

Presented by Max Stottrop

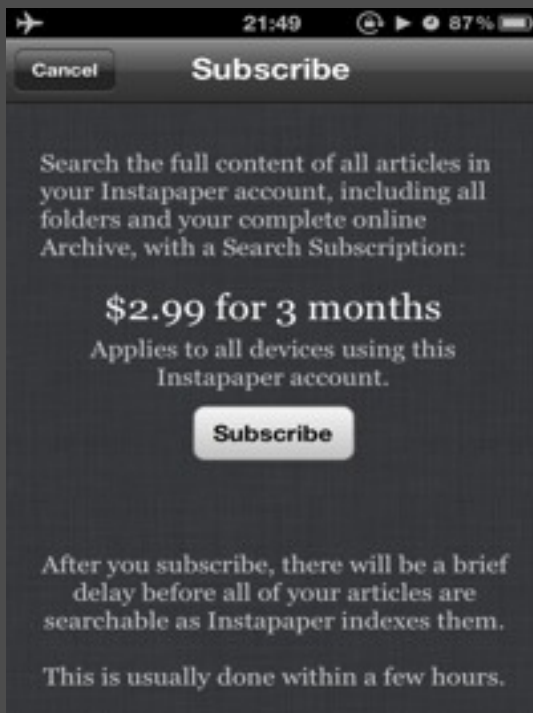
IN-APP PURCHASE

What I will cover today

- Introduction
- Process of implementing In-App Purchase
- StoreKit Framework
- Business & Tips

With In-App Purchase you can...

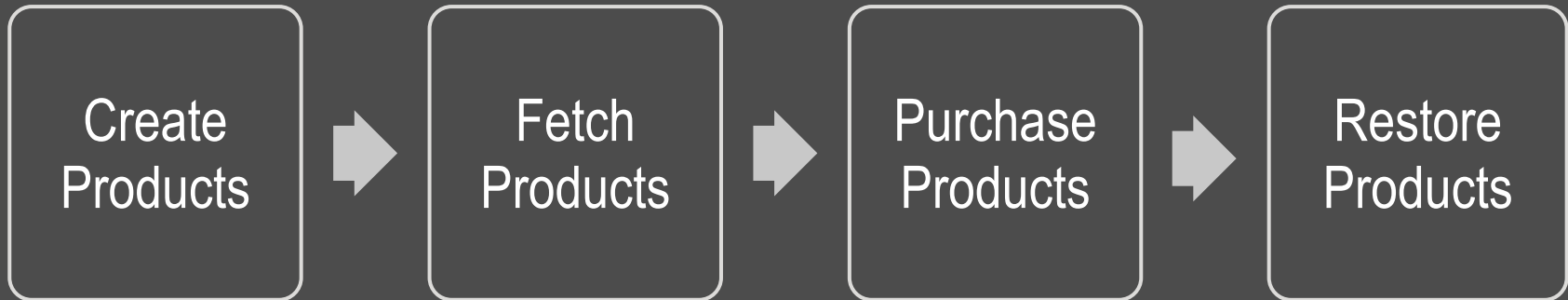
- ... embed a store directly to your application
- It uses the Storekit Framework.



In-App Purchase Types

- Non-Consumable
- Consumable
- Auto-renewable subscription
- Free subscription
- Non-renewable subscription

Walkthrough



Implementing In-App Purchase



Products

- Supported types:
 - Content
 - Functionality
 - Services
 - Subscriptions
- Each product has a unique productIdentifier.
- No real-world goods!

iTunes Connect Setup

- Setup your Products
- Create a Test User
 - One per country
- Sign out of your Store Settings
 - Don't enter your Test User's Ids in the Settings

Store Kit Framework

Validate In-App Purchase Access

Retrieving product information

Show the Store Interface

Make the purchase

Process Transaction

Make the feature available

Finish Transaction

Verify the receipts

Restore previous Purchases

Validate In-App Purchase Acces

- Validate In-App Purchase Acces
- Payment Queue class method `canMakePurchase`

Determine sellable products

- Load Product IDs
- Use SKProductsRequest to determine the sellable subset of the IDs

```
NSSet* products = [NSSet setWithObject: @"productID1", @"productID2", nil];
SKProductsRequest *request= [[SKProductsRequest alloc]
initWithProductIdentifiers: products];
[productRequest start];
```

```
(void)productsRequest:(SKProductsRequest *)request didReceiveResponse:
(SKProductsResponse *)response {
    response.invalidProductIdentifiers == nonsellable;
    response.products == sellable;
```

