#WWDC18

Engineering Subscriptions

Implementation Best Practices

Session 705

Pete Hare, Engineering Manager, App Store
Michael Gargas, Technical Advocate, App Store Operations

Device and Server Architecture

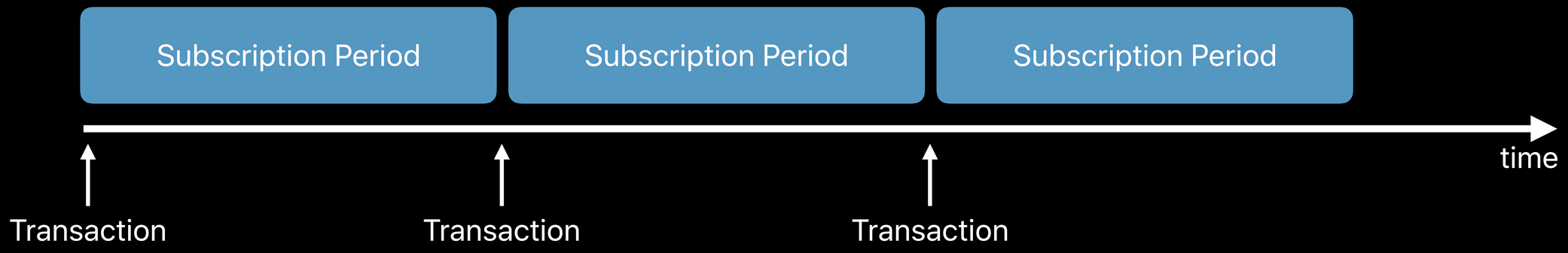
In-App Experience

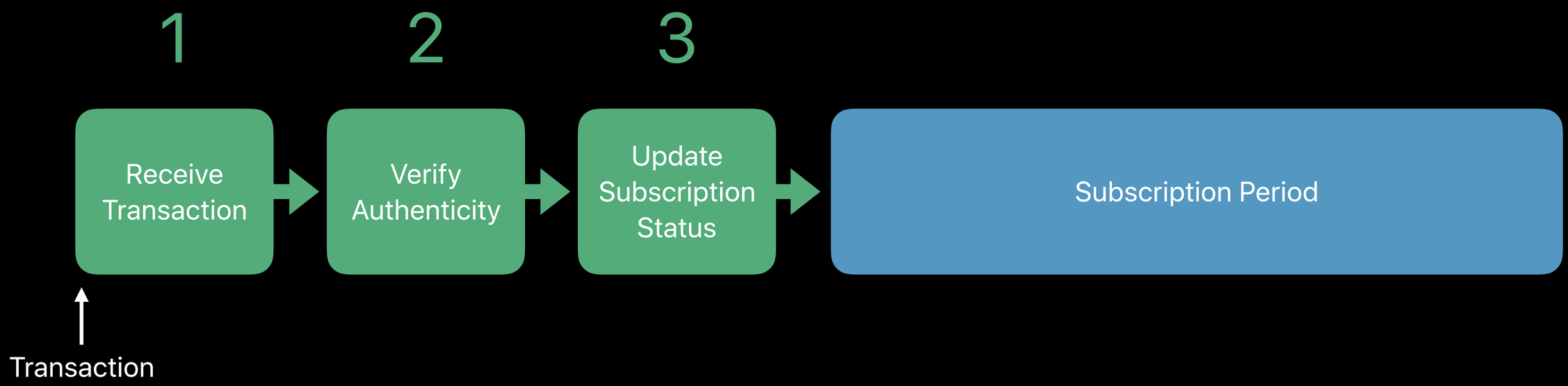
Reducing Subscriber Churn

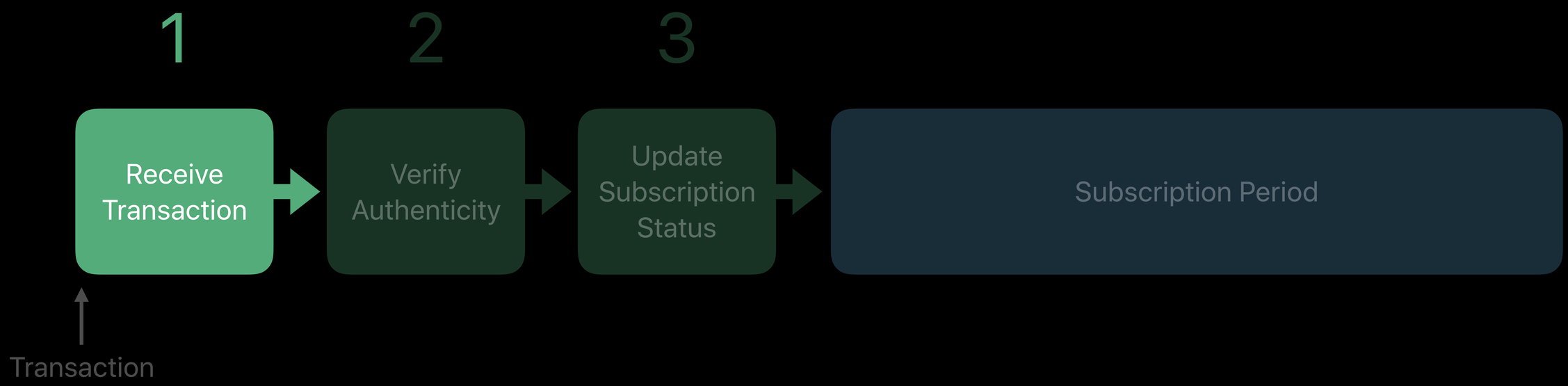
Analytics and Reporting

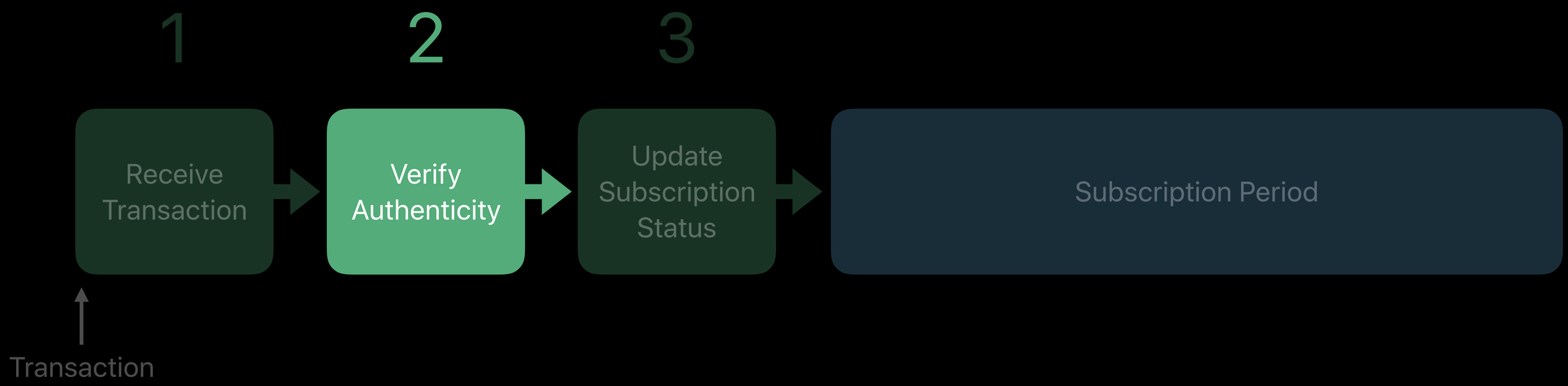
Device and Server Architecture

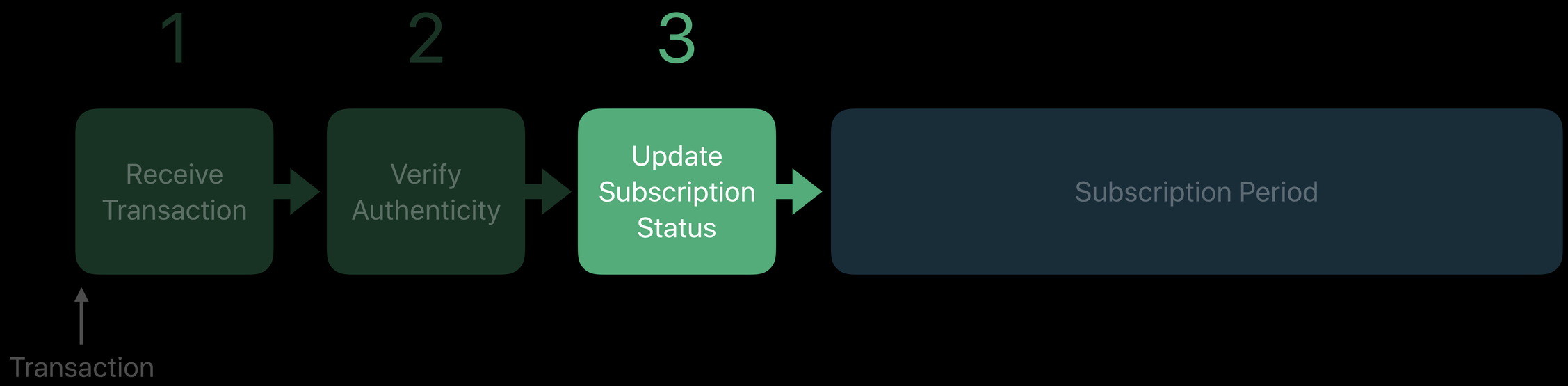
What is a subscription?

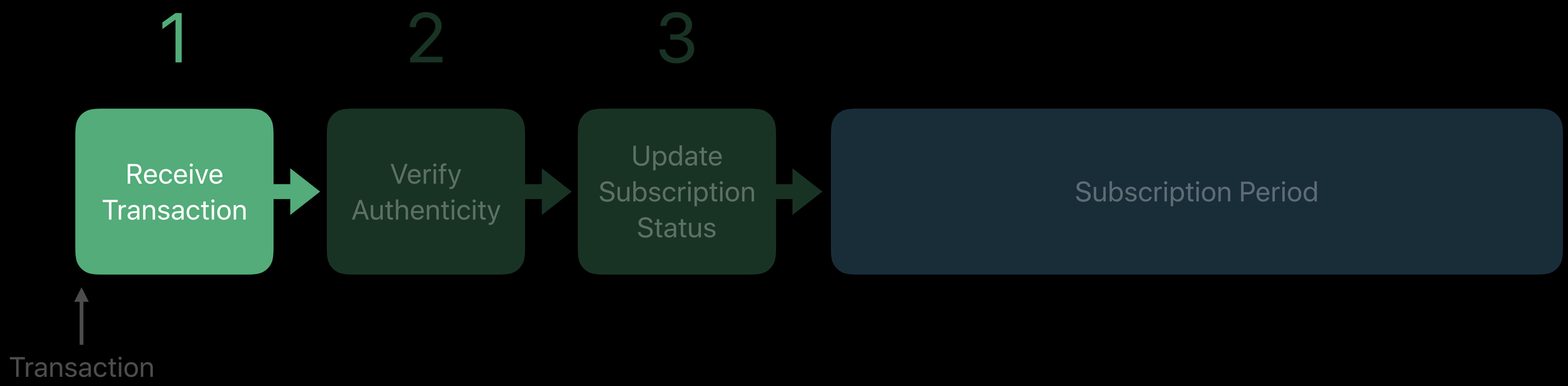

Subscription Period
Subscription Period
Subscription Period
time
Transaction
Transaction
Transaction

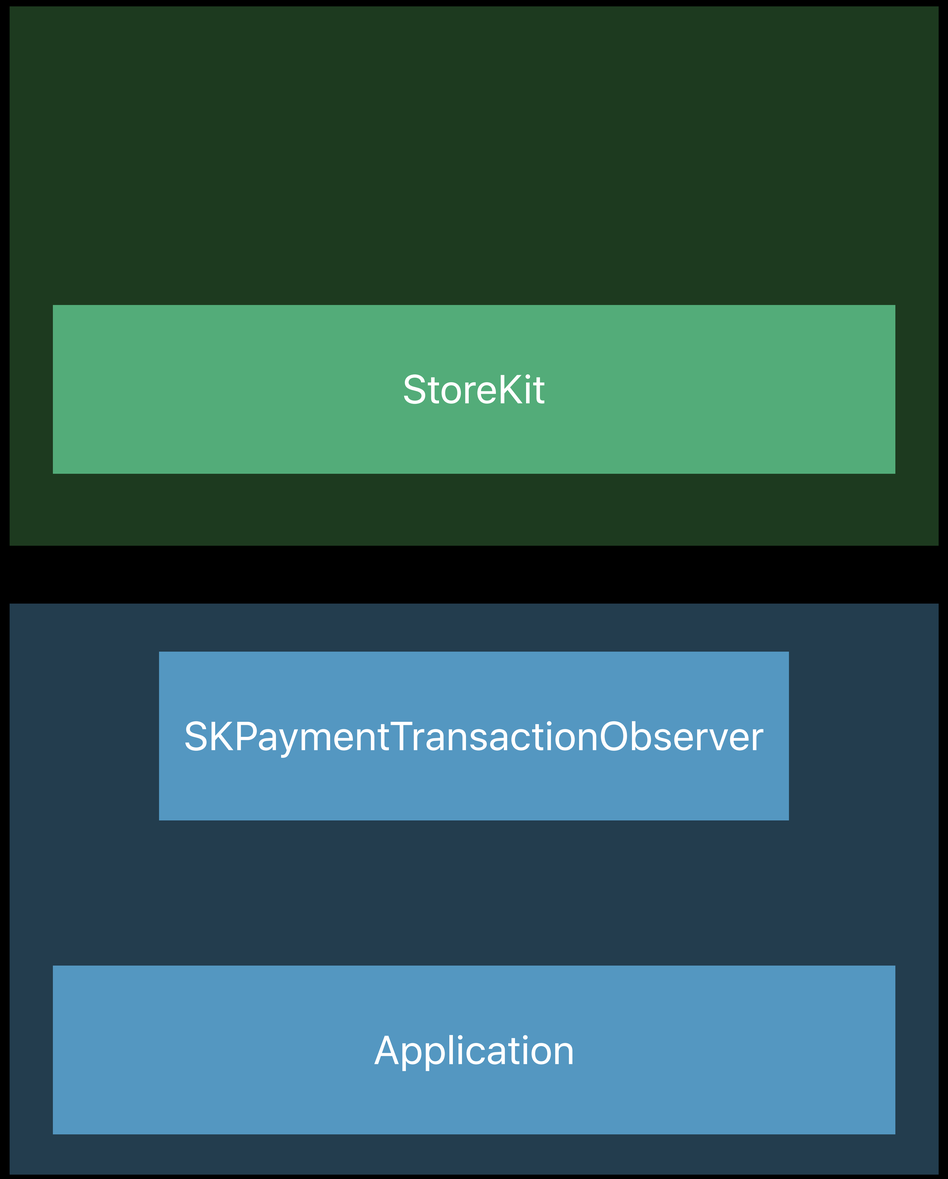

1
2
3
Receive Transaction
Verify Authenticity
Update Subscription Status
Subscription Period
Transaction

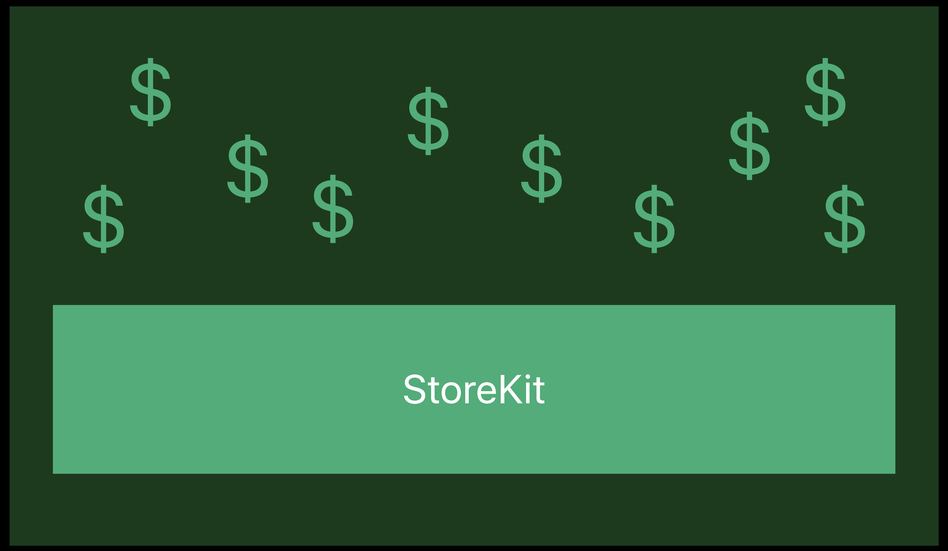

1
2
3
Receive Transaction
Verify Authenticity
Update Subscription Status
Subscription Period
Transaction


1
2
3
Receive Transaction
Verify Authenticity
Update Subscription Status
Subscription Period
Transaction


1
2
3
Receive Transaction
Verify Authenticity
Update Subscription Status
Subscription Period
Transaction

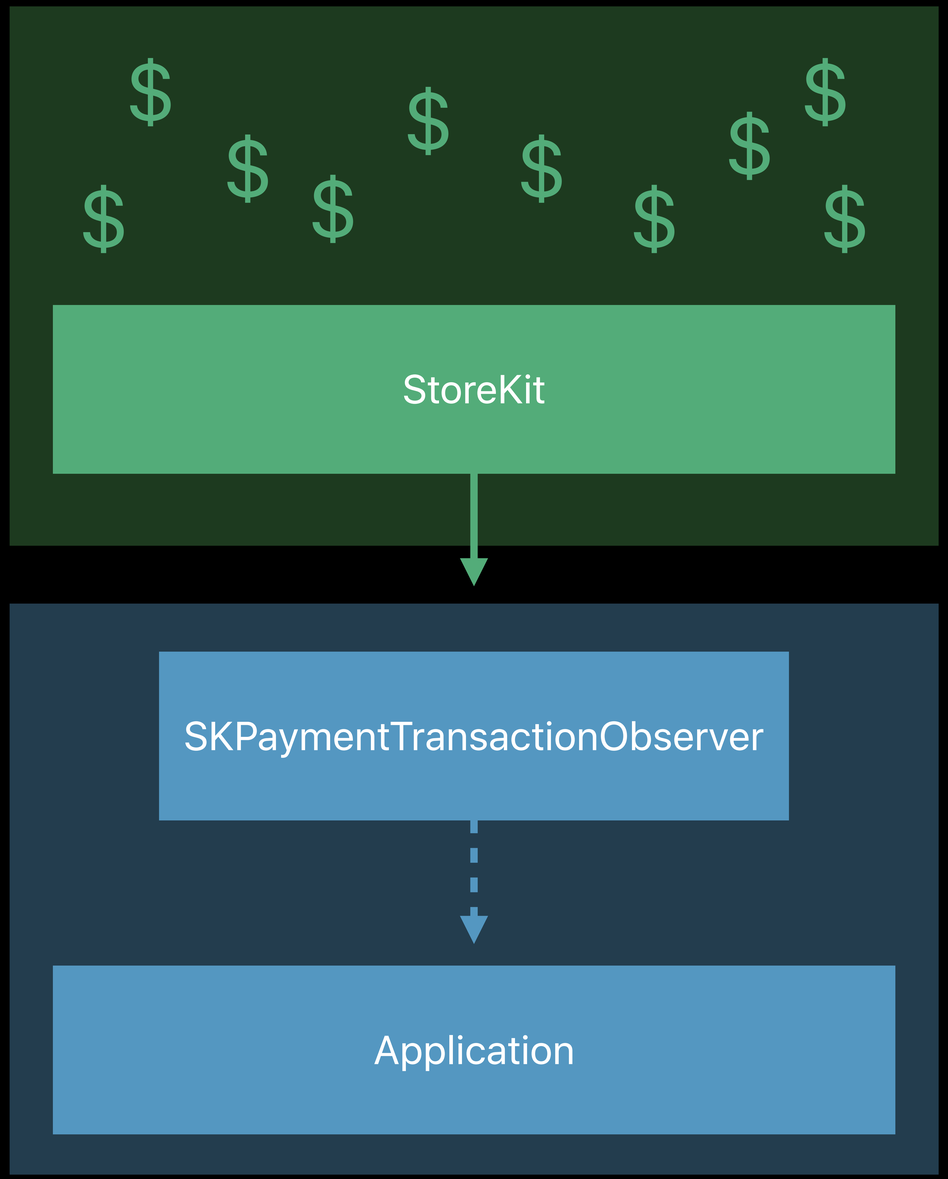

1
2
3
Receive Transaction
Verify Authenticity
Update Subscription Status
Subscription Period
Transaction


StoreKit
SKPaymentTransactionObserver
Application



SKPaymentTransactionObserver

Application


$
$
$
$
$
$
$
$
$
$
StoreKit
SKPaymentTransactionObserver
Application

```swift
// 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
// 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
// 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
    }

}
```

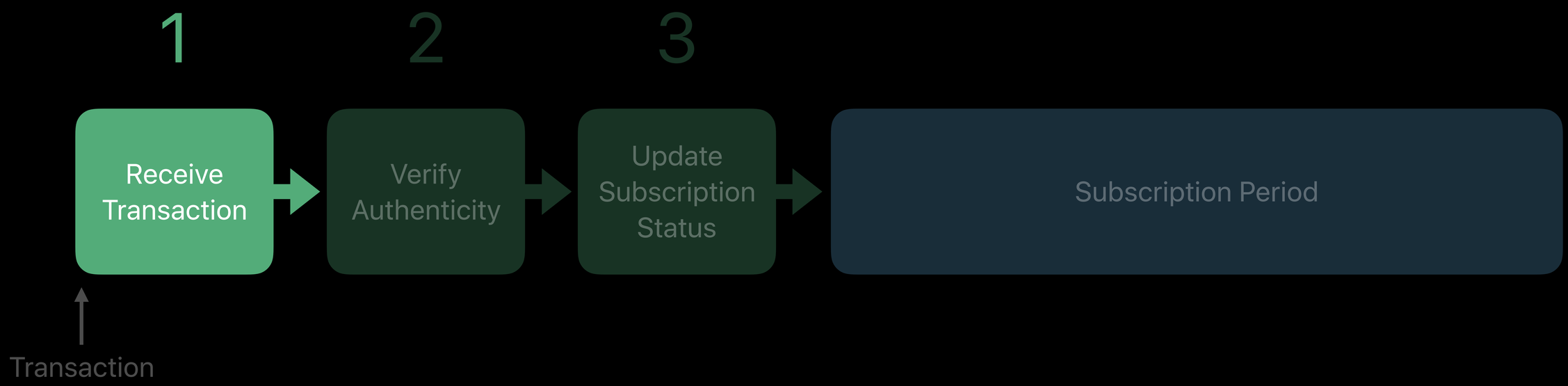
```swift
// Start Observing the Payment Queue

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
    }

}
```

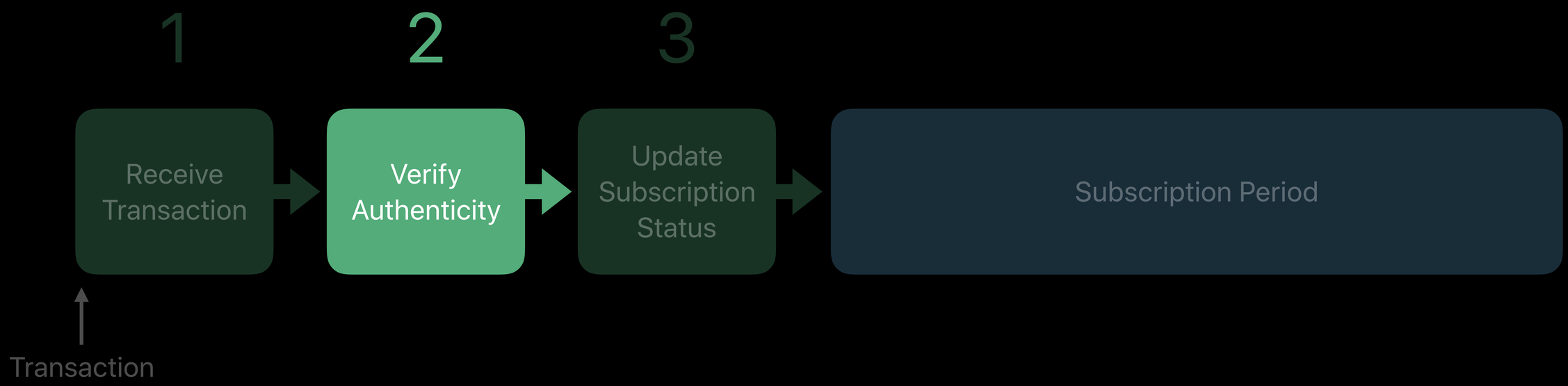
```
class AppDelegate: UIResponder, UIApplicationDelegate, SKPaymentTransactionObserver {

    // 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
            }
        }
    }
}
```


1
2
3
Receive Transaction
Verify Authenticity
Update Subscription Status
Subscription Period
Transaction


1
2
3
Receive Transaction
Verify Authenticity
Update Subscription Status
Subscription Period
Transaction

App Store Receipt

App Store Receipt

Trusted record of App and In-App Purchases

App Store Receipt

Trusted record of App and In-App Purchases

Stored on device

App Store Receipt

Trusted record of App and In-App Purchases

Stored on device

Issued by the App Store

App Store Receipt

Trusted record of App and In-App Purchases

Stored on device

Issued by the App Store

Signed and verifiable

App Store 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

Don’t let your app be fooled!

Receipt Validation

Don’t let your app be fooled!

On-device validation

- Update subscription state within the app

Receipt Validation

Don’t let your app be fooled!

On-device validation

- Update subscription state within the app

Server-to-server validation

- Online validation through a request to the App Store
- Update subscription state on your server

Receipt Validation

Don’t let your app be fooled!

On-device validation

- Update subscription state within the app

Server-to-server validation

- Online validation through a request to the App Store
- Update subscription state on your server

Do not use online validation directly from the device!

Comparing Validation Approaches

Comparing Validation Approaches

	On-Device Receipt Validation	Server-to-Server Receipt Validation

Comparing Validation Approaches

	On-Device Receipt Validation	Server-to-Server Receipt Validation
Validate authenticity of receipt		

Comparing Validation Approaches

	On-Device Receipt Validation	Server-to-Server Receipt Validation
Validate authenticity of receipt		
Receipt includes renewal transactions		

Comparing Validation Approaches

	On-Device Receipt Validation	Server-to-Server Receipt Validation
Validate authenticity of receipt	✓	✓
Receipt includes renewal transactions	✓	✓
Additional user subscription information	✗	✓

Comparing Validation Approaches

	On-Device Receipt Validation	Server-to-Server Receipt Validation
Validate authenticity of receipt	✓	✓
Receipt includes renewal transactions	✓	✓
Additional user subscription information	✗	✓
Always “on” to handle renewals	✗	✓

Comparing Validation Approaches

	On-Device Receipt Validation	Server-to-Server Receipt Validation
Validate authenticity of receipt	✓	✓
Receipt includes renewal transactions	✓	✓
Additional user subscription information	✗	✓
Always “on” to handle renewals	✗	✓
Not susceptible to device “clock change”	✗	✓

Comparing Validation Approaches

	On-Device Receipt Validation	Server-to-Server Receipt Validation
Validate authenticity of receipt	✓	✓
Receipt includes renewal transactions	✓	✓
Additional user subscription information	✗	✓
Always “on” to handle renewals	✗	✓
Not susceptible to device “clock change”	✗	✓
No cryptography needed (OpenSSL, ASN.1)	✗	✓

Comparing Validation Approaches

	On-Device Receipt Validation	Server-to-Server Receipt Validation
Validate authenticity of receipt	✓	✓
Receipt includes renewal transactions	✓	✓
Additional user subscription information	✗	✓
Always “on” to handle renewals	✗	✓
Not susceptible to device “clock change”	✗	✓
No cryptography needed (OpenSSL, ASN.1)	✗	✓

Comparing Validation Approaches

	On-Device Receipt Validation	Server-to-Server Receipt Validation
Validate authenticity of receipt	✓	✓
Receipt includes renewal transactions	✓	✓
Additional user subscription information	✗	✓
Always “on” to handle renewals	✗	✓
Not susceptible to device “clock change”	✗	✓
No cryptography needed (OpenSSL, ASN.1)	✗	✓

Advanced StoreKit https://developer.apple.com/videos/play/wwdc2017/305/ WWDC 2017

```
switch transaction.transactionState {
    case .purchased:

        if let appStoreReceiptURL = Bundle.main.appStoreReceiptURL,
          FileManager.default.fileExists(atPath: appStoreReceiptURL.path) {

            let rawReceiptData = Data(contentsOf: appStoreReceiptURL)
            let receiptData = rawReceiptData.base64EncodedString(options: …)

            // Send up receiptString to server
            currentUser.processTransaction(receiptData) { isValid in

            }
        }
}
```

```
switch transaction.transactionState {
    case .purchased:

        if let appStoreReceiptURL = Bundle.main.appStoreReceiptURL,
          FileManager.default.fileExists(atPath: appStoreReceiptURL.path) {

            let rawReceiptData = Data(contentsOf: appStoreReceiptURL)
            let receiptData = rawReceiptData.base64EncodedString(options: …)

            // Send up receiptString to server
            currentUser.processTransaction(receiptData) { isValid in

            }
        }
}
```

```
switch transaction.transactionState {
    case .purchased:

        if let appStoreReceiptURL = Bundle.main.appStoreReceiptURL,
          FileManager.default.fileExists(atPath: appStoreReceiptURL.path) {

            let rawReceiptData = Data(contentsOf: appStoreReceiptURL)
            let receiptData = rawReceiptData.base64EncodedString(options: …)

            // Send up receiptString to server
            currentUser.processTransaction(receiptData) { isValid in

            }
        }
}
```

```
switch transaction.transactionState {
    case .purchased:

        if let appStoreReceiptURL = Bundle.main.appStoreReceiptURL,
          FileManager.default.fileExists(atPath: appStoreReceiptURL.path) {

            let rawReceiptData = Data(contentsOf: appStoreReceiptURL)
            let receiptData = rawReceiptData.base64EncodedString(options: …)

            // Send up receiptString to server
            currentUser.processTransaction(receiptData) { isValid in

            }
        }
}
```

```
switch transaction.transactionState {
    case .purchased:

        if let appStoreReceiptURL = Bundle.main.appStoreReceiptURL,
          FileManager.default.fileExists(atPath: appStoreReceiptURL.path) {

            let rawReceiptData = Data(contentsOf: appStoreReceiptURL)
            let receiptData = rawReceiptData.base64EncodedString(options: …)

            // Send up receiptString to server
            currentUser.processTransaction(receiptData) { isValid in

            }
        }
}
```

```
switch transaction.transactionState {
    case .purchased:

        if let appStoreReceiptURL = Bundle.main.appStoreReceiptURL,
          FileManager.default.fileExists(atPath: appStoreReceiptURL.path) {

            let rawReceiptData = Data(contentsOf: appStoreReceiptURL)
            let receiptData = rawReceiptData.base64EncodedString(options: …)

            // Send up receiptString to server
            currentUser.processTransaction(receiptData) { isValid in

            }
        }
}
```

Verifying a Transaction

Your Server

/processTransaction?userId=90000001

Verifying a Transaction

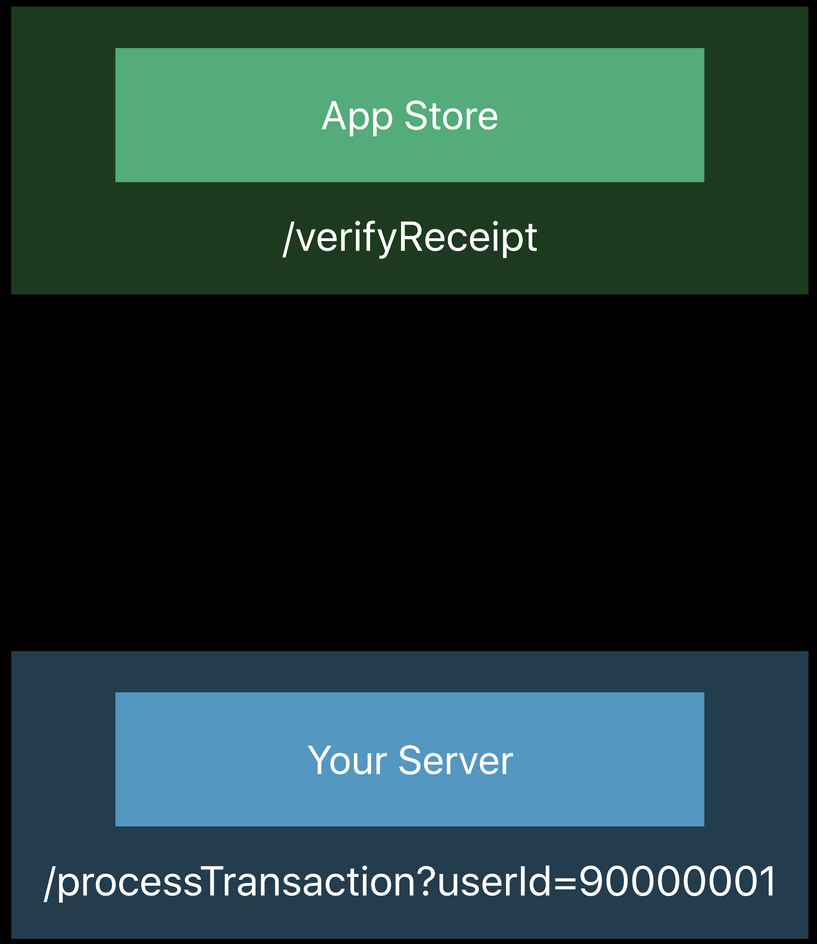
Your Server

/processTransaction?userId=90000001

```
{ receiptData: “e2FzZ…jg3OTgwN30=“ }
```

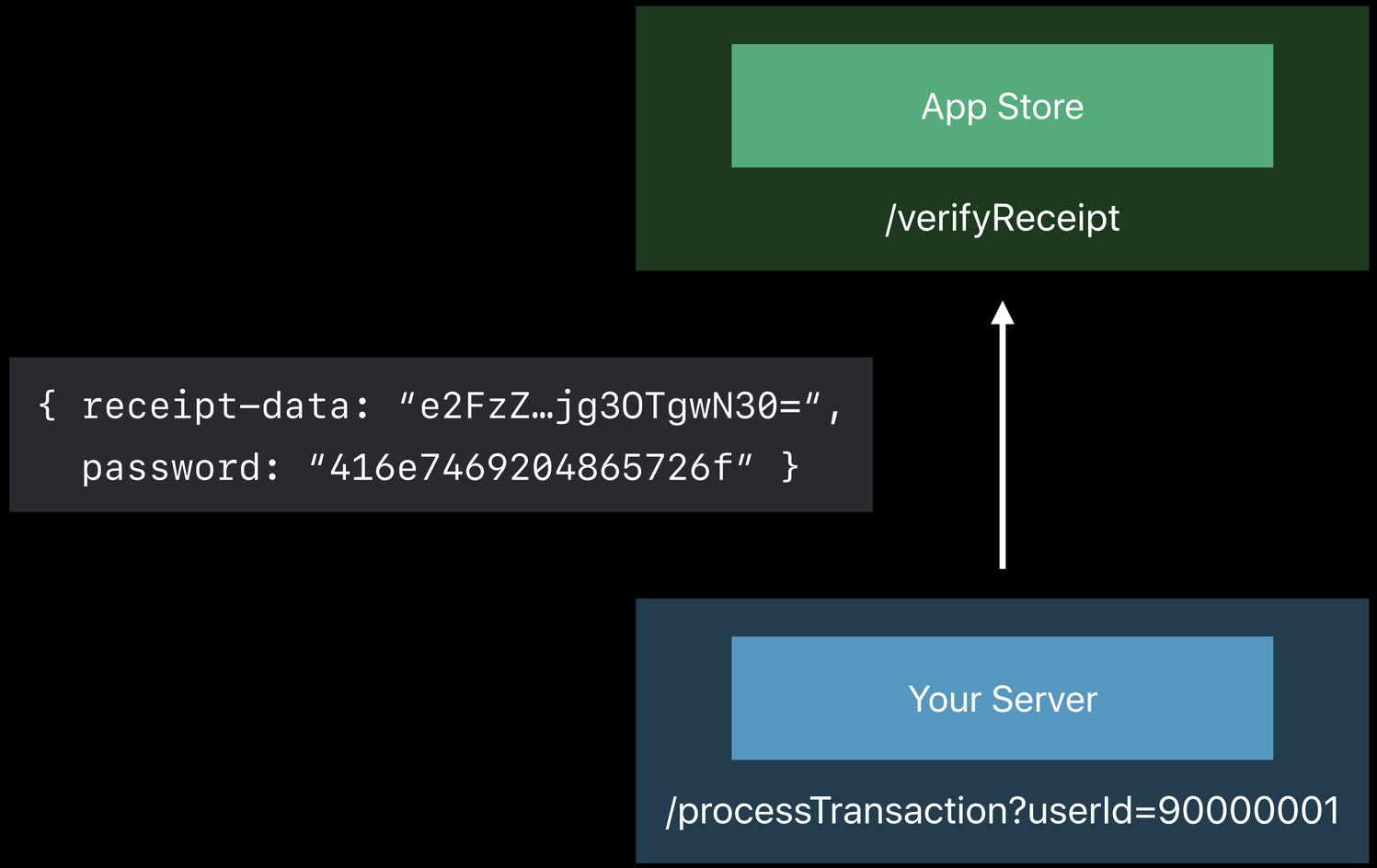
Verifying a Transaction

Server-to-server receipt validation



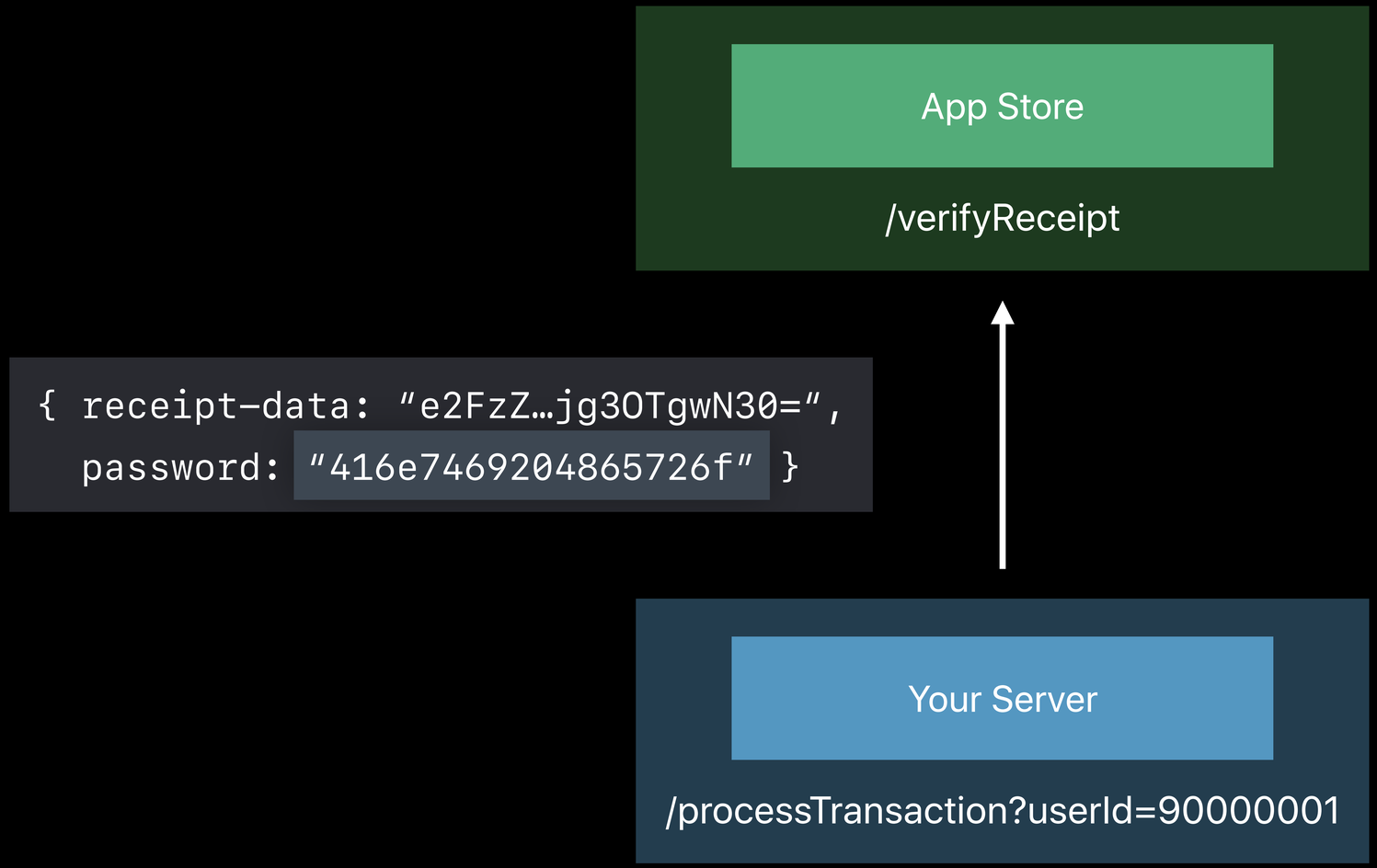
Verifying a Transaction

Server-to-server receipt validation

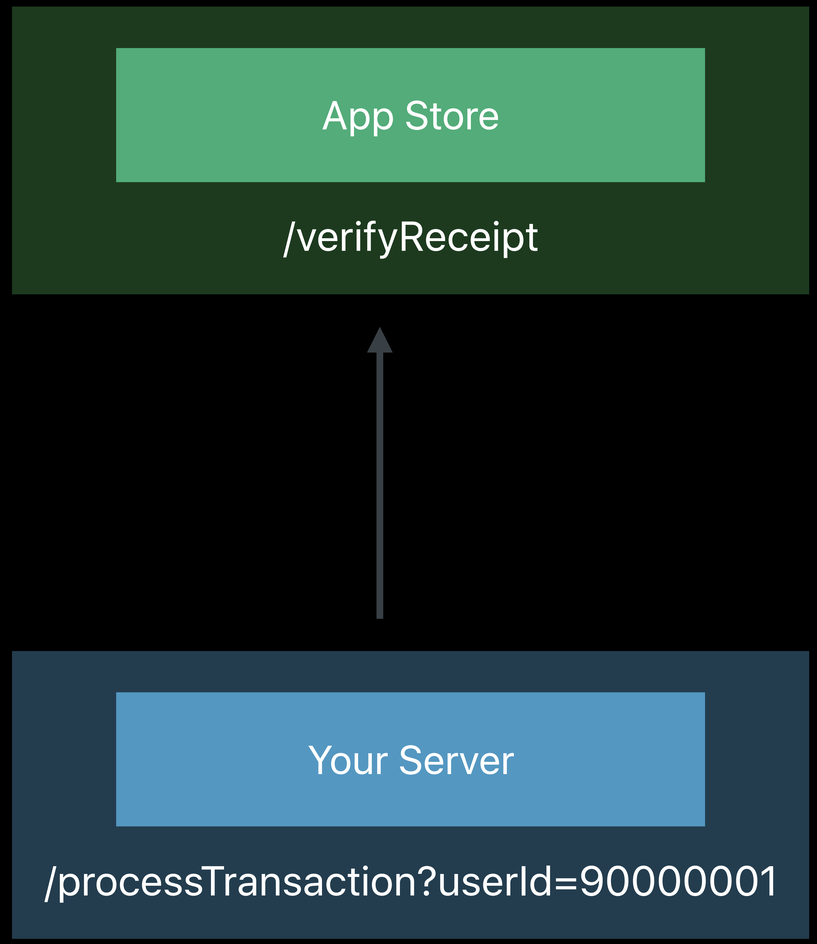


Verifying a Transaction

Server-to-server receipt validation

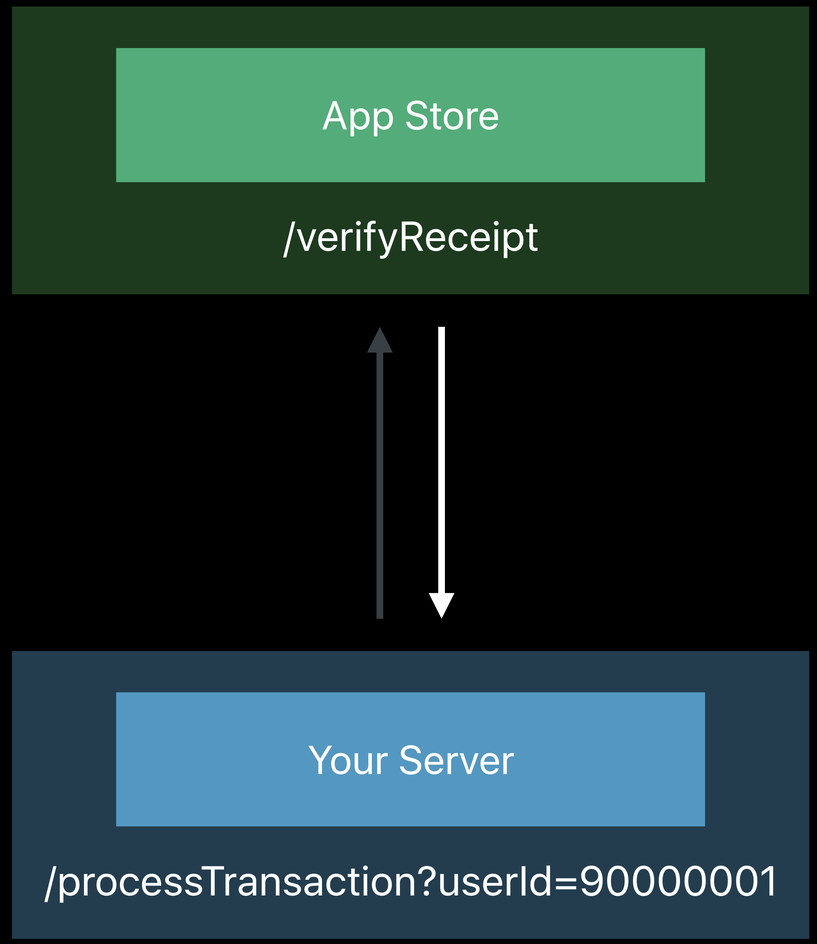


Verifying a Transaction



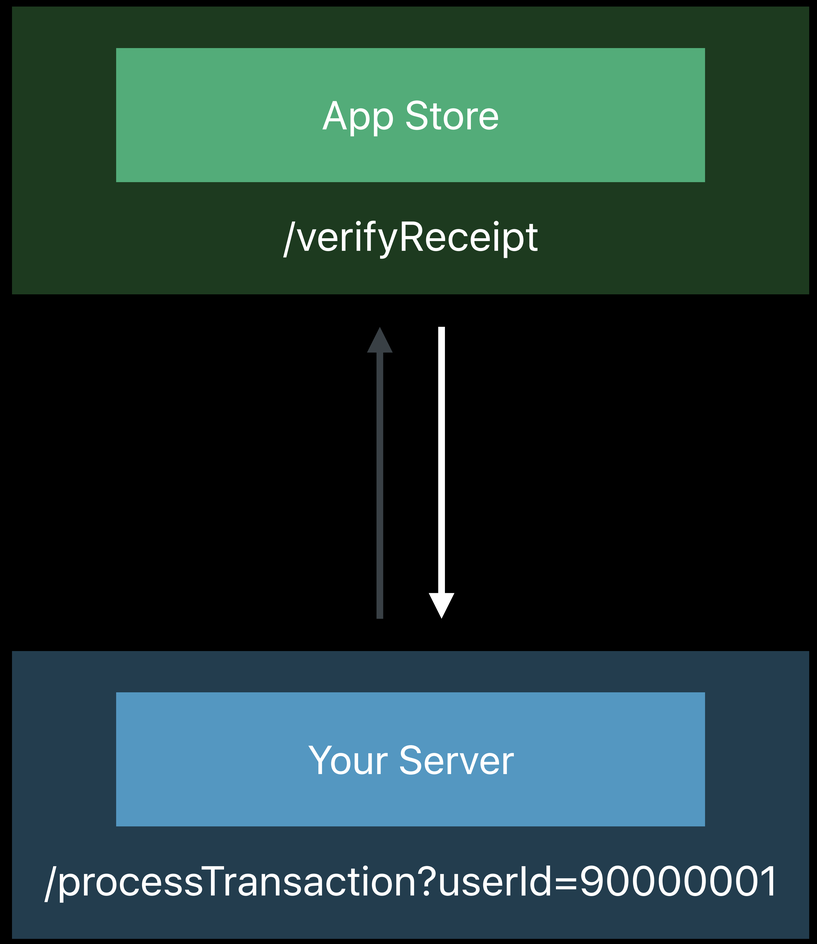
Verifying a Transaction

```
{ status: 0,
  receipt: {
    bundle_id: “com.your.app“,
    in_app: [{
        transaction_id: “1234567890“,
        product_id: “com.your.product.id“,
        original_transaction_id: “1133557799“,
        expires_date: “2018-07-08…“,
        …
    }]
  }
}
```



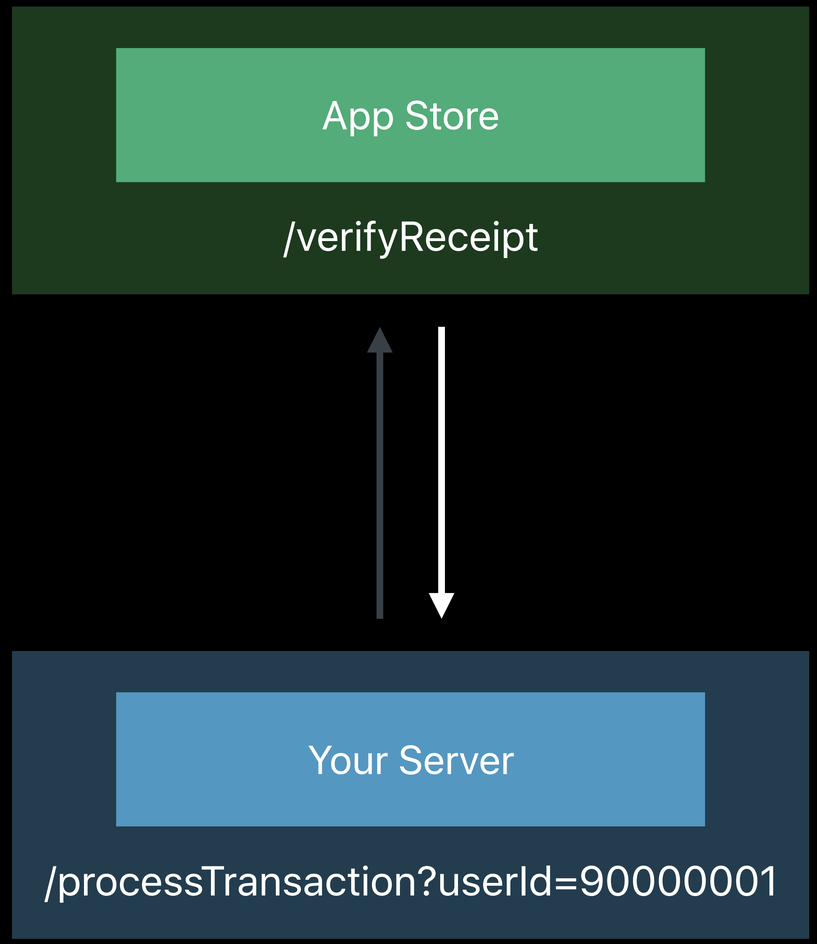
Verifying a Transaction

```
{ status: 0,
  receipt: {
    bundle_id: “com.your.app“,
    in_app: [{
        transaction_id: “1234567890“,
        product_id: “com.your.product.id“,
        original_transaction_id: “1133557799“,
        expires_date: “2018-07-08…“,
        …
    }]
  }
}
```



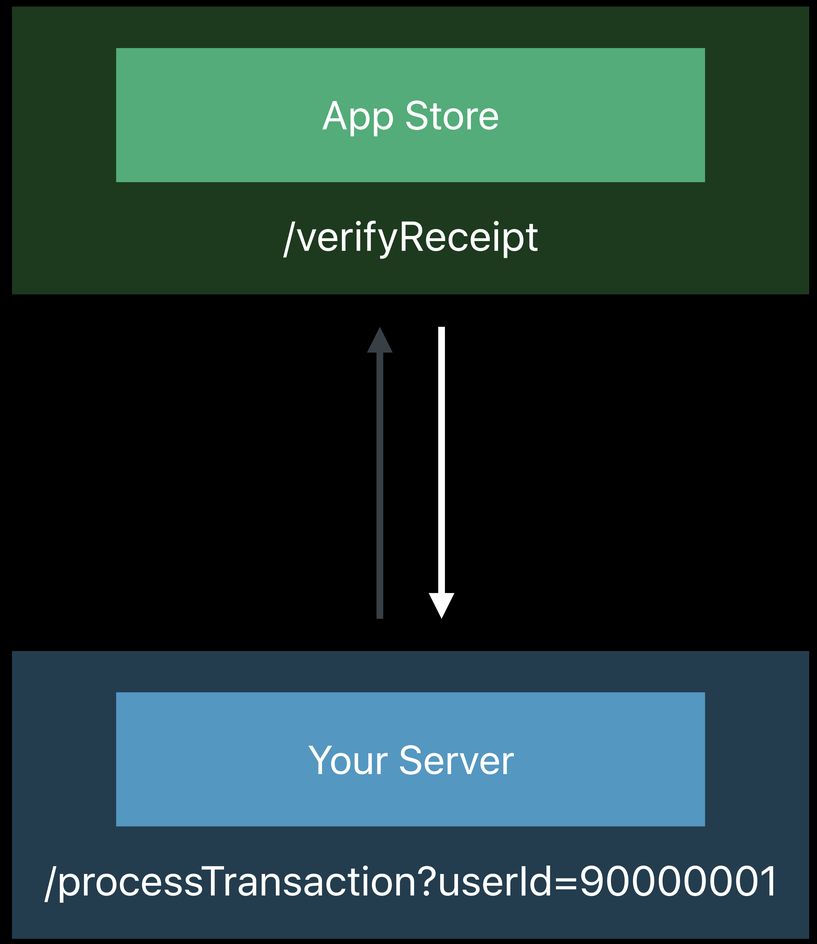
Verifying a Transaction

```
{ status: 0,
  receipt: {
    bundle_id: “com.your.app“,
    in_app: [{
        transaction_id: “1234567890“,
        product_id: “com.your.product.id“,
        original_transaction_id: “1133557799“,
        expires_date: “2018-07-08…“,
        …
    }]
  }
}
```



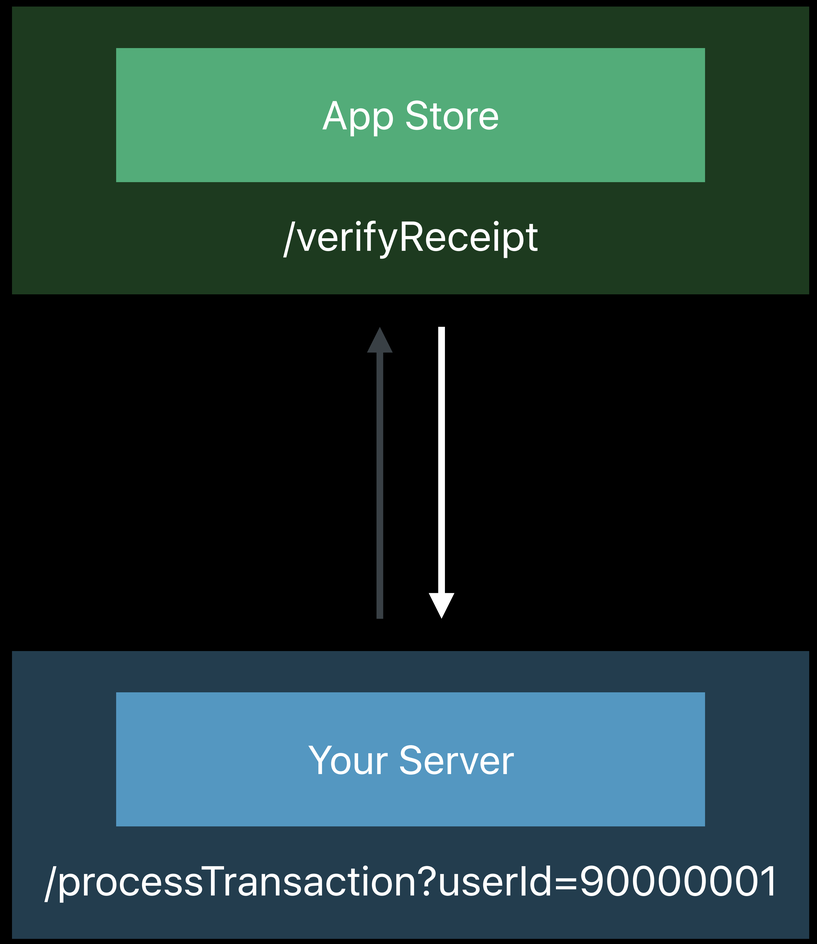
Verifying a Transaction

```
{ status: 0,
  receipt: {
    bundle_id: “com.your.app“,
    in_app: [{
        transaction_id: “1234567890“,
        product_id: “com.your.product.id“,
        original_transaction_id: “1133557799“,
        expires_date: “2018-07-08…“,
        …
    }]
  }
}
```



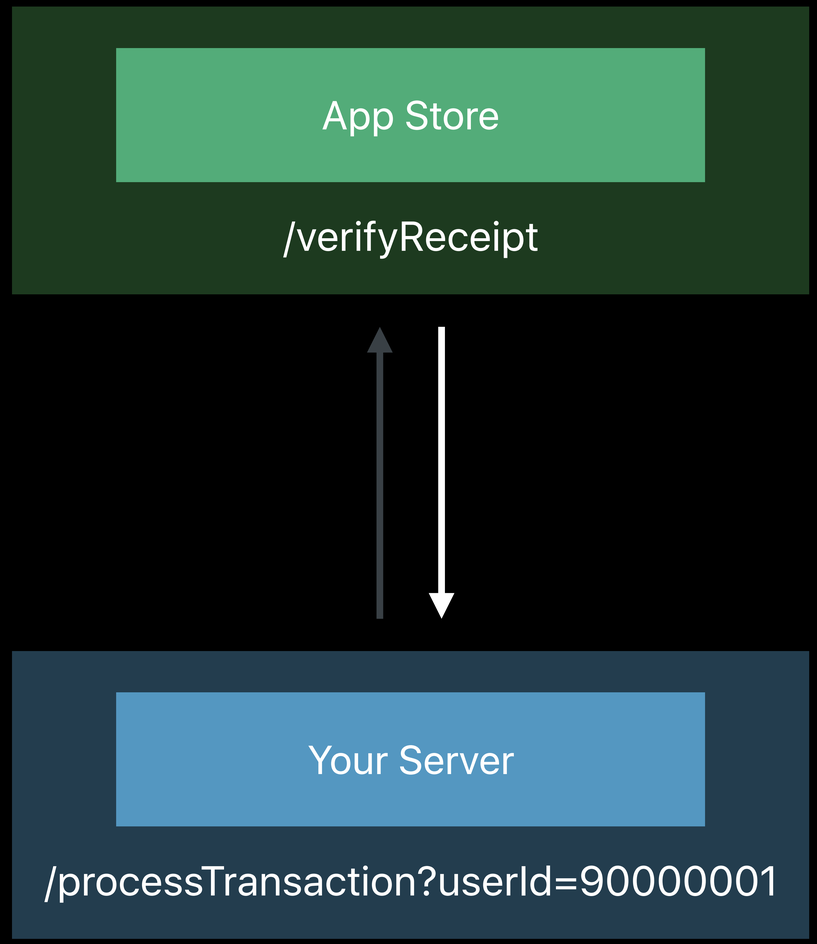
Verifying a Transaction

```
{ status: 0,
  receipt: {
    bundle_id: “com.your.app“,
    in_app: [{
        transaction_id: “1234567890“,
        product_id: “com.your.product.id“,
        original_transaction_id: “1133557799“,
        expires_date: “2018-07-08…“,
        …
    }]
  }
}
```



