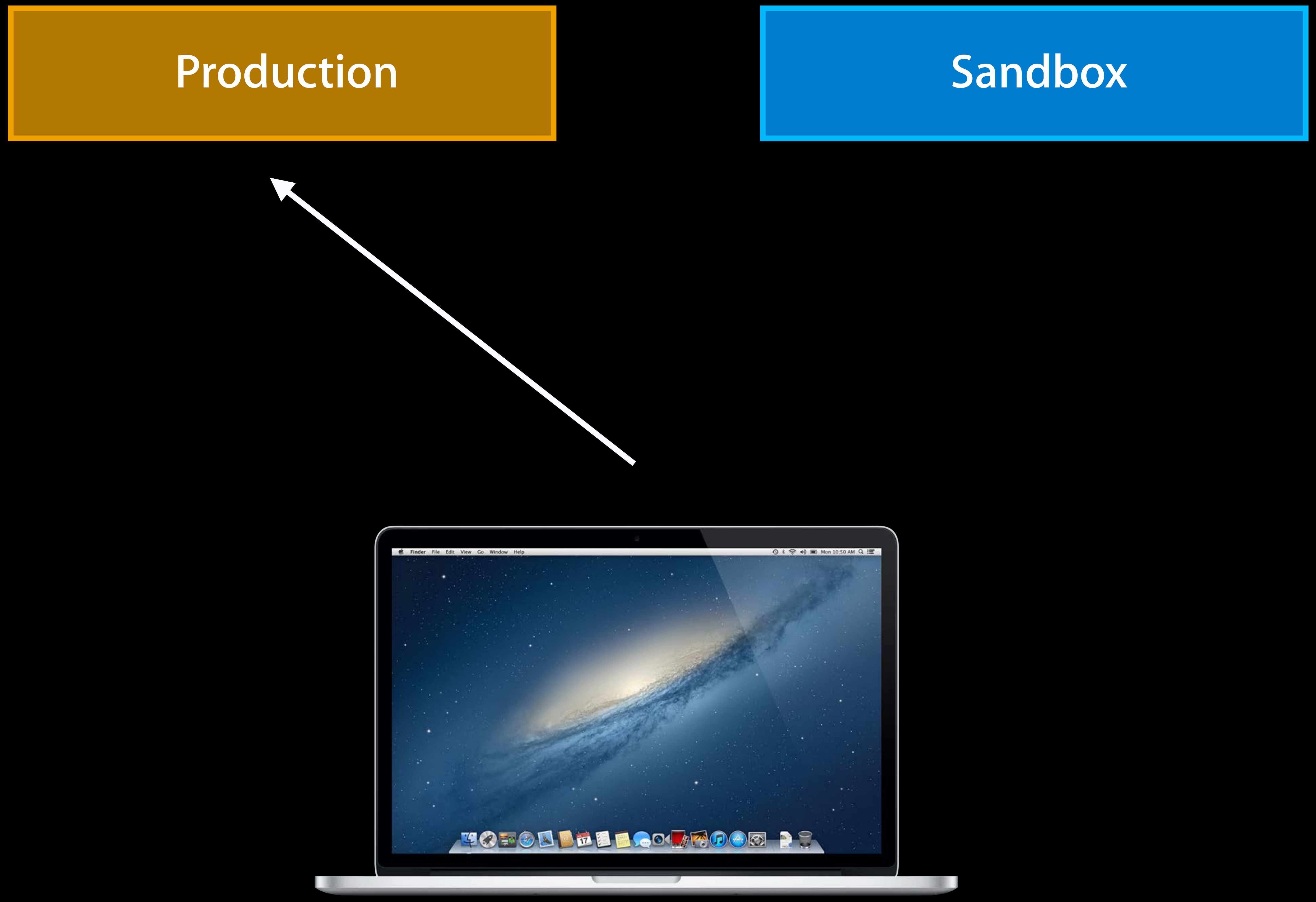
The Test Environment

a.k.a. Sandbox



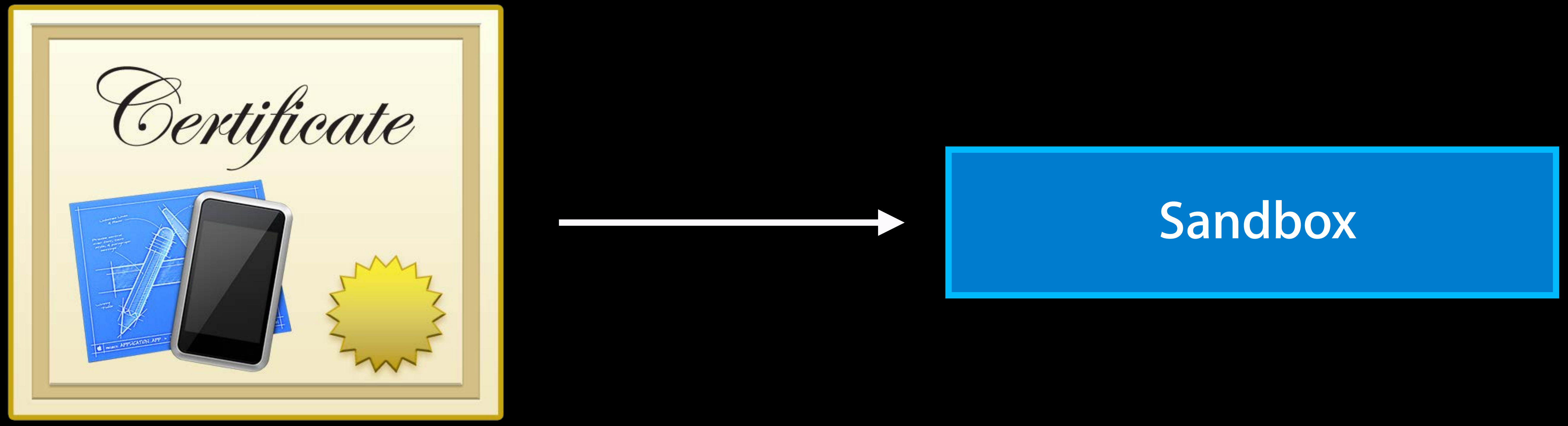
The Test Environment

a.k.a. Sandbox

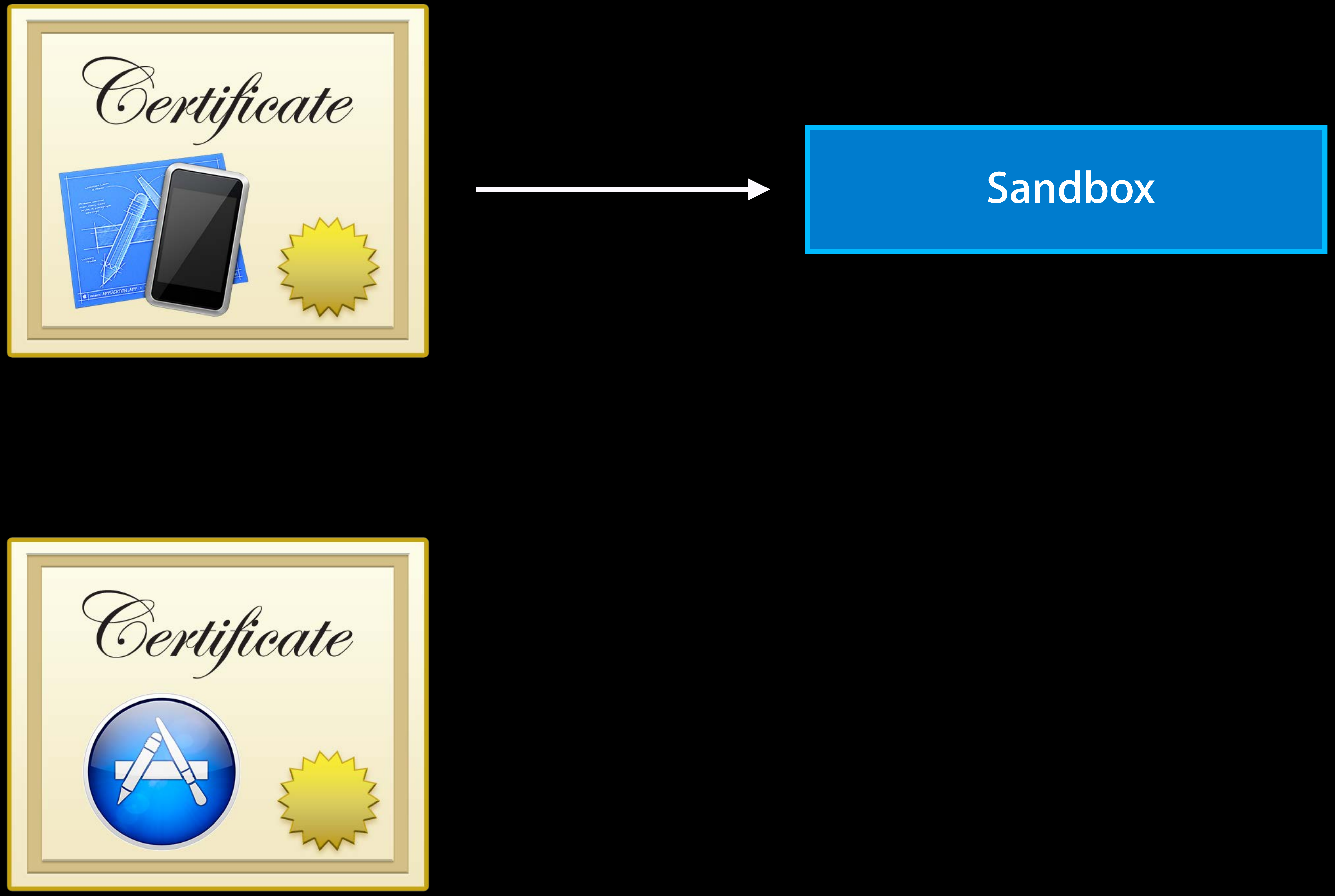


Based on Application Code Signing

