ruToken®

Российское средство аутентификации
и защиты информации

Руководство пользователя

Издание 1.0

Содержание

Часть 1
Введение

Что такое ruToken

ruToken представляет собой компактное электронное устройство, предназначенное для надежной идентификации пользователей в компьютерных системах. ruToken имеет интерфейс USB, что позволяет использовать его с любым компьютером, оборудованным соответствующим интерфейсом. ruToken обладает всеми свойствами, присущими смарт-картам: он обеспечивает надежность, простоту и безопасность процесса идентификации. Для его использования не требуется считывающее устройство (ридер), необходимое для смарт-карт, что делает возможным использование ruToken практически на любом современном компьютере.

ruToken может быть использован для множества приложений, где прежде использовались пароли. Применение токена позволяет избежать затруднений, связанных с запоминанием действительно сильных паролей. В памяти устройства можно хранить гораздо более сложные пароли, чем может запомнить обычный человек, что делает ruToken незаменимым в приложениях, требующих высокого уровня безопасности.

ruToken представляет собой комбинацию активного и пассивного устройств аутентификации, в зависимости от задачи. Он основан на микропроцессоре, способном выполнять криптографические преобразования информации, что делает его использование предпочтительным при обмене конфиденциальной информацией – ключи шифрования не покидают памяти ruToken и, следовательно, не могут быть перехвачены и скомпрометированы.

Важнейшим отличием ruToken от представленных на рынке токенов зарубежного производства является поддержка Российского стандарта шифрования по ГОСТ 28147-89.

ruToken может быть использован для хранения персональной информации, паролей, ключей шифрования, сертификатов, лицензий, удостоверений и других данных, которые могут не просто храниться, но и предъявляться токеном в процессе аутентификации.

Есть четыре основных повода хранить данные в памяти ruToken, а не на диске:

- **Надежность.** Хранение персональной информации в памяти устройства обеспечивает ее безопасность и сохранность в случаях сбоев системы и потери информации, хранящейся на жестком диске
- **Переносимость.** Маленькие габариты и вес ruToken делают его идеальным для сетевых приложений, а также для использования его дома и в офисе
- **Защищенность.** ruToken решает проблемы хранения идентификационной информации, паролей и т. п. на жестких дисках, которые могут стать доступными вирусам, троянским программам и т. п. Он хранит критическую информацию вне жестких дисков, тем не менее, позволяя пользователям иметь эту информацию всегда при себе
- **Невоспроизводимость.** Пользователи не могут дублировать информацию, хранимую в памяти ruToken. Это означает возможность хранения учетных записей, дающих доступ к Интернет-сервисам и другим ресурсам с гарантией того, что никто другой не сможет ими воспользоваться

Безопасность в ruToken

Безопасность является одним из главных аргументов в пользу применения подобных устройств. ruToken предоставляет вычислительную платформу, на которой информация может храниться и обрабатываться безопасно. Вследствие этого ruToken идеально подходит для повышения безопасности других систем.

Информация, хранящаяся в памяти ruToken, может быть организована таким образом, чтобы доступ к ней полностью контролировался его владельцем или поставщиком приложений.

Контроль доступа обычно реализуется требованием ключей и / или PIN-кодов для доступа к данным. Ключи и PIN-коды хранятся в памяти токена в специальных объектах файловой системы. В отличие от ключей, вводимых пользователем или получаемых с каких-либо устройств (например, дискет, жестких и оптических дисков), доступ к этим объектам может иметь имеет только сам токен.

Токены предоставляют целый ряд полезных возможностей:

- Хранение паролей для доступа к системам, сетям и т. п.
- Хранение ключей шифрования для обеспечения секретности
- Хранение ключей для целей аутентификации
- Выполнение алгоритмов шифрования для аутентификации
- Выполнение алгоритмов шифрования для обеспечения секретности информации
- Безопасное хранение информации

Спектр возможностей ruToken

Интерфейс и поддерживаемые ОС

- Имеет встроенный микропроцессор
- Интерфейс USB (USB 1.1 / USB 2.0)
- Скорость USB-обмена – до 1,5 Мбит/с
- Не требует дорогостоящих карт-ридеров
- Поддерживается работа с несколькими устройствами ruToken
- Поддерживаемые ОС: Windows 98/Me/2000/XP

Поддержка стандартов и стандартных API

- Поддержка стандарта ISO/IEC 7816
- Поддержка стандарта PC/SC
- Поддержка стандарта ГОСТ 28147-89
- Поддержка функций Microsoft Crypto API и Microsoft Smartcard API
- Поддержка функций PKCS #11 (v. 2.10+)
- Собственные Cryptographic Service Provider и ICC Service Provider со стандартными наборами интерфейсов и функций API
- Возможность интеграции в любые smartcard-ориентированные программные продукты (e-mail-, internet-, платежные системы и т.п.)

Файловая система

- Энергонезависимая память для хранения конфиденциальных данных
- Встроенная файловая система по стандарту ISO/IEC 7816
- Прозрачное шифрование содержимого памяти по ГОСТ 28147-89 на основе данных, уникальных для каждого экземпляра ruToken
- Возможность создания папок и файлов в памяти ruToken
- Возможность создания объектов данных для security-ориентированных операций и для мер по обеспечению безопасности
- Возможность назначения свойств, прав доступа и атрибутов безопасности каждому объекту файловой системы ruToken: папке, файлу и объекту данных
- Количество создаваемых папок и уровень их вложенности – не ограничено (ограничивается только объемом памяти)
- Количество создаваемых объектов файловой системы в каждой папке – до 255
- Наличие предопределенных папок для объектов данных разных типов для повышения удобства их использования
- Автоматический поиск объекта данных в памяти токена на основе типа и ID объекта
- Администрирование объектов файловой системы: создание, модификация, удаление, использование для различных операций
- Возможность присвоения токену символьного имени для упрощения его визуальной идентификации

Поддержка ГОСТ 28147-89

- Встроенный алгоритм шифрования ГОСТ 28147-89
- Шифрование внешних данных в любом из 3 режимов: простая замена, гаммирование и гаммирование с обратной связью
- Выработка 32-битовой имитовставки

- Генерация 256-битовых случайных чисел
- Защищенное хранение ключей шифрования ГОСТ 28147-89 в объектах данных без возможности их экспорта из токена
- Возможность импорта ключей шифрования ГОСТ 28147-89 в токен

Аутентификация и конфиденциальность

- PIN-коды Пользователя и Администратора ruToken
- 2-факторная аутентификация (по факту наличия ruToken и по факту предъявления PIN-кода)
- Ограничение числа последовательных неудачных попыток аутентификации с предъявлением PIN-кода
- Блокирование PIN-кода по достижении предельного числа последовательных неудачных попыток аутентификации
- Разграничение прав Пользователя и Администратора
- Возможность авторизованного изменения PIN-кодов и администрирования токена
- 32-битовый уникальный серийный номер токена

Дополнительные возможности

- Программно-аппаратная поддержка стандарта X.509
- Программно-аппаратная поддержка алгоритмов RSA, DES (3DES), RC2, RC4, MD4, MD5, SHA-1
- Защищенное хранение ключей асимметричного шифрования и цифровых сертификатов
- Возможность использования ruToken для асимметричного шифрования данных и для работы с цифровыми сертификатами из любых smartcard-приложений, поддерживающих стандарт PC/SC
- Протестирована работа с e-mail-клиентами: Microsoft Outlook, Microsoft Outlook Express
- Протестирована работа с центрами сертификации: Microsoft, Verisign

Программное обеспечение ruToken

- Дополнительные интерфейсы ICC Service Provider для повышения удобства работы и для реализации дополнительных возможностей ruToken
- Сервисная библиотека классов C++, предназначенная для облегчения разработки приложений, использующих ruToken. Является надстройкой над SmartCard API и предоставляет классы, облегчающие работу с ruToken на уровне APDU
- Утилита программирования ruToken:
  - Создание объектов файловой системы ruToken: папок, файлов, ключей шифрования и т.п.
  - Назначение прав доступа к объектам файловой системы
  - Редактирование символьного имени токена
  - Манипулирование объектами файловой системы: получение сведений, просмотр, редактирование, удаление и т.п.
  - Шифрование внешних данных по алгоритму ГОСТ 28147-89
- Утилита администрирования ruToken:
  - Получение сведений о выбранном токене
  - Инициализация памяти ruToken
  - Изменение PIN-кодов Пользователя и Администратора
  - Разблокирование PIN-кода Пользователя
- Подробные примеры работы с ruToken для всех поддерживаемых API
- Набор инсталляторов отдельных компонентов ПО ruToken для удобного переноса ПО на компьютеры конечных пользователей

Применение ruToken

Аутентификация

- Замена парольной защиты при доступе к БД, Web-серверам, VPN-сетям и security-ориентированным приложениям на программно-аппаратную аутентификацию
- Шифрование соединений при доступе к почтовым серверам, серверам баз данных, Web-серверам, файл-серверам, аутентификация при удалённом администрировании и т.п.
- Использование разнообразных методов аутентификации и любых Центров Сертификации

Защита собственных данных

- Защита электронной почты (электронная цифровая подпись, шифрование), защита компьютера (поддержка WinLogon)
- Контроль доступа к компьютеру, поддержка любого числа пользователей (владельцев ruToken) на одном компьютере
- Безопасное хранение в одном устройстве большого количества данных: файлов, ключей шифрования, цифровых сертификатов и т. п.

Корпоративное использование

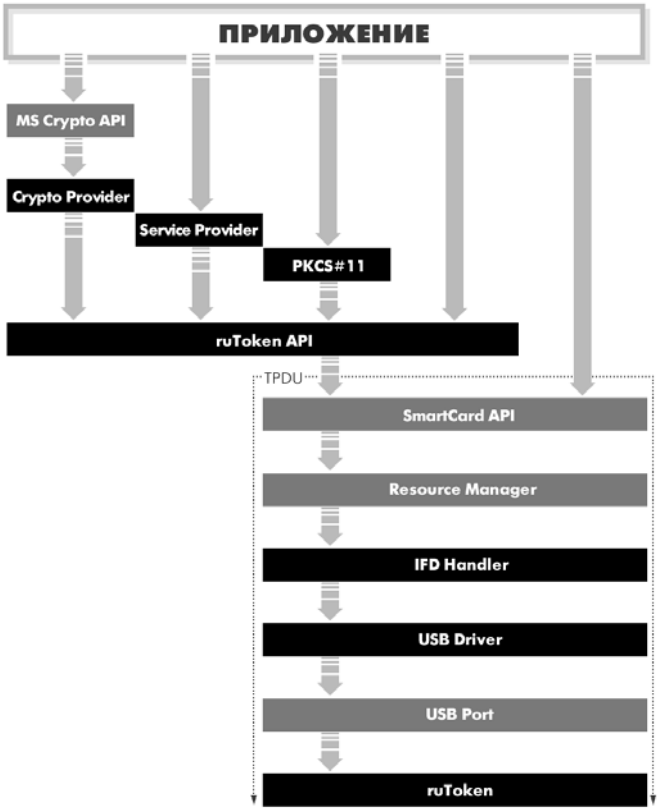
- Использование в любых прикладных программах и системах электронной торговли для хранения служебной информации, персональной информации пользователей, паролей, ключей шифрования, цифровых сертификатов и любой другой конфиденциальной информации
- Использование ruToken как единого идентификационного устройства для доступа пользователя к разным элементам корпоративной системы и для обеспечения, например, необходимого разграничения прав доступа, автоматической цифровой подписи создаваемых документов, аутентификации при доступе к компьютерам и приложениям системы

Обзор архитектуры ruToken

Архитектура ruToken была разработана в соответствии с требованиями индустриального стандарта PC/SC. В ней можно выделить четыре уровня:

- Аппаратный уровень
- Интерфейс низкого уровня, включающий средства операционной системы
- Интерфейс среднего уровня
- Интерфейсы высокого уровня

На рисунке эти уровни изображены в графическом виде.



Аппаратный уровень

Самый нижний уровень представлен физическими устройствами: непосредственно токеном и хост-контроллером USB. Взаимодействие между токеном и хост-контроллером осуществляется по протоколу USB Control Transfer Protocol, который использует Vendor Specific Requests (VSR).

Основным аппаратным элементом ruToken является защищенный микроконтроллер, реализующий функции интерфейса USB, поддержку файловой системы и команд по ISO 7816, выполняющий криптографические преобразования по ГОСТ 28147-89, а также другие функции.

Интерфейс низкого уровня

Над аппаратным уровнем находится интерфейс, обеспечивающий представление ruToken интерфейсам более высокого уровня в качестве смарт-карты, вставленной в ридер. Интерфейс образуется взаимодействием между USB-драйвером ruToken, ruToken IFD Handler, Microsoft Resource Manager и Microsoft SmartCard API. Взаимодействие этого интерфейса с интерфейсом аппаратного уровня происходит путем передачи однозначно определенных команд с использованием Transport Protocol Data Units (TPDU) по протоколу T=0.

Интерфейс среднего уровня

Этот интерфейс располагается над Microsoft SmartCard API – низкоуровневым интерфейсом прикладного программирования и состоит из библиотеки ядра ruToken (rtAPI). Интерфейс оперирует понятием APDU (Application Protocol Data Unit), преобразуя их в TPDU T=0 для передачи интерфейсу низкого уровня.

Интерфейсы высокого уровня

Самый высокий уровень сформирован из реализаций различных стандартов (PKCS #11) и API (Microsoft Crypto API, ICC Service Provider), которые могут взаимодействовать с интерфейсами низкого и среднего уровней и могут оперировать или не оперировать понятием “смарт-карта”. Сообщение с интерфейсами более низких уровней происходит путем вызовов однозначно определенных интерфейсных функций и их трансформацией в APDU на среднем уровне.

Поддержка стандарта PC/SC

PC/SC – это широко используемый промышленный стандарт программного интерфейса для приложений, построенных на платформе PC и оперирующих со смарт-картами.

Назначение и смысл архитектуры и спецификации PC/SC состоит в том, чтобы дать возможность производителям разрабатывать свои продукты независимо. Кроме того, разработчики программного обеспечения, ориентированного на использование смарт-карт, могут теперь не принимать в расчет особенности каждого конкретного ридера, имея стандартный универсальный интерфейс программирования. Рисунок иллюстрирует архитектуру PC/SC в общем виде:

Архитектура PC/SC определяет интерфейс между ридером смарт-карт, менеджером ресурсов (Resource Manager) и приложением, ориентированным на использование смарт-карт. С данной точки зрения, все ридеры ведут себя одинаково. В комплект поставки ruToken входит драйвер ридера, который делает возможным подключение виртуального ридера ruToken к менеджеру ресурсов. Интерфейс прикладного программирования в данном случае обеспечивается Сервис-провайдером ruToken, состоящим из двух компонент: Сервис-провайдера смарт-карты (ICC Service Provider, ICCSP) и Криптографического сервис-провайдера (Cryptographic Service Provider, CSP).

Соответствие общему стандарту, такому как PC/SC, гарантирует способность приложений к взаимодействию, снижает стоимость разработки и помогает поддерживать постоянный уровень безопасности, требуемый при использовании технологий смарт-карт. ruToken включает технологии и стандарты смарт-карт и полностью совместим со стандартом PC/SC.

Любое приложение, ориентированное на использование PC/SC-совместимых смарт-карт, может быть легко приспособлено для поддержки ruToken, точно так же как и для поддержки любой другой PC/SC-совместимой смарт-карты.

Более подробную информацию о стандарте PC/SC можно найти на сайте http://www.pcscworkgroup.com/.

Комплект разработчика ruToken

Установка и обновление Комплекта разработчика

Системные требования

При работе с электронными идентификаторами ruToken предъявляются следующие минимальные требования к аппаратному и программному обеспечению:

- IBM-совместимый компьютер
- Стандартный USB-порт
- Операционная система Windows 98/ME/2000/XP

Состав Комплекта разработчика

Комплект разработчика содержит все необходимое для работы с ruToken:

- Два электронных идентификатора
- Компакт-диск с программным обеспечением ruToken
- Руководство пользователя

Установка программного обеспечения ruToken

Важная информация
Программное обеспечение ruToken должно быть установлено до первого подключения ruToken к USB-порту.

Вставьте компакт-диск ruToken в CD-ROM компьютера. Установка ПО ruToken начнется автоматически (если автозапуск запрещен, то запустите программу AUTORUN.EXE из корневого каталога компакт-диска). Появится диалоговое окно программы установки.

Руководствуясь указаниями программы-установщика, выполните следующие действия:

- Выберите тип установки – полная или выборочная
- При необходимости определите каталоги для размещения ПО ruToken
- В случае выборочной установки отметьте необходимые компоненты Комплекта разработчика
- После установки ПО перезагрузите ОС
- После перезагрузки все готово для начала работы с ruToken

Важная информация

1. Все компоненты, которые могут понадобиться для работы с электронными идентификаторами, уже отмечены по умолчанию. Не рекомендуется изменять конфигурацию ПО без особой надобности.
2. В процессе установки ПО ruToken производится установка в систему драйверов ruToken, а также Service Provider ruToken и библиотеки PKCS #11, если эти компоненты Комплекта разработчика были отмечены вами для установки.
3. Во время установки ПО все приложения должны быть закрыты во избежание ошибки разделения файлов.
4. Пользователь, работающий с Windows 2000/XP, должен обладать правами администратора системы, иначе установка драйверов ruToken будет невозможна.

В процессе установки ПО создается программная группа ruToken.

Установка электронного идентификатора

- Установите драйвер электронного идентификатора ruToken (см. ниже раздел "Драйверы").
- Подсоедините ruToken к USB-порту.

Обновление программного обеспечения ruToken

Для получения новых возможностей и предотвращения возможных конфликтов с программным и аппаратным обеспечением рекомендуется регулярно обновлять программное обеспечение ruToken.

Обновление с компакт-диска

- Вставьте компакт-диск с новой версией ПО ruToken в привод CD-ROM компьютера
- Выберите элемент программной группы ruToken "Обновление Комплекта разработчика"
- Укажите папку, в которой находятся файлы обновления

Далее действуйте, как описано в пункте "Установка программного обеспечения ruToken".

Обновление через Internet

Для обновления ПО ruToken через Интернет воспользуйтесь утилитой GSoftUpd.exe из состава Комплекта разработчика:

- Установите соединение с Internet
- Выберите элемент программной группы ruToken **Обновление Комплекта разработчика On-Line**

- Появится окно GSoftUpd.exe, в котором можно определить директорию размещения файлов обновления и настроить параметры соединения с Internet. Для продолжения нажмите на кнопку **[Далее]**
- Утилита автоматически загрузит с сайта компании «Актив» (www.rutoken.ru) информацию о последнем релизе ruToken
- Выберите из списка в левой части диалогового окна компоненты, которые необходимо загрузить (отсутствующие или устаревшие компоненты заранее отмечены флажками), и нажмите на кнопку **[Загрузить]**
- Запустите программу установки и следуйте ее указаниям

Примечание

Загруженные файлы обновления сохраняются. Их можно использовать повторно, например, при аварийном завершении сеанса связи.

Драйверы ruToken

Драйверы необходимы для работы электронных идентификаторов, утилит Комплекта разработчика, а также любых решений на основе ruToken. В состав драйверов входят IFD Handler и USB-драйвер ruToken.

При установке программного обеспечения ruToken файлы драйверов копируются в подкаталог /DRIVERS.

Для установки и удаления драйверов служит программа SetupDrv.exe.

Важная информация

1. Во время установки драйверов все приложения должны быть закрыты во избежание ошибки разделения файлов.

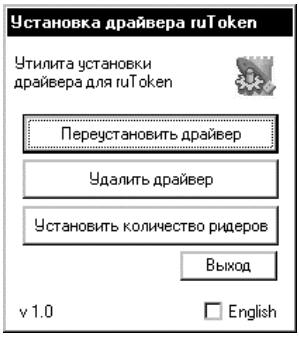
2. Пользователь, работающий с Windows 2000/XP, должен обладать правами администратора системы, иначе установка драйверов будет невозможна.

3. Драйверы ruToken можно устанавливать только при помощи программы SetupDrv. Не следует делать это "вручную" при помощи стандартного мастера Windows.

4. ruToken не следует подключать к компьютеру до установки драйверов, иначе Windows начнет самостоятельный поиск драйвера для них. Если же это произошло, следует отказаться от поиска драйвера мастером Windows, нажав кнопку [Отмена], и отсоединить ruToken от компьютера.

Установка драйверов

- Запустите SetupDrv.exe (элемент программной группы ruToken "Управление драйвером ruToken").



При запуске утилита проверяет наличие драйверов в системе. Если драйверы ruToken не найдены, в главном окне программы будет доступна только одна кнопка – **[Установить драйвер]**

- Нажмите на кнопку **[Установить драйвер]** (**[Переустановить драйвер]**)

При успешной установке программа выдаст соответствующее сообщение, после чего произойдет возврат в главное окно.

Установка числа ридеров ruToken

Ридер смарт-карты – это физическое и / или логическое устройство, необходимое для взаимодействия смарт-карты и операционной системы. Т. к. ruToken для операционной системы представляет собой смарт-карту, для него также нужно устанавливать (резервировать) ридер(ы). Числом ридеров ruToken, установленных в системе, определяется количество идентификаторв ruToken, которые могут обслуживаться системой одновременно.

Чтобы назначить количество ридеров ruToken, нажмите на кнопку **[Задать число ридеров]**.

В появившемся диалоговом окне с помощью развертывающегося списка установите нужное число (от 1 до 10, по умолчанию - 3) и перезагрузите компьютер.

Важная информация

1. В операционной системе Windows максимальное число ридеров смарт-карт ограничено 10-ю.

2. Максимальное число ридеров смарт-карт – это величина, разделяемая между всеми типами смарт-карт, присоединяемых к компьютеру. Поэтому, если помимо идентификаторов ruToken используются устройства других производителей, установку числа ридеров ruToken следует производить с учетом потребностей в других токенах / смарт-картах.

Удаление драйверов

Для удаления загруженных в систему драйверов нужно нажать кнопку **[Удалить драйвер]** в главном окне утилиты SetupDrv.exe

Важная информация

Драйверы ruToken остаются активными до перезагрузки операционной системы.

Использование опций командной строки

Утилиту SetupDrv.exe можно использовать в пакетном режиме из командной строки. При запуске с некоторыми опциями командной строки главное окно SetupDrv.exe не выводится.

Программа установки поддерживает следующие опции командной строки:

/H,/?	Выдать окно подсказки со списком доступных опций.
/U	Удалить драйверы из системы (без опции /Q не имеет особого смысла: вы просто попадаете в главное окно, где должны нажать **[Удалить]**).
/LE	Принудительно установить английский язык для всех сообщений.
/LR	Принудительно установить русский язык для всех сообщений. Если ни /LE, ни /LR не заданы, язык сообщений выбирается автоматически.
/Q	Ничего не показывать на экране. Полезна при запуске SetupDrv из других приложений. По умолчанию инсталлирует, с опцией /U – удаляет драйверы.

/O	Всегда перезаписывать установленные в системе драйверы.
/LOG	Создавать файл отчета SetupDrv.log в папке с SetupDrv.exe. Если в этой папке файл не может быть создан, он создается в папке %WinDir%.
/RB	Игнорировать опцию /Q для вопроса о перезагрузке системы (если указана опция /Q и не указана /RB, то при необходимости перезагрузки системы SetupDrv сделает это без вопросов, что не всегда удобно).
/ED	Игнорировать опцию /Q для сообщений об ошибках.
/NORB	Не перезагружать систему ни в коем случае.

Коды возврата SetupDrv

0	Работа успешно завершена.
1	Работа успешно завершена, требуется перезагрузка ОС (опция /NORB).
2	Операция прервана пользователем (например, нажатие кнопки [Отмена]).
3	Попытка установить компоненту в неподдерживаемой ОС (старая Windows).
4	В системе уже установлена более свежая компонента, установка не произведена (нет опции /O).
5	Ошибка в процессе установки.
6	Недостаточно прав у пользователя для выполнения операции.
-1	Код возврата после выдачи подсказки.

Коды возврата выдаются всегда, вне зависимости от того, работает ли утилита в оконном или в пакетном режиме.

Часть 2
Как устроен ruToken

В этой главе рассказывается об устройстве и принципах функционирования файловой системы ruToken, описаны форматы объектов файловой системы ruToken.

Файловая система ruToken

Энергонезависимая память ruToken содержит полноценную файловую систему, удовлетворяющую требованиям стандарта ISO 7816. Благодаря наличию File Allocation Table (FAT) поддерживается возможность работы с объектами произвольного (а не фиксированного) размера.

Файловая система ruToken образуется **объектами файловой системы**. Все объекты файловой системы функционально делятся на файлы, папки и объекты данных.

В памяти ruToken можно создать иерархическую структуру папок с неограниченным уровнем вложенности. При этом в каждой папке можно создать до 255 объектов файловой системы.

Файлы и папки в ruToken

Файловая система ruToken оперирует файлами следующих типов:

- **Файл** (в терминах ISO 7816 – **EF**, Elementary File)
- **Папка** (в терминах ISO 7816 – **DF**, Dedicated File)
- **Корневая папка** (в терминах ISO 7816 – **MF**, Master File)

Корневая папка – это особая папка (DF-файл), наличие которой в системе обязательно. Корневая папка образует вершину иерархической файловой структуры ruToken, все остальные объекты файловой системы ruToken включены в нее.

Понятие текущих файла / папки

Большинство файловых команд ruToken оперируют такими понятиями, как **текущий файл** и **текущая папка**. Текущим называется файл или папка, специальным образом выбранные в результате выполнения определенных действий. Информация о

текущих файле / папке хранится в оперативной памяти ruToken, что, помимо прочего, ускоряет доступ к ним.

Текущая папка *всегда присутствует* в системе. После подключения ruToken к компьютеру текущей папкой становится Корневая папка. Сменить текущую папку можно путем:

- Создания новой папки командой "Создать файл"
- Выбора новой папки командой "Выбрать файл"
- Удаления текущей папки командой "Удалить файл" (в этом случае текущей становится родительская папка)

Кроме того, в системе *может присутствовать* и текущий файл. Он становится текущим после:

- Его создания командой "Создать файл"
- Его выбора командой "Выбрать файл"

Однако, в отличие от текущей папки, текущий файл может отсутствовать. Так, после подключения ruToken к компьютеру текущего файла в системе нет. Кроме того, в процессе работы с файловой системой текущий файл может *сброситься*. Это происходит при:

- Создании новой папки командой "Создать файл"
- Выборе новой папки командой "Выбрать файл"
- Удалении текущего файла командой "Удалить файл"

Идентификация файлов и папок в ruToken

Файлы и папки в файловой системе ruToken содержат идентификационные данные, при помощи которых можно найти нужный файл / папку и получить к нему доступ. Эти данные хранятся в **заголовке** файла / папки.

Для идентификации используются следующие данные:

- **Дескриптор** – 2-байтовое значение, в котором хранится тип (файл или папка) и другая информация
- **ID** (идентификатор файла / папки) – 2-байтовое числовое значение, интерпретируемое в качестве имени файла / папки

Кроме того, в заголовке хранится:

- **Размер** файла / папки и **длина тела** файла
- **Security Attributes**, определяющие, какие файловые команды в отношении данного файла / папки должны быть предварены особыми security-ориентированными

действиями, и задающие конкретные security-ориентированные действия для каждой из этих команд

- Другая информация

Диапазон ID и зарезервированные ID

Файлы и папки могут иметь ID в диапазоне от 0x0000 до 0xFFFF.

Два ID зарезервированы в файловой системе ruToken для особых нужд:

0x3F00	Идентификатор Корневой папки (Master File).
0x3FFF	"Виртуальный" идентификатор, имеющий для файловой системы значение "Текущая папка".

Во избежание коллизий *настоятельно не рекомендуется* создавать новые файлы / папки с указанными ID.

Объекты данных в ruToken

Объекты данных (в терминах ISO 7816 – **DO**, Data Object) – это объекты файловой системы ruToken, в которых хранится разного рода служебная информация: PIN-коды, наборы прав доступа, окружения безопасности, ключи шифрования для различных алгоритмов и т.п. Главное отличие объекта данных от файла состоит в том, что его содержимое не может быть получено из токена ни при каких условиях.

Все объекты данных (DO) состоят из заголовка и тела. В заголовке DO хранится информация, сходная с информацией файла или папки. Тело объекта либо полностью лежит в заголовке, либо находится вне его – в зависимости от формата объекта данных.

Форматы объектов данных

Различают 2 формата объектов данных DO: **стандартный** и **компактный**.

- DO стандартного формата (стандартный DO) имеет тело, расположенное вне заголовка DO. Длина тела стандартного объекта данных может быть произвольной. Типичный пример DO стандартного формата – объект, содержащий ключ шифрования ГОСТ 28147-89 (GOST-объект)
- DO компактного формата (компактный DO) имеет тело, полностью находящееся в его заголовке. Компактными могут быть DO, имеющие тело длиной не более

16 байт. Типичный пример DO компактного формата – объект, содержащий PIN-код (GCHV-объект)

Если создаваемый объект данных имеет длину тела не более 16 байт, он будет автоматически маркирован как компактный DO.

Формат DO задается во флагах, входящих в состав заголовка объекта данных.

Степень открытости объектов данных

По степени открытости объекты данных делятся на **открытые DO**, **DO с открытой длиной** и **закрытые DO**:

- И длина тела, и само тело **открытого DO** могут быть получены при чтении параметров DO специальной командой. Открытыми могут быть только DO, не содержащие секретной информации (например, SE-объекты)
- У **DO с открытой длиной** можно получить длину тела, но не само тело объекта. У таких DO тело обычно содержит секретную информацию, а длина может быть раскрыта, например, для дополнительной идентификации объекта. Пример подобного DO – это объект с ключом шифрования; длина тела такого DO указывает на длину самого ключа шифрования
- Ни длина тела, ни само тело **закрытого DO** не могут быть получены из токена ни при каких обстоятельствах. Закрытый объект обычно содержит информацию такого рода, что даже открытие объема этой информации недопустимо. Типичный пример такого DO – объект, содержащий PIN-код (GCHV-объект)

Любой объект данных может быть создан закрытым. Однако не каждый DO может быть с открытой длиной и тем более – открытым. Ограничения, налагаемые на степень открытости DO, зависят от типа объекта и будут рассмотрены ниже.

Степень открытости DO также задается во флагах его заголовка.

Типы объектов данных

На сегодняшний день ruToken поддерживает следующие типы объектов данных:

- **Security Environment-объекты (SE-объекты)** – содержат окружения безопасности (Security Environments)
- **Global Cardholder Verification-объекты (GCHV-объекты)** – содержат PIN-коды доступа ко всему токену, а также права доступа, устанавливаемые при успешной аутентификации владельца токена
- **Ключи ГОСТ (GOST-объекты)** – содержат ключи шифрования ГОСТ 28147-89

Ниже дана сводная таблица, показывающая, какие сочетания форматов и степеней открытости могут быть назначены при создании различных объектов данных:

Тип объекта данных	Допустимые форматы	Допустимые степени открытости
SE-объект	Компактный Стандартный	Закрытый С открытой длиной Открытый
GCHV-объект	Компактный	Закрытый
GOST-объект	Стандартный	Закрытый С открытой длиной

Диапазоны ID объектов данных

ID объекта данных, в отличие от ID файла, имеет длину 1 байт. Диапазоны ID DO:

- GCHV-объекты – от 0x01 до 0x1F
- Остальные объекты - от 0x01 до 0xFE

Местоположение DO и предопределенные ID. Структура предопределенных папок.

Объекты данных можно создавать в любой папке. Однако, для повышения удобства работы с объектами, в файловой системе ruToken было введено понятие **предопределенного ID** объекта и соответствующих **предопределенных папок**, а файловая система имеет механизм автоматического поиска объекта данных.

Чтобы различать, в какой папке (предопределенной или произвольной) лежит DO, используется старший бит его ID. Если старший бит ID содержит 0, такой ID будет считаться предопределенным, а сам объект данных при создании будет автоматиче-

ски помещен в предопределенную папку для DO данного типа. Если же старший бит ID содержит 1, такой объект данных будет создан в текущей папке.

Таким образом:

- Все объекты данных с ID не более чем 0x7F будут всегда расположены в предопределенной папке для объектов соответствующего типа
- Объекты данных с ID 0x80 и больше будут расположены в папке, текущей на момент их создания

Использование предопределенных ID имеет ряд преимуществ:

- При создании такой объект данных автоматически помещается в "свою", а не в текущую папку
- При выполнении наиболее часто используемых команд не требуется ни знать, где именно находится нужный объект данных, ни переходить в эту папку. Все, что нужно знать о таком объекте, – это его ID и тип. Пользуясь этими данными, механизм автоматического поиска найдет этот объект, сделает текущей соответствующую папку, а по окончании выполнения команды вернет текущую папку назад
- Объекты с предопределенными ID доступны из любой папки. Какая бы папка ни была текущей, можно прямо из нее работать с таким объектом. Это существенно упрощает и ускоряет работу с объектами

Поэтому при создании объектов данных рекомендуется в первую очередь использовать предопределенные ID.

