# Рутокен для мобильных приложений на iOS



RtPcsc - фреймворк для работы с Рутокен ЭЦП Bluetooth, виртуальными считывателями Рутокен VCR, а также NFC-устройствами семейства Рутокен ЭЦП.

## Требования

Приложения со встроенным RtPcsc.framework собираются с iOS SDK 13 и новее и запускаются на устройствах с iOS 13 и выше.

Для работы с PKCS#11 функциями добавьте в ваше приложение фреймворк rtpkcs11ecp.framework. Он находится в Рутокен SDK в директории *sdk\mobile\ios\pkcs11\lib*.

## Встраивание

**Внимание!** Приложение со встроенным RtPcsc.framework может быть запущено только на физических устройствах Apple, не на эмуляторе. Обратите внимание, что работа с VCR API доступна только на iPad.

## Для работы с NFC устройствами Рутокен необходимо

- в Info.plist добавить ключ ISO7816 application identifiers for NFC Tag Reader Session со значениями:

**Info.plist**

```
<key>com.apple.developer.nfc.readersession.iso7816.select-identifiers</key>
<array>
    <string>F0000000005275746F6B656E</string>
    <string>A00000039742544659</string>
</array>
```

- в Info.plist добавить ключ NFCReaderUsageDescription с описанием причины необходимости доступа, например: Доступ необходим для работы с NFC

**Info.plist**

```
<key>NFCReaderUsageDescription</key>
<string>Allow NFC scanning</string>
```

- в Entitlements.plist добавить ключ com.apple.developer.nfc.readersession.formats со значением:

**Entitlements.plist**

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

**Внимание!** Убедитесь, что ваш сертификат разработчика для iOS позволяет разрабатывать приложения для работы с NFC устройствами.

## Для работы с виртуальными считывателями (Рутокен VCR) необходимо

1. в Info.plist добавить ключ Bonjour services со значениями:

   **Entitlements.plist**

   ```
   <key>NSBonjourServices</key>
   <array>
       <string>_ru-rutoken-vcr._udp</string>
       <string>_ru-rutoken-vcr._tcp</string>
   </array>
   ```

2. в Info.plist добавить ключ Privacy - Local Network Usage Description с описанием причины необходимости доступа, например: Доступ необходим для работы с VCR

   **Entitlements.plist**

   ```
   <key>NSLocalNetworkUsageDescription</key>
   <string>       </string>
   ```

3. Работа в приложениях с VCR API на macOS в режиме совместимости с iPad
   Чтобы в приложении для iPadOS, когда оно запускается в режиме совместимости на macOS, работал VCR API необходимо:

   Добавить в Entitlements.plist ключ keychain-access-groups.

Примеры на GitHub

Примеры готовых приложений можно найти в репозиториях на GitHub: rutoken-demobank-ios и rutoken-demoshift-ios.

## Для работы с USB-токенами необходимо

Начиная с iOS/iPadOS **16.2** стала возможна работа USB-токенов.
Для включения поддержки USB-токенов достаточно только обновления **RtPcsc.framework до версии 4.0.0 или новее.**

## Описание интерфейса RtPcsc для работы с NFC/VCR

Изменилось поведение функции SCardGetStatusChange

Начиная с версии RtPcsc 5.0.0 для обнаружения нового считывателя с помощью "\\?PnP?\Notification", необходимо указывать количество уже известных приложению считывателей в верхнем слове dwCurrentState.

Например, так:

```
SCARD_READERSTATE state;
state.szReader = "\\\\?PnP?\\Notification";
state.dwCurrentState = (1 << 16);
```

Если ожидаемое приложением количество считывателей не совпадет с реальным, управление будет сразу же возвращено, а в верхнем слове dwEventState будет содержаться текущее количество считывателей.

Пример работы с функцией SCardGetStatusChange, учитывающий поведение описанное на GitHub.

1. Перед началом взаимодействия с Рутокен ЭЦП 3.0 NFC запустите функцию *startNFC*. Функция определена во фреймворке RtPcsc.
2. 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".
3. После окончания взаимодействия с NFC токеном запустите функцию *stopNFC* из фреймворка RtPcsc.
4. 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.

### Обнаружение Рутокена с NFC

Для начала работы с NFC устройствами Рутокен нужно перечислить доступные считыватели с помощью функции SCardListReaders. При этом для iPhone будет доступен встроенный NFC считыватель с именем Device Internal NFC-Reader.

После этого необходимо вызвать функцию SCardConnect для этого считывателя, указав dwShareMode == SCARD_SHARE_DIRECT. С полученным SCARDHANDLE вызвать функцию SCardControl с параметром RUTOKEN_CONTROL_CODE_START_NFC.

Для завершения работы с NFC устройствами Рутокен нужно вызвать функцию SCardControl с параметром RUTOKEN_CONTROL_CODE_STOP_NFC и далее вызвать функцию SCardDisconnect.

Описание параметров функции SCardControl:

```
LONG SCardControl(
  [in]  SCARDHANDLE hCard,
  [in]  DWORD       dwControlCode,
  [in]  LPCVOID     pbSendBuffer,
  [in]  DWORD       cbSendLength,
  [out] LPVOID      pbRecvBuffer,
  [in]  DWORD       cbRecvLength,
  [out] LPDWORD     lpBytesReturned
)
```

Параметр **dwControlCode** отвечает за тип операции, которую необходимо выполнить, и может принимать следующие значения:

- RUTOKEN_CONTROL_CODE_START_NFC - запуск обнаружения по NFC;
- RUTOKEN_CONTROL_CODE_STOP_NFC - остановка обнаружения по NFC;
- RUTOKEN_CONTROL_CODE_STOP_NFC_WITH_ERROR - остановка обнаружения по NFC с показом иконки ошибки;
- RUTOKEN_CONTROL_CODE_LAST_NFC_STOP_REASON - возврат причины прекращения обнаружения по NFC.