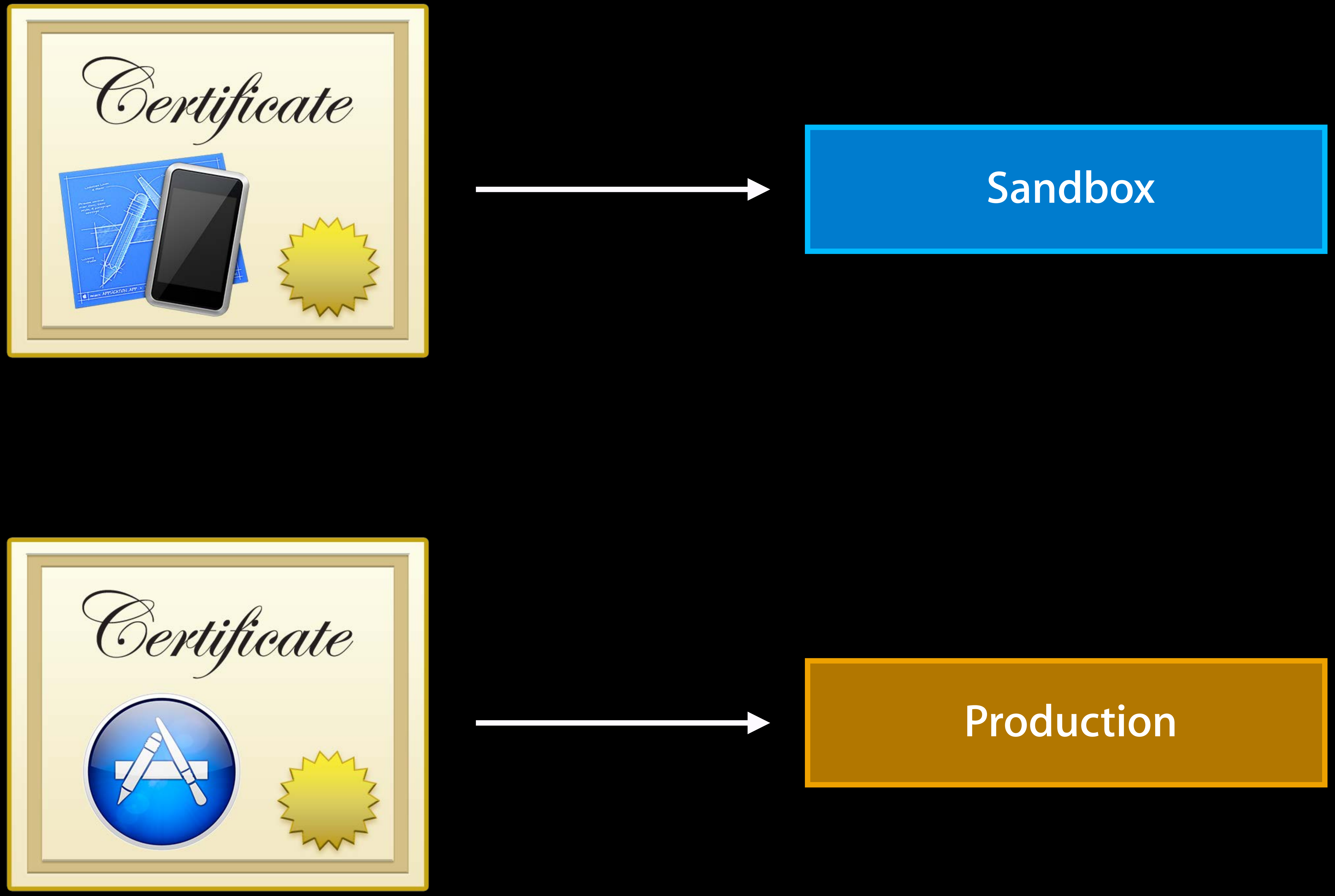
Based on Application Code Signing



Based on Application Code Signing

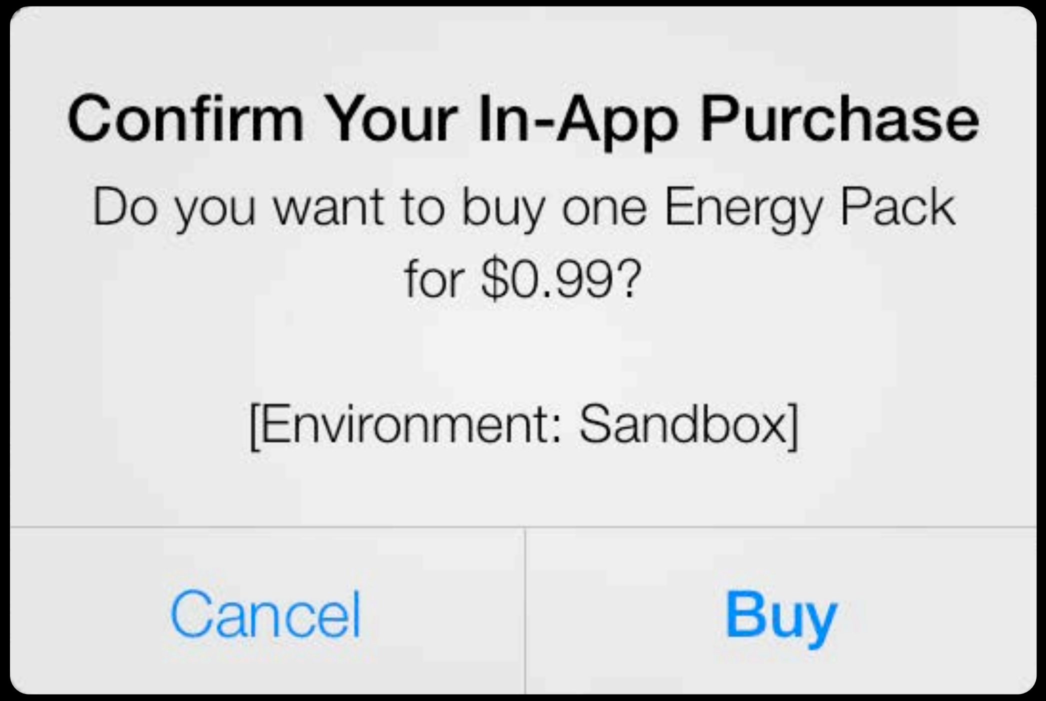


Based on Application Code Signing



How to Be Sure

