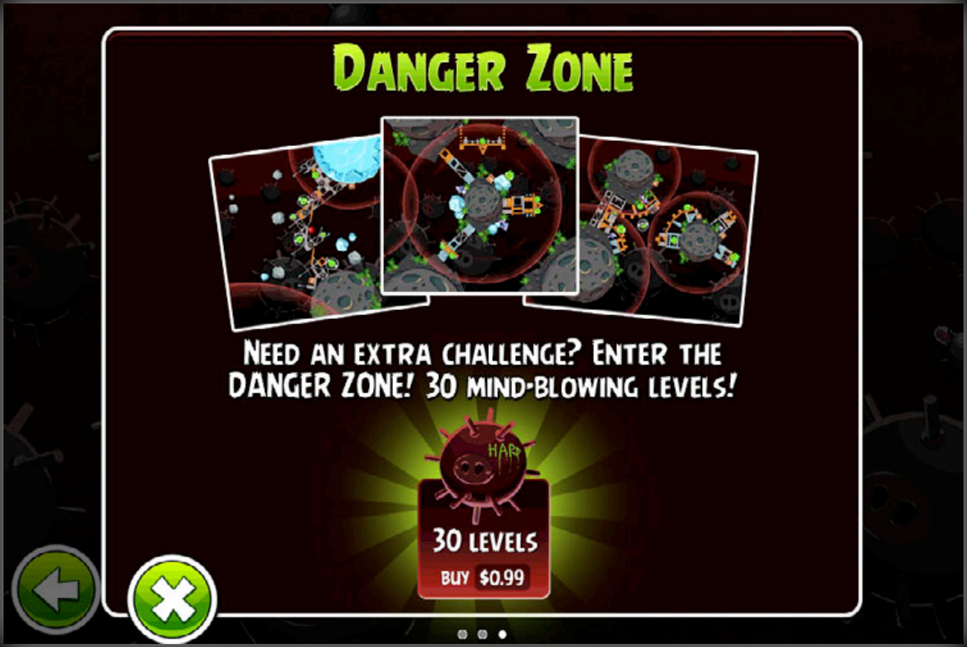
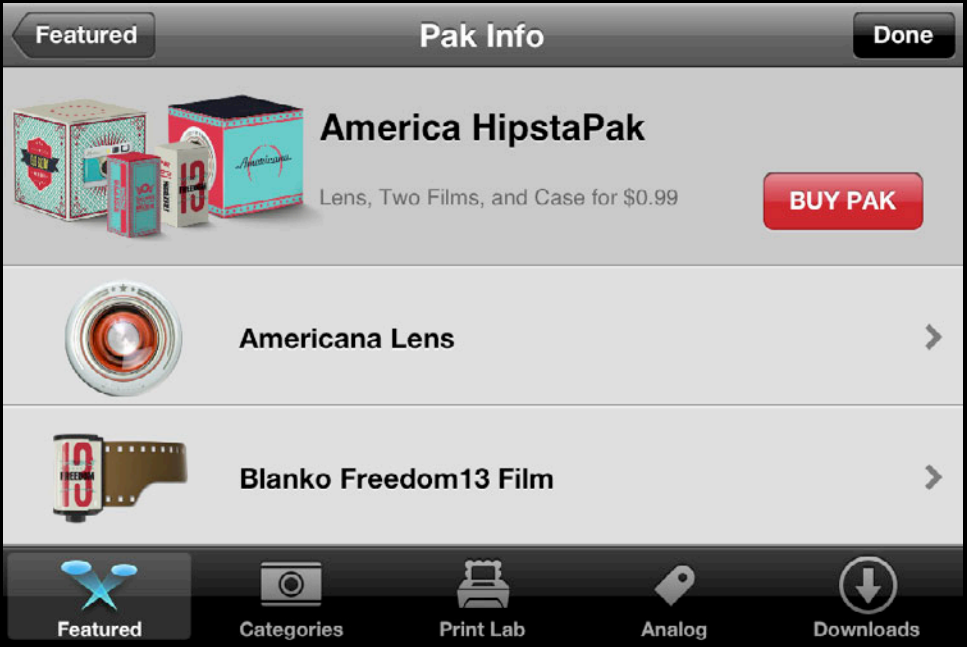
Selling Products with Store Kit

Session 302

Daniel Feldman
Engineering Manager, Mac App Store


DANGER ZONE
NEED AN EXTRA CHALLENGE? ENTER THE DANGER ZONE! 30 MIND-BLOWING LEVELS!
HARD
30 LEVELS
BUY $0.99



Featured
Pak Info
Done
America HipstaPak
Lens, Two Films, and Case for $0.99
BUY PAK
Americana Lens
Blanko Freedom13 Film
Featured
Categories
Print Lab
Analog
Downloads

Shopping
iTunes
Magic...Map
Newsstand
Store
Clean Eating
THE DAILY
Travel
Seafood
Vocabology
Uber

75%

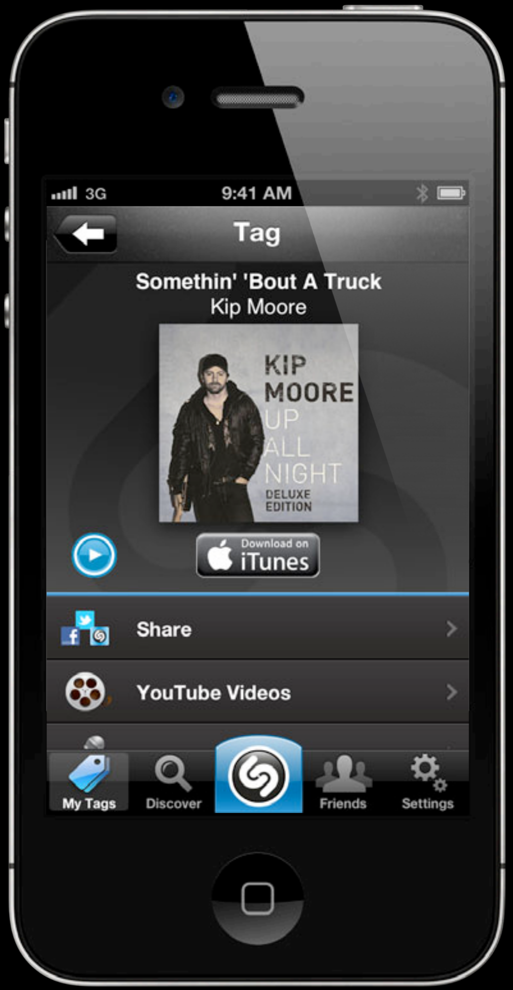
Of the 25 top grossing
iPhone apps use In-App Purchase

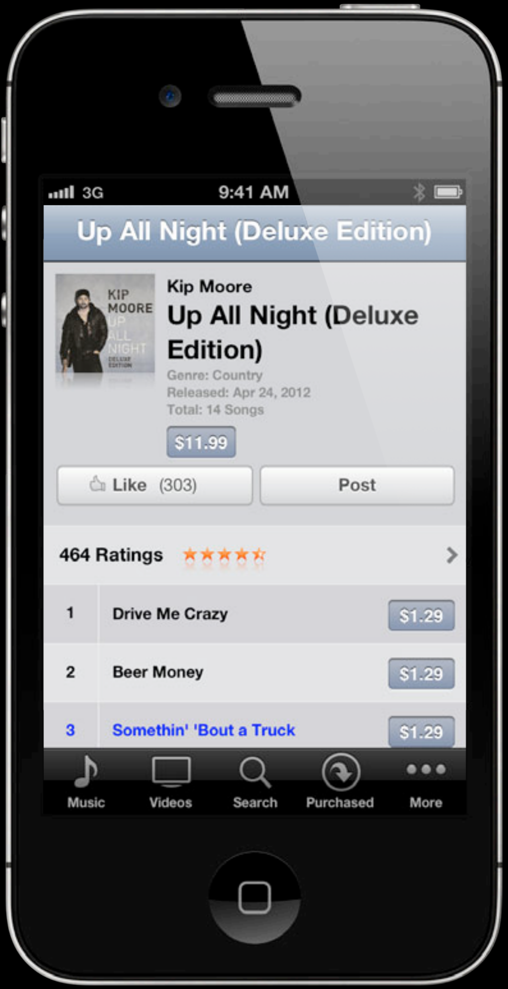
Today’s Agenda

- Selling Store Content
- Using In-App Purchase
- In Detail: The Purchase Queue
- App Store Hosted Content
- Best Practices

Today’s Agenda

- Selling Store Content
- Using In-App Purchase
- In Detail: The Purchase Queue
- App Store Hosted Content
- Best Practices


3G
9:41 AM
Tag
Somethin' 'Bout A Truck
Kip Moore
KIP MOORE UP ALL NIGHT DELUXE EDITION
Download on iTunes
Share
YouTube Videos
My Tags
Discover
Friends
Settings



Kip Moore

Up All Night (Deluxe Edition)

Genre: Country
Released: Apr 24, 2012
Total: 14 Songs

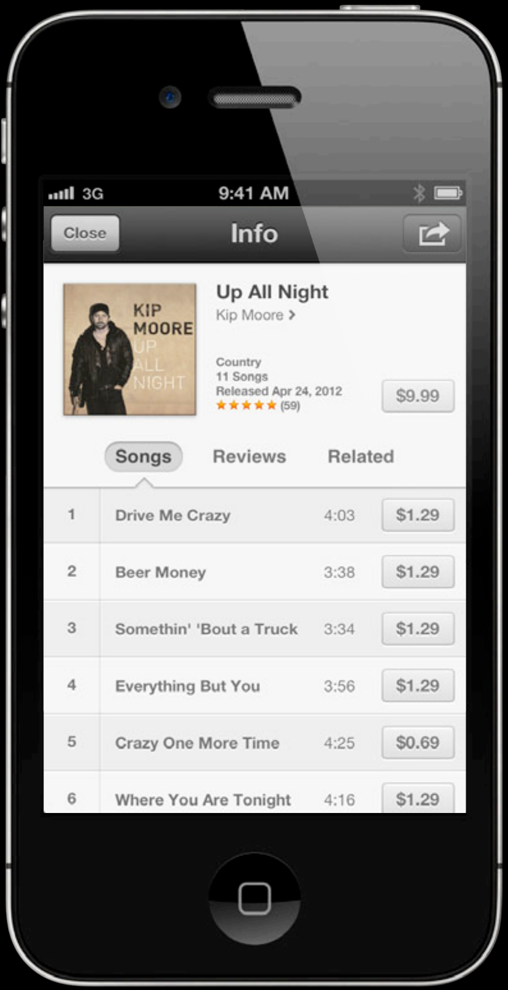
$11.99

464 Ratings ★★★★☆

1	Drive Me Crazy	$1.29
2	Beer Money	$1.29
3	Somethin' 'Bout a Truck	$1.29

3G
9:41 AM
Tag
Somethin' 'Bout A Truck
Kip Moore
KIP MOORE UP ALL NIGHT DELUXE EDITION
Download on iTunes
Share
YouTube Videos
My Tags
Discover
Friends
Settings

iOS



3G 9:41 AM

Close Info

Up All Night

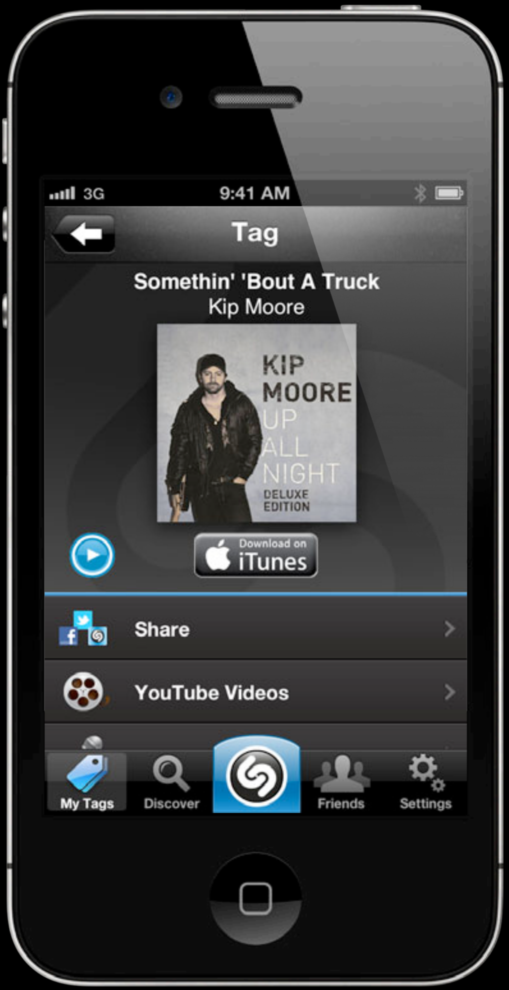
Kip Moore ›

Country
11 Songs
Released Apr 24, 2012
★★★★★ (59)

$9.99

Songs Reviews Related

1	Drive Me Crazy	4:03	$1.29
2	Beer Money	3:38	$1.29
3	Somethin' 'Bout a Truck	3:34	$1.29
4	Everything But You	3:56	$1.29
5	Crazy One More Time	4:25	$0.69
6	Where You Are Tonight	4:16	$1.29


3G
9:41 AM
Tag
Somethin' 'Bout A Truck
Kip Moore
KIP MOORE UP ALL NIGHT DELUXE EDITION
Download on iTunes
Share
YouTube Videos
My Tags
Discover
Friends
Settings

Selling Store Products

Selling Store Products

- Look up item identifier

Selling Store Products

- Look up item identifier
    - Search API

Selling Store Products

- Look up item identifier
    - Search API
    - Enterprise Partner Feed

Selling Store Products

- Look up item identifier
    - Search API
    - Enterprise Partner Feed
    - Parse iTunes Preview URLs

Selling Store Products

- Look up item identifier
  - Search API
  - Enterprise Partner Feed
  - Parse iTunes Preview URLs
- Configure a `SKStoreProductViewController`

Selling Store Products

- Look up item identifier
    - Search API
    - Enterprise Partner Feed
    - Parse iTunes Preview URLs
- Configure a `SKStoreProductViewController`
- Tell the view controller to load

Selling Store Products

- Look up item identifier
  - Search API
  - Enterprise Partner Feed
  - Parse iTunes Preview URLs
- Configure a `SKStoreProductViewController`
- Tell the view controller to load
- Show the view controller

How to Sell a Store Product

How to Sell a Store Product

```
- (void)showProductViewController:(UIButton *)sender {









}
```

How to Sell a Store Product

```
- (void)showProductViewController:(UIButton *)sender {

 SKStoreProductViewController *viewController =
      [[SKStoreProductViewController alloc] init];








}
```

How to Sell a Store Product

```
- (void)showProductViewController:(UIButton *)sender {

 SKStoreProductViewController *viewController =
      [[SKStoreProductViewController alloc] init];

 viewController.delegate = self;







}
```

How to Sell a Store Product

```
- (void)showProductViewController:(UIButton *)sender {

 SKStoreProductViewController *viewController =
      [[SKStoreProductViewController alloc] init];

 viewController.delegate = self;

 NSDictionary *parameters =
 @{SKStoreProductParameterITunesItemIdentifier: [NSNumber
 numberWithInteger: itemIdentifier]};



}
```

How to Sell a Store Product

```
- (void)showProductViewController:(UIButton *)sender {

 SKStoreProductViewController *viewController =
      [[SKStoreProductViewController alloc] init];

 viewController.delegate = self;

 NSDictionary *parameters =
 @{SKStoreProductParameterITunesItemIdentifier: [NSNumber
 numberWithInteger: itemIdentifier]};



}
```

How to Sell a Store Product

```
NSDictionary *parameters =
@{SKStoreProductParameterITunesItemIdentifier: [NSNumber
numberWithInteger: itemIdentifier]};









}
```

How to Sell a Store Product

```
NSDictionary *parameters =
@{SKStoreProductParameterITunesItemIdentifier: [NSNumber
numberWithInteger: itemIdentifier]};









}
```

How to Sell a Store Product

```
 NSDictionary *parameters =
 @{SKStoreProductParameterITunesItemIdentifier: [NSNumber
 numberWithInteger: itemIdentifier]};

   [viewController loadProductWithParameters:parameters
completionBlock: ^(BOOL result, NSError *error) {





   }]

 }
```

How to Sell a Store Product

```
 NSDictionary *parameters =
 @{SKStoreProductParameterITunesItemIdentifier: [NSNumber
 numberWithInteger: itemIdentifier]};

  [viewController loadProductWithParameters:parameters
completionBlock: ^(BOOL result, NSError *error) {
         if (result)
             [[self.window rootViewController]
             presentModalViewController:viewController
             animated:YES];
  }]

 }
```

How to Sell a Store Product

```
- (void)productViewControllerDidFinish:
    (SKStoreProductViewController *)viewController
```

Today’s Agenda

- Selling Store Content
- Using In-App Purchase
- In Detail: The Purchase Queue
- App Store Hosted Content
- Best Practices

In-App Purchase Types

In-App Purchase Types

- Consumable

In-App Purchase Types

- Consumable
- Non-consumable

In-App Purchase Types

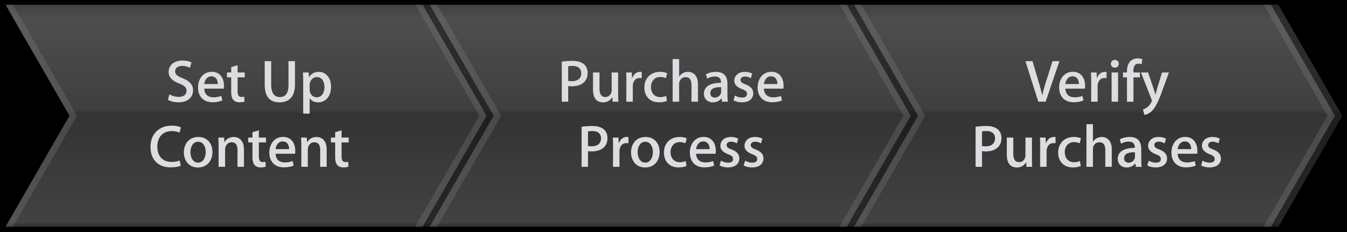
- Consumable
- Non-consumable
- Auto-renewing subscription

In-App Purchase Types

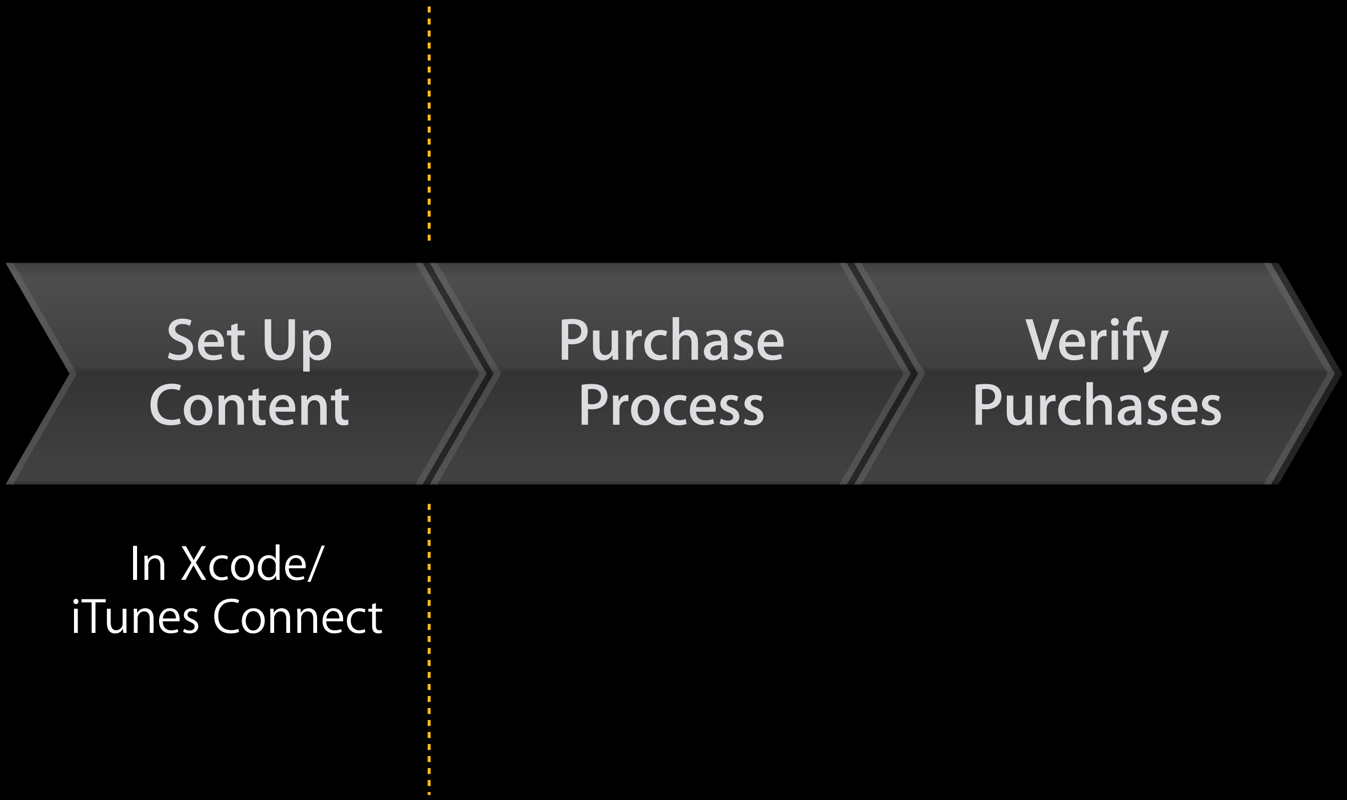
- Consumable
- Non-consumable
- Auto-renewing subscription
- Non-renewing subscription