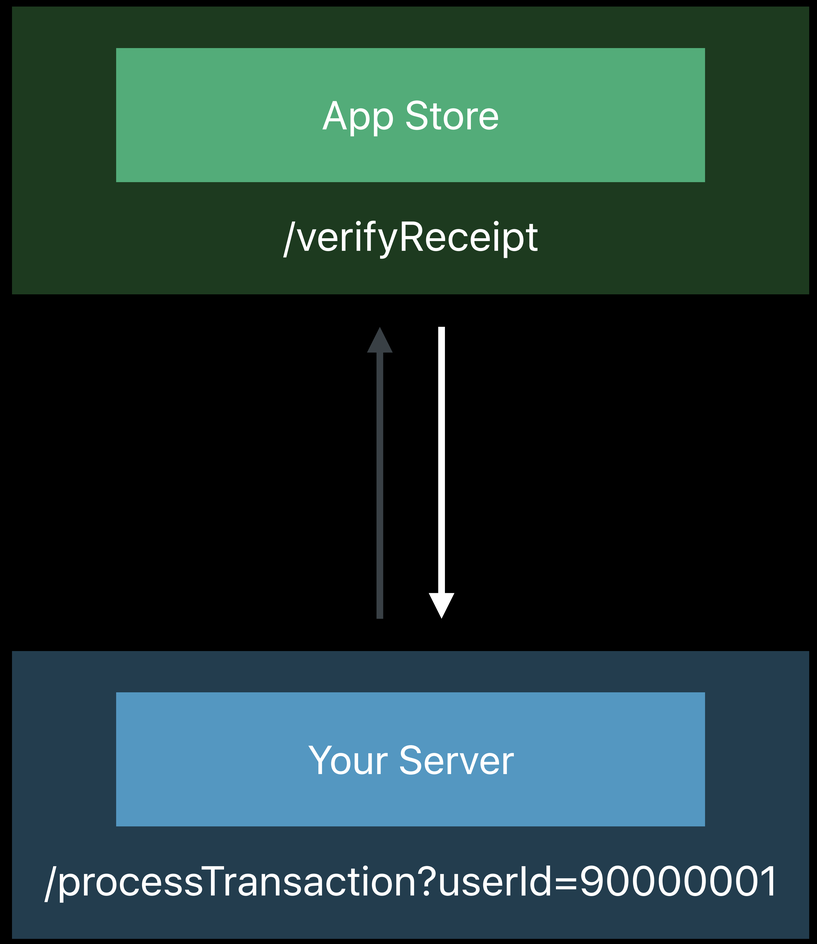
Verifying a Transaction

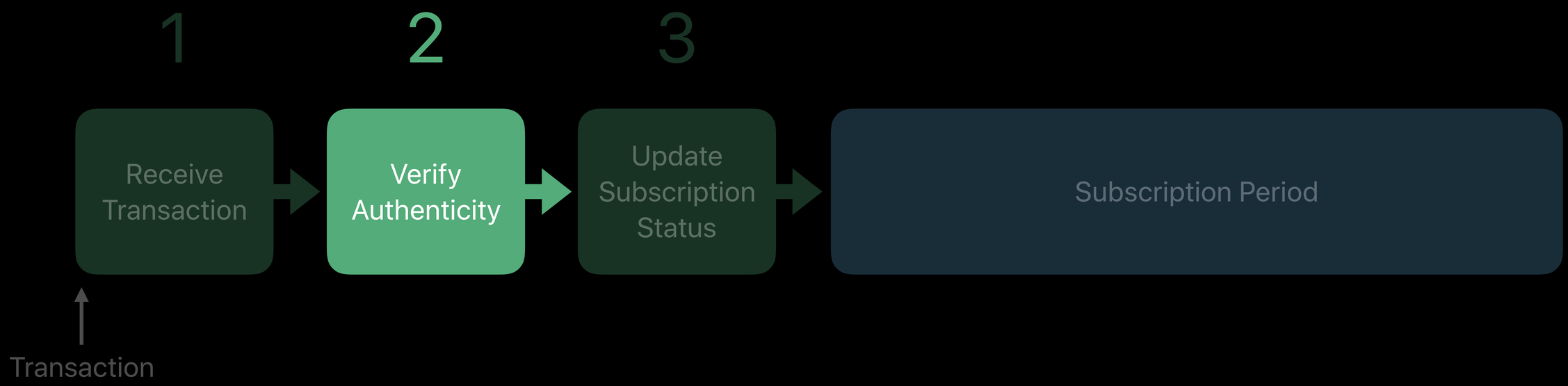
```
{ status: 0,
  receipt: {
    bundle_id: “com.your.app“,
    in_app: [{
        transaction_id: “1234567890“,
        product_id: “com.your.product.id“,
        original_transaction_id: “1133557799“,
        expires_date: “2018-07-08…“,
        …
    }]
  }
}
```

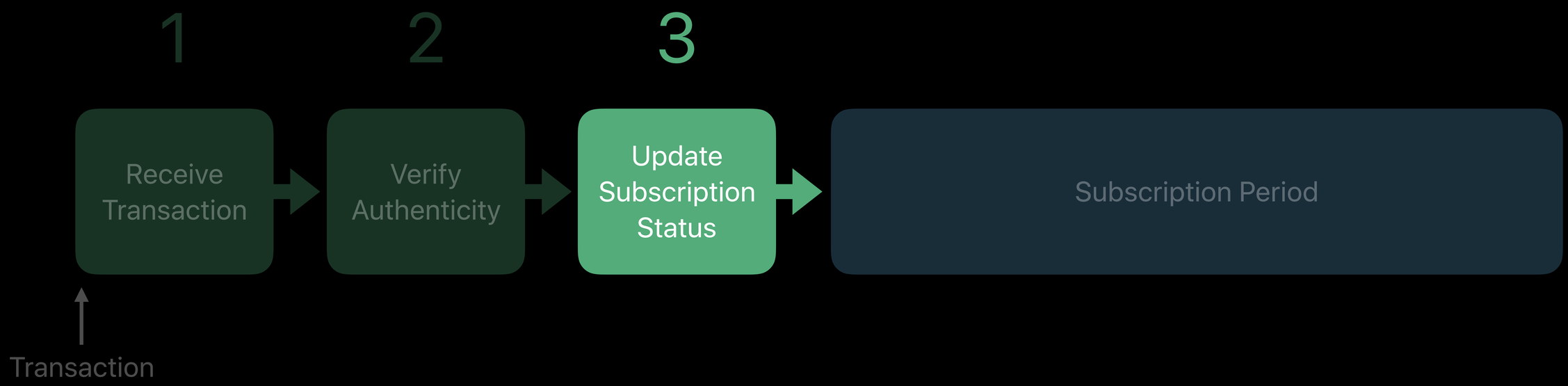


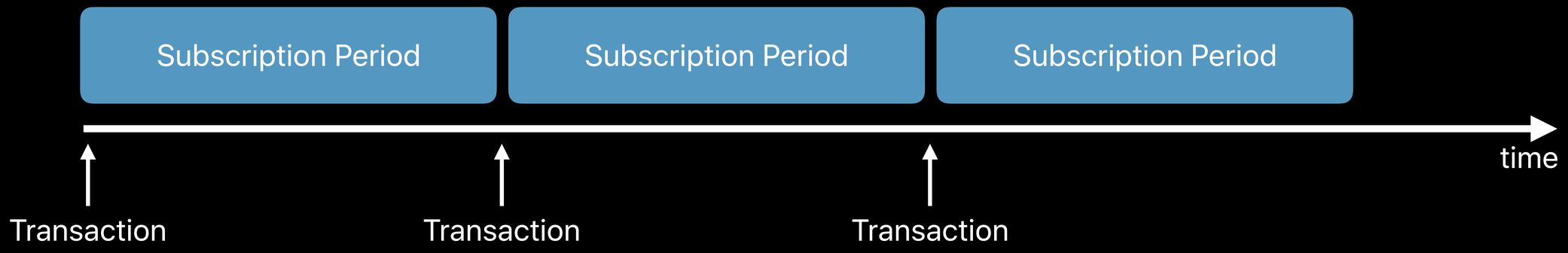
Verifying a Transaction

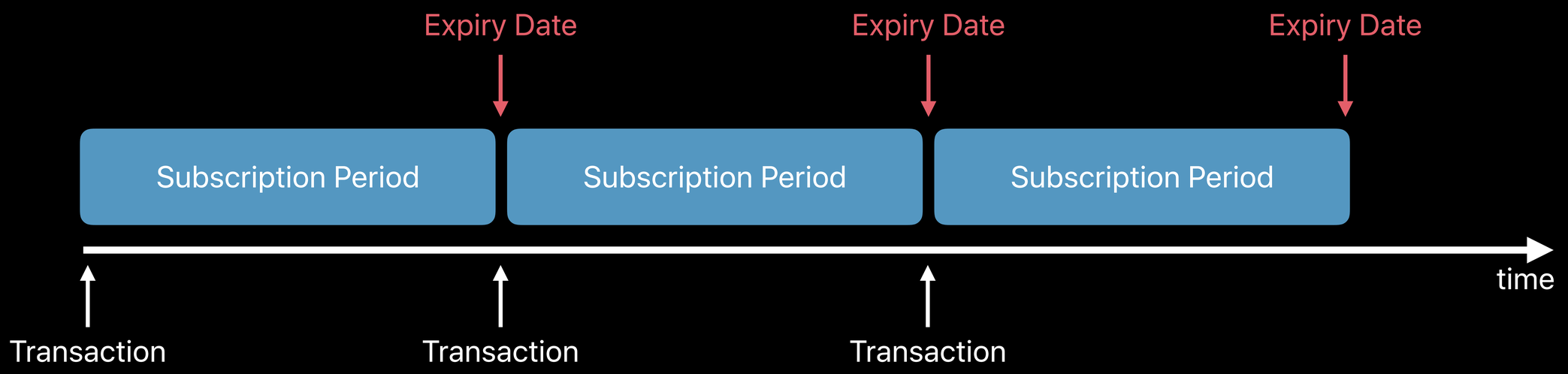
```
{ status: 0,
  receipt: {
    bundle_id: “com.your.app“,
    in_app: [{
        transaction_id: “1234567890“,
        product_id: “com.your.product.id“,
        original_transaction_id: “1133557799“,
        expires_date: “2018-07-08…“,
        …
    }]
  }
}
```




1
2
3
Receive Transaction
Verify Authenticity
Update Subscription Status
Subscription Period
Transaction

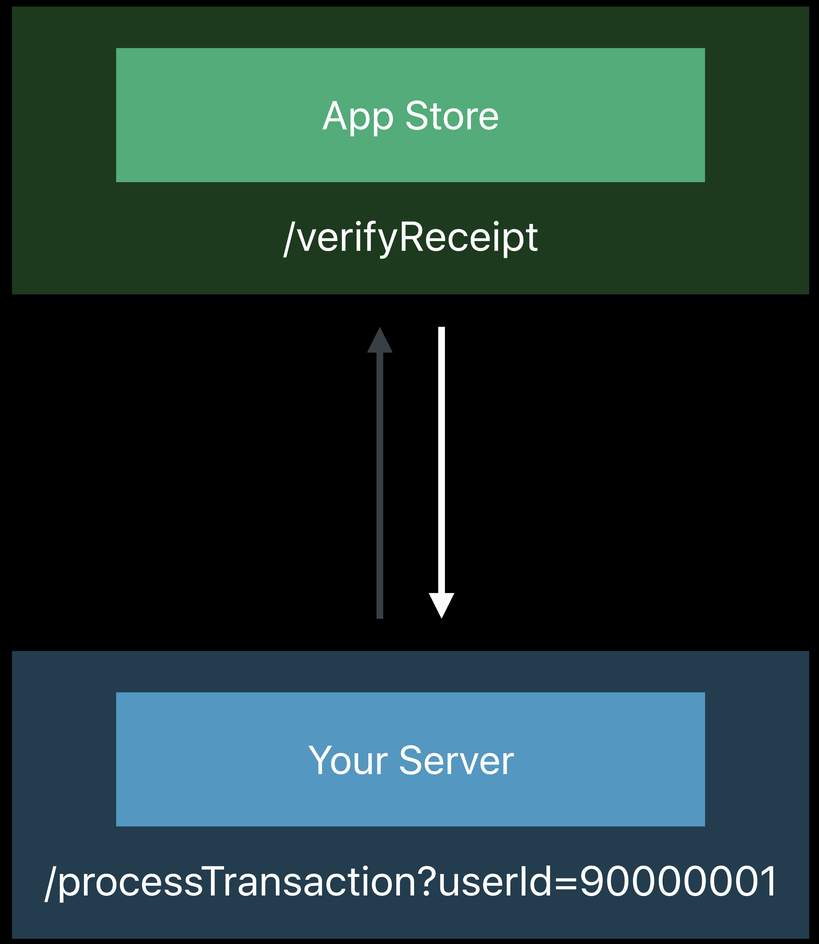

1
2
3
Receive Transaction
Verify Authenticity
Update Subscription Status
Subscription Period
Transaction


Subscription Period
Subscription Period
Subscription Period
time
Transaction
Transaction
Transaction


Expiry Date
Expiry Date
Expiry Date
Subscription Period
Subscription Period
Subscription Period
time
Transaction
Transaction
Transaction

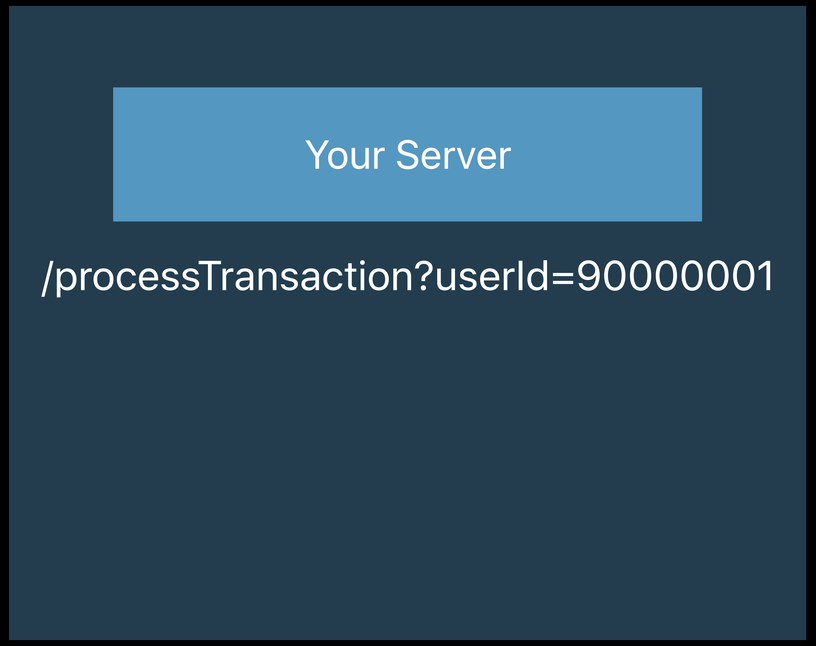
Update Subscription State

```
{ receipt: {
    bundle_id: “com.yourcompany.app”,
    in_app: [{
        transaction_id: “1234567890”,
        product_id: “com.your.product.id”,
        original_transaction_id: “1133557799”,
        expires_date: “2018-07-08…”,
        …
    }]
  }
}
```



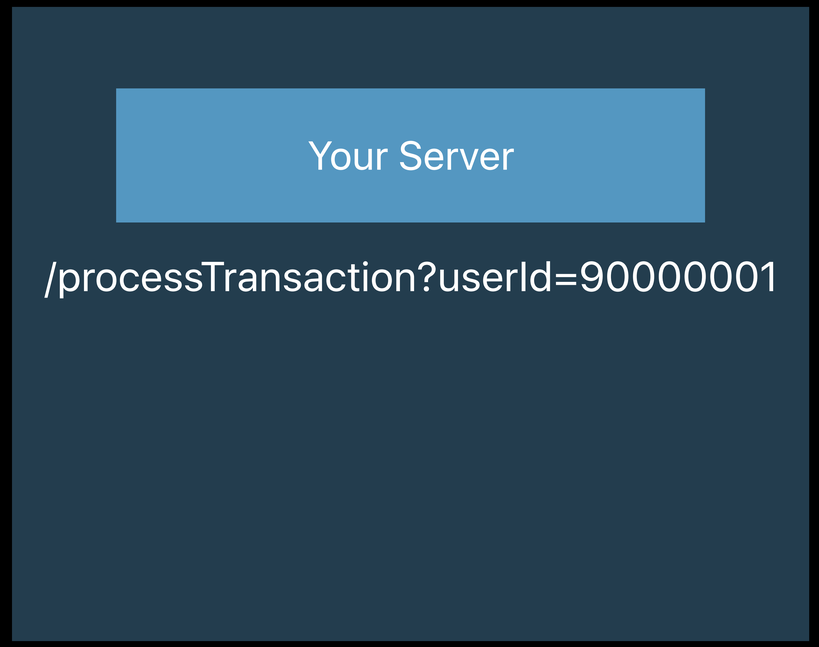
Update Subscription State

```
{ receipt: {
    bundle_id: “com.yourcompany.app”,
    in_app: [{
        transaction_id: “1234567890”,
        product_id: “com.your.product.id”,
        original_transaction_id: “1133557799”,
        expires_date: “2018-07-08…”,
        …
    }]
  }
}
```



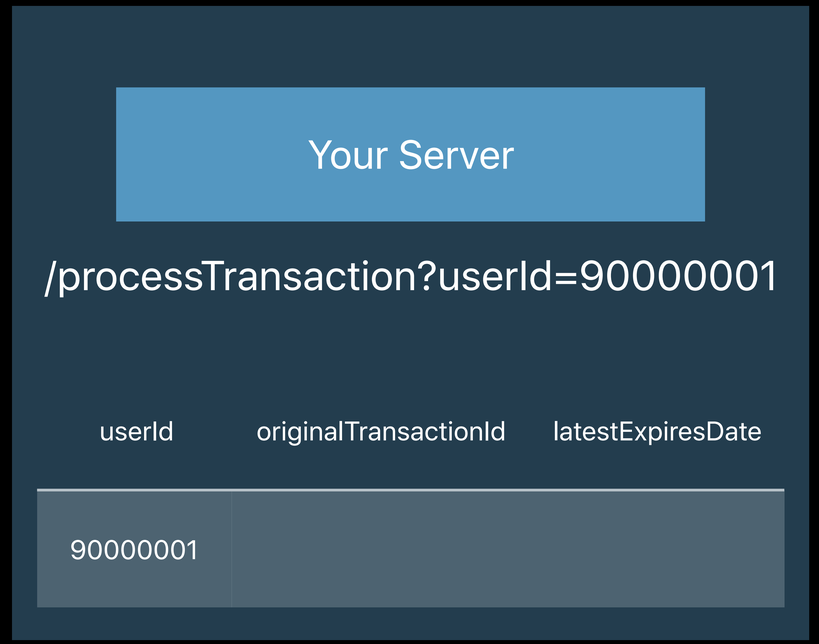
Update Subscription State

```
{ receipt: {
    bundle_id: “com.yourcompany.app”,
    in_app: [{
        transaction_id: “1234567890”,
        product_id: “com.your.product.id”,
        original_transaction_id: “1133557799”,
        expires_date: “2018-07-08…”,
        …
    }]
  }
}
```



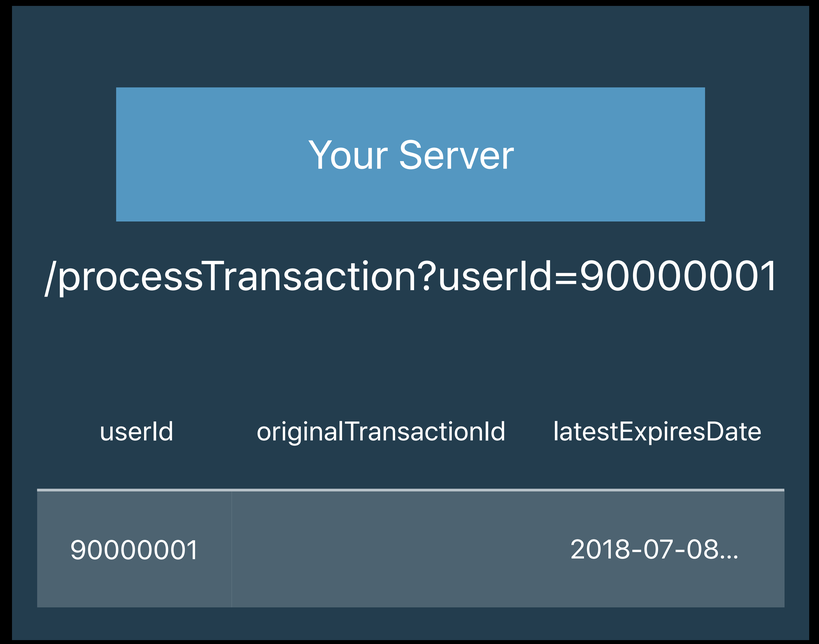
Update Subscription State

```
{ receipt: {
    bundle_id: “com.yourcompany.app”,
    in_app: [{
        transaction_id: “1234567890”,
        product_id: “com.your.product.id”,
        original_transaction_id: “1133557799”,
        expires_date: “2018-07-08…”,
        …
    }]
  }
}
```



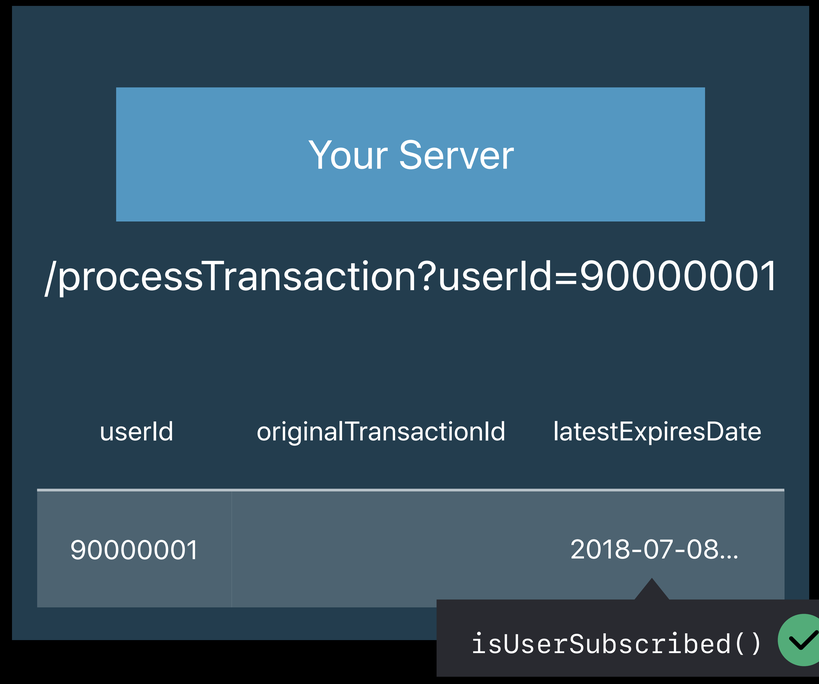
Update Subscription State

```
{ receipt: {
    bundle_id: “com.yourcompany.app”,
    in_app: [{
        transaction_id: “1234567890”,
        product_id: “com.your.product.id”,
        original_transaction_id: “1133557799”,
        expires_date: “2018-07-08…”,
        …
    }]
  }
}
```



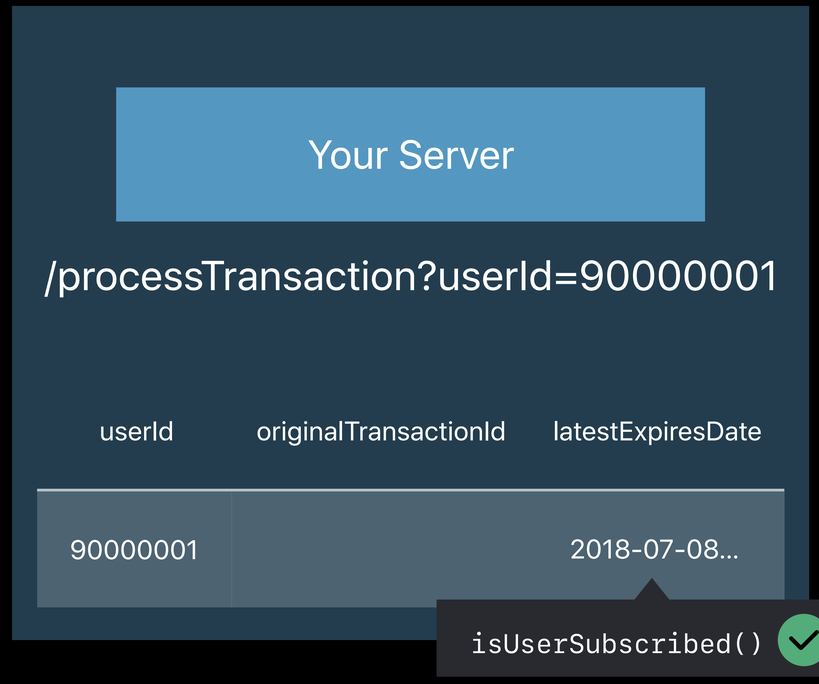
Update Subscription State

```
{ receipt: {
    bundle_id: “com.yourcompany.app”,
    in_app: [{
        transaction_id: “1234567890”,
        product_id: “com.your.product.id”,
        original_transaction_id: “1133557799”,
        expires_date: “2018-07-08…”,
        …
    }]
  }
}
```



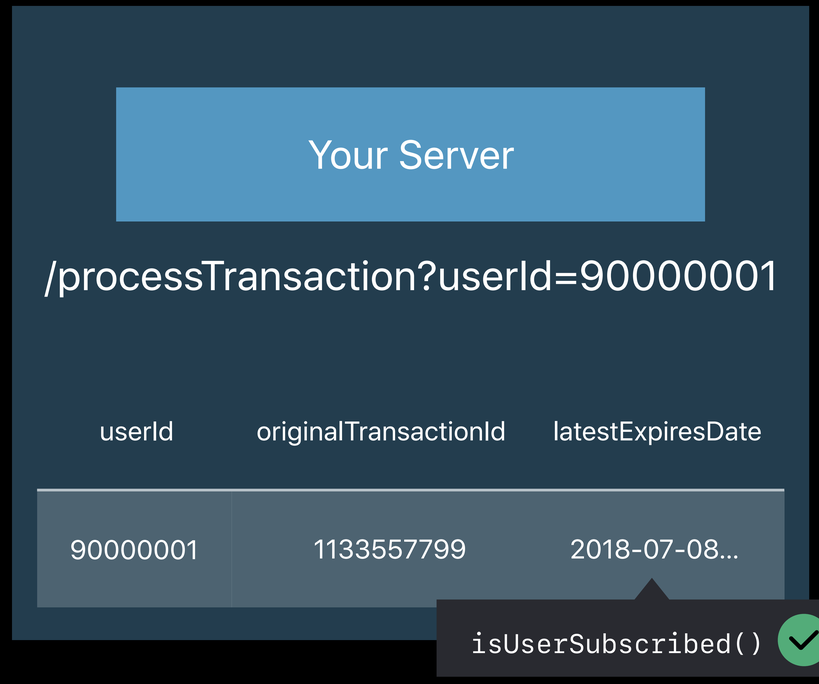
Update Subscription State

```
{ receipt: {
    bundle_id: “com.yourcompany.app”,
    in_app: [{
        transaction_id: “1234567890”,
        product_id: “com.your.product.id”,
        original_transaction_id: “1133557799”,
        expires_date: “2018-07-08…”,
        …
    }]
  }
}
```



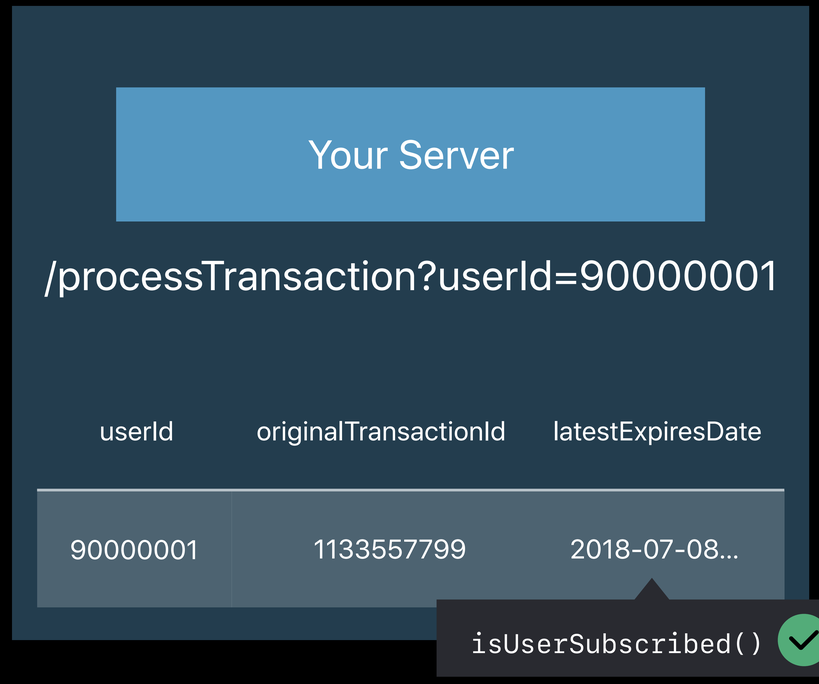
Update Subscription State

```
{ receipt: {
    bundle_id: “com.yourcompany.app”,
    in_app: [{
        transaction_id: “1234567890”,
        product_id: “com.your.product.id”,
        original_transaction_id: “1133557799”,
        expires_date: “2018-07-08…”,
        …
    }]
  }
}
```



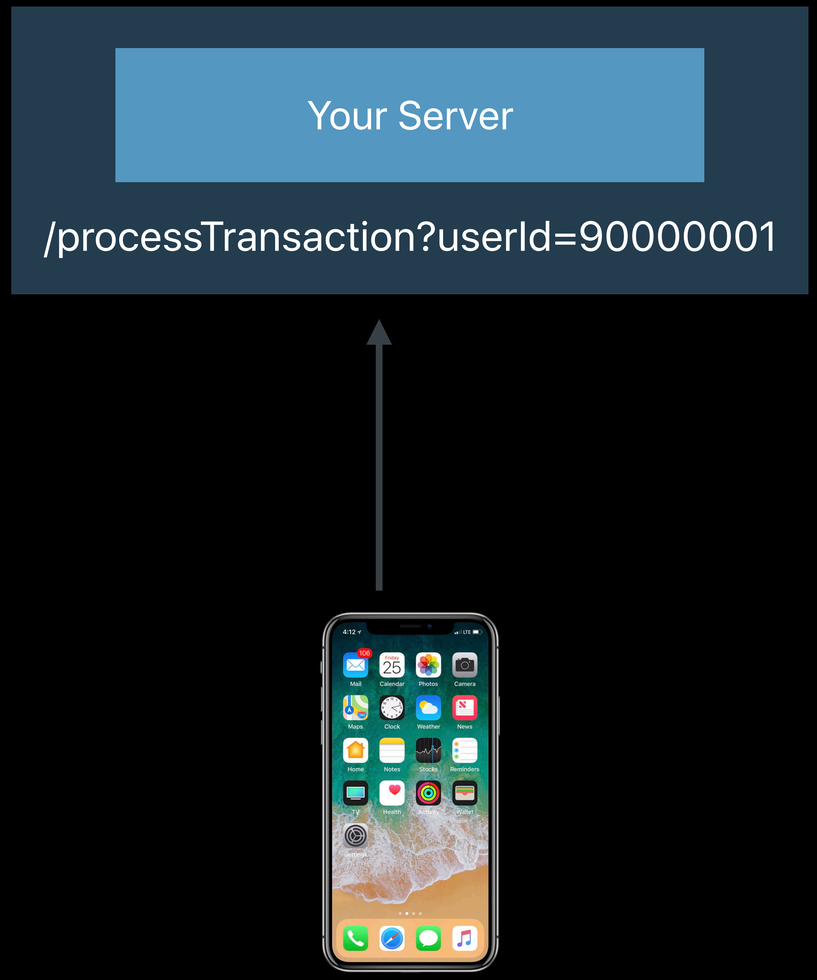
Update Subscription State

```
{ receipt: {
    bundle_id: “com.yourcompany.app”,
    in_app: [{
        transaction_id: “1234567890”,
        product_id: “com.your.product.id”,
        original_transaction_id: “1133557799”,
        expires_date: “2018-07-08…”,
        …
    }]
  }
}
```



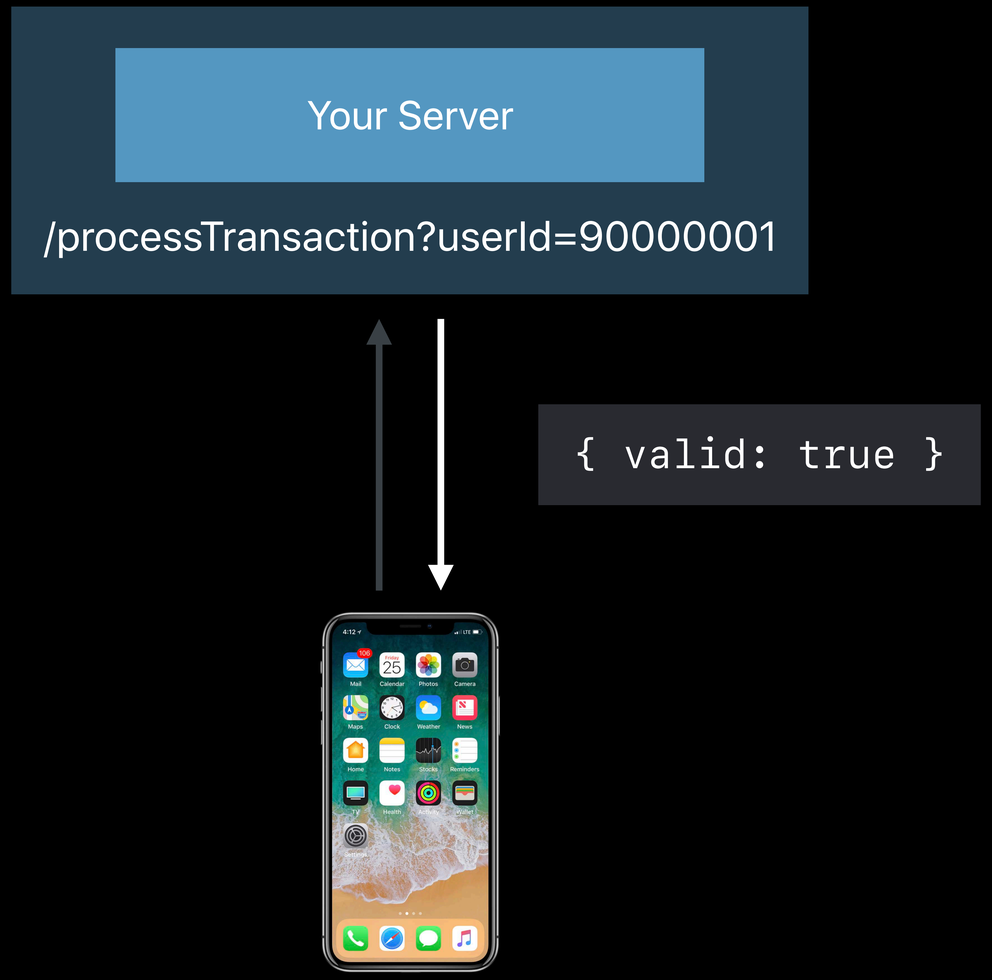
Verifying Transactions

Finishing the transaction



Verifying Transactions

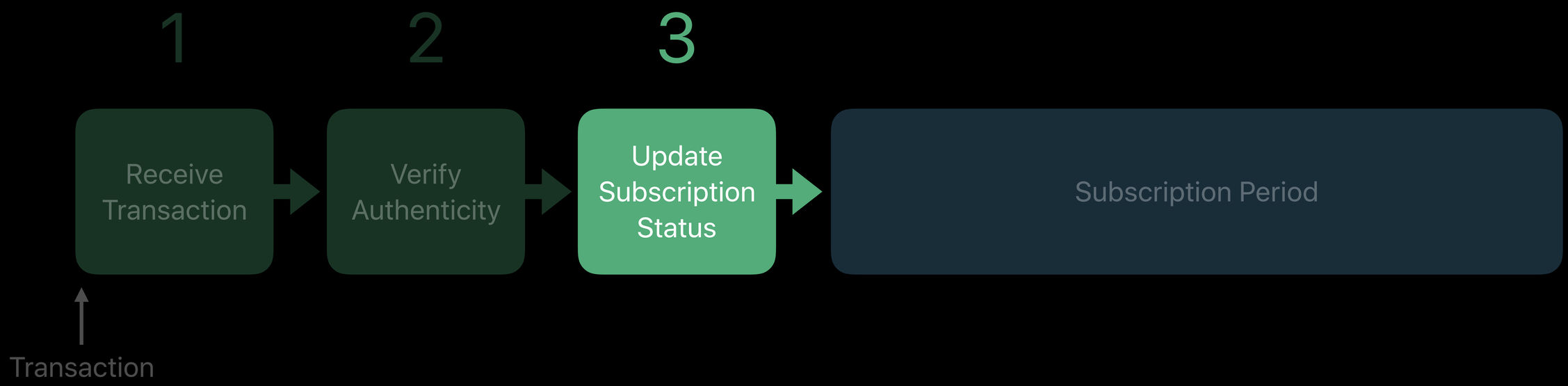
Finishing the transaction

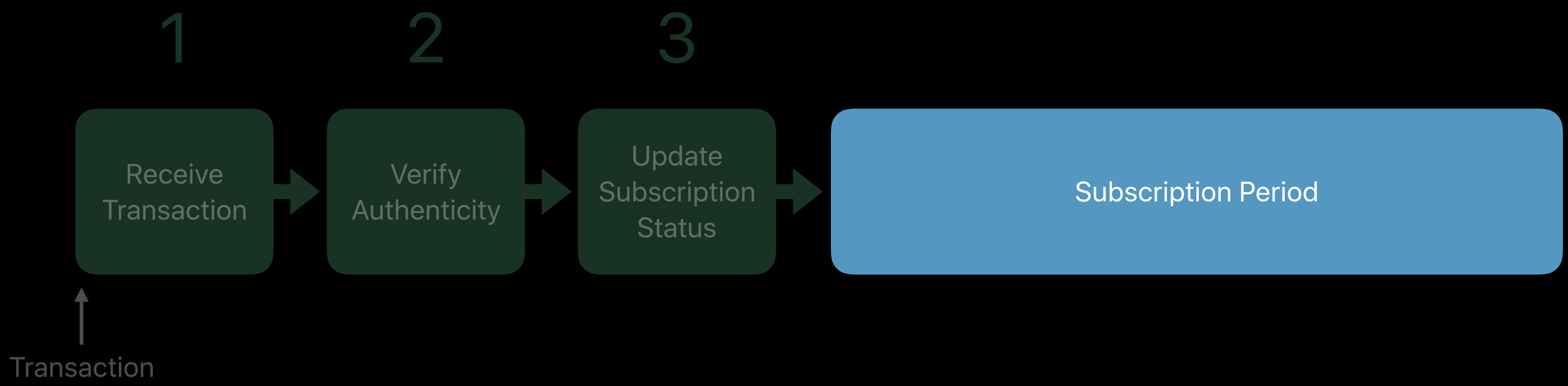


```
switch transaction.transactionState {
    case .purchased:

        if let appStoreReceiptURL = Bundle.main.appStoreReceiptURL,
          FileManager.default.fileExists(atPath: appStoreReceiptURL.path) {

            let rawReceiptData = Data(contentsOf: appStoreReceiptURL)
            let receiptData = rawReceiptData.base64EncodedString(options: …)

            // Send up receiptString to server
            currentUser.processTransaction(receiptData) { isValid in
                if isValid {
                    // Finish the transaction
                    queue.finishTransaction(transaction)
                }
            }
        }
`
```

```
switch transaction.transactionState {
    case .purchased:

        if let appStoreReceiptURL = Bundle.main.appStoreReceiptURL,
          FileManager.default.fileExists(atPath: appStoreReceiptURL.path) {

            let rawReceiptData = Data(contentsOf: appStoreReceiptURL)
            let receiptData = rawReceiptData.base64EncodedString(options: …)

            // Send up receiptString to server
            currentUser.processTransaction(receiptData) { isValid in
                if isValid {
                    // Finish the transaction
                    queue.finishTransaction(transaction)
                }
            }
        }
```

```
switch transaction.transactionState {
    case .purchased:

        if let appStoreReceiptURL = Bundle.main.appStoreReceiptURL,
          FileManager.default.fileExists(atPath: appStoreReceiptURL.path) {

            let rawReceiptData = Data(contentsOf: appStoreReceiptURL)
            let receiptData = rawReceiptData.base64EncodedString(options: …)

            // Send up receiptString to server
            currentUser.processTransaction(receiptData) { isValid in
                if isValid {
                    // Finish the transaction
                    queue.finishTransaction(transaction)
                }
            }
        }
`
```


1
2
3
Receive Transaction
Verify Authenticity
Update Subscription Status
Subscription Period
Transaction

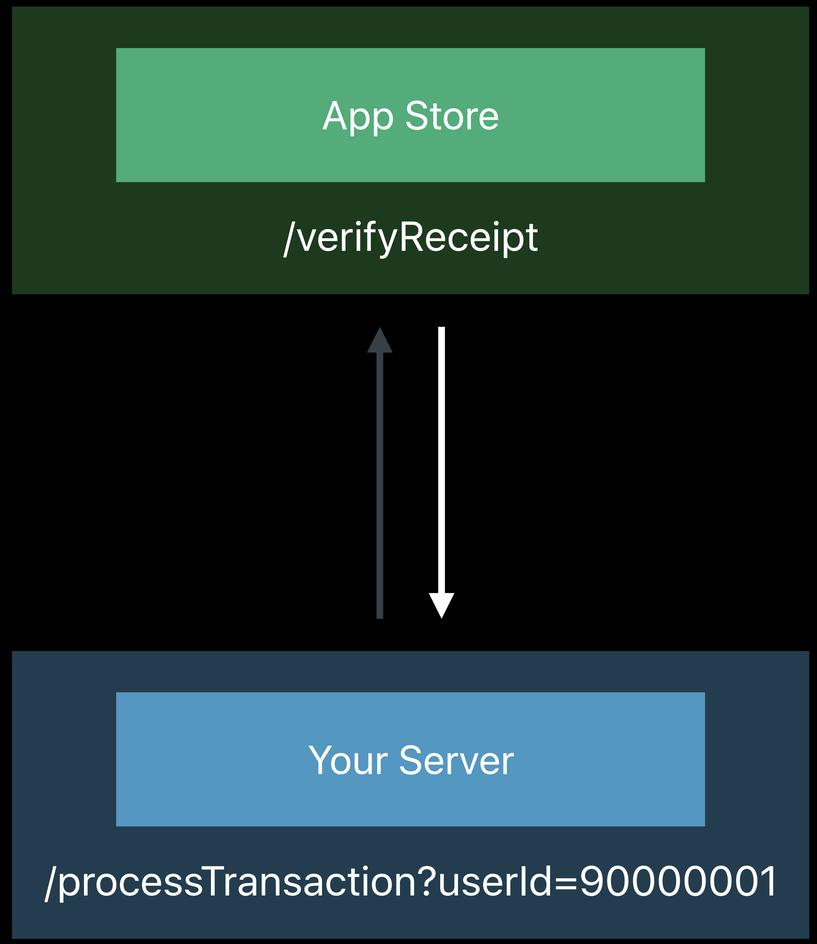

1
2
3
Receive Transaction
Verify Authenticity
Update Subscription Status
Subscription Period
Transaction

userId	originalTransactionId	latestExpiresDate
90000001	1133557799	2018-07-08...
90000002	4820018625	2018-06-02...
90000003	6090522426	2018-06-01...
90000004	3092890314	2018-06-20...
90000005	8426576390	2018-06-21...
90000006	4286343643	2018-06-03...

userId	originalTransactionId	latestExpiresDate
90000001	1133557799	2018-07-08...
90000002	4820018625	2018-06-02...
90000003	6090522426	2018-06-01...
90000004	3092890314	2018-06-20...
90000005	8426576390	2018-06-21...
90000006	4286343643	2018-06-03...

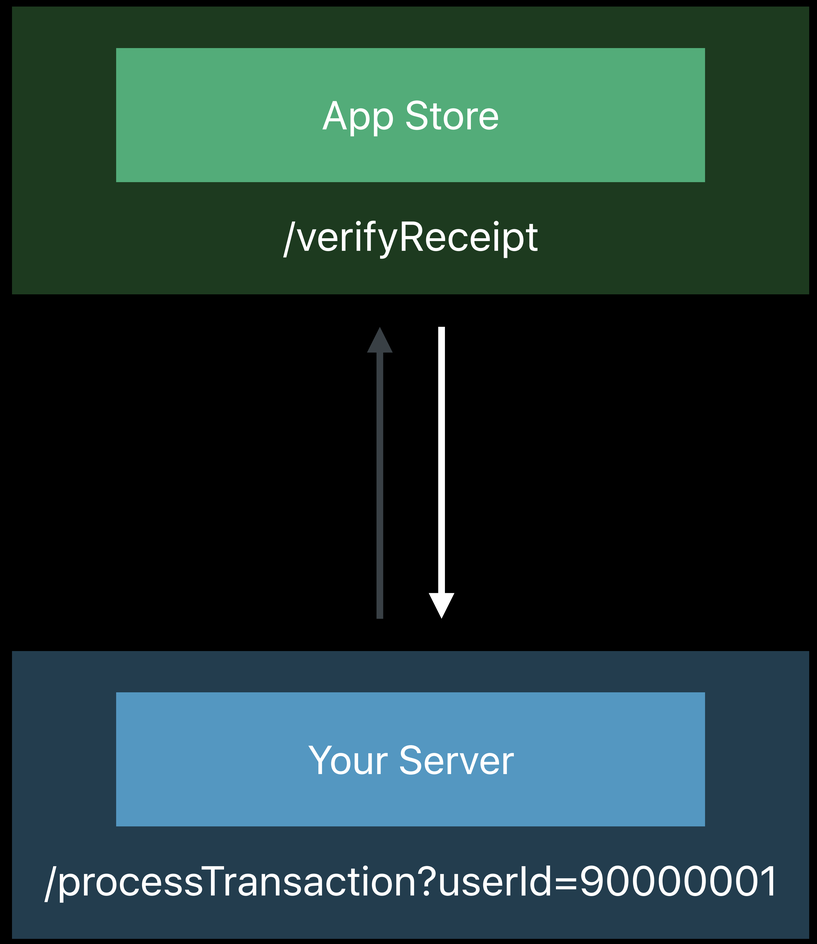
Verifying a Transaction

```
{ status: 0,
  receipt: {
    bundle_id: “com.your.app“,
    in_app: [{
        transaction_id: “1234567890“,
        product_id: “com.your.product.id“,
        original_transaction_id: “1133557799“,
        expires_date: “2018-07-08…“,
        …
    }]
  }
}
```



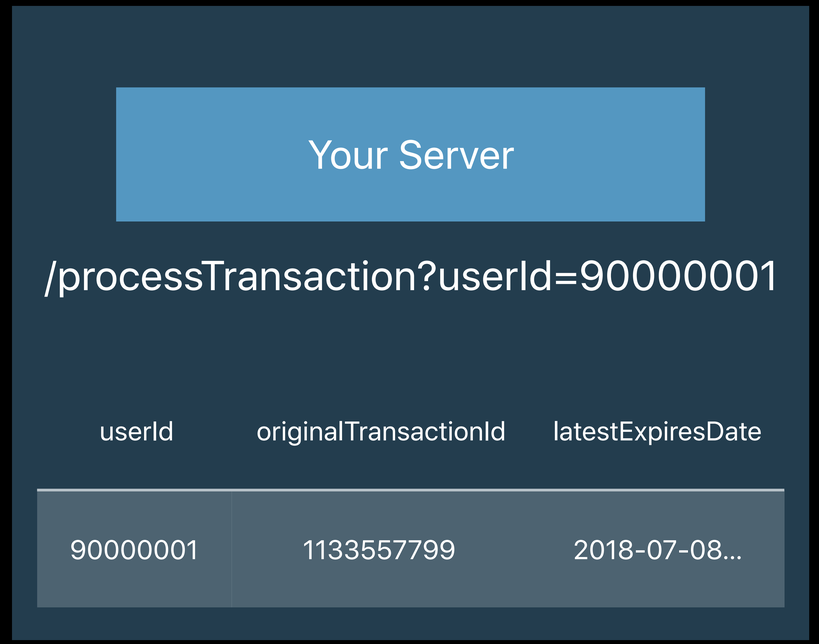
Verifying a Transaction

```
{ status: 0,
  receipt: {
    bundle_id: “com.your.app“,
    in_app: [{
        transaction_id: “1234567890“,
        product_id: “com.your.product.id“,
        original_transaction_id: “1133557799“,
        expires_date: “2018-07-08…“,
        …
    }]
  }
}
```



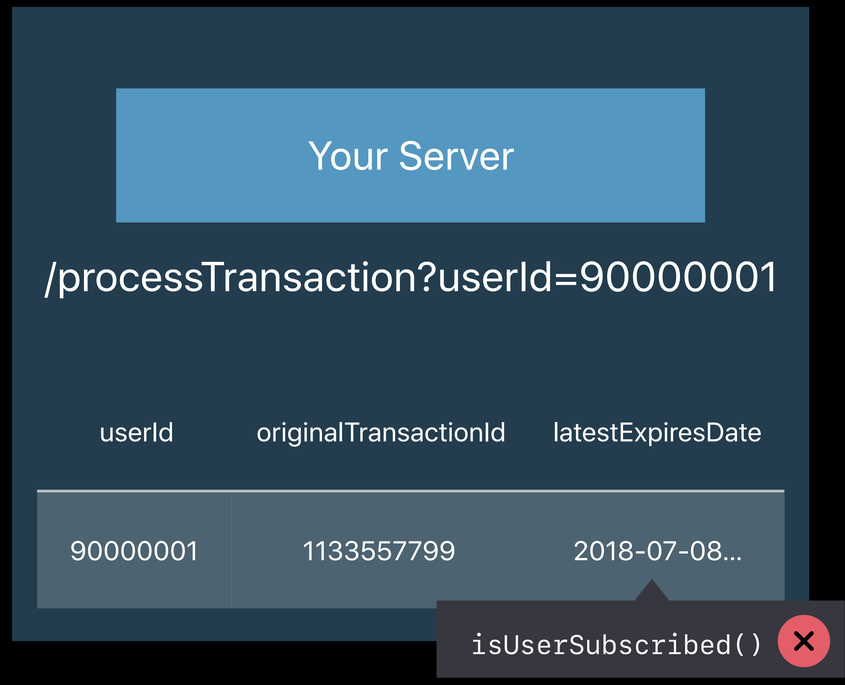
Update Subscription State

```
{ receipt: {
   in_app: [{
     transaction_id: “1234567890”,
     original_transaction_id: “1133557799”,
     expires_date: “2018-07-08…”, …
   }, {
     transaction_id: “2233445566”,
     original_transaction_id: “1133557799”,
     expires_date: “2018-08-08…”, …
   }]
  }
}
```



Update Subscription State

```
{ receipt: {
   in_app: [{
     transaction_id: “1234567890”,
     original_transaction_id: “1133557799”,
     expires_date: “2018-07-08…”, …
   }, {
     transaction_id: “2233445566”,
     original_transaction_id: “1133557799”,
     expires_date: “2018-08-08…”, …
   }]
  }
}
```



Does the User Have an Active Subscription?

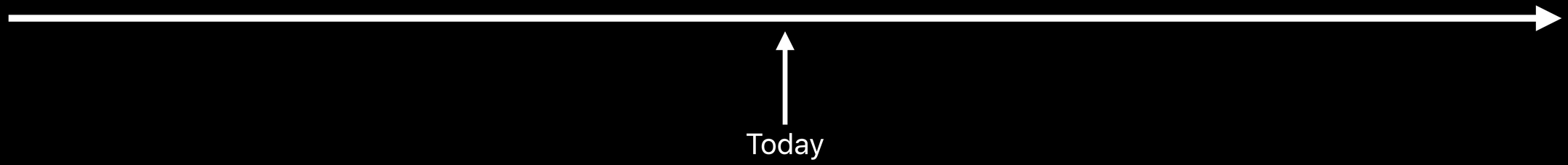
Does the User Have an Active Subscription?

Filter transactions by `original_transaction_id`

Does the User Have an Active Subscription?

Filter transactions by `original_transaction_id`

Find transaction with the latest `expires_date`

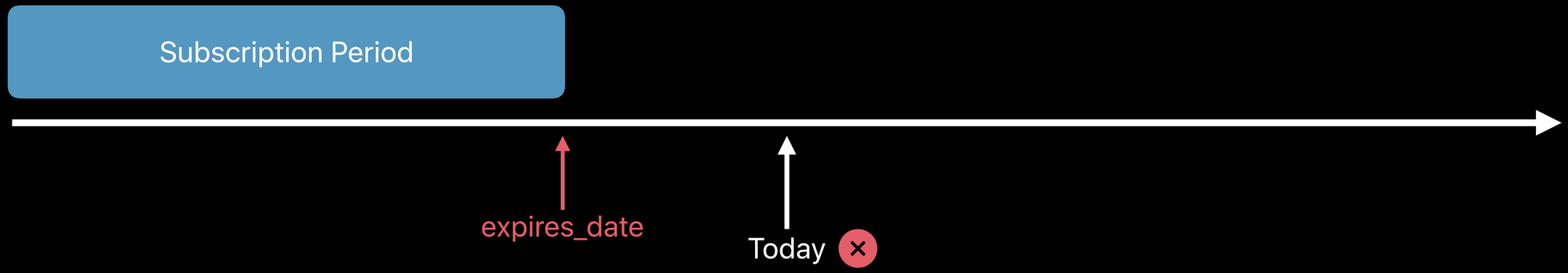


Does the User Have an Active Subscription?

Filter transactions by `original_transaction_id`

Find transaction with the latest `expires_date`

Date in past shows user is not subscribed



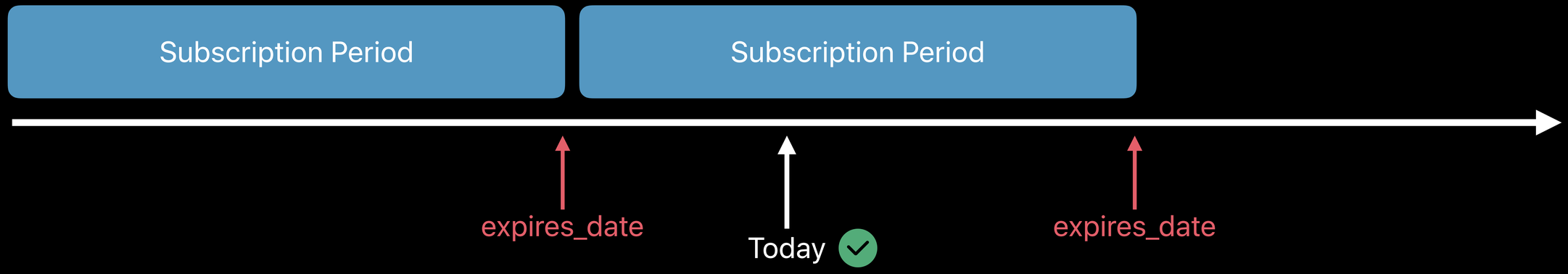
Does the User Have an Active Subscription?