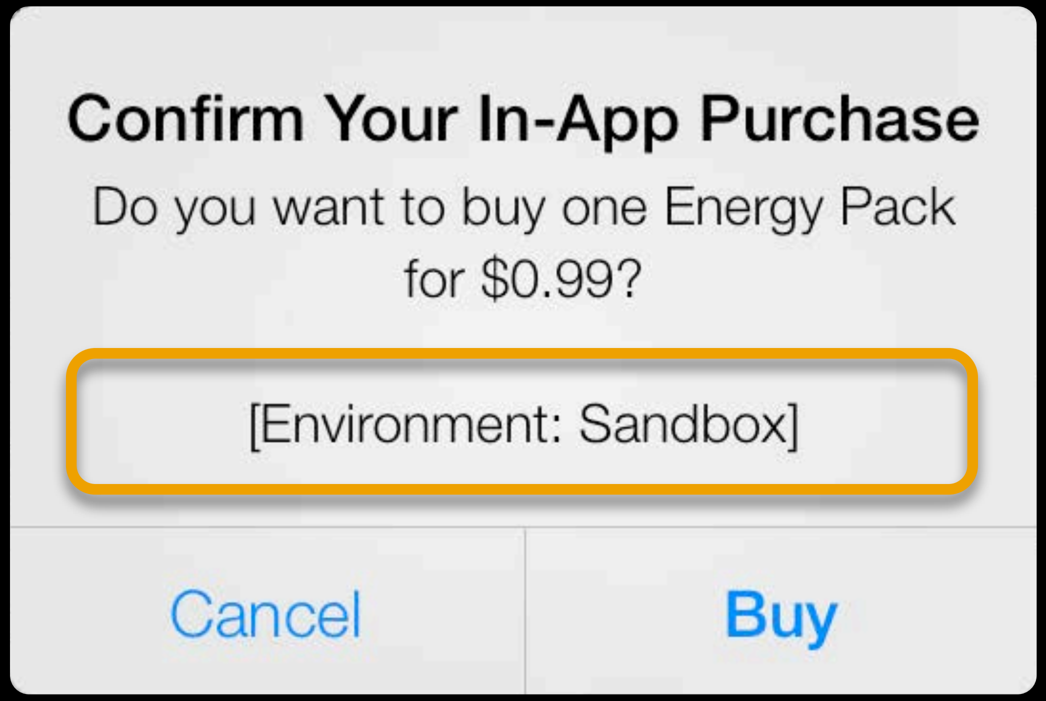
How to Be Sure


Confirm Your In-App Purchase

Do you want to buy one Energy Pack for $0.99?

[Environment: Sandbox]

How to Be Sure


Confirm Your In-App Purchase

Do you want to buy one Energy Pack for $0.99?