In-App Process Overview

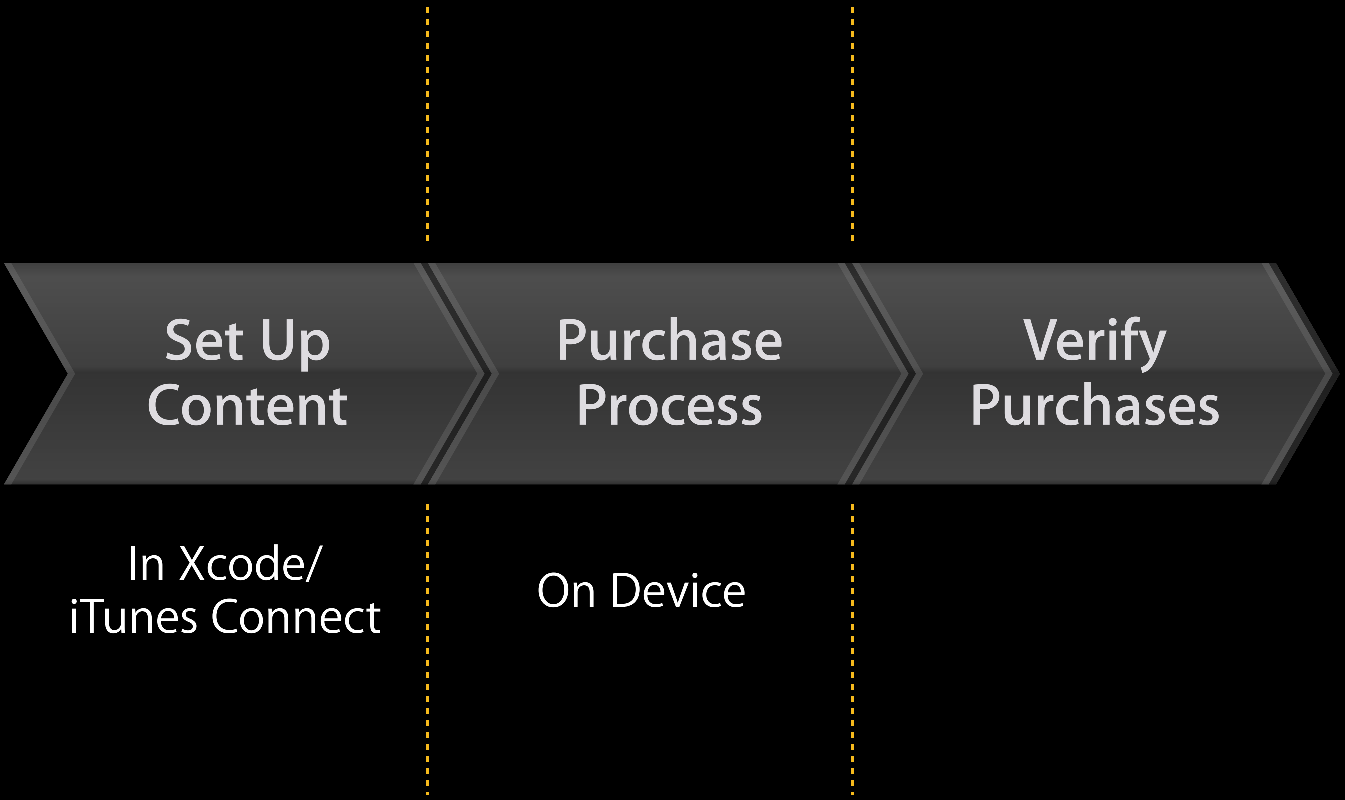
In-App Process Overview



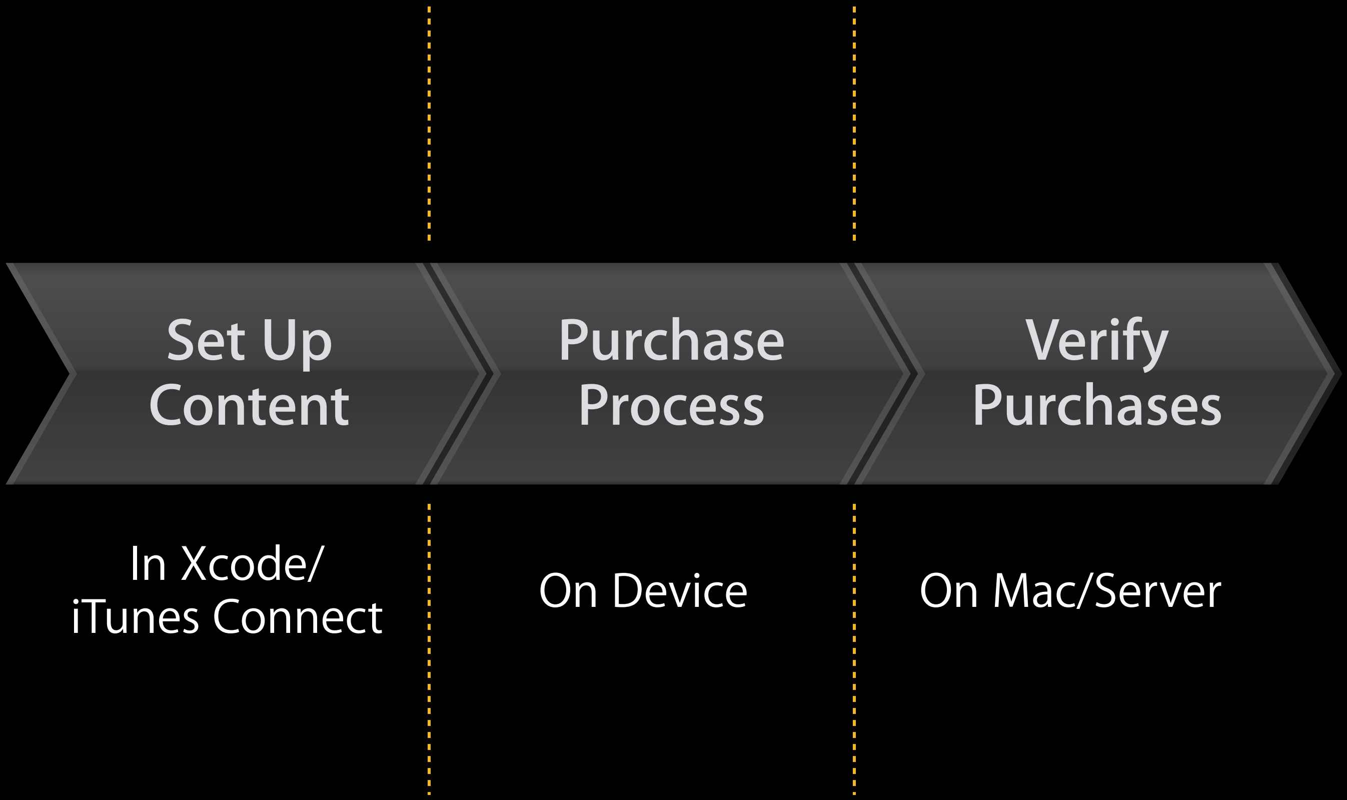
In-App Process Overview



In-App Process Overview

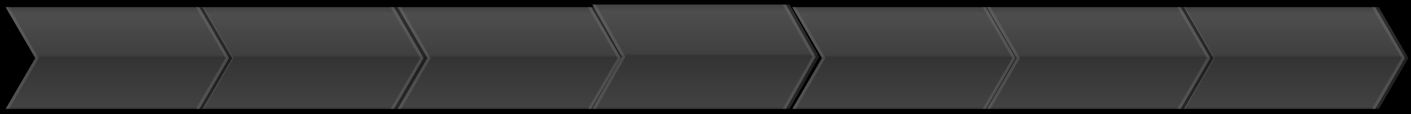


In-App Process Overview

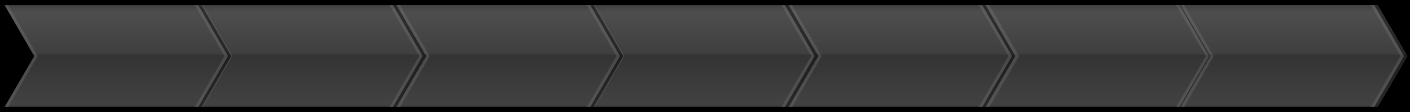


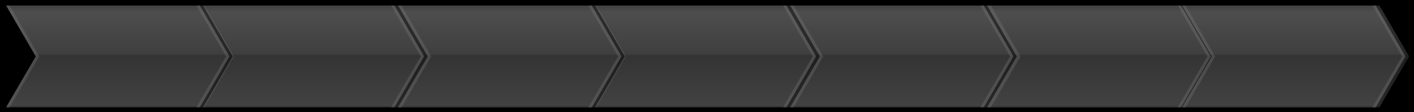
In-App Process Overview

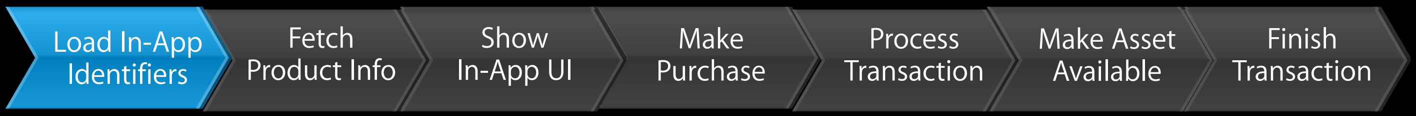
In-App Process Overview



In-App Process Overview



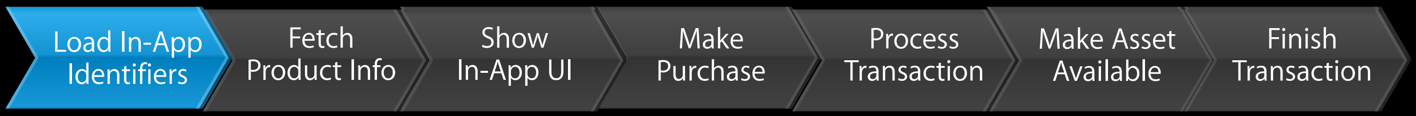



Load In-App Identifiers
Fetch Product Info
Show In-App UI
Make Purchase
Process Transaction
Make Asset Available
Finish Transaction

Load In-App Identifiers → Fetch Product Info → Show In-App UI → Make Purchase → Process Transaction → Make Asset Available → Finish Transaction

- From within your app

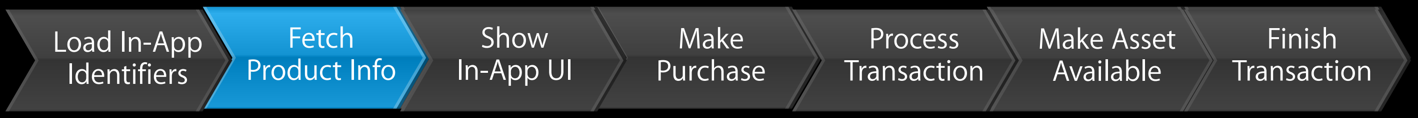
```
NSArray* productIdentifiers = @[@"com.myCompany.myApp.product1",
                                @"com.myCompany.myApp.product2",
                                @"com.myCompany.myApp.product3"];
```



- From within your app

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

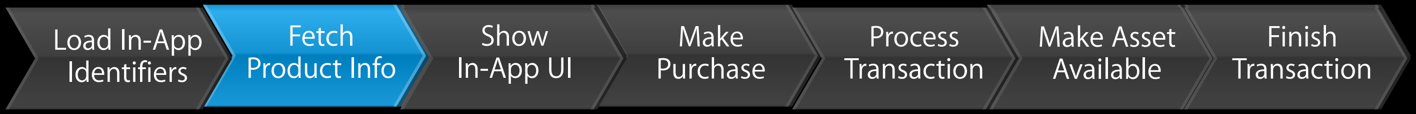
- From your server
    - Develop your own client/server communication



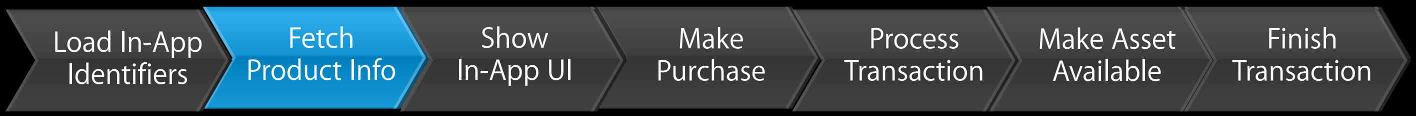
```
NSArray* productIdentifiers = @[@"com.myCompany.myApp.product1",
                                @"com.myCompany.myApp.product2",
                                @"com.myCompany.myApp.product3"];
```

Load In-App Identifiers | **Fetch Product Info** | Show In-App UI | Make Purchase | Process Transaction | Make Asset Available | Finish Transaction

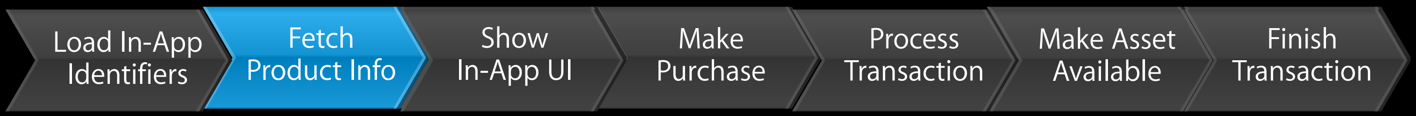
```
NSArray* productIdentifiers = @[@"com.myCompany.myApp.product1",
                                @"com.myCompany.myApp.product2",
                                @"com.myCompany.myApp.product3"];

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



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

NSSet* identifierSet = [NSSet setWithArray:productIdentifiers];

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

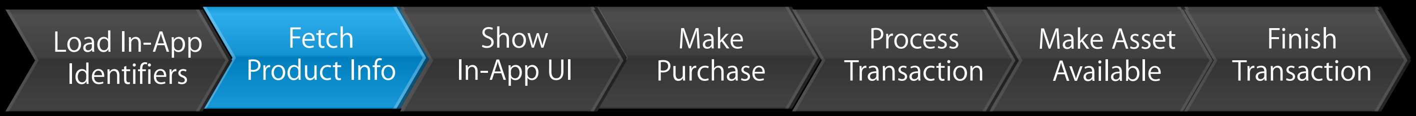


```
NSArray* productIdentifiers = @[@"com.myCompany.myApp.product1",
                                @"com.myCompany.myApp.product2",
                                @"com.myCompany.myApp.product3"];

NSSet* identifierSet = [NSSet setWithArray:productIdentifiers];

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

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


Load In-App Identifiers
Fetch Product Info
Show In-App UI
Make Purchase
Process Transaction
Make Asset Available
Finish Transaction

Load In-App Identifiers
Fetch Product Info
Show In-App UI
Make Purchase
Process Transaction
Make Asset Available
Finish Transaction

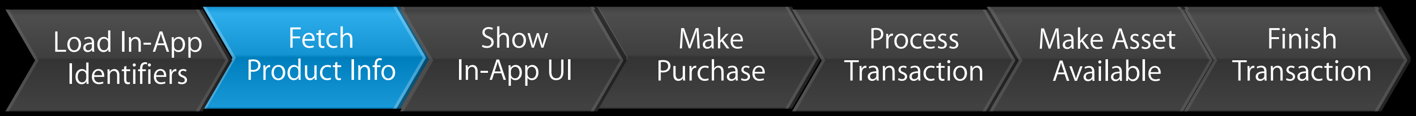
```
- (void)productsRequest:(SKProductsRequest *)request didReceiveResponse:
(SKProductsResponse *)response
```



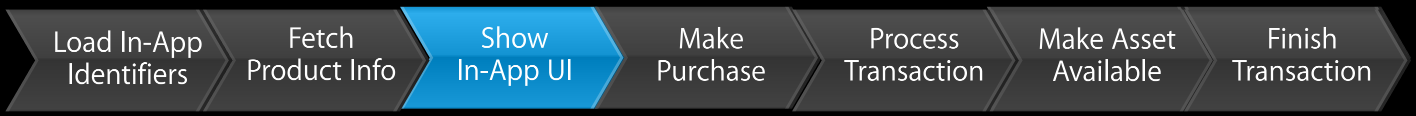
```
- (void)productsRequest:(SKProductsRequest *)request didReceiveResponse:
(SKProductsResponse *)response

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

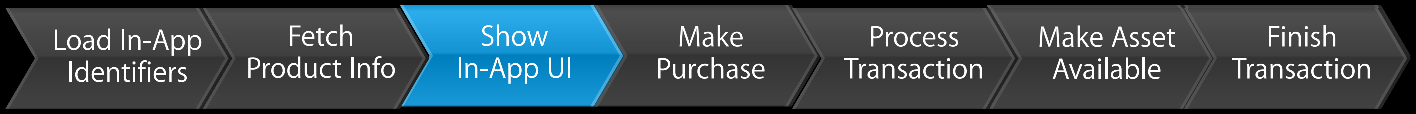
Load In-App Identifiers | **Fetch Product Info** | Show In-App UI | Make Purchase | Process Transaction | Make Asset Available | Finish Transaction

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

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



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

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

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

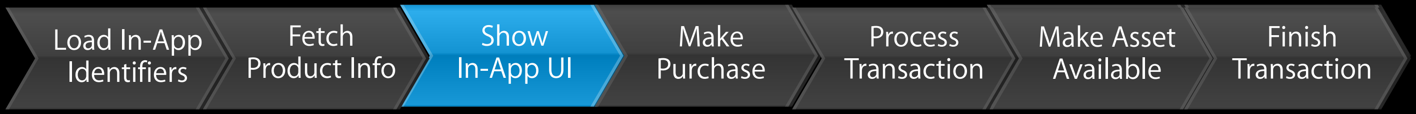

Load In-App Identifiers
Fetch Product Info
Show In-App UI
Make Purchase
Process Transaction
Make Asset Available
Finish Transaction



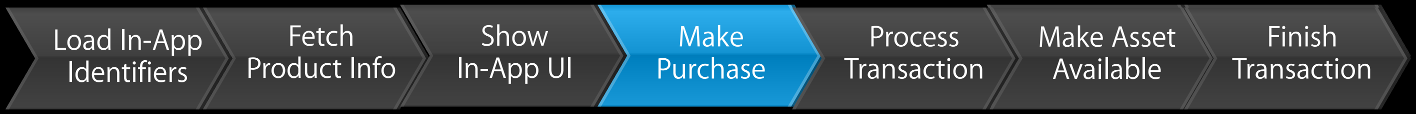
- Your responsibility

Load In-App Identifiers › Fetch Product Info › **Show In-App UI** › Make Purchase › Process Transaction › Make Asset Available › Finish Transaction

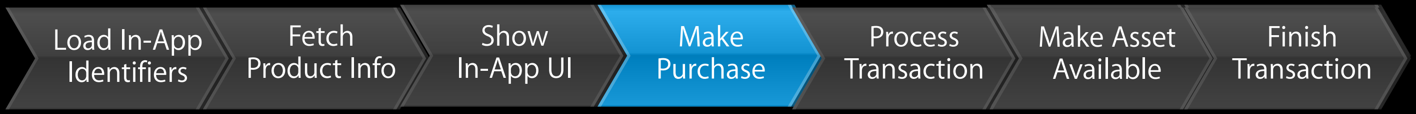
- Your responsibility
- Make it fit your app



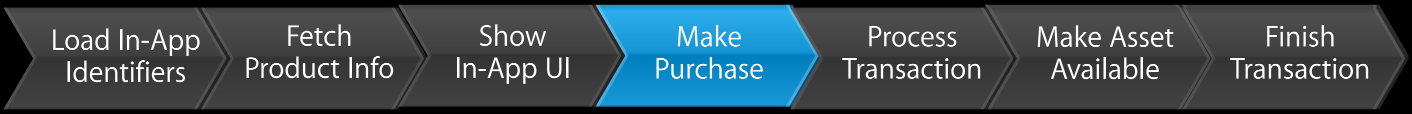
- Your responsibility
- Make it fit your app
- Don’t just show, sell!

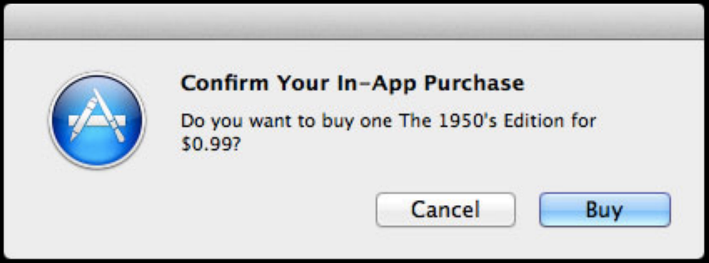


