ISBC
Группа компаний

Смарт-карты в 1С:Предприятие

Содержание

© 2014 ISBC

© 2014 ISBC

© 2014 ISBC

© 2014 ISBC

© 2014 ISBC

1 Комплект разработчика смарт карт конфигураций (SDK) для 1С

1.1 Назначение комплекта

Комплект разработчика предназначен для разработки конфигураций 1С: Предприятия, реализующих различные проекты любого масштаба, в том числе:

- дисконтных и клубных систем;
- систем обслуживания клиентов;
- обеспечения информационной безопасности;
- защиты ПО;
- контроля доступа;
- платежных (биллинговых) систем.

SDK содержит все самые необходимые инструменты и исчерпывающую документацию, может работать как с контактными, так и с бесконтактными картами. Комплектация оборудования зависит от выбора контактной или бесконтактной технологии и предпочтений заказчика.

Продукт предназначен для организации дисконтных или платежных систем, систем учета и статистики продаж в торговых сетях, магазинах, офисах компаний, где используются системы 1С:Предприятие.

7 преимуществ внедрения смарт-карт:

- Исключение ошибок (человеческого фактора) при начислении скидок или бонусов;
- Беспроцентное кредитование при создании платежной системы. Например, подарочный сертификат;
- Конфиденциальность при начислении скидок и бонусов;
- Многофункциональность. Возможность использования одной карты в нескольких программах;
- Гибкость. Возможность создания сложных программ лояльности, проведения акций без замены карт;
- Имиджевые преимущества. Смарт-карта это технологично, современно, стильно;
- Технологические преимущества при обслуживании территориально распределенной системы.

1.2 Системные требования

Поддерживаемые платформы 1С:Предприятие.

- Платформа 1С:Предприятие версии 8.1.

Поддерживаемые устройства чтения контактных карт :

- ACR 38;
- Все семейство устройств Omnikey CardMan.
- Все семейство контактных ридеров SCM Microsystems.

Могут применяться и любые другие устройства, совместимые со стандартом PC/SC, но только для микропроцессорных карт (например, ACOS). Карты памяти (SLE4428, SLE4442) поддерживаются только на означенных устройствах.

Поддерживаемые контактные карты :

- Микропроцессорные карты ACOS1, ACOS2, ACOS3 (вплоть до 256K);
- Карты памяти SLE4442, SLE4428, SLE5542.

Поддерживаемые устройства чтения бесконтактных карт :

- Omnikey CardMan 5121, 5321;
- SCM Microsystems SDI010;
- ACR128.

Поддерживаемые бесконтактные карты :

- Смарт-карты Mifare Classic 1K/4K;
- Смарт-карты Mifare Plus 2K/4K SL1;
- Карты памяти Mifare Ultralight, Mifare Ultralight C.

Для перечисленных типов карт реализована высокоуровневая поддержка в виде удобных обьектов, позволяющих задействовать все функции карт.

Комплект может быть использован для доступа и к другим смарт-картам на низком уровне - уровне APDU.

Для работы с картами PKI, обеспечивающими функции ассиметричной криптографии и ЭЦП, на высоком уровне данный комплект не предназначен.

1.2.1 Таблица совместимости технологий

Обратите внимание, необходимо обязательно устанавливать свежие драйверы устройств. Лучше всего использовать драйверы из SDK, т.к. они были протестированы на совместимость. В более старых версиях некоторых драйверов выявлены проблемы, которые могут повлиять на работоспособность библиотек SDK и стабильность работы не только вашего приложения, но и операционной системы в целом.

В таблицах сравнения считывтелей используются следующие обозначения:

C - контактные считыватели

S - считыватели с SAM-слотом

ID-000 - считыватели формата SIM-карт

CL - бесконтактные считыватели

D - комбинированные считыватели (контактный и бесконтактный интерфейс)

P - встраиваемые (PCI Express или PCMCIA)

v - совместимо;

x - несовместимо;

? - зависит от наличия драйвера;

Серым цветом отмечены поддерживаемые считыватели, снятые с серийного производства.

ACS Ltd

	ACR38	ACR39 /3901	ACR89	ACR 38T	ACR12 22L	ACR12 51	ACR12 81	ACR 128	AC R1 22
	C			ID-000	CL + S	CL	D	CL+S	CL
SLE 4442 (5542), SLE 4428	v	v	v	v	v	x	v	x	x
ACOS 1,2,3	v	v	v	v	v	x	v	v	x
Mifare Classic Mifare Ultra Light	x	x	x	x	v	v	v	v	v
Mifare Ultra Light C**	x	x	x	x	x	x	v	v	v
Mifare Plus**	x	x	x	x	v	v	v	v	v
x64	v	v	v	v	v	v	v	v	v
Windows Vista	v	v	v	v	v	v	v	v	v
Windows 7	v	v	v	v	v	v	v	v	?
Windows 8	v	v	v	v	v	v	v	v	?
Windows Server 2003, 2008, 2012	v	v	v	v	v	v	v	v	v
Terminal Server (Microsoft)	v	v	v	v	v	v	v	v	v
Terminal Server (Citrix)	v	v	v	v	v	v	v	v	v

* Карты Mifare Ultra Light C требуют прошивки ACR128 версии не ниже V17 (июль 2009 года).

** Документацию на карты Mifare Ultra Light C, Mifare Plus и Mifare DESFire EV1 необходимо запрашивать у производителя чипов (NXP Semiconductors). Документы предоставляются производителем только после подписания договора о нераглашении (Non Disclosure Agreement).

HID Omnikey

	OK 3021	OK 3121	OK 6121	OK 4040	OK 4321	OK 5125	OK 5421	OK 6121	OK 5121	OK 5321
	C	C	C	P	P	C + 125kHz	D	S	D	D
SLE 4442 (5542), SLE 4428*	v	v	v	v	v	v	v	v	v	v
ACOS 1,2,3	v	v	v	v	v	v	v	v	v	v
Mifare Classic Mifare Ultra Light	x	x	x	x	x	x	v	x	v	v
Mifare Ultra Light C**	x	x	x	x	x	x	x	x	x	x
Mifare Plus**	x	x	x	x	x	x	v	x	v	v
x64	v	v	v	v	v	v	v	v	v	v
Windows Vista	v	v	v	v	v	v	v	v	v	v
Windows 7	v	v	v	v	v	v	v	v	v	v
Windows 8	v	v	v	v	v	v	v	v	v	v
Windows Server 2003, 2008, 2012	v	v	v	v	v	v	v	v	v	v
Terminal Server (Microsoft)	v	v	v	v	v	v	v	v	v	v
Terminal Server (Citrix)	v	v	v	v	v	v	v	v	v	v

* - не поддерживается на архитектуре x64 по причине отсутствия 64-разрядной библиотеки scardsyn.dll от OmniKey.

В версиях ридеров Omnikey до осени 2007 года обнаружена несовместимость с картами SLE 5542. Убедитесь, что вы приобретате ридер с устраненной проблемой.

** Документацию на карты Mifare Ultra Light C, Mifare Plus и Mifare DESFire EV1 необходимо запрашивать у производителя чипов (NXP Semiconductors). Документы предоставляются производителем только после подписания договора о нераглашении (Non Disclosure Agreement).

Identive

	SCR 331/531	SCR 335	SCR 3320	SCR3 500	2700 R/ F	470 0F	471 0F	SDI0 10	SCR 3311	SCR 3310
	C	C	ID-000	C	C	D	CL	CL	C	C
SLE 4442 (5542), SLE 4428	v	v	v	x	x	x	x	x	v	v
ACOS 1,2,3	v	v	v	v	v	v	x	x	v	v
Mifare Classic Mifare Ultra Light	x	x	x	x	x	v	v	v	x	x
Mifare Ultra Light C**	x	x	x	x	x	v	v	v	x	x
Mifare Plus**	x	x	x	x	x	v	v	v	x	x
x64	v	v	v	v	v	v	v	v	v	v
Windows Vista	v	v	v	v	v	v	v	v	v	v
Windows 7	v	v	v	v	v	v	v	v	v	v
Windows 8	v	v	v	v	v	v	v	v	v	v
Windows Server 2003, 2008, 2012	v	v	v	v	v	v	v	v	v	v
Terminal Server (Microsoft)	v	v	v	v	v	v	v	v	v	v
Terminal Server (Citrix)	v	v	v	v	v	v	v	v	v	v

** Документацию на карты Mifare Ultra Light C, Mifare Plus и Mifare DESFire EV1 необходимо запрашивать у производителя чипов (NXP Semiconductors). Документы предоставляются производителем только после подписания договора о нераглашении (Non Disclosure Agreement).

1.3 Преемственность с SDK 3.x

Smart Card Development Kit для 1С является новым продуктом и максимально обеспечивает функционал Smart Card Development Kit 3.x в конфигурациях 1С: Предприятие.

1.4 Оборудование, входящее в комплект

1.4.1 Устройства чтения смарт-карт (ридеры)

Возможна комплектация следующими устройствами чтения смарт-карт :

Advanced Card Systems :

ACR 38U

Advanced Card Systems :

ACR 39U

Advanced Card Systems :

ACR 3901U

Advanced Card Systems :

ACR 1222L

Advanced Card Systems :

ACR 1251U

Advanced Card Systems :

ACR 1281U

Advanced Card Systems :

ACR 89U

HID :

OMNIKEY 3121

HID :

OMNIKEY 3021

HID :

OMNIKEY 6121

Advanced card systems :

ACR 38T

Identive :

SCR 3320

HID :

OMNIKEY 4321

Identive :

SCR 243

Identive :

SCR 3040

HID :

OMNIKEY 5421

(DUAL)

HID :

OMNIKEY 5125

(RFID)

Identive :

SCL 010

(RFID)

SDK создавался таким образом, чтобы вы могли выбирать устройство, а не производителя. Хотя решения разных производителей могут использовать различные нестандартные программные интерфейсы, SDK практически полностью скрывает эти особенности от разработчика. В дальнейшем, если вы захотите поменять производителя или использовать параллельно устройства нескольких производителей, ваша программа полностью к этому готова без перекомпиляции. (ньюансы, которые невозможно скрыть внутри библиотек, особо оговариваются в документации).

При выборе устройства обязательно ознакомьтесь с таблицей совместимости технологий.

1.4.2 Устройства чтения смарт-карт, снятые с серийного производства

В данной таблице приведены поддерживаемые устройства чтения смарт-карт, которые были сняты с серийного производства.

ACS :
ACR128
(RFID)

ACS :
ACR122
(RFID)

HID :
OMNIKEY 5321
(RFID)

Identive :
SDI 010
(RFID)

Identive :
SCR 3311

Identive :

SCR 3310

Identive :

SCR 335

Identive :

SCR 531

Identive :

SCR 331

1.4.3 Контактные карты

<table>
<tr><td rowspan="3">Размер памяти (EEPROM), Байт</td><td colspan="2">Карты памяти</td><td>Микропроцессорные карты</td></tr>
<tr><td colspan="2">Защита PIN кодом от записи</td><td rowspan="2">IC, PIN, 3DES, ISO 7816-3, 5 APP кодов, файловая структура, активная аутентификация</td></tr>
<tr><td>Двухбайтовый</td><td>Трехбайтовый</td></tr>
<tr><td>256</td><td></td><td>SLE-4442</td><td rowspan="2"></td></tr>
<tr><td>1024</td><td>SLE-4428</td><td rowspan="2"></td></tr>
<tr><td>> 1024</td><td></td><td>ACOS3-8K, ACOS3-16K</td></tr>
</table>

1. ACOS3 - Микропроцессорная смарт-карта:

- Совместимость с ISO7816-1/2/3, T=0;
- Безопасность обеспечивается алгоритмами DES (или Triple DES) и MAC;
- Аутентификация по случайному числу и секретной паре ключей;
- Поддержка электронного кошелька;
- 8K или 16K EEPROM памяти пользовательских данных.

2. SLE-4442 - карта памяти:

- 256 x 8 бит EEPROM;
- побайтовая адресация;
- защищаемые от записи 32 нижних адреса (байты 0..31);
- 3 байтовый секретный код карты.

3. SLE-4428 - карта памяти:

- 1024 x 8 бит EEPROM;
- побайтовая адресация;
- 1024 битная организация защиты памяти;
- 2 байтовый секретный код карты.

Карты памяти защищены наиболее слабо из всех типов смарт-карт. Карты SLE-4442 и SLE-4428 всегда доступны на чтение. Для доступа на запись

необходимо ввести так называемый PIN - код (двухбайтовый для 4428 и трехбайтовый для 4442). Исключение – первые 32 байта, защита от записи которых побайтно регулируется специальной битовой маской. Производитель может на свое усмотрение предварительно блокировать запись в часть этой области. Как правило там содержится информация о производителе, номере партии, и пр. Память этих карт никак не разделяется и представляет собой просто массив байтов.

В отличие от карт памяти, смарт-карты ACOS имеют несколько (а именно 7) ключей защиты, а также развитую организацию памяти. Память можно разбивать на файлы, и каждому файлу задавать привилегии доступа (отдельно на чтение и на запись) по представлению любого набора ключей. Смарт-карты поддерживают так называемую "двустороннюю аутентификацию", позволяющую надежно шифровать все транзакции с картой (при незащищенном соединении с картой это незаменимо). Для реализации этой аутентификации в смарт-карте есть криптоблок, реализующие алгоритмы DES, 3DES и MAC, а так же генератор случайных чисел. Карты ACOS имеют встроенный механизм кошелька, что позволяет их использовать как надежно защищенное платежное средство.

1.4.4 Бесконтактные карты

Mifare 1K - микропроцессорная бесконтактная смарт-карта :

Совместимость с ISO14443 A;

- 4-байтный уникальный серийный номер;
- Организация памяти : 16 секторов по 4 16-байтных блока;
- Для данных пользователя доступно 752 байта;
- 2 6-байтовых кода аутентификации на каждый сектор;
- Гибкая настройка прав доступа на уровне блока;
- Функция электронного кошелька.

Mifare 4K - микропроцессорная бесконтактная смарт-карта :

Совместимость с ISO14443 A;

- Полная обратная совместимость с Mifare 1K;
- 4-байтный уникальный серийный номер;
- Организация памяти : 32 сектора по 4 16-байтных блока + 8 секторов по 16 16-байтных блоков;
- Для данных пользователя доступно 3440 байт;
- 2 6-байтовых кода аутентификации на каждый сектор;
- Гибкая настройка прав доступа на уровне блока (первые 32 сектора) или группы блоков (8 последних секторов);
- Функция электронного кошелька.

Mifare Ultra Light - бесконтактная карта памяти :

Совместимость с ISO14443 A;

- 7-байтный уникальный серийный номер;
- Организация памяти : 16 страниц по 4 байта;
- Для данных пользователя доступно 48 байт;
- Возможность постраничной обратимой и необратимой защиты от записи;
- Функция 32-битного одноразового счетчика (OTP).

Карты памяти Mifare Ultra Light всегда доступны на чтение и запись, хотя запись в отдельные группы страниц может обратимо или необратимо блокироваться. Если вам необходима защита данных, основанная на предъявлении кодов, то подойдут карты Mifare 1K или 4K.

2 Стандарты для работы со смарт-картами

2.1 ISO 7816

Для унификации смарт-карт технологий существует стандарт ISO 7816, который определяет размеры, физические параметры, логическую структуру и протоколы работы карт. Стандарт состоит из трех основных частей:

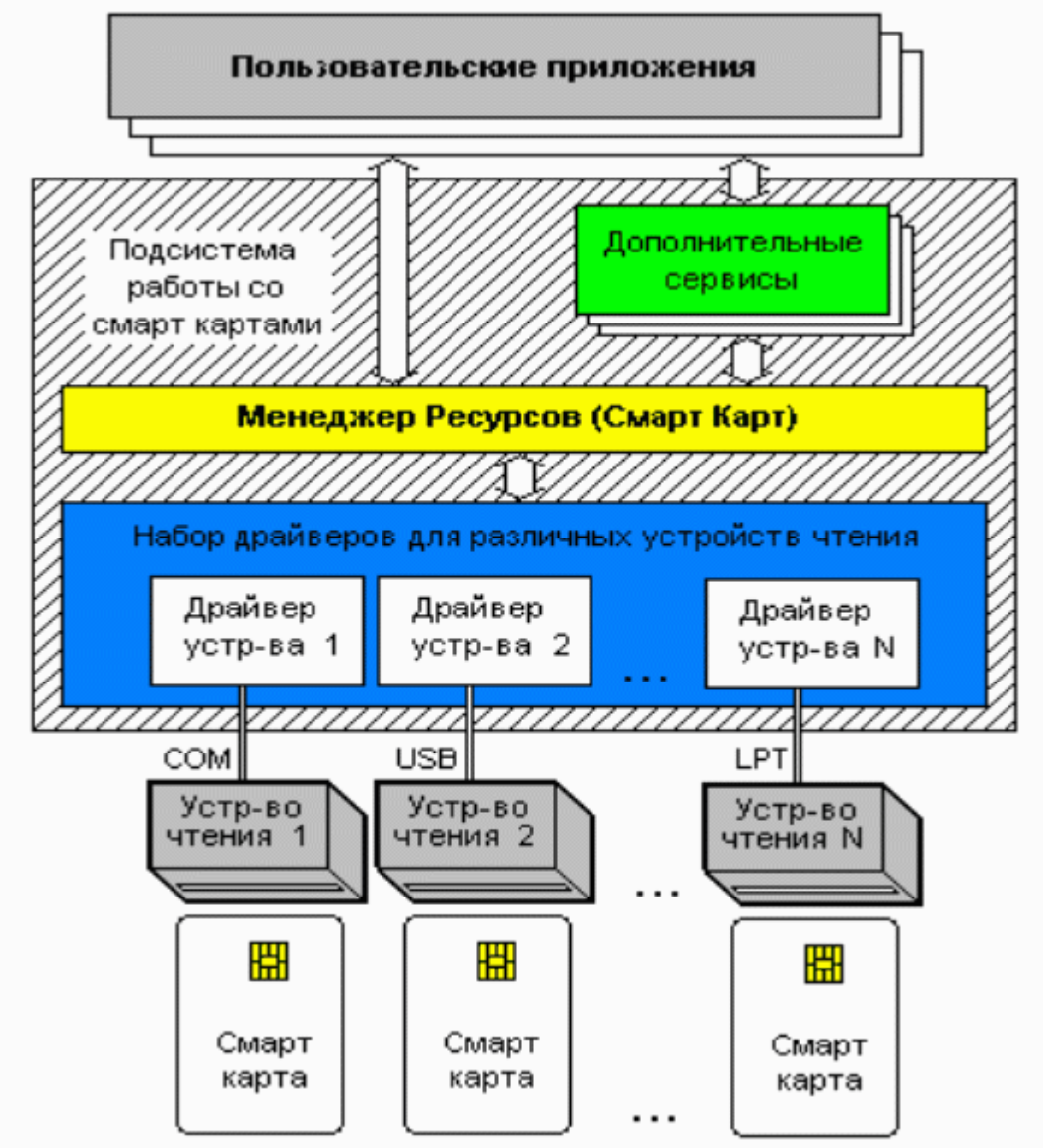
- ISO/IEC 7816-1 - Часть 1: Физические характеристики
- ISO/IEC 7816-2 - Часть 2: Размеры и расположение контактов
- ISO/IEC 7816-3 - Часть 3: Электрические сигналы и протоколы передачи данных

Все прилагаемые в комплекте карты соответствуют первым двум частям стандарта. Микропроцессорные контактные карты также соответствуют 3-й части стандарта, что позволяет работать с ними на любых ридерах любых производителей. Контактные карты памяти не следуют ISO7816-3, поэтому с ними возникают сложности с совместимостью при работе с различными устройствами чтения. Устройства, совместимые с SDK, позволяют работать с картами памяти по их собственному протоколу. Однако, это является нестандартной возможностью, и конкретные реализации такой возможности различаются у производителей ридеров. Программные обьекты SDK практически полностью скрывают различия собственных программных интерфейсов различных производителей оборудования.

2.2 PC/SC

Стандарт PC/SC является стандартом API для работы с устройствами чтения смарт-карт и самими смарт-картами. Во всех поддерживаемых в данном SDK операционных системах PC/SC API уже встроено в ОС и не требует дополнительной установки.

Архитектура интерфейса PC/SC:



Элементы этой архитектуры следующие:

- Смарт-карта - микропроцессорная карта или карта памяти, выполненная в

соответствии со стандартом ISO/IEC 7816.

- Устройство чтения - специальное устройство для сопряжения смарт-карты и компьютера. Может иметь различные интерфейсы для соединения с компьютером, например COM, USB, PCMCIA и т.д. Для соединения со смарт-картой имеет интерфейс, выполненный в соответствии с ISO 7816-1,2,3. Обеспечивает карту питанием, генерирует синхронизирующие сигналы и осуществляет последовательный обмен данными.
- Драйвер устройства чтения - программный компонент, позволяющий осуществлять обмен данными с устройством. Имеет определенную стандартом структуру, которая позволяет менеджеру ресурсов работать с любым устройством чтения по одинаковой схеме.
- Менеджер ресурсов - ключевой компонент архитектуры PC/SC созданный для решения трех основных задач:

  ✓ следит за установленными устройствами чтения и делает эту информацию доступной для других приложений

  ✓ отслеживает известные типы смарт-карт в соответствии с установленными дополнительными сервисами и поддерживаемыми интерфейсами, делая доступной эту информацию другим приложениям.

  ✓ отслеживает вставление и вытаскивание карты.