[Environment: Sandbox]

The Test Environment

Differences

- No charge
- Receipts won’t validate against Apple Production servers
- Can request expired and/or revoked receipts
- Expired receipts won’t prevent your iOS app from launching
- Time contraction

Subscription Timing

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

Maximum 6 renewals per 8-hour window

Setting up the Test Environment

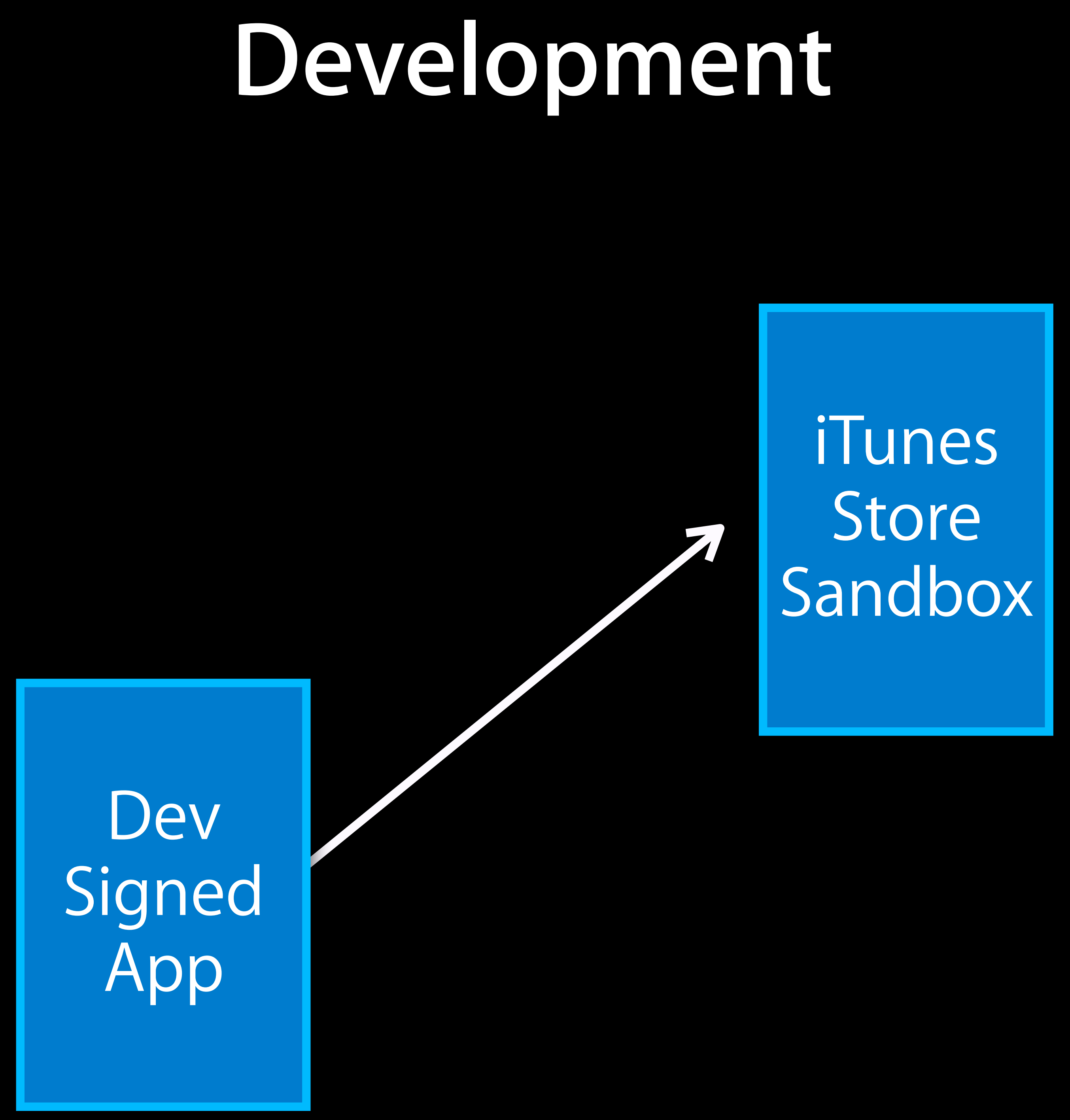
- Setup in iTunes Connect
    - Create a test user
    - Enter products for sale
- Build and sign your app
- Mac: Launch from Finder once to fetch a receipt
- Buy many products!