```
SKPayment *payment = [SKPayment paymentWithProduct:product];
```



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

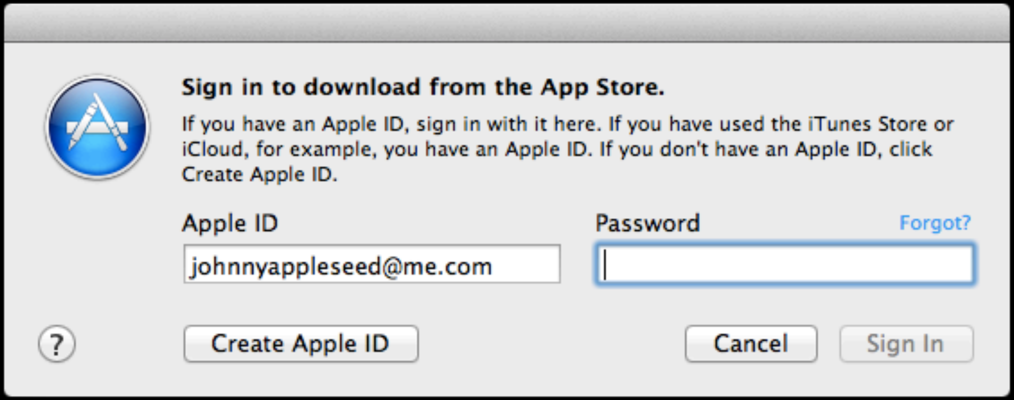




Confirm Your In-App Purchase

Do you want to buy one The 1950's Edition for $0.99?

Load In-App Identifiers → Fetch Product Info → Show In-App UI → **Make Purchase** → Process Transaction → Make Asset Available → Finish Transaction

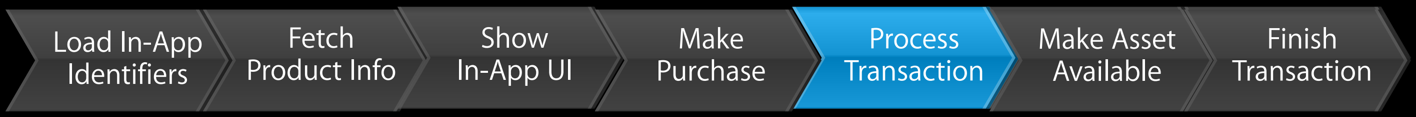


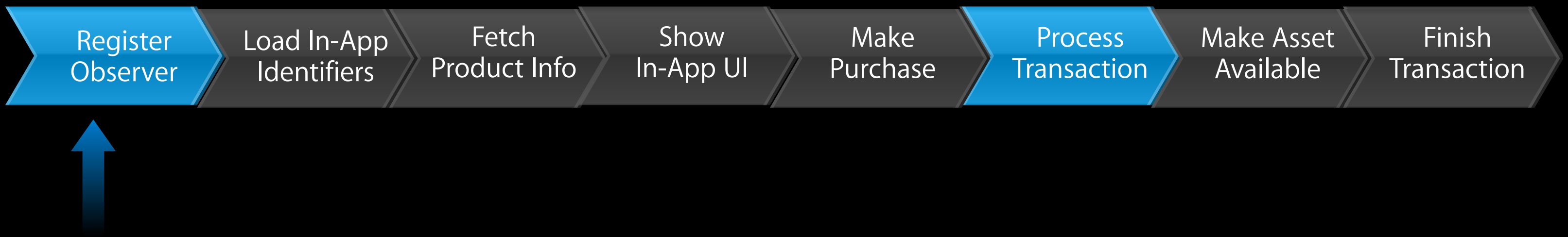
Sign in to download from the App Store.

If you have an Apple ID, sign in with it here. If you have used the iTunes Store or iCloud, for example, you have an Apple ID. If you don't have an Apple ID, click Create Apple ID.

Apple ID: johnnyappleseed@me.com

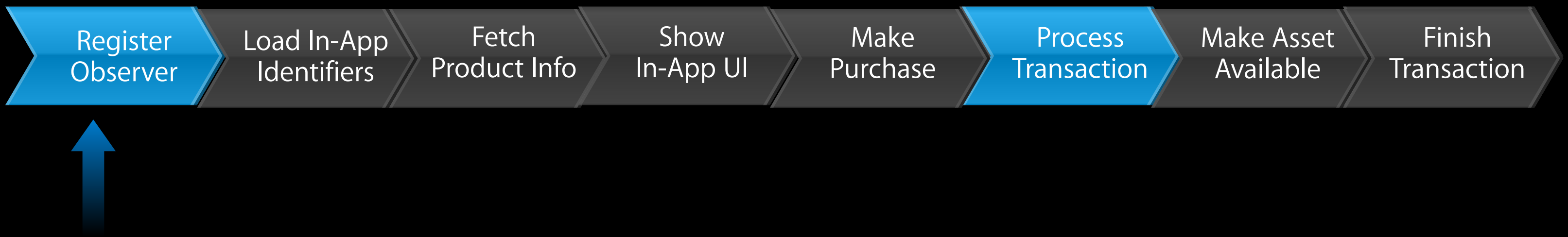
Password:


Load In-App Identifiers
Fetch Product Info
Show In-App UI
Make Purchase
Process Transaction
Make Asset Available
Finish Transaction



- Add an observer at launch

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

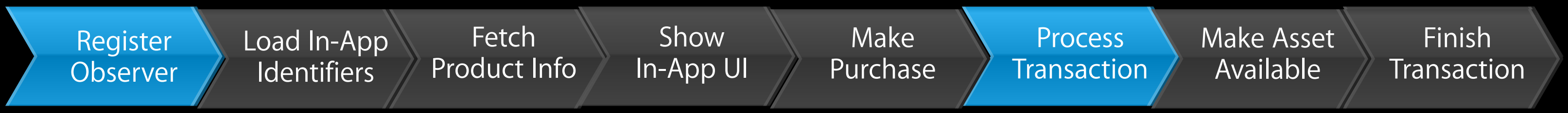


- Add an observer at launch

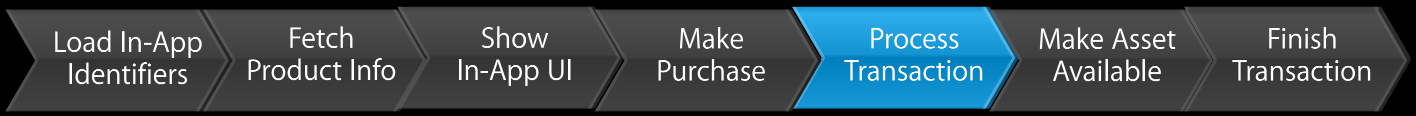
```
[[SKPaymentQueue defaultQueue] addTransactionObserver: self];
```

- Implement `SKPaymentTransactionObserver` protocol

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



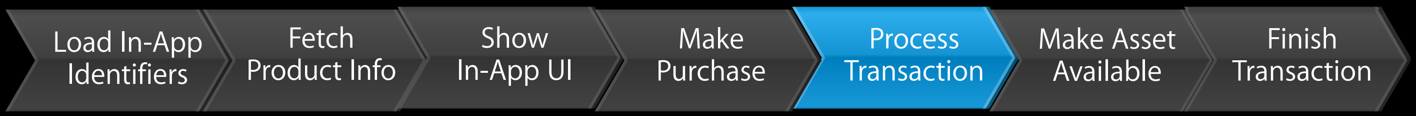
```
- (void)paymentQueue:(SKPaymentQueue *)queue updatedTransactions:(NSArray
*)transactions
```



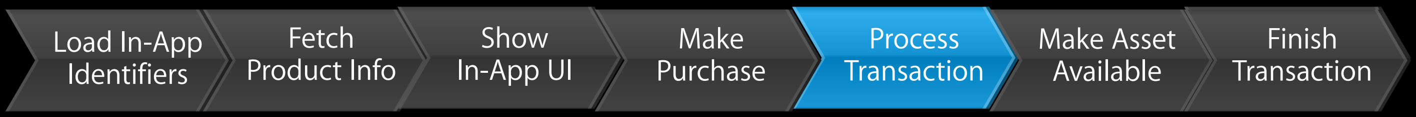
```
- (void)paymentQueue:(SKPaymentQueue *)queue updatedTransactions:(NSArray
*)transactions
```

Load In-App Identifiers | Fetch Product Info | Show In-App UI | Make Purchase | **Process Transaction** | Make Asset Available | Finish Transaction

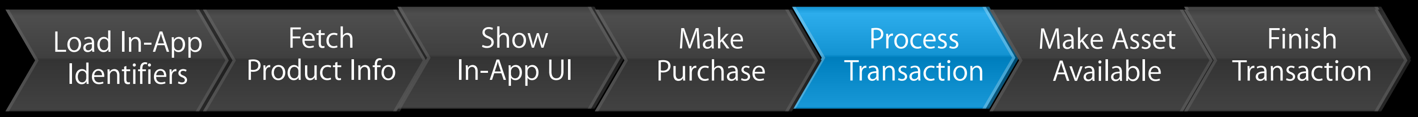
```
- (void)paymentQueue:(SKPaymentQueue *)queue updatedTransactions:(NSArray
*)transactions

    for(SKPaymentTransaction* transaction in transactions)
    {









    }
```



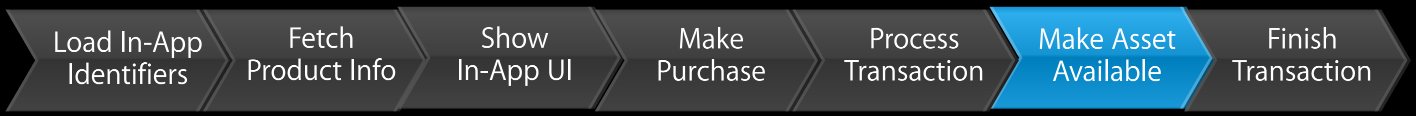
```
- (void)paymentQueue:(SKPaymentQueue *)queue updatedTransactions:(NSArray
*)transactions

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






       }
   }
```



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

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

          case SKPaymentTransactionStatePurchased:

                  …



       }
   }
```

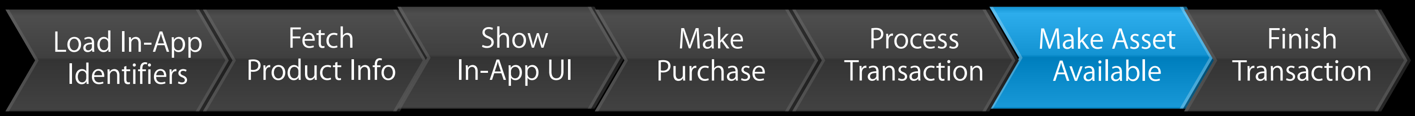


```
- (void)paymentQueue:(SKPaymentQueue *)queue updatedTransactions:(NSArray
*)transactions

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

           case SKPaymentTransactionStatePurchased:

                   …

           case SKPaymentTransactionStateFailed:

                   …

        }
    }
```

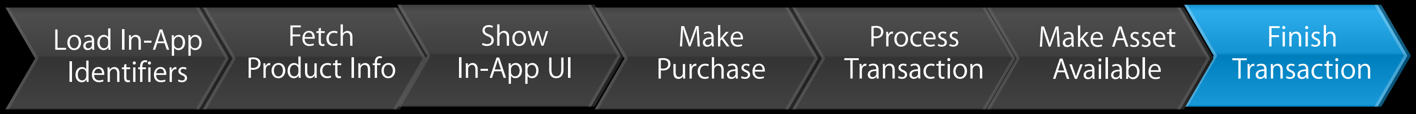

Load In-App Identifiers
Fetch Product Info
Show In-App UI
Make Purchase
Process Transaction
Make Asset Available
Finish Transaction

Load In-App Identifiers → Fetch Product Info → Show In-App UI → Make Purchase → Process Transaction → **Make Asset Available** → Finish Transaction

- Unlock functionality in your app



- Unlock functionality in your app
- Download additional content from your server



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

Restoring Transactions

Restoring Transactions

- A way to get all In-App purchases back

Restoring Transactions

- A way to get all In-App purchases back
- Important for app re-downloads, multi-device scenarios

Restoring Transactions

- A way to get all In-App purchases back
- Important for app re-downloads, multi-device scenarios
- Applications must offer this

Restoring Transactions

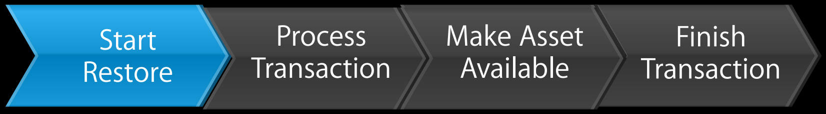
- A way to get all In-App purchases back
- Important for app re-downloads, multi-device scenarios
- Applications must offer this
- Only non-consumable and auto-renew subscription types

Restoring Transactions

- A way to get all In-App purchases back
- Important for app re-downloads, multi-device scenarios
- Applications must offer this
- Only non-consumable and auto-renew subscription types
- Don’t auto-restore on launch

Process Transaction
Make Asset Available
Finish Transaction

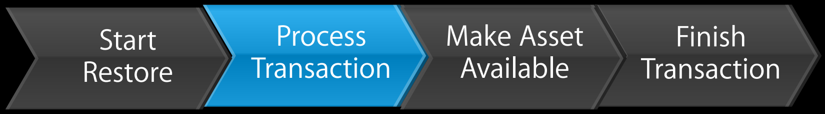
Start Restore
Process Transaction
Make Asset Available
Finish Transaction



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

Start Restore → **Process Transaction** → Make Asset Available → Finish Transaction

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



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

    for(SKPaymentTransaction* transaction in transactions)
    {









    }
```

Start Restore → **Process Transaction** → Make Asset Available → Finish Transaction

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

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






        }
    }
```

Start Restore → **Process Transaction** → Make Asset Available → Finish Transaction

