Advanced StoreKit

Receipt validation and subscriptions

Pete Hare, App Store Engineer

Agenda

Receipt validation

Maintaining subscription state

Developing with the Sandbox

Agenda

Receipt validation

Maintaining subscription state

Developing with the Sandbox

Agenda

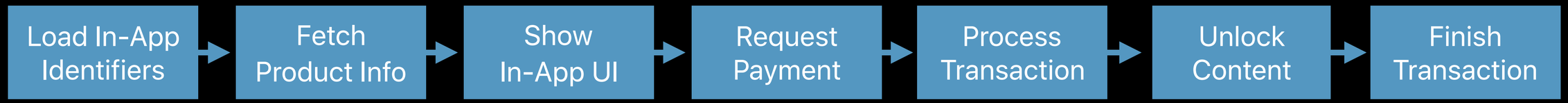
Receipt validation

Maintaining subscription state

Developing with the Sandbox

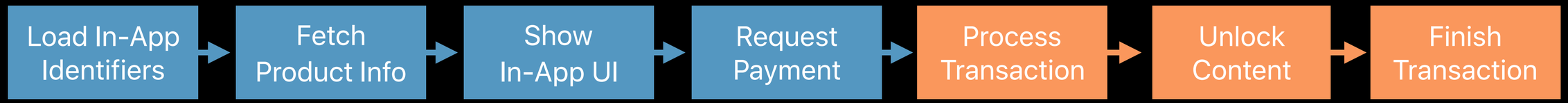
In-App Purchase Process

Processing transactions



In-App Purchase Process

Processing transactions

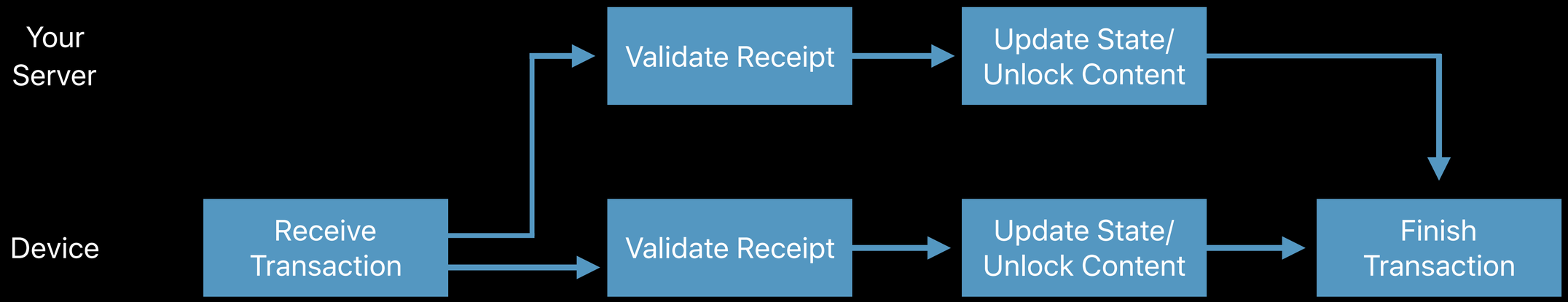


In-App Purchase Process

Processing transactions

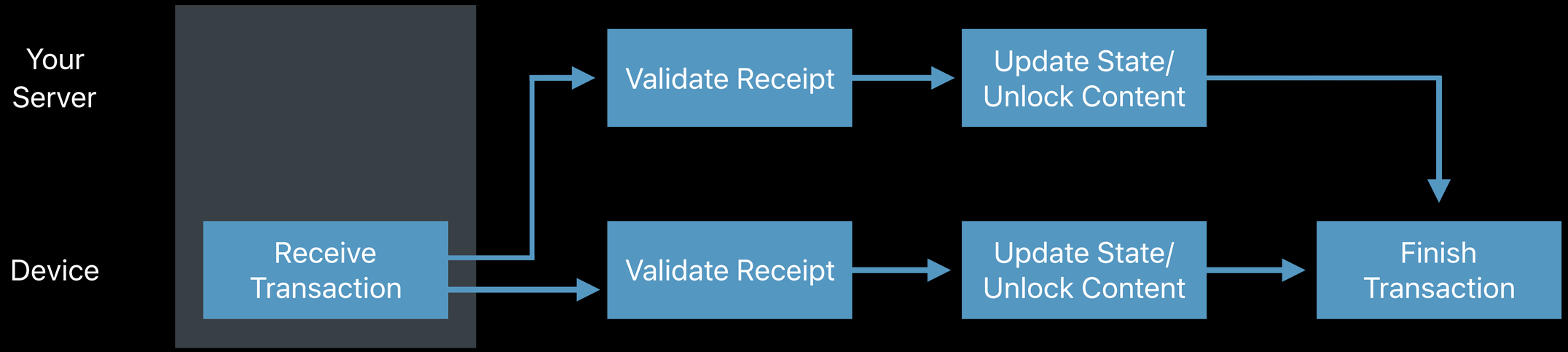
In-App Purchase Process

Processing transactions



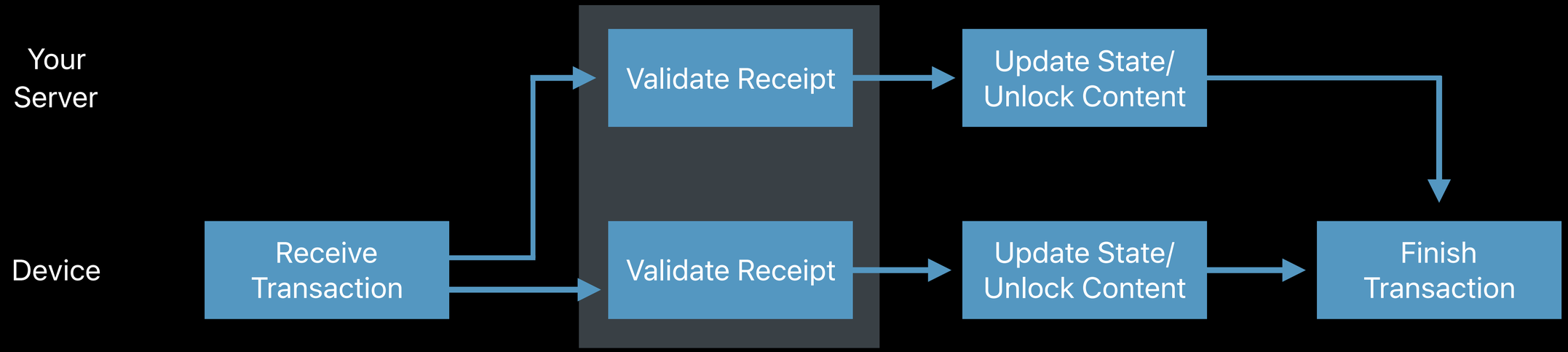
In-App Purchase Process

Processing transactions



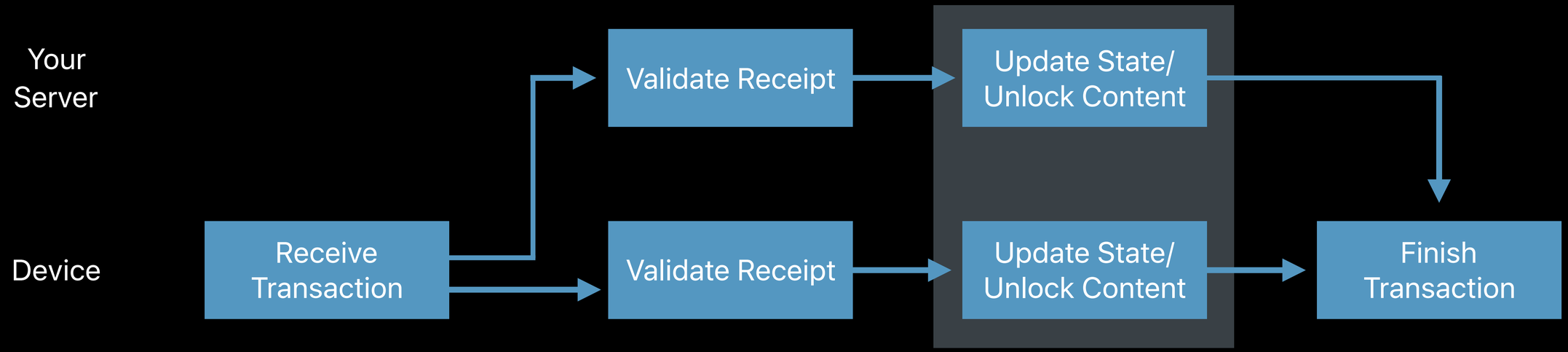
In-App Purchase Process

Processing transactions



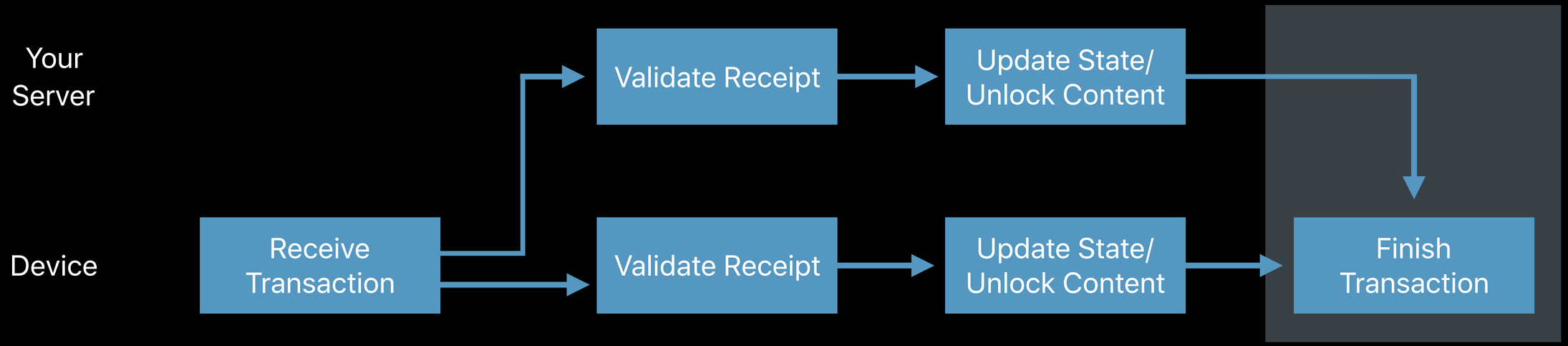
In-App Purchase Process

Processing transactions



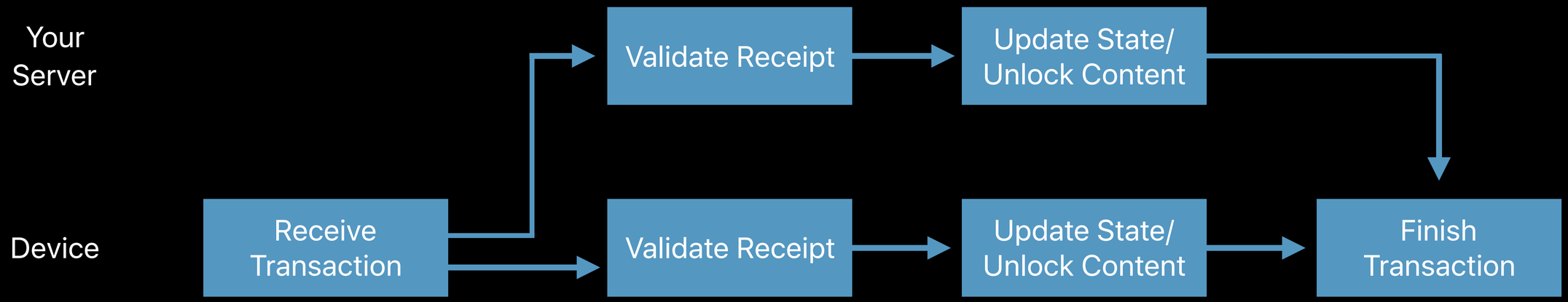
In-App Purchase Process

Processing transactions



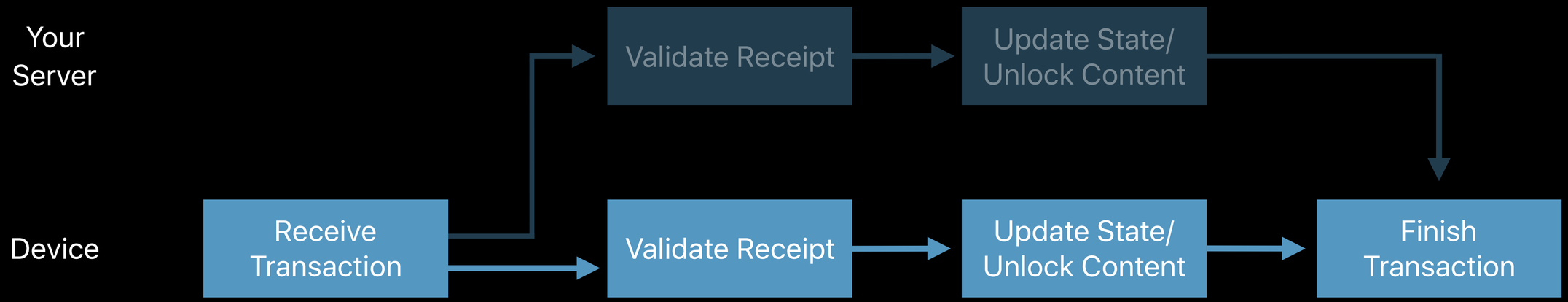
In-App Purchase Process

Processing transactions



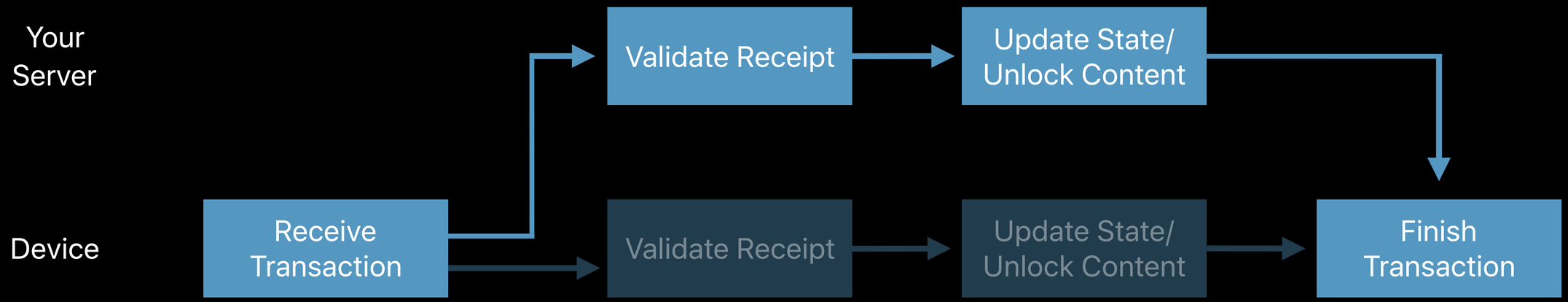
In-App Purchase Process

Processing transactions



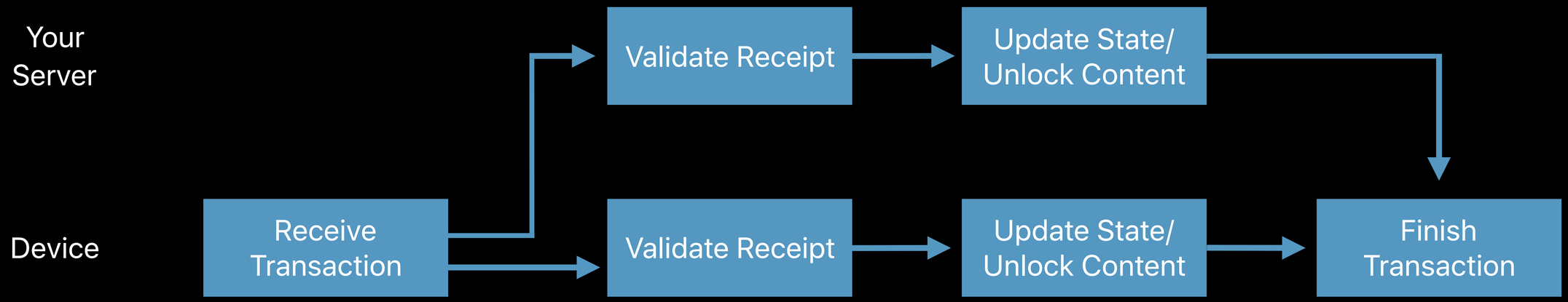
In-App Purchase Process

Processing transactions



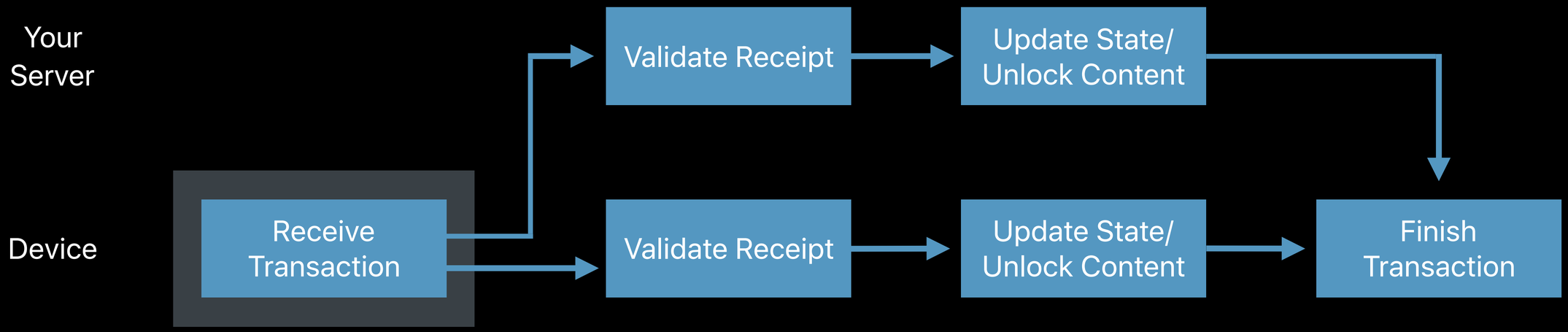
In-App Purchase Process

Processing transactions



In-App Purchase Process

Processing transactions



```
// Start Observing the Payment Queue

import UIKit
import StoreKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate, SKPaymentTransactionObserver {

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

```
// Start Observing the Payment Queue

import UIKit
import StoreKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate, SKPaymentTransactionObserver {

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

```
// Start Observing the Payment Queue

import UIKit
import StoreKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate, SKPaymentTransactionObserver {

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

```swift
class AppDelegate: UIResponder, UIApplicationDelegate, SKPaymentTransactionObserver {

    // …

    // MARK: - SKPaymentTransactionObserver

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

```
class AppDelegate: UIResponder, UIApplicationDelegate, SKPaymentTransactionObserver {

    // …

    // MARK: - SKPaymentTransactionObserver

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

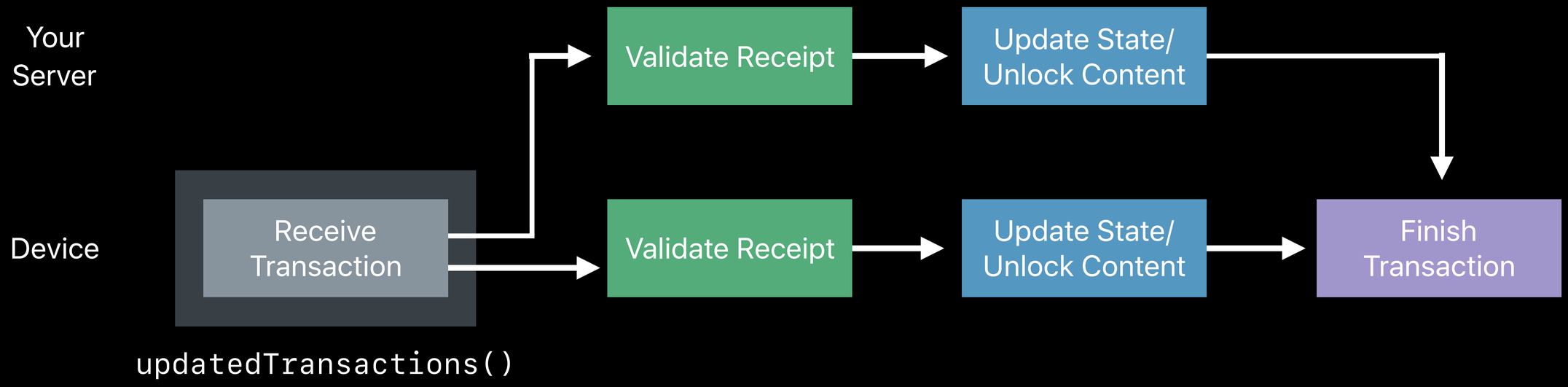
```
class AppDelegate: UIResponder, UIApplicationDelegate, SKPaymentTransactionObserver {

    // …

    // MARK: - SKPaymentTransactionObserver

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

```
class AppDelegate: UIResponder, UIApplicationDelegate, SKPaymentTransactionObserver {

    // …

    // MARK: - SKPaymentTransactionObserver

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

```swift
class AppDelegate: UIResponder, UIApplicationDelegate, SKPaymentTransactionObserver {

    // …

    // MARK: - SKPaymentTransactionObserver

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

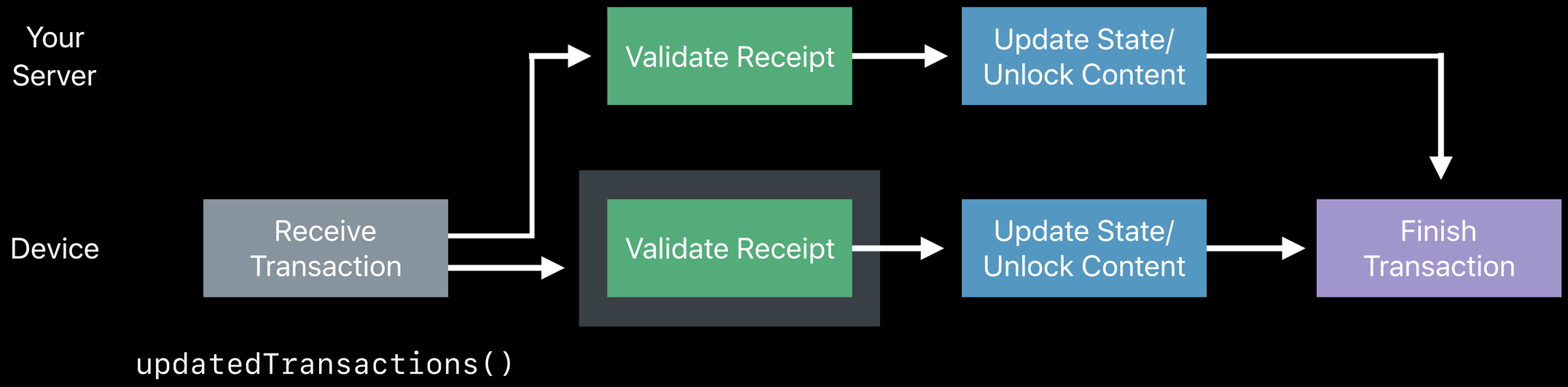
In-App Purchase Process

Processing transactions



In-App Purchase Process

Processing transactions



The Receipt

The Receipt

The Receipt

Trusted record of app and in-app purchases

The Receipt

Trusted record of app and in-app purchases

Stored on device

The Receipt

Trusted record of app and in-app purchases

Stored on device

Issued by the App Store

The Receipt

Trusted record of app and in-app purchases

Stored on device

Issued by the App Store

Signed and verifiable

The Receipt

Trusted record of app and in-app purchases

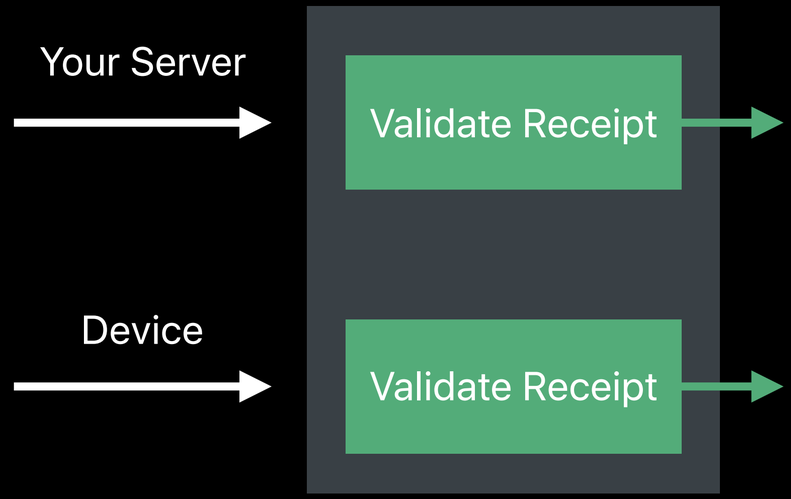
Stored on device

Issued by the App Store

Signed and verifiable

For your app, on that device only

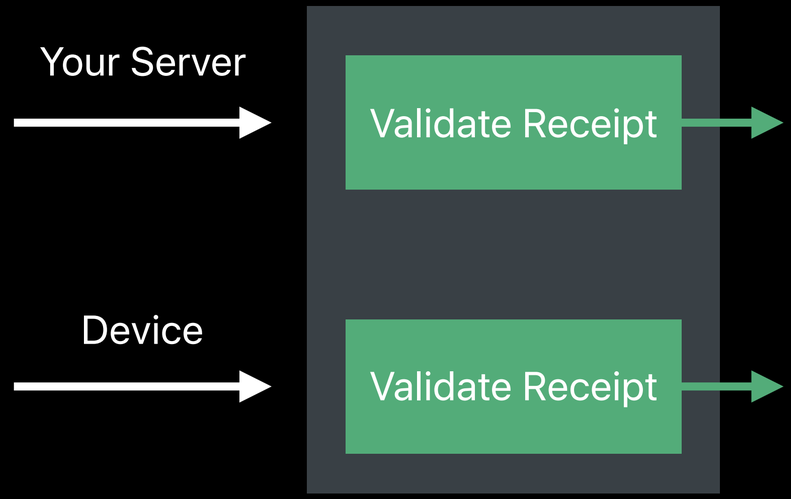
Receipt Validation



Receipt Validation

On-device validation

- Unlock features and content within the app



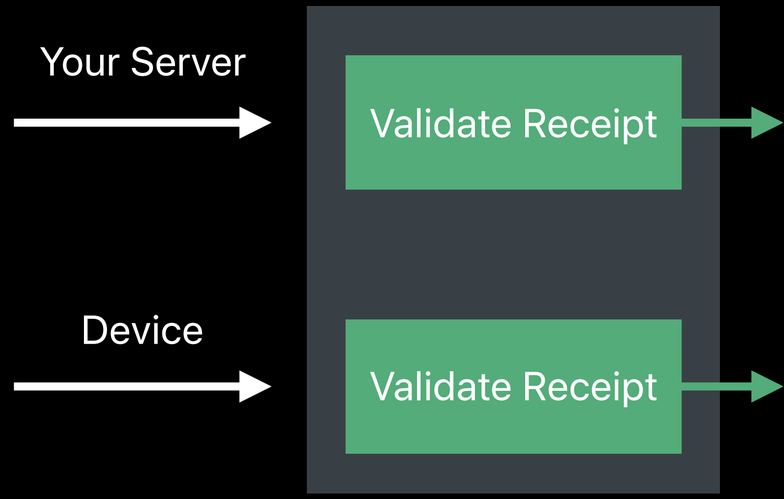
Receipt Validation

On-device validation

- Unlock features and content within the app

Server-to-server validation

- Online validation through a request to the App Store
- Unlock features/subscription state on your server



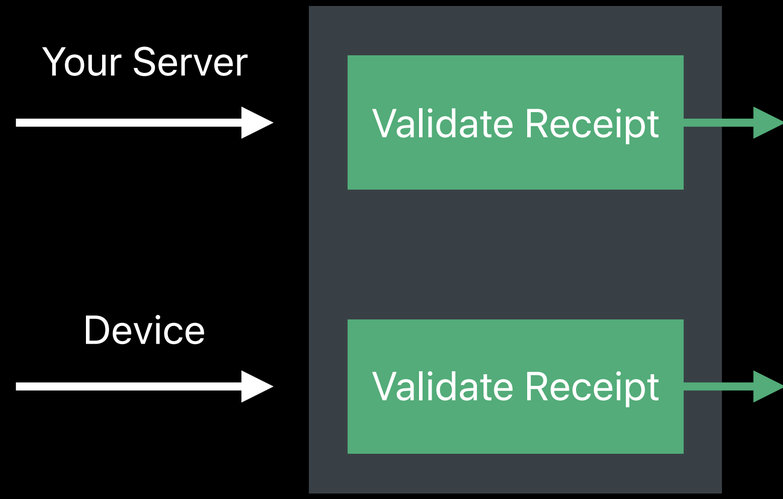
Receipt Validation

On-device validation

- Unlock features and content within the app

Server-to-server validation

- Online validation through a request to the App Store
- Unlock features/subscription state on your server



Do not use online validation directly from the device!

On-Device Receipt Validation

The basics

On-Device Receipt Validation

The basics

Stored in the app bundle

On-Device Receipt Validation

The basics

Stored in the app bundle

- API to get the path

On-Device Receipt Validation

The basics

Stored in the app bundle

- API to get the path

Single file

On-Device Receipt Validation

The basics

Stored in the app bundle

- API to get the path

Single file

- Purchase data

On-Device Receipt Validation

The basics

Stored in the app bundle

- API to get the path

Single file

- Purchase data
- Signature to check authenticity

On-Device Receipt Validation

Standards

On-Device Receipt Validation

Standards

Signing

- PKCS#7 Cryptographic Container

On-Device Receipt Validation

Standards

Signing

- PKCS#7 Cryptographic Container

Data Encoding

- ASN.1

On-Device Receipt Validation

Standards

Signing

- PKCS#7 Cryptographic Container

Data Encoding

- ASN.1

Options for verifying and reading

- OpenSSL, asn1c, and more
- Create your own

On-Device Receipt Validation

The basics

Locate the receipt using Bundle API

```
// Locate the file
guard let url = Bundle.main.appStoreReceiptURL else {
    // Handle failure
    return
}
// Read the contents
let receipt = Data(contentsOf: url)
```

On-Device Receipt Validation

The basics

Locate the receipt using Bundle API

```
// Locate the file
guard let url = Bundle.main.appStoreReceiptURL else {
    // Handle failure
    return
}
// Read the contents
let receipt = Data(contentsOf: url)
```

On-Device Receipt Validation

The basics

Locate the receipt using Bundle API

```
// Locate the file
guard let url = Bundle.main.appStoreReceiptURL else {
    // Handle failure
    return
}
// Read the contents
let receipt = Data(contentsOf: url)
```

On-Device Receipt Validation

The basics

Locate the receipt using Bundle API

```
// Locate the file
guard let url = Bundle.main.appStoreReceiptURL else {
    // Handle failure
    return
}
// Read the contents
let receipt = Data(contentsOf: url)
```

On-Device Receipt Validation

Tips for using OpenSSL

On-Device Receipt Validation

Tips for using OpenSSL

Build your own static library (.a file)

- Not a dynamic library

On-Device Receipt Validation

Tips for using OpenSSL

Build your own static library (.a file)

- Not a dynamic library

Include Apple Root CA Certificate

- Available online
- If bundled in app, watch out for expiry

On-Device Receipt Validation

Tips for using OpenSSL

Build your own static library (.a file)

- Not a dynamic library

Include Apple Root CA Certificate

- Available online
- If bundled in app, watch out for expiry

Documentation online

On-Device Receipt Validation

Downloading pre-built solutions

On-Device Receipt Validation

Downloading pre-built solutions

Convenience comes at a price

- Reusing code brings with it bugs and vulnerabilities
- Single exploit affects many

On-Device Receipt Validation

Downloading pre-built solutions

Convenience comes at a price

- Reusing code brings with it bugs and vulnerabilities
- Single exploit affects many

It’s your revenue stream

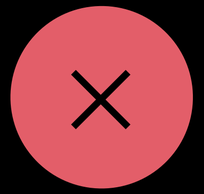
- Make decisions that suit your product
- Know and own the risks

On-Device Receipt Validation

Certificate verification

On-Device Receipt Validation

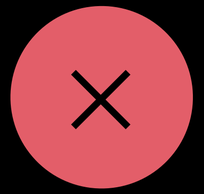
Certificate verification



Do not check the expiry date of the certificate relative to current date

On-Device Receipt Validation

Certificate verification

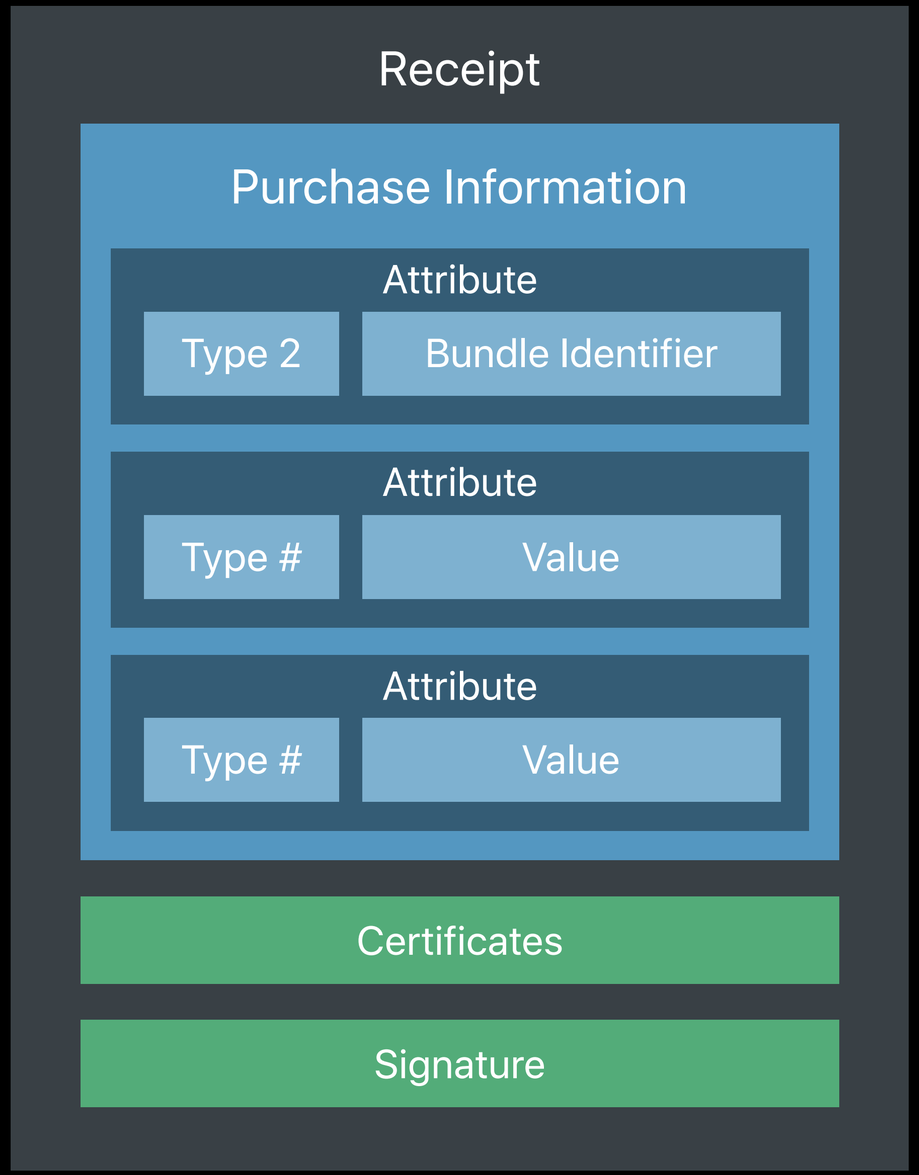


Do not check the expiry date of the certificate relative to current date

Compare expiry date to purchase date of the transaction

On-Device Receipt Validation

Receipt payload

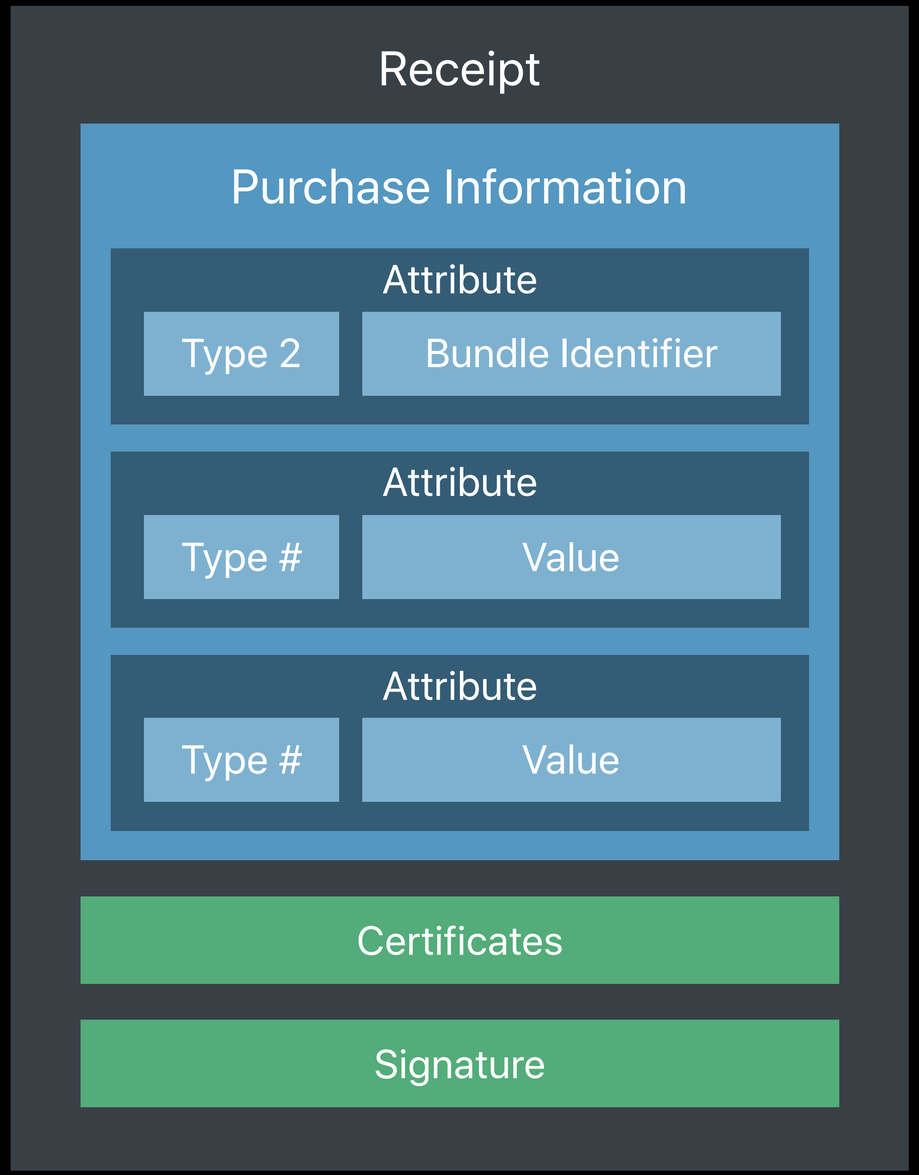


On-Device Receipt Validation

Receipt payload

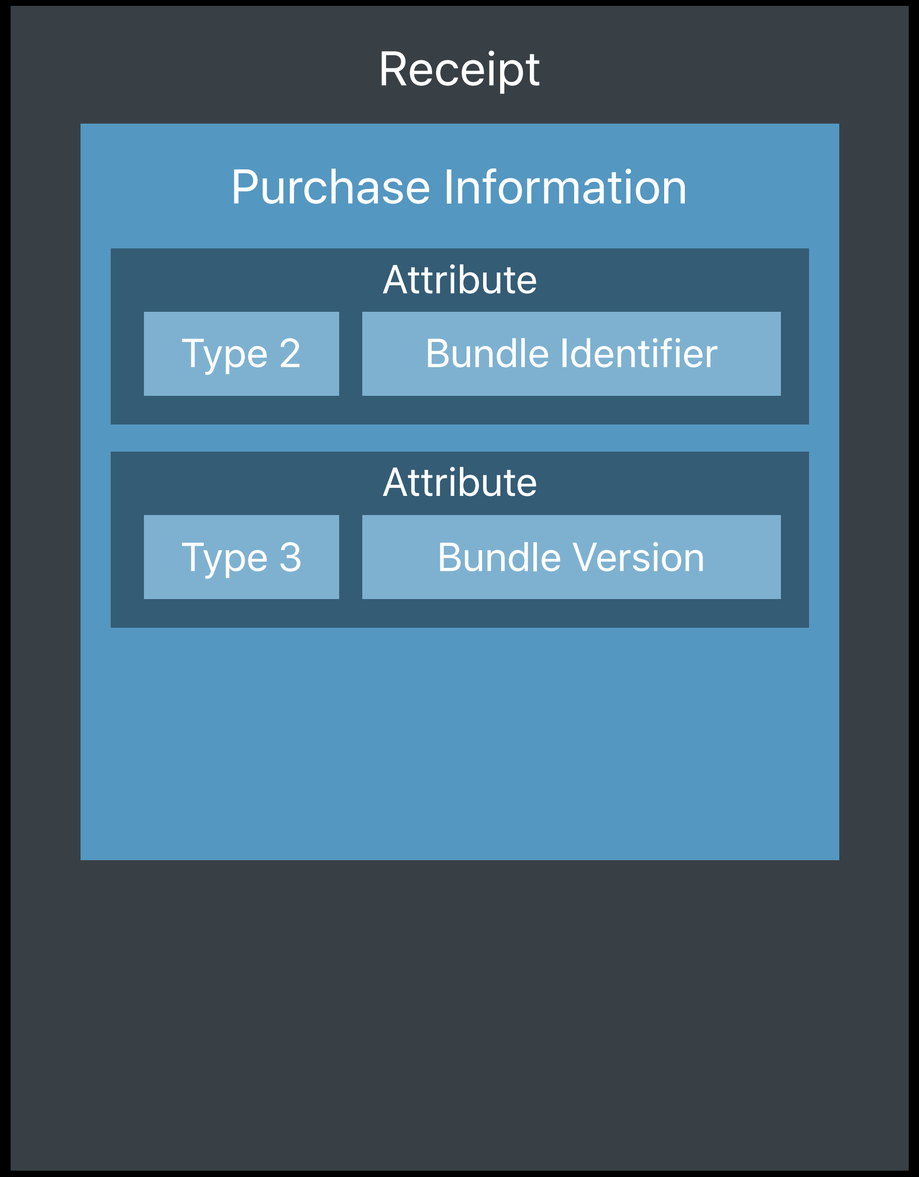
Series of attributes

- Type
- Value
- (Version)



