# Описание функций API

# «IT SM»

## версия 1.0 исполнение 2

(криптомодуль IT SM M)

Редакция 6 / 14.05.2026 г.

105203, Город Москва, вн.тер. г. Муниципальный Округ Восточное Измайлово,
ул. 14-я Парковая, дом 8, помещение 8/5
+7 (499) 270-12-45

Фискальная Безопасность



**105203, Город Москва, вн.тер. г. Муниципальный Округ Восточное Измайлово, ул. 14-я Парковая, дом 8, помещение 8/5**

**+7 (499) 270-12-45**

# Оглавление

# 1. Введение.

Криптомодуль (или просто модуль) предназначен для выполнения криптографических операций, а также для хранения данных (в том числе с обеспечением их подлинности и некорректируемости).

Криптомодуль является ведомым устройством. Он устанавливается в мастер-устройство (далее просто «мастер»). Таким мастер-устройством может служить, например, электронная пломба или устройство сбора и передачи данных.

Мастер по одному из доступных интерфейсов посылает модулю команды и получает от него ответы.

Далее при описании различных команд и параметров вместо непосредственных значений, как правило, используются имена констант, начинающиеся с “SCM”. Эти константы определены в файле scm_api.h.

# 2. Физический интерфейс.

Криптомодуль устанавливается в мастер-устройство посредством интерфейсного разъема, содержащего контакты для питания модуля, коммуникационных интерфейсов и др.

## 2.1 Питание и включение/выключение модуля.

Питается криптомодуль от 5 В.

Для включения/выключения модуля предусмотрена отдельная линия в разъеме. Когда на этой линии нулевое напряжение, модуль выключен, когда на нее подается 5 В, модуль включается.

Если модуль не имеет собственного аккумулятора, то питание на него должно подаваться постоянно. Если аккумулятор имеется, то питание может подаваться только при включении модуля. При этом рекомендуемый порядок действий при включении: сначала подать питание, затем подать 5 В на линию включения. При выключении – обратный порядок.

Необходимо иметь в виду, что для многих криптографических операций чрезвычайно важно, чтобы часы реального времени (RTC) в модуле работали постоянно (см. ниже раздел «Криптография и ключи»). Собственный аккумулятор модуля питает RTC, но его заряд может закончиться. Поэтому рекомендуется всегда подавать питание на модуль и включать/выключать его, используя специальную линию в разъеме.

Модуль должен выключаться только после получения его ответа на последнюю команду или при его «зависании» (некоторый таймаут при ожидании ответа на команду). Между выключением и повторным включением должно пройти не менее 300 миллисекунд.

## 2.2 Коммуникационные интерфейсы.

Модуль поддерживает следующие коммуникационные интерфейсы: USB и UART. При включении ПО модуля ожидает команды на обоих интерфейсах. Как только по одному из интерфейсов приходит понятная команда, другой интерфейс не используется.

Когда модуль готов принимать команды, он выдает в UART канал строку “SCM READY”, заканчивающуюся символом 0x0A. По USB ничего не передается: как только мастер обнаружил USB устройство, оно готово к работе.

Параметры работы по USB: модуль функционирует как композитное USB устройство (USB device) с одним интерфейсом с одной конфигурацией, с двумя bulk endpoint для обмена данными.

Скорость работы UART интерфейса 115200 бод, с одним стоп-битом, без бита четности.

Команды, явно или неявно (например, при доступе к зашифрованным данным) использующие криптографические функции, могут быть доступны не сразу. Модулю требуется время (несколько секунд) прежде, чем его ПО сможет отвечать на такие команды. До тех пор на такие команды будет возвращаться ошибка «Не готов» (SCM_ERROR_NOT_READY). Такая же ошибка может возвращаться в ответ и на другие команды, для обработки которых требуется начальная инициализация внутри модуля. См. также описание команды «Задать идентификатор мастер устройства».

### 2.2.1 Особенности работы по USB.

Разные системы и устройства могут по разному определять окончание USB транзакции: по количеству полученных байтов в рамках транзакции, по получению неполного пакета, т.е. пакета, содержащего данных меньше максимального размера (для подключения по протоколу USB high-speed это 512 байтов, по USB full-speed – 64 байта). Если размер передаваемого буфера кратен максимальному размеру USB пакета, то размер последнего пакета будет равен этому максимальному размеру. Если принимающая сторона полагается только на получение неполного пакета, то она будет считать, что передача все еще не завершена, и операция чтения никогда не завершится (или завершится с ошибкой по таймауту). В этом случае необходимо перезагрузить модуль для восстановления работоспособности.

Модуль при передаче данных по USB по умолчанию использует «безопасный» способ передачи: если размер буфера кратен максимальному размеру USB пакета, то сначала передается весь буфер, кроме последнего байта, а затем последний байт передается отдельной транзакцией (таким образом, мастер должен быть готов к тому, что ответ модуля на переданную команду может прийти не за одну USB транзакцию). Модуль после загрузки всегда начинает работать в этом безопасном режиме. Если мастер для определения окончания транзакции использует количество полученных байтов, то можно отключить безопасный режим, задав командой «Установить параметр модуля» (см. далее) значение «0» для параметра usb.last_packet_fix.

При приеме данных для определения окончания USB транзакции модуль использует как количество полученных байтов, так и получение неполного пакета (что случится раньше). Поэтому от мастера в случае кратности размера передаваемого буфера максимальному размеру USB пакета никаких специальных мер предпринимать не требуется.

Проверить корректность работы по USB протоколу можно, записывая в хранилище модуля файлы определенного размера и считывая их обратно (см.

описание команд «Записать данные в хранилище» и «Получить данные из хранилища»). Для проверки корректности передачи можно записать файл размером 476 байтов (имя файла, передаваемое модулю в качестве имени записи, должно иметь длину 8 байтов, в этом случае размер буфера с командой записи будет составлять 512 байтов). Для проверки корректности приема можно записать и потом считать обратно файл размером 498 байтов (в этом случае размер буфера с ответом модуля на команду чтения будет составлять 512 байтов).

## 2.3 Линия тревоги.

В интерфейсном разъеме модуля имеется специальная линия (Alarm) для экстренного удаления ключей. Эта линия может использоваться в случае критической угрозы безопасности, например, при вскрытии корпуса мастер-устройства.

Принцип ее работы следующий. В экстренной ситуации мастер должен выставить на этой линии логический ноль, после чего включить модуль (хотя бы на 0.5-1 сек.). Ключи будут удалены, и криптофункции модуля, использующие секретные ключи, работать не будут. Восстановить работоспособность криптофункций и создать новые ключи в модуле можно только специальной сервисной процедурой с извлечением модуля из мастер-устройства.

Если в экстренной ситуации модуль уже включен и работает, то мастер может либо просто выключить его и провести вышеуказанную процедуру удаления ключей, либо дать модулю команду «Установить состояние тревоги» с флагом SCM_ALARM_FLAG_DELETE_KEYS (см. описание этой команды). Ключи могут быть удалены и в результате ввода неправильного идентификатора мастер-устройства (см. описание команды «Задать идентификатор мастер-устройства»).

## 2.4 Внешний контур защиты.

В интерфейсном разъеме модуля имеется пара контактов «Protect_1» и «Protect_2», образующих линию защиты. Она может использоваться мастером для построения своего контура защиты (например, для защиты от вскрытия корпуса мастер-устройства). Принцип его работы заключается в том, что пока эти контакты замкнуты между собой, ключи в модуле сохраняются, а когда разомкнуты – ключи удаляются. Этот подход отличается от использования линии тревоги (см. раздел «Линия тревоги») тем, что для удаления ключей от мастера не требуется включения модуля и подачи дополнительных команд модулю, ключи удаляются сразу при размыкании контактов.

Примечание. Терминологическое уточнение: линия защиты – это пара контактов «Protect_1» и «Protect_2», контур защиты – это комплекс программно-аппаратных средств модуля, включающий и линию защиты.

Для управления контуром защиты используется команда «Контур защиты», позволяющая включать и отключать контур защиты и получать информацию о его текущем состоянии.

Пока контур защиты не включен, линия защиты не используется: ее размыкание не приводит к удалению ключей. Включение контура защиты может быть немедленным или автоматическим. Автоматическое включение происходит, когда модуль при старте

или при обработке соответствующей команды «Контур защиты» обнаруживает замкнутую линию защиты.

При этом необходимо учитывать, что ключи модуля кэшируются при старте модуля и остаются доступными для выполнения криптографических операций даже после размыкания линии защиты до тех пор, пока модуль не будет выключен. Поэтому рекомендуется строить контур защиты мастера таким образом, чтобы при размыкании линии защиты также отключалось питание модуля. Проверить, сохраняются ли реальные (некэшированные) ключи, можно с помощью параметра модуля «crypto.keymem_status».

Контур защиты может быть защищен паролем. В этом случае для отключения контура потребуется указать тот же пароль, что был задан при его включении. Отключение контура защиты может использоваться, например, для проведения каких-то регламентных работ, требующих вскрытия корпуса мастер-устройства. В этом случае мастер может получать пароль от некоторого оператора для отключения контура защиты и его последующего включения. Конечно, необходимо позаботиться о защите канала между мастером и оператором, что этот пароль нельзя было перехватить.

Процедура начальной инициализации ключей (см. раздел «Криптография и ключи») отключает контур защиты и сбрасывает пароль. Обновление ключей (см. описание команды «Обновить ключи») никак не меняет состояние контура защиты.

# 3. Работа с модулем.

Конкретный сценарий работы с модулем выбирается мастером в зависимости от решаемых задач.

Типичные варианты использования модуля включают работу с хранилищами данных и создание TLS сессий.

## 3.1 Работа с хранилищами.

Как явствует из названия, хранилища позволяют хранить в модуле различные данные.Хранилища в модуле создаются по командам мастера (см. раздел «Функции хранения данных»). Это может происходить либо в рамках некоторой инициализации/привязки, либо «на лету» во время обычного сеанса работы с модулем: мастер проверяет, существует ли требуемое хранилище и создает его, если оно отсутствует.

Когда необходимо записать какие-то данные в определенное хранилище или считать из него ранее записанные данные, мастер открывает хранилище, указывая его имя, и получает его дескриптор. Используя этот дескриптор, можно давать команды чтения или записи. По окончании работы с хранилищем мастер закрывает дескриптор.

## 3.2 Использование TLS.

TLS сессии могут использоваться для защиты канала передачи данных между мастером и каким-либо сервером или пользователем. Тип канала (сетевое соединение, NFC и т.д.) для модуля несущественны. Модуль работает с буферами данных, транспортные функции (т.е. собственно передачу) осуществляет сам мастер.

Для использования TLS в модуле должны быть установлены криптографические ключи (см. ниже раздел «Криптография и ключи»).

Сначала мастеру необходимо создать сессию командой «Начать TLS сессию» (см. раздел «Функции для организации TLS»), которая возвращает дескриптор сессии. Все дальнейшие операции в рамках данной сессии осуществляются с использованием этого дескриптора. Время создания сессии (до открытия канала передачи или после) мастер выбирает по своему усмотрению. После создания сессии и открытия канала передачи мастер передает удаленной стороне данные, полученные от модуля в результате выполнения начальной команды «Начать TLS сессию» или последующих команд «Установить TLS сессию», и передает модулю в командах «Установить TLS сессию» данные, полученные от удаленной стороны, Когда модуль в ответе на команду вернет сигнал, что сессия установлена, мастер может начать обмен своими данными с удаленной стороной. Для этого мастер шифрует свои данные с помощью модуля (команда «Зашифровать TLS данные»), передает зашифрованные данные удаленной стороне, в свою очередь получает от удаленной стороны зашифрованные данные и расшифровывает их командой «Расшифровать TLS данные».

Когда работа с удаленной стороной завершена, мастер закрывает сессию (команда «Завершить TLS сессию»).

## 3.3 Типичный сеанс работы с модулем.

Общая схема сеанса работы с модулем выглядит следующим образом.

Когда мастеру необходимо использовать функции криптомодуля, он включает модуль и дожидается его готовности (см. раздел «Физический интерфейс»).

Мастер «представляется» модулю командой «Задать идентификатор мастер-устройства». Если модуль был привязан к конкретному идентификатору (см. описание указанной команды), то необходимо указывать именно этот идентификатор.

Мастер записывает в некоторое хранилище какие-то данные или считывает ранее записанные данные. Это могут быть документы, записи журнала и др.

Мастер инициализирует TLS сессию для защиты канала с каким-то сервером или пользователем. Данные, передающиеся между мастером и удаленной стороной модулем не анализируются, только шифруются/расшифровываются. По окончании сеанса связи мастер дает модулю команду завершить TLS сессию.

При необходимости мастер может использовать низкоуровневые криптофункции (см. раздел «Базовые криптофункции»).

Мастер может вызвать функции модуля для собственных нужд или по запросу пользователя.

Если это предусмотрено логикой работы системы, в рамках которой функционирует мастер, то периодически (например, один-два раза в день) мастер может запрашивать модуль, не требуется ли тому какое-либо сервисное соединение (см. раздел «Дополнительные команды»). Сервисные соединения могут быть очень важны для криптографической инфраструктуры модуля (см. ниже раздел «Криптография и ключи»).

По завершении работы с модулем (после получения ответа на последнюю команду) мастер выключает модуль.

Мастер может управлять внутренним состоянием модуля, используя команду «Установить состояние тревоги». Эта команда может использоваться в случаях обнаружения мастером нарушения какого-то контура безопасности.

## 3.4 Криптопровайдеры.

Модуль может выполнять криптографические операции согласно определенным алгоритмам. Алгоритмы объединяются в некоторые группы. Например, это могут быть группы алгоритмов, заданных какими-либо национальными стандартами. В модуле поддержка подобных групп алгоритмов осуществляется так называемыми криптопровайдерами. Каждый криптопровайдер в модуле имеет свой уникальный идентификатор. При выполнении криптографической операции модулю передается идентификатор криптопровайдера, что позволяет выполнять операцию в соответствии с указанной группой алгоритмов.

Определены следующие идентификаторы криптопровайдеров (указаны основные алгоритмы, реализуемые криптопровайдерами):

SCM_CRYPTO_PROVIDER_TYPE_GOST
- хэш ГОСТ 34.11-2012 256 бит
- подпись ГОСТ 34.10-2012 256 бит
- симметричное шифрование ГОСТ 34.12-2015 («Магма»)

SCM_CRYPTO_PROVIDER_TYPE_CN
- хэш SM3
- подпись SM2
- симметричное шифрование SM4
- защищенное соединение согласно GMT 0024-2014

SCM_CRYPTO_PROVIDER_TYPE_RSA
- хэш SHA256
- подпись RSA 2048 бит
- симметричное шифрование AES256

SCM_CRYPTO_PROVIDER_TYPE_BELR
- хэш BELT-HASH256 (СТБ 34.101.31)
- подпись BIGN 256-бит (СТБ 34.101.45)
- симметричное шифрование BELT256 (СТБ 34.101.31)

Криптопровайдеры могут реализовывать и другие алгоритмы для внутренних операций.

Также, если явно не было указано выше, криптопровайдеры обеспечивают защищенные соединения по TLS протоколу с использованием указанных или совместимых алгоритмов.

Можно узнать список доступных криптопровайдеров в модуле командой «Получить список криптопровайдеров».

## 3.5 Криптография и ключи.

Для выполнения многих криптографических операций необходимо, чтобы в модуле были установлены криптографические ключи и сертификаты.

Первоначально ключи и сертификаты устанавливаются в модуле в ходе специальной процедуры инициализации, описание которой выходит за рамки данного документа. Во время дальнейшей эксплуатации модуля ключи и сертификаты могут быть обновлены по команде «Обновить ключи». Команда «Получить состояние модуля» с параметром SCM_MODULE_STATUS_TYPE_KEYINIT позволяет проверить наличие ключей у криптопровайдера. Также для этих целей может использоваться команда «Получить информацию о ключе» и параметр модуля «crypto.keymem_status».

Многие криптографические операции сопровождаются проверкой сертификатов ключей. При этом проверяется срок действия сертификатов, т.е. модулю необходимо знать текущее время. Для этого используется микросхема часов реального времени (RTC) в модуле, которая должна работать постоянно (см. раздел «Питание и включение/выключение модуля»). Мастер может проверить текущее время в модуле командой «Получить время модуля».

Модуль может иметь встроенные ограничения срока действия ключей и сертификатов. Для проверки этих ограничений можно использовать команду «Получить параметр модуля» с именами параметров «crypto.key_expiration», «crypto.key_expiration_days», «crypto.key_expiration_max» и др. Рекомендуется регулярно проверять значение параметра «crypto.key_expiration», который сигнализирует о приближении срока окончания действия ключей и сертификатов, и своевременно обновлять ключи в модуле командой «Обновить ключи». Количество обновлений ключей также может быть ограничено (количество доступных обновлений может быть проверено в параметре «crypto.key_update_remain»). Если доступных обновлений нет, то сменить ключи и сертификаты в модуле можно только процедурой инициализации с извлечением модуля из мастер-устройства.

Модуль выполняет операции с ключами и после истечения срока их действия.

Также при проверке сертификатов используются списки отозванных сертификатов, которые необходимо регулярно обновлять. Поэтому необходимо давать модулю возможность связаться с определенными серверами для получения обновлений. Для этого используются сервисные соединения (см. раздел «Типичный сеанс работы с модулем», команду «Сервисное соединение»). Если возможности для регулярной связи отсутствуют, то списки отзыва могут стать неактуальными, и проверка сертификатов закончится неудачей. Мастер может отключать использование списков отзыва (см. описание команд «Проверить сертификат», «Начать TLS сессию»), но это снижает уровень безопасности. Списки отзыва устанавливаются в модуле во время процедуры инициализации или вручную (см. описание команды «Задать список отзыва»). Также они могут автоматически обновляться по мере истечения срока их действия (см. описание команды «Сервисное соединение» и параметра модуля “crypto.user_cert_cdp”).

# 4. Формат команды и ответа.

Взаимодействие между мастером и модулем строится по принципу команда-ответ. Мастер дает команды, модуль их обрабатывает и на каждую дает ответ. Пока не получен ответ на команду, следующую команду давать нельзя.