Кроме этого менеджер ресурсов контролирует распределение устройств чтения и смарт-карт между несколькими приложениями, реализуя данный механизм с помощью подключения к устройствам в общедоступном или эксклюзивном режиме. Одна из важнейших функций менеджера ресурсов - это обеспечение выполнения транзакций к вставленным смарт-картам. Транзакционный режим работы оказывается очень важным из-за того, что сегодняшние смарт-карты являются однопотоковыми устройствами, для которых выполнение нескольких команд должно происходить в виде завершенной единой функции. Транзакции позволяют выполнять несколько команд последовательно без прерываний, гарантируя, что промежуточная информация не будет испорчена.

Дополнительные сервисы - каждый дополнительный сервис расширяет функциональность, давая возможность работать с различными смарт-картами посредством высокоуровневого программного интерфейса. Местоположение данных сервисов в системе не ограничивается, это может быть программа, запущенная на локальном компьютере, или клиент-серверное приложение.

Рассматривая архитектуру в целом, можно сказать, что это достаточно универсальный многофункциональный интерфейс, позволяющий упростить работу с различными устройствами чтения и смарт-картами.

2.3 Реализация PC/SC в Windows

В Windows менеджер ресурсов смарт-карт выполнен в виде службы SCardSvr ("Smart Card"). Функции PC/SC API называются, начиная с префикса "SCard", и реализованы в библиотеке winscard.dll. Эта библиотека ответственна за взаимодействие с менеджером ресурсов.

Если winscard.dll используется в составе приложения, работающего в терминальной сессии, то winscard не работает с менеджером ресурсов сервера. Вместо этого осуществляется прозрачное перенаправление запросов через виртуальные каналы протоколов RDP и ICA. Таким образом, прикладная программа видит клиентские устройства, а не устройства сервера. Для реализации данной технологии существуют некоторые ограничения :

Стандартный терминальный сервер Microsoft (RDP) :

- Требуемая ОС сервера : Windows XP, 2003, Vista.
- Требуемая ОС клиента : Windows 2000, XP, 2003, Vista
- Терминальный клиент версии не ниже 5.0.

Терминальный сервер citrix (ICA) :

- Требуемая ОС сервера : Windows 2000, XP, 2003, Vista.
- Версия ПО сервера : не ниже Citrix Metaframe XP FR2.
- Citrix ICA client версии не ниже 9.0.
- Требуется для каждого процесса индивидуально разрешать доступ к смарт-картам через команду scconfig.exe.
- Не слишком качественная реализация перехвата PC/SC API. Некоторые функции могут не работать или работать неправильно. По крайней мере это было замечено в версии Citrix Metaframe XP FR3. В программных библиотеках SDK были приняты меры, чтобы максимально обойти проблемы Citrix.

3 Описание программных обьектов SDK

3.1 Описание внешней компоненты для 1С

3.1.1 Установка внешней компоненты.

Для установки модуля скопируйте все файлы из каталога BIN в каталог BIN 1С:Предприятия.

Например для 1С:Предприятия версии 8.1 это будет каталог \1cv81\Bin.

Далее зарегистрируйте компоненту при помощи стандартной программы regsvr32 :

```
regsvr32 ESDK2_1C.dll
```

3.1.2 Подключение внешней компоненты

Модуль ESDK2_1C.DLL, предназначенный для работы со смарт-картами из программы 1С:Предприятие версии 8.1, выполнен в виде COM-компоненты в соответствие с «Технологией создания внешних компонент». Модуль расширяет встроенный язык 1С:Предприятия 8.1 посредством реализации интерфейса ILanguageExtender.

Для загрузки модуля и создания COM-объектов компоненты необходимо выполнить следующие действия:

```
ПодключитьВнешнююКомпоненту("AddIn.<имя объекта>");
ПеременнаяЭкземпляраОбьекта = Новый("AddIn.<имя объекта>");
```

Например :

```
ПодключитьВнешнююКомпоненту("AddIn.PCSCInfo");
Обьект0 = Новый("AddIn.PCSCInfo");
```

Другие возможные варианты :

```
Обьект1 = Новый("AddIn.PCSCard");
Обьект2 = Новый("AddIn.PCSCCard_Memory");
Обьект3 = Новый("AddIn.PCSCCard_SLE4428");
Обьект4 = Новый("AddIn.PCSCCard_SLE4442");
Обьект5 = Новый("AddIn.PCSCCard_ACOS");
Обьект6 = Новый("AddIn.PCSCCard_Mifare");
Обьект7 = Новый("AddIn.PCSCCard_MifareUL");
Обьект8 = Новый("AddIn.PCSCMonitoring");
Обьект9 = Новый("AddIn.PCSCCapture");
Обьект10 = Новый("AddIn.Crypto");
Обьект11 = Новый("AddIn.DataHelper");
```

3.1.3 Константы, используемые в SDK

Список используемых SDK констант.

VARIANT_TRUE	-1
VARIANT_FALSE	0

CT_MCU	0
CT_IIC_Auto	1
CT_IIC_1K	2
CT_IIC_2K	3
CT_IIC_4K	4
CT_IIC_8K	5
CT_IIC_16K	6
CT_IIC_32K	7
CT_IIC_64K	8
CT_IIC_128K	9
CT_IIC_256K	10
CT_IIC_512K	11
CT_IIC_1024K	12
CT_AT88SC153	13
CT_AT88SC1608	14
CT_SLE4418	15
CT_SLE4428	16
CT_SLE4432	17
CT_SLE4442	18
CT_SLE4406	19
CT_SLE4436	20
CT_SLE5536	21
CT_MCUT0	22
CT_MCUT1	23
CT_MCU_Auto	24

SCARD_SCOPE_USER	0
SCARD_SCOPE_TERMINAL	1
SCARD_SCOPE_SYSTEM	2

SCARD_UNKNOWN	0
SCARD_ABSENT	1
SCARD_PRESENT	2
SCARD_SWALLOWED	3
SCARD_POWERED	4
SCARD_NEGOTIABLE	5
SCARD_SPECIFIC	6

SCARD_STATE_UNAWARE	0
SCARD_STATE_IGNORE	1
SCARD_STATE_CHANGED	2
SCARD_STATE_UNKNOWN	4
SCARD_STATE_UNAVAILABLE	8
SCARD_STATE_EMPTY	16
SCARD_STATE_PRESENT	32
SCARD_STATE_ATRMATCH	64
SCARD_STATE_EXCLUSIVE	128
SCARD_STATE_INUSE	256
SCARD_STATE_MUTE	512
SCARD_STATE_UNPOWERED	1024

SCARD_SHARE_EXCLUSIVE	1
SCARD_SHARE_SHARED	2
SCARD_SHARE_DIRECT	3

SCARD_LEAVE_CARD	0

SCARD_RESET_CARD	1
SCARD_RESET_CARD	2
SCARD_EJECT_CARD	3

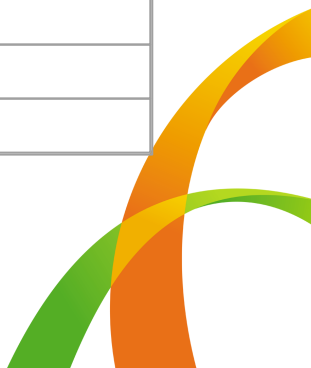
ACOS_TYPE_12	1
ACOS_TYPE_3_8K	308
ACOS_TYPE_3_16K	316
ACOS_TYPE_3_24K	324
ACOS_TYPE_3_64K	364

ACOS_FID_MCUID	65280
ACOS_FID_MANUFACTURER	65281
ACOS_FID_PERSONALIZATION	65282
ACOS_FID_SECURITY	65283
ACOS_FID_USER_FILE_MANAGEMENT	65284
ACOS_FID_ACCOUNT	65285
ACOS_FID_ACCOUNT_SECURITY	65286
ACOS_FID_ATR	65287

ACOS_STAGE_MANUFACTURING	0
ACOS_STAGE_PERSONALIZATION	1
ACOS_STAGE_USER	2

ACOS_MANOPT_BLOW_FUSE	128
ACOS_MANOPT_INQ_ACC_MAC	64
ACOS_MANOPT_RECORD_NUMBERING	32

ACOS_PERSOPT_ACCOUNT	1
ACOS_PERSOPT_3DES	2
ACOS_PERSOPT_PIN_ALT	4
ACOS_PERSOPT_DEB_MAC	8
ACOS_PERSOPT_DEB_PIN	16



ACOS_PERSOPT_REV_DEB	32
ACOS_PERSOPT_TRNS_AUT	64
ACOS_PERSOPT_INQ_AUT	128

ACOS_SECOPT_AC1_DES	2
ACOS_SECOPT_AC2_DES	4
ACOS_SECOPT_AC3_DES	8
ACOS_SECOPT_AC4_DES	16
ACOS_SECOPT_AC5_DES	32
ACOS_SECOPT_PIN_DES	64
ACOS_SECOPT_IC_DES	128

ACOS_SFN_IC	0
ACOS_SFN_PIN	1
ACOS_SFN_CARD	2
ACOS_SFN_CARD2	12
ACOS_SFN_TERM	3
ACOS_SFN_TERM2	13
ACOS_SFN_LAST	13
ACOS_SFN_AC1	5
ACOS_SFN_AC2	6
ACOS_SFN_AC3	7
ACOS_SFN_AC4	8
ACOS_SFN_AC5	9

ACOS_ASFN_DEBIT	0
ACOS_ASFN_CREDIT	1
ACOS_ASFN_INQUIRE	2
ACOS_ASFN_REVOKE	3
ACOS_ASFN_LAST	3

ACOS_CODE_AC1	1

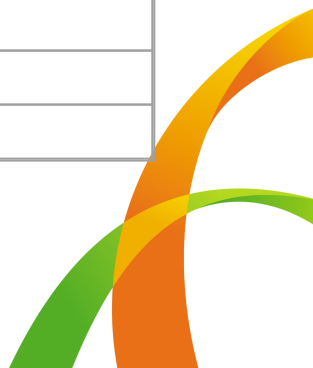
ACOS_CODE_FIRST	1
ACOS_CODE_AC2	2
ACOS_CODE_AC3	3
ACOS_CODE_AC4	4
ACOS_CODE_AC5	5
ACOS_CODE_PIN	6
ACOS_CODE_IC	7
ACOS_CODE_LAST	7

ACOS_AC_IC	128
ACOS_AC_PIN	64
ACOS_AC_AC5	32
ACOS_AC_AC4	16
ACOS_AC_AC3	8
ACOS_AC_AC2	4
ACOS_AC_AC1	2
ACOS_AC_NEVER	1
ACOS_AC_ALWAYS	0

ACOS_TRANSTYPE_DEBIT	1
ACOS_TRANSTYPE_REVOKE_DEBIT	2
ACOS_TRANSTYPE_CREDIT	3

ACOS_FA_BINARY	128
ACOS_FA_READ_SECURE	64
ACOS_FA_WRITE_SECURE	32
ACOS_FIDX_SYSTEM_FILE	255

CL_CARDTYPE_UNKNOWN	0
CL_CARDTYPE_MIFARE_1K	1
CL_CARDTYPE_MIFARE_4K	2
CL_CARDTYPE_MIFARE_ULTRA_LIGHT	3



CL_CARDTYPE_SLE55R_XXXXX	4
CL_CARDTYPE_SR176	6
CL_CARDTYPE_SRIX4K	7
CL_CARDTYPE_AT88RF020	8
CL_CARDTYPE_AT88SC0204CRF	9
CL_CARDTYPE_AT88SC0808CRF	10
CL_CARDTYPE_AT88SC1616CRF	11
CL_CARDTYPE_AT88SC3216CRF	12
CL_CARDTYPE_AT88SC6416CRF	13
CL_CARDTYPE_SRF55V10P	14
CL_CARDTYPE_SRF55V02P	15
CL_CARDTYPE_SRF55V10S	16
CL_CARDTYPE_SRF55V02S	17
CL_CARDTYPE_TAGIT	18
CL_CARDTYPE_LRI512	19
CL_CARDTYPE_ICODESLI	20
CL_CARDTYPE_TEMPSENS	21
CL_CARDTYPE_ICODE1	22
CL_CARDTYPE_PICOPASS_2K	23
CL_CARDTYPE_PICOPASS_2KS	24
CL_CARDTYPE_PICOPASS_16K	25
CL_CARDTYPE_PICOPASS_16KS	26

CL_KEYTYPE_MIFARE_A	96
CL_KEYTYPE_MIFARE_B	97

MEMORY_CARD_TYPE_UNKNOWN	0
MEMORY_CARD_TYPE_SLE4428	1
MEMORY_CARD_TYPE_SLE4442	2

3.1.4 Особенности работы с компонентой

Формат функций.

Число ИмяФункции(Параметры);

Число – код возврата после выполнения функции. При успешном выполнении функции равен Истина(1), при ошибке содержит код ошибки.

ИмяФункции – имя вызываемой функции в английском или русском варианте.

Параметры – значения, передаваемые для выполнения функции. Могут быть как входящими, так и возвращаемыми.

В документации по функциям входящие значения помечены идентификатором [in], возвращаемые значения помечены идентификатором [out].

Значения могут быть или типа *Число*, или типа *Строка*.

Тип *Число* это целое численное значение.

Тип *Строка* это строка в шестнадцатиричной записи. То есть вида "`1122334455667788 99AABBCCDDEEFF`", где каждая пара символов задает шестнадцатиричный код байта.

Внимание! Количество символов в шестнадцатиричной строке должно быть кратно 2.

Для перевода чисел, вещественных чисел и текстовых строк из формата 1С в формат шестнадцатиричных строк в компоненте предусмотрен обьект DataHelper.

3.2 Описание обьектов

3.2.1 PCSCInfo

Класс позволяет получать информацию о ридерах и картах, в них вставленных.

Методы

ListReaders/ПолучитьСписокРидеров	Получить список ридеров
IsCardInserted/ВставленаЛиКарта	Вставлена ли карта в ридер ?
GetCardATR/ПолучитьATRКарты	Получить ATR карты
GetStatusChange/ ОжиданиеСменыСтатусаКарты	Ожидание смены статуса вставленности карты
GetReaderName/ПолучитьИмяРидера	Получить имя ридера.

ListReaders

Получение количества доступных в системе ридеров.

```
Число ListReaders/ПолучитьСписокРидеров(
[out] Число КоличествоРидеров);
```

Параметры

`[out] Число КоличествоРидеров.`

Результат

Функция возвращает код ошибки либо *Истину*. Возвращаются только доступные ридеры (не им

В параметр *КоличествоРидеров* записывается количество доступных в системе ридеров.

IsCardInserted

Позволяет узнать вставлена ли карта в ридер.

```
Число IsCardInserted/ВставленаЛиКарта(
    [in] Строка ИмяРидера,
    [out] Число КартаВставлена);
```

Параметры

[in] Строка Имя ридера.

[out] Число КартаВставлена.

Результат

Функция возвращает код ошибки либо *Истину*.

Функция позволяет определить вставлена ли карта в ридер с именем *ИмяРидера*.

В параметр *КартаВставлена* записывается:

VARIANT_TRUE = карта вставлена.

VARIANT_FALSE = карта не вставлена

GetCardATR

Получение ATR карты.

```
Число GetCardATR/ПолучитьATRКарты(
    [in] Строка ReaderName,
    [out] Строка ATR);
```

Параметры

`[in] Строка ReaderName.`

`[out] Строка ATR.`

Результат

Функция возвращает код ошибки либо *Истину*.

В параметр *ATR* записывается ATR вставленной карты. Если карта не вставлена или не отвечает, то возвращается пустая строка.

Описание

`ATR (Answer-To-Reset)` позволяет опознать тип карты. ATR представляет собой последовательность байтов максимальной длиной 36, и доступен без установления подключения к карте.

GetStatusChange

Функция позволяет выполнить ожидание смены состояния карты в ридере.

```
Число GetStatusChange/ОжиданиеСменыСтатусаКарты(
    [in] Строка ReaderName,
    [in] Число Timeout,
    [in] Число CurrentState,
    [out] Число EventState,
    [out] Число StatusChanged);
```

Параметры

`[in] ReaderName`

Имя ридера.

`[in] Timeout`

Максимальное время ожидания смены состояния карты. Значение INFINITE (-1) означает бесконечное ожидание.

`[in] CurrentState`

Состояние карты как его видит приложение (битовые флаги) :

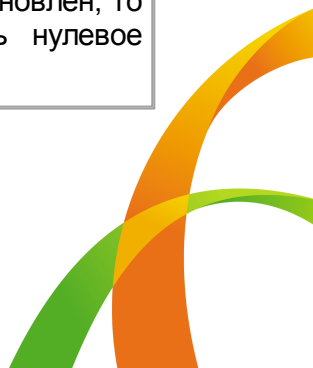
Значение	Описание
SCARD_STATE_UNAWARE	Приложение не знает о состоянии карты. Происходит немедленный возврат из функции.
SCARD_STATE_UNAVAILABLE	Приложение считает, что ридер недоступен для использования. Если этот бит установлен, то все последующие биты игнорируются.
SCARD_STATE_EMPTY	Приложение считает, что карта не вставлена в ридер. Если этот бит установлен, то все

	последующие биты игнорируются.
SCARD_STATE_PRESENT	Приложение считает, что карта вставлена в ридер.
SCARD_STATE_EXCLUSIVE	Приложение считает, что карта находится в эксклюзивном использовании другим приложением. Если этот бит установлен, то SCARD_STATUS_PRESENT подразумевается.
SCARD_STATE_INUSE	Приложение считает, что карта используется одним или более приложением в режиме разделяемого доступа. Если этот бит установлен, то SCARD_STATUS_PRESENT подразумевается.
SCARD_STATE_MUTE	Приложение считает, что в ридере находится карта, не отвечающая на запросы.

[out] EventState

Изменившееся состояние карты (битовые флаги) :

Значение	Смысл
SCARD_STATE_IGNORE	Ридер должен быть проигнорирован.
SCARD_STATE_CHANGED	Произошло изменение состояния.
SCARD_STATE_UNKNOWN	Имя ридера не распознано менеджером ресурсов. Если этот бит установлен, SCARD_STATE_CHANGED и SCARD_STATE_IGNORE также будут установлены.
SCARD_STATE_UNAVAILABLE	Информация о состоянии недоступна. Если этот бит установлен, то все последующие биты будут иметь нулевое значение.
SCARD_STATE_EMPTY	В ридере нет карты. Если этот бит установлен, то все последующие биты будут иметь нулевое значение.



SCARD_STATE_PRESENT	В ридере имеется карта (не обязательно отвечающая на запросы).
SCARD_STATE_EXCLUSIVE	Карта в ридере используется эксклюзивно другим приложением. Если этот бит установлен, то SCARD_STATUS_PRESENT также будет установлен.
SCARD_STATE_INUSE	Карта используется одним или более приложением в режиме разделяемого доступа. Если этот бит установлен, то SCARD_STATUS_PRESENT также будет установлен.
SCARD_STATE_MUTE	Карта в ридере не отвечает на запросы.

Результат

Функция возвращает код ошибки либо *Истину*.

В параметр StatusChanged записывается:

VARIANT_TRUE = статус карты изменился.

VARIANT_FALSE = статус карты не изменился.

GetReaderName

Получение имени ридера.

```
Число GetReaderName/ПолучитьИмяРидера(
    [in] Число НомерРидера,
    [out] Строка ИмяРидера);
```

Параметры

`[in] Число НомерРидера.`

`[out] Строка ИмяРидера.`

Результат

Функция возвращает код ошибки либо *Истину*.

Функция позволяет получить имя ридера по его порядковому номеру. Количество доступных ридеров возвращается функцией `ListReaders/ПолучитьСписокРидеров`.

В параметр `ИмяРидера` записывается название ридера.

3.2.2 PCSCMonitoring

Класс позволяет осуществлять наблюдение за приходом/уходом ридеров и вставлением/извлечением карт.

Методы

InitializeMonitoring/ ЗапуститьМониторинг	Начать мониторинг
StopMonitoring/ ОстановитьМониторинг	Завершить мониторинг

События

ПоявилсяРидер	Появился новый ридер
ПропалРидер	Ридер перестал существовать
КартаВставлена	Карта вставлена
КартаИзвлечена	Карта извлечена

Описание

Функция мониторинга может быть полезна в конфигурациях, работающих с несколькими ридерами. Если ваша конфигурация предполагает работу только с одним (любым) ридером, то вместо мониторинга рекомендуется использовать технику захвата карты.

Мониторинг производится в отдельном потоке. После выполнения метода InitializeMonitoring могут вызываться события.

Внутри класса имеется очередь событий. Это значит, что пока вы обрабатываете событие, цикл опроса продолжается, и все новые события поступают в очередь.

С приходом нового ридера автоматически проверяется наличие карты в нем. Если карта присутствует, то после события прихода ридера вызывается событие вставления карты. С уходом ридера, если в нем была вставлена карта, то перед событием ухода ридера вызывается событие извлечения карты. Таким образом, если вас интересует только мониторинг карт, не обязательно обрабатывать события прихода/ухода ридеров. В конфигурации 1С события должны обрабатываться в процедуре ОбработкаВнешнегоСобытия(Источник, Событие, Данные).