On-Device Receipt Validation

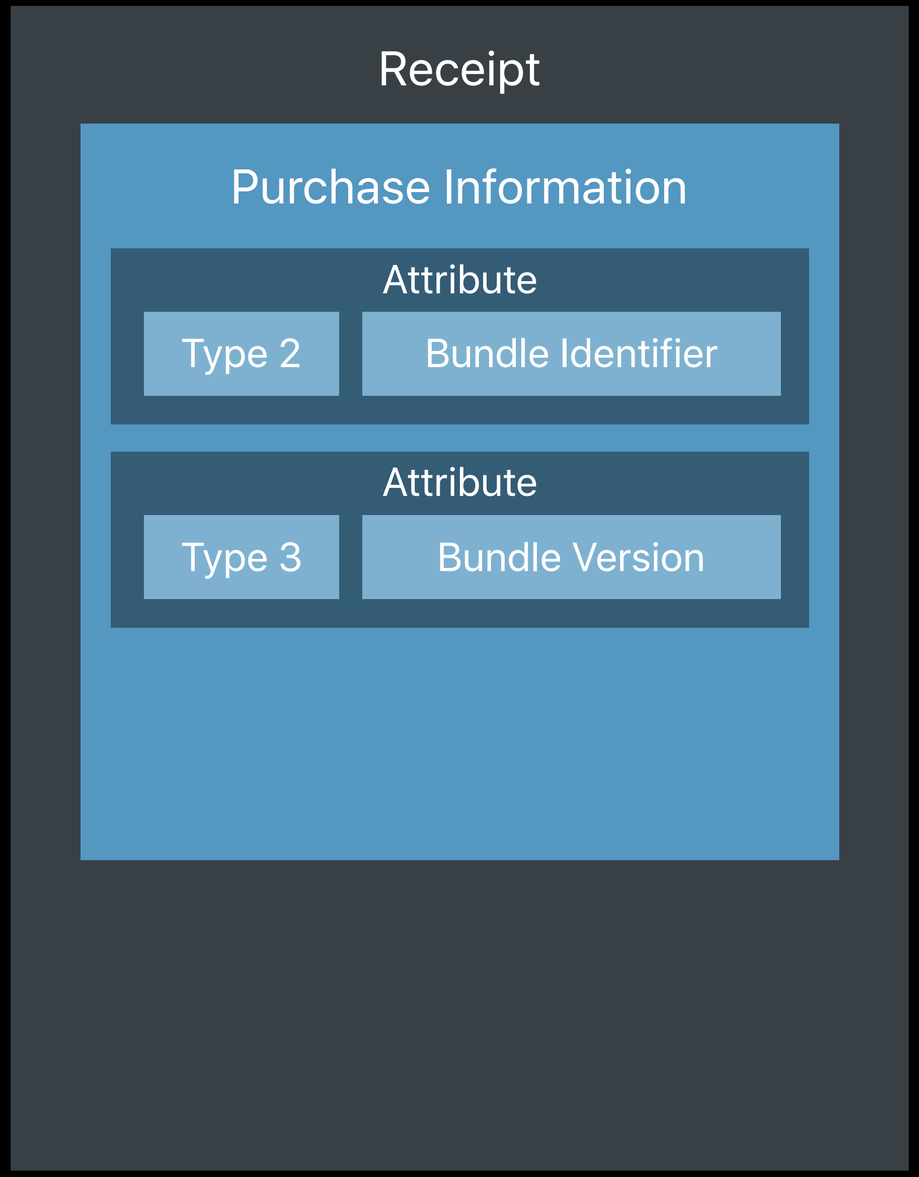
Verify application



On-Device Receipt Validation

Verify application

Check the bundle identifier

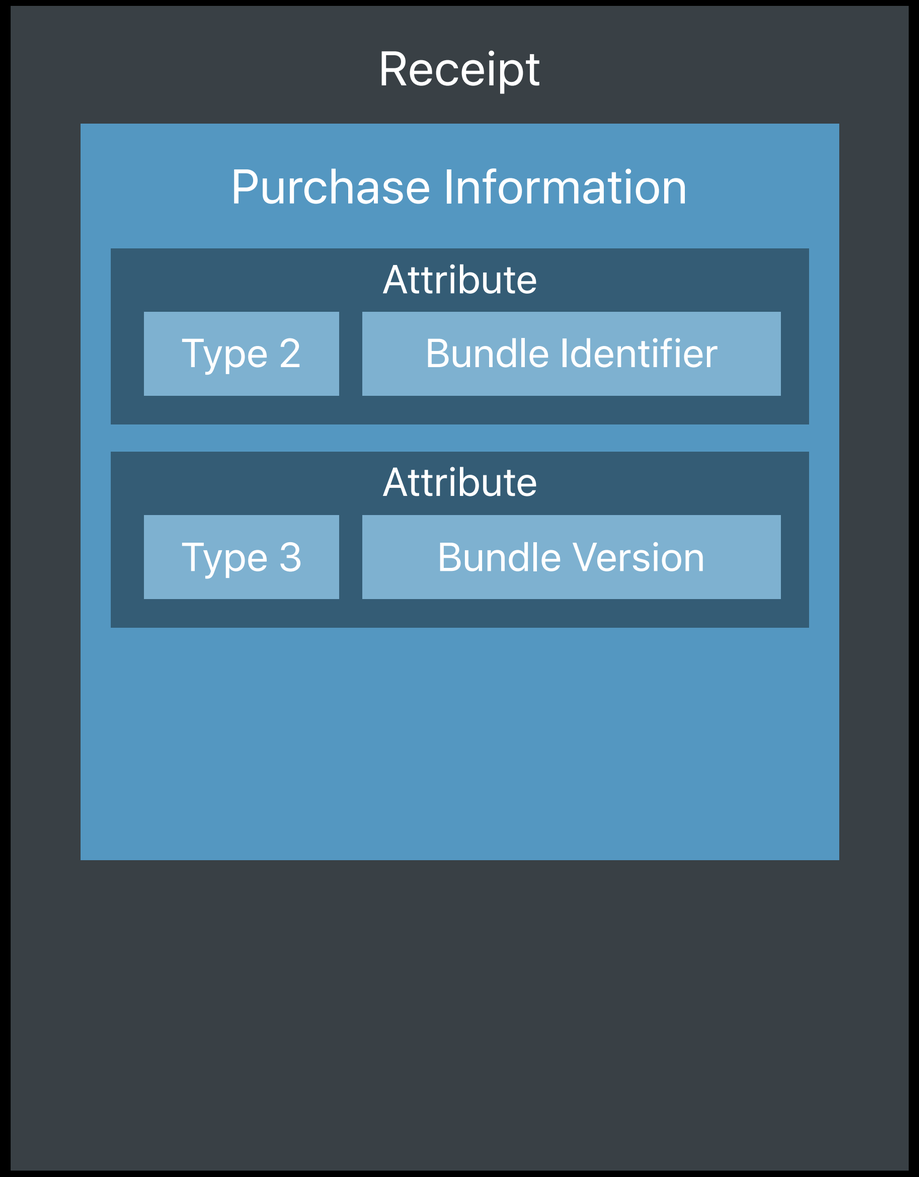


On-Device Receipt Validation

Verify application

Check the bundle identifier

Check the bundle version



On-Device Receipt Validation

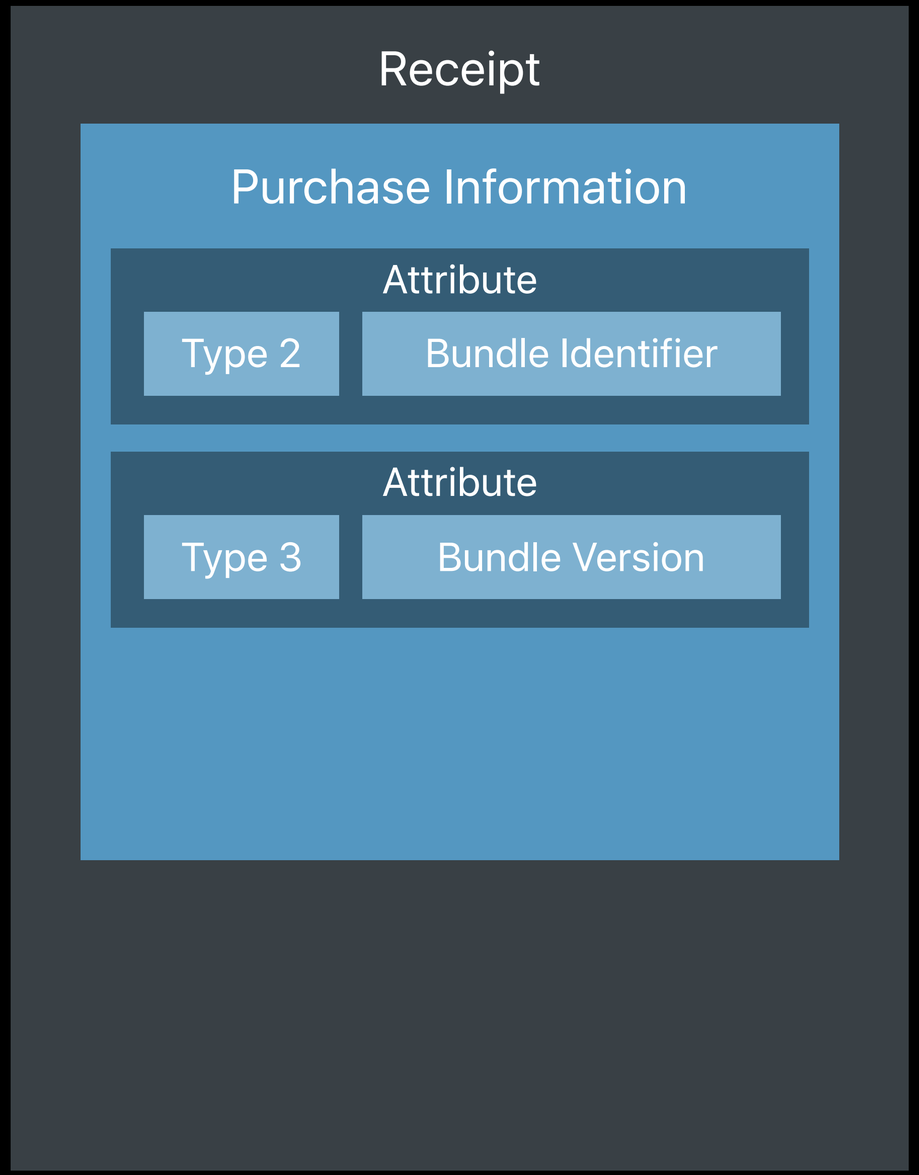
Verify application

Check the bundle identifier

Check the bundle version

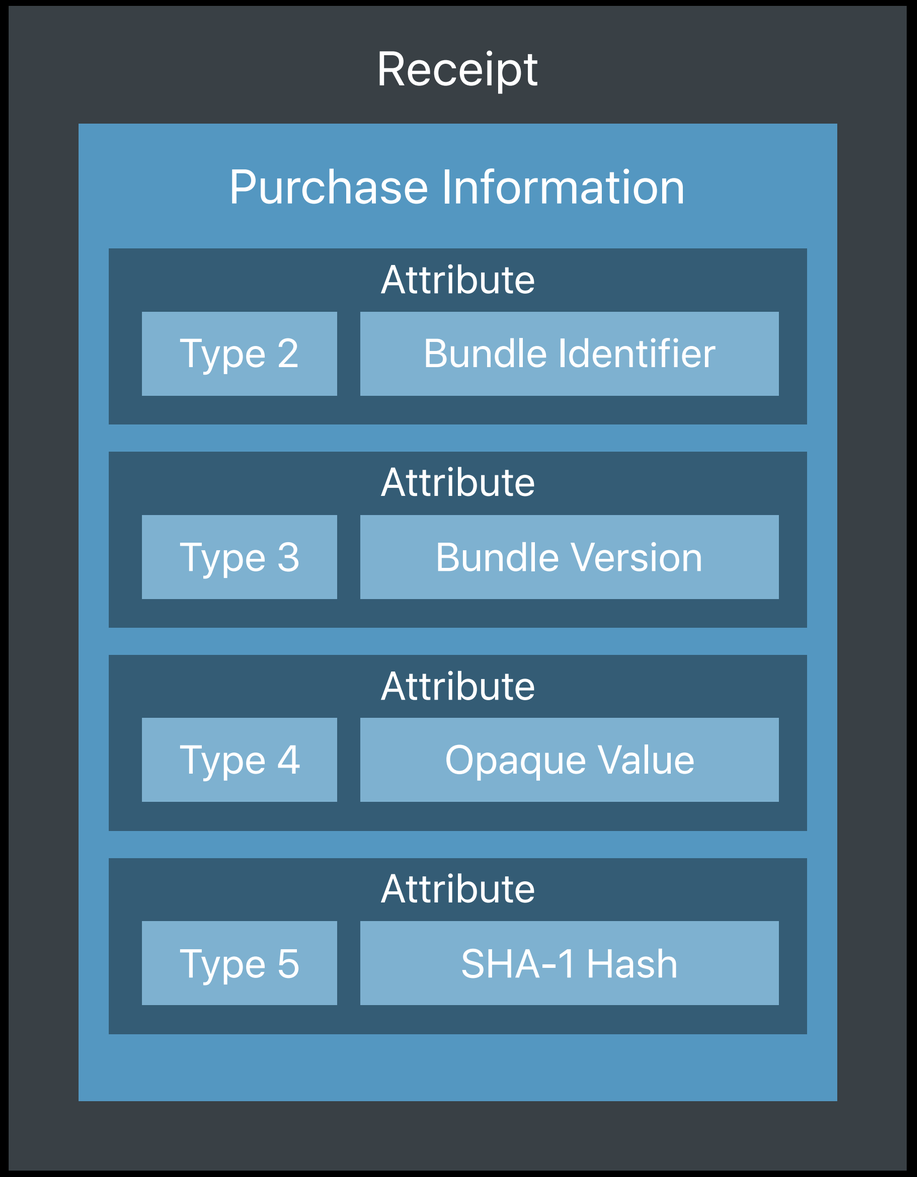
Use hardcoded values

- Not Info.plist values



On-Device Receipt Validation

Verify device

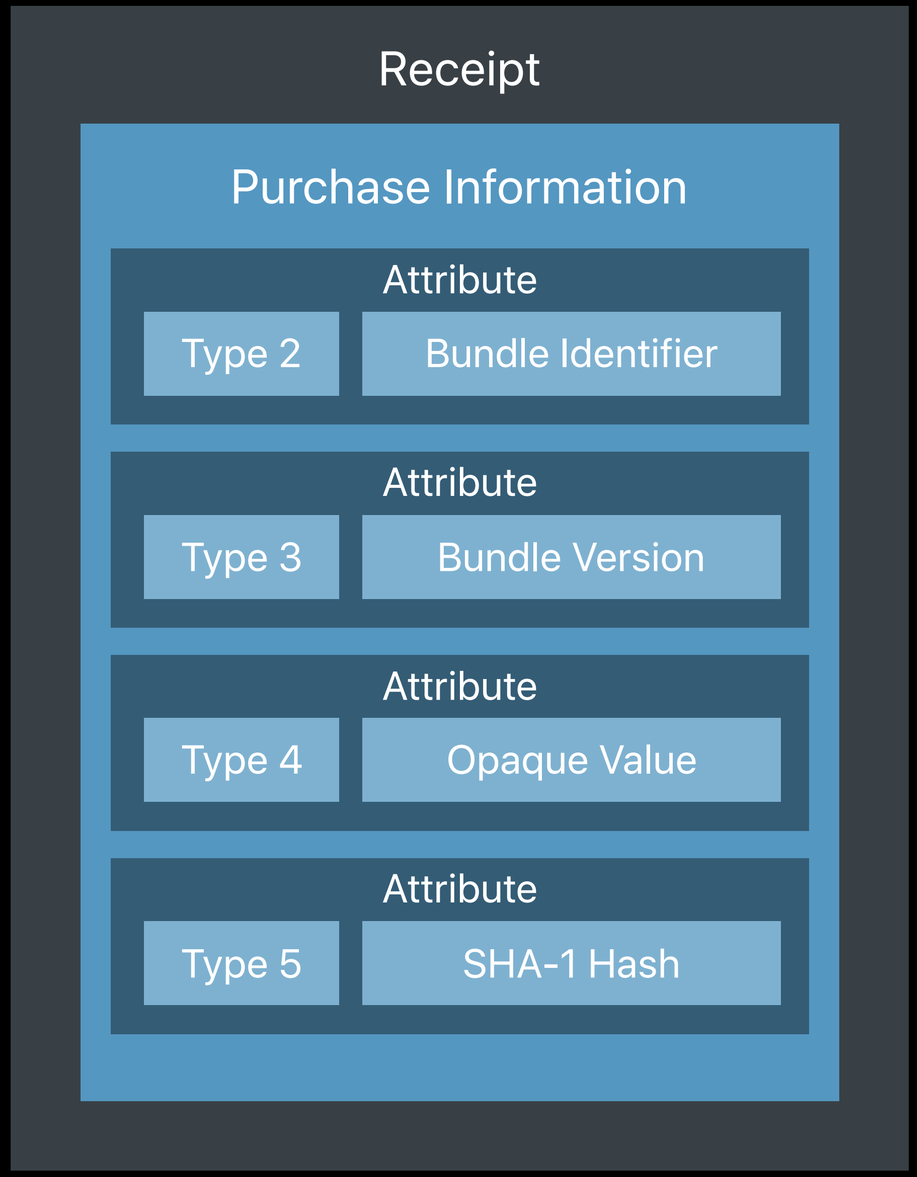


On-Device Receipt Validation

Verify device

Attribute 5 is a SHA-1 hash of three key values

- Bundle identifier
- Device identifier
- Opaque value (Attribute 4)



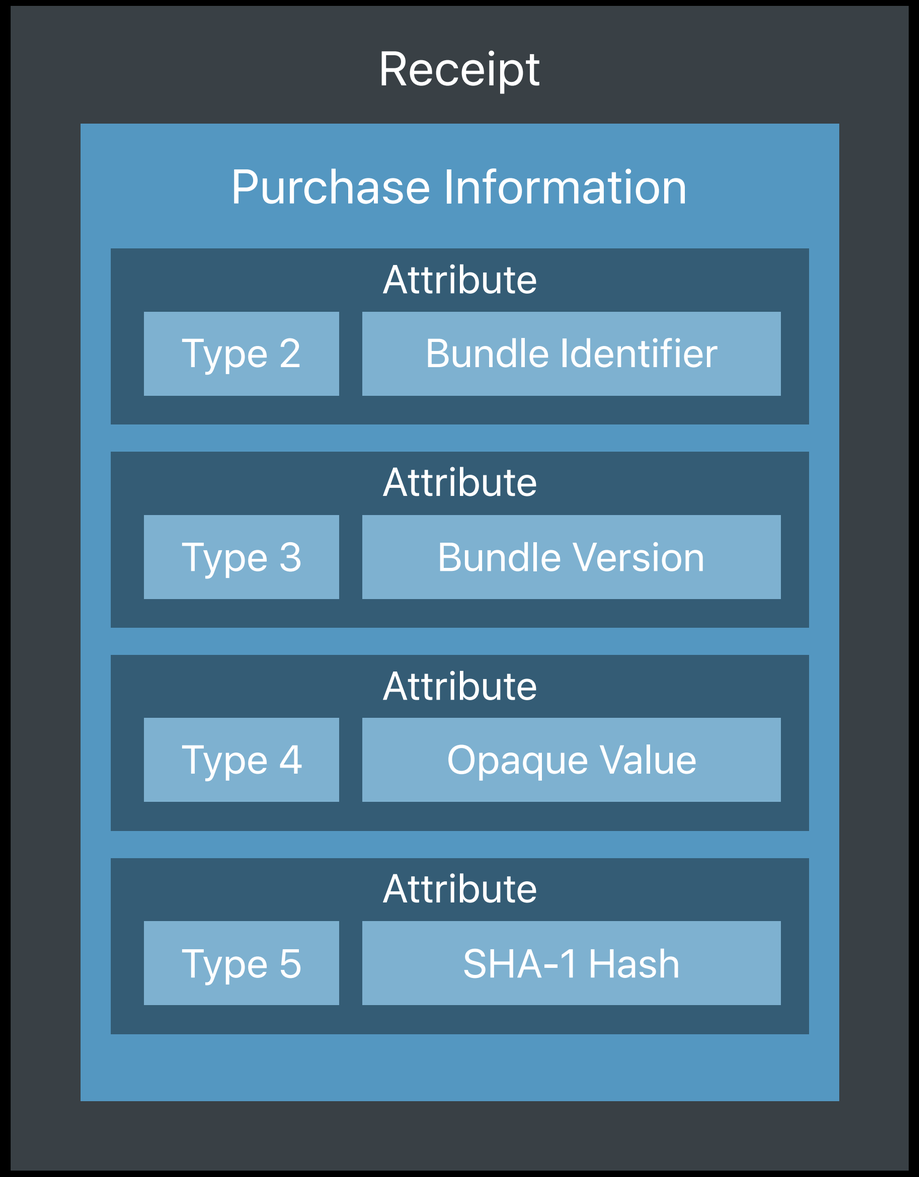
On-Device Receipt Validation

Verify device

Attribute 5 is a SHA-1 hash of three key values

- Bundle identifier
- Device identifier
- Opaque value (Attribute 4)

Unique to your app on this device



On-Device Receipt Validation

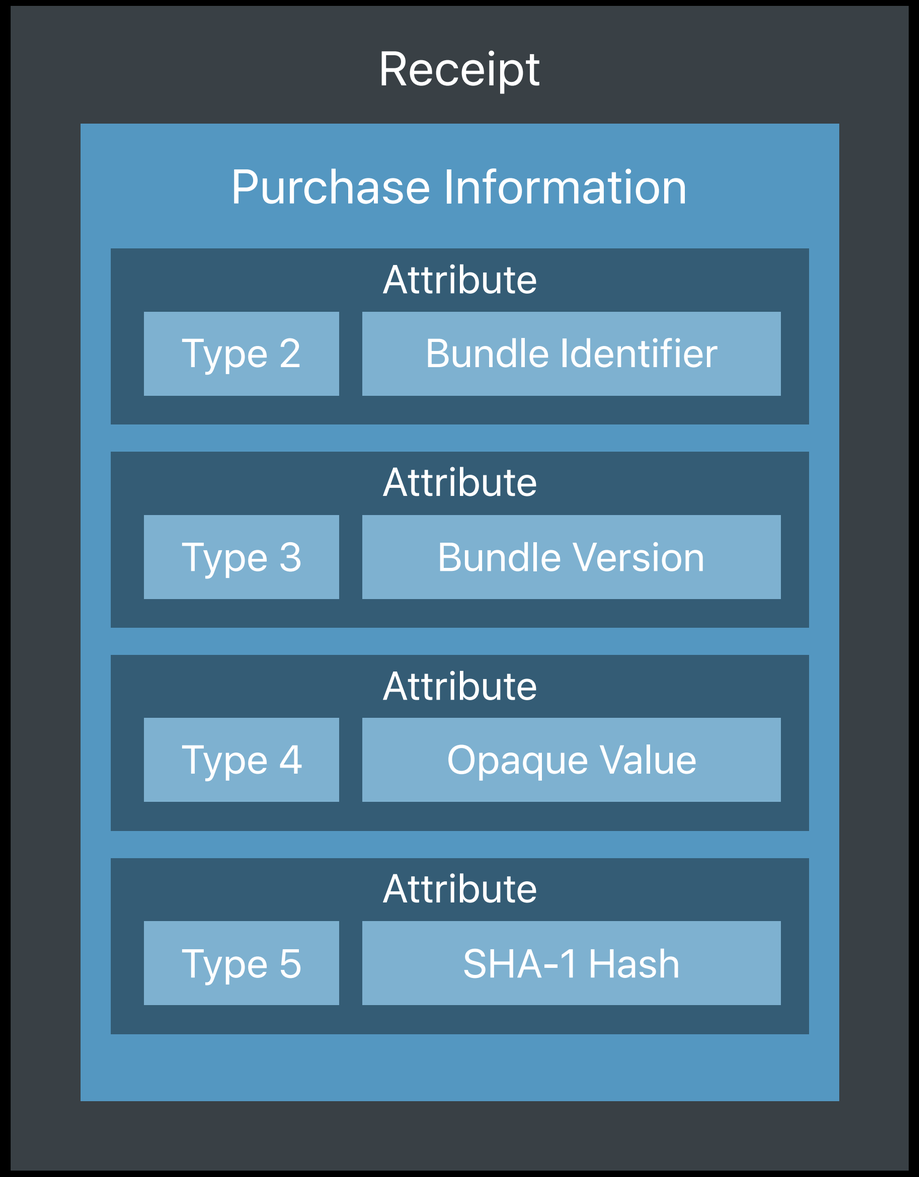
Verify device

Attribute 5 is a SHA-1 hash of three key values

- Bundle identifier
- Device identifier
- Opaque value (Attribute 4)