В ответ на любую команду модулем может быть дан ответ об ошибке (см. «Коды ошибок» и описание ответа «Ошибка»).

Команда и ответ передаются в виде пакета – последовательности байтов, имеющих определенный формат.

Максимальный размер пакета – SCM_PACKET_MAX байтов. Если данные, которые необходимо передать, не помещаются в один пакет, то данные разбиваются на несколько пакетов (см. описание команды «Продолжение»). Рекомендуется по возможности разбивать данные на логически завершенные части для упрощения их последовательной обработки.

Можно использовать таймаут SCM_TIMEOUT_REPLY для обработки модулем команды. Если модуль не успевает завершить обработку команды, то он может сообщить об этом мастеру (см. описание ответа «Ожидание»). Также может быть использован общий таймаут зависания SCM_TIMEOUT_HANG, по истечении которого мастер считает модуль «зависшим» и может перегрузить его - выключить и включить повторно (см. также описание ответа «Ожидание»).

Поля в пакете, имеющие тип размером более 1 байта, кодируются в little-endian (LSB 0).

В начале пакета находится его заголовок, за которым следуют параметры (данные).

## 4.1 Заголовок пакета.

**Заголовок пакета команды:**

Идентификатор протокола – 1 байт.
Код команды – 1 байт.
Флаги – 1 байт.
Длина параметров пакета – 2 байта.
Контрольная сумма заголовка – 2 байта.
Контрольная сумма параметров – 2 байта.

**Заголовок пакета ответа:**

Идентификатор протокола – 1 байт.
Код ответа – 1 байт.
Флаги – 1 байт.
Длина параметров пакета – 2 байта.
Контрольная сумма заголовка – 2 байта.
Контрольная сумма параметров – 2 байта.

### 4.1.1 Идентификатор протокола.

Всегда имеет значение SCM_PROTO_TAG.

### 4.1.2 Код команды или ответа.

См. раздел «Команды и ответы».

### 4.1.3 Флаги.

Битовая маска из следующих значений:

SCM_PACKET_FLAG_CMD – пакет является командой,

SCM_PACKET_FLAG_REPLY – пакет является ответом на команду,

SCM_PACKET_FLAG_CONTINUE – флаг продолжения (см. описание команды «Продолжение»).

Флаги SCM_PACKET_FLAG_CMD и SCM_PACKET_FLAG_REPLY не могут присутствовать одновременно.

### 4.1.4 Длина данных.

Длина параметров пакета, не включая размер заголовка пакета.

### 4.1.5 Контрольная сумма заголовка.

Вычисляется от всего заголовка пакета после вычисления контрольной суммы параметров. При вычислении данное поле должно равняться нулю.

Алгоритм CRC16 CCIT X.25, полином 0x1021 (реверсированный 0x8408), начальное значение 0xFFFF, последний XOR 0xFFFF, переставленные байты результата. Пример: контрольная сумма от строки «123456789» равна 0x6E90.

### 4.1.6 Контрольная сумма параметров.

Вычисляется от всех параметров пакета.

Алгоритм тот же, что и для контрольной суммы заголовка.

## 4.2 Параметры пакета.

Параметры пакета – это входные данные команды или возвращаемые данные ответа. Параметры следуют друг за другом. Параметры могут отсутствовать в пакете. Каждый параметр структурируется по принципу tag-length-value: сначала идет заголовок параметра, содержащий идентификатор типа данных (tag) и их длину (не включая данный заголовок), затем – сами данные. Размер поля tag – 1 байт, размер поля длины – 4 байта.

Длина данных указывает полный размер данных параметра, которые необходимо передать, и может превышать размер свободного места в пакете. Данные, не помещающиеся в пакет, передаются в режиме продолжения (см. описание команды «Продолжение»).

## 4.3 Пример команды и ответа.

Команда «Проверка связи» (Ping). Посылаются 4 байта, в ответ ожидаются те же 4 байта в обратном порядке.

Команда: 47 02 01 09 00 F9 9E 83 72 02 04 00 00 00 04 03 02 01

Где:

---- Заголовок пакета команды

47 - Идентификатор протокола (SCM_PROTO_TAG)

02 - Код команды (SCM_CMD_PING)

01 - Флаги (SCM_PACKET_FLAG_CMD)

09 00 - Длина параметров пакета (9 – заголовок параметра и его значение)
F9 9E - Контрольная сумма заголовка
83 72 - Контрольная сумма параметров
---- Параметры команды
02 – Тип параметра/tag (SCM_DATA_BINARY)
04 00 00 00 – Длина параметра (4)
04 03 02 01 – Значение параметра

Ответ: 47 02 02 09 00 42 B5 F9 8C 02 04 00 00 00 01 02 03 04
Где:
---- Заголовок пакета ответа
47 - Идентификатор протокола (SCM_PROTO_TAG)
02 - Код ответа (SCM_REPLY_PING)
02 - Флаги (SCM_PACKET_FLAG_REPLY)
09 00 - Длина параметров пакета (9 – заголовок параметра и его значение)
42 B5 - Контрольная сумма заголовка
F9 8C - Контрольная сумма параметров
---- Параметры ответа
02 – Тип параметра/tag (SCM_DATA_BINARY)
04 00 00 00 – Длина параметра (4)
01 02 03 04 – Значение параметра

# 5. Команды и ответы.

При описании команд указываются коды команд и ответов на них (см. раздел «Заголовок пакета» выше), для каждого параметра команды и возвращаемых данных ответа указаны идентификаторы типа данных (tag, см. раздел «Параметры пакета» выше). Порядок входных параметров и возвращаемых данных должен соответствовать указанному.

Есть набор общих ошибок, которые могут быть возвращены в ответ на любую команду (см. раздел «Коды ошибок»).

Ниже при описании команд указываются, как правило, только дополнительные коды ошибок (не являющиеся общими), специфические для конкретной команды.

## 5.1 Идентификация и статус.

### 5.1.1 Проверка связи.

Код команды: SCM_CMD_PING.
Входные параметры:
произвольные бинарные данные (SCM_DATA_BINARY, не обязательно).

Код ответа: SCM_REPLY_PING.
Возвращаемые данные:
данные полученные в команде с обратным порядком байтов (SCM_DATA_BINARY).

Возможные ошибки: нет.

Команда для проверки наличия криптомодуля и протокола.

### 5.1.2 Получить идентификаторы модуля.

Код команды: SCM_CMD_MODULE_ID.
Входные параметры: нет.

Код ответа: SCM_REPLY_MODULE_ID.
Возвращаемые данные:
Идентификатор (SCM_DATA_BINARY).

Возможные ошибки: нет.

### 5.1.3 Получить состояние модуля.

Код команды: SCM_CMD_MODULE_STATUS.
Входные параметры:
тип статуса (SCM_DATA_BYTE, не обязательно),
дополнительные параметры (в зависимости от типа статуса).

Код ответа: SCM_REPLY_MODULE_STATUS.
Возвращаемые данные:
статус (SCM_DATA_INTEGER).

Возможные ошибки:
зависит от типа статуса.

Запрашиваемый тип статуса может быть одним из следующих значений:
SCM_MODULE_STATUS_TYPE_COMMON – общее состояние, дополнительные параметры не требуются, специфические ошибки отсутствуют.
SCM_MODULE_STATUS_TYPE_KEYINIT – состояние инициализации ключей криптопровайдера, требуется дополнительный параметр - идентификатор криптопровайдера (SCM_DATA_ALG_ID), возможные ошибки: ошибка криптопровайдера (SCM_ERROR_CRYPT).

Если команда не имеет параметров, то подразумевается тип статуса SCM_MODULE_STATUS_TYPE_COMMON.

Возвращаемое значение общего статуса (тип статуса SCM_MODULE_STATUS_TYPE_COMMON) может принимать одно из следующих значений:
SCM_MODULE_STATE_READY – модуль полностью готов к работе,
SCM_MODULE_STATE_INITIALIZING – модуль находится в состоянии инициализации, функционал может быть ограничен, требуются дополнительные действия (см. описание команды «Задать идентификатор мастер-устройства»),

SCM_MODULE_STATE_ERROR – при инициализации обнаружены ошибки, функционал может быть ограничен.

Если запрашивается состояние инициализации ключей (тип статуса SCM_MODULE_STATUS_TYPE_KEYINIT), то статус «3» означает, что ключи криптопровайдера инициализированы. Другие значения указывают на ошибку инициализации ключей.

Значения статусов других типов определяются соответствующими подсистемами модуля. Как правило, мастер не должен самостоятельно их запрашивать.

### 5.1.4 Получить время модуля.

Код команды: SCM_CMD_MODULE_GET_TIME.
Входные параметры: нет.

Код ответа: SCM_REPLY_MODULE_GET_TIME.
Возвращаемые данные:
время (SCM_DATA_TIME).

Возможные ошибки: нет.

См. также описание команды «Задать время модуля».

### 5.1.5 Задать идентификатор мастер-устройства.

Код команды: SCM_CMD_MASTER_ID.
Входные параметры:
идентификатор (SCM_DATA_BINARY),
флаги (SCM_DATA_INTEGER, битовая маска, не обязательно).

Код ответа: SCM_REPLY_MASTER_ID.
Возвращаемые данные: нет.

Возможные ошибки: нет.

Идентификатор – это произвольный набор байтов.

Флаги – битовая маска значений:
SCM_MASTER_ID_FLAG_RESET (сбросить привязку),
SCM_MASTER_ID_FLAG_SET (сделать привязку к новому идентификатору),
SCM_MASTER_ID_FLAG_DELETE_STORAGE (удалять хранилища при сбросе привязки).

Модуль может быть привязан к конкретному мастер-устройству (идентификатору). В этом случае, сразу после загрузки, модуль находится в состоянии SCM_MODULE_STATE_INITIALIZING (см. описание команды «Получить состояние

модуля»). Чтобы начать работу с модулем, необходимо передать ему идентификатор, к которому модуль привязан. Модуль проверит, что этот идентификатор совпадает с требуемым, и состояние модуля станет SCM_MODULE_STATE_READY. Если привязка отсутствует, то состояние модуля после загрузки будет SCM_MODULE_STATE_READY, и для начала работы все равно необходимо выполнить данную команду (можно передать любой непустой идентификатор). Независимо от наличия или отсутствия привязки передавать в команде флаги для начала работы не требуется (или они должны быть равны 0).

Выполнение этой команды (без флагов или с нулевыми флагами) открывает в модуле специальную внутреннюю сессию, имеющую идентификатор «0» (это внутренний идентификатор сессии, не идентификатор мастера!). До выполнения этой команды модуль будет выполнять только ограниченное число команд (в основном, предоставляющие общую информацию о модуле). После того, как внутренняя сессия открыта, она не закрывается до выключения или перезагрузки модуля (в том числе после повторных команд «Задать идентификатор мастер-устройства» независимо от результата их выполнения).

ПРИМЕЧАНИЕ. Указанный идентификатор сессии «0» может использоваться в некоторых командах, требующих идентификатор пользователя, что означает, что данная команда выполняется самим мастером, а не по запросу некоторого пользователя. Если такая команда дается по запросу пользователя, с которым установлено TLS соединение (см. команду «Начать TLS сессию»), то в таких командах необходимо указывать идентификатор этой TLS сессии в качестве идентификатора пользователя.

Чтобы осуществить привязку к новому идентификатору, данная команда выполняется с флагом SCM_MASTER_ID_FLAG_SET. Если модуль уже имел привязку, то в модуле будут удалены ключи. Если привязки не было, ключи не удаляются.

Отменить привязку можно командой с флагом SCM_MASTER_ID_FLAG_RESET (при этом можно передать любой непустой идентификатор, привязки к нему не будет).

Узнать о наличии привязки можно с помощью команды «Получить параметр модуля» с именем параметра «global.master_bind». Если этот параметр равен 0, то привязка отсутствует. Если параметр больше нуля, то привязка есть, а значение параметра – количество оставшихся попыток для ввода идентификатора мастера командой «Задать идентификатор мастер-устройства» без флагов или с нулевыми флагами. Если мастер введет неправильный идентификатор указанное количество раз, то привязка будет сброшена, ключи в модуле удалены (при этом состояние модуля из SCM_MODULE_STATE_INITIALIZING перейдет в SCM_MODULE_STATE_READY). После ввода правильного идентификатора количество попыток снова принимает значение по умолчанию.

Флаг SCM_MASTER_ID_FLAG_DELETE_STORAGE имеет смысл только в сочетании с флагом SCM_MASTER_ID_FLAG_SET, т.е. при создании новой привязки. Если флаг был задан, то при сбросе по любой причине этой привязки кроме ключей будут удалены также все хранилища.

### 5.1.6 Получить состояние тревоги.

Код команды: SCM_CMD_ALARM_GET.

Входные параметры:
номер состояния (SCM_DATA_BYTE).

Код ответа: SCM_REPLY_ALARM_GET.
Возвращаемые данные:
состояние (SCM_DATA_BYTE),
флаги (SCM_DATA_BYTE),
время (SCM_DATA_TIME).

Возможные ошибки:
неверный номер (SCM_ERROR_ITEM_NOT_FOUND).

Всего возможно 16 номеров состояния тревоги. Конкретное значение каждого типа тревоги определяется мастер-устройством.

Если состояние равно 0, то данная тревога неактивна.

Флаги – это битовая маска значений:
SCM_ALARM_FLAG_UNRECOVERABLE – тревога не может быть отключена (см. команду «Установить состояние тревоги»),
SCM_ALARM_FLAG_READ_ONLY – данная тревога переключает модуль в режим «только для чтения» (изменение состояния тревоги останется возможным),
SCM_ALARM_FLAG_DELETE_KEYS – по данной тревоге модуль удаляет криптографические ключи (криптофункции станут невозможны). Удаление ключей можно также выполнить, используя специальную линию в интерфейсном разъеме модуля (см. выше раздел «Линия тревоги»).

Возвращаемое время – это время последнего переключения состояния (или 0, если тревоги не было).

### 5.1.7 Установить состояние тревоги.

Код команды: SCM_CMD_ALARM_SET.
Входные параметры:
номер состояния (SCM_DATA_BYTE),
состояние (SCM_DATA_BYTE),
флаги (SCM_DATA_BYTE, не обязательно).

Код ответа: SCM_REPLY_ALARM_SET.
Возвращаемые данные: нет.

Возможные ошибки:
неверный номер (SCM_ERROR_ITEM_NOT_FOUND),
неотключаемая тревога (SCM_ERROR_ACCESS_DENIED).

Значение параметров см. в команде «Получить состояние тревоги».

Если флаги отсутствуют, то используется значение 0.

Если состояние равно 0, то флаги игнорируются.

Отключить включенное состояние тревоги, для которого был установлен флаг SCM_ALARM_UNRECOVERABLE, нельзя.

Если ключи были удалены, то отключение тревоги их не восстановит.

Удаление ключей можно также выполнить, используя специальную линию в интерфейсном разъеме модуля (см. выше раздел «Линия тревоги»).

### 5.1.8 Получить параметр модуля.

Код команды: SCM_CMD_GET_OPTION.
Входные параметры:
имя параметра (SCM_DATA_STRING).

Код ответа: SCM_REPLY_GET_OPTION.
Возвращаемые данные:
значение параметра (SCM_DATA_BINARY).

Возможные ошибки:
неизвестное имя параметра (SCM_ERROR_INVALID_PARAM),
прочие ошибки зависят от запрашиваемого параметра.

Список поддерживаемых параметров см. в Приложении 1 «Описание параметров криптомодуля».

Возвращаемое значение всегда трактуется либо как бинарное (если в описании параметра явно указано, что его значение представляет собой бинарный массив байтов), либо как строка символов без концевого нуля (во всех остальных случаях). Например, если значение параметра должно быть целым числом, то возвращается строковое представление этого числа. Если значение параметра должно быть логическим значением «да» или «нет», то возвращается строка, соответственно, «1» или «0».

### 5.1.9 Установить параметр модуля.

Код команды: SCM_CMD_SET_OPTION.
Входные параметры:
имя параметра (SCM_DATA_STRING),
значение параметра (SCM_DATA_BINARY).

Код ответа: SCM_REPLY_SET_OPTION.
Возвращаемые данные: нет.

Возможные ошибки:
неизвестное имя параметра (SCM_ERROR_INVALID_PARAM),
параметр только для чтения (SCM_ERROR_ACCESS_DENIED),
прочие ошибки зависят от конкретного параметра.

Список поддерживаемых параметров см. в Приложении 1 «Описание параметров криптомодуля». Многие параметры являются параметрами только для чтения, и задание их значений невозможно.

Передаваемое значение всегда трактуется модулем либо как бинарное (если в описании параметра явно указано, что его значение представляет собой бинарный массив байтов), либо как строка (во всех остальных случаях). Например, если значение параметра должно быть целым числом, то должно передаваться строковое представление этого числа. Если значение параметра должно быть логическим значением «да» или «нет», то передается строка, соответственно, «1» или «0». Строка не обязана содержать концевой ноль, а его наличие не считается ошибкой.

### 5.1.10 Контур защиты.

Код команды: SCM_CMD_PROTECT.
Входные параметры:
код операции (SCM_DATA_INTEGER, не обязательно),
пароль (SCM_DATA_BINARY, не обязательно).

Код ответа: SCM_REPLY_PROTECT.
Возвращаемые данные:.
общий статус контура защиты (SCM_DATA_INTEGER),
текущий статус линии защиты (SCM_DATA_INTEGER).

Возможные ошибки:
линия защиты незамкнута (SCM_ERROR_OUT_OF_RESOURCE),
неверный пароль (SCM_ERROR_ACCESS_DENIED),
не поддерживается (SCM_ERROR_NOT_SUPPORTED).

Код операции может принимать одно из следующих значений:
SCM_PROTECT_GET – получить текущее состояние контура защиты,
SCM_PROTECT_DISABLE – отключить контур защиты,
SCM_PROTECT_ENABLE – включить контур защиты,
SCM_PROTECT_ENABLE_AUTO – включить контур защиты, когда линия защиты будет замкнута.

Если код операции не указан, то подразумевается SCM_PROTECT_GET.