При инициализации токена в файловой системе создается следующая структура предопределенных папок:

3F00\0000\	SE-объекты
3F00\0000\0000\	Системные папки с файлами: GCHV-объекты и файлы идентификации
3F00\0000\0000\0001\	Файлы и папки для хранения объектов PKCS #11
3F00\0000\0000\0002\	Файлы и папки для хранения объектов Crypto API
3F00\0000\0001\	GOST-объекты

Папки, образующие эту структуру, существуют всегда; их невозможно удалить. В этих папках *не рекомендуется* создавать собственные файлы, а также DO не с предопределенными ID.

При появлении в будущем новых, поддерживаемых токеном, объектов данных в структуру могут быть добавлены новые предопределенные папки.

Безопасность в файловой системе ruToken

На данный момент система безопасности данных, хранящихся в памяти ruToken, обеспечивается системой атрибутов безопасности (**Security Attributes**) в сочетании с механизмом аутентификации владельца токена.

Атрибуты безопасности

Атрибуты безопасности назначаются любому объекту файловой системы ruToken на этапе его создания. С их помощью можно назначить различным операциям с данным объектом файловой системы предваряющие их security-ориентированные действия, успешное выполнение которых является необходимым условием выполнения самой операции.

Атрибут безопасности состоят из:

- Байта **режима доступа** (Access Mode байт), задающего список защищенных операций в виде битовой маски
- Массива **байт условий защиты**, определяющего набор security-ориентированных действий, которыми должна быть предварена каждая из защищенных операций
- **Таблицы объектов безопасности**, задающей ID объектов данных (DO) с эталонными данными и используемых для каждого из security-ориентированных действий перед выполнением каждой из защищенных операций

Атрибуты безопасности объекта файловой системы хранятся в его заголовке.

Таблица 1. Формат атрибутов безопасности

Смещение поля, байт	Длина поля, байт	Значение поля	Примечание
+00	1	Access Mode байт.	См. табл. 2.,4
+01	7	Байты условий защиты в соответствии с Access Mode байтом.	См. табл. 5
+08	32	Таблица объектов безопасности в соответствии с байтом условий защиты и Access Mode байтом.	См. табл. 6

Для файлов, папок и объектов данных существуют разные Access Mode байты, кодирующие специфические для файлов этих типов наборы операций. Для того чтобы защитить ту или иную операцию, нужно установить в 1 соответствующий ей бит в Access Mode байте.

Таблица 2. Формат Access Mode байта папки (MF/DF)

Номер и значение бита								На какие операции (команды APDU) распространяется защита
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
x	-	-	-	-	-	-	-	Не используется (должен быть 0)
-	1	-	-	-	-	-	-	DELETE FILE (саму папку)
-	-	0	-	-	-	-	-	Зарезервирован (должен быть 0)
-	-	-	0	-	-	-	-	Зарезервирован (должен быть 0)
-	-	-	-	0	-	-	-	Зарезервирован (должен быть 0)
-	-	-	-	-	0	-	-	Зарезервирован (должен быть 0)
								CREATE DO (PUT DATA) (внутри папки)
								CREATE FILE (внутри папки)

Таблица 3. Формат Access Mode байта файла (EF)

Номер и значение бита								На какие операции (команды APDU) распространяется защита
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
x	-	-	-	-	-	-		Не используется (должен быть 0)
-	1	-	-	-	-	-	-	DELETE FILE
-	-	0	-	-	-	-	-	Зарезервирован (должен быть 0)
-	-	-	0	-	-	-	-	Зарезервирован (должен быть 0)
-	-	-	-	0	-	-	-	Зарезервирован (должен быть 0)
-	-	-	-	-	0	-	-	Зарезервирован (должен быть 0)
-	-	-	-	-	-	1	-	UPDATE BINARY
-	-	-	-	-	-	-	1	READ BINARY

Таблица 4. Формат Access Mode байта объекта данных (DO)

Номер и значение бита								На какие операции (команды APDU) распространяется защита
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
GCHV-объекты								
x	-	-	-	-	-	-		Не используется (должен быть 0)
-	1	-	-	-	-	-	-	DELETE DO (PUT DATA)
-	-	0	-	-	-	-	-	Зарезервирован (должен быть 0)
-	-	-	0	-	-	-	-	Зарезервирован (должен быть 0)
-	-	-	-	0	-	-	-	Зарезервирован (должен быть 0)
-	-	-	-	-	1	-	-	USE (VERIFY)
-	-	-	-	-	-	1	-	UPDATE DO (CHANGE REFERENCE DATA)
-	-	-	-	-	-	-	1	UNBLOCK DO (RESET RETRY COUNTER)
не- GCHV-объекты								
x	-	-	-	-	-	-		Не используется (должен быть 0)
-	1	-	-	-	-	-	-	DELETE DO (PUT DATA)
-	-	0	-	-	-	-	-	Зарезервирован (должен быть 0)
-	-	-	0	-	-	-	-	Зарезервирован (должен быть 0)
-	-	-	-	0	-	-	-	Зарезервирован (должен быть 0)
-	-	-	-	-	1	-	-	USE (PSO, MSE)
-	-	-	-	-	-	0	-	Зарезервирован (должен быть 0)
-	-	-	-	-	-	-	0	Зарезервирован (должен быть 0)

Массив байтов условий защиты кодирует конкретные виды Security-ориентированных действий, которыми должны быть предварены соответствующие защищенные операции. Порядковый номер байта условий защиты для данной операции равен номеру бита для этой операции в Access Mode байте. Для операций, которым в Access Mode байте соответствует бит 0, значение байта условий защиты не актуально (рекомендуется задать 0x0).

Операция, которой соответствует байт условий защиты со значением 0x0, является незащищенной вне зависимости от значения соответствующего ей бита Access Mode байта.

Операция, которой соответствует байт условий защиты со значением 0xFF, не может быть применена ни при каких условиях.

Таблица 5. Формат одного байта условий защиты

Номер и значение бита						Условия применения операции к файлу / объекту
7	**6..4**	**3**	**2**	**1**	**0**	
-	000	**-**	**-**	x	x	Доступ к файлу / объекту:
-	000	-	-	0	0	Безусловный доступ
-	000	-	-	-	1	Аутентификация владельца токена (VERIFY)
0	000	0	0	0	0	Команда всегда применима к файлу / объекту
1	111	1	1	1	1	Команда никогда не применима к файлу / объекту

Таблица объектов безопасности представляет собой массив структур в соответствии с Access Mode байтом и соответствующими ему байтами условий защиты. Каждая 4-байтовая структура массива содержит список ID объектов данных, используемых при выполнении всех возможных Security-ориентированных действий, защищающих одну операцию. Порядковый номер элемента таблицы безопасности равен номеру бита для этой операции в Access Mode байте. Для операций, которым в Access Mode байте соответствует бит 0, значения полей соответствующего элемента таблицы безопасности не актуальны (рекомендуется задать 0x00..0x00).

Таблица 6. Формат элемента таблицы объектов безопасности

Смещение поля, байт	Длина поля, байт	Назначение поля	Примечание
+00	1	ID объекта, используемого при аутентификации владельца.	
+01	1	Зарезервирован (должен быть 0)	
+02	1	Зарезервирован (должен быть 0)	
+03	1	Зарезервирован (должен быть 0)	

Права доступа

На данный момент ruToken поддерживает 3 типа владельцев: **Гость**, **Пользователь** и **Администратор**, обладающих разным набором прав доступа. Установка текущих прав доступа произво-

дится в процессе аутентификации владельца в зависимости от того, какой объект данных был задействован при аутентификации. До выполнения первой аутентификации (либо после выполнения APDU-команды "RESET ACCESS RIGHTS" – "Сбросить права доступа") ruToken работает с правами доступа "Гость".

На данный момент ruToken содержит 2 объекта данных для аутентификации, предоставляющих соответствующие наборы прав:

ID объекта данных	Предоставляемые права доступа
0x1	Администратор (ADMIN)
0x2	Пользователь (USER)

Каждый из этих объектов данных содержит свой PIN-код (Администратора и Пользователя соответственно). Для того чтобы установить нужные права доступа, требуется выполнить команду аутентификации, указав ID соответствующего объекта данных и передав токену соответствующий PIN-код.

Владельцы имеют следующие права:

Тип владельца	Права владельца
Гость	Выполнение незащищенных операций.
Пользователь	Выполнение незащищенных операций. Выполнение операций, защищенных PIN-кодом Пользователя.
Администратор	Выполнение незащищенных операций. Выполнение операций, защищенных PIN-кодом Администратора. Изменение PIN-кодов Пользователя и Администратора. Разблокирование PIN-кода Пользователя. Инициализация памяти ruToken.

Окружение безопасности (Security Environment)

Окружение безопасности – это логическая структура, содержащая все необходимые настройки для выполнения токеном security-ориентированных команд и команд аутентификации. Структура состоит из компонент, каждая из которых отвечает за свой класс APDU-команд: шифрование, хеширование, аутентификацию.

Окружения безопасности можно создавать и сохранять в файловой памяти ruToken – для этой цели предназначены SE-объекты данных. Перед выполнением security-команд специальной APDU-командой нужное в данной момент окружение безопасности загружается в оперативную память токена. Такое окружение безопасности называется **текущим** (Current Security Environment, CSE).

Текущее окружение безопасности ruToken состоит из трех компонент, каждая из которых содержит ID объекта данных, используемого в своем классе APDU-команд:

Компонента CSE ruToken	**APDU-команды, использующие компоненту**
A-компонента	APDU-команды для аутентификации владельца токена (Login).
C-компонента	APDU-команды для шифрования.
H-компонента	APDU-команды для хеширования.

Использование окружений безопасности – это весьма удобно. Пользователю достаточно один раз создать несколько различных окружений безопасности для различных нужд, вместо того чтобы постоянно помнить, какие, например, ключи шифрования используются им в той или иной ситуации. А перед выполнением самих security-операций нужно всего лишь загрузить нужное окружение из соответствующего SE-объекта, сделав его текущим (security-ориентированные APDU-команды не требуют передачи им сведений об объектах данных – они берут эти сведения самостоятельно из соответствующей компоненты CSE).

Форматы объектов файловой системы ruToken

Ниже даны форматы заголовков объектов файловой системы ruToken. В колонке "Тег поля по ISO 7816" указаны теги ("маркеры") полей, которые используются при создании объектов файловой системы и при выдаче информации о них. Теги указаны в шестнадцатиричной системе счисления.

Заголовок папки

Таблица 7. Заголовок папки

Смещение поля, байт	Длина поля, байт	Значение поля по ISO 7816	Тег поля по ISO 7816	Примечание
+00	2	Общая длина, включая заголовок, в байтах	0x81	
+02	2	Длина тела в байтах	0x80	*У папки поле не используется.* Содержит 0x00
+04	2	Байт описания папки (ISO 7816-4, табл. 3), за которым следует байт кодирования данных (ISO 7816-4, табл. 86).	0x82	
+06	2	ID папки	0x83	
+08	1	Байт состояния цикла жизни папки (LCSInteger)	0x8A	
+09	12	Собственная информация	0xA5	
+21	3	Резерв	-	
+24	40	Атрибуты безопасности папки (см. табл. 1).	0x86	См. табл. 1

Заголовок файла

Таблица 8. Заголовок файла

Смещение поля, байт	Длина поля, байт	Значение поля по ISO 7816	Тег поля по ISO 7816	Примечание
+00	2	Общая длина, включая заголовок, в байтах	0x81	
+02	2	Длина тела в байтах	0x80	

Смещение поля, байт	Длина поля, байт	Значение поля по ISO 7816	Тег поля по ISO 7816	Примечание
+04	2	Байт описания файла (ISO 7816-4, табл. 3), за которым следует байт кодирования данных (ISO 7816-4, табл. 86).	0x82	
+06	2	ID файла	0x83	
+08	1	Байт состояния цикла жизни файла (LCSInteger)	0x8A	
+09	15	Резерв	-	
+24	40	Атрибуты безопасности файла (см. табл. 1).	0x86	См. табл. 1

Заголовок объекта данных

Таблица 9. Формат заголовка объекта данных

Смещение поля, байт	Длина поля, байт	Значение поля	Тег поля	Примечание
+00	2	Длина тела объекта данных в байтах	0x80	
+02	2	1 Byte: Тип объекта	0x83	См. табл. 10
		2 Byte: ID объекта		
+04	3	1 Byte: Опции объекта	0x85	См. табл. 10
		2 Byte: Флаги объекта		
		3 Byte: Максимальное : оставшееся число попыток доступа к объекту		
+07	1	Резерв	-	
+08	16	Резерв (для стандартного DO), ЛИБО тело объекта (для компактного DO)	0xA5	Тег 0xA5 *всегда* используется для передачи тела объекта данных

Смещение поля, байт	Длина поля, байт	Значение поля	Тег поля	Примечание
+24	40	Атрибуты безопасности объекта данных (см. табл. 1).	0x86	См. табл. 1

Таблица 10. Состав некоторых полей заголовка объекта данных

Смещ. байта	Значение байта	Номер бита								Описание
		7	6	5	4	3	2	1	0	
+02	Тип объекта					x	x	x	x	**Тип объекта:**
						0	0	0	0	SE-тип
						0	0	0	1	CHV-тип
						0	0	1	0	KEY-тип
		x	x	x	x					**Подтип объекта:**
		0	0	0	0	0	0	0	0	SE-объект SE-типа
		0	0	0	0	0	0	0	1	GCHV-объект CHV-типа
		0	0	0	0	0	0	1	0	GOST-объект KEY-типа
+03	ID объекта		x	x	x	x	x	x	x	**ID объекта**
		x								**Местоположение объекта:** 0 - Объект находится в предопределенной папке, 1 - Объект находится в текущей папке
+04	Опции объекта	x	x	x	x	x	x	x	x	**См. табл. 11.**

Смещ. байта	Значение байта	Номер бита								Описание
		7	6	5	4	3	2	1	0	
+05	Флаги объекта								x	**Местоположение тела объекта:** 0 – вне заголовка объекта (стандартный DO), 1 – внутри заголовка объекта (компактный DO)
							x	x		**Степень открытости объекта:**
							-	0		Закрытый DO
							0	1		DO с открытой длиной
							1	1		Открытый DO
		x	x	x	x	x				**Резерв** (должны содержать 0)
+06	Число попыток доступа к объекту					x	x	x	x	**Оставшееся число попыток доступа к объекту** (0..15)
		x	x	x	x					**Максимальное число попыток доступа к объекту** (1..15)

Таблица 11. Опции объекта данных (смещение +04 в заголовке объекта)

Номер бита								Описание
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
GCHV-объекты								
					x	**x**	**x**	**Права доступа, предоставляемые GCHV-объектом:**
					0	0	1	Администратор (ADMIN)
					0	1	0	Пользователь (USER)
x	**x**	**x**	**x**	**x**				**Резерв** (должны содержать 0)
GOST-объекты								
					x	**x**	**x**	**Тип шифрования, для которого применяется объект:**
					0	0	0	Простая замена
					0	0	1	Гаммирование
					0	1	0	Гаммирование с обратной связью
x	**x**	**x**	**x**	**x**				**Резерв** (должны содержать 0)
SE-объекты								
x	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**Резерв** (должны содержать 0)

Часть 3 Разработка приложений для ruToken

В этой главе описаны различные интерфейсы прикладного программирования, которые используются для разработки приложений для ruToken. Описания интерфейсов даны в соответствии с архитектурой ruToken – от низкоуровневых к высокоуровневым.

Интерфейсы низкого уровня

Низкоуровневые интерфейсы оперируют с TPDU T=0 по стандарту ISO 7816 и рассматривают ruToken как смарт-карту, вставленную в ридер. В операционных системах Windows существует Win32 API (**Microsoft SmartCard API**), при помощи которого можно обнаруживать смарт-карты и обмениваться с ними информацией. Разработчики могут использовать этот интерфейс для обмена с токеном на уровне TPDU T=0 без использования каких-либо библиотек из Комплекта разработчика ruToken.

Низкоуровневые интерфейсы обеспечивают приложениям доступ ко всем возможностям ruToken. Однако ничто не дается бесплатно. В данном случае расплачиваться приходится большим объемом программирования. Приложение само должно обнаруживать ридер с подключенным токеном, инициировать транзакции и т.д. Наиболее предпочтительной представляется следующий механизм доступа к токену посредством низкоуровневых API:

1. Прежде чем отправлять токену какие-либо команды, необходимо заблокировать ридер, к которому подключен токен.
2. Если необходимо, выполните процедуру аутентификации владельца токена (Login).
3. Выполните необходимые команды.
4. Если была выполнена аутентификация, выполните обратную процедуру (Logout).
5. Разблокируйте ридер.

Такая модель обеспечивает максимальную целостность данных и права доступа при обмене с токеном в случае конкурентного обращения приложений к токену. Однако при использовании низкоуровневых интерфейсов нужно постоянно помнить, что компоненты операционной системы и приложения взаимодействуют с токеном независимо.

Работа с токеном на уровне USB

Существует возможность обмена с ruToken напрямую, через интерфейс USB. Это может понадобиться в тех случаях, когда работа с токеном предполагается на этапе первоначальной загрузки, до загрузки операционной системы, либо при работе с операционными системами, отличными от Windows.

Важно

Производители ruToken не дают никаких гарантий относительно неизменности протоколов обмена через интерфейс USB, формата USB-пакетов, а также других деталей. Детали обмена через интерфейс USB, равно как и производимые изменения, не документируются в Руководствах.

Интерфейсы среднего уровня

В интерфейсах среднего уровня APDU преобразуются в формат TPDU T=0 (являющимися, по сути, теми же APDU, но специальным образом трансформированными для “транспортных” нужд).

APDU ruToken

Несмотря на то, что APDU являются базовым уровнем коммуникации с ruToken, вам вряд ли понадобится работать с ними напрямую – за исключением, быть может, каких-то неожиданных и экзотических случаев. Среди интерфейсов ruToken есть множество более удобных в использовании. Поэтому описание APDU дано в Руководстве скорее для ознакомления с тем “фундаментом”, на котором базируется архитектура ruToken. Вместе с тем, API ruToken включает и интерфейсы, позволяющие вам самостоятельно строить APDU.

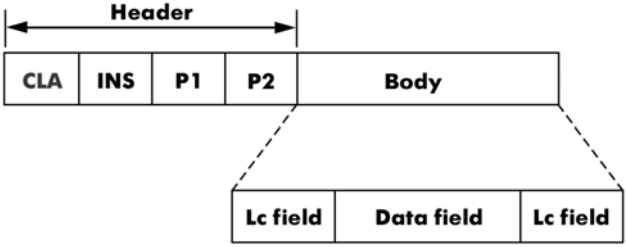
Структура APDU

APDU (Application Protocol Data Unit) – это командная последовательность, которая может быть отправлена приложением смарт-карте. В общем случае APDU может содержать в себе как команду, так и ответ.

Функциональные аспекты APDU, непосредственно используемые в разработке приложений, работающих со смарт-картами, описаны в стандарте ISO 7816-4. Этот стандарт определяет две основные области функциональности:

- Файловая система с иерархической структурой. Также определен набор функций, составляющий API, посредством которого приложения получают доступ к файлам и информации, хранящихся в файловой системе
- Доступ к файлам и информации, хранящейся в этих файлах, в файловой системе смарт-карты, может быть ограничен посредством ряда функций безопасности

Структура APDU-команды, определенная в ISO 7816-4 имеет сходное строение со структурой команды TPDU T=0, описанной в ISO 7816-3:



Заголовок APDU-команды состоит из следующих полей:

- CLA – класс команды
- INS – инструкция (команда)
- P1 и P2 – параметры команды.

Тело APDU-команды – это компонента переменной длины (и формы), которая служит для передачи информации как части команды, либо ответа:

- Lc Field – содержит размер передаваемого смарт-карте в качестве части команды блока данных (длина поля данных)
- Data Field – содержит данные, передаваемые смарт-карте для выполнения APDU-команды

- LE field – указывает количество байтов, которые должны быть возвращены APDU-ответом для данной APDU-команды

APDU-ответ в общем случае имеет следующее строение:

- Data Field – тело APDU-ответа переменной длины
- SW1|SW2 – байты статуса APDU (коды возврата)

Набор APDU ruToken

APDU	ISO 7816-	Описание
APDU для работы с файлами и папками		
CREATE FILE	9	Создать папку или файл в текущей папке
SELECT FILE	4	Выбрать (т.е. сделать текущим) папку или файл
READ BINARY	4	Прочитать текущий файл (или его часть)
UPDATE BINARY	4	Обновить (перезаписать) содержимое текущего файла (или его части)
DELETE FILE	9	Удалить файл в текущей или указанной папке, либо текущую или явно указанную папку
APDU для работы с объектами данных		
CREATE DO	расширение	Создать объект данных указанного типа
GENERATE KEY	расширение	Сгенерировать ключ шифрования и поместить его в создаваемый объект данных
GET DO INFO	расширение	Получить сведения об указанном объекте данных
DELETE DO	расширение	Удалить указанный объект данных
APDU для аутентификации и работы с PIN-кодами		
VERIFY	4	Сравнить данные, присланные в APDU-команде (PIN-код), с данными, хранящимися в памяти, и установить текущие права доступа
RESET ACCESS RIGHTS	расширение	Сбросить текущие права доступа
RESET RETRY COUNTER	8	Установить счетчик повторов в начальное значение
CHANGE REFERENCE DATA	8	Переписать заданные Reference Data (PIN-код)
Security-ориентированные APDU		
MANAGE SECURITY ENVIRONMENT	8	Создать Current Security Environment для последующего выполнения security-ориентированных действий:

APDU	ISO 7816-	Описание
- SET		- Установить (или заменить) заданную в APDU-команде компоненту Current Security Environment (CSE);
- RESTORE		- Заменить CSE на заданное SE, хранящееся в памяти ruToken (установить новое CSE)
PERFORM SECYRITY OPERATION **- HASH** **- ENCIPHER** **- DECIPHER**	8	Выполняет заданное security-ориентированное действие: - Вычислить и вернуть хеш-значение присланных данных; - Зашифровать присланные данные симметричным алгоритмом и вернуть зашифрованные данные - Расшифровать присланные данные симметричным алгоритмом и вернуть расшифрованные данные
GET CHALLENGE	4	Послать challenge (случайное число) для его дальнейшего использования в security-ориентированных APDU.
APDU для идентификации ruToken		
GET RUTOKEN ID	расширение	Получить уникальный серийный номер (ID) ruToken
GET RUTOKEN INFO	расширение	Получить идентификационные данные ruToken
RESET RUTOKEN	расширение	Выполнить сброс ruToken и вернуть ATR-строку
Сервисные APDU		
GET FREE MEMORY	расширение	Получить объем свободной файловой памяти ruToken
GET CURRENT FILE	расширение	Получить ID текущего файла, если он существует
Другие APDU (выступают как базовые для некоторых APDU расширения)		
GET DATA	4	Выдать данные. Может использоваться для получения версии ruToken, информации о текущем файле и т.п.
PUT DATA	4	Занести данные (ключ, номер алгоритма, список прав доступа и т.п.), заданные в APDU-команде

APDU для работы с файлами и папками

Создать файл / папку (APDU "CREATE FILE")

Создать в текущей папке ruToken файл или папку с заданными свойствами.

APDU-команда:

Название поля	Значение поля	Комментарий
CLA	0	Класс команды
INS	0xE0	APDU "CREATE FILE"
P1	0	
P2	0	
Lc	N+2	
DATA	0x62 (Tag, 1 байт)	FCP template по ISO 7816-4
	N (Length, 1 байт)	
	FCP Data (Value, N байт)	
Le	Пусто	Поле отсутствует

Описание используемых структур:

Поле DATA содержит TLV-структуру, представляющую собой FCP template по ISO 7816-4. В поле V (Value) содержится сама структура FCP.

Структура FCP (File Control Parameters) содержит информацию о создаваемом файле в виде набора TLV (Tag-Length-Value) структур. Любая TLV-структура состоит из 3-х полей: Tag (1 байт), Length (1 байт), Value (N байт, где N задано в поле Length). Таким образом, TLV-структура, описывающая 2-байтовый параметр, имеет длину 4 байта: 1 байт (Tag) + 1 байт (Length, содержащий значение 2) + 2 байта (Value).

Для создания файла используются следующие теги:

Значение в поле Tag (T)	Значение в поле Length (L)	Содержимое поля Value (V)
0x80	2	Длина тела файла (длина файла папки равна 0)
0x82	2	Дескриптор файла / папки (см. ниже)
0x83	2	ID, используемый в качестве имени файла / папки
0x86	40	Атрибуты безопасности. См. раздел "Безопасность в файловой системе ruToken"

Дескриптор файла:

В дескрипторе задается тип (файл или папка, младший байт) и режим кодирования данных при их чтении/записи (старший байт).

В текущей версии ruToken приняты следующие типы файлов:

- **Папка** (младший байт содержит 0x38)
- **Файл** (младший байт содержит 0x1)

Остальные значения младшего байта дескриптора не рассматриваются.

Старший байт дескриптора в текущей версии ruToken не используется (рекомендуется занести 0).

Описание команды:

APDU-команда создает в текущей папке новый файл или папку (тип создаваемого объекта задается младшим байтом дескриптора, тег 0x82). 2-байтовое имя файла / папки задается с помощью ID (тег 0x83), размер – тегом 0x80. Для папки должна быть задана нулевая длина.

В пределах одной папки невозможно создать файл и папку с одинаковыми ID.

После создания файла его размер нельзя изменить.

При успешном завершении команды созданный файл или папка становится текущим (к нему можно будет применять все команды, оперирующие с текущим файлом / папкой). При этом, если была создана папка, то текущий файл, возможно установленный ранее другими командами, сбрасывается.

Текущие права доступа должны соответствовать правам, заданным в поле CREATE FILE Access Mode байта текущей папки.

APDU-ответ:

Поле	Длина поля, байт	Содержимое поля
DATA OUT	Пусто	Поле отсутствует
SW1-SW2	2	Код возврата

Выбрать файл / папку (APDU "SELECT FILE")

Выбрать (т.е. сделать текущим) файл или папку.

APDU-команда:

Название поля	Значение поля	Комментарий
CLA	0	Класс команды
INS	0xA4	APDU "SELECT FILE"
P1	см. ниже	Основной режим выбора (см. ниже)
P2	см. ниже	Дополнительный режим выбора (см. ниже)
Lc	Пусто или длина входных данных	Зависит от значений P1-P2 (см. ниже)
DATA		В зависимости от значений P1-P2: Поле отсутствует, ID (имя файла или папки) Путь от Корневой папки (MF, Master File) Путь от текущей папки
Le	M	В ответе APDU ожидать TLV-структуру с FCP template (M байт)

Кодирование параметра P1:

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Означает:
0	0	0	0	0	0	x	x	Выбор по ID
0	0	0	0	0	0	0	0	Выбрать корневую папку (MF), папку или файл в текущей папке (в поле DATA – ID, или поля Lc и DATA отсутствуют)
0	0	0	0	0	0	0	1	Выбрать подпапку (в поле DATA – ID подпапки)
0	0	0	0	0	0	1	0	Выбрать файл в текущей папке (в поле DATA – ID файла)
0	0	0	0	0	0	1	1	Выбрать надпапку (родительскую папку) (поля Lc и DATA отсутствуют)

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Означает:
0	0	0	0	1	0	x	x	Выбор по пути
0	0	0	0	1	0	0	0	Выбрать по пути от корневой папки (т.е. от Master File, MF) (в поле DATA – путь от MF, не включая ID MF)
0	0	0	0	1	0	0	1	Выбрать по пути от текущей папки (в поле DATA – путь от текущей папки, не включая ID текущей папки)

Остальные значения P1 зарезервированы в ISO 7816 и /или не поддерживаются текущей версией ruToken.

Кодирование параметра P2:

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Означает:
0	0	0	0	-	-	x	x	Выбрать включение (актуально для P1=0)
0	0	0	0	-	-	0	0	Выбрать первое или конкретное включение
0	0	0	0	-	-	0	1	Выбрать последнее включение
0	0	0	0	-	-	1	0	Выбрать следующее включение
0	0	0	0	-	-	1	1	Выбрать предыдущее включение
0	0	0	0	x	x	-	-	Выбрать тип возвращаемой информации
0	0	0	0	0	1	-	-	Вернуть FCP-структуру с параметрами выбранного файла

Остальные значения P2 зарезервированы в ISO 7816 и /или не поддерживаются текущей версией ruToken.

Описание команды:

APDU-команда позволяет выбрать (сделать текущим) указанный файл или папку. Основной режим выбора задается параметром P1. Конкретизировать выбор позволяет параметр P2. В

зависимости от значений этих параметров может потребоваться передача дополнительных данных (ID, путь) в поле DATA.

Если P1 = 0, то, в зависимости от значения параметра P2 и содержимого поля DATA, будет выбран либо Master File (MF), либо папка или двоичный файл в текущей папке.

Если P1 задает выбор по пути, поле DATA должно содержать путь к нужной папке или файлу (путь представляет собой строку из идентификаторов папок *без разделителей* и *не включает* ID MF или текущей папки соответственно).

Биты bit1..bit0 параметра P2 позволяют конкретизировать параметры выбора, когда *P1 = 0*:

Значение bit1..bit0	**Содержимое поля DATA**	**Что будет выбрано**
00	Поля Lc и DATA отсутствуют	Первый встреченный в текущей папке файл или папка.
	ID	В текущей папке – файл или папка с указанным ID.
	0x3F00	Корневая папка (Master File, MF).
	0x3FFF	Будут выданы параметры текущей папки.
01	Поля Lc и DATA отсутствуют	Последний встреченный в текущей папке файл или папка.
10	ID	В текущей папке - следующий за указанным файл или папка.
11	ID	В текущей папке - предыдущий относительно указанного файл или папка.

Иные комбинации параметров недопустимы.

В случае успешного завершения команды файл (или папка) становится текущим. Если была выбрана папка, то текущий файл, возможно, установленный ранее другими командами, сбрасывается.

APDU-ответ:

Поле	**Длина поля, байт**	**Содержимое поля**
DATA OUT	M	FCP template
SW1-SW2	2	Код возврата

Если команда завершилась успешно, APDU-ответ содержит TLV-структуру с FCP template (Tag 0x62) + 2 байта кода возвра-

та. В поле Value структуры содержится структура FCP с параметрами выбранного файла. При этом в выходном буфере возвращается набор из следующих TLV-структур:

Значение в поле Tag (T)	Значение в поле Length (L)	Содержимое поля Value (V)
0x80	2	Длина тела файла (для файла папки возвращается 0).
0x81	2	Общая длина (длина заголовка + длина тела).
0x82	2	Дескриптор.
0x83	2	ID, используется в качестве имени файла / папки.
0x8A	1	LCSInteger (значение состояния жизненного цикла файла / папки)
0x86	40	Атрибуты безопасности.

Если в процессе выполнения команды произошла ошибка, APDU-ответ состоит из 2 байт кода возврата.

Считать из / записать в файл (APDU "READ BINARY" / "UPDATE BINARY")

Считать из текущего файла / записать в текущий файл заданное число байт, начиная с заданного смещения.

APDU-команда:

1. Для чтения из файла

Название поля	Значение поля	Комментарий
CLA	0	Класс команды
INS	0xB0	APDU "READ BINARY"
P1	Старший байт смещения	P1||P2 задают файловое смещение 1-го считываемого байта. Старший бит byP1 не должен содержать 1.
P2	Младший байт смещения	
Lc	Пусто	Поле отсутствует
DATA	Пусто	Поле отсутствует
Le	(0..255)	В APDU-ответе ожидать заданное число считанных байт (0 означает 256 байт)

2. Для записи в файл

Название поля	Значение поля	Комментарий
CLA	0	Класс команды
INS	0xD6	APDU "UPDATE BINARY"
P1	Старший байт смещения	byP1||byP2 задают файловое смещение 1-го байта для записи. Старший бит byP1 не должен содержать 1.
P2	Младший байт смещения	
Lc	0..255	Количество записываемых байт (0 означает 256 байт)
DATA		Данные для записи
Le	Пусто	Поле отсутствует

Описание команды:

APDU-команды позволяют считать из текущего файла или записать в текущий файл заданное число байт, начиная с заданного смещения. Файл устанавливается в качестве текущего командами "Создать файл" или "Выбрать файл".

Файловое смещение 1-го байта для чтения / записи задается параметрами P1 и P2 (старший байт смещения – P1). При этом старший бит смещения не должен содержать 1.

За один сеанс можно считать / записать до 256 байт. Т.к. размер файла после его создания нельзя изменить, чтение и запись возможны только в пределах существующего размера (т.е. файл невозможно увеличить / уменьшить командой записи). Задание неверного соотношения "смещение + длина" является ошибкой.