Show Store

- UITableView Power?

```
(void)productsRequest:(SKProductsRequest *)request didReceiveResponse:  
(SKProductsResponse *)response {  
    for(SKProducts* aProduct in response.products)  
        cell.textLabel.text = aProduct.localizedTitle;  
        cell.textLabel.description = aProduct.localizedDescription;  
        cell.textLabel.extra = aProduct.price;  
}
```

Request payment

- SKPayment Class
 - Create payment object
 - Add it to the queue
 - Observe the payment queue

```
SKPayment *payRequest = [SKPayment paymentWithObject:selected];
```

```
[[SKPaymentQueue defaultQueue] addPayment:payRequest];
```

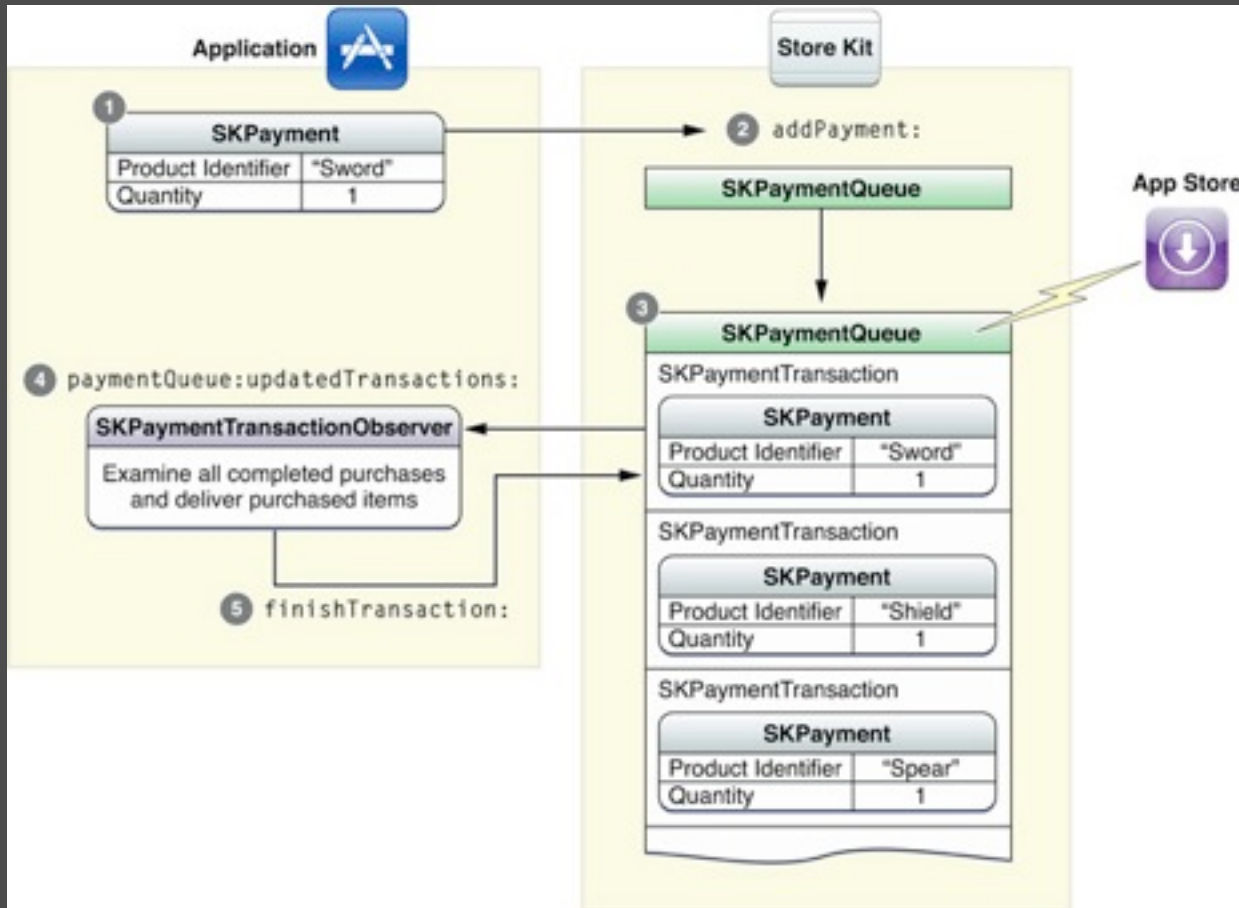
```
- (void)addTransactionObserver:(id <SKPaymentTransactionObserver>) observer
```