Verifying Receipt During App Review

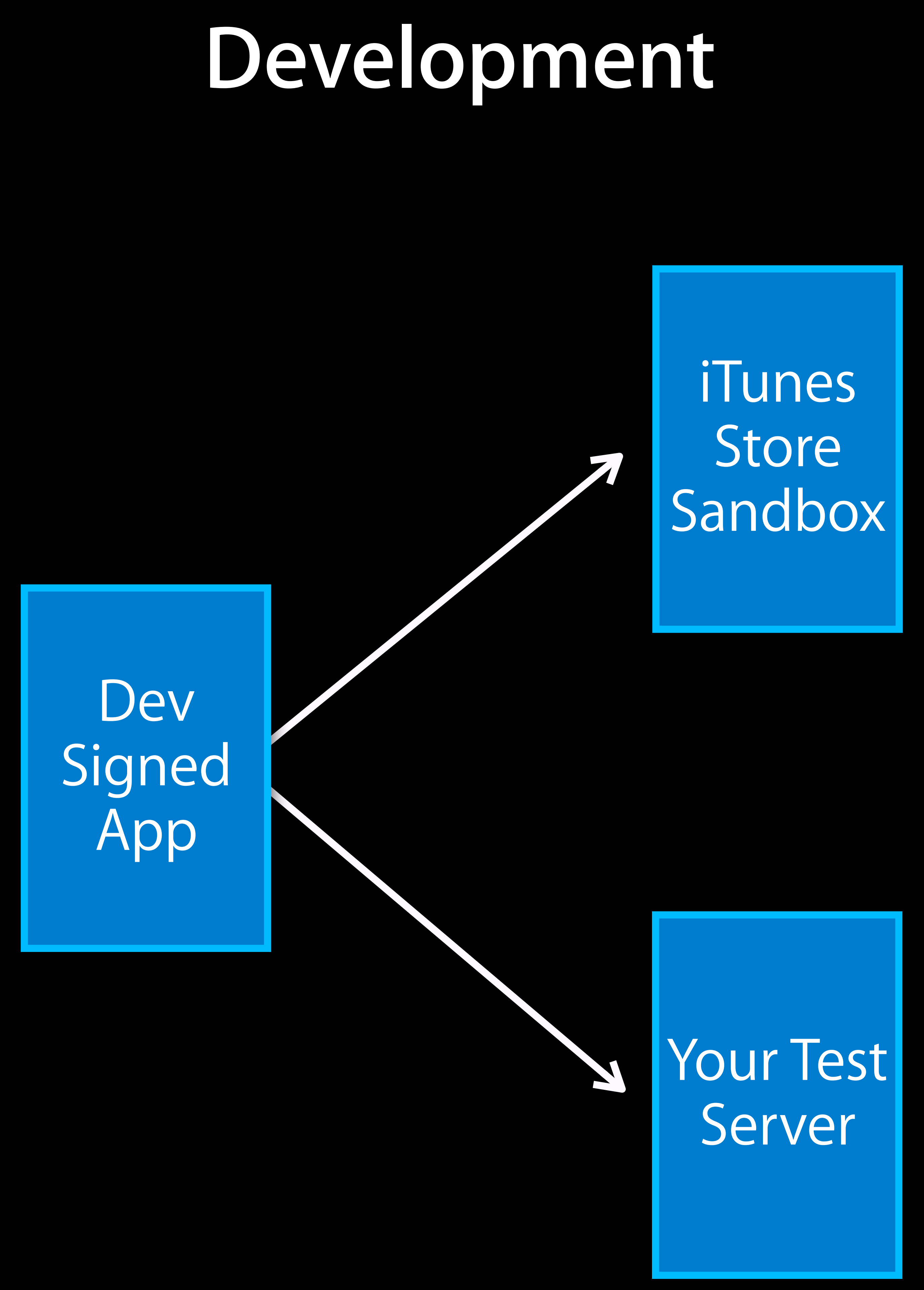
Development

Verifying Receipt During App Review

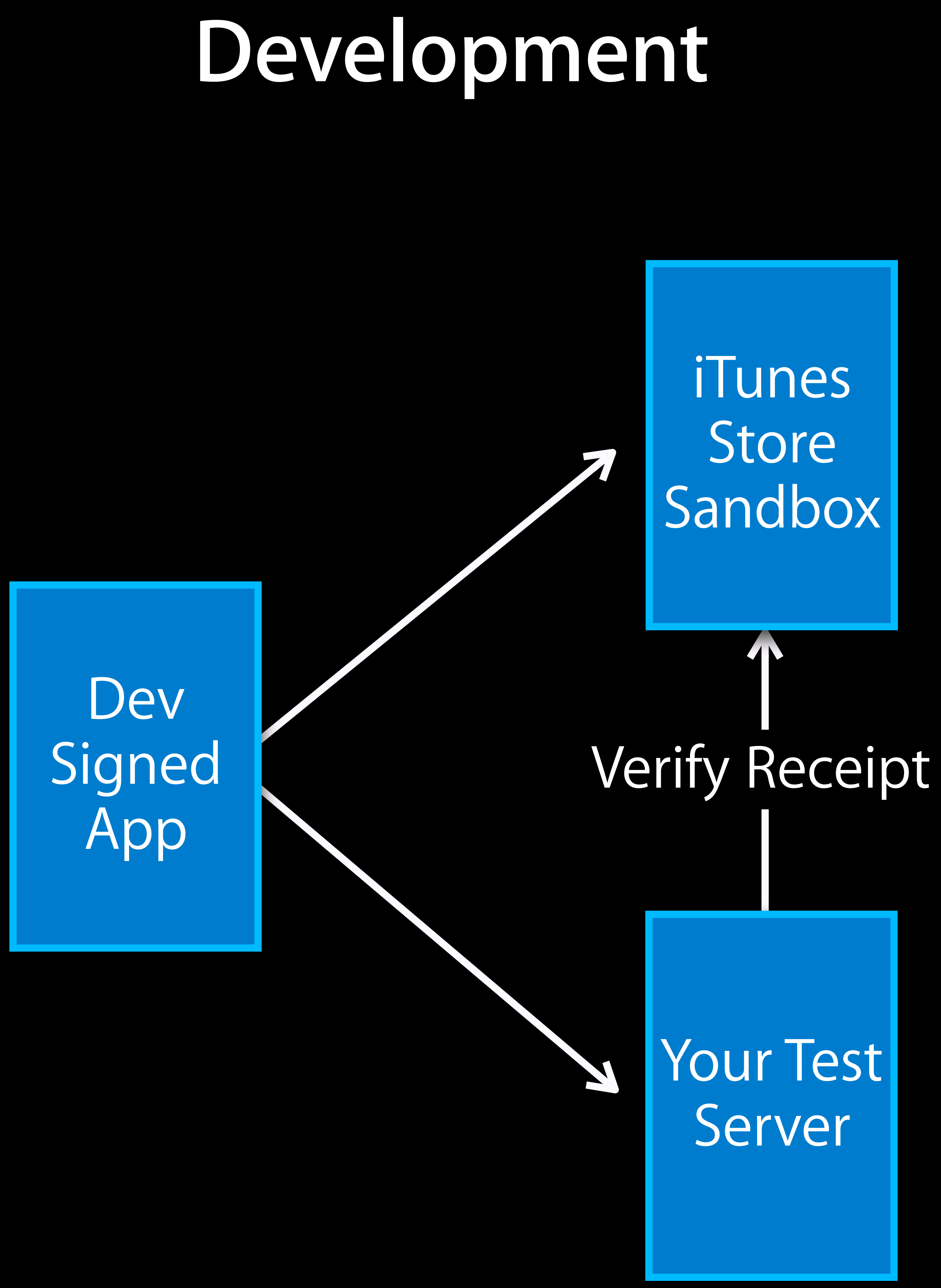
Verifying Receipt During App Review



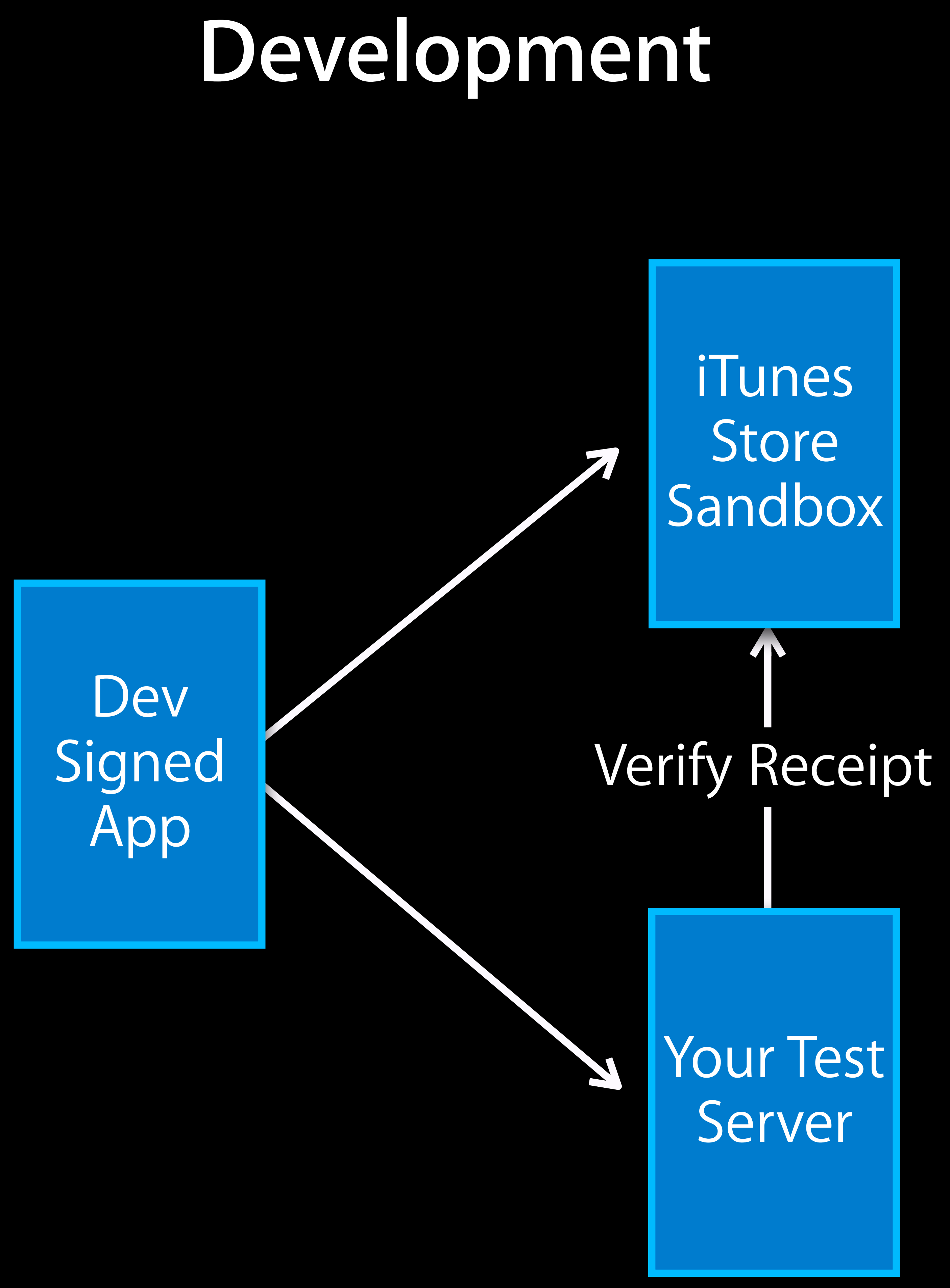
Verifying Receipt During App Review

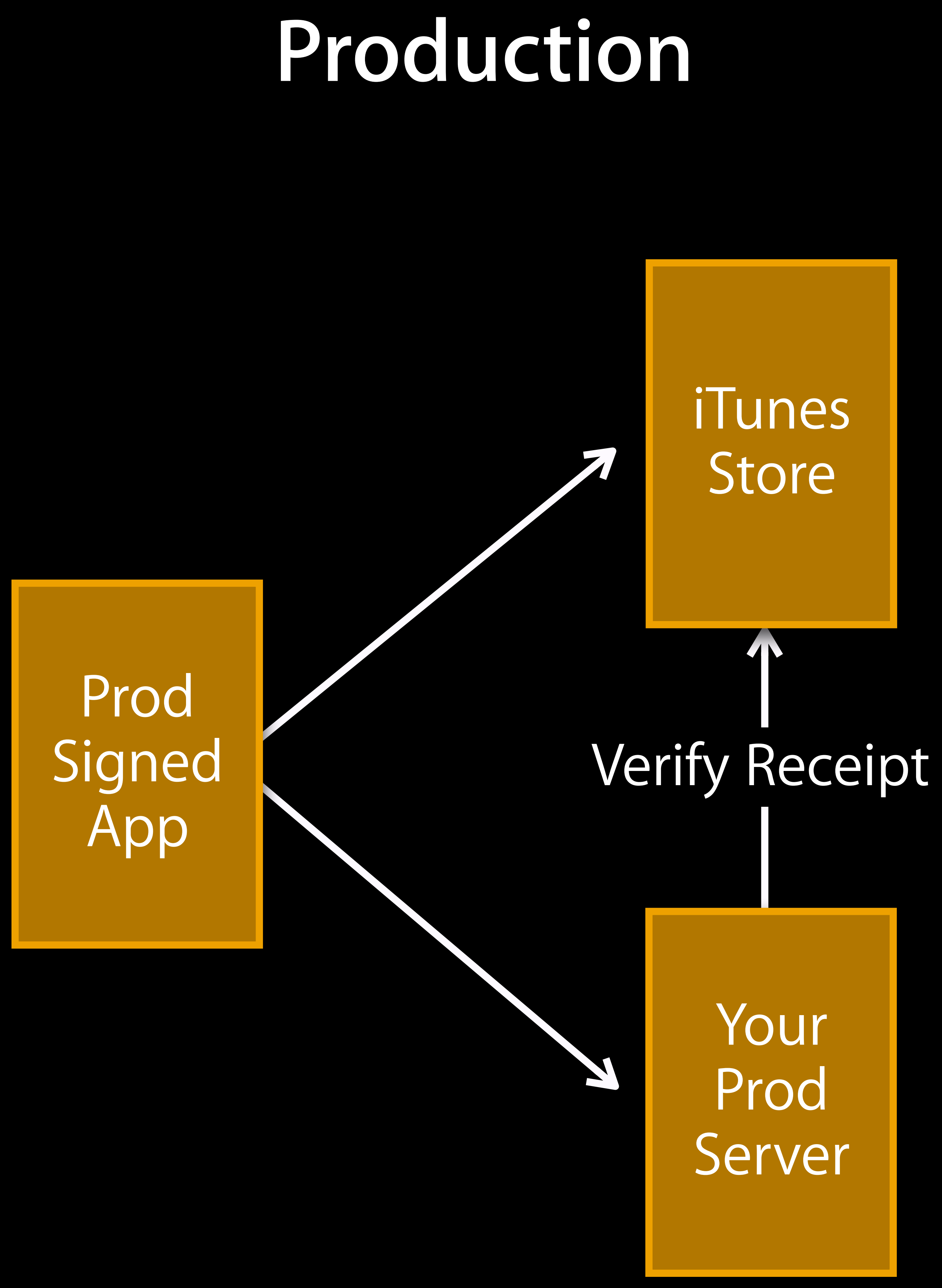


Verifying Receipt During App Review

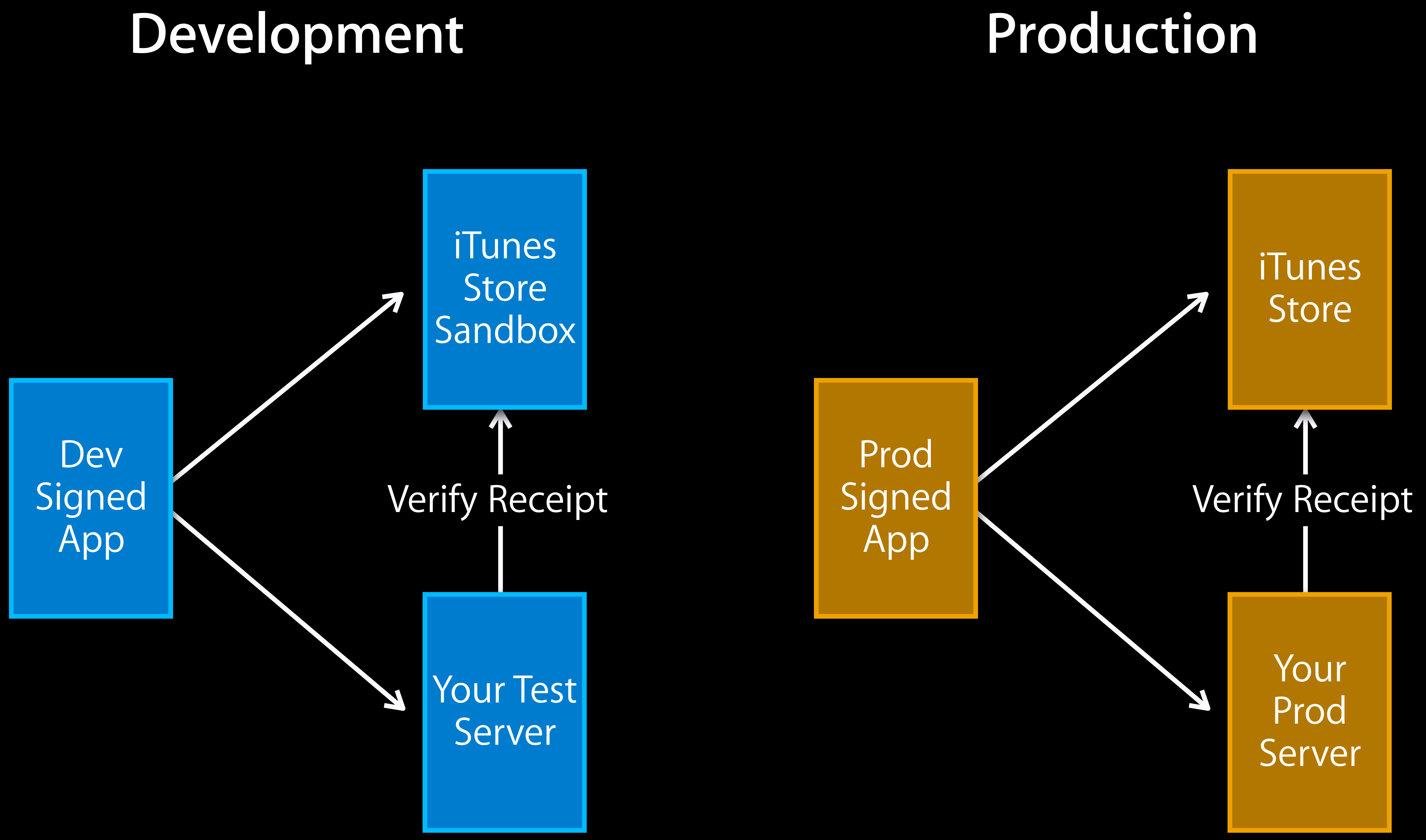


Verifying Receipt During App Review

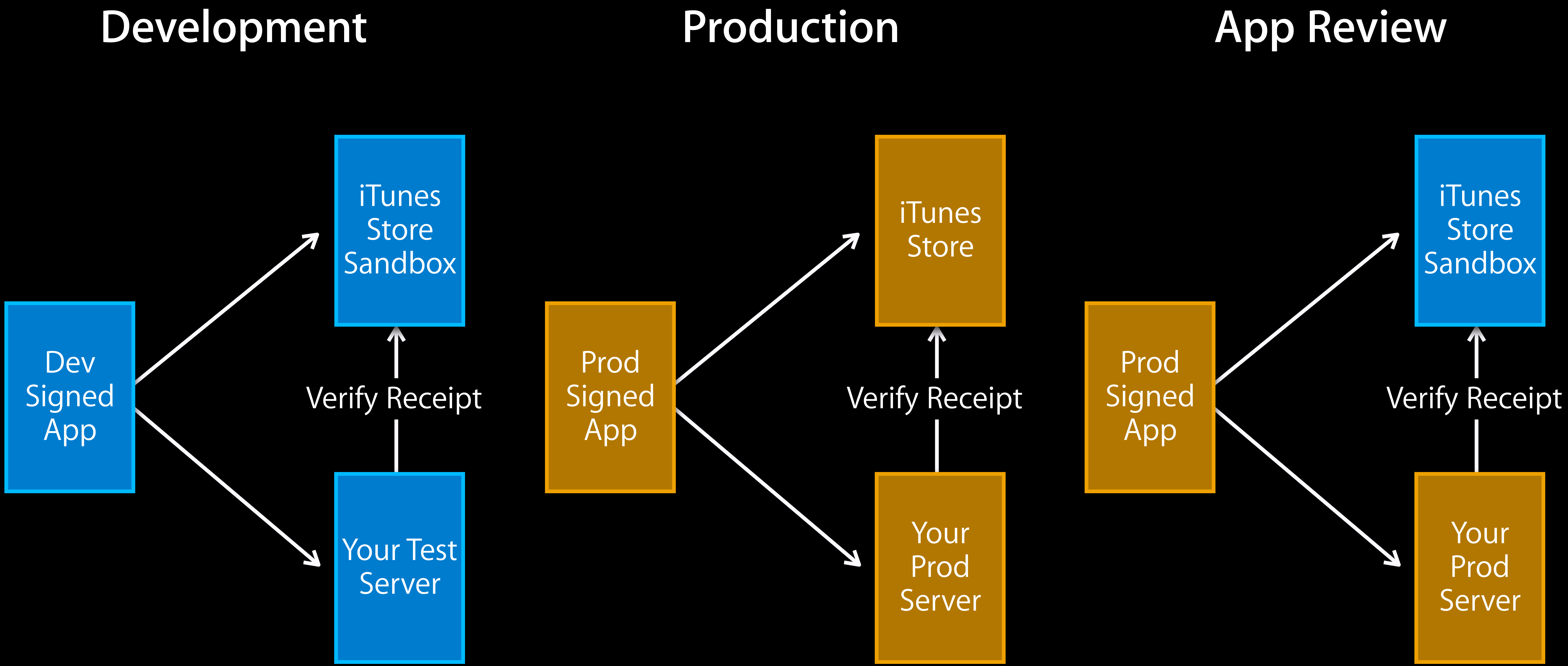




Verifying Receipt During App Review

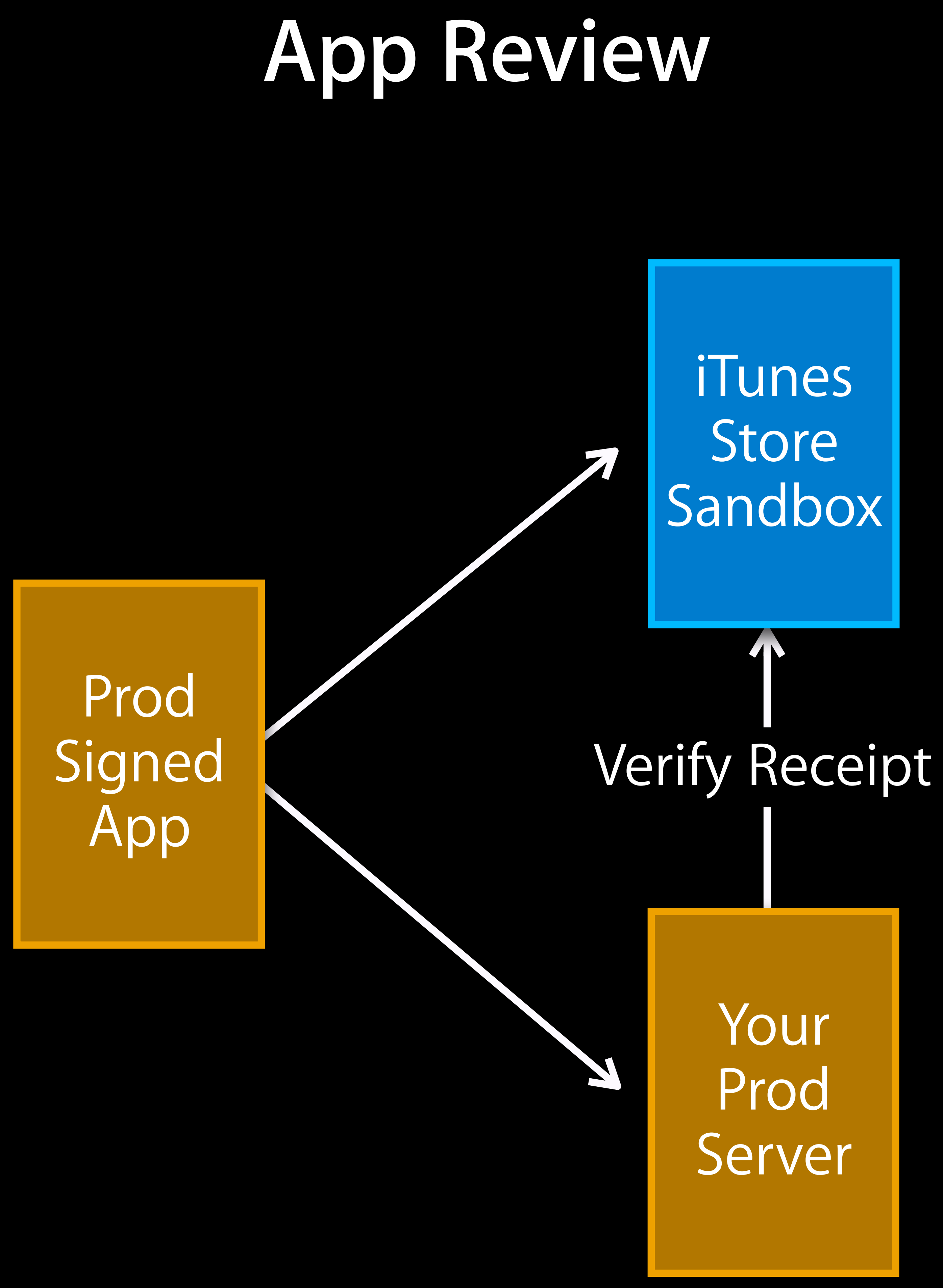


Verifying Receipt During App Review



App Review Considerations