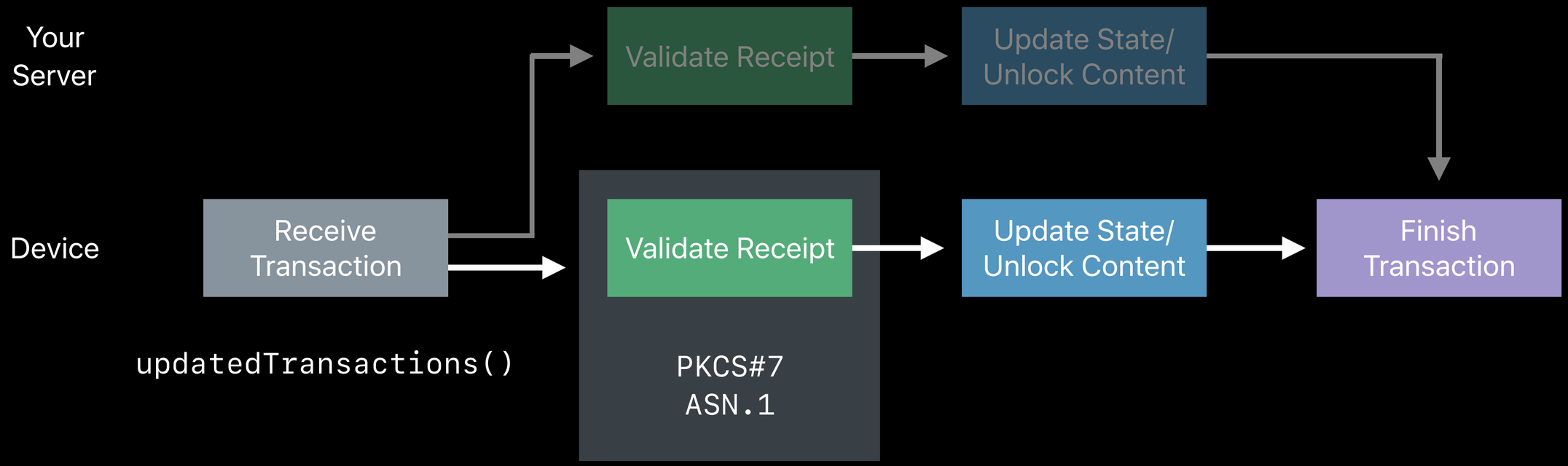
Unique to your app on this device

Create hash using hardcoded values, compare



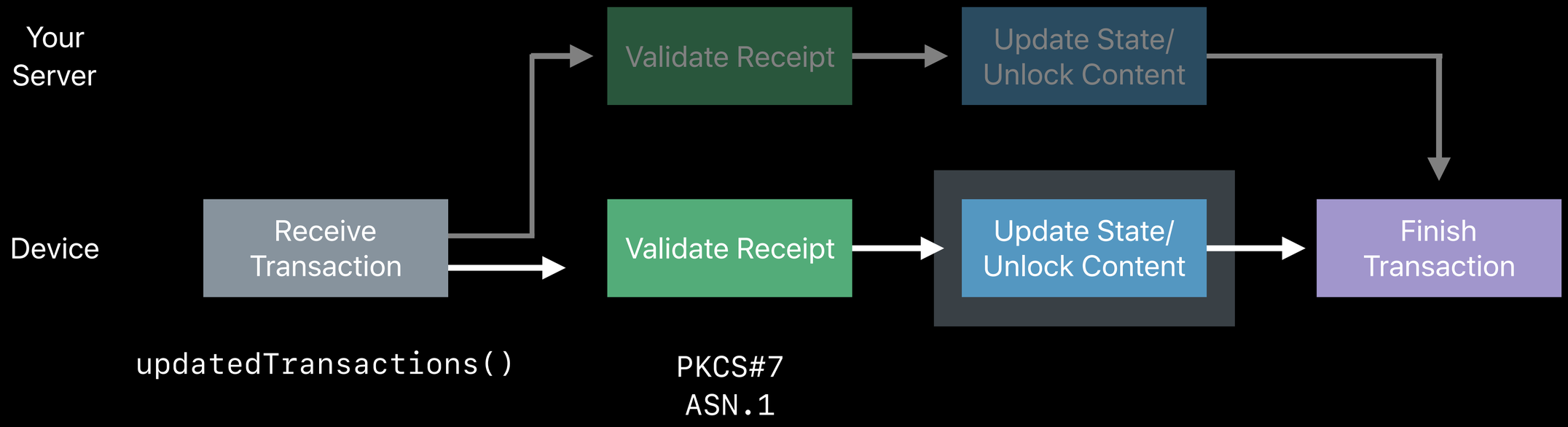
In-App Purchase Process

Processing transactions



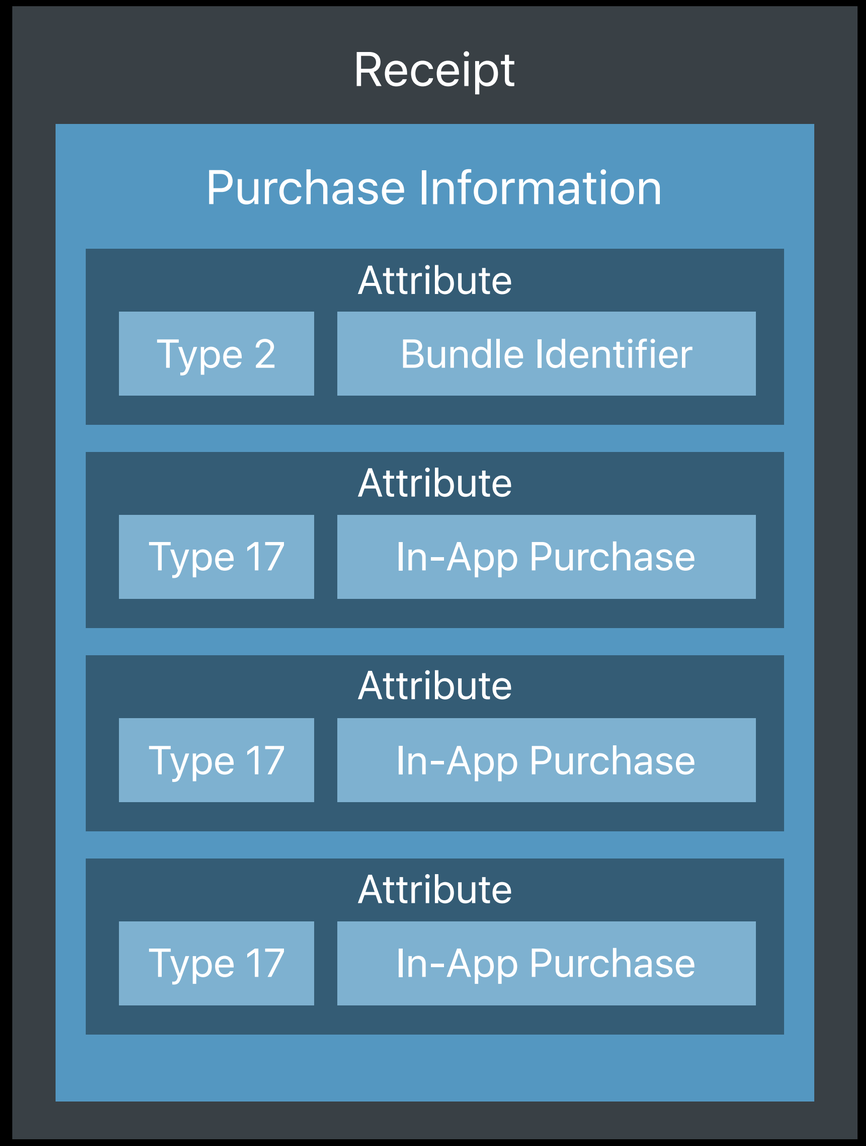
In-App Purchase Process

Processing transactions



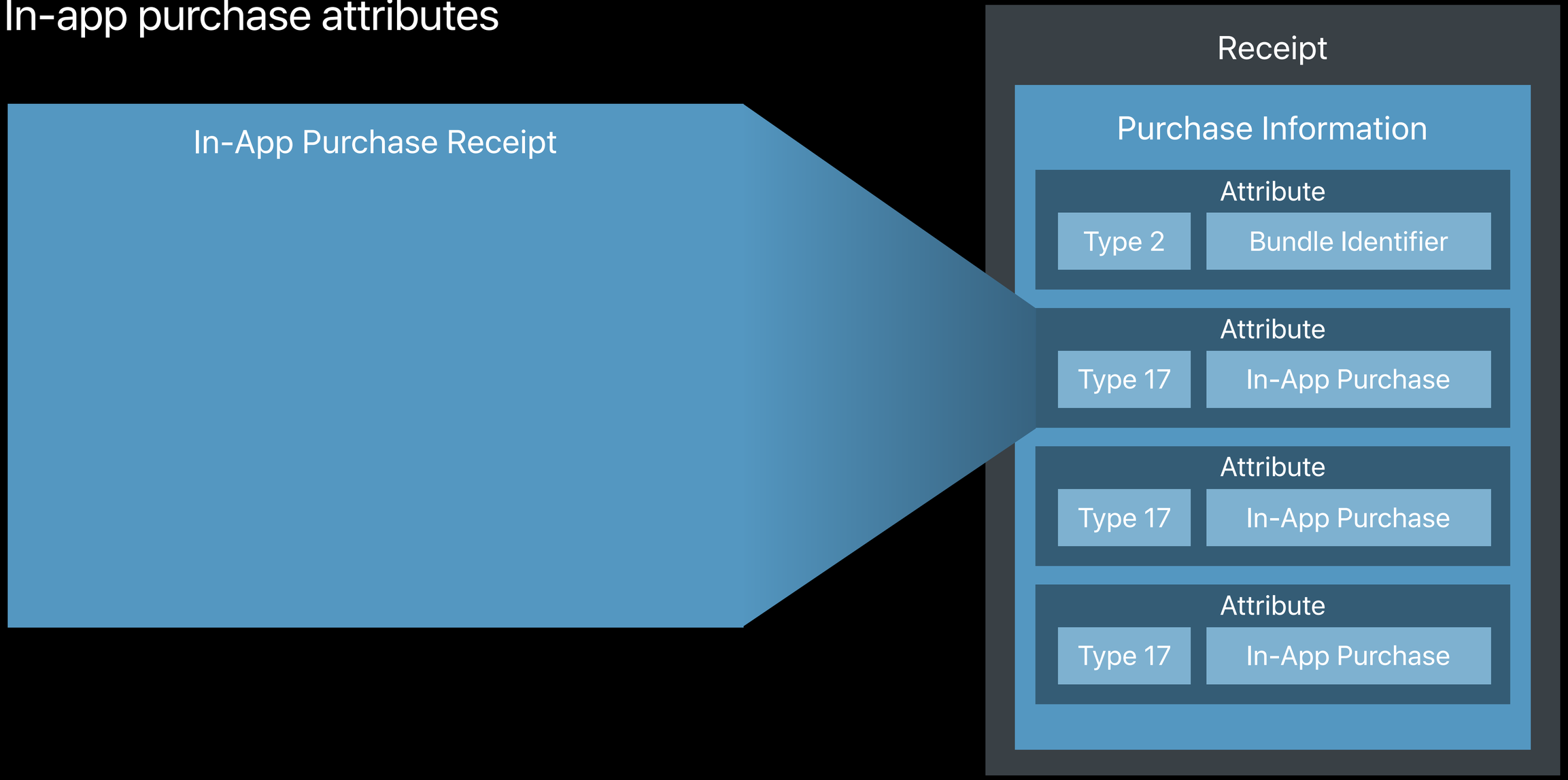
On-Device In-App Purchase State

In-app purchase attributes



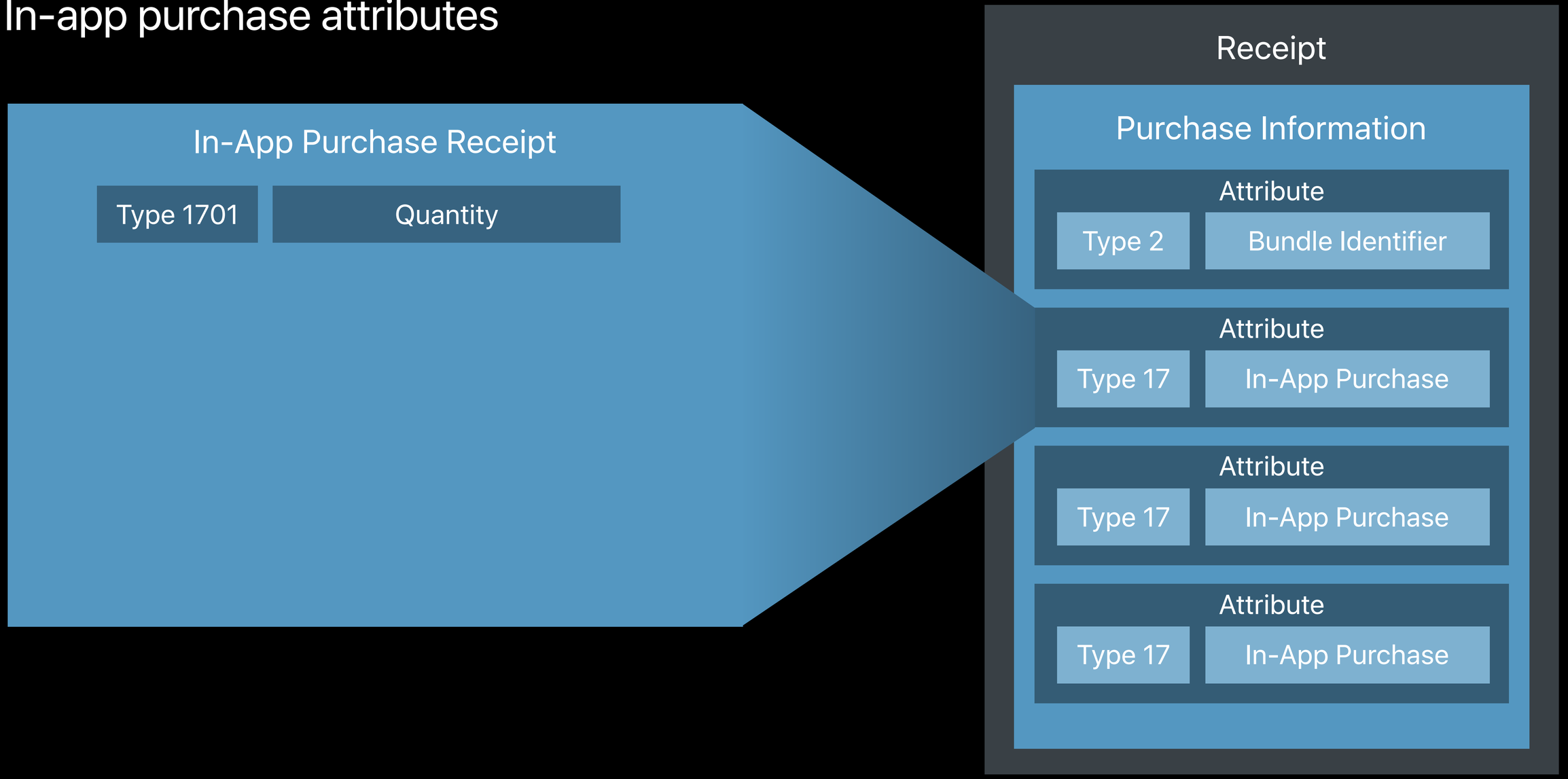
On-Device In-App Purchase State

In-app purchase attributes



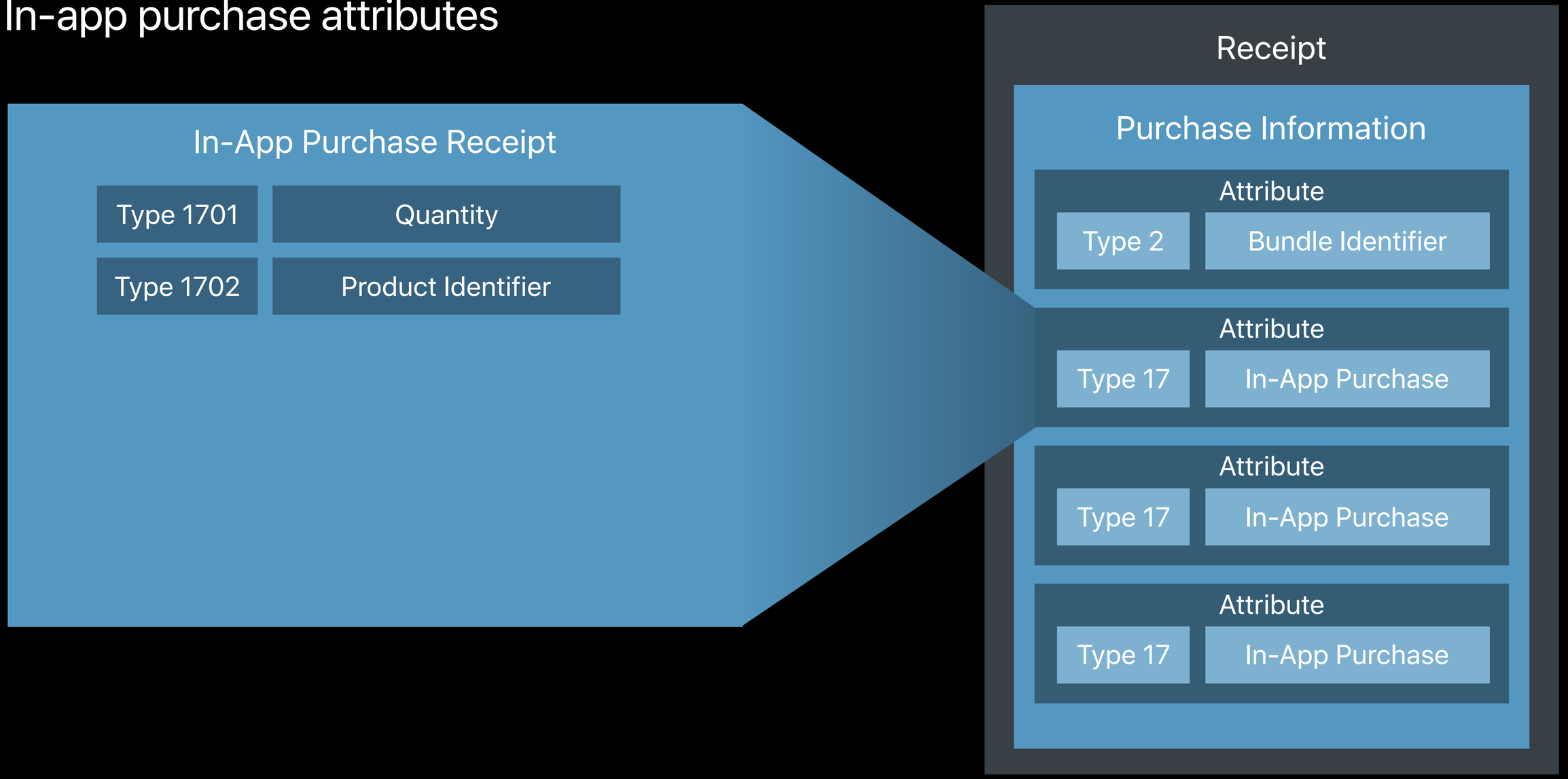
On-Device In-App Purchase State

In-app purchase attributes



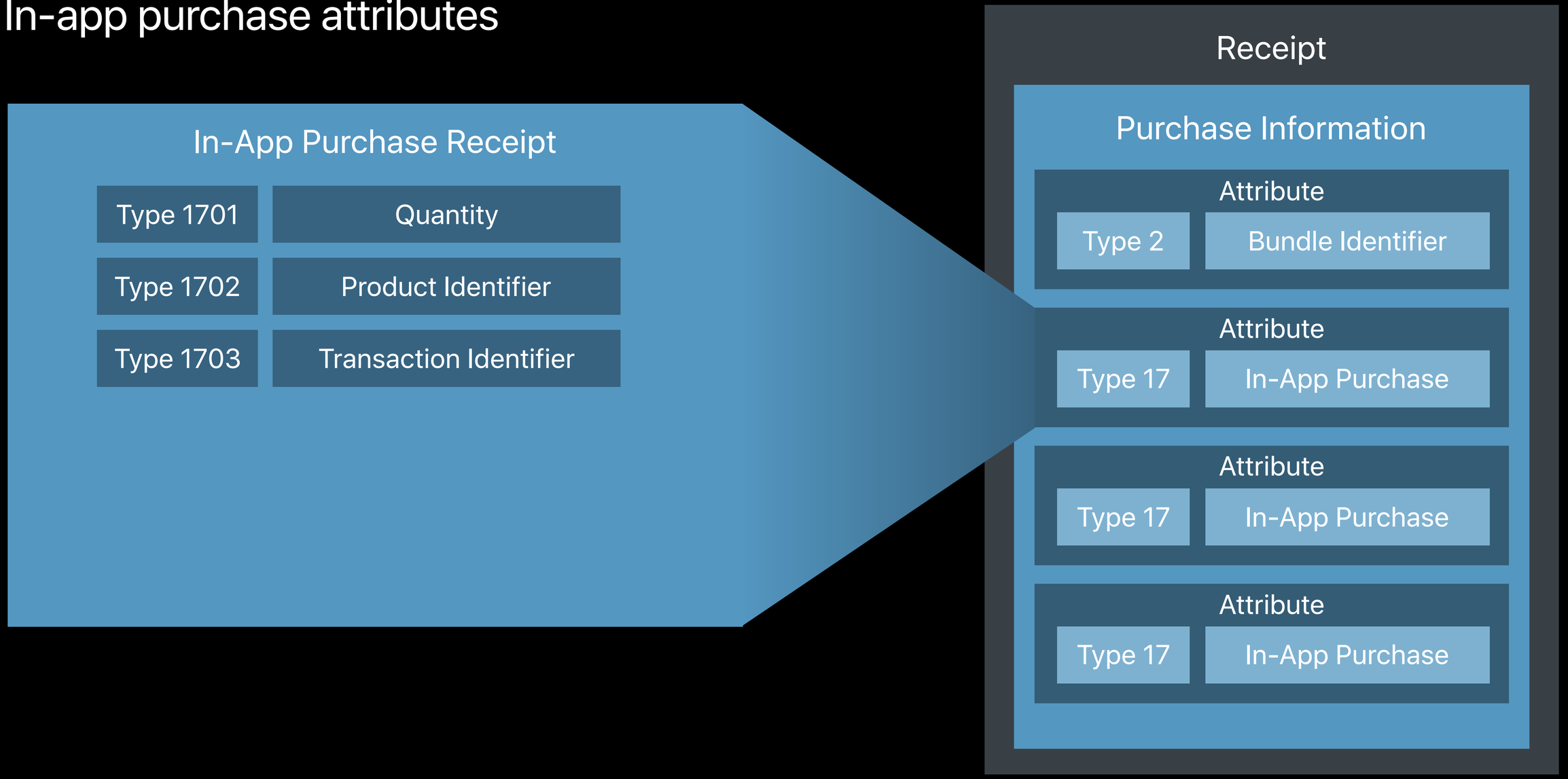
On-Device In-App Purchase State

In-app purchase attributes



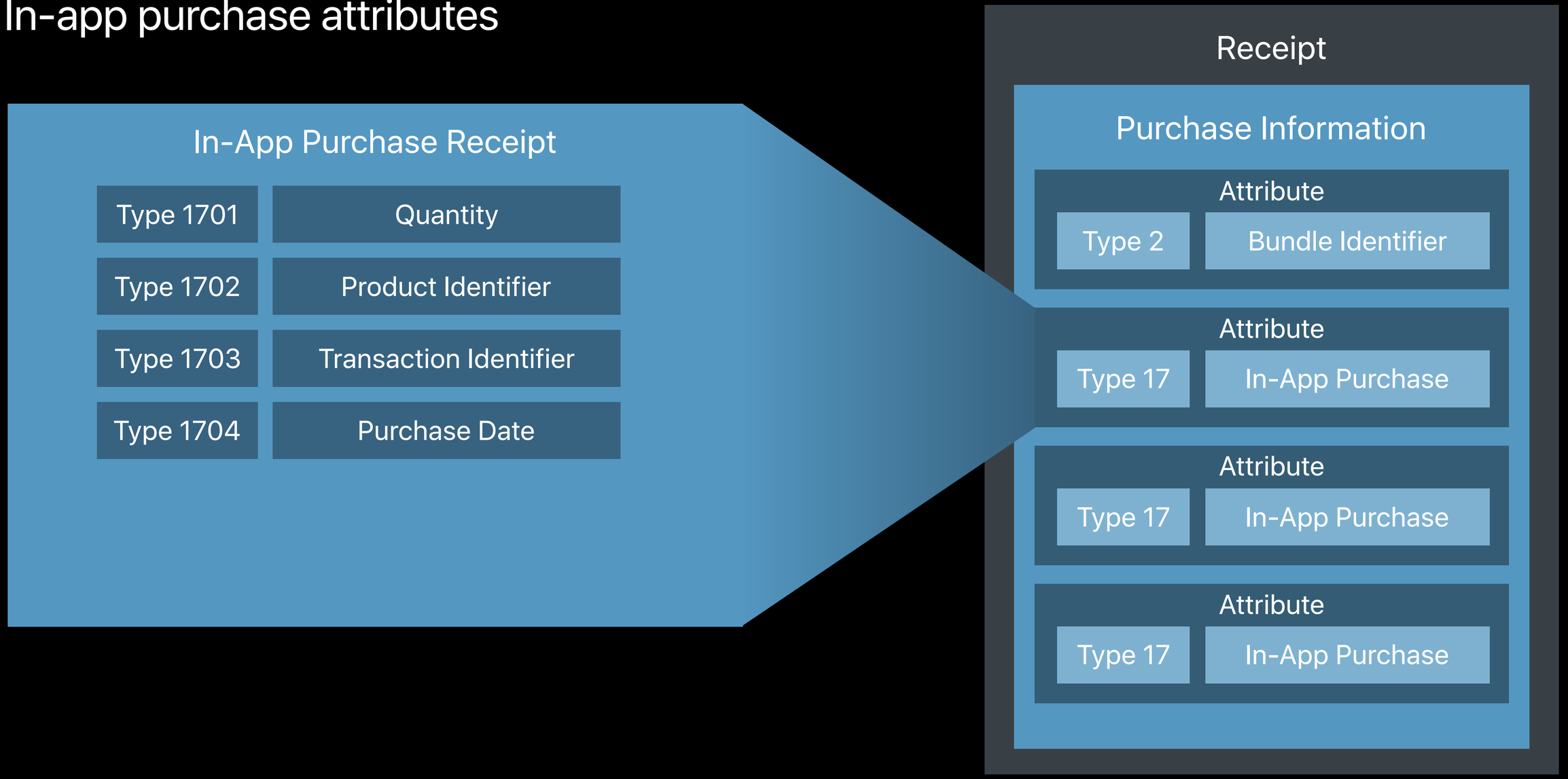
On-Device In-App Purchase State

In-app purchase attributes



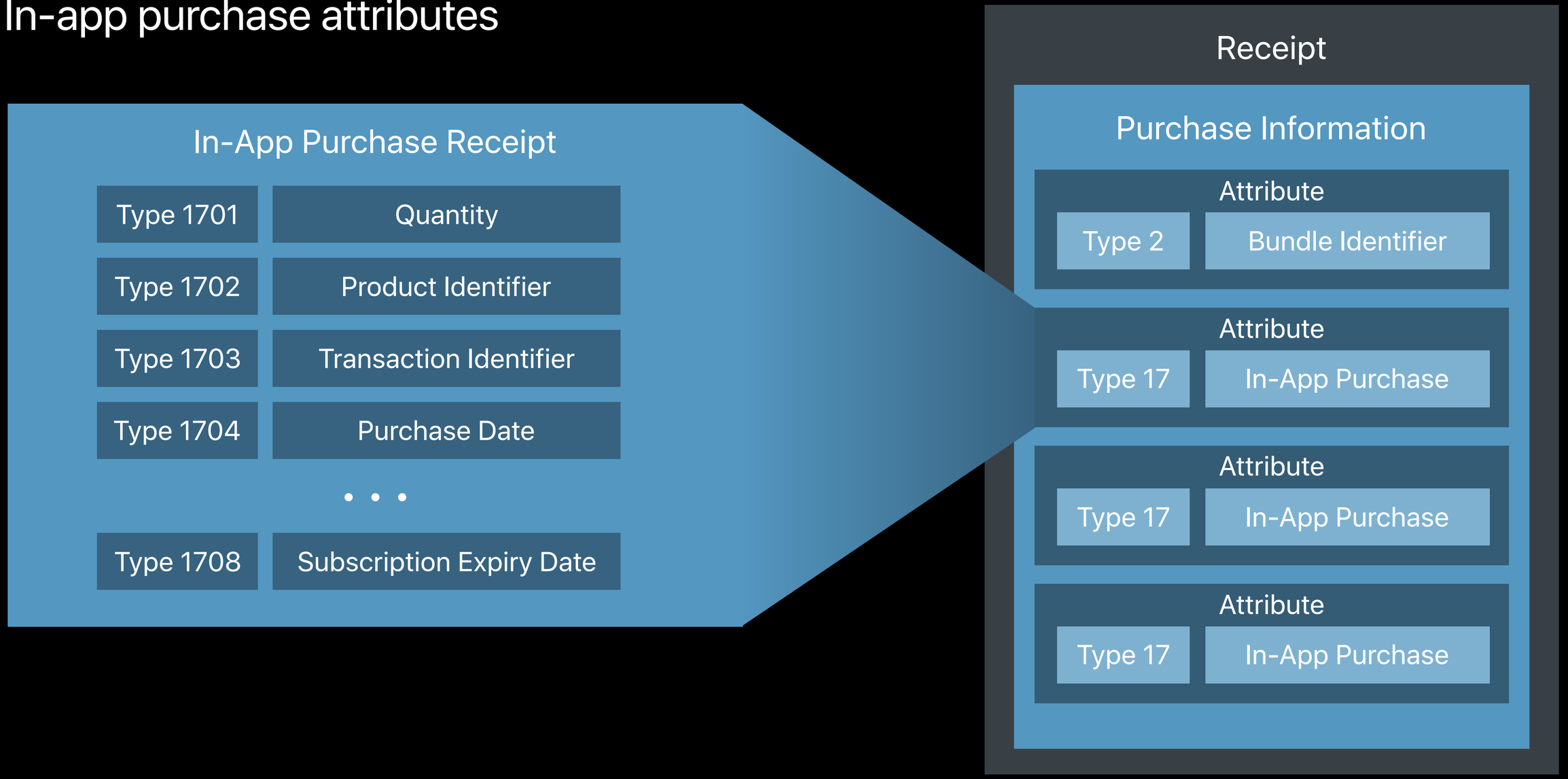
On-Device In-App Purchase State

In-app purchase attributes



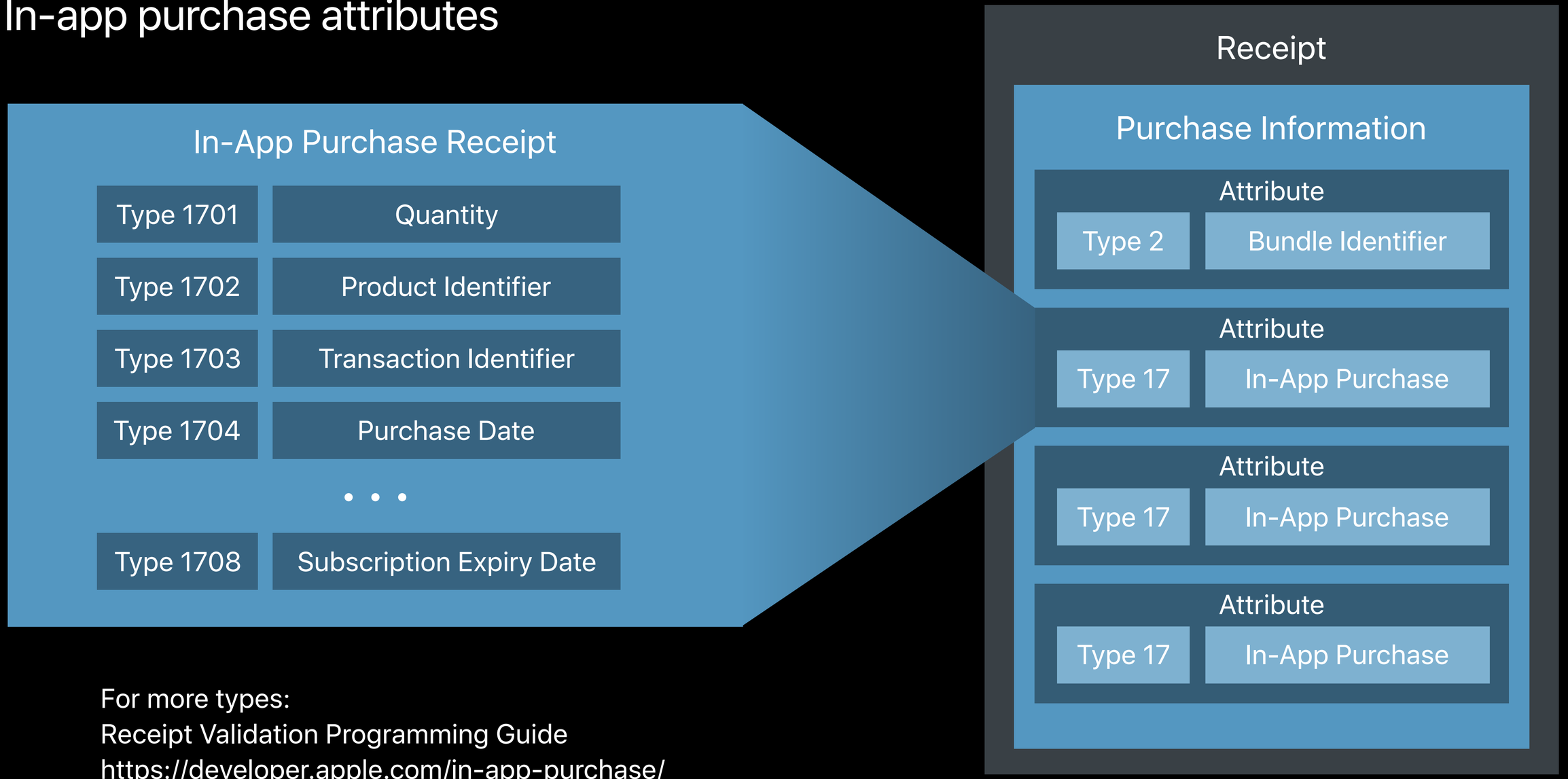
On-Device In-App Purchase State

In-app purchase attributes



On-Device In-App Purchase State

In-app purchase attributes



For more types:
Receipt Validation Programming Guide
https://developer.apple.com/in-app-purchase/

On-Device In-App Purchase State

Unlocking content on-device

On-Device In-App Purchase State

Unlocking content on-device

Transaction will appear in `updatedTransactions` callback

- Transaction observer must be registered early in lifecycle!

On-Device In-App Purchase State

Unlocking content on-device

Transaction will appear in `updatedTransactions` callback

- Transaction observer must be registered early in lifecycle!

Receipt payload contains in-app purchase transactions

- Verify transaction in `updatedTransactions` callback is present in a receipt transaction

On-Device Subscription State

Does the user have an active subscription?

On-Device Subscription State

Does the user have an active subscription?

Valid receipt ≠ subscribed

On-Device Subscription State

Does the user have an active subscription?

Valid receipt ≠ subscribed

Filter transactions by `originalTransactionId`

- Matches the first in-app purchase for that subscription

On-Device Subscription State

Does the user have an active subscription?

Valid receipt ≠ subscribed

Filter transactions by `originalTransactionId`

- Matches the first in-app purchase for that subscription

Check matching transactions for latest expiry date

- Type 1708

On-Device Subscription State

Does the user have an active subscription?

Valid receipt ≠ subscribed

Filter transactions by `originalTransactionId`

- Matches the first in-app purchase for that subscription

Check matching transactions for latest expiry date

- Type 1708

If there’s no valid transaction, can refresh receipt

- Repeat above steps

On-Device Subscription State

Caveat

On-Device Subscription State

Caveat

For determining if subscription is valid, inspect

- Purchase date
- Expiry date
- System date

On-Device Subscription State

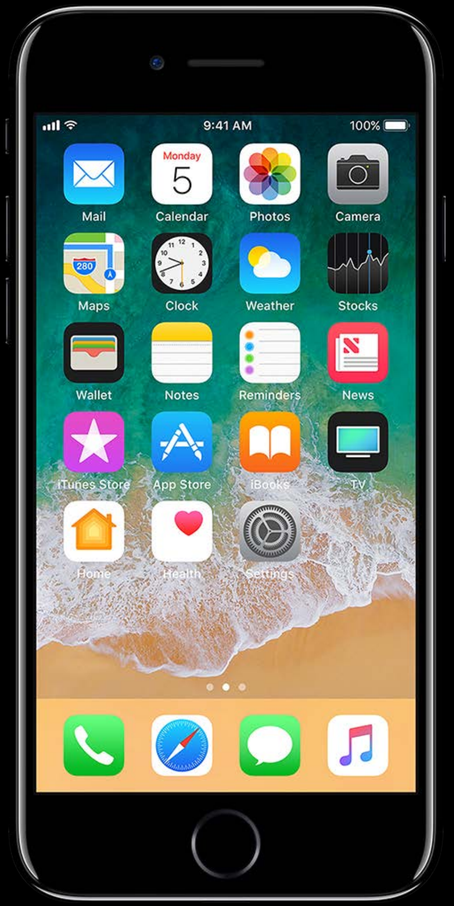
Caveat

For determining if subscription is valid, inspect

- Purchase date
- Expiry date
- System date

Caveat for on-device subscription state

- Device clock could be wound back!



On-Device Receipt Validation

Receipt Refresh on iOS

On-Device Receipt Validation

Receipt Refresh on iOS

If the receipt doesn’t exist or is invalid, refresh the receipt using StoreKit

On-Device Receipt Validation

Receipt Refresh on iOS

If the receipt doesn’t exist or is invalid, refresh the receipt using StoreKit

Receipt refresh will require network

On-Device Receipt Validation

Receipt Refresh on iOS

If the receipt doesn’t exist or is invalid, refresh the receipt using StoreKit

Receipt refresh will require network

Store sign-in will be required

On-Device Receipt Validation

Receipt Refresh on iOS

If the receipt doesn’t exist or is invalid, refresh the receipt using StoreKit

Receipt refresh will require network

Store sign-in will be required

Avoid continuous loop of validate-and-refresh

On-Device Receipt Validation

Receipt Refresh on iOS

If the receipt doesn’t exist or is invalid, refresh the receipt using StoreKit

Receipt refresh will require network

Store sign-in will be required

Avoid continuous loop of validate-and-refresh

```
let request = SKReceiptRefreshRequest()
request.delegate = self
request.start()
```

On-Device Receipt Validation

Receipt Refresh on macOS

On-Device Receipt Validation

Receipt Refresh on macOS

If the receipt is invalid

On-Device Receipt Validation

Receipt Refresh on macOS

If the receipt is invalid

Receipt refresh will require network

On-Device Receipt Validation

Receipt Refresh on macOS

If the receipt is invalid

Receipt refresh will require network

Store sign-in will be required

On-Device Receipt Validation

Receipt Refresh on macOS

If the receipt is invalid

Receipt refresh will require network

Store sign-in will be required

Exit with code 173 to refresh receipt

On-Device Receipt Validation

Receipt Refresh on macOS

If the receipt is invalid

Receipt refresh will require network

Store sign-in will be required

Exit with code 173 to refresh receipt

```
// Receipt is invalid
exit(173)
```

Receipt Tips

Restoring transactions vs. refreshing receipt

Receipt Tips

Restoring transactions vs. refreshing receipt

```
restoreCompletedTransactions()
```

```
SKReceiptRefreshRequest()
```

Receipt Tips

Restoring transactions vs. refreshing receipt

```
restoreCompletedTransactions()
```

API on `SKPaymentQueue`

```
SKReceiptRefreshRequest()
```

Receipt Tips

Restoring transactions vs. refreshing receipt

`restoreCompletedTransactions()`

API on `SKPaymentQueue`

`SKReceiptRefreshRequest()`

Create request instance, call `start()`

Receipt Tips

Restoring transactions vs. refreshing receipt

`restoreCompletedTransactions()`

API on `SKPaymentQueue`

Completed transactions appear in `updatedTransactions()`

`SKReceiptRefreshRequest()`

Create request instance, call `start()`

New receipt document is downloaded onto device

Receipt Tips

Restoring transactions vs. refreshing receipt

`restoreCompletedTransactions()`

API on `SKPaymentQueue`

Completed transactions appear in `updatedTransactions()`

Restores:

- Non-consumable
- Auto-renewable subscriptions

`SKReceiptRefreshRequest()`

Create request instance, call `start()`

New receipt document is downloaded onto device

Receipt Tips

Restoring transactions vs. refreshing receipt

`restoreCompletedTransactions()`

API on `SKPaymentQueue`

Completed transactions appear in `updatedTransactions()`

Restores:

- Non-consumable
- Auto-renewable subscriptions

`SKReceiptRefreshRequest()`

Create request instance, call `start()`

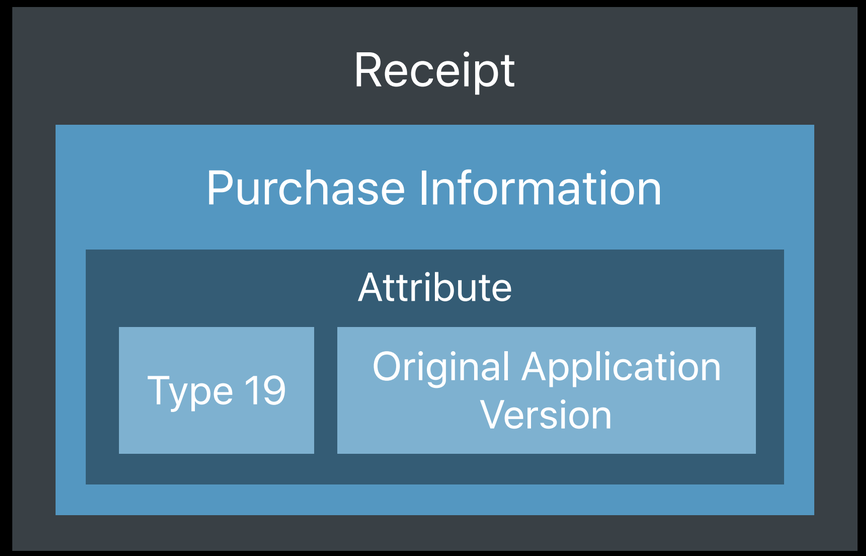
New receipt document is downloaded onto device

Restores:

- Non-consumable
- Non-renewing subscriptions
- Auto-renewable subscriptions

Receipt Tips

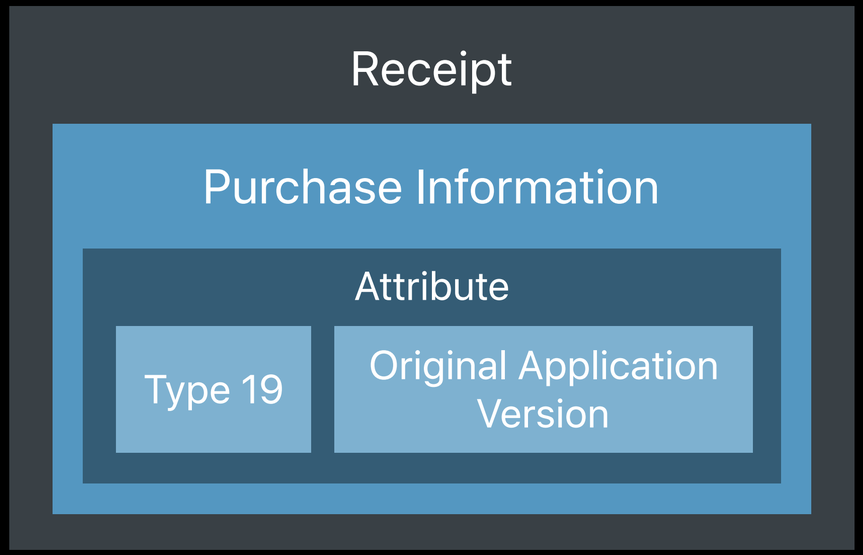
Switching to subscriptions



Receipt Tips

Switching to subscriptions

Original application version in receipt

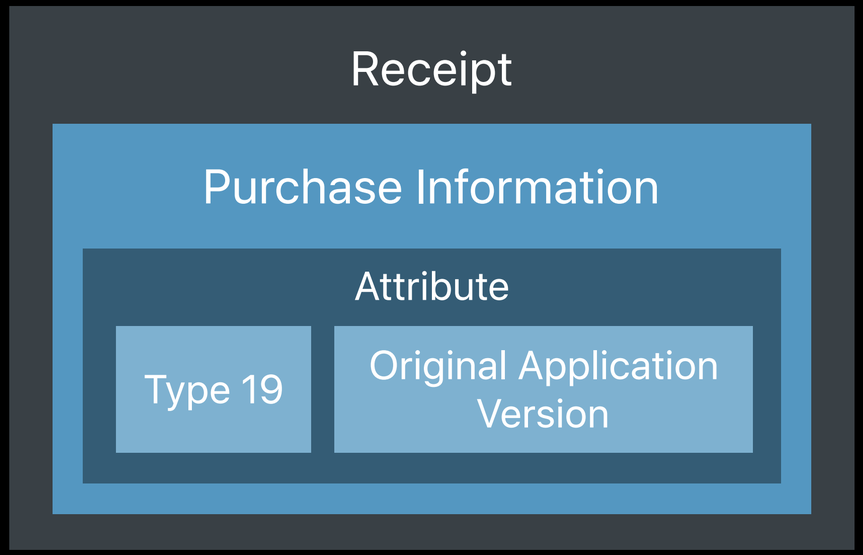


Receipt Tips

Switching to subscriptions

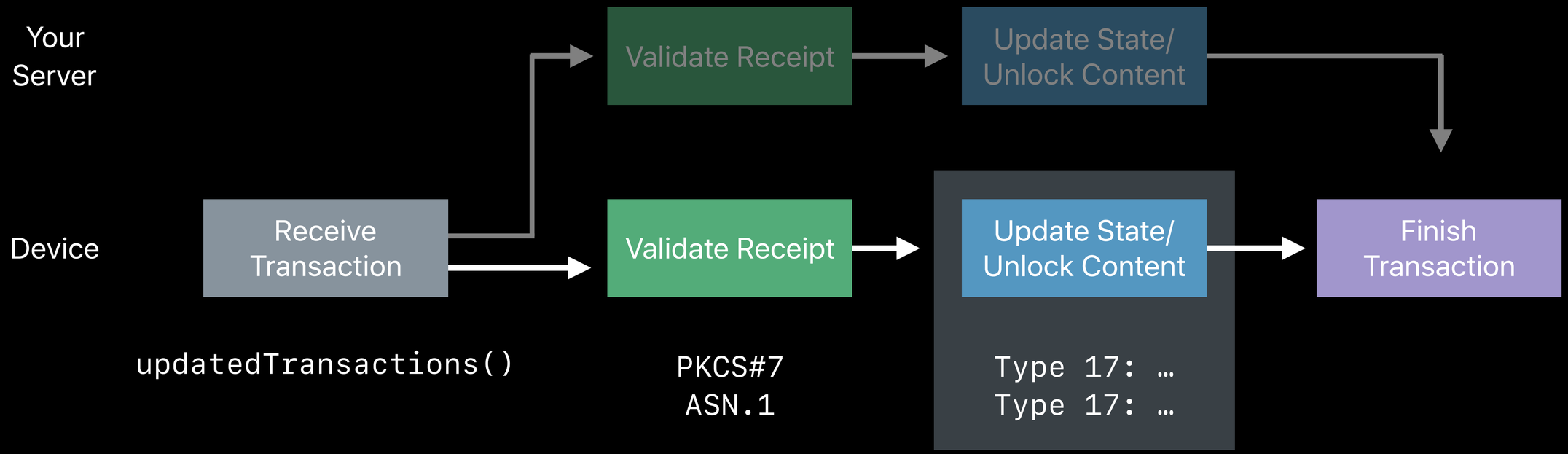
Original application version in receipt

Know whether the app was purchased as paid version, or is a subscription version



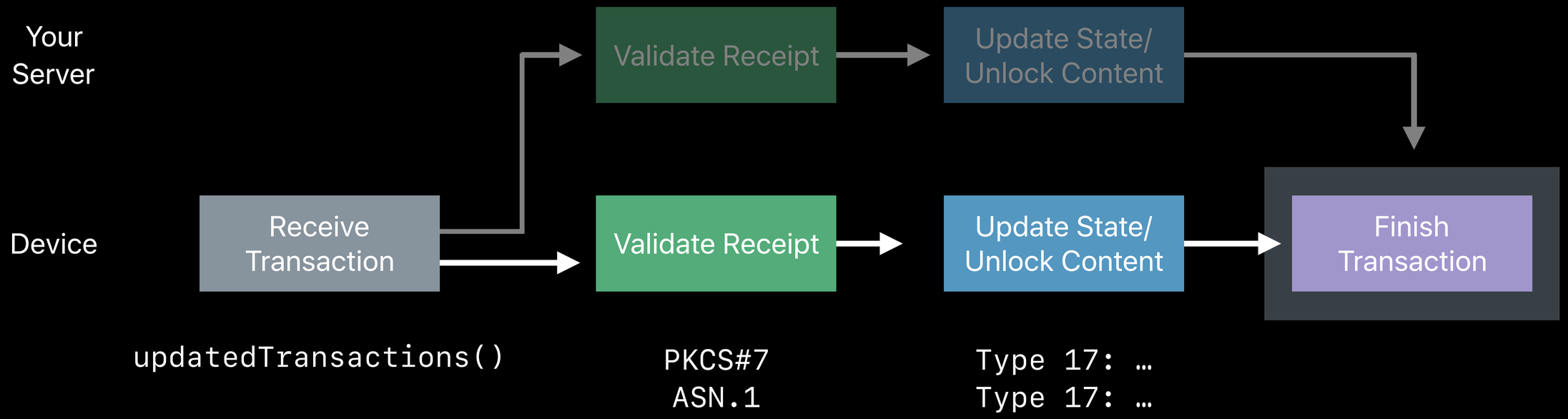
In-App Purchase Process

Processing transactions



In-App Purchase Process

Processing transactions



In-App Purchase Process

Finish the transaction

In-App Purchase Process

Finish the transaction

Finish all transactions once content is unlocked

- If downloading hosted content, wait until after the download completes

In-App Purchase Process

Finish the transaction

Finish all transactions once content is unlocked

- If downloading hosted content, wait until after the download completes

Includes all auto-renewable subscription transactions

In-App Purchase Process

Finish the transaction

Finish all transactions once content is unlocked

- If downloading hosted content, wait until after the download completes

Includes all auto-renewable subscription transactions

Otherwise, the payment will stay in the queue

In-App Purchase Process

Finish the transaction

Finish all transactions once content is unlocked

- If downloading hosted content, wait until after the download completes

Includes all auto-renewable subscription transactions

Otherwise, the payment will stay in the queue

Subscription billing retry depends on up-to-date information about transaction

In-App Purchase Process

Finish the transaction

Finish all transactions once content is unlocked

- If downloading hosted content, wait until after the download completes

Includes all auto-renewable subscription transactions

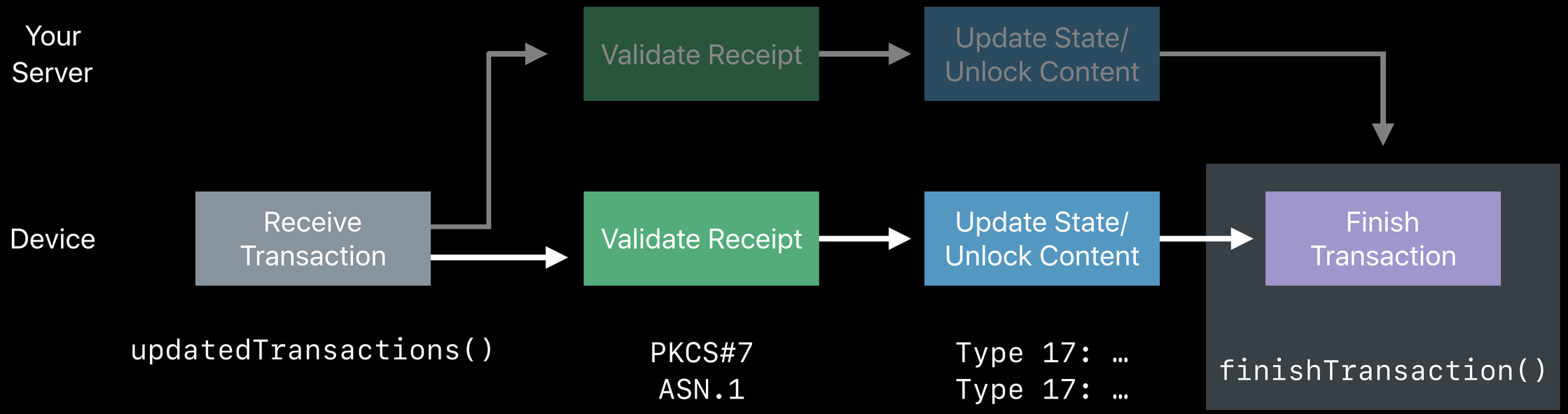
Otherwise, the payment will stay in the queue

Subscription billing retry depends on up-to-date information about transaction

```
SKPaymentQueue.default().finishTransaction(transaction)
```

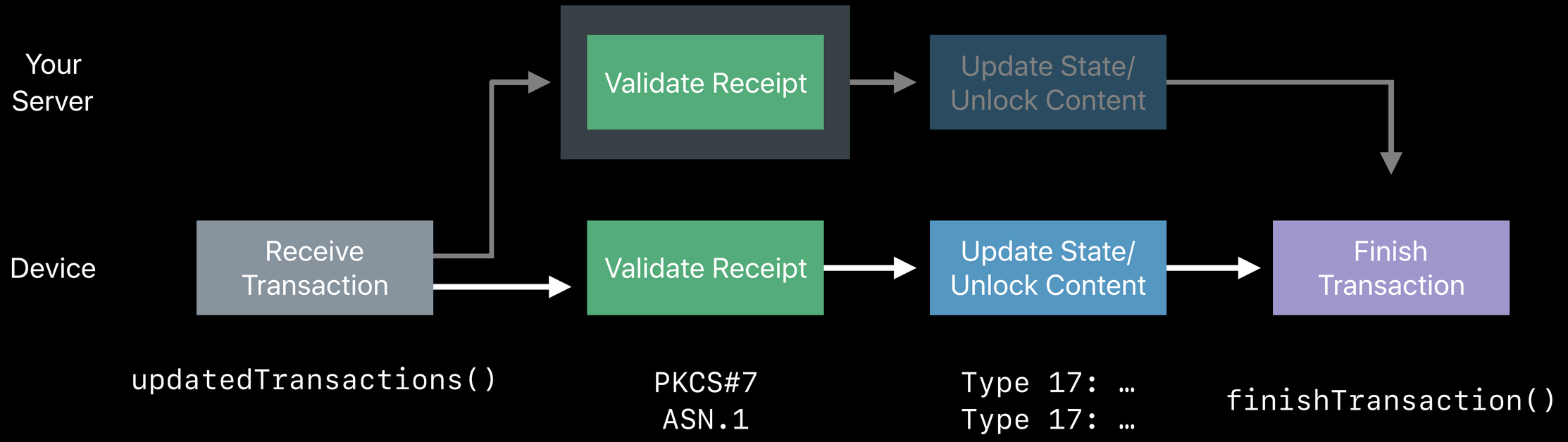
In-App Purchase Process

Processing transactions



In-App Purchase Process

Processing transactions



Server-Side Receipt Validation

Server-to-server validation

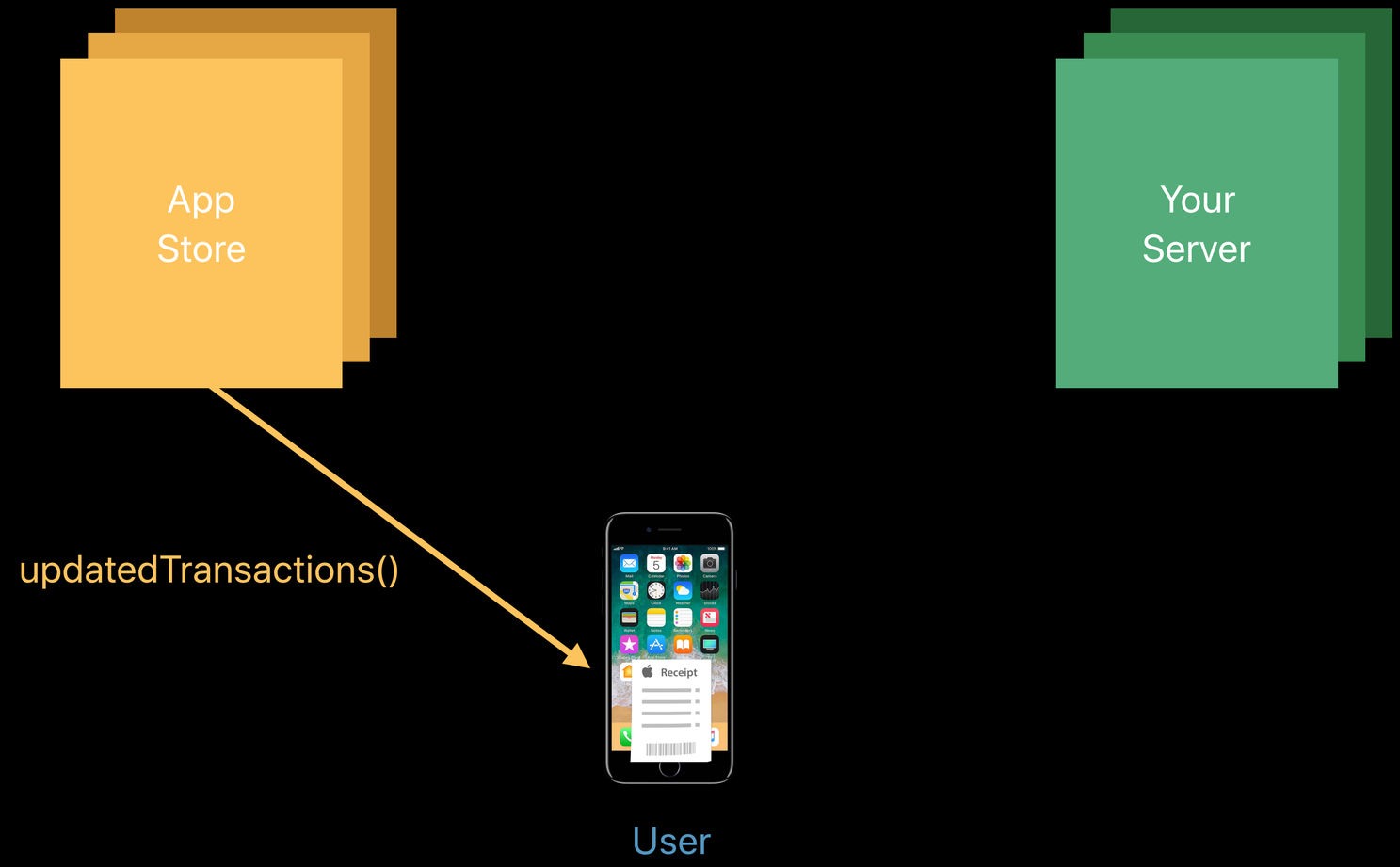
App Store

Your Server

User

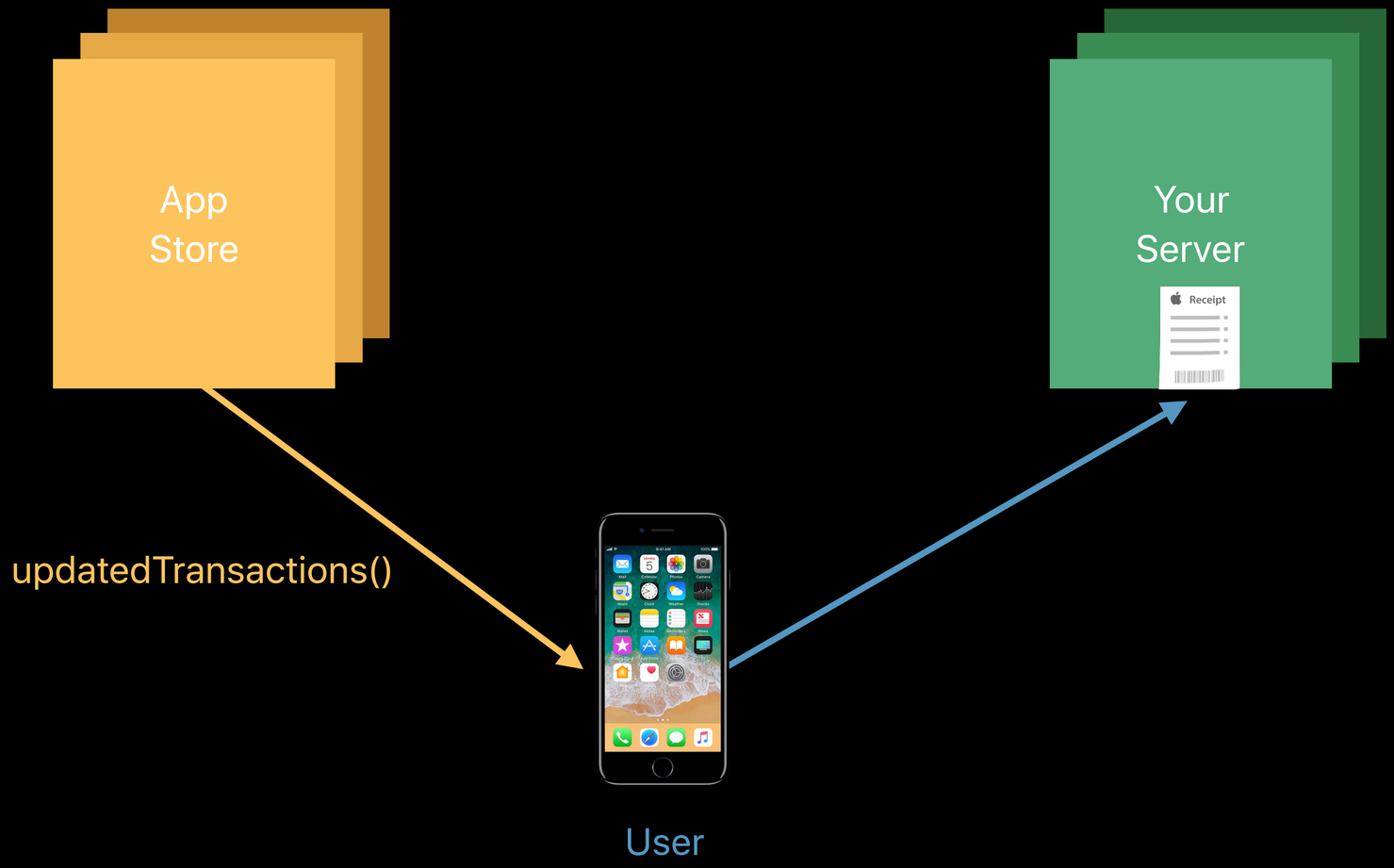
Server-Side Receipt Validation

Server-to-server validation



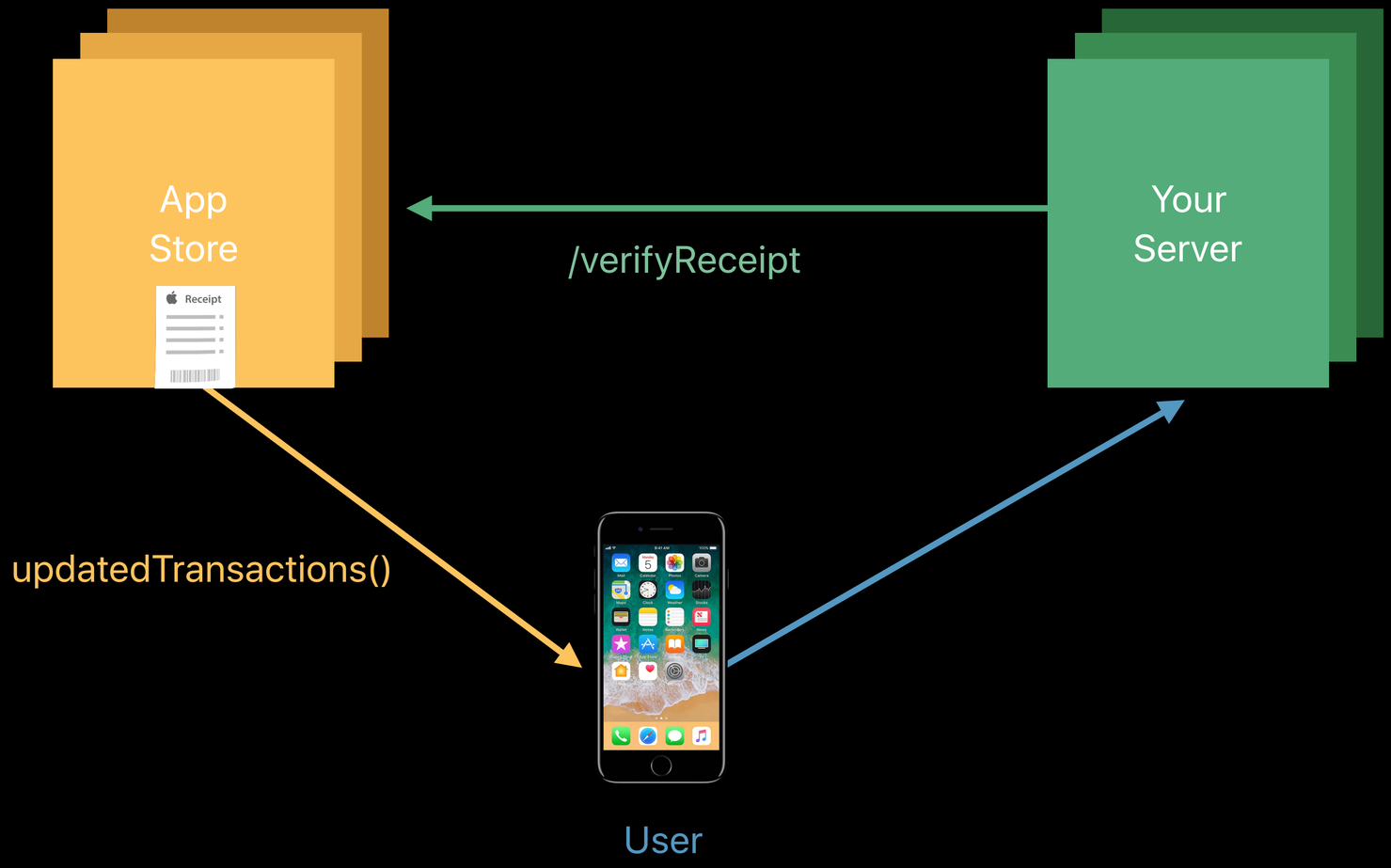
Server-Side Receipt Validation

Server-to-server validation



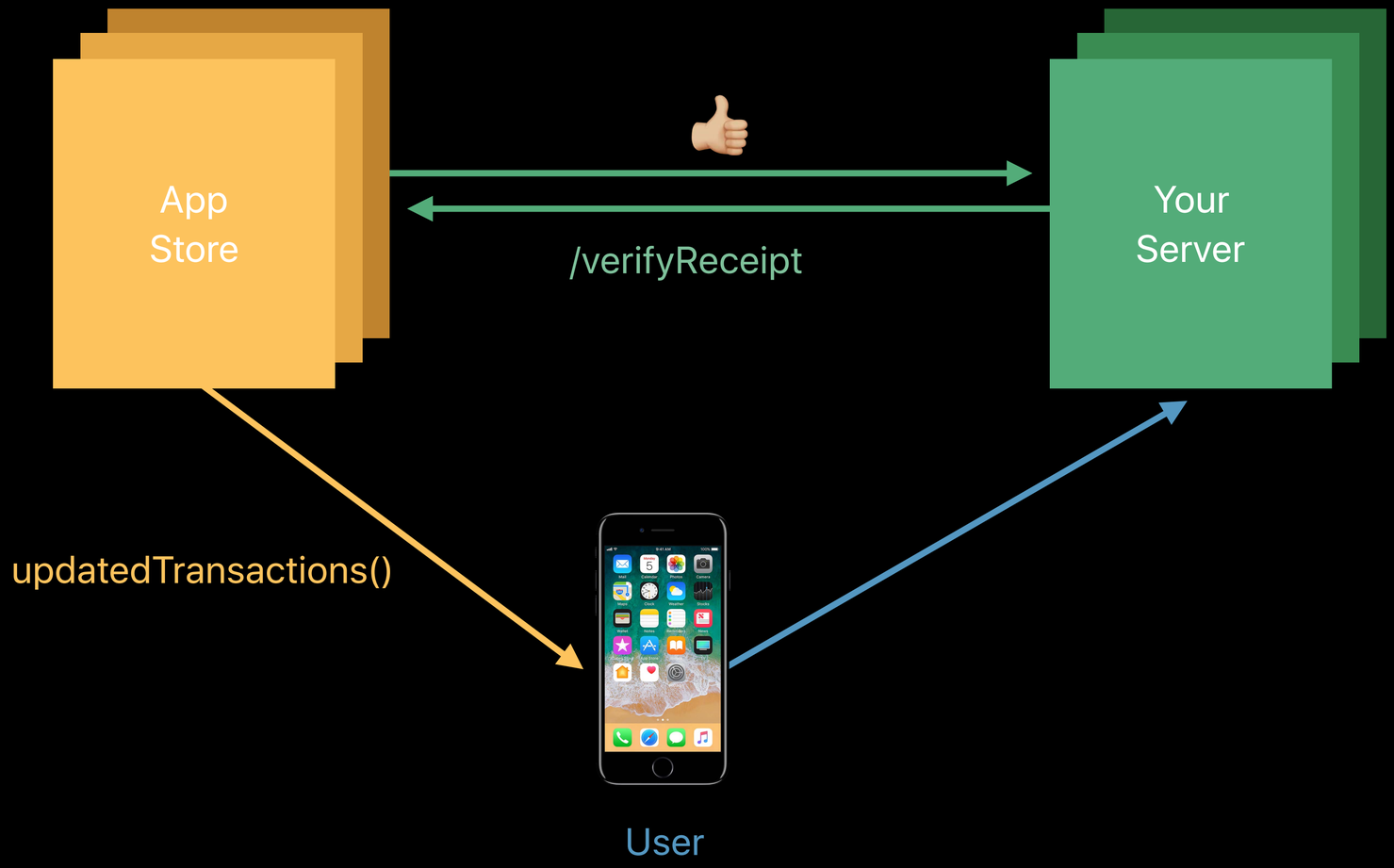
Server-Side Receipt Validation

Server-to-server validation



Server-Side Receipt Validation

Server-to-server validation



Server-Side Receipt Validation

Server-to-server validation

Allows your servers to validate the receipt

Your app sends the receipt to your servers

- Your server sends the receipt to the App Store
- Endpoint is `verifyReceipt`

Response is in JSON

- Returns status on whether receipt is valid or not

Server-Side Receipt Validation

Server-to-server validation

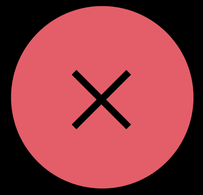
Allows your servers to validate the receipt