Filter transactions by `original_transaction_id`

Find transaction with the latest `expires_date`

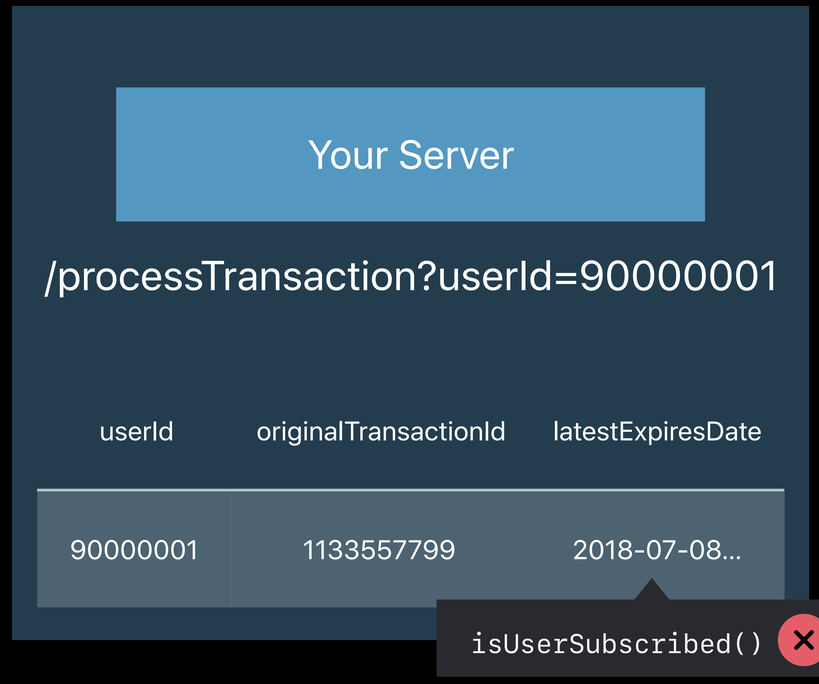
Date in past shows user is not subscribed

Date in future shows user is subscribed



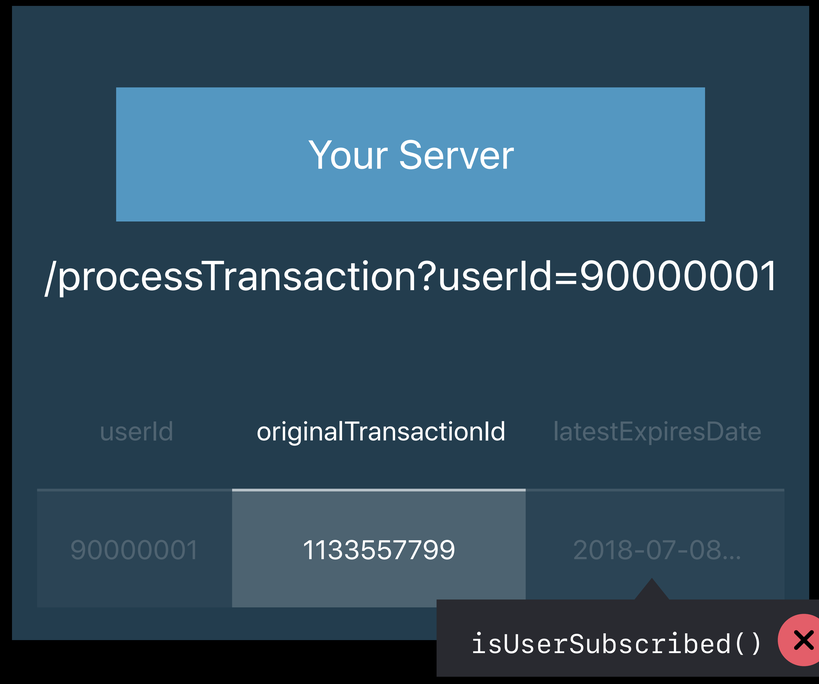
Update Subscription State

```
{ receipt: {
   in_app: [{
     transaction_id: “1234567890”,
     original_transaction_id: “1133557799”,
     expires_date: “2018-07-08…”, …
   }, {
     transaction_id: “2233445566”,
     original_transaction_id: “1133557799”,
     expires_date: “2018-08-08…”, …
   }]
  }
}
```



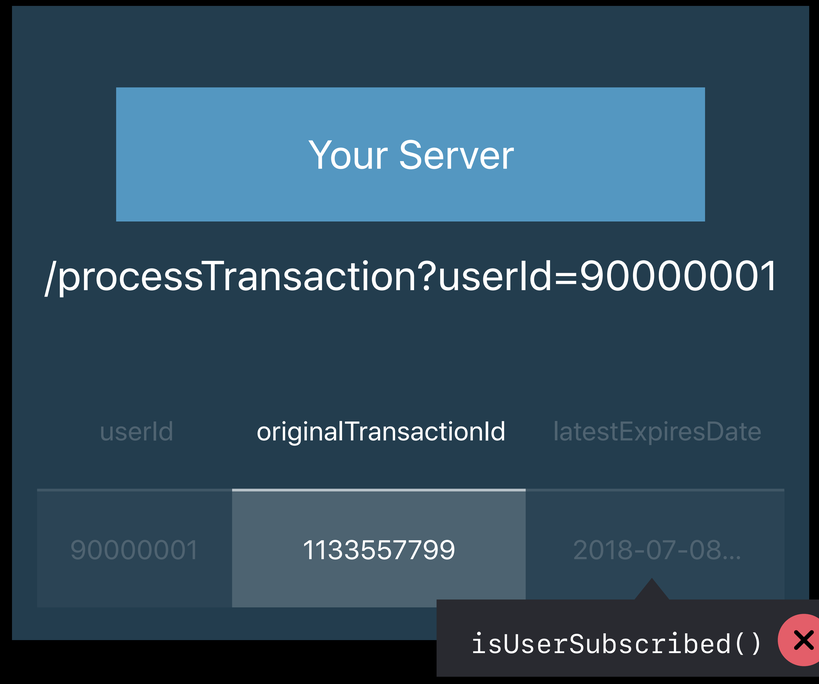
Update Subscription State

```
{ receipt: {
   in_app: [{
     transaction_id: “1234567890”,
     original_transaction_id: “1133557799”,
     expires_date: “2018-07-08…”, …
   }, {
     transaction_id: “2233445566”,
     original_transaction_id: “1133557799”,
     expires_date: “2018-08-08…”, …
   }]
  }
}
```



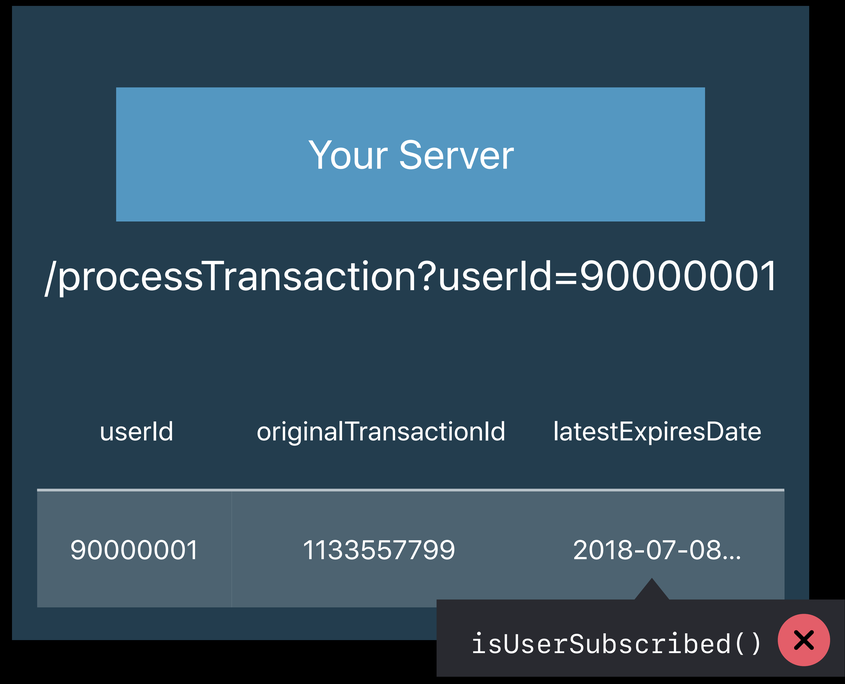
Update Subscription State

```
{ receipt: {
   in_app: [{
     transaction_id: “1234567890”,
     original_transaction_id: “1133557799”,
     expires_date: “2018-07-08…”, …
   }, {
     transaction_id: “2233445566”,
     original_transaction_id: “1133557799”,
     expires_date: “2018-08-08…”, …
   }]
  }
}
```



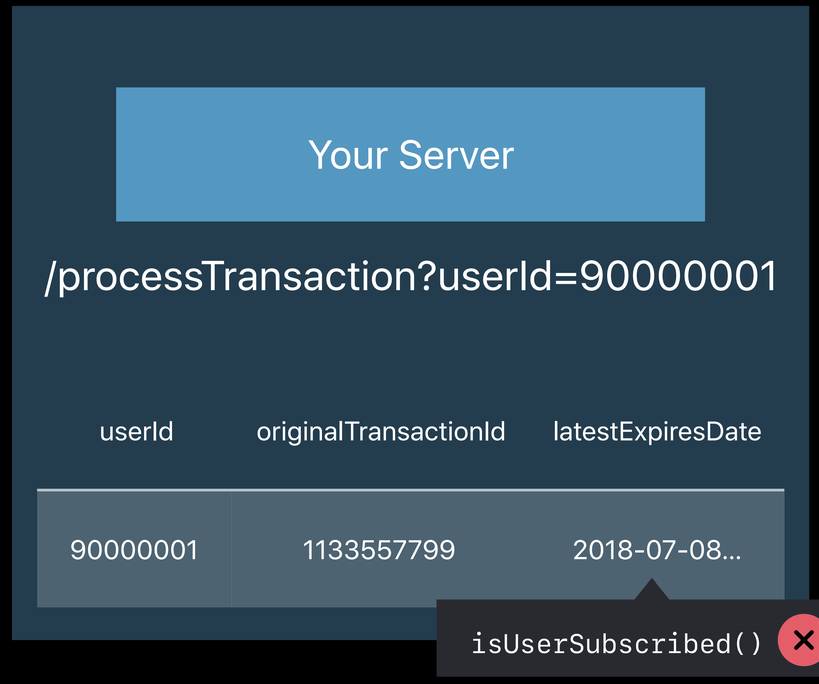
Update Subscription State

```
{ receipt: {
   in_app: [{
     transaction_id: “1234567890”,
     original_transaction_id: “1133557799”,
     expires_date: “2018-07-08…”, …
   }, {
     transaction_id: “2233445566”,
     original_transaction_id: “1133557799”,
     expires_date: “2018-08-08…”, …
   }]
  }
}
```



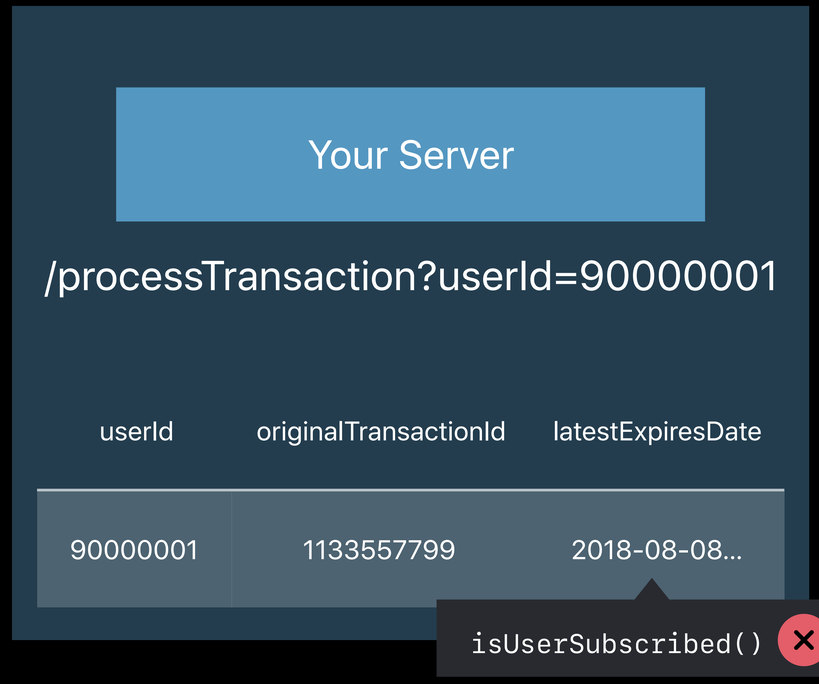
Update Subscription State

```
{ receipt: {
   in_app: [{
     transaction_id: “1234567890”,
     original_transaction_id: “1133557799”,
     expires_date: “2018-07-08…”, …
   }, {
     transaction_id: “2233445566”,
     original_transaction_id: “1133557799”,
     expires_date: “2018-08-08…”, …
   }]
  }
}
```



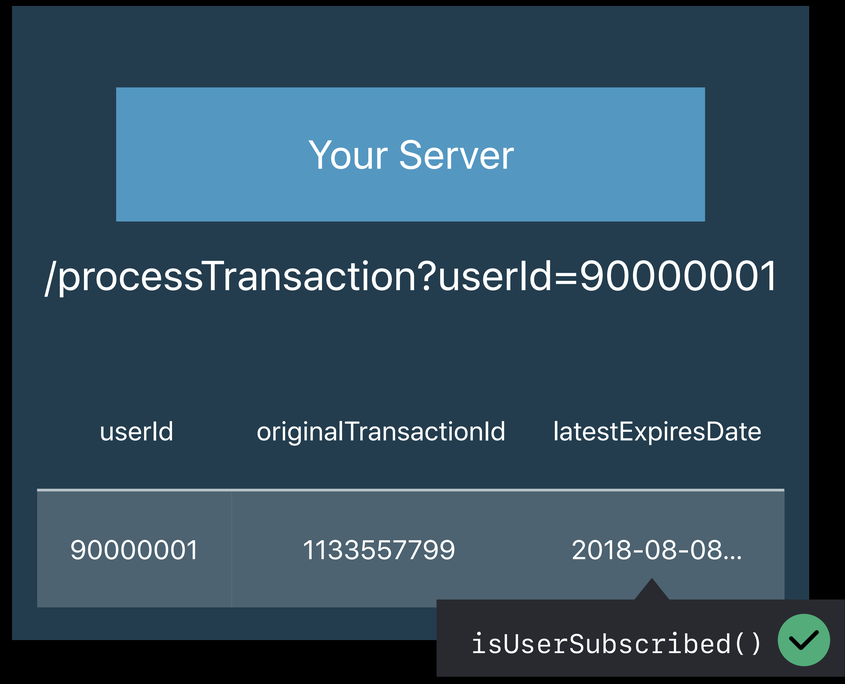
Update Subscription State

```
{ receipt: {
   in_app: [{
     transaction_id: “1234567890”,
     original_transaction_id: “1133557799”,
     expires_date: “2018-07-08…”, …
   }, {
     transaction_id: “2233445566”,
     original_transaction_id: “1133557799”,
     expires_date: “2018-08-08…”, …
   }]
  }
}
```



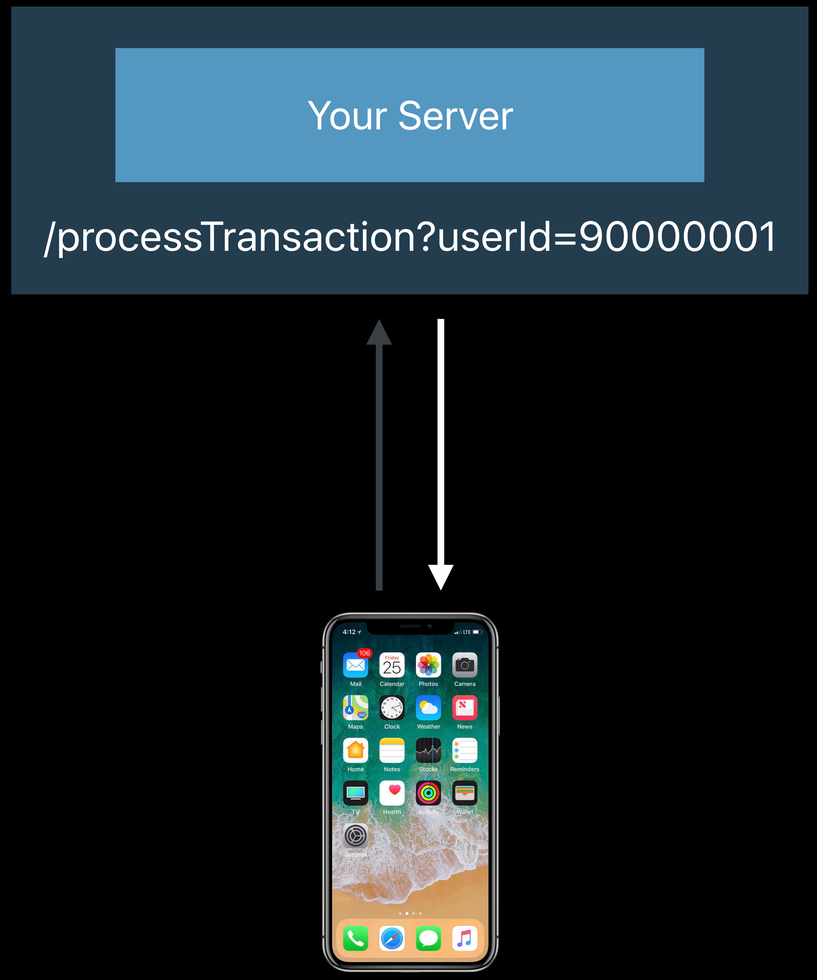
Update Subscription State

```
{ receipt: {
   in_app: [{
     transaction_id: “1234567890”,
     original_transaction_id: “1133557799”,
     expires_date: “2018-07-08…”, …
   }, {
     transaction_id: “2233445566”,
     original_transaction_id: “1133557799”,
     expires_date: “2018-08-08…”, …
   }]
  }
}
```



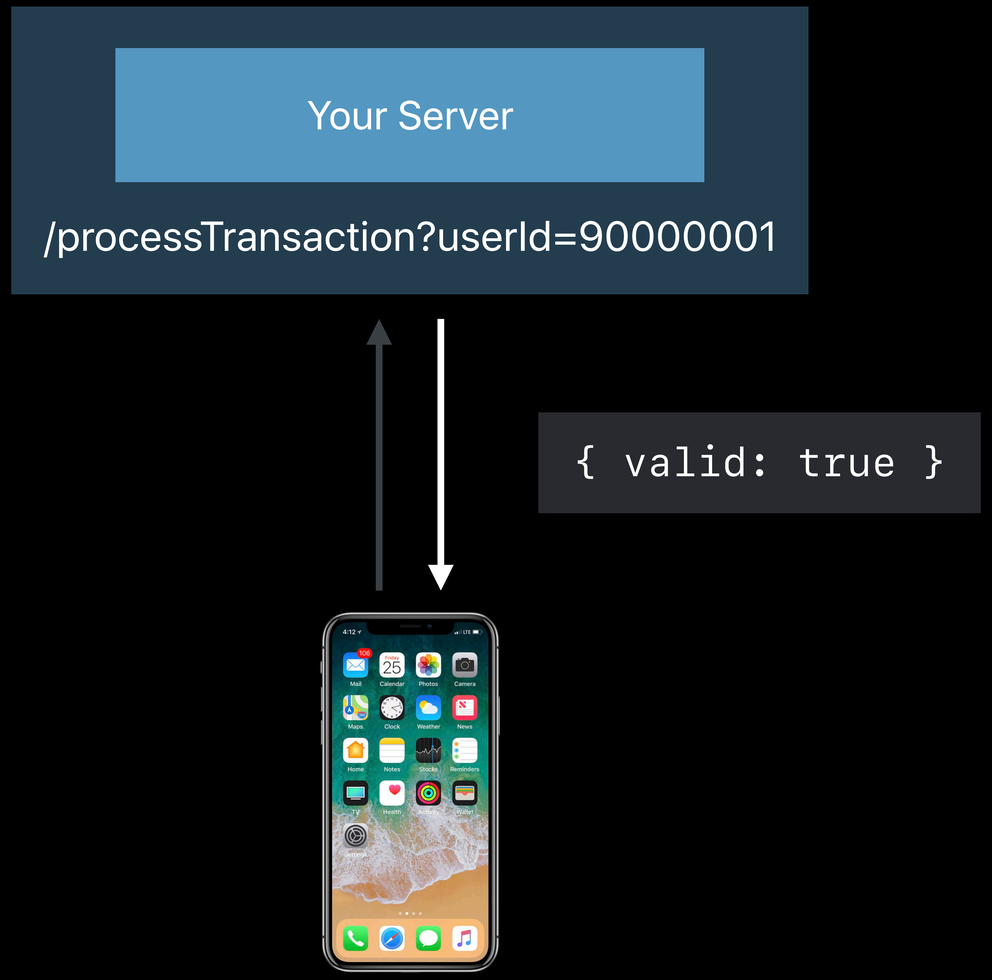
Finish All Transactions on Device

Including all renewal transactions



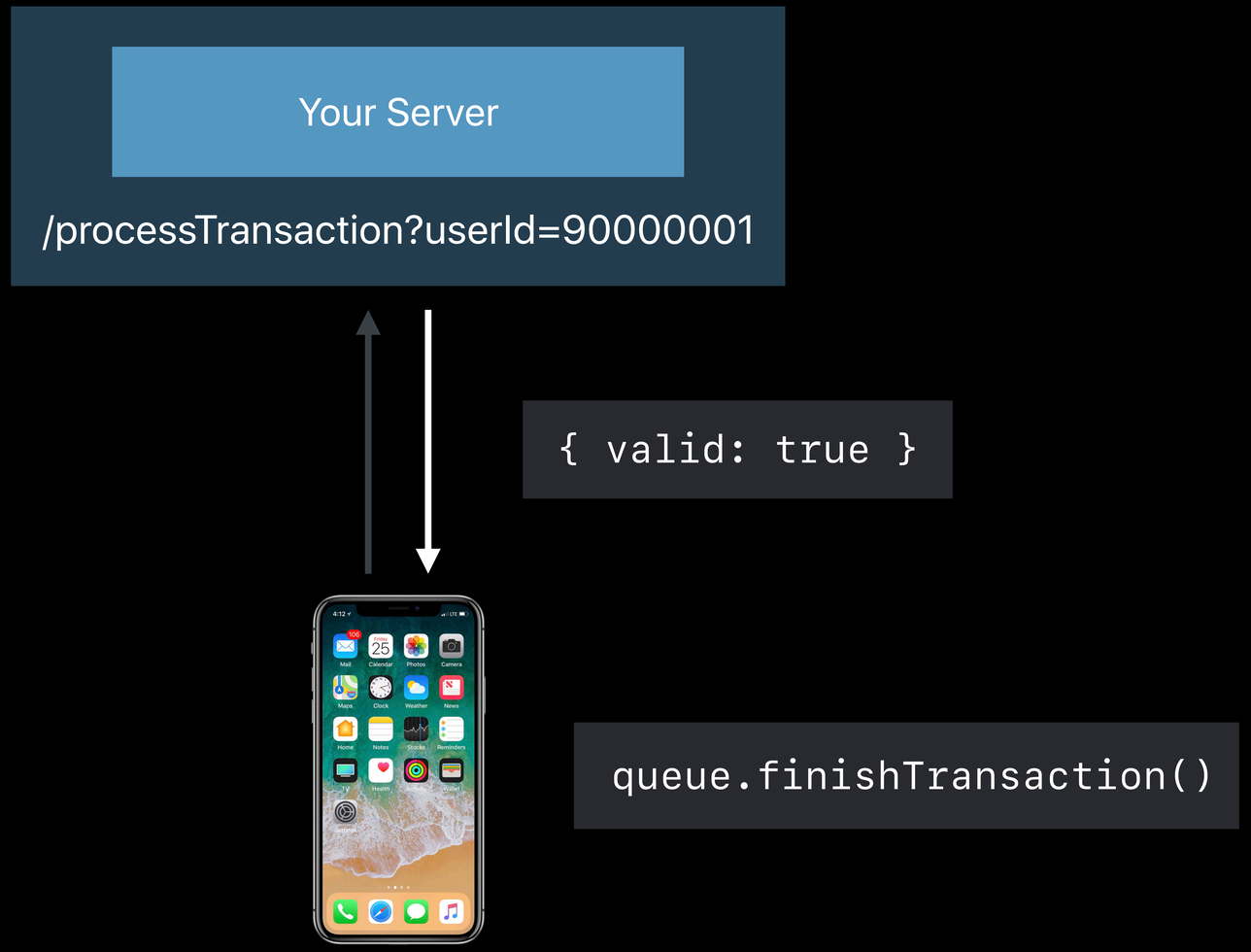
Finish All Transactions on Device

Including all renewal transactions

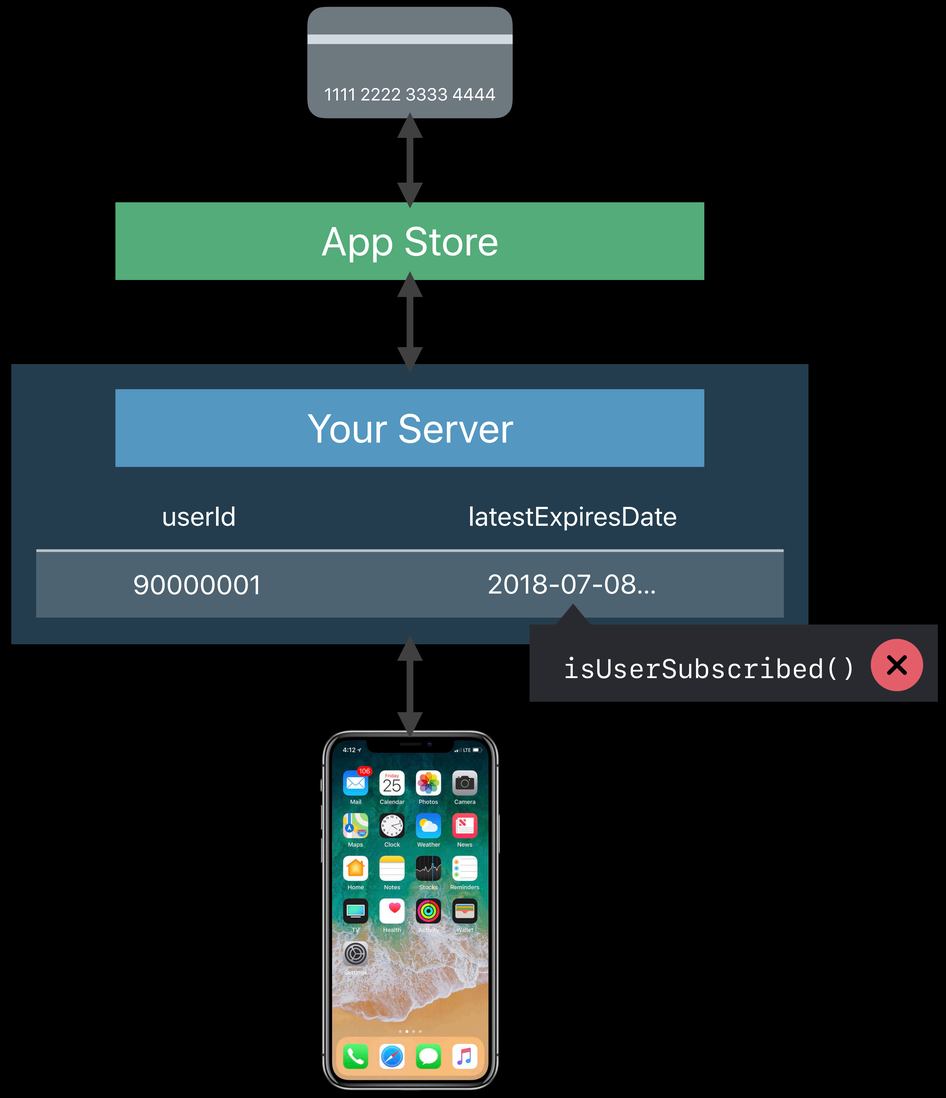


Finish All Transactions on Device

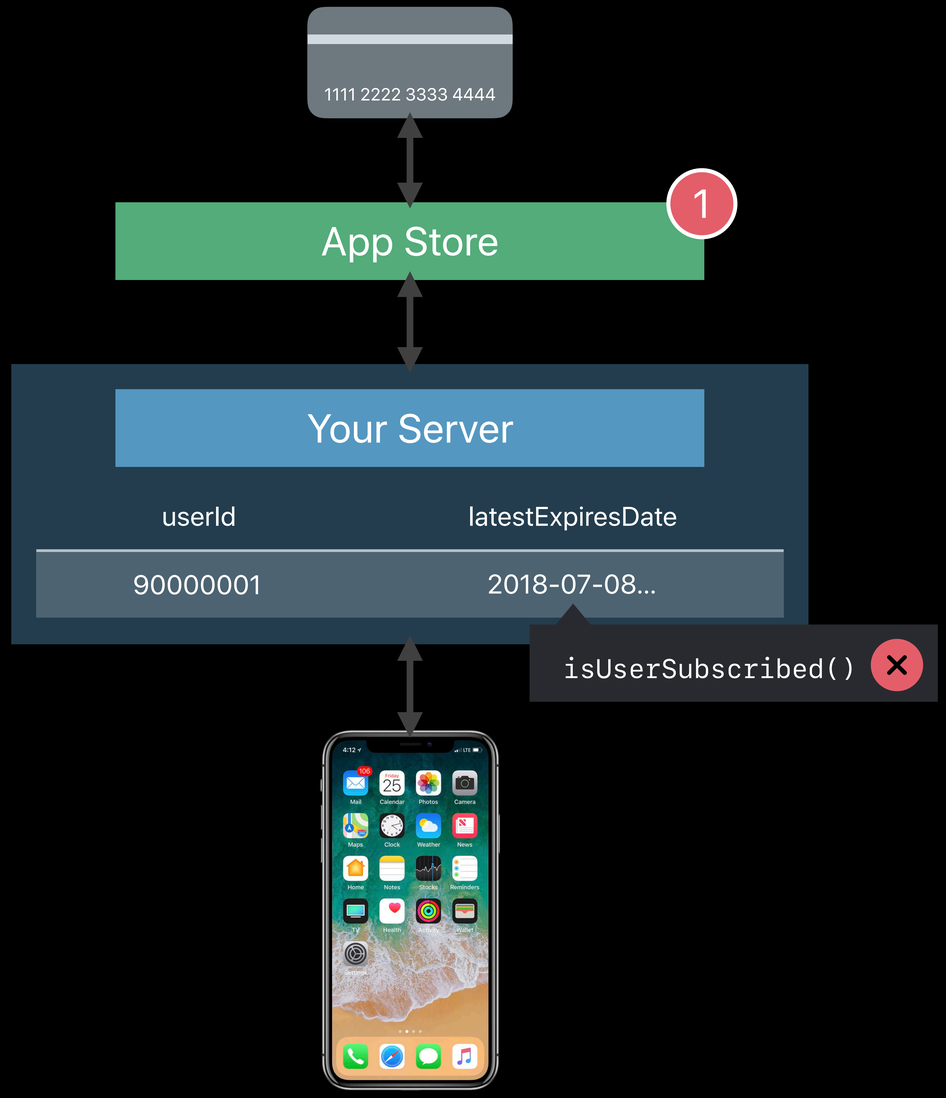
Including all renewal transactions



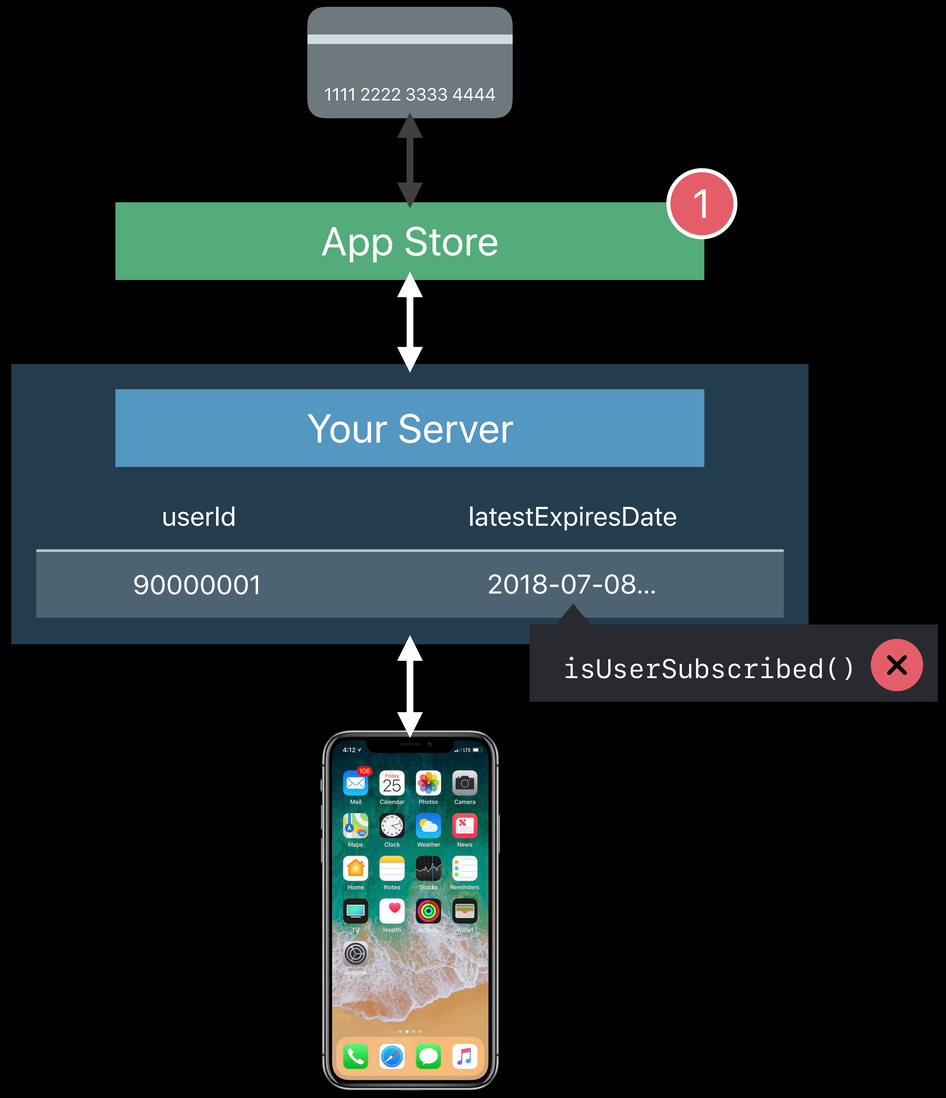
Renewal Transactions



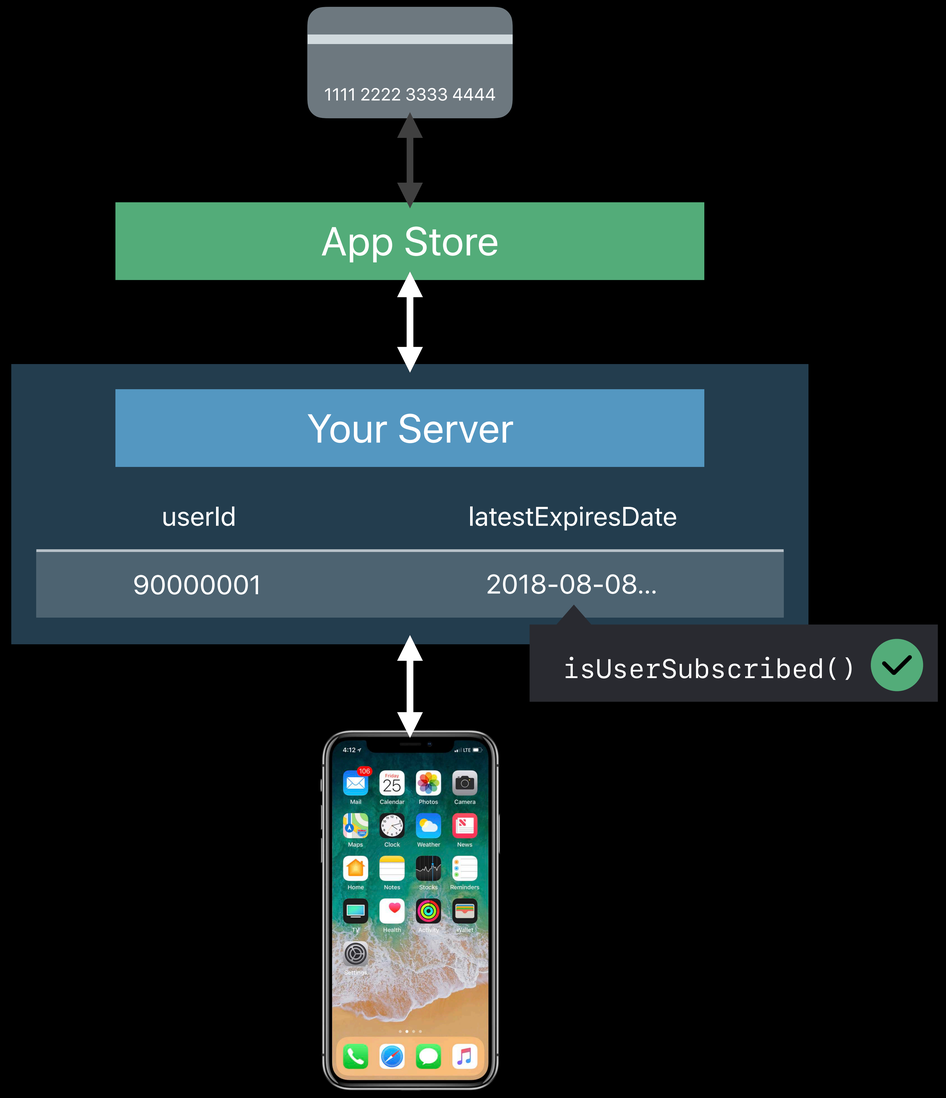
Renewal Transactions



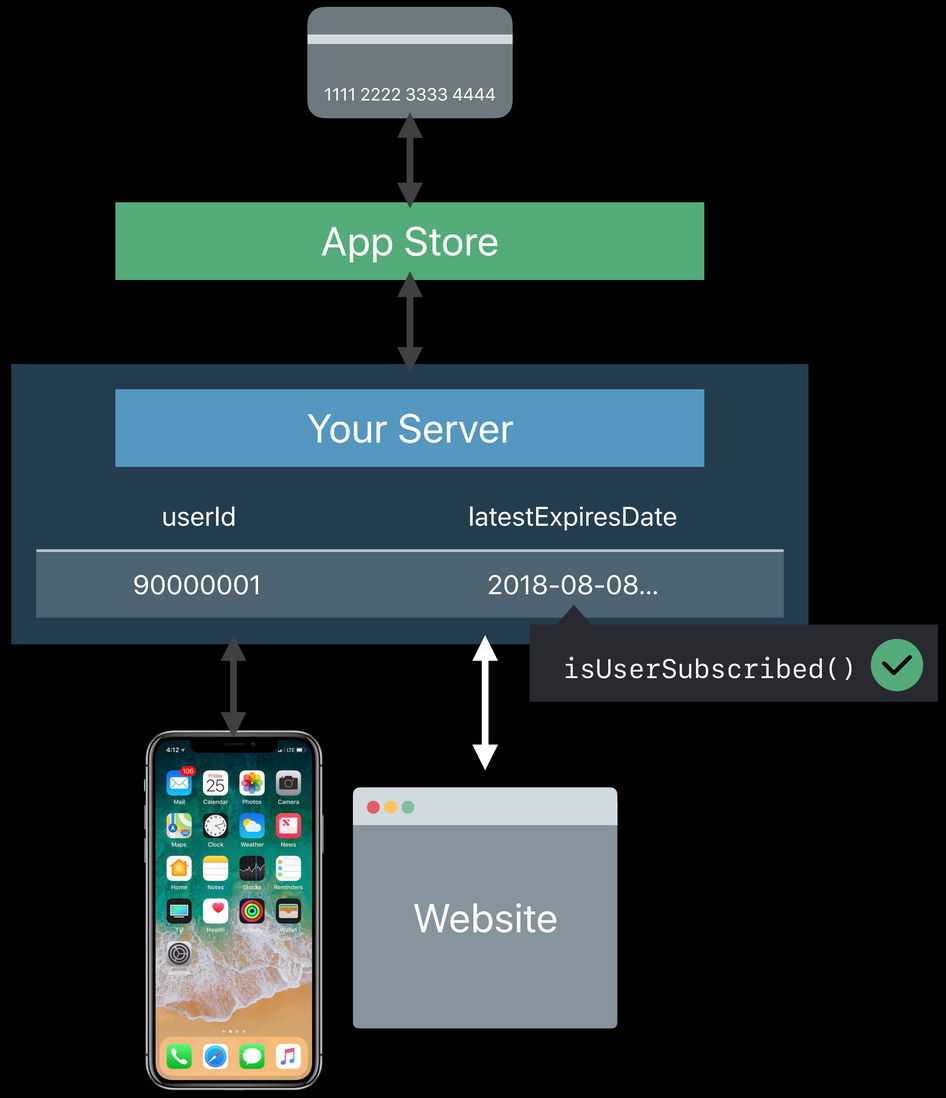
Renewal Transactions



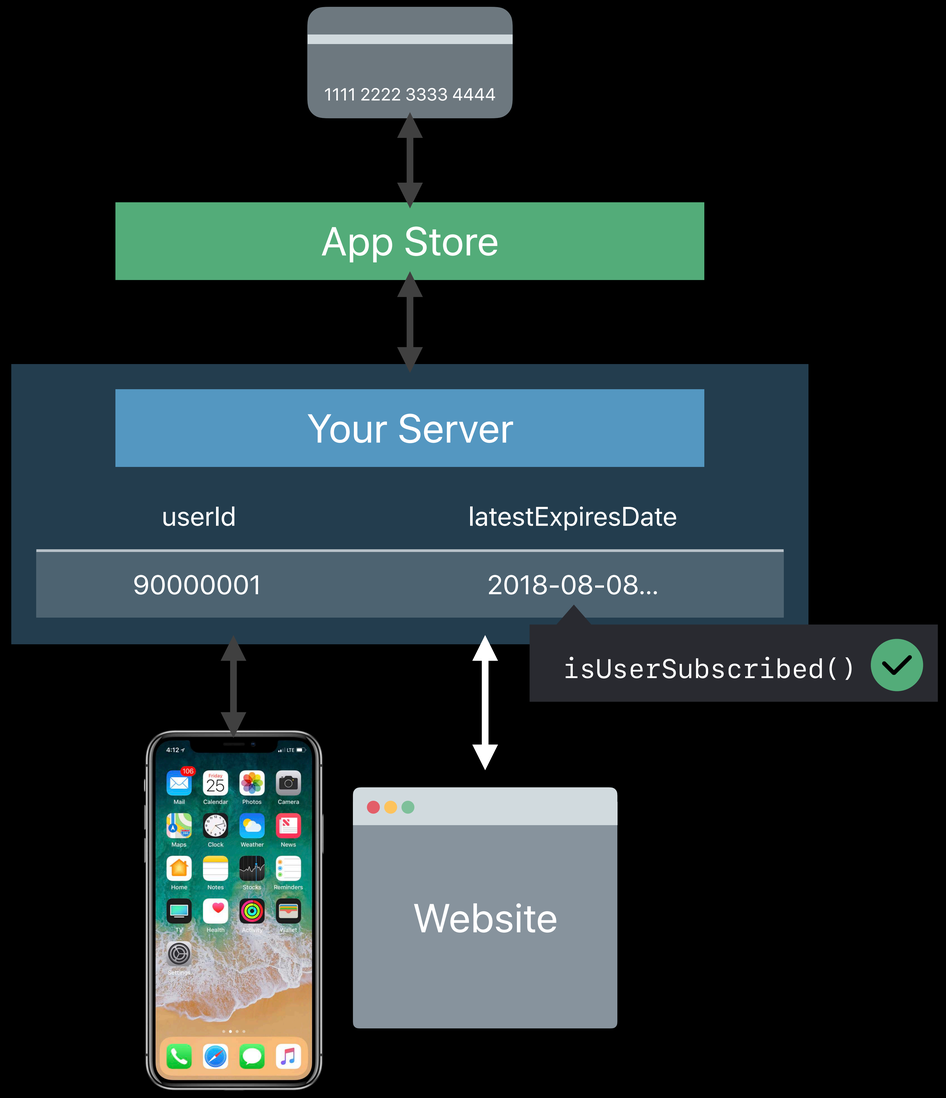
Renewal Transactions



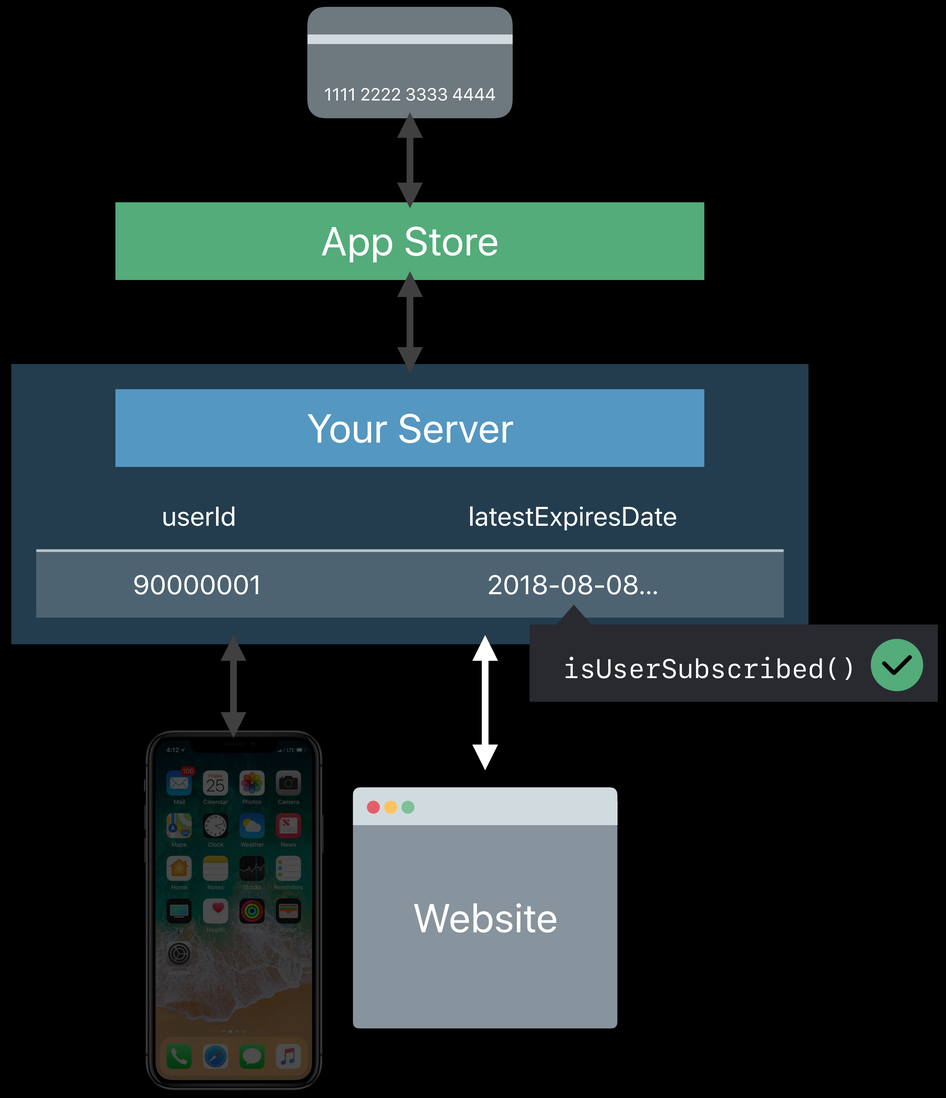
Multiple Platforms



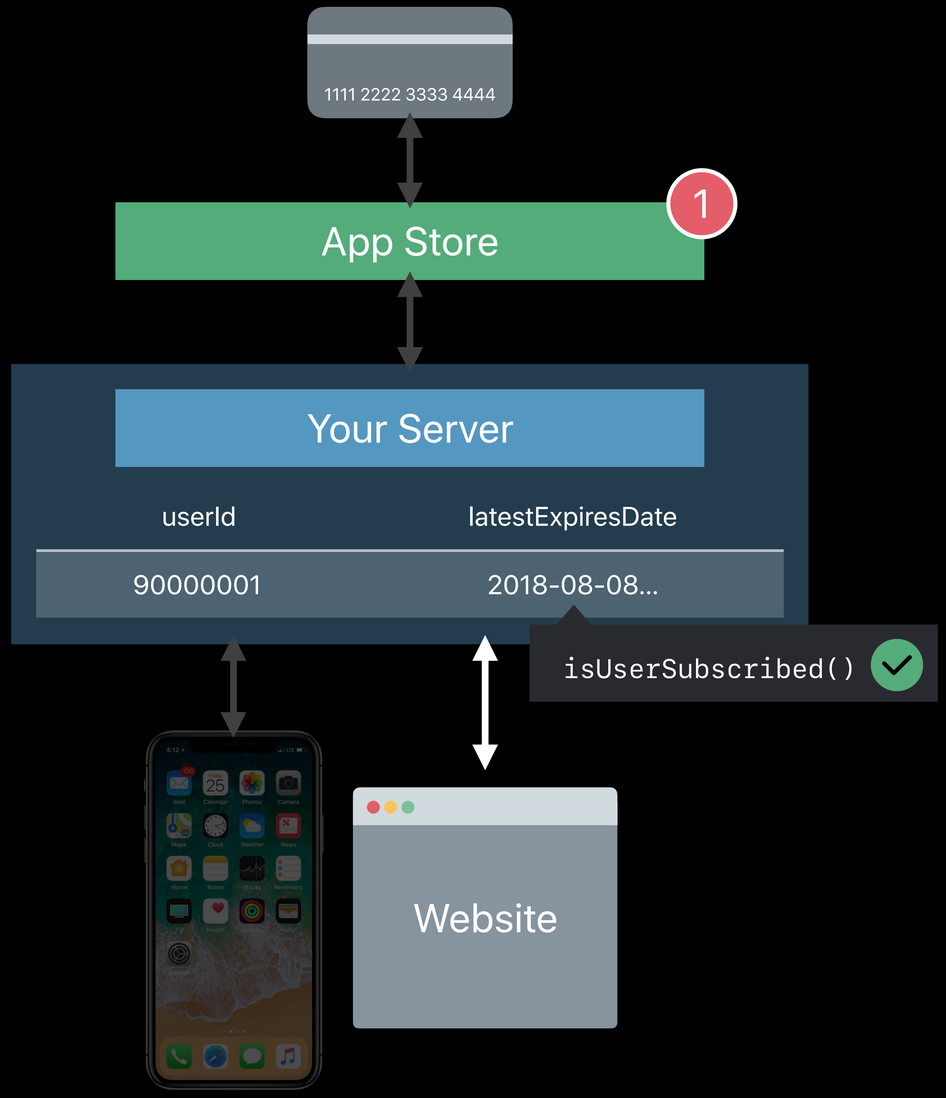
Multiple Platforms



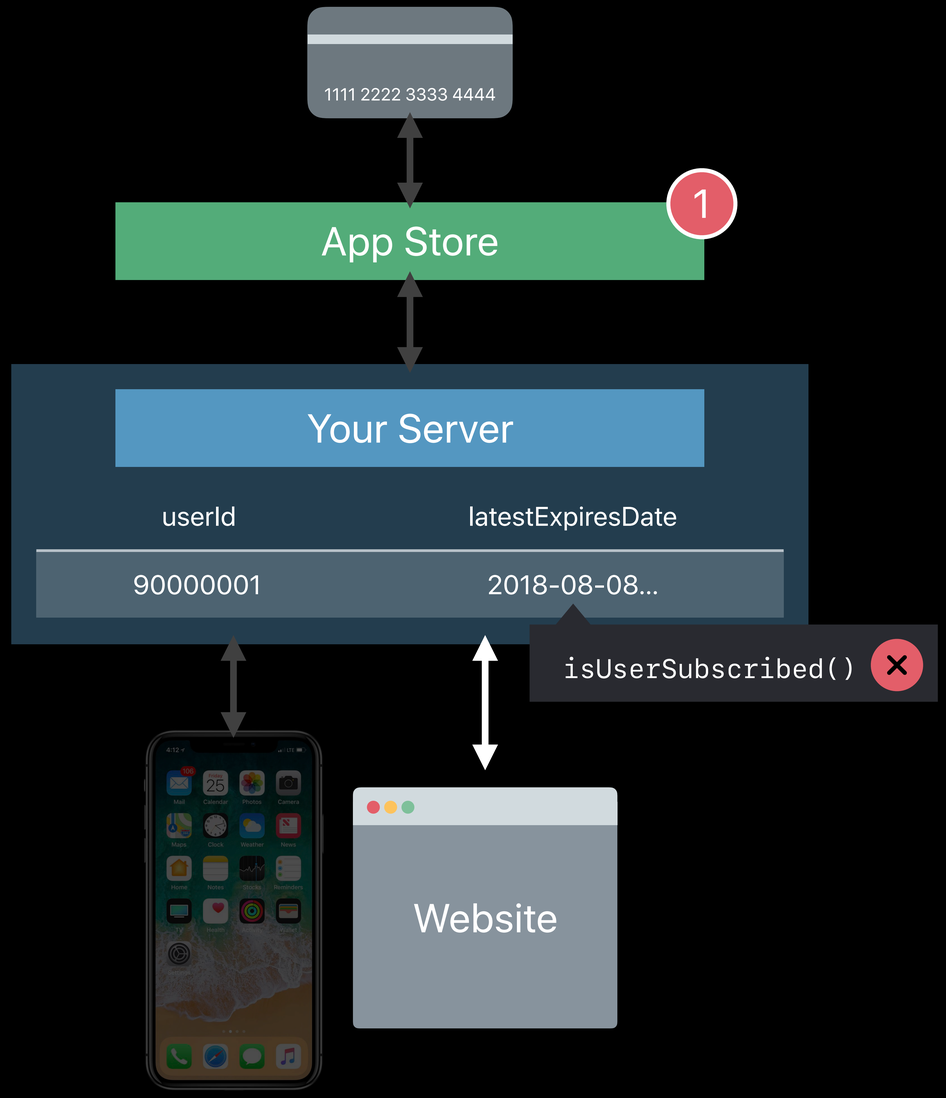
Multiple Platforms



Multiple Platforms



Multiple Platforms



Status Polling

Discover new transactions directly from your server

Status Polling

Discover new transactions directly from your server

Save latest version of encoded receipt data on your server

Status Polling

Discover new transactions directly from your server

Save latest version of encoded receipt data on your server

Treat receipt data like a token

Status Polling

Discover new transactions directly from your server

Save latest version of encoded receipt data on your server

Treat receipt data like a token

`/verifyReceipt` response also includes new transactions

Status Polling

Discover new transactions directly from your server

Save latest version of encoded receipt data on your server

Treat receipt data like a token

`/verifyReceipt` response also includes new transactions

- Located in the `latest_receipt_info` field

Status Polling

Discover new transactions directly from your server

Save latest version of encoded receipt data on your server

Treat receipt data like a token

`/verifyReceipt` response also includes new transactions

- Located in the `latest_receipt_info` field
- Unlock new subscription periods without waiting for app to launch

Verifying Renewal Transactions

Your Server

/processTransaction?userId=90000001

Verifying Renewal Transactions

Your Server

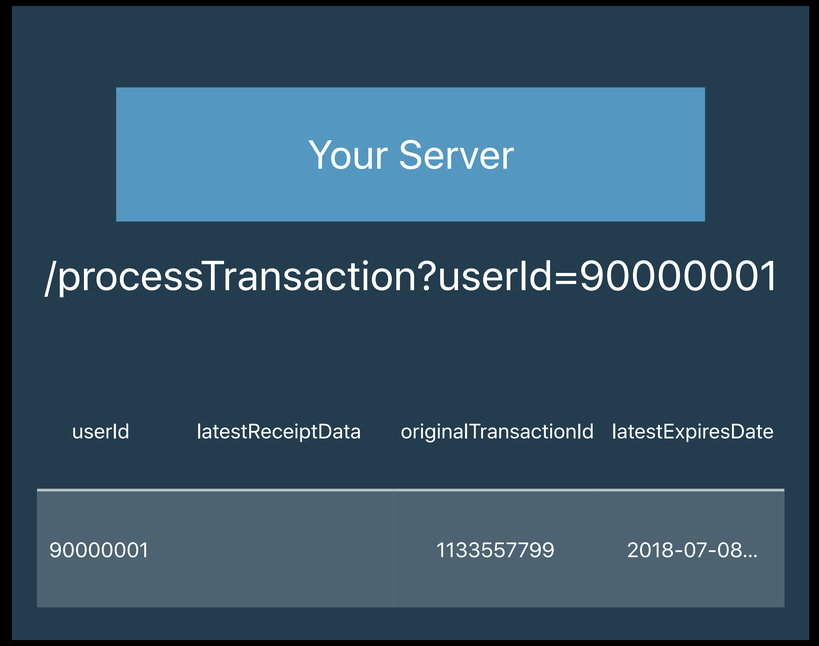
/processTransaction?userId=90000001

```
{ receiptData: “d24Fs…kJ87dDGe3=“ }
```

Update Subscription State

Store receipt data with user

```
{ receipt: {
   in_app: [{
     transaction_id: “1234567890”,
     original_transaction_id: “1133557799”,
     expires_date: “2018-07-08…”, …
   }, {
     transaction_id: “2233445566”,
     original_transaction_id: “1133557799”,
     expires_date: “2018-08-08…”, …
   }]
  }
}
```



Update Subscription State

Store receipt data with user

```
{ receipt: {
    in_app: [{
      transaction_id: "1234567890",
      original_transaction_id: "1133557799",
      expires_date: "2018-07-08…", …
    },
      original_transaction_id: "1133557799",
      expires_date: "2018-08-08…", …
    }]
  }
}
```

```
{ receiptData: "d24Fs…kJ87dDGe3=" }
```

Your Server

/processTransaction?userId=90000001

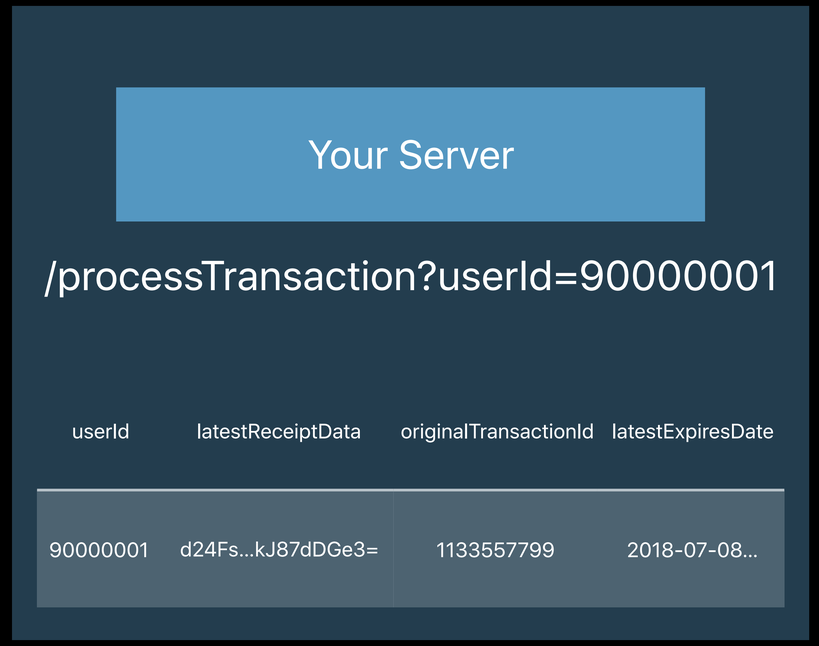
userId	latestReceiptData	originalTransactionId	latestExpiresDate
90000001		1133557799	2018-07-08...

