In-App Purchases

In-App Purchases

Preparing

Paid Applications Agreement

If you haven't already, you'll need to sign the Paid Applications Agreement and set up your banking and tax information in iTunes Connect.

iTunes Connect Developer Help: Agreements, tax, and banking overview

Create Your In-App Purchases

Then, you'll need to configure your in-app purchases in iTunes Connect, and include details such as name, pricing, and description that highlights the features and functionality of your in-app purchase.

iTunes Connect Developer Help: Create an in-app purchase

Change the CFBundleIdentifier

To test In-App Purchase in development with Electron you'll have to change the `CFBundleIdentifier` in `node_modules/electron/dist/Electron.app/Contents/Info.plist`. You have to replace `com.github.electron` by the bundle identifier of the application you created with iTunes Connect.

```
<key>CFBundleIdentifier</key>
<string>com.example.app</string>
```

Code example

Here is an example that shows how to use In-App Purchases in Electron. You'll have to replace the product ids by the identifiers of the products created with iTunes Connect (the identifier of `com.example.app.product1` is `product1`). Note that you have to listen to the `transactions-updated` event as soon as possible in your app.

```
// Main process
const { inAppPurchase } = require('electron')
const PRODUCT_IDS = ['id1', 'id2']

// Listen for transactions as soon as possible.
inAppPurchase.on('transactions-updated', (event, transactions) => {
  if (!Array.isArray(transactions)) {
    return
  }

  // Check each transaction.
  transactions.forEach((transaction) => {
    const payment = transaction.payment

    switch (transaction.transactionState) {
      case 'purchasing':
        console.log(`Purchasing ${payment.productIdentifier}...`)
        break

      case 'purchased': {
        console.log(`${payment.productIdentifier} purchased.`)

        // Get the receipt url.
        const receiptURL = inAppPurchase.getReceiptURL()

        console.log(`Receipt URL: ${receiptURL}`)

        // Submit the receipt file to the server and check if it is valid.
        // @see https://developer.apple.com/library/content/releasenotes/General/ValidateApp
        // ...
        // If the receipt is valid, the product is purchased
        // ...

        // Finish the transaction.
        inAppPurchase.finishTransactionByDate(transaction.transactionDate)

        break
      }

      case 'failed':

        console.log(`Failed to purchase ${payment.productIdentifier}.`)

        // Finish the transaction.
        inAppPurchase.finishTransactionByDate(transaction.transactionDate)

        break
      case 'restored':
        console.log(`The purchase of ${payment.productIdentifier} has been restored.`)

        break
      case 'deferred':

        console.log(`The purchase of ${payment.productIdentifier} has been deferred.`)

        break
      default:
        break
    }
  })
})

// Check if the user is allowed to make in-app purchase.
if (!inAppPurchase.canMakePayments()) {
  console.log('The user is not allowed to make in-app purchase.')
}

// Retrieve and display the product descriptions.
inAppPurchase.getProducts(PRODUCT_IDS).then(products => {
  // Check the parameters.
  if (!Array.isArray(products) || products.length <= 0) {
    console.log('Unable to retrieve the product informations.')
    return
  }

  // Display the name and price of each product.
  products.forEach(product => {
    console.log(`The price of ${product.localizedTitle} is ${product.formattedPrice}.`)
  })

  // Ask the user which product they want to purchase.
  const selectedProduct = products[0]
  const selectedQuantity = 1

  // Purchase the selected product.
  inAppPurchase.purchaseProduct(selectedProduct.productIdentifier, selectedQuantity).then(is
    if (!isProductValid) {
      console.log('The product is not valid.')
      return
    }

    console.log('The payment has been added to the payment queue.')
  })
```

```
})
```