InitializeMonitoring

Запустить мониторинг.

```
Число InitializeMonitoring/ЗапуститьМониторинг();
```

StopMonitoring

Завершить мониторинг. Операция автоматически выполняется при уничтожении класса.

```
Число StopMonitoring/ОстановитьМониторинг();
```

ПоявилсяРидер

Появился новый ридер.

```
ОбработкаВнешнегоСобытия(Источник, Событие, Данные);
```

Параметры

```
Источник        = "PCSCMonitoring".
Событие         = "ПоявилсяРидер".
Данные          = ИмяРидера
```

ПропалРидер

Ридер перестал существовать.

```
ОбработкаВнешнегоСобытия(Источник, Событие, Данные);
```

Параметры

```
Источник        = "PCSCMonitoring".
Событие         = "ПропалРидер".
Данные          = ИмяРидера
```

КартаВставлена

Карта вставлена.

```
ОбработкаВнешнегоСобытия(Источник, Событие, Данные);
```

Параметры

```
Источник        = "PCSCMonitoring".
Событие         = "КартаВставлена".
Данные          = ИмяРидера@ATRКарты
```

КартаИзвлечена

Карта извлечена.

```
ОбработкаВнешнегоСобытия(Источник, Событие, Данные);
```

Параметры

```
Источник        = "PCSCMonitoring".
Событие         = "КартаИзвлечена".
Данные          = ИмяРидера
```

3.2.3 PCSCCapture

Класс позволяет осуществлять захват одной карты по произвольному сценарию, определяемому приложением.

Методы

InitializeCapture/НачатьЗахват	Начать процесс захвата
StopCapture/ ОстановитьЗахват	Завершить процесс захвата
SetATR/УстановитьATR	Установить ATR захватываемой карты

События

КартаЗахвачена	Карта захвачена
КартаОсвобождена	Карта освобождена

Описание

Техника захвата карты полезна, когда приложение предназначено для работы только с одним (любым) ридером и ожидает только определенные карты. В общем случае в системе может быть неограниченное количество ридеров, в каждом из которых может быть вставлена любая карта. Какой ридер выбрать ? Вставлена ли подходящая карта ? Захват карты позволяет удобно разрешить данные вопросы.

Механизм захвата карты базируется на механизме мониторинга и реализует машину состояний. Имеется 2 состояния :

1. Карта не захвачена. Событие вставления карты в любой ридер вызывает обратный запрос от системы захвата : подходит ли карта ? Если карта подходит, то вызывается событие захвата карты, и происходит переход в состояние (2). Иначе продолжается мониторинг событий вставления карты.
2. Карта захвачена. Осуществляется мониторинг только одного ридера - того, в котором была захвачена карта. При поступлении события извлечении карты вызывается событие освобождения карты, и происходит переход в состояние (1).

Возможность принятия решения о том, подходит ли карта, предоставляется приложению через событие КартаЗахвачена. Таким образом, при данном сценарии пользователь может вставить свою карту в любой ридер, и приложение прореагирует только на правильную карту.

Внутри класса имеется очередь событий. Это значит, что пока вы обрабатываете событие, цикл опроса продолжается, и все новые события поступают в очередь. Перед вызовом функции InitializeCapture необходимо установить ATR захватываемой карты вызовом функции SetATR.

InitializeCapture

Запустить процесс захвата.

```
Число InitializeCapture/НачатьЗахват();
```

StopCapture

Завершить процесс захвата. Операция автоматически выполняется при уничтожении класса.

```
Число StopCapture/ОстановитьЗахват();
```

КартаЗахвачена

Карта захвачена.

```
ОбработкаВнешнегоСобытия(Источник, Событие, Данные);
```

Параметры

```
Источник        = "PCSCCapture".
Событие         = "КартаЗахвачена".
Данные          = ИмяРидера.
```

КартаОсвобождена

Карта освобождена.

```
ОбработкаВнешнегоСобытия(Источник, Событие, Данные);
```

Параметры

```
Источник        = "PCSCCapture".
Событие         = "КартаОсвобождена".
Данные          = ИмяРидера.
```

SetATR

Установить ATR захватываемой карты.

```
Число SetATR/УстановитьATR([in] Строка ATR);
```

Параметры

`[in] ATR`

ATR карты.

Описание

Обработчик сравнивает заданный ATR с ATR вставленной карты и в зависимости от успеха сравнения вызывает событие о появлении ожидаемой карты.

3.2.4 PCSCCard

Базовый класс для работы с PC/SC картами.

Методы

Status/ПолучитьСтатусКарты	Получить статус карты и протокола
AdjustReaderMode/ НастроитьРидерНаТипКарт	Настройка ридера на тип карт
AdjustReaderModeForACSReader / НастроитьРидерACSНаТипКарт	Настройка ридеров ACS на тип карт
Connect/Подключиться	Подключение к карте
BeginTransaction/ НачатьТранзакцию	Начать транзакцию
Transmit/ПередатьAPDU	Передать APDU
Control/ ПередатьДрайверуIOCTLЗапрос	Передать драйверу IOCTL запрос
EndTransaction/ ЗавершитьТранзакцию	Завершить транзакцию
Disconnect/Отключиться	Отключение от карты

Свойства

ConnectionReader/ИмяРидера	Имя ридера
ConnectionShareMode/ РежимРазделенияДоступа	Режим разделения доступа
ConnectionPreferredProtocols/ ДопустимыеПротоколыПодключения	Допустимые протоколы подключения
ConnectionDisconnectDisposition/ РежимОтключения	Режим отключения
ATR/ATRПодключеннойКарты	ATR подключенной карты

Response/ОтветКарты	Ответ карты на последнюю команду или результат выполнения операции
StatusWord/СтатусКоманды	Статус последней команды к карте

Описание

PCSCCard является базовым классом, реализующим интерфейс взаимодействия с произвольной картой, совместимой с PC/SC. Это может быть микропроцессорная карта стандарта ISO7816-3, USB токен, карта памяти или бесконтактная карта (при условии, что производитель ридера предусмотрел возможность доступа к картам данного типа через PC/SC). Общение с картой производится на уровне APDU - байтовых посылок.

Карты памяти и бесконтактные карты, как правило, обслуживаются через специальные библиотеки производителей ридеров. В этом случае можно использовать класс PCSCCard как средство управления подключением к карте, а дальше использовать соответствующие библиотеки для работы с конкретным типом карт на конкретном оборудовании.

В настоящем SDK реализована готовая поддержка карт ACOS, SLE4428, SLE4442.

ConnectionReader

Текущее имя ридера.

```
Строка ConnectionReader/ИмяРидера
```

Описание

Перед началом работы с картой нужно обязательно установить имя ридера. Имя ридера можно получить при помощи метода ListReaders класса PCSCInfo или в обработчиках событий классов PCSCCapture и PCSCMonitoring. Значение используется для всех последующих операций.

ConnectionShareMode

Режим разделения доступа к карте между приложениями. Значение используется для всех последующих операций.

```
Число ConnectionShareMode/РежимРазделенияДоступа
```

Описание

Режим разделения доступа. Одно из следующих значений :

Свойство доступно и на запись, и на чтение.

Значение	Описание
SCARD_SHARE_SHARED	Режим разделяемого доступа (рекомендуется). Если в настоящий момент выполняется транзакция с картой, то подключение пройдет успешно. Однако, при попытке начать другую транзакцию, в т.ч. неявно, будет выполнено бессрочное ожидание завершения выполняющейся транзакции. Из состояния ожидания можно выйти принудительно только при помощи прямого вызова функции PC/SC API SCardCancel с хэндлом соединения с картой из другого потока. Метод Cancel отсутствует в классе, поскольку применяется потоковая модель STA, которая сериализует вызовы в один поток.
SCARD_SHARE_EXCLUSIVE	Режим эксклюзивного доступа. Если в настоящий момент имеется другое подключение, то произойдет ошибка SCARD_E_SHARING_VIOLATION.
SCARD_SHARE_DIRECT	Прямой доступ к драйверу. Карта может отсутствовать. Режим может быть полезен для подачи IOCTL команд драйверу. Если в натоящий момент имеется другое подключение, то произойдет ошибка SCARD_E_SHARING_VIOLATION.

ConnectionPreferredProtocols

Допустимые протоколы подключения к карте. Значение используется для всех последующих операций.

```
Число ConnectionPreferredProtocols/

ДопустимыеПротоколыПодключения
```

Описание

Допустимые протоколы. Битовое поле.

Значение	Описание
SCARD_PROTOCOL_T0	Допустим протокол T=0.
SCARD_PROTOCOL_T1	Допустим протокол T=1.
0	Нулевое значение возможно, если используется режим разделения доступа SCARD_SHARE_DIRECT.

ConnectionDisconnectDisposition

Режим отключения. Значение используется для всех последующих операций.

```
Число ConnectionDisconnectDisposition/РежимОтключения
```

Описание

Режим отключения. Одно из следующих значений :

Значение	Описание
SCARD_LEAVE_CARD	Ничего не делать с картой. Используя этот режим, убедитесь, что перед отключением были сброшены установленные права доступа к карте. Например, если вы предъявили карте PIN-код, то после отключения в режиме SCARD_LEAVE_CARD состояние введенного PIN-кода будет доступно другим приложениям.
SCARD_RESET_CARD	Сбросить карту посредством сигнала RESET.
SCARD_UNPOWER_CARD	Отключить питание от карты.
SCARD_EJECT_CARD	Для ридеров с механизмом подачи карты : извлечь карту из ридера.

Status

Получить статус карты и протокола.

```
Число   Status/ПолучитьСтатусКарты([out]   Число   State,[out]
Число Protocol);
```

Параметры

`[out] State`

Статус карты. Может быть 0.

Значение	Описание
SCARD_ABSENT	В ридере нет карты.
SCARD_PRESENT	В ридере есть карта, но она не находится на контактной площадке.
SCARD_SWALLOWED	В ридере есть карта и она находится в правильном положении, однако питание не подано.
SCARD_POWERED	В ридере есть карта, она находится в правильном положении, питание подано, но драйвер ридера не провел процедуру сброса карты и не знает, в каком она находится состоянии.
SCARD_NEGOTIABLE	Карта была сброшена и ждет прохождения процедуры PTS negotiation.
SCARD_SPECIFIC	Был установлен один из протоколов взаимодействия с картой.

`[out] Protocol`

Текущий протокол. Может быть 0.

Значение	Описание
SCARD_PROTOCOL_RAW	Используется протокол низкого уровня.
SCARD_PROTOCOL_T0	Используется протокол T=0.
SCARD_PROTOCOL_T1	Используется протокол T=1.

Описание

Функция возвращает код ошибки либо *Истину*.

Функция работает только с подключенной картой : необходимо выполнение Connect.

Если вы не используете метод подключения SCARD_SHARE_DIRECT, то для вас не представит интереса поле *pdwState*. Оно всегда будет в значении SCARD_SPECIFIC.

AdjustReaderMode

Настройка ридера на тип карт.

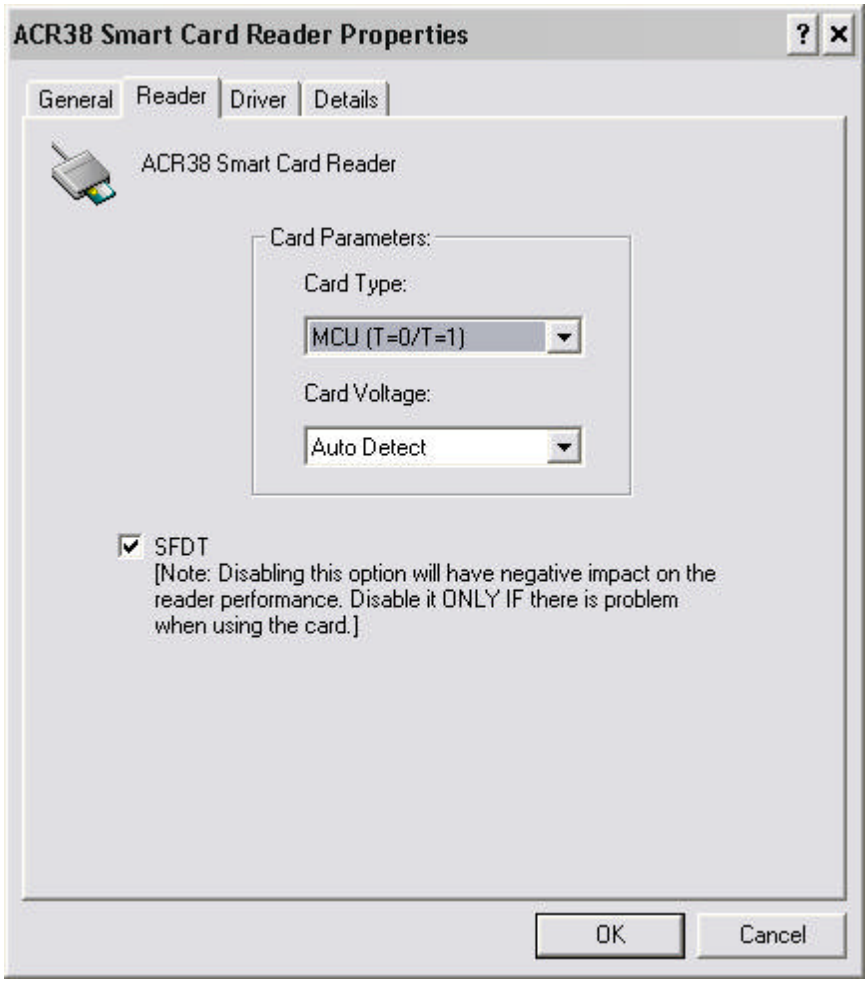
```
Число AdjustReaderMode/НастроитьРидерНаТипКарт();
```

Описание

Некоторые модели ридеров (в частности, ACR38) требуют переключения режимов для доступа к различным картам памяти и к микропроцессорным картам. Реализация метода в классе PCSCCard настраивает ридер на работу с микропроцессорными картами ISO7816-3, в остальных классах, работающих с конкретными картами, может применятся другая настройка.

Функция должна вызываться до подключения к карте как отдельная операция.

Существует одна проблема при использовании данной функции. У ридеров ACS она требует эксклюзивного доступа к ридеру, и поэтому ее применение при каждом подключении может вызывать ошибки. Если для вас это представляет проблему, то существует метод ее решения. Откройте диспетчер устройств и найдите драйвер ридера ACR38. Откройте свойства драйвера.



Измените поле "Card Type" на значение "Auto detection". Исключите из вашей программы вызовы метода AdjustReaderMode. Тип карты будет определяться автоматически. При этом увеличатся задержки подключения. Если вы будете работать только с картами памяти, то для сокращения задержек вместо "Auto detection" можно выбрать нужный тип карт. Это решение имеет недостаток : действие придется выполнить на каждом рабочем месте, где будет выполняться ваша программа.

AdjustReaderModeForACSReader

Настройка ридера ACS на тип карт.

```
Число AdjustReaderModeForACSReader/НастроитьРидерACSНаТипКарт
(
[in] Число CardType);
```

Параметры

`[in] Число CardType`

Тип карты. Возможные значения : `CT_MCU, CT_IIC_Auto, CT_IIC_1K, CT_IIC_2K, CT_IIC_4K, CT_IIC_8K, CT_IIC_16K, CT_IIC_32K, CT_IIC_64K, CT_IIC_128K, CT_IIC_256K, CT_IIC_512K, CT_IIC_1024K, CT_AT88SC153, CT_AT88SC1608, CT_SLE4418, CT_SLE4428, CT_SLE4432, CT_SLE4442, CT_SLE4406, CT_SLE4436, CT_SLE5536, CT_MCUT0, CT_MCUT1, CT_MCU_Auto.`

Описание

Функция может быть полезна, если вы разрабатываете собственный класс для поддержки карт, с которыми работают ридеры ACS, но поддержка которых не реализована в SDK.

Если функция вызывается на ридере, отличном от ACR38, то это не будет ошибкой - просто ничего не произойдет. Поэтому, вам нет необходимости проверять тип ридера самостоятельно.

Функция должна вызываться до подключения к карте как отдельная операция.

Ознакомьтесь с описание возможной проблемы при использовании данной функции.

Connect

Подключиться к карте.

```
Число Connect/Подключиться();
```

Результат

Функция возвращает или код ошибки или *Истину*.

BeginTransaction

Начать транзакцию.

```
Число BeginTransaction/НачатьТранзакцию();
```

Описание

Транзакция - это атомарная операция, состоящая из нескольких операций Transmit. Пока выполняется транзакция, никакой другой поток или другое приложение не могут осуществлять взаимодействие с картой. Таким образом, вы имеете гарантию, что между вашими APDU кто-то другой не вставит свои APDU.

Если вы не выполнили `BeginTransaction`, то каждый вызов Transmit является неявной транзакцией. Между вызовами Transmit другой поток или приложение могут выполнять свои действия. Поэтому, если операция состоит из нескольких APDU, и используется режим разделяемого доступа к карте, то явное обозначение транзакций является обязательным шагом.

Механизм транзакций включает в себя счетчик вложенных вызовов. Количество вызовов EndTransaction должно соответствовать количеству вызовов `BeginTransaction`.

Операция Disconnect завершает транзакцию автоматически.

Transmit

Передать карте APDU команду.

```
Число Transmit/ПередатьAPDU(
    [in] Строка SendBuffer,
    [in] Число MaximumExpectedRecvLength);
```

Параметры

`[in] SendBuffer`

Байтовая посылка.

`[in] MaximumExpectedRecvLength`

Максимальная длина принимаемой информации. По умолчанию 260 байт.

Описание

С помощью этой функции происходит общение со смарт-картой. Вы посылаете команду в виде байтовой посылки и принимаете ответ в виде ответной байтовой посылки. Ответ сохраняется в Response. Как правило, последние 2 байта ответа означают статус операции. Он дополнительно сохраняется в StatusWord для удобства анализа.

Каждый тип карт имеет свое назначение, свой набор APDU команд и свою архитектуру, поэтому не идет речи о совместимости между картами различных типов такого рода, как, например, между устройствами USB flash drive от разных производителей. Чтобы работать с картой на уровне APDU, необходима документация, описывающая архитектуру карты и APDU команды. Если ваше приложение должно работать с несколькими типами карт, то ваша задача состоит в создании уровня абстракции карты, предназначенного для выполнения функций вашего приложения.

Control

Передать драйверу IOCTL запрос.

```
Число Control/ПередатьДрайверуIOCTLЗапрос(
    [in] Число ControlCode,
    [in] Строка InBuffer,
    [in] Число MaximumExpectedRecvLength);
```

Параметры

`[in] ControlCode`

IOCTL код.

`[in] InBuffer`

Входные данные IOCTL команды.

`[in] MaximumExpectedRecvLength`

Максимальная длина принимаемой информации. По умолчанию 260 байт.

Описание

С помощью этой функции происходит передача специальных запросов драйверу ридера. Выходные данные помещаются в Response.

Как правило, IOCTL команды и их структура входных и выходных данных зависят от производителя оборудования.

EndTransaction

Завершить транзакцию.

```
Число EndTransaction/ЗавершитьТранзакцию();
```

Описание

Механизм транзакций включает в себя счетчик вложенных вызовов. Количество вызовов `EndTransaction` должно соответствовать количеству вызовов BeginTransaction.

Операция Disconnect завершает транзакцию автоматически.

При завершении транзакции выполняется действие, определяемое режимом отключения. Возможные режимы отключения см. в описании свойства ConnectionDisconnectDisposition.

Disconnect

Отключиться от карты/ридера.

```
Число Disconnect/Отключиться();
```

Описание

Если имеется незавершенная транзакция, то она автоматически завершается. Выполняется действие, заданное вызовом свойством ConnectionDisconnectDisposition.

`Disconnect` можно вызывать сколько угодно раз. Если подключение отсутствует, то ничего не произойдет.

Метод вызывается автоматически при освобождении PCSCCard.

ATR

Свойство содержат ATR подключенной карты.

```
Строка ATR/ATRПодключеннойКарты
```

Описание

Свойство заполняются после выполнения Connect.

Чтобы узнать ATR, нет необходимости выполнять подключение к карте. Получить ATR вставленной в произвольный ридер карты можно при помощи класса PCSCInfo. Если вы хотите получать асихронные уведомления о вставлении карт, то оцените возможности использования классов PCSCCapture и PCSCMonitoring.

Свойство доступно только на чтение.

Response

Свойство содержит результат выполнения предыдущей команды (если она предполагает получение результата).

```
Строка Response/ОтветКарты
```

Описание

Если предыдущее действие являлось прямым отражением APDU команды (например, выполнение функции Transmit, либо выполнение операции в классе, наследованном от `PCSCCard`, сводящейся к Transmit), то Response содержит полный ответ карты, в т.ч. 2 байта `StatusWord` на конце. В этом случае дополнительно заполняется поле StatusWord. Форматы ответов индивидуальны для различных команд и различных типов карт, и доступны в техническом описании по используемому типу карт.

В наследованных классах `Response` может содержать и другие типы результатов, не обязательно являющиеся ответом карты на APDU команду. Такие виды результатов могут не предполагать наличие `StatusWord`, и тогда поле StatusWord не заполняется.

Свойство доступно только на чтение.

StatusWord

Свойство содержит статус выполнения предыдущей команды (если она предполагает наличие статуса).

```
Число StatusWord/СтатусКоманды
```

Описание

`StatusWord` является стандартным элементом ответа карты и располагается в его последних двух байтах. Сначала идет MSB, потом LSB. Для удобства анализа Transmit выделяет `StatusWord` и помещает его в отдельное поле.

Возможные значения `StatusWord` должны быть описаны в технической документации по вашему типу карт.

Свойство доступно только на чтение.

3.2.5 PCSCCard_Memory

Базовый класс для работы с картами памяти картами.