Update Subscription State

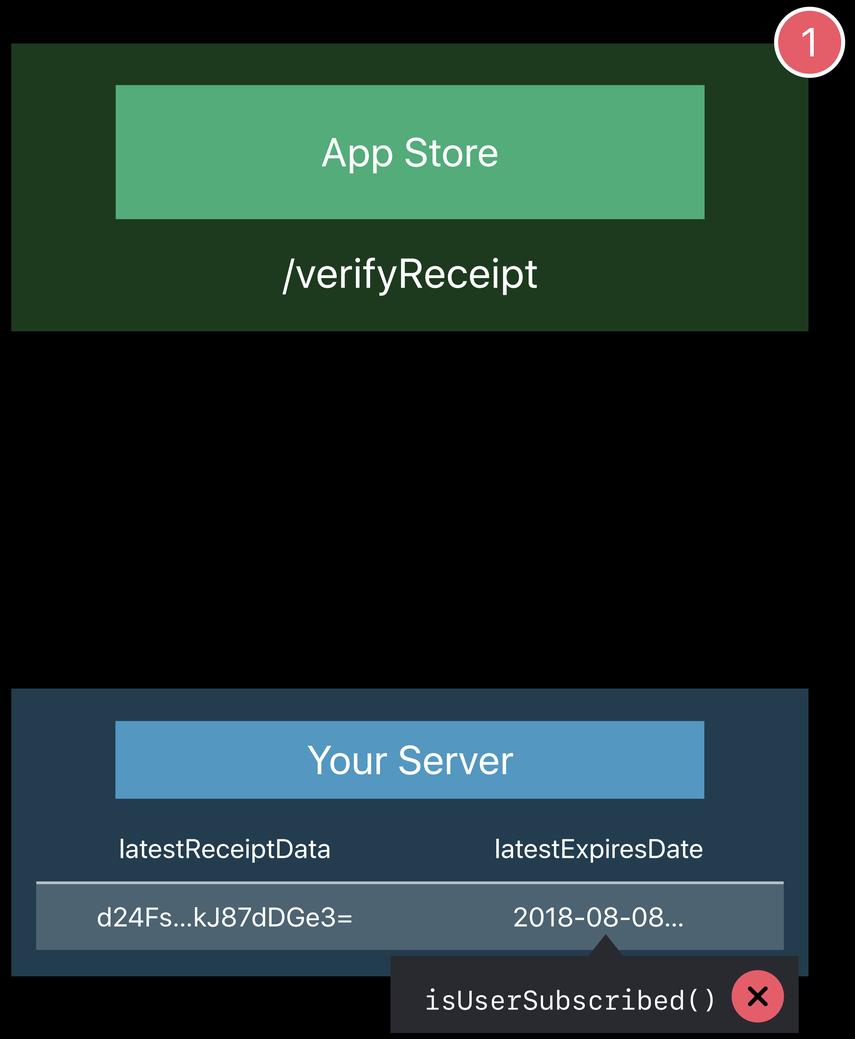
Store receipt data with user

```
{ receipt: {
    in_app: [{
      transaction_id: "1234567890",
      original_transaction_id: "1133557799",
      expires_date: "2018-07-08…", …
    },{
      transaction_id: "1234567890",
      original_transaction_id: "1133557799",
      expires_date: "2018-08-08…", …
    }]
  }
}
```

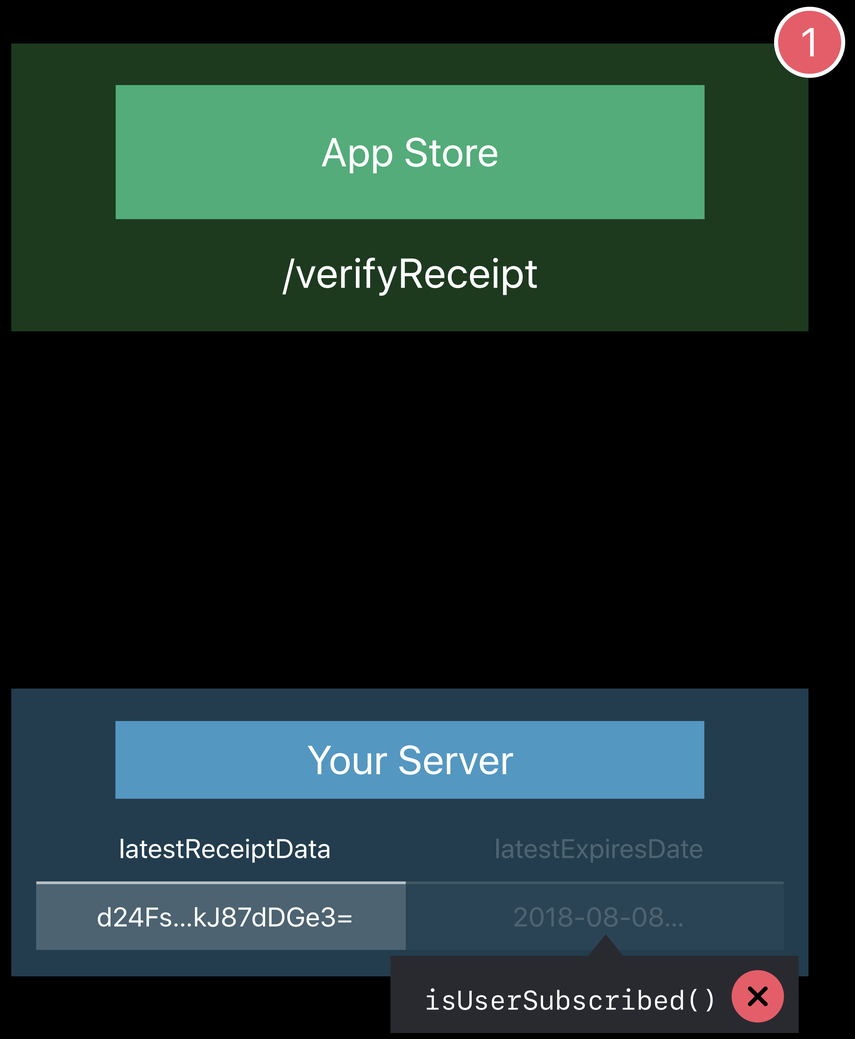
```
{ receiptData: "d24Fs…kJ87dDGe3=" }
```



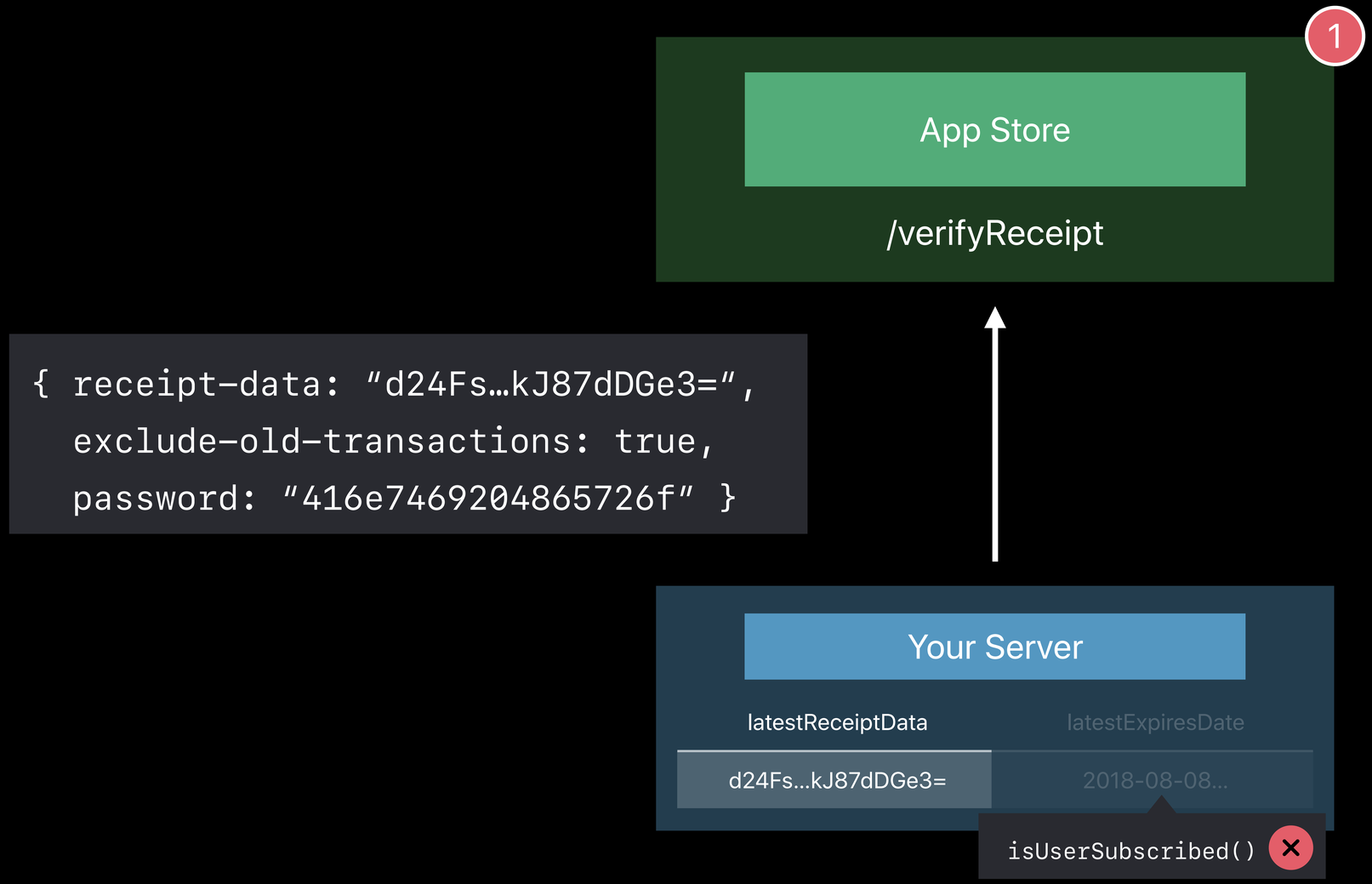
Status Polling



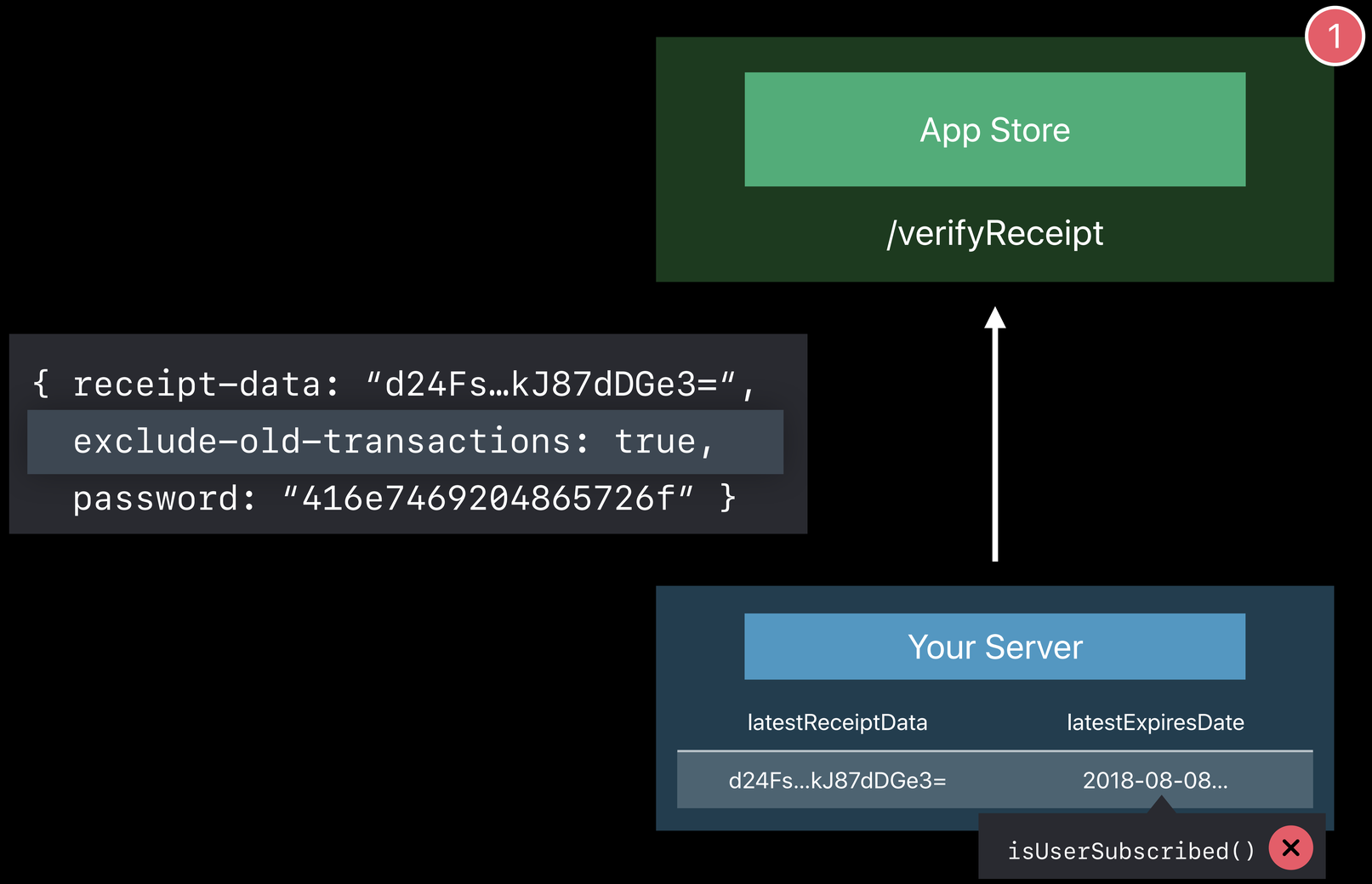
Status Polling



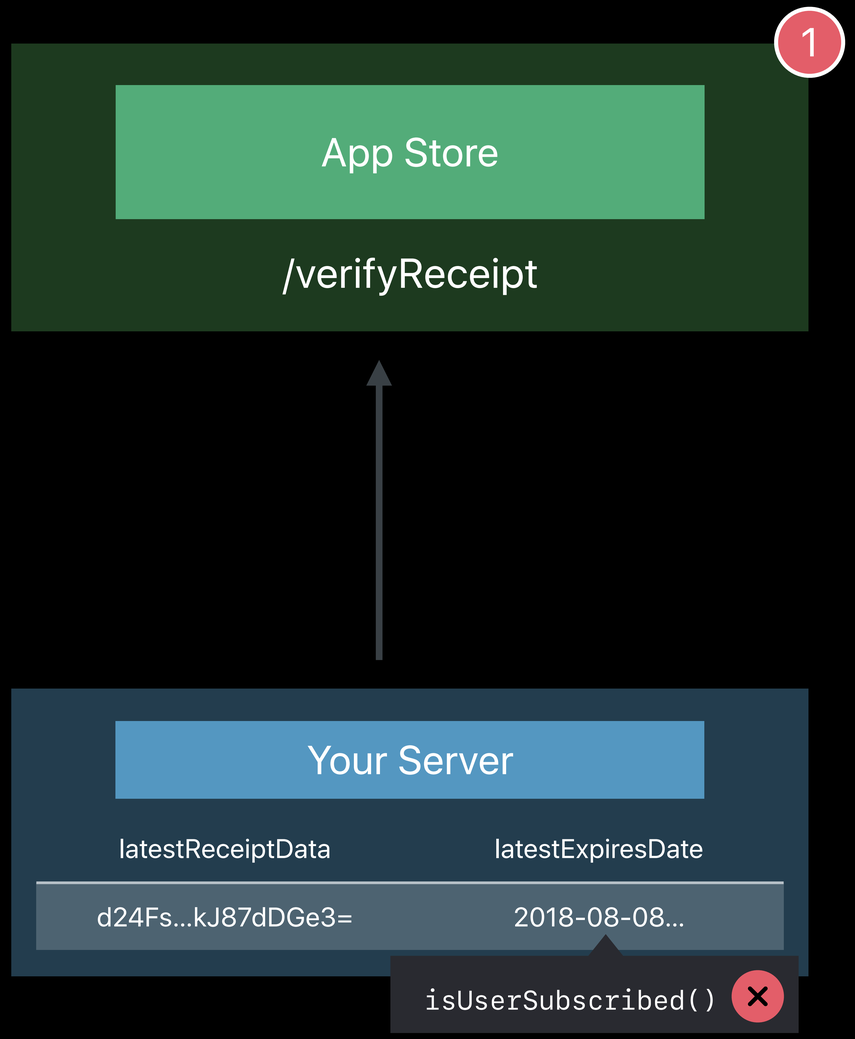
Status Polling



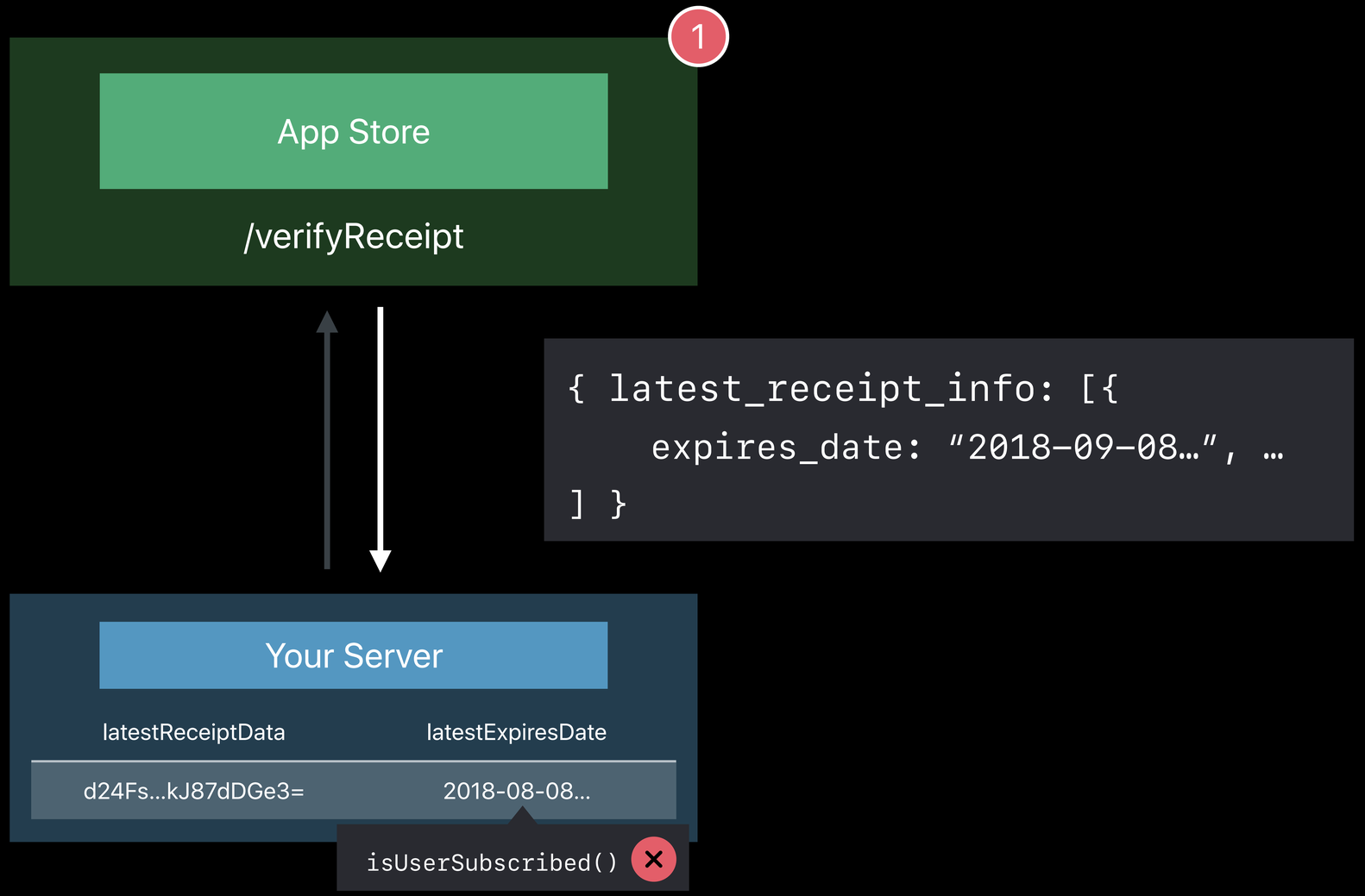
Status Polling



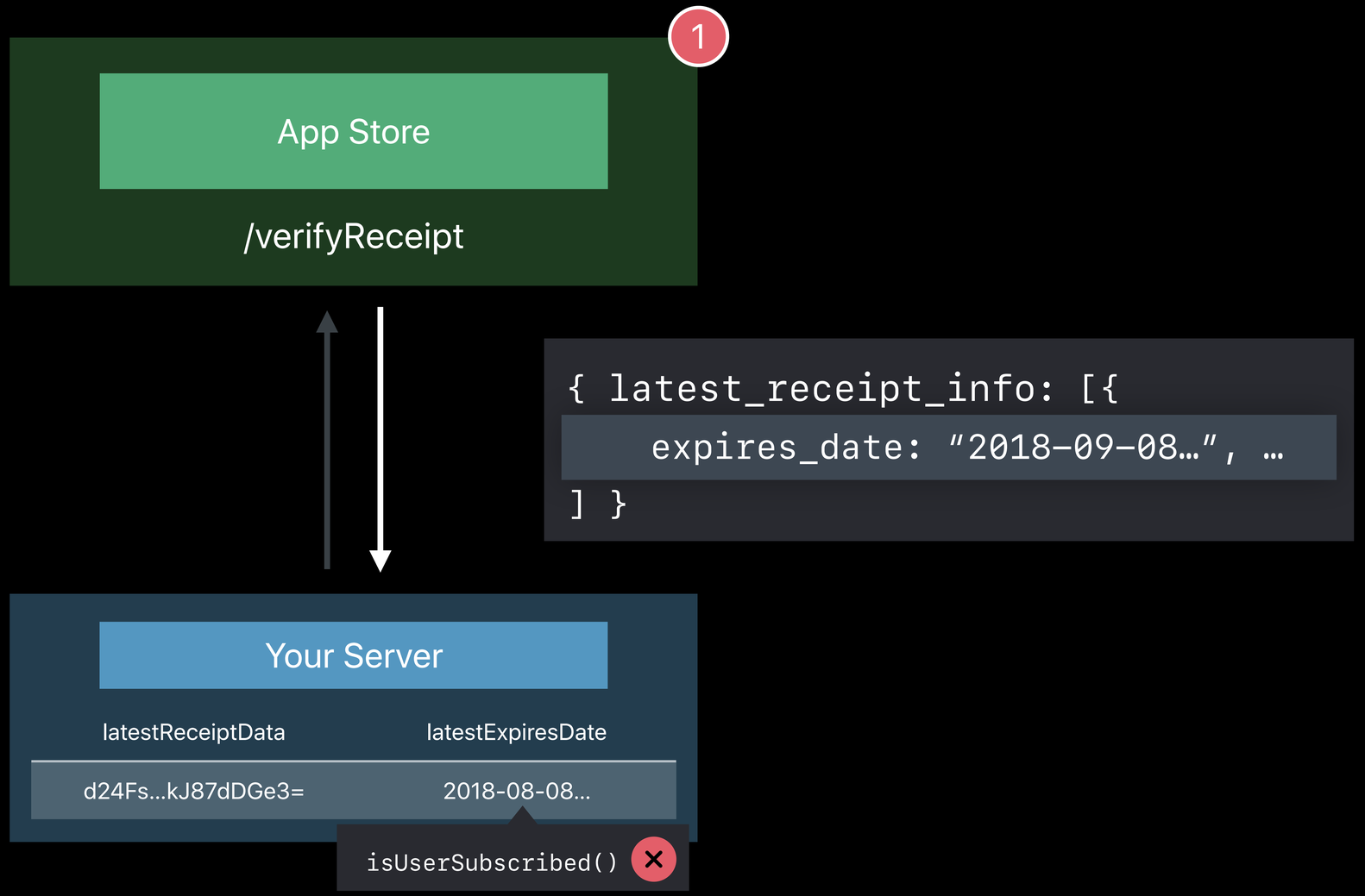
Status Polling



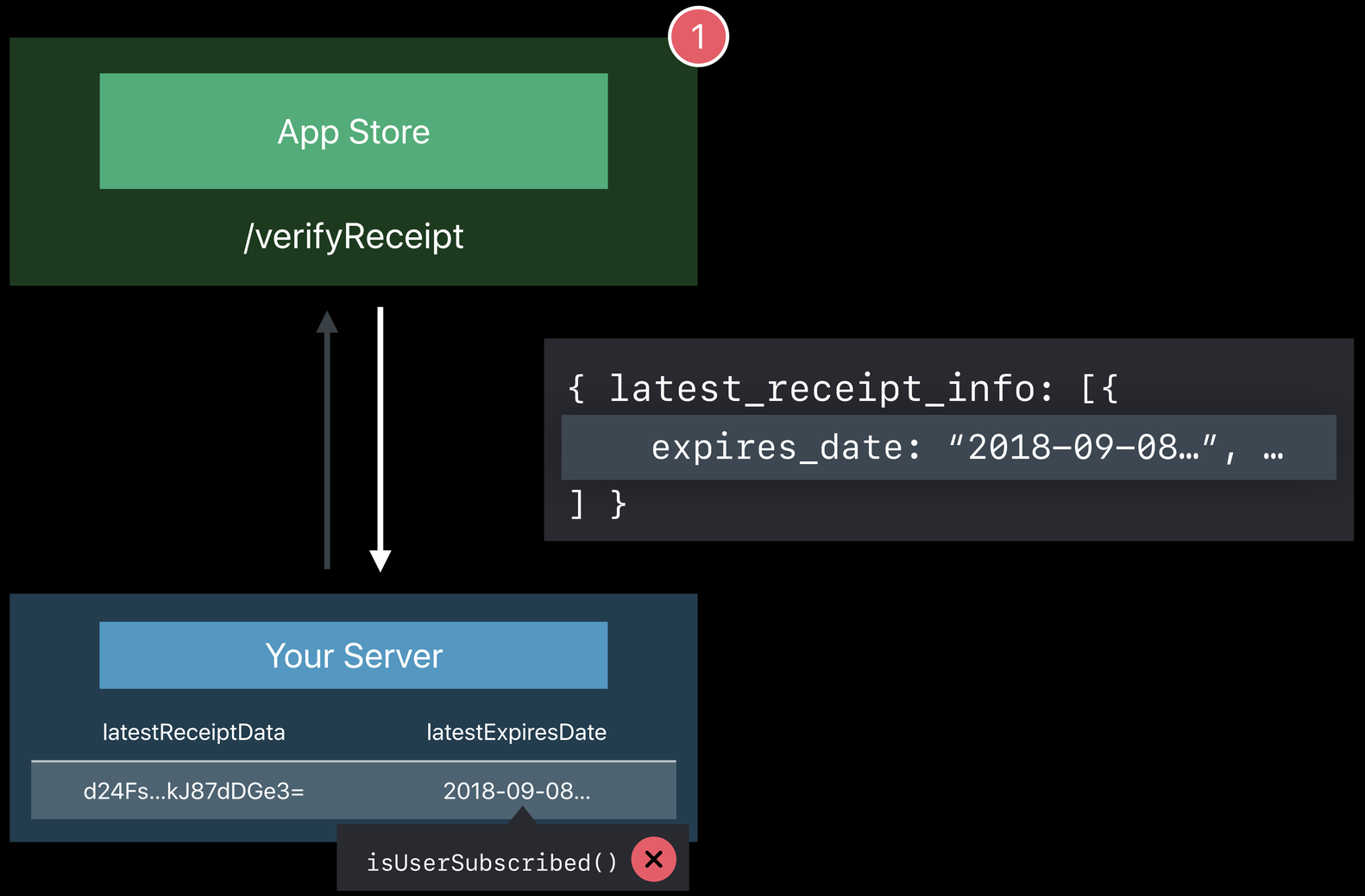
Status Polling



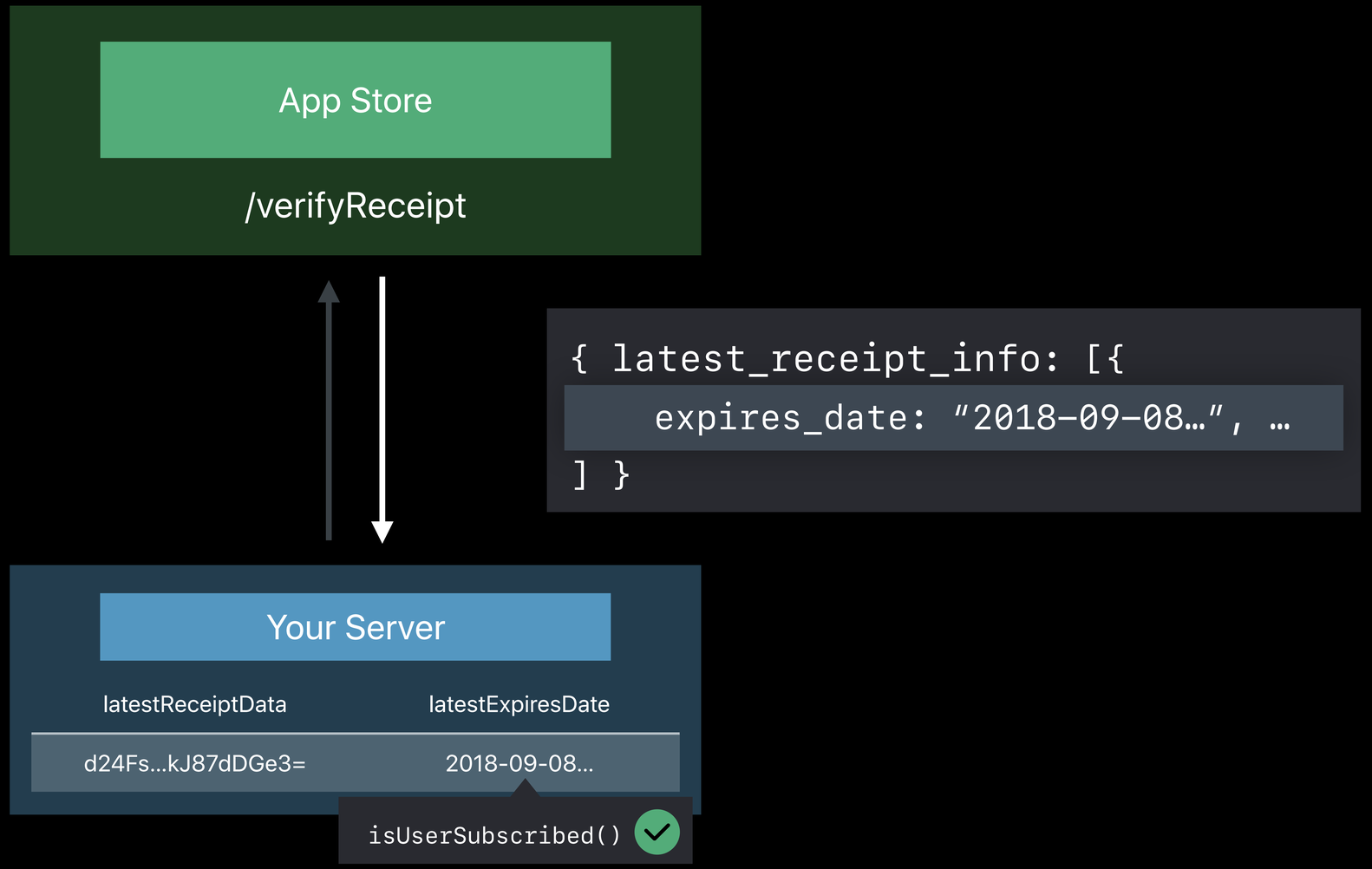
Status Polling



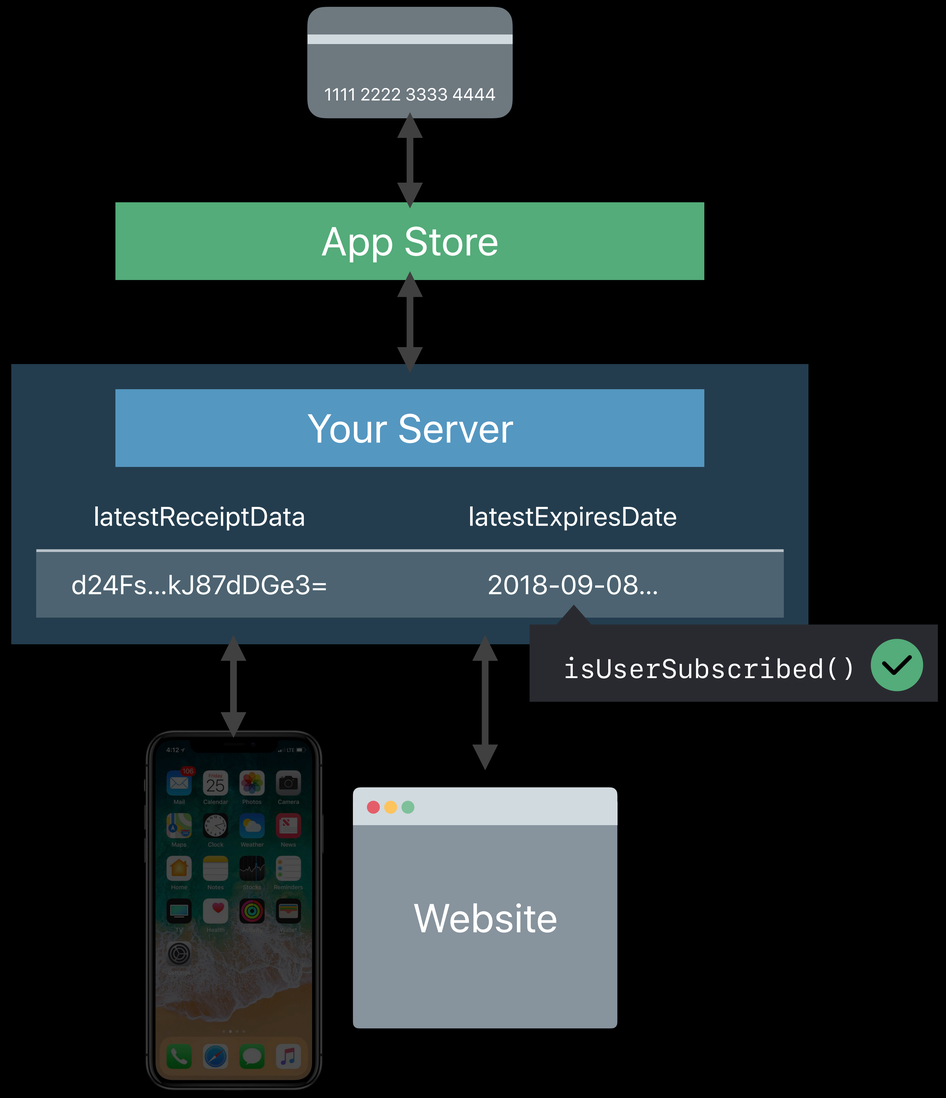
Status Polling



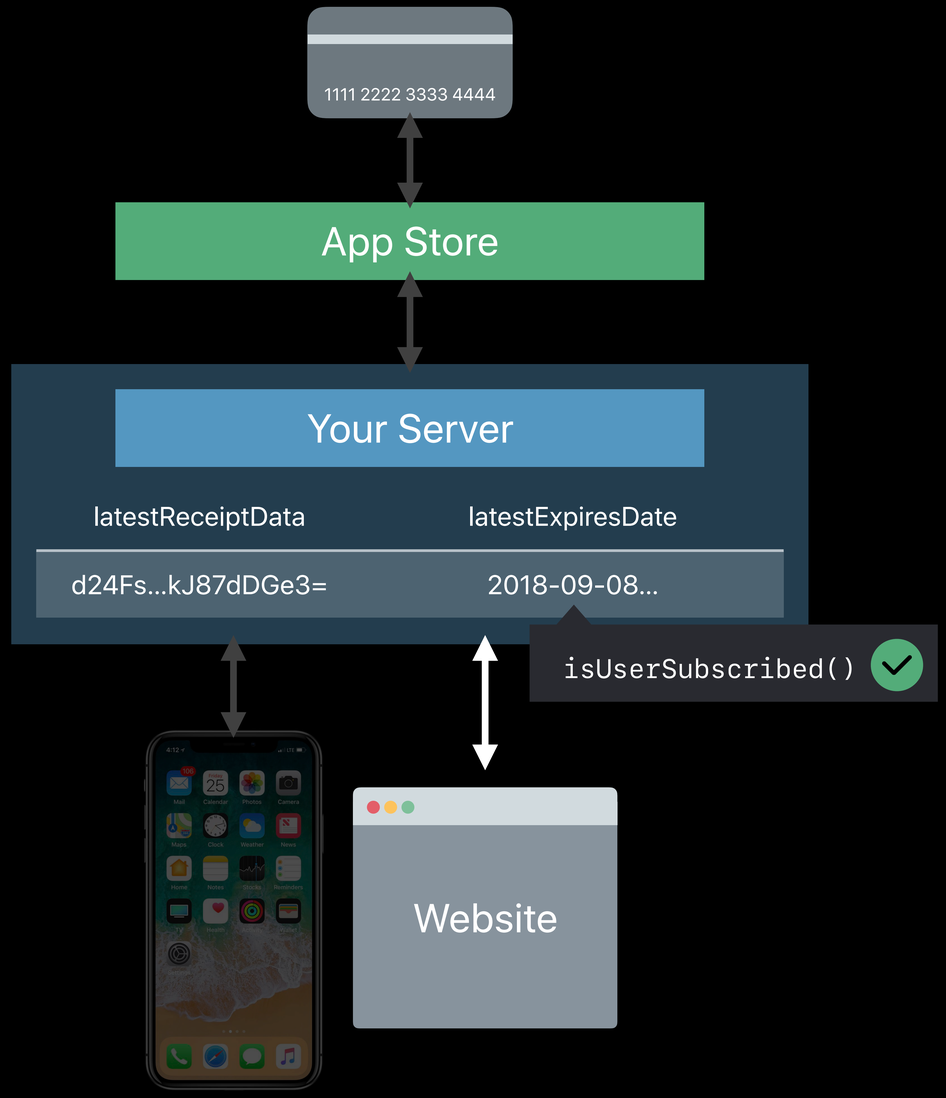
Status Polling



Multiple Platforms



Multiple Platforms



Status Polling

Discover new transactions directly from your server

Status Polling

Discover new transactions directly from your server

New transactions will still appear in StoreKit on next app launch

Status Polling

Discover new transactions directly from your server

New transactions will still appear in StoreKit on next app launch

- Must verify and finish these transactions

Status Polling

Discover new transactions directly from your server

New transactions will still appear in StoreKit on next app launch

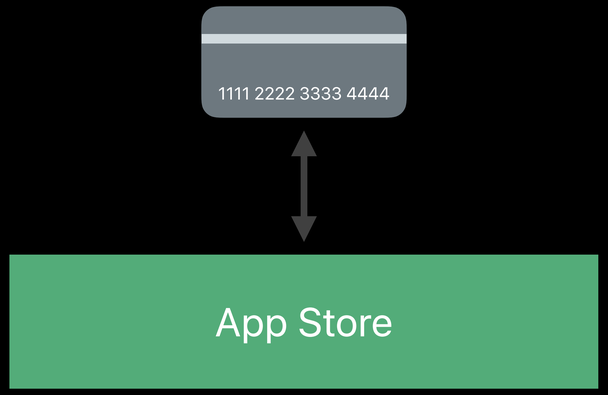
- Must verify and finish these transactions
- Even if your server already knows about them

Status Polling

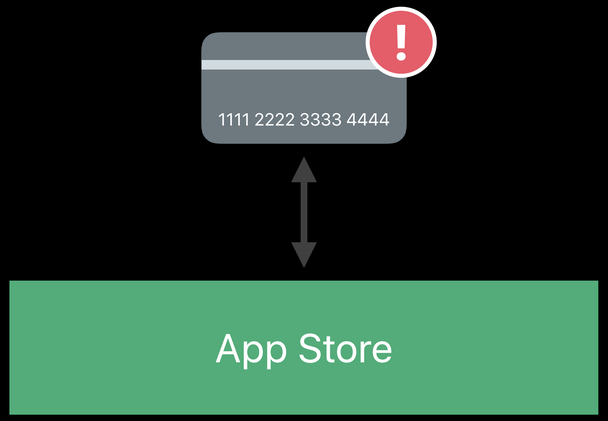
Discover new transactions directly from your server

New transactions will still appear in StoreKit on next app launch

- Must verify and finish these transactions
- Even if your server already knows about them
- Opportunity to update latest receipt data on server


1111 2222 3333 4444
App Store


1111 2222 3333 4444
1
App Store


1111 2222 3333 4444
App Store

Reacting to Billing Issues

Reacting to Billing Issues

1. Observe no renewal transaction appears

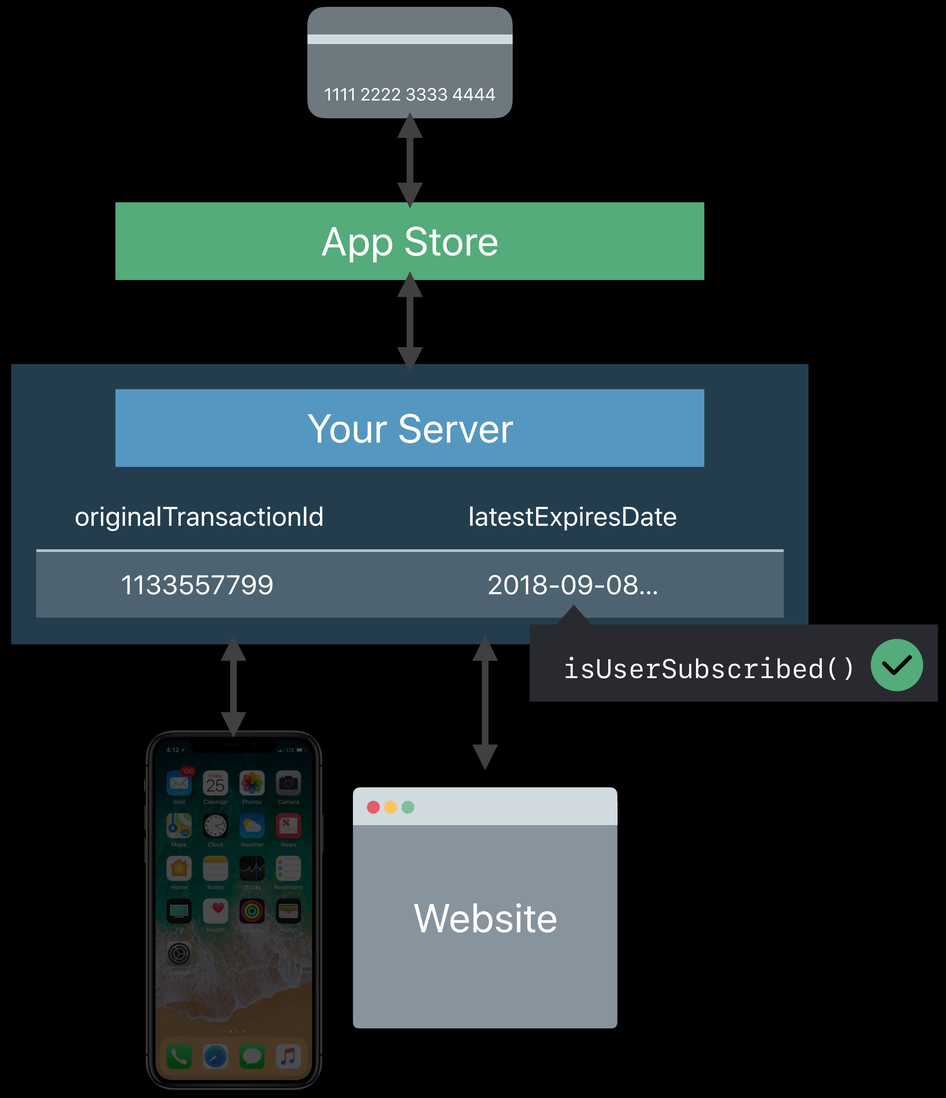
Reacting to Billing Issues

1. Observe no renewal transaction appears
2. Direct user to amend their billing details

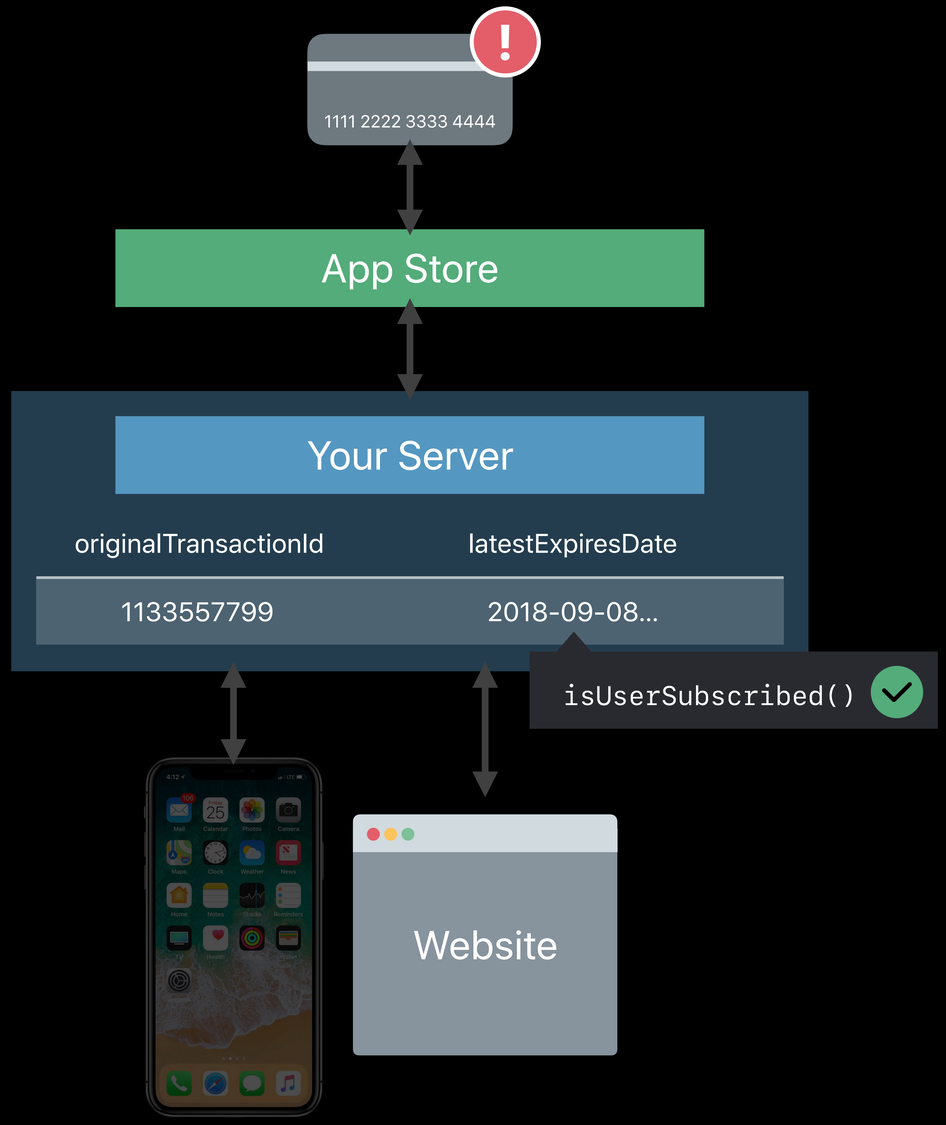
Reacting to Billing Issues

1. Observe no renewal transaction appears
2. Direct user to amend their billing details
3. Unblock user immediately when transaction occurs