С кодом SCM_PROTECT_ENABLE_AUTO модуль попытается немедленно включить контур защиты (как с кодом SCM_PROTECT_ENABLE). Если в данный момент линия защиты разомкнута, то включение контура будет отложено, пока модуль не обнаружит, что линия замкнута. Проверка линии осуществляется при старте модуля или при обработке данной команды с кодом операции SCM_PROTECT_ENABLE или SCM_PROTECT_ENABLE_AUTO.

Пароль позволяет защитить контур защиты от несанкционированного отключения. Он задается данной командой с кодами операции SCM_PROTECT_ENABLE или SCM_PROTECT_ENABLE_AUTO и проверяется для команд с кодами SCM_PROTECT_ENABLE, SCM_PROTECT_ENABLE_AUTO и SCM_PROTECT_DISABLE.

При использовании неверного пароля, следующие попытки использования пароля блокируются на 20 секунд.

При отключении контура защиты пароль сбрасывается.

Возвращаемый общий статус контура защиты может принимать значения:

SCM_PROTECT_DISABLE – контур защиты отключен,

SCM_PROTECT_ENABLE – контур защиты включен,

SCM_PROTECT_ENABLE_AUTO – контур защиты будет включен, когда линия защиты будет замкнута.

Значение SCM_PROTECT_ENABLE_AUTO будет автоматически изменено на SCM_PROTECT_ENABLE, когда произойдет реальное включение контура защиты.

Текущий статус линии защиты принимает значения:

0 – использование линии защиты не поддерживается,

1 – линия защиты замкнута,

-1 – линия защиты разомкнута.

Использование контура защиты может не поддерживаться конкретной версией модуля. В этом случае возвращается ошибка SCM_ERROR_NOT_SUPPORTED.

Для получения информации о контуре и линии защиты можно также использовать параметр модуля «system.protect_line» (см. Приложение 1 «Описание параметров криптомодуля»).

Дополнительную информацию о работе контура защиты см. в разделе «Внешний контур защиты».

### 5.1.11 Задать время модуля.

Код команды: SCM_CMD_MODULE_SET_TIME.
Входные параметры:

время (SCM_DATA_TIME).

Код ответа: SCM_REPLY_MODULE_SET_TIME.
Возвращаемые данные:

нет.

Возможные ошибки: нет.

Задавать время модуля можно только из надежного источника.

См. также описание команды «Задать время модуля».

## 5.2 Базовые криптофункции.

### 5.2.1 Получить список криптопровайдеров.

Код команды: SCM_CMD_GET_ALG_LIST.
Входные параметры: нет.

Код ответа: SCM_REPLY_GET_ALG_LIST.
Возвращаемые данные:
идентификатор криптопровайдера (SCM_DATA_ALG_ID),
название криптопровайдера (SCM_DATA_STRING),
….

Возможные ошибки: нет.

Данные возвращаются в виде последовательности (списка, возможно, пустого): идентификатор, название, идентификатор, название, … . Название может быть пустой строкой.

### 5.2.2 Получить информацию о ключе.

Код команды: SCM_CMD_GET_KEY_INFO.
Входные параметры:
идентификатор криптопровайдера (SCM_DATA_ALG_ID),
идентификатор ключа (SCM_DATA_KEY_ID).

Код ответа: SCM_REPLY_GET_KEY_INFO.
Возвращаемые данные:
сертификат (SCM_DATA_CERT, не обязательно),
текстовое описание (SCM_DATA_STRING, не обязательно).

Возможные ошибки:
элемент не найден (SCM_ERROR_ITEM_NOT_FOUND).

Идентификатор ключа (SCM_DATA_KEY_ID) состоит из
назначения ключа (1 байт):
SCM_KEY_USAGE_CYPHER - симметричное шифрование,
SCM_KEY_USAGE_TLS – ключ для TLS,
SCM_KEY_USAGE_SIGN – ключ для подписи,
SCM_KEY_USAGE_TLS_SERVICE – ключ для сервисного соединения,
и номера ключа (1 байт) - среди ключей с тем же назначением, если их может быть несколько (ключ по умолчанию имеет номер 0).

### 5.2.3 Получить сертификаты УЦ.

Код команды: SCM_CMD_GET_CA.
Входные параметры:
идентификатор криптопровайдера (SCM_DATA_ALG_ID),
флаги (SCM_DATA_INTEGER, не обязательно).

Код ответа: SCM_REPLY_GET_CA.
Возвращаемые данные:
список SCM_DATA_BINARY (не обязательно).

Возможные ошибки: нет.
Сертификаты доверенных УЦ (если они есть в модуле) возвращаются в виде последовательности параметров типа SCM_DATA_BINARY.
Эти сертификаты могут использоваться модулем, например, при установлении TLS соединений (см. команду «Начать TLS сессию») в целях аутентификации, т.е. для проверки сертификата удаленной стороны (предоставленная удаленной стороной цепочка сертификатов должна содержать сертификат, подписанный доверенным УЦ).

Флаги – это битовая маска. Они определяют, какую информация требуется получить. Допустимые значения:
SCM_GET_CA_FLAG_CA – получить сертификаты промежуточных УЦ,
SCM_GET_CA_FLAG_ROOT – получить сертификаты корневых УЦ,
SCM_GET_CA_FLAG_TRUSTED_OIDS – после каждого сертификата УЦ также будет возвращен буфер (возможно, пустой), содержащий доверенные OID для данного УЦ (список из текстовых представлений OID, разделенных запятыми).
Если флаги не заданы, то используется комбинация SCM_GET_CA_FLAG_CA и SCM_GET_CA_FLAG_ROOT.

### 5.2.4 Получить списки отзыва.

Код команды: SCM_CMD_GET_CRL.
Входные параметры:
идентификатор криптопровайдера (SCM_DATA_ALG_ID).

Код ответа: SCM_REPLY_GET_CRL.
Возвращаемые данные:
список SCM_DATA_BINARY (не обязательно).

Возможные ошибки: нет.

Списки отзыва (если они есть в модуле) возвращаются в виде последовательности параметров типа SCM_DATA_BINARY.

### 5.2.5 Задать список отзыва.

Код команды: SCM_CMD_SET_CRL.
Входные параметры:
идентификатор криптопровайдера (SCM_DATA_ALG_ID),
список отзыва (SCM_DATA_BINARY).

Код ответа: SCM_REPLY_SET_CRL.
Возвращаемые данные:
нет.

Возможные ошибки:
криптопровайдер (алгоритм) не найден (SCM_ERROR_ITEM_NOT_FOUND).

### 5.2.6 Проверить сертификат.

Код команды: SCM_CMD_CERT_CHECK.
Входные параметры:
идентификатор криптопровайдера (SCM_DATA_ALG_ID),
параметры проверки (SCM_DATA_BYTE),
имя сервера (SCM_DATA_STRING),
сертификат (SCM_DATA_CERT).

Код ответа: SCM_REPLY_CERT_CHECK.
Возвращаемые данные:
результат проверки (SCM_DATA_BYTE).

Возможные ошибки:
криптопровайдер (алгоритм) не найден (SCM_ERROR_ITEM_NOT_FOUND).

Параметры проверки – это битовая маска из следующих значений:
SCM_CERT_CHECK_DATE – проверить срок действия,
SCM_CERT_CHECK_BASE – выполнить базовую проверку действительности сертификата против установленных в модуле сертификатов и списков отзыва (см. далее).
SCM_CERT_CHECK_TLS_CLIENT – проверить возможность использования сертификата в качестве клиентского для TLS соединения,
SCM_CERT_CHECK_TLS_SERVER – проверить возможность использования сертификата в качестве серверного для TLS соединения,
Дополнительные параметры проверки:
SCM_CERT_CHECK_IGNORE_REVOCATION_CHAIN – не проверять, отозваны ли сертификаты во всей цепочке, удостоверяющей данный сертификат,
SCM_CERT_CHECK_IGNORE_WRONG_USAGE – игнорировать целевое использование сертификатов при построении цепочки.

Имя сервера – адрес (обычно IP адрес или доменное имя) сервера. Используется, только если задан параметр SCM_CERT_CHECK_TLS_SERVER, чтобы проверить, что сертификат выдан именно для этого сервера. Параметр является обязательным. Если имя сервера проверять не надо (или не задан параметр SCM_CERT_CHECK_TLS_SERVER), то передается пустая строка (параметр нулевой длины).

Сертификат – сертификат для проверки.

Значения результата проверки: положительный – «1» или отрицательный – «0». При отрицательном результате проверки не посылается ответ «Ошибка».

ПРИМЕЧАНИЕ. Данная проверка не является средством аутентификации владельца сертификата. Для аутентификации необходимо использование криптоалгоритмов с участием секретного ключа владельца (как это делается, например, в TLS).

### 5.2.7 Начать шифрование.

Код команды: SCM_CMD_CRYPT_INIT.
Входные параметры:
идентификатор криптопровайдера (SCM_DATA_ALG_ID),
идентификатор ключа (SCM_DATA_KEY_ID, не обязательно),
материал для ключа (SCM_DATA_BINARY, не обязательно),
направление операции (SCM_DATA_BYTE),
вектор инициализации (SCM_DATA_BINARY, не обязательно).

Код ответа: SCM_REPLY_CRYPT_INIT.
Возвращаемые данные:
идентификатор сессии шифрования (SCM_DATA_HANDLE).

Возможные ошибки:
криптопровайдер (алгоритм) или ключ не найдены (SCM_ERROR_ITEM_NOT_FOUND).

Команде необходимо передать либо идентификатор ключа (см. описание команды «Задать ключ шифрования»), либо материал для выработки ключа (например, парольная фраза).

Значения для направления операции:
SCM_CIPHER_ENCRYPT – шифрование,
SCM_CIPHER_DECRYPT – расшифрование.

Вектор инициализации – это массив байтов, который задает начальное состояние сессии шифрования. Если какое-то сообщение было зашифровано с определенным вектором инициализации, то его необходимо расшифровывать с тем же вектором. Реальный размер вектора инициализации для разных алгоритмов может различаться. Если в команде передан вектор инициализации большего размера, то используются начальные байты вектора. Если передан вектор меньшего размера, то оставшиеся байты заполняются нулями. В частности, если вектор инициализации в команде отсутствует, то используется вектор, состоящий из нулевых байтов.

По окончании шифрования сессию необходимо завершить командой «Завершить шифрование».

### 5.2.8 Зашифровать/расшифровать.

Код команды: SCM_CMD_CRYPT_UPDATE.
Входные параметры:

идентификатор сессии шифрования (SCM_DATA_HANDLE),
данные для шифрования/расшифрования (SCM_DATA_BINARY).

Код ответа: SCM_REPLY_CRYPT_UPDATE.
Возвращаемые данные:
зашифрованные/расшифрованные данные (SCM_DATA_BINARY, не обязательно).

Возможные ошибки:
сессия не найдена (SCM_ERROR_ITEM_NOT_FOUND),
ошибка шифрования (SCM_ERROR_CRYPT).

В зависимости от используемого алгоритма размер зашифрованных/расшифрованных данных может отличаться от размера исходных данных. Возвращаемые данные могут отсутствовать. Они могут быть получены при следующем вызове команды или по команде «Завершить шифрование».

Команда поддерживает режим продолжения со стороны мастера и со стороны модуля (см. описание команды «Продолжение»).

### 5.2.9 Завершить шифрование.

Код команды: SCM_CMD_CRYPT_FINAL.
Входные параметры:
идентификатор сессии шифрования (SCM_DATA_HANDLE).

Код ответа: SCM_REPLY_CRYPT_FINAL.
Возвращаемые данные:
последние зашифрованные/расшифрованные данные (SCM_DATA_BINARY, не обязательно).

Возможные ошибки:
сессия не найдена (SCM_ERROR_ITEM_NOT_FOUND),
ошибка шифрования (SCM_ERROR_CRYPT).

Последние данные – это данные, которые могли быть закэшированы в сессии, они могут отсутствовать (зависит от алгоритма шифрования).

После данной команды идентификатор сессии становится недействительным.

### 5.2.10 Начать хэш.

Код команды: SCM_CMD_HASH_INIT.
Входные параметры:
идентификатор криптопровайдера (SCM_DATA_ALG_ID),
тип хэша (SCM_DATA_INTEGER, не обязательно).

Код ответа: SCM_REPLY_HASH_INIT.

Возвращаемые данные:
идентификатор сессии хэширования (SCM_DATA_HANDLE).

Возможные ошибки:
криптопровайдер (алгоритм) не найден (SCM_ERROR_ITEM_NOT_FOUND).

Если тип хэша не указан, то используется тип по умолчанию для указанного криптопровайдера.

По окончании хэширования сессию необходимо завершить командой «Завершить хэш».

### 5.2.11 Продолжить хэш.

Код команды: SCM_CMD_HASH_UPDATE.
Входные параметры:
идентификатор сессии хэширования (SCM_DATA_HANDLE),
данные для хэширования (SCM_DATA_BINARY).

Код ответа: SCM_REPLY_HASH_UPDATE.
Возвращаемые данные: нет.

Возможные ошибки:
сессия не найдена (SCM_ERROR_ITEM_NOT_FOUND),
ошибка шифрования (SCM_ERROR_CRYPT).

Команда поддерживает режим продолжения со стороны мастера (см. описание команды «Продолжение»).

### 5.2.12 Завершить хэш.

Код команды: SCM_CMD_HASH_FINAL.
Входные параметры:
идентификатор сессии хэширования (SCM_DATA_HANDLE).

Код ответа: SCM_REPLY_HASH_FINAL.
Возвращаемые данные:
вычисленное значение хэша (SCM_DATA_BINARY).

Возможные ошибки:
сессия не найдена (SCM_ERROR_ITEM_NOT_FOUND),
ошибка шифрования (SCM_ERROR_CRYPT).

После данной команды идентификатор сессии становится недействительным.

### 5.2.13 Начать подпись.

Код команды: SCM_CMD_SIGN_INIT.
Входные параметры:

идентификатор криптопровайдера (SCM_DATA_ALG_ID),
идентификатор ключа (SCM_DATA_KEY_ID).

Код ответа: SCM_REPLY_SIGN_INIT.
Возвращаемые данные:
идентификатор сессии подписи (SCM_DATA_HANDLE).

Возможные ошибки:
криптопровайдер (алгоритм) или ключ не найдены (SCM_ERROR_ITEM_NOT_FOUND).

По окончании подписи сессию необходимо завершить командой «Завершить подпись».

### 5.2.14 Продолжить подпись.

Код команды: SCM_CMD_SIGN_UPDATE.
Входные параметры:
идентификатор сессии подписи (SCM_DATA_HANDLE),
данные для подписи (SCM_DATA_BINARY).

Код ответа: SCM_REPLY_SIGN_UPDATE.
Возвращаемые данные: нет.

Возможные ошибки:
сессия не найдена (SCM_ERROR_ITEM_NOT_FOUND),
ошибка шифрования (SCM_ERROR_CRYPT).

Команда поддерживает режим продолжения со стороны мастера (см. описание команды «Продолжение»).

### 5.2.15 Завершить подпись.

Код команды: SCM_CMD_SIGN_FINAL.
Входные параметры:
идентификатор сессии подписи (SCM_DATA_HANDLE).

Код ответа: SCM_REPLY_SIGN_FINAL.
Возвращаемые данные:
вычисленное значение подписи (SCM_DATA_BINARY).

Возможные ошибки:
сессия не найдена (SCM_ERROR_ITEM_NOT_FOUND),
ошибка шифрования (SCM_ERROR_CRYPT).

После данной команды идентификатор сессии становится недействительным.

### 5.2.16 Начать проверку подписи.

Код команды: SCM_CMD_VERIFY_INIT.
Входные параметры:
идентификатор криптопровайдера (SCM_DATA_ALG_ID),
идентификатор ключа (SCM_DATA_KEY_ID).

Код ответа: SCM_REPLY_VERIFY_INIT.
Возвращаемые данные:
идентификатор сессии проверки подписи (SCM_DATA_HANDLE).

Возможные ошибки:
криптопровайдер (алгоритм) или ключ не найдены (SCM_ERROR_ITEM_NOT_FOUND).

По окончании проверки подписи сессию необходимо завершить командой «Завершить проверку подписи».

### 5.2.17 Продолжить проверку подписи.

Код команды: SCM_CMD_VERIFY_UPDATE.
Входные параметры:
идентификатор сессии проверки подписи (SCM_DATA_HANDLE),
данные для проверки подписи (SCM_DATA_BINARY).

Код ответа: SCM_REPLY_VERIFY_UPDATE.
Возвращаемые данные: нет.

Возможные ошибки:
сессия не найдена (SCM_ERROR_ITEM_NOT_FOUND),
ошибка шифрования (SCM_ERROR_CRYPT).

Команда поддерживает режим продолжения со стороны мастера (см. описание команды «Продолжение»).

### 5.2.18 Завершить проверку подписи.

Код команды: SCM_CMD_VERIFY_FINAL.
Входные параметры:
идентификатор сессии проверки подписи (SCM_DATA_HANDLE).

Код ответа: SCM_REPLY_VERIFY_FINAL.
Возвращаемые данные:
результат проверки подписи (SCM_DATA_BYTE).

Возможные ошибки:
сессия не найдена (SCM_ERROR_ITEM_NOT_FOUND),
ошибка шифрования (SCM_ERROR_CRYPT).

Значения результата проверки: положительный – «1» или отрицательный – «0».

После данной команды идентификатор сессии становится недействительным.

### 5.2.19 Получить случайные данные.

Код команды: SCM_CMD_GET_RND.
Входные параметры:
идентификатор криптопровайдера (SCM_DATA_ALG_ID),
размер запрашиваемых данных (SCM_DATA_INTEGER).

Код ответа: SCM_REPLY_GET_RND.
Возвращаемые данные:
случайные данные (SCM_DATA_BINARY).

Возможные ошибки:
криптопровайдер не найден (SCM_ERROR_ITEM_NOT_FOUND),
ошибка шифрования (SCM_ERROR_CRYPT).

В одной команде можно запросить не более 512 байтов.
Первая команда SCM_CMD_GET_RND после загрузки модуля может потребовать больше времени для инициализации генератора случайных данных.