Your app sends the receipt to your servers

- Your server sends the receipt to the App Store
- Endpoint is `verifyReceipt`

Response is in JSON

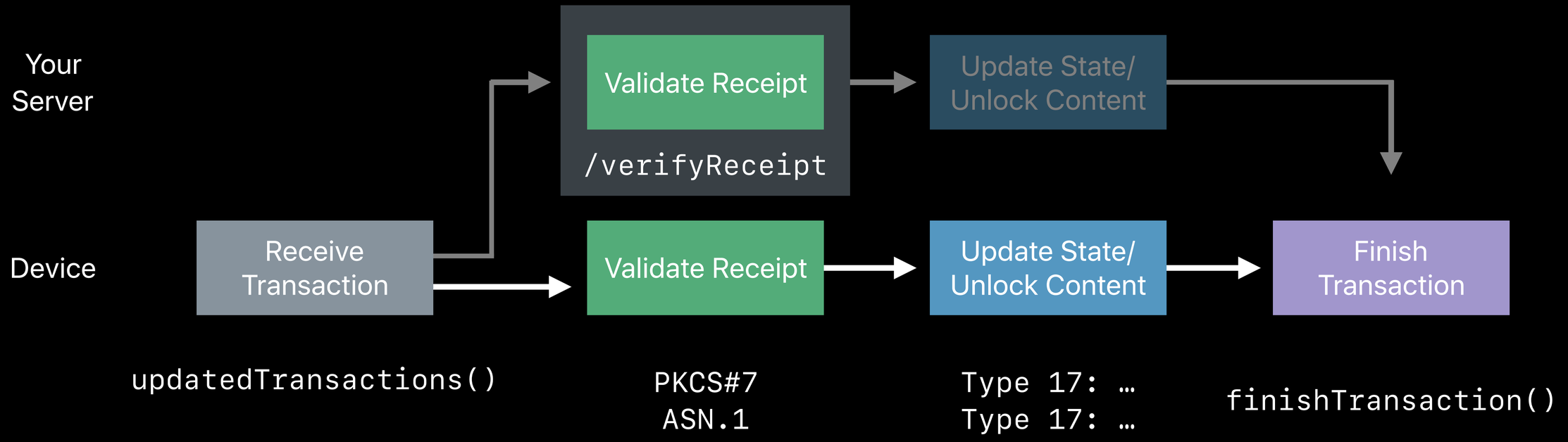
- Returns status on whether receipt is valid or not



Never send the receipt directly from your app to the App Store server

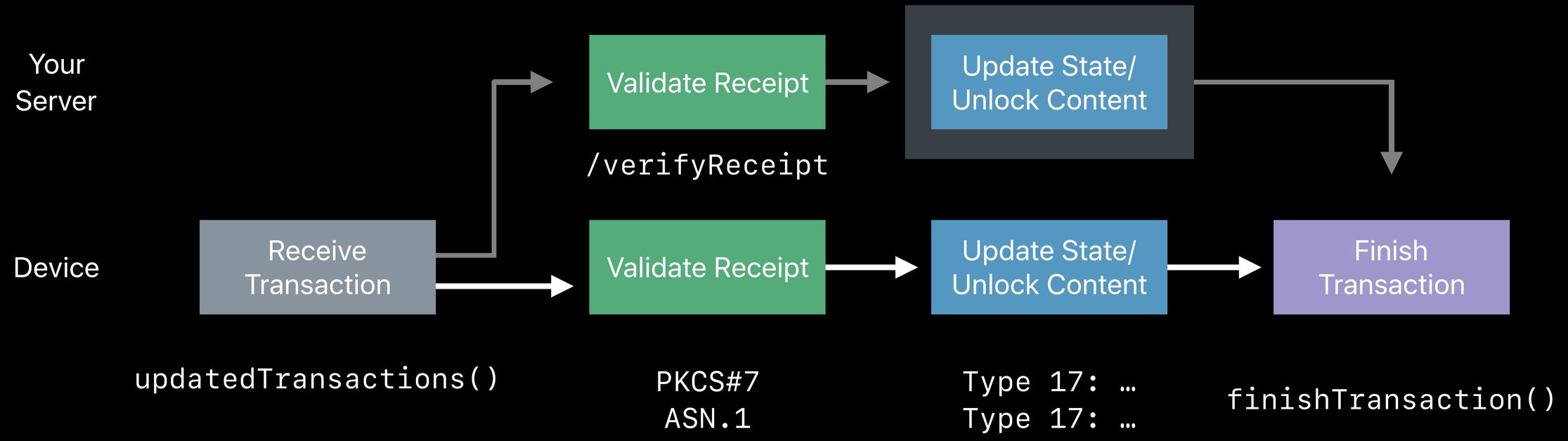
In-App Purchase Process

Processing transactions



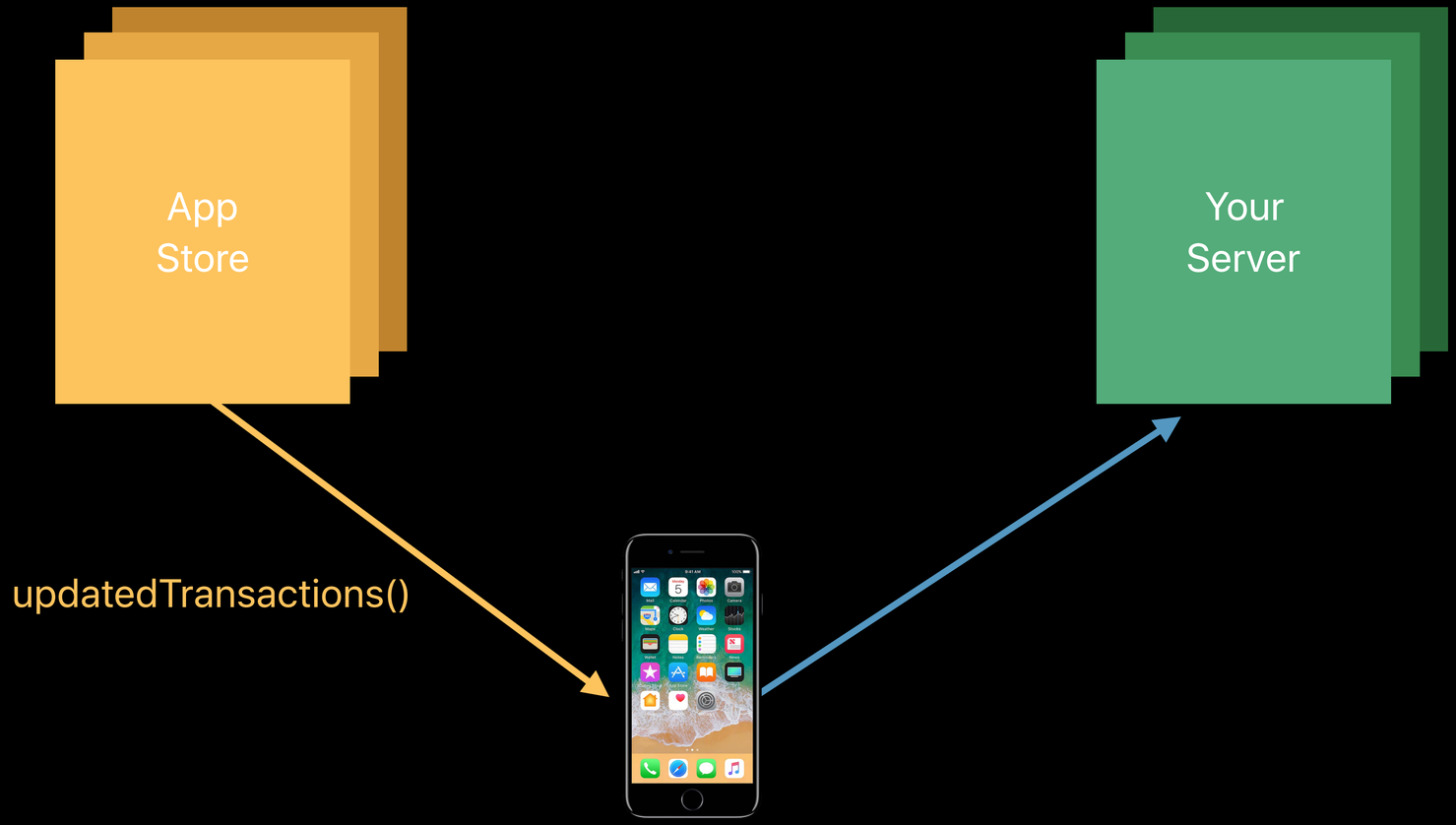
In-App Purchase Process

Processing transactions



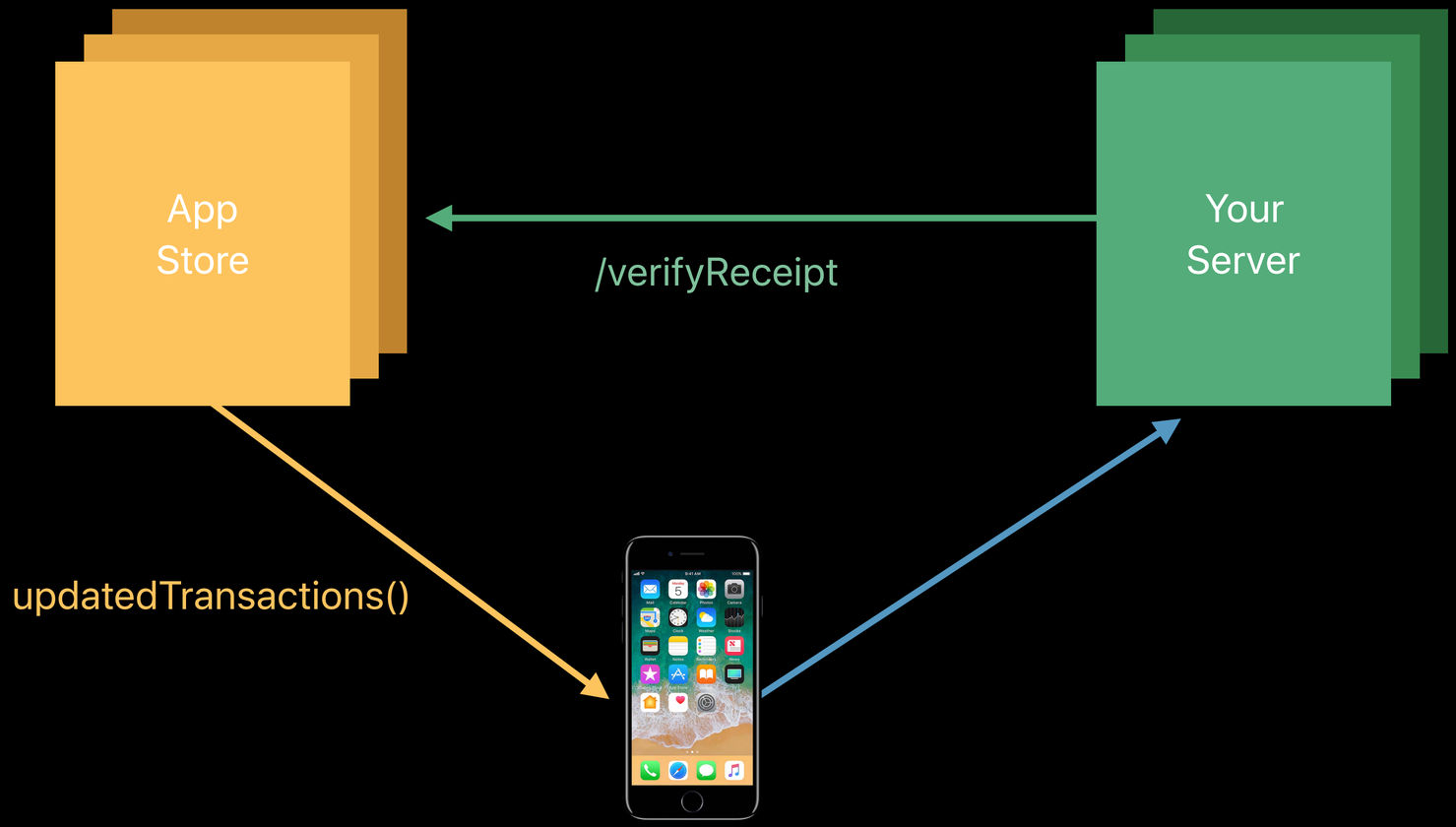
Server-Side In-App Purchase State

Verifying a purchase on your server



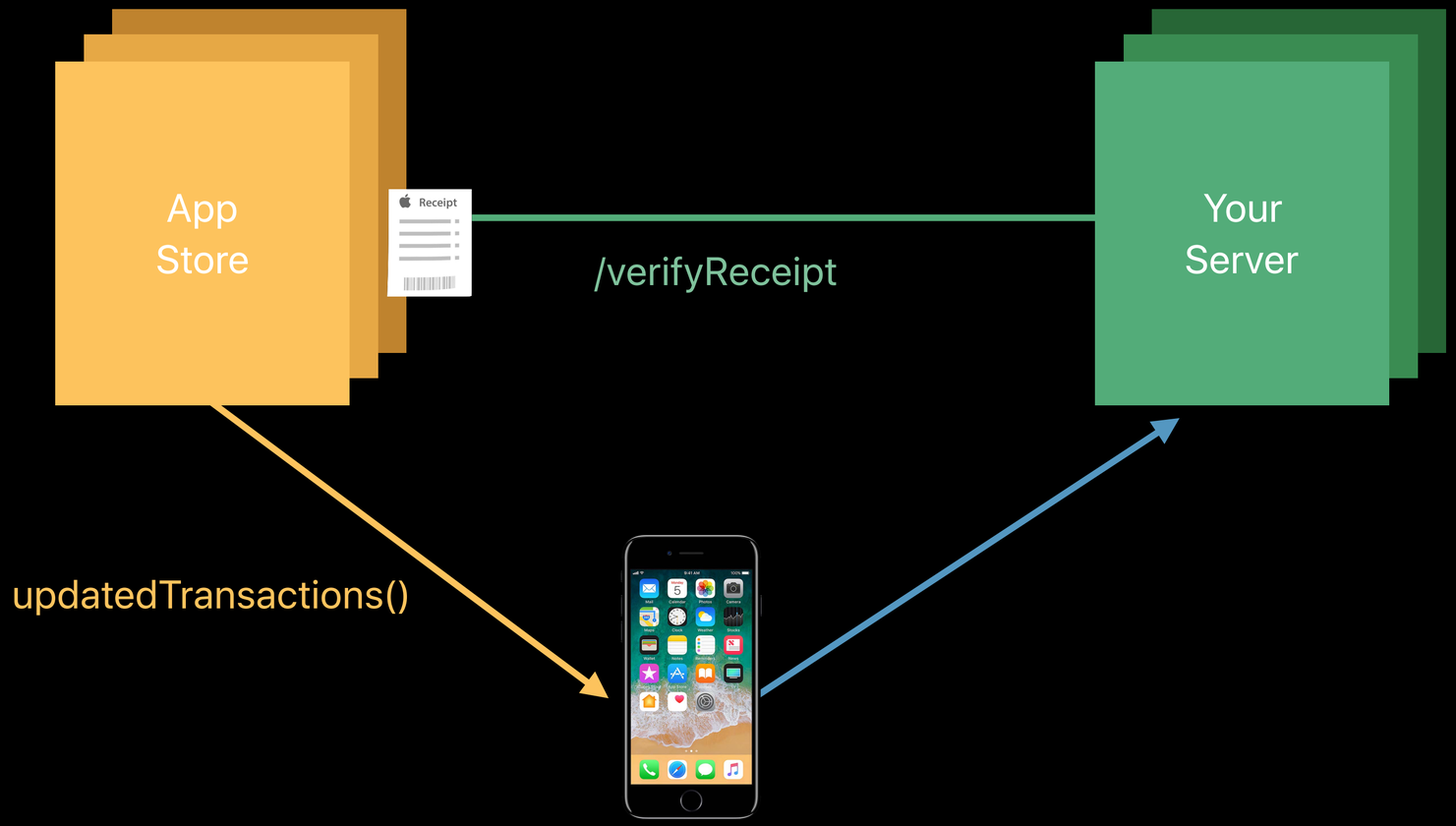
Server-Side In-App Purchase State

Verifying a purchase on your server



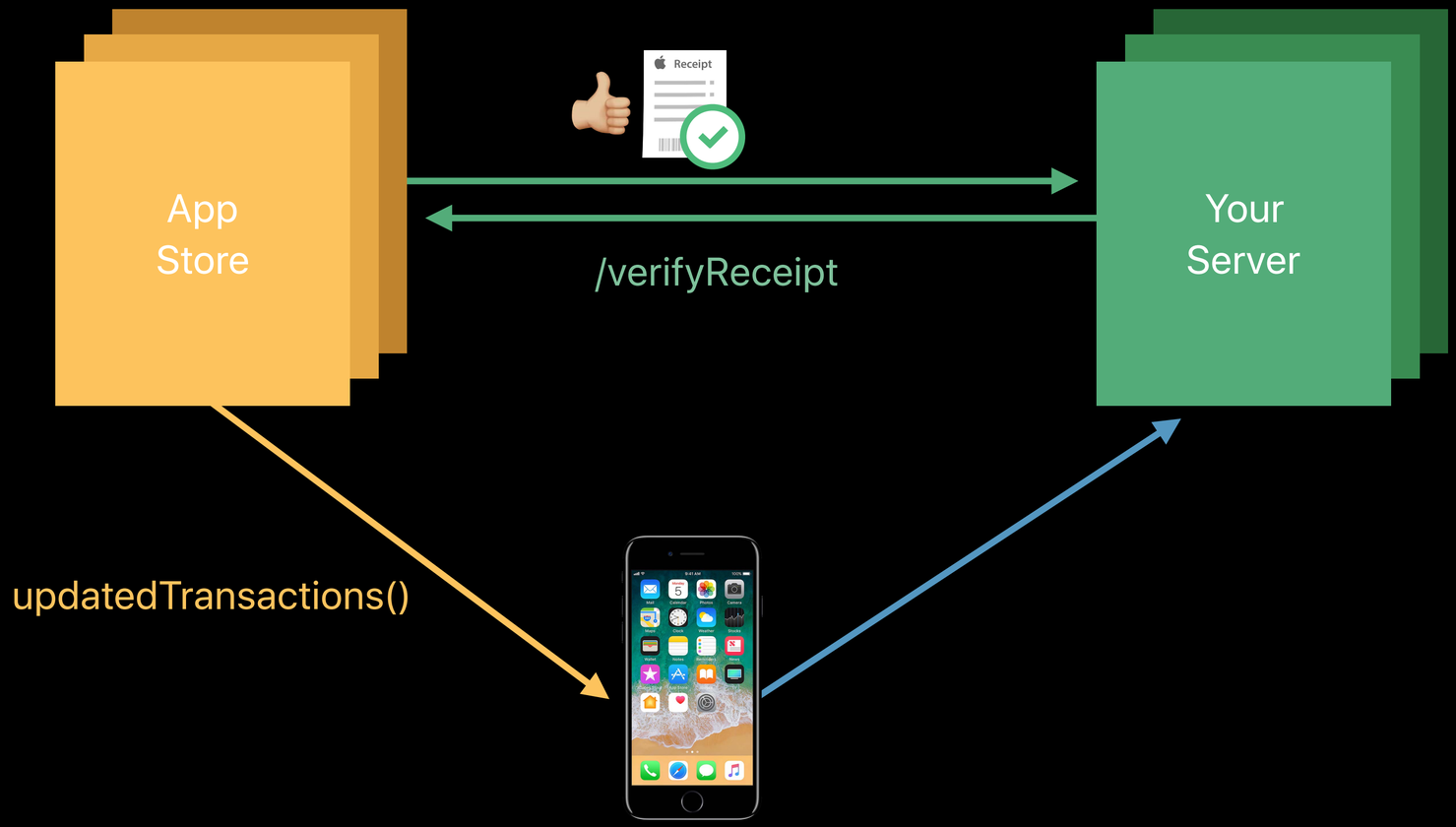
Server-Side In-App Purchase State

Verifying a purchase on your server



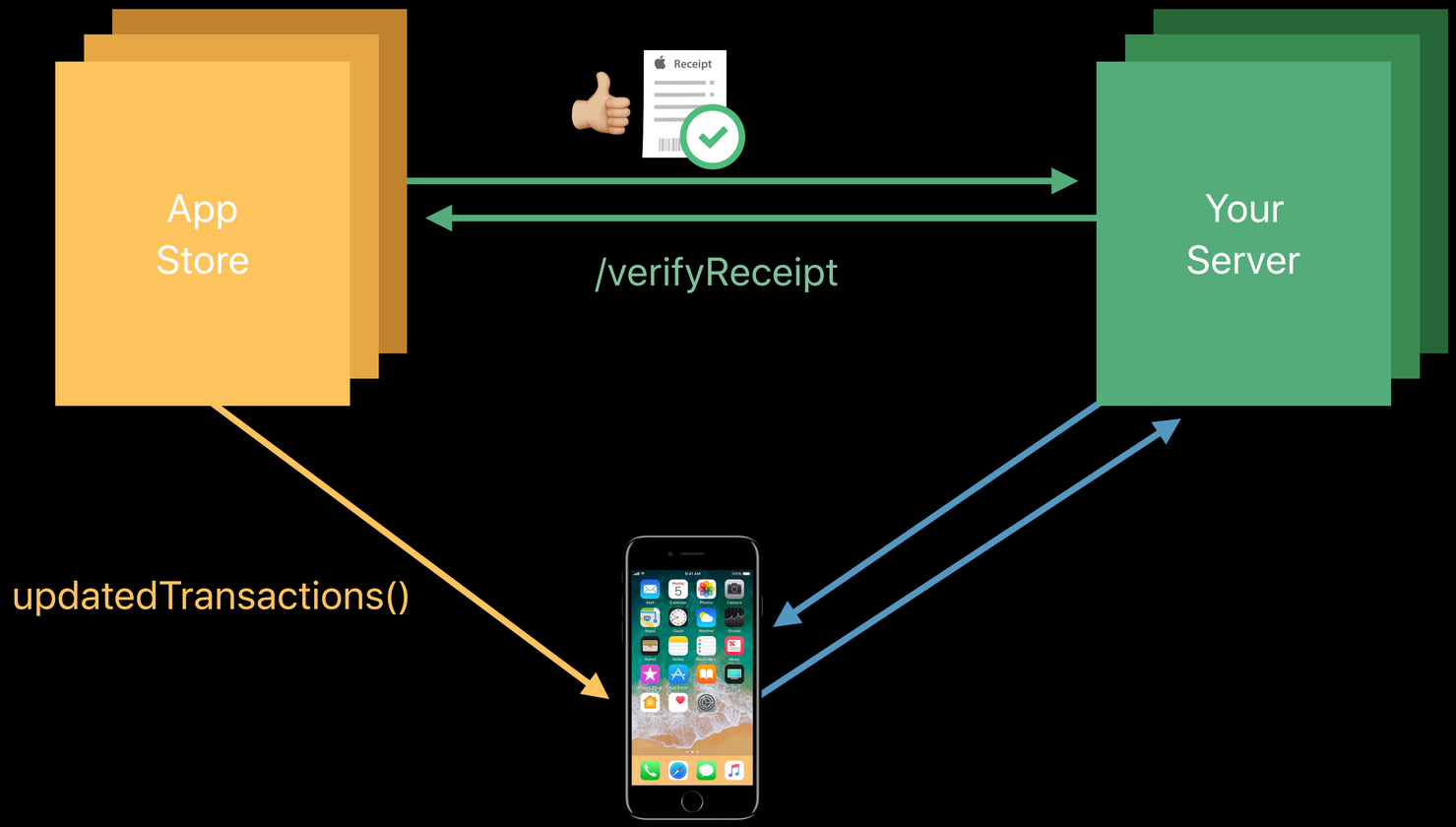
Server-Side In-App Purchase State

Verifying a purchase on your server



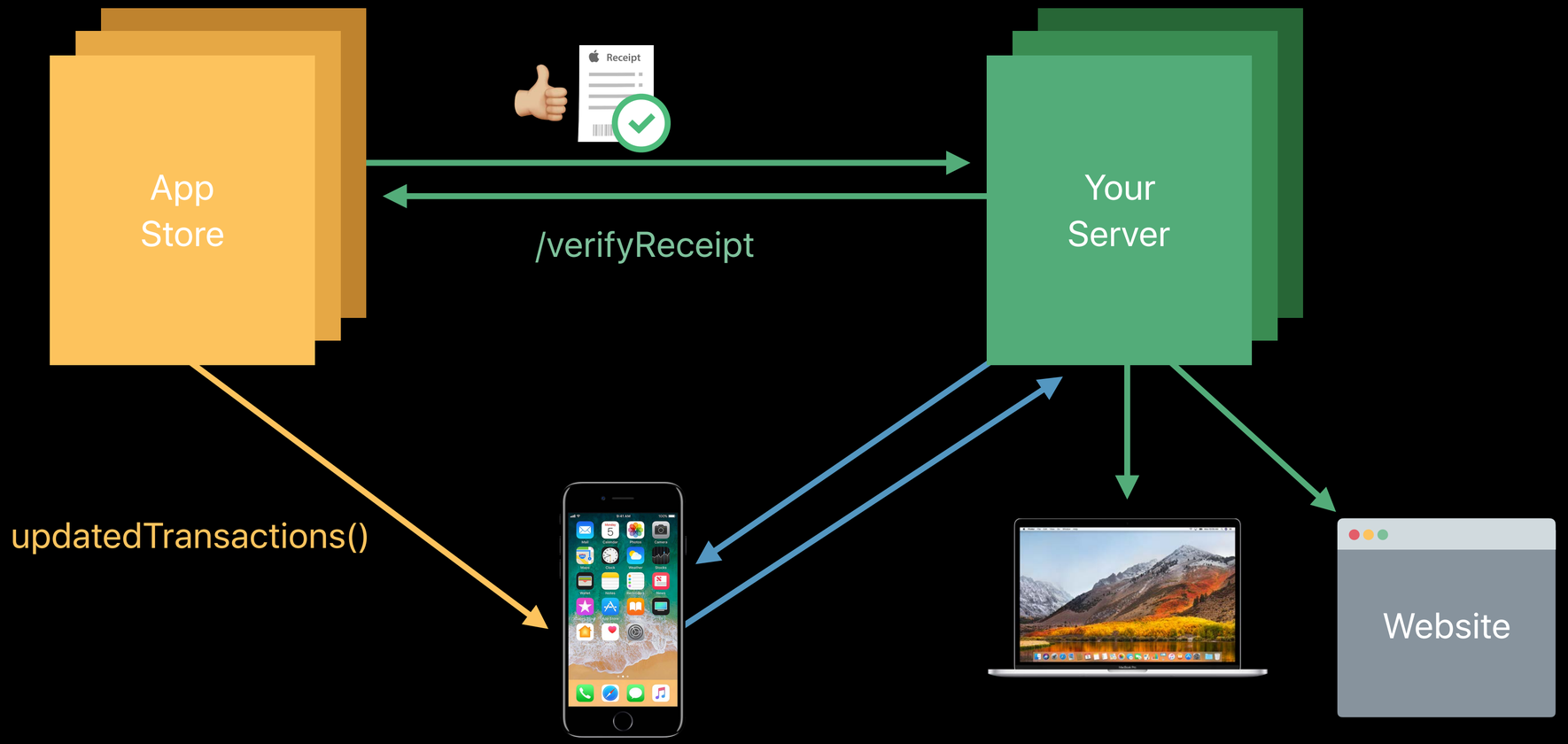
Server-Side In-App Purchase State

Verifying a purchase on your server



Server-Side In-App Purchase State

Verifying a purchase on your server



Server-Side In-App Purchase State

Unlocking content on your server

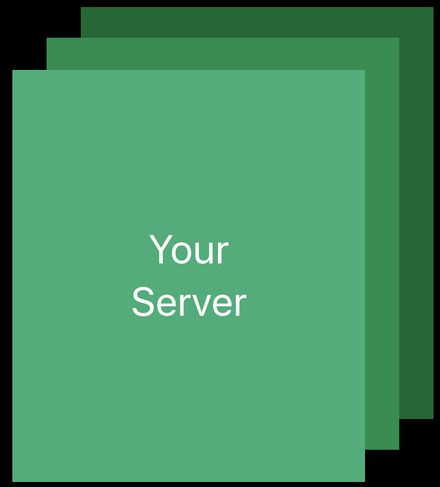
In addition to receipt validity, `verifyReceipt` returns

- Latest decoded application receipt

Contains array of in-app purchase transactions

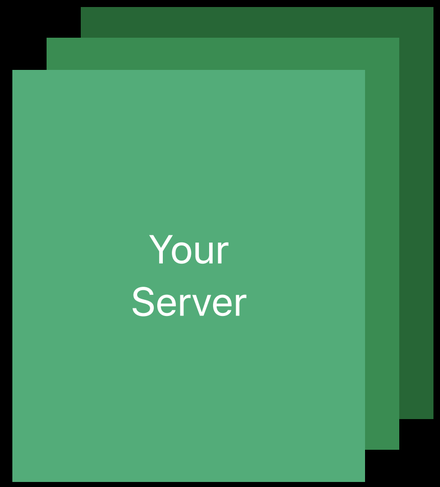
- Verify product in the `updatedTransactions` callback on device is present in a transaction

Tell the device to `finishTransaction()`



Server-Side In-App Purchase State

Does the user have an active subscription?

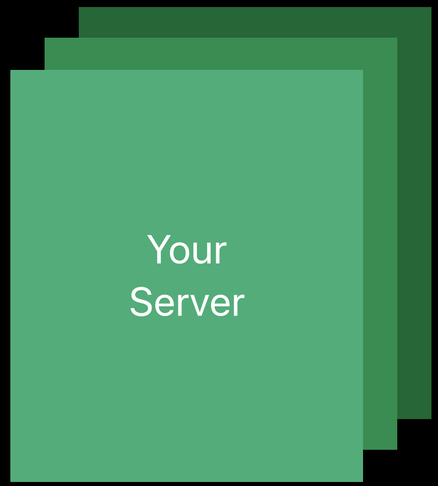


Server-Side In-App Purchase State

Does the user have an active subscription?

Filter transactions by `originalTransactionId`

- Matches the first in-app purchase for that subscription



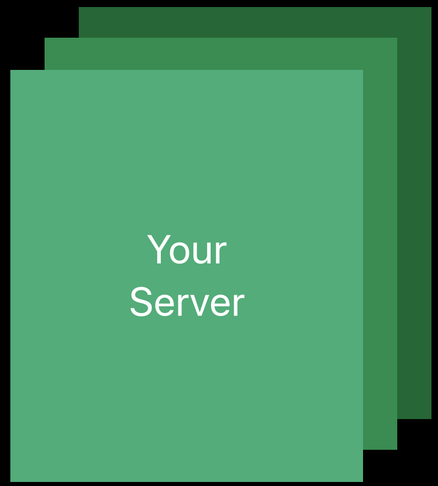
Server-Side In-App Purchase State

Does the user have an active subscription?

Filter transactions by `originalTransactionId`

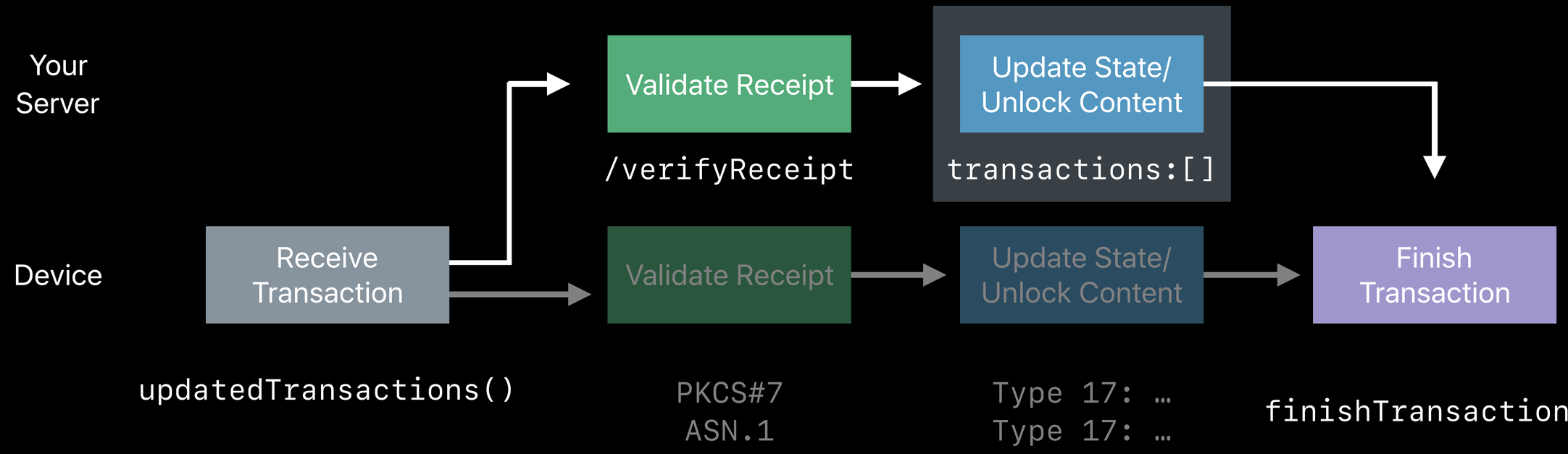
- Matches the first in-app purchase for that subscription

Check matching transactions for latest expiry date



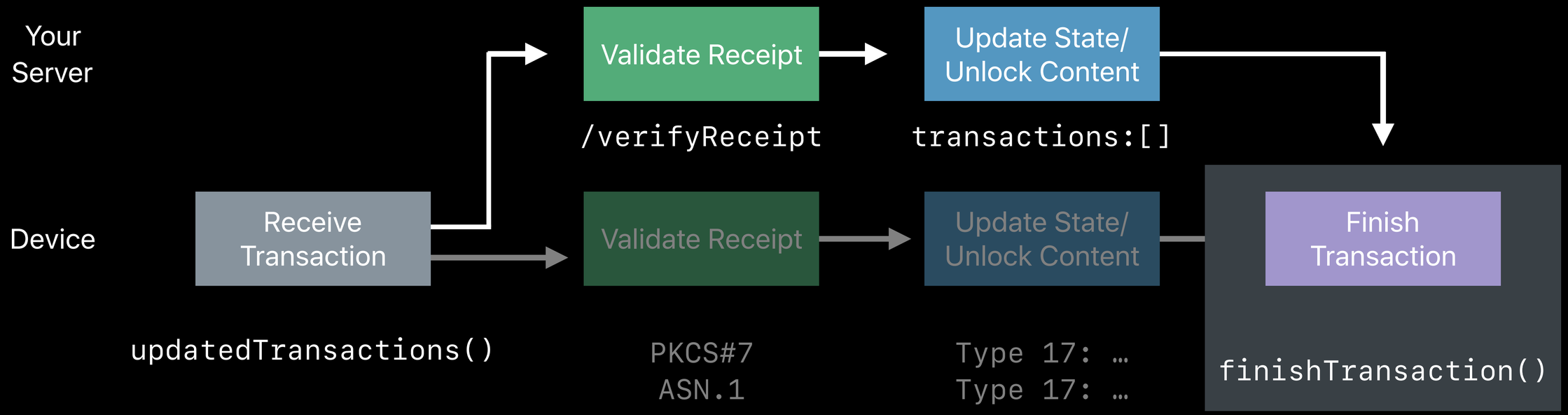
In-App Purchase Process

Processing transactions



In-App Purchase Process

Processing transactions



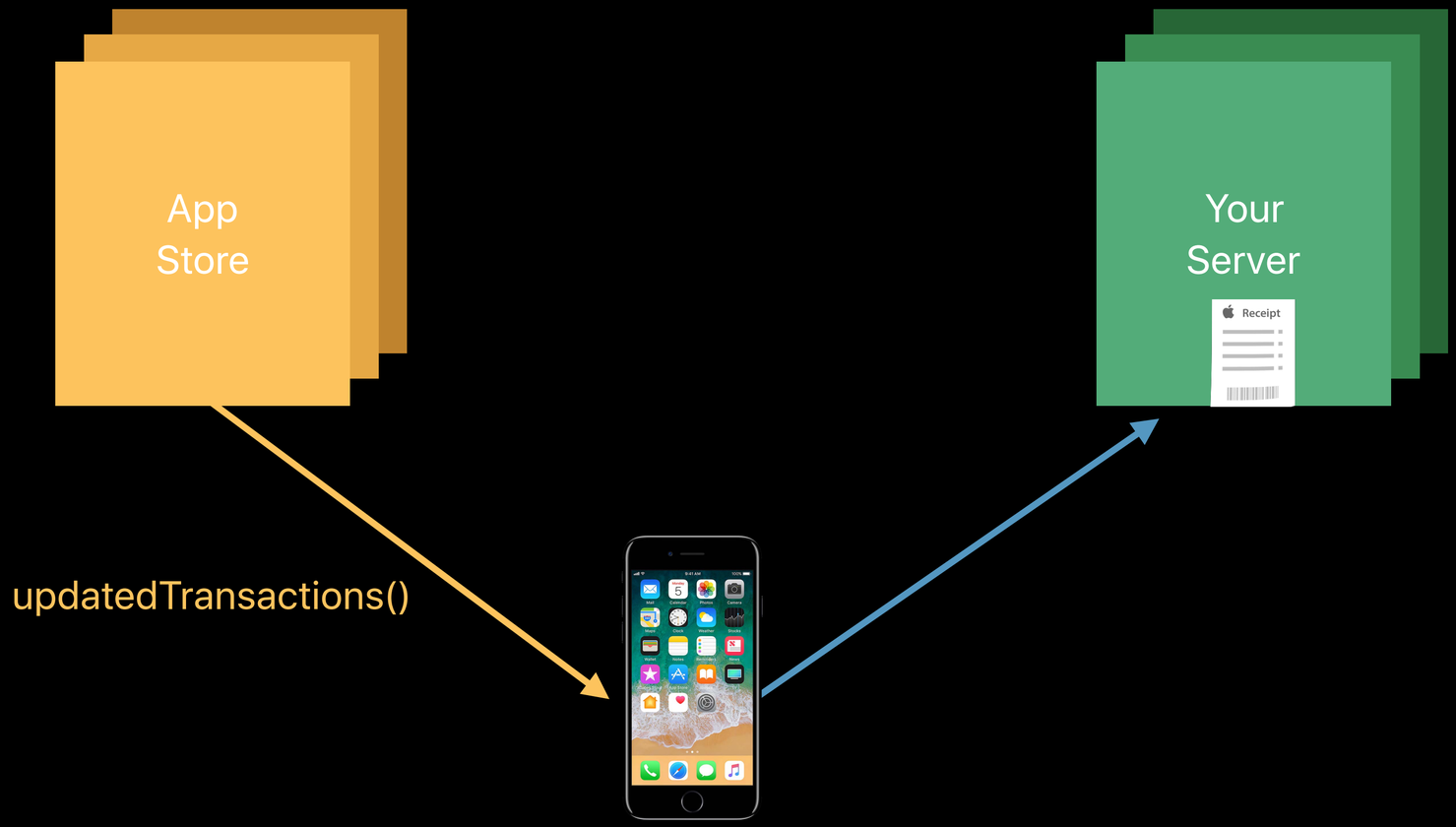
Server-Side Subscription State

App Store

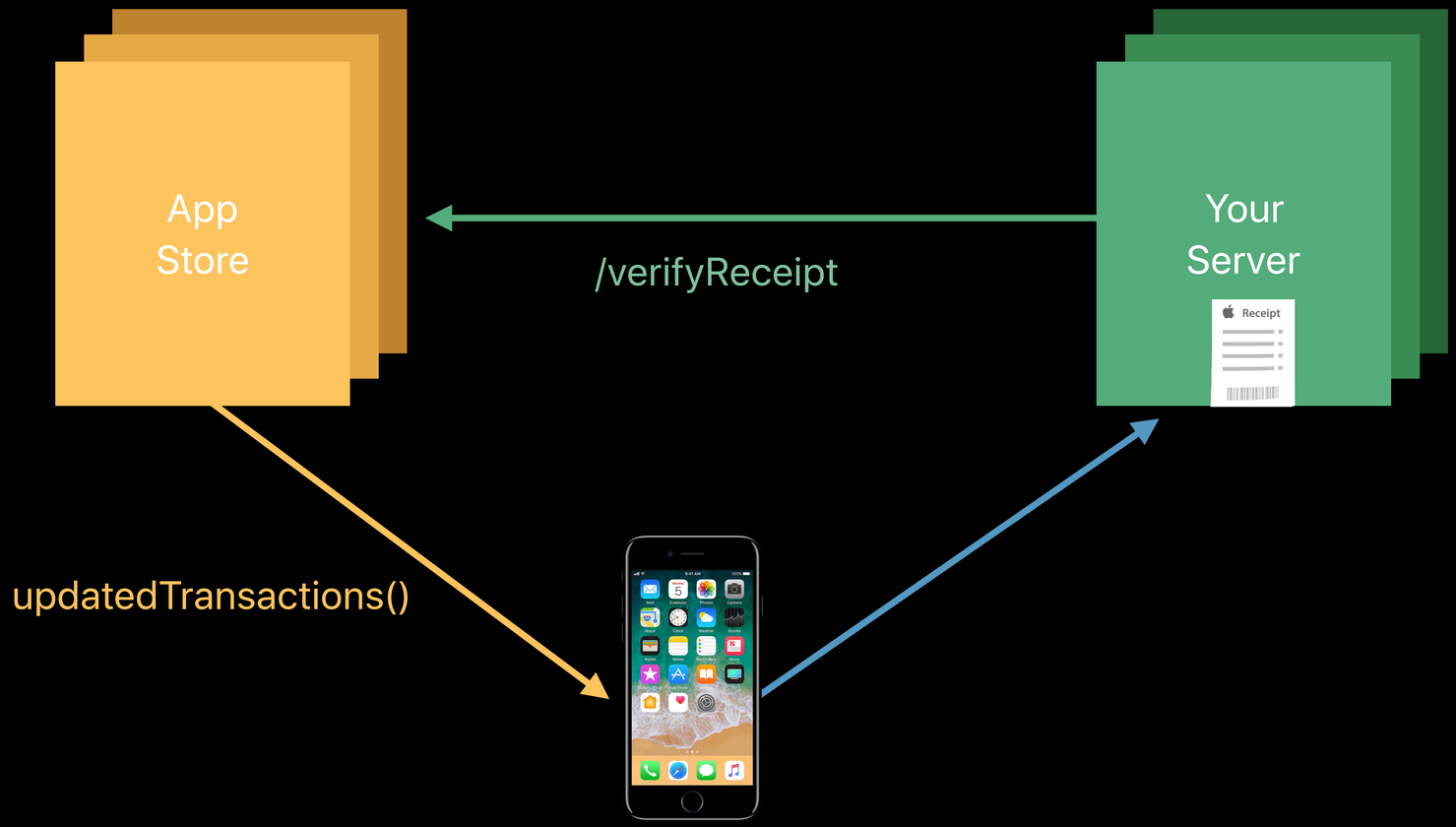
Your Server

updatedTransactions()