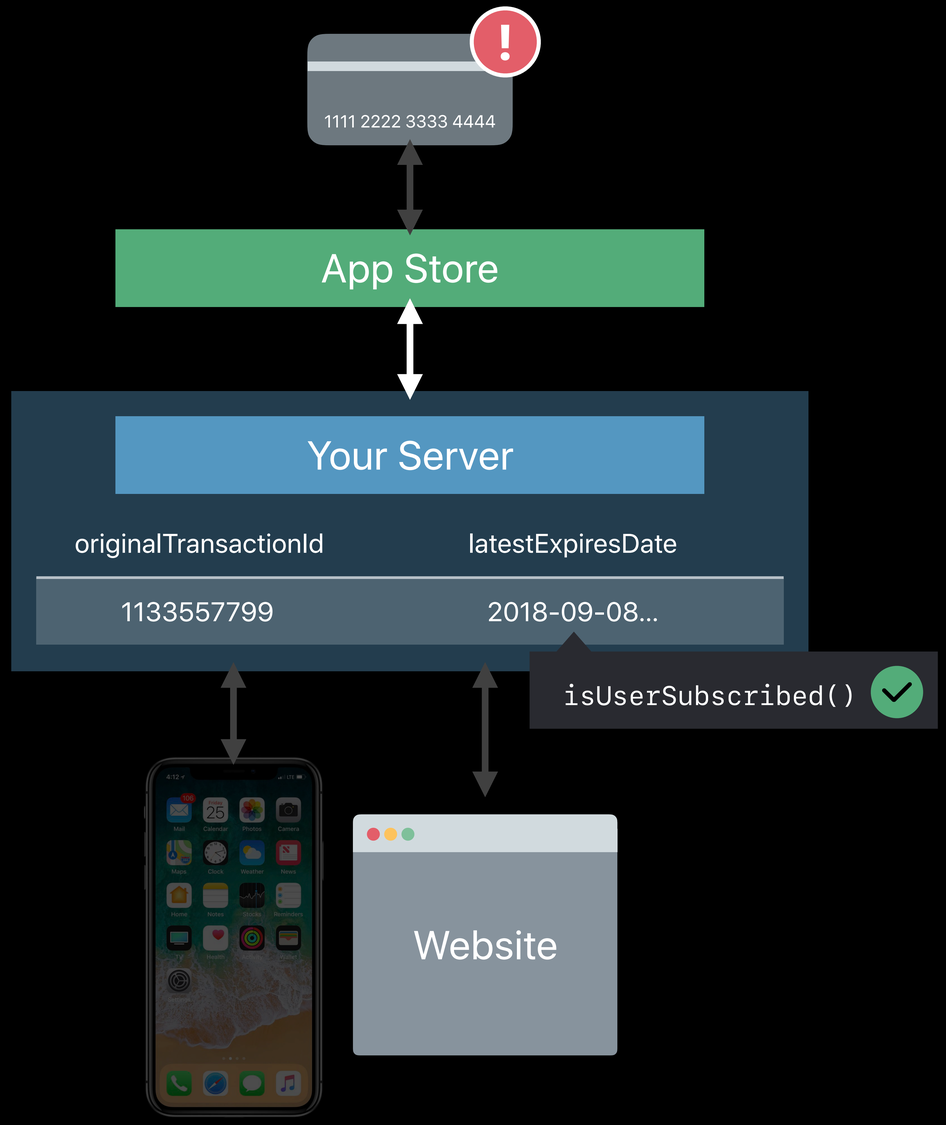
Server-to-Server Notifications



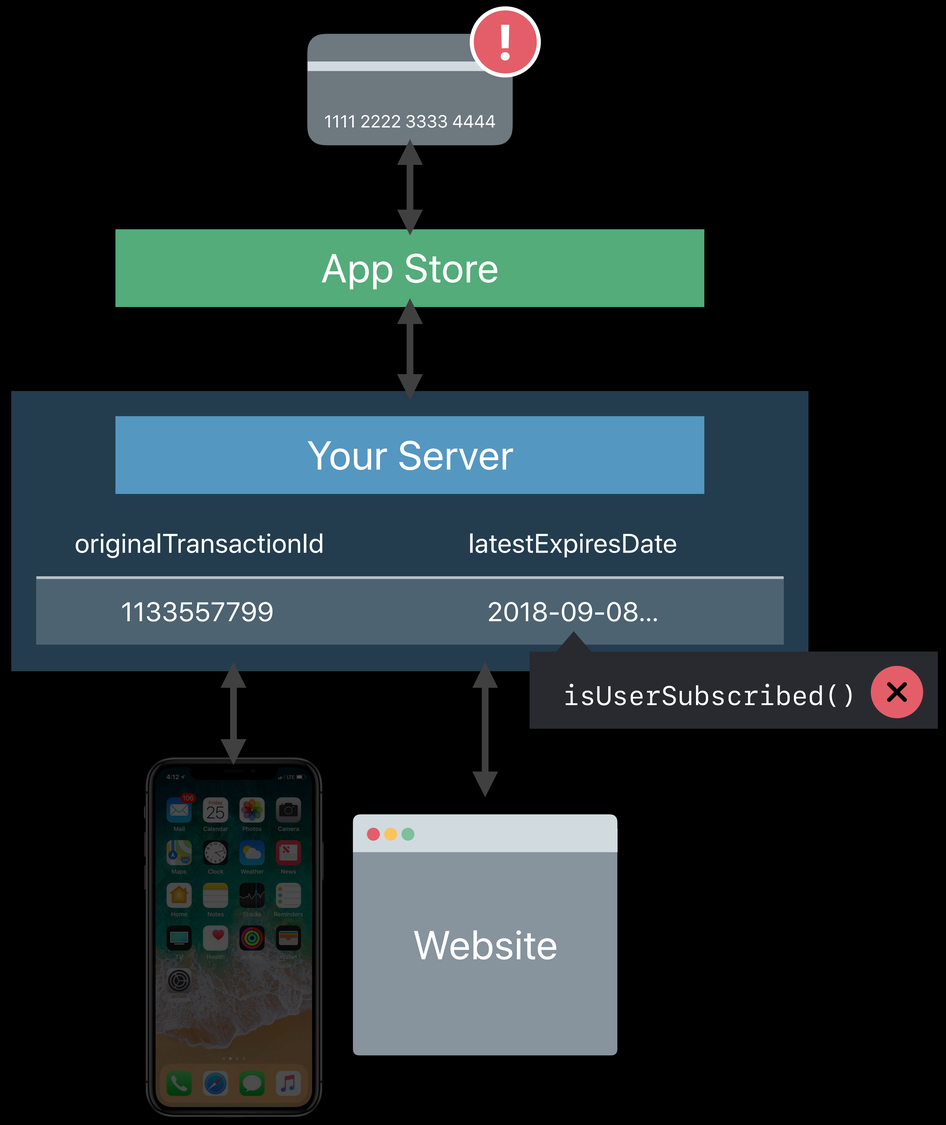
Server-to-Server Notifications



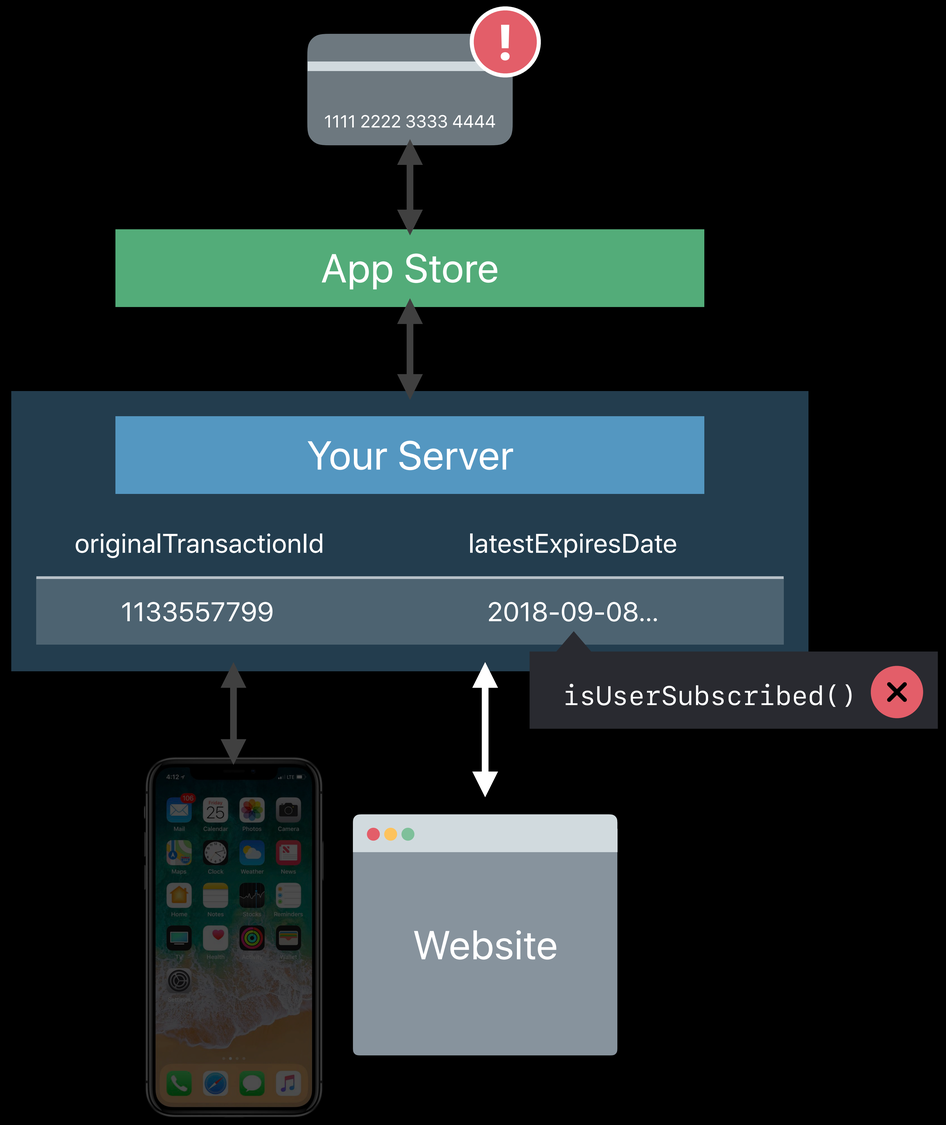
Server-to-Server Notifications



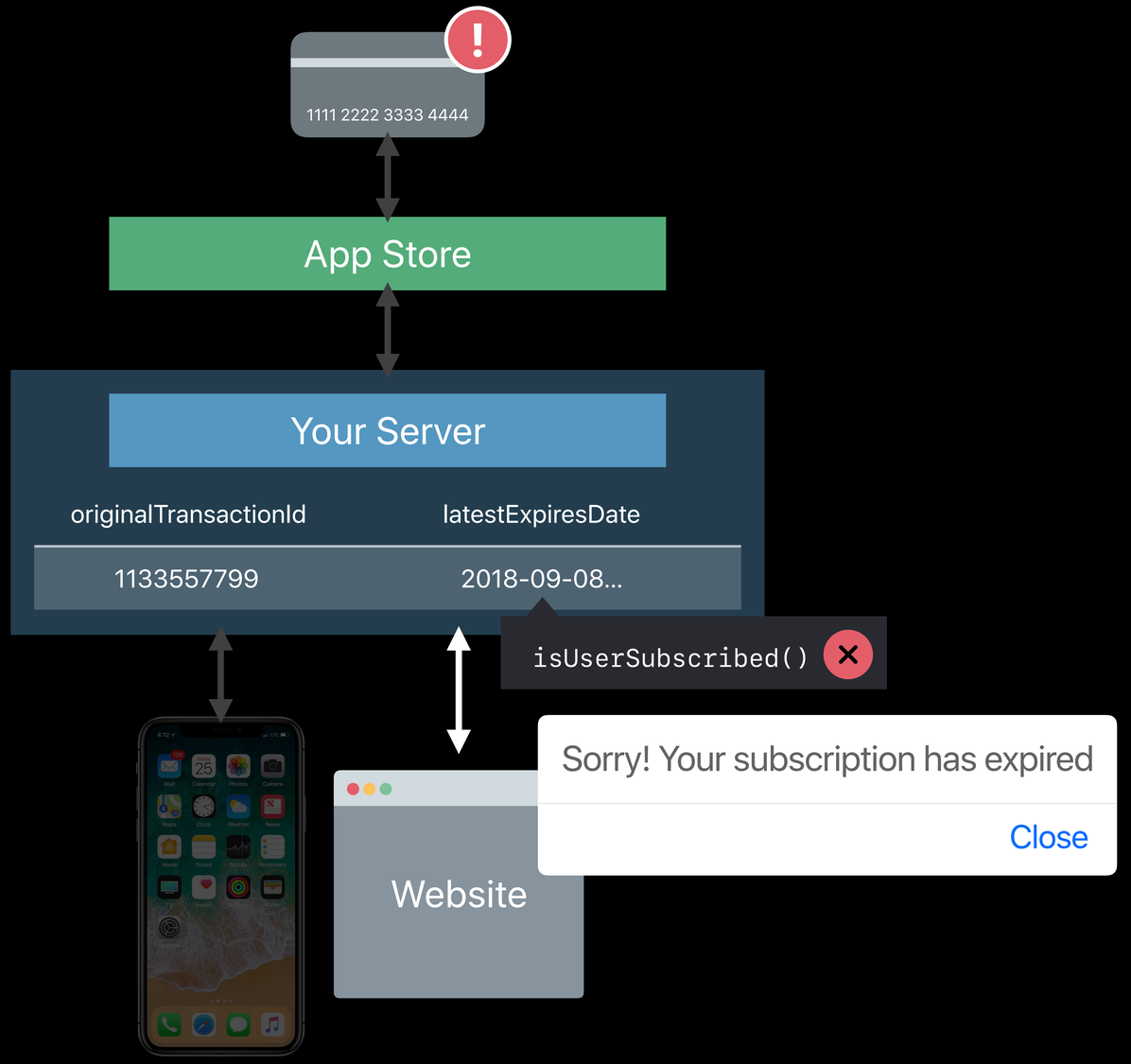
Server-to-Server Notifications



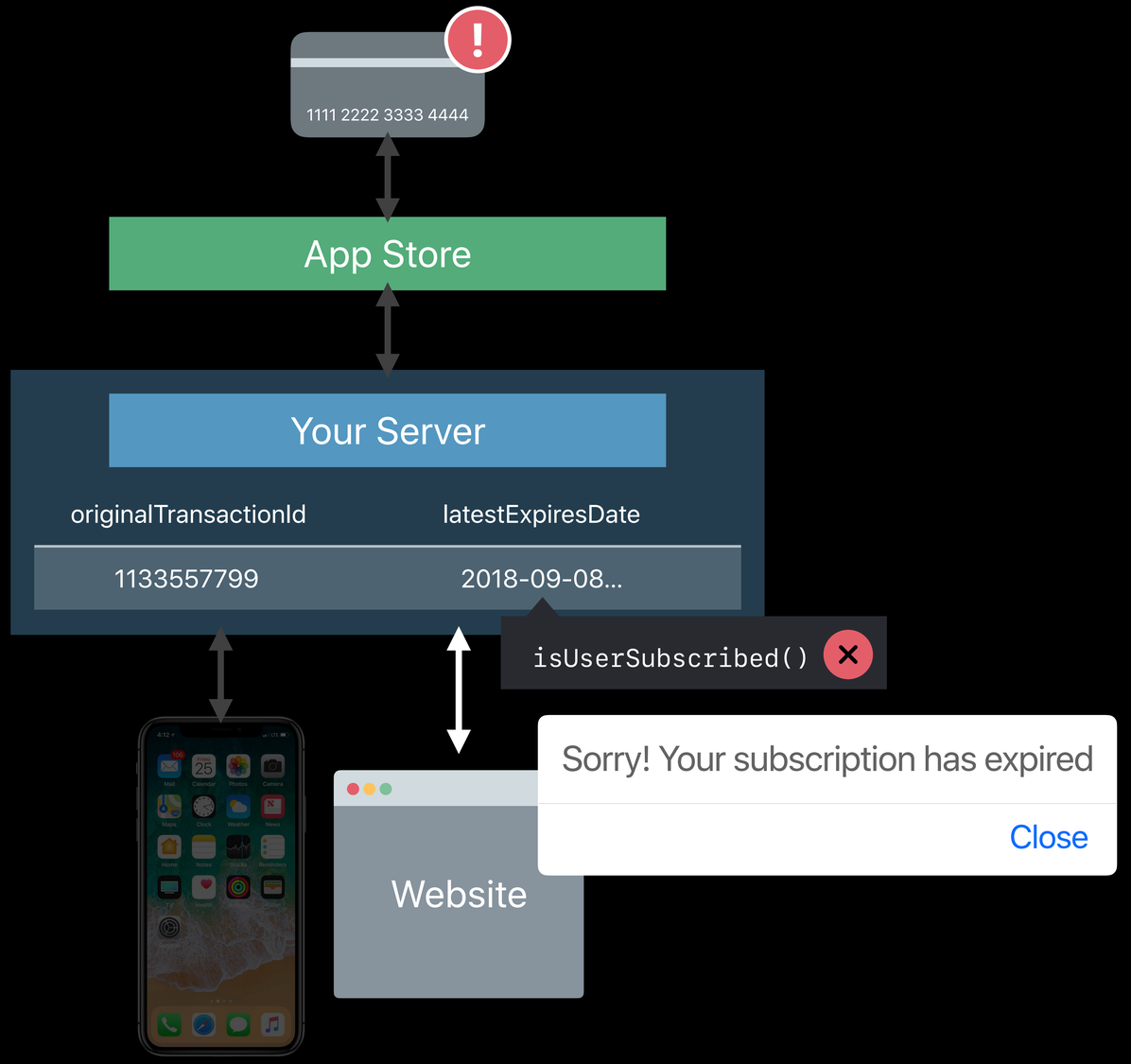
Server-to-Server Notifications



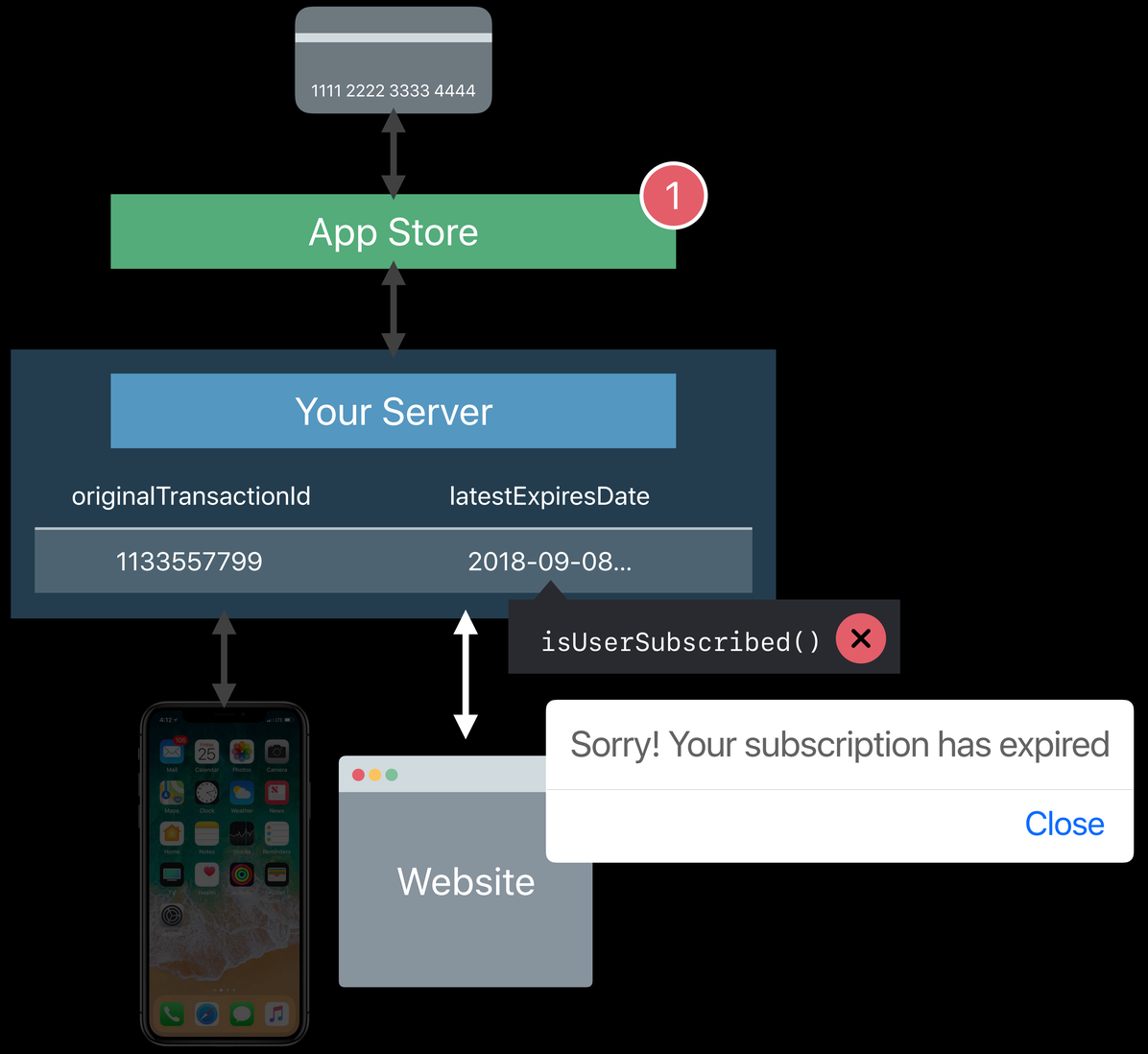
Server-to-Server Notifications



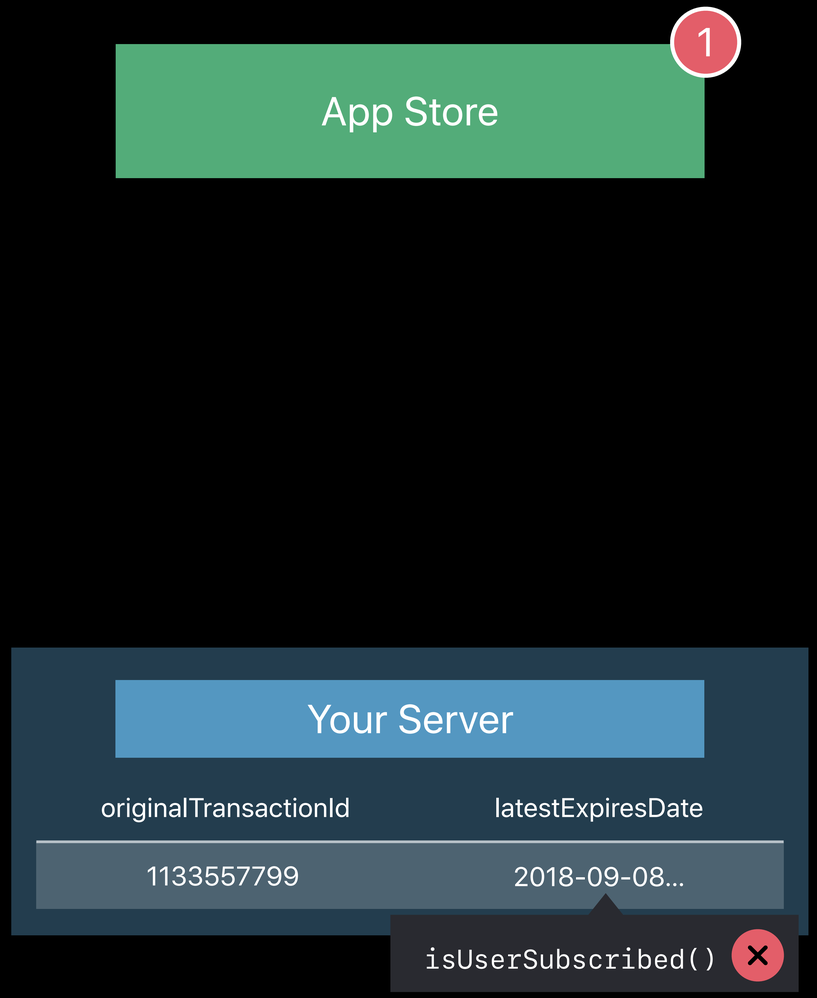
Server-to-Server Notifications



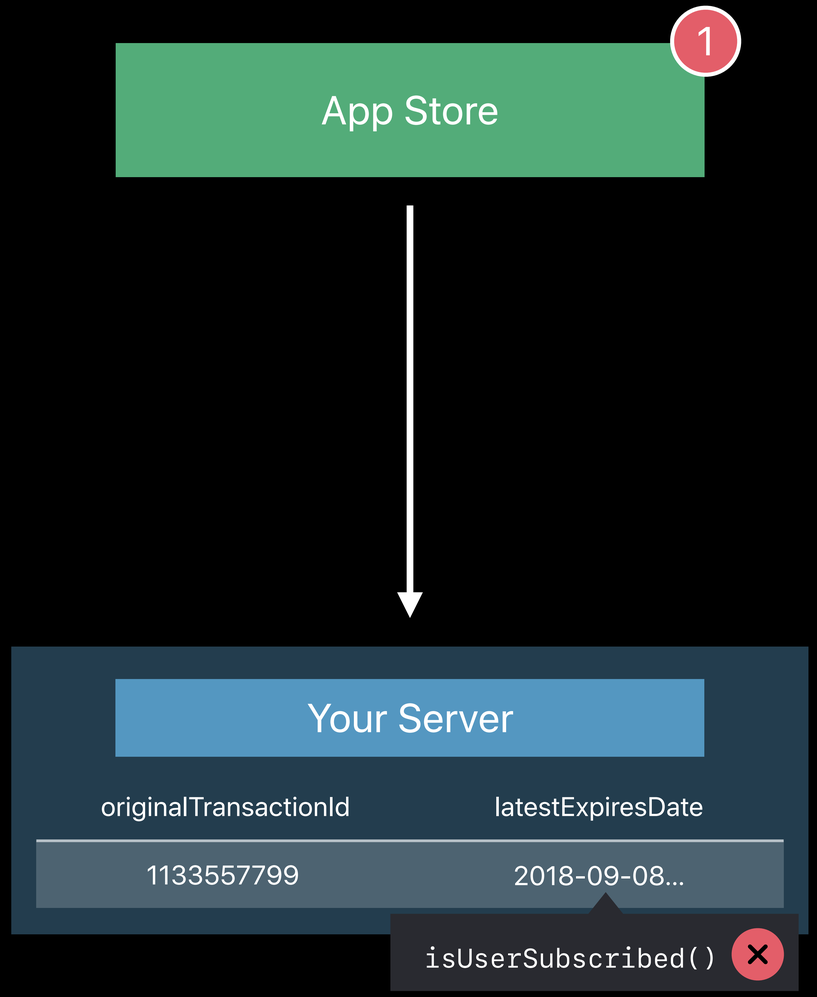
Server-to-Server Notifications



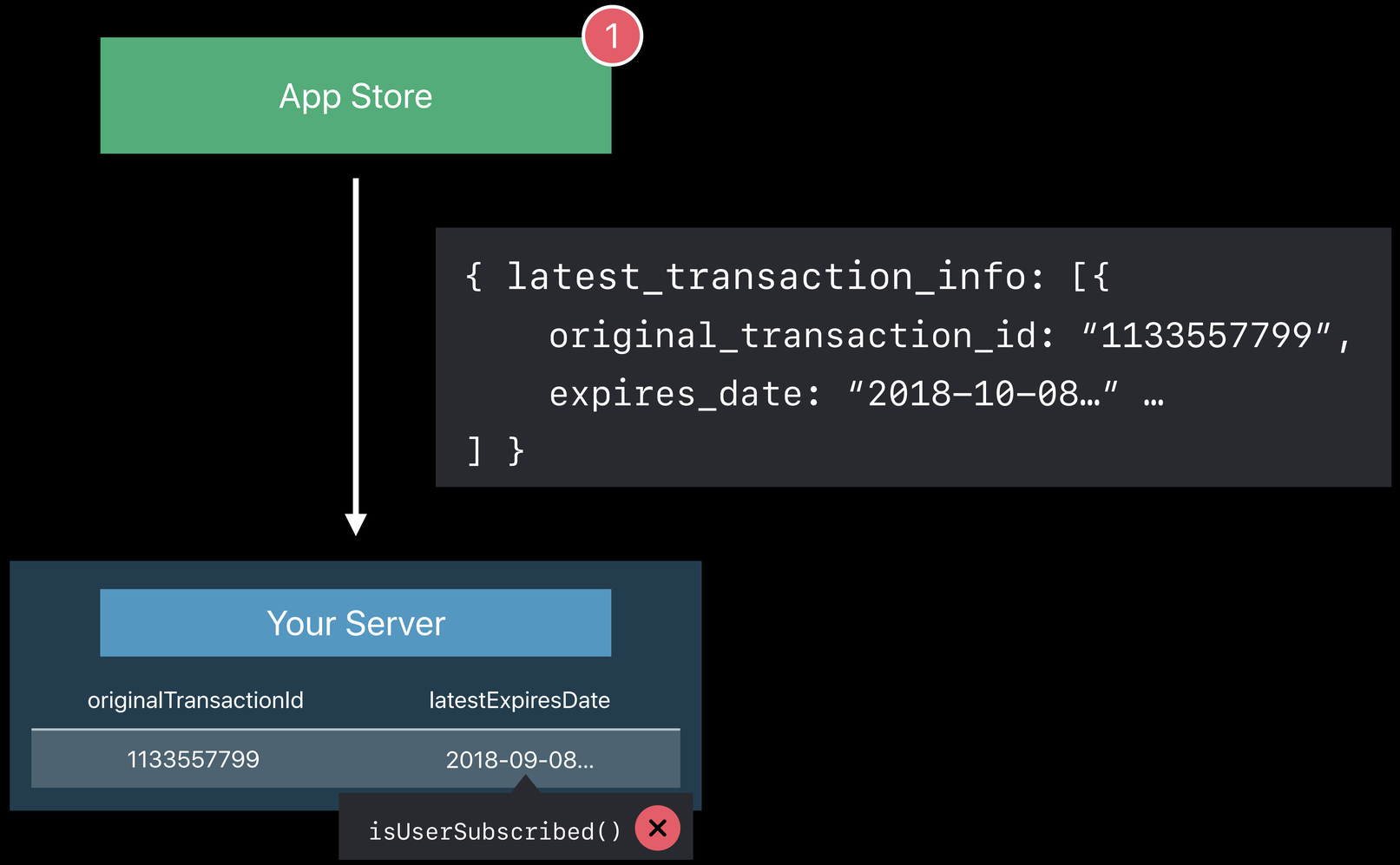
Server-to-Server Notifications



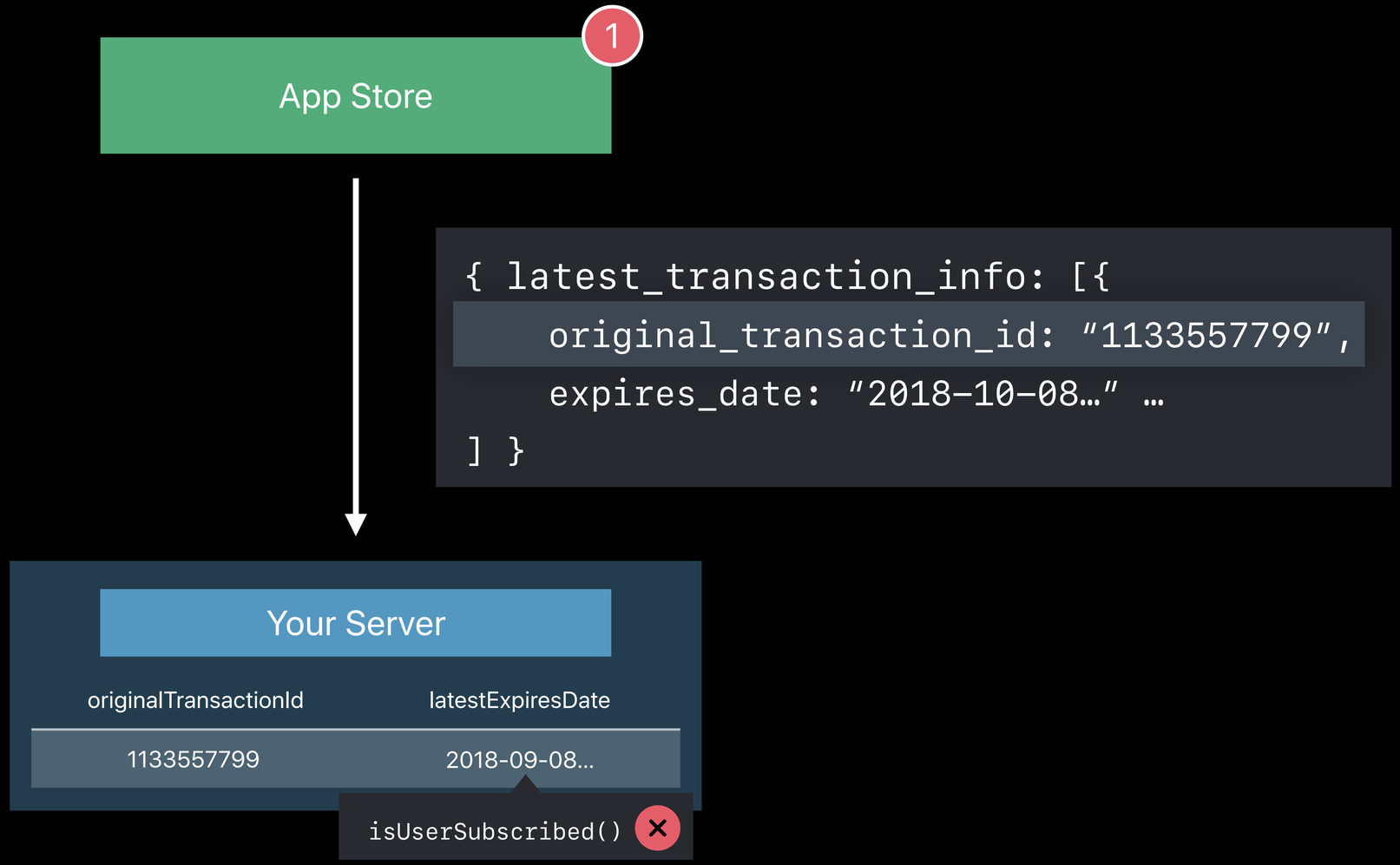
Server-to-Server Notifications



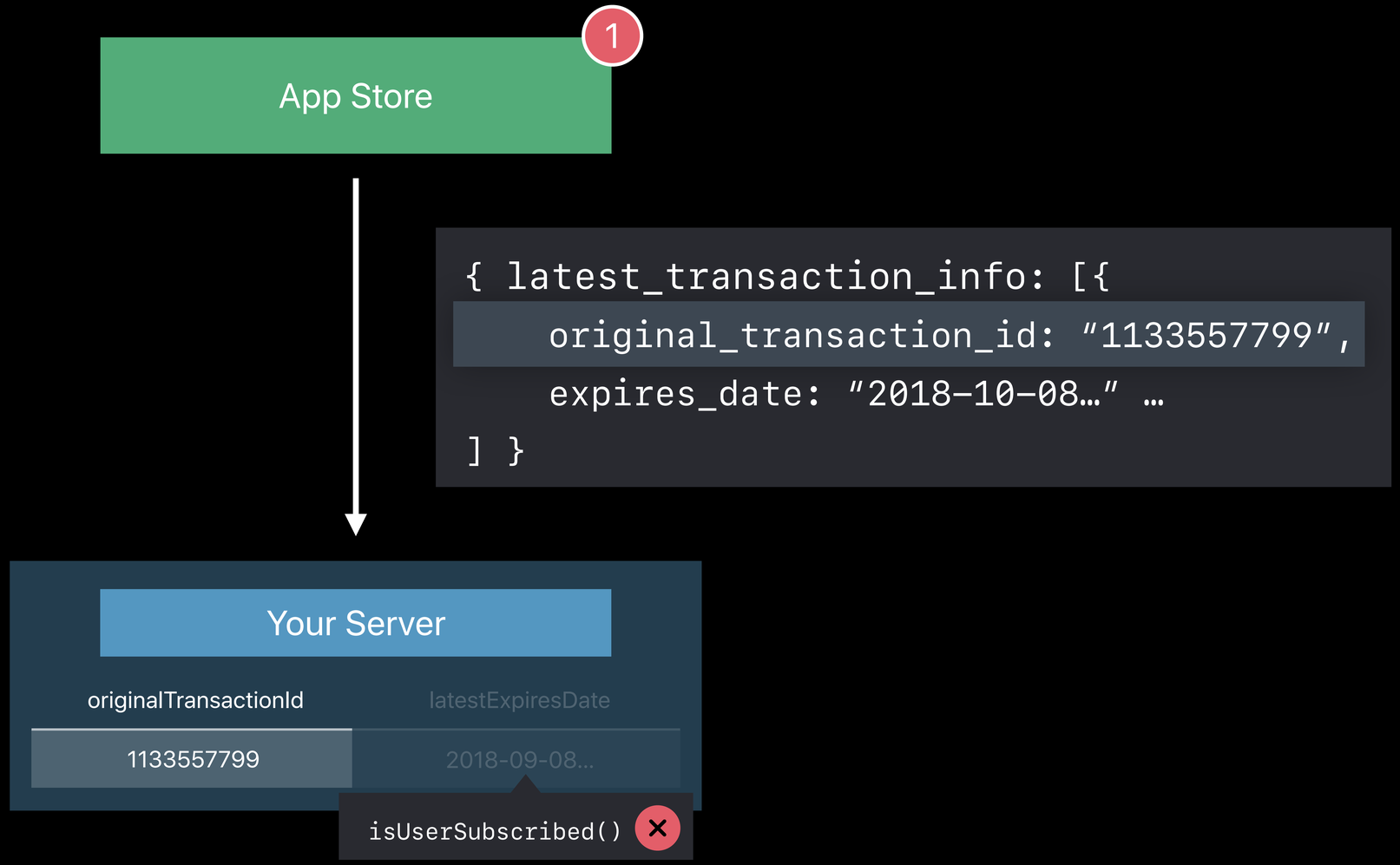
Server-to-Server Notifications



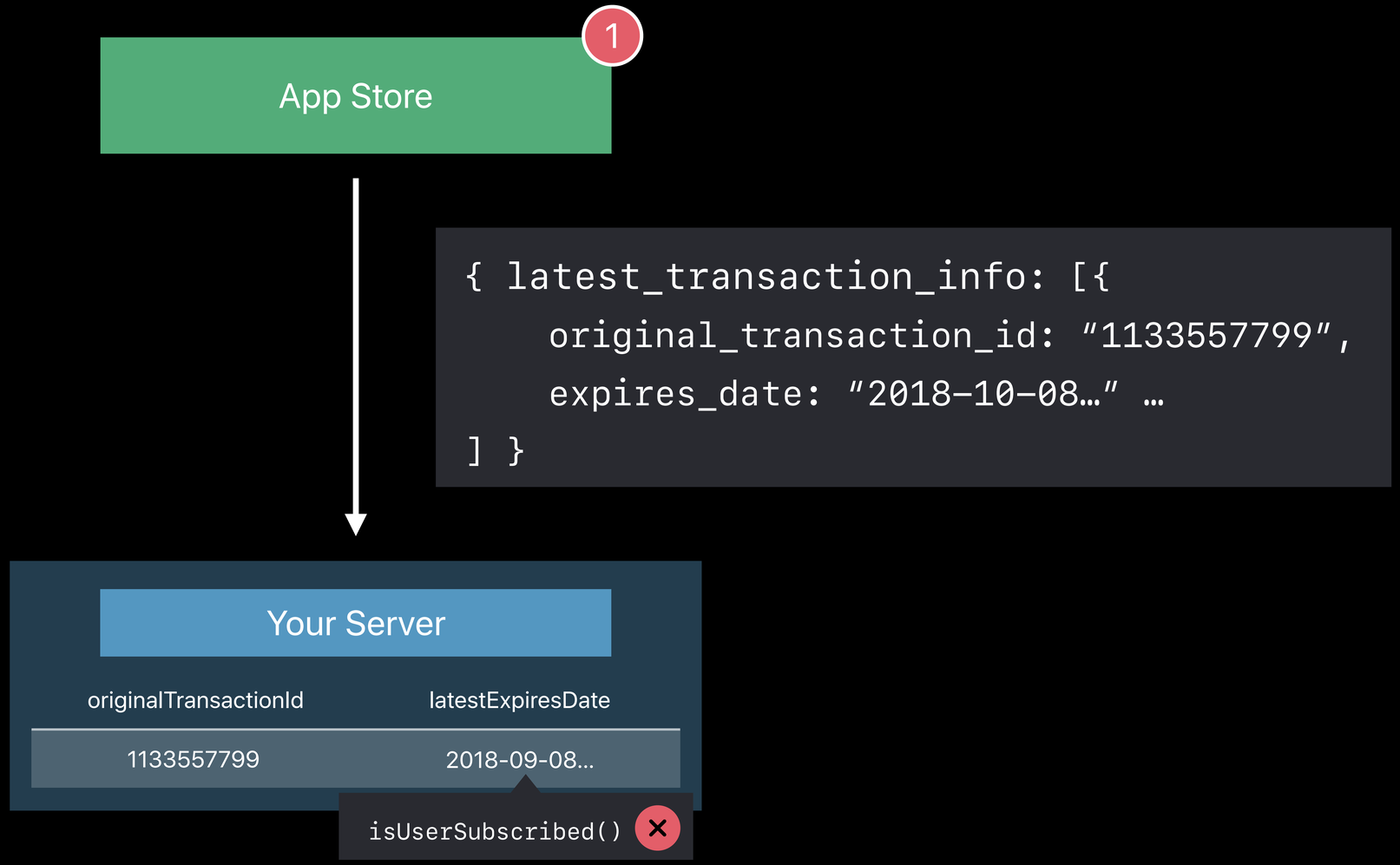
Server-to-Server Notifications



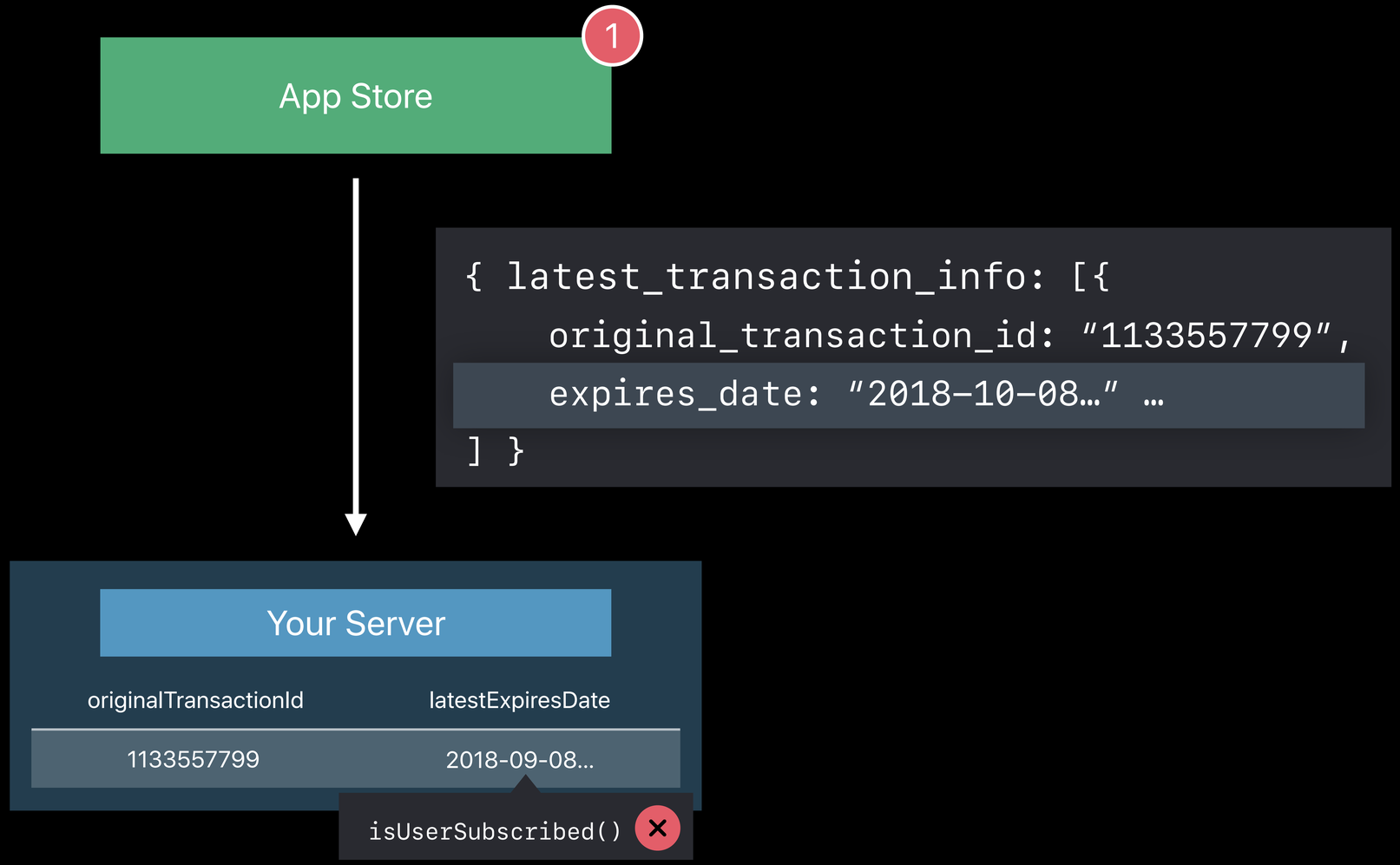
Server-to-Server Notifications



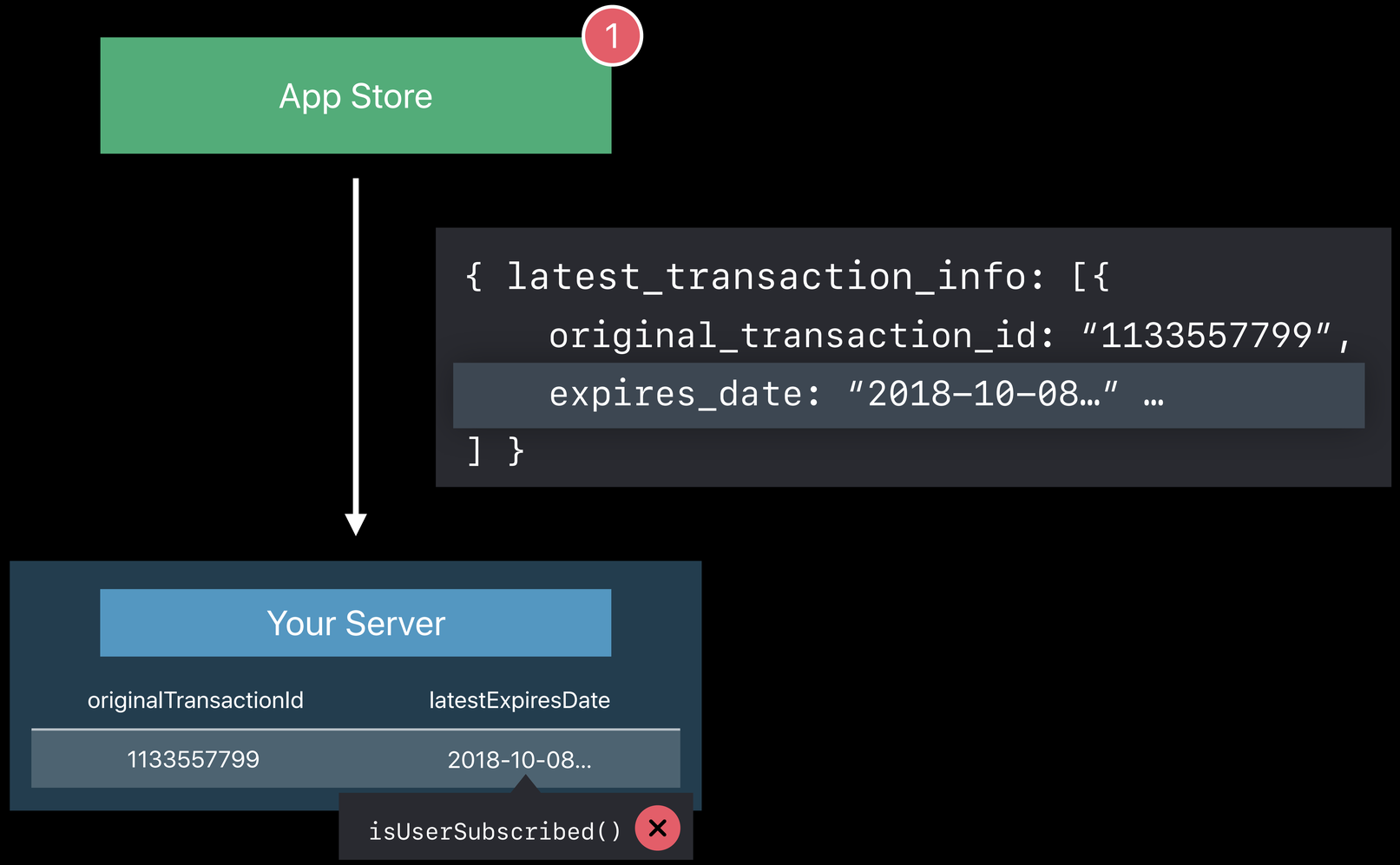
Server-to-Server Notifications



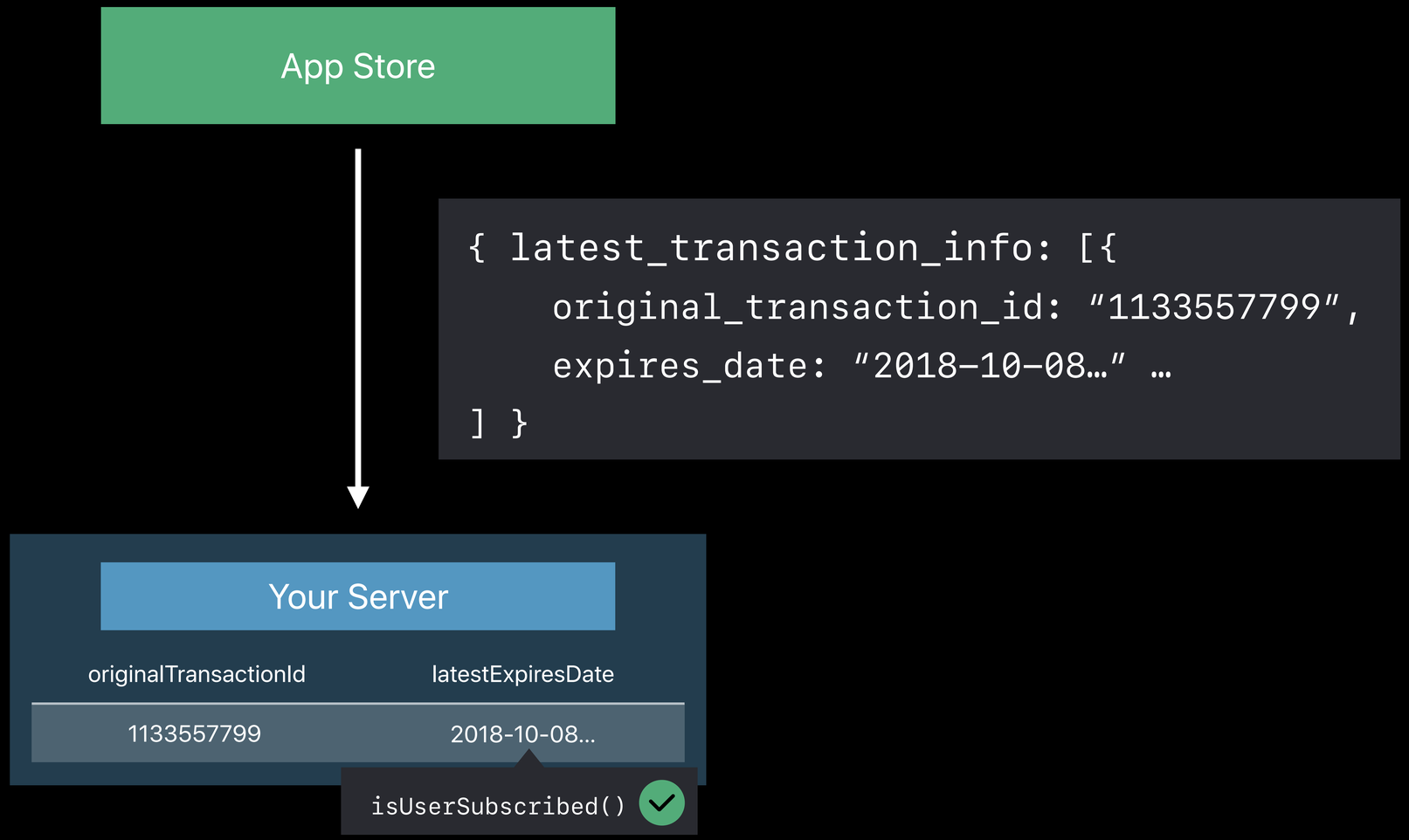
Server-to-Server Notifications



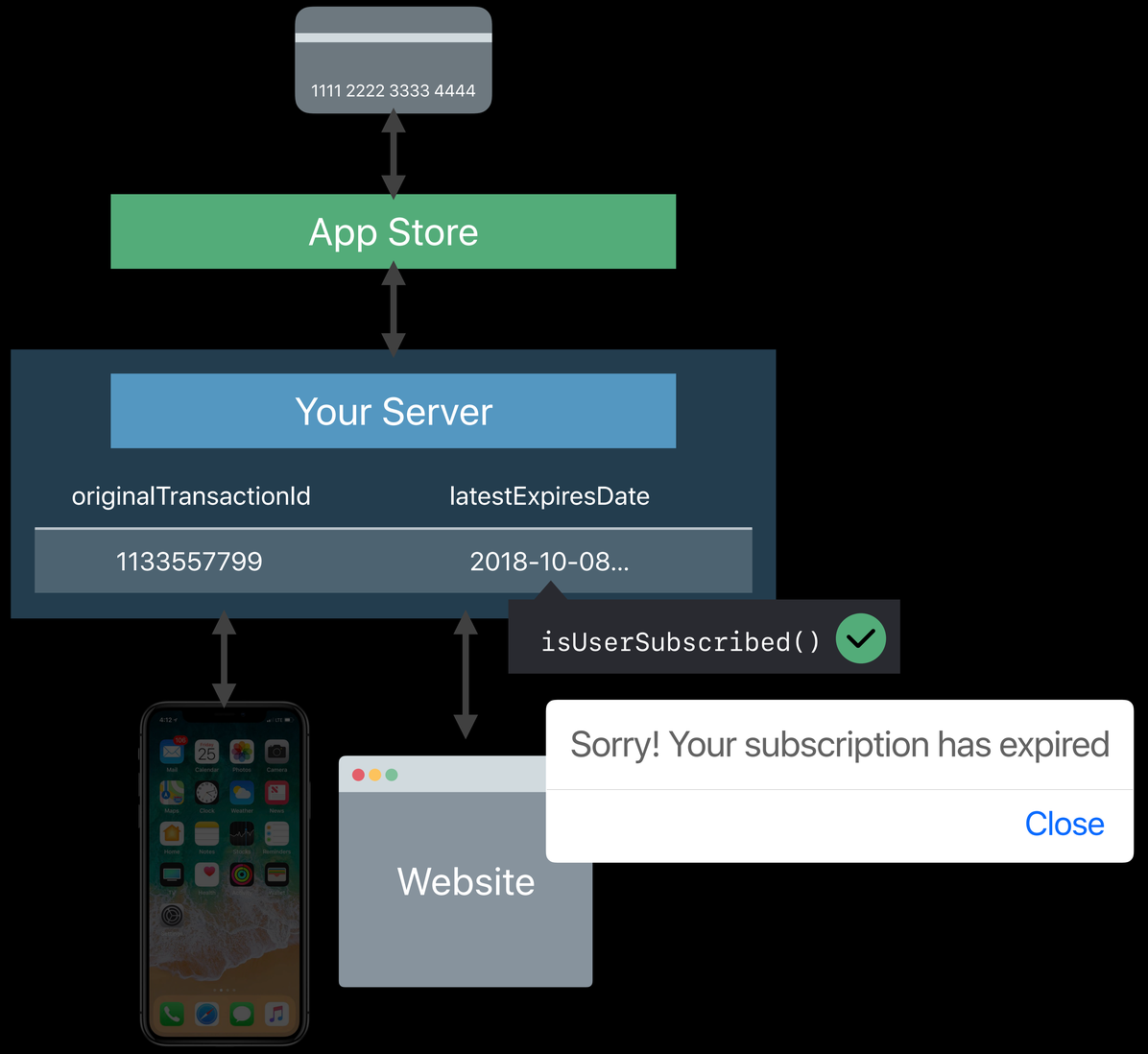
Server-to-Server Notifications



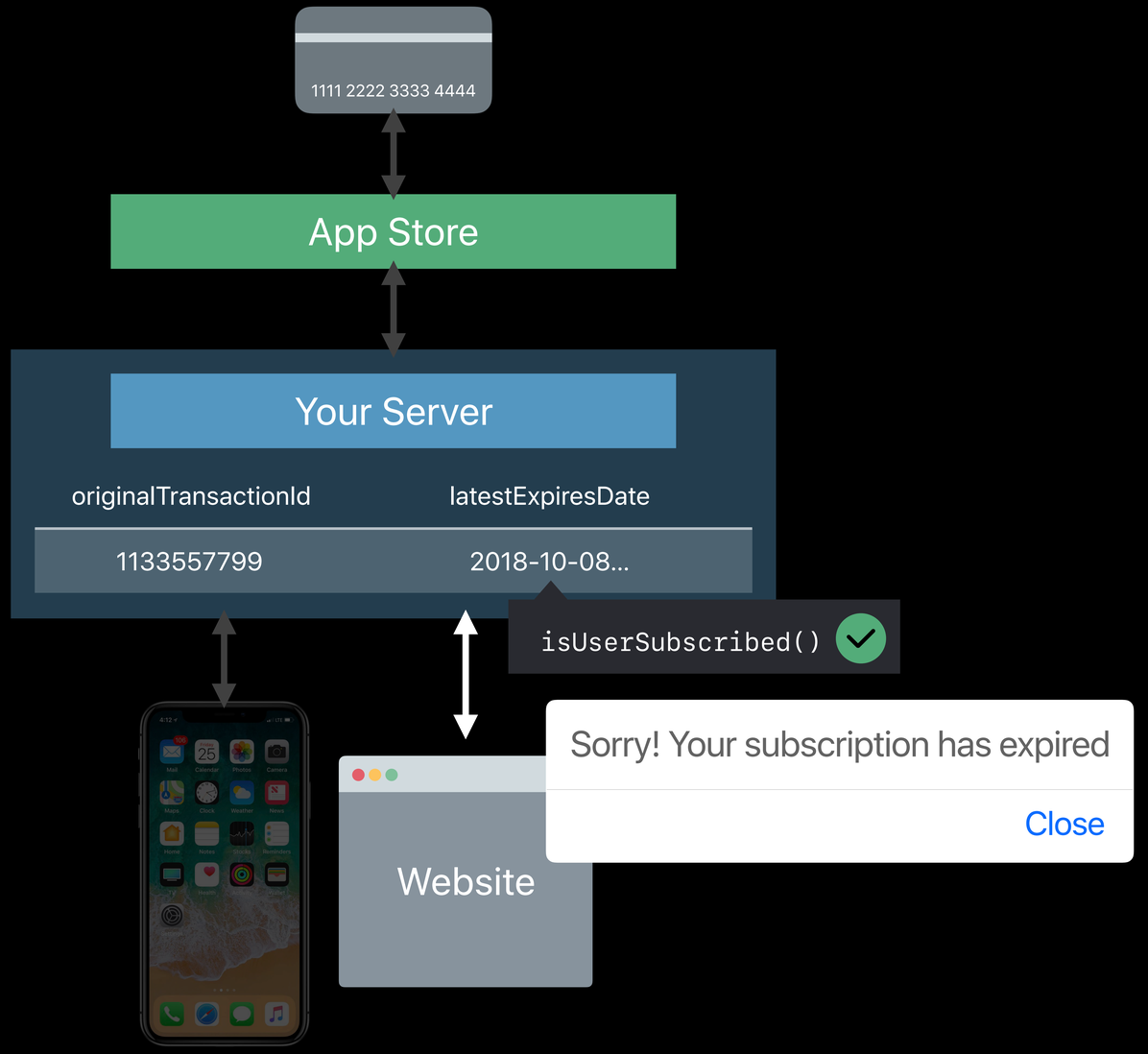
Server-to-Server Notifications



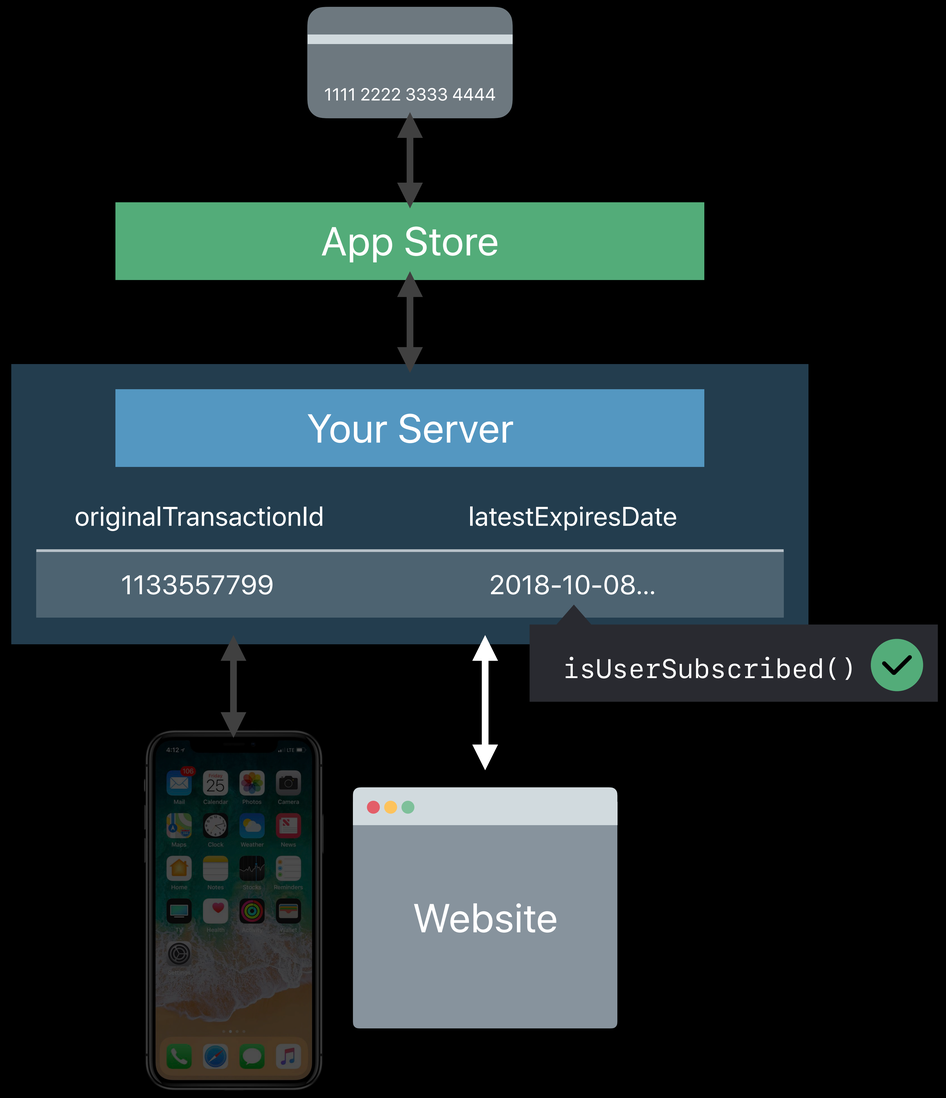
Server-to-Server Notifications



Server-to-Server Notifications



Server-to-Server Notifications



Server-to-Server Notifications

Server-to-Server Notifications

Status URL in App Store Connect

Server-to-Server Notifications

Status URL in App Store Connect

HTTPS POST to your server for status changes

Server-to-Server Notifications

Status URL in App Store Connect

HTTPS POST to your server for status changes

Includes `latest_transaction_info` for the transaction in question

In-App Experience

userId	originalTransactionId	latestExpiresDate
90000001	1133557799	2018-07-08...
90000002	4820018625	2018-06-02...
90000003	6090522426	2018-06-01...
90000004	3092890314	2018-06-20...
90000005	8426576390	2018-06-21...
90000006	4286343643	2018-06-03...

Creating User Accounts

Creating User Accounts

Offer in-app purchase before account creation

Creating User Accounts

Offer in-app purchase before account creation

- Better user experience

Creating User Accounts

Offer in-app purchase before account creation

- Better user experience
- Higher conversion

Creating User Accounts

Offer in-app purchase before account creation

- Better user experience
- Higher conversion

Use anonymous account

userId	email	originalTransactionId	latestReceiptData	latestExpiresDate
90000001	NULL	1133557799	d24Fs...kJ87dDGe3=	2018-10-08...

Creating User Accounts

Offer in-app purchase before account creation

- Better user experience
- Higher conversion

Use anonymous account

- Rely on `original_transaction_id` to associate multiple devices

userId	email	originalTransactionId	latestReceiptData	latestExpiresDate
90000001	NULL	1133557799	d24Fs...kJ87dDGe3=	2018-10-08...

Creating User Accounts

Offer in-app purchase before account creation

- Better user experience
- Higher conversion

Use anonymous account

- Rely on `original_transaction_id` to associate multiple devices
- Optionally take user through account “creation” later

userId	email	originalTransactionId	latestReceiptData	latestExpiresDate
90000001	NULL	1133557799	d24Fs...kJ87dDGe3=	2018-10-08...

Creating User Accounts

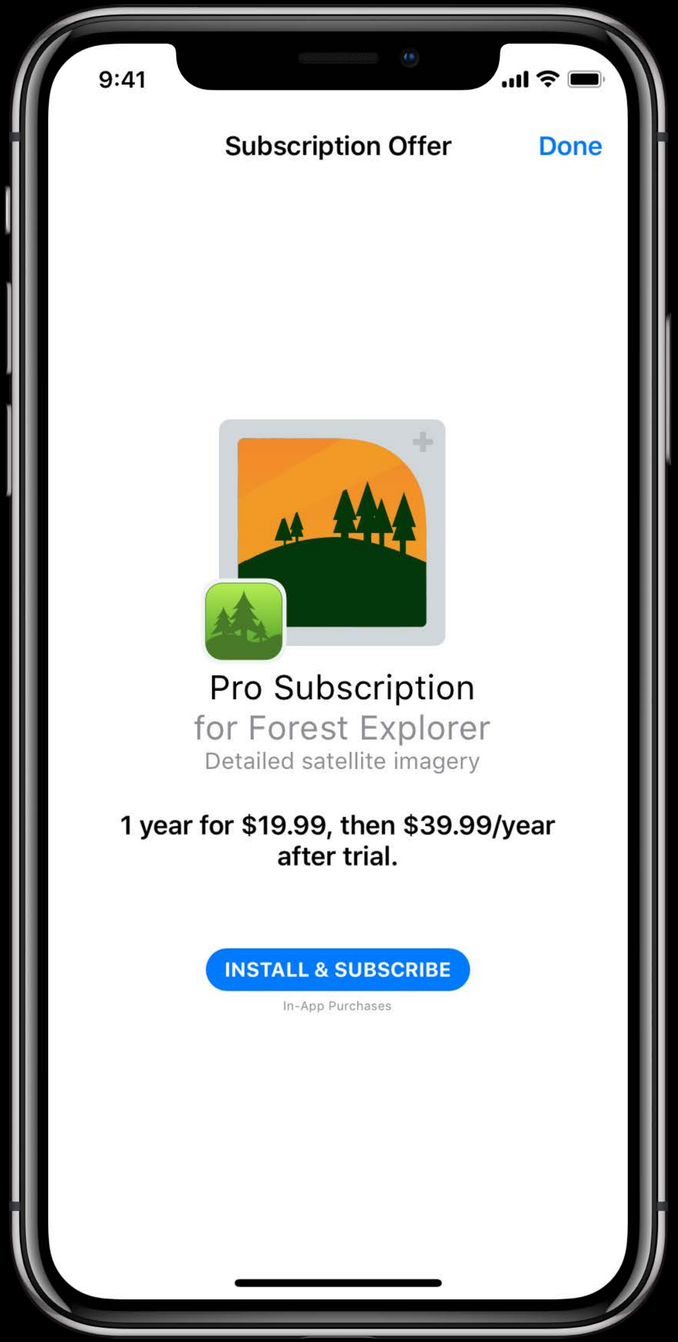
Offer in-app purchase before account creation

- Better user experience
- Higher conversion

Use anonymous account

- Rely on `original_transaction_id` to associate multiple devices
- Optionally take user through account “creation” later

userId	email	originalTransactionId	latestReceiptData	latestExpiresDate
90000001	johnappleseed@...	1133557799	d24Fs...kJ87dDGe3=	2018-10-08...