### 5.2.20 Задать ключ шифрования.

Код команды: SCM_CMD_KEY_SET.
Входные параметры:
идентификатор криптопровайдера (SCM_DATA_ALG_ID),
идентификатор ключа (SCM_DATA_KEY_ID),
имя/описание ключа (SCM_DATA_STRING, не обязательно),
флаги (SCM_DATA_INTEGER, битовая маска),
данные ключа (SCM_DATA_BINARY).

Код ответа: SCM_REPLY_KEY_SET.
Возвращаемые данные: нет.

Возможные ошибки:
ошибка шифрования (SCM_ERROR_CRYPT).

Данная команда явно задает ключ для симметричного шифрования с указанным идентификатором. В идентификаторе ключа назначение должно быть SCM_KEY_USAGE_CYPHER (симметричное шифрование). Этот идентификатор затем может быть использован в команде «Начать шифрование».
Не все криптопровайдеры поддерживают явное задание ключа. Проверить эту возможность можно, запросив командой «Получить параметр модуля» значение параметра «crypto.key_set_supported.<PROVIDER_ID>» (см. Приложение 1 «Описание параметров криптомодуля»). Если данная возможность не поддерживается, то для шифрования доступен только вариант команды «Начать шифрование» с материалом для ключа (парольной фразой).

Имя (описание) ключа – необязательный параметр. Если он задан, то его значение будет возвращено в качестве описание ключа в ответе на команду «Получить информацию о ключе» с указанным выше идентификатором ключа.

Флаги – битовая маска значений:

SCM_KEY_SET_FLAG_PERSISTENT – постоянный ключ (ключ будет сохранен и доступен после перезагрузки модуля). Если этот флаг не задан, то ключ будет временным (исчезнет после перезагрузки модуля).

Данные ключа – это сам ключ для симметричного шифрования, который будет использоваться криптопровайдером. Длина необходимых данных определяется криптопровайдером. Недостающие данные будут дополнены нулями, лишние данные – отброшены.

Заданный данной командой ключ можно в любой момент удалить командой «Удалить ключ шифрования».

### 5.2.21 Удалить ключ шифрования.

Код команды: SCM_CMD_KEY_DELETE.
Входные параметры:
идентификатор криптопровайдера (SCM_DATA_ALG_ID),
идентификатор ключа (SCM_DATA_KEY_ID).

Код ответа: SCM_REPLY_KEY_DELETE.
Возвращаемые данные: нет.

Возможные ошибки:
криптопровайдер или ключ не найдены (SCM_ERROR_ITEM_NOT_FOUND).

Команда удаляет ключ для симметричного шифрования, заданный ранее командой «Задать ключ шифрования».

### 5.2.22 Черный список.

Код команды: SCM_CMD_BLACK_LIST.
Входные параметры:
код операции (SCM_DATA_INTEGER),
идентификатор криптопровайдера (SCM_DATA_ALG_ID),
сертификат (SCM_DATA_CERT, не обязательно),
номер сертификата (SCM_DATA_INTEGER, не обязательно).

Код ответа: SCM_REPLY_BLACK_LIST.

Возвращаемые данные:
сертификат (SCM_DATA_CERT, не обязательно).

Возможные ошибки:
криптопровайдер или запрошенный сертификат не найдены (SCM_ERROR_ITEM_NOT_FOUND).

Допустимые коды операции:
SCM_BLACK_LIST_ADD – добавить сертификат в черный список (сертификат должен присутствовать во входных параметрах),
SCM_BLACK_LIST_DELETE – удалить сертификат из черного списка (во входных параметрах должен присутствовать сертификат или его номер; если указаны оба, то используется сам сертификат),
SCM_BLACK_LIST_GET – получить сертификат из черного списка по его номеру (во входных параметрах указывается только номер, но не сам сертификат); для получения всего списка необходимо последовательно перебирать номера, пока модуль не вернет ошибку.

Черный список используется при любой проверке сертификатов: если сертификат включен в черный список, то этот сертификат отвергается до каких-бы то ни было проверок. Эта проверка выполняется до всех прочих проверок сертификата.
Это не очень надежный способ блокировки сертификатов. Он может использоваться для оперативного запрета пользования каким-то сертификатом, пока соответствующий УЦ не выпустит новый список отзыва.

Используя параметр модуля «crypto.use_black_list» (см. описание команды «Получить параметр модуля»), можно также включать и выключать применение всего черного списка сертификатов.

### 5.2.23 Белый список.

Код команды: SCM_CMD_WHITE_LIST.
Входные параметры:
код операции (SCM_DATA_INTEGER),
идентификатор криптопровайдера (SCM_DATA_ALG_ID),
тип элемента (SCM_DATA_INTEGER),
номер элемента (SCM_DATA_INTEGER, не обязательно),
данные элемента (SCM_DATA_BINARY, не обязательно),
дополнительные данные элемента (SCM_DATA_BINARY, не обязательно).

Код ответа: SCM_REPLY_WHITE_LIST.

Возвращаемые данные:
данные элемента (SCM_DATA_BINARY, не обязательно),
дополнительные данные элемента (SCM_DATA_BINARY, не обязательно).

Возможные ошибки:
криптопровайдер или запрошенный элемент не найдены (SCM_ERROR_ITEM_NOT_FOUND).

Допустимые коды операции:

SCM_WHITE_LIST_ADD – добавить элемент в белый список (данные должны присутствовать во входных параметрах),

SCM_WHITE_LIST_DELETE – удалить элемент из белого списка (во входных параметрах должен присутствовать данные элемента или его номер; если указаны оба, то используются данные),

SCM_WHITE_LIST_GET – получить элемент из белого списка по его номеру (во входных параметрах указывается только номер, но не данные); для получения всего списка необходимо последовательно перебирать номера, пока модуль не вернет ошибку.

Допустимые типы элементов:

SCM_WHITE_LIST_ITEM_TYPE_CERT – сертификат,

SCM_WHITE_LIST_ITEM_TYPE_CERT – OID.

Номер элемента может указываться только при получении или удалении элемента.

Белый список используется при проверке сертификатов: если сертификат не удовлетворяет белому список, то этот сертификат отвергается. Данная проверка осуществляется после всех остальных проверок. Т.е., чтобы сертификат был принят, он должен сначала пройти все запрошенные проверки и затем пройти проверку по белому списку.

Элементами белого списка могут быть сертификаты или OID (идентфикаторы расширений сертификатов). При проверке какого либо сертификата проверяется, что он идентичен одному из сертификатов в белом списке или содержит расширение с OID из белого списка. При этом, если в белом списке для OID было также задано само значение (тело) расширения, то проверяется и точное совпадение этого значения с расширением в проверяемом сертификате.

Если элемент списка сертификат, то при добавлении элемента в список сертификат указывается в первом буфере данных («данные элемента»), второй буфер («дополнительные данные») не требуется (игнорируется). Аналогично передается сертификат при его удалении из списка (если не производится удаление по номеру). При получении сертификата из белого списка он возвращается в первом параметре («данные элемента»), второй параметр («дополнительные данные») также присутствует, но является пустым.

Если элемент списка OID, то при добавлении элемента в список строковое представление OID (оно может содержать или не содержать концевой 0) указывается в первом буфере данных («данные элемента»). Если необходимо задать конкретное значение соответствующего расширения, то оно передается без всяких модификаций в бинарном виде во втором буфере («дополнительные данные»). Если значение расширения задавать не требуется, то второй буфер может отсутствовать или быть пустым. Аналогично передается OID при его удалении из списка (если не производится удаление по номеру). При получении OID из белого списка его строковое представление (всегда содержащее концевой 0) возвращается в первом параметре («данные элемента»). Второй параметр («дополнительные данные») также всегда присутствует и либо является пустым, если значение расширения не было задано, либо содержит значение расширения.

Нумерация элементов разных типов независимая. При добавлении нового элемента в список он не обязательно становится последним по номеру!

Используя параметр модуля «crypto.use_white_list» (см. описание команды «Получить параметр модуля»), можно также включать и выключать применение всего белого списка сертификатов.

## 5.3 Функции для организации TLS.

TLS служит для защиты какого-либо коммуникационного канала (например, TCP соединения, NFC, Bluetooth и др.). Сам канал (соединение) создается мастером. Затем мастер посылает модулю команду «Начать TLS сессию» (указывая, является ли мастер/модуль TLS клиентом/сервером), после чего является посредником в передаче данных между модулем и удаленной стороной, используя команду «Установить TLS сессию». Как только ответ на команду «Установить TLS сессию» сообщит об установлении сессии, мастер может обмениваться данными с удаленной стороной в рамках защищенной сессии. Для этого все данные, получаемые от удаленной стороны, передаются модулю командой «Расшифровать TLS данные», которая возвращает мастеру расшифрованные данные. Аналогично, все данные, которые мастер хочет передать удаленной стороне, передаются модулю командой «Зашифровать TLS данные», которая возвращает зашифрованные данные для передачи удаленной стороне. Для закрытия TLS сессии (по результатам обмена данными с удаленной стороной или в результате закрытия канала) необходимо завершить TLS сессию командой «Завершить TLS сессию» (которая может вернуть некоторые данные для передачи удаленной стороне, если канал еще не закрыт, также могут потребоваться повторные команды «Завершить TLS сессию»). Закрытие самого канала осуществляется мастером.

Автоматическое переустановление сессии (renegotiation) не поддерживается.

### 5.3.1 Начать TLS сессию.

Код команды: SCM_CMD_TLS_INIT.
Входные параметры:
- транспорт (SCM_DATA_BYTE),
- версия протокола (SCM_DATA_BYTE),
- клиент или сервер (SCM_DATA_BYTE),
- необходимость аутентификации (SCM_DATA_BYTE),
- параметры аутентификации (SCM_DATA_BYTE),
- идентификатор криптопровайдера (SCM_DATA_ALG_ID),
- идентификатор ключа (SCM_DATA_KEY_ID, не обязательно),
- адрес удаленной стороны (SCM_DATA_STRING, не обязательно).

Код ответа: SCM_REPLY_TLS_INIT.
Возвращаемые данные:
- идентификатор TLS сессии (SCM_DATA_HANDLE),

данные для передачи удаленной стороне (SCM_DATA_BINARY, не обязательно).

Возможные ошибки:

криптопровайдер (алгоритм) или ключ не найдены (SCM_ERROR_ITEM_NOT_FOUND).

Транспорт указывает, каким образом осуществляется связь. Возможные значения:

SCM_CONNECTION_TRANSPORT_UNKNOWN - конкретный транспорт не имеет значения;

SCM_CONNECTION_TRANSPORT_TCP_CLIENT - данные передаются по TCP соединению, мастер является TCP клиентом.

Данный параметр имеет значение только при установлении сервисного соединения (см. команду «Сервисное соединение»).

Версия протокола – битовая маска значений:

SCM_CONNECTION_PROTOCOL_TLS_1_0 (TLS версии 1.0),

SCM_CONNECTION_PROTOCOL_TLS_1_1 (TLS версии 1.1),

SCM_CONNECTION_PROTOCOL_TLS_1_2 (TLS версии 1.2),

SCM_CONNECTION_PROTOCOL_GMTLS (только для криптопровадера SCM_CRYPTO_PROVIDER_TYPE_CN, защищенное соединение по GMT 0024-2014).

Флаг «Клиент или сервер» указывает, является ли мастер/модуль TLS клиентом или сервером. Возможные значения:

SCM_CONNECTION_SERVER - сервер,

SCM_CONNECTION_CLIENT - клиент.

Необходимость аутентификации (проверки сертификата удаленной стороны) - одно из значений:

SCM_TLS_VERIFY_NONE (аутентификация не требуется),

SCM_TLS_VERIFY_IF_PRESENT (аутентификация, если удаленная сторона предоставила сертификат),

SCM_TLS_VERIFY_ALWAYS (всегда требовать сертификат удаленной стороны).

Параметры аутентификации указывают, каким образом надо проверять сертификат удаленной стороны. Возможные значения (битовая маска):

SCM_TLS_FLAG_MUTUAL_AUTH (проверять сертификаты на обеих сторонах),

SCM_TLS_FLAG_IGNORE_REVOCATION (при проверке игнорировать наличие списков отзыва, см. также описание параметра модуля «crypto.tls_ignore_crl» а Приложении 1 «Описание параметров криптомодуля»),

SCM_TLS_FLAG_IGNORE_UNKNOWN_CA (при проверке допускать УЦ, чьи сертификаты не установлены в модуле),

SCM_TLS_FLAG_IGNORE_WRONG_USAGE (при проверке позволять несоответствие использования указанному в сертификате),

SCM_TLS_FLAG_IGNORE_CERT_CN_INVALID (при проверке на стороне клиента не проверять соответствие имени сервера заданного как адрес удаленной стороны имени указанному в сертификате),
SCM_TLS_FLAG_IGNORE_CERT_DATE_INVALID (не проверять срок действия сертификатов).

Идентификатор ключа указывать не обязательно (будет использован ключ по умолчанию).

Адрес удаленной стороны может использоваться клиентом при проверке сертификата сервера (см. также флаг SCM_TLS_FLAG_IGNORE_CERT_CN_INVALID в параметрах аутентификации). На стороне сервера этот параметр не требуется.

По окончании работы сессию необходимо завершить командой «Завершить TLS сессию».

Возвращаемый идентификатор TLS сессии может использоваться в командах, требующих идентификатора пользователя, чтобы указать, что некоторая команда исполняется от имени пользователя, с которым установлено TLS соединение.

### 5.3.2 Установить TLS сессию.

Код команды: SCM_CMD_TLS_SETUP.
Входные параметры:
идентификатор TLS сессии (SCM_DATA_HANDLE),
данные полученные от удаленной стороны (SCM_DATA_BINARY, не обязательно).

Код ответа: SCM_REPLY_TLS_SETUP.
Возвращаемые данные:
статус соединения (SCM_DATA_BYTE),
данные для передачи удаленной стороне (SCM_DATA_BINARY, не обязательно).

Возможные ошибки:
идентификатор сессии не найден (SCM_ERROR_ITEM_NOT_FOUND),
ошибка шифрования (SCM_ERROR_CRYPT).

Возможные значения статуса соединения:
SCM_CONNECTION_STATUS_CONNECTED - соединение установлено,
SCM_CONNECTION_STATUS_CLOSED - соединение закрыто,
SCM_CONNECTION_STATUS_ERROR - ошибка соединения,
SCM_CONNECTION_STATUS_NEED_MORE_DATA - требуются дополнительные данные.

Начальное состояние сессии после команды «Начать TLS сессию» - SCM_CONNECTION_STATUS_NEED_MORE_DATA. Мастер передает данные между модулем и удаленной стороной, неоднократно используя данную команду до тех пор,

пока статус соединения не станет либо SCM_CONNECTION_STATUS_CONNECTED (в этом случае мастер может начинать обмен данными с удаленной стороной, используя команды «Зашифровать TLS данные» и «Расшифровать TLS данные»), либо SCM_CONNECTION_STATUS_CLOSED или SCM_CONNECTION_STATUS_ERROR (необходимо завершить сессию командой «Завершить TLS сессию» и закрыть канал).

Команда поддерживает режим продолжения со стороны мастера и со стороны модуля (см. описание команды «Продолжение»).

### 5.3.3 Получить удаленный сертификат TLS.

Код команды: SCM_CMD_TLS_GET_REMOTE.
Входные параметры:
идентификатор TLS сессии (SCM_DATA_HANDLE).

Код ответа: SCM_REPLY_TLS_GET_REMOTE.
Возвращаемые данные:
статус соединения (SCM_DATA_BYTE),
сертификат (SCM_DATA_CERT, не обязательно),
роль владельца сертификата (SCM_DATA_BYTE, не обязательно).

Возможные ошибки:
идентификатор сессии не найден (SCM_ERROR_ITEM_NOT_FOUND).
Также необходимо контролировать статус соединения (см. описание команды «Установить TLS сессию»).

О роли владельца сертификата см. также команду «Проверить сертификат».

Данная команда позволяет получить сертификат, предъявленный удаленной стороной. Может вызываться после успешного установления TLS сессии командами «Установить TLS сессию».
Сертификат может отсутствовать. Если сертификата нет, или в команде «Начать TLS сессию» была отключена проверка сертификата, то роль также будет отсутствовать.

### 5.3.4 Зашифровать TLS данные.

Код команды: SCM_CMD_TLS_ENCRYPT.
Входные параметры:
идентификатор TLS сессии (SCM_DATA_HANDLE),
данные для шифрования (SCM_DATA_BINARY).

Код ответа: SCM_REPLY_TLS_ENCRYPT.
Возвращаемые данные:
статус соединения (SCM_DATA_BYTE),
зашифрованные данные (SCM_DATA_BINARY).

Возможные ошибки:

идентификатор сессии не найден (SCM_ERROR_ITEM_NOT_FOUND),
ошибка шифрования (SCM_ERROR_CRYPT).

Также необходимо контролировать статус соединения (см. описание команды «Установить TLS сессию»).

Зашифрованные данные передаются удаленной стороне. Их размер может превышать размер предоставленных в команде данных для шифрования.

Команда поддерживает режим продолжения со стороны мастера и со стороны модуля (см. описание команды «Продолжение»).

### 5.3.5 Расшифровать TLS данные.

Код команды: SCM_CMD_TLS_DECRYPT.
Входные параметры:
идентификатор TLS сессии (SCM_DATA_HANDLE),
данные для расшифрования (SCM_DATA_BINARY).

Код ответа: SCM_REPLY_TLS_DECRYPT.
Возвращаемые данные:
статус соединения (SCM_DATA_BYTE),
расшифрованные данные (SCM_DATA_BINARY, не обязательно).

Возможные ошибки:
идентификатор сессии не найден (SCM_ERROR_ITEM_NOT_FOUND),
ошибка шифрования (SCM_ERROR_CRYPT).

Также необходимо контролировать статус соединения (см. описание команды «Установить TLS сессию»).

Зашифрованные данные получены от удаленной стороны.

Расшифрованные данные могут отсутствовать, если требуются дополнительные данные от удаленной стороны (текущие данные буферизованы в модуле, повторно их посылать не требуется).

Статус соединения может быть SCM_CONNECTION_STATUS_CLOSED, но расшифрованные данные могут присутствовать. В этом случае завершать TLS сессию следует после обработки всех расшифрованных данных.

Команда поддерживает режим продолжения со стороны мастера и со стороны модуля (см. описание команды «Продолжение»).