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

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

          case SKPaymentTransactionStateRestored:
                 …


       }
   }
```

Start Restore
Process Transaction
Make Asset Available
Finish Transaction

Start Restore → Process Transaction → **Make Asset Available** → Finish Transaction

- Unlock functionality in your app

Start Restore | Process Transaction | **Make Asset Available** | Finish Transaction

- Unlock functionality in your app
- Download additional content from *your* server

Start Restore
Process Transaction
Make Asset Available
Finish Transaction

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

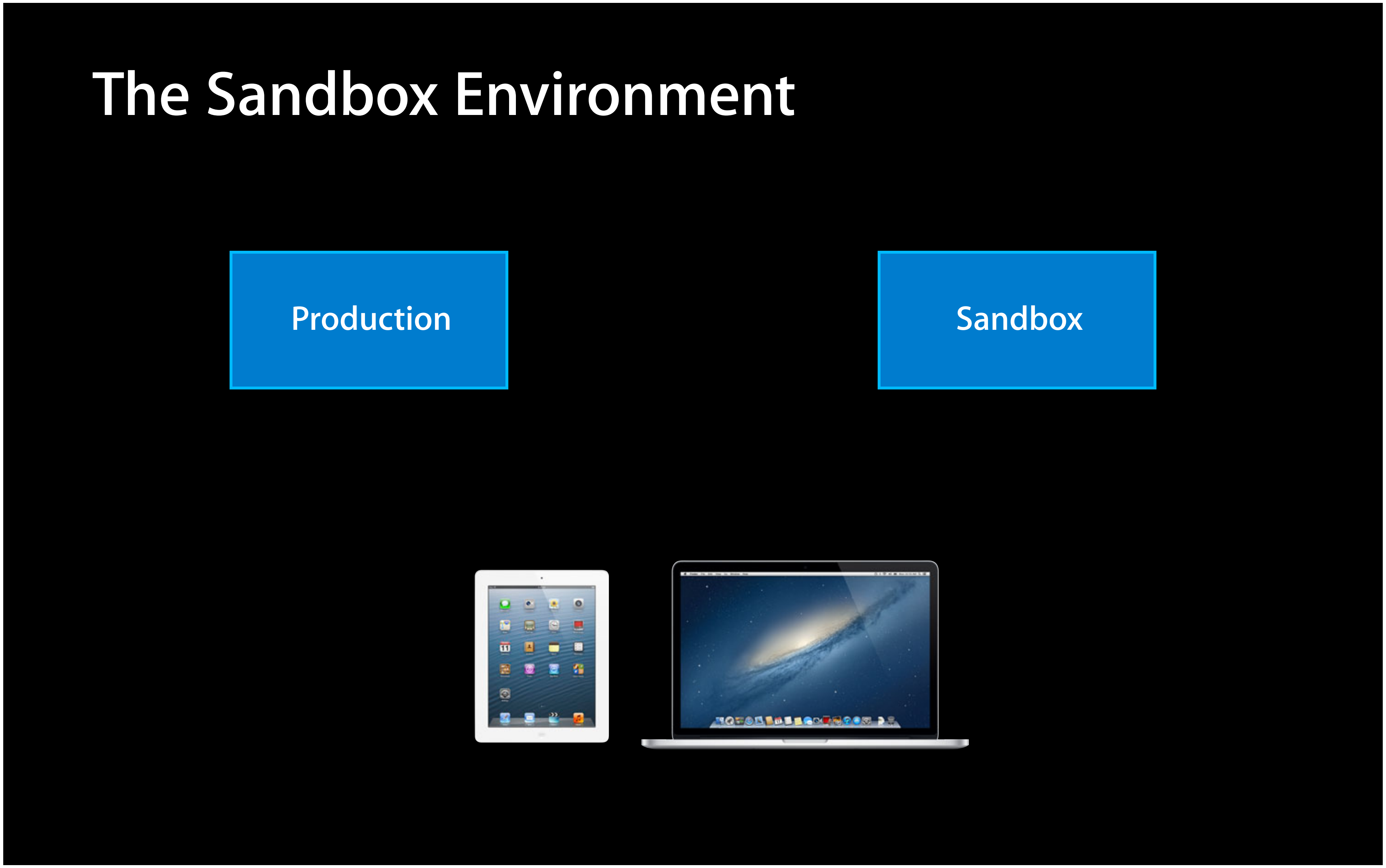
The Sandbox Environment

The Sandbox Environment

The Sandbox Environment

Production

Sandbox

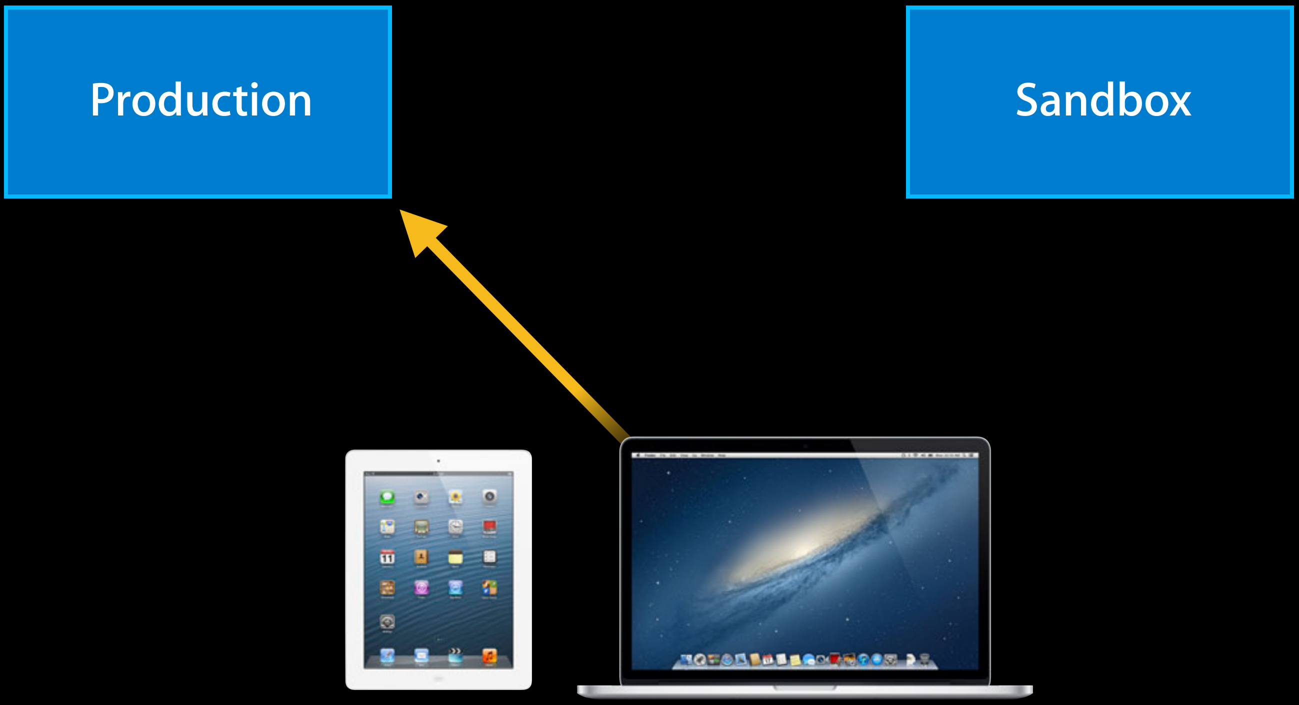


The Sandbox Environment

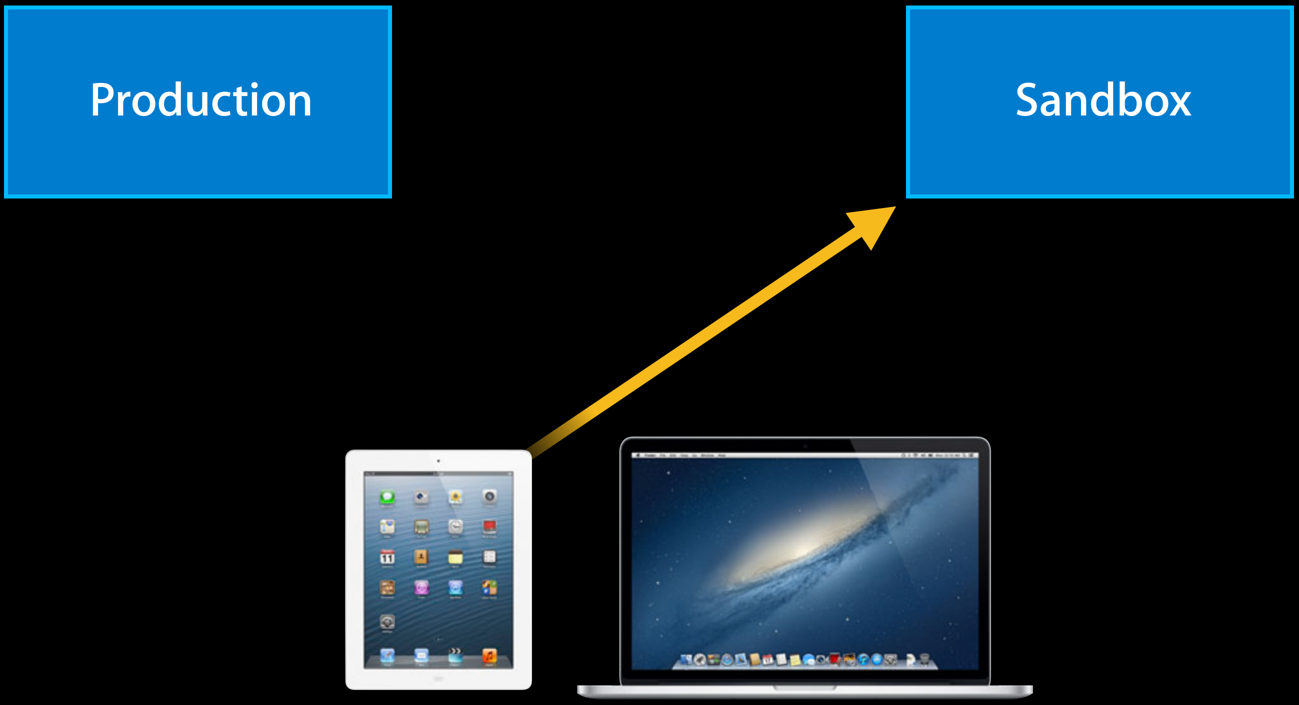
Production

Sandbox

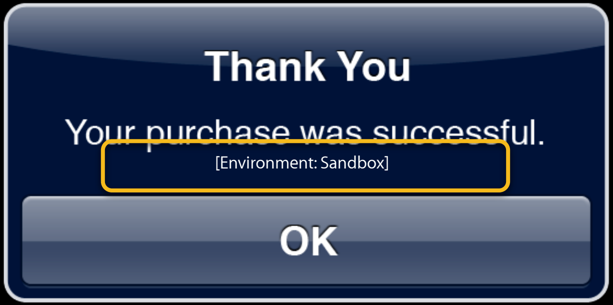
The Sandbox Environment



The Sandbox Environment



The Sandbox Environment



Thank You

Your purchase was successful.

[Environment: Sandbox]

The Sandbox Environment

How

The Sandbox Environment

How

- Setup in iTunes Connect

The Sandbox Environment

How

- Setup in iTunes Connect
    - Create a test user

The Sandbox Environment

How

- Setup in iTunes Connect
  - Create a test user
  - Enter products for sale

The Sandbox Environment

How

- Setup in iTunes Connect
    - Create a test user
    - Enter products for sale
- Build and sign

The Sandbox Environment

How

- Setup in iTunes Connect
  - Create a test user
  - Enter products for sale
- Build and sign
- Mac: Fetch a receipt

The Sandbox Environment

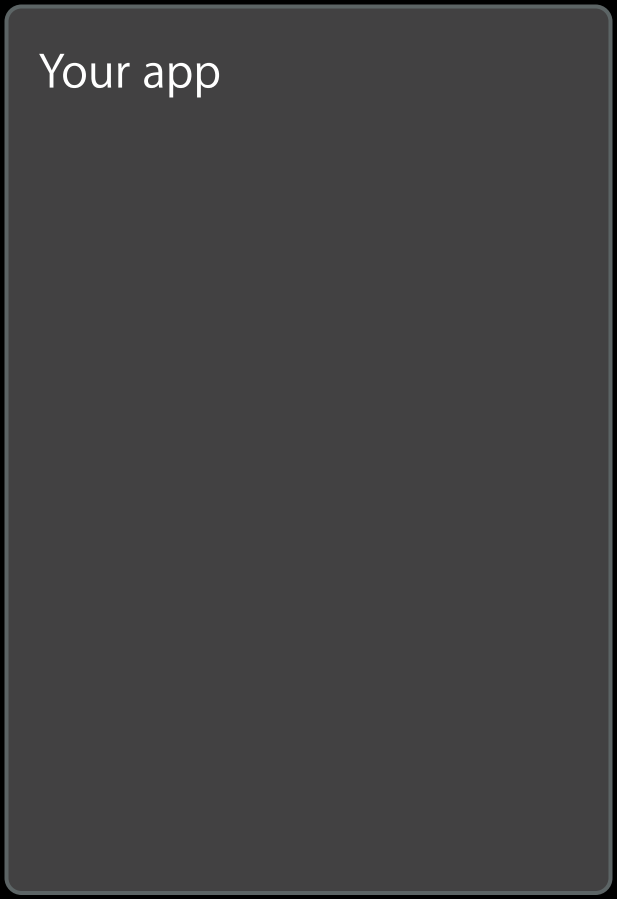
How

- Setup in iTunes Connect
  - Create a test user
  - Enter products for sale
- Build and sign
- Mac: Fetch a receipt
- Buy a product!

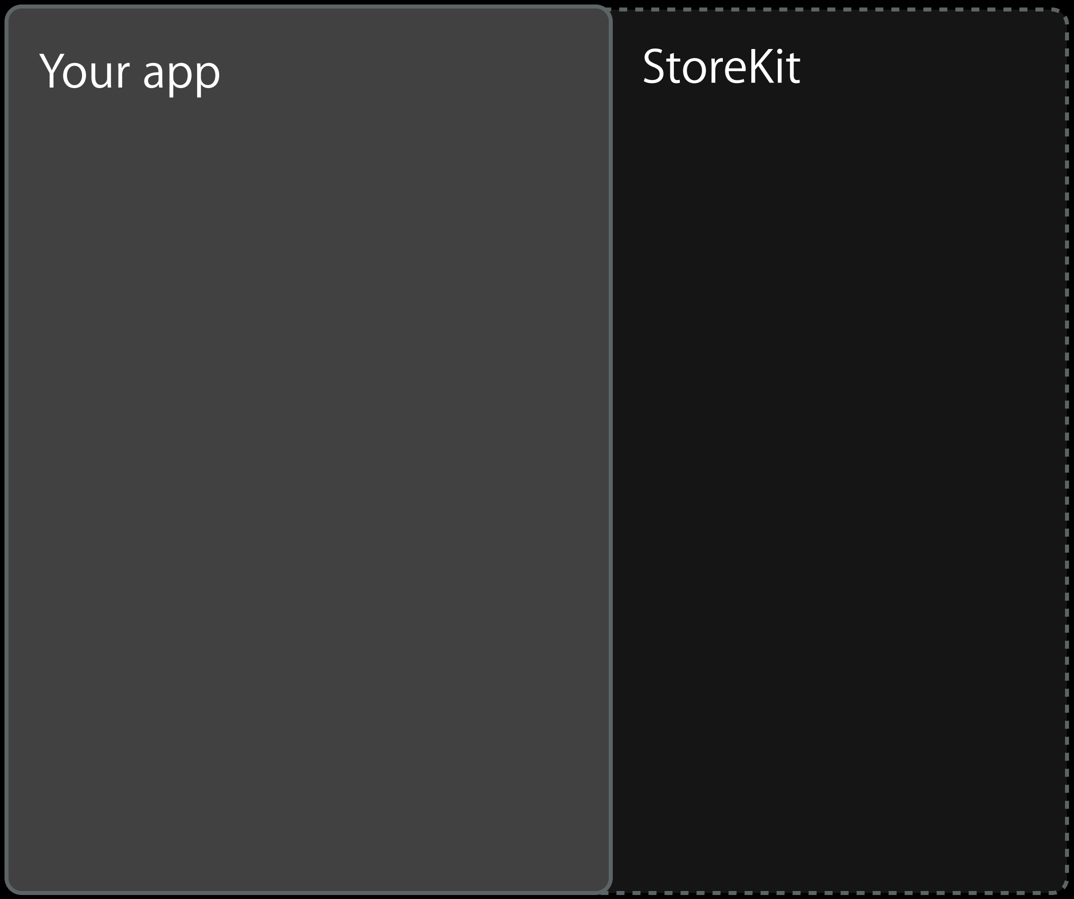
Today’s Agenda

- Selling Store Content
- Using In-App Purchase
- In Detail: The Purchase Queue
- App Store Hosted Content
- Best Practices

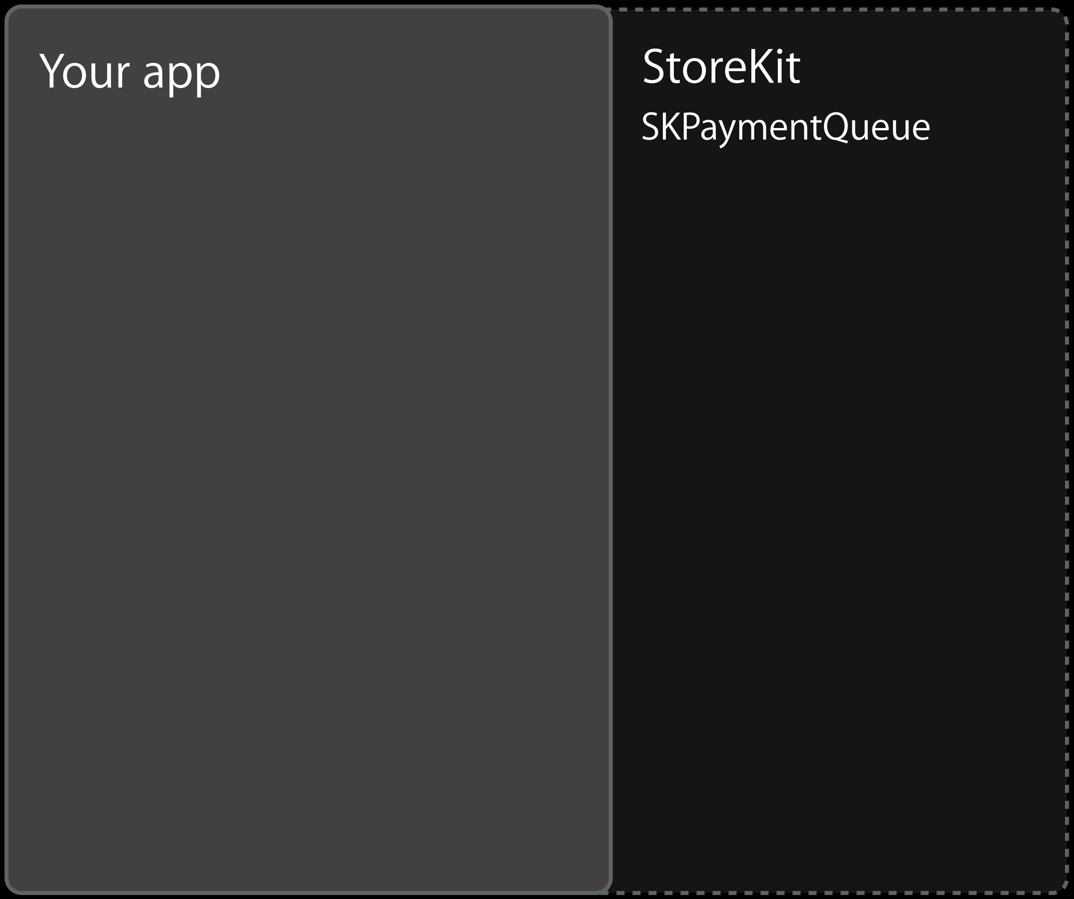
SKPaymentQueue



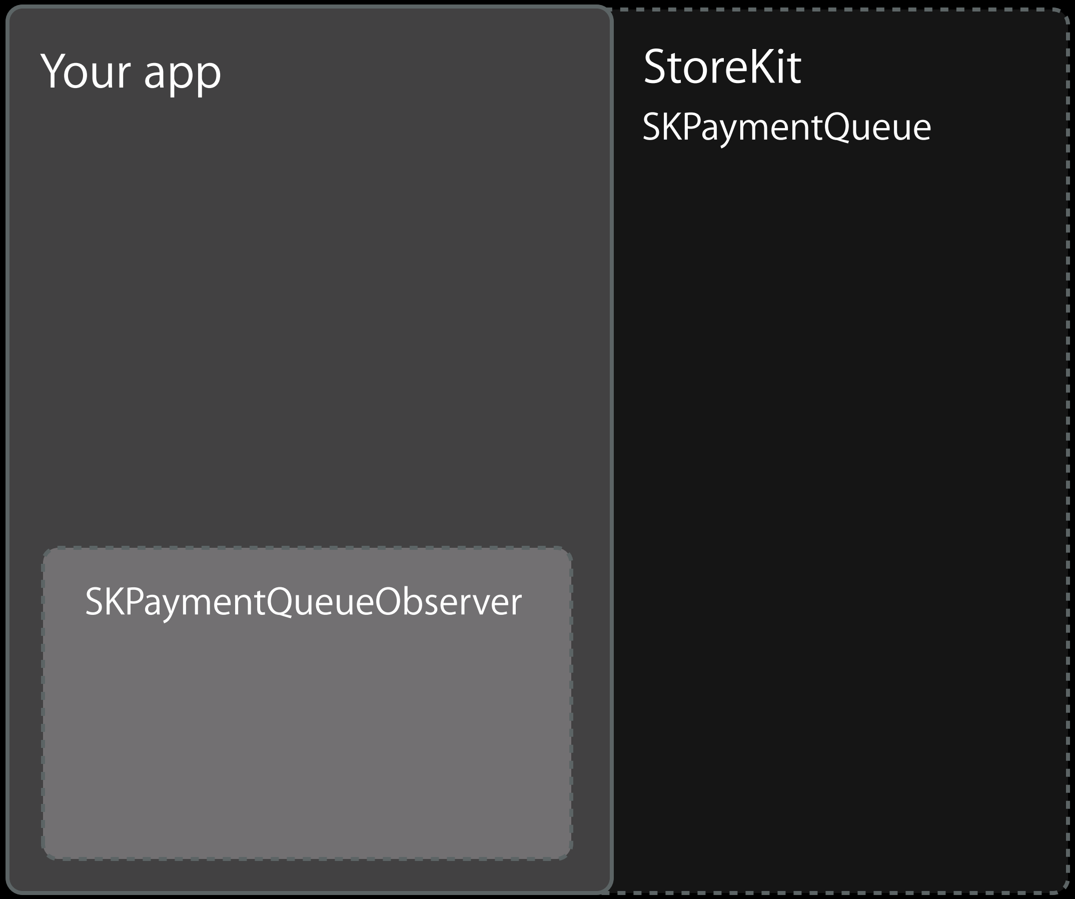
SKPaymentQueue



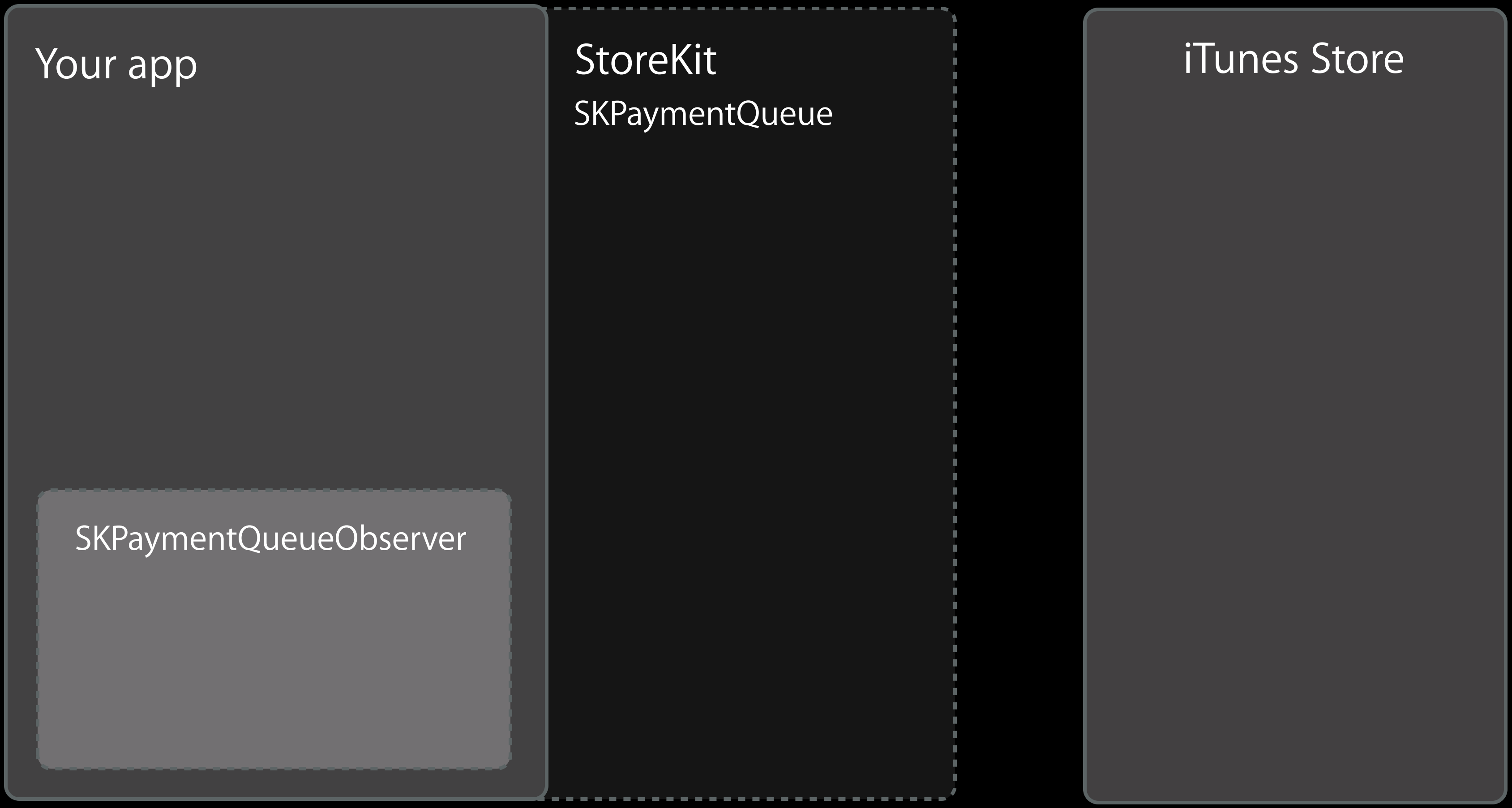
SKPaymentQueue



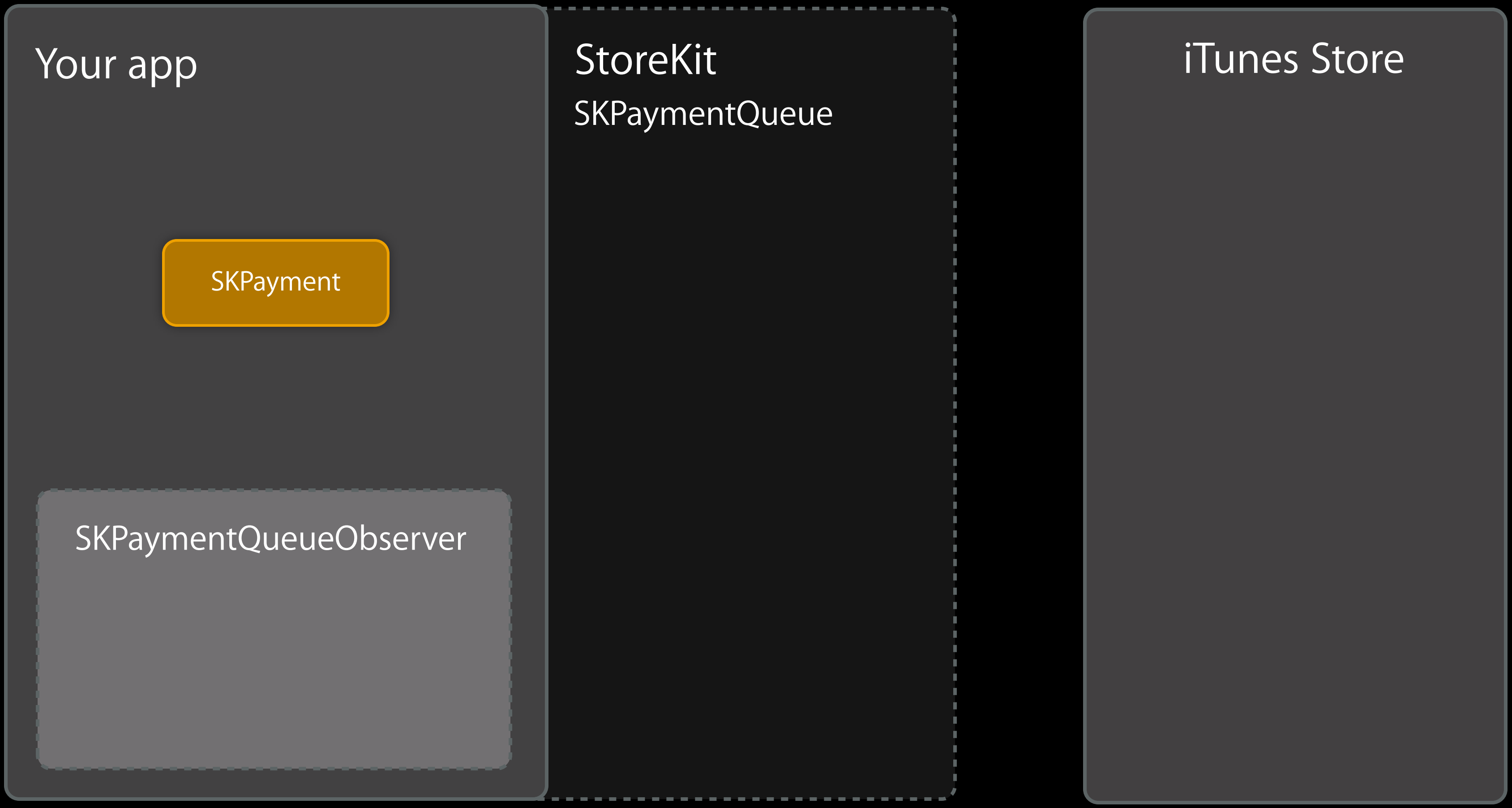
SKPaymentQueue



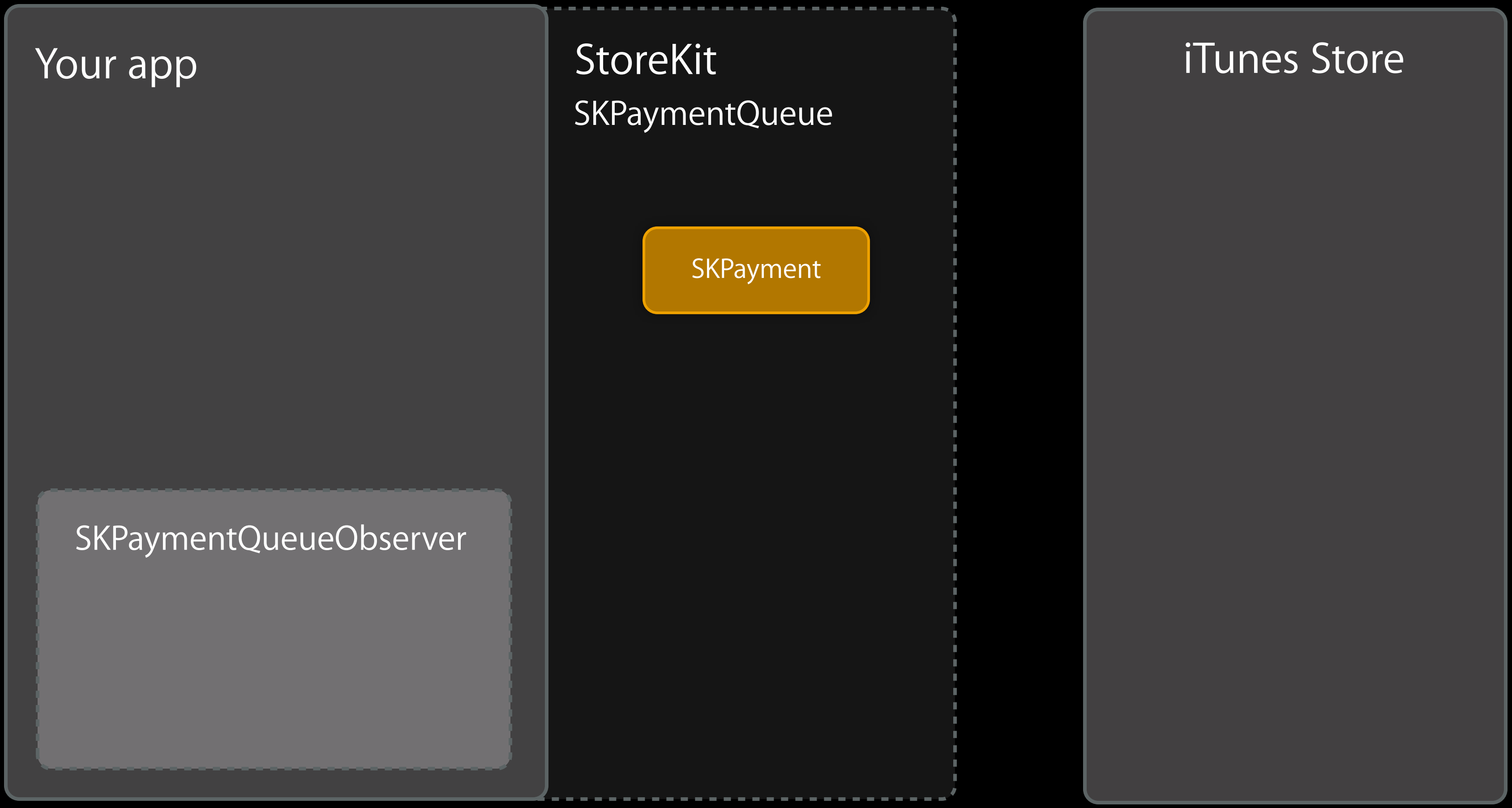
SKPaymentQueue



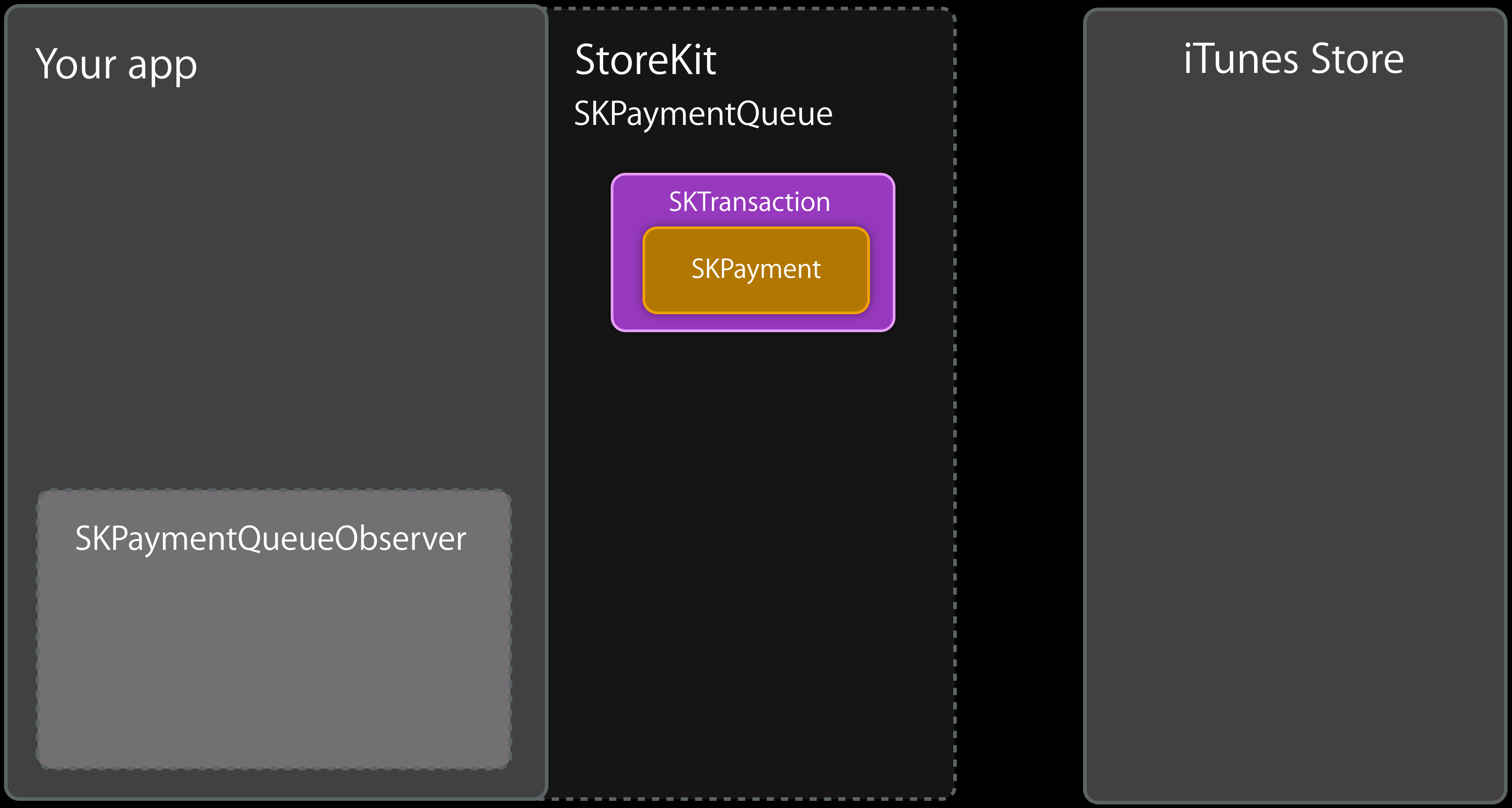
SKPaymentQueue



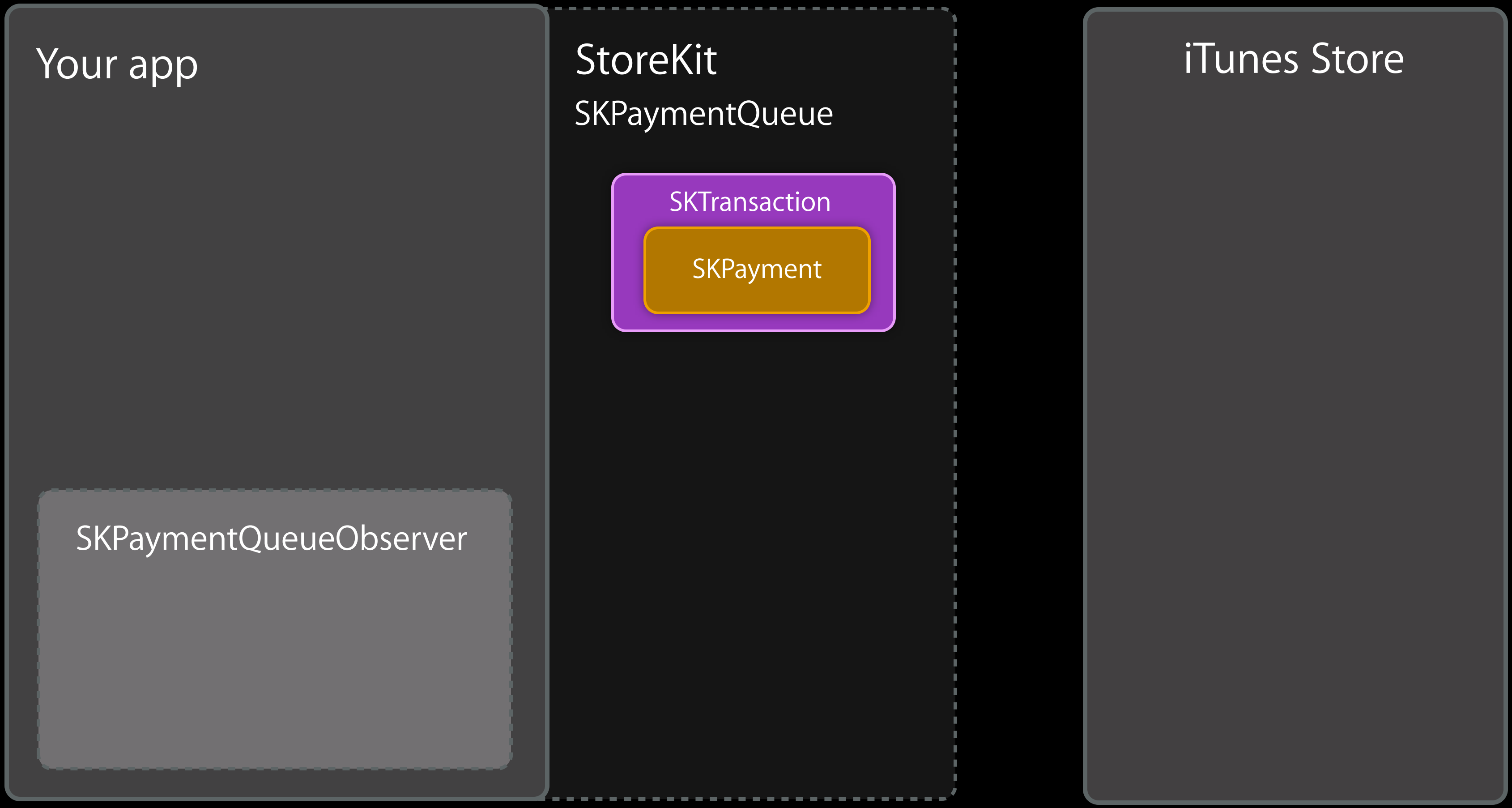
SKPaymentQueue



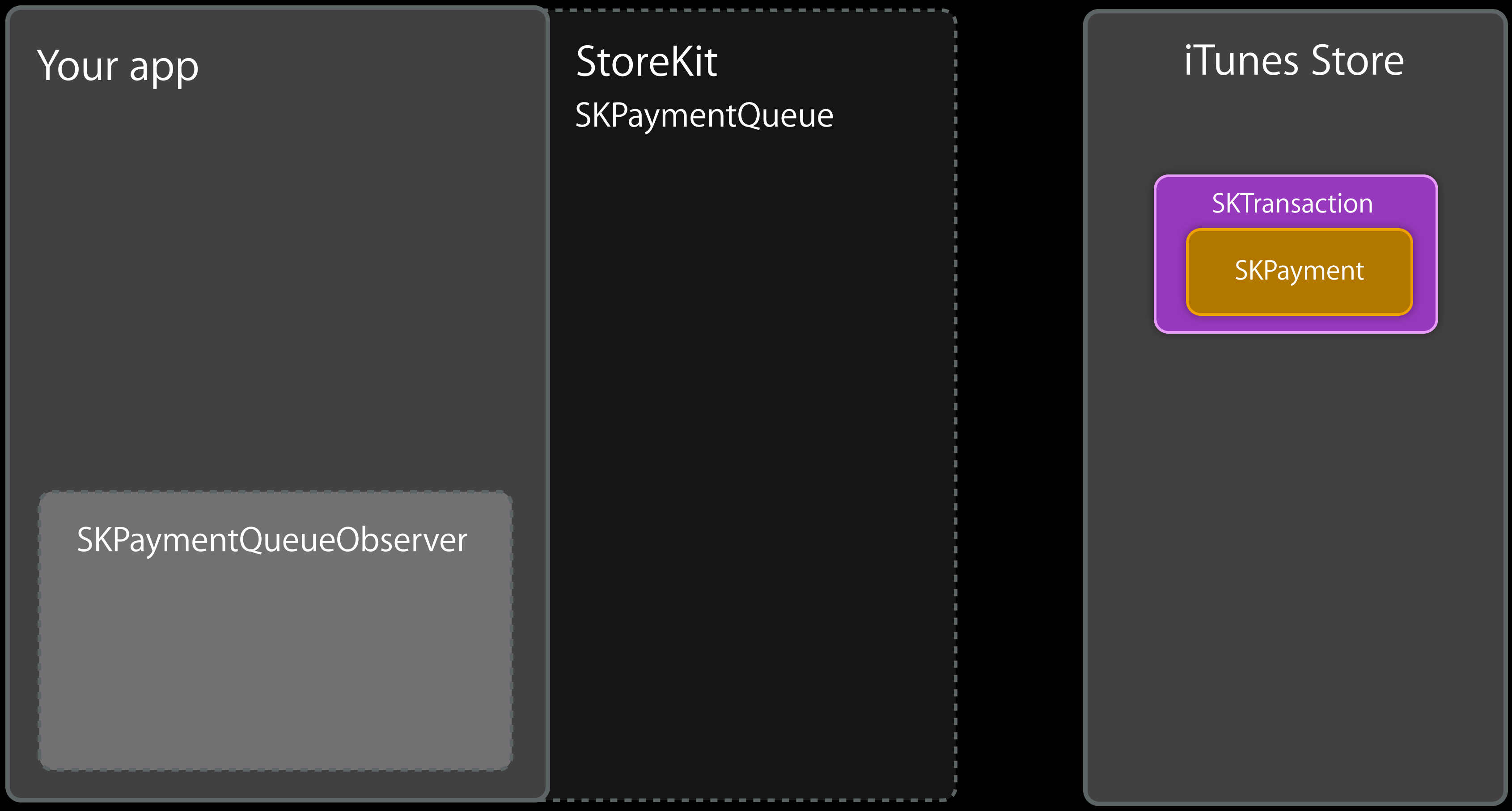
SKPaymentQueue



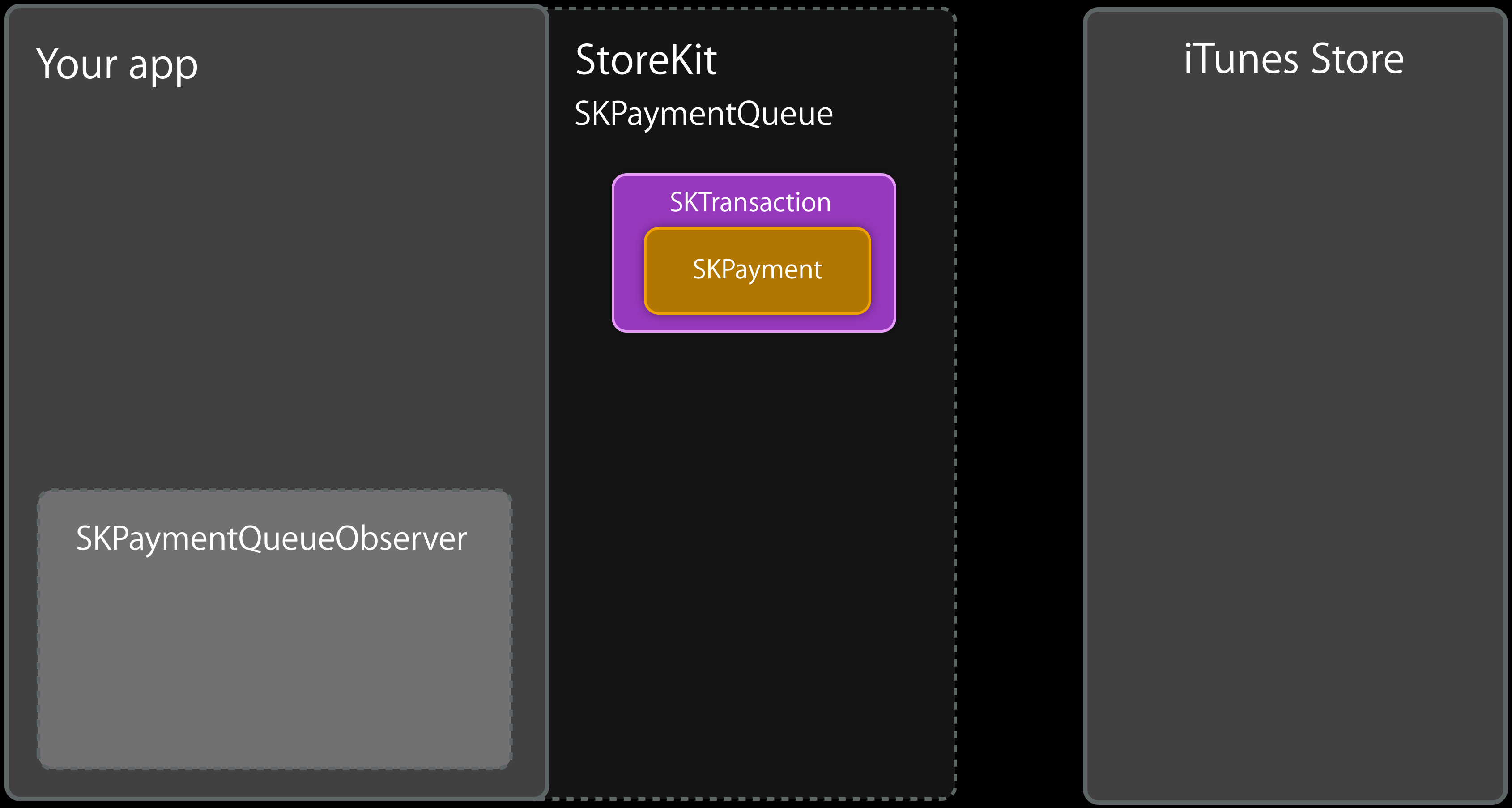
SKPaymentQueue



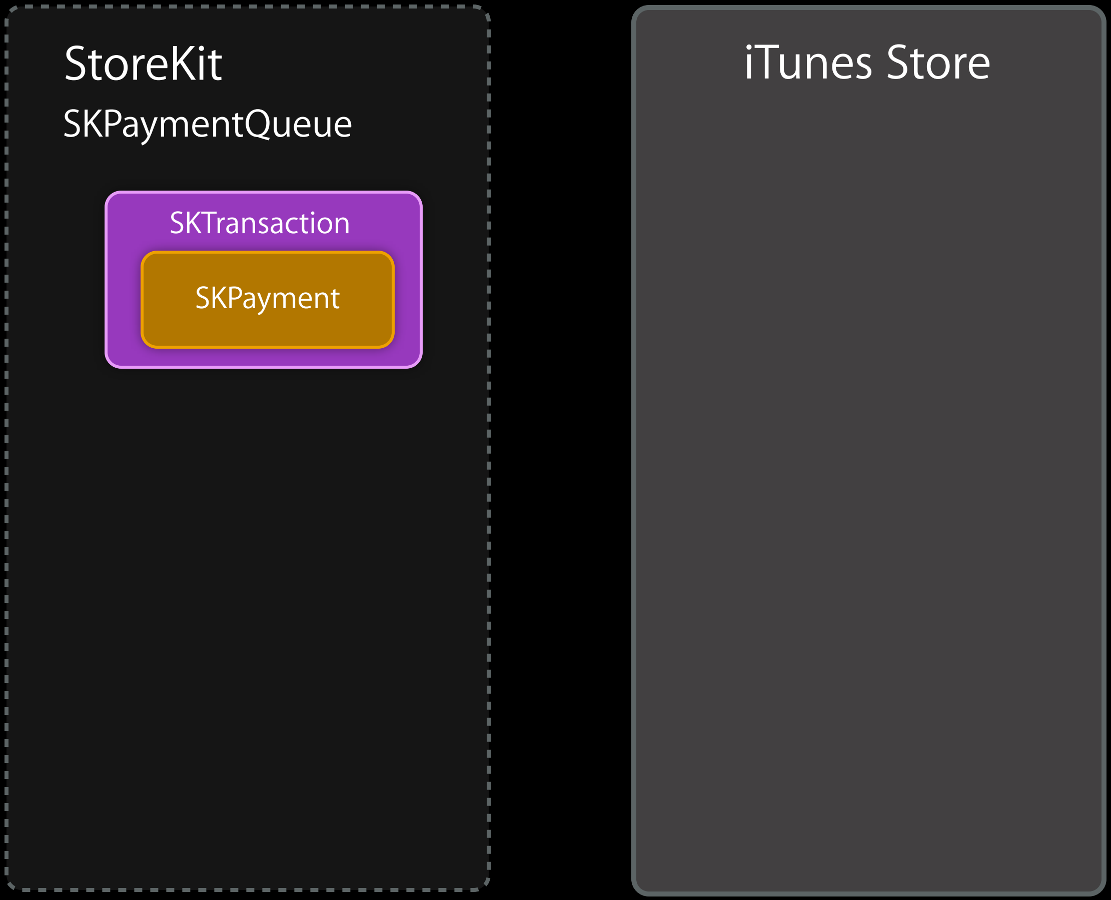
SKPaymentQueue



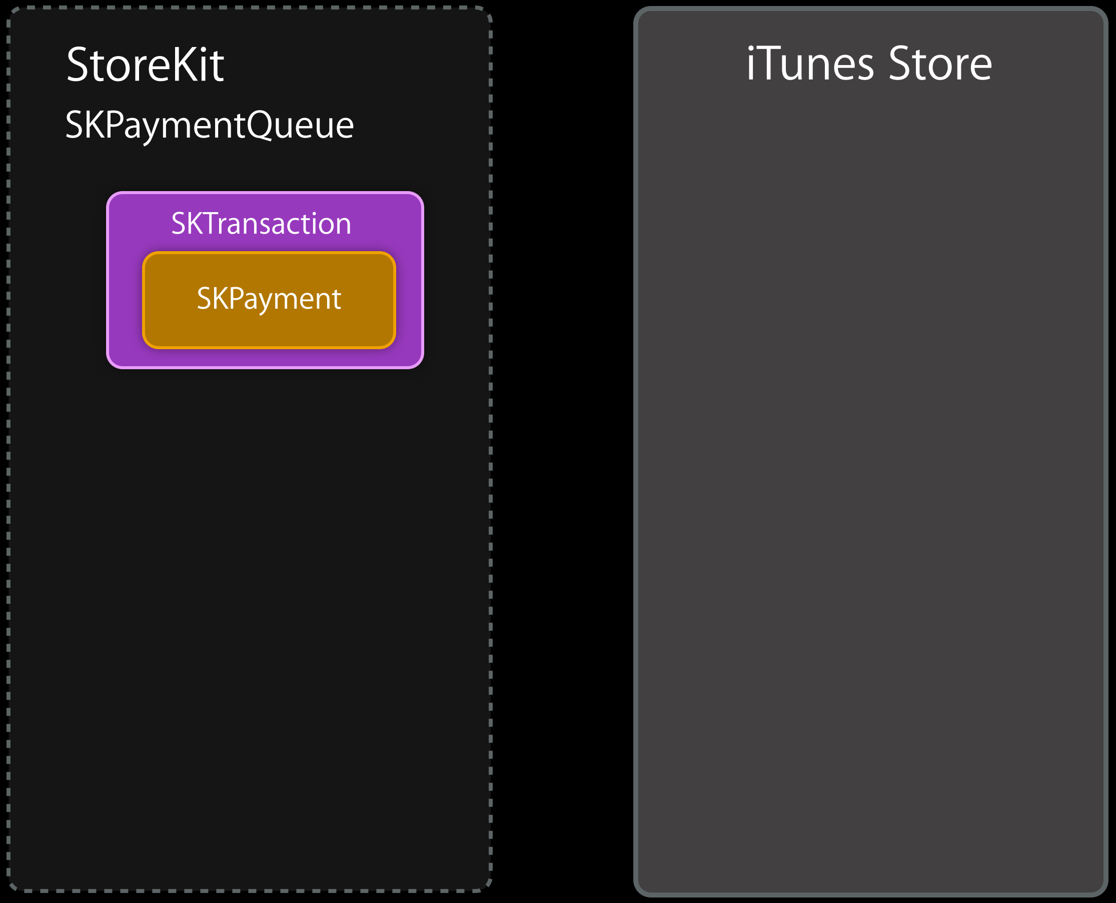
SKPaymentQueue



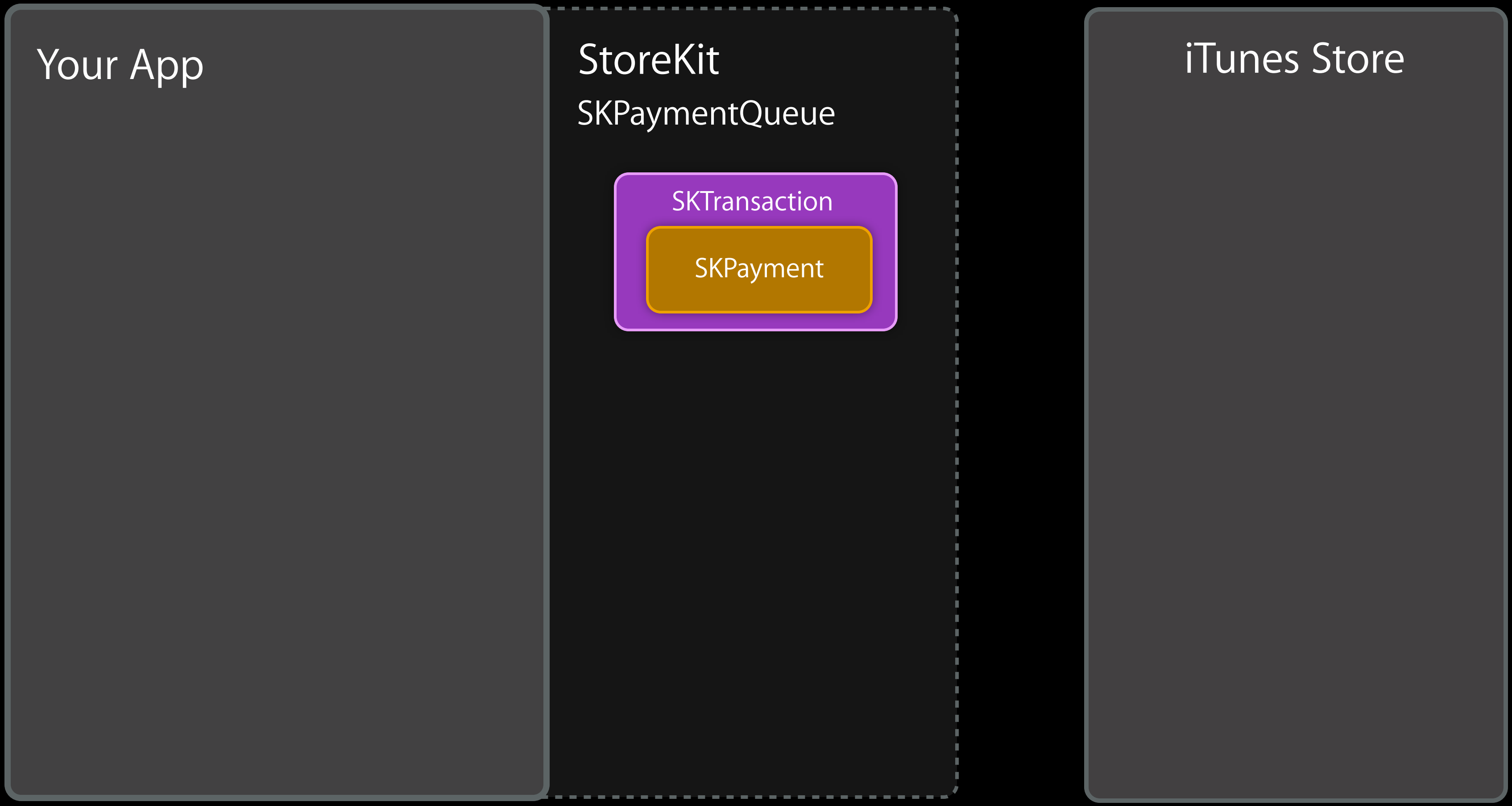
SKPaymentQueue



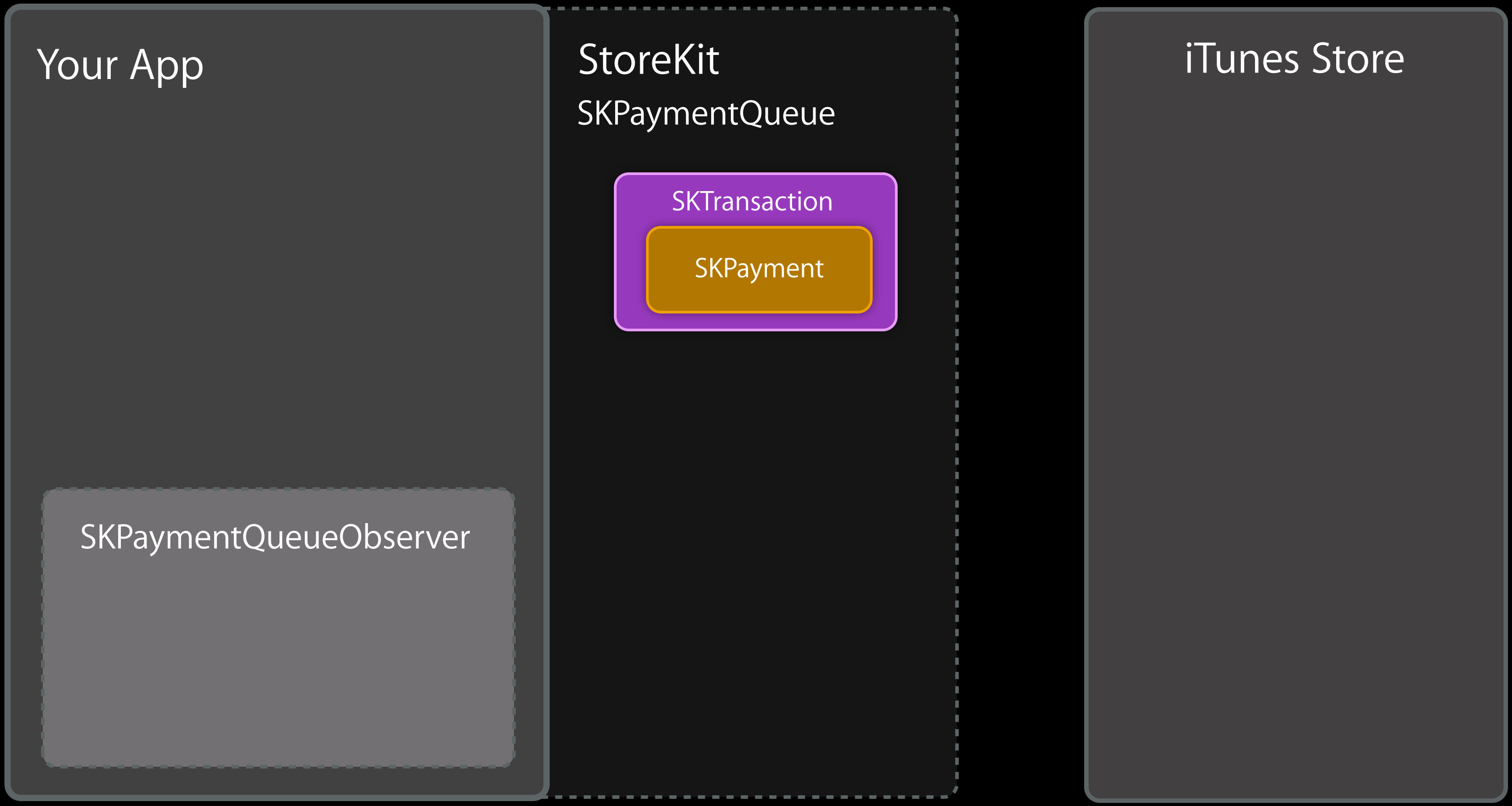
SKPaymentQueue



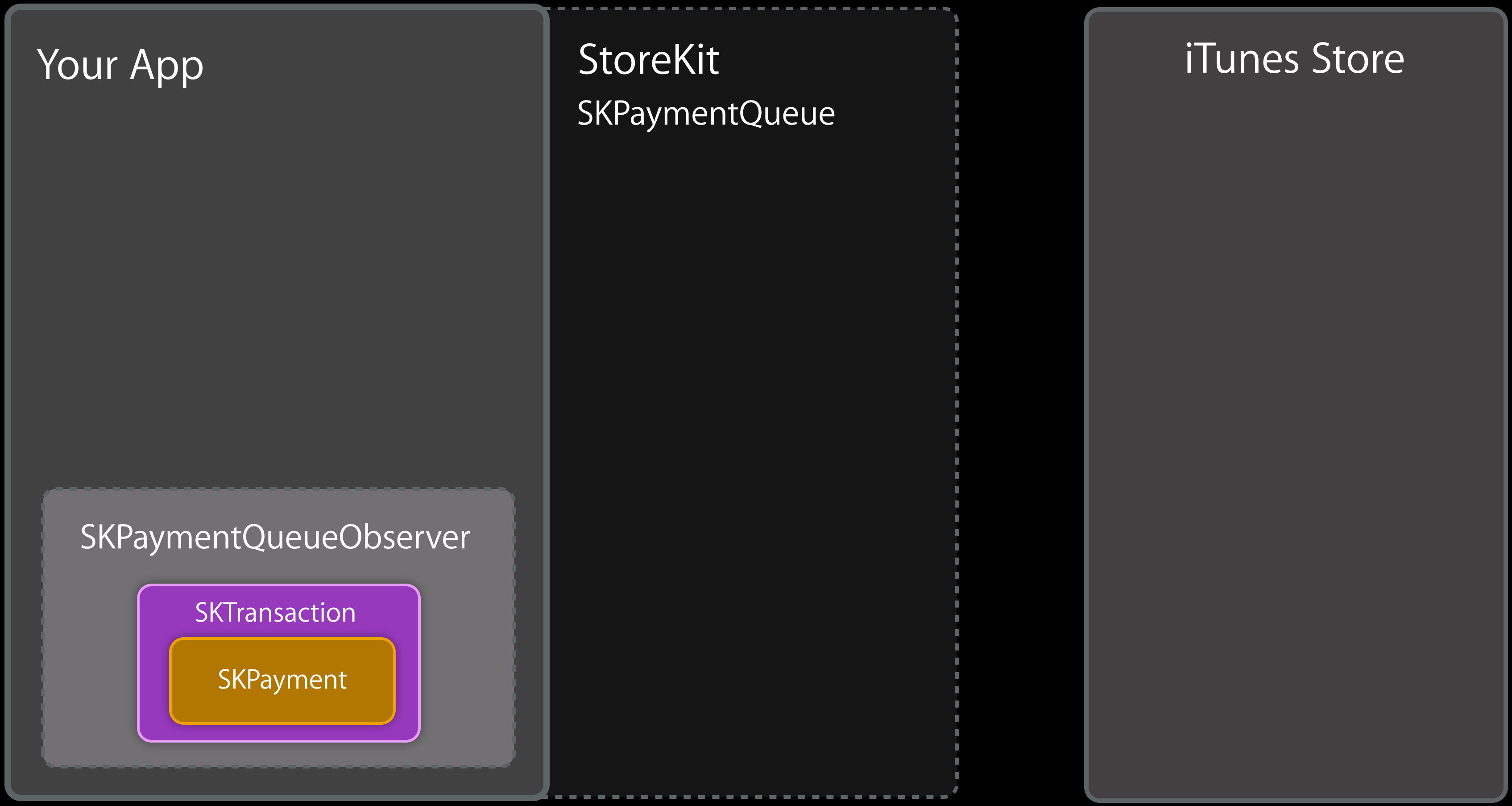
SKPaymentQueue



SKPaymentQueue

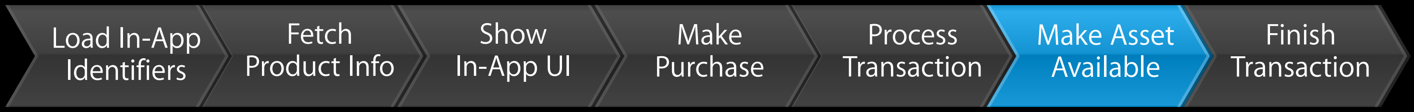


SKPaymentQueue



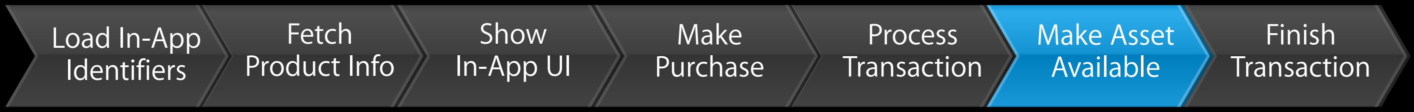
Today's Agenda

- Selling Store Content
- Using In-App Purchase
- In Detail: The Purchase Queue
- App Store Hosted Content
- Best Practices


Load In-App Identifiers
Fetch Product Info
Show In-App UI
Make Purchase
Process Transaction
Make Asset Available
Finish Transaction

New

The App Store will now host your In-App content for you!



App Store Hosted Content

App Store Hosted Content

- Don’t need to host your own content

App Store Hosted Content

- Don’t need to host your own content
  - Save time, money, and bugs

App Store Hosted Content

- Don’t need to host your own content
  - Save time, money, and bugs
  - Scalable and reliable

App Store Hosted Content

- Don’t need to host your own content
  - Save time, money, and bugs
  - Scalable and reliable
- Easy API

App Store Hosted Content

- Don’t need to host your own content
  - Save time, money, and bugs
  - Scalable and reliable
- Easy API
  - Save development time

App Store Hosted Content