9:41
Subscription Offer
Done
Pro Subscription
for Forest Explorer
Detailed satellite imagery
1 year for $19.99, then $39.99/year after trial.
INSTALL & SUBSCRIBE
In-App Purchases

Introductory Pricing

Determining user eligibility

Introductory Pricing

Determining user eligibility

Need to show correct price to your user

Introductory Pricing

Determining user eligibility

Need to show correct price to your user

Check transactions to see if discount has been used

Introductory Pricing

Determining user eligibility

Need to show correct price to your user

Check transactions to see if discount has been used

```
{… transaction_id: “9988776655”,
  product_id: “com.your.product.id”,
  is_trial_period: “false”,
  is_in_intro_offer_period: “true” },
```

Introductory Pricing

Determining user eligibility

Need to show correct price to your user

Check transactions to see if discount has been used

```
{… transaction_id: “9988776655”,
  product_id: “com.your.product.id”,
  is_trial_period: “false”,
  is_in_intro_offer_period: “true” },
```

Introductory Pricing

Determining user eligibility

Need to show correct price to your user

Check transactions to see if discount has been used

```
{… transaction_id: “9988776655”,
  product_id: “com.your.product.id”,
  is_trial_period: “false”,
  is_in_intro_offer_period: “true” },
```

userId	originalTransactionId	latestReceiptData	latestExpiresDate	consumedProductDiscounts
90000001	1133557799	d24Fs…kJ87dDGe3=	2018-10-08…	com.your.product.id

```
// Displaying the correct product price for “Product A”
```

NEW

```
func fetchConsumedGroupDiscounts() {
  let consumedProducts = currentUser.consumedProductDiscounts
  let productsRequest = SKProductsRequest(productIdentifiers: Set(consumedProducts))
  productsRequest.delegate = self
  productsRequest.start()
}

// MARK: - SKProductsRequestDelegate

func productsRequest(_ request: SKProductsRequest, didReceive response: SKProductsResponse) {
  for product in response.products {
    currentUser.consumedGroupDiscounts.append(product.subscriptionGroupIdentifier)
  }
}
```

```
// Displaying the correct product price for “Product A”
```

NEW

```
func fetchConsumedGroupDiscounts() {
  let consumedProducts = currentUser.consumedProductDiscounts
  let productsRequest = SKProductsRequest(productIdentifiers: Set(consumedProducts))
  productsRequest.delegate = self
  productsRequest.start()
}

// MARK: - SKProductsRequestDelegate

func productsRequest(_ request: SKProductsRequest, didReceive response: SKProductsResponse) {
  for product in response.products {
    currentUser.consumedGroupDiscounts.append(product.subscriptionGroupIdentifier)
  }
}
```

```
// Displaying the correct product price for “Product A”
```

NEW

```
func fetchConsumedGroupDiscounts() {
  let consumedProducts = currentUser.consumedProductDiscounts
  let productsRequest = SKProductsRequest(productIdentifiers: Set(consumedProducts))
  productsRequest.delegate = self
  productsRequest.start()
}

// MARK: - SKProductsRequestDelegate

func productsRequest(_ request: SKProductsRequest, didReceive response: SKProductsResponse) {
  for product in response.products {
    currentUser.consumedGroupDiscounts.append(product.subscriptionGroupIdentifier)
  }
}
```

```
// Displaying the correct product price for “Product A”
```

NEW

```
func fetchConsumedGroupDiscounts() {
  let consumedProducts = currentUser.consumedProductDiscounts
  let productsRequest = SKProductsRequest(productIdentifiers: Set(consumedProducts))
  productsRequest.delegate = self
  productsRequest.start()
}

// MARK: - SKProductsRequestDelegate

func productsRequest(_ request: SKProductsRequest, didReceive response: SKProductsResponse) {
  for product in response.products {
    currentUser.consumedGroupDiscounts.append(product.subscriptionGroupIdentifier)
  }
}
```

```
// Displaying the correct product price for “Product A”
```

NEW

```
func fetchConsumedGroupDiscounts() {
  let consumedProducts = currentUser.consumedProductDiscounts
  let productsRequest = SKProductsRequest(productIdentifiers: Set(consumedProducts))
  productsRequest.delegate = self
  productsRequest.start()
}

// MARK: - SKProductsRequestDelegate

func productsRequest(_ request: SKProductsRequest, didReceive response: SKProductsResponse) {
  for product in response.products {
    currentUser.consumedGroupDiscounts.append(product.subscriptionGroupIdentifier)
  }
}
```

```
// Displaying the correct product price for “Product A”
```

NEW

```
func fetchConsumedGroupDiscounts() {
  let consumedProducts = currentUser.consumedProductDiscounts
  let productsRequest = SKProductsRequest(productIdentifiers: Set(consumedProducts))
  productsRequest.delegate = self
  productsRequest.start()
}

// MARK: - SKProductsRequestDelegate

func productsRequest(_ request: SKProductsRequest, didReceive response: SKProductsResponse) {
  for product in response.products {
    currentUser.consumedGroupDiscounts.append(product.subscriptionGroupIdentifier)
  }
}
```

```
// Displaying the correct product price for “Product A”

  let price: NSDecimalNumber
  let priceLocale: Locale
  if currentUser.consumedGroupDiscounts.contains(productA.subscriptionGroupIdentifier) {
    price = productA.price
    priceLocale = productA.priceLocale
  } else {
    price = productA.introductoryPrice.price
    priceLocale = productA.introductoryPrice.priceLocale
  }

  let formatter = NumberFormatter()
  formatter.numberStyle = .currency
  formatter.locale = priceLocale
  let formattedString = formatter.string(from: price)
```

```
// Displaying the correct product price for “Product A”

  let price: NSDecimalNumber
  let priceLocale: Locale
  if currentUser.consumedGroupDiscounts.contains(productA.subscriptionGroupIdentifier) {
    price = productA.price
    priceLocale = productA.priceLocale
  } else {
    price = productA.introductoryPrice.price
    priceLocale = productA.introductoryPrice.priceLocale
  }

  let formatter = NumberFormatter()
  formatter.numberStyle = .currency
  formatter.locale = priceLocale
  let formattedString = formatter.string(from: price)
```

```
// Displaying the correct product price for “Product A”

  let price: NSDecimalNumber
  let priceLocale: Locale
  if currentUser.consumedGroupDiscounts.contains(productA.subscriptionGroupIdentifier) {
    price = productA.price
    priceLocale = productA.priceLocale
  } else {
    price = productA.introductoryPrice.price
    priceLocale = productA.introductoryPrice.priceLocale
  }

  let formatter = NumberFormatter()
  formatter.numberStyle = .currency
  formatter.locale = priceLocale
  let formattedString = formatter.string(from: price)
```

```
// Displaying the correct product price for “Product A”

  let price: NSDecimalNumber
  let priceLocale: Locale
  if currentUser.consumedGroupDiscounts.contains(productA.subscriptionGroupIdentifier) {
    price = productA.price
    priceLocale = productA.priceLocale
  } else {
    price = productA.introductoryPrice.price
    priceLocale = productA.introductoryPrice.priceLocale
  }

  let formatter = NumberFormatter()
  formatter.numberStyle = .currency
  formatter.locale = priceLocale
  let formattedString = formatter.string(from: price)
```

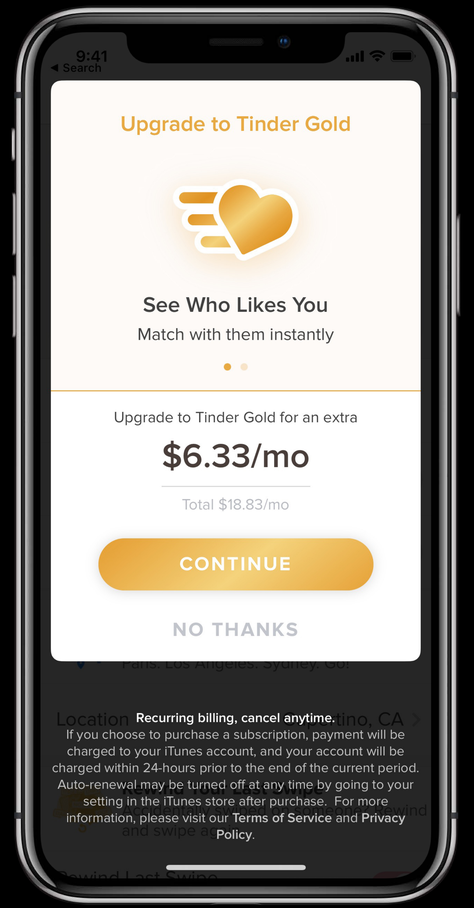
```
// Displaying the correct product price for “Product A”

  let price: NSDecimalNumber
  let priceLocale: Locale
  if currentUser.consumedGroupDiscounts.contains(productA.subscriptionGroupIdentifier) {
    price = productA.price
    priceLocale = productA.priceLocale
  } else {
    price = productA.introductoryPrice.price
    priceLocale = productA.introductoryPrice.priceLocale
  }

  let formatter = NumberFormatter()
  formatter.numberStyle = .currency
  formatter.locale = priceLocale
  let formattedString = formatter.string(from: price)
```

Introductory Pricing

Best Practices and What’s New with In-App Purchases	Hall 1	Tuesday 2:00PM
What’s New in App Store Connect	Hall 3	Wednesday 5:00PM

Upgrade/Downgrade Subscriptions

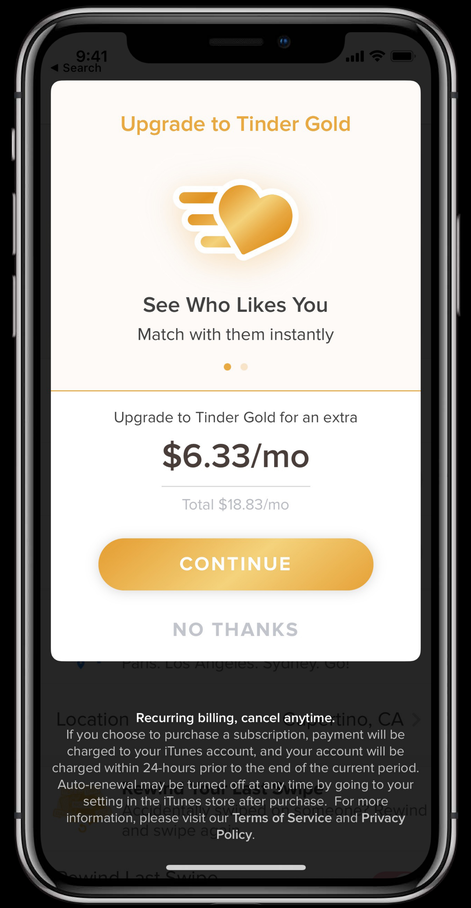
Inside your app



Upgrade/Downgrade Subscriptions

Inside your app

Just like selling an initial subscription

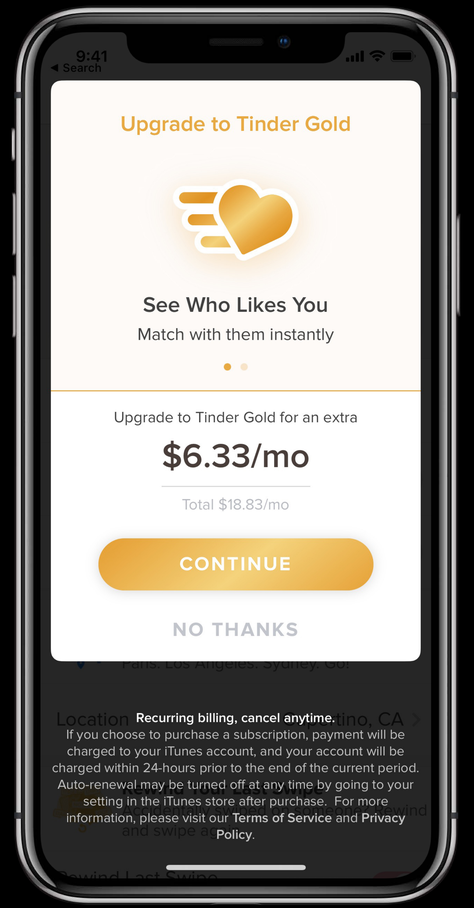


Upgrade/Downgrade Subscriptions

Inside your app

Just like selling an initial subscription

Must be a subscription in the same subscription group



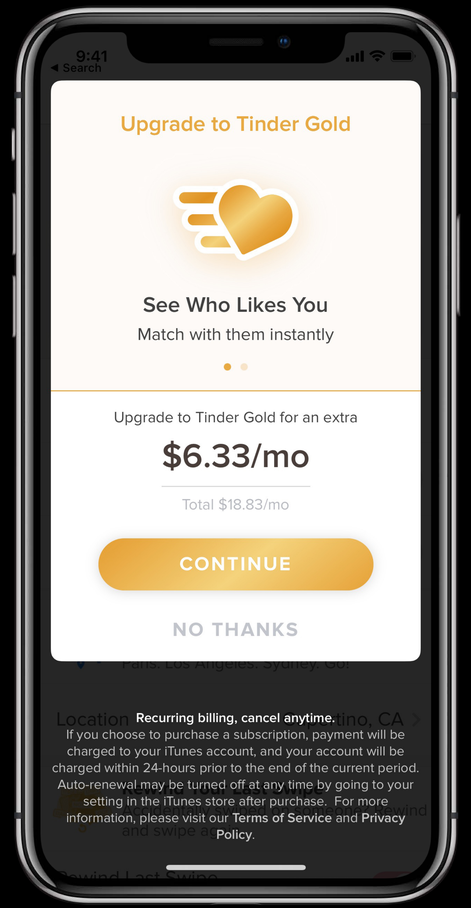
Upgrade/Downgrade Subscriptions

Inside your app

Just like selling an initial subscription

Must be a subscription in the same subscription group

```
let payment = SKPayment(product: product)
SKPaymentQueue.default().add(payment)
```



Upgrade/Downgrade Subscriptions

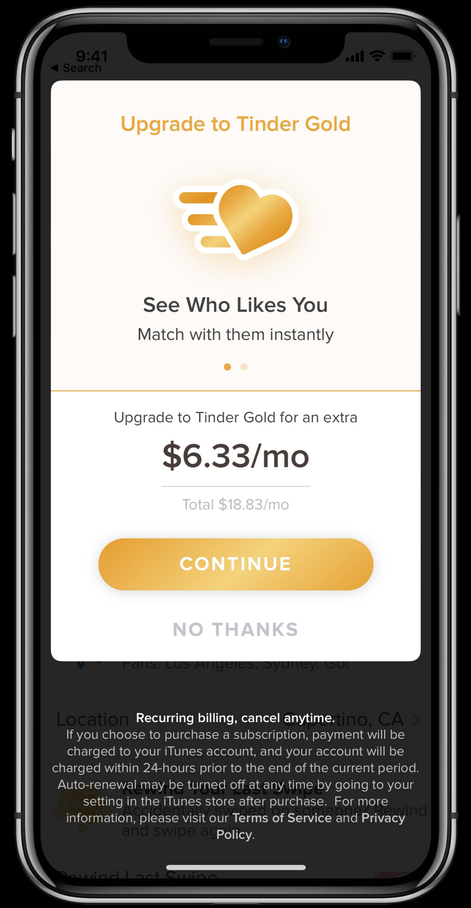
Inside your app

Just like selling an initial subscription

Must be a subscription in the same subscription group

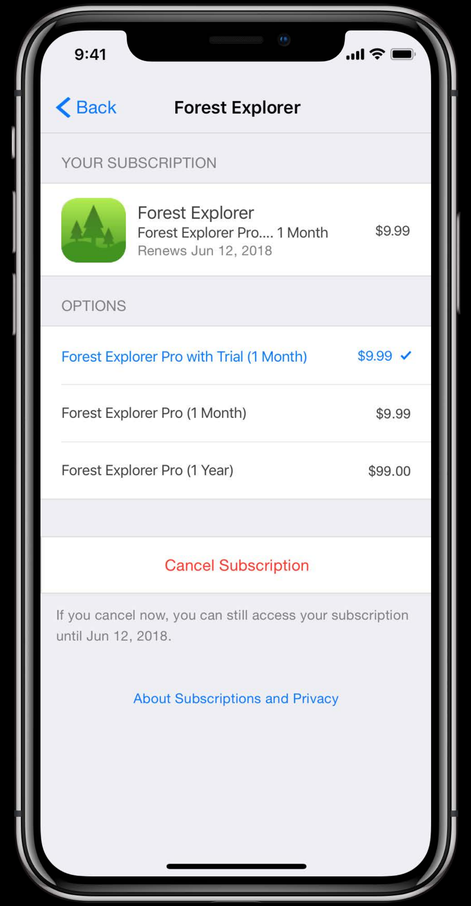
```
let payment = SKPayment(product: product)
SKPaymentQueue.default().add(payment)
```

StoreKit handles the upgrade/downgrade



Subscription Management

In App Store account settings

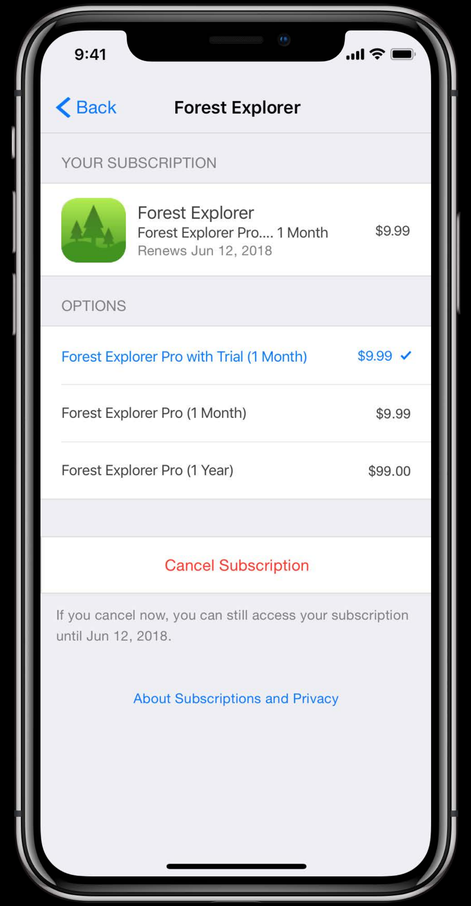


Subscription Management

In App Store account settings

Provide a link to “Manage Your Subscription”

- Upgrades/downgrades
- Cancellation



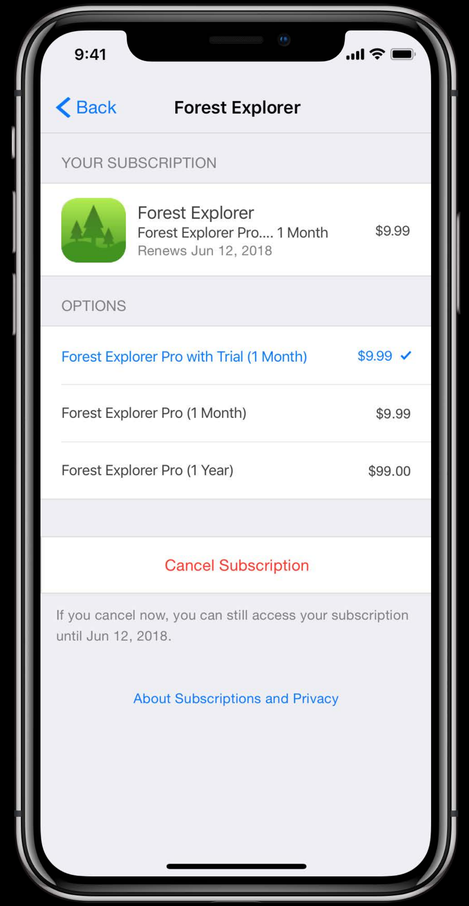
Subscription Management

In App Store account settings

Provide a link to “Manage Your Subscription”

- Upgrades/downgrades
- Cancellation

Link on the In-App Purchase Programming Guide



Subscription Management

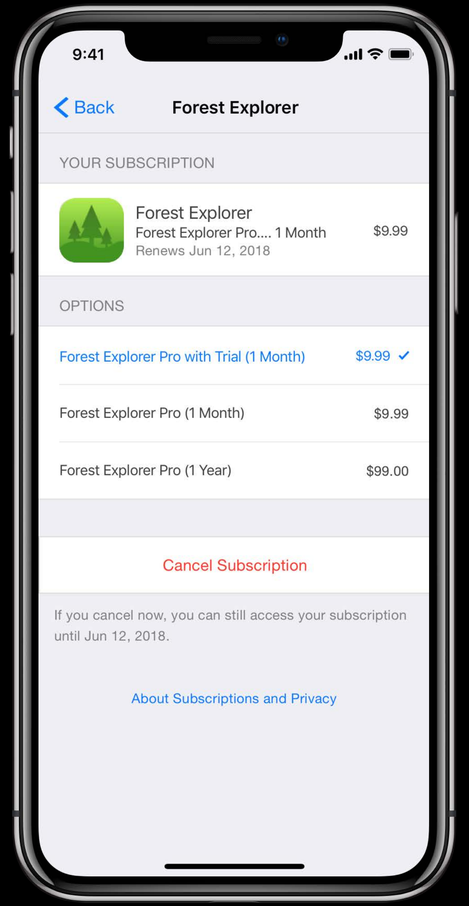
In App Store account settings

Provide a link to “Manage Your Subscription”

- Upgrades/downgrades
- Cancellation

Link on the In-App Purchase Programming Guide

https://apps.apple.com/account/subscriptions



Reducing Subscriber Churn

Michael Gargas, Technical Advocate, App Store Operations

Reducing Subscriber Churn

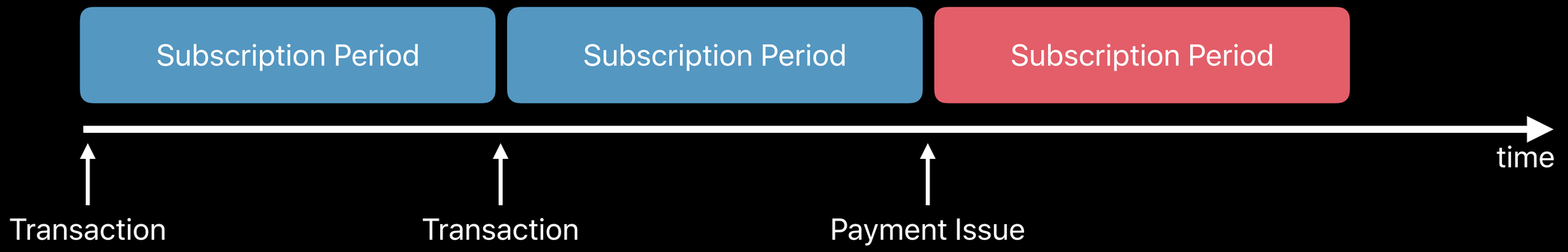
Involuntary churn

Voluntary churn

Winback

Involuntary Churn

Involuntary Churn



Involuntary Churn

Billing retry

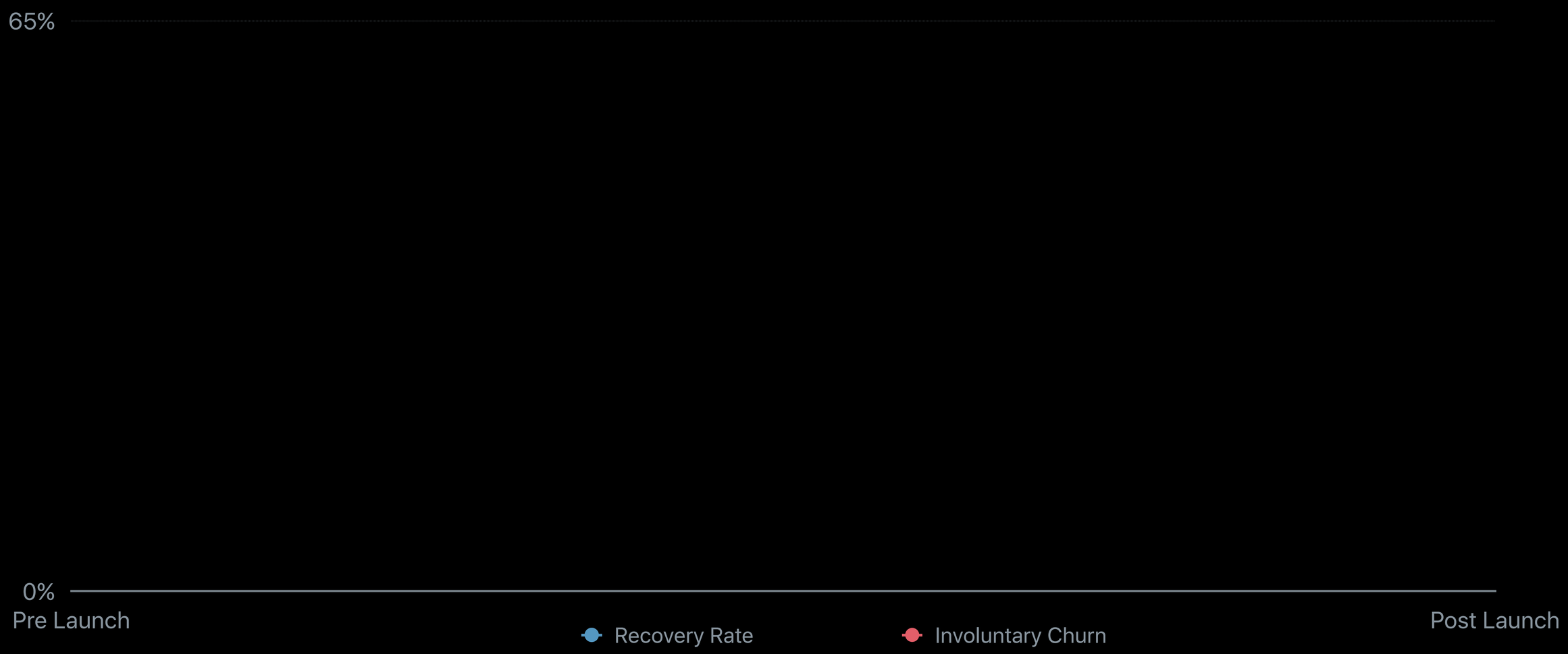
Updated failed payment logic

- Expanded retry duration
- New retry strategies
- Tuning

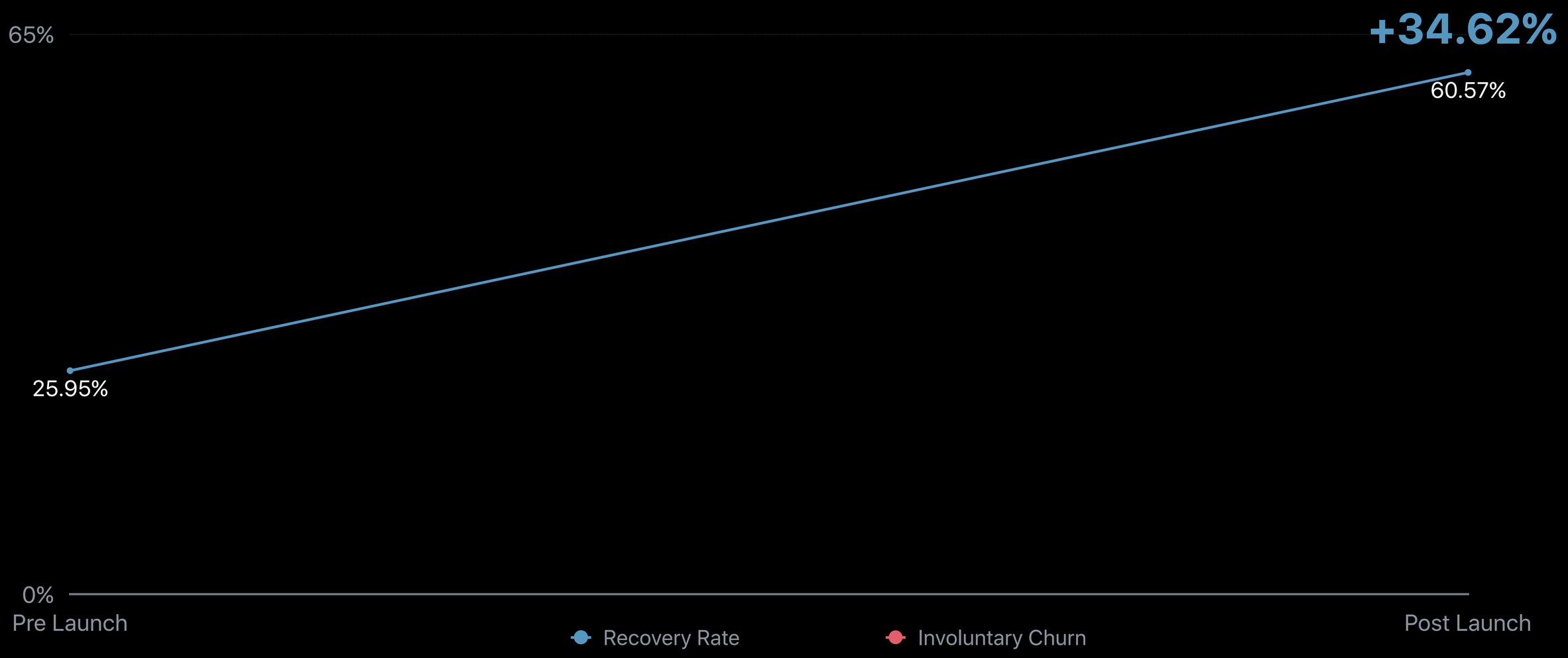
July 13, 2017

After Billing Retry

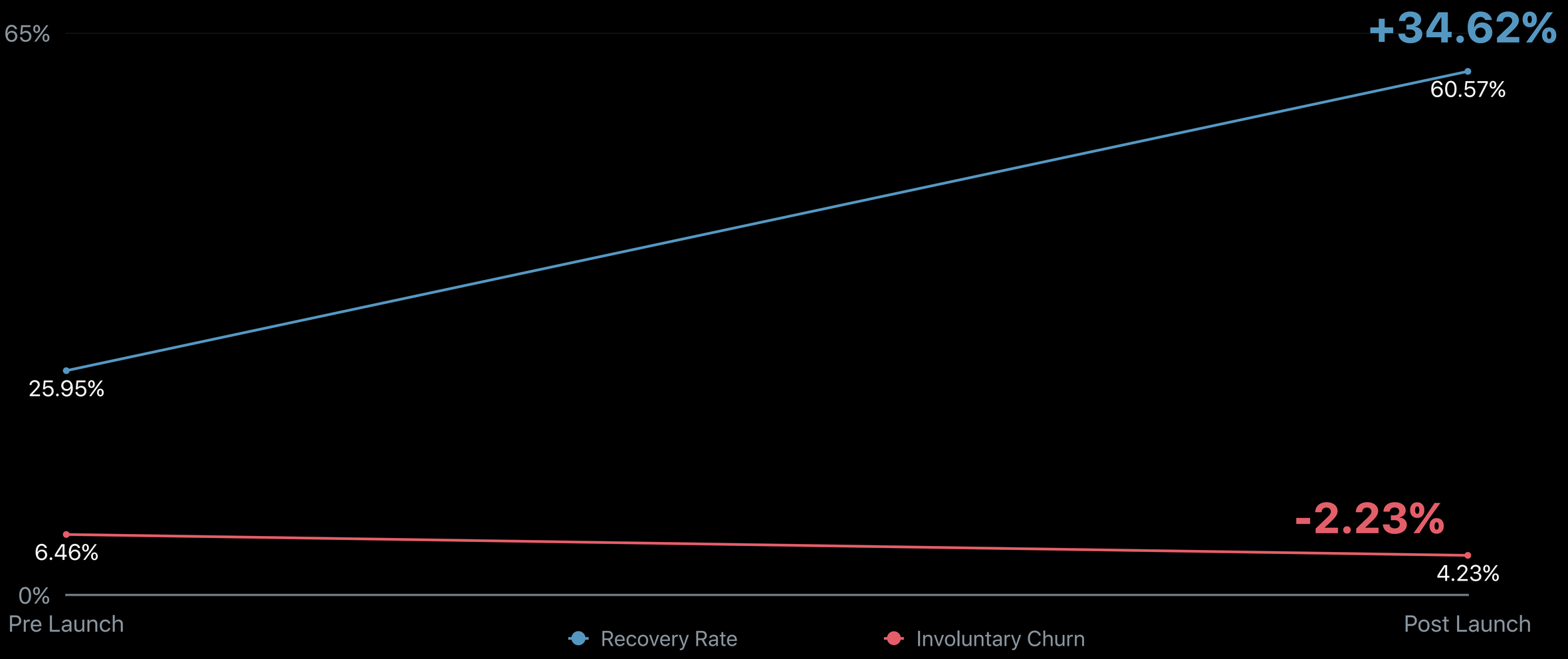
Performance



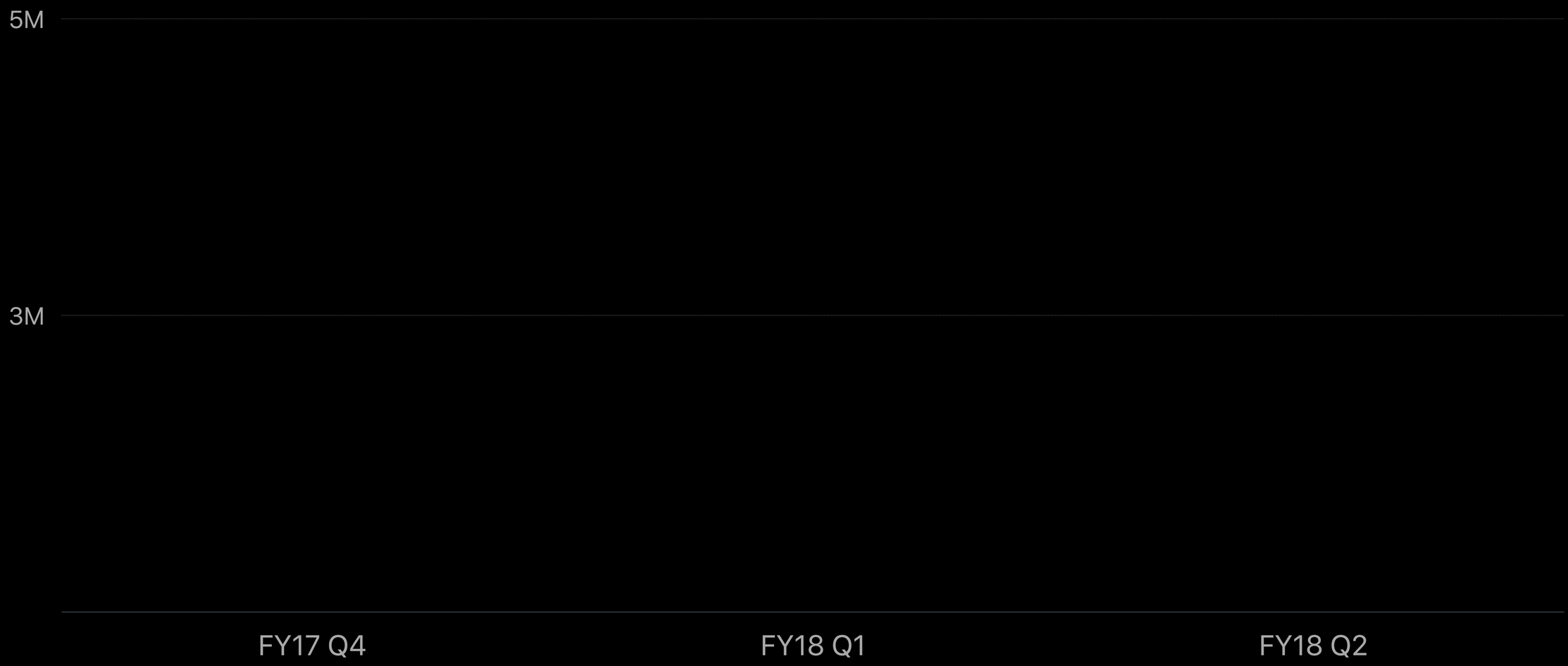
Performance



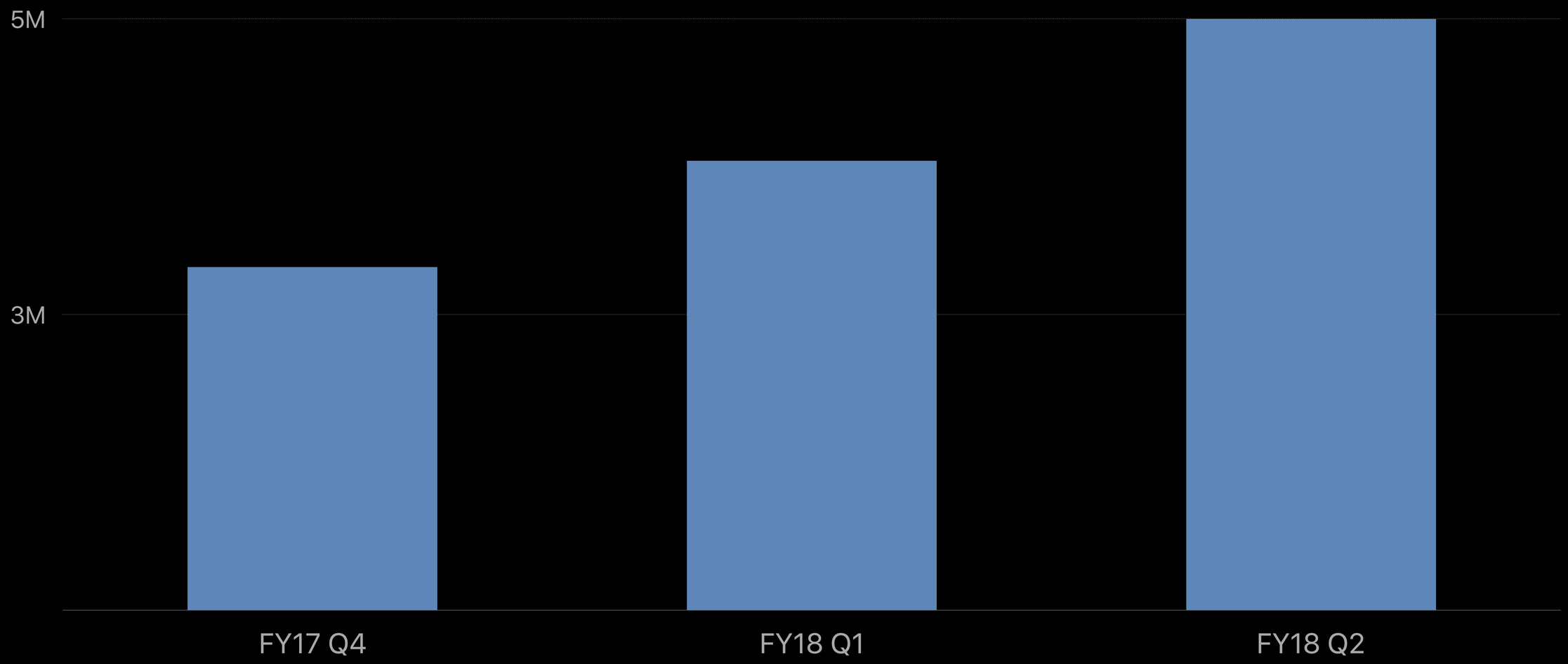
Performance



Incremental Subscriptions Recovered



Incremental Subscriptions Recovered



12 Million

Subscriptions Recovered

Minimizing Involuntary Churn

Leverage subscription receipt fields

Implement a grace period

Customer messaging

Example Subscription


```
in_app: [{
  “purchase_date”: “2018-03-26…”,
  “expires_date”: “2018-04-26…”,…
}]
```

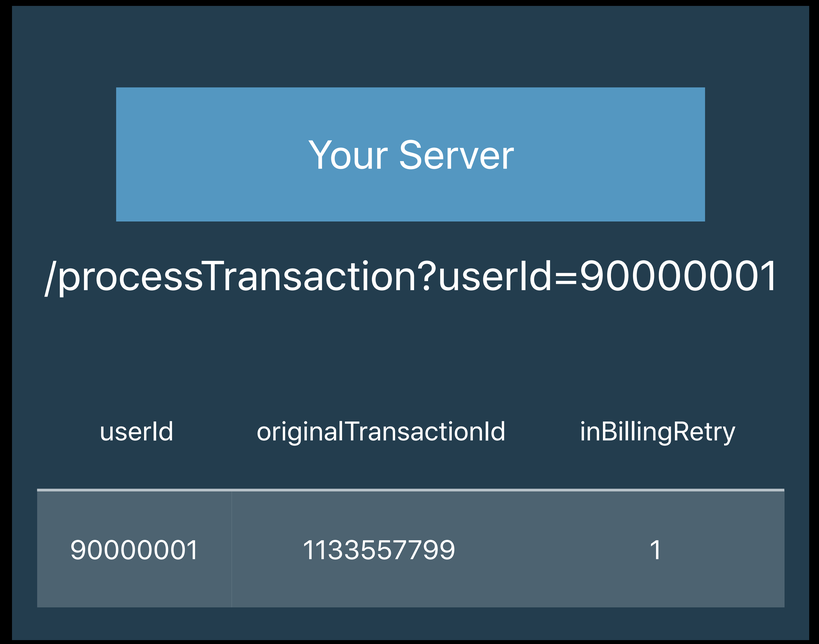
Example Subscription


```
in_app: [{
  “purchase_date”: “2018-03-26…”,
  “expires_date”: “2018-04-26…”,…
}]
```

Billing Retry

Optimization strategies

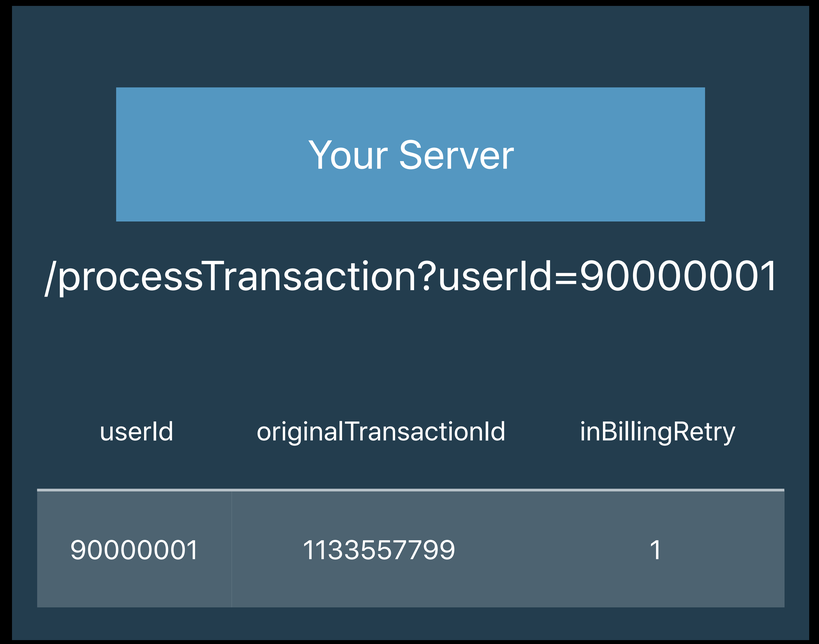
```
pending_renewal_info: [{
  “is_in_billing_retry_period”: “1”,
  “original_transaction_id”: “1133557799”,
  “product_id”: “plus_subscription_999_6m”,
  “auto_renew_status”: “1”, …
}]
```



Billing Retry

Optimization strategies

```
pending_renewal_info: [{
  “is_in_billing_retry_period”: “1”,
  “original_transaction_id”: “1133557799”,
  “product_id”: “plus_subscription_999_6m”,
  “auto_renew_status”: “1”, …
}]
```



Example Subscription


```
pending_renewal_info: [{
  “original_transaction_id”: “1133557799”,
  “product_id”: “plus_subscription_999_6m”,
  “auto_renew_status”: “1”,
  “is_in_billing_retry_period”: “1”,…
}]
```

Grace Period

Free subscription access

- While in a billing retry state
- Before a subscriber has churned

Improve recovery

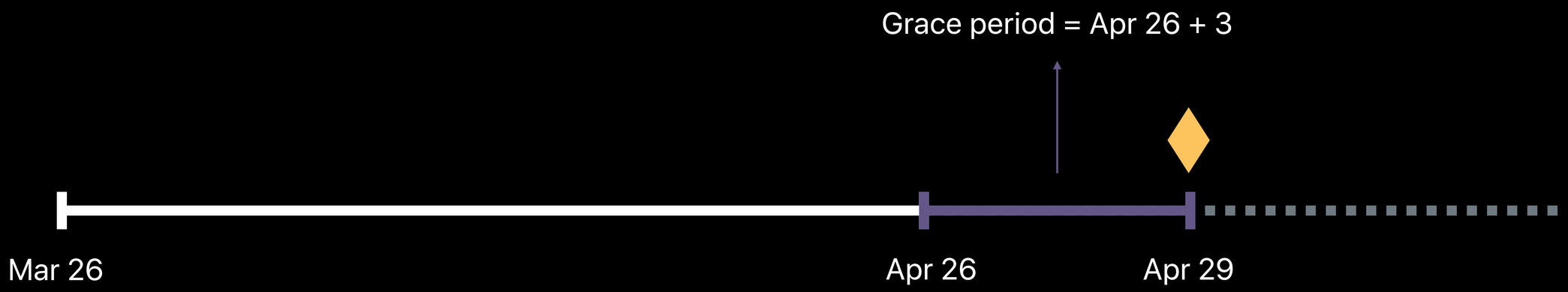
Example Grace Period


```
pending_renewal_info: [{
  “original_transaction_id”: “1133557799”,
  “product_id”: “plus_subscription_999_6m”,
  “auto_renew_status”: “1”,
  “is_in_billing_retry_period”: “1”,…
}]
```

Example Grace Period


```
pending_renewal_info: [{
  “original_transaction_id”: “1133557799”,
  “product_id”: “plus_subscription_999_6m”,
  “auto_renew_status”: “1”,
  “is_in_billing_retry_period”: “1”,…
}]
```

Example Grace Period



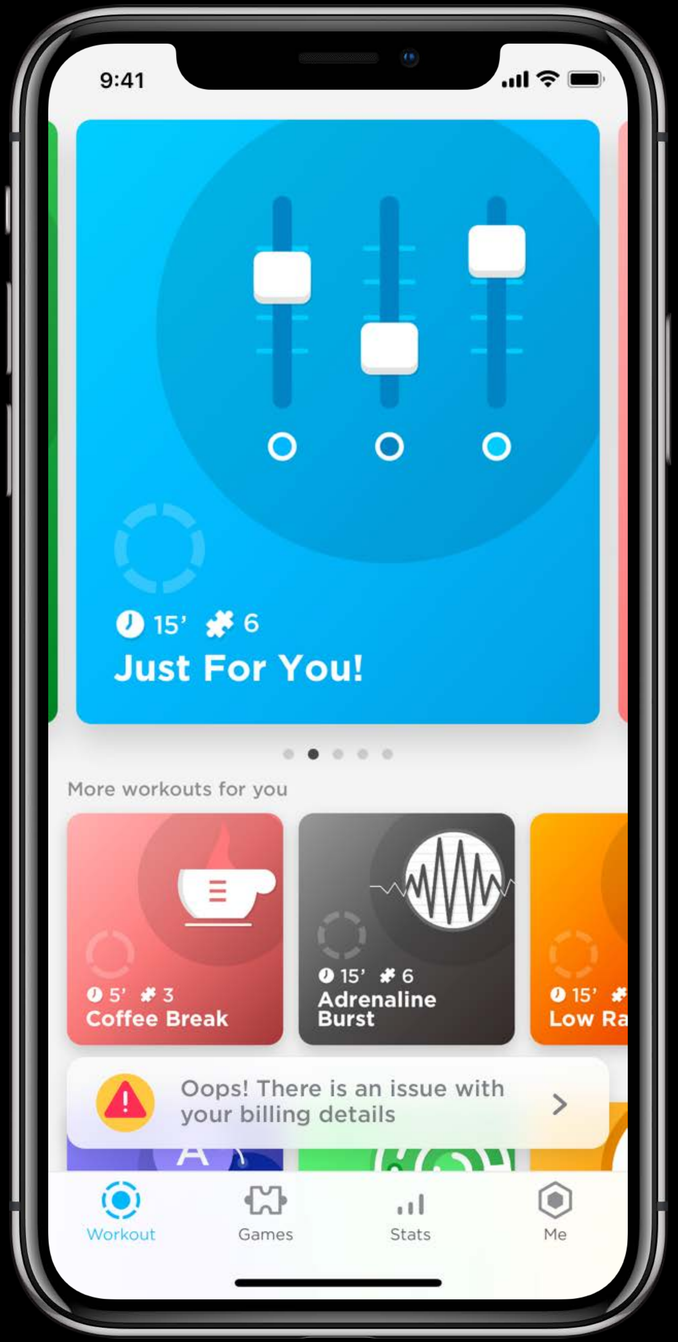

```
pending_renewal_info: [{
  “original_transaction_id”: “1133557799”,
  “product_id”: “plus_subscription_999_6m”,
  “auto_renew_status”: “1”,
  “is_in_billing_retry_period”: “1”,…
}]
```

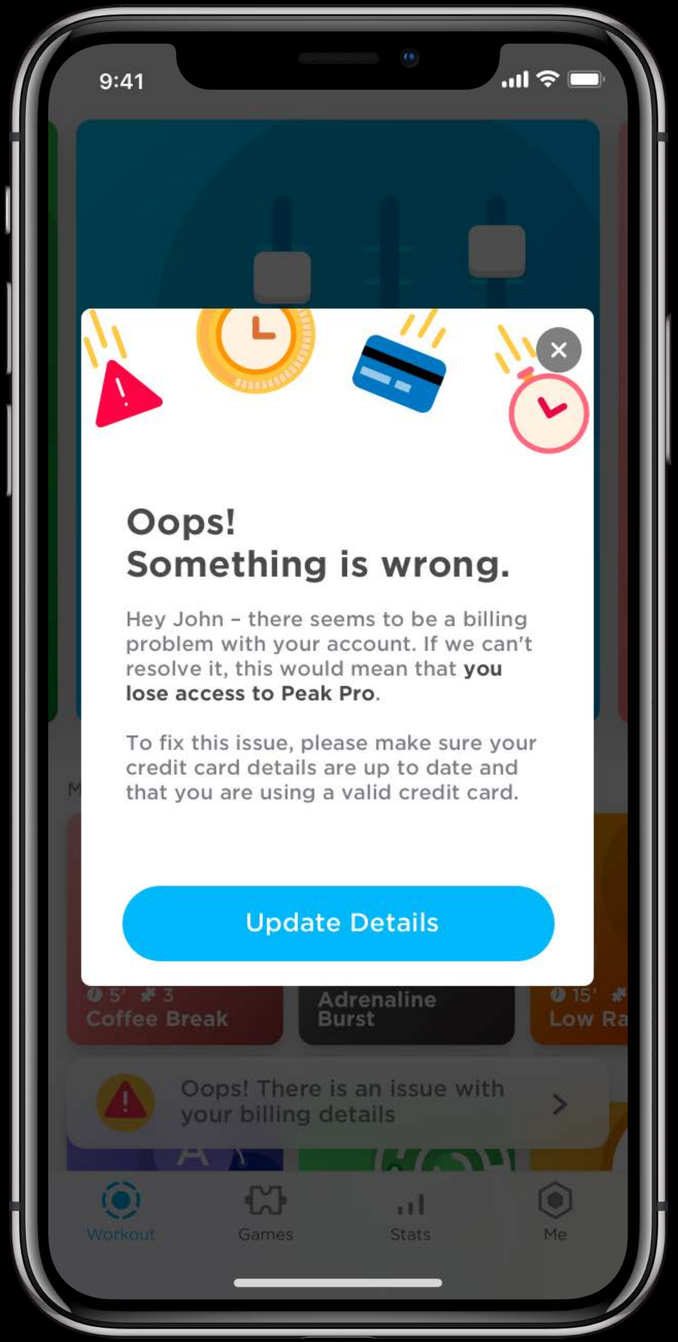
Customer Messaging

Messaging

- Update payment methods
- Restate the value proposition

Offer limited service


9:41
15’ 6
Just For You!
More workouts for you
5’ 3
Coffee Break
15’ 6
Adrenaline Burst
15’
Low Ra
Oops! There is an issue with your billing details
Workout
Games
Stats
Me


9:41
Oops!
Something is wrong.
Hey John – there seems to be a billing problem with your account. If we can't resolve it, this would mean that **you lose access to Peak Pro**.
To fix this issue, please make sure your credit card details are up to date and that you are using a valid credit card.
Update Details
Coffee Break
Adrenaline Burst
Oops! There is an issue with your billing details
Workout
Games
Stats
Me

Customer Messaging

NEW

Edit billing information

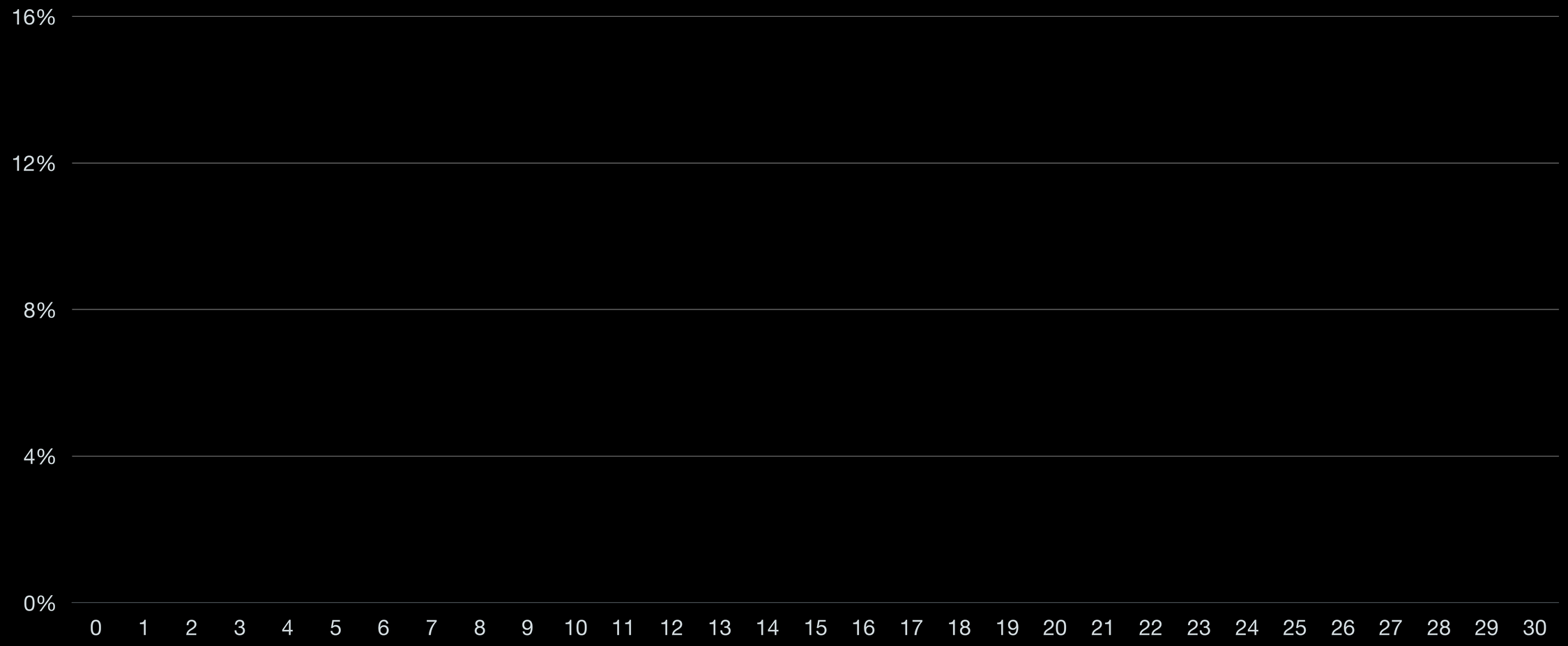
- https://apps.apple.com/account/billing

Manage subscriptions

- https://apps.apple.com/account/subscriptions

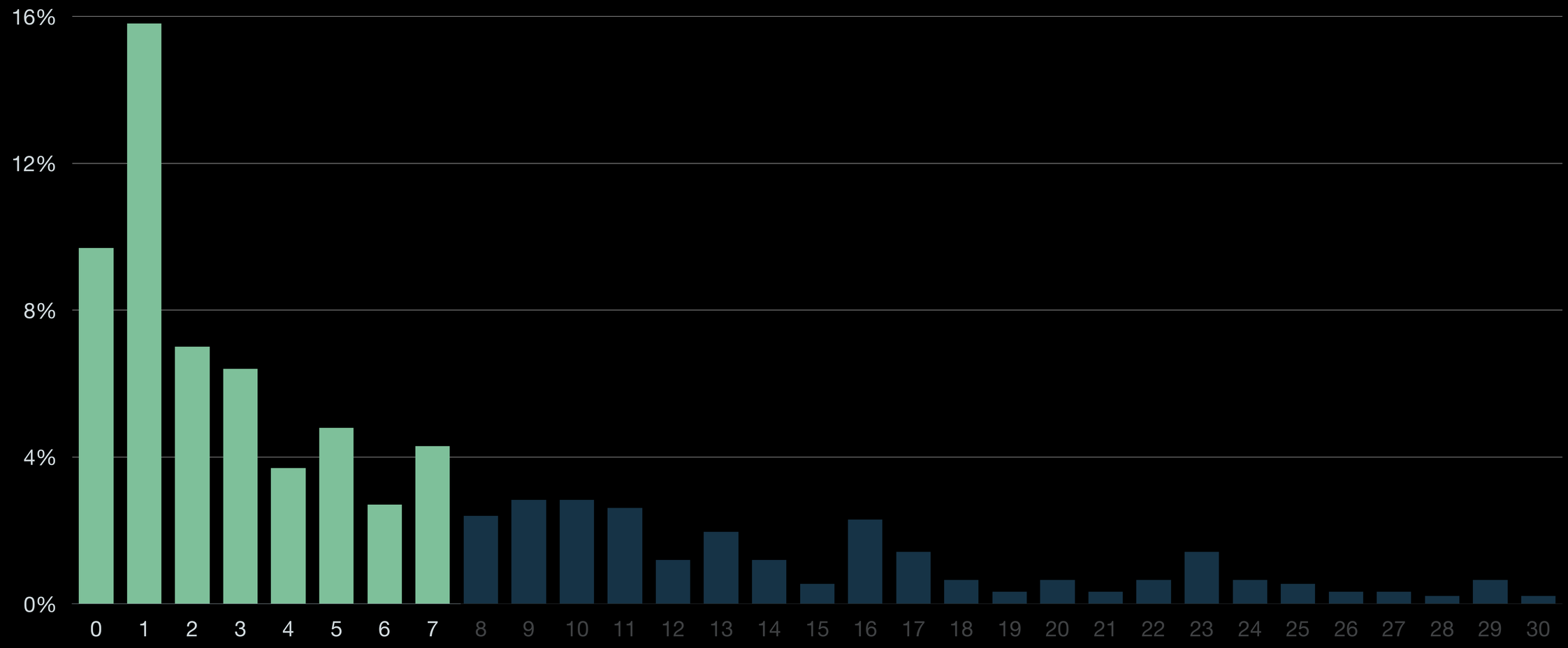
Grace Period

Customer recovery

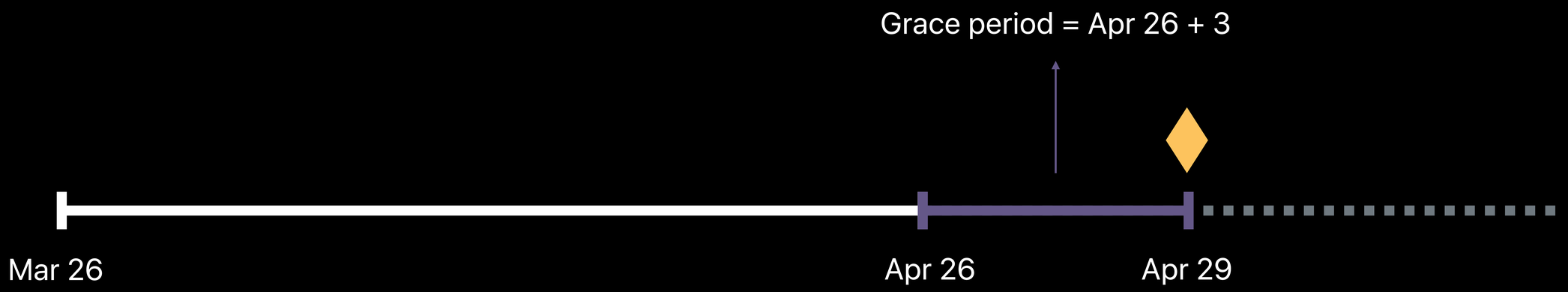


Grace Period

Customer recovery



Example Grace Period




```
pending_renewal_info: [{
  “original_transaction_id”: “1133557799”,
  “product_id”: “plus_subscription_999_6m”,
  “auto_renew_status”: “1”,
  “is_in_billing_retry_period”: “1”,…
}]
```

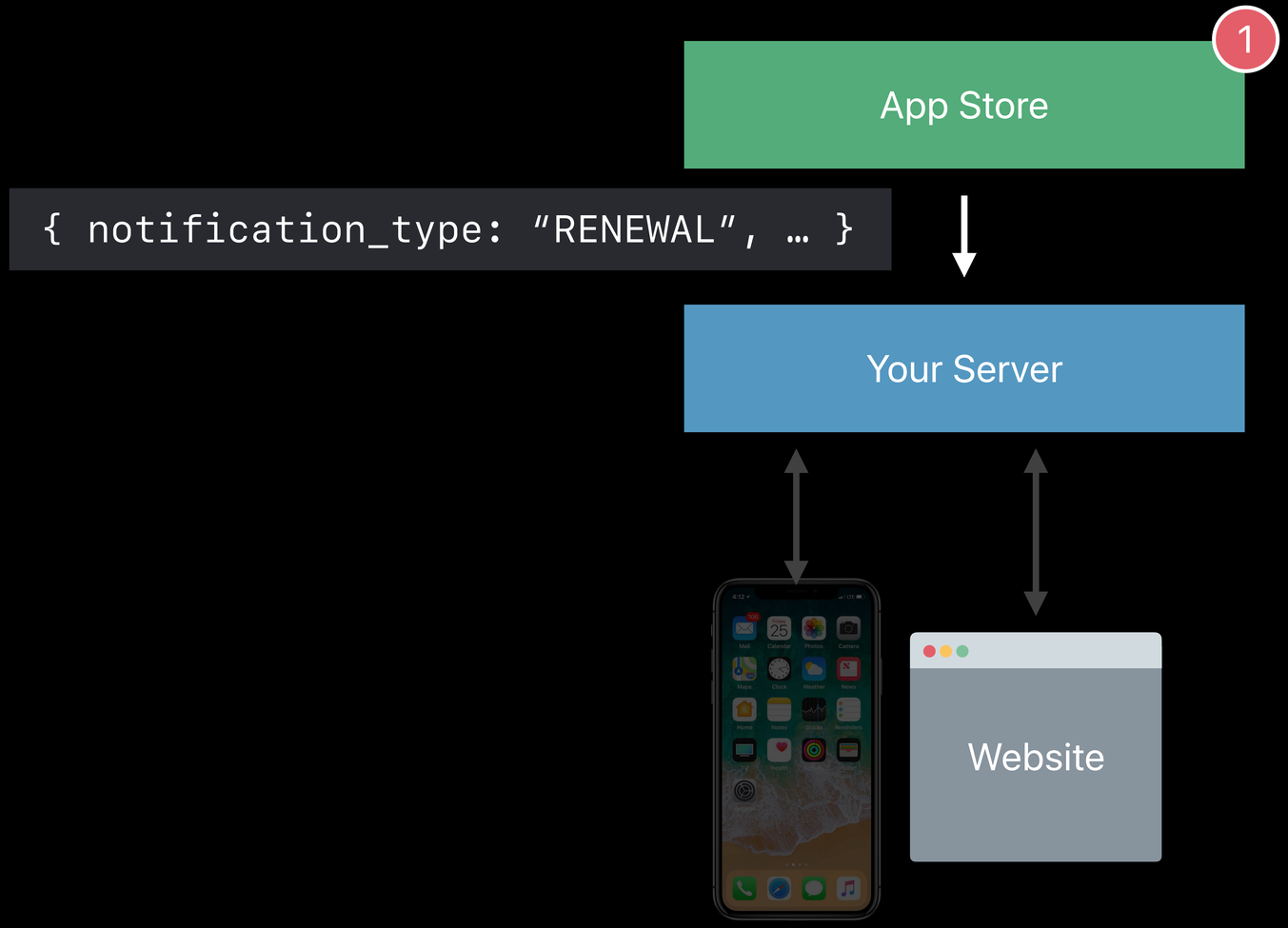
Example Subscription


```
latest_receipt_info: [{
  “purchase_date”: “2018-05-15…”,
  “expires_date”: “2018-06-15…”,…
}],
```