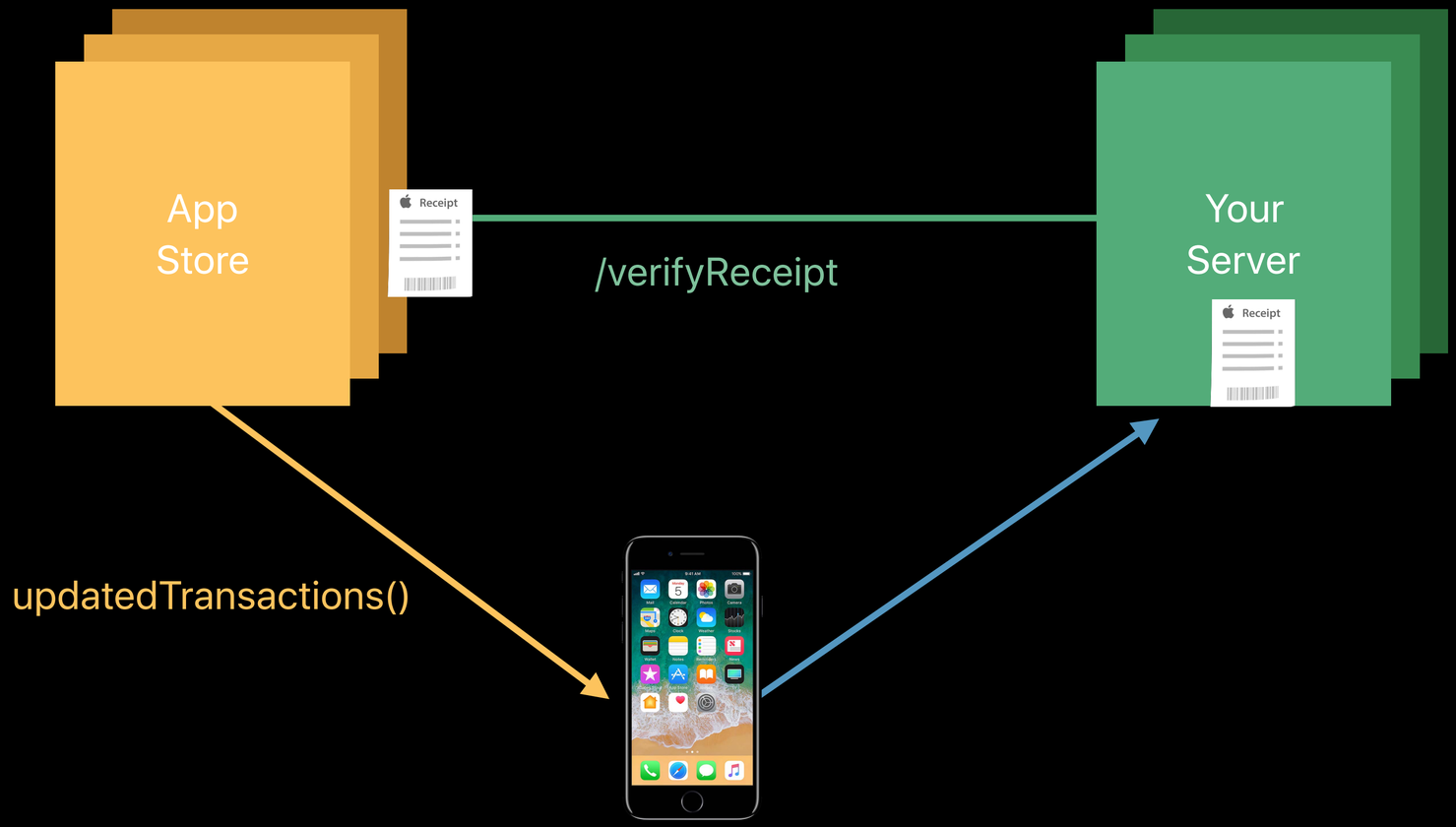
Server-Side Subscription State



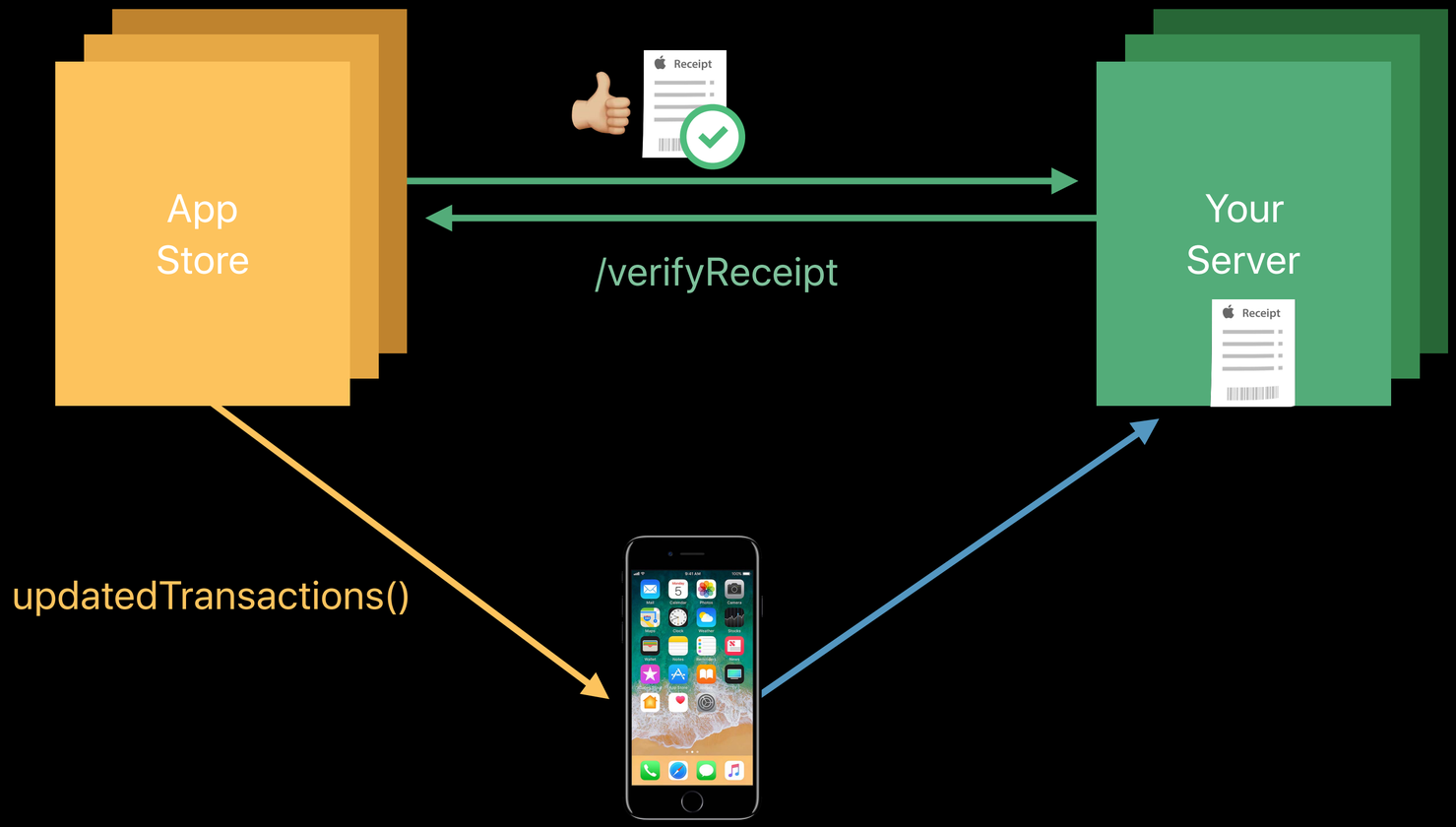
Server-Side Subscription State



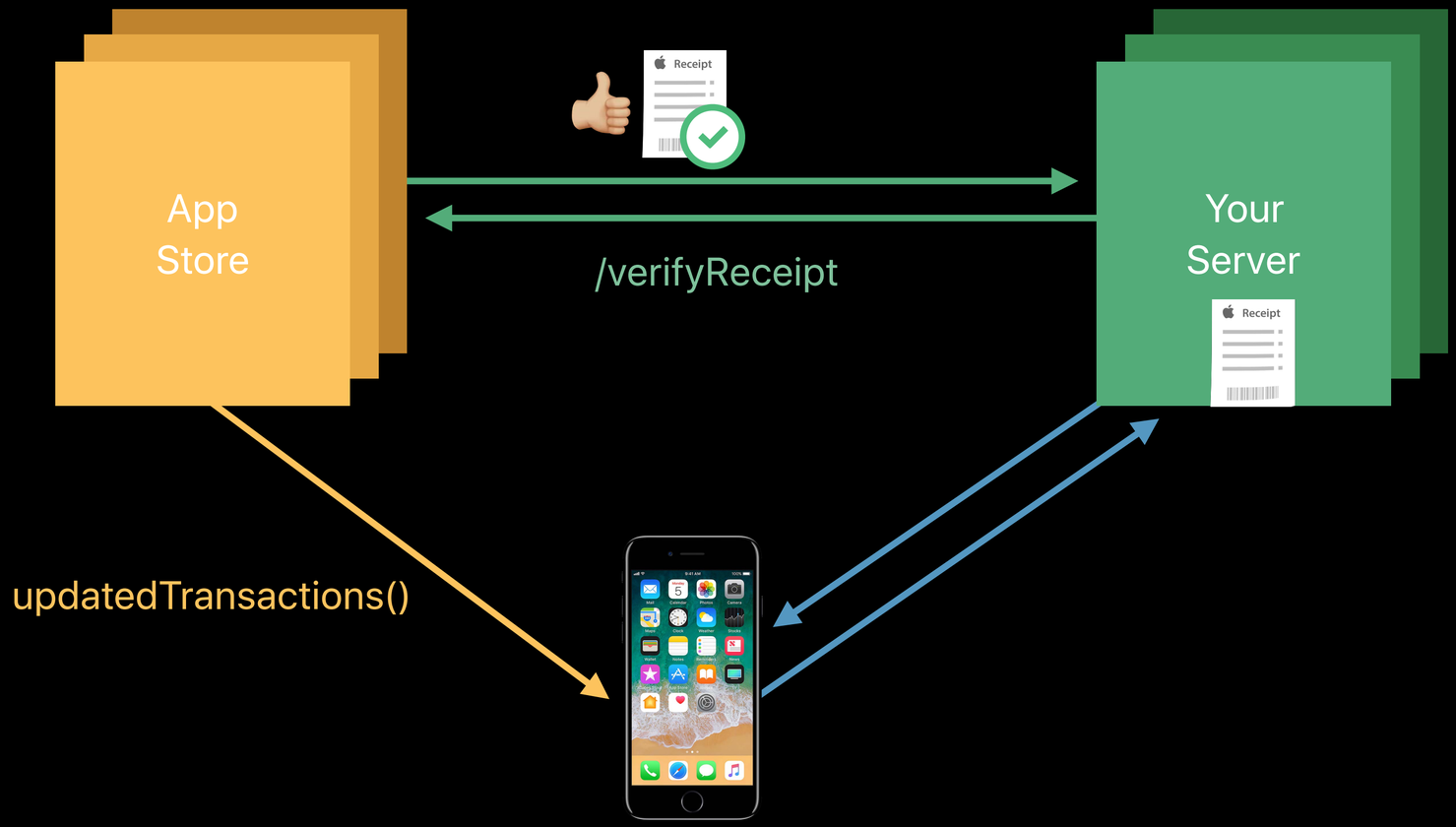
Server-Side Subscription State



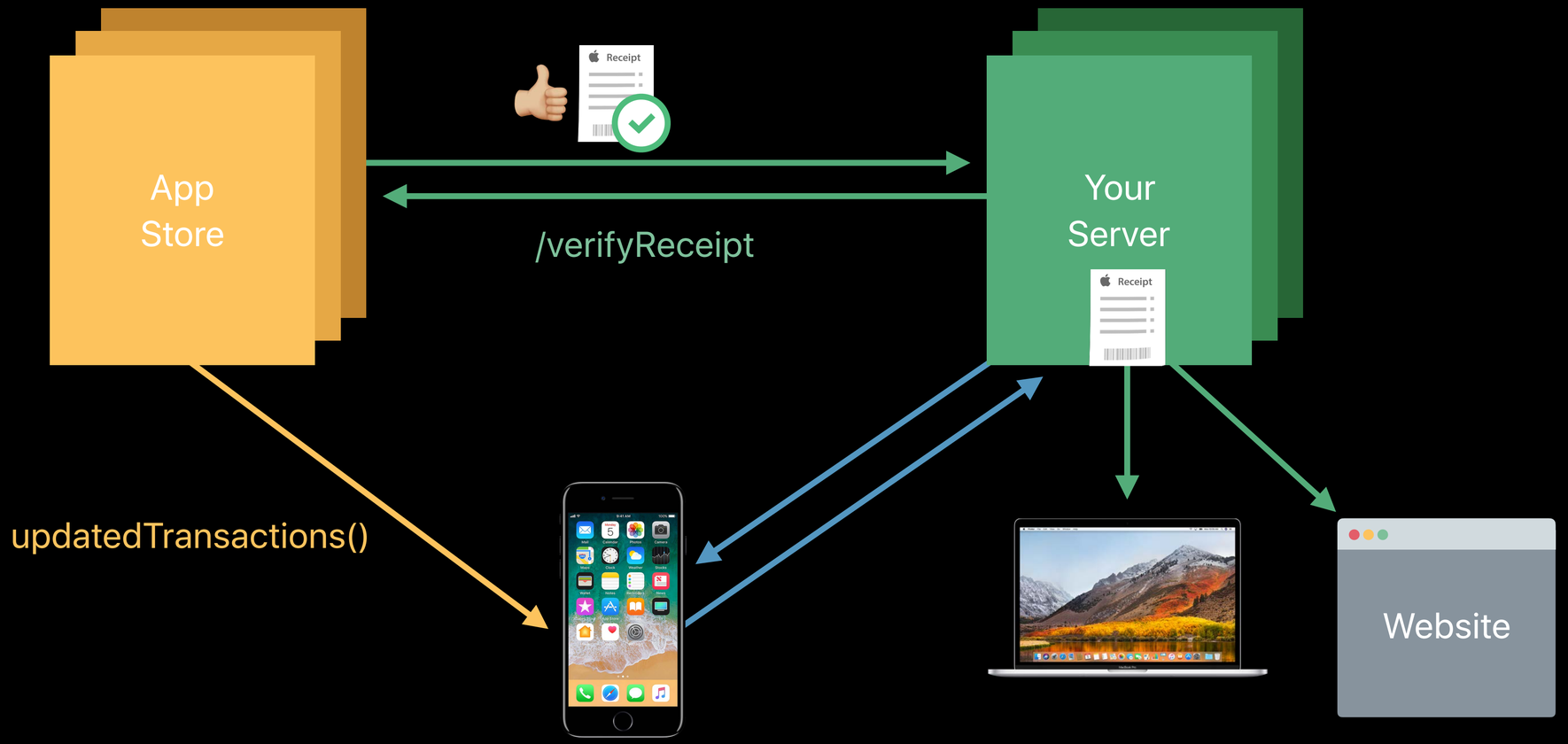
Server-Side Subscription State



Server-Side Subscription State

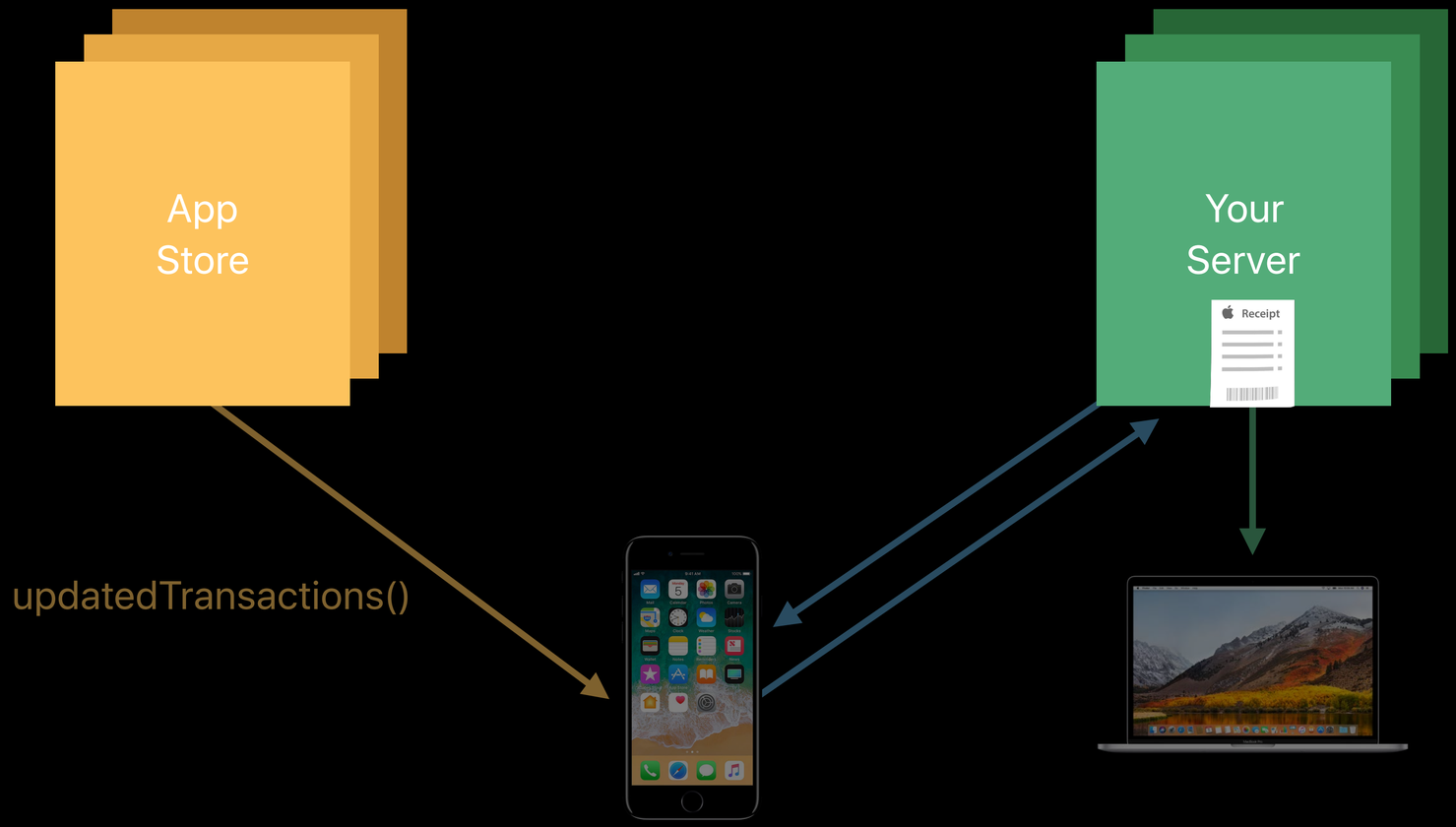


Server-Side Subscription State



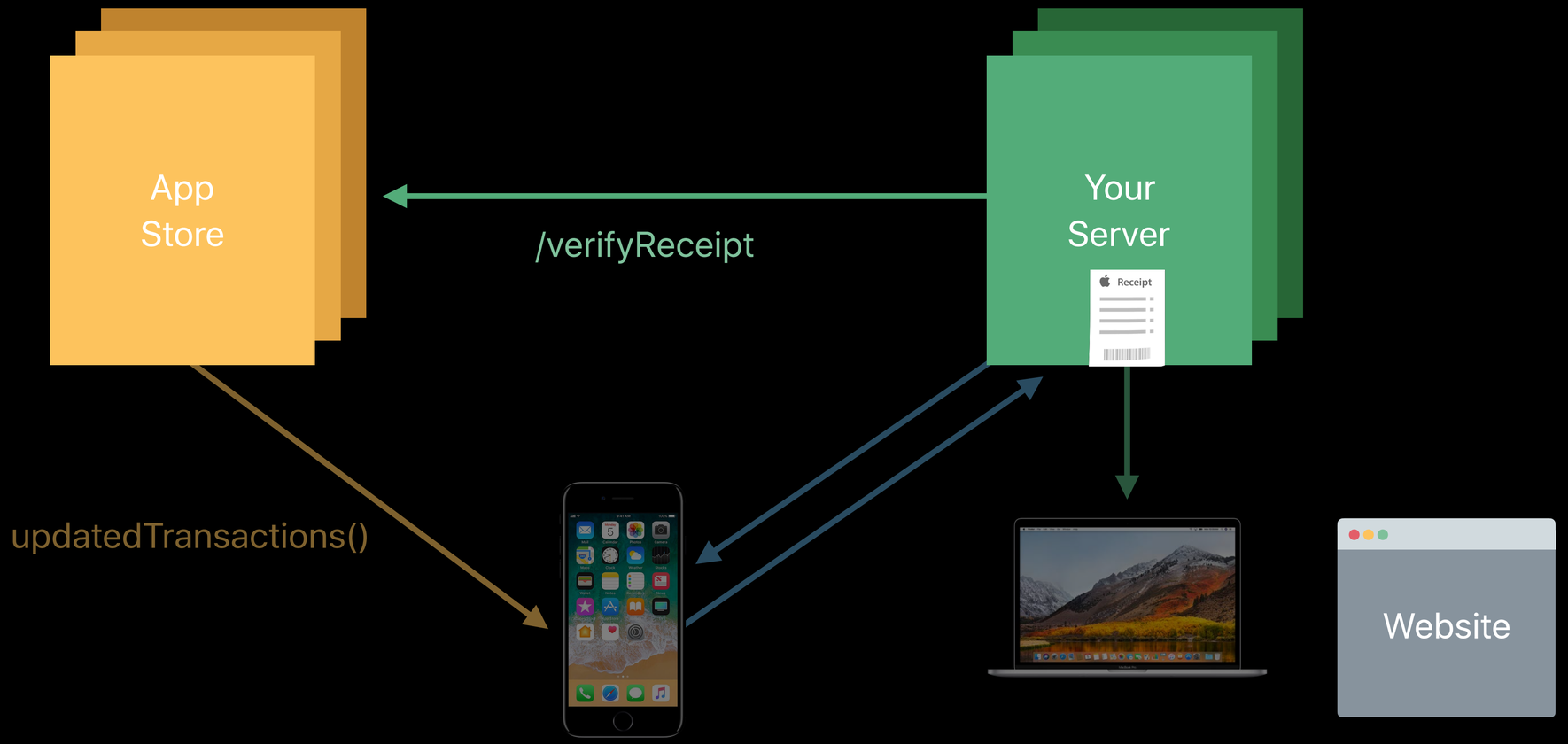
Server-Side Subscription State

Polling for subscription state



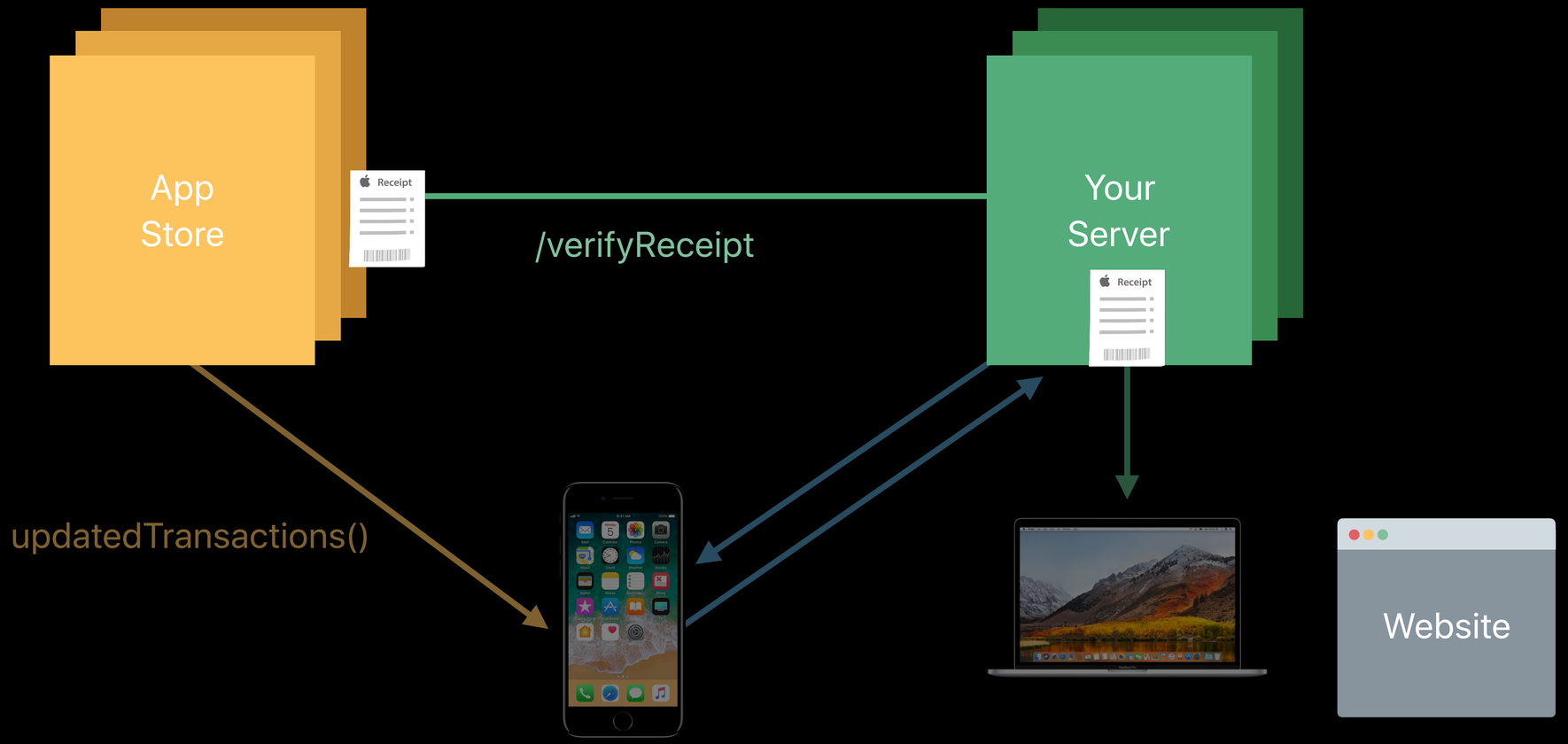
Server-Side Subscription State

Polling for subscription state



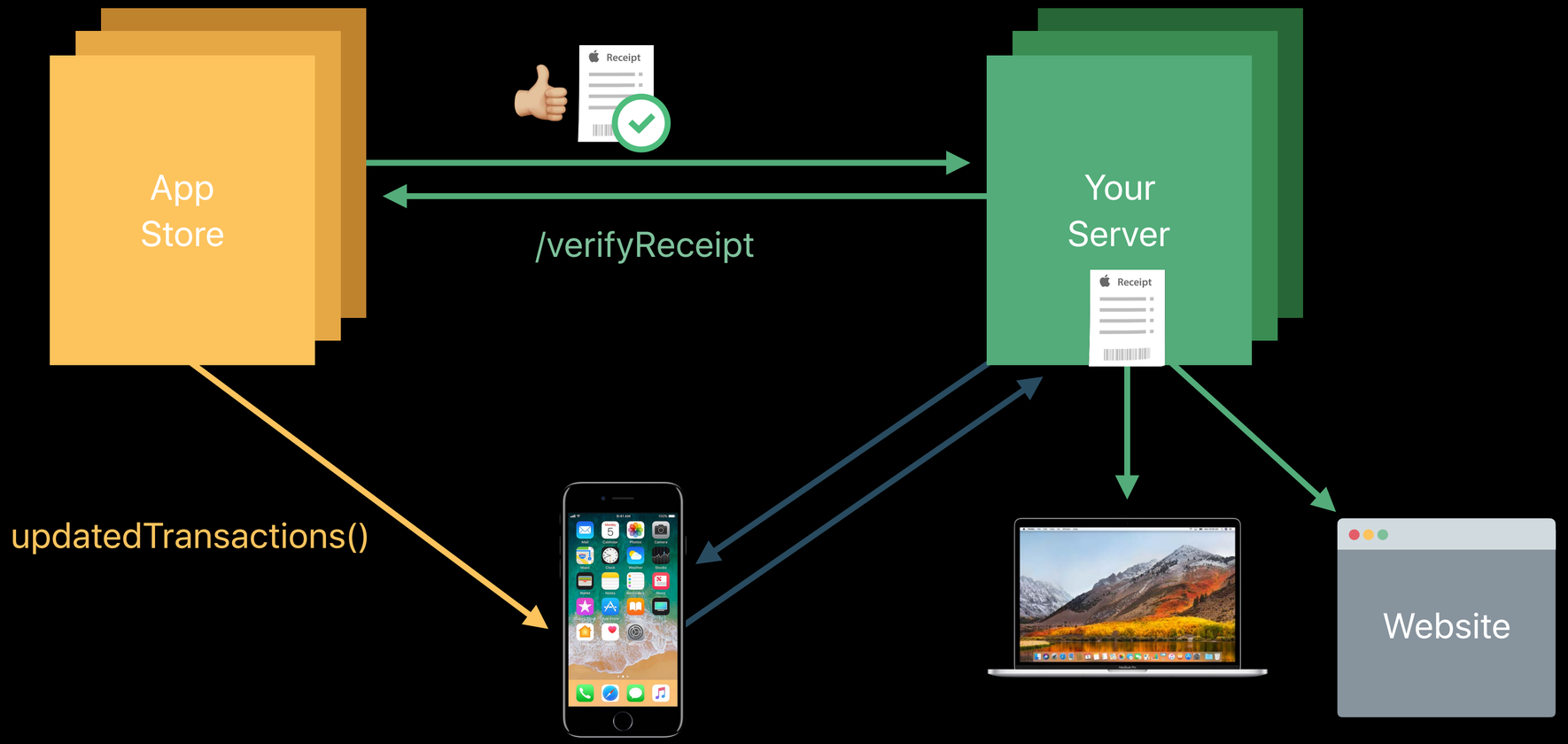
Server-Side Subscription State

Polling for subscription state



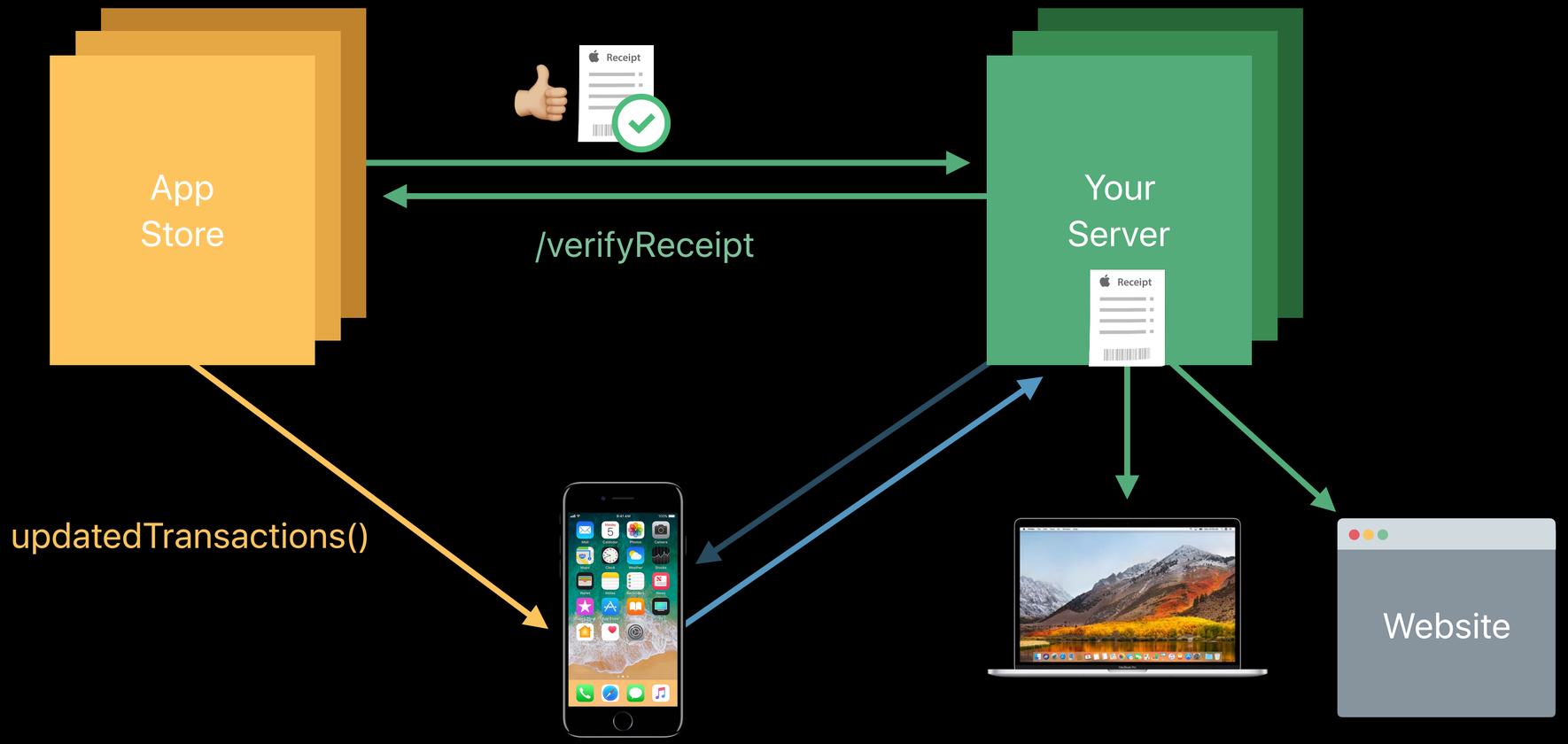
Server-Side Subscription State

Polling for subscription state



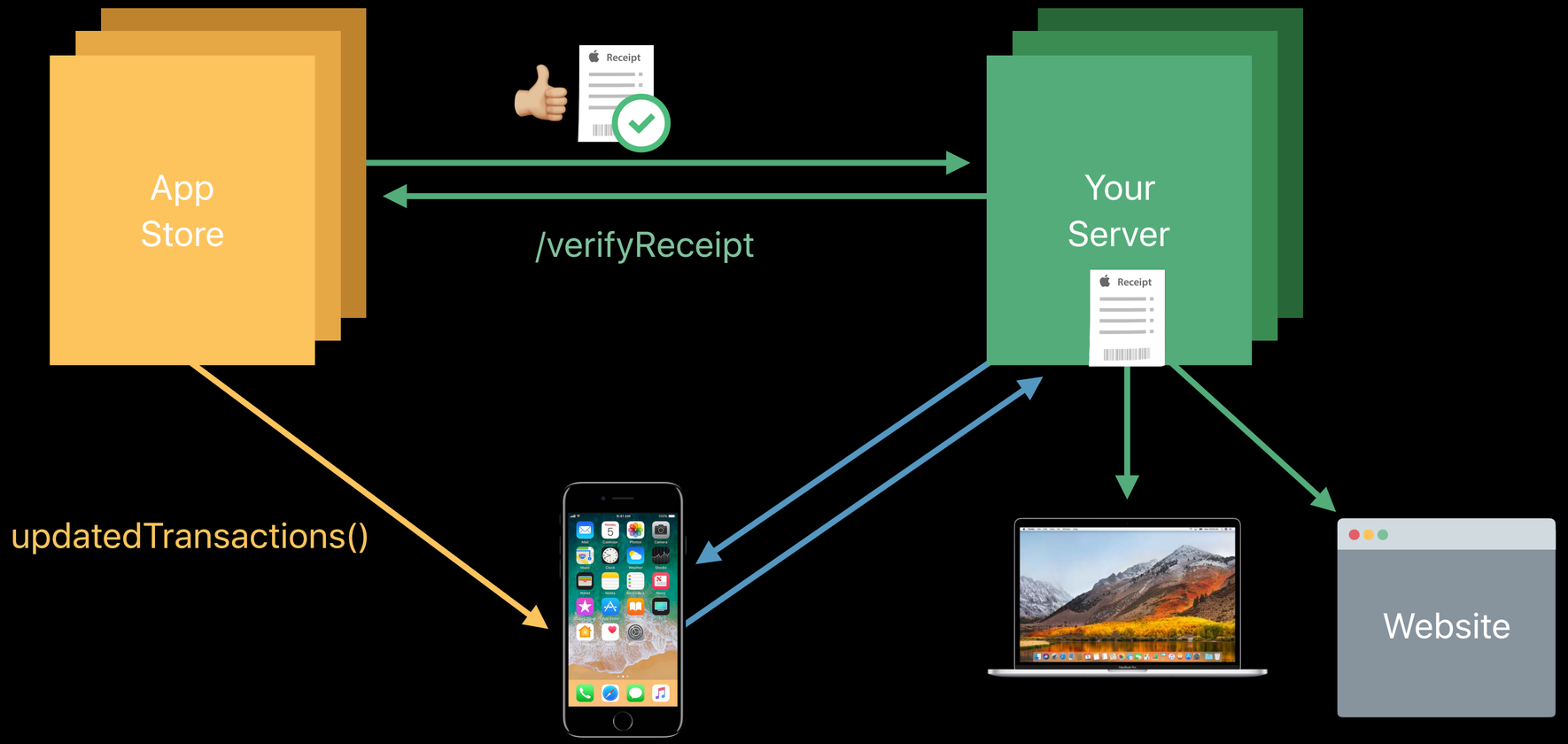
Server-Side Subscription State

Polling for subscription state



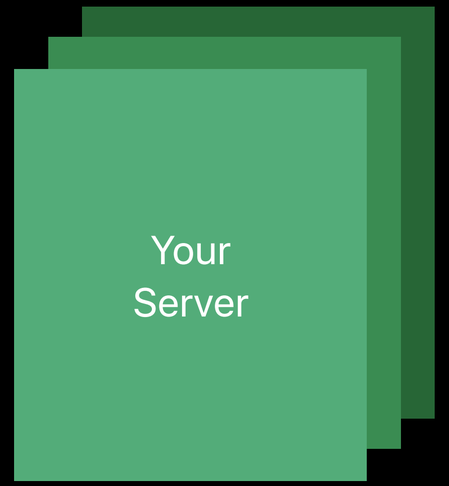
Server-Side Subscription State

Polling for subscription state



Server-Side In-App Purchase State

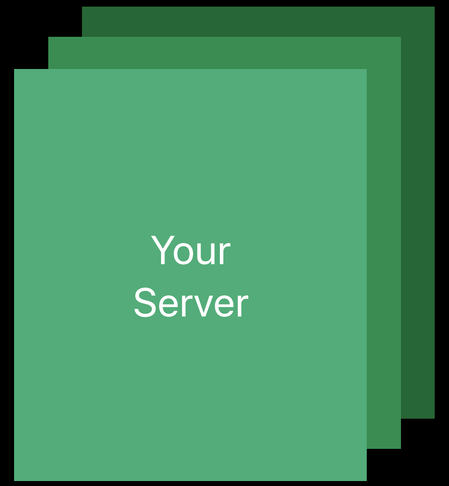
For auto-renewable subscriptions



Server-Side In-App Purchase State

For auto-renewable subscriptions

Treat the receipt data like a “token”

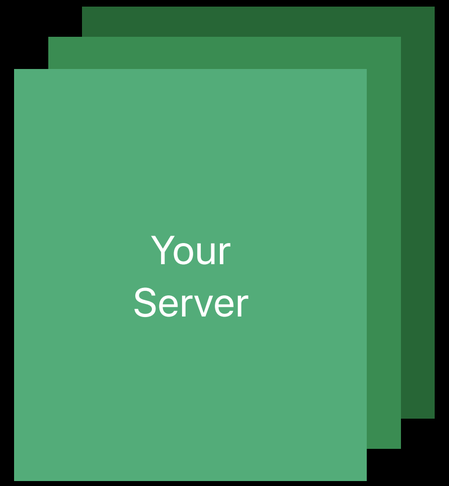


Server-Side In-App Purchase State

For auto-renewable subscriptions

Treat the receipt data like a “token”