- Don’t need to host your own content
  - Save time, money, and bugs
  - Scalable and reliable
- Easy API
  - Save development time
  - Comes with a security model

App Store Hosted Content

- Don’t need to host your own content
  - Save time, money, and bugs
  - Scalable and reliable
- Easy API
  - Save development time
  - Comes with a security model
  - Take advantage of background downloads

App Store Hosted Content

App Store Hosted Content

- No additional cost to use this

App Store Hosted Content

- No additional cost to use this
- Limit of 2GB

App Store Hosted Content

- No additional cost to use this
- Limit of 2GB
- They go through review

App Store Hosted Content

- No additional cost to use this
- Limit of 2GB
- They go through review
- No code

App Store Hosted Content

- No additional cost to use this
- Limit of 2GB
- They go through review
- No code
- Same content rules as apps

App Store Hosted Content

Workflow

App Store Hosted Content

Workflow

- Build and test your content

App Store Hosted Content

Workflow

- Build and test your content
- Upload your content to iTunes Connect

App Store Hosted Content

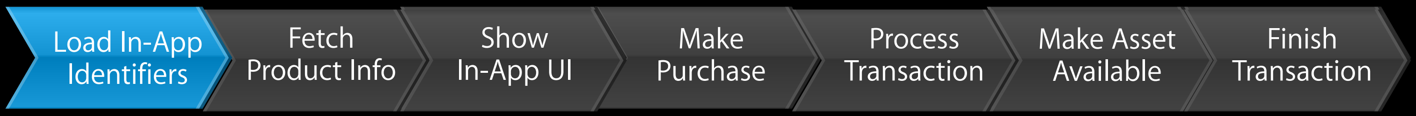
Workflow

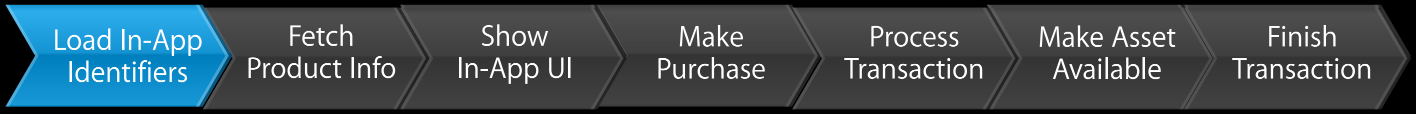
- Build and test your content
- Upload your content to iTunes Connect
- App Store will host that content

App Store Hosted Content

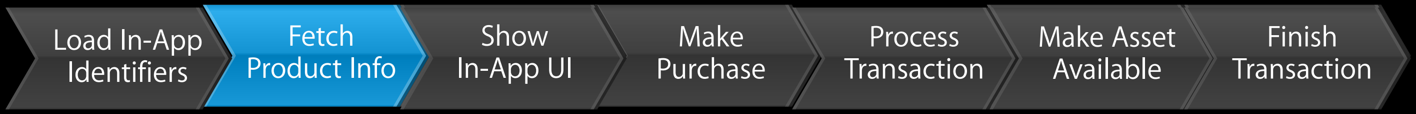
Workflow

- Build and test your content
- Upload your content to iTunes Connect
- App Store will host that content
- Use new Store Kit API to download content


Load In-App Identifiers
Fetch Product Info
Show In-App UI
Make Purchase
Process Transaction
Make Asset Available
Finish Transaction

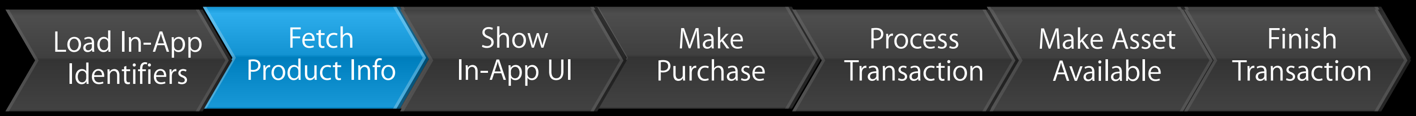

Load In-App Identifiers
Fetch Product Info
Show In-App UI
Make Purchase
Process Transaction
Make Asset Available
Finish Transaction

No Change


Load In-App Identifiers
Fetch Product Info
Show In-App UI
Make Purchase
Process Transaction
Make Asset Available
Finish Transaction

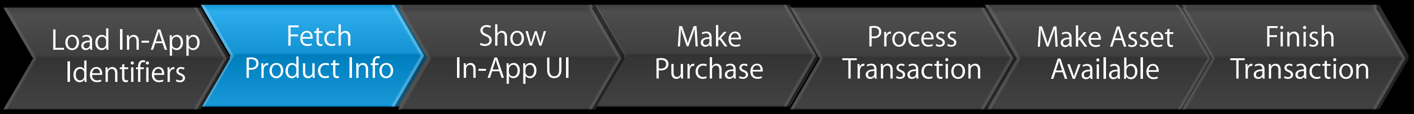
Load In-App Identifiers | **Fetch Product Info** | Show In-App UI | Make Purchase | Process Transaction | Make Asset Available | Finish Transaction

`SKProduct` has new properties



`SKProduct` has new properties

- BOOL downloadable;



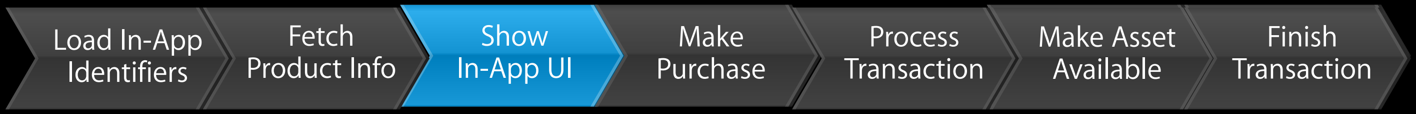
`SKProduct` has new properties

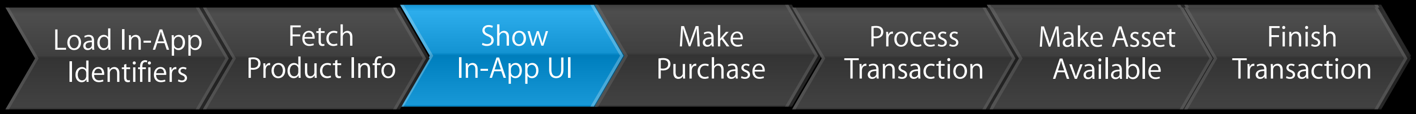
- BOOL downloadable;
- NSString* contentVersion;

Load In-App Identifiers | **Fetch Product Info** | Show In-App UI | Make Purchase | Process Transaction | Make Asset Available | Finish Transaction

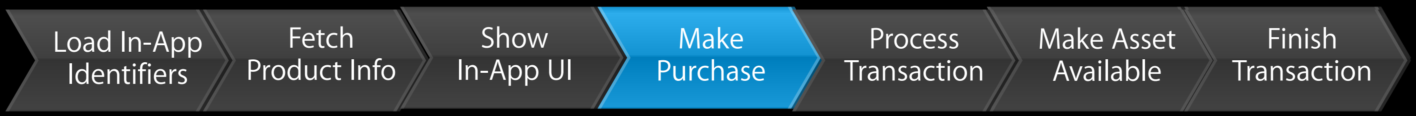
`SKProduct` has new properties

- BOOL downloadable;
- NSString* contentVersion;
- NSArray* contentLengths;


Load In-App Identifiers
Fetch Product Info
Show In-App UI
Make Purchase
Process Transaction
Make Asset Available
Finish Transaction

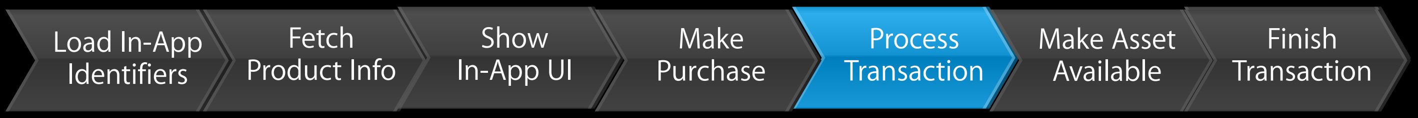

Load In-App Identifiers
Fetch Product Info
Show In-App UI
Make Purchase
Process Transaction
Make Asset Available
Finish Transaction

No Change


Load In-App Identifiers
Fetch Product Info
Show In-App UI
Make Purchase
Process Transaction
Make Asset Available
Finish Transaction

Load In-App Identifiers → Fetch Product Info → Show In-App UI → **Make Purchase** → Process Transaction → Make Asset Available → Finish Transaction

No Change


Load In-App Identifiers
Fetch Product Info
Show In-App UI
Make Purchase
Process Transaction
Make Asset Available
Finish Transaction

Load In-App Identifiers | Fetch Product Info | Show In-App UI | Make Purchase | **Process Transaction** | Make Asset Available | Finish Transaction

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

Load In-App Identifiers → Fetch Product Info → Show In-App UI → Make Purchase → **Process Transaction** → Make Asset Available → Finish Transaction

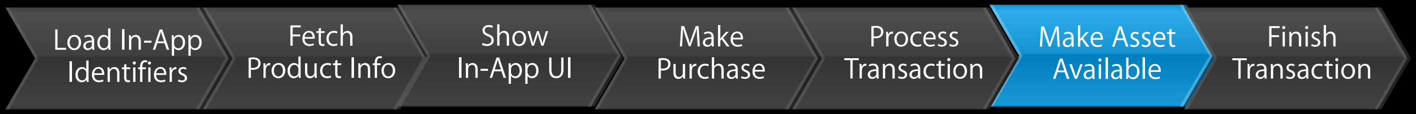
```
- (void)paymentQueue:(SKPaymentQueue *)queue updatedTransactions:(NSArray
*)transactions