Для успешного выполнения APDU-команд требуется, чтобы текущие права доступа соответствовали правам, заданным в поле READ BINARY (для чтения из файла) или UPDATE BINARY (для записи в файл) Access Mode байта текущего файла.

APDU-ответ:

1. Для чтения из файла

Поле	Длина поля, байт	Содержимое поля
DATA OUT	1..256	Считанные данные
SW1-SW2	2	Код возврата

2. Для записи в файл

Поле	Длина поля, байт	Содержимое поля
DATA OUT	Пусто	Поле отсутствует
SW1-SW2	2	Код возврата

Если команда завершилась успешно, APDU-ответ содержит либо считанные данные + 2 байта кода возврата (для команды чтения), либо 2 байта кода возврата (для команды записи).

Если в процессе выполнения команды произошла ошибка, APDU-ответ содержит 2 байта кода возврата.

Удалить файл / папку (APDU "DELETE FILE")

Удалить текущий или указанный файл / папку.

APDU-команда:

Название поля	Значение поля	Комментарий
CLA	0	Класс команды
INS	0xE4	APDU "DELETE FILE"
P1	0	
P2	0	
Lc	Пусто либо 2	Поле отсутствует либо содержит длину поля DATA
DATA	Пусто либо ID	Поле DATA отсутствует либо содержит ID
Le	Пусто	Поле отсутствует

Описание команды:

APDU-команда позволяет удалить текущий или указанный в поле DATA файл или папку.

Если поля Lc и DATA отсутствуют, это означает удаление текущего файла или папки. При этом, если текущий файл существует, будет удален он, если нет – будет удалена текущая папка.

Если поля Lc и DATA присутствуют, то поле DATA содержит ID. В этом случае будет удален файл с этим ID.

Удаляемая папка должна быть пустой (не должна содержать вложенных объектов файловой системы, в т.ч. объектов данных).

Если удаляемый файл является текущим, после удаления текущий файл в системе отсутствует.

Если удаляемая папка является текущей, после удаления текущей становится родительская папка.

Текущие права доступа должны соответствовать правам, заданным в поле DELETE FILE Access Mode байта удаляемой папки или файла.

APDU-ответ:

Поле	Длина поля, байт	Содержимое поля
DATA OUT	Пусто	Поле отсутствует
SW1-SW2	2	Код возврата

APDU для работы с объектами данных (DO)

Создать объект данных (DO) (APDU "CREATE DO")

Создать DO указанного типа.

APDU-команда:

Название поля	Значение поля	Комментарий
CLA	0	Класс команды
INS	0xDA	APDU "PUT DATA"
P1	1	
P2	0x62	Подкоманда "Создать DO"
Lc	N	
DATA	N байт	Входной буфер с набором TLV-структур (см. ниже)
Le	Пусто	Поле отсутствует

Описание команды:

APDU-команда позволяет создать DO указанного формата в текущей или предопределенной папке. Папка, в которой создается DO (текущая или предопределенная), выбирается в соответствии со значением старшего бита его ID (см. табл. 10).

Для создания DO используются следующие TLV-структуры:

Значение в поле Tag (T)	Значение в поле Length (L)	Содержимое поля Value (V)
0x80	2	Общая длина тела DO (см. ниже).
0x83	2	Тип DO + ID DO (см. табл. 10)
0x85	3	Опции DO + флаги DO + число попыток доступа к DO (см. табл. 10)
0x86	40	Атрибуты безопасности. См. раздел "Безопасность в файловой системе ruToken"
0xA5	N	Тело DO. Для компактных DO тело имеет длину 1..16 байт (т.е. N=1..16).

Тег 0x80 задает *общую длину тела DO*. Все поддерживаемые текущей версией ruToken типы DO имеют тело, полностью помещающееся в поле DATA вместе с остальными описанными выше TLV-структурами. Поэтому общая длина сейчас должна быть равна N – значению в поле L TLV-структуры с тегом 0xA5. В будущем, с появлением поддержки "больших" DO – например, асимметричных ключей RSA большой длины – их тело при создании DO надо будет передавать по частям, в несколько итераций. И тогда тег 0x80 будет задавать общую длину тела DO, а значение N (тег 0xA5) – длину *части* тела DO, передаваемой в данной итерации.

Во флагах DO (2-й байт, кодируемый тегом 0x85) задается формат DO (стандартный или компактный), а также степень открытости DO. Для вашего удобства, флаг "Компактный DO" будет установлен автоматически при создании DO с общей длиной тела, подходящей для компактных DO (не более 16 байт).

Число попыток доступа (3-й байт, кодируемый тегом 0x85) для SE- и GOST-объектов игнорируется; рекомендуется задавать значение 0.

Форматы атрибутов безопасности для DO идентичны атрибутам для папок и файлов.

В текущей версии ruToken создавать новые GCHV-объекты *невозможно*.

Таким образом, в текущей версии ruToken при помощи этой команды можно:

- Создать SE-объект (DO любой степени открытости, компактного формата, общая длина тела DO – 6 байт)
- *Импортировать* ключ шифрования ГОСТ в создаваемый GOST-объект (закрытый DO или DO с открытой длиной, *только* стандартного формата, тело DO содержит ключ шифрования ГОСТ, общая длина тела DO – 32 байта)

Формат тела создаваемого **SE-объекта** (тег 0xA5):

Смещение поля	**Длина поля, байт**	**Значение в поле**	**Описание поля**
+00	1	ID или 0	ID объекта данных для A-компоненты текущего SE
+01	1	ID или 0	ID объекта данных для C-компоненты текущего SE

Смещение поля	Длина поля, байт	Значение в поле	Описание поля
+02	1	ID или 0	ID объекта данных для H-компоненты текущего SE
+03	3	0..0	Резерв

Поля, соответствующие не используемым в данном окружении безопасности компонентам, должны содержать 0. Например, при создании SE-объекта, который будет задавать окружение безопасности только для аутентификации пользователя и для шифрования, надо задать ID только для A- и C-компонент, а в поле для H-компоненты надо занести 0.

Текущие права доступа должны соответствовать правам, заданным в поле CREATE DO Access Mode байта текущей папки.

APDU-ответ:

Поле	Длина поля, байт	Содержимое поля
DATA OUT	Пусто	Поле отсутствует
SW1-SW2	2	Код возврата

Сгенерировать ключ шифрования (APDU "GENERATE KEY")

Сгенерировать ключ шифрования и поместить его в создаваемый DO.

APDU-команда:

Название поля	Значение поля	Комментарий
CLA	0	Класс команды
INS	0xDA	APDU "PUT DATA"
P1	1	
P2	0x65	Подкоманда "Сгенерировать ключ шифрования"
Lc	N	
DATA	N байт	Входной буфер с набором TLV-структур (см. ниже)
Le	Пусто	Поле отсутствует

Описание команды:

В отличие от APDU-команды создания DO, данная APDU-команда позволяет *сгенерировать* ключ шифрования внутри токена и поместить его в создаваемый DO, а не импортировать

ключ шифрования извне. DO создается в текущей или предопределенной папке – в зависимости от значения старшего бита его ID (см. табл. 10).

В текущей версии ruToken можно генерировать только ключи шифрования ГОСТ.

Используются следующие TLV-структуры:

Значение в поле Tag (T)	Значение в поле Length (L)	Содержимое поля Value (V)
0x80	2	Общая длина тела DO (32 байта).
0x83	2	Тип DO + ID DO (см. табл. 10).
0x85	3	Опции DO + флаги DO + число попыток доступа к DO (см. табл. 10).
0x86	40	Атрибуты безопасности. См. раздел "Безопасность в файловой системе ruToken".

В отличие от предыдущей APDU-команды, TLV-структура с тегом 0xA5 (Тело DO) не передается.

Длина тела DO (тег 0x80) фиксирована и составляет 32 байта.

Во флагах DO (2-й байт, кодируемый тегом 0x85) должен быть задан закрытый DO или DO с открытой длиной, стандартного формата.

Число попыток доступа (3-й байт, кодируемый тегом 0x85) игнорируется, рекомендуется задавать значение 0.

В будущих версиях ruToken, с увеличением числа аппаратно поддерживаемых алгоритмов шифрования, при помощи этой команды можно будет генерировать ключи разных типов и длин.

Текущие права доступа должны соответствовать правам, заданным в поле CREATE DO Access Mode байта текущей папки.

APDU-ответ:

Поле	Длина поля, байт	Содержимое поля
DATA OUT	Пусто	Поле отсутствует
SW1-SW2	2	Код возврата

Получить сведения об объекте данных (DO) (APDU "GET DO INFO")

Получить набор сведений об указанном DO.

APDU-команда:

Название поля	Значение поля	Комментарий
CLA	0x80	Класс команды
INS	0x30	Команда
P1	0x0	Основной режим выбора
P2	См. ниже	Дополнительный режим выбора (см. ниже)
Lc	Пусто или 1	В зависимости от значения P2 – поле отсутствует или содержит 1
DATA	Пусто или ID	В зависимости от значения P2 – поле отсутствует или содержит ID объекта данных
Le	256	Ожидать набор TLV-структур, содержащих сведения о DO

APDU-команда позволяет получить сведения об объекте данных, находящемся *в текущей папке.*

Параметр P2 позволяет конкретизировать параметры выбора DO, информацию о котором нужно получить:

Значение в P2	Содержимое поля DATA	Сведения о каком DO будут выданы
0x0	Поля Lc и DATA отсутствуют	О первом DO, созданном в текущей папке
	ID	О DO с ID, указанным в поле DATA
0x2	ID	О DO, следующем за указанным в поле DATA

Иные комбинации параметров недопустимы.

APDU-ответ:

Поле	Длина поля, байт	Содержимое поля
DATA OUT	M	Набор TLV-структур, содержащих сведения об объекте данных
SW1-SW2	2	Код возврата

Если команда завершилась успешно, в APDU-ответ содержит набор TLV-структур со сведениями о DO + 2 байта кода возврата. Набор состоит из следующих TLV-структур:

Значение в поле Tag (T)	Значение в поле Length (L)	Содержимое поля Value (V)	Обязательная TLV-структура?	Комментарий
0x80	2	Длина тела DO	Нет	Только для открытых DO и DO с открытой длиной тела
0x83	2	Тип DO	Да	
		ID DO		
0x85	3	Опции DO	Да	
		Флаги DO		
		MAX / оставшееся число попыток доступа к DO		
0x86	40	Атрибуты безопасности DO	Да	
0xA5	M	Тело DO	Нет	Только для открытых DO

Не все TLV-структуры из этого набора являются обязательными. Так, структуры с тегами 0x80 и 0xA5 могут присутствовать, только если это разрешает степень открытости DO, задаваемая в его флагах на этапе создания DO (см. табл. 10).

Если в процессе выполнения команды произошла ошибка, APDU-ответ состоит из 2 байт кода возврата.

Удалить объект данных (DO) (APDU "DELETE DO")

Удалить указанный объект данных.

APDU-команда:

Название поля	Значение поля	Комментарий
CLA	0	Класс команды
INS	0xDA	APDU "PUT DATA"
P1	1	
P2	0x64	Подкоманда "Удалить DO"
Lc	1	
DATA	ID DO	ID удаляемого DO
Le	Пусто	Поле отсутствует

Описание команды:

Команда позволяет удалить *в текущей папке* DO с указанным ID.

Текущие права доступа должны соответствовать правам, заданным в поле DELETE DO Access Mode байта удаляемого объекта данных.

APDU-ответ:

Поле	Длина поля, байт	Содержимое поля
DATA OUT	Пусто	Поле отсутствует
SW1-SW2	2	Код возврата

APDU для аутентификации и работы с PIN-кодами

Выполнить аутентификацию владельца токена (APDU "VERIFY")

APDU-команда позволяет выполнить аутентификацию владельца токена путем проверки переданного PIN-кода. В случае успешной аутентификации в токене устанавливаются текущие права доступа, ассоциированные с данным PIN-кодом.

В другом режиме работы APDU-команда позволяет проверить, была ли уже проведена аутентификация с этими правами доступа, либо получить оставшееся количество попыток аутентификации с этими правами доступа.

Входные параметры:

Название поля	Значение поля	Комментарий
CLA	0	Класс команды
INS	0x20	APDU "VERIFY"
P1	0	
P2	ID или 0	ID – ID объекта данных (DO), участвующего в аутентификации; 0 – для идентификации используется DO из A-компоненты текущего окружения безопасности (CSE).
Lc	Пусто или N	Поле отсутствует либо содержит длину поля DATA
DATA	Пусто или PIN-код	Поле отсутствует либо содержит N-байтовый PIN-код
Le	Пусто	Поле отсутствует

Описание команды:

APDU-команда позволяет:

- Произвести аутентификацию владельца токена с установкой соответствующих прав доступа.
- Проверить, была ли уже проведена аутентификация с данными правами доступа или получить оставшееся количество попыток аутентификации с данными правами доступа.

Режим работы APDU-команды (аутентификация или проверка) определяется наличием или отсутствием полей Lc и DATA.

Если поля Lc и DATA присутствуют, будет выполнена аутентификация. В этом случае поле DATA содержит N-байтовый PIN-код владельца токена.

Если поля Lc и DATA отсутствуют:

- Если аутентификация с данными правами доступа уже была произведена ранее, будет возвращен код 0x9000
- В противном случае будет возвращен код 0x63CX, где X – оставшееся количество попыток аутентификации с данными правами доступа.

В поле P2 передается ID GCHV-объекта, который должен использоваться при выполнении APDU-команды. Каждый объект данных содержит эталонный PIN-код владельца токена и соответствующие ему права доступа. На данный момент ruToken содержит 2 таких GCHV-объекта:

ID объекта данных	Права доступа
0x1	Администратор (ADMIN)
0x2	Пользователь (USER)

Таким образом, если требуется получить, например, права Администратора, нужно выполнить аутентификацию с указанием в P2 0x1 и передачей токену PIN-кода Администратора.

Если P2 = 0, токен будет использовать DO, указанный в A-компоненте CSE (текущего окружения безопасности, устанавливаемого с помощью APDU “MSE”).

Аутентификация владельца возможна только в том случае, когда токен работает с правами доступа “Гость”. В любом другом случае будет возвращена ошибка 0x6F86.

В случае предъявления неверного PIN-кода будет уменьшен на 1 счетчик попыток доступа к соответствующему GCHV-объекту данных. Если счетчик достигнет 0, объект данных блокируется, а последующие аутентификации становятся невозможными. Для разблокирования объекта данных служит APDU “RESET RETRY COUNTER”.

В случае успешной проверки токеном переданного PIN-кода соответствующие права доступа будут установлены как текущие, а счетчик попыток доступа к GCHV-объекту будет установлен в максимальное значение. С этого момента владелец токена может выполнять все операции, защищенные данными правами.

Текущие права доступа должны соответствовать правам, заданным в поле USE Access Mode байта соответствующего GCHV-объекта.

APDU-ответ:

Поле	Длина поля, байт	Содержимое поля
DATA OUT	Пусто	Поле отсутствует
SW1-SW2	2	Код возврата

Очистить текущие права доступа (APDU “RESET ACCESS RIGHTS”)

APDU-команда выполняет сброс текущих прав доступа (т.е. текущими становятся права доступа “Гость”). После выполнения команды владелец токена сможет выполнять только те операции, для которых не требуется аутентификация.

APDU-команда:

Название поля	Значение поля	Комментарий
CLA	0x80	Класс команды
INS	0x40	Команда
P1	0	
P2	0	
Lc	Пусто	Поле отсутствует
DATA	Пусто	Поле отсутствует
Le	Пусто	Поле отсутствует

APDU-ответ:

Поле	Длина поля, байт	Содержимое поля
DATA OUT	Пусто	Поле отсутствует
SW1-SW2	2	Код возврата

Восстановить значение счетчика попыток доступа к GCHV-объекту (APDU "RESET RETRY COUNTER")

Команда позволяет установить счетчик попыток доступа к GCHV-объекту в его исходное, максимальное значение (т.е. разблокировать объект данных).

APDU-команда:

Название поля	Значение поля	Комментарий
CLA	0x0	Класс команды
INS	0x2C	APDU "RESET RETRY COUNTER"
P1	3	
P2	ID	ID разблокируемого DO
Lc	Пусто	Поле отсутствует.
DATA	Пусто	Поле отсутствует.
Le	Пусто	Поле отсутствует.

Описание команды:

В действительности APDU-команда может разблокировать только объект, предоставляющий права доступа "Пользователь" (ID = 0x2). *Объект 0x1 (права доступа "Администратор") не подлежит разблокированию.*

Текущие права доступа должны соответствовать правам, заданным в поле UNBLOCK DO Access Mode байта соответствующего GCHV-объекта.

APDU-ответ:

Поле	Длина поля, байт	Содержимое поля
DATA OUT	Пусто	Поле отсутствует
SW1-SW2	2	Код возврата

Изменить PIN-код владельца токена (APDU "CHANGE REFERENCE DATA")

APDU-команда позволяет сменить PIN-код Пользователя или Администратора.

APDU-команда:

Название поля	Значение поля	Комментарий
CLA	0x0	Класс команды
INS	0x24	APDU "CHANGE REFERENCE DATA"

Название поля	Значение поля	Комментарий
P1	1	
P2	ID	ID GCHV-объекта, содержащего изменяемый PIN-код.
Lc	1..16	Длина нового PIN-кода.
DATA	PIN-код	Новый PIN-код.
Le	Пусто	Поле отсутствует.

Описание команды:

Параметр P2 содержит ID GCHV-объекта с PIN-кодом, который требуется изменить. Поле DATA содержит *новый PIN-код.*

Текущие права доступа должны соответствовать правам, заданным в поле UPDATE DO Access Mode байта соответствующего GCHV-объекта.

APDU-ответ:

Поле	Длина поля, байт	Содержимое поля
DATA OUT	Пусто	Поле отсутствует
SW1-SW2	2	Код возврата

APDU для Security-ориентированных операций

Создать текущее окружение безопасности (APDU "Manage Security Environment", "MSE")

APDU-команда позволяет создать в оперативной памяти токена текущее окружение безопасности (Current Security Environment, CSE) – структуру данных, содержащую сведения об объектах данных, используемых в различных security-ориентированных операциях.

APDU-команда:

Название поля	Значение поля	Комментарий
CLA	0x0	Класс команды
INS	0x22	APDU "MSE"
P1	3 или 1	Режим работы команды (см. ниже)
P2	ID или Тег компоненты CSE	Зависит от режима работы (см. ниже)
Lc	Пусто или 3	Зависит от режима работы (см. ниже).
DATA	Пусто или Шаблон компоненты CSE	Зависит от режима работы (см. ниже).
Le	Пусто	Поле отсутствует.

Описание команды:

Команда позволяет создать CSE двумя способами:

- Загрузкой всех компонент CSE из указанного SE-объекта (режим RESTORE)
- Прямой установкой компоненты CSE (режим SET)

Режим работы команды кодируется полем P1.

Наличие и вид остальных входных параметров зависит от значения поля P1 следующим образом:

Режим работы	P1	P2	Lc	DATA
RESTORE	0x3	ID	Отсутствует	Отсутствует
SET	0x1	Тег компоненты CSE	3	Шаблон компоненты CSE

Режим работы RESTORE:

В режиме RESTORE все компоненты текущего окружения безопасности загружаются из SE-объекта, ID которого указано в поле P2. SE-объект должен быть предварительно создан APDU-командой "CREATE DO" в предопределенной или текущей папке (местоположение зависит от значения старшего бита ID объекта).

Поля Lc и DATA в этом режиме работы отсутствуют.

Текущие права доступа должны соответствовать правам, заданным в поле USE Access Mode байта данного SE-объекта.

Режим работы SET:

В режиме SET производится прямая установка указанного компонента CSE. Информация о том, какая компонента CSE должна быть установлена, содержится в поле P2, а устанавливаемое значение в виде TLV-структуры – в поле DATA:

Компонента CSE	Тэг компоненты (в P2)	Шаблон компоненты (в DATA)		
		T	L	V
A-компонента	0xA4	0x83	1	ID GCHV-объекта
C-компонента	0xB8	0x83	1	ID GOST-объекта
H-компонента	0xAA	0x83	1	ID GOST-объекта

В режиме SET может быть установлена только одна компонента.

Общие сведения:

В процессе выполнения APDU-команды токен проверяет наличие объектов данных, составляющих CSE, в предопределенной или в текущей папке (местоположение DO зависит от значения старшего бита его ID). Если все объекты данных найдены, информация о них заносится в CSE токена. С этого момента становится возможным выполнение security-ориентированных команд, оперирующих с компонентами CSE.

Если какая-либо из компонент CSE ссылается на DO с ID 0x80 и больше, токен сможет оперировать с этой компонентой, только если в текущей папке в данный момент присутствует DO данного типа и с данным ID.

Если при создании CSE указать в качестве ID объекта данных 0x00, это приведет к *сбросу* соответствующей компоненты CSE. С этого момента выполнение security-ориентированных команд, оперирующих с этой компонентой CSE, станет невозможным.

APDU-ответ:

Поле	Длина поля, байт	Содержимое поля
DATA OUT	Пусто	Поле отсутствует
SW1-SW2	2	Код возврата

Выполнить security-операцию (APDU "Perform Security Operation", "PSO")

APDU-команда позволяет выполнить определенную security-операцию: шифрование, вычисление контрольной суммы, hash-значения, цифровой подписи и т.п.

Текущая версия ruToken позволяет выполнять 3 security-операции:

- Зашифрование по ГОСТ 28147-89
- Расшифрование по ГОСТ 28147-89
- Вычисление hash-значения (имитовставки по ГОСТ 28147-89)

Все security-операции производятся на ключах шифрования ГОСТ, содержащихся в предварительно созданных GOST-объектах данных. Перед выполнением операции при помощи APDU-команды "MSE" должно быть создано текущее окружение безопасности (CSE), соответствующие компоненты которого ссылаются на используемые в операциях GOST-объекты.

Одна APDU-команда может обработать не более чем 256 байт данных. Для обработки большего объема данных следует организовать цепочку APDU (т.е. вызывать APDU-команду и получать APDU-ответ в цикле). При этом должен быть соответствующим образом установлен класс APDU-команды (поле CLA):

- 0x10 - если данная APDU-команда не является последней в цепочке APDU
- 0x00 – если данная APDU-команда завершает цепочку APDU

До завершения выполнения цепочки APDU невозможно выполнить никакую другую APDU-команду.

Текущие права доступа должны соответствовать правам, заданным в поле USE Access Mode байта GOST-объекта.

1. Зашифрование по ГОСТ 28147-89 (APDU "PSO: ENCIPHER")

APDU-команда:

Название поля	Значение поля	Комментарий
CLA	0x00	Класс команды (для единственной APDU-команды или последней команды в цепочке APDU)
	0x10	Класс команды (для не последней APDU-команды в цепочке APDU)
INS	0x2A	APDU "PSO"
P1	0x86	Тип выходных данных – криптограмма (зашифрованные данные), предваренная padding indicator байтом.
P2	0x80	Тип входных данных – не зашифрованные данные.
Lc	N	Длина входных данных.
DATA	Входные данные	Данные для шифрования.
Le	M	Длина выходных данных, ожидаемых в APDU-ответе.

Описание команды:

APDU-команда позволят зашифровать данные по алгоритму ГОСТ 28147-89.

Зашифрование производится на ключе, содержащемся в предварительно созданном GOST-объекте данных. Ссылка на этот

объект данных должна быть задана в C-компоненте текущего окружения безопасности (CSE).

Режим зашифрования – простая замена, гаммирование или гаммирование с обратной связью – выбирается автоматически в соответствии с опциями GOST-объекта данных (см. табл. 11).

Число байт для зашифрования должно быть *кратным 8.*

Дополнительно для режима простой замены:

Если выполняется единственная APDU-команда (цепочка APDU не организуется), либо выполняется первая APDU-команда цепочки, в поле DATA OUT APDU-ответа возвращается:

- Padding indicator байт (значение 0x00, согласно ISO 7816-8 и ISO 7816-4)
- Зашифрованные данные

В остальных командах цепочки поле DATA OUT APDU-ответа содержит только зашифрованные данные.

Таким образом:

Если выполняется единственная APDU-команда либо первая команда цепочки APDU:

- Командой можно зашифровать не более (256-8) байт, т.е. N = 8,16,..,(256-8), кратное 8
- В поле Le – значение N+1 (M = N+1, т.е. + padding indicator байт)

Если выполняются остальные команды цепочки APDU:

- Командой можно зашифровать до 256 байт включительно, т.е. N = 0,8,16,..,(256-8), кратное 8 (0 – для зашифрования 256 байт)
- В поле Le – значение N (M = N)

Дополнительно для остальных режимов шифрования:

Если выполняется единственная APDU-команда (цепочка APDU не организуется), либо выполняется первая APDU-команда цепочки – в первые 8 байт поля DATA должна быть помещена синхропосылка. В остальных APDU-командах цепочки поле DATA содержит только данные для зашифрования.

Если выполняется единственная APDU-команда либо первая команда цепочки APDU, в поле DATA OUT APDU-ответа возвращается:

- Padding indicator байт (значение 0x00, согласно ISO 7816-8 и ISO 7816-4)
- 8-байтовая синхропосылка
- Зашифрованные данные

В остальных командах цепочки поле DATA OUT APDU-ответа содержит только зашифрованные данные.

Таким образом:

Если выполняется единственная APDU-команда либо первая команда цепочки APDU:

- Командой можно зашифровать не более (256-16) байт, т.е. N = 8,16,..,(256-16), кратное 8
- В поле Le – значение N+1 (M = N+1, т.е. + padding indicator байт)

Если выполняются остальные команды цепочки APDU:

- Командой можно зашифровать до 256 байт включительно, т.е. N = 0,8,16,..,(256-8), кратное 8 (0 – для зашифрования 256 байт)
- В поле Le – значение N (M = N)

APDU-ответ:

Поле	Длина поля, байт	Содержимое поля
DATA OUT	M	Выходные данные
SW1-SW2	2	Код возврата

2. Расшифрование по ГОСТ 28147-89 (APDU "PSO: DECIPHER")

APDU-команда:

Название поля	Значение поля	Комментарий
CLA	0x00	Класс команды (для единственной APDU-команды или последней команды в цепочке APDU)
	0x10	Класс команды (для не последней APDU-команды в цепочке APDU)
INS	0x2A	APDU "PSO"
P1	0x80	Тип выходных данных – не зашифрованные данные.
P2	0x86	Тип входных данных –

Название поля	Значение поля	Комментарий
		криптограмма (зашифрованные данные), предваренная padding indicator байтом.
Lc	N	Длина входных данных.
DATA	Входные данные	Данные для расшифрования.
Le	M	Длина выходных данных, ожидаемых в APDU-ответе.

Описание команды:

APDU-команда позволят расшифровать данные по алгоритму ГОСТ 28147-89. Расшифрование производится на ключе, содержащемся в предварительно созданном GOST-объекте данных. Ссылка на этот объект данных должна быть задана в C-компоненте текущего окружения безопасности (CSE).

Режим расшифрования – простая замена, гаммирование или гаммирование с обратной связью – выбирается автоматически в соответствии с опциями GOST-объекта данных (см. табл. 11).

Число байт для расшифрования должно быть *кратным 8.*

Дополнительно для режима простой замены:

Если выполняется единственная APDU-команда (цепочка APDU не организуется), либо выполняется первая APDU-команда цепочки, в поле DATA передаются:

- Padding indicator байт (значение 0x00, согласно ISO 7816-8 и ISO 7816-4)
- Данные для расшифрования

В остальных командах цепочки поле DATA содержит только данные для расшифрования.

Поле DATA OUT APDU-ответа во всех случаях содержит только расшифрованные данные (т.е. при расшифровании padding indicator байт теряется).

Таким образом:

Если выполняется единственная APDU-команда либо первая команда цепочки APDU:

- В поле Lc – значение N = 1+n (padding indicator байт + число байт для расшифрования)
- Командой можно расшифровать не более (256-8) байт, т.е. n = 8,16,..,(256-8), кратное 8

- В поле Le – значение n (M = n)

Если выполняются остальные команды цепочки APDU:

- Командой можно расшифровать до 256 байт включительно, т.е. N = 0,8,16,..,(256-8), кратное 8 (0 – для расшифрования 256 байт)
- В поле Le – значение N (M = N)

Дополнительно для остальных режимов шифрования:

Если выполняется единственная APDU-команда либо первая команда цепочки APDU, в поле DATA передаются:

- Padding indicator байт (значение 0x00, согласно ISO 7816-8 и ISO 7816-4)
- 8-байтовая синхропосылка
- Данные для расшифрования

В остальных APDU-командах цепочки поле DATA содержит только данные для расшифрования.

Поле DATA OUT APDU-ответа во всех случаях содержит только расшифрованные данные (т.е. при расшифровании синхропосылка и padding indicator байт теряются).

Таким образом:

Если выполняется единственная APDU-команда либо первая команда цепочки APDU:

- В поле Lc – значение N = 1+8+n (padding indicator байт + синхропосылка + число байт для расшифрования)
- Командой можно расшифровать не более (256-16) байт, т.е. n = 8,16,..,(256-16), кратное 8
- В поле Le – значение n (M = n)

Если выполняются остальные команды цепочки APDU:

- Командой можно расшифровать до 256 байт включительно, т.е. N = 0,8,16,..,(256-8), кратное 8 (0 – для расшифрования 256 байт)
- В поле Le – значение N (M = N)

APDU-ответ:

Поле	Длина поля, байт	Содержимое поля
DATA OUT	M	Расшифрованные данные
SW1-SW2	2	Код возврата

3. Вычисление имитовставки по ГОСТ 28147-89 (APDU "PSO: HASH")

APDU-команда:

Название поля	Значение поля	Комментарий
CLA	0x00	Класс команды (для единственной APDU-команды или последней команды в цепочке APDU)
	0x10	Класс команды (для не последней APDU-команды в цепочке APDU)
INS	0x2A	APDU "PSO"
P1	0x90	Тип выходных данных – hash-значение.
P2	0x80	Тип входных данных – не зашифрованные данные.
Lc	0..255	Длина входных данных (0 – для 256 байт).
DATA	Входные данные	Данные для подсчета имитовставки.
Le	4	В APDU-ответе ожидать 4-байтовую имитовставку (для единственной APDU-команды или последней команды в цепочке APDU).
	Пусто	Поле отсутствует (для не последней команды в цепочке APDU)

Описание команды:

APDU-команда позволят выполнить функцию хеширования входных данных. В текущей версии ruToken в качестве функции хеширования используется функция вычисления имитовставки по ГОСТ 28147-89. Вычисление производится на ключе, содержащемся в предварительно созданном GOST-объекте данных. Ссылка на этот объект данных должна быть задана в H-компоненте текущего окружения безопасности (CSE).

При организации цепочки APDU поля Le и DATA OUT присутствуют только в последней APDU-команде цепочки; в остальных командах цепочки поле Le игнорируется, а поле DATA OUT - отсутствует.

APDU-ответ:

Поле	Длина поля, байт	Содержимое поля
DATA OUT	4	Имитовставка (для единственной APDU-команды или последней команды в цепочке APDU).
	Пусто	Поле отсутствует (для не последней команды в цепочке APDU).
SW1-SW2	2	Код возврата

Получить случайное число (APDU "GET CHALLENGE")

APDU-команда позволяет получить из токена случайное число размером 32 байта.

APDU-команда:

Название поля	Значение поля	Комментарий
CLA	0	Класс команды
INS	0x84	APDU "GET CHALLENGE"
P1	0	
P2	0	
Lc	Пусто	Поле отсутствует.
DATA	Пусто	Поле отсутствует.
Le	32	Ожидать 32-байтовое случайное число в APDU-ответе

APDU-ответ:

Поле	Длина поля, байт	Содержимое поля
DATA OUT	32	Случайное число
SW1-SW2	2	Код возврата

Если APDU-команда завершилась успешно, APDU-ответ содержит 32-байтовое случайное число + 2 байта кода возврата.

Если в процессе выполнения APDU-команды произошла ошибка, APDU-ответ содержит 2 байта кода возврата.

APDU для идентификации ruToken

Получить ruToken ID (серийный номер) (APDU "GET RUTOKEN ID")

APDU-команда позволяет получить 4-байтовый уникальный серийный номер (ID) ruToken.

APDU-команда:

Название поля	Значение поля	Комментарий
CLA	0	Класс команды
INS	0xCA	APDU "GET DATA"
P1	1	
P2	0x81	Подкоманда "Получить ruToken ID"
Lc	Пусто	Поле отсутствует.
DATA	Пусто	Поле отсутствует.
Le	4	Ожидать 4 байта ruToken ID в APDU-ответе.

APDU-ответ:

Поле	Длина поля, байт	Содержимое поля
DATA OUT	4	ruToken ID
SW1-SW2	2	Код возврата

Если APDU-команда завершилась успешно, APDU-ответ содержит ruToken ID + 2 байта кода возврата.

Если в процессе выполнения APDU-команды произошла ошибка, APDU-ответ содержит 2 байта кода возврата.

