Рутокен для мобильных приложений на C#

RutokenPkcs11Interop – расширение библиотеки Pkcs11Interop для работы через интерфейс PKCS#11 с устройствами семейства Рутокен ЭЦП. Поддерживает ГОСТ-2012, работу с CMS-форматом, запросами на сертификаты PKCS#10 и многое другое.

Для демонстрации работы в нашем аккаунте на GitHub размещены примеры приложений на Xamarin для iOS и Android. Их решение находятся в файле Xamarin.Samples.sln.

Фреймворк Xamarin позволяет удобно разрабатывать кроссплатформенные C# приложения за счет разделения внутренней логики и особенностей мобильных платформ.

Особенности сборки для Android

Добавьте в ваш проект PKCS#11 библиотеку. Для этого просто подключите к вашему проекту NuGet пакет Aktiv.RtPkcs11Ecp.Natives.Android.

Для работы с PKCS#11 вам также необходимо добавить в проект две библиотеки: *rtserviceconnection.aar* и *pkcs11jna.jar.* Их можно взять из Рутокен SDK в *директориях sdk\mobile\android\libs* и *sdk\java\samples\lib.* Для того, чтобы добавить их в свой проект опишите опишите их в файле проекта:

добавление jar и aar бибилотек в проект

```
<ItemGroup>
  <AndroidAarLibrary Include="Jars\rtserviceconnection-*.aar" />
  <AndroidJavaLibrary Include="Jars\pkcs11jna-*.jar" />
</ItemGroup>
```

В этом примере мы положили библиотеки в папку *Jars*.

Помимо jar и aar библиотек вам нужно добавить в свой проект саму pkcs#11 библиотеку.

Особенности сборки для iOS

Минимальная настройка

Добавьте в ваш проект PKCS#11 библиотеку. Для этого просто подключите к вашему проекту nuget пакет Aktiv.RtPkcs11Ecp.Natives.iOS. Также в файл Info.plist добавьте строчки:

Info.plist

```
<key>UISupportedExternalAccessoryProtocols</key>
<array>
  <string>com.aktivco.rutokenecp</string>
</array>
```

Этого будет достаточно для работы с Рутокен ЭЦП Bluetooth.

Добавление поддержки устройств с NFC

Для того, чтобы ваше приложение умело работать также и с Рутокен ЭЦП 3.0 NFC:

1. Добавьте в файл Info.plist строчки:

Info.plist

```
<key>NFCReaderUsageDescription</key>
<string>Allow NFC scanning</string>
<key>com.apple.developer.nfc.readersession.iso7816.select-identifiers</key>
<array>
    <string>F000000000100000000000100</string>
    <string>A0000001510000000</string>
    <string>A00000039742544659</string>
</array>
```

2. В файл Entitlements.plist добавьте строчки:

Entitlements.plist

```
<key>com.apple.developer.nfc.readersession.formats</key>
<array>
    <string>NDEF</string>
    <string>TAG</string>
</array>
```

Сертификат разработчика iOS

Убедитесь, что ваш сертификат разработчика для iOs позволяет разрабатывать приложения с использованием NFC меток.

3. Перед началом взаимодействия с Рутокен ЭЦП 3.0 NFC запустите функцию *startNFC*. Функция определена во фреймворке RtPcsc, который можно взять из Рутокен SDK в директории *sdk\mobile\ios\pcsc\lib*.
4. Функция *startNFC* запускает **в отдельном потоке** окно с просьбой приложить NFC карту к телефону или планшету. На вход она принимает callback, который будет вызван в случае ошибок, например если окно закрылось по таймауту или пользователь нажал на клавишу "Отмена".

   Напрямую вызвать эти функции не получится т.к. они имеют Obj-C сигнатуру и в качестве callback принимает литеральный блок. Но это ограничение можно обойти воспользовавшись Xamarin оберткой над блоками, например, как это сделано здесь. Подробнее о том, как писать такие обертки написано здесь.
5. После окончания взаимодействия с NFC токеном запустите функцию *stopNFC* из фреймворка RtPcsc.
6. Поток с окном предложения приложить токен и поток работы с PKCS#11 функциями надо синхронизировать: токен не сразу распознается системой и нужно некоторое время подождать прежде чем начать работать с ним. У нас в коде это сделано с помощью функции C_WaitForSlotEvent и **атомарный** флаг остановки работы с nfc.