Методы

GetMemoryCardType/ ПолучитьТипКартыПамяти	Получить тип карты памяти

Описание

Класс реализует базовый функционал по абстракции операций подключения к картам памяти на разных моделях ридеров. Создание экземпляров данного класса целесообразно только в случае, если вам необходимо автоматически определять тип вставленной карты. Класс также реализует все методы и свойства класса PCSCCard.

GetMemoryCardType

Получить тип текущей подключенной карты памяти.

```
Число   GetMemoryCardType/ПолучитьТипКартыПамяти([out]   Число
Type);
```

Результат

Одно из следующих значений :

```
MEMORY_CARD_TYPE_UNKNOWN
MEMORY_CARD_TYPE_SLE4428
MEMORY_CARD_TYPE_SLE4442
```

Описание

Функция позволяет определить тип карты памяти, вставленной в ридер. В силу несоответствия карт памяти стандарту PCSC, определение типа карты посредством ATR в общем случае невозможно. Поэтому, сначала требуется установить подключение к карте через метод Connect, затем вызвать функцию `GetMemoryCardType()`.

Ридер ACR38 не поддерживает автоматическое определение типа карты памяти. Функция всегда будет возвращать `MEMORY_CARD_TYPE_UNKNOWN`.

3.2.6 PCSCCard_SLE4428

Класс для работы с картами SLE4428.

Наследование

PCSCCard	Класс содержит все методы и свойства класса PCSCCard.
PCSCCard_Memory	Класс содержит все методы и свойства класса PCSCCard_Memory.

Методы

PresentPIN/ ПредъявитьPIN	Предъявить PIN
WritePIN/ СменитьPIN	Сменить PIN
Read/ ЧтениеДанных	Чтение данных
Write/ ЗаписьДанных	Запись данных
IsByteProtected/ ЗащищенБайт	Защищен ли байт по адресу ... от записи ?
ProtectByte/ ЗащититьОтПерезаписи	Защитить байт по адресу ... от записи.

Описание

Карты памяти накладывают некоторые ограничения на использование функций класса PCSCCard. В частности, ридеры SCM не поддерживают режим разделяемого доступа (SCARD_SHARE_SHARED) при работе с картами памяти. В любом случае перед началом работы с картами памяти следует настроить ридер для работы с ними. Это можно сделать при помощи функции AdjustReaderMode.

PresentPIN

Предъявить PIN.

```
Число PresentPIN/ПредъявитьPIN([in] Строка PIN);
```

Параметры

`[in] PIN`

Двухбайтовый PIN-код.

Описание

Предъявление PIN-кода открывает память карты на запись. Максимальное количество неправильных попыток - 8, после чего PIN-код блокируется. Разблокировка PIN невозможна.

WritePIN

Сменить PIN.

```
Число WritePIN/СменитьPIN([in] Строка PIN);
```

Параметры

`[in] PIN`

Двухбайтовый новый PIN-код.

Описание

Функция записывает в карту новый PIN-код. Для выполнения действия требуется предъявление текущего PIN-кода.

Read

Чтение данных.

```
Число Read/ЧтениеДанных(
    [in] Число Address,
    [in] Число Length);
```

Параметры

`[in] Address`

Адрес. Допустимы значения 0..1020.

`[in] Length`

Длина читаемого блока. Сумма (Address + Length) не должна превышать 1021.

Описание

Чтение данных не требует предъявления PIN. Прочитанный блок сохраняется в свойстве Response.

Write

Запись данных.

```
Число Write/ЗаписьДанных(
    [in] Число Address,
    [in] Строка Data);
```

Параметры

`[in] Address`

Адрес. Допустимы значения 0..1020.

`[in] Data`

Записываемые данные.

Описание

Запись данных требует предъявления PIN. Возможна запись по адресам 0..1020, за исключением защищенных от записи адресов. Попытка записи в защищенные области после предъявления PIN завершится успешно, но фактически защищенные байты не будут перезаписаны.

Обратите внимание : производители карт оставляют за собой право защищать от записи некоторые из первых 32 байтов памяти по своему усмотрению. Мы рекомендуем использовать только ячейки с адресами 32..1020.

IsByteProtected

Проверить защищен ли байт от записи.

```
Число IsByteProtected/ЗащищенБайт(
    [in] Число Address,
    [out] Число Protected);
```

Параметры

`[in] Address`

Адрес. Допустимы значения 0..1023.

Результат

`VARIANT_TRUE` = байт не может быть записан,
`VARIANT_FALSE` = байт может быть записан.

Описание

Функция позволяет определить статус защиты от записи каждой ячейки. Ячейка по адресу 1021 содержит счетчик попыток предъявления PIN-кода, ячейки 1022 и 1023 – PIN-код. Однако, получить статус защищенности байтов 1021..1023 невозможно на ридерах SCM.

Если PIN-код не был предъявлен, то независимо от битов защиты запись в эти ячейки будет запрещена.

ProtectByte

Защитить байт от записи.

```
Число ProtectByte/ЗащититьОтПерезаписи([in] Число Address);
```

Параметры

`[in] Address`

Адрес. Допустимы значения 0..1023.

Описание

Операция требует предъявления PIN и является необратимым действием. Выполнение операции над уже защищенной ячейкой не вызывает ошибки. Ячейка по адресу 1021 содержит счетчик попыток предъявления PIN-кода, ячейки 1022 и 1023 - PIN-код. Вы можете запретить изменение PIN-кода, если защитите от записи байты по адресам 1022 и 1023. Однако, эта операция недоступна на ридерах SCM.

3.2.7 PCSCCard_SLE4442

Класс для работы с картами SLE4442.

Наследование

PCSCCard	Класс содержит все методы и свойства класса PCSCCard.
PCSCCard_Memory	Класс содержит все методы и свойства класса PCSCCard._Memory

Методы

PresentPIN/ ПредъявитьPIN	Предъявить PIN.
WritePIN/ СменитьPIN	Сменить PIN.
Read/ ЧтениеДанных	Чтение данных.
Write/ ЗаписьДанных	Запись данных.
ReadProtectionMemory/ ПрочитатьМаскуЗащиты	Прочитать маску защиты первых 32 байтов от записи.
WriteProtectionMemory/ ЗаписатьМаскуЗащиты	Защитить первые 32 байта от записи по указанной битовой маске.

Описание

Карты памяти накладывают некоторые ограничения на использование функций класса PCSCCard. В частности, ридеры SCM не поддерживают режим разделяемого доступа (SCARD_SHARE_SHARED) при работе с картами памяти. В любом случае перед началом работы с картами памяти следует настроить ридер для работы с ними. Это можно сделать при помощи функции AdjustReaderMode.

PresentPIN

Предъявить PIN.

```
Число PresentPIN/ПредъявитьPIN([in] Строка PIN);
```

Параметры

`[in] PIN`

Трехбайтовый PIN-код.

Описание

Предъявление PIN-кода открывает память карты на запись. Максимальное количество неправильных попыток – 3, после чего PIN-код блокируется. Разблокировка PIN невозможна.

WritePIN

Сменить PIN.

```
Число WritePIN/СменитьPIN([in] Строка PIN);
```

Параметры

`[in] PIN`

Трехбайтовый новый PIN-код.

Описание

Функция записывает в карту новый PIN-код. Для выполнения действия требуется предъявление текущего PIN-кода.

Read

Чтение данных.

```
Число Read/ЧтениеДанных(
    [in] Число Address,
    [in] Число Length);
```

Параметры

`[in] Address`

Адрес.

`[in] Length`

Длина читаемого блока. Сумма (Address + Length) не должна превышать 256.

Описание

Чтение данных не требует предъявления PIN. Прочитанный блок сохраняется в свойстве Response.

Write

Запись данных.

```
Число Write/ЗаписьДанных(
    [in] Число Address,
    [in] Строка Data);
```

Параметры

`[in] Address`

Адрес.

`[in] Data`

Записываемые данные.

Описание

Запись данных требует предъявления PIN. Возможна запись во все ячейки, за исключением защищенных от записи адресов (реализована побайтовая защита адресов 0..31). Попытка записи в защищенные области после предъявления PIN завершится успешно, но фактически защищенные байты не будут перезаписаны.

Обратите внимание – производители карт оставляют за собой право защищать от записи некоторые из первых 32 байтов памяти по своему усмотрению.

ReadProtectionMemory

Прочитать маску защиты первых 32 байтов от записи.

```
Число ReadProtectionMemory/ПрочитатьМаскуЗащиты([out] Число
ProtectionBitMask);
```

Результат

Битовая маска защиты от записи. Номер бита (0 – младший, 31 – старший) определяет адрес. 1 означает разрешение записи, 0 – запрет записи.

WriteProtectionMemory

Защитить первые 32 байта от записи по указанной битовой маске.

```
Число WriteProtectionMemory/ЗаписатьМаскуЗащиты(
[in] Число ProtectionBitMask);
```

Параметры

`[in] ProtectionBitMask`

Битовая маска защиты от записи. Возьмите за основу значение 0xFFFFFFFF. Сбросьте в ноль биты, соответствующие адресам ячеек, которые вы хотите защитить от записи. Номер бита (0 – младший, 31 – старший) определяет адрес ячейки.

Описание

Операция требует предъявления PIN-кода. Защита битов необратима. Попытка снять защиту с ячеек не приведет к успеху.

3.2.8 PCSCCard_ACOS

Класс для работы с картами ACOS.

Наследование

PCSCCard	Класс содержит все методы и свойства класса PCSCCard.

Методы

Получение информации о карте

GetVersion/ВерсияКарты	Получить версию карты.
IsBuggedACOS3_KcKt/КартаСОшибкой	Определить наличие ошибки, связанной с ключами взаимной аутентификации.
GetStage/ПолучитьСтадиюКарты	Получить стадию карты.
GetSerialNumber/ПолучитьСерийныйНомер	Получить серийный номер.
ReadManufacturerOptions/ПолучитьНастройкиСтадииПроизводства	Получить настройки стадии производства.
ReadPersonalizationOptions/ПолучитьНастройкиПерсонализации	Получить настройки персонализации.
ReadFileDefinitionBlockEx/ПрочитатьДескрипторФайла	Прочитать дескриптор файла.
ReadBinaryFileDefinitionBlock/ПрочитатьДескрипторДвоичногоФайла	Прочитать дескриптор двоичного файла.
GetMemorySpace/ПолучитьОбьемПамяти	Получить объем памяти.

Управление картой и персонализация

SetCardSpeed/ УстановитьСкоростьДоступа	Изменить скорость доступа к карте
GetCardSpeed/ ПолучитьСкоростьДоступа	Получить скорость доступа к карте
Manufacture/ УстановитьНастройкиПроизводства	Изменить настройки стадии производства.
Personalize/ УстановитьНастройкиПерсонализации	Изменить настройки стадии персонализации.
ClearCard/ОчиститьКарту	Очистить карту
GetMaximumFileRecordsCount/ ПолучитьМаксимальныйРазмерФайла	Получить максимально возможный размер файла записей.
GetMaximumBinaryFileSize/ ПолучитьМаксимальныйРазмерДвоичногоФайла	Получить максимально возможный размер двоичного файла.
WriteFileDefinitionBlockEx/ ЗаписатьДескрипторФайла	Записать дескриптор файла.
WriteBinaryFileDefinitionBlock/ ЗаписатьДескрипторДвоичногоФайла	Записать дескриптор двоичного файла.
ZeroFileDefinitionBlock/ ОбнулитьДескрипторФайла	Обнулить дескриптор файла.
ZeroAllFileDefinitionBlocksFrom/ ОбнулитьВсеДескрипторыФайлов	Обнулить все дескрипторы файлов, начиная с указанного.
WriteSecurityFile/ СохранитьКодДоступа	Сохранить код доступа.
WriteSecurityFile3/ СохранитьКодДоступа3DE	Сохранить код доступа (3DES).

S	
WriteAccountSecurityFile1/ СохранитьКлючКошелькаDES	Сохранить ключ доступа кошелька (DES).
WriteAccountSecurityFile3/ СохранитьКлючКошелька3DES	Сохранить ключ доступа кошелька (3DES).
InitAccount/ ИнициализироватьКошелек	Инициализировать кошелек.

Аутентификация

Verify/ ПроверитьКодДоступа	Проверить код доступа.
ChangePIN/ИзменитьPIN	Сменить PIN-код.
MakeBinaryCode/ СоздатьДвоичныйКод	Получить 8-байтный код из строки.
MutualAuthenticate1/ ВзаимнаяАутентификацияDES	Взаимная аутентификация по алгоритму DES.
MutualAuthenticate3/ ВзаимнаяАутентификация3DES	Взаимная аутентификация по алгоритму 3DES.
StartSession/ НачатьСессиюАутентификации	Начать сессию взаимной аутентификации.
GetRndT/ СлучайноеЧислоТерминала	Сформировать случайное число терминала.
EncryptWithSessionKey/ Зашифровать	Зашифровать по ключу сессии.
DecryptWithSessionKey/ Расшифровать	Расшифровать по ключу сессии.

Операции с файлами

SelectFile/ВыбратьФайл	Выбрать файл.
ReadRecord/ ПрочитатьЗапись	Прочитать запись.
WriteRecord/ СохранитьЗапись	Сохранить запись.
ReadBinaryFile/ ЧтениеДвоичногоФайла	Чтение двоичного файла.
WriteBinaryFile/ ЗаписьДвоичногоФайла	Запись в двоичный файл.

Операции с кошельком

InquireAccount/ ОпросКошелька	Опрос кошелька.
Credit/ЗачислитьНаСчет	Зачисление суммы на счет.
Debit/СнятьСоСчета	Снятие суммы со счета.
RevokeDebit/ ОтменитьСнятиеСоСчета	Отмена снятия суммы со счета.

Secure messaging

SMEnable/ВклВыклSM	Включить/отключить secure messaging.
SMIsEnabled/SMВключен	Проверить включен ли secure messaging.
TransmitSM/ПередатьCSM	Передать команду с использованием secure messaging
TransmitSMWithCommonSWCheck/ ПередатьSMПроверитьСтатус	Передать команду с использованием secure messaging + проверить статус

Другие операции

GetResponse/ ПолучитьОтвет	Получить ответ.

Свойства

ATRCheckEnforced/ ATRКартыACOS	Проверять ли соответствие ATR карте ACOS

Описание

С подробным описанием архитектуры карт ACOS можно ознакомиться по оригинальным документам : ACOS2.PDF, ACOS3.PDF, ACOS3-24*.PDF. Обратим внимание лишь на особенности, касающиеся реализации функций ACOS в SDK.

- Класс позволяет задействовать все возможности карт ACOS, за исключением одной : не поддерживается опция производителя `RECORD_NUMBERING` (нумерация записей файлов с единицы).
- Алгоритм 3DES в реализации ACOS по сути является 2DES (1 и 3 1DES ключи совпадают). Везде, где упоминается алгоритм 3DES в отношении карт ACOS, следует это понимать как алгоритм 2DES. Ключ 2DES делится на 2 составляющих по 8 байт : левая половина (первая часть) и правая половина (вторая часть).

Получение информации о карте

GetVersion

Получить версию карты.

```
Число GetVersion/ВерсияКарты([out,retval] Число Version);
```

Результат

Версия карты. Одно из следующих значений :

Значение	Описание
ACOS_TYPE_12	ACOS1 и ACOS2. Эти карты отличаются только скоростью передачи данных. В остальном, с точки зрения программиста они неотличимы.
ACOS_TYPE_3_8K	ACOS3 8K
ACOS_TYPE_3_16K	ACOS3 16K
ACOS_TYPE_3_24K	ACOS3 24K (добавлены binary files, secure messaging)
ACOS_TYPE_3_64K	ACOS3 64K (добавлены binary files, secure messaging)

Примечание

Числовые значения констант версий идут в порядке возрастания – старшая версия карты имеет большее числовое значение версии. Поэтому возможны операции сравнения : например, `Version>=ACOS_TYPE_3_24K`.

IsBuggedACOS3_KcKt

Определить наличие ошибки, связанной с ключами взаимной аутентификации.

```
Число IsBuggedACOS3_KcKt/КартаСОшибкой(
[out,retval] Число Bugged);
```

Результат

VARIANT_TRUE - ошибка присутствует,

VARIANT_FALSE - ошибка отсутствует.

Описание

Суть проблемы в картах ACOS3 ранних версий описана в файле ACOS3_bug.pdf. Скорее всего, для вас ее наличие или отсутствие останется прозрачным : реализация класса автоматически обходит ошибку, если она присутствует. Ошибка относится к разряду несовместимостей. Урозы для безопасности нет.

GetStage

Получить стадию карты.

```
Число GetStage/ПолучитьСтадиюКарты([out,retval] Число Stage);
```

Результат

Стадия карты. Одно из следующих значений :

Значение	Описание
ACOS_STAGE_MANUFACTURING	Стадия производства.
ACOS_STAGE_PERSONALIZATION	Стадия персонализации.
ACOS_STAGE_USER	Стадия использования.

Примечание

Карты могут поставляться в стадии производства или в стадии персонализации. Ваше приложение не должно быть завязано на одну из этих стадий и должно корректно персонализировать карту в обоих случаях. Вы можете расчитывать, что карты не будут в стадии использования.

GetSerialNumber

Получить серийный номер карты.

```
Число       GetSerialNumber/ПолучитьСерийныйНомер([out,retval]
Строка Serial);
```

Результат

Серийный номер карты длиной 8 байт.

GetMemorySpace

Получить объем памяти.

```
Число GetMemorySpace/ПолучитьОбъемПамяти(
    [out] Число TotalMemory,
    [out] Число FreeMemory);
```

Параметры

`[out] TotalMemory`

Полный объем памяти, байт.

`[out] FreeMemory`

Свободно памяти, байт.

Описание

Полученные значения могут быть использованы только для наглядного представления пользователю информации о свободной памяти на карте. Для получения максимального размера вновь создаваемых файлов используйте методы GetMaximumFileRecordsCount и GetMaximumBinaryFileSize.

Управление картой и персонализация

SetCardSpeed

Изменить скорость доступа к карте.

```
Число    SetCardSpeed/УстановитьСкоростьДоступа([in]    Число
Speed);
```

Параметры

`[in] BaudRate`

Возможные значения : 9600,13950,27900,55800,111600,223200, ACOS_SPEED_MAX (максимально возможная скорость для текущей карты).

Описание

Скорость доступа к карте настраивается внутри самой карты (не в ридере) и сохраняется в энергонезависимой памяти. Изменение скорости происходит при следующем подключении к карте после ее сброса.

Карты ACOS2 не имеют возможности менять скорость : единственная допустимая скорость - 9600.

Карты ACOS3-8K,ACOS3-16K допускают установку скорости 9600..111600.

Карты ACOS3-24K и выше поддерживают все скорости.

При указании `dwBaudRate=ACOS_SPEED_MAX` функция не возвращает ошибки независимо от типа вставленной карты.

Не все ридеры способны работать на повышенных скоростях. Старые модели ридеров могут не иметь такой возможности.

GetCardSpeed

Получить скорость доступа к карте.

```
Число GetCardSpeed/ПолучитьСкоростьДоступа([out,retval] Число
Speed);
```

Результат

Скорость доступа к карте (бит/сек).

Manufacture

Изменить настройки стадии производства.

```
Число Manufacture/УстановитьНастройкиПроизводства(
[in] Число OptionRegister);
```

Параметры

Регистр настроек стадии производства. Битовое поле :

Значение	Описание
ACOS_MANOPT_INQ_ACC_MAC	Учитывать TTREF_C, TTREF_D при вычислении криптографической подписи в процедуре InquireAccount.
ACOS_MANOPT_BLOW_FUSE	Перевести карту в стадию персонализации. Операция необратима.

Описание

Мы рекомендуем менять стадию карты в процессе персонализации, только если ваше приложение будет работать с электронным кошельком, или существует угроза компрометации IC кода. Во всех остальных случаях с точки зрения безопасности нет угрозы оставлять карту в стадии производства или в стадии персонализации. Выполнение необратимых действий неоправданно.

Для успешного выполнения операции требуется предъявление IC кода.

ReadManufacturerOptions

Получить настройки стадии производства.

```
Число                                    ReadManufacturerOptions/
ПолучитьНастройкиСтадииПроизводства([out,retval]         Число
OptionRegister);
```

Результат

Регистр настроек стадии производства. Если значение не требуется, можно указать 0. Битовое поле. Значения битов см. в описании метода Manufacture.

Personalize

Изменить настройки стадии персонализации.

```
Число Personalize/УстановитьНастройкиПерсонализации(
    [in] Число OptionRegister,
    [in] Число SecurityOptionRegister,
    [in] Число N_OF_FILE,
    [in] Число BlowFuse);
```

Параметры

`[in] OptionRegister`

Регистр настроек стадии персонализации. Битовое поле :

Значение	Описание
ACOS_PERSOPT_ACCOUNT	Поддерживается работа с электронным кошельком.
ACOS_PERSOPT_3DES	Использовать алгоритм 3DES вместо DES во всех операциях.
ACOS_PERSOPT_PIN_ALT	Разрешена смена PIN-кода операцией ChangePIN.
ACOS_PERSOPT_DEB_MAC	Для списания суммы с кошелька требуется ключ дебетования.
ACOS_PERSOPT_DEB_PIN	Для списания суммы с кошелька требуется проверка PIN-кода.
ACOS_PERSOPT_REV_DEB	Разрешена отмена последнего дебетования операцией RevokeDebit.
ACOS_PERSOPT_TRNS_AUT	Для проведения транзакций с кошельком (Credit, Debit, RevokeDebit) требуется проведение взаимной аутентификации (MutualAuthenticate1, MutualAuthenticate3).
ACOS_PERSOPT_I	Для опроса кошелька операцией InquireAccount

NQ_AUT	требуется проведение взаимной аутентификации (MutualAuthenticate1, MutualAuthenticate3).