- Store on your own server

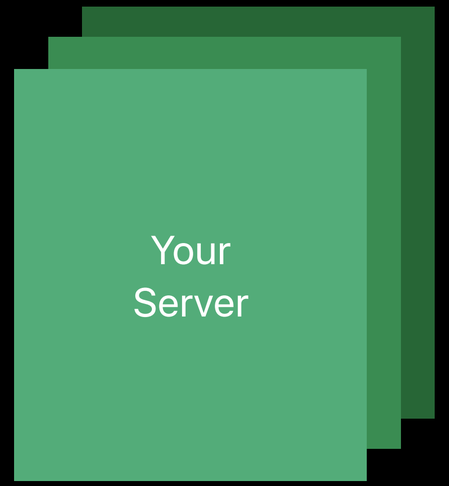


Server-Side In-App Purchase State

For auto-renewable subscriptions

Treat the receipt data like a “token”

- Store on your own server
- Same binary data, multiple times



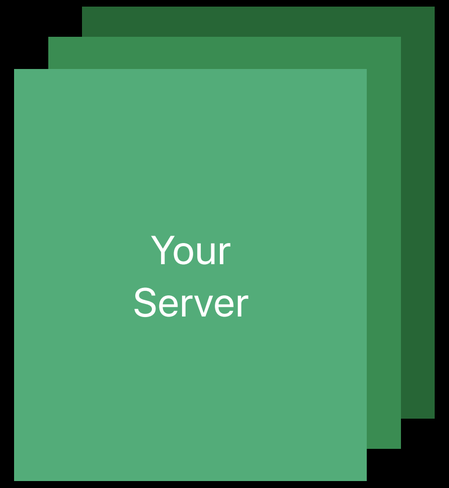
Server-Side In-App Purchase State

For auto-renewable subscriptions

Treat the receipt data like a “token”

- Store on your own server
- Same binary data, multiple times

Propagate subscription state across devices



Server-Side In-App Purchase State

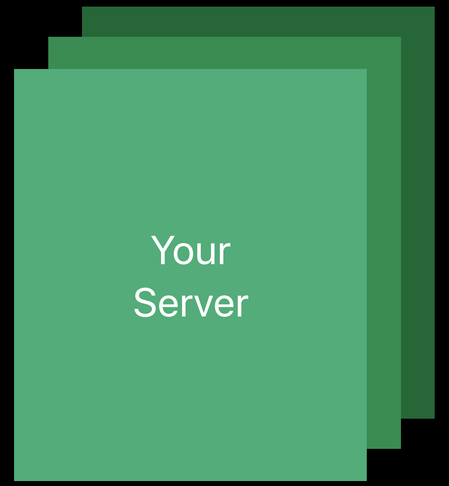
For auto-renewable subscriptions

Treat the receipt data like a “token”

- Store on your own server
- Same binary data, multiple times

Propagate subscription state across devices

Still need to process all `updateTransactions()`



Server-Side In-App Purchase State

For auto-renewable subscriptions

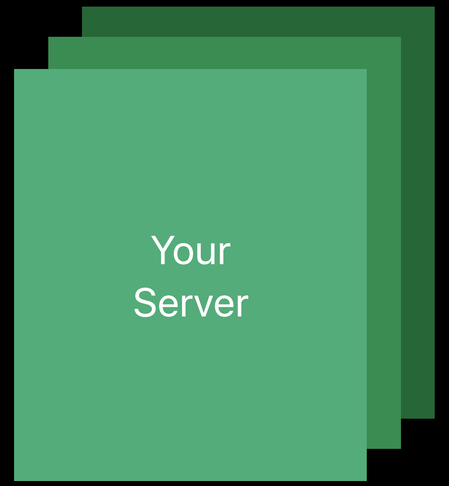
Treat the receipt data like a “token”

- Store on your own server
- Same binary data, multiple times

Propagate subscription state across devices

Still need to process all `updateTransactions()`

- Including all renewal transactions



Server-Side In-App Purchase State

For auto-renewable subscriptions

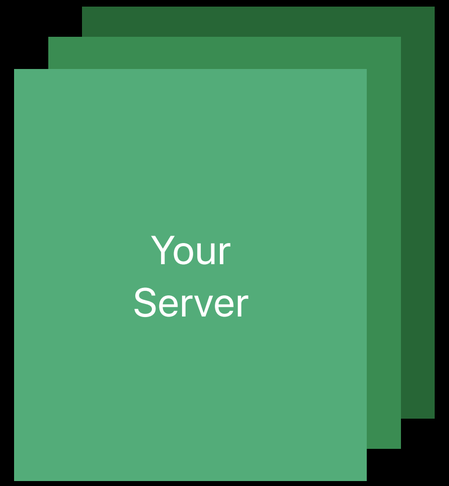
Treat the receipt data like a “token”

- Store on your own server
- Same binary data, multiple times

Propagate subscription state across devices

Still need to process all `updateTransactions()`

- Including all renewal transactions
- All the way to `finishTransaction()`



Server-Side In-App Purchase State

Optimizing app receipts

NEW

Server-Side In-App Purchase State

Optimizing app receipts

NEW

App receipts include all subscription renewals

Server-Side In-App Purchase State

Optimizing app receipts

NEW

App receipts include all subscription renewals

Many developers only care about latest

Server-Side In-App Purchase State

Optimizing app receipts

NEW

App receipts include all subscription renewals

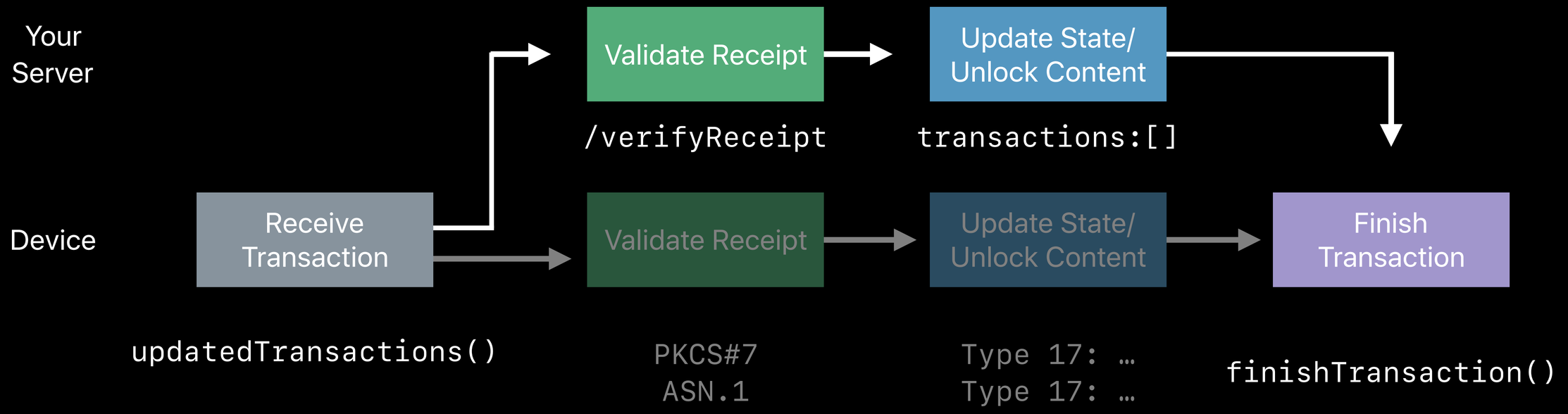
Many developers only care about latest

Optional query parameter `exclude-old-transactions`

- true: only return latest transaction
- false: include all transactions (default)

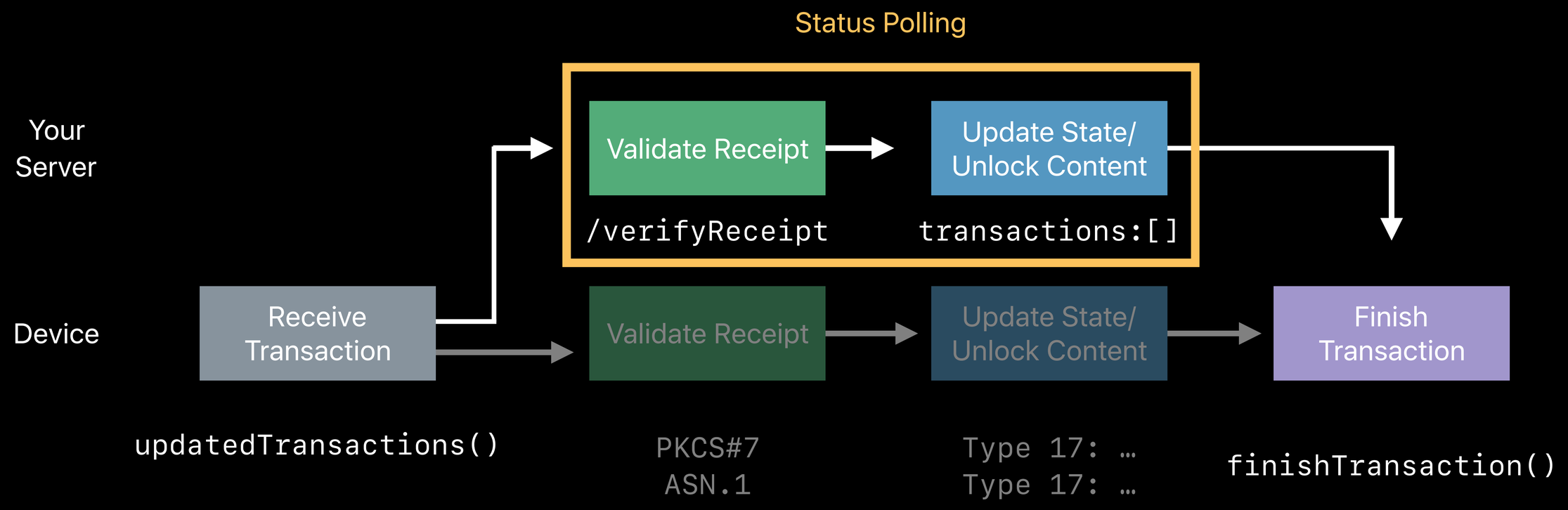
In-App Purchase Process

Processing transactions



In-App Purchase Process

Processing transactions



Server-Side Subscriptions

Server-Side Subscriptions

Why did a user’s subscription expire?

Server-Side Subscriptions

Why did a user’s subscription expire?

Will this user’s subscription be renewed?

Server-Side Subscriptions

Why did a user’s subscription expire?

Will this user’s subscription be renewed?

Will the user be downgraded after this billing cycle?

Server-Side Subscriptions

Why did a user's subscription expire?

Will this user's subscription be renewed?

Why did AppleCare give the user a refund?

Will the user be downgraded after this billing cycle?

Server-Side Subscriptions

Why did a user's subscription expire?

Will this user's subscription be renewed?

Why did AppleCare give the user a refund?

Has the user agreed to a price increase?

Will the user be downgraded after this billing cycle?

Server-Side Subscriptions

Why did a user's subscription expire?

What does the user need to know about their subscription?

Will this user's subscription be renewed?

Why did AppleCare give the user a refund?

Has the user agreed to a price increase?

Will the user be downgraded after this billing cycle?

Why?

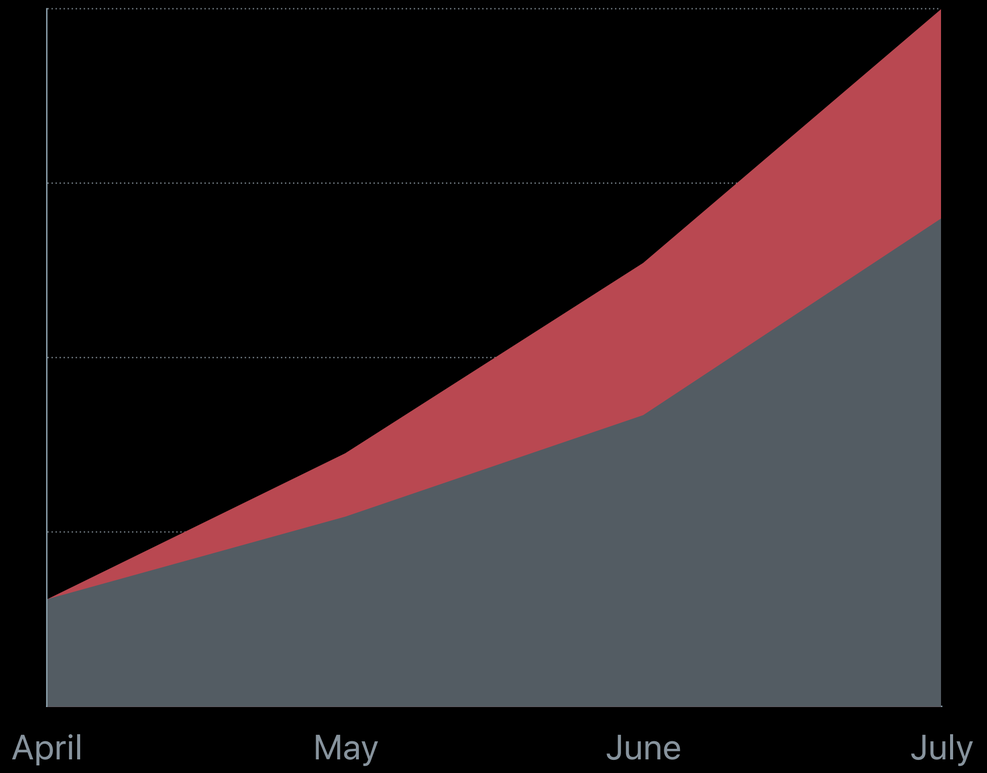
Subscription Churn

Losing subscribers

Subscription Churn

Losing subscribers

Every subscription that lapses is lost revenue

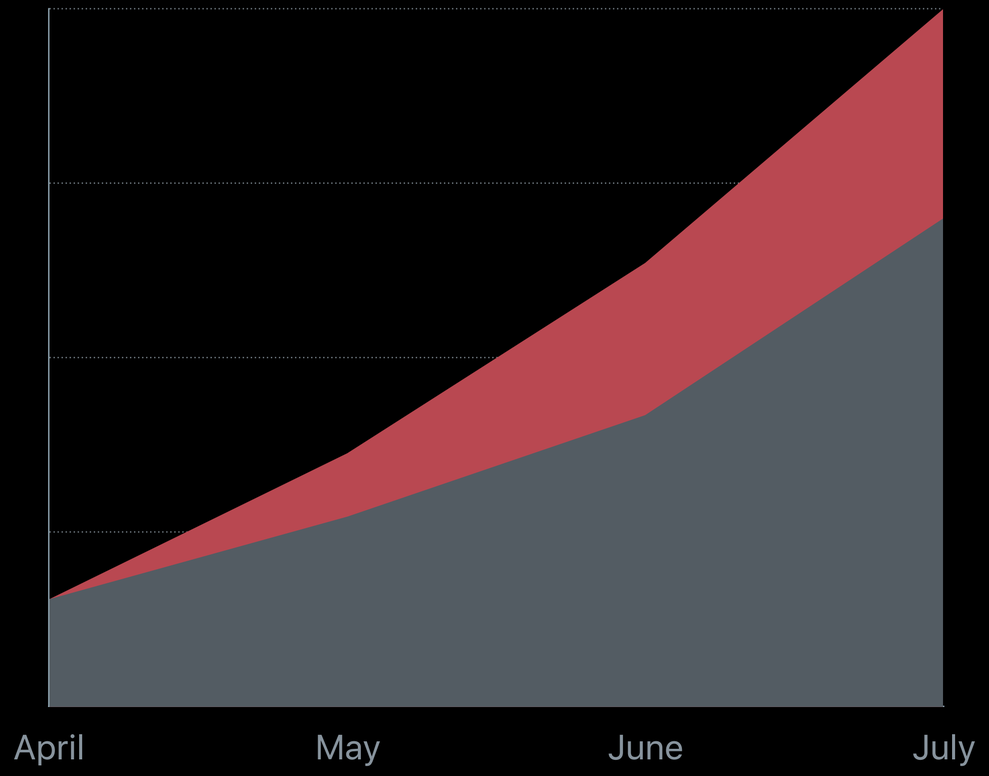


Subscription Churn

Losing subscribers

Every subscription that lapses is lost revenue

Involuntary



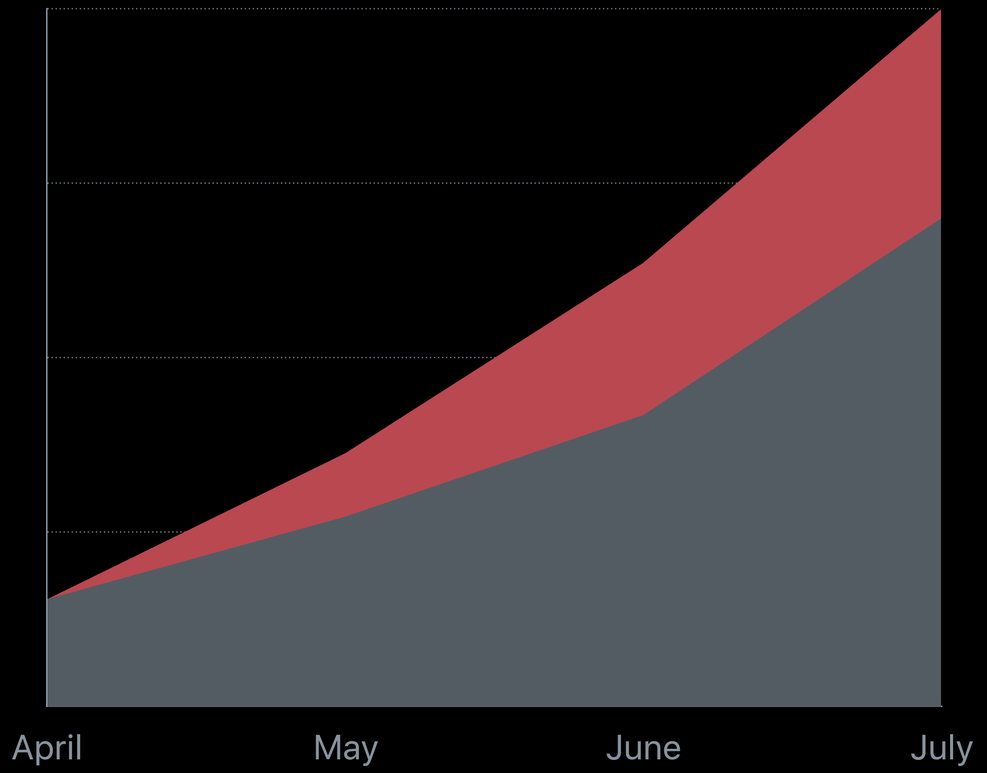
Subscription Churn

Losing subscribers

Every subscription that lapses is lost revenue

Involuntary

- Credit card expires



Subscription Churn

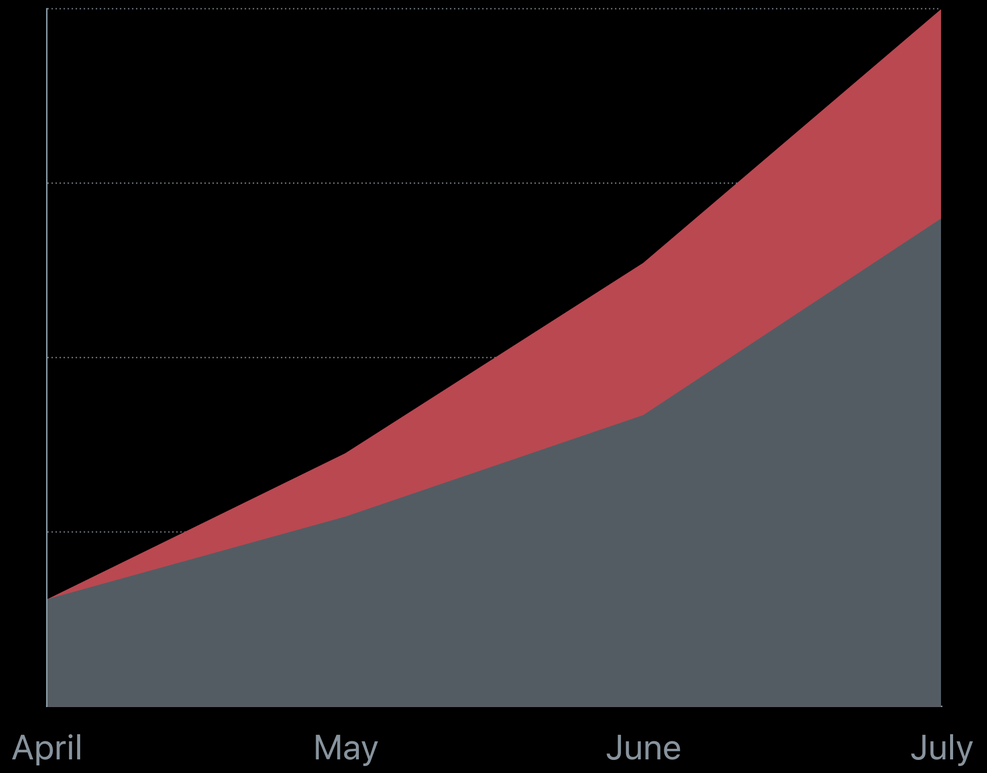
Losing subscribers

Every subscription that lapses is lost revenue

Involuntary

- Credit card expires

Voluntary



Subscription Churn

Losing subscribers

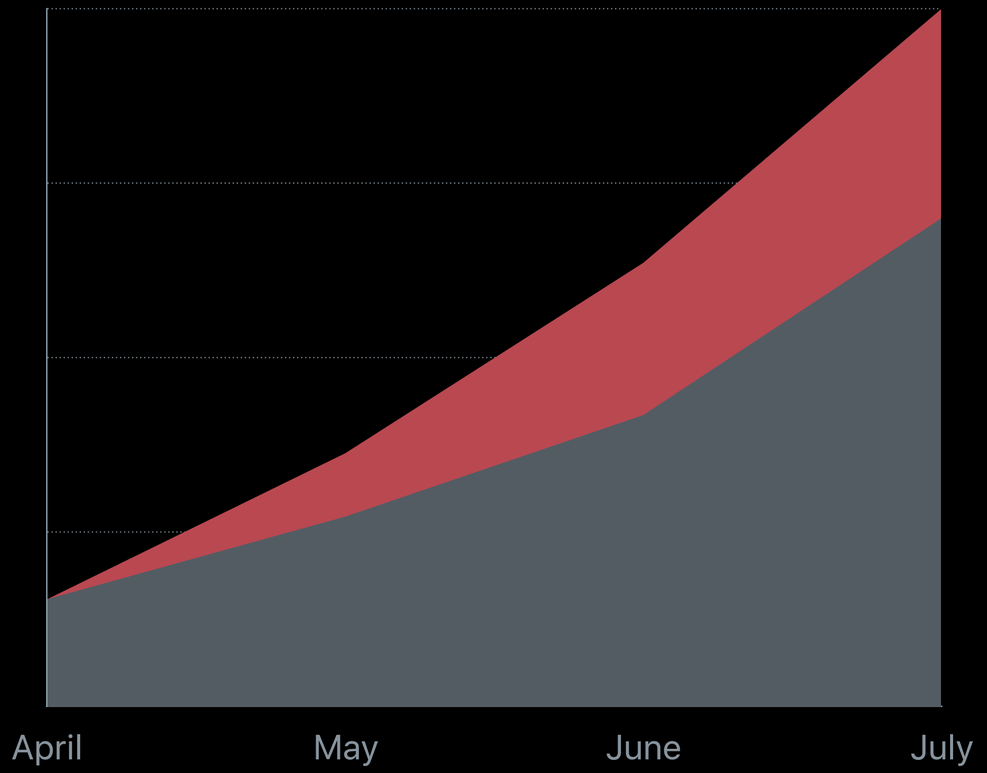
Every subscription that lapses is lost revenue

Involuntary

- Credit card expires

Voluntary

- Turns off auto-renew



Subscription Churn

Losing subscribers

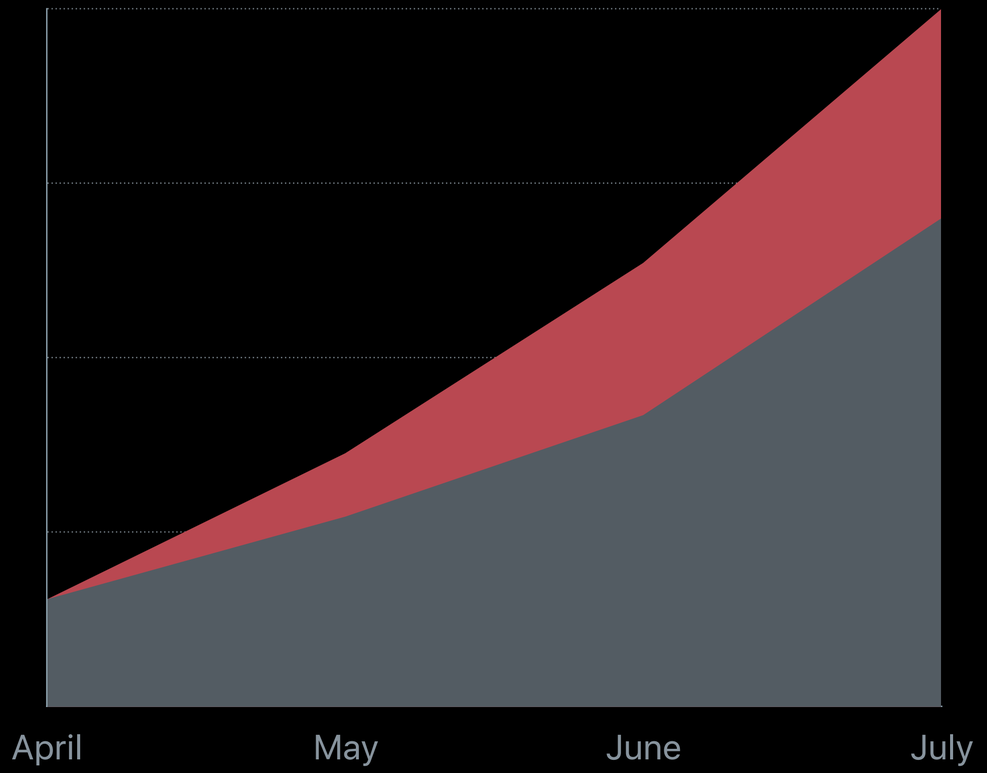
Every subscription that lapses is lost revenue