Параметр **pbSendBuffer** используется для передачи дополнительной информации:

- при RUTOKEN_CONTROL_CODE_START_NFC: параметр задан в формате `"\(waitMessage)\0\(workMessage)\0\0"` и содержит два сообщения:
  - **waitMessage** отображается во время ожидания карты,
  - **workMessage** отображается во время работы с картой;
- при RUTOKEN_CONTROL_CODE_STOP_NFC: параметр содержит сообщение о завершении работы с картой;
- при RUTOKEN_CONTROL_CODE_STOP_NFC_WITH_ERROR: параметр содержит сообщение о завершении работы с картой с уведомлением об ошибке;
- при RUTOKEN_CONTROL_CODE_LAST_NFC_STOP_REASON: параметр не используется.

## Рекомендуемый порядок работы с Рутокеном с NFC

- получить список доступных ридеров с помощью функции SCardListReaders;
- если ридер еще не появился – дождаться его появления с помощью функции SCardGetStatusChange;
- вызов функции SCardConnect для нужного ридера с параметром dwShareMode == SCARD_SHARE_DIRECT;
- вызов функции SCardControl с параметром RUTOKEN_CONTROL_CODE_START_NFC;
- работа с Рутокеном;
- вызов функции SCardControl с параметром RUTOKEN_CONTROL_CODE_STOP_NFC;
- вызов функции SCardDisconnect.

Причину завершения обнаружения NFC устройств можно получить с помощью вызова функции SCardControl с параметром RUTOKEN_CONTROL_CODE_LAST_NFC_STOP_REASON.

Возможные причины:

- RUTOKEN_NFC_STOP_REASON_FINISHED - вызов SCardControl с параметром RUTOKEN_CONTROL_CODE_STOP_NFC;
- RUTOKEN_NFC_STOP_REASON_UNKNOWN - причина завершения неизвестна;
- RUTOKEN_NFC_STOP_REASON_TIMEOUT - системный таймаут (на устройствах под управлением iOS предоставляется 20 секунд на одну NFC сессию);
- RUTOKEN_NFC_STOP_REASON_CANCELLED_BY_USER - нажатие кнопки "Отмена" на системном окне обнаружения NFC;
- RUTOKEN_NFC_STOP_REASON_NO_ERROR - системное окно работы с NFC еще не опускалось, ошибок нет.

## Получение типа устройства Рутокен

Получение типа устройства Рутокен доступно с помощью функции:

```
LONG SCardGetAttrib (
  [in]      SCARDHANDLE hCard,
  [in]      DWORD       dwAttrId,
  [out]     LPBYTE      pbAttr,
  [in, out] LPDWORD     pcbAttrLen
)
```

В качестве параметра dwAttrId необходимо передать SCARD_ATTR_VENDOR_IFD_TYPE.

Возможные типы:

- RUTOKEN_UNKNOWN_TYPE
- RUTOKEN_BT_TYPE
- RUTOKEN_NFC_TYPE
- RUTOKEN_VCR_TYPE

## Работа с виртуальным считывателем

Для запуска процедуры создания пары с новым виртуальным считывателем необходимо вызвать функцию **NSString* generatePairingQR(void)**, она возвращает изображение QR-кода в виде base64 строки. Для этого необходимо отобразить QR-код на экране и считать его с помощью приложения Рутокен VCR.

Функция **NSString* generatePairingQR(void)** служит для создания сертификата и временного секрета для сопряжения. Процедура сопряжения начнется после вызова хотя бы одной функции из интерфейса RtPcsc.framework.

Для получения списка сопряженных считывателей необходимо вызвать функцию **NSArray* listPairedVCR(void).**
Функция возвращает массив словарей, содержащих информацию о считывателях. Для каждого считывателя Рутокен VCR существует один сертификат, который хранится в keychain. Он необходим, чтобы удостовериться, что данные устройства (iPad и iPhone) сопряжены.

Ключи словаря:

- name - имя считывателя
- cert - сертификат считывателя в виде BASE64-строки
- fingerprint - SHA1-хеш от сертификата считывателя

Для разрыва пары со считывателем необходимо вызвать функцию **BOOL unpairVCR(NSData* vcrId).**

Функция возвращает true, если разрыв произошёл успешно, и false в противном случае. В качестве параметра она принимает SHA1-хеш от сертификата считывателя, пару с котором необходимо разорвать.

## Рекомендуемый порядок работы с VCR

- Вызвать функцию SCardEstablishContext;
- Сгенерировать QR-код для создания пары с помощью функции generatePairingQR;
- Запустить процесс ожидания подключения ридеров, вызвав функцию SCardGetStatusChange;
- Осуществить сопряжение с VCR и дождаться подключения ридера на уровне RtPcsc (для iPad будут отображаться доступные виртуальные считыватели);
- Вызов функции SCardConnect для нужного считывателя с параметром dwShareMode == SCARD_SHARE_DIRECT;
- Вызов функции SCardControl с параметром RUTOKEN_CONTROL_CODE_START_NFC;
- Работа с Рутокеном;
- Вызов функции SCardControl с параметром RUTOKEN_CONTROL_CODE_STOP_NFC;
- Вызов функции SCardDisconnect.

## Пример работы с API

Примеры работы с API есть в репозитории на GitHub rutoken-demoshift-ios в файле https://github.com/AktivCo/rutoken-demoshift-ios/blob/master/demoshift/PcscWrapper/PcscWrapper.swift