### 5.3.6 Завершить TLS сессию.

Код команды: SCM_CMD_TLS_CLOSE.
Входные параметры:
идентификатор TLS сессии (SCM_DATA_HANDLE),
причина (SCM_DATA_BYTE),
данные полученные от удаленной стороны (SCM_DATA_BINARY, не обязательно).

Код ответа: SCM_REPLY_TLS_CLOSE.
Возвращаемые данные:
статус соединения (SCM_DATA_BYTE),
данные для передачи удаленной стороне (SCM_DATA_BINARY, не обязательно).

Возможные ошибки:
идентификатор сессии не найден (SCM_ERROR_ITEM_NOT_FOUND),
ошибка шифрования (SCM_ERROR_CRYPT).

Причина завершения сессии определяет процедуру закрытия. Возможные значения:

SCM_SESSION_CLOSE_REASON_COMPLETED – сессия завершена согласно протоколу между мастером и удаленной стороной, канал передачи данных еще открыт, необходимо провести корректное завершение TLS сессии.

SCM_SESSION_CLOSE_REASON_DISCONNECTED – канал передачи закрыт, корректное завершение TLS сессии невозможно.

SCM_SESSION_CLOSE_REASON_CLOSING – сессия принудительно завершается по какой-то причине, канал передачи данных еще открыт, необходимо провести корректное завершение TLS сессии (если возможно).

SCM_SESSION_CLOSE_REASON_ABORTED – сессия принудительно завершена по какой-то причине, корректное завершение TLS сессии невозможно.

Данные полученные от удаленной стороны могут появиться только при повторных вызовах данной команды.

Данные для передачи удаленной стороне могут отсутствовать (если корректное завершение TLS сессии невозможно, или TLS соединение уже фактически было завершено внутри модуля в команде «Расшифровать TLS данные»). Если они присутствуют, и канал еще открыт, то эти данные необходимо передать удаленной стороне независимо от значения статуса соединения.

Команда может вызываться несколько раз, пока статус соединения не покажет, что сессия завершена или пока не вернется ошибка SCM_ERROR_ITEM_NOT_FOUND.

Команда должна быть вызвана (по крайней мере один раз), даже если одна из предыдущих команд ранее вернула статус, показывающий, что соединение закрыто.

## 5.4 CMS.

Если необходимо передать сообщение по открытому каналу, то его можно упаковать в CMS (Cryptographic Message Syntax) контейнер. Сообщение в CMS будет подписано и, если необходимо, зашифровано. Информация о ключах подписи и шифрования также будет включена в контейнер. Принимающая сторона должна уметь расшифровывать, проверять и раскодировать CMS сообщения.

Поддерживаются либо простые подписанные сообщения (signed data), либо подписанные сообщения, запечатанные и зашифрованные на ключе получателя (enveloped and encrypted data). Сертификат отправителя, как правило, должен быть вложен в сообщение.

### 5.4.1 Закодировать CMS.

Код команды: SCM_CMD_CMS_ENCRYPT.
Входные параметры:
параметры кодирования (SCM_DATA_BYTE),
идентификатор криптопровайдера (SCM_DATA_ALG_ID),
идентификатор ключа (SCM_DATA_KEY_ID),
сертификаты получателя (последовательность SCM_DATA_CERT, не обязательно),
сообщение (SCM_DATA_BINARY).

Код ответа: SCM_REPLY_CMS_ENCRYPT.
Возвращаемые данные:
данные CMS (SCM_DATA_BINARY).

Возможные ошибки:
криптопровайдер (алгоритм) или ключ не найдены (SCM_ERROR_ITEM_NOT_FOUND),
ошибка шифрования (SCM_ERROR_CRYPT).

Параметры кодирования – битовая маска следующих значений:
SCM_CMS_FLAG_SIGNING_TIME – включить в сообщение время подписи,
SCM_CMS_FLAG_DO_NOT_SIGN – не подписывать CMS (при этом игнорируется идентификатор ключа и не включается время подписи; шифрование на сертификате получателя возможно).
SCM_CMS_FLAG_DETACHED – создать отсоединенную подпись. Т.е., CMS не будет содержать самого сообщения, только подпись.
SCM_CMS_FLAG_ENCRYPT_BEFORE_SIGN – зашифровать сообщение, потом подписать зашифрованное сообщение (обычный порядок обратный). Это аналогично двум последовательным вызовам данной команды: сначала с флагом SCM_CMS_FLAG_DO_NOT_SIGN и сертификатами получателей, потом без флага SCM_CMS_FLAG_DO_NOT_SIGN и без сертификатов получателей.
SCM_CMS_FLAG_ENCRYPT_RAW – зашифровать сообщение как есть, т.е., не упаковывать его дополнительно в подписанный контейнер. Данный флаг имеет смысл только с флагами SCM_CMS_FLAG_DO_NOT_SIGN и SCM_CMS_FLAG_ENCRYPT_BEFORE_SIGN, в остальных случаях игнорируется. Если вызвать команду с флагами SCM_CMS_FLAG_DO_NOT_SIGN или SCM_CMS_FLAG_ENCRYPT_BEFORE_SIGN, но без SCM_CMS_FLAG_ENCRYPT_RAW, то зашифрованное сообщение будет помещено в подписанный контейнер с пустым списком подписантов.

Идентификатор ключа указывает, каким ключом подписать CMS сообщение.

Если сертификаты получателя не указаны, то сообщение будет только подписано указанным ключом. Если сертификаты получателя указаны, то подписанное сообщение будет также запечатано (enveloped) и зашифровано таким образом, что расшифровать его смогут только владельцы сертификатов.

Команда поддерживает режим продолжения со стороны мастера и со стороны модуля (см. описание команды «Продолжение»).

### 5.4.2 Раскодировать CMS.

Код команды: SCM_CMD_CMS_DECRYPT.
Входные параметры:
параметры проверки (SCM_DATA_BYTE),
идентификатор криптопровайдера (SCM_DATA_ALG_ID),
сертификат отправителя (SCM_DATA_CERT, необязательно),
данные CMS (SCM_DATA_BINARY),
отсоединенные данные (SCM_DATA_BINARY, не обязательно).

Код ответа: SCM_REPLY_CMS_DECRYPT.
Возвращаемые данные:
результат проверки (SCM_DATA_BYTE),
раскодированное сообщение (SCM_DATA_BINARY).

Возможные ошибки:
криптопровайдер (алгоритм) не найден (SCM_ERROR_ITEM_NOT_FOUND),
ошибка шифрования (SCM_ERROR_CRYPT).

Параметры проверки – битовая маска следующих значений:
SCM_CMS_FLAG_SIGNING_TIME – проверять время подписи.
SCM_CMS_FLAG_CHECK_SIGNER – проверять сертификат отправителя.
SCM_CMS_FLAG_DO_NOT_SIGN – не проверять подпись CMS (при этом игнорируется идентификатор ключа и не включается время подписи; шифрование на сертификате получателя возможно).
SCM_CMS_FLAG_DETACHED – проверить отсоединенную подпись. Т.е., сам CMS не будет содержать самого сообщения, только подпись, а сообщение будет передано в отдельном параметре (отсоединенные данные). Это значение можно не указывать – если мастер предоставил отсоединенные данные, то модуль будет рассматривать этот CMS как отсоединенный.
SCM_CMS_FLAG_ENCRYPT_BEFORE_SIGN – сообщение сначала было зашифровано, а потом подписано. Т.е., использование этого флага аналогично двум последовательным вызовам данной команды: сначала только проверка подписи, затем только расшифровка (с флагом SCM_CMS_FLAG_DO_NOT_SIGN).
SCM_CMS_FLAG_ENCRYPT_RAW – сообщение было зашифровано как есть, т.е., не упаковано дополнительно в подписанный контейнер, поэтому расшифрованные данные будут возвращены как есть, без дополнительной распаковки. Данный флаг имеет смысл только с флагами SCM_CMS_FLAG_DO_NOT_SIGN и SCM_CMS_FLAG_ENCRYPT_BEFORE_SIGN, в остальных случаях игнорируется.

Если в параметрах команды указан сертификат отправителя, то проверка подписи сообщения происходит только с использованием открытого ключа этого сертификата. Сертификат также проверяется на действительность.

Если сертификат отправителя не указан, то используется сертификат, находящийся в самом сообщении. Для успешной проверки этого сертификата он должен быть выпущен УЦ, чей сертификат установлен в модуле.

Если сообщение зашифровано, то сертификат, на открытом ключе которого было произведено шифрование, должен быть установлен в модуле вместе с секретным ключом.

Раскодированное сообщение может иметь нулевую длину, если расшифровка завершилась неудачей.

Значения результата проверки:

SCM_CMS_RESULT_SIGNATURE_NOT_VERIFIED – подпись не проверена (неверная подпись или не найден сертификат); при этом раскодированное сообщение может присутствовать,

SCM_CMS_RESULT_SIGNATURE_VERIFIED – подпись проверена (если явно не был указан сертификат отправителя, то это слабый результат, практически означающий, что сообщение просто корректно декодировано),

SCM_CMS_RESULT_CERTIFICATE_VERIFIED – проверена подпись и сертификат отправителя (указанный в команде или извлеченный из сообщения); действительность сертификата определялась на текущий момент,

SCM_CMS_RESULT_TIME_VERIFIED – проверены подпись и сертификат отправителя (указанный в команде или извлеченный из сообщения); действительность сертификата определялась на момент подписи сообщения. Для получения такого результата в параметрах проверки должен присутствовать флаг SCM_CMS_FLAG_SIGNING_TIME.

Команда поддерживает режим продолжения со стороны мастера и со стороны модуля (см. описание команды «Продолжение»).

## 5.5 Функции хранения данных.

Модуль может использоваться для хранения каких-либо данных (разбитых на документы/записи). Для этого в модуле создаются хранилища различных типов.

**Атрибуты хранилища:**

Тип хранилища (SCM_DATA_BYTE).

Идентификатор (имя) хранилища (SCM_DATA_STRING, максимальная длина – 32 байта).

Флаги (SCM_DATA_INTEGER, битовая маска).

Флаг «встроенное хранилище» (SCM_STORAGE_FLAG_INTERNAL) - некоторые хранилища могут создаваться автоматически ПО модуля исключительно для своих нужд (недоступны для мастера).

Флаг «только для чтения» (SCM_STORAGE_FLAG_READ_ONLY) - содержимое хранилища инициализируется при его создании.

Флаг цикличности (SCM_STORAGE_FLAG_CYCLIC) - если указано максимальное количество записей, то при добавлении новой записи (если это поддерживается типом хранилища) и достижении максимального количества запись будет производиться в начало хранилища.

Флаг шифрования (SCM_STORAGE_FLAG_ENCRYPT). Надо ли шифровать записи. Шифрование производится с использованием внутреннего ключа определенным алгоритмом по умолчанию.

Флаг «подписывать» (SCM_STORAGE_FLAG_SIGN). Записи будут подписываться всеми алгоритмами, для которых в модуле заданы соответствующие ключи.

Флаг «системное хранилище» (SCM_STORAGE_FLAG_SYSTEM) - некоторые хранилища могут создаваться автоматически ПО модуля и быть доступными для мастера.

Максимальное количество записей (SCM_DATA_INTEGER). Если 0 – то произвольное.

Фиксированный размер записи (SCM_DATA_INTEGER). Если 0 – то запись может быть произвольного размера. Некоторые типа хранилищ могут требовать фиксированного размера записей (см. ниже).

Количество записей (SCM_DATA_INTEGER). Сколько было записано. Для хранилищ с нефиксированным количеством записей обновляется после добавления новых записей.

Начальная позиция (SCM_DATA_INTEGER). Номер первой записи. Для нециклического хранилища – всегда 0 (не используется).

Временная метка (SCM_DATA_TIME). Время последнего доступа и изменения.

**Атрибуты записей** (слотов, документов) в хранилище:

Номер записи (SCM_DATA_INTEGER).

Имя записи (SCM_DATA_STRING, максимальная длина 32 байта).

Размер данных (SCM_DATA_INTEGER).

Временная метка (SCM_DATA_TIME). Время последнего изменения записи.

Флаги (SCM_DATA_INTEGER, битовая маска).

Флаг подтверждения (SCM_STORAGE_ITEM_FLAG_CONFIRMATION). Например, может использоваться для простой фиксации факта успешной передачи данных на какой-то сервер. Если от сервера получаются некоторые квитанции подтверждения, то их можно сохранять как отдельные атрибуты записи.

Подписи (SCM_DATA_BINARY).

Квитанция подтверждения (SCM_DATA_BINARY).

Типы хранилищ:

**Состояние** (SCM_STORAGE_TYPE_STATE). Данное хранилище имеет фиксированное количество ячеек (слотов) для хранения данных. В нем можно хранить какие-то фиксированные блоки информации. Требуется задавать фиксированный размер записи.

**Журнал** (SCM_STORAGE_TYPE_LOG). Дополняемое хранилище. Операции записи добавляют данные в хранилище.

**Документы** (SCM_STORAGE_TYPE_DOCUMENT). Дополняемое хранилище. Операции записи добавляют данные в хранилище. Практически тот же тип, что и журнал (самостоятельный тип сделан, чтобы отделить от журналов).

Имена хранилищ и записей могут не оканчиваться нулевым байтом. Мастер может указывать имена произвольной длины, модуль будет их усекать или дополнять

нулями до фиксированного размера. Модуль, когда выдает имя хранилища, всегда передает фиксированный размер.

Если записи в хранилище шифруются, то при удалении ключей в модуле (например, в результате команды «Установить состояние тревоги») эти записи восстановить не удастся.

Удалить отдельное хранилище нельзя. Однако при некоторых операциях в модуле могут удаляться все хранилища (см. описания команд «Обновить ключи», «Задать идентификатор мастер-устройства»).

### 5.5.1 Получить список хранилищ данных.

Код команды: SCM_CMD_STORAGE_LIST.
Входные параметры: нет.

Код ответа: SCM_REPLY_STORAGE_LIST.
Возвращаемые данные:
Список типов (SCM_DATA_INTEGER) и имен/идентификаторов (SCM_DATA_STRING) хранилищ.

Возможные ошибки: нет.

Команда поддерживает режим продолжения со стороны модуля (см. описание команды «Продолжение»).

### 5.5.2 Создать хранилище.

Код команды: SCM_CMD_STORAGE_NEW.
Входные параметры:
идентификатор пользователя (SCM_HANDLE_ID),
тип хранилища (SCM_DATA_BYTE).
идентификатор (имя) хранилища (SCM_DATA_STRING, максимальная длина – 32 байта).
флаги (SCM_DATA_INTEGER, битовая маска).
максимальное количество записей (SCM_DATA_INTEGER). Если 0 – то произвольное.
фиксированный размер записи (SCM_DATA_INTEGER). Если 0 – то запись может быть произвольного размера.

Код ответа: SCM_REPLY_STORAGE_NEW.
Возвращаемые данные:
дескриптор открытого хранилища (SCM_DATA_HANDLE).

Возможные ошибки:
хранилище уже существует (SCM_ERROR_ALREADY_EXISTS).

Идентификатор пользователя – либо «0» (см. команду «Задать идентификатор мастер-устройства»), либо идентификатор TLS сессии (см. команду «Начать TLS сессию»).

Атрибуты нового хранилища можно указывать не все, обязательны только тип и идентификатор. Остальные атрибуты будут инициализированы значениями по умолчанию в зависимости от типа хранилища. Атрибуты должны следовать в строгом порядке. Поэтому, если необходимо задать фиксированный размер записи, то требуется указать и все предыдущие параметры.

Идентификатор хранилища должен быть уникальным для хранилищ данного типа. Рекомендуется задавать уникальные имена среди хранилищ всех типов (см. команду «Открыть хранилище»).

Возвращаемый дескриптор имеет права на запись, даже если создается хранилище «только для чтения», что позволяет инициализировать хранилище.Команда может посылать ответ «Ожидание».

### 5.5.3 Информация о хранилище.

Код команды: SCM_CMD_STORAGE_INFO.
Входные параметры:
тип хранилища (SCM_DATA_BYTE),
идентификатор хранилища (SCM_DATA_STRING).

Код ответа: SCM_REPLY_STORAGE_INFO.
Возвращаемые данные:
все атрибуты хранилища в указанном выше (см. начало раздела «Функции хранения данных») порядке.

Возможные ошибки:
идентификатор хранилища не найден (SCM_ERROR_ITEM_NOT_FOUND).

Тип хранилища можно не указывать, если известно, что его идентификатор уникален среди хранилищ всех типов.

Возвращаемое значение атрибута может быть пустым (нулевой длины).

### 5.5.4 Открыть хранилище

Код команды: SCM_CMD_STORAGE_OPEN.
Входные параметры:
идентификатор пользователя (SCM_HANDLE_ID),
тип хранилища (SCM_DATA_BYTE),
идентификатор хранилища (SCM_DATA_STRING),
тип доступа (SCM_DATA_BYTE).

Код ответа: SCM_REPLY_STORAGE_OPEN.
Возвращаемые данные:
дескриптор открытого хранилища (SCM_DATA_HANDLE).

Возможные ошибки:
идентификатор хранилища не найден (SCM_ERROR_ITEM_NOT_FOUND),
отказ в доступе (SCM_ERROR_ACCESS_DENIED).

Идентификатор пользователя – либо «0» (см. команду «Задать идентификатор мастер-устройства»), либо идентификатор TLS сессии (см. команду «Начать TLS сессию»).

Тип хранилища можно не указывать, если известно, что его идентификатор уникален среди хранилищ всех типов.

Тип доступа – битовая маска флагов «Доступ на чтение» (SCM_ACCESS_READ) и «Доступ на запись» (SCM_ACCESS_WRITE). Если тип доступа не указан, подразумевается доступ на чтение.

По окончании работы с хранилищем необходимо закрыть его командой «Закрыть хранилище».

### 5.5.5 Закрыть хранилище.

Код команды: SCM_CMD_STORAGE_CLOSE.
Входные параметры:
дескриптор открытого хранилища (SCM_DATA_HANDLE).

Код ответа: SCM_REPLY_STORAGE_CLOSE.
Возвращаемые данные: нет.