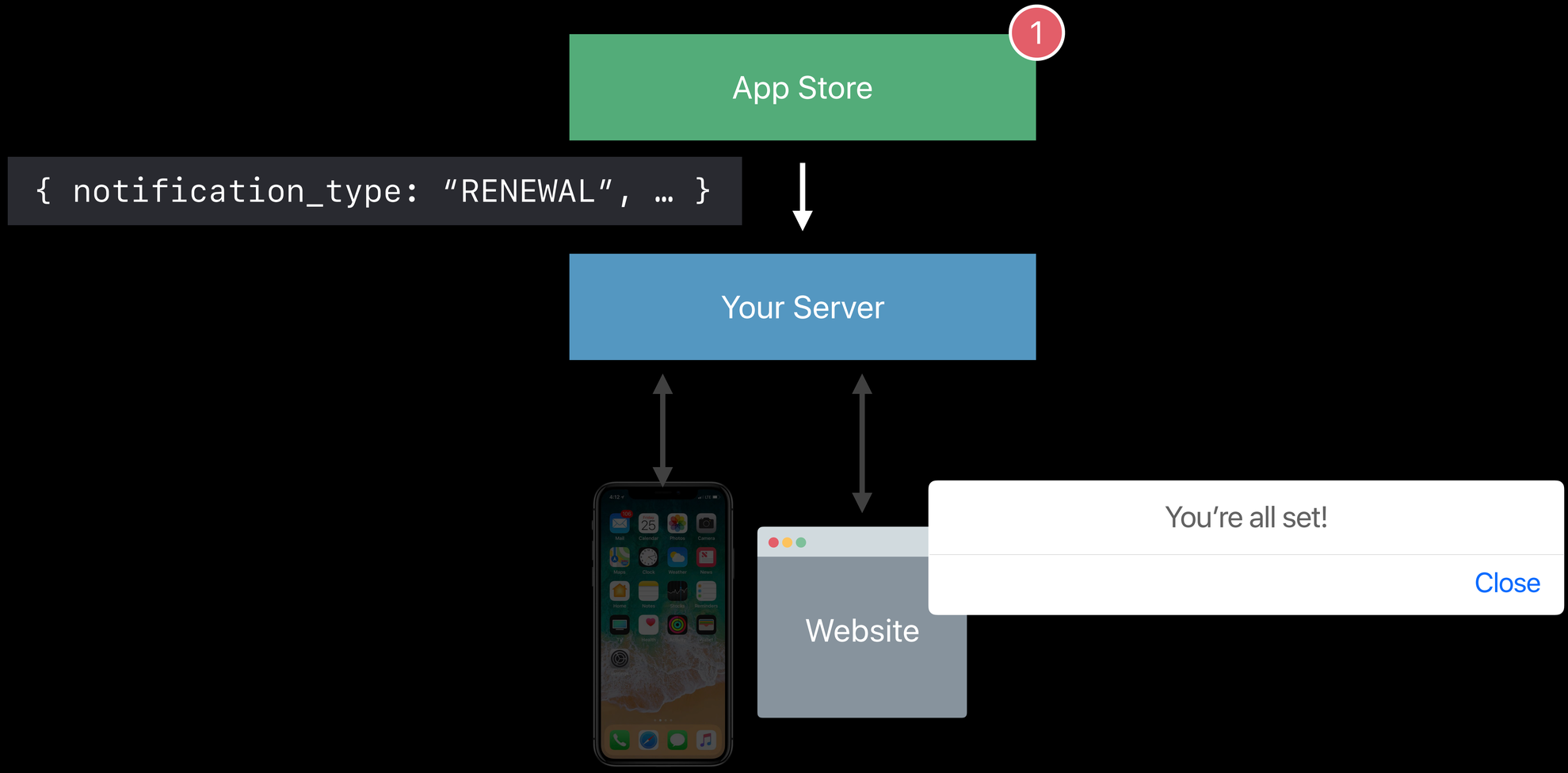
Example Subscription


```
latest_receipt_info: [{
  “purchase_date”: “2018-05-15…”,
  “expires_date”: “2018-06-15…”,…
}],
```

Server-to-Server Notifications

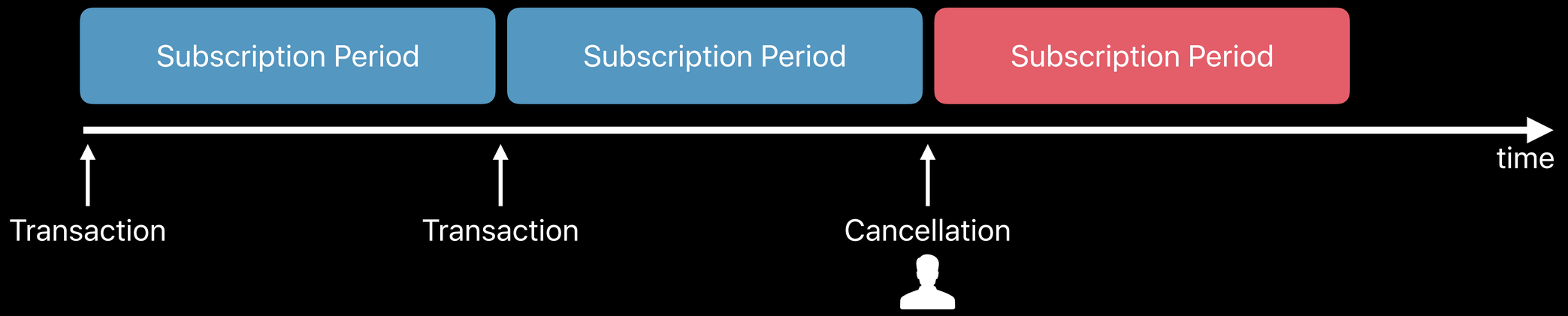


Server-to-Server Notifications



Voluntary Churn

Voluntary Churn



Minimizing Voluntary Churn

Implement status polling

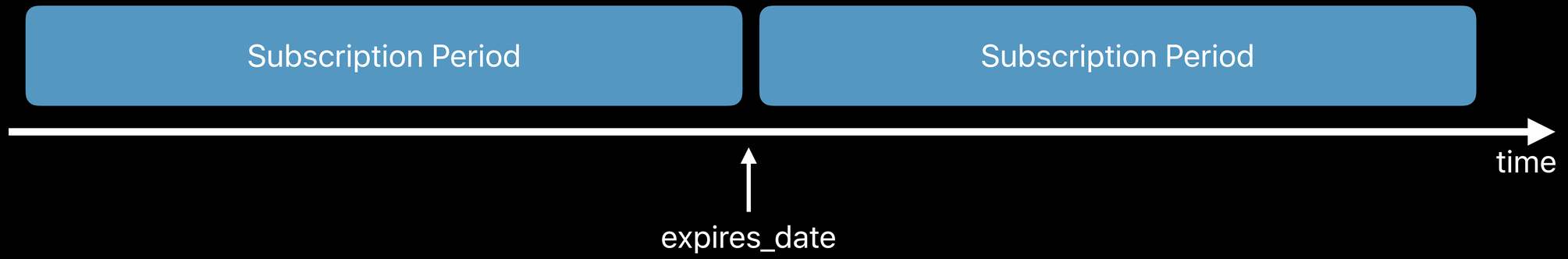
Offer attractive alternative subscriptions

Status Polling

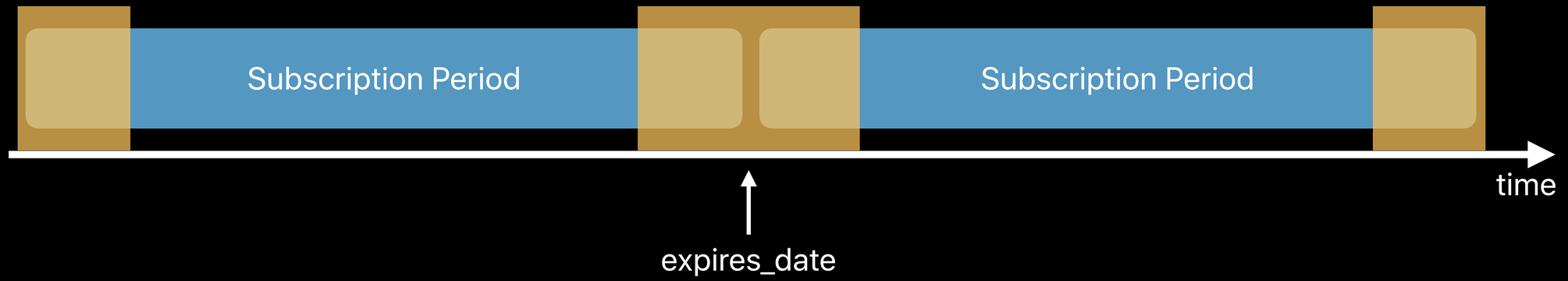
Will my subscriber churn?

Has my subscriber renewed?

When to Status Poll



When to Status Poll



Additional Receipt Fields

Storing subscriber status fields

Additional Receipt Fields

Storing subscriber status fields

Save decoded JSON from `/verifyReceipt`

Additional Receipt Fields

Storing subscriber status fields

Save decoded JSON from `/verifyReceipt`

userId	originalTransactionIdentifier	latestReceiptData	latestExpiresDate	latestDecodedReceipt
90000001	1234567890	d24Fs...kJ87dDGe3=	2018-08-08...	{ ... auto_renew_status: 1 ...}

Additional Receipt Fields

Storing subscriber status fields

Save decoded JSON from `/verifyReceipt`

Or parse out specific fields

userId	originalTransactionIdentifier	latestReceiptData	latestExpiresDate	inBillingRetry
90000001	1234567890	d24Fs...kJ87dDGe3=	2018-08-08...	1

Additional Receipt Fields

Storing subscriber status fields

Save decoded JSON from `/verifyReceipt`

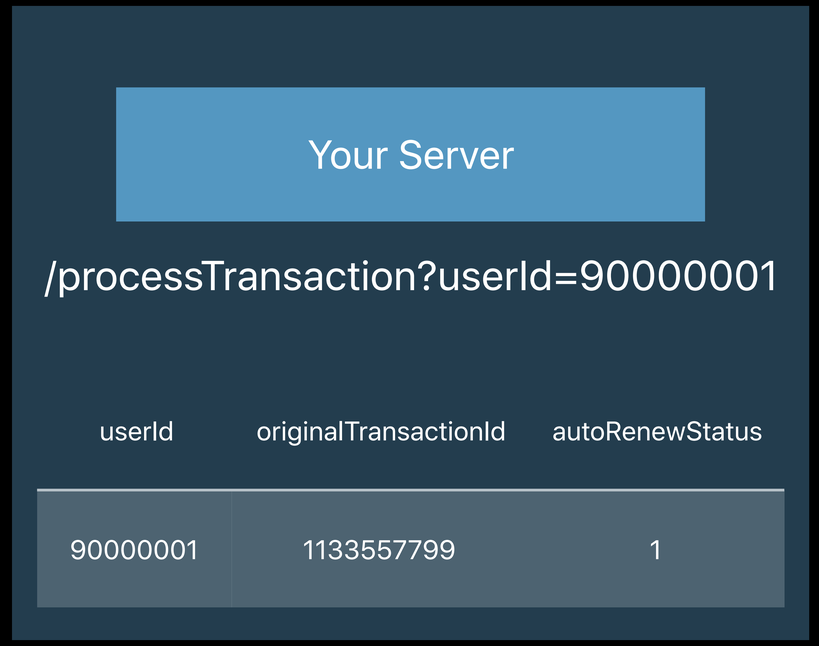
Or parse out specific fields

userId	originalTransactionIdentifier	latestReceiptData	latestExpiresDate	inBillingRetry
90000001	1234567890	d24Fs...kJ87dDGe3=	2018-08-08...	1

Auto Renew Status

Voluntary churn

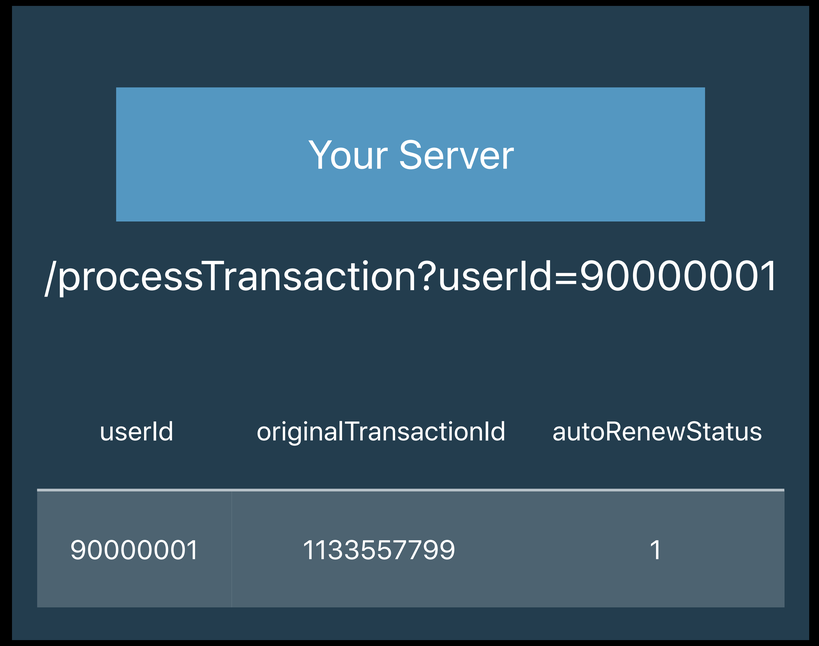
```
pending_renewal_info: [{
  “is_in_billing_retry_period”: “1”,
  “original_transaction_id”: “1133557799”,
  “product_id”: “plus_subscription_999_6m",
  “auto_renew_status”: “1”, …
}]
```



Auto Renew Status

Voluntary churn

```
pending_renewal_info: [{
  “is_in_billing_retry_period”: “1”,
  “original_transaction_id”: “1133557799”,
  “product_id”: “plus_subscription_999_6m",
  “auto_renew_status”: “1”, …
}]
```



Example Subscription

Example Subscription

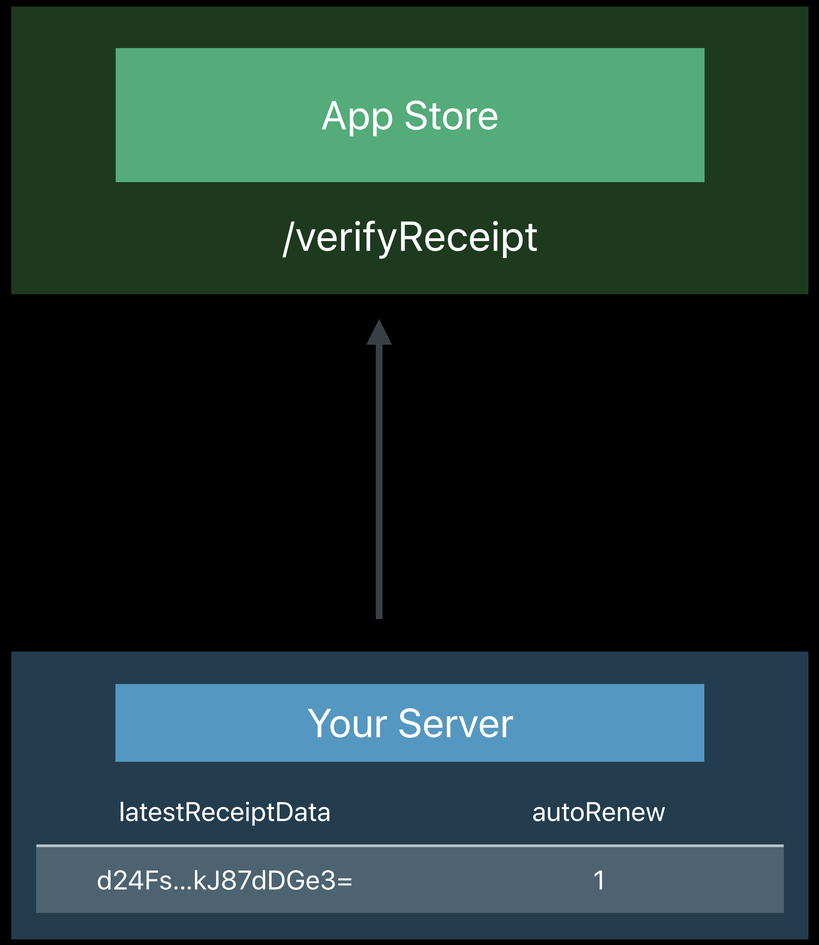

```
in_app: [{
  “purchase_date”: “2018-03-26…”,
  “expires_date”: “2018-04-26…”,…
}]
```

Example Subscription


```
in_app: [{
  “purchase_date”: “2018-03-26…”,
  “expires_date”: “2018-04-26…”,…
}]
```

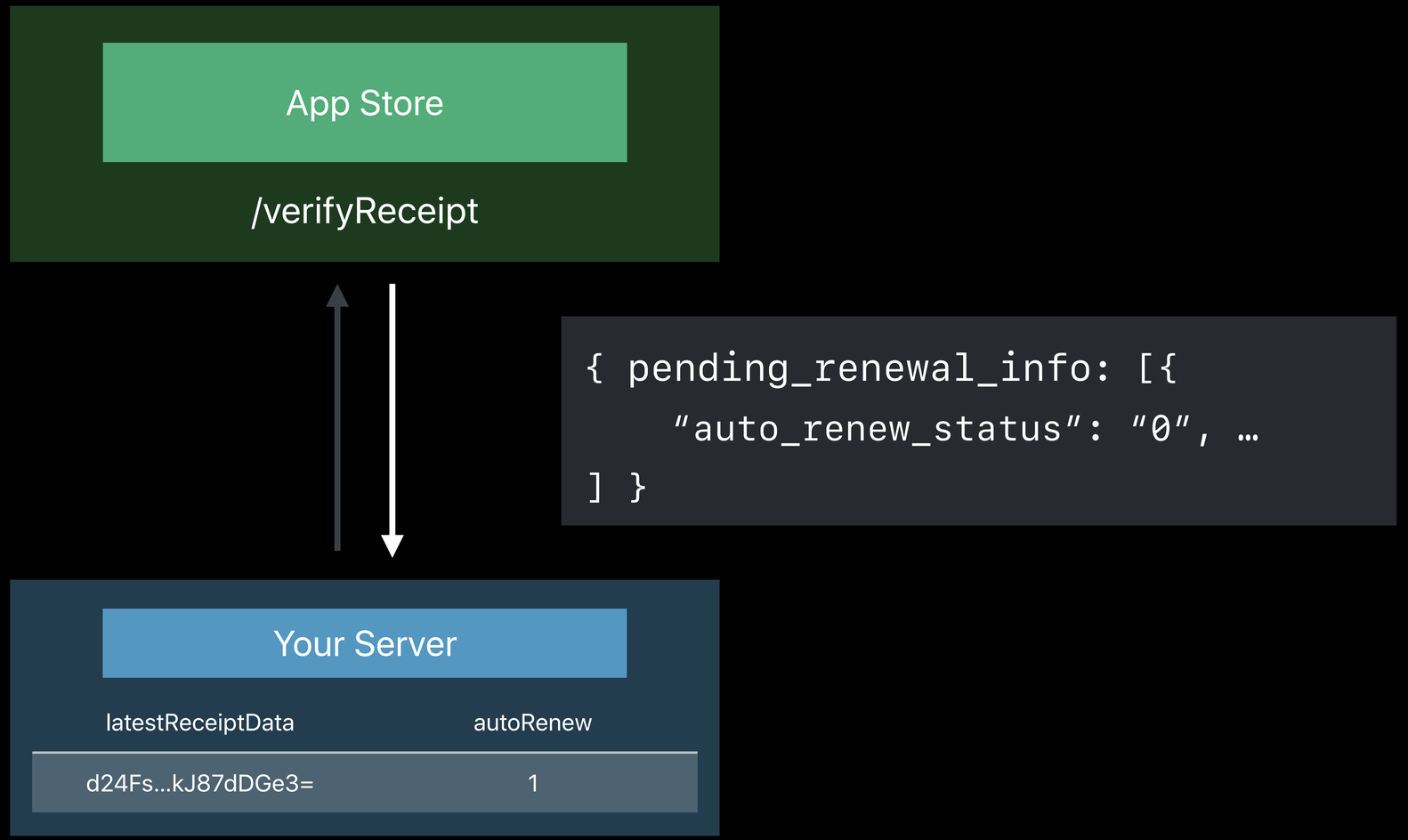
Status Polling

Update auto renew status from server



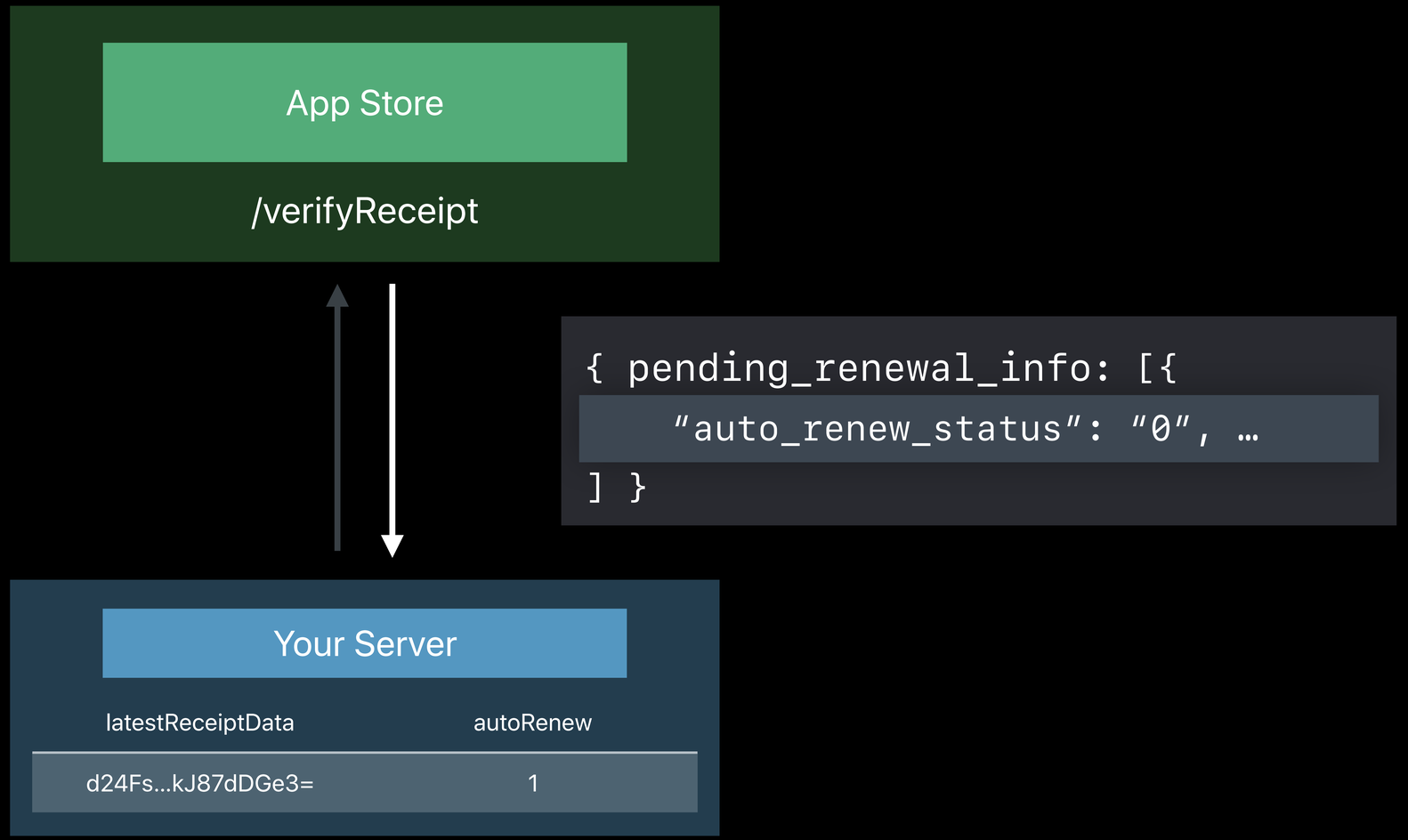
Status Polling

Update auto renew status from server



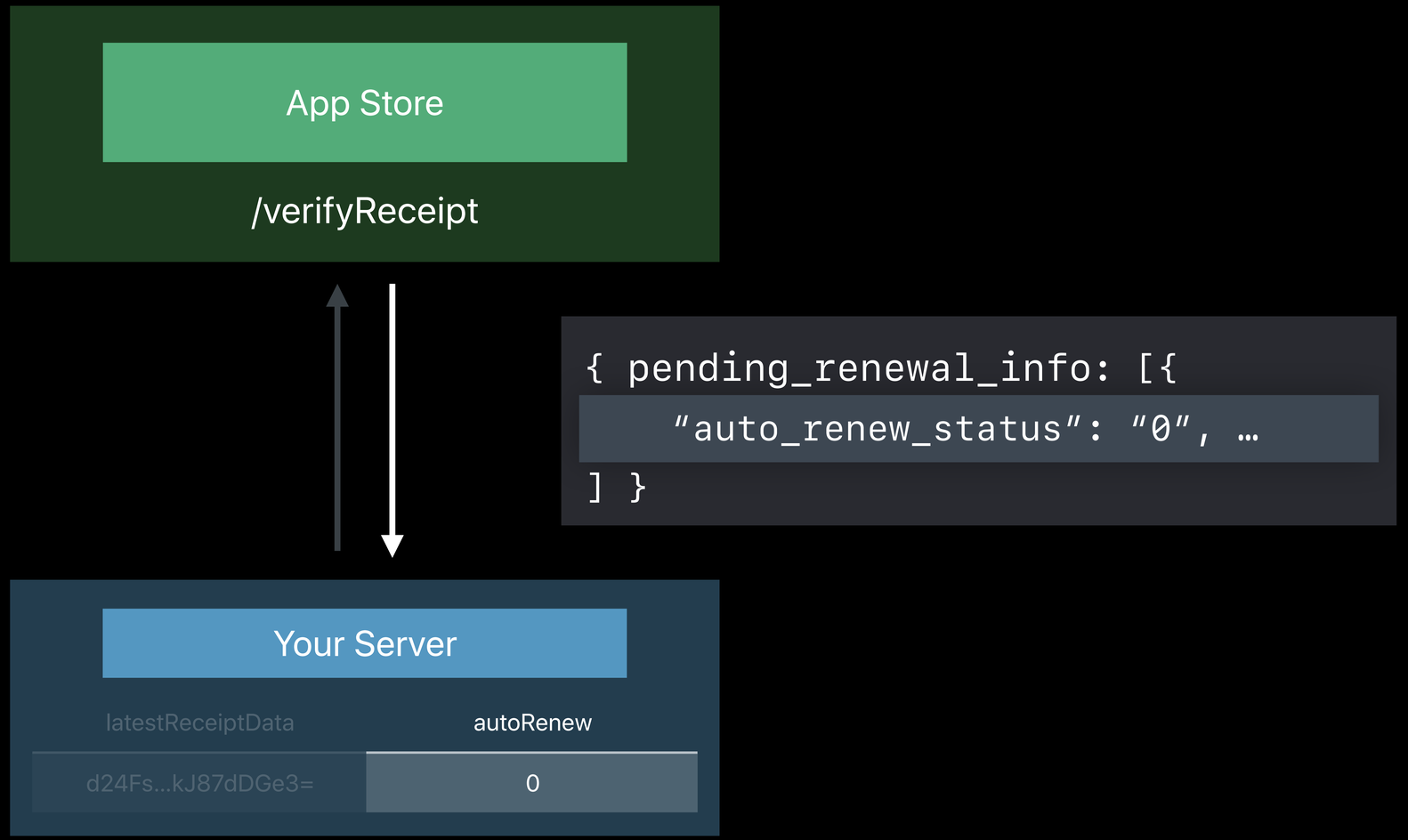
Status Polling

Update auto renew status from server



Status Polling

Update auto renew status from server



Example Subscription


```
pending_renewal_info: [{
  “original_transaction_id”: “1133557799”,
  “product_id”: “plus_subscription_999_6m",
  “auto_renew_status”: “0”, …
}]
```

Example Subscription

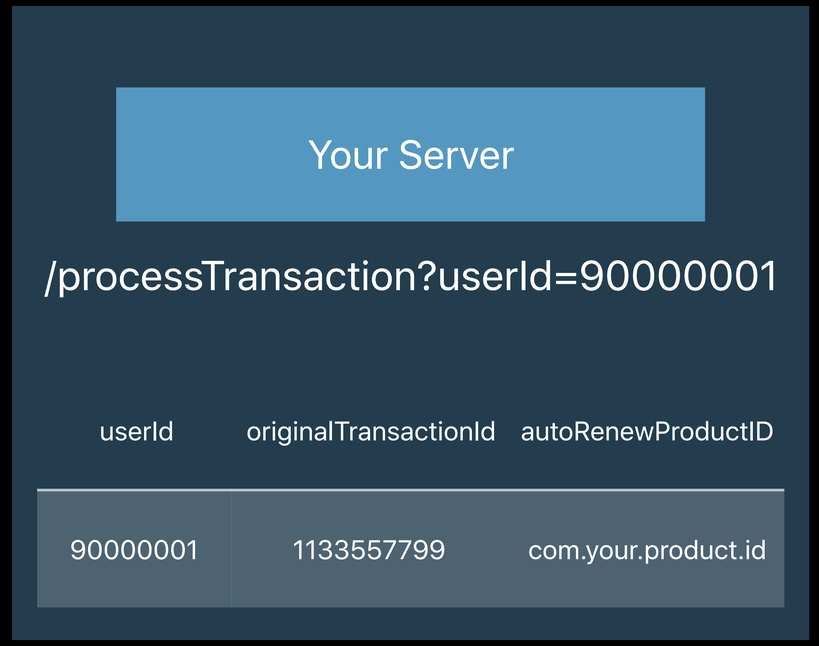

```
pending_renewal_info: [{
  “original_transaction_id”: “1133557799”,
  “product_id”: “plus_subscription_999_6m",
  “auto_renew_status”: “0”, …
}]
```


9:41
PRO
Here's a reason to come back!
Go Pro again for
60% less
Workout
Games
Stats
Me


9:41
PRO
Here's a reason to come back!
Go Pro again for
60% less
Workout
Games
Stats
Me

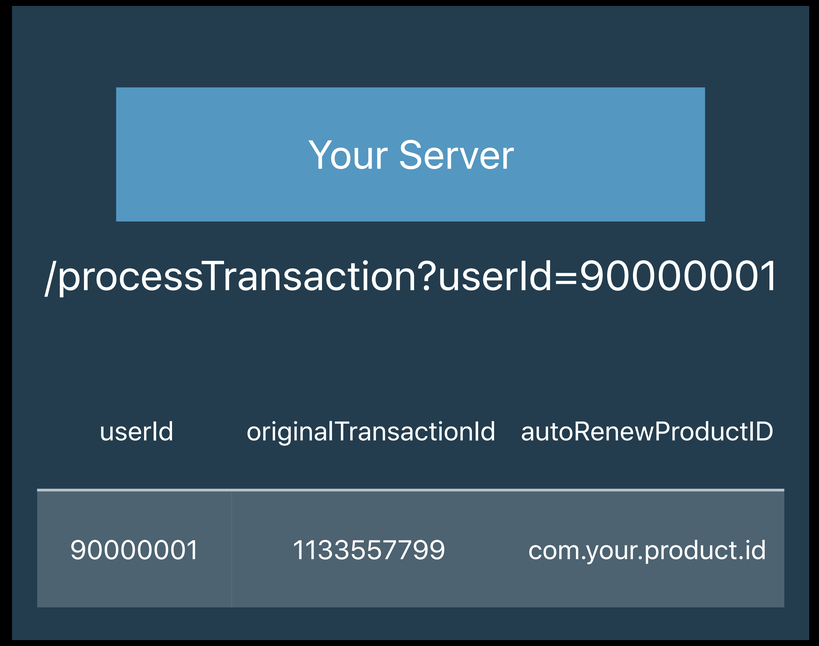
Auto Renew Preference

```
pending_renewal_info: [{
  “original_transaction_id”: “1133557799”,
  “product_id”: “plus_subscription_999_6m",
  “auto_renew_status”: “1”,
  “auto_renew_product_id”: “com.your.product.id“, …
}]
```

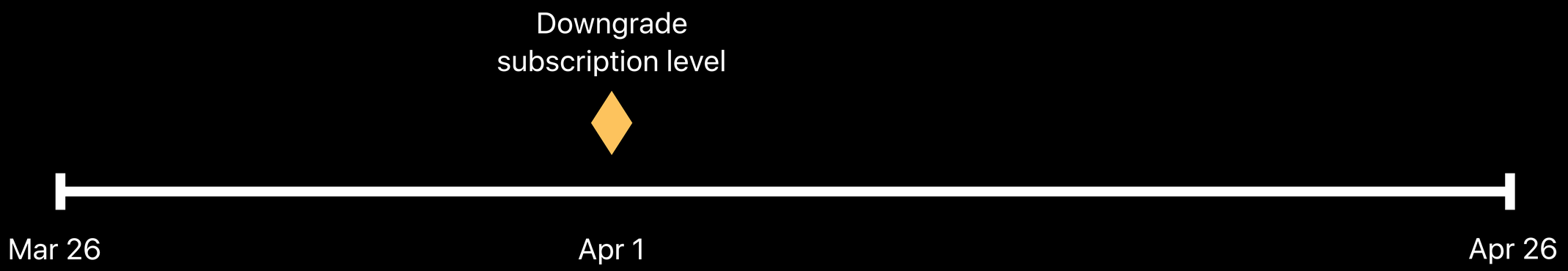


Auto Renew Preference

```
pending_renewal_info: [{
  “original_transaction_id”: “1133557799”,
  “product_id”: “plus_subscription_999_6m",
  “auto_renew_status”: “1”,
  “auto_renew_product_id”: “com.your.product.id”, …
}]
```

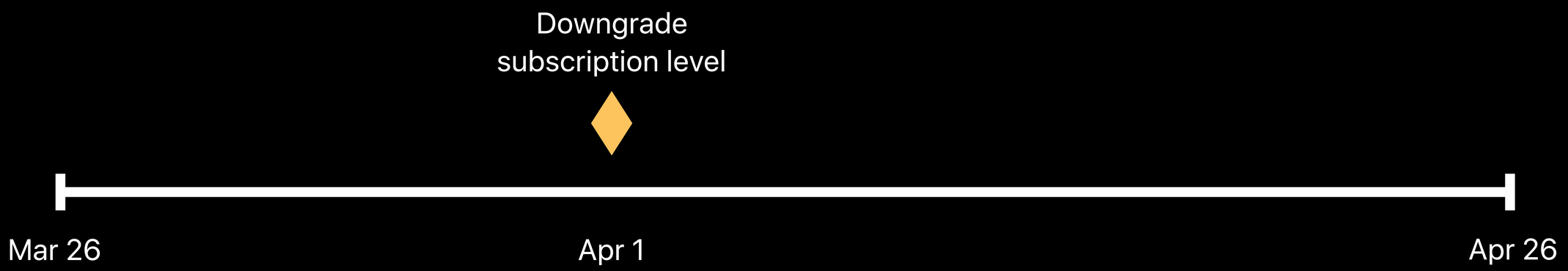


Example Subscription




```
pending_renewal_info: [{
  “original_transaction_id”: “1133557799”,
  “product_id”: “plus_subscription_999_6m",
  “auto_renew_status”: “1”,
  “auto_renew_product_id”: “com.your.product.id”, …
}]
```

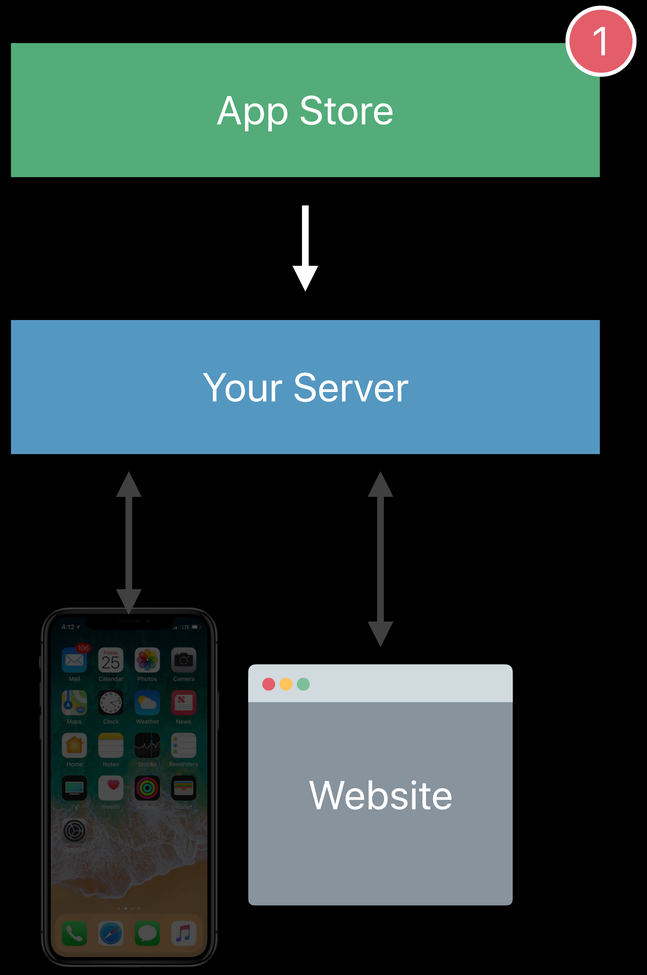
Example Subscription




```
pending_renewal_info: [{
  “original_transaction_id”: “1133557799”,
  “product_id”: “plus_subscription_999_6m",
  “auto_renew_status”: “1”,
  “auto_renew_product_id”: “com.your.product.id”, …
}]
```

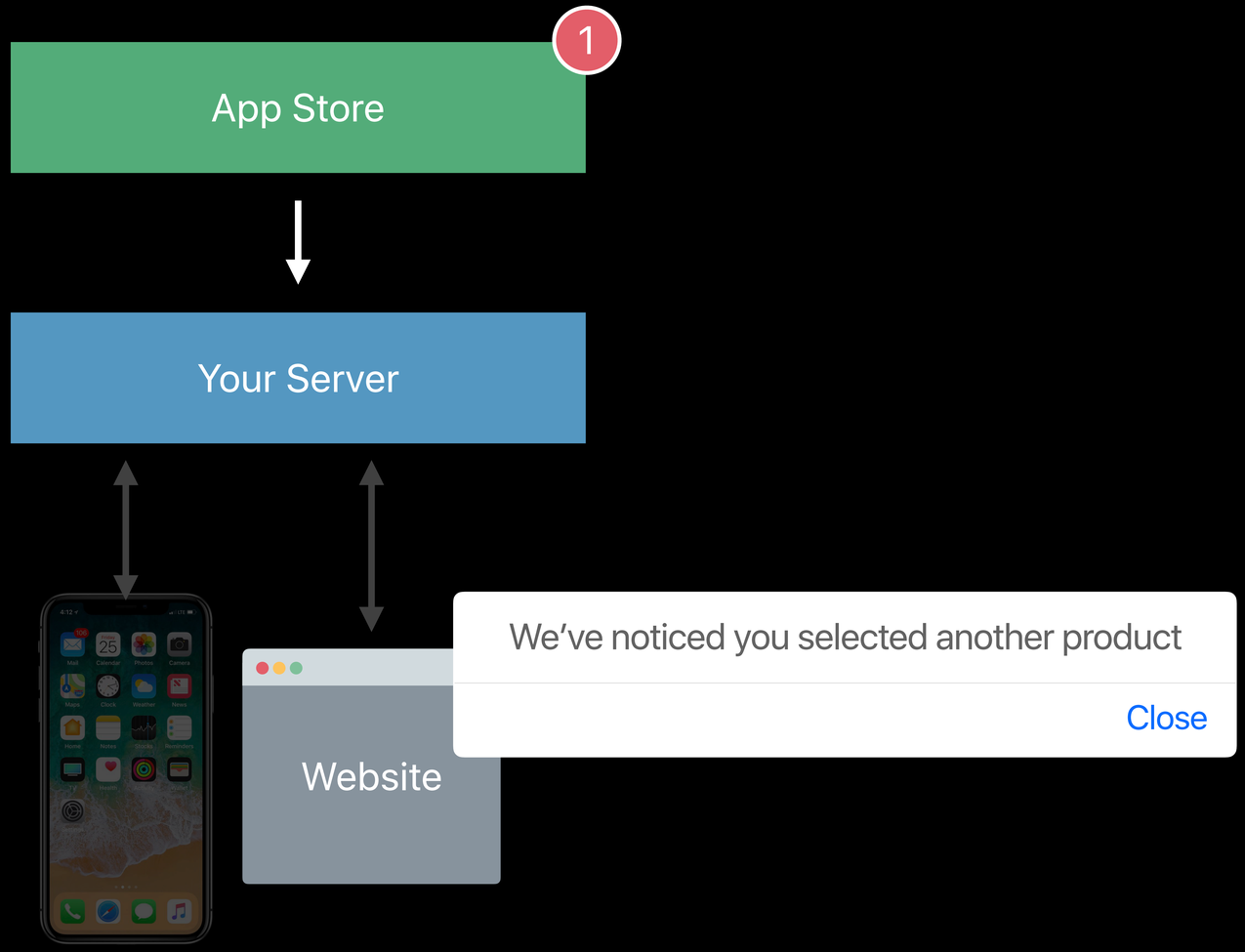
Server-to-Server Notifications

```
{ notification_type:
“DID_CHANGE_RENEWAL_PREFERENCE”, … }
```



Server-to-Server Notifications

```
{ notification_type:
“DID_CHANGE_RENEWAL_PREFERENCE”, … }
```



Winback

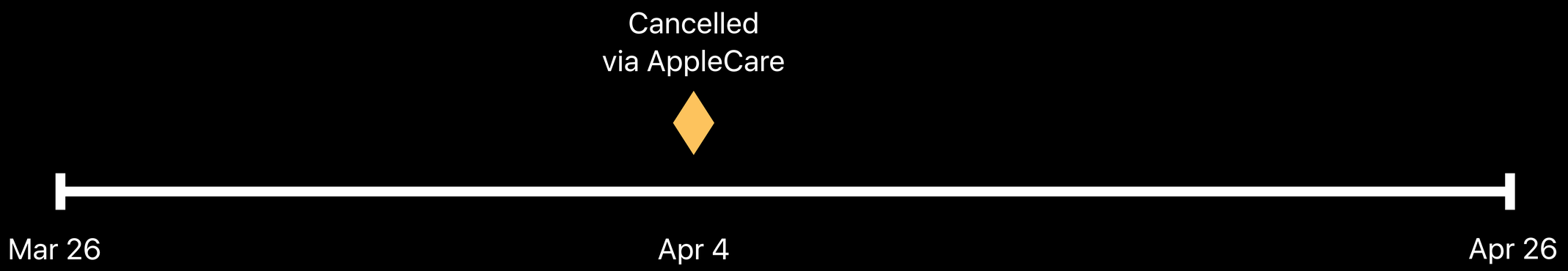
Reengage after subscribers churn

- Resubscription offers
- Customer surveys

Example Subscription

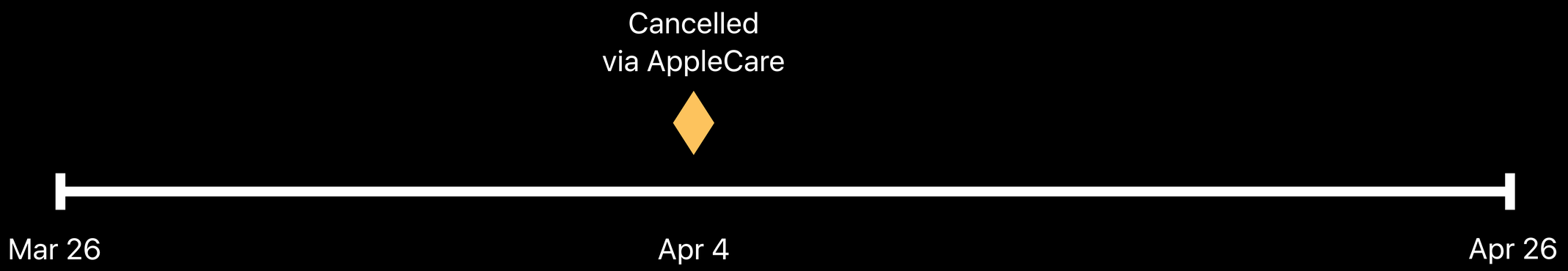

```
in_app: [{
  “purchase_date”: “2018-03-26…”,
  “expires_date”: “2018-04-26…”,
}]
```

Example Subscription



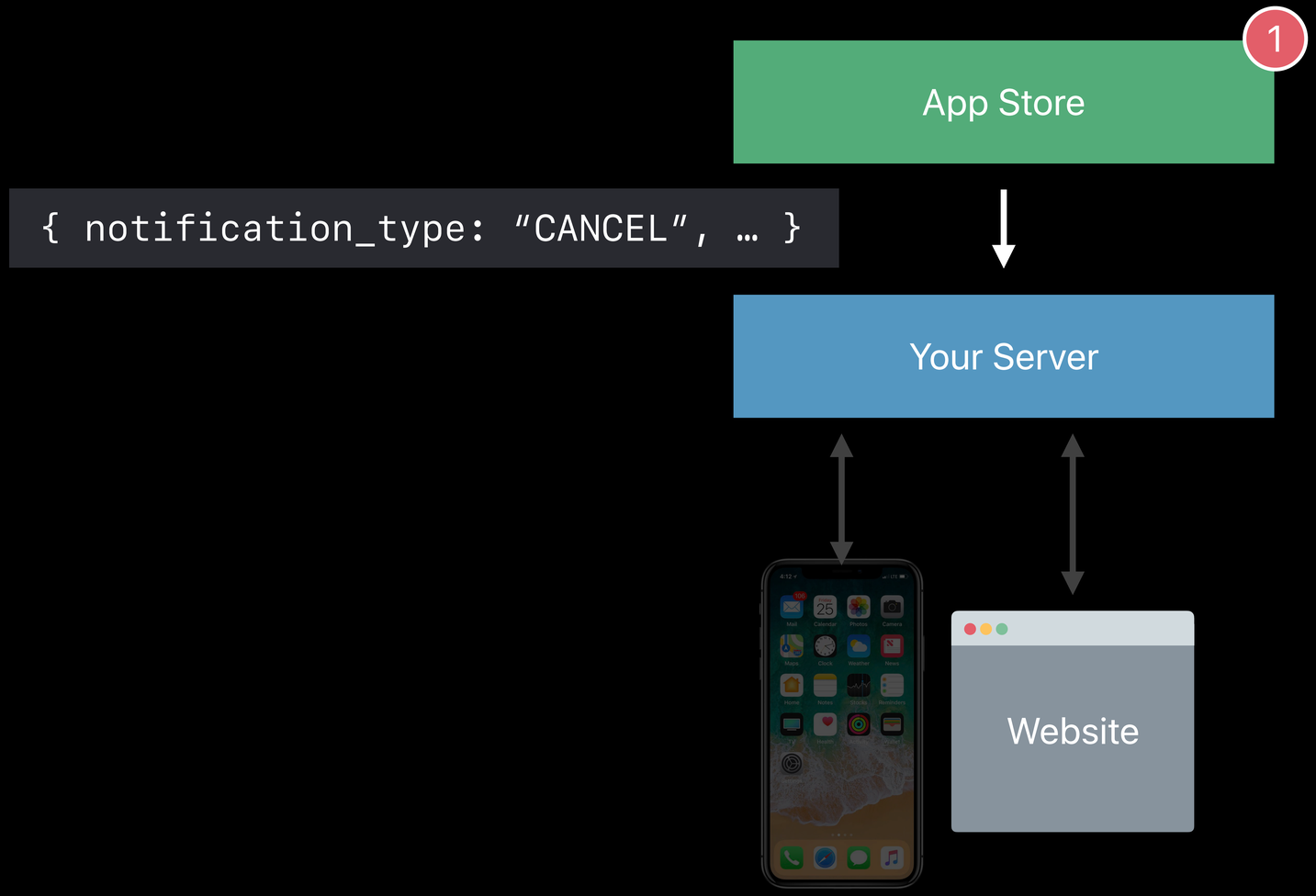

```
in_app: [{
  “purchase_date”: “2018-03-26…”,
  “expires_date”: “2018-04-26…”,
  “cancellation_date”: “2018-04-04…”,
…
}]
```

Example Subscription

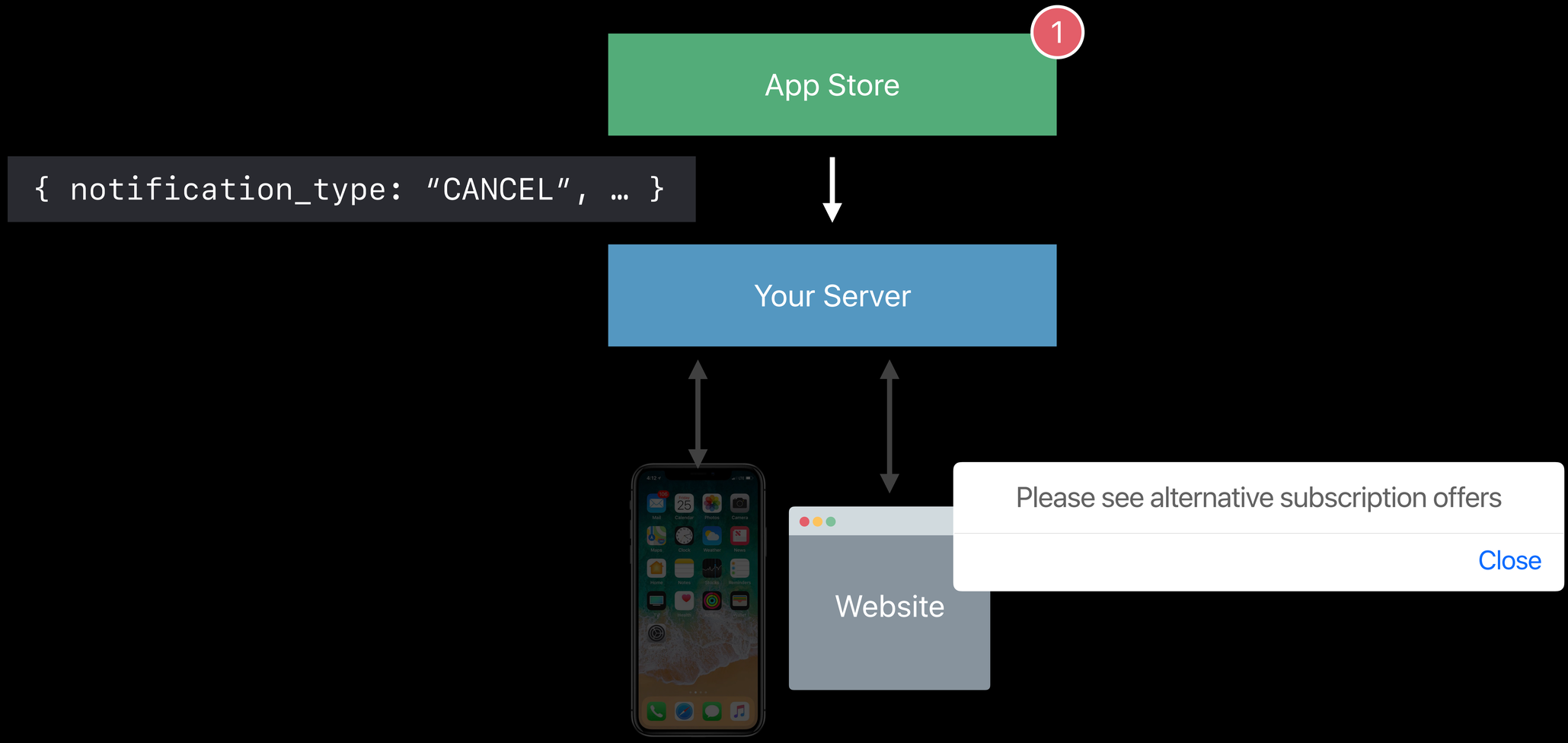



```
in_app: [{
  “purchase_date”: “2018-03-26…”,
  “expires_date”: “2018-04-26…”,
  “cancellation_date”: “2018-04-04…”,
…
}]
```

Server-to-Server Notifications



Server-to-Server Notifications



Expiration Intent

```
pending_renewal_info: [{
  “original_transaction_id”: “1133557799”,
  “product_id”: “plus_subscription_999_6m”,
  “auto_renew_status”: “0”
  “expiration_intent”: “1”, …
}]
```

Your Server

/processTransaction?userId=90000001

userId	originalTransactionId	expirationIntent
90000001	1133557799	1

Expiration Intent

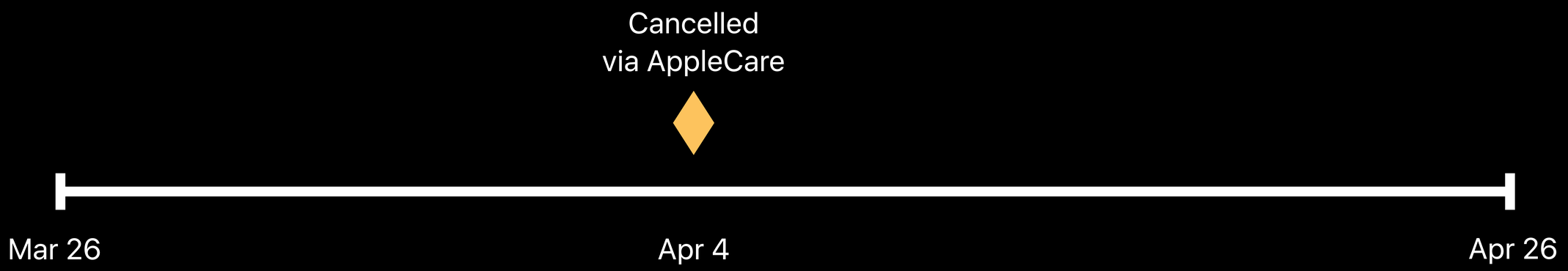
```
pending_renewal_info: [{
  “original_transaction_id”: “1133557799”,
  “product_id”: “plus_subscription_999_6m”,
  “auto_renew_status”: “0”
  “expiration_intent”: “1”, …
}]
```

Your Server

/processTransaction?userId=90000001

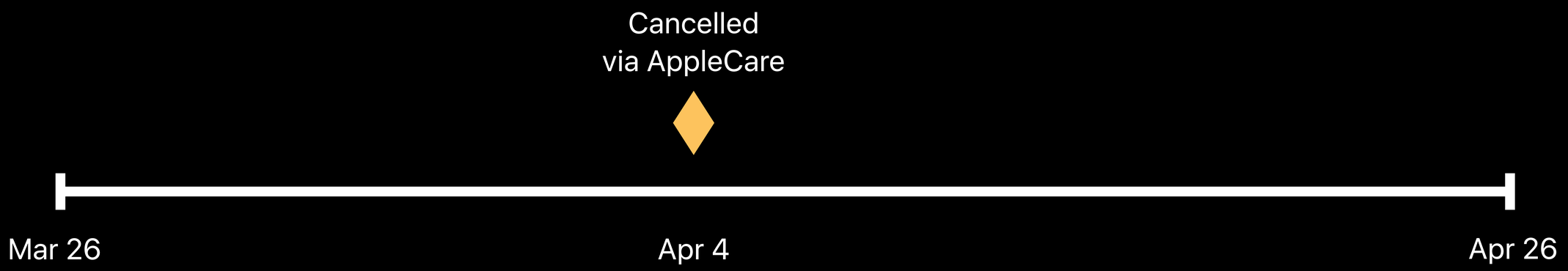
userId	originalTransactionId	expirationIntent
90000001	1133557799	1

Example Subscription




```
in_app: [{
  “purchase_date”: “2018-03-26…”,
  “expires_date”: “2018-04-26…”,
  “cancellation_date”: “2018-04-04…”,
  “expiration_intent”: “1”, …
}]
```

Example Subscription



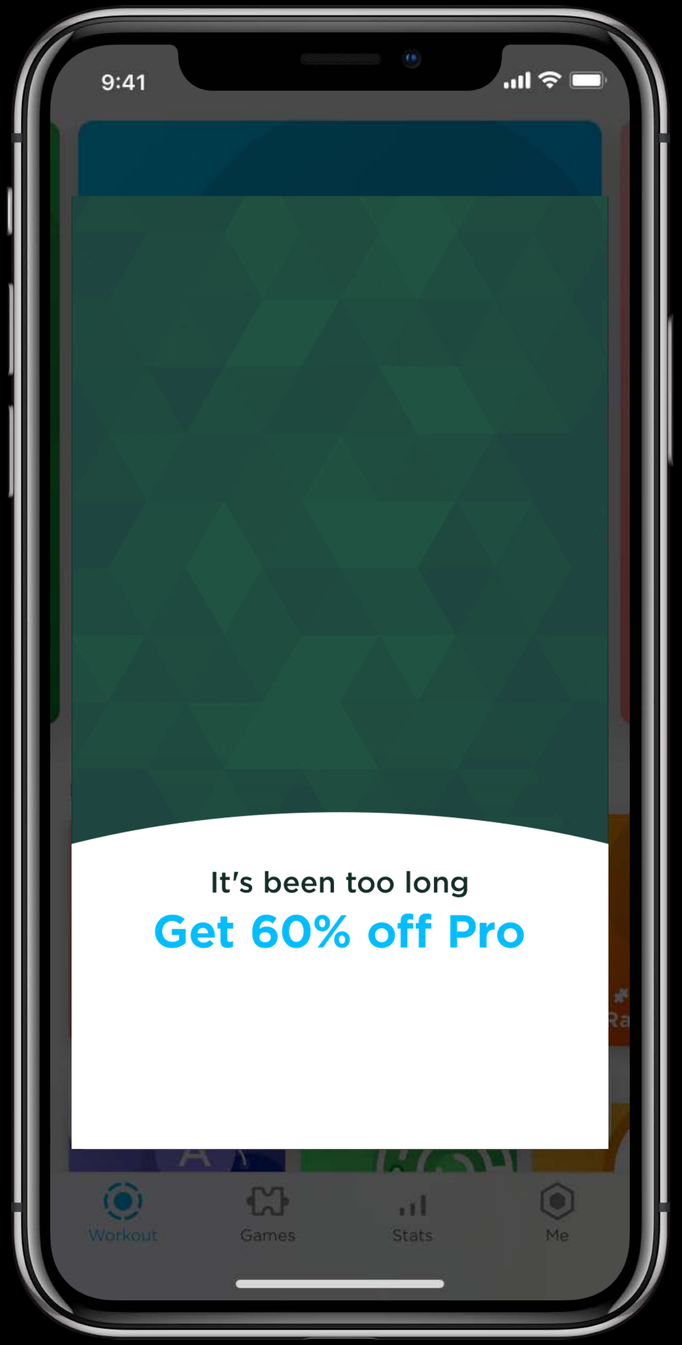

```
in_app: [{
  “purchase_date”: “2018-03-26…”,
  “expires_date”: “2018-04-26…”,
  “cancellation_date”: “2018-04-04…”,
  “expiration_intent”: “1”, …
}]
```

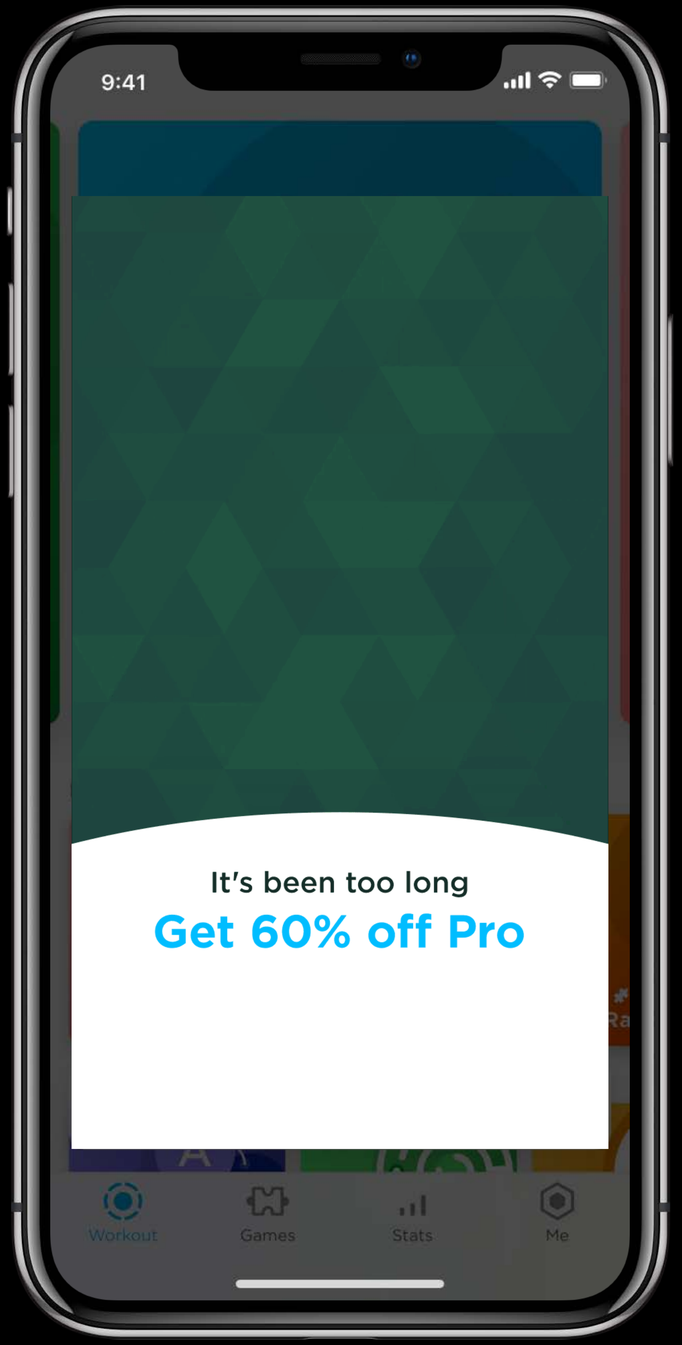
Winback Lapsed Subscribers

Voluntary churn

Survey lapsed subscribers

Show alternative subscription products


9:41
It's been too long
Get 60% off Pro
Workout
Games
Stats
Me


9:41
It's been too long
Get 60% off Pro
Workout
Games
Stats
Me

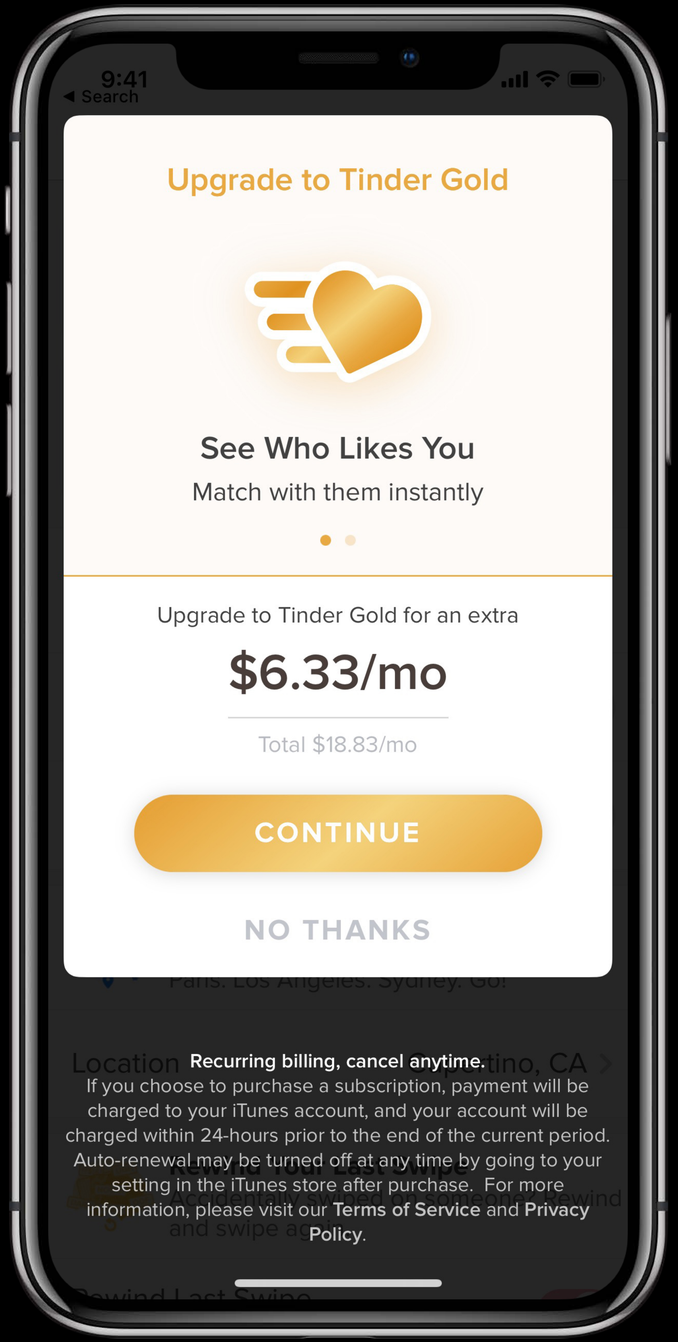
Winback Lapsed Subscribers

Involuntary churn

Show alternative subscription products

Persistent messaging in-app

Limited subscription experience


9:41
Search
Upgrade to Tinder Gold
See Who Likes You
Match with them instantly
Upgrade to Tinder Gold for an extra
$6.33/mo
Total $18.83/mo
CONTINUE
NO THANKS
Recurring billing, cancel anytime.
If you choose to purchase a subscription, payment will be charged to your iTunes account, and your account will be charged within 24-hours prior to the end of the current period. Auto-renewal may be turned off at any time by going to your setting in the iTunes store after purchase. For more information, please visit our Terms of Service and Privacy Policy.