    for(SKPaymentTransaction* transaction in transactions)
    {





    }
```

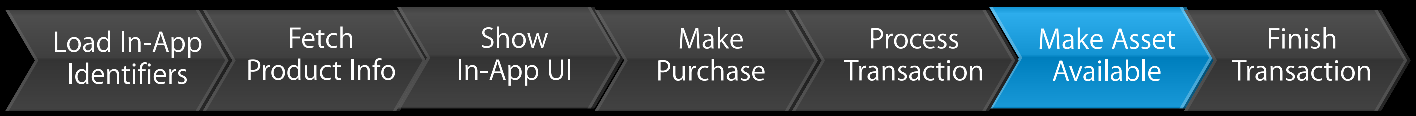
Load In-App Identifiers › Fetch Product Info › Show In-App UI › Make Purchase › **Process Transaction** › Make Asset Available › Finish Transaction

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

    for(SKPaymentTransaction* transaction in transactions)
    {

       if(transaction.downloads)



    }
```

Load In-App Identifiers › Fetch Product Info › Show In-App UI › Make Purchase › **Process Transaction** › Make Asset Available › Finish Transaction

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

   for(SKPaymentTransaction* transaction in transactions)
   {

      if(transaction.downloads)

          [[SKPaymentQueue defaultQueue] startDownloads:

          transaction.downloads];