Получить идентификационные данные ruToken (APDU "GET RUTOKEN INFO")

APDU-команда позволяет получить набор параметров, идентифицирующих данный ruToken: его тип, аппаратную версию, общий объем EEPROM-памяти, №протокола обмена с токеном,№ микропрограммы, прошитой в токен, №заказа токена.

APDU-команда:

Название поля	Значение поля	Комментарий
CLA	0	Класс команды
INS	0xCA	APDU "GET DATA"
P1	1	
P2	0x89	Подкоманда "Получить идентифи-

		идентификационные данные ruToken"
Lc	Пусто	Поле отсутствует.
DATA	Пусто	Поле отсутствует.
Le	8	Ожидать 8 байт идентификационных данных в APDU-ответе.

APDU-ответ:

Поле	Длина поля, байт	Содержимое поля
DATA OUT	8	Идентификационные данные ruToken
SW1-SW2	2	Код возврата

Если APDU-команда завершилась успешно, APDU-ответ содержит структуру с идентификационными данными ruToken + 2 байта кода возврата. Структура имеет следующий вид:

Если APDU-команда завершилась успешно, APDU-ответ содержит структуру с идентификационными данными ruToken + 2 байта кода возврата. Структура имеет следующий вид:

Смещение поля, байт	Длина поля, байт	Содержимое поля
+00	1	Тип ruToken (текущий тип – 0).
+01	1	Аппаратная версия ruToken (4 бита maj: 4 бита min).
+02	1	Общий объем EEPROM-памяти ruToken в *8-килобайтовых* единицах (0 означает 256 единиц).
+03	1	№ протокола обмена с ruToken.
+04	1	№ микропрограммы, прошитой в ruToken.
+05	1	№ заказа ruToken.
+06	2	Резерв

Если в процессе выполнения APDU-команды произошла ошибка, APDU-ответ содержит 2 байта кода возврата.

Выполнить сброс токена и получить ATR-строку (APDU "RESET RUTOKEN")

APDU-команда позволяет выполнить "горячий" сброс токена в его начальное состояние. В APDU-ответе возвращается ATR-строка в формате ISO 7816-3, предназначенная для идентификации токена стандартными средствами.

Состояние токена после "горячего" сброса:

- Текущая папка – MF.

- Текущий EF отсутствует.
- Все компоненты CSE сброшены.
- Текущие права доступа – “Гость”.

APDU-команда:

Название поля	Значение поля	Комментарий
CLA	0x80	Класс команды
INS	0x0	Команда
P1	0	
P2	0	
Lc	Пусто	Поле отсутствует.
DATA	Пусто	Поле отсутствует.
Le	19	Ожидать 19-байтовую ATR-строку в APDU-ответе

ATR-строка ruToken:

3B 6F 00 FF 00 56 72 75 54 6F 6B 6E 73 30 20 00 00 90 00

APDU-ответ:

Поле	Длина поля, байт	Содержимое поля
DATA OUT	19	ATR-строка ruToken
SW1-SW2	2	Код возврата

Если APDU-команда завершилась успешно, APDU-ответ содержит ATR-строку в формате ISO 7816-3 + 2 байта кода возврата.

Если в процессе выполнения APDU-команды произошла ошибка, APDU-ответ содержит 2 байта кода возврата.

Сервисные APDU

Получить объем свободной файловой памяти ruToken (APDU “GET FREE MEMORY”)

APDU возвращает объем свободной файловой памяти (т.е. суммарную длину всех свободных секторов памяти).

APDU-команда:

Название поля	Значение поля	Комментарий
CLA	0	Класс команды
INS	0xCA	APDU “GET DATA”
P1	1	
P2	0x8A	Подкоманда “Получить объем свободной файловой памяти ”

Название поля	Значение поля	Комментарий
Lc	Пусто	Поле отсутствует.
DATA	Пусто	Поле отсутствует.
Le	4	Ожидать 4-байтовое значение свободной памяти в APDU-ответе.

APDU-ответ:

Поле	Длина поля, байт	Содержимое поля
DATA OUT	4	Значение свободной памяти
SW1-SW2	2	Код возврата

Если APDU-команда завершилась успешно, APDU-ответ содержит 4-байтовое значение свободной памяти в ruToken + 2 байта кода возврата.

Если в процессе выполнения APDU-команды произошла ошибка, APDU-ответ содержит 2 байта кода возврата.

Получить ID текущего файла (APDU "GET CURRENT FILE")

APDU возвращает ID текущего файла, если он существует. Если текущего файла в системе нет, будет возвращен соответствующий код возврата.

APDU-команда:

Название поля	Значение поля	Комментарий
CLA	0	Класс команды
INS	0xCA	APDU "GET DATA"
P1	1	
P2	0x11	Подкоманда "Получить ID текущего EF"
Lc	Пусто	Поле отсутствует.
DATA	Пусто	Поле отсутствует.
Le	2	Ожидать 2-байтовый ID текущего EF

APDU-ответ:

Поле	Длина поля, байт	Содержимое поля
DATA OUT	2	ID текущего EF
SW1-SW2	2	Код возврата

Если APDU-команда завершилась успешно, APDU-ответ содержит 2-байтовый ID текущего EF + 2 байта кода возврата.

Если в процессе выполнения APDU-команды произошла ошибка, APDU-ответ содержит 2 байта кода возврата.

Коды возврата

Любой APDU-ответ обязательно содержит 2 байта, в которые занесены коды возврата. Коды возврата всегда заносятся в 2 последних байта APDU-ответа.

В терминах ISO 7816 коды возврата обозначаются как SW1‖SW2. Код SW1 заносится в предпоследний (более младший) байт APDU-ответа, код SW2 – в последний байт.

Код SW1 задает класс ошибки, код SW2 позволяет конкретизировать данную ошибку в пределах класса.

ruToken может возвращать следующие сочетания кодов SW1‖SW2:

SW1	Смысл SW1	SW2	Смысл SW2
0x90	Команда выполнена успешно	**0x00**	Нет дополнительной информации
0x63	Неудачная аутентификация	**0x00**	Нет дополнительной информации
		0xCX	X – количество оставшихся попыток доступа к данному DO
0x64	Состояние памяти ruToken не изменилось (неожиданный конец файла / DO или ошибка перед началом записи в память)	**0x00**	Нет дополнительной информации
0x65	Состояние памяти ruToken изменилось (ошибка в процессе записи в память)	**0x00**	Нет дополнительной информации
		0x81	Сбой памяти (memory failure)
0x67	Неверная длина входного и/или выходного буфера (т.е. Lc и/или Le)	**0x00**	Нет дополнительной информации
0x68	Ошибка, связанная с информацией в байте CLA	**0x83**	Ожидается завершающая команда цепочки (выдается при попытке выполнить другую команду до завершения цепочки команд)
0x69	Команда не позволена (command not allowed)	**0x82**	Права доступа неудовлетворительны для выполнения команды (нет прав для выполнения команды)
		0x83	DO заблокирован
		0x85	Непригодные условия выполнения команды (например, попытка удаления непустой папки)
		0x86	Нет текущего EF
0x6A	Неверные параметры команды	**0x80**	Неверные параметры (данные) во входном буфере (в поле DATA)
		0x81	Затребованная функция этой команды не поддерживается

SW1	Смысл SW1	SW2	Смысл SW2
		0x82	Файл / DO не найден
		0x84	Недостаточно места в EEPROM-памяти токена
		0x86	Неверные параметры команды (P1-P2)
		0x89	Файл / DO уже существует
0x6B	Заданное смещение – за границей двоичного файла	**0x00**	Нет дополнительной информации
0x6C	Неверная длина выходного буфера (Le)	**0xXX**	XX – рекомендуемая точная длина выходного буфера
0x6D	Неверная или неподдерживаемая команда	**0x00**	Нет дополнительной информации
0x6F	Собственный класс ошибки	**0x01**	ruToken имеет протокол обмена, не поддерживаемый USB-драйвером (более новый, чем в драйвере)
		0x83	Обнаружено нарушение протокола обмена с ruToken.
		0x84	ruToken занят обработкой другой команды
		0x85	В данной папке уже создано максимальное количество объектов файловой системы. Создание нового объекта невозможно.
		0x86	Токен работает не с правами доступа "Гость". Перед выполнением аутентификации владельца требуется очистить права доступа при помощи APDU "RESET ACCESS RIGHTS".

Список кодов возврата может быть дополнен в будущих версиях ruToken.

rtAPI. Библиотека ядра ruToken

Библиотека ядра ruToken rtAPI – это сервисная библиотека классов C++, предназначенная для облегчения разработки приложений, использующих ruToken. Она является надстройкой над SmartCard API и предоставляет классы, облегчающие работу с ruToken на уровне APDU. Высокоуровневые компоненты архитектуры ruToken – Сервис-провайдер, библиотека PKCS #11 – а также утилиты для работы с ruToken прямо или косвенно используют классы rtAPI.

Библиотека ядра существует в динамическом (.dll-модуль, COFF-формат) и в статическом (.lib-модуль, COFF- и OMF-форматы) вариантах, являющихся эквивалентными. Статическую бибилиотеку рекомендуется использовать только в том случае, если особенности используемых вами языков программирования и средств разработки не позволяют работать с dll-модулями COFF-формата.

Комплект разработчика содержит следующие файлы, относящиеся к библиотеке ядра:

Drivers\rtAPI.dll	Динамическая библиотека ядра (COFF-формат).
Lib\rtAPI\rtAPI.lib	Библиотека импорта для rtAPI.dll.
Lib\rtAPI\Static\rtAPIs.lib	Статическая библиотека ядра (COFF-формат).
Lib\rtAPI\Static\rtAPIso.lib	Статическая библиотека ядра (OMF-формат).
Drivers\SetupDrv.exe	Инсталлятор динамической библиотеки ядра.

Установка библиотеки ядра ruToken

Динамическая библиотека ядра rtAPI.dll устанавливается в системную папку Windows и удаляется из нее утилитой SetupDrv.exe вместе в драйверами ruToken (см. "Установка драйверов" раздела "Драйверы ruToken"), что гарантирует ее постоянное присутствие в системе вне зависимости от того, какие из высокоуровневых компонент архитектуры ruToken используются вами для работы с токеном.

В процессе инсталляции Комплекта разработчика динамическая библиотека ядра rtAPI.dll устанавливается в систему автоматически.

Ниже даны описания классов rtAPI, сгруппированные в соответствии с файлами объявлений.

BASIC.H

Назначение файла: Содержит основные макроопределения для использования в библиотеке, а также объявления общеупотребимых типов.

RTAPI_ASSERT – Макрос, который пользователь может определить как ему удобнее для отладочных сообщений. В inline-функциях rtAPI он используется довольно часто.

SCAPI.H

Назначение файла: Содержит классы, "оборачивающие" в себя Microsoft SmartCard API.

class Contextable;

Описание:

Является базовым классом для всех классов, так или иначе использующих Resource Manager Context (SCARDCONTEXT). Хранит хендл на SCARDCONTEXT. Также запрещает копирование и присвоение для всех наследуемых классов.

Методы:

SCARDCONTEXT GetContext() const; - возвращает хранимый хендл на SCARDCONTEXT.

class Context : class Contextable;

Описание:

Реализует понятие Resource Manager контекста (**SCARDCONTEXT**) и функции по его управлению в виде класса. rtAPI не вынуждает пользователя обязательно создавать объект класса **Context**, однако если он используется, то перед тем, как работать с ruToken, пользователь должен создать объект типа **Context** и вызывать его метод **Establish**. Причем продолжительность жизни этого объекта должна быть больше, чем остальных, использующих его, т.к. в деструкторе класса **Context** вызывается метод Release, который закрывает хранимый хендл **SCARDCONTEXT**.

Методы:

Context(); - Конструктор. Строит объект, инициализируя хранимый хендл на **SCARDCONTEXT** значением **NULL**. Для продолжения работы нужно инициализировать данный объект корректным хендлом, вызвав метод **Establish** или **Attach.**

~Context(); - Деструктор. Уничтожает объект, вызвав предварительно Release, если хранимый хендл не равен NULL.

void Attach(SCARDCONTEXT hCtx) – присоединяет хендл **hCtx** к данному объекту, предварительно вызвав **Release** для старого хендла, если он не равен **NULL**.

SCARDCONTEXT Detach() – отсоединяет хранимый хендл от данного объекта, возвращая его значение. После выполнения данной операции хендл, который хранит объект, становится равным **NULL**. Т.е. **GetContext()** вернет **NULL**.

LONG Establish(DWORD dwScope =SCARD_SCOPE_USER) – Вызывает **SCardEstablishContext** для хранимого хендла (передавая последним параметром его адрес). Возвращает то, что вернет **SCardEstablishContext**.

LONG Release() - Вызывает **SCardReleaseContext** для хранимого хендла. Возвращает то, что вернет **SCardReleaseContext**. При успешном выполнении данной функции, значение хранимого хендла становится равным **NULL**.

operator SCARDCONTEXT () const; - возвращает хранимый хендл. Аналог **GetContext();**

template <class Tp> class AutoAllocPtr : public Contextable;

Описание:

Шаблонный класс-"обертка" вокруг указателя, для которого была выделена память в любой **SCardXXX** функции с помощью **SCARD_AUTOALLOCATE**.

Tp – шаблонный параметр указатель. Т.е., например, **AutoAllocPtr<LPBYTE>**, а не **AutoAllocPtr<BYTE>**.

Класс использует стратегию "захват ресурса" (подобно **std::auto_ptr**), т.е. при копировании, присвоении объектов данного класса происходит передача владения указателем.

Методы:

explicit AutoAllocPtr(SCARDCONTEXT hCtx) – Конструктор. Создает неинициализированный объект с хранимым указателем, равным **NULL**.

~AutoAllocPtr() – Деструктор. Уничтожает объект, предварительно вызвав **Reset()**.

void Grab(Tp p, DWORD dwLen) throw() – Вызывает **Reset()**, а затем присоединяет указатель **p** и длину **dwLen** к объекту, т. е. **GetPtr()** вернет **p**, а GetLength() вернет **dwLen**.

void Reset() throw() – уничтожает указатель с помощью **SCardFreeMemory** и устанавливает хранимую длину равной **SCARD_AUTOALLOCATE**.

Tp Release() throw() – возвращает хранимый указатель, теряя право владения им. Т.е. **GetPtr()** вернет **NULL**, а **GetLength()** - **SCARD_AUTOALLOCATE**.

Tp GetPtr() const – возвращает хранимый указатель.

DWORD GetLength() const – возвращает длину буфера, на который указывает хранимый указатель. Для неинициализированного (сброшенного) объекта (**GetPtr() = NULL**) возвращаемый результат не имеет значения, за исключением некоторых ситуаций (см. **AutoAllocPtr<LPTSTR> Database::ListReaders(...)**)

class Database : class Contextable;

Описание:

Дает доступ к database-функциям Resource Manager. Большая часть методов этого класса здесь не описывается, поскольку они являются аналогами соответствующих SCard-функций, где вместо хендла на SCARDCONTEXT передается хендл, хранимый объектом класса Database, и для каждой функции, возвращающей буфер с данными, есть функция-двойник, возвращающая этот буфер через объект **AutoAllocPtr**, причем хендлы на **SCARDCONTEXT** у Database-объекта и AutoAllocPtr-объекта должны быть равны.

Методы (только те, что заслуживают внимания):

explicit Database(SCARDCONTEXT hCtx); - Конструктор. Строит объект, инициализируя хранимый хендл на **SCARDCONTEXT** значением **hCtx**.

AutoAllocPtr<LPTSTR> ListReaders(LPCTSTR pmszGroups = NULL); - возвращает список ридеров, имеющихся в системе через AutoAllocPtr-указатель. Если **SCardListReaders** завершилась неудачно, то **AutoAllocPtr::GetPtr()** вернет **NULL**,

AutoAllocPtr::GetLength() – код ошибки возвращаемый **SCardListReaders**.

LONG MakeRutokenList () – делает список (снимок) имен ридеров, в которых на момент вызова этой функции находились ruToken. Доступ к этим именам в последствии можно получить с помощью функции **Database::GetNextRutoken**. В случае успеха возвращает **SCARD_S_SUCCESS**, в противном – код ошибки, который вернул Resource Manager при вызове одной из его функций.

LPCTSTR GetNextRutoken (DWORD& dwIndex**)** – возвращает имя ридера с индексом **dwIndex** в списке ридеров, в которых на момент успешного вызова функции **MakeRutokenList** находились ruToken и в случае успеха увеличивает **dwIndex** на единицу. Функция возвращает указатель на 0-терминированную строку с именем ридера или NULL если **dwIndex** >= количеству записей в списке. Если функция вернет NULL, то dwIndex не увеличивается на 1, а остается прежним. Перед тем как вызывать эту функцию надо вызвать функцию **MakeRutokenList**

```
Context ctx;
// ….
Database db(ctx);
LONG lResult = db.MakeRutokenList();
if (SCARD_S_SUCCESS == lResult)
{
LPCTSTR pszReader = NULL;
for (DWORD dwI = 0; pszReader =
db.GetNextRutoken(dwI); )
{
            // что-то делаем
}
}
```

class Card : class Contextable;

Описание:

Реализует понятие Смарт-Карты, как это представляет Resource Manager (**SCARDHANDLE**), и функции по ее управлению в виде класса. Перед тем как посылать команды APDU на ruToken, нужно создать объект класса **Card**. В дальнейшем, для ссылки на

хранимый хендл на **SCARDHANDLE** будет употребляться идентификатор **hCard_**. Это и есть хранимый private-хендл.

Методы:

explicit Card(SCARDCONTEXT hCtx) – Конструктор. Строит объект, инициализируя хранимый хендл на **SCARDCONTEXT** значением **hCtx**, а хранимый хендл на **SCARDHANDLE** – значением **NULL**. **hCtx** в дальнейшем будет использоваться для подключения к ruToken. Для продолжения работы нужно инициализировать данный объект корректным хендлом смарт-карты (**SCARDHANDLE**), вызвав метод **Connect** или **Attach**.

~Card(); – Деструктор. Уничтожает объект, предварительно вызвав **Disconnect,** если хранимый **SCARDHANDLE** не равен **NULL**.

void Attach(SCARDHANDLE hCard) – присоединяет хендл **hCard** к данному объекту, предварительно вызвав **Disconnect** для старого хендла, если он не равен **NULL**.

SCARDHANDLE Detach() – отсоединяет хранимый **SCARDHANDLE** от данного объекта, возвращая его значение. После выполнения данной операции хендл, который хранит объект, становится равным **NULL**. Т.е. **operator SCARDHANDLE()** вернет **NULL**.

LONG Connect(LPCTSTR pszReader, DWORD dwShareMode =SCARD_SHARE_SHARED, DWORD dwPrefProts =SCARD_PROTOCOL_T0 | SCARD_PROTOCOL_T1, LPDWORD pdwActiveProt =NULL) - Вызывает **SCardConnect** для хранимого **SCARDHANDLE**, передавая предпоследним параметром его адрес. Возвращает то, что вернет **SCardConnect.**

LONG Disconnect() – Вызывает **SCardDisconnect** для хранимого хендла (**SCARDHANDLE**). Возвращает то, что вернет **SCardDisconnect**. При успешном выполнении данной функции (если результат равен **SCARD_S_SUCCESS**), значения хранимого **SCARDHANDLE** становится равным **NULL**.

LONG Reconnect(DWORD dwShareMode =SCARD_SHARE_SHARED, DWORD dwPrefProts =SCARD_PROTOCOL_T0 | SCARD_PROTOCOL_T1, DWORD

dwInit =SCARD_LEAVE_CARD, LPDWORD pdwActiveProt =NULL) – Вызывает **SCardReconnect** для хранимого **SCARDHANDLE**, передавая его первым параметром. Возвращает то, что вернет **SCardReconnect**.

LONG BeginTransaction() – Вызывает SCardBeginTransaction(hCard_). Возвращает то, что вернет **SCardBeginTransaction.**

LONG EndTransaction (DWORD dwDisposition =SCARD_LEAVE_CARD) – Вызывает **SCardEndTransaction(hCard_, dwDisposition**. Возвращает то, что вернет **SCardEndTransaction**.

LONG Lock(DWORD dwTimeout =INFINITE) – Позволяет получить монопольный доступ к ruToken. Монопольный доступ основывается на захвате ruToken мьютекса, который создает библиотека rtAPI, и последующем вызове **BeginTransaction. dwTimeout** указывает сколько времени пытаться получить монопольный доступ, если мьютекс уже захвачен. Возвращает ***SCARD_S_SUCCESS,*** если монопольный доступ был получен; ***SCARD_E_TIMEOUT*** - если указанное время **dwTimeout** истекло; в противном случае вернет код ошибки.

LONG Unlock(DWORD dwDisposition =SCARD_LEAVE_CARD) – Позволяет снять монопольный доступ с ruToken. Снятие монопольного доступа основывается на освобождении ruToken мьютекса и последующем вызове **EndTransaction(dwDisposition**). Возвращает код обозначающий успешное/неуспешное выполнение функции.

LONG Status(...) const; – Аналог **SCardStatus** (см. scapi.h и MSDN).

LONG Status(AutoAllocPtr<LPTSTR>& aapReaderName, LPDWORD pdwState, LPDWORD pdwProtocol, AutoAllocPtr<LPBYTE>& aapAtr) const – Аналог вышеуказанной функции, только имя ридера и ATR-строка возвращаются через указатели **AutoAllocPt**.

LONG Transmit(LPCBYTE pInBuf, DWORD cbInLen, LPBYTE pOutBuf, LPDWORD pcbOutLen) – Аналог **ScardTransmit** (см. scapi.h и MSDN).

pInBuf – указатель на входной буфер

cbInBuf – длина в байтах входного буфера

pOutBuf – указатель на выходной буфер

pcbOutLen – указатель на длину выходного буфера

LONG GetAttrib(DWORD dwAttr, LPBYTE pbAttr, LPDWORD pcbAttrLen) const – Аналог **SCardGetAttrib** (см. scapi.h и MSDN).

LONG GetAttrib(DWORD dwAttr, AutoAllocPtr<LPBYTE>& aapAttr) const – Аналог вышеуказанной функции, только значение атрибута возвращается через указатель **AutoAllocPtr**.

LONG GetAtr(AutoAllocPtr<LPBYTE>& aapAtr) – Возвращает ATR-строку. Вызывает **GetAttrib(SCARD_ATTR_ATR_STRING, aapAtr);**

LONG Control(...) const – Аналог **SCardControl** (см. scapi.h и MSDN).

LONG Control(DWORD dwControlCode, LPCVOID pInBuffer, DWORD nInBufferSize, AutoAllocPtr<LPVOID>& aapOutBuffer, LPDWORD pBytesReturned) const – Аналог вышеуказанной функции, только выходной буфер возвращается через указатель **AutoAllocPtr**.

operator SCARDHANDLE () const; – возвращает хранимый **SCARDHANDLE хендл (hCard_).**

DWORD GetActiveProtocol() const; – возвращает активный протокол, который использует смарт-карта. Этот протокол возвращается также и через последний параметр функции **SCardConnect** или через 5-й параметр функции **ScardStatus**

APDUBASIC.H

Назначение файла: Содержит базовые классы для APDU.

class ApduBasic;

Описание:

Базовый класс для всех APDU-классов. Определяет базовый интерфейс для доступа к разным частям APDU и отправкой его на смарт-карту. Напрямую объекты класса ApduBasic создать нельзя.

Методы:

virtual LONG Send(Card& card) =0; – виртуальный метод, который переопределяется в наследуемых классах. Отправляет APDU-команду на смарт-карту card и возвращает результат в виде кода ошибки **Resource Manager** (т.е. результат вызова **card.Transmit(...)**)

LPCBYTE GetCommand() – Возвращает указатель на буфер APDU-команды.

DWORD GetCommandLen() – Возвращает длину буфера APDU-команды.

LPCBYTE GetReply() – Возвращает указатель на буфер APDU-ответа. Следует вызывать только после вызова Send.

DWORD GetReplyLen() – Возвращает длину буфер APDU-ответа, исключая 2 байта слова состояния (SW).

WORDGetReplySW() – Возвращает слово состояния (код возврата, SW) выполненной APDU-команды.

APDUS.H

Назначение файла: Содержит класс, описывающий короткие APDU, а также наследуемые от него классы APDU, специфичных для ruToken (часть из них в прямом смысле специфична, основная же часть описывается стандартом ISO 7816).

class ApduS : public ApduBasic;

Описание:

Базовый класс для всех классов коротких APDU. Реализует всю необходимую функциональность для преобразования коротких APDU в TPDU T=0 и отправки их на смарт-карту с помощью замещения метода Send, унаследованного от ApduBasic. ApduS гарантирует инвариантность буфера APDU перед и после отправки на смарт-карту (т.к. преобразование в TPDU сказывается на APDU).

Методы:

ApduS(BYTE byCla, BYTE byIns, BYTE byP1, BYTE byP2, LPCBYTE pLc =0, LPCBYTE pData =0, LPCBYTE pLe =0); - Конструктор. Строит короткий APDU из входных параметров:

byCla – CLA байт

byIns – INS байт

byP1 – P1 байт

byP2 – P2 байт

pLc – указатель на значение поля **Lc**. Если **pLc = NULL** – поле **Lc** отсутствует.

pData – указатель на буфер данных APDU-команды (т.е. поле **DATA**). Его длина должна быть равной ***pLc**.

pLe – указатель на значение поля **Le**. Если **pLe = NULL** – поле **Lc** отсутствует.

Примечание

Если *pLc и/или *pLe равен 0 – имеется в виду максимальная длина (256 байт).

void Rebuild(BYTE byCla, BYTE byIns, BYTE byP1, BYTE byP2, LPCBYTE pLc =0, LPCBYTE pData =0, LPCBYTE pLe =0). – Перестраивает данный APDU заново. См. конструктор для описания входных параметров.

```
enum {
None =0x00,
Admin=0x01, //ID GCHV-объекта
// с правами Администратора
User =0x02, //ID GCHV-объекта
// с правами Пользователя
};
```

Определение ID GCHV-объектов.

Важно

Далее будут обсуждаться классы, относящиеся к безопасности в ruToken. **НАСТОЯТЕЛЬНО** рекомендуется прочитать раздел "Безопасность в файловой системе ruToken".

class Sot;

Описание:

Реализует Таблицу объектов безопасности (Security Object Table, SOT) (См. Таблицу 6.).

Внутренние константы:

```
enum {
ixAuth      =0,
ixExtauth   =1,
// В текущей версии ruToken не используется
ixEnctraf   =2,
// В текущей версии ruToken не используется
ixSigtraf   =3,
// В текущей версии ruToken не используется
};
```

Индексы соответствующих записей в таблице. **ixAuth**, **ixExtauth**, **ixEnctraf**, **ixSigtraf** – соответственно индексы ID объектов данных для аутентификации, EXTERNAL AUTHENTICATE, шифрования трафика, подписывания трафика. В текущей версии ruToken используется только ID объекта данных для аутентификации.

Методы:

explicit Sot(BYTE byID0 =None, BYTE byID1 =None, BYTE byID2 =None, BYTE byID3 =None); - Строит таблицу из **byID0**, **byID1**, и т.д., записывая их по индексам в порядке возрастания, начиная с нуля.

static SizeType Size(); – размер таблицы (всегда одинаков и пока равен 4).

BYTE operator[] (SizeType ix) const;

BYTE& operator[] (SizeType ix); - операторы обращения к ID GCHV-объекта (записи в таблице) по индексу **ix**.

class Scb;

Описание:

Реализует Байт условий защиты (Security Condition Byte, SCB) (См. Табл. 5). Помимо хранения самого SCB (private член - **bySCb_**), хранит и реализует доступ к ассоциированной с ним

Таблицей объектов безопасности (Security Object Table, SOT) (**class Sot**) (private член - **assocSot_**). Под "пустым" состоянием объекта класса Scb понимается: **byScb_ = 0 (Always)**, и все записи в **assocSot_** равны 0.

Внутренние константы:

```
enum {
Always              =0x00,
Never       =0xff,
Auth        =0x01,
};
```

Возможные значения SCB-байта (**byScb_**):

- Always – операция всегда доступна
- Never – операция никогда не доступна

Auth – для выполнения операции требуется аутентификация DO с ID, который записан в ассоциированной с этим SCB-байтом SOT-таблице (т.е. в **assocSot_**) по смещению **Sot::ixAuth (0x00)**.

Методы:

Scb(); - Конструктор. Вызывает **Reset()**. См. описание метода Reset.

explicit Scb(BYTE byCond); - Конструктор. Вызывает **Set(byCond)**; См. описание метода **Set(BYTE)**.

Scb(BYTE byCond, const Sot& sot) - Конструктор. Вызывает **Set(byCond, sot)**; См. описание метода **Set(BYTE, const Sot&)**.

BYTE Get() – Возвращает значение SCB-байта (**byScb_**).

const Sot& GetAssocSot() const

Sot& GetAssocSot() – Возвращают ассоциированную SOT. Как видно, второй метод возвращает ее через неконстантную ссылку, что позволяет пользователю манипулировать этой таблицей.

void Set(BYTE byCond) – Меняет значение **byScb_** на **byCond**, не изменяя **assocSot_**.

void Set(BYTE byCond, const Sot& sot) – меняет значение **byScb_** на **byCond**, а **assocSot_** на **sot**.

void Reset(); - возвращает SCB-объект в "пустое" состояние (см. Описание класса)

operator BYTE() const – Возвращает **Get()**.

class SecAttrs;

Описание:

Реализует Security Attributes (атрибуты безопасности, см. раздел "Безопасность в файловой системе ruToken"). Хранит Access Mode байт и вектор из 7 объектов SCB (Security Condition Byte, байт условий защиты). С каждым битом в Access Mode байте ассоциируется объект SCB, причем если данный бит не установлен, то объекта SCB нет; если же бит установлен, то можно получить доступ к соответствующему SCB-объекту и производить над ним манипуляции. Под "пустым" состоянием объекта класса **SecAttrs** понимается, что Access Mode байт равен нулю, т.е. никакой защиты нет.

Методы:

SecAttrs(); - Конструктор. Вызывает **Reset()**. См. описание метода Reset.

explicit SecAttrs(SaType st); - Конструктор. Вызывает **AssumeType(st);** См. описание метода **Assume(BYTE)**, а для определения **SaType** см. следующее описание после этого класса.

void Reset(); - Возвращает объект в пустое состояние (см. Описание класса).

static SizeType Count(); - Возвращает количество (возможно) ассоциированных объектов SCB. В текущей версии ruToken – 7.

BYTE GetAmByte() const; - Возвращает значение хранимого Access Mode байта.

Scb* operator[] (SizeType ix);

const Scb* operator[] (SizeType ix) const; - Возвращает указатель на ассоциированный SCB-объект с битом номер ix в Access

Mode байте. Если этот бит сброшен, функция вернет NULL; если же бит установлен, функция вернет корректный указатель, и соответствующий SCB-объект можно изменять.

bool SetAt(SizeType ix, const Scb* pScb); - Если **ix >= SecAttrs::Size()**, то функция возвращает **false**. В противном случае она вернет **true** и, если **pScb = NULL**, то бит с номером **ix** в Access Mode байте сбрасывается; в противном же случае (если **pScb != NULL**), бит с номером **ix** в Access Mode байте устанавливается, и ассоциированному с этим битом SCB-объекту присваивается значение ***pScb**.

bool AssumeType(SaType st); - Изменяет состояние (значение) объекта класса **SecAttrs** так, что он начинает соответствовать предопределенному набору **(st)** атрибутов безопасности. См. следующее за этим классом описание.

void GetAssumedTypes(SaType* pst, size_t& sz) const; - Возвращает вектор предопределенных наборов атрибутов безопасности, под которые подходит состояние данного объекта. Если **pst = NULL**, в sz возвращается количество этих типов. Если **pst != NULL**, **sz** равен входному количеству на входе и выходному количеству на выходе.

```
enum SaType {
satPublicEF =1,     // public файл
satPrivateEF,       // private файл
satDefaultDF,       // default папка
satDefaultDO,       // default DO
};
```

Для удобства введено понятие “предопределенный набор атрибутов безопасности”. Определенным комбинациям битов в Access Mode байте, соответствующим им SCB-байтам и, в свою очередь, соответствующим им Security Object Table (таблицам объектов безопасности), поставлены в соответствие термины “private файл”, “public файл” и т.д. Эти названия пронумерованы в виде вышеуказанного перечисляемого типа **SaType**. Класс **SecAttrs** “понимает” эти значения и изменяется таким образом, чтобы подходить под семантику соответствующего набора атрибутов безопасности.