Reducing Subscriber Churn

Summary

Reducing Subscriber Churn

Summary

Leverage subscription specific receipt fields

Reducing Subscriber Churn

Summary

Leverage subscription specific receipt fields

Implement status polling

Reducing Subscriber Churn

Summary

Leverage subscription specific receipt fields

Implement status polling

Implement customer messaging

Reducing Subscriber Churn

Summary

Leverage subscription specific receipt fields

Implement status polling

Implement customer messaging

Present contextual subscription offers

Analytics and Reporting

Pete Hare, Engineering Manager, App Store

Apple Inc.

App Store Connect Sales and Trends

Ailish Kimber
Nature Lab

- Overview

SUBSCRIPTIONS

- Summary
- Retention

SALES

- Units
- Sales
- Proceeds
- Pre-Orders

SAVED

- Sales and Trends Reports

Add Filters +

Ending on Jun 5, 2018

UTC

App Units

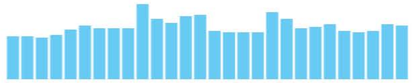
May 6 - Jun 5, 2018

16.8M ▲59%

In-App Purchases

May 6 - Jun 5, 2018

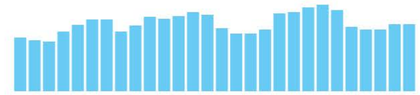
3.29M ▲16%



Sales

May 6 - Jun 5, 2018

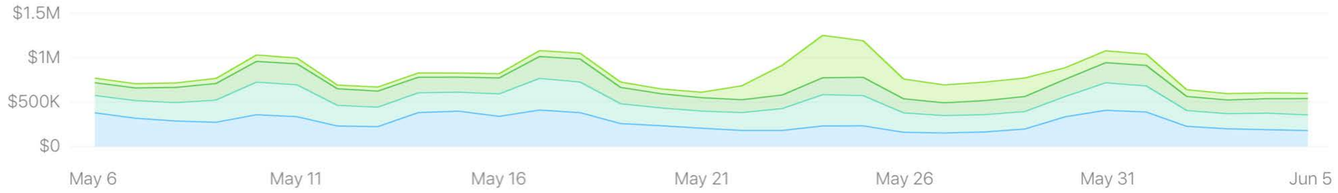
$23M ▲17%



Top Apps and Bundles

Sales

Sales per Day: May 6 - Jun 5, 2018



	Name	Apple ID	Sales	Percentage of Total	Change
1	Forest Explorer iOS	432	$8.95M	38.9%	▲ 12%
2	Mountain Climber iOS	554	$6.06M	26.3%	▲ 4.2%

Apple Inc.

App Store Connect Sales and Trends

Ailish Kimber
Nature Lab

- Overview

SUBSCRIPTIONS

- Summary
- Retention

SALES

- Units
- Sales
- Proceeds
- Pre-Orders

SAVED

- Sales and Trends Reports

Add Filters +

Ending on Jun 5, 2018

Active Paid Subscriptions

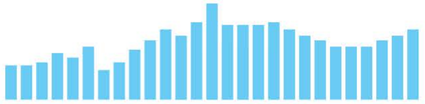
Jun 5, 2018

221K ▲ 0.48%

Daily Subscription Units

Jun 5, 2018

8.60K ▲ 1.1%



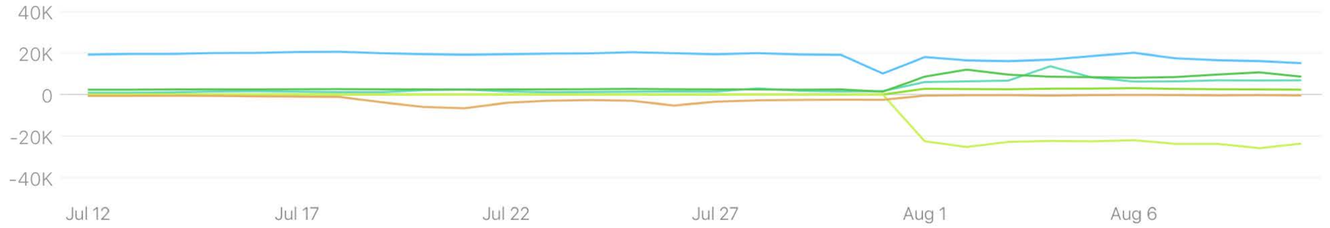
Daily Subscription Sales

Jun 5, 2018

$87.5 ▲ 1.3%

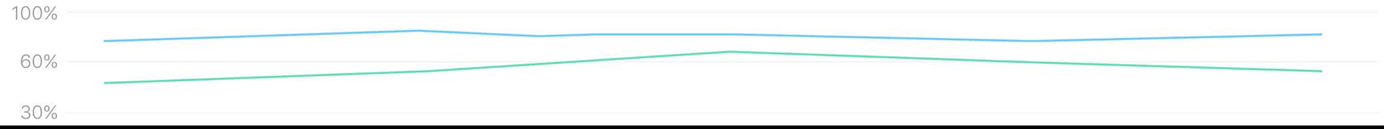
Subscription Activity

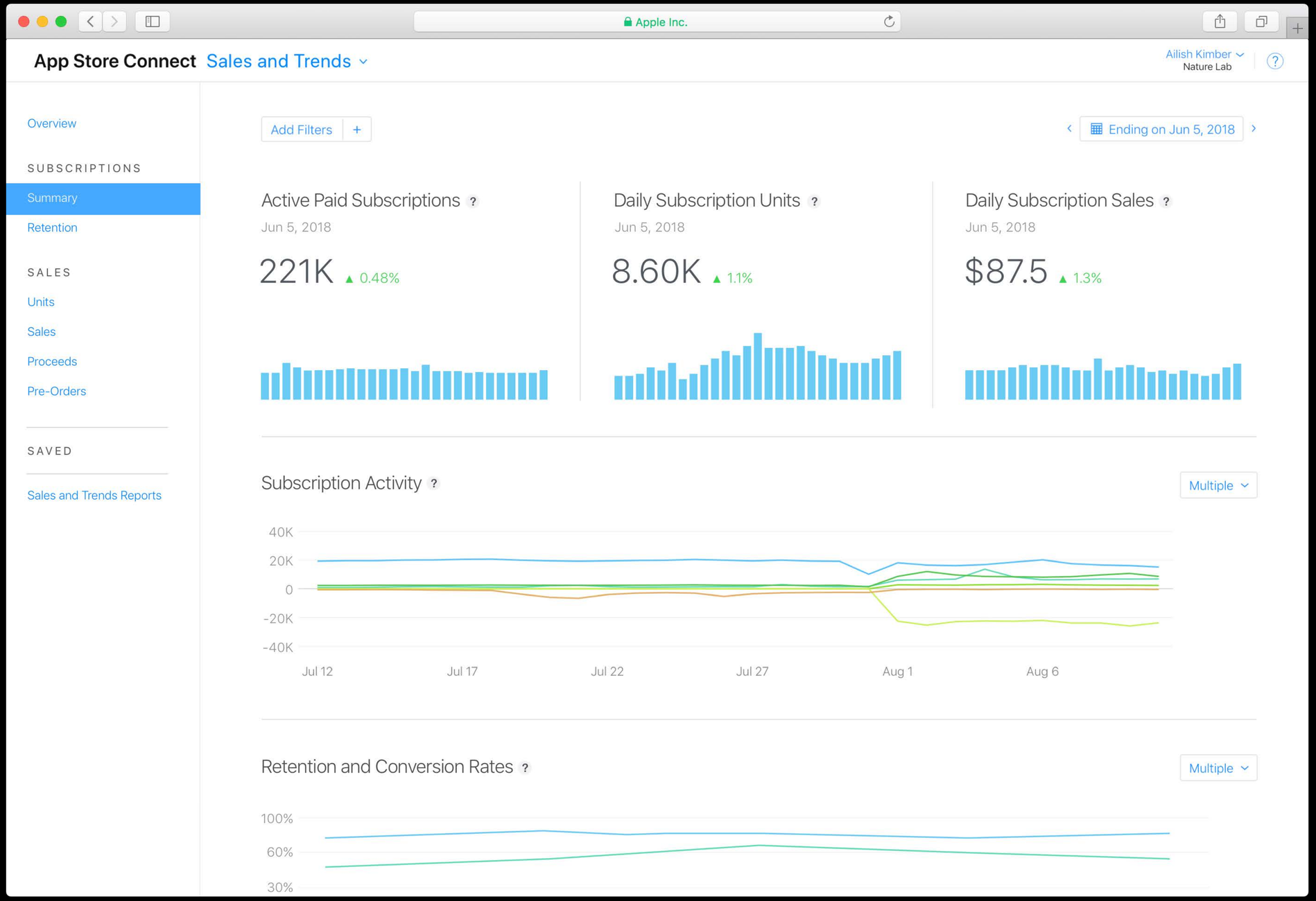
Multiple



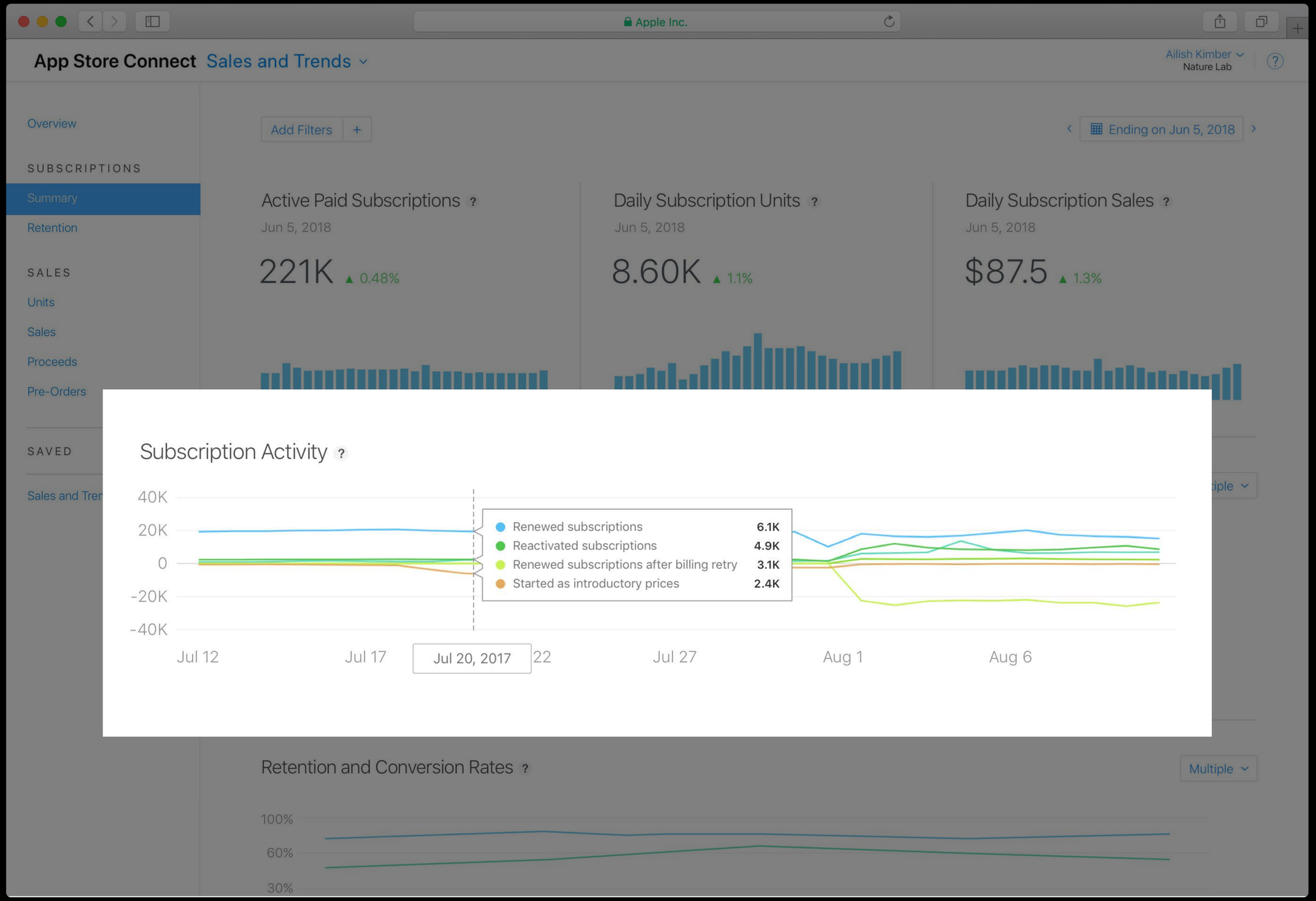
Retention and Conversion Rates

Multiple

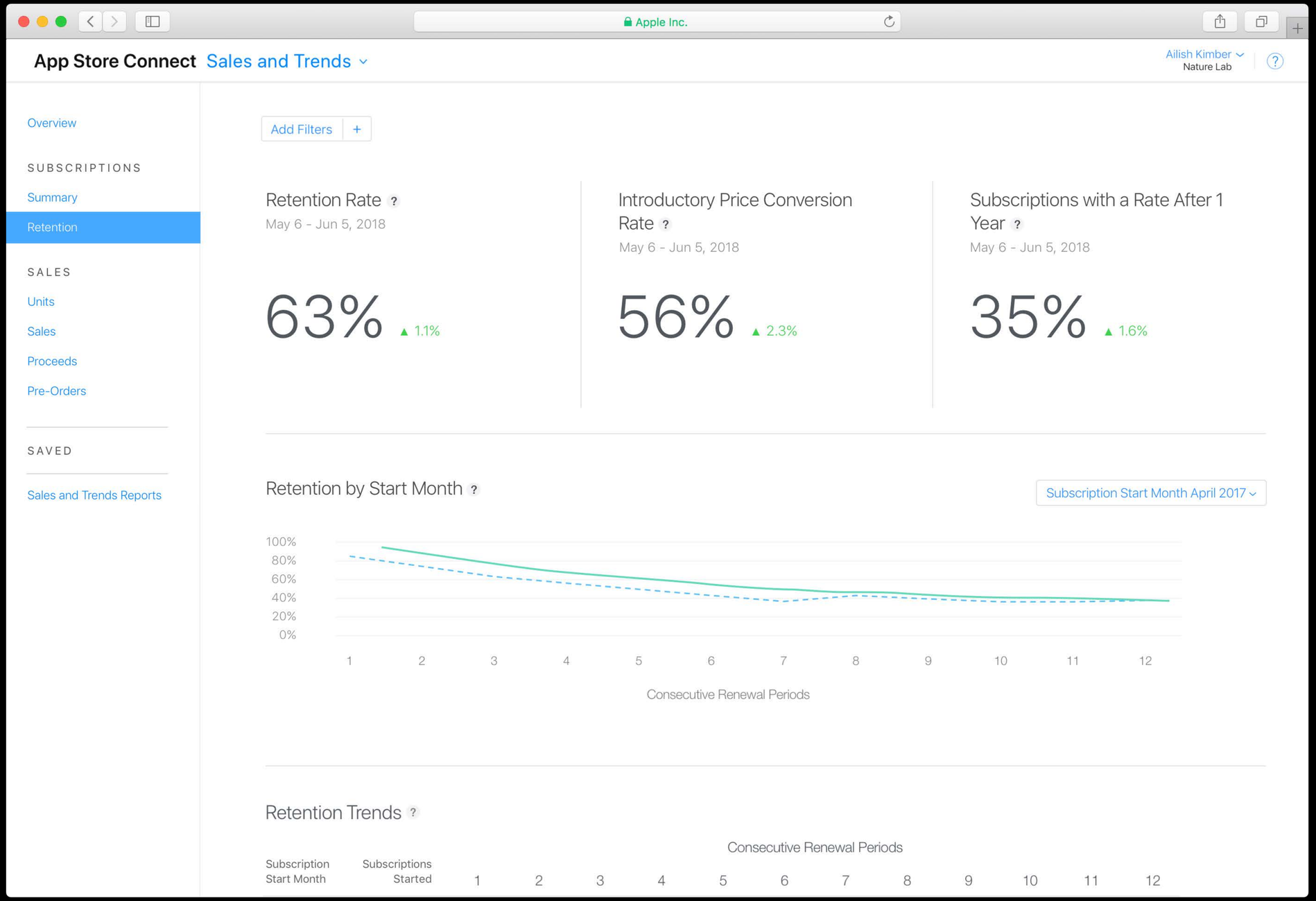



Apple Inc.
App Store Connect
Sales and Trends
Ailish Kimber
Nature Lab
Overview
SUBSCRIPTIONS
Summary
Retention
SALES
Units
Sales
Proceeds
Pre-Orders
SAVED
Sales and Trends Reports
Add Filters
+
Ending on Jun 5, 2018
Active Paid Subscriptions
Jun 5, 2018
221K
▲ 0.48%
Daily Subscription Units
Jun 5, 2018
8.60K
▲ 1.1%
Daily Subscription Sales
Jun 5, 2018
$87.5
▲ 1.3%
Subscription Activity
Multiple
40K
20K
0
-20K
-40K
Jul 12
Jul 17
Jul 22
Jul 27
Aug 1
Aug 6
Retention and Conversion Rates
Multiple
100%
60%
30%

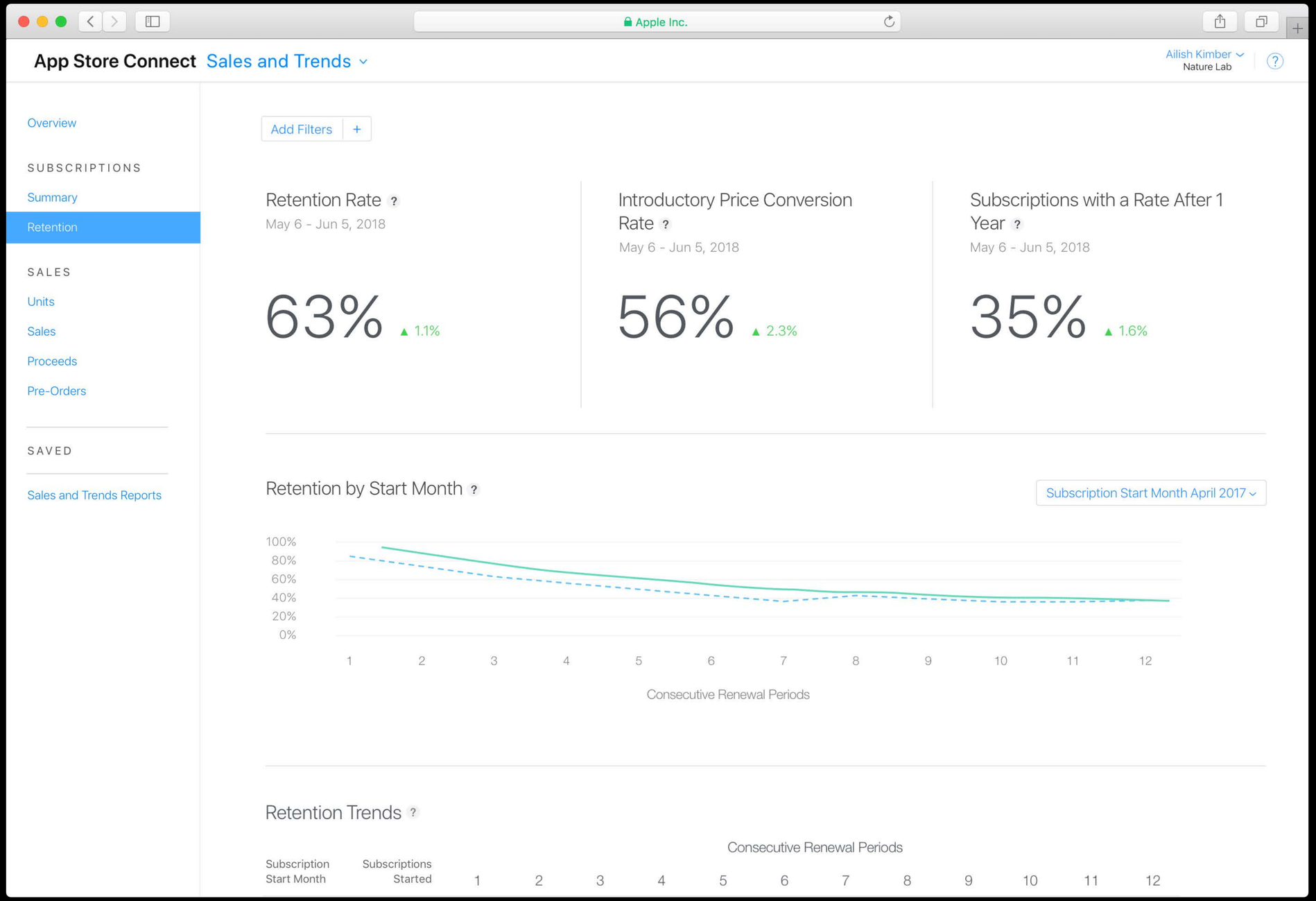
NEW


Apple Inc.
App Store Connect
Sales and Trends
Ailish Kimber
Nature Lab
Overview
SUBSCRIPTIONS
Summary
Retention
SALES
Units
Sales
Proceeds
Pre-Orders
SAVED
Sales and Trer
Add Filters
+
Ending on Jun 5, 2018
Active Paid Subscriptions
Jun 5, 2018
221K
▲ 0.48%
Daily Subscription Units
Jun 5, 2018
8.60K
▲ 1.1%
Daily Subscription Sales
Jun 5, 2018
$87.5
▲ 1.3%
Subscription Activity
40K
20K
0
-20K
-40K
Renewed subscriptions 6.1K
Reactivated subscriptions 4.9K
Renewed subscriptions after billing retry 3.1K
Started as introductory prices 2.4K
Jul 12
Jul 17
Jul 20, 2017
22
Jul 27
Aug 1
Aug 6
Retention and Conversion Rates
Multiple
100%
60%
30%

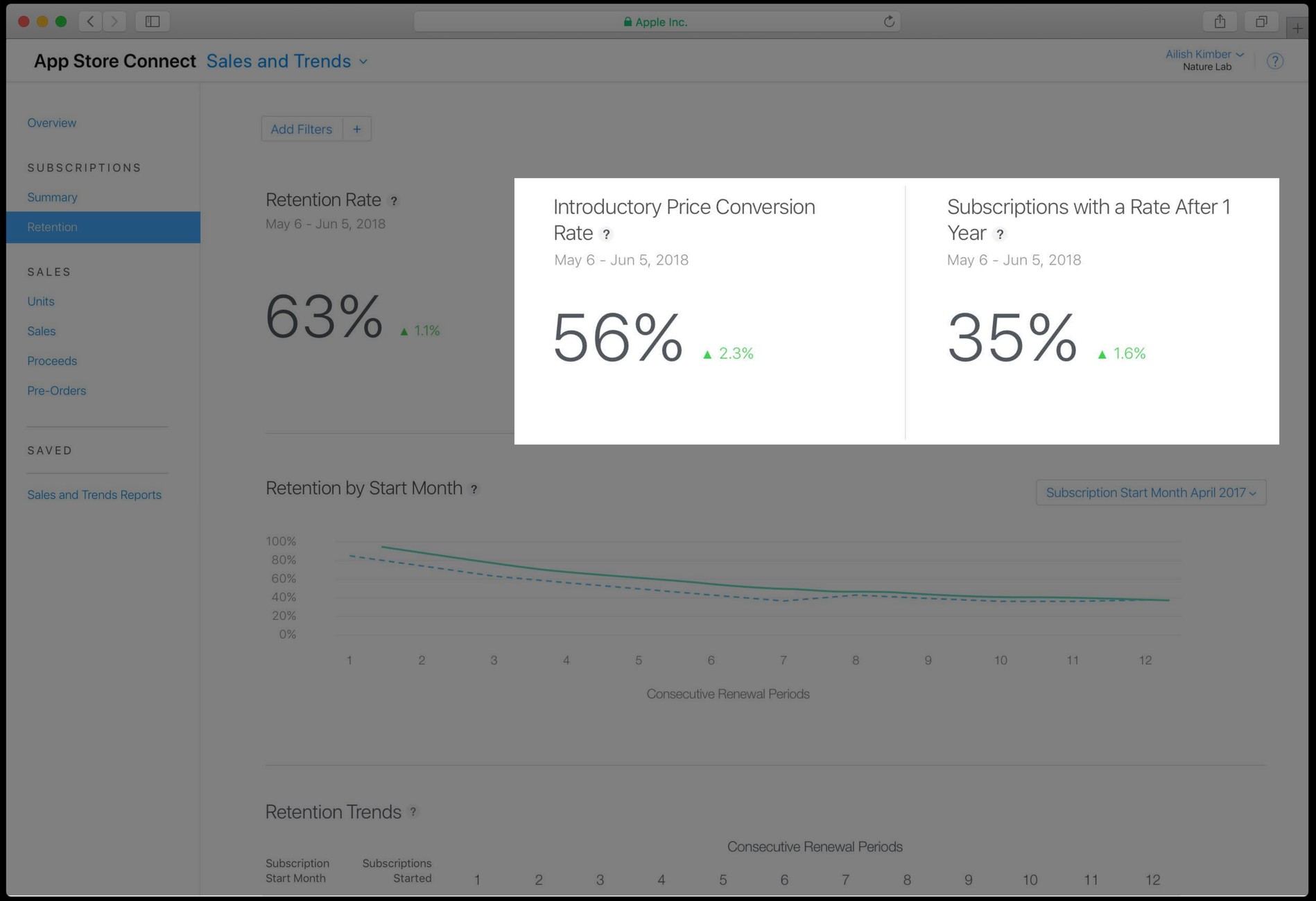
NEW


Apple Inc.
App Store Connect
Sales and Trends
Ailish Kimber
Nature Lab
Overview
SUBSCRIPTIONS
Summary
Retention
SALES
Units
Sales
Proceeds
Pre-Orders
SAVED
Sales and Trends Reports
Add Filters
+
Retention Rate
May 6 - Jun 5, 2018
63%
▲ 1.1%
Introductory Price Conversion Rate
May 6 - Jun 5, 2018
56%
▲ 2.3%
Subscriptions with a Rate After 1 Year
May 6 - Jun 5, 2018
35%
▲ 1.6%
Retention by Start Month
Subscription Start Month April 2017
100%
80%
60%
40%
20%
0%
1 2 3 4 5 6 7 8 9 10 11 12
Consecutive Renewal Periods
Retention Trends
Consecutive Renewal Periods
Subscription Start Month
Subscriptions Started
1 2 3 4 5 6 7 8 9 10 11 12

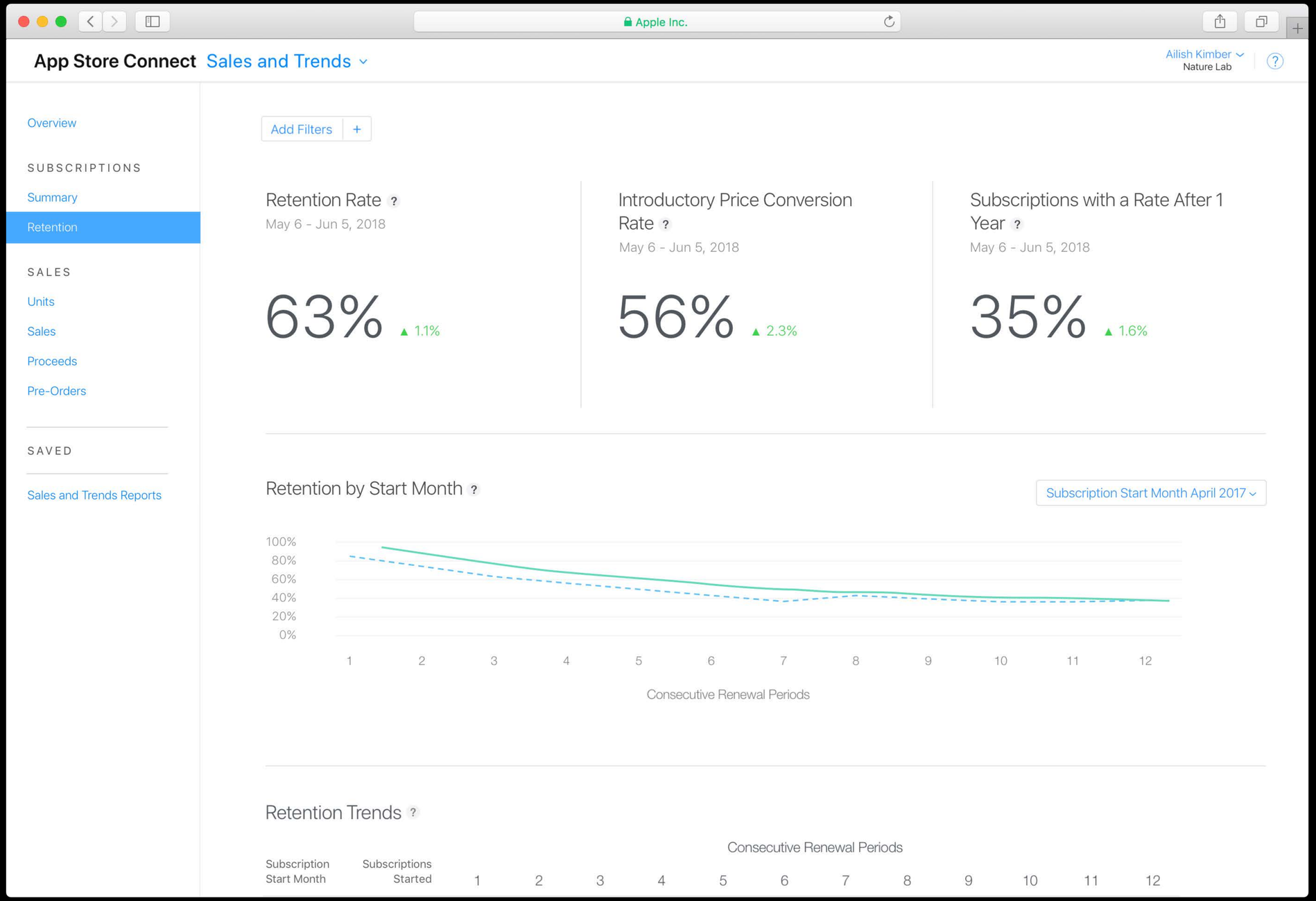
NEW


Apple Inc.
App Store Connect
Sales and Trends
Ailish Kimber
Nature Lab
Overview
SUBSCRIPTIONS
Summary
Retention
SALES
Units
Sales
Proceeds
Pre-Orders
SAVED
Sales and Trends Reports
Add Filters
+
Retention Rate
May 6 - Jun 5, 2018
63%
▲ 1.1%
Introductory Price Conversion Rate
May 6 - Jun 5, 2018
56%
▲ 2.3%
Subscriptions with a Rate After 1 Year
May 6 - Jun 5, 2018
35%
▲ 1.6%
Retention by Start Month
Subscription Start Month April 2017
100%
80%
60%
40%
20%
0%
1 2 3 4 5 6 7 8 9 10 11 12
Consecutive Renewal Periods
Retention Trends
Consecutive Renewal Periods
Subscription Start Month
Subscriptions Started
1 2 3 4 5 6 7 8 9 10 11 12

NEW


Apple Inc.
App Store Connect Sales and Trends
Ailish Kimber
Nature Lab
Overview
SUBSCRIPTIONS
Summary
Retention
SALES
Units
Sales
Proceeds
Pre-Orders
SAVED
Sales and Trends Reports
Add Filters +
Retention Rate
May 6 - Jun 5, 2018
63% ▲ 1.1%
Introductory Price Conversion Rate
May 6 - Jun 5, 2018
56% ▲ 2.3%
Subscriptions with a Rate After 1 Year
May 6 - Jun 5, 2018
35% ▲ 1.6%
Retention by Start Month
Subscription Start Month April 2017
100%
80%
60%
40%
20%
0%
1 2 3 4 5 6 7 8 9 10 11 12
Consecutive Renewal Periods
Retention Trends
Consecutive Renewal Periods
Subscription Start Month
Subscriptions Started
1 2 3 4 5 6 7 8 9 10 11 12

NEW


Apple Inc.
App Store Connect
Sales and Trends
Ailish Kimber
Nature Lab
Overview
SUBSCRIPTIONS
Summary
Retention
SALES
Units
Sales
Proceeds
Pre-Orders
SAVED
Sales and Trends Reports
Add Filters
+
Retention Rate
May 6 - Jun 5, 2018
63%
▲ 1.1%
Introductory Price Conversion Rate
May 6 - Jun 5, 2018
56%
▲ 2.3%
Subscriptions with a Rate After 1 Year
May 6 - Jun 5, 2018
35%
▲ 1.6%
Retention by Start Month
Subscription Start Month April 2017
100%
80%
60%
40%
20%
0%
1 2 3 4 5 6 7 8 9 10 11 12
Consecutive Renewal Periods
Retention Trends
Consecutive Renewal Periods
Subscription Start Month
Subscriptions Started
1 2 3 4 5 6 7 8 9 10 11 12

NEW

NEW

Apple Inc.

App Store Connect Sales and Trends

Ailish Kimber
Nature Lab

Overview

SUBSCRIPTIONS
Summary
Retention

SALES
Units
Sales
Proceeds
Pre-Orders

SAVED
Sales and Tren

Add Filters +

Retention Rate ?
May 6 - Jun 5, 2018
63% ▲ 1.1%

Introductory Price Conversion Rate ?
May 6 - Jun 5, 2018
56% ▲ 2.3%

Subscriptions with a Rate After 1 Year ?
May 6 - Jun 5, 2018
35% ▲ 1.6%

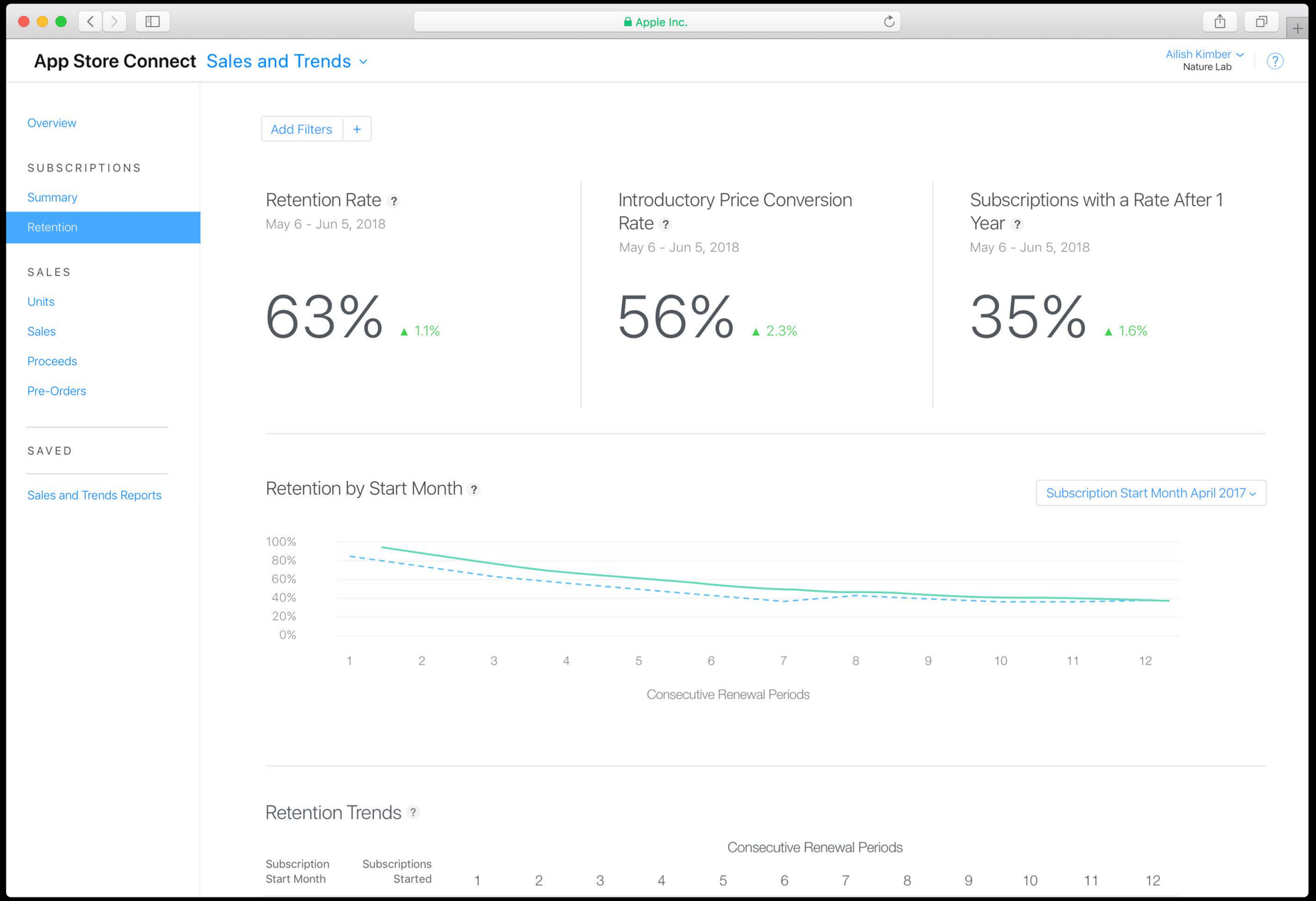
Retention by Start Month ?

Subscription Start Month April 2017

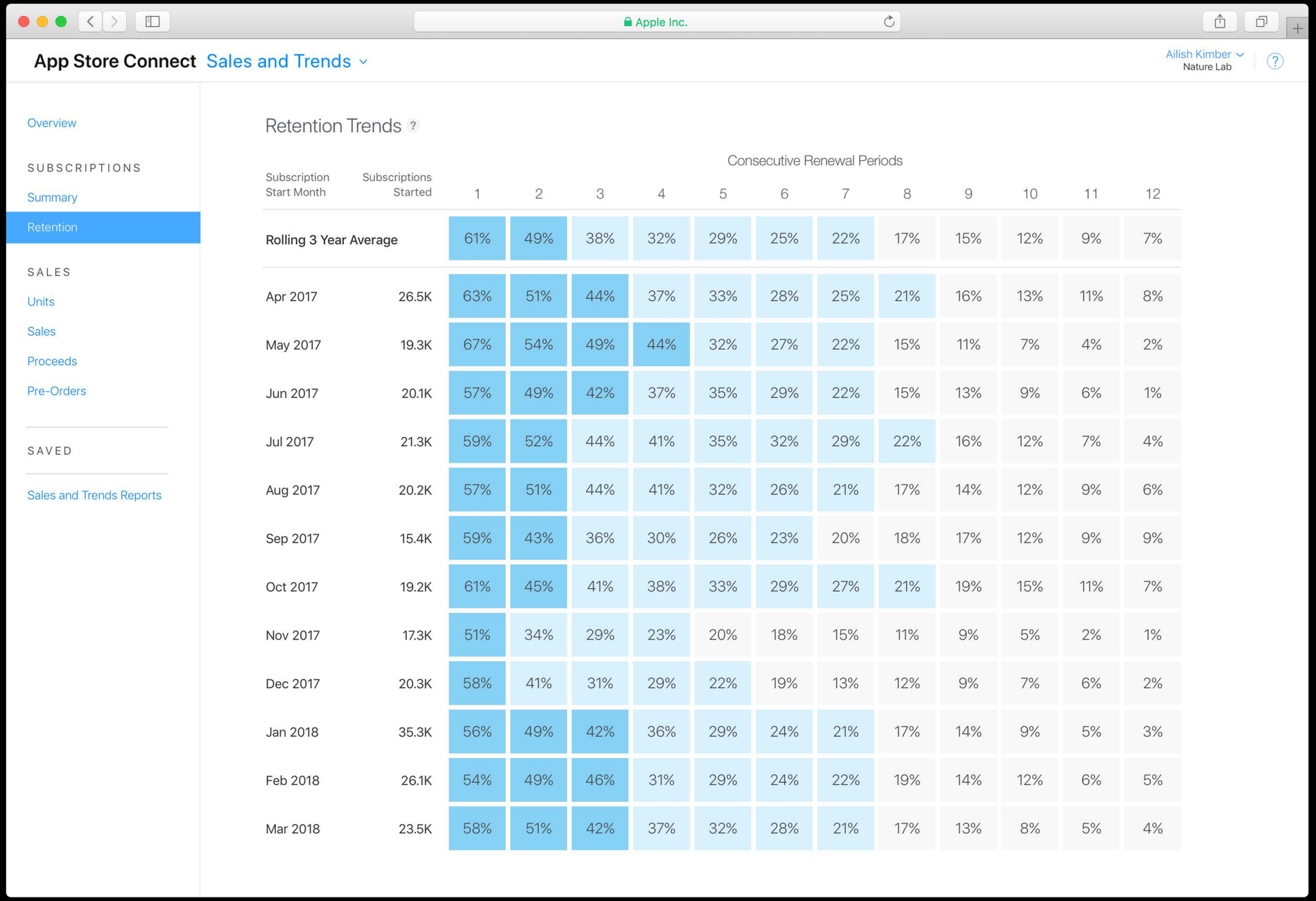
Retention Trends ?

Consecutive Renewal Periods

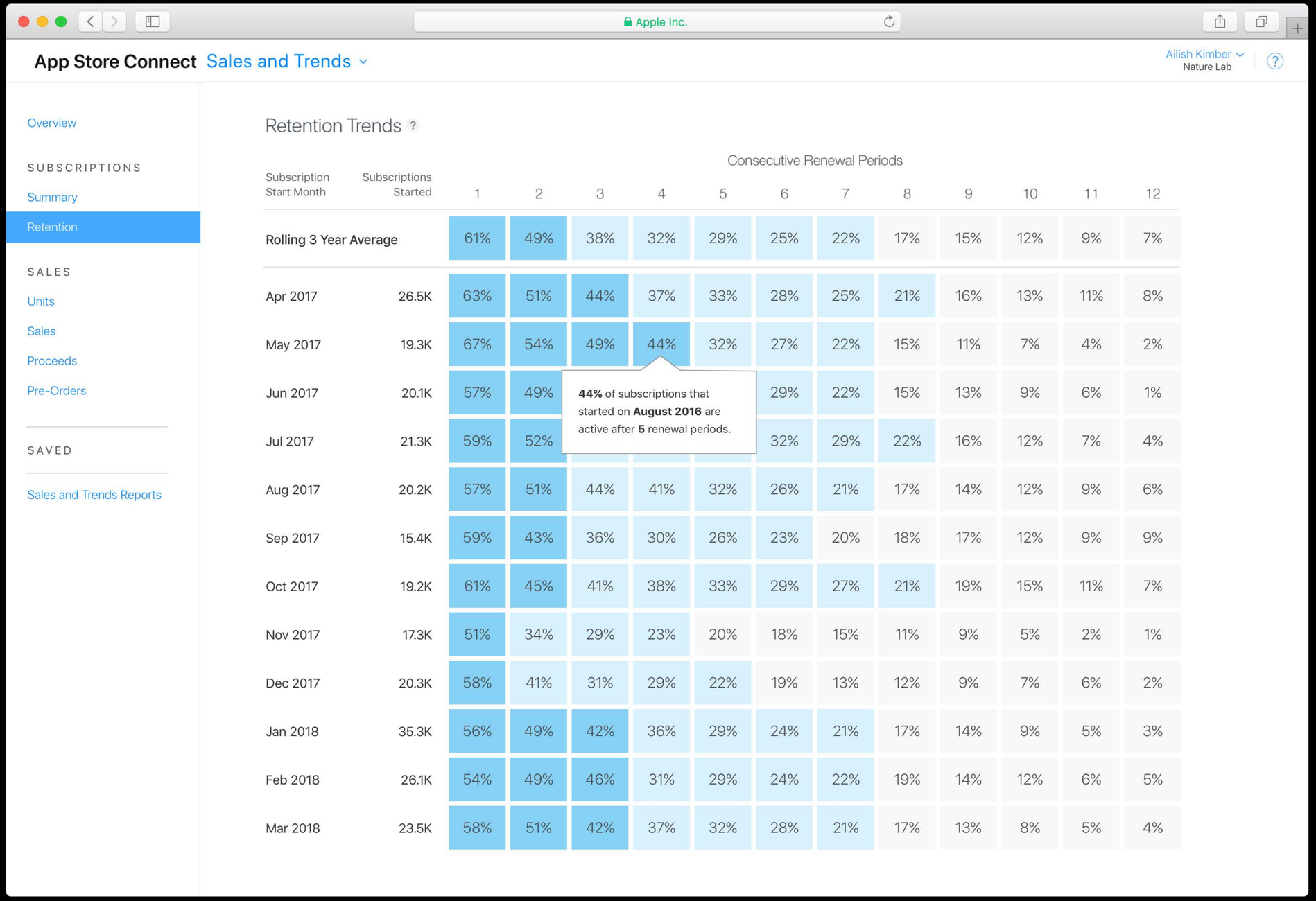
Subscription Start Month	Subscriptions Started	1	2	3	4	5	6	7	8	9	10	11	12


Apple Inc.
App Store Connect
Sales and Trends
Ailish Kimber
Nature Lab
Overview
SUBSCRIPTIONS
Summary
Retention
SALES
Units
Sales
Proceeds
Pre-Orders
SAVED
Sales and Trends Reports
Add Filters
+
Retention Rate
May 6 - Jun 5, 2018
63%
▲ 1.1%
Introductory Price Conversion Rate
May 6 - Jun 5, 2018
56%
▲ 2.3%
Subscriptions with a Rate After 1 Year
May 6 - Jun 5, 2018
35%
▲ 1.6%
Retention by Start Month
Subscription Start Month April 2017
100%
80%
60%
40%
20%
0%
1 2 3 4 5 6 7 8 9 10 11 12
Consecutive Renewal Periods
Retention Trends
Consecutive Renewal Periods
Subscription Start Month
Subscriptions Started
1 2 3 4 5 6 7 8 9 10 11 12

NEW


Apple Inc.
App Store Connect
Sales and Trends
Ailish Kimber
Nature Lab
Overview
SUBSCRIPTIONS
Summary
Retention
SALES
Units
Sales
Proceeds
Pre-Orders
SAVED
Sales and Trends Reports
Retention Trends
Consecutive Renewal Periods
Subscription Start Month
Subscriptions Started
1 2 3 4 5 6 7 8 9 10 11 12
Rolling 3 Year Average 61% 49% 38% 32% 29% 25% 22% 17% 15% 12% 9% 7%
Apr 2017 26.5K 63% 51% 44% 37% 33% 28% 25% 21% 16% 13% 11% 8%
May 2017 19.3K 67% 54% 49% 44% 32% 27% 22% 15% 11% 7% 4% 2%
Jun 2017 20.1K 57% 49% 42% 37% 35% 29% 22% 15% 13% 9% 6% 1%
Jul 2017 21.3K 59% 52% 44% 41% 35% 32% 29% 22% 16% 12% 7% 4%
Aug 2017 20.2K 57% 51% 44% 41% 32% 26% 21% 17% 14% 12% 9% 6%
Sep 2017 15.4K 59% 43% 36% 30% 26% 23% 20% 18% 17% 12% 9% 9%
Oct 2017 19.2K 61% 45% 41% 38% 33% 29% 27% 21% 19% 15% 11% 7%
Nov 2017 17.3K 51% 34% 29% 23% 20% 18% 15% 11% 9% 5% 2% 1%
Dec 2017 20.3K 58% 41% 31% 29% 22% 19% 13% 12% 9% 7% 6% 2%
Jan 2018 35.3K 56% 49% 42% 36% 29% 24% 21% 17% 14% 9% 5% 3%
Feb 2018 26.1K 54% 49% 46% 31% 29% 24% 22% 19% 14% 12% 6% 5%
Mar 2018 23.5K 58% 51% 42% 37% 32% 28% 21% 17% 13% 8% 5% 4%

NEW


Apple Inc.
App Store Connect
Sales and Trends
Ailish Kimber
Nature Lab
Overview
SUBSCRIPTIONS
Summary
Retention
SALES
Units
Sales
Proceeds
Pre-Orders
SAVED
Sales and Trends Reports
Retention Trends
Consecutive Renewal Periods
Subscription Start Month
Subscriptions Started
1 2 3 4 5 6 7 8 9 10 11 12
Rolling 3 Year Average 61% 49% 38% 32% 29% 25% 22% 17% 15% 12% 9% 7%
Apr 2017 26.5K 63% 51% 44% 37% 33% 28% 25% 21% 16% 13% 11% 8%
May 2017 19.3K 67% 54% 49% 44% 32% 27% 22% 15% 11% 7% 4% 2%
Jun 2017 20.1K 57% 49% 29% 22% 15% 13% 9% 6% 1%
Jul 2017 21.3K 59% 52% 32% 29% 22% 16% 12% 7% 4%
Aug 2017 20.2K 57% 51% 44% 41% 32% 26% 21% 17% 14% 12% 9% 6%
Sep 2017 15.4K 59% 43% 36% 30% 26% 23% 20% 18% 17% 12% 9% 9%
Oct 2017 19.2K 61% 45% 41% 38% 33% 29% 27% 21% 19% 15% 11% 7%
Nov 2017 17.3K 51% 34% 29% 23% 20% 18% 15% 11% 9% 5% 2% 1%
Dec 2017 20.3K 58% 41% 31% 29% 22% 19% 13% 12% 9% 7% 6% 2%
Jan 2018 35.3K 56% 49% 42% 36% 29% 24% 21% 17% 14% 9% 5% 3%
Feb 2018 26.1K 54% 49% 46% 31% 29% 24% 22% 19% 14% 12% 6% 5%
Mar 2018 23.5K 58% 51% 42% 37% 32% 28% 21% 17% 13% 8% 5% 4%
44% of subscriptions that started on August 2016 are active after 5 renewal periods.

NEW

App Store Connect Reports

NEW

App Store Connect Reports

NEW

Available through the new App Store Connect API

App Store Connect Reports

NEW

Available through the new App Store Connect API

Report data available daily

App Store Connect Reports

NEW

Available through the new App Store Connect API

Report data available daily

Can be imported for further analysis

App Store Connect Reports

NEW

Available through the new App Store Connect API

Report data available daily

Can be imported for further analysis

Automating App Store Connect | Hall 3 | Thursday 3:00PM

Receipts vs Reports

Receipts vs Reports

App Store Receipts

- Validating StoreKit transactions
- Updating user subscription state
- Understand individual subscriber behavior

Receipts vs Reports

App Store Receipts

- Validating StoreKit transactions
- Updating user subscription state
- Understand individual subscriber behavior

App Store Connect Reports

- Aggregated sales activity
- Subscriptions activity
- Financial payments

Summary

Summary

Server side state management offers more flexibility

Summary

Server side state management offers more flexibility

Use notifications from the App Store

Summary

Server side state management offers more flexibility

Use notifications from the App Store

Offer introductory pricing

Summary

Server side state management offers more flexibility

Use notifications from the App Store

Offer introductory pricing

Reduce churn with simple messaging

Summary

Server side state management offers more flexibility

Use notifications from the App Store

Offer introductory pricing

Reduce churn with simple messaging

Win back users with alternative subscription options

Summary

Server side state management offers more flexibility

Use notifications from the App Store

Offer introductory pricing

Reduce churn with simple messaging

Win back users with alternative subscription options

New reporting tools in App Store Connect

More Information

https://developer.apple.com/wwdc18/705

StoreKit / Subscriptions Lab	Technology Lab 12	Tuesday 4:00PM
StoreKit / Subscriptions Lab	Technology Lab 12	Thursday 9:00AM


WWDC18