Involuntary

- Credit card expires

Voluntary

- Turns off auto-renew
- Receives a refund from AppleCare



Subscription Churn

Losing subscribers

Every subscription that lapses is lost revenue

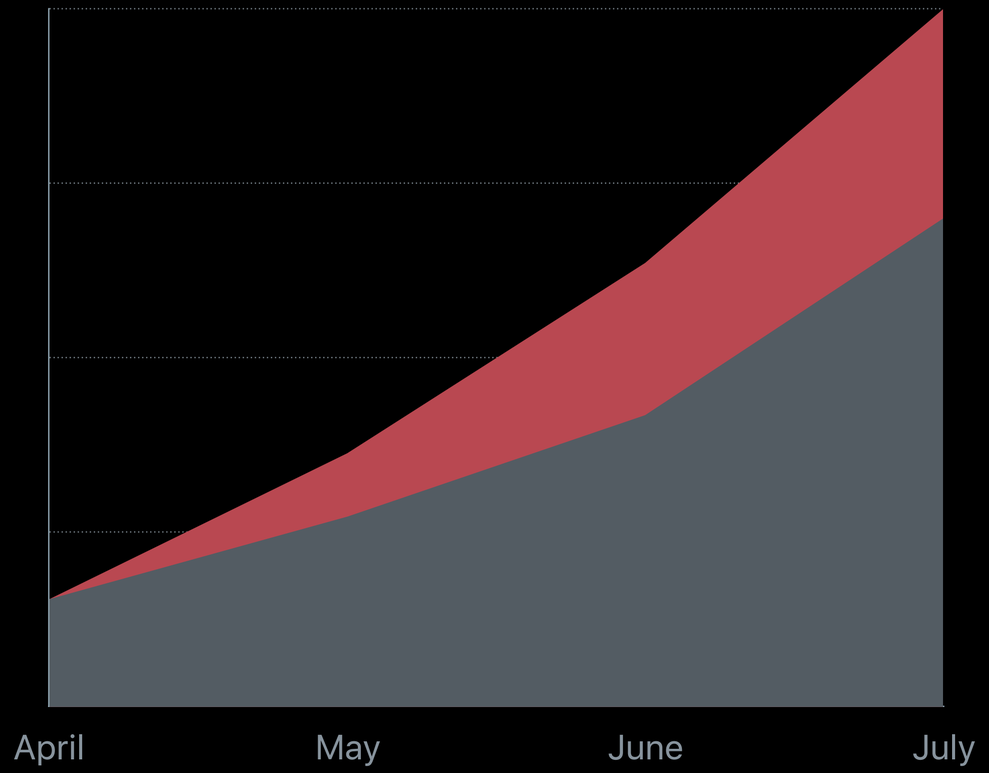
Involuntary

- Credit card expires

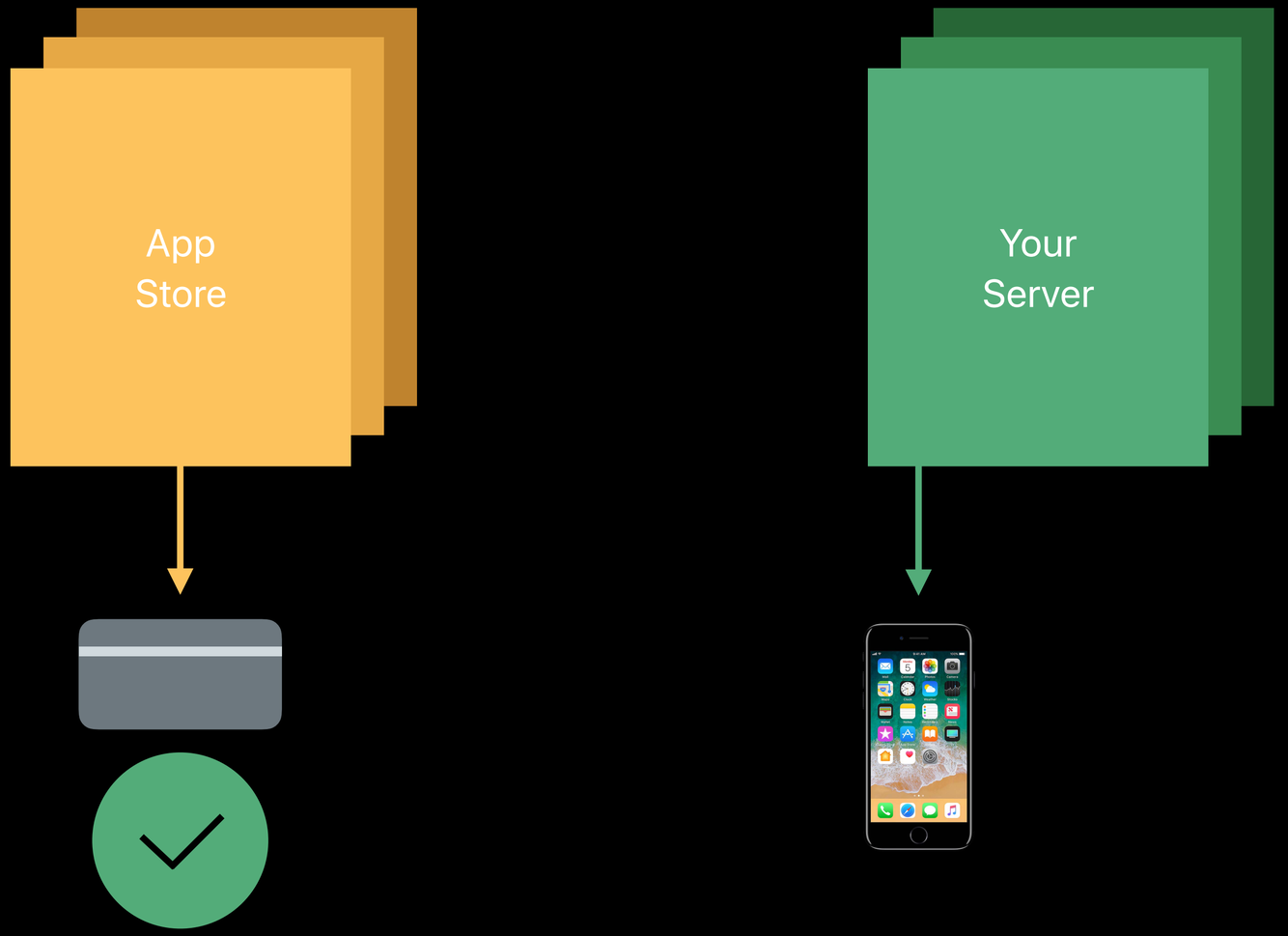
Voluntary

- Turns off auto-renew
- Receives a refund from AppleCare

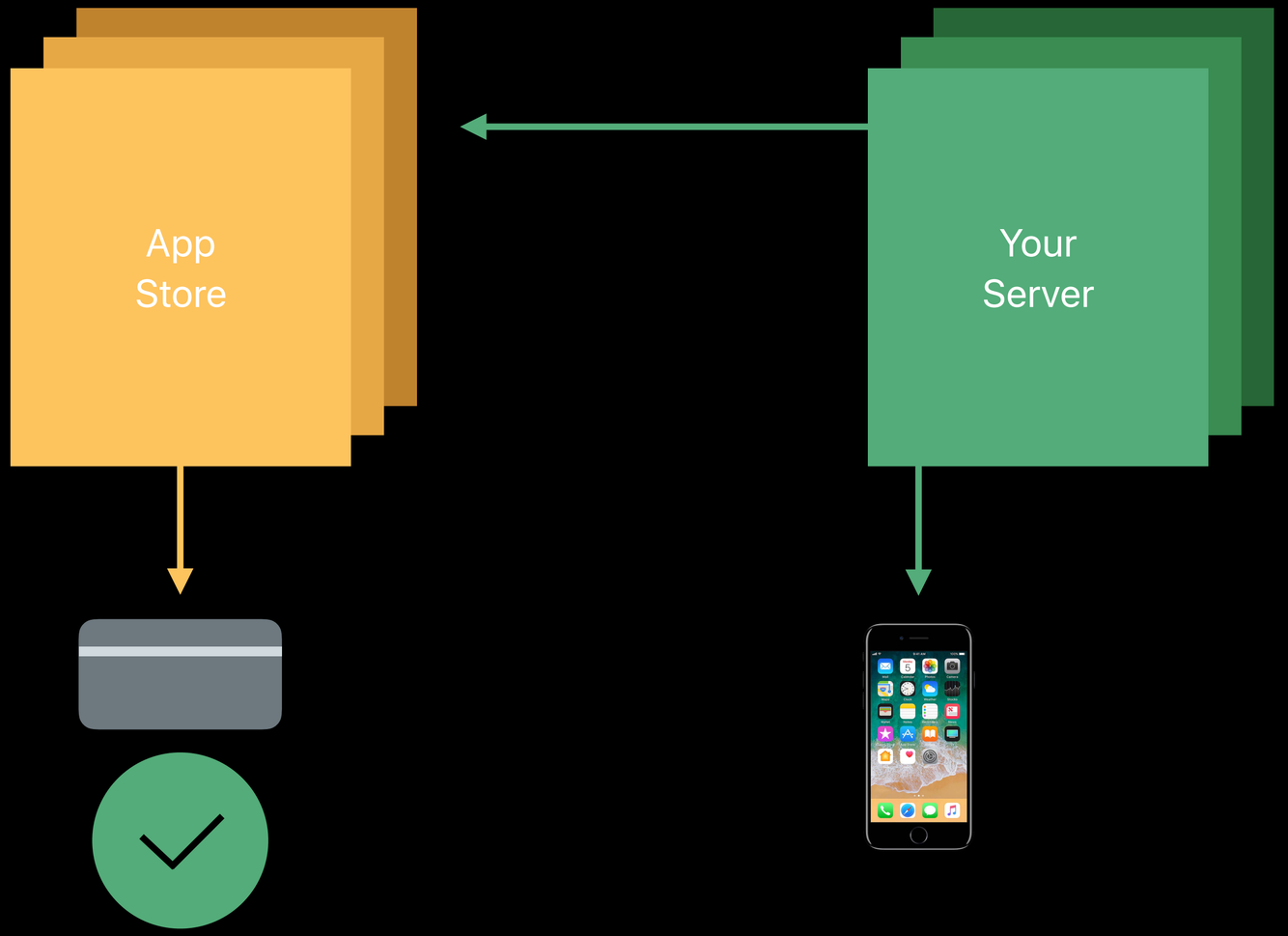
New tools to help reduce churn



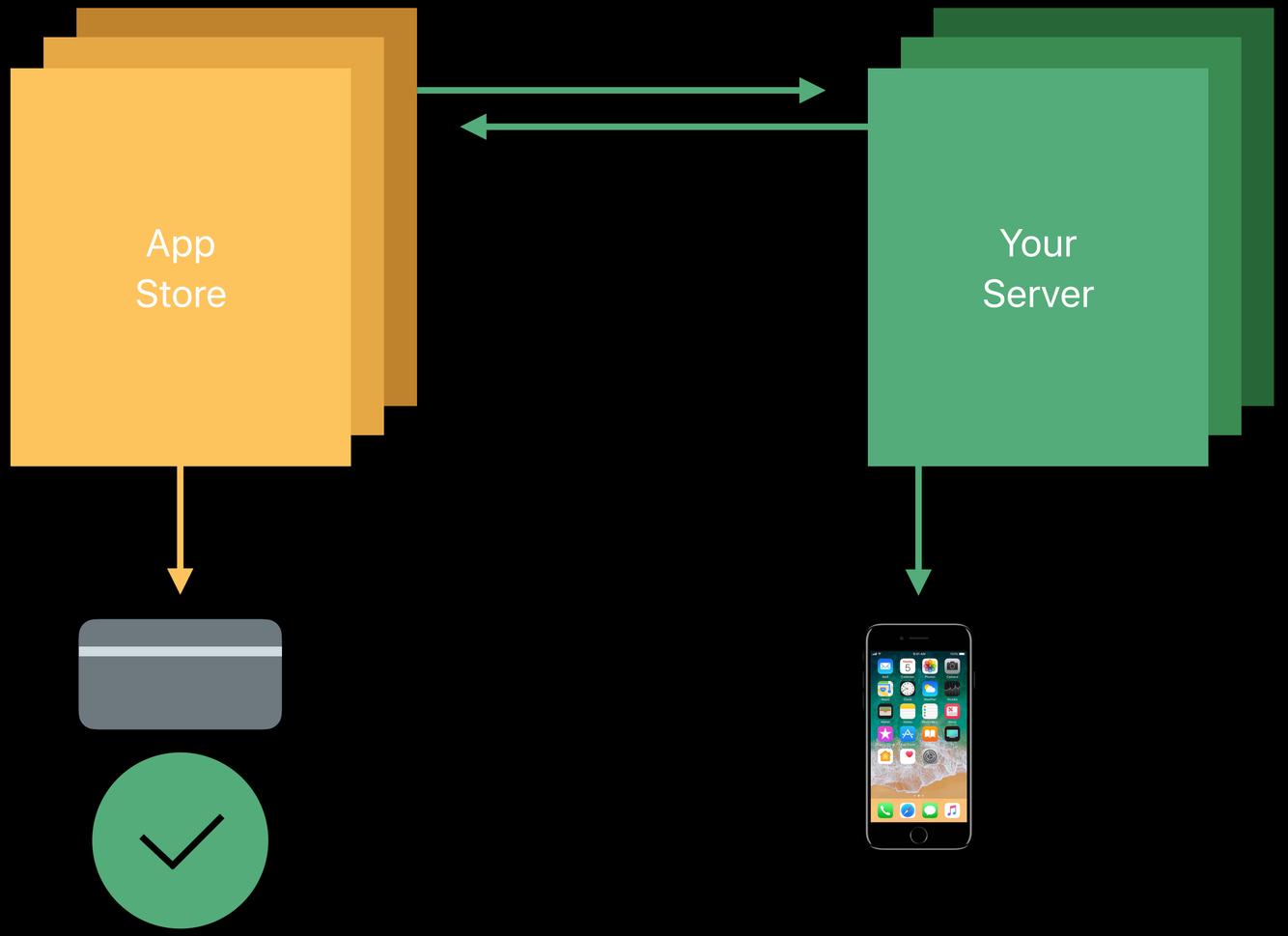
Example



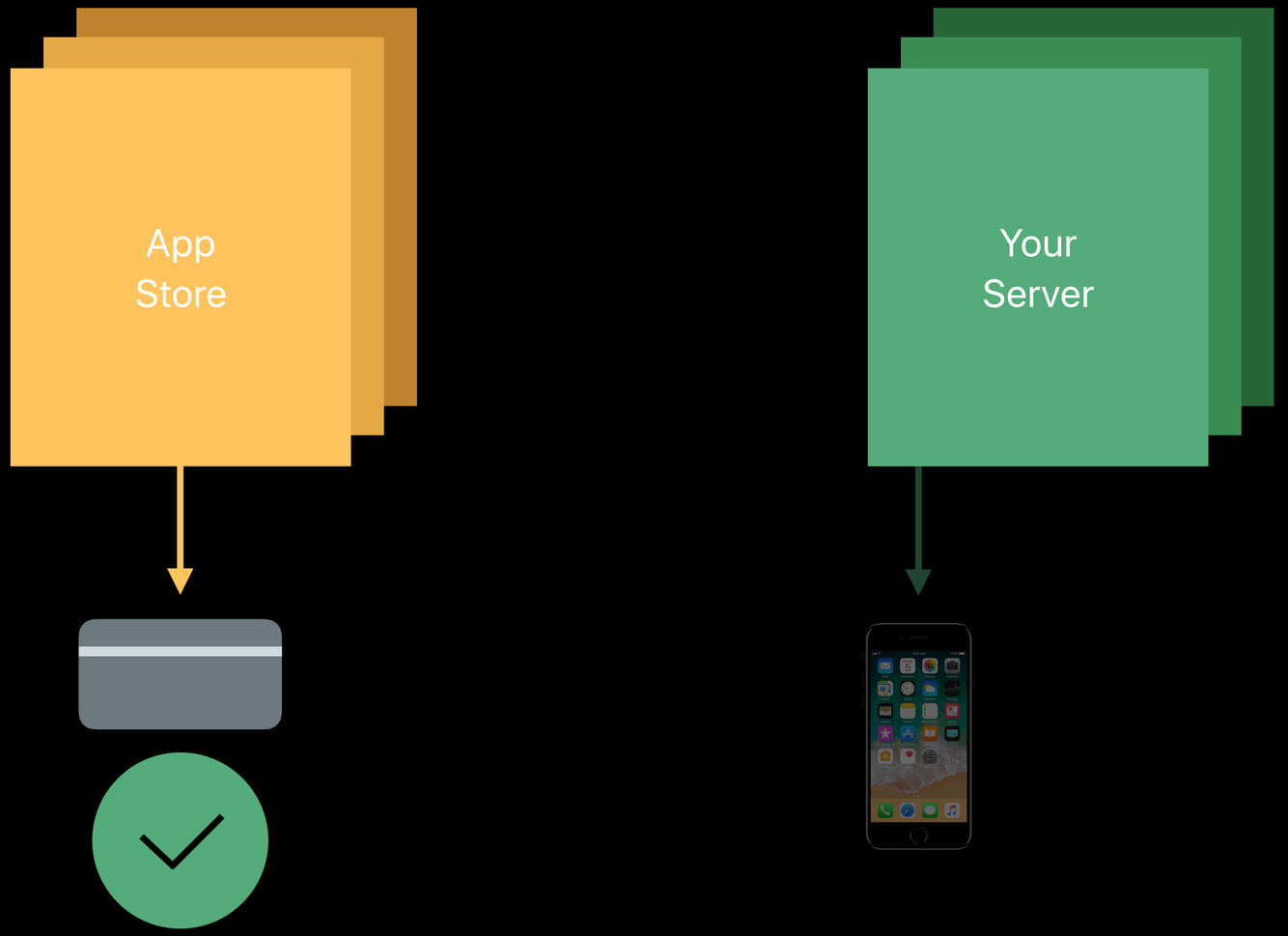
Example



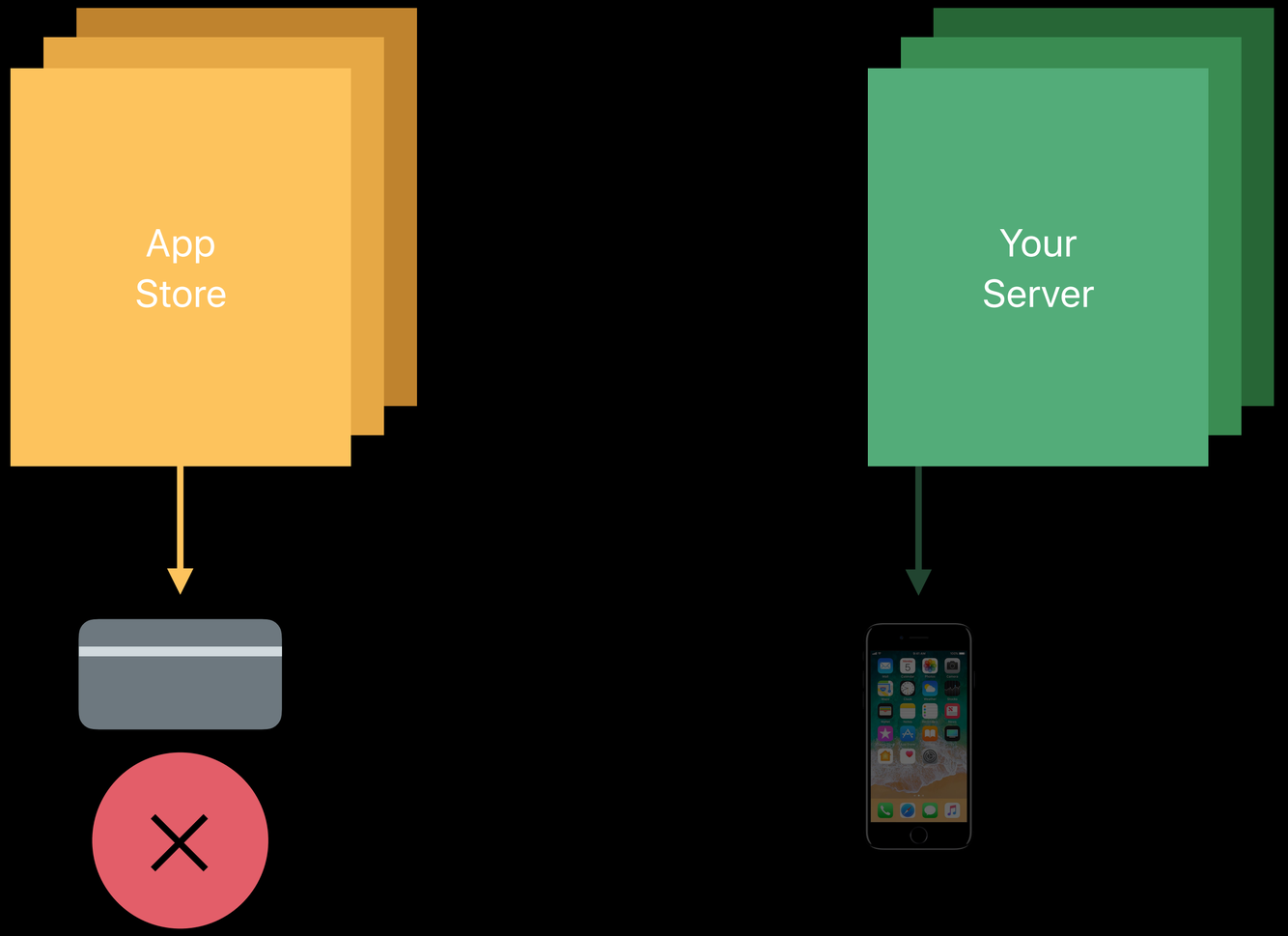
Example



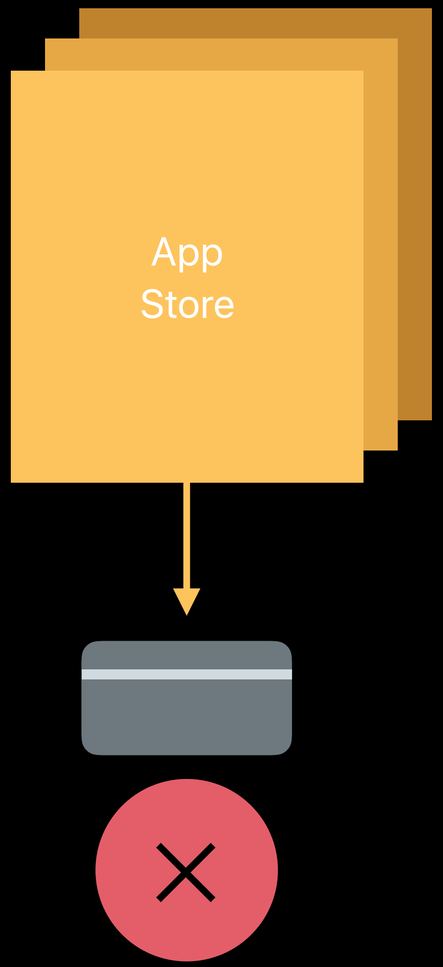
Example

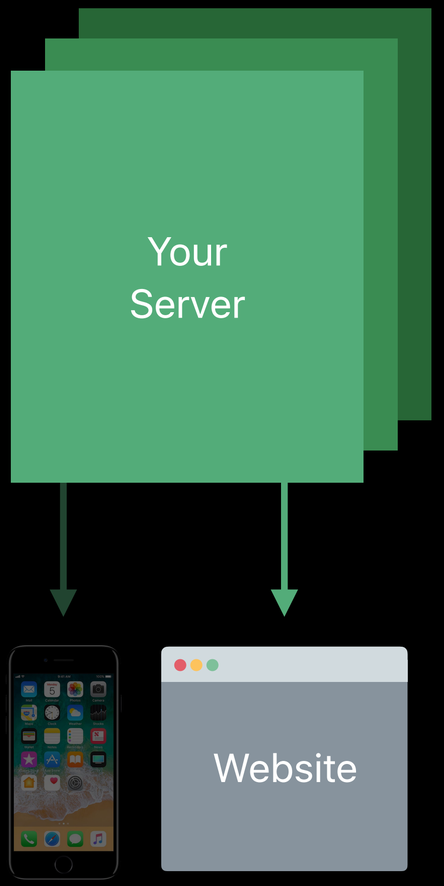


Example

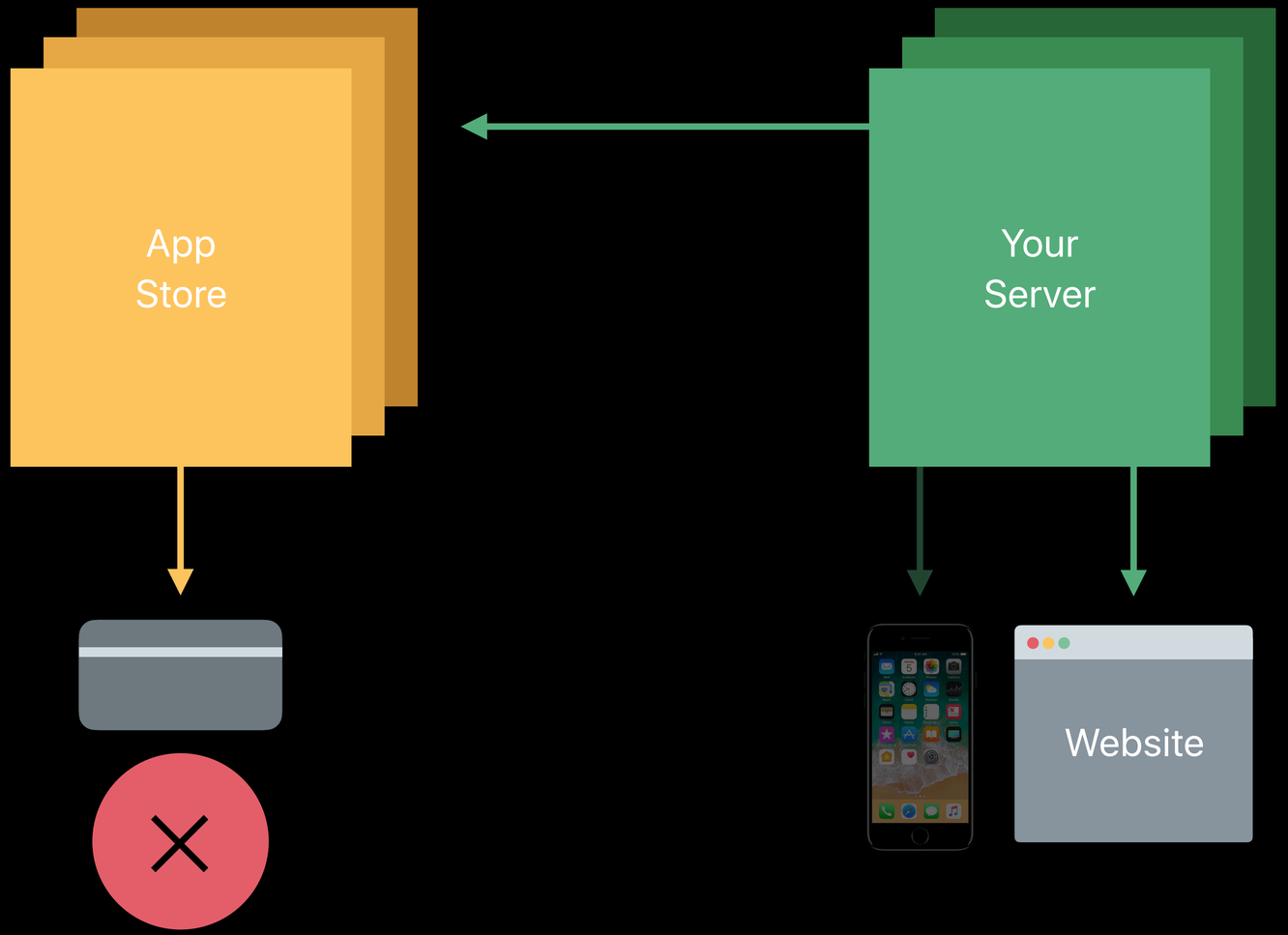


Example

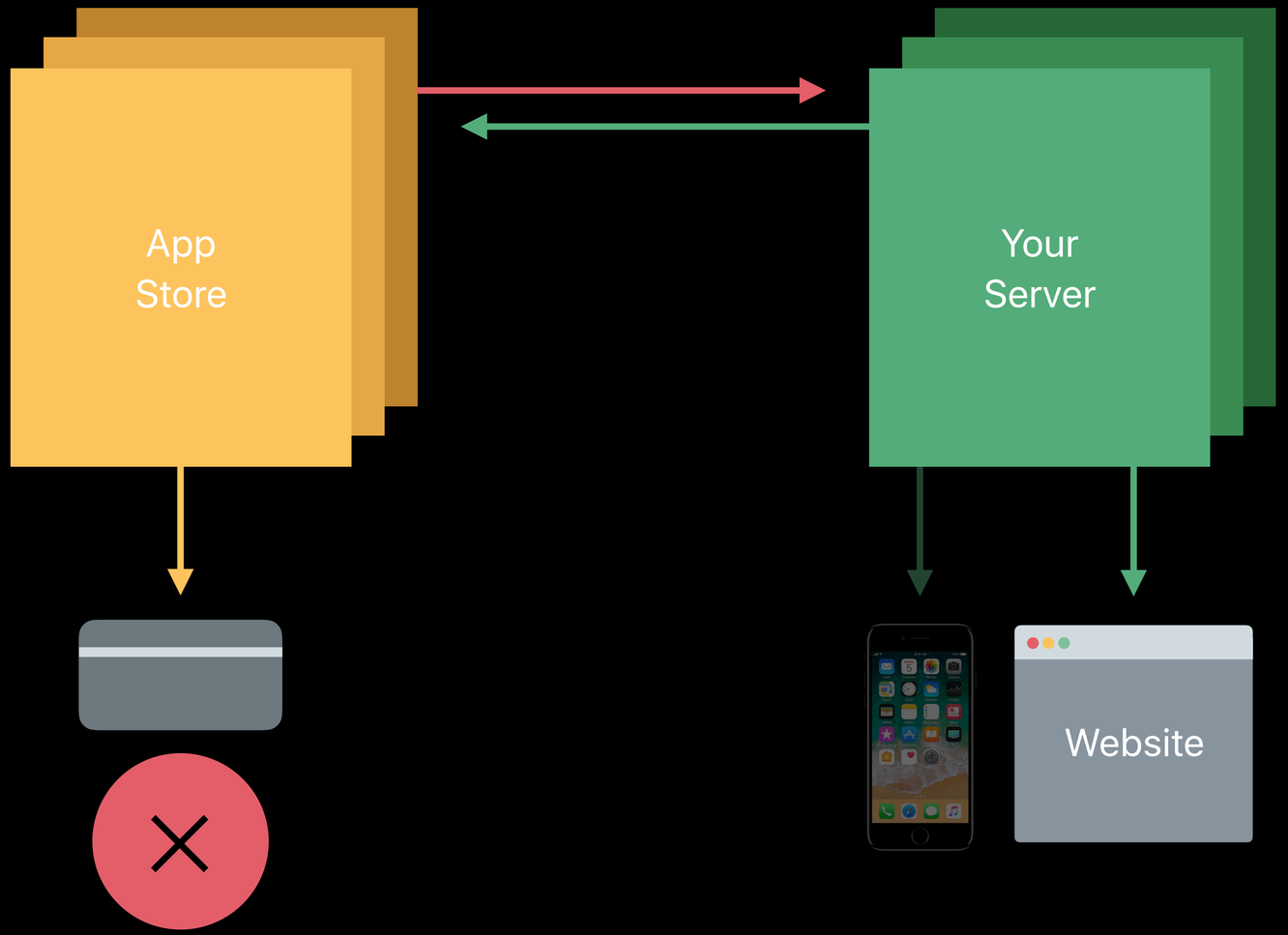




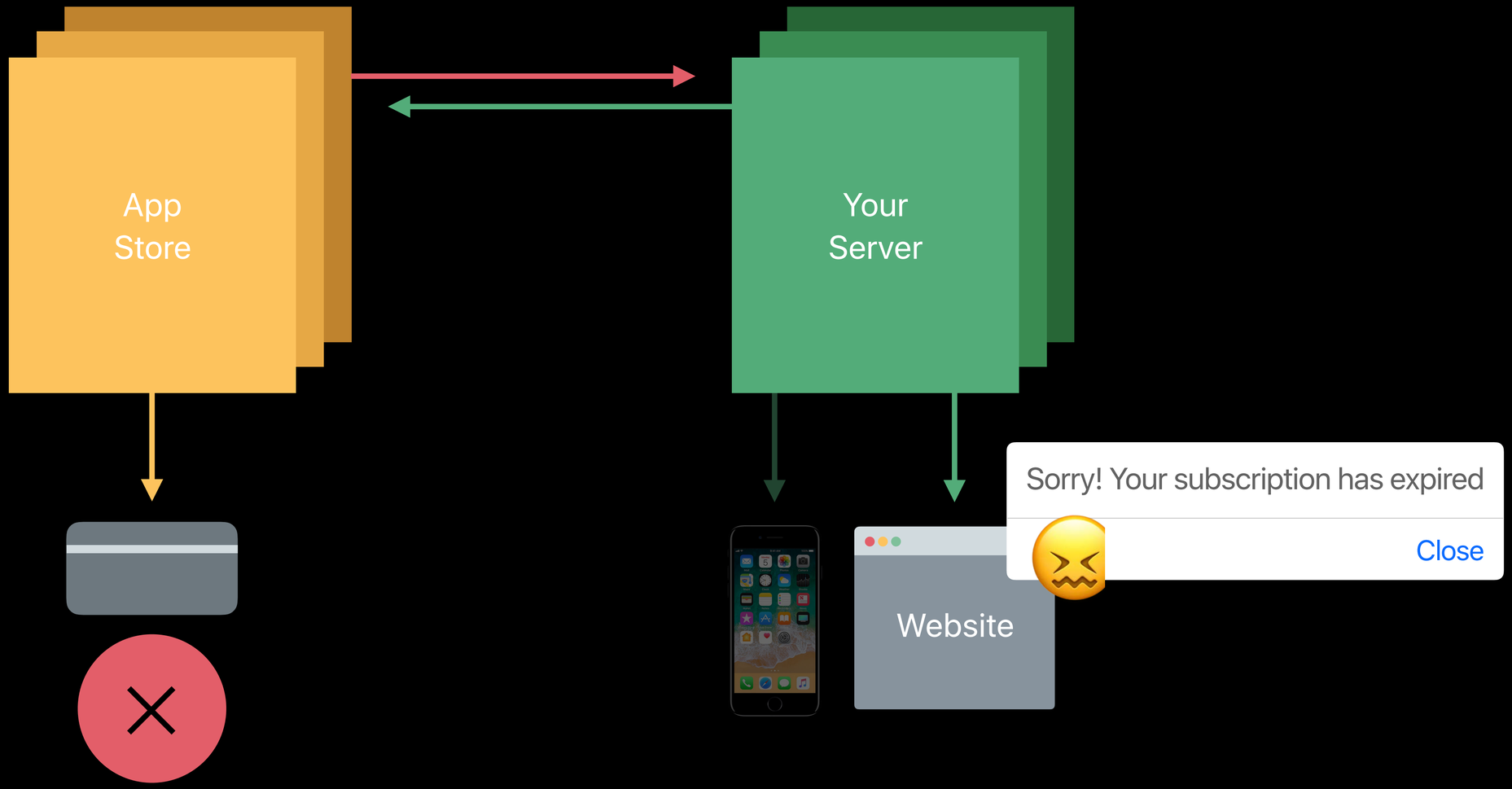
Example



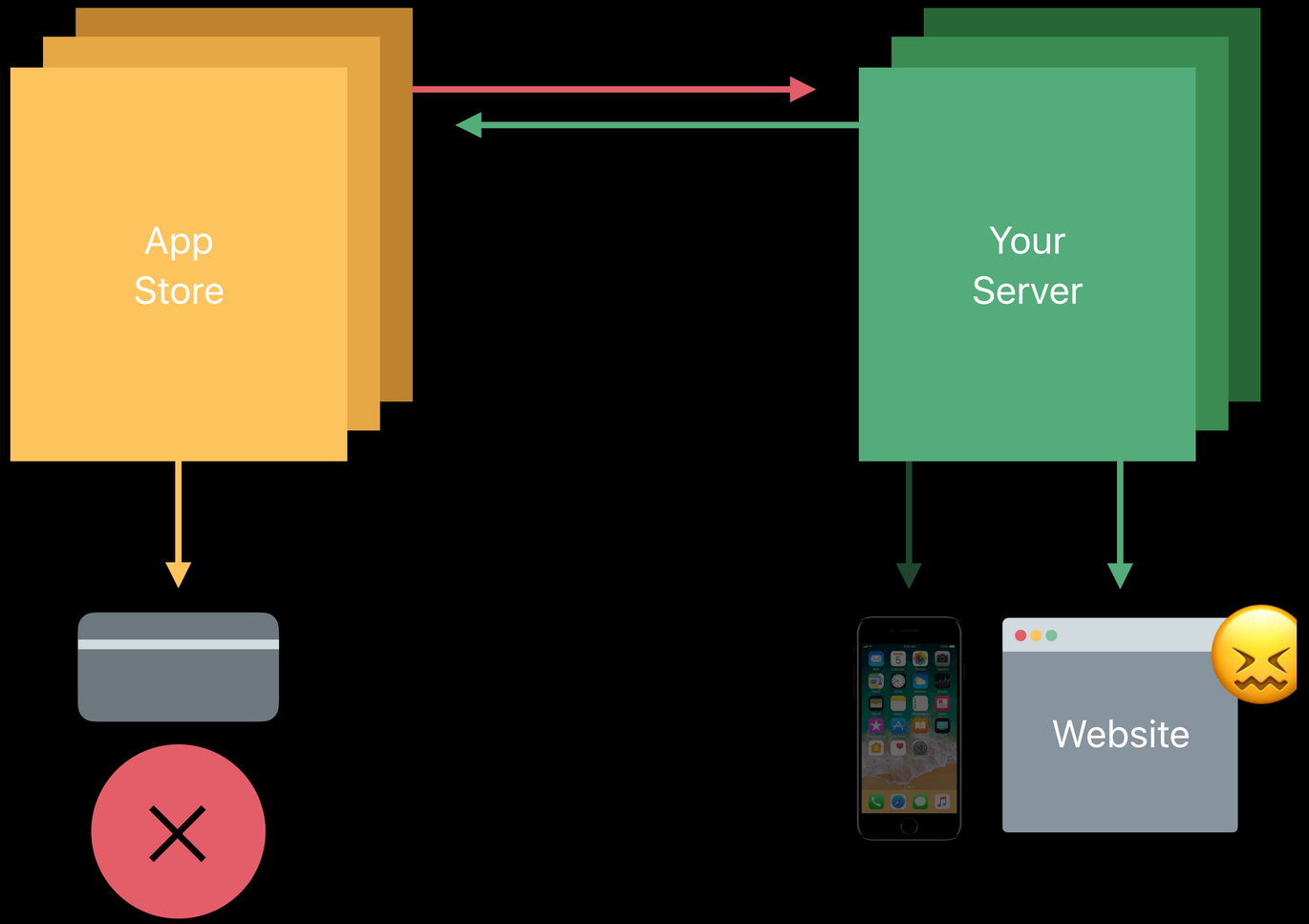
Example



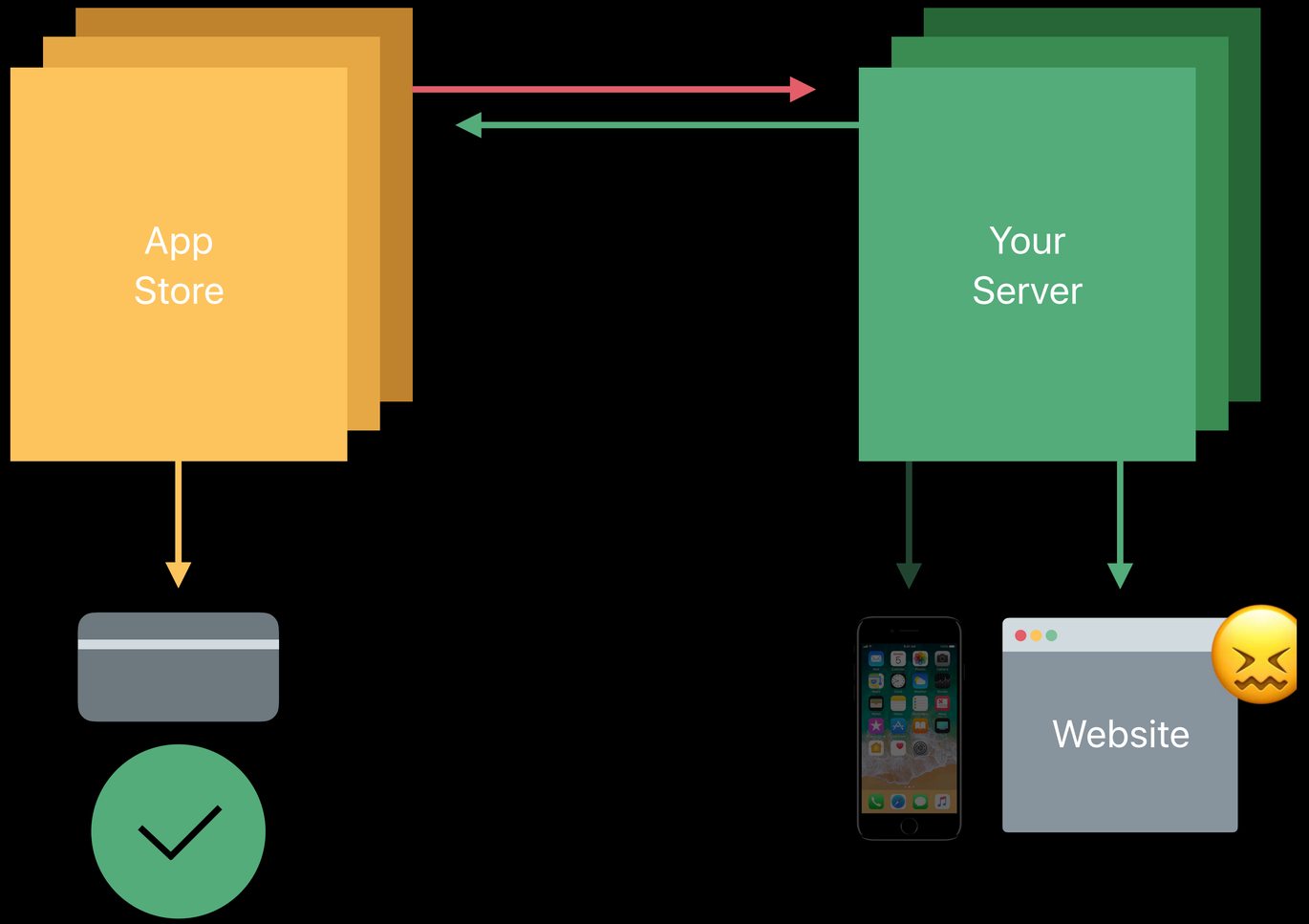
Example


App
Store
Your
Server
Website
Sorry! Your subscription has expired
Close

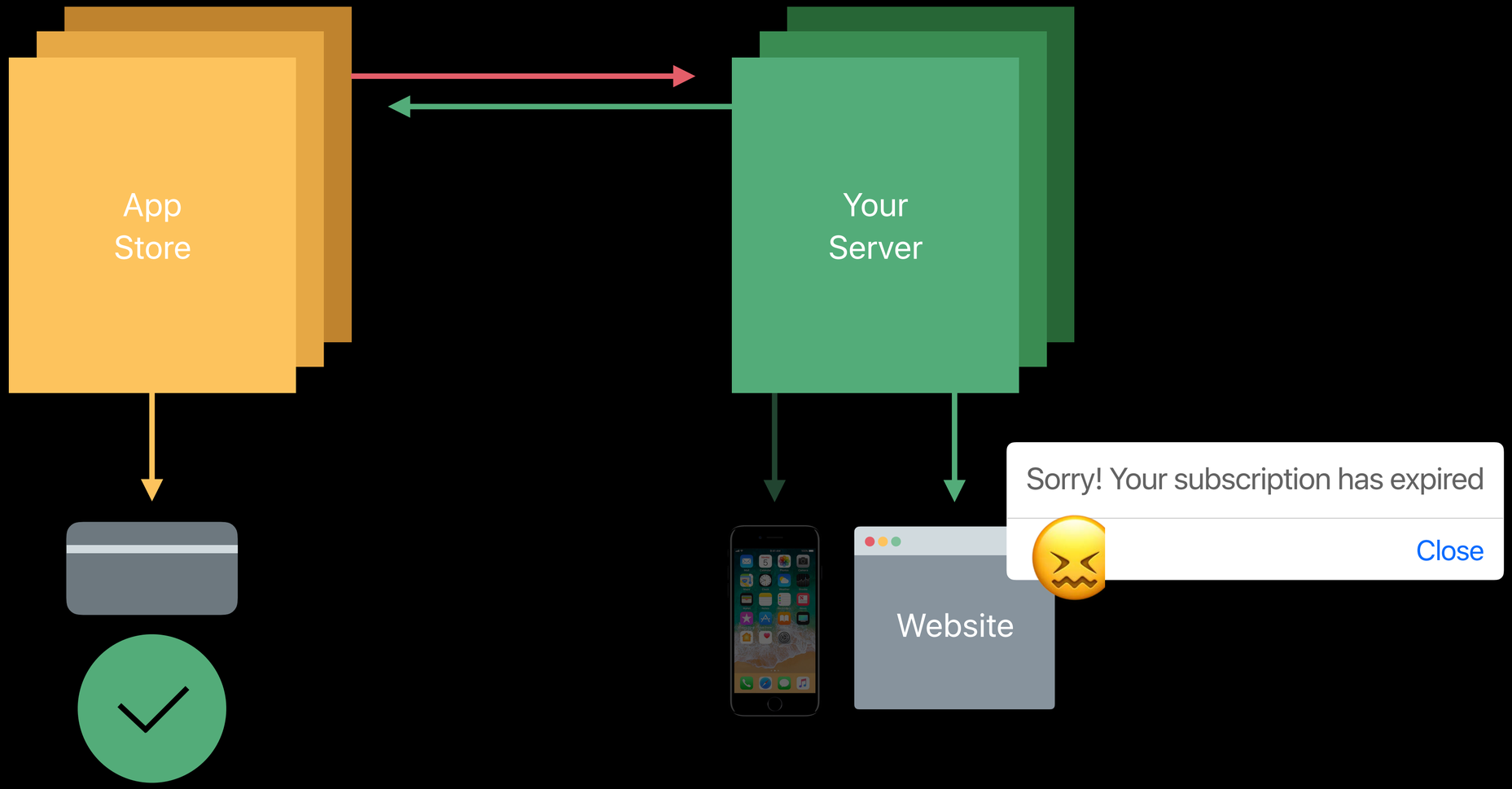
Example



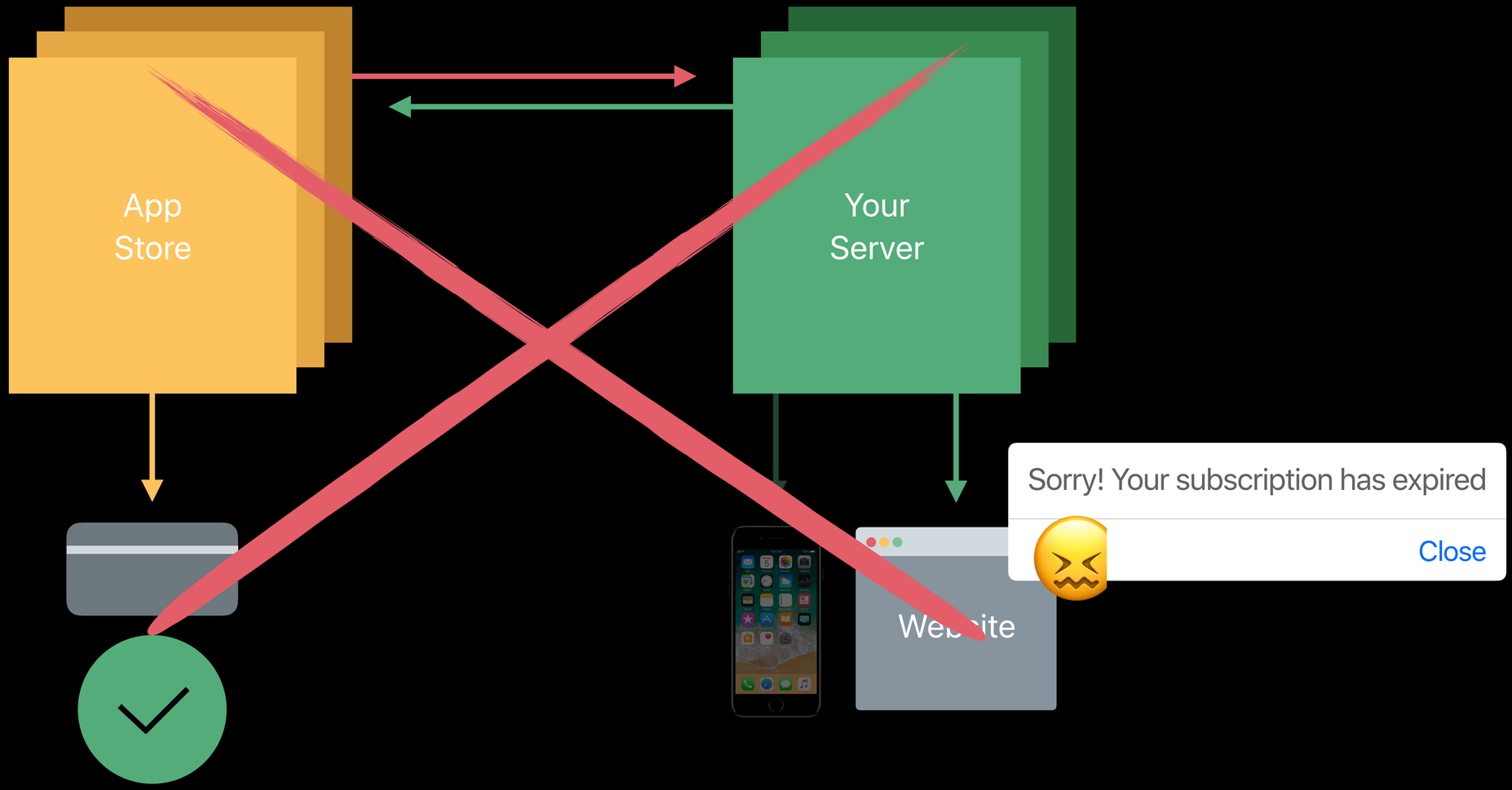
Example



Example

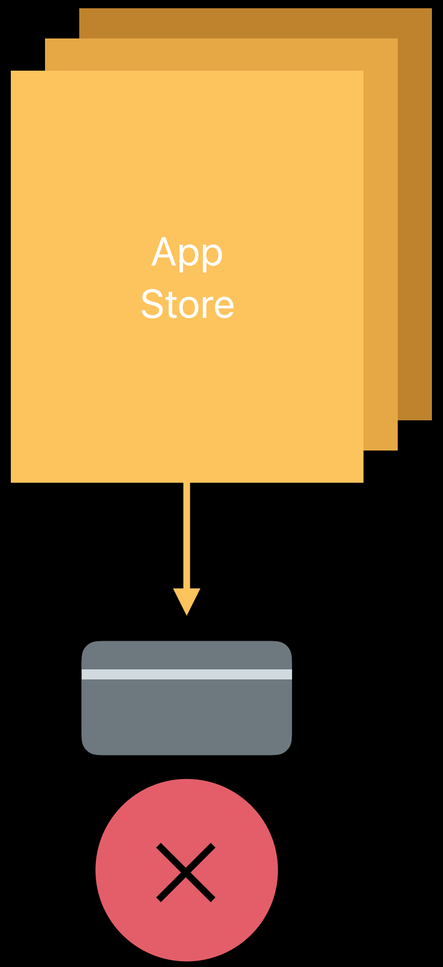

App
Store
Your
Server
Website
Sorry! Your subscription has expired
Close

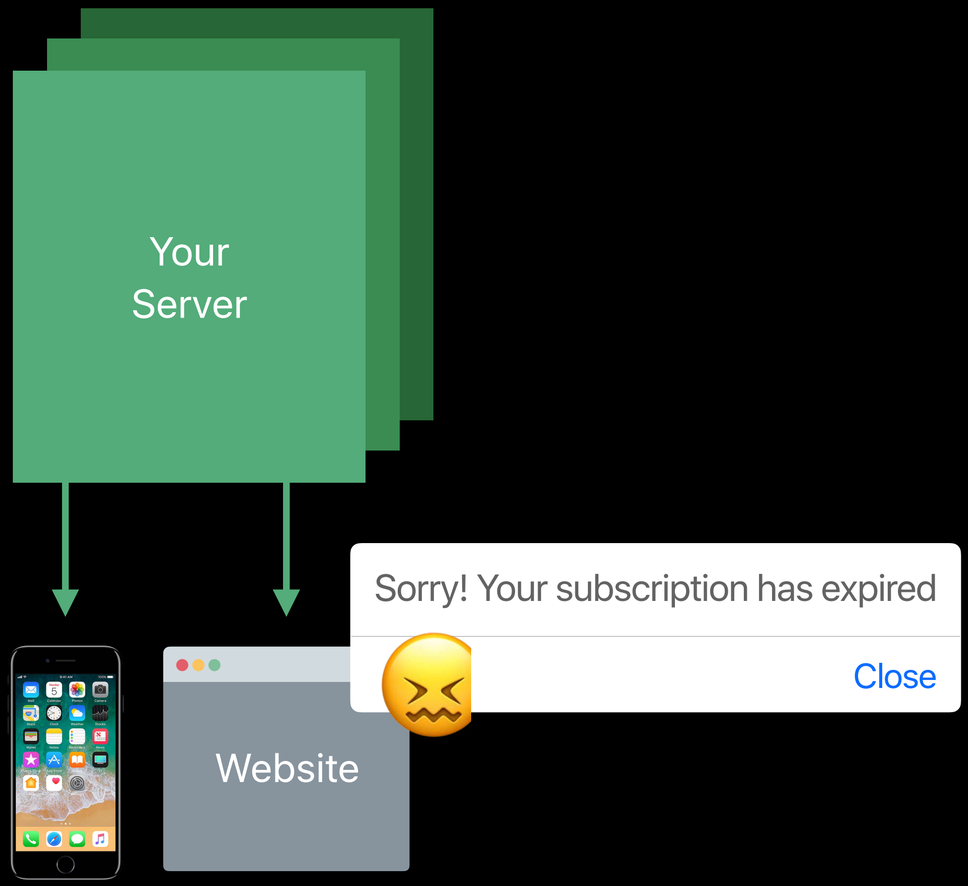
Example


App
Store
Your
Server
Sorry! Your subscription has expired
Close

Server Notifications

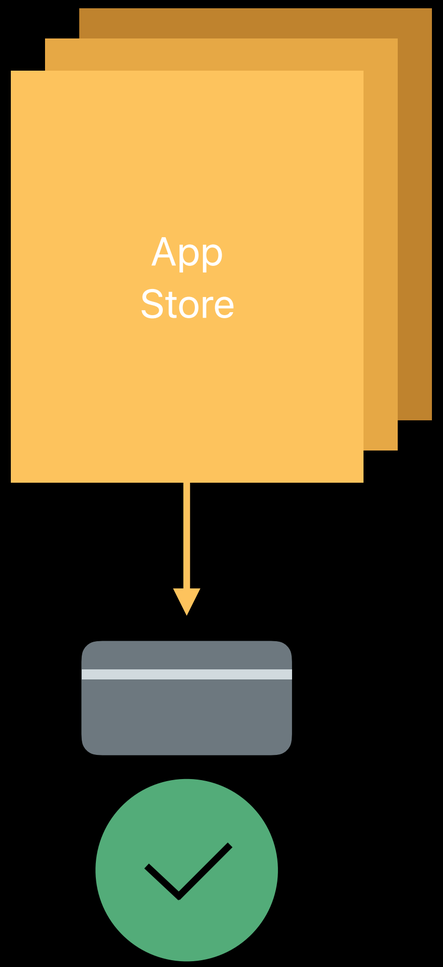
NEW

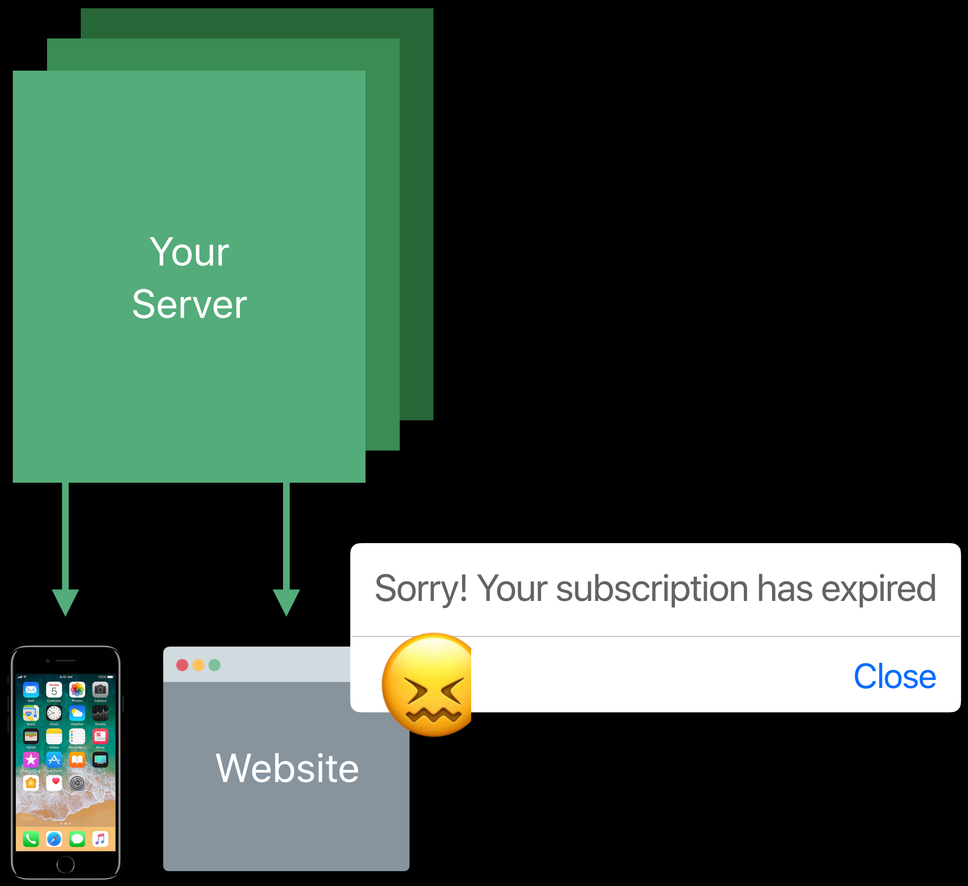




Server Notifications

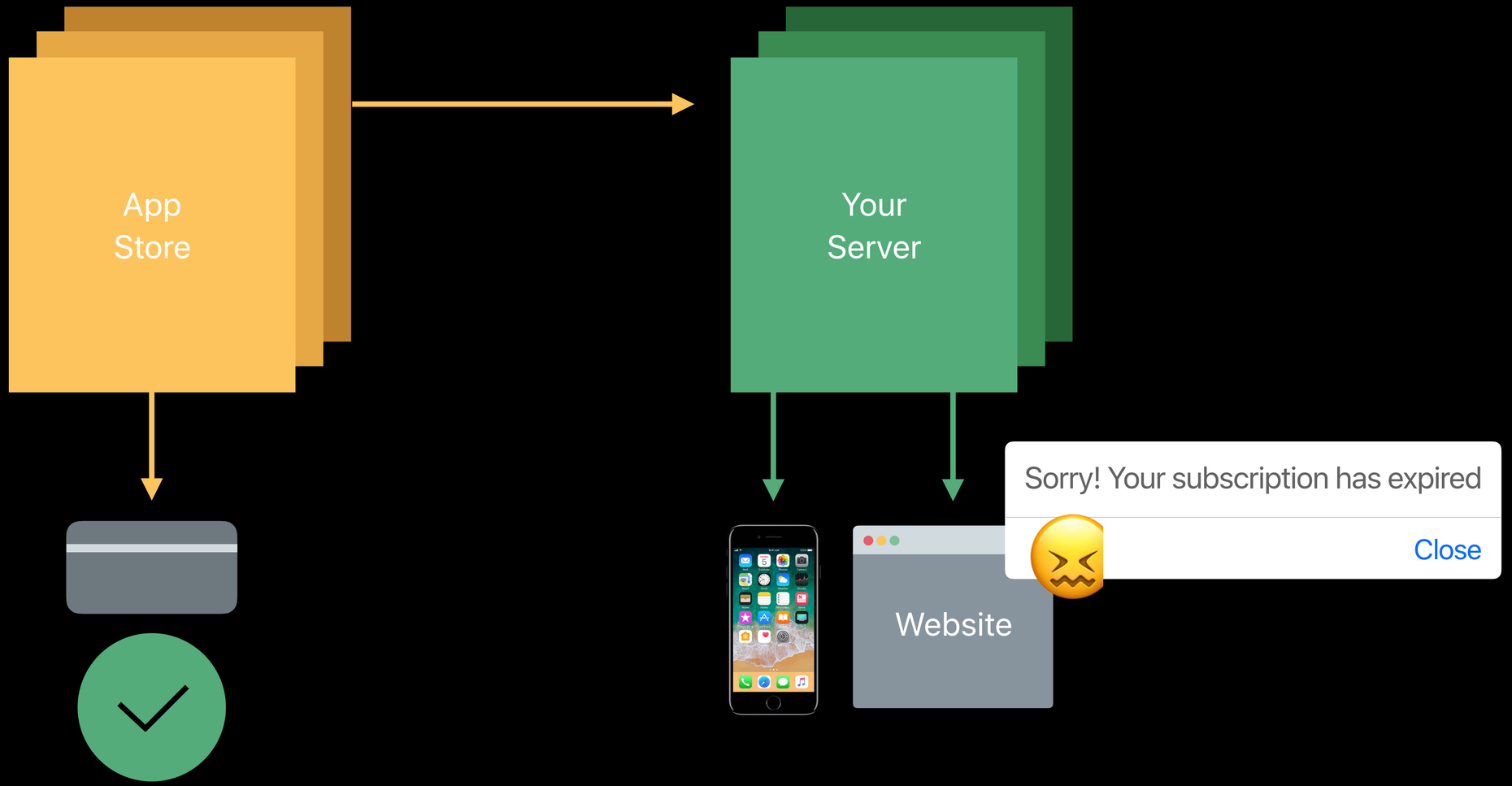
NEW





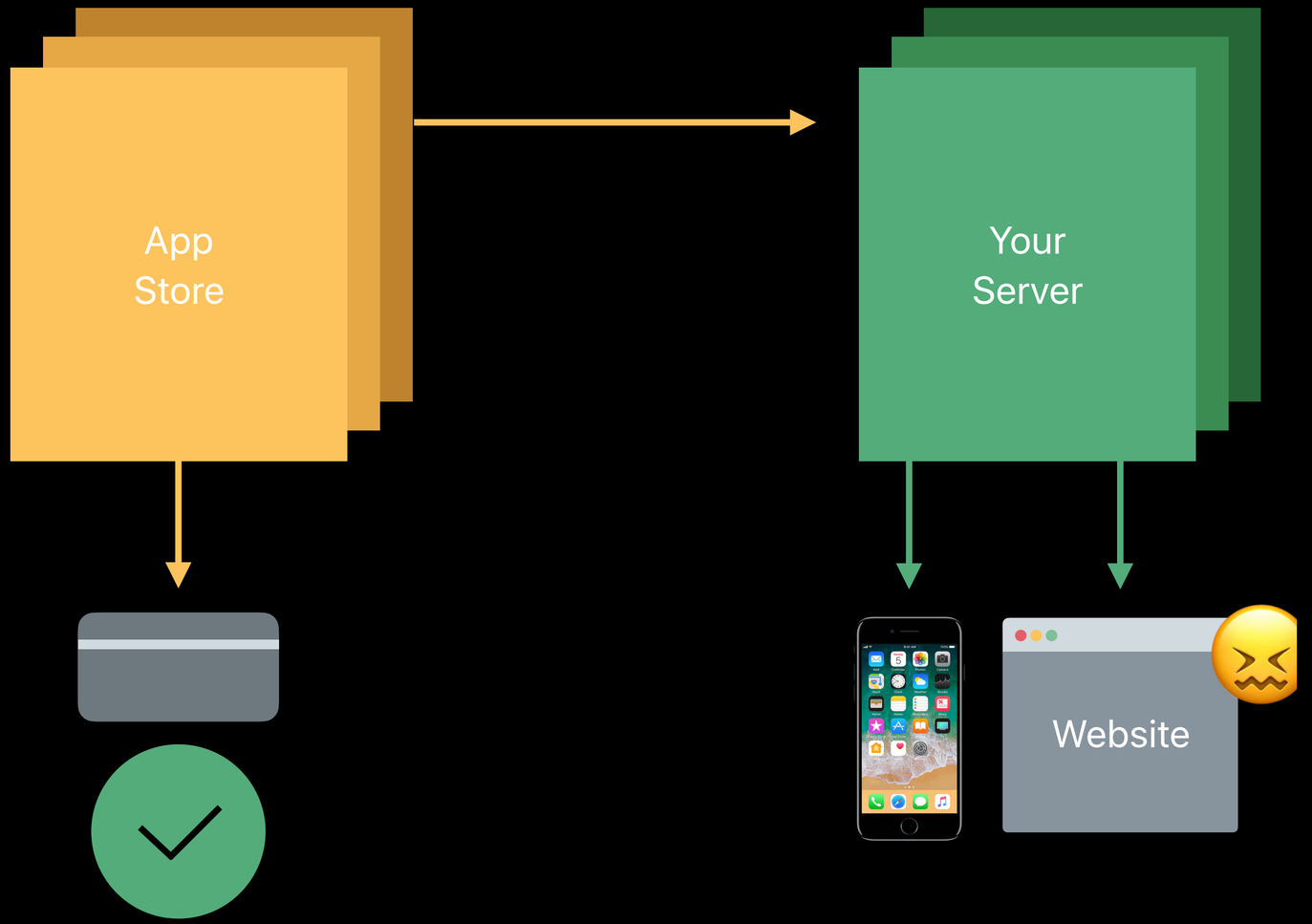
Server Notifications

NEW



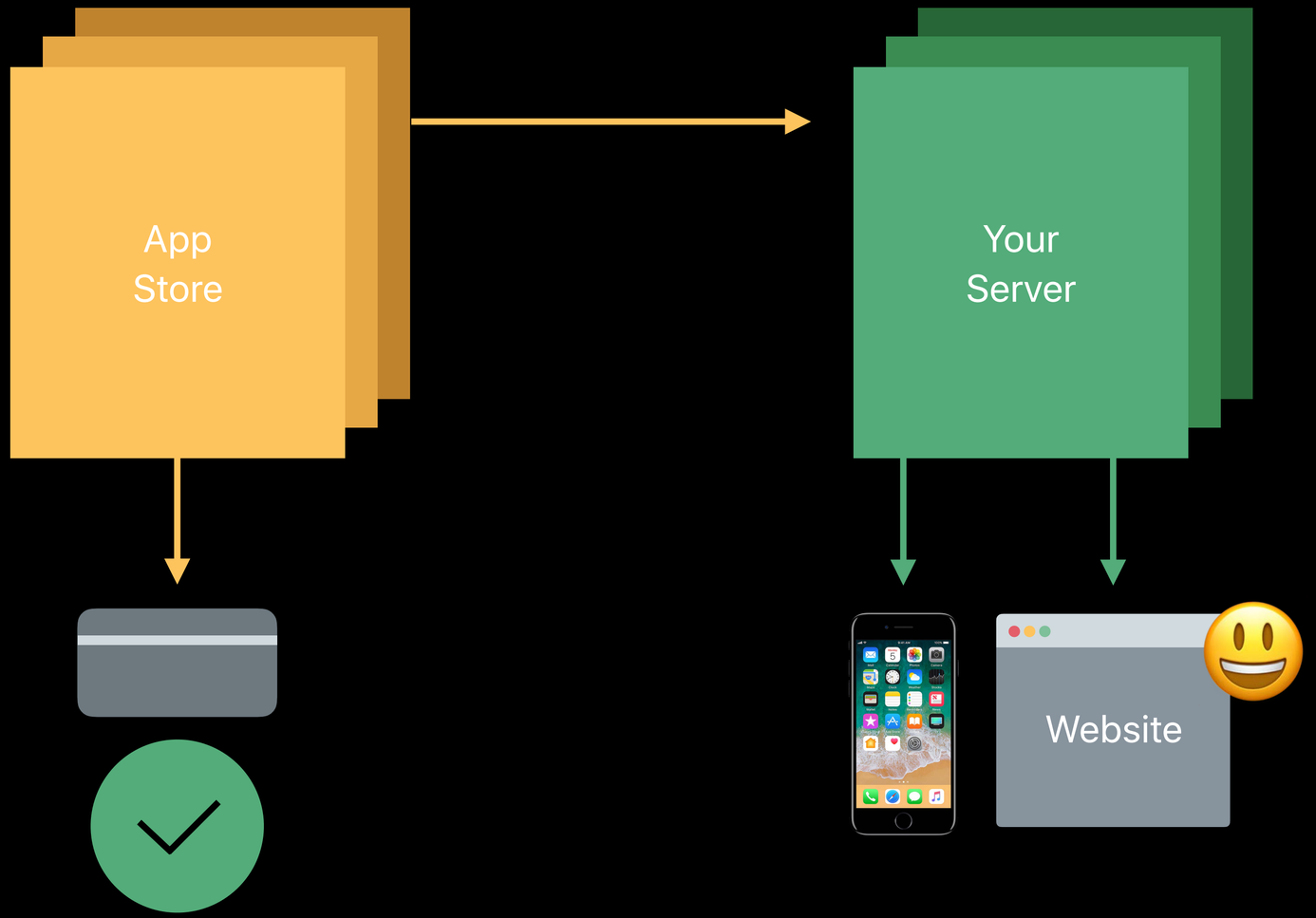
Server Notifications

NEW



Server Notifications

NEW



Server Notifications

NEW

Server Notifications

NEW

Status URL in iTunes Connect

Server Notifications

NEW

Status URL in iTunes Connect

App Transport Security requirements apply

Server Notifications

NEW

Status URL in iTunes Connect

App Transport Security requirements apply

HTTP POST to your server for status changes

Server Notifications

NEW

Status URL in iTunes Connect

App Transport Security requirements apply

HTTP POST to your server for status changes

- Initial purchase of a subscription

Server Notifications

NEW

Status URL in iTunes Connect

App Transport Security requirements apply

HTTP POST to your server for status changes

- Initial purchase of a subscription
- Subscription cancellation by AppleCare

Server Notifications

NEW

Status URL in iTunes Connect

App Transport Security requirements apply

HTTP POST to your server for status changes

- Initial purchase of a subscription
- Subscription cancellation by AppleCare
- Subscription downgrades

Server Notifications

NEW

Status URL in iTunes Connect

App Transport Security requirements apply

HTTP POST to your server for status changes

- Initial purchase of a subscription
- Subscription cancellation by AppleCare
- Subscription downgrades
- Successful renewal or re-purchase for an expired subscription

Server Notifications

NEW

Status URL in iTunes Connect

App Transport Security requirements apply

HTTP POST to your server for status changes

- Initial purchase of a subscription
- Subscription cancellation by AppleCare
- Subscription downgrades
- Successful renewal or re-purchase for an expired subscription

Includes latest transaction for the transaction in question

Server Notifications

NEW

Server Notifications

NEW

No longer need to poll verifyReceipt every day

Server Notifications

NEW

No longer need to poll verifyReceipt every day

Still call verifyReceipt to update your state

Server Notifications

NEW

No longer need to poll verifyReceipt every day

Still call verifyReceipt to update your state

- App Store notification may not reach your server

Server Notifications

NEW

No longer need to poll verifyReceipt every day

Still call verifyReceipt to update your state

- App Store notification may not reach your server
- Be smart about when to poll

Server Notifications

NEW

No longer need to poll verifyReceipt every day

Still call verifyReceipt to update your state

- App Store notification may not reach your server
- Be smart about when to poll
  - Day before, day of expiry date

Server Notifications

NEW

No longer need to poll verifyReceipt every day

Still call verifyReceipt to update your state

- App Store notification may not reach your server
- Be smart about when to poll
  - Day before, day of expiry date

Notifications coming later this year

Server-Side Subscriptions

Server-Side Subscriptions

Why did a user's subscription expire?

What does the user need to know about their subscription?

Will this user's subscription be renewed?

Why did AppleCare give the user a refund?

Has the user agreed to a price increase?

Will the user be downgraded after this billing cycle?

Server-Side Subscriptions

New fields in verifyReceipt endpoint

NEW

Server-Side Subscriptions

New fields in verifyReceipt endpoint

NEW

Server-Side Subscriptions

New fields in verifyReceipt endpoint

NEW

Server-Side Subscriptions

New fields in verifyReceipt endpoint

NEW

Now included in JSON from verifyReceipt

Server-Side Subscriptions

New fields in verifyReceipt endpoint

NEW

Now included in JSON from verifyReceipt

- Auto-Renew Status

Server-Side Subscriptions

New fields in verifyReceipt endpoint

NEW

Now included in JSON from verifyReceipt

- Auto-Renew Status
- Auto-Renew Preference

Server-Side Subscriptions

New fields in verifyReceipt endpoint

NEW

Now included in JSON from verifyReceipt

- Auto-Renew Status
- Auto-Renew Preference
- Price Consent Status

Server-Side Subscriptions

New fields in verifyReceipt endpoint

NEW

Now included in JSON from verifyReceipt

- Auto-Renew Status
- Auto-Renew Preference
- Price Consent Status
- Subscription Retry Flag

Server-Side Subscriptions

New fields in verifyReceipt endpoint

NEW

Now included in JSON from verifyReceipt

- Auto-Renew Status
- Auto-Renew Preference
- Price Consent Status
- Subscription Retry Flag
- Expiration Intent

Server-Side Subscriptions

New fields in verifyReceipt endpoint

NEW

Now included in JSON from verifyReceipt

- Auto-Renew Status
- Auto-Renew Preference
- Price Consent Status
- Subscription Retry Flag
- Expiration Intent
- Cancellation Reason

Keeping Your Subscribers

Involuntary expiration

Keeping Your Subscribers

Involuntary expiration

Expanded retry window up to 60 days

Keeping Your Subscribers

Involuntary expiration

Expanded retry window up to 60 days

Use the Expiration Intent and Subscription Retry Flag to

Keeping Your Subscribers

Involuntary expiration

Expanded retry window up to 60 days

Use the Expiration Intent and Subscription Retry Flag to

1. Provide messaging to user

Keeping Your Subscribers

Involuntary expiration

Expanded retry window up to 60 days

Use the Expiration Intent and Subscription Retry Flag to

1. Provide messaging to user
2. Offer a downgraded or temporary experience

Keeping Your Subscribers

Involuntary expiration

Expanded retry window up to 60 days

Use the Expiration Intent and Subscription Retry Flag to

1. Provide messaging to user
2. Offer a downgraded or temporary experience