Однако, т.к. сам по себе объект **SecAttrs** "не знает", к какому объекту файловой системы он относится – к файлу, папке или объекту данных, – а значения в Access Mode байте, Security Condition Byte и Security Object Table для предопределенных типов могут совпадать для разных сущностей (файлов, папок, объектов данных), то **SecAttrs::GetAssumedTypes** может вернуть значения нескольких предопределенных типов.

```
enum SaIndex {
saiEFReadBinary   =0,
saiEFUpdateBinary,
saiEFDeactivateFile,
saiEFActivateFile ,
saiEFChangeFile   =5,
saiEFDeleteFile   =6,

saiDFCreateFile   =0,
saiDFCreateDO,
saiDFDeactivateFile,
saiDFActivateFile ,
saiDFChangeFile   =5,
saiDFDeleteSelf   =6,

saiDOUpdate =1,
saiDOUse          =2,
saiDOChange =5,
saiDODelete =6,

saiGCHVUnblock    =0,
};
```

Для удобства в библиотеке введен вышеуказанный перечисляемый тип, который содержит номера битов в Access Mode байте атрибутов безопасности. Например, чтобы построить Security Attributes для файла, который ничем не отличается от public файла, за исключением того, что никто не должен иметь права его изменять, можно использовать следующий код:

```
// Создать таблицу атрибутов безопасности
// для Public файла
SecAttrs saEF(satPublicEF);
```

```
// Поставить запрет на обновление – никогда
// нельзя записать
saEF.SetAt(saiEFUpdateBinary, &Scb(Scb::Never));
```

Важно
Далее будут обсуждаться форматы объектов файловой системы ruToken. **НАСТОЯТЕЛЬНО** рекомендуется прочитать соответствующий раздел Руководства.

class Fcp;

Описание:

Реализует понятие File Control Parameters (FCP) для файла / папки (См. "**Заголовок файла" и "Заголовок папки" в разделе "Форматы объектов файловой системы ruToken"**). Хранит данные и предоставляет доступ к свойствам конкретного файла / паки.

Методы:

Fcp(WORD wID, const SecAttrs& sa); - Первый конструктор. Строит FCP для папки.

wID – ID папки.

sa – Security Attributes для данной папки.

Fcp(WORD wID, const SecAttrs& sa, WORD wSize); - Второй конструктор. Строит FCP для файла.

wID – ID файла.

sa – Security Attributes для данного файла.

wSize – размер файла.

WORDGetID() const; - Возвращает ID файла / папки.

WORDGetSize() const; - Возвращает размер файла. Имеет смысл только если **IsEF() = true**

bool IsEF() const; - **true,** если данный FCP описывает файл, **false** – в противном случае.

const SecAttrs& GetSecAttr() const;

SecAttrs& GetSecAttr(); - Возвращают ссылку на **SecAttrs** (Security Attributes) для данного файла / папки. Вторая версия возвращает неконстантную ссылку, что позволяет изменять

SecAttrs для данного файла / папки. Изменениям подвергнется только C++ объект типа **Fcp**, а не реальный файл в ruToken.

void SetID(WORD wID); - Устанавливает ID файла / папки. Изменениям подвергнется только C++ объект типа **Fcp**, а не реальный файл в ruToken.

void SetSize(WORD wSize); - Устанавливает размер файла. Имеет смысл только если **IsEF() = true**. Изменениям подвергнется только C++ объект типа Fcp, а не реальный файл в ruToken.

Дополнительные константы:

```
enum {
SEBodySize =6, // Длина тела SE-объекта данных
GOSTKeyBodySize =32, // Длина тела GOST-объекта
// данных

SEBodyIxCompA =0, // Индекс A-компоненты в
// теле SE-объекта
SEBodyIxCompC =1, // Индекс
// C-компоненты в теле SE-объекта
SEBodyIxCompH =2, // Индекс H-компоненты
// в теле SE-объекта

// Ниже -- значения тегов  соответствующих
// компонент в CSE (Current Security
// Environment, текущее окружение безопасности).

SETagCompA =0xa4, // Тег A-компоненты CSE
SETagCompC =0xb8, // Тег C-компоненты CSE
SETagCompH =0xaa, // Тег H-компоненты CSE
};
```

class Ocp;

Описание:

Реализует понятие Object Control Parameters (OCP – по аналогии с FCP, но для объектов данных) (См. **“Заголовок объекта данных”** в разделе “**Форматы объектов файловой системы**

ruToken"). Хранит данные и предоставляет доступ к свойствам конкретного объекта данных (DO).

Внутренние типы и константы:

```
enum Type {
tSE  =0x00,
tGCHV=0x01,
tLCHV=0x11, // В данной версии
//ruToken не используется
tGOSTKey    =0x02,
};
```

Type - Тип объекта данных.

```
enum Options {
// Маски для получения различных опций DO
omskGCHVAcs =0x07, // Права доступа
// в GCHV-объекте
omskGOSTMod =0x07, // Режим шифрования
//в GOST-объекте

// Опции GCHV-объектов (права доступа)
oGCHVAcsAdmin      =0x01, // Права доступа
// Администратора
oGCHVAcsUser       =0x02, // Права доступа
// Пользователя

// Опции GOST-объектов (режимы шифрования)
oGOSTModPZ         =0x00, // Режим простой замены
oGOSTModGamm       =0x01, // Режим гаммирования
oGOSTModGammOS     =0x02, // Режим гаммирования
// с обратной связью
};
```

Options – значения опций объекта данных.

```
enum Flags {
// Маски для вычленения различных флагов
// объекта данных
```

```
fmskFmt     =0x01, // Формат DO
fmskAcs     =0x06, // Степень открытости DO

// Флаги формата объекта данных
fFmtCompact =0x01, // Компактный DO
//(иначе - стандартный DO)

// Флаги степени открытости объекта данных
fAcsClosed  =0x00, // Закрытый DO
fAcsOpenLen =0x02, // DO с открытой длиной тела
fAcsOpen    =0x06, // Открытый DO
};
```

Flags – значения флагов объекта данных.

Как видно, у Enum'ов есть значения масок для разных категорий опций и флагов. Т.е., например, если пользователь хочет проверить режим шифрования у GOST-объекта или степень открытости у объекта данных, то надо использовать примерно следующий код:

```
Ocp ocp // каким-то образом полученный
// объект типа Ocp

if (ocp.GetType() == Ocp::tGOSTKey)
{
// если это GOST-объект
switch (ocp.GetOptions() & Ocp::omskGOSTMod)
{
case Ocp::oGOSTModPZ:
// Выполнить какой-то код, если режим
// простой замены
break;
case Ocp::oGOSTModGamm:
// Выполнить какой-то код, если режим
// гаммирования
break;
case Ocp::oGOSTModGammOS:
// Выполнить какой-то код, если режим
```

```
// гаммирования с ОС
break;
}

if ((ocp.GetFlags() & Ocp::fmskAcs) ==
Ocp::fAcsOpen) {
// Если это открытый DO, то, например,
// считать его тело
}
}
```

Методы:

Ocp(BYTE byID, ObjectType ot, const SecAttrs& sa, BYTE byOptions, BYTE byFlags, BYTE byRetryCounter =0x00); - Конструктор. Параметры:

byID – ID объекта данных.

ot – тип объекта данных

sa – Security Attributes для данного объекта данных.

byOptions – опции объекта данных

byFlags – флаги объекта данных

byRetryCounter – значение счетчика числа попыток доступа к DO. Старший нибл (четверка бит) - максимальное число попыток доступа к DO, младший нибл - оставшееся число попыток доступа к DO. Т.к. в текущей версии ruToken нельзя создавать GCHV-объекты, а для объектов других типов этот параметр не используется, он установлен в умалчиваемое значение 0.

Ocp(const Ocp&)

Ocp& operator= (const Ocp&); - Конструктор копирования и присваивающее копирование. Они могут возбудить исключение, если указатель на тело некорректен или оператор new не может выделить достаточно памяти. В этом случае гарантируется, что объект остается в устойчивом, неизмененном состоянии, т.е. таким, каким он был до входа в функцию, которая возбудила исключение.

BYTE GetID() const; - Возвращает ID объекта данных.

ObjectType GetType() const; - Возвращает тип объекта данных.

BYTE GetRetryCntr() const; - Возвращает значение счетчика числа попыток доступа к DO. Старший нибл (четверка бит) - максимальное число попыток доступа к DO, младший нибл - оставшееся число попыток доступа к DO.

BYTE GetOptions() const; - Возвращает значение байта опций объекта данных.

BYTE GetFlags() const; - Возвращает значение байта флагов объекта данных.

bool IsInPredefinedDF() const; - true, если объект данных имеет предопределенный ID (т.е. находится в предопределенной папке), false – в противном случае.

LPCBYTE GetBody() const; - Возвращает указатель на тело объекта данных.

WORD GetBodySize() const; - Возвращает длину тела объекта данных.

const SecAttrs& GetSecAttr() const;

SecAttrs& GetSecAttr(); - Возвращают ссылку на **SecAttrs** (Security Attributes) для данного объекта данных. Вторая версия возвращает неконстантную ссылку, что позволяет изменять **SecAttrs** для данного объекта данных. Изменениям подвергнется только C++ объект типа Ocp, а не реальный объект данных в ruToken.

Примечание

Все нижеследующие **SetXxx** методы изменяют только C++ объект типа Ocp, а не реальный объект данных в ruToken.

void SetID(BYTE byID); - Устанавливает ID объекта данных.

void SetType(ObjectType t); - Устанавливает тип объекта данных.

void SetRetryCntr(BYTE rc); - Устанавливает значение счетчика числа попыток доступа к DO. Старший нибл (четверка бит) -

максимальное число попыток доступа к DO, младший нибл - оставшееся число попыток доступа к DO.

void SetOptions(BYTE opt); - Устанавливает значение байта опций объекта данных.

void SetFlags(BYTE f); - Устанавливает значение байта флагов объекта данных.

void SetBody(LPCBYTE pbyBody, WORD wBodySize); - Устанавливает тело объекта данных. Если **pbyBody = NULL**, тогда текущее тело уничтожается (т.е. **GetBody()** вернет **NULL**), но длина тела устанавливается в **wBodySize** (т.е. **GetBodySize()** вернет **wBodySize**). Это используется в тех случаях, когда в процессе создания объекта данных нужно установить длину его тела, не зная самого тела (которое будет, например, сгенерировано). Эта функция может возбудить исключение. В этом случае гарантируется, что сам объект остается в устойчивом (неизмененном) состоянии (т.е. таким, каким он был до входа в функцию) и может быть безопасно далее использован.

Важно

Далее будет обсуждаться APDU "SELECT FILE". НАСТОЯТЕЛЬНО рекомендуется прочитать "Выбрать файл / папку" в разделе "APDU ruToken".

Т.к. APDU-команда "SELECT FILE" достаточно сложна и объемна в плане используемых ею флагов и опций, сама семантика APDU-команды разбита на 2 типа классов: первый класс определяет параметры (значения P1, P2, Lc и DATA APDU-команды), а второй – реализует саму APDU-команду, принимая в свой конструктор в качестве параметра ссылку на объект класса параметров для APDU-команды. Сначала будут описаны все эти классы, затем – приведены примеры их использования.

```
enum SfFlags
{
sffFirst    = 0x00,// Выбрать первый файл
sffLast     = 0x01,// Выбрать последний файл
sffNext     = 0x02,// Выбрать следующий файл,
// после указанного
sffPrev     = 0x03,// Выбрать предыдущий файл,
// перед указанным
```

```
sffRetFCP   = 0x04 // Вернуть FCP для выбранного
// файла (всегда задается)
};
```

SfFlags – определяет значения (маску) флагов для APDU-команды “SELECT FILE” (эти флаги передаются через параметр **P2**).

class SfParams;

Описание:

Класс, реализующий параметры для APDU-команды “SELECT FILE”. Можно создавать напрямую объекты данного класса, однако гораздо удобнее (и рекомендуется) использовать вспомогательные функции по созданию объектов этого класса (см. их описания ниже).

SfParams SfFirst()

Описание:

Функция создает объект класса SfParams, реализующий параметры для APDU-команды “SELECT FILE” в случае, когда **P1** равен 0x00, **P2** равен 0x04, а поле **DATA** отсутствует, т.е. надо выбрать ***первый*** файл/папку в текущей папке.

SfParams SfLast()

Описание:

Функция создает объект класса SfParams, реализующий параметры для APDU-команды “SELECT FILE” в случае, когда **P1** равен 0x00, **P2** равен 0x05, а поле **DATA** отсутствует, т.е. надо выбрать ***последний*** файл/папку в текущей папке.

SfParams SfAny(WORD wID, BYTE sff =sffNext)

Описание:

Функция создает объект класса SfParams, реализующий параметры для APDU-команды “SELECT FILE” в случае, когда **P1** равен 0x00, **P2** комбинация флага 0x04 (вернуть FCP) с ***sff*** параметром, а в поле **DATA** передается ***wID***, т.е. надо выбрать объект любого типа – папку или файл в текущей папке, который удовлетворяет заданным условиям. По умолчанию ***sff*** настроен на

sffNext, т.е. выбрать ***следующий*** файл/папку в текущей папке после файла/папки с ID = ***wID***.

SfParams SfSelMF()

Описание:

Функция создает объект класса SfParams, реализующий параметры для APDU-команды "SELECT FILE" в случае, когда **P1** равен 0x00, **P2** равен 0x04, а в поле **DATA** передается 0x3f00, т.е. надо выбрать Корневую папку (MF). Того же результата можно добиться, используя **SfAny(0x3f00, sffFirst)**.

SfParams SfChildEF(WORD wID)

Описание:

Функция создает объект класса SfParams, реализующий параметры для APDU-команды "SELECT FILE" в случаях, когда **P1** равен 0x02, а в поле **DATA** передается ID файла в текущей папке, который надо выбрать.

SfParams SfChildDF(WORD wID)

Описание:

Функция создает объект класса SfParams, реализующий параметры для APDU-команды "SELECT FILE" в случаях, когда **P1** равен 0x01, а в поле **DATA** передается ID подпапки в текущей папке, которую надо выбрать, т.е. надо выбрать подпапку.

SfParams SfParentDF()

Описание:

Функция создает объект класса SfParams, реализующий параметры для APDU-команды "SELECT FILE" в случаях, когда **P1** равен 0x03, а поле **DATA** отсутствует, т.е. надо выбрать родительскую папку текущей папки.

SfParams SfFromMF(const WORD* pwPath, DWORD dwCnt)

Описание:

Функция создает объект класса SfParams, реализующий параметры для APDU-команды “SELECT FILE” в случаях, когда **P1** равен 0x08, а поле **DATA** содержит путь от MF, не включая ID MF, т.е. надо выбрать по пути от Корневой папки.Параметры:

pwPath – путь от MF, не включая ID MF.

dwCnt – количество ID в буфере pwPath.

Примечание
Объект SfParams, созданный с помощью этой функции, является “легкой” оберткой вокруг данного буфера пути (**pwPath**), т.е. он не копирует данный буфер в свое внутреннее состояние. Поэтому продолжительность жизни объекта класса SfParams должна быть меньше или равна продолжительности жизни буфера **pwPath**. Вообще не рекомендуется создавать объекты **SfParams** для долгого хранения (См. пример по их использованию ниже, после описания класса **SelectFileApdu)**.

SfParams SfFromDF(const WORD* pwPath, DWORD dwCnt);

Описание:

Функция создает объект класса SfParams, реализующий параметры для APDU-команды “SELECT FILE” в случаях, когда **P1** равен 0x09, а поле **DATA** содержит путь от текущей папки, не включая ID текущей папки, т.е. надо выбрать по пути от текущей папки.Параметры:

pwPath – путь от текущей папки, не включая ID текущей папки.

dwCnt – количество ID в буфере **pwPath**.

Примечание
См. Примечания для SfFromMF.

class SelectFileApdu : public ApduS;

Описание:

Реализует APDU-команду “SELECT FILE”.

Методы:

explicit SelectFileApdu(const SfParams& p); - Конструктор. Строит объект класса SelectFileApdu. Параметры:

p – Ссылка на объект класса параметров для APDU-команды "SELECT FILE".

Fcp GetFcp() const; - Возвращает **FCP** для выбранного файла/папки. Следует вызывать только после вызова **Send**, если **GetReplySW()** вернула 0x9000. Может возбудить исключение **BadBuffer** (класс исключения, описание см. ниже), если при анализе выходного буфера для построения **FCP** были обнаружены ошибки.

Пример:

```
// Выбрать первый файл в текущей папке
SelectFileApdu sfa1(SfFirst());
sfa1.Send(card);

// Выбрать следующий за файлом 0x0001
// файл в текущей папке
SelectFileApdu sfa2(SfAny(0x0001, sffNext));
sfa2.Send(card);

// Выбрать родительскую папку
SelectFileApdu sfa3(SfParentDF());
sfa3.Send(card);

// Выбрать файл по пути
// 0x3f00/0x0001/0x0002/0x0003 от MF
WORD vwPath[] = { 0x0001, 0x0002, 0x0003 };

SelectFileApdu sfa4( SfFromMF(vwPath,
sizeof(vwPath)/sizeof(vwPath[0])) );

// Получить FCP выбранного файла
if (SCARD_S_SUCCESS == sfa4.Send(card) &&
sfa4.GetReplySW() == SW_SUCCESS)
{
   try
```

```
    {
       Fcp fcp( sfa4.GetFcp() );
    }
    catch (BadBuffer&)
    {
     // По идее такого не должно случиться, но...
     // Обрабатываем исключение.
    }
}
```

class CreateFileApdu : public ApduS;

Описание:

Реализует APDU-команду “CREATE FILE”. Создает файл или папку в текущей папке.

Методы:

explicit CreateFileApdu(const Fcp& fcp); - Конструктор. Реализует построение APDU-команды “CREATE FILE” на основе данных, хранящихся в Fcp. Параметры:

fcp – задает ID, параметры, опции для создаваемого файла или папки.

class DeleteFileApdu : public ApduS;

Описание:

Реализует APDU-команду “DELETE FILE”.

Методы:

DeleteFileApdu(); - Конструктор по умолчанию. Реализует построение APDU-команды “DELETE FILE” для удаления текущего файла или папки. Если текущий файл существует, будет удален он, если нет – будет удалена текущая папка.

explicit DeleteFileApdu(WORD wID); - Конструктор. Реализует построение APDU-команды “DELETE FILE” для удаления в текущей папке файла или папки с ID, равным **wID**.

Примечание

Удаляемая папка должна быть пустой (не должна содержать вложенных объектов файловой системы).

class ReadBinaryApdu : public ApduS;

Описание:

Реализует APDU-команду “READ BINARY”.

Методы:

explicit ReadBinaryApdu(WORD wOffset, BYTE byOutLen =MAX_LE_VAL) - Конструктор. Реализует построение APDU-команды “READ BINARY”. Параметры:

wOffset – смещение в файле, с которого надо начинать считывание.

byOutLen – размер данных для считывания. Если **byOutLen = 0**, считать максимальное количество (256 байт).

Примечание
Считанные данные надо получать через **GetReply()**, а их длину через **GetReplyLen()**.

class UpdateBinaryApdu : public ApduS;

Описание:

Реализует APDU-команду “UPDATE BINARY”.

Методы:

UpdateBinaryApdu(WORD wOffset, LPCBYTE pbyBuffer, BYTE byLen) - Конструктор. Реализует построение APDU-команды “UPDATE BINARY”. Параметры:

wOffset – Смещение в файле, с которого надо начинать запись.

pbyBuffer – Буфер для записи.

byLen – Размер буфер для записи. Если **byLen = 0**, записать максимальное количество (256 байт).

class VerifyApdu : public ApduS;

Описание: см. "Выполнить аутентификацию владельца токена" в разделе "APDU ruToken".

Реализует APDU-команду "VERIFY".

APDU-команда работает в двух режимах:

1. Позволяет выполнить аутентификацию владельца токена путем проверки переданного PIN-кода. В случае успешной аутентификации в токене устанавливаются текущие права доступа, ассоциированные с данным PIN-кодом.

2. Позволяет проверить, была ли уже проведена аутентификация с этими правами доступа, либо получить оставшееся количество попыток аутентификации с этими правами доступа.

Режим выбирается посредством вызова определенного конструктора.

Чтобы у пользователя была возможность отслеживать момент аутентификации, класс предоставляет строку сообщения Windows, которую пользователь может зарегистрировать у себя в приложении посредством Win32 API функции **RegisterWindowMessage(),** и тогда он сможет получать сообщение Windows, говорящее, что текущие права доступа в токене изменились.

Методы:

VerifyApdu(BYTE bySecObjID, LPCBYTE pbyPin, BYTE byLen, HWND hWndSender =NULL) – Конструктор, реализующий первый режим. Параметры:

bySecObjID - ID GCHV-объекта, используемого при аутентификации; 0 – для идентификации используется GCHV-объект, указанный в A-компоненте CSE.

pbyPin – PIN-код.

byLen – длина PIN-кода. Если **byLen = 0**, значит максимальное количество (256 байт).

hWndSender – хендл окна, который будет передан в качестве параметра WPARAM сообщения Windows, посланного при аутентификации.

explicit VerifyApdu(BYTE bySecObjID) – Конструктор, реализующий второй режим. Параметры:

bySecObjID - ID GCHV-объекта, для которого надо проверить, была ли уже проведена аутентификация с данными правами доступа или получить оставшееся количество попыток аутентификации с данными правами доступа.

bool Broadcast() const;

void Broadcast(bool bBroadcast); - Первый метод позволяет узнать, установлен ли флаг бродкастинга (т.е. будет ли сообщение рассылаться всем окнам). По умолчанию он всегда установлен. Второй метод позволяет снять/установить этот флаг.

static LPCTSTR GetWindowMessage(); - Возвращает строку нашего сообщения Windows, которую пользователь должен передать в Win32 API функцию **RegisterWindowMessage()**.

class ResetAccessRightsApdu : public ApduS;

Описание: См. "Очистить текущие права доступа" в разделе "APDU ruToken".

Реализует APDU-команду "RESET ACCESS RIGHTS", выполняющую сброс текущих прав доступа (т.е. текущими становятся права доступа "Гость"). После выполнения APDU-команды владелец токена сможет выполнять только те операции, для которых не требуется аутентификация.

Чтобы у пользователя была возможность отслеживать момент сброса текущих прав, класс **VerifyApdu** предоставляет строку сообщения Windows, которую пользователь может зарегистрировать у себя в приложении посредством Win32 API функции **RegisterWindowMessage(),** и тогда он сможет получать сообщение Windows, говорящее, что текущие права доступа в токене изменились. (См. **VerifyApdu**)

Методы:

explicit ResetAccessRightsApdu(HWND hWndSender =NULL) – Конструктор. Параметры:

hWndSender – хендл окна, который будет передан в качестве параметра **WPARAM** сообщения Windows, посланного при аутентификации.

bool Broadcast() const;

void Broadcast(bool bBroadcast); - Первый метод позволяет узнать, установлен ли флаг бродкастинга (т.е. будет ли сообщение рассылаться всем окнам). Второй метод позволяет снять/установить этот флаг.

Примечание
Чтобы получить строку сообщения Windows, пользователь должен вызвать функцию **VerifyApdu::GetWindowMessage().**

class ChangeReferenceDataApdu : public ApduS;

Описание: см. "Изменить PIN-код владельца токена" в разделе "APDU ruToken".

Реализует APDU-команду "CHANGE REFERENCE DATA", позволяющую сменить PIN-код Пользователя или Администратора.

Методы:

ChangeReferenceDataApdu(BYTE byDoId, LPCBYTE pbyNewPin, BYTE byLen) – Конструктор. Параметры:

byDoId – ID GCHV-объекта, содержащего PIN-код, который требуется изменить.

pbyNewPin - новый PIN-код.

byLen – длина нового PIN-кода. Если **byLen = 0**, значит максимальное количество (256 байт).

class ResetRetryCounterApdu : public ApduS;

Описание:

Реализует APDU-команду "RESET RETRY COUNTER", позволяющую установить счетчик попыток доступа к GCHV-объекту в его исходное, максимальное значение (т.е. разблокировать GCHV-объект).

Методы:

explicit ResetRetryCounterApdu(BYTE byDoId) – Конструктор. Параметры:

byDoId – ID GCHV-объекта, подлежащего разблокировке.

Примечание
В действительности команда может разблокировать только объект, предоставляющий права доступа "Пользователь" (**ID = 0x2**). Объект с правами "Администратор" (ID = 0x1) не подлежит разблокированию.

class PutDataApdu : public ApduS;

Описание:

Базовые классы для других. Реализуют APDU-команды "GET DATA" и "PUT DATA" соответственно.

class CreateDOApdu : public PutDataApdu;

Описание:

Реализует APDU-команду "CREATE DO", позволяющую создать объект данных (DO) в памяти ruToken.

Методы:

explicit CreateDOApdu(const Ocp& ocp); – Конструктор. Параметры:

ocp – объект типа **Ocp**. Предоставляет тип, параметры, опции, флаги, ID для создания DO.

class GetDOInfoApdu : public ApduS;

Описание:

Реализует APDU-команду "GET DO INFO", позволяющую получить набор сведений об указанном объекте данных (OCP).

Типы:

```
enum Flags {
     getThis  = 0x00, // флаг для получения
// OCP указанного DO
     getNext = 0x02,  // флаг для получения
// OCP DO, следующего за указанным
};
```

Методы:

GetDOInfoApdu() – Конструктор по умолчанию. Создает APDU-команду для получения сведений о первом DO в текущей папке.

explicit GetDOInfoApdu(BYTE byDoId, Flags flag =getThis) - Конструктор. Создает APDU-команду получения сведений о DO с ID, равным **byDoId,** либо о DO, следующим после DO с ID, равным **byDoId**.

Ocp GetOcp() const; - Возвращает **OCP** для выбранного объекта данных. Следует вызывать только после вызова **Send**, если **GetReplySW()** вернула 0x9000. Может возбудить исключение **BadBuffer** (класс исключения, описание см. ниже), если при анализе выходного буфера для построения **OCP** были обнаружены ошибки.

class DeleteDOApdu : public PutDataApdu;

Описание:

Реализует APDU-команду "DELETE DO", позволяющую удалить указанный объект данных.

Методы:

explicit DeleteDOApdu(BYTE byDoId) - Конструктор. Строит APDU-команду для удаления DO с ID, равным **byDoId**.

class GenerateKeyApdu : public PutDataApdu;

Описание:

Реализует APDU-команду "GENERATE KEY ", позволяющую сгенерировать ключ шифрования и поместить его в создаваемый DO.

Методы:

explicit GenerateKeyApdu(const Ocp& ocp); - Конструктор. Строит APDU-команду для генерации ключа шифрования. В **ocp** передается ID DO, его тип (**Ocp::otGostKey**), опции, флаги, атрибуты безопасности и т.п. Тело в **ocp** не должно быть, а длина должна быть указана (32 байта для ключей ГОСТ – **константа GOSTKeyBodySize**). Для удобства, если длина тела в **ocp** будет равна 0 (т.е. **ocp.GetBodySize() = 0**), **GenerateKeyApdu** для ключа ГОСТ установит длину по умолчанию (32 байта).

class ManageSEApdu : public ApduS;

Описание:

Реализует APDU-команду "MANAGE SECURITY ENVIRONMENT" ("MSE") (см. "Создать текущее окружение безопасности" в разделе "APDU ruToken"). APDU-команда позволяет создать в оперативной памяти ruToken текущее окружение безопасности (Current Security Environment, CSE) – структуру данных, содержащую сведения об объектах данных, используемых в различных security-ориентированных операциях. Команда позволяет создать CSE двумя способами:

1. Загрузкой всех компонент CSE из указанного SE-объекта (режим **RESTORE**);

2. Прямой установкой компоненты CSE (режим **SET**).

Режим работы команды определяется соответствующим конструктором.

Методы:

explicit ManageSEApdu(BYTE bySeId); - Конструктор. Строит APDU-команду "MSE" в режиме RESTORE. Параметры:

bySeId – ID SE-объекта, загрузка всех компонент которого будет произведена.

explicit ManageSEApdu(BYTE byCompTag, BYTE byVal); - Конструктор. Строит APDU-команду "MSE" в режиме SET. Параметры:

byCompTag – Тег компоненты, которую надо установить (т.е. одно из **SETagCompA**, **SETagCompC** или **SETagCompH**, см. выше).

byVal – значение, которое нужно установить этой компоненте (ID какого-нибудь DO). Чтобы сбросить компоненту, надо передать 0x00 в **byVal**.

class PerformSOApdu : public ApduS;

Описание:

Реализует APDU-команду "PERFORM SECURITY OPERATION" ("PSO") (см. "Выполнить Security-операцию" в разделе "APDU ruToken").

APDU-команда позволяет выполнить определенную security-операцию: шифрование, вычисление контрольной суммы, hash-значения, цифровой подписи и т.п.

Текущая версия ruToken позволяет выполнять 3 security-операции:

- Зашифрование по ГОСТ 28147-89
- Расшифрование по ГОСТ 28147-89
- Вычисление hash-функции (имитовставки по ГОСТ 28147-89)

Все security-операции производятся на ключах шифрования ГОСТ, содержащихся в предварительно созданных GOST-объектах данных. Перед выполнением операции при помощи APDU-команды “MSE” должно быть создано текущее окружение безопасности (CSE), соответствующие компоненты которого ссылаются на используемые в операциях GOST-объекты.

Для удобства класс **PerformSOApdu** был оставлен как базовый, а от него были унаследованы еще 3 класса **EncipherApdu, DecipherApdu** и **HashApdu**.

Методы:

PerformSOApdu(BYTE byCla, BYTE byP1, BYTE byP2, BYTE byLc, LPCBYTE pData, BYTE byLe) – Конструктор. Параметры:

byCla – **CLA** байт.

byP1 – **P1** байт

byP2 – **P2** байт

byLc – Значение поля **Lc**. Если **Lc = 0x00** – то максимум (256 байт).

pData – указатель на данные в APDU-команде (т.е. поле **DATA**). Их длина должна быть равной **byLc**.

byLe – Значение поля **Le**. Если **Le = 0x00** – то максимум (256 байт).

void Rebuild(BYTE byCla, BYTE byP1, BYTE byP2, BYTE byLc, LPCBYTE pData, BYTE byLe); - Перестраивает APDU-команду. Для значения параметров см. конструктор.

class EncipherApdu : public PerformSOApdu;

Описание:

Реализует APDU-команду “PSO:ENCIPHER”.

ВНИМАНИЕ! Описание формата данных в APDU-команде и APDU-ответе см. в “**Зашифрование по ГОСТ 28147-89”** в разделе “APDU ruToken”.

Методы:

EncipherApdu(bool bLast, BYTE byLc, LPCBYTE pData, BYTE byLe = MAX_LE_VAL); - Конструктор. Параметры:

bLast – **true**, если это единственная или последняя APDU-команда в цепочке APDU.

byLc – размер поля DATA с данными для зашифрования. Если **byLc = 0x00**, то максимум (256 байт).

pData – поле DATA с данными для зашифрования.

byLe – размер данных, ожидаемых в APDU-ответе. Если **byLe = 0x00**, то максимум (256 байт).

void Rebuild(bool bLast, BYTE byLc, LPCBYTE pData, BYTE byLe =MAX_LE_VAL); - Перестраивает APDU-команду. Для значения параметров см. конструктор.

Примечание
Зашифрованные данные и их длина могут быть получены через **GetReply()** и **GetReplyLen()**.

class DecipherApdu : public PerformSOApdu;

Описание:

Реализует APDU-команду “PSO:DECIPHER”.

ВНИМАНИЕ! Описание формата данных в APDU-команде и APDU-ответе см. в “**Расшифрование по ГОСТ 28147-89”** в разделе “APDU ruToken”.

Методы:

DecipherApdu(bool bLast, BYTE byLc, LPCBYTE pData, BYTE byLe = MAX_LE_VAL); - Конструктор. Параметры:

bLast – **true**, если это единственная или последняя APDU-команда в цепочке APDU.

byLc – размер поля DATA с данными для расшифрования. Если **byLc = 0x00**, то максимум (256 байт).

pData – поле DATA с данными для расшифрования.

byLe – размер данных, ожидаемых в APDU-ответе. Если **byLe = 0x00**, то максимум (256 байт).

void Rebuild(bool bLast, BYTE byLc, LPCBYTE pData, BYTE byLe =MAX_LE_VAL); - Перестраивает APDU-команду. Для значения параметров см. конструктор.

Примечание
Расшифрованные данные и их длина могут быть получены через **GetReply()** и **GetReplyLen()**.

class HashApdu : public PerformSOApdu;

Описание:

Реализует APDU-команду “PSO:HASH”.

ВНИМАНИЕ! Описание формата данных в APDU-команде и APDU-ответе см. в “ **Вычисление имитовставки по ГОСТ 28147-89”** в разделе “APDU ruToken”.

Методы:

HashApdu(bool bLast, BYTE byLc, LPCBYTE pData, BYTE byLe =MAX_LE_VAL); - Конструктор. Параметры:

bLast – **true**, если это единственная или последняя APDU-команда в цепочке APDU.

byLc – размер поля DATA с данными для вычисления имитовставки. Если **byLc = 0x00**, то максимум (256 байт).

pData – поле DATA с данными для вычисления имитовставки.

byLe – размер данных, ожидаемых в APDU-ответе. Если **byLe = 0x00**, то максимум (256 байт).

void Rebuild(bool bLast, BYTE byLc, LPCBYTE pData, BYTE byLe =MAX_LE_VAL); - Перестраивает APDU-команду. Для значения параметров см. конструктор.