[in] SecurityOptionRegister

Регистр настроек способов аутентификации. Битовое поле :

Значение	Описание
ACOS_SECOPT_IC_DES	IC код предъявляется в шифрованном виде.
ACOS_SECOPT_PIN_DES	PIN код предъявляется в шифрованном виде.
ACOS_SECOPT_AC1_DES	AC1 код предъявляется в шифрованном виде.
ACOS_SECOPT_AC2_DES	AC2 код предъявляется в шифрованном виде.
ACOS_SECOPT_AC3_DES	AC3 код предъявляется в шифрованном виде.
ACOS_SECOPT_AC4_DES	AC4 код предъявляется в шифрованном виде.
ACOS_SECOPT_AC5_DES	AC5 код предъявляется в шифрованном виде.

[in] N_OF_FILE

Количество файловых дескрипторов (максимальное количество файлов).

[in] BlowFuse

VARIANT_TRUE = перевести карту в стадию использования,
VARIANT_FALSE = не переводить. Изменение стадии необратимо !

Описание

Мы рекомендуем менять стадию карты в процессе персонализации, только если ваше приложение будет работать с электронным кошельком, или существует угроза компрометации IC-кода. Во всех остальных случаях с точки зрения безопасности нет угрозы оставлять карту в стадии производства или в стадии персонализации. Выполнение необратимых действий неоправданно.

Для успешного выполнения операции требуется предъявление IC-кода.

ReadPersonalizationOptions

Получить настройки стадии персонализации.

```
Число                                   ReadPersonalizationOptions/
ПолучитьНастройкиПерсонализации(
    [out] Число OptionRegister,
    [out] Число SecurityOptionRegister,
    [out] Число N_OF_FILE);
```

Параметры

`[out] OptionRegister`

Регистр настроек стадии персонализации.

`[out] SecurityOptionRegister`

Регистр настроек способов аутентификации.

`[out] N_OF_FILE`

Количество файловых дескрипторов (максимальное количество файлов).

Примечание

Описание полученных значений см. в описании метода Personalize.

ClearCard

Очистить карту.

```
Число ClearCard/ОчиститьКарту();
```

Описание

Привести карту в исходное состояние. Все структуры данных и файлы уничтожаются, IC-код сбрасывается в значение "ACOSTEST". Функция доступна только в картах ACOS3 в стадии производства или персонализации. После очистки стадия не меняется.

Для успешного выполнения операции требуется предъявление IC-кода.

GetMaximumFileRecordsCount

Получить максимально возможный размер файла записей.

```
Число                                   GetMaximumFileRecordsCount/
ПолучитьМаксимальныйРазмерФайла(
    [in] Число RecordLength,
    [out,retval] Число MaxRecords);
```

Параметры

`[in] RecordLength`

Размер записи нового файла.

Результат

Максимальное количество записей в файле.

Описание

Функция позволяет определить максимально возможный размер вновь создаваемого файла в записях на основе имеющейся свободной памяти, не занятой другими файлами и структурами данных.

GetMaximumBinaryFileSize

Получить максимально возможный размер двоичного файла.

```
Число GetMaximumBinaryFileSize/
ПолучитьМаксимальныйРазмерДвоичногоФайла([out,retval] Число
MaxSize);
```

Результат

Максимально возможный размер двоичного файла, байт.

Описание

Функция позволяет определить максимально возможный размер вновь создаваемого двоичного файла на основе имеющейся свободной памяти, не занятой другими файлами и структурами данных.

Поддержка двоичных файлов отсутствуют в картах ACOS. Однако, для удобства использования SDK предоставляет свою программную эмуляцию, которая основывается на файлах записей.

WriteFileDefinitionBlockEx

Записать дескриптор файла.

```
Число WriteFileDefinitionBlockEx/ЗаписатьДескрипторФайла(
    [in] Число FileDescriptorIdx,
    [in] Число FID,
    [in] Число RecordLength,
    [in] Число NumberOfRecords,
    [in] Число BinaryFileSize,
    [in] Число ReadSA,
    [in] Число WriteSA,
    [in] Число FileAccessFlags);
```

Параметры

`[in] FileDescriptorIdx`

Номер файлового дескриптора. Допустимый диапазон - 0..(N_OF_FILE-1).

`[in] FID`

Идентификатор файла. Любое значение из диапазона 0x0001..0xFEFF.

`[in] RecordLength`

Длина записи. 1..32 для карт ACOS1/2. 1..255 для карт ACOS3. Значение используется только при отсутствии параметра `ACOS_FA_BINARY`. В этом случае `wBinaryFileSize` игнорируется.

`[in] NumberOfRecords`

Количество записей в файле. Максимально допустимое количество записей можно определить с помощью метода GetMaximumFileRecordsCount. Значение используется только при отсутствии параметра `ACOS_FA_BINARY`. В этом случае `BinaryFileSize` игнорируется.

`[in] BinaryFileSize`

Длина двоичного файла (байт). Значение используется только при указании параметра ACOS_FA_BINARY. В этом случае RecordLength, NumberOfRecords игнорируются.

`[in] ReadSA`

Аттрибут безопасности на чтение файла (см. Аттрибуты доступа к файлам).

`[in] WriteSA`

Аттрибут безопасности на запись файла (см. Аттрибуты доступа к файлам).

`[in] FileAccessFlags`

Параметры доступа к файлу. Битовое поле :

Значение	Описание
ACOS_FA_BINARY	Двоичный файл, поддерживаемый аппаратно. (ACOS3-24K и выше)
ACOS_FA_READ_SECURE	Чтение требует secure messaging.(ACOS3-24K и выше)
ACOS_FA_WRITE_SECURE	Запись требует secure messaging.(ACOS3-24K и выше)

Описание

Карты ACOS не поддерживают динамического создания/удаления файлов в полной мере. Адрес расположения N-го файла внутри EEPROM определяется на основании дескрипторов 0..(N-1). Это значит, что если вы измените дескриптор некоторого файла, то все последующие файлы "съедут". Поменяется их базовый адрес в EEPROM, и из файла будет читаться уже совсем другая информация. Учитывайте этот факт при переразметке карты.

Для успешного выполнения операции требуется предъявление IC кода.

WriteBinaryFileDefinitionBlock

Записать дескриптор двоичного файла.

```
Число                              WriteBinaryFileDefinitionBlock/
ЗаписатьДескрипторДвоичногоФайла(
    [in] Число FileDescriptorIdx,
    [in] Число FID,
    [in] Число FileSize,
    [in] Число ReadSA,
    [in] Число WriteSA,
    [in] Число FileAccessFlags,
    [out,retval] Число RealFileSize);
```

Параметры

`[in] FileDescriptorIdx`

Номер файлового дескриптора. Допустимый диапазон - 0..(N_OF_FILE-1).

`[in] FID`

Идентификатор файла. Любое значение из диапазона 0x0001..0xFEFF.

`[in] FileSize`

Длина файла, байт.

`[in] ReadSA`

Аттрибут безопасности на чтение файла (см. Аттрибуты доступа к файлам).

`[in] WriteSA`

Аттрибут безопасности на запись файла (см. Аттрибуты доступа к файлам).

`[in] FileAccessFlags`

Параметры доступа к файлу. Битовое поле :

Значение	Описание
`ACOS_FA_READ_SECURE`	Чтение требует secure messaging.(ACOS3-24K и выше)
`ACOS_FA_WRITE_SECURE`	Запись требует secure messaging.(ACOS3-24K и выше)

Результат

Реальный размер созданного файла. Может быть больше, но не меньше, запрашиваемого..

Описание

Поддержка двоичных файлов отсутствуют в картах ACOS1,ACOS2,ACOS3-8K, ACOS3-16K и присутствует в картах ACOS3-24K и выше. Однако, в случае использования старой модели карты, SDK предоставляет прозрачную программную эмуляцию, которая основывается на файлах записей. Функция сама решает использовать для аппаратную реализацию двоичных файлов или эмуляцию через файлы записей. Для потребителя двоичный файл представляет собой непрерывную последовательность байтов, независимо от фактической реализации хранилища данных.

Функции чтения и записи двоичных файлов могут работать с любыми типами файлами (двоичные, файлы записей), с любым количеством записей и размером записи. При использовании эмуляции для уменьшения количества операций с картой, и, соответственно, повышения скорости обмена, метод `WriteBinaryFileDefinitionBlock` автоматически выбирает оптимальный размер файла. Для карт ACOS1/2 - это 32 байта, для карт ACOS3-8K,ACOS3-16K - 128 байт.

Для успешного выполнения операции требуется предъявление IC-кода.

ReadFileDefinitionBlockEx

Прочитать дескриптор файла.

```
Число ReadFileDefinitionBlockEx/ПрочитатьДескрипторФайла(
    [in] Число FileDescriptorIdx,
    [out] Число FID,
    [out] Число RecordLength,
    [out] Число NumberOfRecords,
    [out] Число BinaryFileSize,
    [out] Число ReadSA,
    [out] Число WriteSA,
    [out] Число FileAccessFlags);
```

Параметры

`[in] FileDescriptorIdx`

Номер файлового дескриптора. Допустимый диапазон - 0..(N_OF_FILE-1).

`[out] FID`

Идентификатор файла.

`[out] RecordLength`

Длина записи. Значение используется только если файт не имеет флага доступа `ACOS_FA_BINARY`. В этом случае `pwBinaryFileSize` заполняется нулями.

`[out] NumberOfRecords`

Количество записей в файле. Значение используется только если файт не имеет флага доступа `ACOS_FA_BINARY`. В этом случае `BinaryFileSize` заполняется нулями.

`[out] BinaryFileSize`

Длина двоичного файла (байт). Значение используется только если файт имеет флаг доступа `ACOS_FA_BINARY`. В этом случае RecordLength, NumberOfRecords заполняются нулями.

`[out] ReadSA`

Аттрибут безопасности на чтение файла (см. Аттрибуты доступа к файлам).

`[out] WriteSA`

Аттрибут безопасности на запись файла (см. Аттрибуты доступа к файлам).

`[out] FileAccessFlags`

Параметры доступа к файлу. Битовое поле :

Значение	Описание
`ACOS_FA_BINARY`	Двоичный файл, поддерживаемый аппаратно. (ACOS3-24K и выше)
`ACOS_FA_READ_SECURE`	Чтение требует secure messaging.(ACOS3-24K и выше)
`ACOS_FA_WRITE_SECURE`	Запись требует secure messaging.(ACOS3-24K и выше)

.

ReadBinaryFileDefinitionBlock

Прочитать дескриптор двоичного файла.

```
Число                                ReadBinaryFileDefinitionBlock/
ПрочитатьДескрипторДвоичногоФайла(
    [in] Число FileDescriptorIdx,
    [out] Число FID,
    [out] Число FileSize,
    [out] Число ReadSA,
    [out] Число WriteSA,
    [out] Число FileAccessFlags);
```

Параметры

`[in] FileDescriptorIdx`

Номер файлового дескриптора. Допустимый диапазон - 0..(N_OF_FILE-1).

`[out] FID`

Идентификатор файла.

`[out] FileSize`

Длина файла, байт.

`[out] ReadSA`

Аттрибут безопасности на чтение файла (см. Аттрибуты доступа к файлам).

`[out] WriteSA`

Аттрибут безопасности на запись файла (см. Аттрибуты доступа к

файлам).

```
[out] FileAccessFlags
```

Параметры доступа к файлу. Битовое поле :

Значение	Описание
ACOS_FA_READ_SECURE	Чтение требует secure messaging.(ACOS3-24K и выше)
ACOS_FA_WRITE_SECURE	Запись требует secure messaging.(ACOS3-24K и выше)

Описание

Функция аналалогична ReadFileDefinitionBlockEx, только в случае файла записей вместо размера и количества записей возвращается их произведение, соответствующее размеру двоичного файла, построенного на файле записей. В случае двоичного файла, реализованного аппаратно, возвращается его фактическая длина.

ZeroFileDefinitionBlock

Обнулить дескриптор файла.

```
Число ZeroFileDefinitionBlock/ОбнулитьДескрипторФайла(
    [in] Число FileDescriptorIdx);
```

Параметры

`[in] FileDescriptorIdx`

Номер файлового дескриптора. Допустимый диапазон - 0..(N_OF_FILE-1).

Описание

Карты ACOS не проверяют корректность записанной в дескриптор файла информации. В случае неправильного заполнения ошибки проявляются уже в процессе работы с файлом. Используйте функцию `ZeroFileDefinitionBlock`, чтобы заполнить дескриптор нейтральными данными. Такое заполнение не повлияет на корректность работы с остальными файлами и на правильность вычисления свободной памяти.

Карты ACOS не поддерживают динамического создания/удаления файлов в полной мере. Адрес расположения N-го файла внутри EEPROM определяется на основании дескрипторов 0..(N-1). Это значит, что если вы измените дескриптор некоторого файла, то все последующие файлы "съедут". Поменяется их базовый адрес в EEPROM, и из файла будет читаться уже совсем другая информация. Учитывайте этот факт при переразметке карты.

Для успешного выполнения операции требуется предъявление IC-кода.

ZeroAllFileDefinitionBlocksFrom

Обнулить все дескрипторы файлов, начиная с указанного.

```
Число                          ZeroAllFileDefinitionBlocksFrom/
ОбнулитьВсеДескрипторыФайлов(
    [in] Число FileDescriptorIdx);
```

Параметры

`[in] FileDescriptorIdx`

Номер файлового дескриптора. Допустимый диапазон - 0..(N_OF_FILE-1).

Описание

Производится обнуление всех дескрипторов файлов в диапазоне btFileDescriptorIdx..(N_OF_FILE-1). Функция `ZeroAllFileDefinitionBlocksFrom` работает быстрее, чем многократный вызов ZeroFileDefinitionBlock.

Для успешного выполнения операции требуется предъявление IC-кода.

WriteSecurityFile

Сохранить код доступа.

```
Число WriteSecurityFile/СохранитьКодДоступа(
    [in] Число CodeIdx,
    [in] Строка Code);
```

Параметры

`[in] CodeIdx`

Номер кода доступа. Возможны следующие значения :

Значение	Описание
ACOS_SFN_IC	IC-код (код издателя, транспортный ключ).
ACOS_SFN_PIN	PIN-код.
ACOS_SFN_CARD	Ключ карты. Используется при взаимной аутентификации.
ACOS_SFN_CARD 2	Ключ карты (правая половина 3DES ключа).
ACOS_SFN_TERM	Ключ терминала. Используется при взаимной аутентификации.
ACOS_SFN_TERM 2	Ключ терминала (правая половина 3DES ключа).
ACOS_SFN_AC1	Код приложения AC1.
ACOS_SFN_AC2	Код приложения AC2.
ACOS_SFN_AC3	Код приложения AC3.
ACOS_SFN_AC4	Код приложения AC4.
ACOS_SFN_AC5	Код приложения AC5.

```
[in] Code
```

Значение кода доступа длиной 8 байт.

Описание

Все коды доступа в карте ACOS имеют размер 8 байт. Если идет работа с 3DES ключами, то они разделяются на 2 половины - левую и правую.

Вероятно, вам потребуется представлять такие коды, как IC, PIN, ACx в виде строки, которую может ввести пользователь. Для перевода строки в бинарный 8-байтовый блок согласно стандарту используйте метод MakeBinaryCode. Ключи взаимной аутентификации, как правило, формируются как случайная последовательность байт.

В процессе персонализации обязательно меняйте IC код. В противном случае защита карты неэффективна.

Метод WriteSecurityFile автоматически и прозрачно обходит ошибку карт ACOS3.

Для успешного выполнения операции требуется предъявление IC кода.

WriteSecurityFile3

Сохранить код доступа 3DES.

```
Число WriteSecurityFile3/СохранитьКодДоступа3DES(
    [in] Число CodeIdx,
    [in] Строка Code);
```

Параметры

`[in] CodeIdx`

Номер кода доступа. Возможны следующие значения :

Значение	Описание
ACOS_SFN_CARD	Ключ карты. Используется при взаимной аутентификации.
ACOS_SFN_TERM	Ключ терминала. Используется при взаимной аутентификации.

`[in] Code`

Значение кода доступа длиной 16 байт (левая + правая половины).

Описание

Все коды доступа в карте ACOS имеют размер 8 байт. Если идет работа с 3DES ключами, то они разделяются на 2 половины - левую и правую. Функция `WriteSecurityFile3` позволяет сохранить сразу 2 половины ключа в нужные ячейки.

Метод `WriteSecurityFile3` автоматически и прозрачно обходит ошибку карт ACOS3.

Для успешного выполнения операции требуется предъявление IC-кода.

WriteAccountSecurityFile1

Сохранить ключ доступа кошелька (DES).

```
Число WriteAccountSecurityFile1/СохранитьКлючКошелькаDES(
    [in] Число CodeIdx,
    [in] Строка Code);
```

Параметры

`[in] CodeIdx`

Номер ключа. Возможны следующие значения :

Значение	Описание
ACOS_ASFN_DEBIT	Код дебетования.
ACOS_ASFN_CREDIT	Код кредитования.
ACOS_ASFN_INQUIRE	Код опроса кошелька.
ACOS_ASFN_REVOKE	Код отмены дебетования.

`[in] Code`

Значение ключа длиной 8 байт.

Описание

Ключи доступа кошелька, как правило, формируются как случайная последовательность байт.

Для успешного выполнения операции требуется предъявление IC-кода.

WriteAccountSecurityFile3

Сохранить ключ доступа кошелька (3DES).

```
Число WriteAccountSecurityFile3/СохранитьКлючКошелька3DES(
    [in] Число CodeIdx,
    [in] Строка Code);
```

Параметры

`[in] CodeIdx`

Номер ключа. Возможны следующие значения :

Значение	Описание
ACOS_ASFN_DEBIT	Код дебетования.
ACOS_ASFN_CREDIT	Код кредитования.
ACOS_ASFN_INQUIRE	Код опроса кошелька.
ACOS_ASFN_REVOKE	Код отмены дебетования.

`[in] Code`

Значение ключа длиной 16 байт (левая + правая половины).

Описание

Все коды доступа в карте ACOS имеют размер 8 байт. Если идет работа с 3DES ключами, то они разделяются на 2 половины - левую и правую. Функция `WriteAccountSecurityFile3` ползволяет записать сразу 2 половины ключа в нужные ячейки.

Ключи доступа кошелька, как правило, формируются как случайная последовательность байт.

Для успешного выполнения операции требуется предъявление IC-кода.

InitAccount

Инициализация кошелька.

```
Число InitAccount/ИнициализироватьКошелек(
    [in] Число AID,
    [in] Число InitialBalance,
    [in] Число MaximumBalance,
    [in] Число TTREF_C,
    [in] Число TTREF_D);
```

Параметры

`[in] AID`

Идентификатор приложения. Произвольное значение, не интерпретируется картой.

`[in] InitialBalance`

Начальная сумма на счету.

`[in] MaximumBalance`

Максимальная сумма на счету. Допустимые значения 0..16777215 (0..0xFFFFFF).

`[in] TTREF_C`

Идентификатор терминала кредитования. Произвольное значение, не интерпретируется картой.

`[in] TTREF_D`

Идентификатор терминала дебетования. Произвольное значение, не интерпретируется картой.

Описание

В картах ACOS сумма представляет собой трехбайтовое целое число. Если необходимо хранить копейки или центы, используйте деление/умножение на 100 в вашей программе. Оцените устроит ли вас получившийся диапазон возможных сумм.

Для успешного выполнения операции требуется предъявление IC-кода.

Аттрибуты доступа к файлам

Аттрибуты доступа позволяют определить условия доступа к файлам и представляют собой битовое поле размером 1 байт.

Значение	Описание
ACOS_AC_IC	Требуется IC-код.
ACOS_AC_PIN	Требуется PIN-код.
ACOS_AC_AC1 ACOS_AC_AC2 ACOS_AC_AC3 ACOS_AC_AC4 ACOS_AC_AC5	Требуется любой из указанных кодов AC1..AC5.
ACOS_AC_NEVER	Доступ всегда запрещен.
ACOS_AC_ALWAYS	Доступ всегда разрешен.

IC и PIN соединяются по логическому "И", ACx - по логическому "ИЛИ".
Например, аттрибут доступа

```
AC = ACOS_AC_IC | ACOS_IC_PIN | ACOS_IC_AC2 | AC_IC_AC5
```

означает следующую схему требования кодов :

```
IC and PIN and (AC2 or AC5).
```

Аутентификация

Verify

Проверить код доступа.

```
Число Verify/ПроверитьКодДоступа(
    [in] Число CodeNumber,
    [in] Строка Code,
    [in] Число SubmitEncrypted);
```

Параметры

`[in] CodeNumber`

Номер кода доступа. Возможны следующие значения : ACOS_CODE_AC1, ACOS_CODE_AC2, ACOS_CODE_AC3, ACOS_CODE_AC4, ACOS_CODE_AC5, ACOS_CODE_PIN, ACOS_CODE_IC.

`[in] Code`

Значение кода доступа. Указатель на область памяти длиной 8 байт.

`[in] SubmitEncrypted`

`VARIANT_TRUE` = проверять код в зашифрованном виде,
`VARIANT_FALSE` = проверять код в открытом виде.

Описание

Карта имеет внутренний регистр состояния проверок кодов доступа. Состояние "проверенности" кода сохраняется до сброса карты. Нет команды, позволяющей аннулировать статус проверки, поэтому при работе с картами ACOS по умолчанию устаналивается режим отключения `SCARD_RESET_CARD`.