3. Unblock your user immediately if subscription renews

Keeping Your Subscribers

Voluntary expiration

Keeping Your Subscribers

Voluntary expiration

Use the Expiration Intent field to

Keeping Your Subscribers

Voluntary expiration

Use the Expiration Intent field to

- Offer messaging to user

Keeping Your Subscribers

Voluntary expiration

Use the Expiration Intent field to

- Offer messaging to user

User canceled?

Keeping Your Subscribers

Voluntary expiration

Use the Expiration Intent field to

- Offer messaging to user

User canceled?

- Apply win-back

Keeping Your Subscribers

Voluntary expiration

Use the Expiration Intent field to

- Offer messaging to user

User canceled?

- Apply win-back

Did not consent to higher price?

Keeping Your Subscribers

Voluntary expiration

Use the Expiration Intent field to

- Offer messaging to user

User canceled?

- Apply win-back

Did not consent to higher price?

- Offer attractive downgrade option

Server-Side Subscriptions

New fields in verifyReceipt endpoint

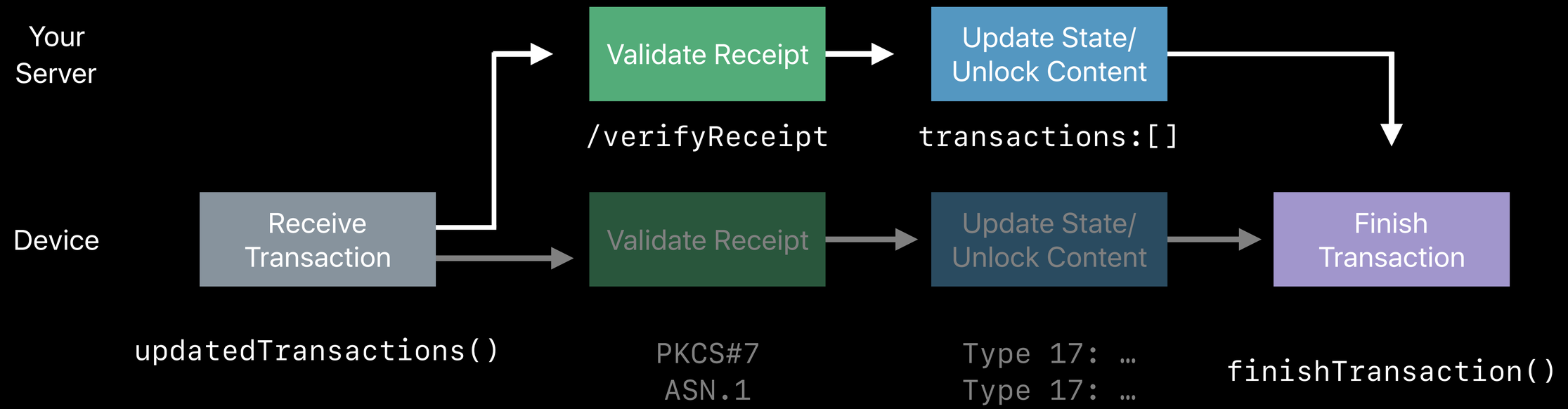
Server-Side Subscriptions

New fields in verifyReceipt endpoint

- Auto-Renew Status
- Auto-Renew Preference
- Price Consent Status
- Subscription Retry Flag
- Expiration Intent
- Cancellation Reason

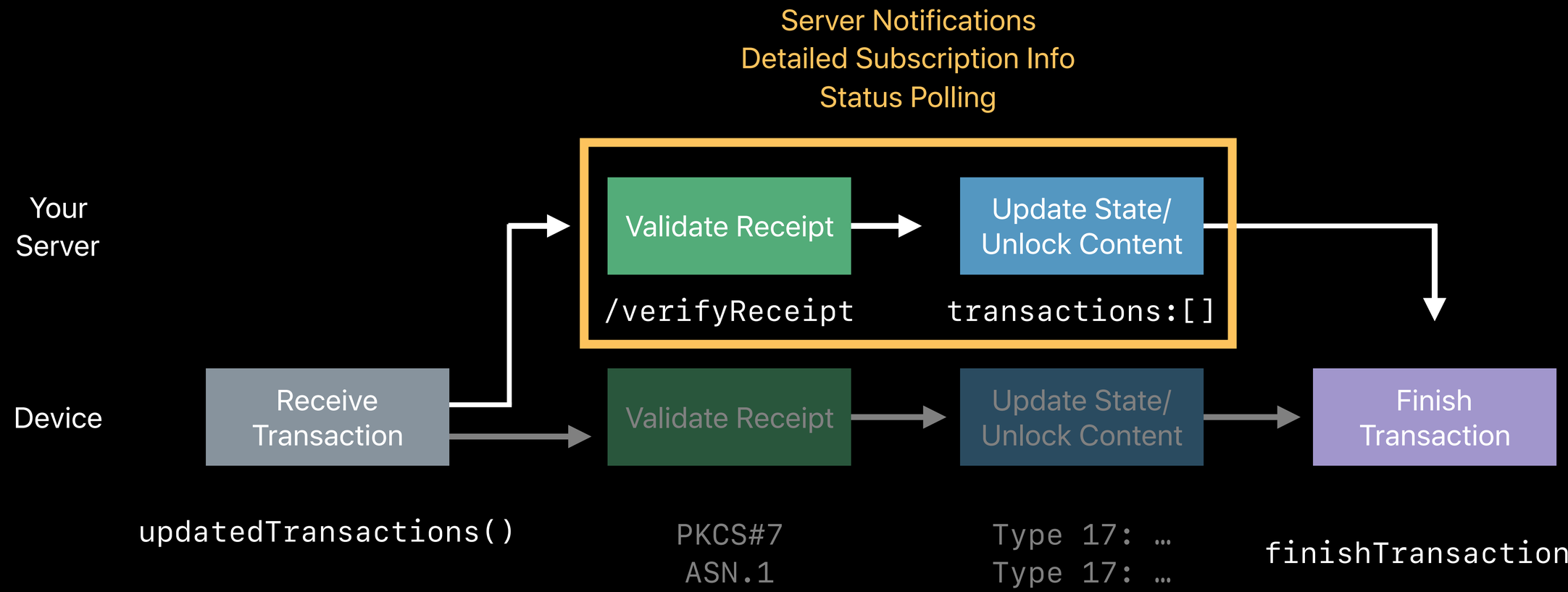
In-App Purchase Process

Processing transactions



In-App Purchase Process

Processing transactions



Auto-Renewable Subscriptions

Free trials

Auto-Renewable Subscriptions

Free trials

Subscription begins immediately

Auto-Renewable Subscriptions

Free trials

Subscription begins immediately

Not billed until free trial period is over

Free Trials

Previously

Subscription Length	Available Free Trial Duration	Available Bonus Duration
7 days	7 days	7 days
1 month	7 days 1 month	7 days 1 month
2 months	7 days 1 month	7 days 1 month
3 months	1 month	1 month
6 months	1 month 2 months	1 month 2 months
1 year	1 month 2 months 3 months	1 month 2 months 3 months

Free Trials

Recent changes to trial durations

NEW

Subscription Length	Available Free Trial Duration	Available Bonus Duration
Any	3 days 1 week 2 weeks 1 month 2 months 3 months 6 months 1 year	3 days 1 week 2 weeks 1 month 2 months 3 months 6 months 1 year

Free Trials

Recent changes to trial durations

NEW

Subscription Length	Available Free Trial Duration	Available Bonus Duration
Any	**3 days** 1 week **2 weeks** 1 month 2 months 3 months 6 months 1 year	**3 days** 1 week **2 weeks** 1 month 2 months 3 months 6 months 1 year

Developing with the Sandbox

The Sandbox

The Sandbox

Environment for testing purchases during development

The Sandbox

Environment for testing purchases during development

Based on certificate used to sign your application

The Sandbox

Environment for testing purchases during development

Based on certificate used to sign your application

Developer Certificate

The Sandbox

Environment for testing purchases during development

Based on certificate used to sign your application

The Sandbox

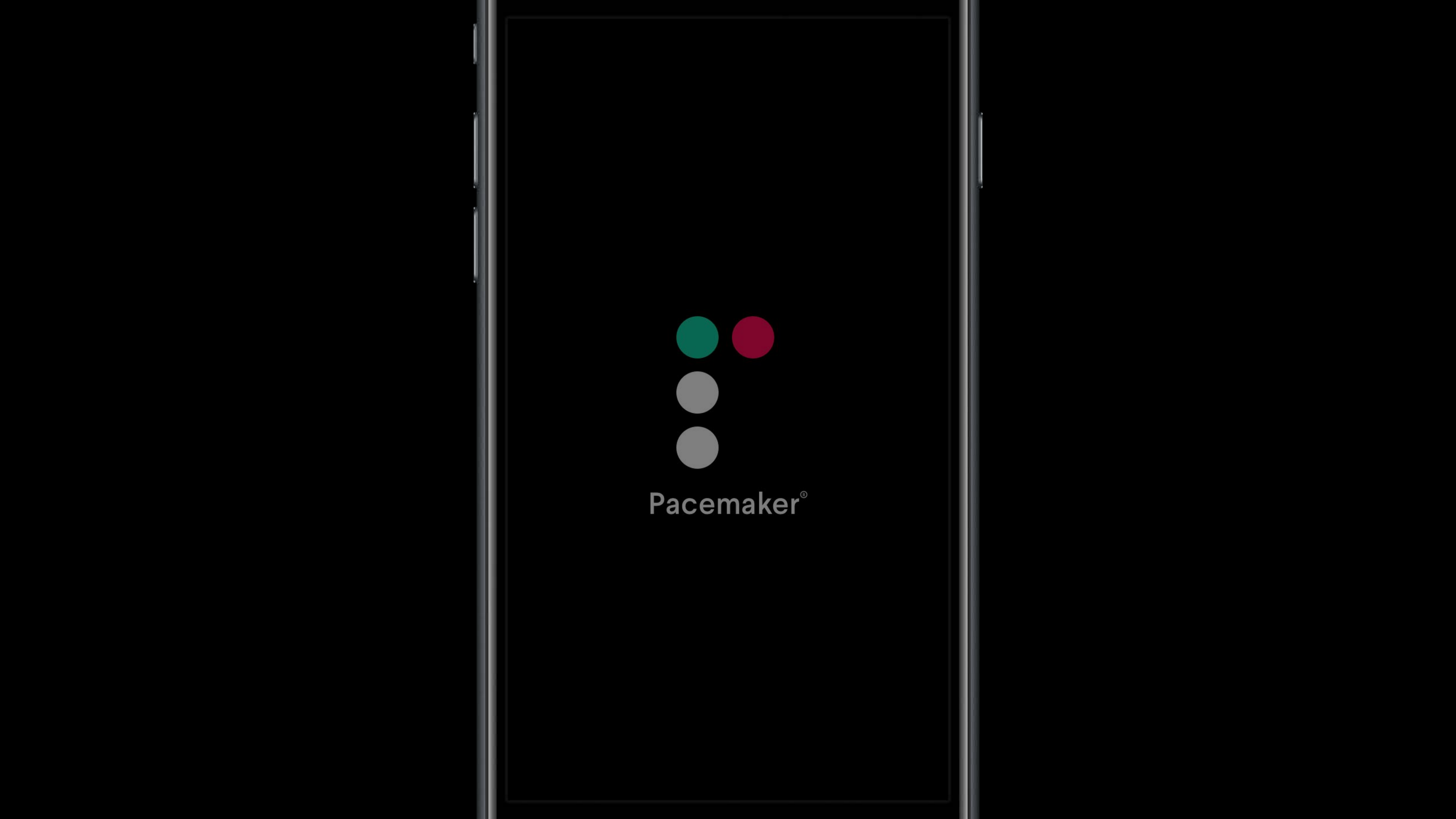
Environment for testing purchases during development

Based on certificate used to sign your application

The Sandbox

Environment for testing purchases during development

Based on certificate used to sign your application


Pacemaker®

App Store [SANDBOX] Cancel

BEATSKIP
PACEMAKER
IN-APP PURCHASE

RATING 4+

ACCOUNT J.APPLESEED@ICLOUD.COM

PAY APP STORE $1.99

Buy with Touch ID

App St[illegible] [SANDBOX] Cancel

BEATSKIP
PACEMAKER
IN-APP PURCHASE

RATING 4+

ACCOUNT J.APPLESEED@ICLOUD.COM

PAY APP STORE $1.99

Buy with Touch ID

The Sandbox

Differences

The Sandbox

Differences

No charge

The Sandbox

Differences

No charge

Different endpoint for server-to-server validation for App Store Sandbox servers

The Sandbox

Differences

No charge

Different endpoint for server-to-server validation for App Store Sandbox servers

Can request expired and/or revoked receipts

The Sandbox

Differences

No charge

Different endpoint for server-to-server validation for App Store Sandbox servers

Can request expired and/or revoked receipts

Time contraction for subscriptions

Subscription Timing

Condensed intervals

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

Subscription Timing

Condensed intervals

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

The Sandbox

Setting up the test environment

The Sandbox

Setting up the test environment

Setup in iTunes Connect

- Create test user
- Enter products for sale

The Sandbox

Setting up the test environment

Setup in iTunes Connect

- Create test user
- Enter products for sale

Build and sign your app

The Sandbox

Setting up the test environment

Setup in iTunes Connect

- Create test user
- Enter products for sale

Build and sign your app

Buy products

- Sign in with test user when prompted

The Sandbox

Setting up the test environment

Setup in iTunes Connect

- Create test user
- Enter products for sale

Build and sign your app

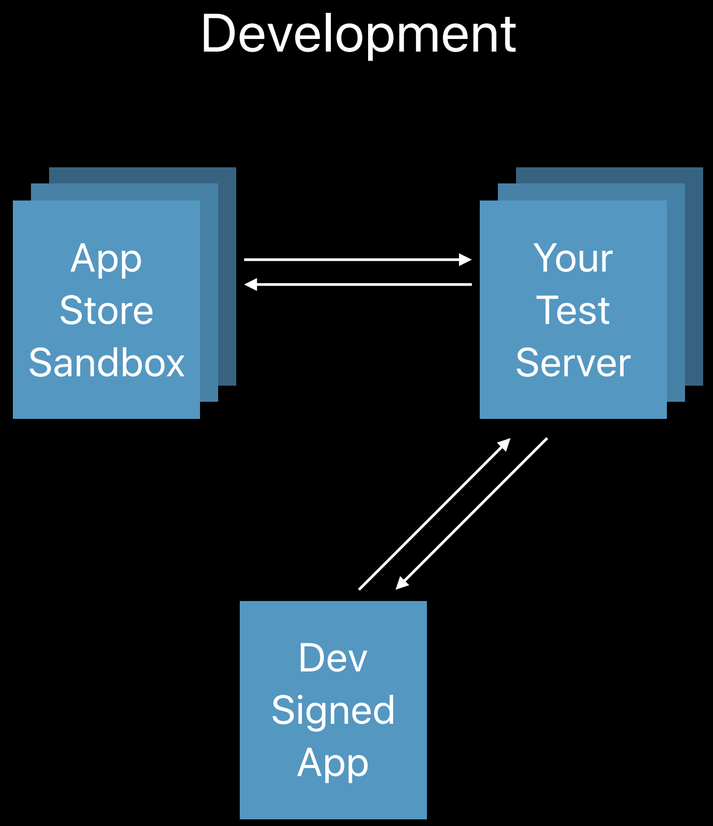
Buy products

- Sign in with test user when prompted

macOS only: Launch app from Finder once to fetch receipt

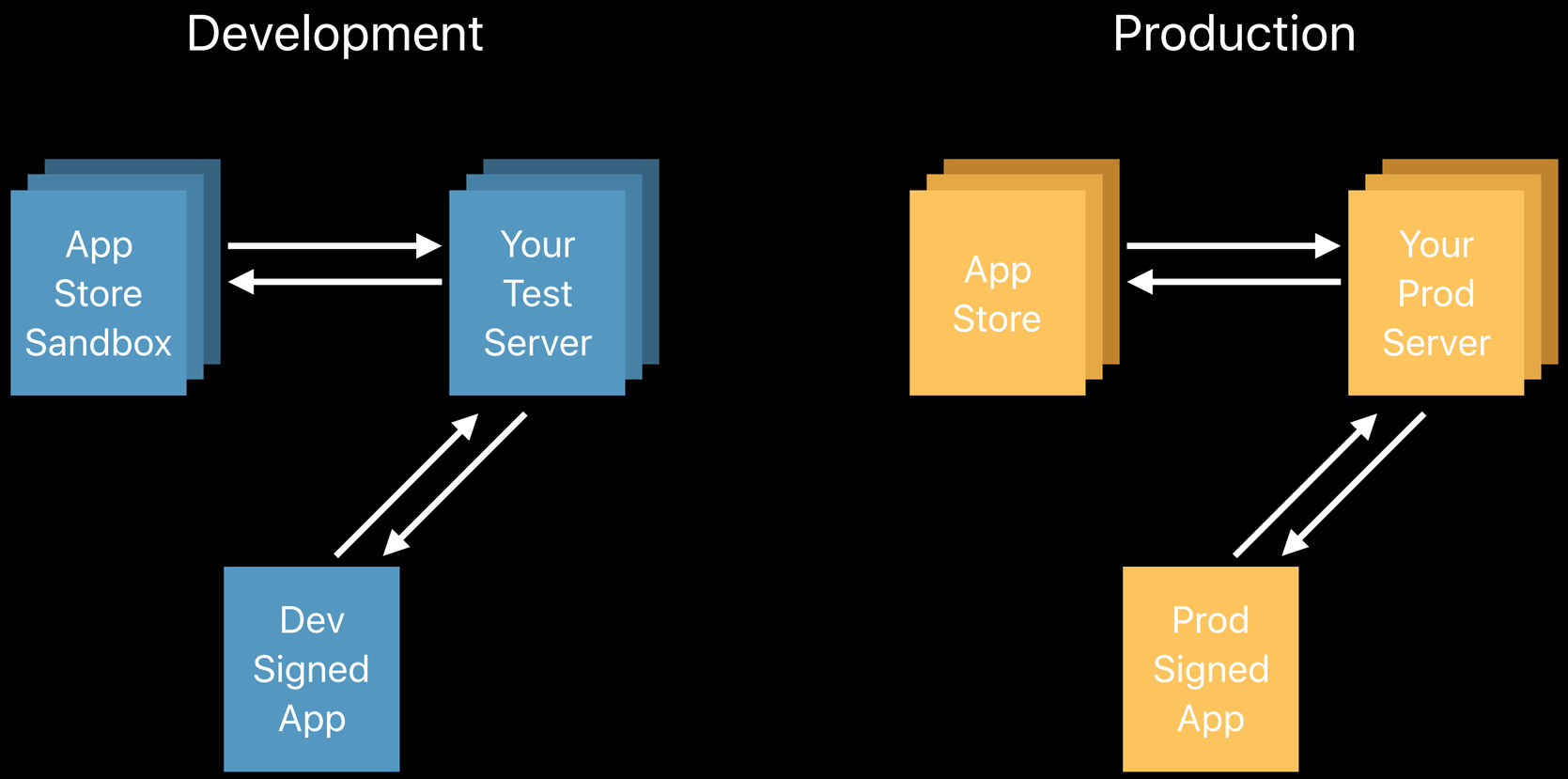
The Sandbox

Using the Sandbox from server



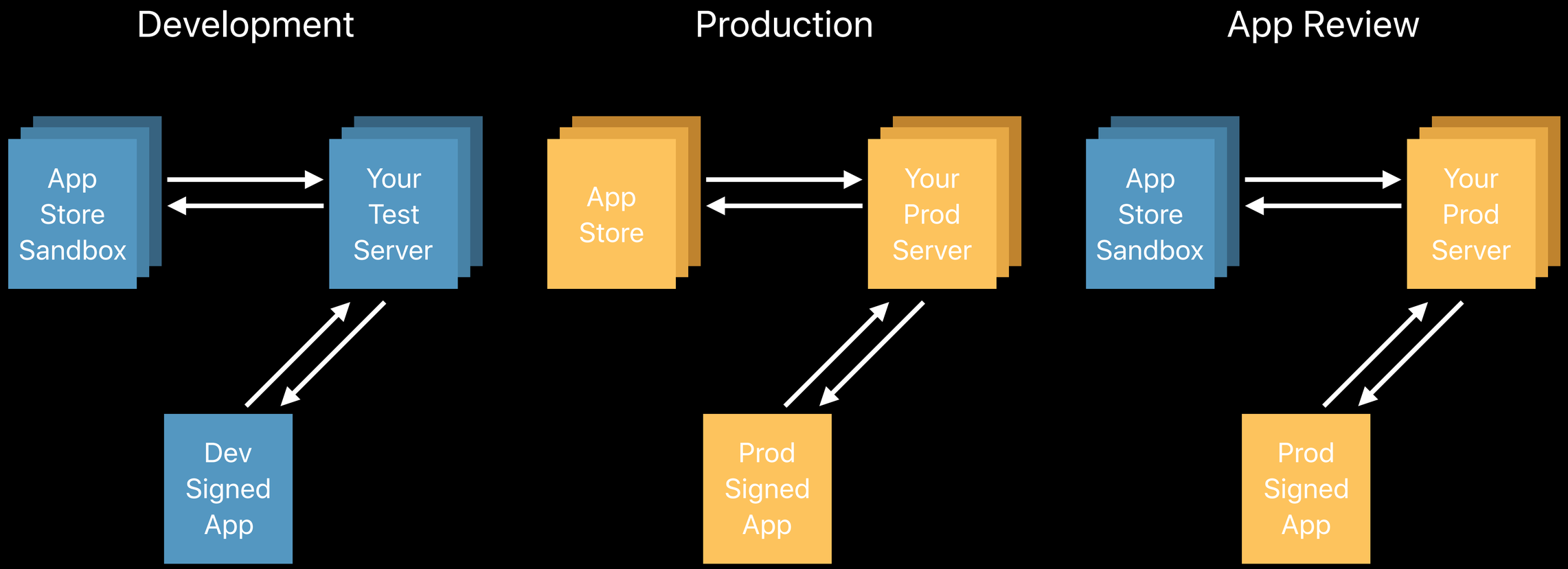
The Sandbox

Using the Sandbox from server



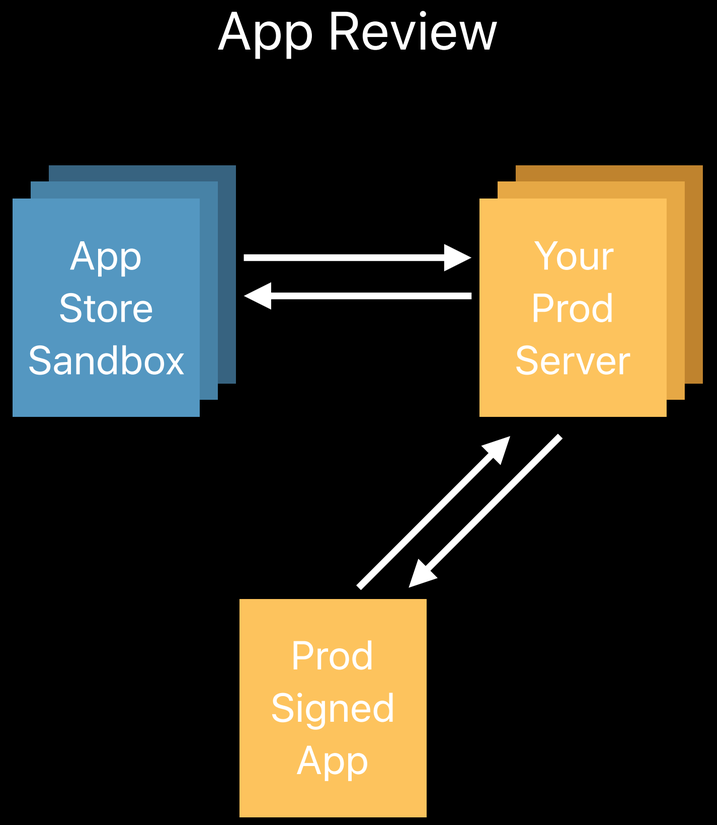
The Sandbox

Using the Sandbox from server



The Sandbox

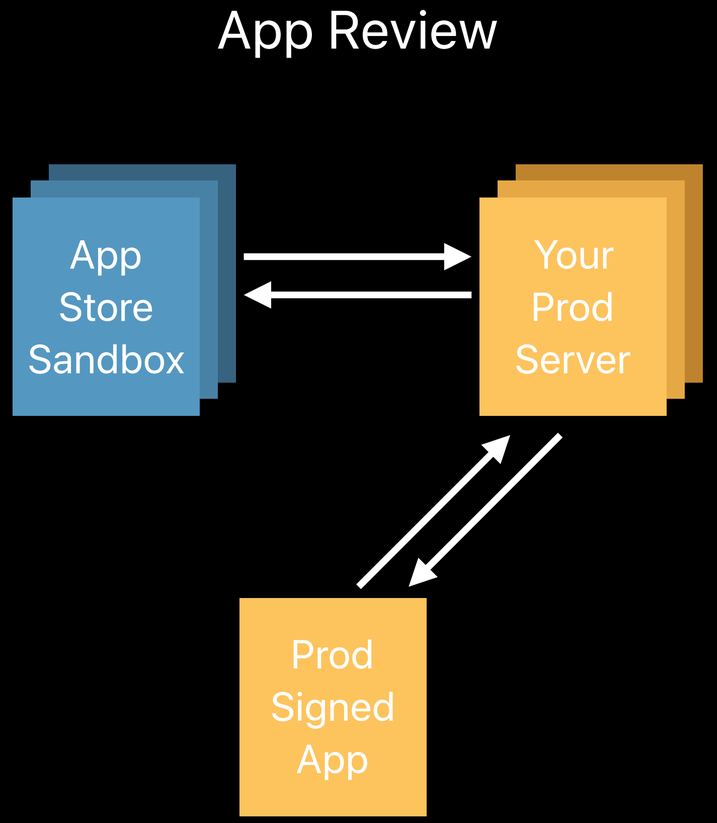
App review considerations



The Sandbox

App review considerations

Try the production environment



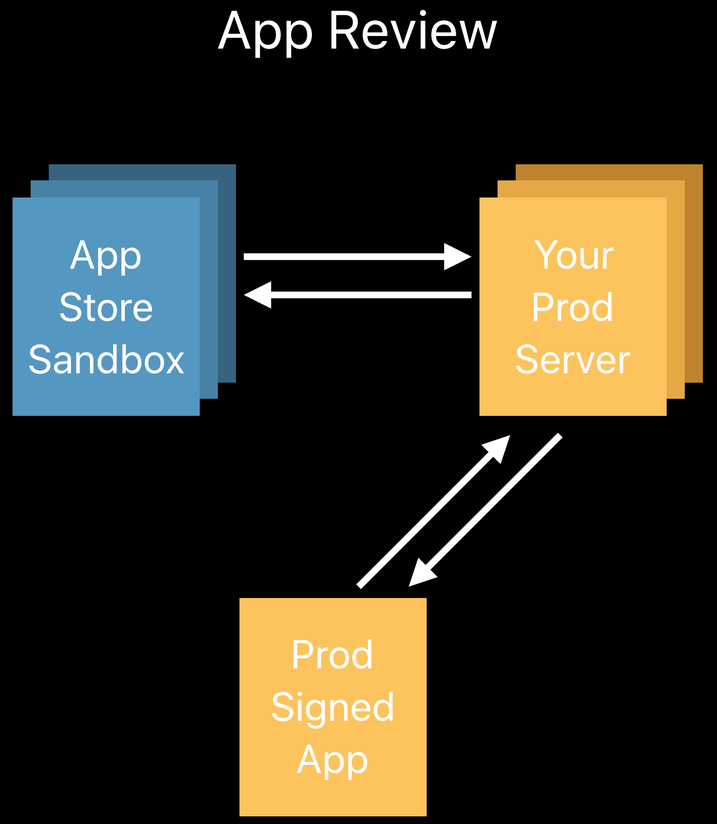
The Sandbox

App review considerations

Try the production environment

If receipt is from Sandbox

- Receive error `21007`



The Sandbox

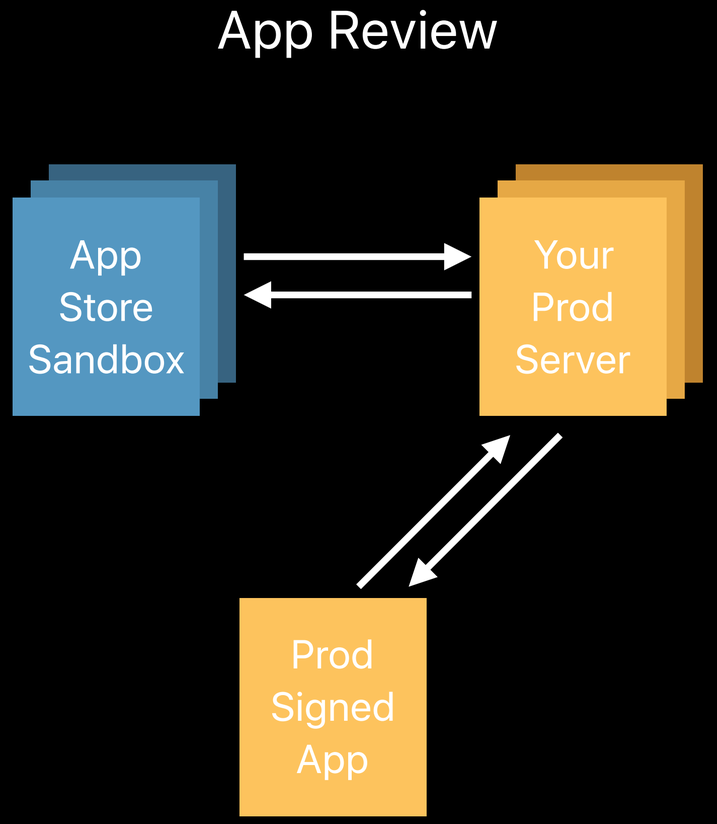
App review considerations

Try the production environment

If receipt is from Sandbox

- Receive error `21007`

Then try against Sandbox



The Sandbox

Server notifications

The Sandbox

Server notifications

Not separate production and Sandbox servers

The Sandbox

Server notifications

Not separate production and Sandbox servers

Sandboxing is handled by a parameter in the payload

The Sandbox

Server notifications

Not separate production and Sandbox servers

Sandboxing is handled by a parameter in the payload

- Use the `environment` key in payload

Summary

Summary

Receipt Validation

Summary

Receipt Validation

Maintaining Subscription State

Summary

Receipt Validation

Maintaining Subscription State

Server-to-Server Notifications

Summary

Receipt Validation

Maintaining Subscription State

Server-to-Server Notifications

New Receipt Fields

Summary

Receipt Validation

Maintaining Subscription State

Server-to-Server Notifications

New Receipt Fields

Easy Steps to Retain Subscribers

Summary

Receipt Validation

Maintaining Subscription State

Server-to-Server Notifications

New Receipt Fields

Easy Steps to Retain Subscribers

Developing with the Sandbox

More Information

https://developer.apple.com/wwdc17/305

Related Sessions

What’s New in iTunes Connect	WWDC 2017
Introducing the New App Store	WWDC 2017
What's New in StoreKit	WWDC 2017

Labs

StoreKit Lab	Technology Lab E	Thu 3:10PM-6:00PM
StoreKit Lab	Technology Lab E	Fri 1:50PM-4:00PM


WWDC17