Примечание
Имитовставка и ее длина могут быть получены через **GetReply()** и **GetReplyLen()**.

class GetRuTokenIdApdu : public GetDataApdu;

Описание:

Реализует APDU-команду "GET RUTOKEN ID" Эта APDU-команда возвращает 4-байтовый уникальный серийный номер (ID) ruToken.

Методы:

explicit GetRuTokenIdApdu(BYTE byIdLen =4); - Конструктор. Параметры:

byIdLen – длина ruToken ID (в текущей версии ruToken – 4 байта).

Примечание
ruToken ID может быть получен через **GetReply()**.

class GetRuTokenSysinfoApdu : public GetDataApdu;

Описание:

Реализует APDU-команду "GET RUTOKEN INFO ". Эта APDU-команда возвращает набор параметров, идентифицирующих данный ruToken: его тип, аппаратную версию, общий объем EEPROM-памяти, номер протокола обмена с токеном, номер микропрограммы, номер заказа токена.

Методы:

explicit GetRuTokenSysinfoApdu(BYTE byLen =8); - Конструктор. Параметры:

byLen – ожидать **byLen** (в текущей версии ruToken – 8) байт идентификационных данных.

bool GetRtSysInfo(LPRTSYSINFO psi); - Преобразует выходной буфер идентификационных данных в структуру **RTSYSINFO** (см. ее описание ниже). Возвращает **false**, если **GetReplySW() != 0x9000** или **psi = NULL**; В противном случае заполняет структуру ***psi** и возвращает **true**.

```
typedef struct tagRTSYSINFO
{
      BYTE          byType ;
      BYTE          byMajVer;
      BYTE          byMinVer;
```

```
    DWORD  dwMemory;
    BYTE          byProtNo;
    BYTE          byMicroProgNo;
    BYTE          byOrderNo;
    WORD   wReserved;
} RTSYSINFO, *LPRTSYSINFO;
```

Описание полей:

byType – байт типа ruToken; текущий тип – 0.

byMajVer – Аппаратная версия ruToken (Major).

byMinVer – Аппаратная версия ruToken (Minor).

dwMemory - Общий объем EEPROM-памяти ruToken в 8-килобайтовых единицах.

byProtNo – Номер протокола обмена с ruToken.

byMicroProgNo – Номер микропрограммы в ruToken.

byOrderNo – Номер заказа ruToken.

wReserved – Резерв

class GetRuTokenFreememApdu : public GetDataApdu;

Описание:

Реализует APDU-команду “GET FREE MEMORY ”. APDU-команда возвращает объем свободной файловой памяти (т.е. суммарную длину всех свободных секторов памяти) в ruToken.

Методы:

explicit GetRuTokenFreememApdu(BYTE byLen =4); – Конструктор. Параметры:

byLen – Ожидать **byLen**-байтовое значение свободной памяти длина ruToken ID.

Примечание
Объем файловой памяти может быть получен через **GetReply()**.

class GetCurrEFIdApdu : public GetDataApdu;

Описание:

Реализует APDU-команду “GET CURRENT FILE”. Команда возвращает ID текущего файла, если он существует. Если теку-

щего файла в системе нет, будет возвращен соответствующий код ошибки.

Методы:

GetCurrEFIdApdu(); - Конструктор.

Примечание
ID этого файла может быть получено через GetReply().

class ResetTokenApdu : public ApduS;

Описание:

Реализует APDU-команду “RESET RUTOKEN ”. APDU-команда позволяет выполнить “горячий” сброс токена в его начальное состояние и возвращает ATR-строку в формате ISO 7816-3, предназначенную для идентификации токена стандартными средствами.

Состояние ruToken после “горячего” сброса:

- Текущая папка – Корневая папка.
- Текущий файл отсутствует.
- Все компоненты CSE сброшены.
- Права доступа – “Гость”.

Методы:

explicit ResetTokenApdu(BYTE byAtrLen =19); - Конструктор. Параметры:

byAtrLen – Ожидать **byAtrLen**-байтовую ATR-строку (19 байт).

Примечание
ATR-строка может быть получена через **GetReply()**.

class GetChallengeApdu : public ApduS;

Описание:

Реализует APDU-команду “GET CHALLENGE”, позволяющую получить случайное число размером 32 байта.

Методы:

explicit GetChallengeApdu(BYTE byLen =32) Конструктор. Параметры:

byLen – Ожидать byLen-байтовое случайное число (32 байта).

Примечание
Случайное число может быть получено через **GetReply()**.

bool InitializeToken(Card& card, LONG* pLR =NULL, WORD* pSW =NULL);

Описание:

Функция инициализирует ruToken. По окончании процесса инициализации в памяти токена содержится только дерево предопределенных папок со служебными файлами, а также GCHV-объекты с PIN-кодами Пользователя и Администратора по умолчанию.

Возвращает:

true – если ruToken был успешно инициализирован.

false – в противном случае. Для дополнительной информации см. Примечание. В случае неудачного завершения функции не гарантируется, что токен находится в устойчивом состоянии и может быть безопасно использован в дальнейшем.

Параметры:

card – объект класса Card, описывающий токен, который надо инициализировать.

pLR – указатель на LONG. См. Примечание.

pSW – указатель на WORD. См. Примечание.

Примечание
1. Для выполнения функции требуется, чтобы текущими правами доступа были права Администратора.

2. **pLR** и **pSW** служат источниками дополнительной информации, если инициализация токена была неудачной. При входе в функцию ***pLR** принимает значение **SCARD_S_SUCCESS**, а ***pSW** - **SW_SUCCESS** (0x9000). Если функция вернула **false**, то один из этих параметров будет содержать код ошибки. Т.е. если ***pLR != SCARD_S_SUCCESS**, то ***pLR** содержит код ошибки Resource Manager; а если ***pSW != SW_SUCCESS**, то ***pSW** содержит слово состояния,

вернувшееся с ruToken. Если **pLR** или/и **pSW** равен **NULL**, дополнительная информация будет недоступна.

bool GetTokenName(Card& card, LPSTR pString, DWORD nMaxCount, LONG* pLR =NULL, WORD* pSW =NULL);

Описание:

Функция возвращает символьное имя ruToken.

Возвращает:

true – если имя было успешно получено.

false – в противном случае. Для дополнительной информации См. Примечание 2 для функции **InitializeToken**.

Параметры:

card – объект класса Card, описывающий токен, символьное имя которого надо получить.

pString – указатель, куда надо скопировать символьное имя токена

nMaxCount – максимальное количество символов, которое может быть скопировано в **pString**, включая завершающий 0-символ. Если имя превышает этот предел, оно обрезается.

pLR – указатель на LONG. См. Примечание 2 для функции **InitializeToken**.

pSW – указатель на WORD. См. Примечание 2 для функции **InitializeToken**.

bool SetTokenName(Card& card, LPCSTR pString, LONG* pLR =NULL, WORD* pSW =NULL);

Описание:

Функция изменяет символьное имя ruToken. Для выполнения функции требуется, чтобы текущими правами доступа были права Пользователя.

Возвращает:

true – если символьное имя было успешно изменено.

false – в противном случае. Для дополнительной информации См. Примечание 2 для функции **InitializeToken**.

Параметры:

card – объект класса Card, описывающий токен, символьное имя которого надо изменить.

pString – указатель на новое символьное имя токена, которое должно завершаться нулевым символом.

pLR – указатель на LONG. См. Примечание 2 для функции **InitializeToken**.

pSW – указатель на WORD. См. Примечание 2 для функции **InitializeToken**.

EXCEPT.H

Назначение файла: Содержит классы, описывающие исключения, которые могут возбудить функции (методы) библиотеки rtAPI.

---

class Exception;

Описание:

Базовый класс для исключений.

class BadBuffer : public Exception;

Описание:

Класс, описывающий исключение при анализе “плохого” буфера.

class BadAlloc : public Exception;

Описание:

Класс, описывающий исключение при недостатке памяти.

Высокоуровневые интерфейсы прикладного программирования

Высокоуровневые интерфейсы обеспечивают наиболее абстрактное представление токенов и их функциональности. Обычно эти интерфейсы обеспечивают протоколы обмена, арбитраж, учитывают особенности различных типов токенов и освобождают разработчиков приложений для концентрации на функциональности приложений. За это приходится расплачиваться снижением производительности и ограничениями, накладываемыми спецификой каждого такого интерфейса.

В комплект разработчика включены высокоуровневые API, предназначенные для работы с Microsoft Crypto API и PKCS #11.

При соблюдении механизмов арбитража, возможно одновременное использование высокоуровневых и низкоуровневых интерфейсов без каких бы то ни было проблем.

Сервис-провайдер ruToken

Сервис-провайдер ruToken представляет собой компонент подсистемы смарт-карт, который обеспечивает доступ к специфичным для смарт-карт сервисам токена посредством программного интерфейса.

В соответствии со стандартом PC/SC, Сервис-провайдер ruToken состоит из двух компонент: Сервис-провайдера смарт-карты и Криптографического сервис-провайдера.

Комплект разработчика содержит следующие файлы, относящиеся к Сервис-провайдеру ruToken:

Lib\Providers\rtcsp.dll	Криптографический сервис-провайдер ruToken.
Lib\Providers\rtcsp.sig	Файл подписи Криптографического сервис-провайдера.
Lib\Providers\rticcsp.dll	Сервис-провайдер смарт-карты ruToken.
Lib\Providers\SetupProv.exe	Инсталлятор Сервис-провайдера ruToken.

Установка Сервис-провайдера ruToken

Для установки Сервис-провайдера ruToken служит утилита SetupProv.exe, имеющая оконный интерфейс. С ее помощью можно установить (переустановить) оба компонента Сервис-провайдера ruToken в системную папку Windows, а также удалить его из системы.

Порядок работы с утилитой аналогичен порядку работы с утилитой установки драйверов ruToken SetupDrv.exe (см. “Установка драйверов” раздела “Комплект разработчика”). Подобно этой утилите, SetupProv поддерживает пакетный режим из командной строки, ее можно использовать для установки Сервис-провайдера ruToken из программы инсталляции ПО, использующего ruToken.

В процессе инсталляции Комплекта разработчика Сервис-провайдер ruToken устанавливается в систему автоматически.

Сервис-провайдер смарт-карты

Сервис-провайдер смарт-карты (Integrated Circuit Card Service Provider, ICCSP) предоставляет интерфейс к стандартным некриптографическим сервисам, а также к расширенным сервисам ruToken.

Интерфейс состоит из предопределенного набора сервисов, протоколов, необходимых для обращения к сервисам токена.

Поддержка какого-либо интерфейса регистрируется ассоциированием с его глобальным уникальным идентификатором (Globally Unique Identifier, GUID).

Привязка токена к какому-либо интерфейсу происходит в тот момент, когда токен впервые вводится в систему – как правило, при установке ICCSP. С этого момента приложения могут обращаться к ruToken, используя специфичный для нее интерфейс, либо GUID.

ICCSP ruToken представляет собой in-proc COM-сервер (динамическую библиотеку классов, реализующих различные COM-интерфейсы). На данный момент он содержит 13 COM-объектов, каждый из которых реализует по одному интерфейсу (не считая стандартного IDispatch, который наследует IUnknown).

Потоковая модель COM-объектов – Both.

Примечание
Условимся понимать под выражением, например, "объект ISCardManage" объект, реализующий интерфейс ISCardManage.

ICCSP ruToken поддерживает следующие **стандартные** интерфейсы:

IbyteBuffer
ISCard
ISCardCmd
ISCardDatabase
ISCardISO7816
ISCardManage
ISCardFileAccess
ISCardVerify
ISCardAuth

Кроме того, в ICCSP ruToken были введены **интерфейсы расширения**, реализующие дополнительные возможности ruToken и / или повышающие удобство работы со стандартными интерфейсами:

ISCardObjectAccess
IsecOperation
IserviceInfoCmd
ISCardError

Базовые интерфейсы

IbyteBuffer
ISCard
ISCardCmd
ISCardDatabase
ISCardISO7816

IByteBuffer

Стандартный интерфейс IByteBuffer предназначен для чтения, записи, копирования потоков данных. Внутри него происходят операции с байтовым массивом. Он используется почти всеми другими интерфейсами и является оберткой для стандартного интерфейса IStream. Поэтому, несмотря на то, что он описывается в MSDN, далее приводится описание его наиболее важных методов.

Для удобства определены следующие типы:

```
typedef IByteBuffer* LPBYTEBUFFER;

typedef const IByteBuffer *LPCBYTEBUFFER;
```

Seek pointer – это "текущий указатель" объекта IbyteBuffer. Его можно перемещать по всему потоку и за его конец для проведения записи или чтения. Если seek pointer находится выше конца потока, то при записи N байт в поток его размер сначала увеличивается до seek pointer, а потом – на N байт, в которые и производится запись.

Методы интерфейса IByteBuffer:

```
HRESULT IByteBuffer::Initialize(LONG lSize,
BYTE* pData,);
```

Вызывается после создания объекта IByteBuffer для его инициализации.

lSize	/*= 1*/ - длина буфера, по умолчанию 1, может быть 0

pData	/*= NULL*/ - сам буфер [in] (если нужно не инициализировать объект конкретными данными, а просто выделить память для потока, то pData = NULL)

HRESULT IByteBuffer::Seek(LONG dLibMove, LONG dwOrigin, LONG* pLibnewPosition);

Перемещение seek pointer на dLibMove байт, возвращая новое значение seek pointer в pLibnewPosition.

dLibMove	На сколько байтов переместить указатель
dwOrigin	STREAM_SEEK_CUR - перемещение от текущей позиции STREAM_SEEK_SET - перемещение от начала потока STREAM_SEEK_END - перемещение от конца потока
pLibnewPosition	новое значение Seek pointer, смещение от начала потока

Текущее значение Seek pointer возвращается в pLibnewPosition, при задании параметров:

dLibMove = 0;

dwOrigin = STREAM_SEEK_CUR;

HRESULT IByteBuffer::Read(BYTE* pByte, LONG cb, LONG* pcbRead);

Чтение из потока cb байт, начиная с текущей позиции в pByte.

pByte	Массив-приемник
cb	Сколько байт читать
pcbRead	Сколько байт удалось считать из потока в pByte. Если это не интересно, то NULL

Если seek pointer, к примеру, в конце, то по адресу pcbRead будет возвращен 0, но при этом ошибки в коде возврата (HRESULT) не будет.

HRESULT IByteBuffer::Write(BYTE* pByte, LONG cb, LONG* pcbWritten)

Запись в поток cb байт из массива pByte.

pByte	Массив-источник
cb	Сколько байт копировать
pcbWritten	Сколько байт удалось скопировать

Если размер потока мал для записи, то происходит автоматическое увеличение его размера. Т.е. если, например seek pointer находится в самом конце, а мы хотим записать в поток N байт, то при вызове метода Write размер потока увеличится на N байт.

HRESULT IByteBuffer::SetSize(LONG libNewSize)

Изменение длины потокового объекта. При этом текущий указатель сохраняется, даже если он находится за концом потока.

libNewSize	Новый размер буфера в байтах

HRESULT IByteBuffer::Clone(LPBYTEBUFFER* ppByteBuffer);

Создание еще одного объекта, с таким же seek pointer, который будет соответствовать тому же массиву, что и уже существующий объект (т.е. они будут разделять один буфер памяти).

ppByteBuffer	Указатель на вновь созданный объект

HRESULT IByteBuffer::CopyTo(LPBYTEBUFFER * ppByteBuffer, LONG cb, LONG* pcbRead, LONG* pcbWritten);

Копирование cb байт из текущей позиции потока в поток, на который указывает *ppByteBuffer.

ppByteBuffer	Поток-приемник
cb	Сколько байт копировать
pcbRead	Сколько байт удалось прочитать из источника
pcbWritten	Сколько байт удалось записать в приемник

Если ***ppByteBuffer = NULL**, то создается новый объект, указатель на который помещается в ***ppByteBuffer**, и в него копируются данные.

ISCard

Стандартный интерфейс используется для создания и управления соединением с ruToken и для выполнения команд (можно использовать и с другими токенами или смарт-картами, которые поддерживают MS Resource Manager).

При создании автоматически создает контекст SCARDCONTEXT. Использует SCard* функции. Для создания соединения к определенному ридеру надо вызвать

```
ISCard::AttachByReader( BSTR bstrReaderName,
                        SCARD_SHARE_MODES ShareMode,
                        SCARD_PROTOCOLS PrefProtocol);
```

Для завершения соединения – ISCard::Detach()

ISCard::Transaction(LPSCARDCMD *ppCmd)

*ppCmd	Указатель на объект ISCardCmd, который предназначен для формирования APDU–команды

Используется для выполнения APDU-команд, внутри себя вызывает ScardTransmit.

Все методы ISCard описаны в MSDN.

ISCardCMD

Класс для создания APDU-команды. Используется в ISCard. Все методы предназначены для различных вариантов формирования APDU-команды.

1 вариант – APDU-команда формируется сразу в одной функции:

HRESULT ISCardCmd::BuildCmd(

BYTE byClassId,

BYTE byInsId,

BYTE byP1,

BYTE byP2,

IByteBuffer* pbyData,

ong* plLe)

byClassId	Класс APDU-команды
byInsId	APDU-команда
byP1	Параметр P1
byP2	Параметр P2
pbyData	поле данных
plLe	Le – число байт, ожидаемое в APDU-ответе. Если поля Le нет, то plLe = NULL. Если ожидается 256байт, то ***plLe** = 256 или 0.

Если функция отработала успешно, то объект можно использовать для выполнения данной APDU-команды в ICard::Transaction.

2-й вариант – APDU-команда формируется по частям.

HRESULT ISCardCmd::put_ClassID(BYTE bClassID);

HRESULT ISCardCmd:: put_InstructionId(BYTE pbyIns);

и так далее, пока не сформируется вся APDU-команда.

3-й вариант используется, когда уже есть сформированная APDU-команда в IByteBuffer:

HRESULT ICardCmd::put_Apdu(IByteBuffer* ppApdu)

ppApdu	Буфер, содержащий APDU-команду

ISCardISO7816

Методы стандартного интерфейса формируют часть APDU-команд ISO 7816.

Иногда проще вызвать один из методов ISCardISO7816, чем использовать для этой цели методы ISCardCmd. Результат выдается в виде объекта ISCardCmd, в котором лежит нужная APDU-команда и который можно использовать в ISCard.

В данной версии **не используются** следующие методы интерфейса:

AppendRecord

EraseBinary

ExternalAuthenticate

InternalAuthenticate

ManageChannel

ReadRecord

UpdateRecord

WriteBinary

WriteRecord

Все они возвращают E_NOTIMPL.

ISCardDatabase

Стандартный интерфейс используется для различных операций с базой данных Resource Manager:

перечисление имен смарт-карт, соответствующим определенным интерфейсам (GUID) или ATR-строкам (и наоборот, – пе-

речисление интерфейсов, соответствующих данной смарт-карте), перечисление зарегистрированных в системе ридеров, а также групп ридеров.

Высокоуровневые интерфейсы

ISCardManage

ISCardError

ISCardFileAccess

ISCardObjectAccess

ISCardVerify

IsecOperation

IserviceInfoCmd

ISCardManage

Для работы с любым из высокоуровневых интерфейсов необходимо сначала создать объект ISCardManage, вызвать его метод AttachByIFD или AttachByHandle, а затем уже создавать любое количество соответствующих COM-объектов. Для создания COM-объектов вызывается метод CreateInterface. В его параметрах задается название интерфейса или его IID. При этом CreateInterface возвращает указатель на IUnknown, после чего указатель на искомый интерфейс получается при помощи метода QueryInterface. При создании COM-объектов, ISCardManage инициализирует их HSCARDHANDLE хэндлом соединения, которое он установил, в результате чего объектам не нужно устанавливать соединение с ридером, вызывая функцию Resource Manager'a SCardConnect.

ISCardError

Интерфейс расширения выполняет преобразование и расшифровку кодов ошибок.

```
ISCardError::Status2Hresult(WORD, HRESULT*)
```

Переводит поля значения SW1SW2 в HRESULT. В ruToken ICCSP определен следующий формат:

- Если произошла ошибка, старшее слово HRESULT будет равно 0xFFFF. Например, если токен вернул значение 0x1234, соответствующий HRESULT будет равен 0xFFFF1234.

- Если ошибки нет, а получено просто уведомление, то старшее слово HRESULT будет равно 0x7FFF. Т.е. макрос FAILED(0x7FFFxxxx) вернет FALSE.

ISCardError::Status2Text(WORD wStatus, BSTR* pbstrText, LANGID langID)

wStatus	Код ошибки
pbstrText	Буфер, в который возвращается строка, соответствующая заданному коду ошибки
LANGID	Язык сообщений: LANG_RUSSIAN - русский LANG_ENGLISH - английский

Метод использует динамическую библиотеку, в ресурсах которой хранятся соответствия SW1SW2 -> строка_поясняющая_ошибку.

ISecOperation

Интрфейс расширения позволяет настроить текущее окружение безопасности и выполнить security-ориентированную операцию с использованием алгоритма ГОСТ 28147-89.

ISecOperation::ManageSecEnvironment(BYTE bID,
CSE_CREATE_TYPE cseCreateType,
CSE_COMP cseComp)

Устанавливка текущего окружения безопасности (CSE). После установки становится возможным зашифровывать, расшифровывать, вычислять имитовставку.

bID	ID объекта данных
cseCreateType	Определяет режим установки CSE: **RESTORE** означает загрузку всех компонент CSE из указанного SE-объекта (bID – ID SE-объекта). Последний параметр игнорируется. **SET** означает прямую установку указанного в cseComp компонента CSE.
cseComp	Компонента окружения безопасности: • CSE_COMP_A - A-компонента; bID - ID GCHV-объекта • CSE_COMP_C - C-компонента; bID - ID GOST-объекта • CSE_COMP_H - H-компонента; bID – ID GOST-объекта

ISecOperation::Encipher(BYTE byKEY_ID, IByte-Buffer *pData)

Зашифрование данных по алгоритму ГОСТ 28147-89.

byKEY_ID	ID GOST-объекта, на котором производится зашифрование. Если byKEY_ID = 0, ID GOST-объекта берется из C-компоненты текущего окружения безопасности (CSE). В противном случае C-компоненте CSE присваивается значение byKEY_ID.
pData	Буфер данных для зашифрования. Зашифрованные данные сохраняются в том же буфере.

Если GOST-объект реализует шифрование в режиме простой замены, объем данных для зашифрования должен быть кратен 8, иначе в последних байтах будет получена некорректная последовательность.

Если GOST-объект реализует шифрование в режиме гаммирования или гаммирования с обратной связью, в первые 8 байт буфера pData должна быть помещена синхропосылка, а затем - данные для зашифрования. После зашифрования синхропосылка сохраняется и остается там же, за ней следуют зашифрованные данные.

ISecOperation::Decipher(BYTE byKEY_ID, IByte-Buffer *pData)

Расшифрование данных по алгоритму ГОСТ 28147-89.

byKEY_ID	ID GOST-объекта, на котором производится расшифрование. Если byKEY_ID = 0, ID GOST-объекта берется из C-компоненты текущего окружения безопасности (CSE). В противном случае C-компоненте CSE присваивается значение byKEY_ID.
pData	Буфер данных для расшифрования. Расшифрованные данные сохраняются в том же буфере.

Если GOST-объект реализует шифрование в режиме простой замены, объем данных для расшифрования должен быть кратен 8, иначе в последних байтах будет получена некорректная последовательность.

Если GOST-объект реализует шифрование в режиме гаммирования или гаммирования с обратной связью, в первые 8 байт

буфера pData должна быть помещена синхропосылка, а затем - данные для расшифрования. После расшифрования синхропосылка не возвращается, в pData сохраняются только расшифрованные данные.

ISecOperation::Hash(BYTE byKEY_ID, IByteBuffer *pData)

Выполнить операцию хеширования (вычисления имитовставки по ГОСТ 28147-89) данных, хранящихся в буфере pData.

byKEY_ID	ID GOST-объекта, на котором производится вычисление имитовставки. Если byKEY_ID = 0, ID GOST-объекта берется из H-компоненты текущего окружения безопасности (CSE). В противном случае H-компоненте CSE присваивается значение byKEY_ID.
pData	Буфер данных для вычисления имитовставки. В этом же буфере сохраняется и вычисленная имитовставка.

Результат вычисления возвращается в pData, при этом размер буфера становится равен длине имитовставки (4 байта).

ISCardObjectAccess

Интерфейс расширения выполняет создание, уничтожение, получение информации оо объектах данных (DO), а также создание security attributes и генерацию ключей шифрования.

```
ISCardObjectAccess::CreateDO(BYTE      byID,
DoType             doType,
DoFlags            doFlags,
DoOptions          doOptions,
LPBYTEBUFFER       pDOBody,
LPBYTEBUFFER       pSecAttr,
SCARD_FLAGS        Flags,
BYTE               bAccessRetryNum,
BYTE               bAccessRetryMaxNum)
```

Создать объект данных (DO) указанного типа.

В данной версии с помощью этой функции можно:

- Создать SE-объект.
- Импортировать ключ шифрования ГОСТ в создаваемый GOST-объект.

byID	ID создаваемого объекта. Если старший бит ID = 0, объект данных будет создан в предопределенной папке (здесь и далее см. "Объекты данных в ruToken" в разделе "Файловая система ruToken")
doType	Тип объекта данных: • T_SE – SE-объект. • T_GOST_KEY – GOST-объект.
doFlags	Формат объекта данных: • F_FMT_COMPACT – компактный DO, иначе – стандартный DO. • F_MSK_FMT – маска для вычленения формата DO • Степень открытости объекта данных: • F_ACS_CLOSED – закрытый DO • F_ACS_OPEN_LEN – DO с открытой длиной тела. • F_ACS_OPEN – Открытый DO. • F_MSK_ACS – маска для вычленения степени открытости DO.
doOptions	При создании SE: 0 При импортировании ключа в создаваемый GOST-объект задается режим шифрования, который будет реализовать этот объект: • O_GOST_MOD_PZ – простая замена • O_GOST_MOD_GAMM – гаммирование • O_GOST_MOD_GAMM_OS – гаммирование с обратной связью • O_MSK_GOST_MOD – маска для вычленения режима шифрования
pDOBody	• При создании SE-объекта – его тело (6 байт, см. "Создать объект данных" в разделе "APDU ruToken"). • При импортировании ключа в создаваемый GOST-объект – сам ключ ГОСТ (32 байта).
pSecAttr	IByteBuffer, содержащий атрибуты безопасности. Предопределенные атрибуты безопасности можно сформировать при помощи метода CreateSecAttributes.

Flags	В текущей версии не используется (по умолчанию равен SC_FL_NO_FLAGS).
bAccessRetryNum	В текущей версии не используется (по умолчанию 0)
bAccessRetryMaxNum	В текущей версии не используется (по умолчанию 0)

ISCardObjectAccess::CreateSecAttributes(SaType saType, IByteBuffer **ppSecAttr)

Сформировать предопределенные атрибуты безопасности для любых объектов файловой системы ruToken и вернуть их через буфер ppSecAttr.

saType	Тип создаваемого объекта файловой системы: • SAT_PUBLIC_EF – public файл • SAT_PRIVATE_EF – private файл • SAT_DEFAULT_DF – папка по умолчанию • SAT_DEFAULT_DO – объект данных по умолчанию • SAT_DEFAULT_KEY – ключ по умолчанию
ppSecAttr	Сформированные атрибуты безопасности.

Если указатель на IByteBuffer*, который передается в параметр равен NULL, то метод сам создает объект – буфер и возвращает на него указатель через параметр

PpSecAttr. Иначе считается, что передался указатель на готовый объект и атрибуты безопасности записываются в его начало, после чего размер буфера устанавливается равным длине атрибутов безопасности.

ISCardObjectAccess::DeleteDO(BYTE byID, SCARD_FLAGS Flags)

Удалить указанный объект данных (DO) в текущей папке.

byID	ID удаляемого DO.
Flags	В текущей версии не используется (по умолчанию равен SC_FL_NO_FLAGS).

ISCardObjectAccess::GenKey(BYTE byID,
DoOptions doOptions,
LONG lKeyLen,
DoFlags doFlags,

```
                                    LPBYTEBUFFER pSecAttr,
SCARD_FLAGS Flags)
```

Сгенерировать ключ шифрования ГОСТ и поместить его в создаваемый GOST-объект.

byID	ID создаваемого GOST-объекта, в который будет помещен ключ.
doOptions	Режим шифрования, который будет реализовывать создаваемый GOST-объект: • O_GOST_MOD_PZ – простая замена • O_GOST_MOD_GAMM – гаммирование • O_GOST_MOD_GAMM_OS – гаммирование с обратной связью
lKeyLen	длина ключав битах. Для алгоритма ГОСТ - всегда 256 (по умолчанию – 256).
doFlags	Степень открытости GOST-объекта: • F_ACS_CLOSED – закрытый DO • F_ACS_OPEN_LEN – DO с открытой длиной
pSecAttr	IByteBuffer, содержащий атрибуты безопасности. Предопределенные атрибуты безопасности можно сформировать при помощи метода CreateSecAttributes.
Flags	В текущей версии не используется (по умолчанию равен SC_FL_NO_FLAGS).

Местоположение создаваемого DO (предопределенная или текущая папка) определяется состоянием старшего бита его ID.

ISCardObjectAccess::GetDOInfo(DOGETINFO doInfo, BYTE byID, LPBYTEBUFFER* ppDOInfo, SCARD_FLAGS Flags)

Получить информацию об объекте данных (DO) в текущей папке.

doInfo	• SC_DO_GETINFO_THIS – получить информацию об объекте данных с указанным ID • SC_DO_GETINFO_FIRST – получить информацию о первом встреченном в текущей папке объекте данных • SC_DO_GETINFO_NEXT – получить информацию о следующем объекте данных после указанного Когда все DO в текущей папке будут перебраны, метод вернет значение, полностью совпадающее с предыдущем.
bID	ID объекта данных (актуально для

	SC_DO_GETINFO_THIS и SC_DO_GETINFO_NEXT)
ppDOInfo	• Если готовый IByteBuffer уже существует (так считается, если ***ppDOInfo** != NULL), то устанавливается определенный размер буфера (256 байт), в который записывается набор TLV – структур • Если IByteBuffer не существует, то он создается внутри метода и инициализируется набором TLV – структур с информацией об объекте
Flags	В текущей версии не используется (по умолчанию равен SC_FL_NO_FLAGS)

Набор TLV-структур аналогичен описанному в “Получить сведения об объекте данных” раздела “APDU ruToken”.

```
ISCardObjectAccess::CreateSE(BYTE byID,
                             BYTE byA,
                             BYTE byC,
                             BYTE byH,
                             DoFlags doFlags,
                             LPBYTEBUFFER pSecAttr,
SCARD_FLAGS Flags)
```

Создать SE-объект.

byID	ID создаваемого SE-объекта
byA	ID GCHV-объекта для A-компоненты создаваемого SE-объекта
byC	ID GOST-объекта для C-компоненты создаваемого SE-объекта
byH	ID GOST-объекта для H-компоненты создаваемого SE-объекта
doFlags	Формат SE-объекта: • F_FMT_COMPACT – компактный DO, иначе – стандартный DO • F_MSK_FMT – маска для вычленения формата DO Степень открытости SE-объекта: • F_ACS_CLOSED – закрытый DO • F_ACS_OPEN_LEN – DO с открытой длиной тела • F_ACS_OPEN – Открытый DO • F_MSK_ACS – маска для вычленения степени открытости DO
pSecAttr	IByteBuffer, содержащий атрибуты безопасности. Предопределенные атрибуты безопасности можно сформировать при помощи метода CreateSecAttributes

Flags	В текущей версии не используется (по умолчанию равен SC_FL_NO_FLAGS)

Местоположение создаваемого SE-объекта (предопределенная или текущая папка) определяется состоянием старшего бита его ID.

ISCardFileAccess

В данной версии **не используются** следующие методы интерфейса:

SetProperties

GetFileCapabilities

Invalidate

Rehabilitate

Все они вернут код ошибки E_NOTIMPL.