Способ проверки кода определяется содержимым регистра настроек способов аутентификации. Параметр `SubmitEncrypted` должен соответствовать установленному режиму проверки кода.

Если код проверяется в зашифрованном виде, то требуется предварительное прохождение процедуры взаимной аутентификации (методы MutualAuthenticate1,

MutualAuthenticate3).

Вероятно, вам потребуется представлять коды доступа в виде строки, которую может ввести пользователь. Для перевода строки в бинарный 8-байтовый блок согласно стандарту используйте метод MakeBinaryCode.

ChangePIN

Сменить PIN код.

```
Число ChangePIN/ИзменитьPIN(
    [in] Строка NewPin,
    [in] Число PinIsEncrypted);
```

Параметры

`[in] NewPin`

Новое значение PIN-кода длиной 8 байт.

`[in] SubmitEncrypted`

`VARIANT_TRUE` = менять код в зашифрованном виде, `VARIANT_FALSE` = менять код в открытом виде.

Описание

Для изменения PIN кода требуется сначала проверить его текущее значение. Необходимо, чтобы в регистре настроек стадии персонализации была включана опция ACOS_PERSOPT_PIN_ALT.

Способ проверки и изменения PIN кода определяется содержимым регистра настроек способов аутентификации. Параметр SubmitEncrypted должен соответствовать установленному режиму проверки кода.

Вероятно, вам потребуется представлять PIN код в виде строки, которую может ввести пользователь. Для перевода строки в бинарный 8-байтовый блок согласно стандарту используйте метод MakeBinaryCode.

MakeBinaryCode

Получить 8-байтный код из строки.

```
Число MakeBinaryCode/СоздатьДвоичныйКод(
    [in] Строка CodeStr,
    [out,retval] Строка CodeBin);
```

Параметры

[in] lpCode

Строковое значение кода.

Результат

[SAFEARRAY(byte)] pCode

Бинарное значение кода длиной 8 байт.

Описание

Все коды в картах ACOS имеют фиксированную длину 8 байт. Вероятно, вам потребуется запрашивать у пользователя некоторые из них. Пользователь вводит строку от 0 до 8 символов, далее вы вызываете метод `MakeBinaryCode`, чтобы получить бинарный код, совместимый с картой. Согласно наиболее распространенному стандарту, строка дополняется байтами FF до необходимой длины. Например, код 123 на выходе даст 8-байтный блок 313233FFFFFFFFFF.

Строка CodeStr может содержать любые символы. Однако, не рекомендуется использовать не-ASCII символы. Использование русских букв допускается, но с осторожностью. Перевод строки в байты осуществляется по кодовой странице не-UNICODE программ (см. Control Panel -> Regional Options). Если на входе присутствуют символы, для которых не задано соответствие ANSI<>UNICODE в используемой кодовой странице, то на выходе вы можете получить блок данных, состоящий из символов-заменителей (например, "????????"). Понятно, что надежность кода, состоящего из вопросительных знаков, невысока, хотя на вход поступала вполне надежная строка. Ваша программа - UNICODE, поэтому с виду все в порядке, но на деле имеется серьезная проблема.

Возможно, будет иметь смысл проверять задаваемый пользователем код на наличие символов с кодом вне диапазона 0x20..0x7F и не допускать установки таких кодов. Тогда у вас гарантированно не возникнет проблем с функцией

`MakeBinaryCode`. Учтите и тот факт, что в русской версии Windows русский язык является языком по умолчанию. После входа в систему пользователь набирает по-русски, даже если курсор переведен на текстовый элемент со стилем ES_PASSWORD.

Вместо `MakeBinaryCode` вы можете использовать свою схему конверсии строки в 8-байтный бинарный блок данных.

MutualAuthenticate1

Взаимная аутентификация по алгоритму DES.

```
Число MutualAuthenticate1/ВзаимнаяАутентификацияDES(
    [in] Строка Kc,
    [in] Строка Kt,
    [out,retval] Строка Ks);
```

Параметры

`[in] Kc`

Ключ терминала длиной 8 байт.

`[in] Kc`

Ключ карты длиной 8 байт.

Результат

Ks - Ключ сессии длиной 8 байт.

Описание

Для успешного выполнения операции необходимо, чтобы в регистре настроек стадии персонализации была сброшена опция `ACOS_PERSOPT_3DES`.

После успешного прохождения процедуры взаимной аутентификации ключ сессии автоматически сохраняется внутри класса. Последующие вызовы методов при необходимости используют сохраненное значение.

MutualAuthenticate3

Взаимная аутентификация по алгоритму 3DES.

```
Число MutualAuthenticate3/ВзаимнаяАутентификация3DES(
    [in] Строка Kc,
    [in] Строка Kt,
    [out,retval] Строка Ks);
```

Параметры

`[in] Kc`

Ключ терминала длиной 16 байт.

`[in] Kc`

Ключ карты длиной 16 байт.

Результат

Ks - Ключ сессии длиной 16 байт.

Описание

Для успешного выполнения операции необходимо, чтобы в регистре настроек стадии персонализации была установлена опция `ACOS_PERSOPT_3DES`.

После успешного прохождения процедуры взаимной аутентификации ключ сессии автоматически сохраняется внутри класса. Последующие вызовы методов при необходимости используют сохраненное значение.

StartSession

Начать сессию взаимной аутентификации.

```
Число StartSession/НачатьСессиюАутентификации();
```

Описание

После выполнения функции Response содержит случайное число карты + `StatusWord`.

Вам, скорее всего, не потребуется вызывать эту функцию, т.к. методы MutualAuthenticate1 и MutualAuthenticate3 берут на себя все действия по выполнению взаимной аутентификации.

GetRndT

Получить случайное число терминала.

```
Число    GetRndT/СлучайноеЧислоТерминала([out,retval]    Строка
Random);
```

Результат

Сгенерированный 8-байтный блок случайных данных.

Описание

Вам, скорее всего, не потребуется вызывать эту функцию, т.к. методы MutualAuthenticate1 и MutualAuthenticate3 берут на себя все действия по выполнению взаимной аутентификации.

EncryptWithSessionKey

Зашифровать по ключу сессии.

```
Число EncryptWithSessionKey/Зашифровать(
    [in] Строка Block,
    [out,retval] Строка Encrypted);
```

Параметры

`[in] Block`

8-байтовый шифруемый блок данных.

Результат

`[out] Encrypted`

8-байтовый зашифрованный блок данных.

Описание

Если взаимная аутентификация выполнялась функцией MutualAuthenticate1, то шифрование производится по алгоритму DES; если функцией MutualAuthenticate3 - по алгоритму 3DES. Если взаимная аутентификация не выполнялась, то результат непредсказуем.

Вам, скорее всего, не потребуется вызывать эту функцию, т.к. все действия, в которых может потребоваться данная операция, реализуются высокоуровневыми методами класса.

DecryptWithSessionKey

Расшифровать по ключу сессии.

```
Число DecryptWithSessionKey/Расшифровать(
    [in] Строка Block,
    [out,retval] Строка Decrypted);
```

Параметры

`[in] Block`

8-байтовый расшифровываемый блок данных.

Результат

`[out] Decrypted`

8-байтовый расшифрованный блок данных.

Описание

Если взаимная аутентификация выполнялась функцией MutualAuthenticate1, то шифрование производится по алгоритму DES; если функцией MutualAuthenticate3 - по алгоритму 3DES. Если взаимная аутентификация не выполнялась, то результат непредсказуем.

Вам, скорее всего, не потребуется вызывать эту функцию, т.к. все действия, в которых может потребоваться данная операция, реализуются высокоуровневыми методами класса.

Операции с файлами

SelectFile

Выбрать файл.

```
Число SelectFile/ВыбратьФайл(
    [in] Число FID,
    [out,retval] Число LastSelectedFileDefinitionBlockIdx);
```

Параметры

`[in] FID`

Идентификатор файла.

Результат

Номер файлового дескриптора, соответствующего выбранному файлу. Если выбран системный файл, то возвращается значение 0xFF.

Описание

Выбор файла необходим перед использованием функций ReadRecord и WriteRecord.

Следующие методы гарантированно не меняют текущий выбранный файл : Verify, ChangePIN, MakeBinaryCode, ReadRecord, WriteRecord, StartSession, GetResponse. Остальные методы могут внутри выполнять SelectFile.

ReadRecord

Прочитать запись.

```
Число ReadRecord/ПрочитатьЗапись(
    [in] Число RecordNumber,
    [in] Число RecordLength);
```

Параметры

`[in] RecordNumber`

Номер записи в диапазоне от 0 до количества записей в файле минус 1.

`[in] RecordLength`

Длина читаемого блока в диапазоне от 1 до длины записи файла.

Описание

Перед выполнением ReadRecord необходимо выбрать файл.

После успешного выполнения операции Response содержит прочитанный блок данных + StatusWord.

WriteRecord

Сохранить запись.

```
Число WriteRecord/СохранитьЗапись(
    [in] Число RecordNumber,
    [in] Строка Data);
```

Параметры

`[in] RecordNumber`

Номер записи в диапазоне от 0 до количества записей в файле минус 1.

`[in] Data`

Сохраняемые данные. Длина сохраняемого блока должна быть в диапазоне от 1 до длины записи файла.

Описание

Перед выполнением WriteRecord необходимо выбрать файл.

ReadBinaryFile

Чтение двоичного файла.

```
Число ReadBinaryFile/ЧтениеДвоичногоФайла(
    [in] Число FID,
    [in] Число Offset,
    [in] Число Length);
```

Параметры

`[in] FID`

Идентификатор файла.

`[in] Offset`

Смещение в байтах относительно начала файла.

`[in] Length`

Длина читаемого блока данных.

Описание

После успешного выполнения операции Response содержит прочитанный блок данных.

WriteBinaryFile

Запись в двоичный файл.

```
Число WriteBinaryFile/ЗаписьДвоичногоФайла(
    [in] Число FID,
    [in] Число Offset,
    [in] Строка Data);
```

Параметры

`[in] FID`

Идентификатор файла.

`[in] Offset`

Смещение в байтах относительно начала файла.

`[in] Data`

Записываемые данные.

Описание

Поскольку в картах ACOS1,ACOS2,ACOS3-8K,ACOS3-16K двоичные файлы являются высокоуровневой абстракцией и базируются на файлах записей, и карты не позволяют изменять записи не с их начала, то может потребоваться промежуточное чтение файла. Это в свою очередь потребует удовлетворения аттрибутов доступа на чтение. Чтобы избежать промежуточного чтения, выполняйте запись только по смещениям, кратным длине записи файла. Длины записей двоичных файлов указаны в описании метода WriteBinaryFileDefinitionBlock. В картах ACOS3-24K и выше применяется их аппаратная возможность по ведению двоичных файлов, поэтому чтение не потребуется никогда.

Операции с кошельком

Прежде чем проектировать схему аутентификации действий с кошельком, тщательно изучите архитектуру карты. Продумайте все возможные угрозы. Допустите, что в терминал могут вместо подлинной карты вставить сколь угодно сложное техническое устройство. Единственное, что устройство не может знать, это ключи доступа к карте. В противном случае систему можно считать скомпрометированной.

Мы рекомендуем всегда использовать настройки персонализации `ACOS_PERSOPT_3DES`, `ACOS_PERSOPT_TRNS_AUT`, `ACOS_PERSOPT_INQ_AUT`. Тем самым вы надежно устанавливаете подлинность карты и защищаетесь от атак, связанных с повторным подсовыванием перехваченных ранее данных (такого рода атаки называются ***replay***).

Самые изощренные атаки реализуются с помощью схемотехнической интеграции подлинной карты и устройства злоумышленника. Такой гибрид способен задействовать настоящую карту, читать все сообщения между картой и терминалом и, когда это необходимо, перехватывать процесс и вставлять свои посылки данных. Например, если используется взаимная аутентификация, но терминал не выполняет достаточных проверок, этим способом можно проэмулировать успешное списание средств с кошелька, когда в действительности оно не произошло.

InquireAccount

Опрос кошелька.

```
Число InquireAccount/ОпросКошелька(
    [in] Число AccountCodeIdx,
    [in] Строка AccountCode,
    [out] Число Balance,
    [out] Число MaximumBalance,
    [out] Число ATC,
    [out] Число AID,
    [out] Число TTREF_C,
    [out] Число TTREF_D,
    [out] Число LastTransType);
```

Параметры

`[in] AccountCodeIdx`

Номер ключа кошелька, по которому будет будет вычисляться криптографическая подпись получаемого от карты результата. Возможные значения : `ACOS_ASFN_DEBIT`, `ACOS_ASFN_CREDIT`, `ACOS_ASFN_INQUIRE`, `ACOS_ASFN_REVOKE`.

`[in] AccountCode`

Ключ проверки криптографической подписи (8 байт для DES, 16 байт 3DES) . Если проверка подписи не требуется – вместо массива укажите 0.

`[out] Balance`

Баланс кошелька.

`[out] MaximumBalance`

Максимальный баланс кошелька.

`[out] ATC`

Порядковый номер предыдущей транзакции.

`[out] AID`

Идентификатор приложения.

`[out] TTREF_C`

Идентификатор терминала дебетования.

`[out] TTREF_D`

Идентификатор терминала кредитования.

`[out] LastTransType`

Тип предыдущей транзакции. Возможные значения : `ACOS_TRANSTYPE_DEBIT`, `ACOS_TRANSTYPE_REVOKE_DEBIT`, `ACOS_TRANSTYPE_CREDIT`. Любое другое значение означает, что не было проведено ни одной транзакции.

Описание

Функция автоматически читает настройки стадии производства и персонализации и проводит операцию согласно прочитанным настройкам.

Длина ключа должна соответствовать опции персонализации `ACOS_PERSOPT_3DES` и быть 8 байт для DES, 16 байт для 3DES

Проверка криптографической подписи является необязательной. Она позволяет убедиться, что полученные данные действительно были сгенерированы подлинной картой, но не позволяет ответить на вопрос "когда". Чтобы исключить атаки класса ***replay***, используйте взаимную аутентификацию и опцию персонализации `ACOS_PERSOPT_INQ_AUT`.

Credit

Зачисление суммы на счет.

```
Число Credit/ЗачислитьНаСчет(
    [in] Строка CreditCode,
    [in] Число Amount);
```

Параметры

`[in] CreditCode`

Ключ кредитования (8 байт для DES, 16 байт 3DES).

`[in] Amount`

Зачисляемая сумма.

Описание

Функция автоматически читает настройки стадии персонализации и проводит операцию согласно прочитанным настройкам.

Длина ключа должна соответствовать опции персонализации `ACOS_PERSOPT_3DES` и быть 8 байт для DES, 16 байт для 3DES.

Debit

Списание суммы со счета.

```
Число Debit/СнятьСоСчета(
    [in] Строка DebitCode,
    [in] Число Amount,
    [in] Число RequireDebitCertificate);
```

Параметры

`[in] DebitCode`

Ключ дебетования (8 байт для DES, 16 байт 3DES). Может быть 0, если не задействована опция персонализации `ACOS_PERSOPT_DEB_MAC`.

`[in] Amount`

Списываемая сумма.

`[in]` RequireDebitCertificate

`VARIANT_TRUE` = требовать сертификат дебетования,
`VARIANT_FALSE` = не требовать.

Описание

Функция автоматически читает настройки стадии персонализации и проводит операцию согласно прочитанным настройкам.

Длина ключа должна соответствовать опции персонализации `ACOS_PERSOPT_3DES` и быть 8 байт для DES, 16 байт для 3DES.

Сертификат дебетования доступен только в картах ACOS3. Если `RequireDebitCertificate=VARIANT_TRUE`, и использована карта ACOS1/2, то произойдет ошибка. Если `RequireDebitCertificate=VARIANT_FALSE`, то проверка сертификата дебетования проводится только в том случае, если использована карта ACOS3. На картах ACOS1/2 сертификат не проверяется, и это не считается ошибкой.

Проверка сертификата дебетования позволяет защититься от устройств,

представляющих собой гибрид подлинной карты и устройства злоумышленника. На картах ACOS1/2 так же возможно защититься от этого класса устройств : необходимо перед дебетованием и после дебетования выполнить InquireAccount с ключами и убедиться, что баланс счета уменьшился должным образом.

RevokeDebit

Отмена списания суммы со счета.

```
Число RevokeDebit/ОтменитьСнятиеСоСчета(
    [in] Строка RevokeDebitCode,
    [in] Число PreviousBalance);
```

Параметры

`[in] RevokeDebitCode`

Ключ отмены дебетования (8 байт для DES, 16 байт 3DES).

`[in] PreviousBalance`

Сумма, которая была до последней операции дебетования.

Описание

Функция автоматически читает настройки стадии персонализации и проводит операцию согласно прочитанным настройкам.

Длина ключа должна соответствовать опции персонализации `ACOS_PERSOPT_3DES` и быть 8 байт для DES, 16 байт для 3DES.

Если последней операцией не было дебетование или сумма dwAmount не соответствует предыдущей сумме, то произойдет ошибка.

Secure messaging

Secure messaging - механизм карт ACOS3-24K и выше, позволяющий зашифровать трафик между терминалом и картой (в оба направления). Ключом шифрования является ключ сессии, поэтому перед задействованием механизма необходимо выполнить команду MutualAuthenticate1 или MutualAuthenticate3, в зависимости от опций персонализации,

Secure messaging - сложный механизм, требующий учета множества факторов и параметров. Для разработчиков, использующих настоящий SDK, он становится полностью прозрачным. Это значит, что вам лишь достаточно его включить, а остальные операции будут автоматически шифроваться.

Следует так же отметить, что secure messaging замедляет работу с картой примерно в 3 раза. Включайте шифрование только для тех операций, где перехват или подмена передаваемых данных представляет реальную угрозу безопасности.

SMEnable

Включить/отключить secure messaging.

```
Число SMEnable/ВклВыклSM([in] Число Enable);
```

Параметры

`[in] Enable`

`VARIANT_TRUE` - включить,
`VARIANT_FALSE` - выключить

Описание

`Secure messaging` автоматически выключается после каждой операции подключения к карте.

Включение `secure messaging` означает его обязательное требование. Если карта не поддерживает `secure messaging`, последующие команды вызовут ошибку. Если вам нужно продолжать работу без `secure messaging` в случае его неподдержки, то проверяйте версию карты перед включением.

SMIsEnabled

Проверить включен ли `secure messaging`.

```
Число SMIsEnabled/SMBвключен([out,retval] Число Enabled);
```

Результат

`VARIANT_TRUE` – включен,
`VARIANT_FALSE` – выключен.

TransmitSM

Передать карте APDU команду в режиме `secure messaging`, если он включен, и без, если он выключен.

```
Число TransmitSM/ПередатьCSM([in] Строка SendBuffer);
```

Параметры

`[in] SendBuffer`

Оригинальная команда без SM.

Описание

Функция может быть полезна для передачи произвольных команд карте в режиме `secure messaging`. Если команда является командой ISO-IN-OUT, то функция автоматически выполнит действия по получению ответа. Все будет выглядеть так, как если бы без secure messaging после подачи основной команды и в случае получения успешного кода завершения выполнили бы функцию GetResponse. Response и StatusWord заполняются из SM-ответа карты. Анализ StatusWord не производится, поэтому функция не вызывает исключения по причине ошибки выполнения самой команды, но вызывает исключения в случае ошибки самого механизма secure messaging.

TransmitSMWithCommonSWCheck

Передать карте APDU команду в режиме secure messaging, если он включен, и без, если он выключен, с проверкой статуса команды.

```
Число TransmitSMWithCommonSWCheck/ПередатьSMПроверитьСтатус(
[in] Строка SendBuffer);
```

Параметры

`[in] SendBuffer`

Оригинальная команда без SM.

Описание

Функция аналогична TransmitSM, только дополнительно выполняется общая проверка StatusWord и вызывается исключение при получении статуса, соответствующего ошибке.

Другие операции

GetResponse

Получить ответ.

```
Число GetResponse/ПолучитьОтвет();
```

Описание

Протокол T=0 не позволяет в одной команде отсылать блок данных и принимать блок данных в ответ. Эта проблема решается таким образом : карта запоминает блок данных, который нужно вернуть терминалу, и возвращает его по следующей команде GetResponse. `GetResponse` работает только в том случае, если `StatusWord` последней команды был 61xx или 6Cxx. xx означает число байтов данных, которые вернет `GetResponse`.

В результате выполнения команды Response содержит блок данных + `StatusWord`.

Скорее всего, эта операция вам не понадобится, поскольку все действия, в которых она вовлечена, реализуются высокоуровневыми методами класса `PCSCCard_ACOS`.

ATRCheckEnforced

Проверять ли соответствие ATR карте ACOS.

```
Число ATRCheckEnforced/ATRКартыACOS
```

Описание

Карты ACOS обладают возможностью менять ATR. Если вы меняете ATR, то вам необходимо запретить проверку ATR при подключении к карте. В этом случае присвойте `ATRCheckEnforced = FALSE`.

3.2.9 PCSCCard_Mifare

Класс для работы с бесконтактными картами Mifare.