Возможные ошибки:
дескриптор хранилища не найден (SCM_ERROR_ITEM_NOT_FOUND).

### 5.5.6 Получить данные из хранилища.

Код команды: SCM_CMD_STORAGE_READ.
Входные параметры:
дескриптор открытого хранилища (SCM_DATA_HANDLE),
номер записи (SCM_DATA_INTEGER),
тип данных (SCM_DATA_BYTE, см. ниже),
начальная позиция (SCM_DATA_INTEGER, не обязательно, см. ниже).

Код ответа: SCM_REPLY_STORAGE_READ.
Возвращаемые данные:
зависит от запрашиваемого типа данных (см. ниже).

Возможные ошибки:
дескриптор хранилища или номер записи не найден (SCM_ERROR_ITEM_NOT_FOUND),
произошла ошибка при работе с хранилищем (SCM_ERROR_STORAGE).

Номер записи отсчитывается от номера первой записи указанной в атрибутах хранилища (для некоторых типов хранилищ первая запись не обязательно имеет

номер 0). Если номер записи равен «-1», то имеется в виду последняя запись, «-2» - предпоследняя и т.д.

Тип запрашиваемых данных может быть одним из следующих значений:

SCM_STORAGE_DATA_TYPE_ATTRIBUTES - атрибуты,
SCM_STORAGE_DATA_TYPE_CONTENT - данные,
SCM_STORAGE_DATA_TYPE_TICKET - квитанции подтверждения,
SCM_STORAGE_DATA_TYPE_SIGNATURE - подписи.

Если тип запрашиваемых данных - SCM_STORAGE_DATA_TYPE_ATTRIBUTES, то возвращаются следующие параметры:

Номер записи (SCM_DATA_INTEGER).
Имя записи (SCM_DATA_STRING, максимальная длина 32 байта).
Временная метка (SCM_DATA_TIME).
Флаги (SCM_DATA_INTEGER).
Размер данных (SCM_DATA_INTEGER).
Размер подписи (SCM_DATA_INTEGER).
Размер квитанции подтверждения (SCM_DATA_INTEGER).

Если запрашиваются прочие типы данные, то они считываются из хранилища и возвращаются (SCM_DATA_BINARY).

Если задана начальная позиция, то данные записи из хранилища считываются не с начала, а начиная с этой позиции (размер возвращаемых данных соответственно уменьшается на эту величину). Эта возможность поддерживается только для хранилищ без шифрования записей (в этом случае будет возвращена ошибка). Если задан тип данных SCM_STORAGE_DATA_TYPE_ATTRIBUTES, то начальная позиция игнорируется.

Квитанции или подписи могут иметь нулевую длину, если их нет в хранилище или не поддерживаются хранилищем.

Команда поддерживает режим продолжения со стороны модуля (см. описание команды «Продолжение»).

Команда может посылать ответ «Ожидание».

### 5.5.7 Записать данные в хранилище.

Код команды: SCM_CMD_STORAGE_WRITE.
Входные параметры:

дескриптор открытого хранилища (SCM_DATA_HANDLE),
номер записи (SCM_DATA_INTEGER, см. ниже),
имя записи (SCM_DATA_STRING, не обязательно),
записываемые данные (SCM_DATA_BINARY).

Код ответа: SCM_REPLY_STORAGE_WRITE.
Возвращаемые данные:

номер записи (SCM_DATA_INTEGER),
временная метка (SCM_DATA_TIME).

Возможные ошибки:

дескриптор хранилища или номер записи не найден (SCM_ERROR_ITEM_NOT_FOUND),
нет места (SCM_ERROR_OUT_OF_RESOURCE, эта ошибка относится к общим),
отказ в доступе (SCM_ERROR_ACCESS_DENIED, например, для хранилища доступного только на чтение),
произошла ошибка при работе с хранилищем (SCM_ERROR_STORAGE).

Указывать во входных параметрах номер записи имеет смысл только для хранилищ с фиксированным числом записей. Для остальных типов хранилищ номер будет проигнорирован (его можно опустить в параметрах команды), и запись будет добавлена после последней записи. Возвращается всегда реальный номер записи.
Если указано имя записи, оно будет сохранено в атрибутах записи (длина этого поля ограничена, см. выше описание атрибутов записи).

Команда поддерживает режим продолжения со стороны мастера (см. описание команды «Продолжение»).

Данная команда может использоваться, если заранее известен размер записываемых данных. В этом случае в поле длины входного параметра «Записываемые данные» указывается полный размер данных (и их первая часть, помещающаяся в пакет), после чего подряд следуют пакеты с данными с командой «Продолжение».
Если размер данных заранее неизвестен, то можно использовать последовательность команд «Начать запись данных в хранилище», «Продолжить запись данных в хранилище», «Завершить запись данных в хранилище» (см. ниже).

### 5.5.8 Начать запись данных в хранилище.

Код команды: SCM_CMD_STORAGE_WRITE_BEGIN.
Входные параметры:
дескриптор открытого хранилища (SCM_DATA_HANDLE),
номер записи (SCM_DATA_INTEGER, см. ниже),
имя записи (SCM_DATA_STRING, не обязательно).

Код ответа: SCM_REPLY_STORAGE_WRITE_BEGIN.
Возвращаемые данные:
номер записи (SCM_DATA_INTEGER).

Возможные ошибки:
дескриптор хранилища или номер записи не найден (SCM_ERROR_ITEM_NOT_FOUND),
отказ в доступе (SCM_ERROR_ACCESS_DENIED, например, для хранилища доступного только на чтение).

Указывать во входных параметрах номер записи имеет смысл только для хранилищ с фиксированным числом записей. Для остальных типов хранилищ номер

будет проигнорирован (его можно опустить в параметрах команды), и запись будет добавлена после последней записи. Возвращается всегда реальный номер записи.

Если указано имя записи, оно будет сохранено в атрибутах записи (длина этого поля ограничена, см. выше описание атрибутов записи).

Данная команда является началом операции продолженной записи. После этой команды должны следовать команды «Продолжить запись данных в хранилище», «Завершить запись данных в хранилище» (см. ниже). Если во время операции продолженной записи мастер пошлет новую команду «Начать запись данных в хранилище» (для любого хранилища), то будет возвращена ошибка. Если во время операции продолженной записи мастер пошлет команду «Записать данные в хранилище» для того же, хранилища, куда уже идет запись, то будет возвращена ошибка доступа (запись в другие хранилища командой «Записать данные в хранилище» возможна)..

### 5.5.9 Продолжить запись данных в хранилище.

Код команды: SCM_CMD_STORAGE_WRITE_APPEND.
Входные параметры:
записываемые данные (SCM_DATA_BINARY).

Код ответа: SCM_REPLY_STORAGE_WRITE_APPEND.
Возвращаемые данные: нет.

Возможные ошибки:
некорректное состояние, т.е. операция записи не была начата или уже завершена, см. описание команды «Начать запись данных в хранилище» (SCM_ERROR_INVALID_STATE, эта ошибка относится к общим),
нет места (SCM_ERROR_OUT_OF_RESOURCE, эта ошибка относится к общим),
отказ в доступе (SCM_ERROR_ACCESS_DENIED, например, для хранилища доступного только на чтение),
произошла ошибка при работе с хранилищем (SCM_ERROR_STORAGE).

Эта команда записывает данные в хранилище в рамках операции продолженной записи, начатой командой «Начать запись данных в хранилище» (см. выше).

Команда поддерживает режим продолжения со стороны мастера (см. описание команды «Продолжение»).

### 5.5.10 Завершить запись данных в хранилище.

Код команды: SCM_CMD_STORAGE_WRITE_END.
Входные параметры:
флаг корректного завершения (SCM_DATA_BYTE).

Код ответа: SCM_REPLY_STORAGE_WRITE_END.
Возвращаемые данные:

размер записанных данных (SCM_DATA_INTEGER, не обязательно),
временная метка (SCM_DATA_TIME, не обязательно).

Возможные ошибки:
некорректное состояние, т.е. операция записи не была начата или уже завершена, см. описание команды «Начать запись данных в хранилище» (SCM_ERROR_INVALID_STATE, эта ошибка относится к общим),
нет места (SCM_ERROR_OUT_OF_RESOURCE, эта ошибка относится к общим),
отказ в доступе (SCM_ERROR_ACCESS_DENIED, например, для хранилища доступного только на чтение),
произошла ошибка при работе с хранилищем (SCM_ERROR_STORAGE).

Эта команда завершает операцию продолженной записи, начатую командой «Начать запись данных в хранилище» (см. выше).
Если флаг корректного завершения равен 0, то текущая операция продолженной записи данных отменяется, при этом никаких параметров в ответе не возвращается. Если флаг корректного завершения не равен 0, то запись корректно завершается и возвращается размер записанных данных и временная метка.

### 5.5.11 Подтвердить отправку данных.

Код команды: SCM_CMD_STORAGE_CONFIRM.
Входные параметры:
дескриптор открытого хранилища (SCM_DATA_HANDLE),
номер записи (SCM_DATA_INTEGER),
квитанция подтверждения (SCM_DATA_BINARY, не обязательно).

Код ответа: SCM_REPLY_STORAGE_CONFIRM.
Возвращаемые данные:
временная метка квитанции подтверждения (SCM_DATA_TIME, не обязательно).

Возможные ошибки:
дескриптор хранилища или номер записи не найден (SCM_ERROR_ITEM_NOT_FOUND),
нет места (SCM_ERROR_OUT_OF_RESOURCE, эта ошибка относится к общим),
отказ в доступе (SCM_ERROR_ACCESS_DENIED, например, для хранилища доступного только на чтение),
произошла ошибка при работе с хранилищем (SCM_ERROR_STORAGE).

Квитанция может отсутствовать. В этом случае для данной записи будет установлен только флаг подтверждения.
Обратная операция (сброс флага подтверждения или удаление квитанции подтверждения) невозможна.
Команда поддерживает режим продолжения со стороны мастера (см. описание команды «Продолжение»).

### 5.5.12 Найти неподтвержденную запись.

Код команды: SCM_CMD_STORAGE_FIND_UNCONFIRMED.
Входные параметры:
дескриптор открытого хранилища (SCM_DATA_HANDLE),
номер начальной записи (SCM_DATA_INTEGER, не обязательно),
наличие квитанции (SCM_DATA_BYTE, не обязательно).

Код ответа: SCM_REPLY_STORAGE_FIND_UNCONFIRMED.
Возвращаемые данные:
номер записи (SCM_DATA_INTEGER, не обязательно),
временная метка (SCM_DATA_TIME, не обязательно),
размер квитанции (SCM_DATA_INTEGER, не обязательно).

Возможные ошибки:
дескриптор хранилища или номер записи не найден (SCM_ERROR_ITEM_NOT_FOUND),
отказ в доступе (SCM_ERROR_ACCESS_DENIED),
произошла ошибка при работе с хранилищем (SCM_ERROR_STORAGE).

Если номер начальной записи отсутствует, то поиск начинается с первой.
Если флаг наличия квитанции присутствует и не равен «0», то ищутся записи без квитанции подтверждения.
Если неподтвержденной записи (или записи без квитанции подтверждения) не найдено, то возвращается пустой ответ, иначе возвращаются все указанные параметры.
Возвращенный номер записи отсчитывается от начальной позиции указанной в атрибутах хранилища. Т.е. этот номер можно непосредственно передавать в параметрах команд.

## 5.6 Управляющие команды.

### 5.6.1 Продолжение.

Код команды: SCM_CMD_CONTINUE.
Входные параметры: зависит от предыдущей (оригинальной) команды.

Код ответа: зависит от предыдущей команды.
Возвращаемые данные: продолжение ответа на предыдущую команду.

Режим продолжения со стороны мастера.
Если мастер хочет передать команду, параметры которой не помещаются в один пакет, то в пакете команды передается начальная часть данных и устанавливается флаг продолжения. При этом длина параметра выходит за пределы пакета. Следующие части данных (если продолжается значение какого-то параметра, то заголовок параметра не требуется) передаются мастером командой «Продолжение» с установленным флагом продолжения; последняя часть данных (может быть нулевой длины) – командой «Продолжение» без установленного флага продолжения. На

команду «Продолжение» модуль отвечает кодом ответа для оригинальной команды, при этом данные возвращаются только в ответ на последнюю команду «Продолжение» без флага продолжения (предыдущие ответы приходят без данных).

Режим продолжения со стороны модуля.

Если модуль хочет передать ответ, данные которого не помещаются в один пакет, то в пакете ответа передается начальная часть данных и устанавливается флаг продолжения. В этом случае мастер должен выдать модулю команду «Продолжение» без установленного флага продолжения, на которую модуль отвечает следующей порцией данных (с кодом ответа определяемым кодом начальной команды) с установленным в ответе флагом продолжения, и т.д. Последний ответ (с последней порцией данных или с пустыми данными, если они закончились) от модуля приходит без установленного в ответе флага продолжения. Мастер может в любой момент вместо команды «Продолжение» выдать команду «Прервать», по которой модуль завершает обработку начальной команды, возвращая ошибку SCM_ERROR_ABORTED, результат команды - указанная ошибка. Если модуль в ответ на команду «Продолжение» вернул ошибку, мастер должен прервать начальную команду, ее результат – указанная ошибка. Если после команды «Прервать» в течение таймаута SCM_TIMEOUT_HANG не приходит ответа, то модуль считается «зависшим» (см. описание ответа «Ожидание»).

Заголовок очередного параметра команды должен целиком помещаться в один пакет.

Не все команды поддерживают режимы продолжения, только те, входные или возвращаемые параметры которых могут иметь неопределенно большой размер (например, «Записать данные в хранилище» или «Получить данные из хранилища»). В описании команд указывается эта возможность.

### 5.6.2 Прервать.

Код команды: SCM_CMD_ABORT.
Входные параметры: нет.

Код ответа: нет.
Возвращаемые данные: нет.

Всегда возвращается ошибка SCM_ERROR_ABORTED.

Команда может даваться только после получения ответа от модуля с флагом продолжения.

См. описание команды «Продолжение».

### 5.6.3 Ответ «Ожидание».

Код ответа: SCM_REPLY_WAIT.
Возвращаемые данные: нет.

Если обработка модулем какой-либо команды затягивается, то модуль может с какой-то периодичностью (меньше таймаута SCM_TIMEOUT_REPLY) посылать ответ «Ожидание», получив который, мастер должен послать либо команду «Продолжение», либо команду «Прервать» (например, при достижении некоторого общего таймаута для начальной команды). Команды посылаются без параметров. Реакция модуля на команду «Прервать» зависит от выполняемой команды, но в любом случае модуль не будет больше посылать ответ «Ожидание». Если команду можно быстро прервать, то модуль прервет ее и пошлет ответ с ошибкой. Если команда не прерывается, то по истечении «таймаута ожидания» мастер может считать, что модуль «завис» и осуществить его аварийное выключение и повторное включение. При этом все сессии, дескрипторы, полученные от модуля, должны считаться недействительными.

Команды отличные от «Продолжение» или «Прервать» приведут к ошибке «Некорректное состояние».

### 5.6.4 Ответ «Ошибка».

Код ответа: SCM_REPLY_ERROR.
Возвращаемые данные:
код ошибки (SCM_DATA_ERROR),
текстовое сообщение (SCM_DATA_STRING, необязательно).

Текстовое сообщение не обязательно. Оно может использоваться для журнала мастера или для передачи во внешний мир.

## 5.7 Дополнительные команды.

### 5.7.1 Сервисное соединение.

Код команды: SCM_CMD_CONNECTION_INIT.
Входные параметры:
тип соединения (SCM_DATA_BYTE),
транспорт (SCM_DATA_BYTE),
клиент или сервер (SCM_DATA_BYTE),
параметры аутентификации (SCM_DATA_BYTE),
идентификатор криптопровайдера (SCM_DATA_ALG_ID, не обязательно),
идентификатор ключа (SCM_DATA_KEY_ID, не обязательно),
адрес удаленной стороны (SCM_DATA_STRING, не обязательно).

Код ответа: SCM_REPLY_CONNECTION_INIT.
Возвращаемые данные:
идентификатор сервисной сессии (SCM_DATA_HANDLE),
тип соединения (SCM_DATA_BYTE),
транспорт (SCM_DATA_BYTE),
клиент или сервер (SCM_DATA_BYTE),
адрес удаленной стороны (SCM_DATA_STRING),
данные для передачи удаленной стороне (SCM_DATA_BINARY, не обязательно).

Возможные ошибки:
криптопровайдер (алгоритм) или ключ не найдены (SCM_ERROR_ITEM_NOT_FOUND).
Если тип соединения или транспорт не поддерживаются модулем, то может быть возвращена ошибка «Неверное значение параметра» (SCM_ERROR_INVALID_PARAM_VALUE) или «Не поддерживается» (SCM_ERROR_NOT_SUPPORTED).

Входной параметр «клиент или сервер» должен принимать значение SCM_CONNECTION_CLIENT или SCM_CONNECTION_SERVER.

Может возникнуть необходимость соединения самого модуля с каким-либо сервером (например, по команде какого-либо пользователя или из собственных потребностей модуля). Механизм такого соединения аналогичен TLS, но мастер в этом случае выступает только посредником между модулем и сервером, не обрабатывая самостоятельно передаваемые данные.
Данная команда может выдаваться либо по чьему-то требованию, либо самим мастером с какой-то периодичностью (если это разрешено в его настройках).
Рекомендуется давать эту команду до создания канала (соединения), и создавать канал по ответу модуля.