   }
```


Load In-App Identifiers
Fetch Product Info
Show In-App UI
Make Purchase
Process Transaction
Make Asset Available
Finish Transaction



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

SKDownload

SKDownload

`download.progress`	`0.128`

SKDownload

`download.progress`	`0.128`
`download.timeRemaining`	`213 (seconds)`

SKDownload

`download.progress`	`0.128`
`download.timeRemaining`	`213 (seconds)`

`download.state`	`SKDownloadStateActive`
	`SKDownloadStateWaiting`
	`SKDownloadStateFinished`
	`SKDownloadStateFailed`
	`SKDownloadStatePaused`
	`SKDownloadStateCancelled`

SKDownload

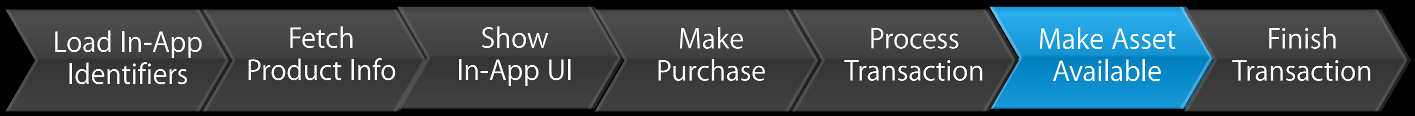
download.progress	0.128
download.timeRemaining	213 (seconds)
download.state	SKDownloadStateActive SKDownloadStateWaiting SKDownloadStateFinished SKDownloadStateFailed SKDownloadStatePaused SKDownloadStateCancelled
download.error	NSError

SKDownload

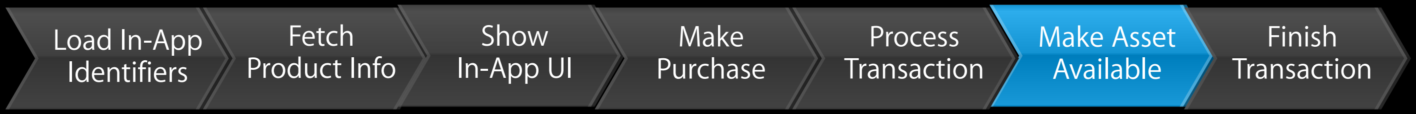
`download.progress`	`0.128`
`download.timeRemaining`	`213 (seconds)`

`download.state`	`SKDownloadStateActive`
	`SKDownloadStateWaiting`
	`SKDownloadStateFinished`
	`SKDownloadStateFailed`
	`SKDownloadStatePaused`
	`SKDownloadStateCancelled`

`download.error`	`NSError`
`download.contentURL`	`file:// URL`

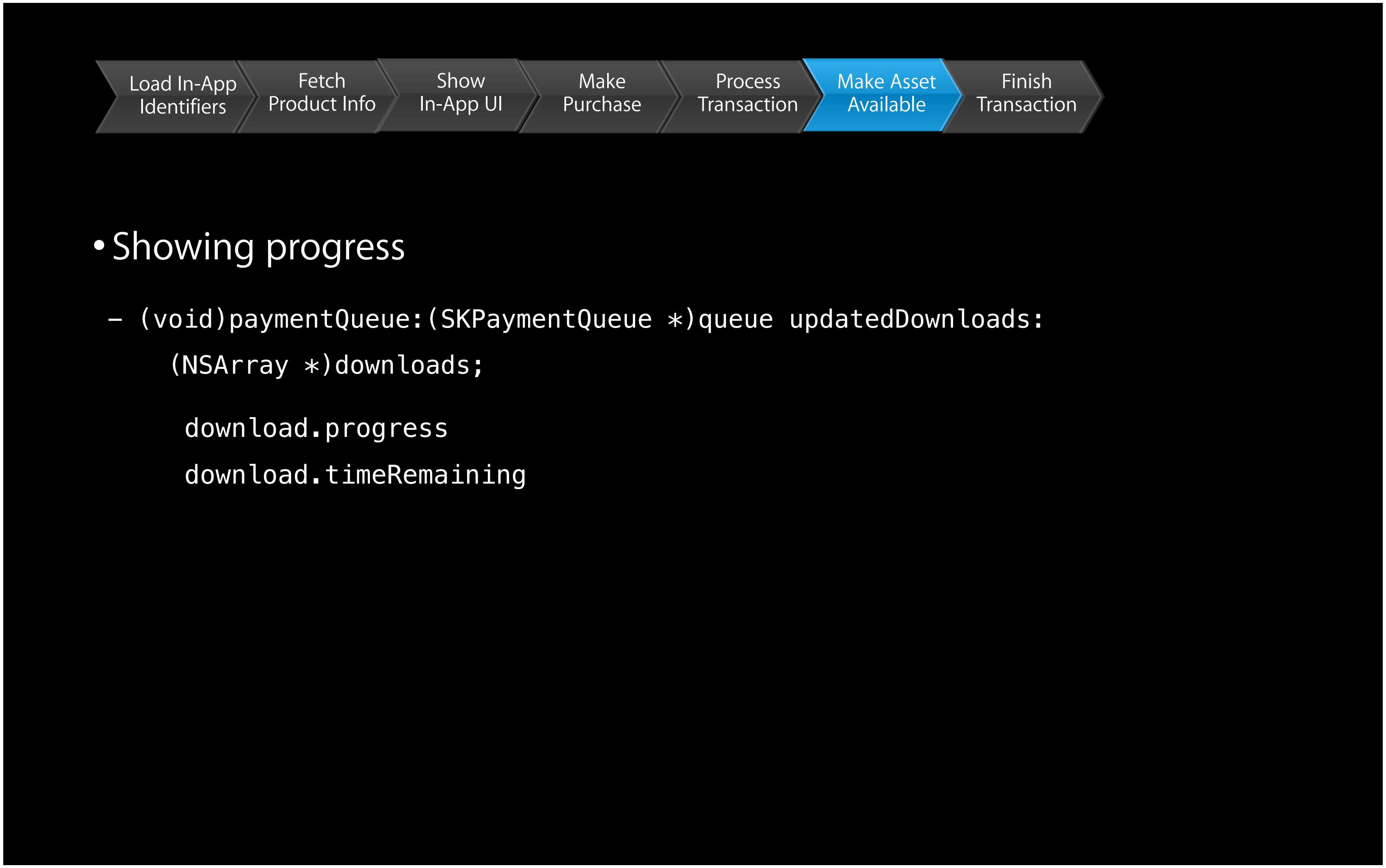


- Showing progress



- Showing progress

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



Load In-App Identifiers → Fetch Product Info → Show In-App UI → Make Purchase → Process Transaction → **Make Asset Available** → Finish Transaction

- Showing progress

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

     download.progress
     download.timeRemaining
```

Load In-App Identifiers → Fetch Product Info → Show In-App UI → Make Purchase → Process Transaction → **Make Asset Available** → Finish Transaction

- Showing progress

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

     download.progress
     download.timeRemaining
     download.state
     download.error
```

Load In-App Identifiers › Fetch Product Info › Show In-App UI › Make Purchase › Process Transaction › **Make Asset Available** › Finish Transaction

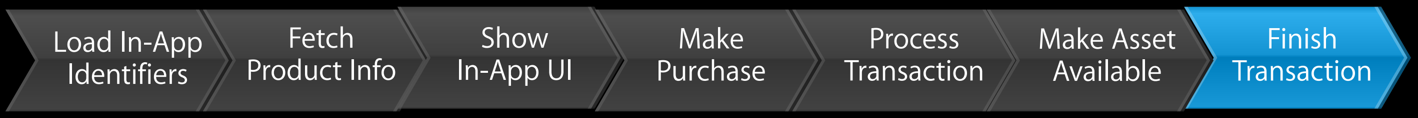
- Pausing and resuming

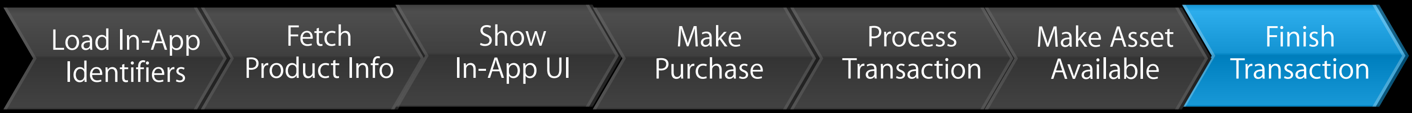
```
SKPaymentQueue
- (void) pauseDownloads:(NSArray *)downloads;
- (void) resumeDownloads:(NSArray *)downloads;
- (void) cancelDownloads:(NSArray *)downloads;
```

Load In-App Identifiers → Fetch Product Info → Show In-App UI → Make Purchase → Process Transaction → **Make Asset Available** → Finish Transaction

- Accessing the content

```
When SKDownload is in the SKDownloadStateFinished state:
  download.contentURL
```


Load In-App Identifiers
Fetch Product Info
Show In-App UI
Make Purchase
Process Transaction
Make Asset Available
Finish Transaction


Load In-App Identifiers
Fetch Product Info
Show In-App UI
Make Purchase
Process Transaction
Make Asset Available
Finish Transaction

No Change

Restoring App Store Hosted Content

Restoring App Store Hosted Content

- Just like non-hosted content

Restoring App Store Hosted Content

- Just like non-hosted content
- Check for `transaction.downloads`

Restoring App Store Hosted Content

- Just like non-hosted content
- Check for `transaction.downloads`
- Downloading is your decision

Restoring App Store Hosted Content

- Just like non-hosted content
- Check for `transaction.downloads`
- Downloading is your decision
    - Always call `finishTransaction:`

App Store Hosted Content

What form does it take?

App Store Hosted Content

What form does it take?

- A folder with any data you need

App Store Hosted Content

What form does it take?

- A folder with any data you need

  `ContentInfo.plist` at root level

App Store Hosted Content

What form does it take?

- A folder with any data you need

  `ContentInfo.plist` at root level

  `ContentVersion`

App Store Hosted Content

What form does it take?

- A folder with any data you need

  `ContentInfo.plist` at root level

  ```
  ContentVersion
  IAPProductIdentifier
  ```

App Store Hosted Content

What form does it take?

- A folder with any data you need

  `ContentInfo.plist` at root level

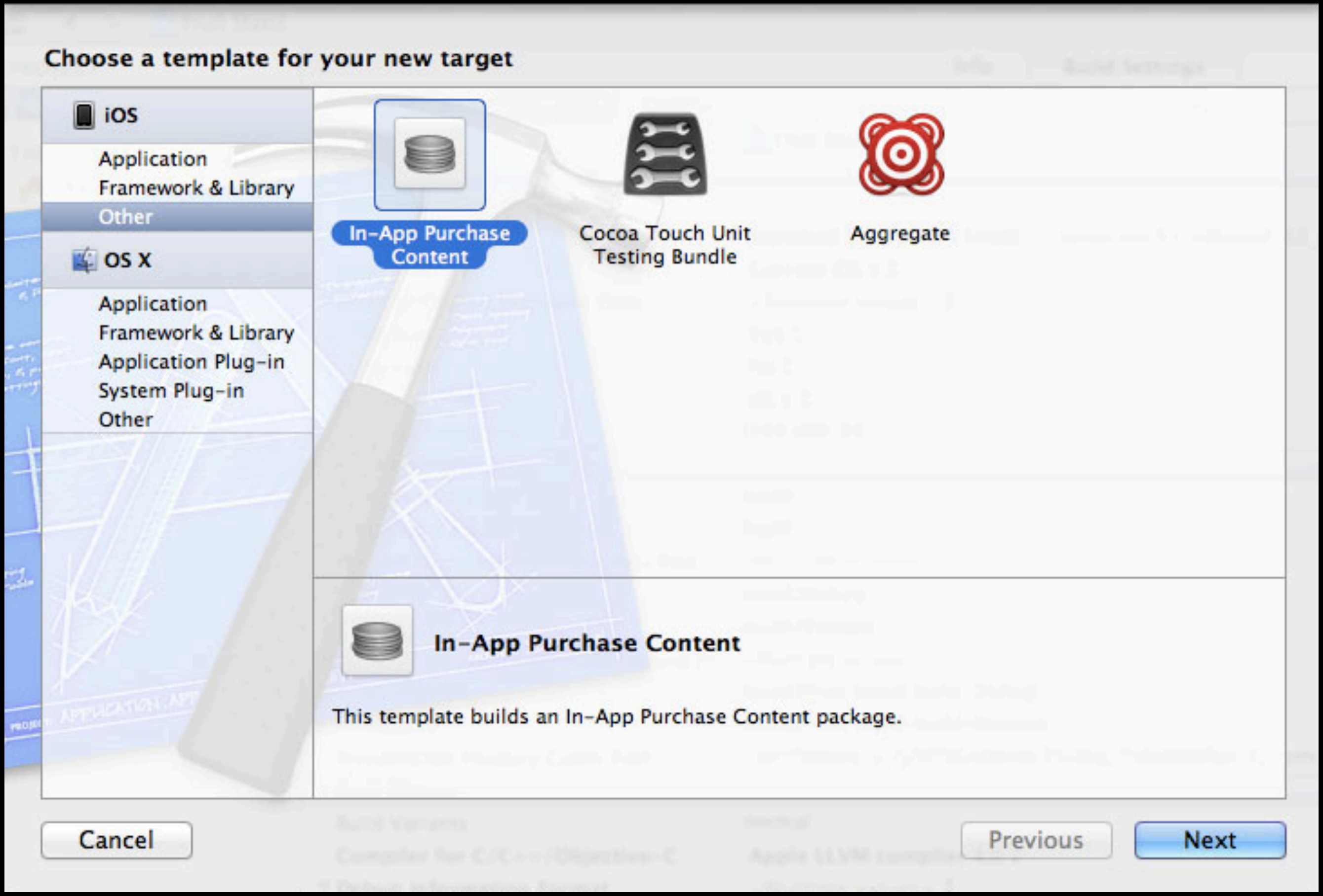
  ```
  ContentVersion
  IAPProductIdentifier
  ```

- Other data in `Contents` subfolder

App Store Hosted Content

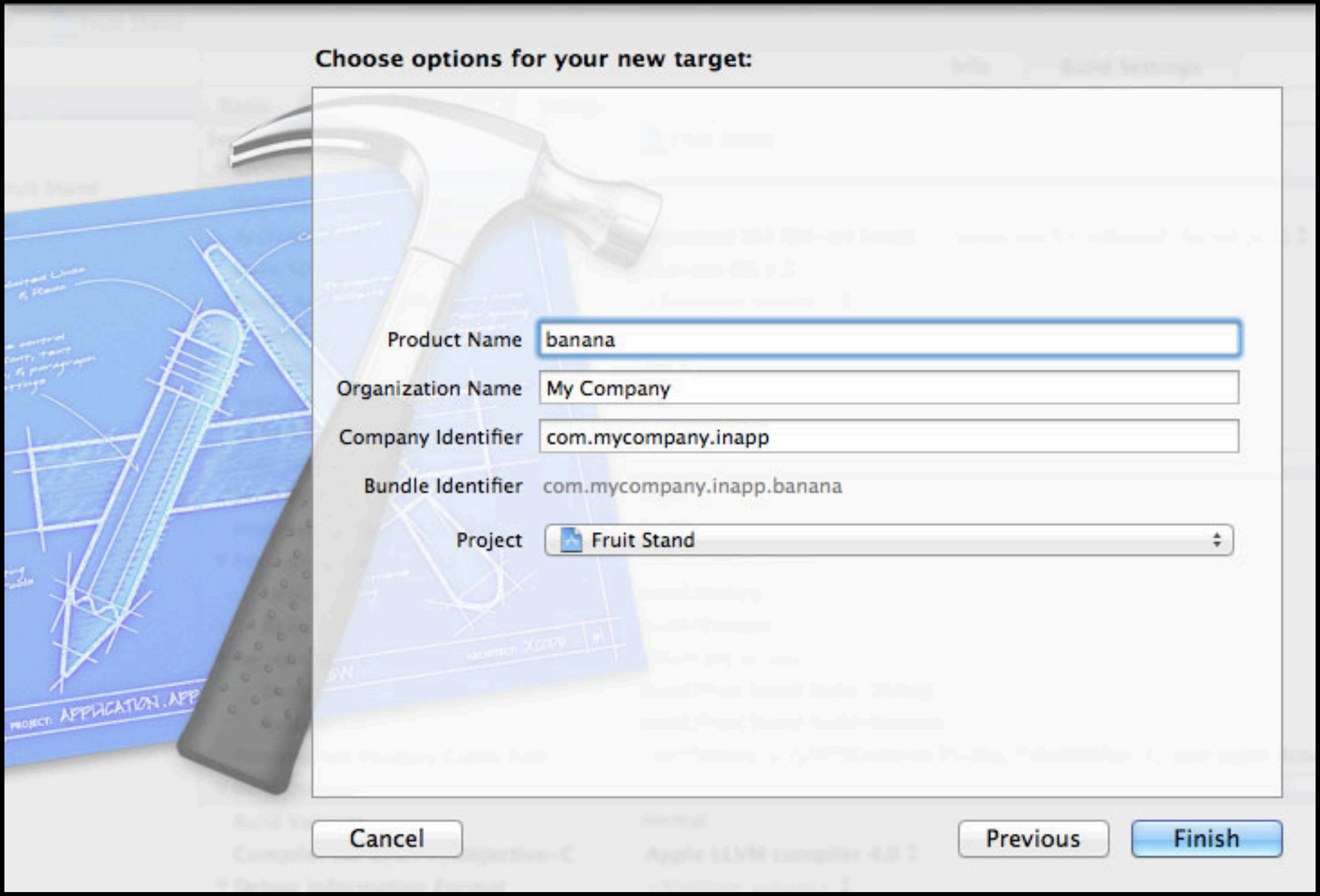
What form does it take?