Остальные методы интерфейса используются так, как это описано в MSDN Library. Ниже дано описание методов, отличных от MSDN.

```
ISCardFileAccess::Create(REFTYPE refType,
        BSTR bstrPathSpec,
        LPTLV_TABLE* ppTLVs,
        LONG* lpcTLVs,
        SCARD_FLAGS Flags,
        LPBYTEBUFFER pDataBuffer);
```

Создать файл или папку в памяти ruToken.

refType	Должен быть SC_TYPE_BY_ID
bstrPathSpec	Если требуется создать файл или папку в текущей папке, bstrPathSpec должен содержать ID создаваемого файла / папки. При этом в TLV_TABLE структуре не обязательно должен присутствовать tag с ID. Если же необходимо создать в другой папке, bstrPathSpec должен содержать путь к папке (без разделителей) + ID создаваемого файла/папки. Путь можно указать двумя способами: От Корневой папки (в начале пути – 3F00). От текущей папки (в начале пути нет 3F00' ID текущей папки в путь не включается)

ppTLVs	Структура TLV_TABLE включает в себя: LONG cbSize – длина последовательности TLV – структур. BYTE byData* – указатель на последовательность TLV – структур.
IpcTLVs	В текущей версии не используется
Flags	SC_FL_DONT_SAVE_CURRENT_DIR – после создании файла / папки не в текущей папке, текущей становится папка, в которой был создан новый файл. SC_FL_NO_FLAGS – после создания файла / папки не в текущей папке, исходная текущая папка восстанавливается.
pDataBuffer	пока не используется

IServiceInfoCmd

Интерфейс расширения содержит методы, позволяющие получить информацию о ruToken.

IServiceInfoCmd::GetRtFreeFileMemory(LONG *plVol)

Возвращает объем свободной файловой памяти (т.е. суммарную длину всех свободных секторов памяти).

IServiceInfoCmd::GetRuTokenID(LONG *plID)

Возвращает по адресу plID 4-байтовый уникальный серийный номер (ID) ruToken.

IServiceInfoCmd::ResetAndGetAtr(LPBYTEBUFFER *ppAtr)

Позволяет выполнить “горячий” сброс токена в его начальное состояние и возвращает ATR-строку в формате ISO 7816-3, предназначенную для идентификации токена стандартными средствами.

IServiceInfoCmd::GetIdentificationData (LPBYTE-BUFFER *ppData)

Возвращает набор параметров, идентифицирующих данный ruToken: его тип, аппаратную версию, общий объем EEPROM-памяти, №протокола обмена с токеном, №микропрограммы, прошитой в токен, №заказа токена.

Более подробно – см в "Получить идентификационные данные ruToken" в разделе "APDU ruToken ".

ISCardVerify

Стандартный интерфейс содержит методы для аутентификации и работы с PIN – кодами.

```
ISCardVerify::Verify(LPBYTEBUFFER pCode,
                     SCARD_FLAGS Flags,
                     LONG* lpRef)
```

Выполнить аутентификацию владельца токена (Login) путем проверки переданного PIN-кода и установить текущие права доступа.

pCode	PIN – код (ANSI-строка без конечного 0-символа).
Flags	Должен быть SC_FL_IHV_GLOBAL.
*lpRef	ID GCHV-объекта, используемого для аутентификации: GCHV_ADMIN_ID – Login с правами Администратора. GCHV_USER_ID – Login с правами Пользователя.

```
ISCardVerify::ResetSecurityState(SCARD_FLAGS
Flags)
```

Выполнить сброс текущих прав доступа (т.е. текущими становятся права доступа "Гость").

Flags	Должен быть SC_FL_IHV_GLOBAL.

После выполнения команды владелец токена сможет выполнять только те операции, для которых не требуется аутентификация.

```
ISCardVerify::ChangeCode(
                         LPBYTEBUFFER pOldCode,
                         LPBYTEBUFFER pNewCode,
                         SCARD_FLAGS Flags,
                         LONG lRef)
```

Сменить PIN-код Пользователя или Администратора.

pOldCode	В текущей версии не используется.
pNewCode	Новый PIN-код.
Flags	Должен быть SC_FL_IHV_GLOBAL
lRef	ID GCHV-объекта, PIN-код в котором надо изменить: • GCHV_ADMIN_ID - Администратора • GCHV_USER_ID – Пользователя

Для выполнения команды необходимы права Администратора.

```
ISCardVerify::Unblock(LONG lData,
                      SCARD_FLAGS Flags,
                      LONG lRef)
```

Установить счетчик попыток доступа к GCHV-объекту в его исходное, максимальное значение (т.е. разблокировать объект GCHV-объект).

lData	В текущей версии не используется
lRef	GCHV_USER_ID
Flags	Должен быть SC_FL_IHV_GLOBAL

В действительности команда может разблокировать только объект, предоставляющий права доступа "Пользователь" (ID = GCHV_USER_ID). *Объект с ID = GCHV_ADMIN_ID (права доступа "Администратор") не подлежит разблокированию.*

Для выполнения команды необходимы права Администратора.

ISCardAuth

В данной версии реализован единственный метод стандартного интерфейса:

```
ISCardAuth::GetChallenge(LONG lAlgoID,
                         LPBYTEBUFFER pParam,
                         LONG *lpLengthOfChallenge,
                         LPBYTEBUFFER *ppBuffer)
```

Получить challenge, которое можно использовать как случайное число.

lAlgoID	В текущей версии не используется (должен быть равен 0)
pParam	В текущей версии не используется (должен

	быть равен 0)
lpLengthOfChallenge	В текущей версии значение, на которое указывает lpLengthOfChallenge, должно быть равным 32
ppBuffer	Параметр, в который передается указатель на буфер, содержащий challenge

Криптографический сервис-провайдер ruToken и поддержка MS CAPI

Microsoft Cryptography API (MS CAPI) – это стандартная библиотека, разработанная Microsoft для поддержки таких наиболее распространенных криптографических функций, как зашифрование/расшифрование сообщений, управление цифровыми сертификатами, аутентификация сообщений и т.д.

Уровнем ниже, чем MS CAPI, находится Криптографический сервис-провайдер (Cryptographic Service Provider, CSP). CSP обеспечивает реальное воплощение криптографических возможностей MS CAPI и, по возможности, служит каналом для сообщения непосредственно с устройством, в данном случае с ruToken. CSP берет на себя выполнение криптографических алгоритмов, генерацию и безопасное хранение криптографических ключей. При этом приложение вызывает функции MS CAPI, которые, в свою очередь, используют соответствующие функции CSP. Это непосредственно связано с проблемами безопасности. Каждый CSP имеет подпись, полученную непосредственно в Microsoft, которая при использовании CAPI неявно проверяется системой.

Все функции MS CAPI полностью документированы в Microsoft Developer's Network Library (MSDN Library). Поэтому в данном Руководстве они не описываются, а за информацией и рекомендациями по использованию MS CAPI следует обращаться к MSDN Library.

Криптографический сервис-провайдер ruToken (далее - криптопровайдер) поддерживает MS CAPI версии 1.0 и старше. Он представляет собой динамическую библиотеку (dll), предоставляющую стандартный набор из 25 функций.

В MS CAPI используются следующие термины:

Сессионный ключ – симметричный ключ.

Блоб или BLOB (binary large object) – Байтовый массив, содержащий ключ (иногда зашифрованный). Он нужен для обмена

ключами между провайдерами. Поддерживаются следующие типы блобов:

- SIMPLEBLOB – для сессионных ключей
- PUBLICKEYBLOB – public-ключ
- PRIVATEKEYBLOB – пара ключей: public-ключ и private-ключ

Ключевой контейнер (key container)

Используется для хранения асимметричных пар. Криптопровайдер ruToken для этой цели использует устройство ruToken. В одном контейнере может находиться максимум две пары (одна для подписи (AT_SIGNATURE) , другая – для обмена ключами и подписи (AT_KEYEXCHANGE)) Контейнер имеет имя, уникальное для данного криптопровайдера (например "MyContainer"). В контейнере нельзя храненить сессионные ключи.

Экспорт ключа

Сохранить сесионный ключ можно только вне криптопровайдера. Это делается при помощи экспорта его блоба (SIMPLEBLOB) и записи последнего, к примеру, на ruToken (не через Crypto API) или на дискету.

В чистом виде сессионый ключ не экспортируется. Для экспорта надо использовать public-ключ, которым криптопровайдер зашифрует симметричный. В результате получится зашифрованный симметричный ключ в блобе. Расшифровать и использовать ключ может только владелец private-ключа. (в данном случае – только владелец ruToken, в памяти которого он хранится).

Экспортировать можно также открытый ключ (PUBLICKEYBLOB) или же всю пару открытый/закрытый ключ (PRIVATEKEYBLOB) (последнее не поддерживается криптопровайдером ruToken).

Импорт ключа

В криптопровайдер можно импортировать:

- Пару private/public (PRIVATEKEYBLOB). В результате пара оказывается в контейнере, т.е. сохраняется в криптопровайдере
- Открытый ключ (PUBLICKEYBLOB) . Не сохраняется в криптопровайдере и используется только в течение сессии

- Сессионный ключ (SIMPLEBLOB) . Импортируется только при наличии private-ключа, соответствующего public-ключу, на котором сессионный ключ был зашифрован. Т.е. при импорте в криптопровайдер автоматически происходит расшифрование

Экспортабельность ключа (флаг - CRYPT_EXPORTABLE)

Атрибут ключа. Его можно задать при генерации или импорте ключа.

Ключ с этим атрибутом можно экспортировать. Из пары *всегда* можно экспортировать открытый ключ.

Функциональность криптопровайдера главным образом состоит в следующем:

- Симметричное шифрование (в криптопровайдере ruToken - алгоритмы RC2, RC4, DES, TripleDES)
- Цифровая подпись (в криптопровайдере ruToken - алгоритм RSA)
- Асимметричное шифрование (в криптопровайдере ruToken шифруется симметричный ключ на открытом ключе RSA)
- Управление ключами (генерация, хранение, экспорт, импорт ключей)
- Хеширование

Список функций криптопровайдера ruToken

Функция API	**Описание функции API**	**Особенности реализации функции в криптопровайдере ruToken**
Функции криптопровайдера		
CPAcquireContext	Получение хендла указанного ключевого контейнера	Флаг CRYPT_MACHINE_KEYSET не поддерживается
CPGetProvParam	Получение информации о CSP	Флаг PP_KEYSET_SEC_DESCR не поддерживается
CPReleaseContext	Освобождение контекста, созданного CPAcquireContext	Флаг CRYPT_USER_PROTECTED не поддерживается
CPSetProvParam	Настройка параметров CSP	Флаг PP_KEYSET_SEC_DESCR не поддерживается
Функции генерации и обмена ключами		
CPDeriveKey	Генерация криптографических сессионных	Флаги CRYPT_USER_PROTECTED,

Функция API	Описание функции API	Особенности реализации функции в криптопровайдере ruToken
	сессионных ключей на основе базовых данных. Гарантируется, что все ключи, сгенерированные на основе одних и тех же базовых данных будут идентичны за счет использования одного и того же алгоритма генерации.	CRYPT_UPDATE_KEY не поддерживается
CPDestroyKey	Освобождает хендл, на который сслается параметр hKey. После освобождения хендл становится непригодным к повторному использованию	Пара ключей не удаляется из памяти токена
CPDuplicateKey	Создает точную копию объекта ключ.	
CPExportKey	Экспортирует криптографические ключи из CSP безопасным способом	Private-ключ не экспортируется
CPGenKey	Генерирует случайные криптографические ключи	Flag CRYPT_USER_PROTECTED is not supported. Will overwrite RSA key pair
CPGenRandom	Заполняет буфер случайными байтами	
CPGetKeyParam	Получает управляющие параметры ключа	
CPGetUserKey	Получает хендл на одну из постоянных пар ключей в ключевом контейнере hProv	
CPImportKey	Импортирует криптографический ключ из ключевого блоба в CSP	
CPSetKeyParam	Настраивает параметры ключа	CRYPT_MODE_CFB режим шифрования не поддерживается. CRYPT_MODE_OFB режим шифрования не поддерживается. Флаг KP_PERMISSIONS не поддерживается.
Функции шифрования данных		
CPDecrypt	Расшифровывает данные,	

Функция API	Описание функции API	Особенности реализации функции в криптопровайдере ruToken
	которые предварительно были зашифрованы функцией CPEncrypt	
CPEncrypt	Зашифровывает данные	
Функции хеширования и цифровой подписи		
CPCreateHash	Инициирует хеширование потока данных	
CPDestroyHash	Уничтожает хеш-объект, на который ссылется параметр hHash	
CPDuplicateHash	Создает точную копию хеш-объекта	
CPGetHashParam	Получает управляющие параметры хеш-объекта. Значение хеша также получается через эту функцию	
CPHashData	Передает данные указанному хеш-объекту	
CPHashSessionKey	Передает криптографический ключ для хеширования	Флаг CRYPT_LITTLE_ENDIAN не поддерживается.
CPSetHashParam	Настраивает параметры хеш-объекта	
CPSignHash	Подписывает значение хеша	Флаг CRYPT_NOHASHOID не поддерживается.
CPVerifySignature	Проверяет цифровую подпись сравнением с хеш-объектом	Флаг CRYPT_NOHASHOID не поддерживается

Все функции MS CAPI используют следующую форму обработки ошибок:

В случае успеха возвращается TRUE, в случае неудачи – FALSE. Код ошибки получается при помощи функции GetLastError().

Использование криптопровайдера начинается с вызова следующей функции MS CAPI:

```
BOOL WINAPI CryptAcquireContext(
HCRYPTPROV *phProv, //[out] указатель на
// дескриптор CSP
LPCTSTR pszContainer, //[in] имя ключевого
// контейнера
```

```
LPCTSTR pszProvider, //[in] имя криптопровайдера
DWORD dwProvType, //[in] тип криптопровайдера
DWORD dwFlags //[in] флаги
);
```

Флаги могут принимать одно из следующих значений :

CRYPT_VERIFYCONTEXT	Приложению не требуется доступа к private-ключам. Флаг используется, когда приложению необходима проверка подписи или информация о криптопровайдере. Второй параметр – имя контейнера, должен в этом случае быть NULL.
CRYPT_NEWKEYSET	Создание нового контейнера с заданным именем (pszContainer). Если pszContainer = NULL, то создается контейнер с именем по умолчанию. В данном случае имя по умолчанию - это имя текущего пользователя операционной системы.
CRYPT_DELETEKEYSET	Уничтожение контейнера с заданным именем.
Если вышеописанные флаги отсутствуют, это означает, что запрашивается уже существующий контейнер. Если он не существует, то возвращается FALSE и GetLastError() вернет код ошибки NTE_BAD_KEYSET.	
CRYPT_SILENT	Этот флаг совместим со всеми предыдущими. Он запрещает использовать пользовательский интерфейс криптопровайдера (в данном случае – это окно для ввода PIN-кода и выбора ruToken) при вызове функций MS CAPI с *именно этим* дескриптором криптопровайдера. Поэтому при необходимости организации пользовательского интерфейса в какой–нибудь функции криптопровайдера возвращается FALSE. GetLastError() вернет NTE_SILENT_CONTEXT. Таким образом, сначала необходимо ввести PIN-код программно, при помощи функции CryptSetProvParam с параметром PP_KEYEXCHANGE_PIN и строкой, содержащей PIN-код. При проблеме выбора токена из нескольких доступных надо задавать имя ридера и имя контейнера в параметре pszContainer в следующем формате: \\.\Aktiv Co. ruToken 0\ ContainerName или \\.\Aktiv Co. ruToken 1\ ContainerName и тд.

Примечание
Не рекомендуется использовать так называемый контейнер по умолчанию (default container) (пример - "\\\\.\\Aktiv Co. ruToken 0\\" означает default container на токене в ридере 'Aktiv Co. ruToken 0') так как он может использоваться операционной системой.

Коды возврата Криптографического сервис-провайдера ruToken

Помимо стандартных кодов возврата Windows, GetLastError() в редких случаях может возвращать коды, несущие информацию об ошибках на уровне APDU. В этом случае старшее слово кода возврата содержит 0xFFFF, а младшее равно значению поля SW1|SW2 APDU-ответа.

Кроме того, Криптографический сервис-провайдер ruToken может возвращать следующие специфические ошибки, связанные с map-файлом (файл с ID 0x0002, лежащий в папке 3F00\0000\0000\0002\ и содержащий информацию об объектах Crypto API):

Имя	Значение	Описание
RTCSP_STRUCT_ERR_BAD_MAPFILE_LABEL	0xFFFE0001	В файле 0x0002 отсутствует идентификационная метка map-файла
RTCSP_STRUCT_ERR_BAD_MAPFILE_STRUCT	0xFFFE0002	Неверная структура map-файла

Поддержка стандарта PKCS #11

Подобно PC/SC, обеспечивающему стандарт взаимодействия со смарт-картами, стандарт PKCS #11, разработанный RSA Laboratories, решает проблемы, относящиеся к криптографическим токенам. PKCS #11 – это один из стандартов криптографии с открытым ключом (Public Key Cryptography Standards, PKCS). Эти стандарты были разработаны для обеспечения совместимости между различными реализациями криптографии с открытым ключом, поставляемыми различными разработчиками устройств и программного обеспечения.

Стандарт PKCS #11, также известный под названием Cryptoki, распространяется на программирование криптографических аппаратных токенов, таких как ruToken. Он описывает интерфейс прикладного программирования (API) для устройств, которые несут в себе криптографическую информацию и способны выполнять криптографические преобразования. Стандарт

определяет общий набор команд для выполнения криптографических функций. Эти команды могут использоваться независимо от конкретной аппаратной реализации токена.

Для того чтобы ruToken можно было использовать с приложениями PKI, в состав Комплекта разработчика включена специальная библиотека. Перед тем как ее использовать, необходимо ознакомиться с основами стандарта PKCS #11, который доступен на сайте RSA Laboratories: http://www.rsasecurity.com/rsalabs/pkcs/pkcs-11/index.html. Документы, предоставляемые RSA Laboratories разработчикам, также находятся на диске, входящем в состав Комплекта разработчика.

Библиотека PKCS #11 ruToken реализует стандарт версии 2.10 и старше. Она реализована в виде динамически загружаемого модуля (.DLL), содержащего функции с интерфейсом вызова на языке C/C++.

Комплект разработчика содержит следующие файлы, относящиеся к библиотеке PKCS #11:

Lib\PKCS11\SetupPKCS.exe	Инсталлятор библиотеки PKCS #11 ruToken.
Lib\PKCS11\rtpkcs11.dll	Библиотека PKCS #11 ruToken
Include\PKCS11*.h	Заголовочные файлы

Для использования библиотеки PKCS #11 в исходные тексты программы необходимо включить заголовочный файл rtPKCS11.h, который, в свою очередь, автоматически подключит заголовочные файлы PKCS11f.h и PKCS11t.h.

Установка библиотеки PKCS #11 ruToken

Для установки библиотеки PKCS #11 служит утилита SetupPKCS.exe, имеющая оконный интерфейс. С ее помощью можно установить (переустановить) библиотеку PKCS #11 в системную папку Windows, а также удалить библиотеку из системы.

Порядок работы с утилитой аналогичен порядку работы с утилитой установки драйверов ruToken SetupDrv.exe (см. “Установка драйверов” раздела “Комплект разработчика”). Подобно этой утилите, SetupPKCS поддерживает пакетный режим из командной строки, ее можно использовать для установки библиотеки PKCS #11 из программы инсталляции ПО, использующего ruToken.

В процессе инсталляции Комплекта разработчика ruToken библиотека PKCS #11 устанавливается в систему автоматически.

Список поддерживаемых функций PKCS #11

Функции общего назначения	
C_Initialize	Инициализировать библиотеку PKCS #11
C_Finalize	Деинициализировать библиотеку
C_GetInfo	Получить информацию о библиотеке
C_GetFunctionList	Получить список всех функций в библиотеке
Функции для работы со слотами (USB-портами) и токенами:	
C_GetSlotList	Получить список слотов в системе
C_GetSlotInfo	Получить информацию о конкретном слоте
C_GetTokenInfo	Получить информацию о ruToken в конкретном слоте
C_WaitForSlotEvent	Ожидать события в любом слоте
C_GetMechanismList	Получить список механизмов, поддерживаемых ruToken
C_InitToken	Инициализировать память ruToken (прерогатива Администратора)
C_InitPIN	Установить PIN-код Пользователя ruToken (прерогатива Администратора)
C_SetPIN	Сменить PIN-код владельца токена, выполнившего Login (либо Пользователя, либо Офицера безопасности (Администратора); ruToken поддерживает эту функцию только для Администратора)
Функции для работы с сессиями:	
C_OpenSession	Открыть новую сессию с ruToken
C_CloseSession	Закрыть сессию
C_CloseAllSessions	Закрыть все сессии
C_GetSessionInfo	Получить информацию о конкретной сессии
C_Login	Выполнить Login (либо Пользователя, либо Офицера безопасности (Администратора))
C_Logout	Выполнить Logout
Функции для работы с объектами (ключами, сертификатами и т. п.):	
C_CreateObject	Создать новый объект
C_CopyObject	Копировать объект
C_DestroyObject	Уничтожить объект
C_GetObjectSize	Получить информацию о размере объекта
C_GetAttributeValue	Получить информацию об атрибутах объекта
C_SetAttributeValue	Изменить значение атрибута объекта
C_FindObjectsInit	Инициализировать процесс поиска объектов
C_FindObjects	Найти объекты по шаблону
C_FindObjectsFinal	Завершить процесс поиска объектов
Функции для зашифрования / расшифрования:	
C_EncryptInit	Инициализировать процесс зашифрования
C_Encrypt	Зашифровать и завершить процесс зашифрования

C_DecryptInit	Инициализировать процесс расшифрования
C_Decrypt	Расшифровать и завершить процесс расшифрования
Функции для подписи / верификации:	
C_SignInit	Инициализировать процесс подписывания
C_Sign	Подписать и завершить процесс подписывания
C_VerifyInit	Инициализировать процесс верификации
C_Verify	Верифицировать и завершить процесс верификации
Функции для работы с ключами:	
C_GenerateKey	Сгенерировать ключ на ruToken
C_GenerateKeyPair	Сгенерировать пару открытого / закрытого ключа
Функции генерации случайных чисел:	
C_SeedRandom	Задать инициализирующее значение для генерации случайных чисел
C_GenerateRandom	Сгенерировать случайное число

При этом поддерживаются следующие механизмы PKCS #11:

1. CKM_RSA_PKCS_KEY_PAIR_GEN (если установлен MS CryptoProvider).
2. CKM_RSA_PKCS (только зашифрование / расшифрование / подписывание / верификация; если установлен MS Enchanced CryptoProvider).
3. CKM_GOST_KEY_GEN - Механизм генерации ключей ГОСТ.
4. CKM_GOST - Механизм зашифрования / расшифрования по алгоритму ГОСТ 28147-89.

Список не поддерживаемых функций PKCS #11

Реализация стандарта PKCS#11 в библиотеке ruToken имеет некоторые ограничения, связанные с аппаратными характеристиками токена. Реализация PKCS#11 для рутокен не поддерживает некоторые функции, которые, тем не менее, остаются доступными, но при их вызове возвращается код CKR_FUNCTION_NOT_SUPPORTED.

Наличие подобных ограничений допускается стандартом PKCS #11.

Полный список функций, не поддерживаемых реализацией PKCS #11 для ruToken:

C_GetFunctionStatus
C_CancelFunction
C_WrapKey
C_UnwrapKey
C_DeriveKey
C_EncryptUpdate

C_EncryptFinal
C_DecryptUpdate
C_DecryptFinal
C_SignUpdate
C_SignFinal
C_SignRecoverInit
C_SignRecover
C_VerifyUpdate
C_VerifyFinal
C_VerifyRecoverInit
C_VerifyRecover
C_DigestEncryptUpdate
C_DecryptDigestUpdate
C_SignEncryptUpdate
C_DecryptVerifyUpdate
C_DigestInit
C_Digest
C_DigestUpdate
C_DigestKey
C_DigestFinal
C_GetOperationState
C_SetOperationState

Поддержка некоторых из перечисленных функций будет добавлена в будущих версиях ruToken.

Часть 4
Утилиты для работы с ruToken

Токены поставляются инициализированными, но не персонифицированными, то есть в них записаны PIN-коды по умолчанию и не содержится никакой значимой для пользователя информации. Для подготовки токена к использованию в реальных приложениях предназначены утилиты администрирования и редактирования.

rtAdmin. Утилита администрирования ruToken.

Утилита администрирования предназначена, в первую очередь, для использования администраторами систем безопасности или, иначе говоря, офицерами безопасности.

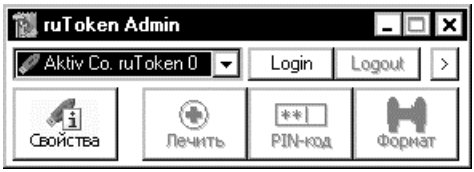
Утилита администрирования позволяет выполнять следующие операции:

- Получение информации о подключенных токенах.
- Разблокирование PIN-кода Пользователя.
- Установка новых PIN-кодов Пользователя и Администратора.
- Инициализация памяти токена.

Запуск утилиты

Для того чтобы начать работу с утилитой администрирования, нужно запустить приложение rtAdmin.exe.

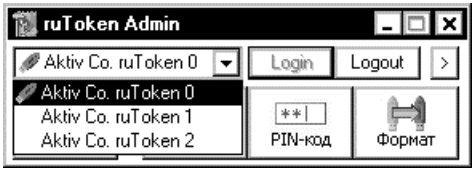
Перед запуском rtAdmin.exe убедитесь в том, что в системе установлен драйвер ruToken. Подключите токен токен к свободному USB-порту, если он не подкючен. Также это можно сделать позже. При успешном запуске утилиты появится ее главное окно:



Главное окно содержит элементы управления основными функциями. Неактивные элементы будут недоступны до выполнения операции Login и ввода PIN-кода Администратора.

Выбор ридера

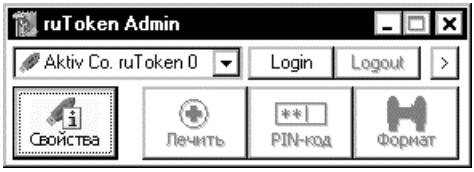
Для выполнения операций непосредственно с токеном прежде всего необходимо выбрать текущий ридер с подключенным к нему токеном. Выбор ридера осуществляется из выпадающего списка, содержащего имена ридеров. Ридеры, к которым подключены токены, отмечены в списке пиктограммой. Для выбора ридера нажмите на соответствующий элемент списка:



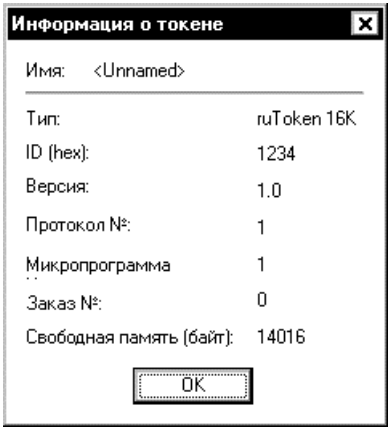
Получение информации о токене

Эта операция доступна в том случае, когда к текущему ридеру подключен токен. Для ее выполнения также не требуется ввод PIN-кода Администратора, поскольку выдается открытая (не секретная) информация и для ее получения достаточно прав доступа "Гость".

Для получения информации о токене нужно нажать на кнопку **[Свойства]:**



После нажатия появится следующее информационное окно:



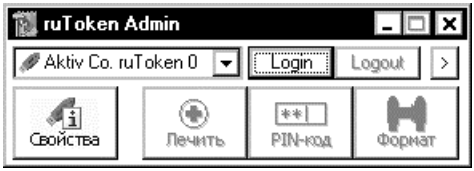
В нем отображается следующая информация:

- Символьное имя токена (Имя)
- Тип токена (Тип), включая общий объем памяти токена в килобайтах.
- ID токена в виде шестнадцатиричного числа (ID)
- Версия ruToken (Версия)
- Номер протокола обмена с драйвером ruToken (Протокол №
- Номер микропрограммы, прошитой в токен (Микропрограмма №
- Номер заказа (Заказ №
- Доступная память токена в байтах (Свободная память)

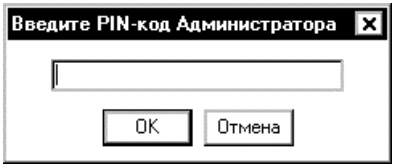
Операции Login и Logout

Для выполнения операций разблокирования токена, изменения PIN-кода и инициализации памяти нужно иметь права доступа

Администратора. Для установки текущих прав доступа выполняется операция Login.



После нажатия на кнопку **[Login]** отображается диалог для ввода PIN-кода Администратора:



Набираемый PIN-код отображается в виде символов '*'. Всего может быть предпринято 15 до попыток ввода PIN-кода Администратора. **По исчерпании этого количества попыток PIN-код Администратора блокируется и становится невозможным выполнять любые действия, требующие прав Администратора.** Для восстановления полной работоспособности такой токен следует направлять поставщику (в процессе восстановления токена все данные на нем будут уничтожены).

В случае ввода правильного PIN-кода активизируются ранее недоступные операции, что отображается в главном окне:

После завершения операций с токеном следует выполнить сброс текущих прав доступа (Logout), для чего нужно нажать на соответствующую кнопку. **Если Logout не выполнен, по завершении работы утилиты текущими остаются права доступа Администратора.**

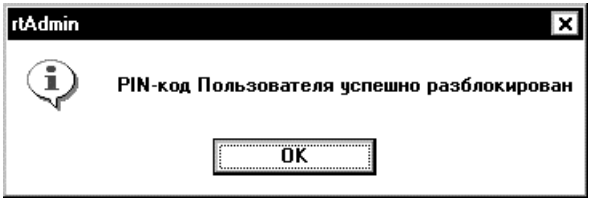
Разблокирование PIN-кода Пользователя

Разблокирование PIN-кода Пользователя ruToken выполняется в тех случаях, когда он был заблокирован после 15 последовательных неудачных попыток установки прав Пользователя (т. е.

если PIN-код Пользователя был введен неправильно 15 раз подряд).

Для его разблокирования нужно нажать на кнопку **[Лечить]**:

При выполнении этой операции счетчик попыток доступа к соответствующему GCHV-объекту восстанавливается в свое исходное, максимальное значение. В случае корректного завершения операции отображается следующее окно:



Смена PIN-кода

По умолчанию для PIN-кода Пользователя и PIN-кода Администратора установлены значения '12345678' и '87654321' соответственно.

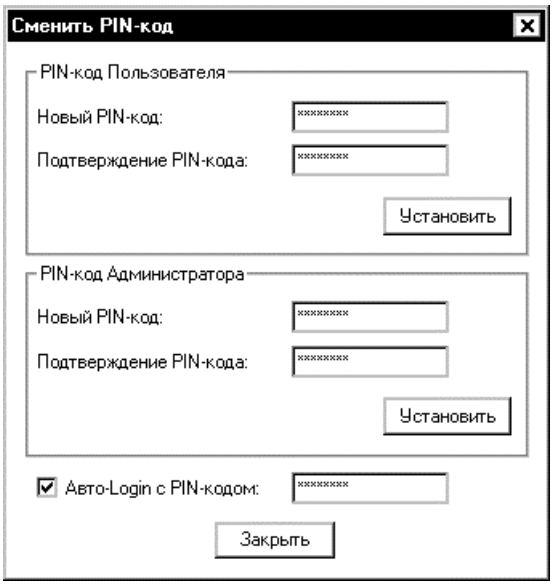
При работе с ruToken значения PIN-кода по умолчанию **необходимо** изменить на собственные в целях обеспечения безопасности. Установка новых значений PIN-кодов производится после нажатия на кнопку **[PIN-код]**:

Диалог установки новых значений PIN состоит из трех групп:

- PIN-код Пользователя – группа предназначена для установки нового PIN-кода Пользователя
- PIN-код Администратора – группа предназначена для установки нового PIN-кода Администратора

- Авто-Login с PIN-кодом – опция предназначена для изменения PIN-кодов у партии токенов в пакетном режиме. При подсоединении очередного токена операция Login будет выполнена автоматически с использованием введенного здесь PIN-кода.

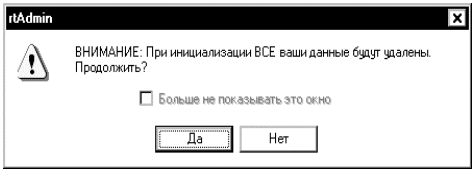
Установка PIN-кода Пользователя и Администратора производится по отдельности. Для установки каждого из них необходимо нажать кнопку **[Установить]** в соответствующей группе. Максимальная длина PIN-кода – 16 символов, регистр учитывается.



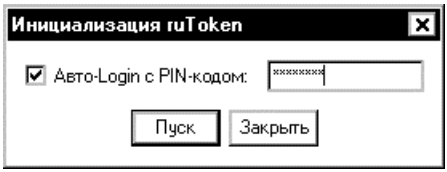
Инициализация памяти токена

Инициализация памяти предназначена в основном для того, чтобы удалить содержимое его памяти – например, при передаче токена другому владельцу. После выполнения инициализации в памяти токена содержится только дерево предопределенных папок со служебными файлами, а также GCHV-объекты с PIN-кодами Пользователя и Администратора по умолчанию.