Если не задано ни одного параметра, то это просто запрос к модулю, не требуется ли ему какое-либо соединение. Мастер может периодически давать такой запрос при наличии внешней связи (Интернет и т.п.). В случае команды без параметров модуль вернет ошибку SCM_ERROR_ITEM_NOT_FOUND, если ему не требуется сервисное соединение. См. также описание команды «Обновить ключи».
Тип соединения – определяет цель, задачу соединения. Мастер по своей инициативе не посылает данную команду с параметрами, только по требованию пользователя, который указывает запрашиваемый тип соединения.
Модуль возвращает корректное значение транспорта и адрес удаленной стороны, чтобы мастер мог предпринять действия по установлению канала связи. Например, если модуль возвращает параметр транспорт SCM_CONNECTION_TRANSPORT_TCP_CLIENT, то мастер должен установить TCP соединение с сервером по указанному адресу (формат адреса: АДРЕС:ПОРТ).
Остальные входные параметры аналогичны параметрам команды «Начать TLS сессию».
Если команда выполняется по требованию, то все параметры или их часть задаются требователем. Адрес может не указываться требователем, если в модуле для типа соединения прошит какой-то фиксированный адрес.
Если модуль вернул идентификатор сессии, то он готов к установлению соединения. По окончании работы с ним (в том числе при невозможности установления соединения) идентификатор необходимо закрыть командой «Завершить сервисное соединение». Мастер может проверить, что транспорт и адрес совпадают с указанными в команде (если они были). Мастер устанавливает соединение и передает данные между модулем и удаленной стороной командами «Передать данные сервисного соединения» и «Получить данные сервисного соединения», пока статус соединения (значения см. в описании команды «Установить TLS сессию») в ответе на одну из команд не покажет, что соединение завершено. После чего мастер завершает

соединение командой «Завершить сервисное соединение» и закрывает канал. Во время работы в качестве посредника мастер постоянно слушает канал для получения данных от удаленной стороны, а также периодически посылает модулю команду «Получить данные сервисного соединения». Мастер может иметь таймаут ожидания данных от удаленной стороны, по которому он может дать модулю команду «Состояние сервисного соединения». Если модуль сообщит, что соединение завершено (например, достигнут некоторый таймаут в самом модуле), то мастер завершает соединение. Также мастер может завершить соединение по достижении некоторого таймаута при отсутствии обмена в канале или по иным причинам (например, для экономии электроэнергии).

Рекомендуется регулярно (например, раз в день) запрашивать модуль о необходимости сервисного соединения, так как у модуля может возникнуть потребность, например, в обновлении списков отзыва сертификатов (CRL).

### 5.7.2 Передать данные сервисного соединения.

Код команды: SCM_CMD_CONNECTION_PUT.
Входные параметры:
идентификатор сервисной сессии (SCM_DATA_HANDLE),
данные полученные от удаленной стороны (SCM_DATA_BINARY).

Код ответа: SCM_REPLY_CONNECTION_PUT.
Возвращаемые данные:
статус соединения (SCM_DATA_BYTE),
данные для передачи удаленной стороне (SCM_DATA_BINARY, не обязательно).

Возможные ошибки:
идентификатор сессии не найден (SCM_ERROR_ITEM_NOT_FOUND),
произошла ошибка при обработке данных в сервисной сессии (SCM_ERROR_SERVICE).

Также необходимо контролировать статус соединения (см. описание команды «Сервисное соединение»).

Команда поддерживает режим продолжения со стороны мастера (см. описание команды «Продолжение»).

### 5.7.3 Получить данные сервисного соединения.

Код команды: SCM_CMD_CONNECTION_GET.
Входные параметры:
идентификатор сервисной сессии (SCM_DATA_HANDLE).

Код ответа: SCM_REPLY_CONNECTION_GET.
Возвращаемые данные:
статус соединения (SCM_DATA_BYTE),
данные для передачи удаленной стороне (SCM_DATA_BINARY, не обязательно).

Возможные ошибки:
идентификатор сессии не найден (SCM_ERROR_ITEM_NOT_FOUND),
произошла ошибка при обработке данных в сервисной сессии (SCM_ERROR_SERVICE).

Также необходимо контролировать статус соединения (см. описание команды «Сервисное соединение»).

### 5.7.4 Состояние сервисного соединения.

Код команды: SCM_CMD_CONNECTION_STATUS.
Входные параметры:
идентификатор сервисной сессии (SCM_DATA_HANDLE).

Код ответа: SCM_REPLY_CONNECTION_STATUS.
Возвращаемые данные:
статус соединения (SCM_DATA_BYTE).

Возможные ошибки:
идентификатор сессии не найден (SCM_ERROR_ITEM_NOT_FOUND).

До первой команды обмена данными («Передать данные сервисного соединения» или «Получить данные сервисного соединения») статус соединения может быть неопределен (может зависеть от типа соединения).

### 5.7.5 Завершить сервисное соединение.

Код команды: SCM_CMD_CONNECTION_CLOSE.
Входные параметры:
идентификатор сервисной сессии (SCM_DATA_HANDLE),
причина (SCM_DATA_BYTE),
данные полученные от удаленной стороны (SCM_DATA_BINARY, не обязательно).

Код ответа: SCM_REPLY_CONNECTION_CLOSE.
Возвращаемые данные:
статус соединения (SCM_DATA_BYTE),
данные для передачи удаленной стороне (SCM_DATA_BINARY, не обязательно).

Возможные ошибки:
идентификатор сессии не найден (SCM_ERROR_ITEM_NOT_FOUND).

Причина завершения сессии определяет процедуру закрытия. Возможные значения см. в описании команды «Завершить TLS сессию».

Данные полученные от удаленной стороны могут появиться только при повторных вызовах данной команды.

Данные для передачи удаленной стороне могут отсутствовать (если корректное завершение сессии невозможно, или соединение уже фактически было завершено внутри модуля в команде «Передать данные сервисного соединения»). Если они присутствуют, и канал еще открыт, то эти данные необходимо передать удаленной стороне независимо от значения статуса соединения.

Команда может вызываться несколько раз, пока статус соединения не покажет, что сессия завершена или пока не вернется ошибка SCM_ERROR_ITEM_NOT_FOUND.

Команда должна быть вызвана (по крайней мере один раз), даже если одна из предыдущих команд ранее вернула статус, показывающий, что соединение закрыто.

Команда поддерживает режим продолжения со стороны мастера (см. описание команды «Продолжение»).

### 5.7.6 Получить информацию о криптопровайдере.

Код команды: SCM_CMD_GET_ALG_INFO.
Входные параметры:
идентификатор криптопровайдера (SCM_DATA_ALG_ID),
тип лицензии (SCM_DATA_INTEGER).

Код ответа: SCM_REPLY_GET_ALG_INFO.
Возвращаемые данные:
состояние криптопровайдера (SCM_DATA_BYTE),
дата окончания лицензии (SCM_DATA_TIME, не обязательно).

Возможные ошибки:
ошибка криптопровайдера (SCM_ERROR_CRYPT).

Тип лицензии – одно из следующих значений:
общая лицензия (SCM_ALG_LICENSE_GENERIC).

Состояние криптопровайдера - битовая маска следующих значений:
криптопровайдер отключен (SCM_ALG_STATE_DISABLED),
отсутствуют ключи (SCM_ALG_STATE_NO_KEYS),
критическая тревога (SCM_ALG_STATE_ALARM),
требуется обновление лицензии (SCM_ALG_STATE_LICENSE, необходимо проверить дату окончания лицензии, если она возвращается).
Значение 0 означает нормальную работу.

Дата окончания лицензии может отсутствовать, если обновление лицензии не требуется.

Как правило, данная команда не должна посылаться модулю мастером, только специальным программным обеспечением.

### 5.7.7 Задать лицензию криптопровайдера.

Код команды: SCM_CMD_SET_ALG_INFO.

Входные параметры:
идентификатор криптопровайдера (SCM_DATA_ALG_ID),
тип лицензии (SCM_DATA_INTEGER),
время начала лицензии (SCM_DATA_TIME, не обязательно),
лицензия (SCM_DATA_BINARY).

Код ответа: SCM_REPLY_SET_ALG_INFO.
Возвращаемые данные:
состояние криптопровайдера (SCM_DATA_BYTE),
дата окончания лицензии (SCM_DATA_TIME, не обязательно).

Возможные ошибки:
уже существует (SCM_ERROR_ALREADY_EXISTS, обновление лицензии не требуется),
ошибка криптопровайдера (SCM_ERROR_CRYPT).

Если указанные криптопровайдер и тип лицензии не требуют задавать время начала лицензии, то соответствующий параметр может быть опущен или равен 0.

Состояния криптопровайдера описаны в команде «Получить информацию о криптопровайдере».
Дата окончания лицензии может отсутствовать, если обновление лицензии не требуется.

Данная команда не должна посылаться модулю мастером, только специальным программным обеспечением.

### 5.7.8 Перезагрузка.

Код команды: SCM_CMD_REBOOT.

Входные параметры:
флаг (SCM_DATA_BYTE, не обязательно).

Код ответа: SCM_REPLY_REBOOT (см. ниже).

Возвращаемые данные: нет.

Возможные ошибки: нет.
Флаг может принимать значения:
SCM_REBOOT_TYPE_NORMAL – обычная перезагрузка,
SCM_REBOOT_TYPE_BOOTLOADER – перезагрузка в режиме прошивки.
Если флаг не указан, то подразумевается SCM_REBOOT_TYPE_NORMAL.

Примечание. В более старых версиях ответ на эту команду не приходил, так как модуль сразу перезагружался. В новых версиях модуль сначала возвращает ответ и затем перезагружается.

Команда перезагрузки в режиме прошивки должна посылаться только специальным ПО.

### 5.7.9 Обновить ключи.

Код команды: SCM_CMD_KEY_UPDATE.

Входные параметры:
идентификатор криптопровайдера (SCM_DATA_ALG_ID, не обязательно),
флаги (SCM_DATA_BYTE, не обязательно).

Код ответа: SCM_REPLY_KEY_UPDATE.

Возвращаемые данные: нет.

Возможные ошибки: нет (см. ниже об ограничении количества обновлений ключей).

Флаги - это битовая маска значений:
SCM_KEY_UPDATE_FLAG_UPDATE – обновить ключи модуля.
Если флаги в команде отсутствуют вовсе, то подразумевается значение SCM_KEY_UPDATE_FLAG_UPDATE.

Команда задает криптопровайдеру состояние обновления ключей. Если флаг содержит ненулевое значение , то при последующем запросе к модулю, не требуется ли ему какое-либо соединение (см. описание команды «Сервисное соединение»), модуль затребует соединение с соответствующим сервером для обновления ключей. Если флаг не задан нулевое значение флага, то ключи обновляться не будут (такой вариант команды может использоваться для отмены ранее заданного обновления ключей).
Если криптопровайдер в команде не указан, то состояние обновления ключей задается для всех криптопровайдеров. При этом может потребоваться несколько соединений с разными серверами.
Состояние обновления ключей действует только до перезагрузки/выключения модуля.
Примечание. Модулю могут потребоваться и другие сервисные соединения, например, для обновления списков отзыва сертификатов. Поэтому после команды «Обновить ключи» мастер должен несколько раз запрашивать модуль о сервисном соединении (или до тех пор, пока модуль в ответ на запрос о сервисном соединении не вернет ошибку SCM_ERROR_ITEM_NOT_FOUND).

Модуль может иметь ограничения в количестве обновлений ключей. Количество доступных обновлений может быть проверено в значении параметра «crypto.key_update_remain», который можно получить из модуля командой «Получить

параметр модуля». Если доступных обновлений нет, то в ответ на команду «Обновить ключи» модуль вернет ошибку «Недостаточно ресурсов».

Рекомендуется следующая процедура обновления ключей.

Решение об обновлении ключей принимает оператор.

Оператор имеет коммуникационный канал с мастер-устройством и дает мастеру команды, которые тот транслирует в команды модулю и передает ответы модуля оператору.

Оператор запрашивает текущие сертификаты каждого провайдера в модуле (команда «Получить информацию о ключе»), после чего дает команду «Обновить ключи». Мастер должен по команде оператора (или самостоятельно после передачи модулю команды «Обновить ключи») обеспечить модулю возможность связаться с серверами ключей (команда «Сервисное соединение» без параметров). Когда модулю больше не требуются соединения с серверами, мастер сообщает оператору о завершении операции. Оператор должен проверить, что обновились сертификаты всех криптопровайдеров, для которых проводилось обновление ключей. Если для какого-либо криптопровайдера сертификаты не обновились, то оператор может попробовать повторить процедуру обновления ключей для этого провайдера. Если ключи криптопровайдера не обновятся, то он может оказаться неработоспособным. В этом случае рекомендуется перезагрузить модуль (выключить и включить). При старте модуль попытается восстановить прежние ключи, и мастер сможет повторить процедуру обновления ключей.

Также решение об обновлении ключей может быть принято непосредственно мастер-устройством на основании проверки значений параметров модуля «crypto.key_expiration» или «crypto.key_expiration_days», которые могут быть получены из модуля командой «Получить параметр модуля».

ПРИМЕЧАНИЕ. При обновлении ключей в модуле удаляются все хранилища.

### 5.7.10 Управление журналом.

Код команды: SCM_CMD_LOG_CONTROL.

Входные параметры:

управляющая команда (SCM_DATA_INTEGER)

дополнительные параметры в зависимости от управляющей команды.

Код ответа: SCM_REPLY_LOG_CONTROL.

Возвращаемые данные: в зависимости от управляющей команды.

Возможные ошибки: нет.

Допустимые управляющие команды, дополнительные параметры и возвращаемые данные:

SCM_LOG_CONTROL_TIMEZONE_GET – получить часовой пояс.

Дополнительные параметры: нет.

Возвращаемые данные:

количество секунд – разница с временем UTC (SCM_DATA_INTEGER).

SCM_LOG_CONTROL_TIMEZONE_SET – задать часовой пояс.

Дополнительные параметры:

количество секунд – разница с временем UTC (SCM_DATA_INTEGER)

Возвращаемые данные: нет.

Примечание. Модуль указывает в журнале для каждой записи 2 метки времени (о формате записей см. описание команды «Получить журнал») – по UTC и по заданному часовому поясу. Если мастер не задает часовой пояс, то эти метки совпадают. Часовой пояс сохраняется после перезагрузки модуля.

### 5.7.11 Получить журнал модуля.

Код команды: SCM_CMD_GET_LOG.

Входные параметры:
флаги (SCM_DATA_INTEGER),
дата (SCM_DATA_TIME, не обязательно),
идентификатор криптопровайдера (SCM_DATA_ALG_ID, не обязательно),
идентификатор ключа (SCM_DATA_KEY_ID, не обязательно).

Код ответа: SCM_REPLY_GET_LOG.

Возвращаемые данные:
подпись журнала (SCM_DATA_BINARY, не обязательно),
данные журнала (SCM_DATA_BINARY).

Возможные ошибки: нет.

Флаги – это битовая маска. Они задают параметры ответа модуля. Допустимые значения:

SCM_LOG_FLAG_SIGNED – вернуть также подпись данных журнала.

Если флаг SCM_LOG_FLAG_SIGNED не указан, то возвращаются только данные журнала. Если флаг SCM_LOG_FLAG_SIGNED указан, то перед данными журнала возвращается их отсоединенная подпись в формате CMS, при этом необходимо передать модулю идентификатор криптопровайдера и идентификатор ключа, которым будет подписан журнал. В идентификаторе ключа поле «Назначение» может быть только SCM_KEY_USAGE_TLS_SERVICE. Поскольку ключ с таким назначением не может использоваться мастером для подписи данных, это обеспечивает доверенность подписи журнала. Если при подписи журнала в модуле происходит какая-либо ошибка (например, указанный ключ не существует), то, модуль не возвращает ошибку, но в качестве подписи возвращается пустой буфер.

Модуль возвращает полный журнал за указанную в команде дату. Если журнал за эту дату не найден, то возвращается пустой буфер. Подпись, если она затребована, пустого журнала не производится (возвращается пустая подпись).

Модуль хранит журнал 90 дней.

Каждая запись журнала содержит следующие поля:
время по UTC (пример: 2025.09.14-09:38:09)
время по часовому поясу (см. описание команды «Управление журналом»)
источник (название некоторой подсистемы модуля)
тип сообщения
текст сообщения

Все поля разделяются символом табуляции.

Возможные типы сообщений (к указанным значениям также добавляется символ ":" и пробел):
EMERGENCY
ALERT
CRITITICAL
ERROR
WARNING
NOTICE
INFO
DEBUG

# 6. Коды ошибок.

В описании кодов ошибок указаны ситуации, в которых они могут возникать.
Если в описании ошибки указано, что она является общей, то это означает, что ответ «Ошибка» (SCM_REPLY_ERROR) с таким кодом ошибки может быть получен мастером в ответ на любую команду. Остальные коды ошибок являются более специфическими и явно указаны в описании команд, при обработке которых они могут быть посланы модулем.

| № п.п | Сообщение об ошибке | Расшифровка | Описание |
|---|---|---|---|
| 1. | SCM_ERROR_INVALID_CMD | Неверная команда | Общая ошибка. Неизвестный код команды. См. также описание ошибки «Не поддерживается» (SM_ERROR_NOT_SUPPORTED). Длина пакета превышает максимальное значение. Ошибка CRC. |
| 2. | SCM_ERROR_INVALID_PARAM | Неверные параметры | Общая ошибка. Отсутствует обязательный параметр или присутствует неподдерживаемый параметр. Ошибка выдается при обнаружении первого неверного параметра. |
| 3. | SCM_ERROR_INVALID_PARAM_VALUE | Неверное значение параметра | Общая ошибка. Один из параметров команды имеет некорректное значение. Ошибка |

| | | | |
|---|---|---|---|
| | | | выдается при обнаружении первого неверного значения параметра. |
| 4. | SCM_ERROR_OUT_OF_RESOURCE | Недостаточно ресурсов | Общая ошибка.<br>Недостаток оперативной памяти, места в хранилище и др. |
| 5. | SCM_ERROR_NOT_READY | Не готов | Общая ошибка.<br>Ошибка возвращается в ответ на команду, для выполнения которой требуется некоторая инициализация внутри модуля, которая еще не завершена. Например, на команды явно или неявно (при доступе к зашифрованным данным) использующие криптографические функции. Такая ошибка возвращается при начале работы модуля, когда происходит инициализация всех подсистем. Если инициализация завершится неудачей, то в дальнейшем на такие команды будет возвращаться ошибка «Не поддерживается» (SM_ERROR_NOT_SUPPORTED) |
| 6. | SCM_ERROR_NOT_SUPPORTED | Не поддерживается | Общая ошибка.<br>Некоторая подсистема модуля, необходимая для выполнения команды, не функционирует. Также может означать, что в данной версии модуля эта подсистема отсутствует или отключена в настройках |
| 7. | SCM_ERROR_INVALID_STATE | Некорректное состояние | Общая ошибка.<br>Команда не может быть выполнена в текущем состоянии.<br>Как правило, эта ошибка возвращается, если мастер-устройство посылает новую команду, когда не завершена предыдущая.<br>См. также описание ответа «Ожидание». |
| 8. | SCM_ERROR_CRYPT | Ошибка шифрования | Произошла ошибка в криптографической операции. |
| 9. | SCM_ERROR_STORAGE | Ошибка хранилища | Произошла ошибка при доступе к хранилищу |
| 10. | SCM_ERROR_SERVICE | Ошибка сервисного соединения | Произошла ошибка при обработке данных сервисного соединения |
| 11. | SCM_ERROR_ABORTED | Прервано | Обработка текущей команды прервана.<br>Ошибка возникает в ответ на команду «Прервать». |
| 12. | SCM_ERROR_ACCESS_DENIED | Отказ в доступе | Выполнение операции запрещено правилами доступа |
| 13. | SCM_ERROR_ITEM_NOT_FOUND | Элемент не найден | Элемент, указанный в параметре команды не найден (не существует). |
| 14. | SCM_ERROR_ALREADY_EXISTS | Уже существует | Попытка создать объект, который уже существует |

# Приложение 1.

# Описание параметров криптомодуля.

## 1. Общие замечания.

Данный документ содержит перечень параметров криптомодуля, значения которых запрашиваются командой модуля SCM_CMD_GET_OPTION или устанавливаются командой SCM_CMD_SET_OPTION.