Complete Transaction

- updatedTransaction method
 - SKPaymentTransactionStatePurchased = Success
 - SKPaymentTransactionStateRestored = Success
- finishTransaction method

```
-(void)paymentQueue:(SKPaymentQueue *)queue updatedTransactions:(NSArray *)  
transactions {  
    if (a.transaction.transactionState == SKPaymentTransactionStatePurchased ) {  
        //Unlock content  
        [queue finishTransaction:a.transaction]; }  
}
```

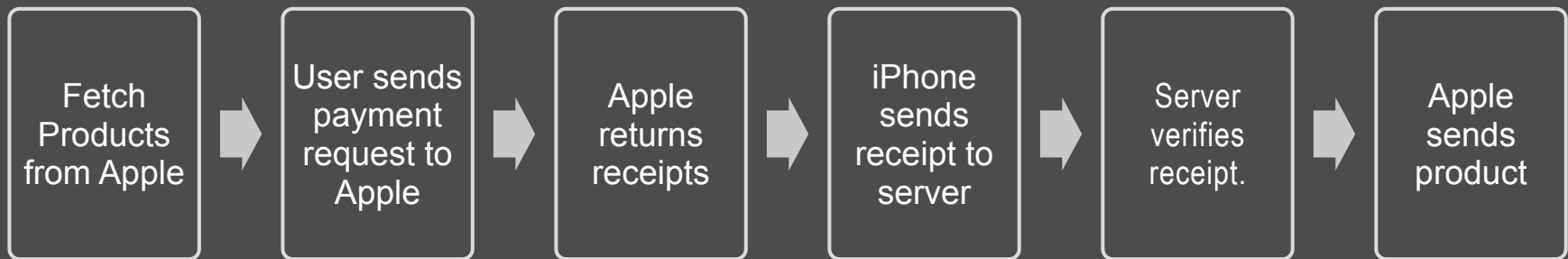
Collecting Payments



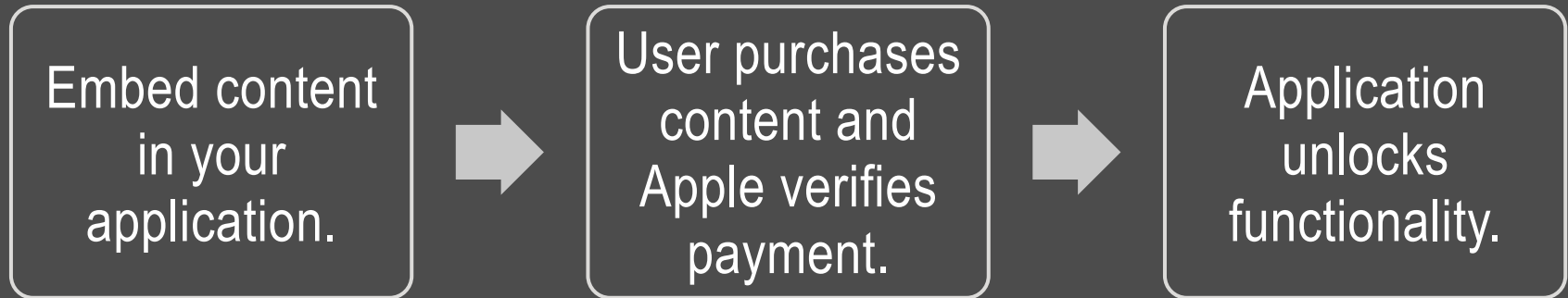
Feature Delivery

- Self contained
- Downloadable

Downloadable Content



In-App Content



Remember Product Purchase

- Use NSUserDefaults class and preference file
 - unsecure, easy to „hack“ .
 - saves to Application Support directory
- Use Keychain API
 - Secure
 - Saved to app Keychain Slice