Наследование

PCSCCard	Класс содержит все методы и свойства класса PCSCCard

Методы

TBlockFromBlock/ НомерТБлока	Вычислить номер Т-блока для указанного блока
FirstBlockOfSector/ ПервыйСекторБлока	Вычислить номер первого блока сектора
SectorFromBlock/ СекторПоБлоку	Вычислить номер сектора по номеру блока
GetSectorCount/ КоличествоСекторов	Получить количество секторов
GetDataBlockCount/ КоличествоБлоковДанных	Получить количество блоков данных
GetKeySlotCount/ КоличествоСлотовКлючей	Получить количество слотов ключей
LoadKey/ ЗагрузитьКлюч	Загрузить ключ
MakeBinaryCode/ СоздатьДвоичныйКод	Получить 6-байтный код из строки.
Authenticate/ Аутентификация	Аутентификация
ReadBinary/ СчитатьДБлок	Чтение D-блока
UpdateBinary/ ЗаписатьДБлок	Запись D-блока
CheckBlockMap/	Проверить цепочку блоков

ПроверитьЦепочкуБлоков	
ReadBinaryLong/ ЧтениеФайла	Чтение виртуального файла
WriteBinaryLong/ ЗаписьФайла	Запись виртуального файла
ReadTBlock/ СчитатьТБлок	Чтение T-блока
WriteTBlock/ ЗаписатьТБлок	Запись T-блока
ReadVBlock/ СчитатьВБлок	Чтение V-блока (получение текущей суммы кошелька)
WriteVBlock/ ЗаписатьВБлок	Запись V-блока (инициализация кошелька)
Increment/ Инкремент	Инкремент V-блока (кредитование кошелька)
Decrement/ Декремент	Декремент V-блока (дебетование кошелька)
IsContactlessStorageCardATR/ ATRКонтактнойКарты	Принадлежит ли ATR бесконтактной карте ?
CardTypeFromATR/ ПолучитьТипПоATR	Получить тип бесконтактной карты по ATR
GetCardType/ ПолучитьТипКарты	Получить тип текущей подключенной карты
OverrideCardType/ ЗадатьТипКарты	Задать тип карты вручную или вернуться к автоматическому определению.
GetUID/ ПолучитьСерийныйНомер	Получить серийный номер карты

Описание

Прежде чем начать работу с картами Mifare, ознакомьтесь с документацией

по этим картам.

TBlockFromBlock

Вычислить номер Т-блока для указанного блока.

```
Число TBlockFromBlock/НомерТБлока(
    [in] Число Block,
    [out,retval] Число TBlock);
```

Параметры

`[in] Block`

Номер блока.

Результат

Номер Т-блока, сответствующего Block.

Описание

Размер блока в картах Mifare составляет 16 байт. Нумерация блоков - сквозная. В картах Mifare 1K имеется 16 секторов по 4 блока каждый. В картах Mifare 4K первые 32 сектора содержат 4 блока, остальные 8 секторов содержат по 16 блоков. Последний блок каждого сектора является Т-блоком.

Метод не зависит от подключения к карте.

FirstBlockOfSector

Вычислить номер первого блока сектора.

```
Число FirstBlockOfSector/ПервыйСекторБлока(
    [in] Число Sector,
    [out,retval] Число Block);
```

Параметры

`[in] Sector`

Номер сектора.

Результат

Номер первого блока сектора Sector.

Описание

Размер блока в картах Mifare составляет 16 байт. Нумерация блоков - сквозная. В картах Mifare 1K имеется 16 секторов по 4 блока каждый. В картах Mifare 4K первые 32 сектора содержат 4 блока, остальные 8 секторов содержат по 16 блоков. Последний блок каждого сектора является Т-блоком.

Метод не зависит от подключения к карте.

SectorFromBlock

Вычислить номер сектора по номеру блока.

```
Число SectorFromBlock/СекторПоБлоку(
    [in] Число Block,
    [out,retval] Число Sector);
```

Параметры

`[in] Block`

Номер блока.

Результат

Номер сектора, которому принадлежит блок `Block`.

Описание

Размер блока в картах Mifare составляет 16 байт. Нумерация блоков - сквозная. В картах Mifare 1K имеется 16 секторов по 4 блока каждый. В картах Mifare 4K первые 32 сектора содержат 4 блока, остальные 8 секторов содержат по 16 блоков. Последний блок каждого сектора является Т-блоком.

Метод не зависит от подключения к карте.

GetSectorCount

Получить количество секторов текущей подключенной карты.

```
Число   GetSectorCount/КоличествоСекторов([out,retval]   Число
Count);
```

Результат

Количество секторов.

Описание

Число секторов для карты Mifare 1K – 16, для карты Mifare 4K – 40.

GetDataBlockCount

Получить количество блоков данных текущей подключенной карты.

```
Число    GetDataBlockCount/КоличествоБлоковДанных([out,retval]
Число Count);
```

Результат

Количество блоков данных.

Описание

Блоками данных считаются все блоки, кроме Т-блоков и нулевого блока нулевого сектора. Число блоков данных для карты Mifare 1K – 47, для карты Mifare 4K – 215.

GetKeySlotCount

Получить количество слотов для хранения ключей доступа.

```
Число GetKeySlotCount/КоличествоСлотовКлючей(
    [in] Число NonvolatileMemory,
    [out,retval] Число Count);
```

Параметры

`[in] NonvolatileMemory`

`VARIANT_TRUE` = энергонезависимая память ридера,
`VARIANT_FALSE` = временное хранение в оперативной памяти.

Результат

Количество слотов.

Описание

В целях достижения максимальной совместимости между ридерами в SDK была построена следующая модель хранения ключей. Есть 2 хранилища : постоянное в энергонезависимой памяти ридера и временное в оперативной памяти компьютера. Временное хранилище привязано к экземпляру класса и гарантированно существует до момента его уничтожения. При использовании временного хранилища обязательно использовать либо эксклюзивное подключение к карте, либо открывать транзакцию.

Оба хранилища состоят из слотов. Каждый слот содержит 2 ячейки : для ключа A и ключа B. Однако, прежде чем использовать произвольные номера слотов, следует определить их количество. Гарантированно в любом ридере имеется только нулевой слот энергонезависимой памяти. Его и рекомендуется использовать, если необходимость не диктует иное. В последнем случае вы теряете совместимость между моделями ридеров.

LoadKey

Загрузить ключ.

```
Число LoadKey/ЗагрузитьКлюч(
    [in] Число NonvolatileMemory,
    [in] Число KeyType,
    [in] Число KeyIdx,
    [in] Строка Key);
```

Параметры

`[in] NonvolatileMemory`

`VARIANT_TRUE` = энергонезависимая память ридера,
`VARIANT_FALSE` = временное хранение в оперативной памяти.

`[in] KeyType`

Тип ключа : `CL_KEYTYPE_MIFARE_A` или `CL_KEYTYPE_MIFARE_B`.

`[in] KeyIdx`

Номер ключевого слота.

`[in] Key`

6-байтный ключ.

Описание

В целях достижения максимальной совместимости между ридерами в SDK была построена следующая модель хранения ключей. Есть 2 хранилища : постоянное в энергонезависимой памяти ридера и временное в оперативной памяти компьютера. Временное хранилище привязано к экземпляру класса и гарантированно существует до момента его уничтожения. При использовании временного хранилища обязательно использовать либо эксклюзивное

подключение к карте, либо открывать транзакцию.

Оба хранилища состоят из слотов. Каждый слот содержит 2 ячейки : для ключа A и ключа B. Однако, прежде чем использовать произвольные номера слотов, следует определить их количество. Гарантированно в любом ридере имеется только нулевой слот энергонезависимой памяти. Его и рекомендуется использовать, если необходимость не диктует иное. Иначе вы теряете совместимость между моделями ридеров.

MakeBinaryCode

Получить 6-байтный код из строки.

```
Число MakeBinaryCode/СоздатьДвоичныйКод(
    [in] Строка CodeStr,
    [out,retval] Строка CodeBin);
```

Параметры

`[in] CodeStr`

Строковое значение кода.

Результат

Бинарное значение кода.

Описание

Ключи A и B в картах Mifare имеют длину 6 байт. Возможно, вам потребуется запрашивать их у пользователя. Пользователь вводит строку от 0 до 6 символов, далее вы вызываете метод `MakeBinaryCode`, чтобы получить бинарный код, совместимый с картой. Согласно наиболее распространенному стандарту, строка дополняется байтами FF до необходимой длины. Например, код 123 на выходе даст 6-байтный блок 313233FFFFFF. Пустая строка будет соответствовать ключу по умолчанию в транспортной конфигурации карты - FFFFFFFFFFFF.

Строка CodeStr может содержать любые символы. Однако, не рекомендуется использовать не-ASCII символы. Использование русских букв допускается, но с осторожностью. Перевод строки в байты осуществляется по кодовой странице не-UNICODE программ (см. Control Panel -> Regional Options). Если на входе присутствуют символы, для которых не задано соответствие ANSI<>UNICODE в используемой кодовой странице, то на выходе вы можете получить блок данных, состоящий из символов-заменителей (например, "????????"). Понятно, что надежность кода, состоящего из вопросительных знаков, невысока, хотя на вход поступала вполне надежная строка. Ваша программа – UNICODE, поэтому с виду все в порядке, но на деле имеется серьезная проблема.

Возможно, будет иметь смысл проверять задаваемый пользователем код на наличие символов с кодом вне диапазона 0x20..0x7F и не допускать установки

таких кодов. Тогда у вас гарантированно не возникнет проблем с функцией `MakeBinaryCode`. Учтите и тот факт, что в русской версии Windows русский язык является языком по умолчанию. После входа в систему пользователь набирает по-русски, даже если курсор переведен на текстовый элемент со стилем ES_PASSWORD.

Вместо `MakeBinaryCode` вы можете использовать свою схему конверсии строки в 6-байтный бинарный блок данных.

Authenticate

Аутентификация.

```
Число Authenticate/Аутентификация(
    [in] Число Sector,
    [in] Число NonvolatileMemory,
    [in] Число KeyType,
    [in] ЧислоKeyIdx);
```

Параметры

`[in] Sector`

Номер сектора.

`[in] NonvolatileMemory`

`VARIANT_TRUE` = энергонезависимая память ридера,
`VARIANT_FALSE` = временное хранение в оперативной памяти.

[in] KeyType

Тип ключа : `CL_KEYTYPE_MIFARE_A` или `CL_KEYTYPE_MIFARE_B`.

`[in] KeyIdx`

Номер ключевого слота.

Описание

В целях достижения максимальной совместимости между ридерами в SDK была построена следующая модель хранения ключей. Есть 2 хранилища : постоянное в энергонезависимой памяти ридера и временное в оперативной памяти компьютера. Временное хранилище привязано к экземпляру класса и гарантированно существует до момента его уничтожения. При использовании временного хранилища обязательно использовать либо эксклюзивное

подключение к карте, либо открывать транзакцию.

Оба хранилища состоят из слотов. Каждый слот содержит 2 ячейки : для ключа A и ключа B. Однако, прежде чем использовать произвольные номера слотов, следует определить их количество. Гарантированно в любом ридере имеется только нулевой слот энергонезависимой памяти. Его и рекомендуется использовать, если необходимость не диктует иное. Иначе вы теряете совместимость между моделями ридеров.

ReadBinary

Чтение D-блока.

```
Число ReadBinary/СчитатьДБлок([in] Число Block);
```

Параметры

`[in] Block`

Номер блока.

Описание

После успешного выполнения операции Response содержит 16 прочитанных байт + `StatusWord`.

UpdateBinary

Запись D-блока.

```
Число UpdateBinary/ЗаписатьДБлок(
    [in] Число Block,
    [in] Строка Data);
```

Параметры

`[in] Block`

Номер блока.

`[in] Data`

Записываемые данные длиной 16 байт.

Описание

Запись блока возможна только целиком. В целях предотвращения необратимого повреждения секторов, если `Block` соответствует номеру T-блока, то возвращается ошибка. Для записи T-блоков следует использовать функцию WriteTBlock.

CheckBlockMap

Проверить цепочку блоков.

```
Число     CheckBlockMap/ПроверитьЦепочкуБлоков([in]     Строка
BlockMap);
```

Параметры

`[in] BlockMap`

Массив номеров блоков.

Описание

Для проверки используются параметры текущей подключенной карты. Правильными считаются ненулевые номера блоков, не выходящие за количество секторов карты и не являющиеся Т-блоками. Если массив блоков содержит неправильные или повторяющиеся номера, возвращается ошибка.

ReadBinaryLong

Чтение виртуального файла.

```
Число ReadBinaryLong/ЧтениеФайла(
    [in] Строка BlockMap,
    [in] Число Offset,
    [in] Число Length);
```

Параметры

`[in] BlockMap`

Массив номеров блоков. 0 означает использовать последовательно все D-блоки карты.

`[in] Offset`

Смещение внутри файла.

`[in] Length`

Длина читаемых данных.

Описание

Память карт Mifare разделена на блоки. Однако, часто возникает необходимость хранить структуры данных, не влезающие в один блок. Для решения этой проблемы SDK предлагает удобную возможность. Вы можете составить цепочку блоков, которая будет восприниматься, как непрерывный файл. К этому файлу возможно обращение по любому смещению и с любой длиной данных. Вам не нужно думать о пересечении границ блоков – метод все сделает за вас.

Размер виртуального файла вычисляется как количество задействованных блоков умноженное на 16. Если используется вся память карты, то количество блоков можно получить из метода GetDataBlockCount.

Обязательное условие состоит в том, чтобы настройки прав доступа и ключи были идентичными для всех блоков. Выполните Authenticate к сектору любого блока, включенного в цепочку. Класс запомнит последние параметры и будет их

использовать автоматически по мере необходимости.

После успешного выполнения операции Response содержит прочитанную область данных.

WriteBinaryLong

Запись виртуального файла.

```
Число WriteBinaryLong/ЗаписьФайла(
    [in] Строка BlockMap,
    [in] Число Offset,
    [in] Строка Data);
```

Параметры

`[in] BlockMap`

Массив номеров блоков. 0 означает использовать последовательно все D-блоки карты.

`[in] Offset`

Смещение внутри файла.

`[in] Data`

Записываемые данные.

Описание

Память карт Mifare разделена на блоки. Однако, часто возникает необходимость хранить структуры данных, не влезающие в один блок. Для решения этой проблемы SDK предлагает удобную возможность. Вы можете составить цепочку блоков, которая будет восприниматься как непрерывный файл. К этому файлу возможно обращение по любому смещению и с любой длиной данных. Вам не нужно думать о пересечении границ блоков - метод все сделает за вас.

Размер виртуального файла вычисляется как количество задействованных блоков умноженное на 16. Если используется вся память карты, то количество блоков можно получить из метода GetDataBlockCount.

Обязательное условие состоит в том, чтобы настройки прав доступа и ключи были идентичными для всех блоков. Выполните Authenticate к сектору любого блока, включенного в цепочку. Класс запомнит последние параметры и будет их

использовать автоматически по мере необходимости.

ReadTBlock

Чтение Т-блока.

```
Число ReadTBlock/СчитатьТБлок(
    [in] Число Sector,
    [out] Число P0,
    [out] Число P1,
    [out] Число P2,
    [out] Число P3,
    [out] Строка KeyB);
```

Параметры

`[in] Sector`

Номер сектора.

`[out] P0`

Биты C1-C2-C3 прав доступа к блоку 0 или группе блоков 0..4 для секторов 33..40.

`[out] P1`

Биты C1-C2-C3 прав доступа к блоку 1 или группе блоков 5..9 для секторов 33..40.

`[out] P2`

Биты C1-C2-C3 прав доступа к блоку 2 или группе блоков 10..14 для секторов 33..40.

`[out] P3`

Биты C1-C2-C3 прав доступа к Т-блоку сектора. Параметр может быть NULL.

`[out] KeyB`

Значение ключа B. Ключ B читается только при определенных настройках прав

доступа на T-блок.

Описание

C1 соответствует биту 2, C2 - биту 1, C3 - биту 0. Например, значение C1-C2-C3=110 будет означать 6. Информацию по правами доступа можно найти в документации по картами Mifare.

WriteTBlock

Запись Т-блока.

```
Число WriteTBlock/ЗаписатьТБлок(
    [in] Число Sector,
    [in] Строка KeyA,
    [in] Строка KeyB,
    [in] Число P0,
    [in] Число P1,
    [in] Число P2,
    [in] Число P3);
```

Параметры

`[in] Sector`

Номер сектора.

`[in] KeyA`

Значение ключа A (6 байт).

`[in] KeyB`

Значение ключа B (6 байт).

`[in] P0`

Биты C1-C2-C3 прав доступа к блоку 0 или группе блоков 0..4 для секторов 33..40.

`[in] P1`

Биты C1-C2-C3 прав доступа к блоку 1 или группе блоков 5..9 для секторов 33..40.

`[in] P2`

Биты C1-C2-C3 прав доступа к блоку 2 или группе блоков 10..14 для секторов 33..40.

`[in] P3`

Биты C1-C2-C3 прав доступа к T-блоку сектора.

Описание

C1 соответствует биту 2, C2 – биту 1, C3 – биту 0. Например, значение C1-C2-C3=110 будет означать 6. Информацию по правами доступа можно найти в документации по картами Mifare.

ReadVBlock

Чтение V-блока.

```
Число ReadVBlock/СчитатьВБлок(
    [in] Число Block,
    [out,retval] Число Value);
```

Параметры

`[in] Block`

Номер блока.

Результат

Значение V-блока.

Описание

Блок должен быть предварительно отформатирован как V-блок. Самый простой способ это сделать - записать значение при помощи WriteVBlock.

WriteVBlock

Запись V-блока.

```
Число WriteVBlock/ЗаписатьВБлок(
    [in] Число Block,
    [in] Число Value,
    [in] Число Adr);
```

Параметры

`[in] Block`

Номер блока.

`[in] Value`

Записываемое значение.

`[in] Adr`

Произвольное значение, фигурирующее в формате V-блока.

Описание

Метод автоматически форматирует блок как V-блок.

Increment

Инкремент V-блока.

```
Число Increment/Инкремент(
    [in] Число Block,
    [in] Число Value);
```

Параметры

`[in] Block`

Номер блока.

`[in] Value`

Величина инкремента.

Описание

Увеличить значение V-блока на Value. V-блоки могут содержать и отрицательные значения.

Decrement

Декремент V-блока.

```
Число Decrement/Декремент(
    [in] Число Block,
    [in] Число Value);
```

Параметры

`[in] Block`

Номер блока.

`[in] Value`

Величина декремента.

Описание

Уменьшить значение V-блока на Value. V-блоки могут содержать и отрицательные значения.

3.2.10 PCSCCard_MifareUL

Класс для работы с бесконтактными картами Mifare Ultra Light.

Наследование

PCSCCard	Класс содержит все методы и свойства класса PCSCCard

Методы

ReadBinary/ ПрочитатьСтраницу	Чтение страницы
UpdateBinary/ ЗаписатьСтраницу	Запись страницы
ReadBinaryLong/ ЧтениеФайла	Чтение виртуального файла
WriteBinaryLong/ ЗаписьФайла	Запись виртуального файла
CheckPageMap/ ПроверитьЦепочкуСтраниц	Проверить цепочку страниц
IsContactlessStorageCardATR/ АТRКонтактнойКарты	Принадлежит ли ATR бесконтактной карте ?
CardTypeFromATR/ ПолучитьТипПоATR	Получить тип бесконтактной карты по ATR
GetCardType/ ПолучитьТипКарты	Получить тип текущей подключенной карты
OverrideCardType/ ЗадатьТипКарты	Задать тип карты вручную или вернуться к автоматическому определению.
GetUID/ ПолучитьСерийныйНомер	Получить серийный номер карты

Описание

Прежде чем начать работу с картами Mifare Ultra Light, ознакомьтесь с документацией по этим картам.

ReadBinary

Чтение страницы.

```
Число ReadBinary/ПрочитатьСтраницу([in] Число Page);
```

Параметры

`[in] Page`

Номер страницы.

Описание

После успешного выполнения операции Response содержит 4 прочитанных байта + `StatusWord`.

UpdateBinary

Запись страницы.

```
Число UpdateBinary/ЗаписатьСтраницу(
    [in] Число Page,
    [in] Строка Data);
```

Параметры

`[in] Page`

Номер страницы.

`[in] Data`

Записываемые данные длиной 4 байта.

Описание

Запись страницы возможна только целиком.

ReadBinaryLong

Чтение виртуального файла.

```
Число ReadBinaryLong/ЧтениеФайла(
    [in] Строка PageMap,
    [in] Число Offset,
    [in] Число Length);
```

Параметры

`[in] PageMap`

Массив номеров страниц. NULL означает использовать последовательно все страницы данных.

`[in] Offset`

Смещение внутри файла.

`[in] Length`

Длина читаемых данных.

Описание

Память карт Mifare Ultra Light разделена на страницы. Однако, часто возникает необходимость хранить структуры данных, не влезающие в одну страницу. Для решения этой проблемы SDK предлагает удобную возможность. Вы можете составить цепочку страниц, которая будет восприниматься как непрерывный файл. К этому файлу возможно обращение по любому смещению и с любой длиной данных. Вам не нужно думать о пересечении границ блоков – метод все сделает за вас.

Размер виртуального файла вычисляется как количество задействованных страниц умноженное на 4. Вы можете использовать номера страниц в пределах 4..15. Если используется вся память карты, то количество страниц равно 12.

После успешного выполнения операции Response содержит прочитанную область данных.

WriteBinaryLong

Запись виртуального файла.

```
Число WriteBinaryLong/ЗаписьФайла(
    [in] Строка PageMap,
    [in] Число Offset,
    [in] Строка Data);
```

Параметры

`[in] PageMap`

Массив номеров страниц. NULL означает использовать последовательно все страницы данных.

`[in] Offset`

Смещение внутри файла.