```
…/
   ContentInfo.plist
   Contents/
       PieceOfContent1.mov
       PieceOfContent2.mov
       …
```



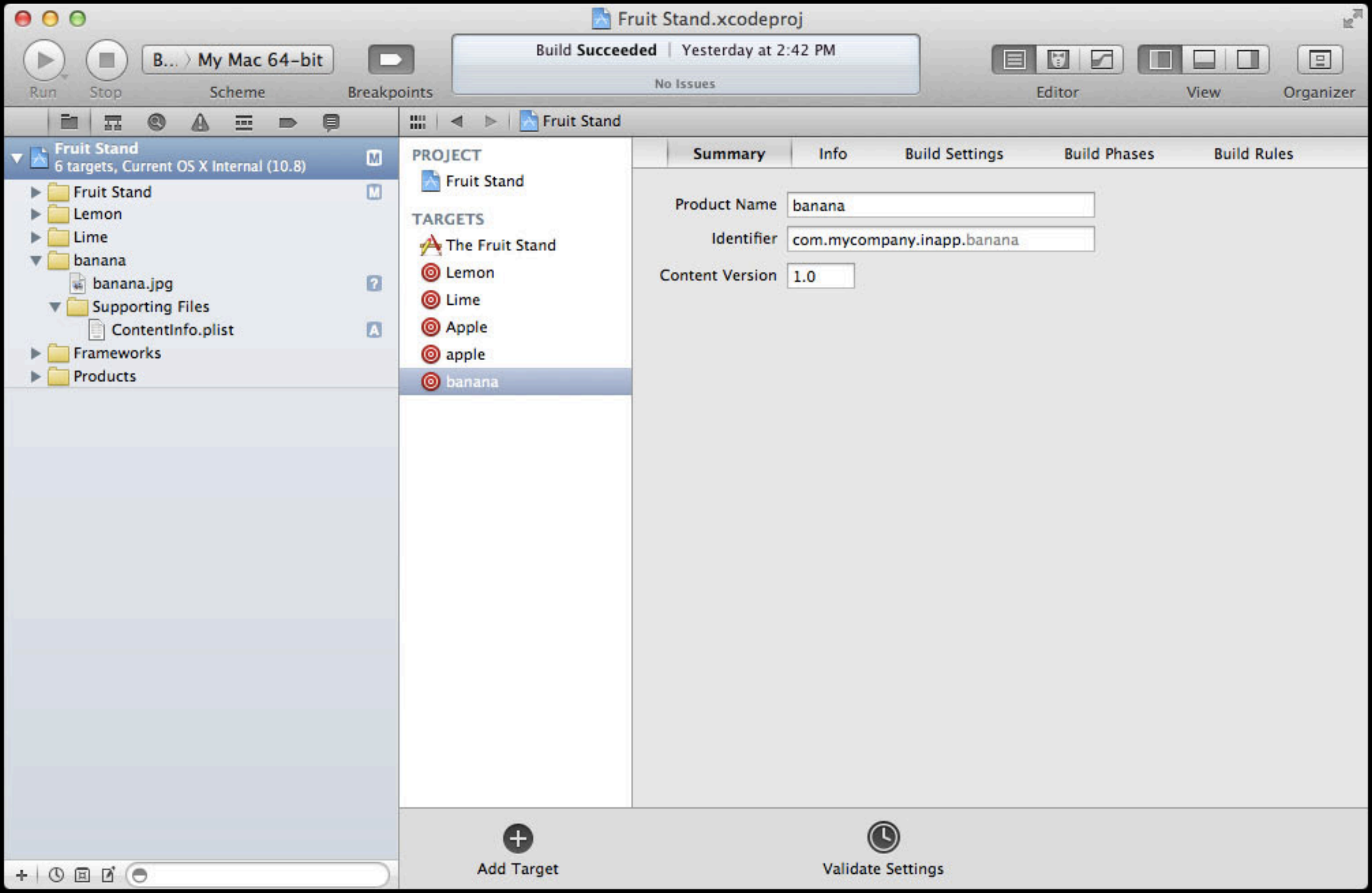
In-App Purchase Content

This template builds an In-App Purchase Content package.

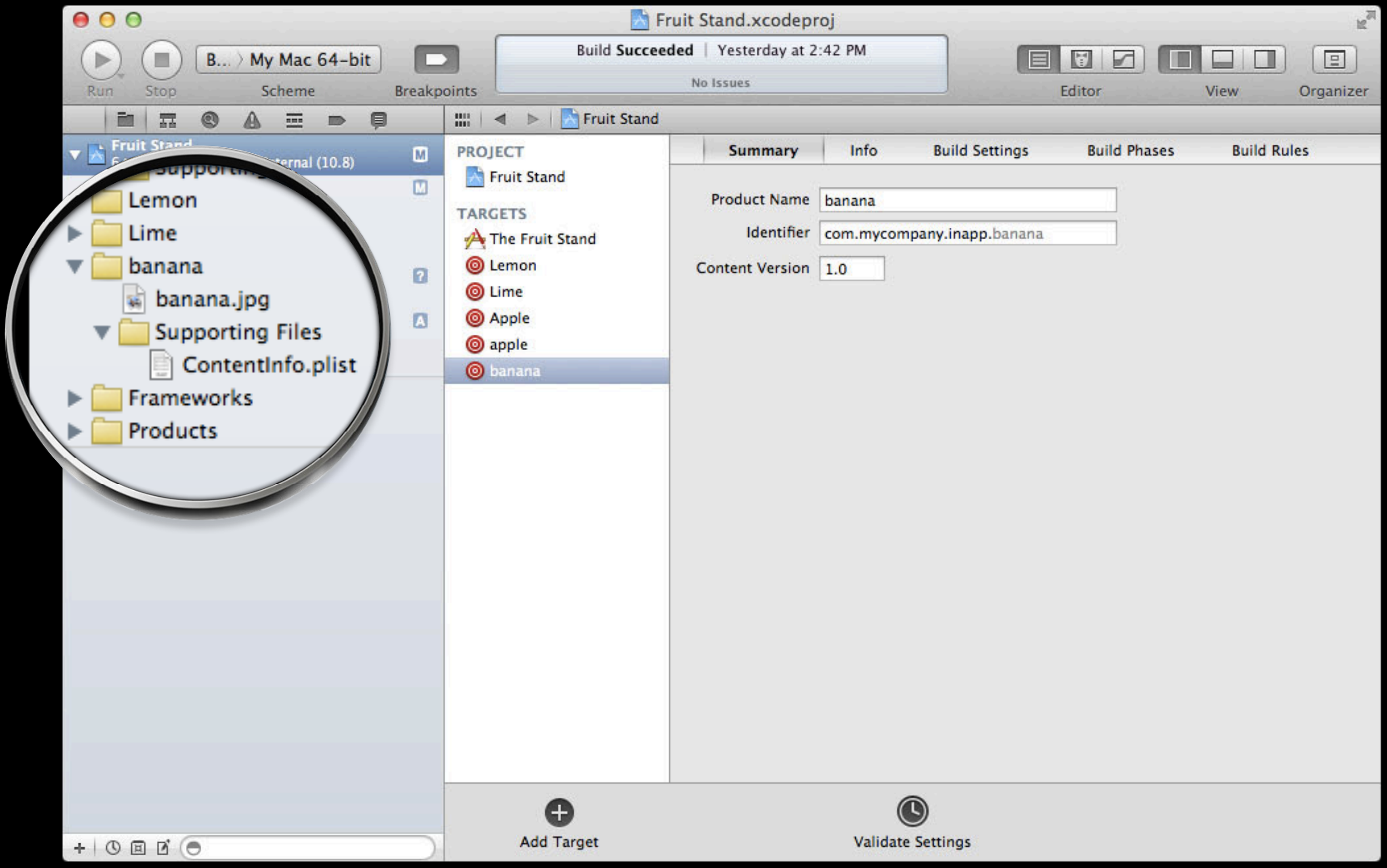


Choose options for your new target:

Product Name	banana
Organization Name	My Company
Company Identifier	com.mycompany.inapp
Bundle Identifier	com.mycompany.inapp.banana
Project	Fruit Stand

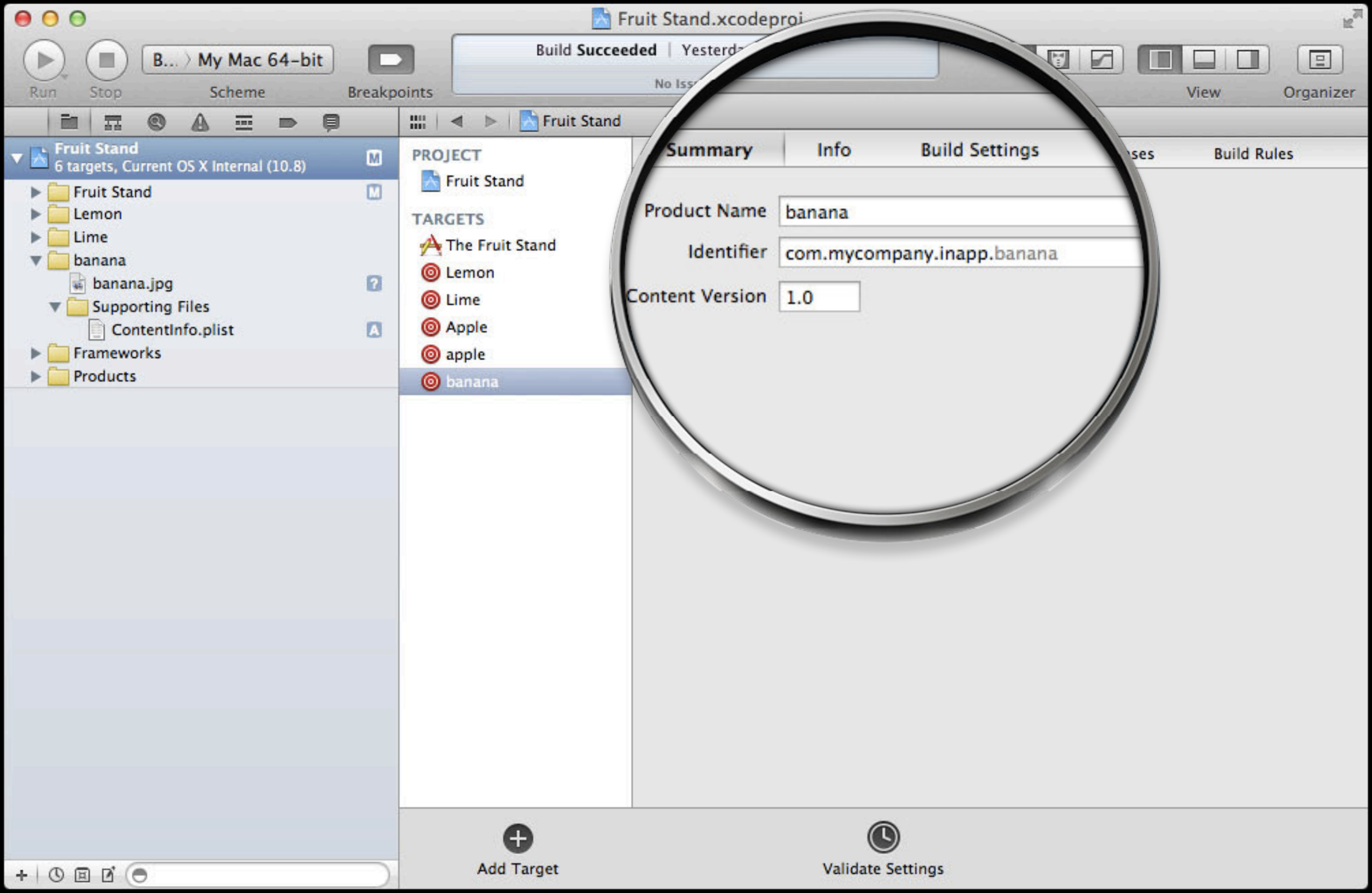


Product Name	banana
Identifier	com.mycompany.inapp.banana
Content Version	1.0

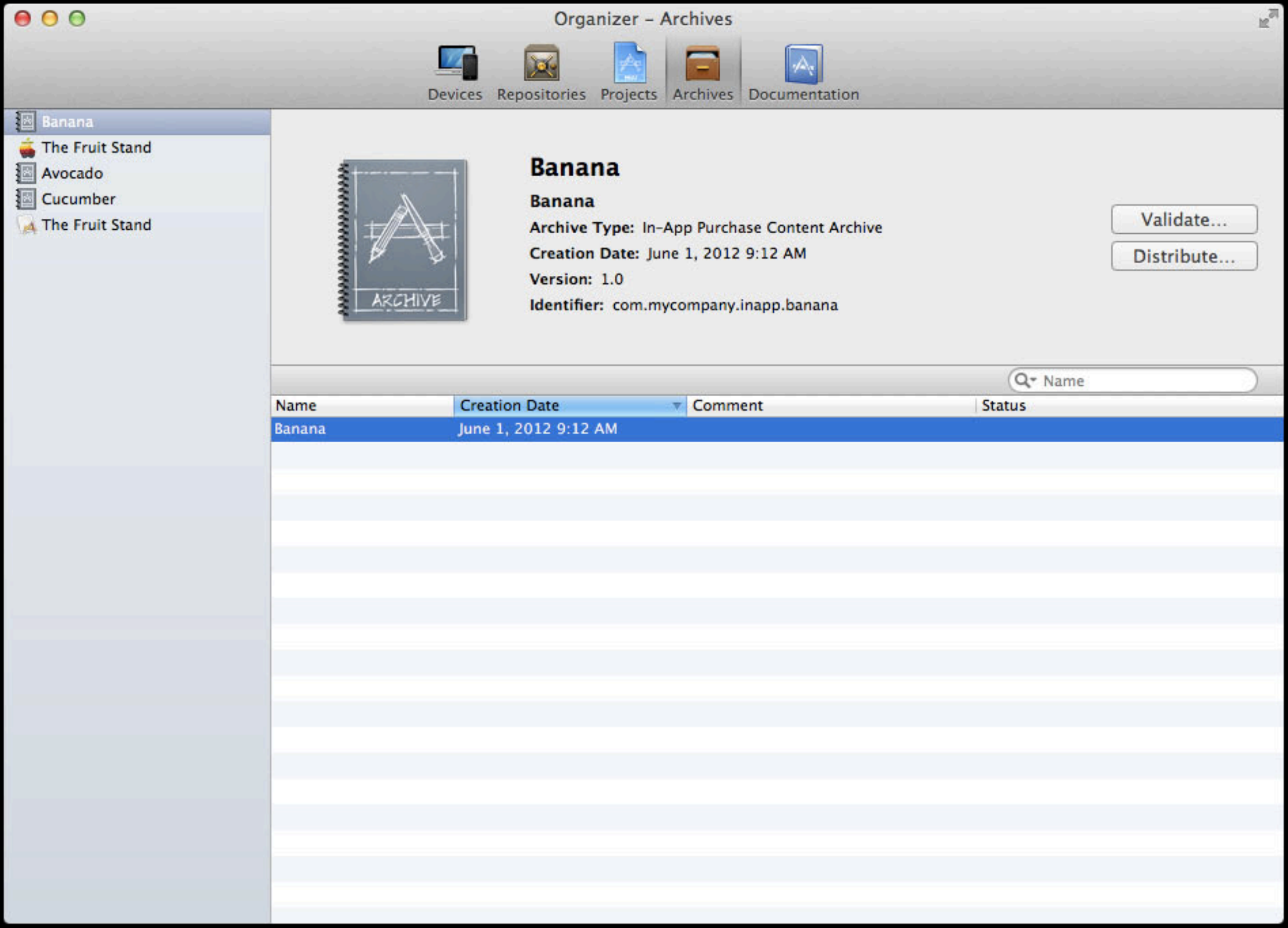


Build Succeeded | Yesterday at 2:42 PM
No Issues

Product Name: banana
Identifier: com.mycompany.inapp.banana
Content Version: 1.0



Product Name: banana
Identifier: com.mycompany.inapp.banana
Content Version: 1.0



Banana

Banana

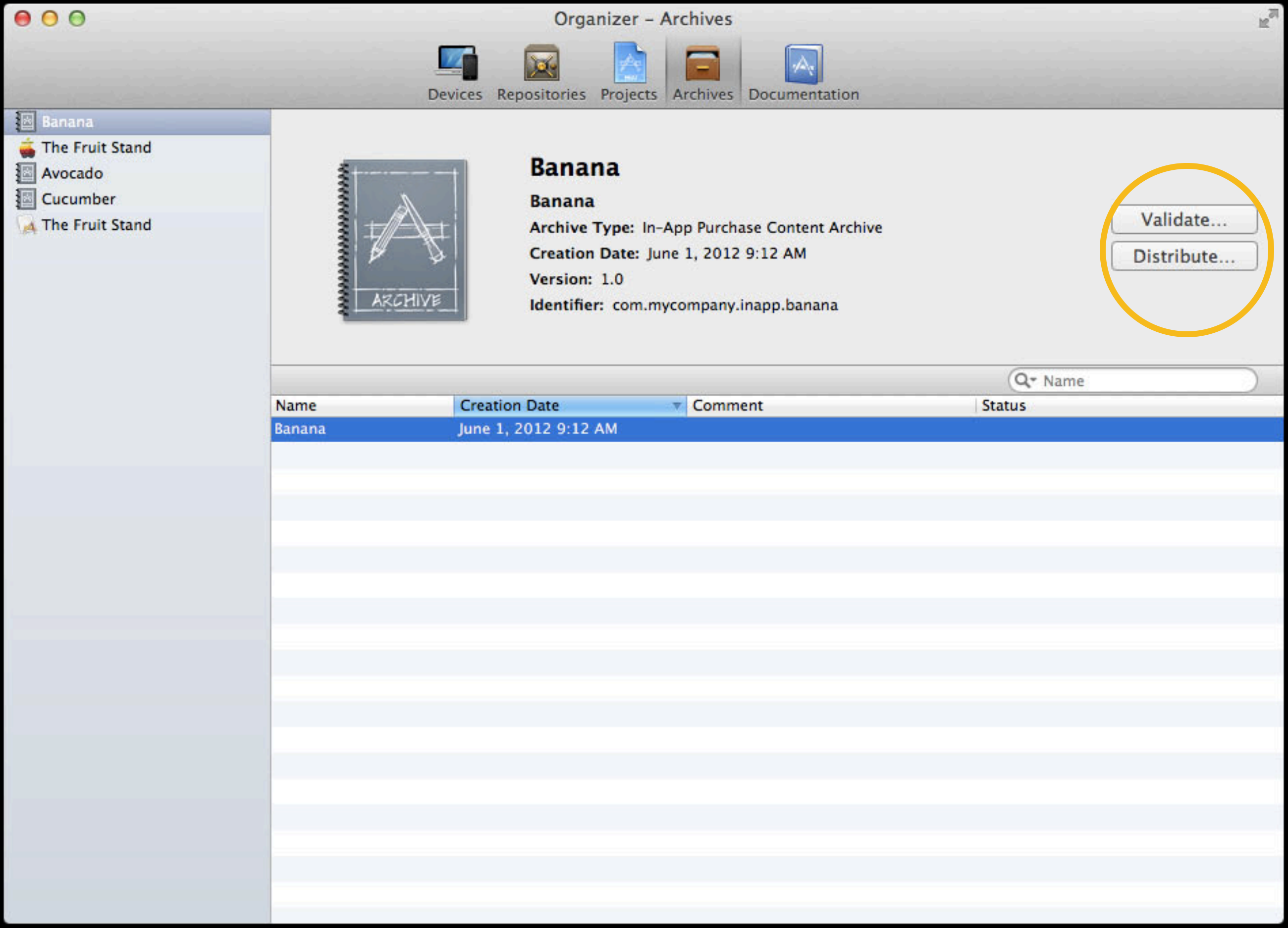
Archive Type: In-App Purchase Content Archive

Creation Date: June 1, 2012 9:12 AM

Version: 1.0

Identifier: com.mycompany.inapp.banana

Name	Creation Date	Comment	Status
Banana	June 1, 2012 9:12 AM		



Banana

Banana

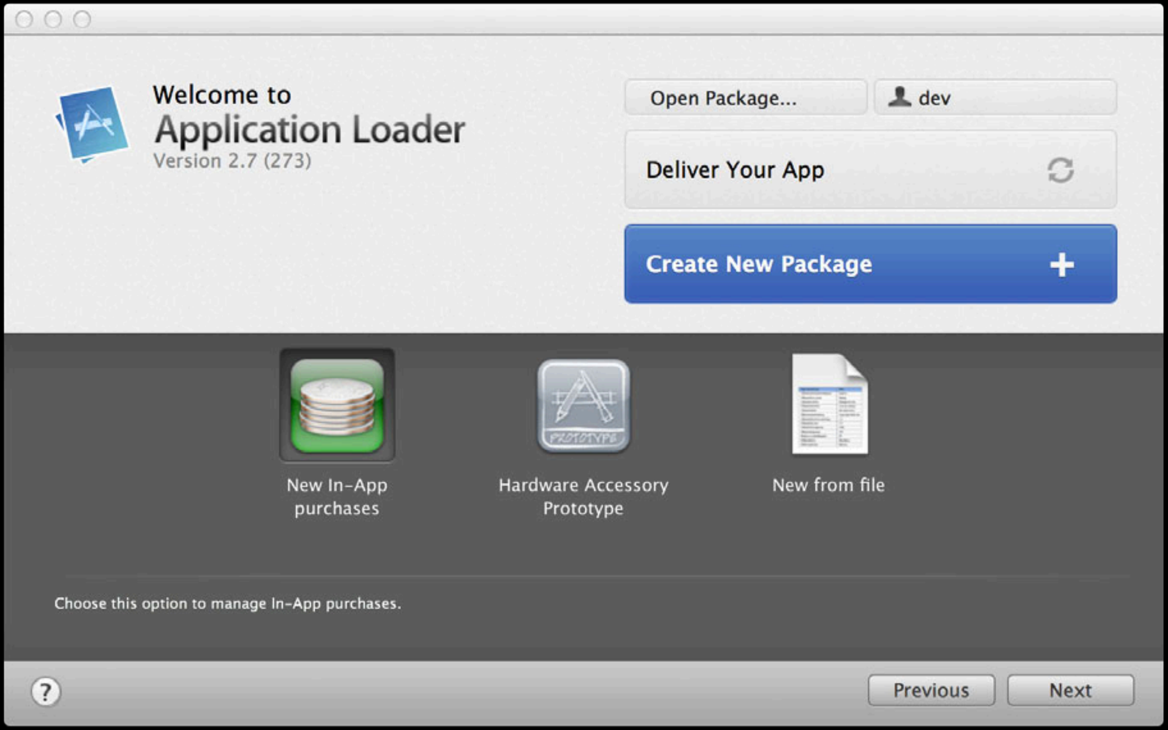
Archive Type: In-App Purchase Content Archive

Creation Date: June 1, 2012 9:12 AM

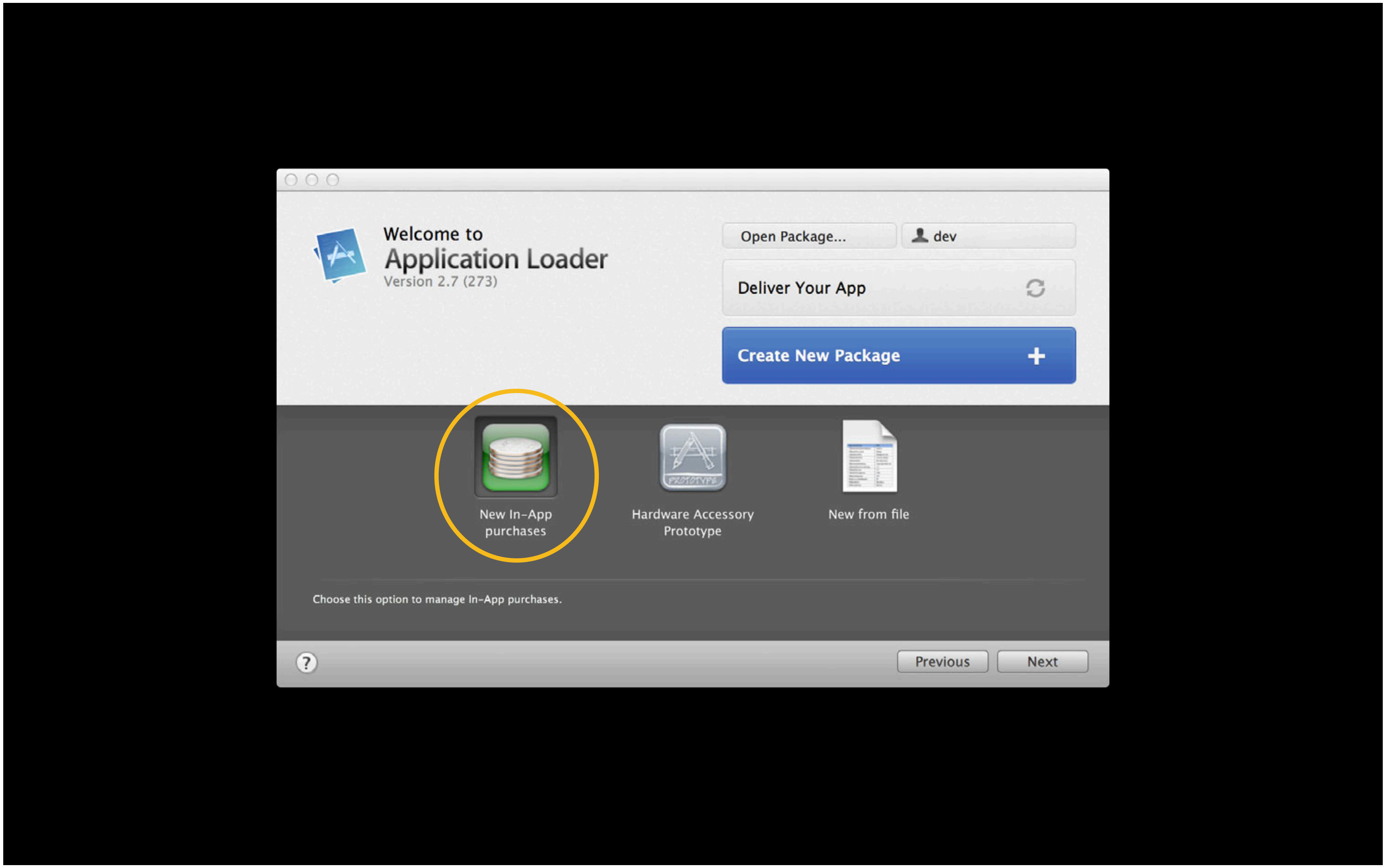
Version: 1.0

Identifier: com.mycompany.inapp.banana

Name	Creation Date	Comment	Status
Banana	June 1, 2012 9:12 AM		



Choose this option to manage In-App purchases.



Choose this option to manage In-App purchases.

App Store Hosted Content

Where does it get installed?

- On iOS

App Store Hosted Content

Where does it get installed?

- On iOS

	Purgeable	Stays Local	Backed Up
Caches	✓		

App Store Hosted Content

Where does it get installed?

- On iOS

	Purgeable	Stays Local	Backed Up
Caches	✓		
Documents		✓	

App Store Hosted Content

Where does it get installed?

- On iOS

	Purgeable	Stays Local	Backed Up
Caches	✓		
Documents		✓	
Documents (with backup flag set)		✓	✓

App Store Hosted Content

Where does it get installed?

App Store Hosted Content

Where does it get installed?

- On OS X

App Store Hosted Content

Where does it get installed?

- On OS X
    - Special Application Support folder

App Store Hosted Content

Where does it get installed?

- On OS X
  - Special Application Support folder
  - Use API to access it

App Store Hosted Content

Where does it get installed?

- On OS X
  - Special Application Support folder
  - Use API to access it

```
+ (NSURL *) contentURLForProductID:(NSString *)productID;
```

App Store Hosted Content

Where does it get installed?

- On OS X
  - Special Application Support folder
  - Use API to access it

```
+ (NSURL *) contentURLForProductID:(NSString *)productID;

+ (void) deleteContentForProductID:(NSString *)productID;
```

App Store Hosted Content

Updating content

App Store Hosted Content

Updating content

- Edit your content

App Store Hosted Content

Updating content

- Edit your content
- Update version in `ContentInfo.plist`

App Store Hosted Content

Updating content

- Edit your content
- Update version in `ContentInfo.plist`
- Re-upload to iTunes Connect

App Store Hosted Content

Updating content

- Edit your content
- Update version in `ContentInfo.plist`
- Re-upload to iTunes Connect
- Requires restore to get new content

App Store Hosted Content
Updating content

- Edit your content
- Update version in `ContentInfo.plist`
- Re-upload to iTunes Connect
- Requires restore to get new content
- To determine if something has changed

App Store Hosted Content

Updating content

- Edit your content
- Update version in `ContentInfo.plist`
- Re-upload to iTunes Connect
- Requires restore to get new content
- To determine if something has changed
  - Fetch `SKProducts`

App Store Hosted Content

Updating content

- Edit your content
- Update version in `ContentInfo.plist`
- Re-upload to iTunes Connect
- Requires restore to get new content
- To determine if something has changed
  - Fetch `SKProducts`
  - Compare to `ContentInfo.plist`

App Store Hosted Content

Transitioning from self-hosted content

App Store Hosted Content

Transitioning from self-hosted content

- Must be added as new products in iTunes Connect

Demo

Today's Agenda

- Selling Store Content
- Using In-App Purchase
- In Detail: The Purchase Queue
- App Store Hosted Content
- Best Practices

Best Practices

Best Practices

- Check queue on launch

Best Practices

- Check queue on launch
- Call `finishTransaction:`

Best Practices

- Check queue on launch
- Call `finishTransaction:`
- Restoring purchases is required

Best Practices

- Check queue on launch
- Call `finishTransaction:`
- Restoring purchases is required
- iOS: Move out of `~/Caches` if you want it to persist

Best Practices

- Check queue on launch
- Call `finishTransaction:`
- Restoring purchases is required
- iOS: Move out of `~/Caches` if you want it to persist
- Test in sandbox before deploying

More Information

Paul Marcos
Application Services Evangelist
pmarcos@apple.com

Documentation
In-App Purchase Programming Guide and Validating App Store Receipts
http://developer.apple.com

Search API and Enterprise Partner Feed
http://www.apple.com/itunes/affiliates

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

Related Sessions

What's New in iTunes Connect for App Developers	Nob Hill Thursday 9:00AM
Building Great Newsstand Apps	Nob Hill Thursday 2:00PM
Managing Subscriptions with In-App Purchase	Mission Thursday 3:15PM

Labs

In-App Purchase Lab	App Services Lab A Wednesday 3:15PM
iTunes Connect for App Developers Lab	App Services Lab A Thursday 11:30AM
In-App Purchase Lab	App Services Lab B Thursday 4:30PM
Newsstand Lab	App Services Lab A Friday 9:00AM
App Store Lab	App Store Lab (Level 3) Monday-Friday 9:00AM

WWDC2012

The last 3 slides after the logo are intentionally left blank for all presentations.

The last 3 slides after the logo are intentionally left blank for all presentations.

The last 3 slides after the logo are intentionally left blank for all presentations.