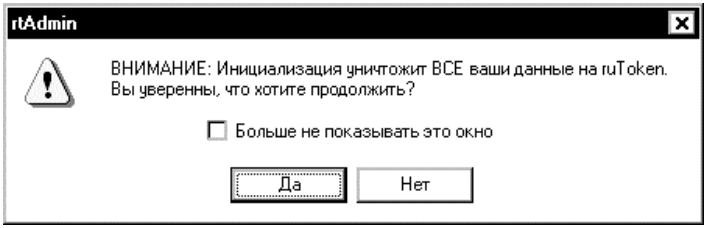
Для запуска инициализации нужно нажать на кнопку **[Формат]**:



Перед началом инициализации отображается диалог, в котором можно ввести PIN-код Администратора для того, чтобы отформатировать партию токенов в пакетном режиме (аналогично рассмотренному выше для функции смены PIN-кодов).



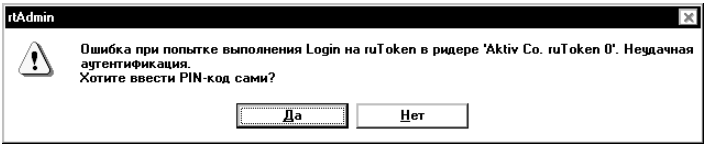
После нажатия на кнопку **[Пуск]** выводится предупреждение о том, что все данные в ruToken при инициализации будут уничтожены:



Если отметить галочку 'Больше не показвать это окно', то перед следующей операцией инициализации в пакетном режиме окно предупреждения показано не будет.

После подтверждения запускается сам процесс инициализации, прерывать который нельзя:

При вводе неверного PIN-кода в строке “Авто-Login с PIN-кодом:” будет выдано сообщение об ошибке:



rtEditor. Редактор памяти ruToken.

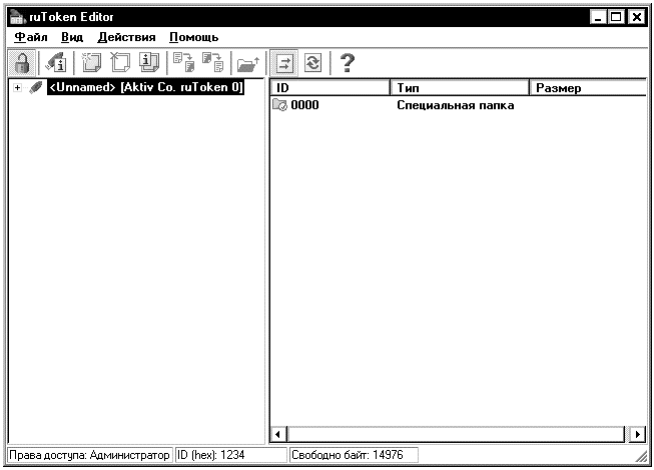
Утилиту rtAdmin могут использовать как Администратор, так и Пользователь ruToken. Она позволяет просматривать содержимое памяти токена, создавать и редактировать объекты файловой системы, определять их свойства, создавать символьное имя токена, производить шифрование данных по алгоритму ГОСТ 28147-89, выполнять другие сервисные действия.

Запуск утилиты

Для того чтобы начать работу с редактором памяти токена, нужно запустить приложение rtEditor.exe.

Перед запуском rtEditor.exe убедитесь в том, что в системе установлен драйвер ruToken и, по крайней мере, один токен подключен к одному из доступных портов USB. Подключение можно будет выполнить также после запуска утилиты.

При успешном запуске редактора появится его главное окно rtEditor:



В главном окне отображаются следующие элементы управления:

- Меню
- Панель инструментов
- Окно просмотра
- Панель состояния

Меню редактора имеет следующую структуру:

Файл			
	Создать		
		Файл	Создать новый файл.
		Папку	Создать новую папку.
		Ключ	Создать ГОСТ-объект (ключ шифрования ГОСТ).
		SE	Создать SE-объект (окружение безопасности).
	Открыть		Открыть файл для редактирования или зайти в папку.
	Удалить		Удалить объект файловой системы.
	Свойства		Получить информацию об объекте файловой системы.
	Выход		Закрыть приложение.
Вид			
	Обновить		Заново считать структуру папок.
	Использовать кеш		Использовать кэш (вкл/выкл).
	Отображать системную папку		Отображать системную папку (вкл/выкл).
	Панель инструментов		Отображать панель инструментов (вкл/выкл).
	Панель состояния		Отображать панель состояния (вкл/выкл).
Действия			

	Login	Выполнить аутентификацию владельца токена (Login).
	Logout	Выполнить сброс текущих прав доступа (Logout).
	Зашифровать	Зашифровать данные по ГОСТ 28147-89.
	Расшифровать	Расшифровать данные по ГОСТ 28147-89.
	Сменить имя токена	Изменить символьное имя токена.
	Информация о токене	Получить информацию о токене.
Помощь		
	О программе	Получить информацию о приложении.

Панель инструментов дублирует следующие функции меню:

	Действия|Login/Logout
	Действия|Информация о токене
	Файл|Создать
	Файл|Удалить
	Файл|Свойства
	Действия|Зашифровать
	Действия|Расшифровать
	View|Использовать кеш
	View|Обновить
	Помощь|О программе

Окно просмотра разделено на два фрейма:

- В левом фрейме отображается список доступных токенов и структура папок каждого из них. Каждый доступный токен обозначается в списке пиктограммой, символьным именем токена (если оно отсутствует, токен маркируется как <Unnamed>) и именем ридера. Структура папок отображается в виде дерева
- В правом фрейме отображается содержимое текущей папки: вложенные папки, файлы и объекты данных с указанием ID, типа и размера объекта файловой системы.

В зависимости от установок и атрибутов безопасности папки, файлы и объекты данных обозначаются пиктограммами:

В зависимости от установок и атрибутов безопасности папки, файлы и объекты данных обозначаются пиктограммами:

	Папка с атрибутами безопасности по умолчанию ("Папка")
	Папка с измененными атрибутами безопасности ("Системная папка")
	Папка с измененными атрибутами безопасности ("Специальная папка")
	Файл с атрибутами безопасности по умолчанию ("Общий файл")
	Файл с атрибутами безопасности по умолчанию ("Личный файл")
	Файл с измененными атрибутами безопасности ("Специальный файл")
	Объект данных ("GOST-объект", "SE-объект", "GCHV-объект")

В панели состояния отображается следующая информация:

- Текущие права доступа: Гость, Пользователь, Администратор.
- ID токена в 16-ном формате (ID (hex)).
- Доступный объем памяти токена в байтах (Свободно байт).

Выбор текущего токена

Список доступных токенов отображается в левом фрейме окна просмотра. Для установки текущего токена нужно выбрать его в списке и щелкнуть левой кнопкой мыши на его имени.

Операции Login и Logout

После выбора текущего токена текущие права доступа установлены в нем на уровне "Гость".

Для выполнения операций и доступа к файлам и объектам, требующим прав доступа Пользователя или Администратора нужно выполнить аутентификацию владельца токена с соответствующими правами. Для этого служит команда меню Действия|Login или кнопка в панели инструментов.

В диалоге требуется выбрать уровень прав доступа из выпадающего списка и ввести соответствующий PIN-код.

Регистрация будет успешно завершена, если введен правильный PIN-код. Для ввода PIN-кода дается до 15 последовательных попыток. По исчерпании числа попыток доступа:

- PIN-код Пользователя блокируется и может быть разблокирован только Администратором при помощи утилиты администрирования токена rtAdmin.
- PIN-код Администратора **блокируется без возможности его разблокирования и становится невозможным выполнять любые действия, требующие прав Администратора**. Такой токен подлежит возврату поставщику для восстановления его полной работоспособности (в процессе восстановления токена все данные на нем будут уничтожены).

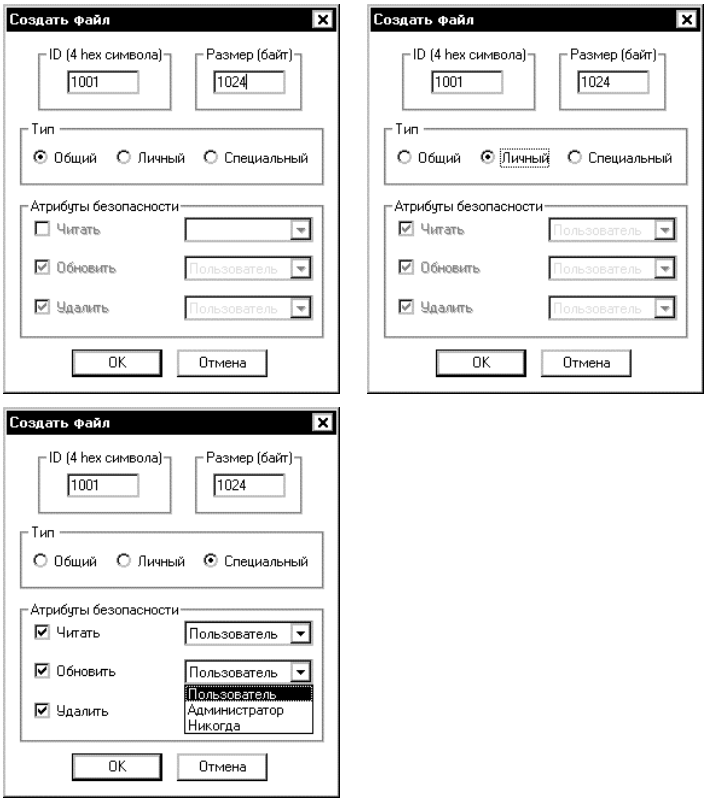
Создание и удаление объектов файловой системы ruToken

Создание файла

Файлы могут быть созданы как а Корневой папке (Master File, MF), так и в других папках. Для создания нового файла нужно сделать текущей папку, в которой он будет создан. Затем, после выбора команды меню Файл|Создать|Файл (либо аналогичной

команды контекстного меню или кнопки в панели инструментов), появится диалог созания файла.

Для создания файлов требуются права доступа, определенные атрибутом безопасности 'Создать файл' той папки, в которой будет создаваться новый файл.



В этом диалоге необходимо ввести имя создаваемого файла, его размер, установить его тип и атрибуты безопасности.

ID (идентификатор файла, используемый в качестве его имени) – двухбайтовое числовое значение, вводимое в виде шестнадцатиричного числа.

Размер (байт) – размер файла. Указывается в байтах и не может быть изменен в дальнейшем.

Тип – тип файла. Выбирается в соответствии с предустановленными наборами атрибутов безопасности:

Атрибут безопасности	Тип файла		
	Общий	Личный	Специальный
Читать	– (не защищено)	Пользователь	*
Обновить	Пользователь	Пользователь	*
Удалить	Пользователь	Пользователь	*

* Атрибут безопасности устанавливается вручную и может принимать значения: Администратор, Пользователь или Никогда, либо атрибут безопасности может быть не установлен (соответствующая операция будет не защищена).

Атрибуты безопасности (Security attributes) назначают различным операциям с данным объектом файловой системы, предваряющие их security-ориентированные действия, успешное выполнение которых является необходимым условием выполнения самой операции. В данном случае под такими действиями понимается установка текущих прав доступа Пользователя или Администратора путем ввода соответствующего PIN-кода:

–	Операция может быть выполнена при любых правах доступа (операция не защищена)
Пользователь	Для выполнения операции требуются права доступа Пользователя
Администратор	Для выполнения операции требуются права доступа Администратора
Никогда	Выполнение операции невозможно

Важно

Атрибут безопасности 'Читать' разрешает выполнять только операцию чтения из файла, а атрибут 'Обновить' – только операцию записи в файл (но не чтения). Поэтому, если предусматривается возможность *редактирования* специального файла – например, в редакторе утилиты rtEditor – важно, чтобы эти 2 атрибута безопасности не противоречили друг другу. То есть атрибуты должны быть заданы таким образом, чтобы владелец, имеющий право писать в файл, имел бы также возможность и читать из файла – в противном случае он не сможет загрузить файл для редактирования. Если же возможность редактирования файла не нужна (например, Администратор будет лишь записывать в файл какие-либо отчеты для Пользователя), данные атрибуты безопасности могут быть установлены в любом сочетании.

Примечание

Вновь созданный файл содержит случайные данные. При необходимости файл можно заполнить любым символом, открыв его для редактирования и выбрав в редакторе команду Правка|Заполнить символом..., а затем сохранив его при помощи команды редактора Файл|Сохранить.

Удаление файла

Для удаления файла требуются права доступа, определенные атрибутом безопасности 'Удалить файл' данного файла. Если для операции Удалить (Delete file) установлен атрибут 'Никогда', файл не может быть удален ни Пользователем, ни Администратором – его удаление возможно только при инициализации памяти токена.

Чтобы удалить файл, нужно сделать его текущим, а затем выбрать команду меню Файл|Удалить (либо аналогичную команду контекстного меню или кнопку в панели инструментов).

Создание папки

Папки могут быть созданы как в Корневой папке, так и в других папках. Для создания новой папки необходимо сделать текущей папку, в которой она будет создана. Затем, после выбора команды меню Файл|Создать|Папку (либо аналогичной команды контекстного меню или кнопки в панели инструментов), появится диалог созания папки.

Для создания папки требуются права доступа, определенные атрибутом безопасности 'Создать файл' той папки, в которой будет создаваться новая.

В этом диалоге необходимо ввести имя создаваемой папки, установить ее тип и атрибуты безопасности.

ID (идентификатор папки, использующийся в качестве ее имени) – двухбайтовое числовое значение, вводимое в виде шестнадцатиричного числа.

Тип – тип папки. Выбирается в соответствии с предустановленными наборами атрибутов безопасности:

Атрибут безопасности	Тип папки	
	Папка	Специальная папка
Создать файл	Пользователь	*
Создать объект	Пользователь	*
Удалить (себя)	Пользователь	*

* Атрибут безопасности устанавливается вручную и может принимать значения: Администратор, Пользователь или Никогда, либо атрибут безопасности может быть не установлен (соответствующая операция будет не защищена).

Атрибуты безопасности (Security attributes) назначают различным операциям с данным объектом файловой системы, предваряющие их security-ориентированные действия, успешное выполнение которых является необходимым условием выполнения самой операции. В данном случае под такими действиями понимается установка текущих прав доступа Пользователя или Администратора путем ввода соответствующего PIN-кода.

–	Операция может быть выполнена при любых правах доступа (операция не защищена)
Пользователь	Для выполнения операции требуются права доступа Пользователя
Администратор	Для выполнения операции требуются права доступа Администратора
Никогда	Выполнение операции невозможно

Удаление папки

Для удаления папки требуются права доступа, определенные атрибутом безопасности 'Удалить' этой папки. Если установлен атрибут 'Никогда', папка не может быть удалена ни Пользователем, ни Администратором – ее удаление возможно только при инициализации памяти токена.

Чтобы удалить папку, нужно сделать ее текущей, а затем выбрать команду меню Файл|Удалить (либо аналогичную команду контекстного меню или кнопку в панели инструментов).

Создание GOST-объектов (ключей шифрования ГОСТ)

Ключи шифрования по ГОСТ 28147-89 могут создаваться в папке по умолчанию (3F00\0000\0001\), либо в текущей папке. Местоположение объекта данных определяется значением старшего бита его ID. Если старший бит ID содержит 0, объект будет создан в папке по умолчанию, если 1 – в текущей папке. Если предполагается, что объект данных будет создан не в папке по умолчанию, нужно сделать папку, в которой он будет создан,

текущей. Затем, вне зависимости от предполагаемого местоположения объекта, нужно выбрать команду меню Файл|Создать|Ключ (либо аналогичной команду контекстного меню или кнопки в панели инструментов). После этого появится диалог созания ключа.

Для создания объектов данных требуются права доступа, определенные атрибутом безопасности 'Создать объект' той папки, в которой будет создаваться объект данных.

В этом диалоге необходимо ввести идентификатор создаваемого объекта, установить способ получения ключа шифрования (будет ли он сгенерирован самим токеном или импортирован в токен из файла), установить требуемые свойства ключа, флаги и атрибуты безопасности.

ID - идентификатор объекта данных. Представляет собой однобайтовое числовое значение, вводимое в виде шестнадцатиричного числа.

Создать как - определяет метод создания: генерацию ключа токеном, либо импорт ключа из внешнего двоичного файла длиной 32 байта (256 бит).

Флаги определяют, будет ли доступна для чтения длина ключа или он будет закрытым объектом.

Опции задают режим шифрования ГОСТ, который будет использовать данный GOST-объект: простая замена, гаммирование или гаммирование с обратной связью.

Атрибуты безопасности (Security attributes) назначают различным операциям с данным объектом, предваряющие их security-ориентированные действия, успешное выполнение которых является необходимым условием выполнения самой операции. В данном случае под такими действиями понимается установка текущих прав доступа Пользователя или Администратора путем ввода соответствующего PIN-кода:

Атрибут безопасности	Значение атрибута
Использовать	*
Удалить	*

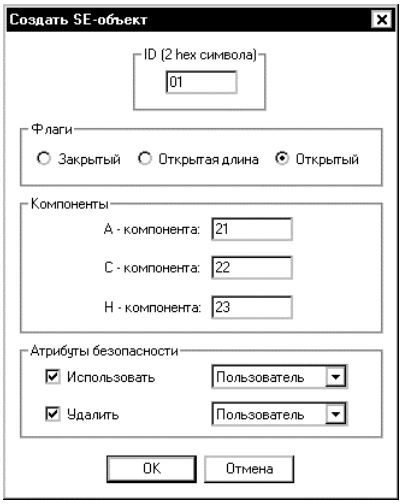
* Атрибут безопасности устанавливается вручную и может принимать значения: Администратор, Пользователь или Никогда, либо атрибут безопасности может быть не установлен (соответствующая операция будет не защищена).

–	Операция может быть выполнена при любых правах доступа (операция не защищена)
Пользователь	Для выполнения операции требуются права доступа Пользователя.
Администратор	Для выполнения операции требуются права доступа Администратора.
Никогда	Выполнение операции невозможно.

Создание SE-объектов (окружений безопасности)

Объекты окружения безопасности (SE-объект) могут создаваться в папке по умолчанию (3F00\0000\), либо в текущей папке. Местоположение объекта данных, содержащего окружение безопасности, определяется значением старшего бита его ID. Если старший бит ID содержит 0, объект будет создан в папке по умолчанию, если 1 – в текущей папке. Если предполагается, что объект данных будет создан не в папке по умолчанию, нужно сделать папку, в которой он будет создан, текущей. Затем, вне зависимости от предполагаемого местоположения объекта, нужно выбрать команду меню Файл|Создать|SE (либо аналогичную команду контекстного меню или кнопки в панели инструментов). После этого появится диалог создания SE-объекта.

Для создания SE-объекта требуются права доступа, определенные атрибутом безопасности 'Создать объект' той папки, в которой будет создаваться объект.



В этом диалоге необходимо ввести идентификатор создаваемого объекта, задать значения требуемым компонентам окружения безопасности, установить флаги и атрибуты безопасности.

ID - идентификатор объекта данных. Представляет собой однобайтовое числовое значение, вводимое виде шестнадцатиричного числа.

Флаги – определяют уровень открытости SE-объекта. Если установлен флаг 'Закрытый', то ни длина, ни содержимое объекта не будут доступны для чтения. Установленный флаг 'Открытая длина' означает, что для чтения будет доступна только информация о длине тела, но не само тело объекта. Установка флага 'Открытый' делает доступным для чтения и длину тела, и само тело объекта.

Компоненты – содержат ID объектов данных, которые будет использованы токеном для выполнения security-ориентированных операций после того, как окружение безопасности из этого SE-объекта будет загружено как текущее (Current Security Environment, CSE

Компонента	Значение	Комментарий
A-компонента	ID GCHV-объекта	ID объекта, используемого для аутентификации. Компоненте могут быть присвоены значения 0x01 (Администратор) и 0x02 (Пользователь).

Компонента	Значение	Комментарий
C-компонента	ID GOST-объекта	ID объекта, используемого для шифрования данных по ГОСТ 28147-89
H-компонента	ID GOST-объекта	ID объекта, используемого для хеширования данных (выработки имитовставки по ГОСТ 28147-89)

Если в данном окружении безопасности какие-то из компонент не используются, то поля, соответствующие этим можно оставить пустыми, что равноценно записи 0. Например, при создании SE-объекта, который будет задавать окружение безопасности только для аутентификации владельца токена и для шифрования данных, достаточно задать ID только для A- и C-компонент, а в поле для H-компоненты надо занести 0 или оставить пустым. Когда такое окружение безопасности будет загружено как текущее, выполнение команды хеширования данных (APDU-команда "PSO:HASH") будет невозможно.

ID объектов, записанные в полях компонент, определяют папки, в которых будет производиться поиск объектов данных в момент выполнения security-команды. Если старший бит ID содержит 0, поиск будет производиться в папке по умолчанию для объектов данного типа, если 1 – в текущей папке. Если в процессе поиска объект с данным ID не будет обнаружен, выполнение security-ориентированной команды, оперирующих с этой компонентой, будет невозможно.

Атрибуты безопасности (Security attributes) назначают различным операциям с данным объектом, предваряющие их security-ориентированные действия, успешное выполнение которых является необходимым условием выполнения самой операции. В данном случае под такими действиями понимается установка текущих прав доступа Пользователя или Администратора путем ввода соответствующего PIN-кода:

Атрибут безопасности	Значение атрибута
Использовать	*
Удалить	*

* Атрибут безопасности устанавливается вручную и может принимать значения: Администратор, Пользователь или Никогда, либо атрибут безопасности может быть не установлен (соответствующая операция будет не защищена).

–	Операция может быть выполнена при любых правах доступа (операция не защищена)

Пользователь	Для выполнения операции требуются права доступа Пользователя.
Администратор	Для выполнения операции требуются права доступа Администратора.
Никогда	Выполнение операции невозможно.

Создание GCHV-объектов

Объекты Global Cardholder Verification (GCHV-объекты) создаются в системной папке 3F00\0000\0000\ при инициализации памяти токена. В текущей версии ruToken создавать новые GCHV-объекты невозможно.

Примечание

Чтобы получить возможность просмотра системной папки, используйте команду меню Вид|Отображать системную папку. **ВНИМАНИЕ: Настоятельно НЕ рекомендуется изменять содержимое этой папки, а также содержимое вложенных в нее папок.**

Удаление объектов данных

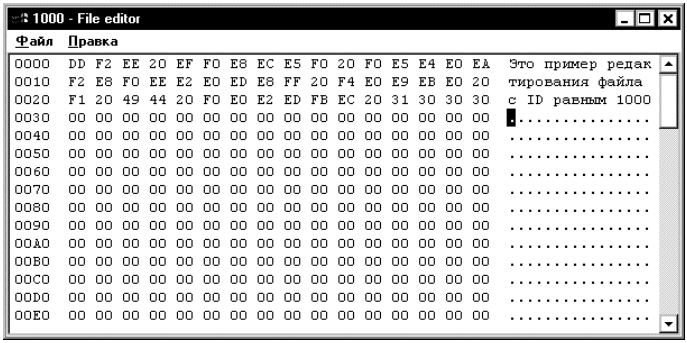
Для удаления объекта данных требуются права доступа, определенные атрибутом безопасности 'Удалить' удаляемого объекта. Если установлен атрибут 'Никогда', объект не может быть удален ни Пользователем, ни Администратором – его удаление возможно только при инициализации токена.

Чтобы удалить объект данных, нужно сделать его текущим, а затем выбрать команду меню Файл|Удалить (либо аналогичную команду контекстного меню или кнопку в панели инструментов).

Просмотр и редактирование файлов

Для чтения файла требуются права доступа, определенные атрибутом безопасности 'Читать', для записи отредактированного файла – права доступа, определенные атрибутом безопасности 'Обновить' данного файла.

После установки соответствующих прав доступа файл можно открыть для просмотра и (или) редактирования двойным щелчком левой кнопки мыши на его ID, либо при помощи команды 'Открыть' контекстного меню. При этом откроется окно редактора, который позволяет просматривать и редактировать файлы в шестнадцатиричном или символьном режиме.



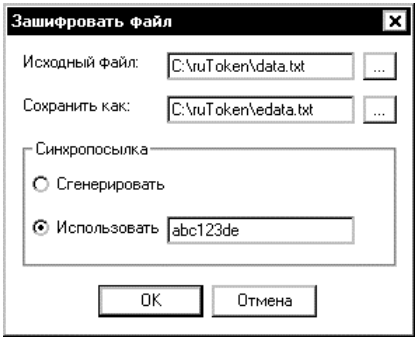
Отредактированный файл можно сохранить в памяти токена командой Файл|Сохранить или оставить неизмененным, закрыв окно редактора и отказавшись от сохранения.

Команда меню Правка|Заполнить символом... позволяет заполнить файл символом с выбранным шестнадцатиричным значением.

Зашифрование и расшифрование данных по ГОСТ 28147-89

Для выполнения операций зашифрования и расшифрования данных по ГОСТ 28147-89 требуются права доступа, определенные атрибутом безопасности 'Использовать' GOST-объекта с ключом шифрования.

Чтобы зашифровать данные, нужно сделать выбранный GOST-объект текущим и при помощи команды меню Действия|Зашифровать (либо аналогичной команды контекстного меню или кнопки в панели инструментов) инициировать процедуру зашифрования. При этом появится следующий диалог:



В диалоге указывается файл с данными для зашифрования (Исходный файл), имя файла, в котором будут сохранены зашифрованные данные (Сохранить как) и способ генерации синхропосылки. Последняя может быть сгенерирована токеном (Сгенерировать) или введена вручную (Использовать). Если выбранный GOST-объект реализует шифрование в режиме **простой замены**, группа "Синхропосылка" будет недоступна, т.к. в этом режиме шифрования синхропосылка не используется.

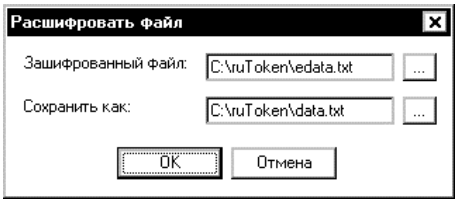
После того как все необходимые параметры введены, процесс зашифрования запускается нажатием на кнопку [Ok].

По окончании процесса зашифрованные данные помещаются в указанный файл; если применялся режим шифрования, использующий синхропосылку, она будет записана в первые 8 байт файла.

Важно

Если выбранный GOST-объект реализует шифрование в режиме **простой замены**, данные для зашифрования должны иметь длину, кратную 8! В противном случае данные будут искажены при расшифровании.

Процедура расшифрования выполняется аналогичным способом командой меню Действия|Расшифровать (либо аналогичной команды контекстного меню или кнопки в панели инструментов). Появится следующий диалог:



В качестве параметров в диалоге указываются имена файла с зашифрованными данными (Зашифрованный файл) и файла для сохранения расшифрованных данных (Сохранить как). Если выбранный GOST-объект реализует шифрование в режиме гаммирования или гаммирования с обратной связью, в первых 8 байтах зашифрованного файла должна находиться синхропосылка. По окончании процесса расшифрованные данные помещаются в указанный файл; синхропосылка теряется.

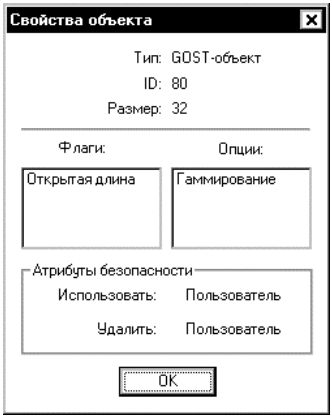
Важно

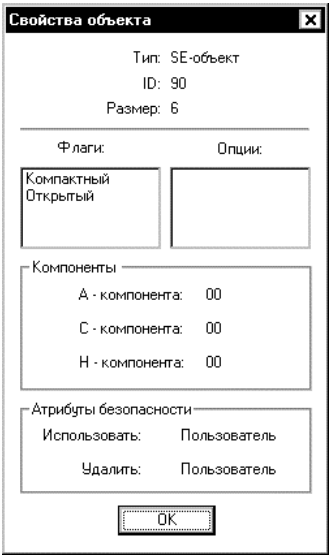
Расшифрование должно производиться на том же ключе (GOST-объекте) и с использованием той же синхропосылки, что и зашифрование! В противном случае данные будут расшифрованы неверно.

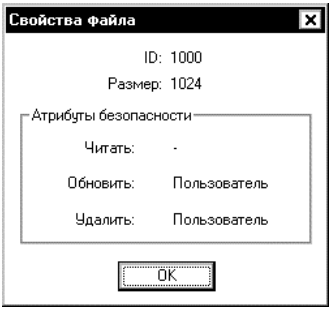
Получение информации об объектах файловой системы

Для получения информации об объекте файловой системе ruToken (о файле, папке, объекте данных) надо сделать этот объект текущим. После чего нужно использовать команду меню Файл|Свойства (либо аналогичную команду в контекстном меню или кнопку в панели инструментов. При этом будет отображено окно, содержащее доступную для отображения информацию о выбранном объекте.

Ниже приведены примеры отображаемой информации для различных объектов.







Изменение символьного имени токена

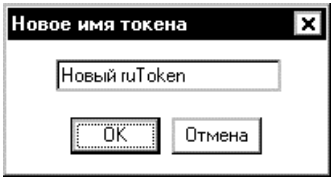
Символьное имя токена служит для облегчения его идентификации и хранится в общем файле 0x1000, лежащем в папке 3F00\0000\0000\ и имеющем следующую структуру:

- Метка файла (2 байта, содержит 'TN')
- Версия файла (2 байта, maj:min)
- Длина символьного имени в байтах (1 байт)
- Символьное имя

При отсутствии этого файла токен будет отображаться как <Unnamed>. Для изменения символьного имени токена требуются права доступа Пользователя.

После установки текущих прав доступа и выбора команды меню Действия|Сменить имя токена появится диалог, в котором следует указать новое имя токена.

Максимальная длина символьного имени токена – 255 байт. Допустимыми являются все отображаемые символы, регистр учитывается.



Часть 5
Комплект поставки для конечных пользователей

Для обеспечения корректной работы с ruToken созданное вами ПО должно включать в себя ряд компонентов Комплекта разработчика. Ниже показано, какие компоненты ПО ruToken нужно поставлять конечным пользователям в зависиости от используемых вами интерфейсов ruToken и от методов работы с токеном:

Каким образом используется ruToken	**Какие компоненты ПО ruToken нужно передать конечному пользователю**				
	Драйверы	**Библ-ка ядра rtAPI.dll**	**Сервис-провайдер**	**Библ-ка PKCS #11**	**Утилиты**
При помощи MS SmartCard API	Да	Только для утилит	Нет	Нет	По желанию
При помощи библиотеки ядра rtAPI	Да	Да (если используется lib-модуль – только для утилит).	Нет	Нет	По желанию
При помощи MS CAPI	Да	Только для утилит	Да	Нет	По желанию
При помощи API ICC Service Provider	Да	Только для утилит	Да	Нет	По желанию
При помощи API PKCS #11	Да	Только для утилит	Да	Да	По желанию
Для PKI Logon	Да	Нет	Да	Нет	Нет
Для работы с e-mail-клиентами	Да	Нет	Да	Нет	Нет
Для работы с центрами сертификации	Да	Нет	Да	Нет	Нет

Для удобства переноса ПО ruToken на компьютеры конечных пользователей все основные компоненты архитектуры ruToken имеют собственные программы установки. Ниже показано, где расположены эти компоненты и какие программы служат для установки каждлго из них:

Компонент ПО ruToken	**Расположение**	**Инсталлятор**
Драйверы ruToken	Drivers\	Drivers\SetupDrv.exe
Библиотека ядра rtAPI.dll	Drivers\	Drivers\SetupDrv.exe
Сервис-провайдер ruToken	Lib\Providers\	Lib\Providers\SetupProv.exe
Библиотека PKCS #11	Lib\PKCS11\	Lib\PKCS11\SetupPKCS.exe
Утилиты для работы с ruToken	Bin\	Нет

Приложения

Приложение 1. PIN-коды по умолчанию.

Владелец	PIN-код владельца по умолчанию (строка символов)
Пользователь	'12345678'
Администратор	'87654321'

Приложение 2. Спецификация ruToken.

ATR	3B 6F 00 FF 00 56 72 75 54 6F 6B 6E 73 30 20 00 00 90 00
Контроллер USB	CY7C63613
Тактовая частота контроллера USB	6 МГц
EEPROM память	8 Кбайт, 16 Кбайт, 128 Кбайт
Тип USB порта	Тип A
Габаритные размеры	58x16x8 мм
Масса	~ 7 г.
Потребляемая мощность	120 mV
Диапазон рабочих температур	От 0 до +70 °C
Диапазон температур хранения	От –10 до +80 °C
Допустимая относительная влажность	От 0 до 100 % (без конденсата)
Время хранения данных	До 100 лет
Количество циклов чтения из памяти	Не ограничено
Количество циклов записи в память	До 1,000,000
Внешние источники питания/батареи	Нет
Гарантированное количество подключений к USB порту	До 5,000