- Try the Production environment
- If receipt is from Sandbox, you will receive error `21007`
- Then try against Sandbox



Agenda

- What’s new in Store Kit
- Implementing in-app purchases
- Using the test environment
- **Tips for passing app review**

Tips for Passing App Review

And make your users happy

Restore Button

- You must have a Restore button
- Should be used only for
  - Non-consumables
  - Auto-renewable subscriptions
- Restore and Purchase should be separate buttons

Newsstand Apps

- You must indicate a privacy policy URL
- Auto-renewable subscription must be in marketing text
    - Schedule 2, 3.8b
- Free subscriptions are only for free publications
    - Not for temporary promotions
- Even free subscriptions must use a Subscribe button
- Always offer the option to subscribe to your publication

Auto-Renewable Subscriptions

- You must indicate a privacy policy URL
- Auto-renewable subscription must be in marketing text
    - Schedule 2, 3.8b
- After subscribing, the latest issue must become downloadable
- Paid subscription must provide non-free content
- Apps that offer services should use non-renewing subscriptions

Non-Renewing Subscriptions

- Asking users to register should be optional
    - Unless you offer account-based features

Purchases

Purchases

Purchases must work!

Agenda

- What's new in Store Kit
- Implementing in-app purchases
- Using the test environment
- Tips for passing app review

More Information

Paul Marcos
Application Services Evangelist
pmarcos@apple.com

Documentation
In-App Purchase Programming Guide
Receipt Validation Programming Guide
Search http://developer.apple.com

Apple Developer Forums
http://devforums.apple.com

Labs

Store Kit and Receipts Lab	Services Lab B Thursday 3:15PM	🟩
App Store Lab	Level 3 Thursday until 6:00PM Friday 9:00AM to 12:00PM	⬜
iTunes Connect Lab	Services Lab B Thursday 11:30AM	🟩
iTunes Connect Lab	Services Lab B Friday 10:15AM	🟩

WWDC2013