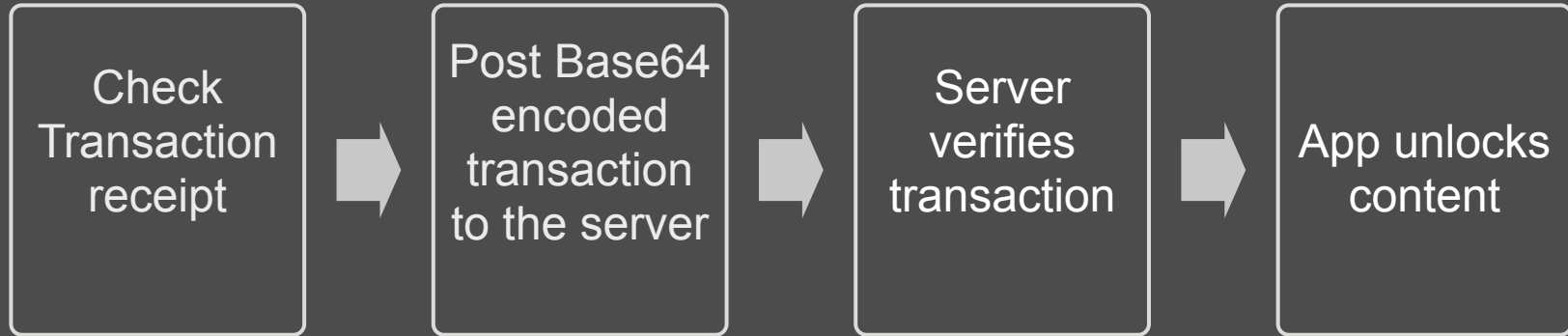
```
[[NSUserDefaults standardUserDefaults] setObject:bought forKey:@"boughtItems"];
```

```
SecItemAdd ((CFDictionaryRef) boughtItems, NULL);
```

Verify Product Purchase



Verify Product Purchase

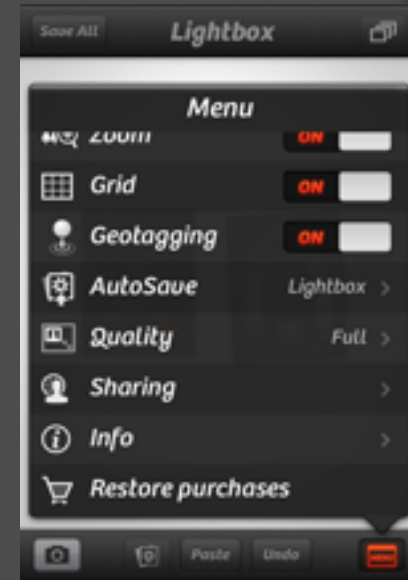


Verify Product Purchase

- Send JSON object with encoded transaction receipt
 - {"receipt - data" : "7b4187e54587ert27e2"}
- Post to iTunes verification service
 - <http://buy.itunes.apple.com/verifyReceipt>
 - <http://sandbox.itunes.apple.com/verifyReceipt>
- Get a JSON object
 - {"status" : 0, "receipt" : { ... } }
 - 0 == Succes,
 - everything else == BAD PIRATES AT WORK


Restore Previous Purchases

- Free Subscription, Auto-Renewing Subscription, Non-Consumable items
- `[[SKPaymentQueue defaultQueue] restoreCompletedTransactions]`
- Subscription & Consumable need own mechanism



Business Model

- Which Model to choose?
- What to content sell?
 - Game
 - Everything else



Extra Levels
Social content
In-game Currency
Optional Content
Money for time/Time