`[in] Data`

Записываемые данные.

Описание

Память карт Mifare Ultra Light разделена на страницы. Однако, часто возникает необходимость хранить структуры данных, не влезающие в одну страницу. Для решения этой проблемы SDK предлагает удобную возможность. Вы можете составить цепочку страниц, которая будет восприниматься как непрерывный файл. К этому файлу возможно обращение по любому смещению и с любой длиной данных. Вам не нужно думать о пересечении границ блоков - метод все сделает за вас.

Размер виртуального файла вычисляется как количество задействованных страниц умноженное на 4. Вы можете использовать номера страниц в пределах 4..15. Если используется вся память карты, то количество страниц равно 12.

CheckPageMap

Проверить цепочку страниц.

```
Число CheckPageMap/ПроверитьЦепочкуСтраниц([in] Строка
PageMap);
```

Параметры

`[in] PageMap`

Массив номеров страниц.

Описание

Правильными считаются страницы с номерами из диапазона 4..15. Если массив страниц содержит неправильные или повторяющиеся номера, возвращается ошибка.

IsContactlessStorageCardATR

Принадлежит ли ATR бесконтактной карте ?

```
Число IsContactlessStorageCardATR/ATRКонтактнойКарты(
    [in] Строка ATR,
    [out] Число Truth);
```

Параметры

`[in] ATR`

ATR карты.

Результат

`VARIANT_TRUE` = принадлежит,
`VARIANT_FALSE` = не принадлежит.

Описание

Структура ATR бесконтактных карт описана в стандарте PC/SC 2.0.
Метод может вызываться без подключения к карте.

CardTypeFromATR

Получить тип бесконтактной карты по ATR.

```
Число CardTypeFromATR/ПолучитьТипПоATR(
    [in] Строка ATR,
    [out,retval] Число Type);
```

Параметры

`[in] ATR`

ATR карты.

Результат

Одно из следующих значений :

```
CL_CARDTYPE_UNKNOWN
CL_CARDTYPE_MIFARE_1K
CL_CARDTYPE_MIFARE_4K
CL_CARDTYPE_MIFARE_ULTRA_LIGHT
CL_CARDTYPE_SLE55R_XXXXX
CL_CARDTYPE_SR176
CL_CARDTYPE_SRIX4K
CL_CARDTYPE_AT88RF020
CL_CARDTYPE_AT88SC0204CRF
CL_CARDTYPE_AT88SC0808CRF
CL_CARDTYPE_AT88SC1616CRF
CL_CARDTYPE_AT88SC3216CRF
CL_CARDTYPE_AT88SC6416CRF
CL_CARDTYPE_SRF55V10P
CL_CARDTYPE_SRF55V02P
CL_CARDTYPE_SRF55V10S
CL_CARDTYPE_SRF55V02S
CL_CARDTYPE_TAGIT
CL_CARDTYPE_LRI512
```

```
CL_CARDTYPE_ICODESLI
CL_CARDTYPE_TEMPSENS
CL_CARDTYPE_ICODE1
CL_CARDTYPE_PICOPASS_2K
CL_CARDTYPE_PICOPASS_2KS
CL_CARDTYPE_PICOPASS_16K
CL_CARDTYPE_PICOPASS_16KS
```

Описание

Структура ATR бесконтактных карт описана в стандарте PC/SC 2.0. Метод может вызываться без подключения к карте.

GetCardType

Получить тип текущей подключенной карты.

```
Число GetCardType/ПолучитьТипКарты([out,retval] Число Type);
```

Результат

Одно из следующих значений :

```
CL_CARDTYPE_UNKNOWN
CL_CARDTYPE_MIFARE_1K
CL_CARDTYPE_MIFARE_4K
CL_CARDTYPE_MIFARE_ULTRA_LIGHT
CL_CARDTYPE_SLE55R_XXXXX
CL_CARDTYPE_SR176
CL_CARDTYPE_SRIX4K
CL_CARDTYPE_AT88RF020
CL_CARDTYPE_AT88SC0204CRF
CL_CARDTYPE_AT88SC0808CRF
CL_CARDTYPE_AT88SC1616CRF
CL_CARDTYPE_AT88SC3216CRF
CL_CARDTYPE_AT88SC6416CRF
CL_CARDTYPE_SRF55V10P
CL_CARDTYPE_SRF55V02P
CL_CARDTYPE_SRF55V10S
CL_CARDTYPE_SRF55V02S
CL_CARDTYPE_TAGIT
CL_CARDTYPE_LRI512
CL_CARDTYPE_ICODESLI
CL_CARDTYPE_TEMPSENS
CL_CARDTYPE_ICODE1
CL_CARDTYPE_PICOPASS_2K
CL_CARDTYPE_PICOPASS_2KS
CL_CARDTYPE_PICOPASS_16K
CL_CARDTYPE_PICOPASS_16KS
```

OverrideCardType

Задать тип карты вручную или вернуться к автоматическому определению.

```
Число OverrideCardType/ЗадатьТипКарты([in] Число Type);
```

Параметры

`[in] Type`

Одно из следующих значений :

`CL_CARDTYPE_UNKNOWN`
`CL_CARDTYPE_MIFARE_1K`
`CL_CARDTYPE_MIFARE_4K`
`CL_CARDTYPE_MIFARE_ULTRA_LIGHT`

Описание

Ридеры могут быть не в состоянии распознать автоматически тип некоторых неоригинальных mifare карт (клонов). В этом случае все же можно заставить ридер работать с такими картами, если определить тип вручную. `OverrideCardType` достаточно вызвать 1 раз после создания экземпляра класса. `CL_CARDTYPE_UNKNOWN` означает вернуться к автоопределению типа карт.

GetUID

Получить серийный номер карты.

```
Число GetUID/ПолучитьСерийныйНомер([out,retval] Строка UID)
```

Результат

Серийный номер карты.

Описание

Количество байт в серийном номере зависит от типа карты.

3.2.11 Crypto

Класс для выполнения криптографических операций.

Методы

Des/ ШифрованиеDes	DES Encrypt/Decrypt.
Des2/ ШифрованиеDes2	DES Encrypt/Decrypt.
Des3/ ШифрованиеDes3	DES2 Encrypt/Decrypt.
DesCbc/ ШифрованиеDesCbc	Шифрование массива данных алогоритмом DES CBC.
DesCbc2/ ШифрованиеDesCbc2	Шифрование массива данных алогоритмом DES2 CBC.
DesCbc3/ ШифрованиеDesCbc3	Шифрование массива данных алогоритмом DES3 CBC.
Mac/ПодписьMac	Вычисление криптографической подписи алгоритмом MAC.
Mac2/ ПодписьMac2	Вычисление криптографической подписи алгоритмом MAC2.
Mac3/ ПодписьMac3	Вычисление криптографической подписи алгоритмом MAC3.
MacEx/ ПодписьMacEx	Вычисление криптографической подписи алгоритмом MAC с вектором инициализации.
Mac2Ex/ ПодписьMac2Ex	Вычисление криптографической подписи алгоритмом MAC2 с вектором инициализации.
Mac3Ex/ ПодписьMac3Ex	Вычисление криптографической подписи алгоритмом MAC3 с вектором инициализации.
MD5/ХешMD5	Хэш-функция MD5

Des

DES.

```
Число Des/ШифрованиеDes(
    [in] Строка Key,
    [in] Строка BlockIn,
    [in] Число Encrypt,
    [out,retval] Строка BlockOut);
```

Параметры

`[in] Key`

8-байтовый ключ шифрования.

`[in] BlockIn`

8-байтовый исходный блок данных.

`[in] Encrypt`

`VARIANT_TRUE` = шифровка,
`VARIANT_FALSE` = расшифровка.

Результат

Зашифрованный или расшифрованный 8-байтовый блок данных.

Des2

2DES = DES_Encrypt(K1,DES_Decrypt(K2,DES_Encrypt(K1,data)))

```
Число Des2/ШифрованиеDes2(
    [in] Строка Key,
    [in] Строка BlockIn,
    [in] Число Encrypt,
    [out,retval] Строка BlockOut);
```

Параметры

`[in] Key`

2 8-байтовые половины ключа шифрования, совмещенные вместе в одном блоке.

`[in] BlockIn`

8-байтовый исходный блок данных.

`[in] Encrypt`

`VARIANT_TRUE` = шифровка,
`VARIANT_FALSE` = расшифровка.

Результат

Зашифрованный или расшифрованный 8-байтовый блок данных.

Des3

3DES = DES_Encrypt(K3,DES_Decrypt(K2,DES_Encrypt(K1,data)))

```
Число Des3/ШифрованиеDes3(
    [in] Строка Key,
    [in] Строка BlockIn,
    [in] Число Encrypt,
    [out,retval] Строка BlockOut);
```

Параметры

`[in] Key`

3 8-байтовые части ключа шифрования 3DES : K1,K2,K3, совмещенные вместе в одном блоке.

`[in] BlockIn`

8-байтовый исходный блок данных.

`[in] Encrypt`

`VARIANT_TRUE` = шифровка,
`VARIANT_FALSE` = расшифровка.

Результат

Зашифрованный или расшифрованный 8-байтовый блок данных.

DesCbc

Шифрование массива данных алгоритмом DES CBC.

```
Число DesCbc/ШифрованиеDesCbc(
    [in] Строка Key,
    [in] Строка DataIn,
    [in] Строка IvIn,
    [in] Число Encrypt,
    [out] Строка DataOut,
    [out] Строка IvOut);
```

Параметры

`[in] Key`

Ключ шифрования DES.

`[in] DataIn`

Входные данные.

`[in] IvIn`

Исходный вектор прерывания. Может быть 0 для первого шифрования.

`[in] Encrypt`

`VARIANT_TRUE` = шифровка, `VARIANT_FALSE` = расшифровка.

`[out] DataOut`

Выходные данные. Т.к. DES является блочным шифром, длина выравнивается на 8 байт.

`[out] IvOut`

Результирующий вектор прерывания.

Описание

Алгоритм DES CBC (Chain Block Cipher) предназначен для шифрования массивов данных любой длины. Его специфика такова, что для каждого последующего 8-байтного блока ключ получается с использованием предыдущего результата шифрования. Таким образом, шифруя однотипные данные, вы не получите повторяющиеся зашифрованные блоки.

Т.к. DES является блочным шифром, длина выходного блока данных `DataOut` выравнивается на 8 байт..

При дешифровке нет возможности определить точную длину расшифрованного блока - если он не был кратен 8, то производится дополнение нулями. Если это критично, то храните длину оригинального блока вместе с зашифрованными данными.

Вектор прерывания позволяет разделить шифрование большого массива данных на несколько операций. При первом вызове заполните `IvIn` нулями или укажите 0. По возвращению функция поместит в `IvOut` значение, необходимое для продолжения шифрования.

DesCbc2

Шифрование массива данных алгоритмом DES2 CBC.

```
Число DesCbc2/ШифрованиеDesCbc2(
    [in] Строка Key,
    [in] Строка DataIn,
    [in] Строка IvIn,
    [in] Число Encrypt,
    [out] Строка DataOut,
    [out] Строка IvOut);
```

Параметры

`[in] Key`

2 8-байтовые половины ключа шифрования, совмещенные вместе в одном блоке.

`[in] DataIn`

Входные данные.

`[in] IvIn`

Исходный вектор прерывания. Может быть 0 для первого шифрования.

`[in] Encrypt`

`VARIANT_TRUE` = шифровка, `VARIANT_FALSE` = расшифровка.

[out] DataOut

Выходные данные. Т.к. DES является блочным шифром, длина выравнивается на 8 байт.

[out] IvOut

Результирующий вектор прерывания.

Описание

Алгоритм DES CBC (Chain Block Cipher) предназначен для шифрования массивов данных любой длины. Его специфика такова, что для каждого последующего 8-байтного блока ключ получается с использованием предыдущего результата шифрования. Таким образом, шифруя однотипные данные, вы не получите повторяющиеся зашифрованные блоки.

Т.к. DES является блочным шифром, длина выходного блока данных `DataOut` выравнивается на 8 байт..

При дешифровке нет возможности определить точную длину расшифрованного блока - если он не был кратен 8, то производится дополнение нулями. Если это критично, то храните длину оригинального блока вместе с зашифрованными данными.

Вектор прерывания позволяет разделить шифрование большого массива данных на несколько операций. При первом вызове заполните `IvIn` нулями или укажите 0. По возвращению функция поместит в `IvOut` значение, необходимое для продолжения шифрования.

DesCbc3

Шифрование массива данных алгоритмом DES3 CBC.

```
Число DesCbc3/ШифрованиеDesCbc3(
    [in] Строка Key,
    [in] Строка DataIn,
    [in] Строка IvIn,
    [in] Число Encrypt,
    [out] Строка DataOut,
    [out] Строка IvOut);
```

Параметры

`[in] Key`

3 8-байтовые части ключа шифрования 3DES : K1,K2,K3, совмещенные вместе в одном блоке.

`[in] DataIn`

Входные данные.

`[in] IvIn`

Исходный вектор прерывания. Может быть 0 для первого шифрования.

`[in] Encrypt`

`VARIANT_TRUE` = шифровка, `VARIANT_FALSE` = расшифровка.

`[out] DataOut`

Выходные данные. Т.к. DES является блочным шифром, длина выравнивается на 8 байт.

`[out] IvOut`

Результирующий вектор прерывания.

Описание

Алгоритм DES CBC (Chain Block Cipher) предназначен для шифрования массивов данных любой длины. Его специфика такова, что для каждого последующего 8-байтного блока ключ получается с использованием предыдущего результата шифрования. Таким образом, шифруя однотипные данные, вы не получите повторяющиеся зашифрованные блоки.

Т.к. DES является блочным шифром, длина выходного блока данных `DataOut` выравнивается на 8 байт..

При дешифровке нет возможности определить точную длину расшифрованного блока - если он не был кратен 8, то производится дополнение нулями. Если это критично, то храните длину оригинального блока вместе с зашифрованными данными.

Вектор прерывания позволяет разделить шифрование большого массива данных на несколько операций. При первом вызове заполните `IvIn` нулями или укажите 0. По возвращению функция поместит в `IvOut` значение, необходимое для продолжения шифрования.

Mac

Вычисление криптографической подписи Mac на алгоритме DES.

```
Число Mac/ПодписьMac(
    [in] Строка Key,
    [in] Строка DataBlock,
    [out,retval] Строка Mac);
```

Параметры

`[in] Key`

Ключ шифрования DES.

`[in] DataBlock`

Входные данные.

Результат

Значение криптографической подписи.

Описание

Алгоритм MAC предназначен для вычисления криптографической подписи блока данных произвольной длины.

Mac2

Вычисление криптографической подписи Mac на алгоритме 2DES.

```
Число Mac2/ПодписьMac2(
    [in] Строка Key,
    [in] Строка DataBlock,
    [out,retval] Строка Mac);
```

Параметры

`[in] Key`

2 8-байтовые половины ключа шифрования, совмещенные вместе в одном блоке.

`[in] DataBlock`

Входные данные.

Результат

Значение криптографической подписи.

Описание

Алгоритм MAC предназначен для вычисления криптографической подписи блока данных произвольной длины.

Mac3

Вычисление криптографической подписи Mac на алгоритме 3DES.

```
Число Mac3/ПодписьMac3(
    [in] Строка Key,
    [in] Строка DataBlock,
    [out,retval] Строка Mac);
```

Параметры

`[in] Key`

3 8-байтовые части ключа шифрования 3DES : K1,K2,K3, совмещенные вместе в одном блоке.

`[in] DataBlock`

Входные данные.

Результат

Значение криптографической подписи.

Описание

Алгоритм MAC предназначен для вычисления криптографической подписи блока данных произвольной длины.

MacEx

Вычисление криптографической подписи Mac на алгоритме DES с вектором инициализации.

```
Число МасЕх/ПодписьМасЕх(
    [in] Строка Key,
    [in] Строка DataBlock,
    [in] Строка InitVector,
    [out,retval] Строка Mac);
```

Параметры

`[in] Key`

Ключ шифрования DES.

`[in] DataBlock`

Входные данные.

`[in] InitVector`

Вектор инициализации (8 байт).

Результат

Значение криптографической подписи.

Описание

Алгоритм MAC предназначен для вычисления криптографической подписи блока данных произвольной длины.

Mac2Ex

Вычисление криптографической подписи Mac на алгоритме 2DES с вектором инициализации.

```
Число Mac2Ex/ПодписьMac2Ex(
    [in] Строка Key,
    [in] Строка DataBlock,
    [in] Строка InitVector,
    [out,retval] Строка Mac);
```

Параметры

`[in] Key`

2 8-байтовые половины ключа шифрования, совмещенные вместе в одном блоке.

`[in] DataBlock`

Входные данные.

`[in] InitVector`

Вектор инициализации (8 байт).

Результат

Значение криптографической подписи.

Описание

Алгоритм MAC предназначен для вычисления криптографической подписи блока данных произвольной длины.

Mac3Ex

Вычисление криптографической подписи Mac на алгоритме 3DES с вектором инициализации.

```
Число Mac3Ex/ПодписьMac3Ex(
    [in] Строка Key,
    [in] Строка DataBlock,
    [in] Строка InitVector,
    [out,retval] Строка Mac);
```

Параметры

`[in] Key`

3 8-байтовые части ключа шифрования 3DES : K1,K2,K3, совмещенные вместе в одном блоке.

`[in] DataBlock`

Входные данные.

`[in] InitVector`

Вектор инициализации (8 байт).

Результат

Значение криптографической подписи.

Описание

Алгоритм MAC предназначен для вычисления криптографической подписи блока данных произвольной длины.

MD5

Вычисление хэш-функции MD5.

```
Число MD5/ХешMD5(
    [in] Строка DataBlock,
    [out,retval] Строка MD5);
```

Параметры

`[in] DataBlock`

Входные данные.

Результат

Значение MD5.

3.2.12 DataHelper

Класс для преобразования данных из их обычного формата записи в HEX строки и обратно.

Методы

StringToDouble/СтрокуВДробное	Преобразование строки в дробное.
DoubleToString/ДробноеВСтроку	Преобразование дробного в строку.
StringToValue/СтрокуВЧисло	Преобразование строки в число.
ValueToString/ЧислоВСтроку	Преобразование числа в строку.
StringToHexString/ СтрокуВБинарнуюСтроку	Преобразование текстовой строки в HEX строку.
HexStringToString/ БинарнуюСтрокуВСтроку	Преобразование HEX строки в текстовую строку.

StringToDouble

Функция преобразует HEX строку в дробное число.

```
Число StringToDouble/СтрокуВДробное(
    [in] Строка НехСтрока,
    [out,retval] ДробноеЧисло Число);
```

Параметры

`[in] НехСтрока`

Строка, преобразуемая в ДробноеЧисло.

`[out] Число`

Преобразованное ДробноеЧисло.

Результат

Преобразованное из строки в дробное число.
Например строка "000000000000f03f" преобразуется в дробное число 1.0

DoubleToString

Функция преобразует HEX строку в дробное число.

```
Число DoubleToString/ДробноеВСтроку(
    [out,retval] Строка НехСтрока,
    [in] ДробноеЧисло Число);
```

Параметры

`[out] НехСтрока`

Строка с преобразованным числом ДробноеЧисло.

`[in] Число`

Преобразуемое ДробноеЧисло.

Результат

Преобразованная строка из дробного числа.
Например дробное число 1.0 преобразуется в строку "000000000000f03f".

ValueToString

Функция преобразует Число в HEX строку.

```
Число ValueToString/ЧислоВСтроку(
    [out,retval] Строка НехСтрока,
    [in] Число Число);
```

Параметры

`[out] НехСтрока`

Строка с преобразованным числом Число.

`[in] Число`

Преобразуемое Число.

Результат

Преобразованная строка из числа.
Например, число 100 преобразуется в строку "64000000".

StringToValue

Функция преобразует HEX строку в Число.

```
Число StringToValue/СтрокуВЧисло(
    [in] Строка НехСтрока,
    [out,retval] Число Число);
```

Параметры

`[in] НехСтрока`

Строка с преобразуемым числом Число.

`[out] Число`

Преобразованное Число.

Результат

Преобразованное число из HEX строки.
Например, строка "64000000" преобразуется в число 100.

HexStringToString

Функция преобразует HEX строку текстовую строку.

```
Число HexStringToString/БинарнуюСтрокуВСтроку(
    [in] Строка HEXСтрока,
    [out,retval] Строка Строка);
```

Параметры

`[in] HEXСтрока`

Преобразуемая HEXСтрока строка.

`[out] Строка`

Преобразованная текстовая строка.

Результат

Преобразованная текстовая строка.
Например, строка "54455354" преобразуется в строку "TEST".

StringToHexString

Функция преобразует текстовую строку в HEX-строку.

```
Число StringToHexString/СтрокуВБинарнуюСтроку(
    [out,retval] Строка HEXСтрока,
    [in] Строка Строка);
```

Параметры

`[out] HEXСтрока`

Преобразованная в HEX текстовая строка.

`[in] Строка`

Преобразуемая текстовая строка.

Результат

Преобразованная текстовая строка.

Например строка "TEST" преобразуется в строку "54455354".

4 Техническая поддержка и контакты

Техническая поддержка SDK для 1С осуществляется компанией "Интеллектуальные системы управления бизнесом".

Телефон офиса в Москве : (495) 739-8699

Web-site : http://www.isbc.ru, http://www.smart-card.ru

Вопросы в электронном виде можно отправить через форму обратной связи : http://isbc.ru/_mail/post_form_

Версия документации : 2.1