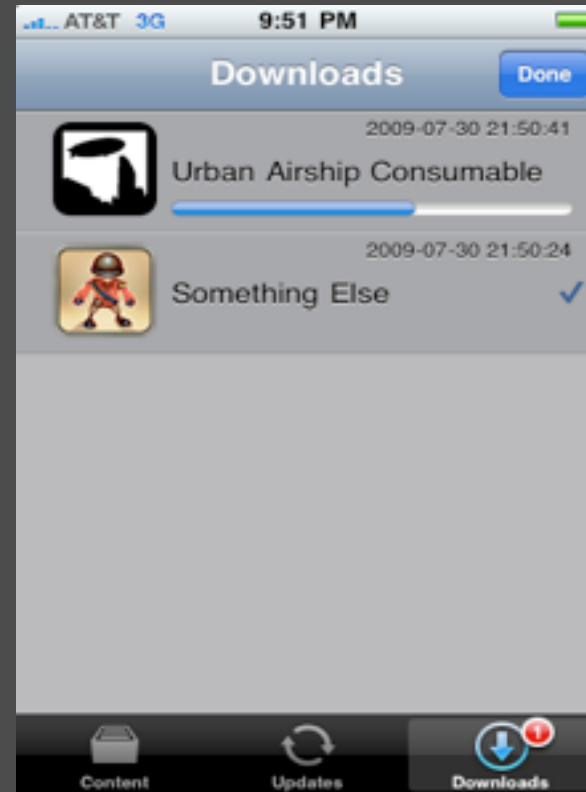
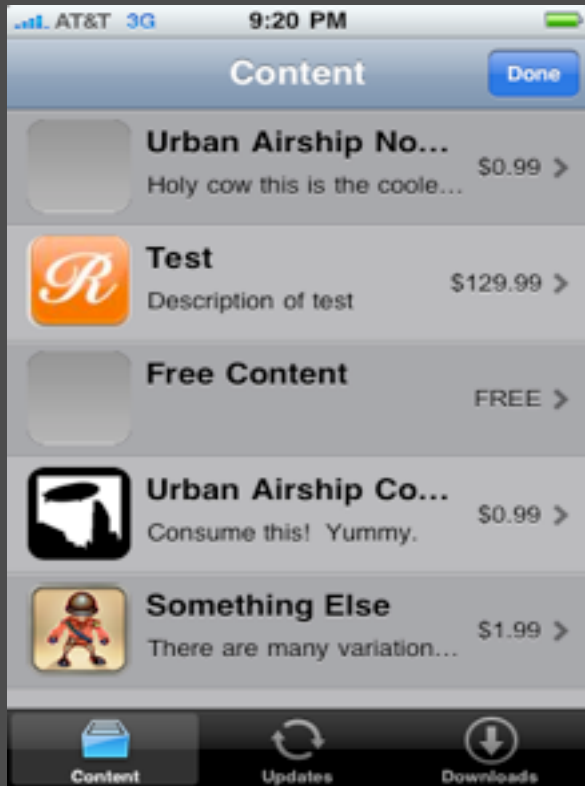
Tips

- Product Ids are unique across apps!
- Built Store with sellable products only!
- Use MBBase64 category from the CocoaDev website
- Auto-renewing Subscription only for apps with dynamic content.
- Only show the button when Internet Connection is available.
- To test In-App Purchase you must add the Binary - Reject it!
- After you added your Products, wait a bit!

- MKStoreKit: Open Source helper.

ONE LAST THING

StoreFront



StoreFront

- Works via JSON to display Content, Downloads, Updates.
- Allows to sell free content.
- No additional server costs/bandwidth.
- Restoring Auto-renewable Subscriptions - YAY

Q & A

THANKS!

Links

- Receipt validation: <http://bit.ly/zMuTKk>, <http://bit.ly/we2Pt0> & <http://bit.ly/AB09Sv>
- Apple: <http://bit.ly/xy7aGV>, <http://bit.ly/wyLec6> & <http://bit.ly/weMm3p>
- MKStoreKit: <https://github.com/MugunthKumar/MKStoreKit>
- StoreFront: http://urbanairship.com/docs/inapp_client.html
- Dr. Touch's Purchase Button: <http://bit.ly/zxGjcY>