Значения параметров в командах передаются с типом SCM_DATA_BINARY и всегда должны трактоваться либо как бинарное (если ниже при описании параметра явно указано, что его значение представляет собой бинарный массив байтов), либо как строка (во всех остальных случаях). Например, если значение параметра должно быть целым числом, то передается строковое представление этого числа в десятичной системе. Если значение параметра должно быть логическим значением «да» или «нет», то передается строка, соответственно, «1» или «0».

## 2. Параметры модуля.

### 2.1 global.hardware_version

Параметр только для чтения.
Значение: версия аппаратного обеспечения модуля.

### 2.2 global.software_version

Параметр только для чтения.
Значение: версия программного обеспечения модуля.

### 2.3 global.software_time

Параметр только для чтения.
Значение: время сборки программного обеспечения модуля.
Формат: строка вида YYYY-MM-DDThh:mm:ss
Пример: 2022-12-23T17:06:28

### 2.4 global.master_bind

Параметр только для чтения.

Значение: количество оставшихся попыток ввода (командой SCM_CMD_MASTER_ID) идентификатора мастера, к которому привязан модуль. Если это число равно 0, то привязка к конкретному идентификатору отсутствует, если больше 0, то привязка есть, и допускается не более указанного количества подряд попыток ввода неправильного идентификатора. Когда количество оставшихся попыток достигает 0, привязка сбрасывается (при этом в модуле удаляются ключи).

Формат: целое неотрицательное число в строковом представлении.

## 2.5 global.debug

Параметр только для чтения.
Значение: индикатор отладочной версии программного обеспечения модуля.
Формат: строка вида «0» или «1».

## 2.6 system.uptime

Параметр только для чтения.
Значение: время в секундах с момента включения интерфейсов модуля (т.е., когда мастер мог начать работу с модулем).
Формат: строковое представление целого числа.
Примечание. Поддерживается дополнительный необязательный подпараметр “power” (.т.е. полное имя параметра “system.uptime.power”), для которого возвращается количество секунд со времени включения модуля.

## 2.7 system.recovery_supported

Параметр только для чтения.
Значение: поддерживает ли модуль режим восстановления (перепрошивки).
Формат: строка вида «0» или «1».

## 2.8 system.cpu_high_performance

Значение: высокопроизводительный режим работы модуля.
Формат: строка вида «1» (означает, что используется высокопроизводительный режим работы), «0» (модуль динамически регулирует производительность в зависимости от нагрузки), «-1» (модуль работает в режиме минимальной производительности).
Примечание. Модуль после загрузки всегда начинает работать в режиме динамического регулирования производительности (соответствует значению «0» данного параметра). Таким образом, задание другого значения действует только до перезагрузки модуля.

## 2.9 system.battery_rtc

Параметр только для чтения.
Значение: текущее напряжение (в вольтах) батареи, питающей RTC. Если модуль не поддерживает эту функцию, то при попытке считывания этого параметра будет возвращена ошибка.
Формат: строковое представление действительного числа с двумя знаками после запятой.
Примечание. Поддерживаются дополнительные необязательные подпараметры: “real”, “percent”, “raw” (.т.е. полное имя параметра будет, соответственно, “system.battery_rtc.real”, “system.battery_rtc.percent”, “system.battery_rtc.raw”), для которых возвращаются значения: напряжение, процент от

номинального напряжения или неприведенное показание датчика, соответственно. Если подпараметр не указан, подразумевается “real”.

## 2.10 system.protect_line

Параметр только для чтения.

Значение: флаг, показывающий состояние линии защиты в интерфейсном разъеме.

Формат: строка «2» (означает, что линия защиты замкнута и используется), «1» (линия защиты замкнута и не используется), «0» (состояние линии защиты не определено, контур защиты не используется), «-1» (линия защиты не замкнута, но не используется), «-2» (линия защиты не замкнута, но используется, т.е. произошло нарушение защиты).

Примечание. Использование линии защиты означает, что был включен внешний контур защиты (см. описание команды «Контур защиты»).

## 2.11 usb.last_packet_fix

Значение: отправлять неполные пакеты для завершения USB транзакции.

Формат: строковое представление числа, являющегося битовой маской следующих флагов:

SCM_USB_FLAG_FIX_LAST_BYTE (значение 1)

SCM_USB_FLAG_SEND_ZLP (значение 2).

Флаг SCM_USB_FLAG_FIX_LAST_BYTE означает, что модуль будет работать в безопасном режиме (см. примечание ниже), флаг SCM_USB_FLAG_SEND_ZLP означает, что модуль будет отправлять пакеты нулевой длины (ZLP).

Таким образом, этому параметру можно задавать значения «0» (отправлять USB пакеты как есть), «1» (отправлять последний байт в отдельной транзакции), «2» (отправлять ZLP).

Значение «3», являющееся комбинацией обоих флагов, допустимо, но специального смысла не имеет, так как, если модуль отправляет последний байт в отдельной транзакции, то ZLP никогда отправляться не будут.

Примечание. Разные системы и устройства могут по разному определять окончание USB транзакции: по количеству полученных байтов в рамках транзакции, по получению неполного пакета, т.е. пакета, содержащего данных меньше максимального размера USB пакета (для подключения по протоколу USB high-speed это 512 байтов, по USB full-speed – 64 байта). Если размер передаваемого буфера кратен максимальному размеру USB пакета, то размер последнего пакета будет равен этому максимальному размеру. Если принимающая сторона полагается только на получение неполного пакета, то она будет считать, что передача все еще не завершена, и операция чтения никогда не завершится (или завершится с ошибкой по таймауту). Поэтому модуль по умолчанию использует «безопасный» способ передачи: если размер буфера кратен максимальному размеру USB пакета, то сначала передается весь буфер, кроме последнего байта, а затем последний байт передается отдельной транзакцией (мастер должен быть готов к тому, что ответ модуля может прийти не за одну USB транзакцию). Модуль после загрузки всегда начинает работать в этом безопасном режиме. Если принимающая сторона для определения окончания

транзакции использует количество полученных байтов, то можно отключить безопасный режим, задав значение «0» для этого параметра.

См. также раздел «Особенности работы по USB» в документе «Описание функций криптомодуля».

### 2.12 proto.crc_verify

Значение: должен ли модуль проверять контрольные суммы пакетов команд, входящие в состав заголовка пакета (см. описание пакета API модуля в документе «Описание функций криптомодуля»).

Пример: строка вида «0» или «1».

Примечание. Значение будет действовать, начиная со следующей команды посылаемой модулю (т.е. после отключения проверки мастер в следующей команде модулю может не вычислять контрольные суммы). ВНИМАНИЕ: байты с контрольными суммами должны всегда присутствовать в заголовке пакета, независимо от того, проверяются они или не проверяются! Модуль после загрузки всегда проверяет контрольные суммы пакетов команд (что соответствует значению «1» данного параметра). Отключать такую проверку не рекомендуется. Делать это стоит только, если используется надежный протокол нижнего уровня (USB), и критически важна производительность. См. также описание следующего параметра.

### 2.13 proto.crc_calculate

Значение: должен ли модуль вычислять контрольные суммы пакетов своих ответов.

Формат: строка вида «0» или «1».

Примечание. Значение параметра будет действовать, начиная с ответа модуля на команду, задавшую это значение (т.е. после отключения вычисления мастер не должен проверять контрольные суммы уже при разборе ответа модуля на команду, задавшую это значение). ВНИМАНИЕ: байты с контрольными суммами должны всегда присутствовать в заголовке пакета, независимо от того, вычисляются они или не вычисляются! Модуль после загрузки всегда вычисляет контрольные суммы пакетов своих ответов (что соответствует значению «1» данного параметра). Отключать вычисление контрольных сумм не рекомендуется. Делать это стоит только, если используется надежный протокол нижнего уровня (USB), и критически важна производительность. См. также описание предыдущего параметра.

### 2.14 crypto.keymem_status

Параметр только для чтения.

Значение: состояние ключевого хранилища.

Формат: строка вида «0» или «1».

Примечание. Если информация в ключевом хранилище корректная, то вернется значение «1», иначе вернется «0», что означает, что ключи отсутствуют и после перезагрузки ключей в модуле не будет, даже если сейчас они доступны.

### 2.15 crypto.key_server.<PROVIDER_ID>

Параметр только для чтения.

Значение: адрес сервера ключей для указанного криптопровайдера.

Формат: строка вида АДРЕС:ПОРТ

Пример: 192.168.1.55:2345

Примечание. В имени параметра вместо “<PROVIDER_ID>” указывается идентификатор криптопровайдера (он может быть получен, например, командой SCM_CMD_GET_ALG_LIST). Например: crypto.key_server.1

### 2.16 crypto.key_update_last.<PROVIDER_ID>

Параметр только для чтения.

Значение: время последнего обновления ключей для указанного криптопровайдера.

Формат: строка вида YYYY-MM-DDThh:mm:ss

Пример: 2022-12-23T17:06:28

Примечание. В имени параметра вместо “<PROVIDER_ID>” указывается идентификатор криптопровайдера (он может быть получен, например, командой SCM_CMD_GET_ALG_LIST). Например: crypto.key_update_last.1

### 2.17 crypto.key_set_supported.<PROVIDER_ID>

Параметр только для чтения.

Значение: поддерживает ли указанный криптопровайдер явное задание ключа для симметричного шифрования командой SCM_CMD_KEY_SET.

Формат: строка вида «0» или «1».

Примечание. В имени параметра вместо “<PROVIDER_ID>” указывается идентификатор криптопровайдера (он может быть получен, например, командой SCM_CMD_GET_ALG_LIST). Например: crypto.key_set_supported.1

### 2.18 crypto.key_expiration.<PROVIDER_ID>

Параметр только для чтения.

Значение: флаг, показывающий общий статус срока действия ключей и сертификатов (т.е. минимума из срока действия ключей и сертификатов модуля). См. также о параметрах модуля crypto.key_expiration_days.<PROVIDER_ID> и crypto.key_expiration_days_notify.<PROVIDER_ID>.

Формат: строка вида «-1» (означает, что срок действия не определен, например, если ключи модуля не инициализированы), «0» (ключи и сертификаты были инициализированы и до окончания срока их действия остается времени больше, чем задается параметром crypto.key_expiration_days_notify.<PROVIDER_ID>), «1» (до окончания срока действия ключей и сертификатов остается времени меньше, чем задается параметром crypto.key_expiration_days_notify.<PROVIDER_ID>), «2» (срок действия ключей и сертификатов истек).

Примечание. В имени параметра вместо “<PROVIDER_ID>” указывается идентификатор криптопровайдера (он может быть получен, например, командой SCM_CMD_GET_ALG_LIST). Например: crypto.key_expiration.1

### 2.19 crypto.key_expiration_days.<PROVIDER_ID>

Параметр только для чтения.

Значение: количество дней, оставшихся до окончания срока действия ключей и сертификатов (т.е. минимума из срока действия ключей и сертификатов модуля). См. также о параметрах модуля crypto.key_expiration.<PROVIDER_ID> и crypto.key_expiration_days_notify.<PROVIDER_ID>.

Формат: строковое представление целого числа (может быть отрицательным, если срок действия истек) или пустая строка (если ключи модуля не инициализированы).

Примечание. В имени параметра вместо “<PROVIDER_ID>” указывается идентификатор криптопровайдера (он может быть получен, например, командой SCM_CMD_GET_ALG_LIST). Например: crypto.key_expiration_days.1

### 2.20 crypto.key_expiration_days_notify.<PROVIDER_ID>

Параметр только для чтения.

Значение: количество дней, за которое необходимо предупреждать об окончании срока действия ключей и сертификатов (т.е. минимума из срока действия ключей и сертификатов модуля). См. также о параметрах модуля crypto.key_expiration.<PROVIDER_ID> и crypto.key_expiration_days.<PROVIDER_ID>.

Формат: строковое представление целого числа (может быть отрицательным, если значение параметра не используется).

Примечание. В имени параметра вместо “<PROVIDER_ID>” указывается идентификатор криптопровайдера (он может быть получен, например, командой SCM_CMD_GET_ALG_LIST). Например: crypto.key_expiration_days_notify.1

### 2.21 crypto.key_expiration_max.<PROVIDER_ID>

Параметр только для чтения.

Значение: максимальный срок действия ключей модуля (в днях). Определяется политикой использования ключей (см. о параметре crypto.key_profile).

Формат: строковое представление целого числа (может быть нулевым или отрицательным, если ограничение срока действия не задано).

Примечание. В имени параметра вместо “<PROVIDER_ID>” указывается идентификатор криптопровайдера (он может быть получен, например, командой SCM_CMD_GET_ALG_LIST). Например: crypto.key_expiration_max.1

### 2.22 crypto.key_update_remain.<PROVIDER_ID>

Параметр только для чтения.

Значение: количество доступных обновлений ключей модуля.

Формат: строковое представление целого числа (если значение больше 0, то обновление ключей возможно; если значение равно 0, обновление ключей невозможно; если значение меньше 0, то количество обновлений не ограничено).

Примечание. В имени параметра вместо “<PROVIDER_ID>” указывается идентификатор криптопровайдера (он может быть получен, например, командой SCM_CMD_GET_ALG_LIST). Например: crypto.key_update_remain.1

### 2.23 crypto.key_profile.<PROVIDER_ID>

Параметр только для чтения.

Значение: имя политики использования ключей модуля (политика определяет максимальный срок действия ключей, количество доступных обновлений ключей и т.п.).

Формат: строка (возможно, пустая).

Также можно использовать параметр в виде crypto.key_profile.full.<PROVIDER_ID>. В этом случае будет получен полный список опций политики использования ключей в JSON формате.

Примечание. В имени параметра вместо “<PROVIDER_ID>” указывается идентификатор криптопровайдера (он может быть получен, например, командой SCM_CMD_GET_ALG_LIST). Например: crypto.key_profile.1

## 2.24 crypto.hash_info.<INFO_TYPE>.<PROVIDER_ID>

Параметр только для чтения.

Значение: информация об алгоритме хэширования.

Формат: строка (возможно, пустая).

Примечание. В имени параметра вместо “<INFO_TYPE>” указывается одно из значений: name, oid, size (в зависимости от того, какая информация требуется), а вместо “<PROVIDER_ID>” указывается идентификатор криптопровайдера (он может быть получен, например, командой SCM_CMD_GET_ALG_LIST). Например: crypto.hash_info.name.1

## 2.25 crypto.sign_info.<INFO_TYPE>.<PROVIDER_ID>

Параметр только для чтения.

Значение: информация об алгоритме подписи.

Формат: строка (возможно, пустая).

Примечание. В имени параметра вместо “<INFO_TYPE>” указывается одно из значений: name, oid, size (в зависимости от того, какая информация требуется), а вместо “<PROVIDER_ID>” указывается идентификатор криптопровайдера (он может быть получен, например, командой SCM_CMD_GET_ALG_LIST). Например: crypto.sign_info.name.1

## 2.26 crypto.sym_info.<INFO_TYPE>.<PROVIDER_ID>

Параметр только для чтения.

Значение: информация об алгоритме симметричного шифрования.

Формат: строка (возможно, пустая).

Примечание. В имени параметра вместо “<INFO_TYPE>” указывается одно из значений: name, oid, size (в зависимости от того, какая информация требуется), а вместо “<PROVIDER_ID>” указывается идентификатор криптопровайдера (он может быть получен, например, командой SCM_CMD_GET_ALG_LIST). Например: crypto.sym_info.name.1

## 2.27 crypto.tls_max_message

Значение: максимальный размер (в байтах) TLS пакета при отправке.

Формат: строковое представление целого числа.

Примечание.

Если известно, что удаленная сторона имеет ограничение на максимальный размер TLS принимаемого пакета, но она не сообщает об этом при установлении TLS сессии (используя какое-либо расширение протокола TLS), то данный параметр позволяет задать максимальный размер отправляемого модулем TLS пакета. Если значение равно 0, то никаких дополнительных ограничений не вводится.

Это значение никак не влияет на максимальный размер принимаемого TLS пакета (модуль поддерживает установленный стандартами TLS максимальный размер пакета 16 КБ). Это значение используется только в момент создания TLS сессии в модуле командой SCM_CMD_TLS_INIT (если используемый криптопровайдер поддерживает использование такого параметра). После того, как сессия создана, любое изменение данного параметра не влияет на размер пакетов сессии.

Значение данного параметра никак не используется в процессе обмена между сторонами при установке TLS сессии (т.е., никаких расширений TLS протокола со стороны модуля не добавляется).

## 2.28 crypto.tls_ignore_crl.<PROVIDER_ID>

Значение: использовать ли CRL для проверки сертификатов во время установления TLS сессии.

Формат: строка вида «0» или «1».

Примечание. В имени параметра вместо “<PROVIDER_ID>” указывается идентификатор криптопровайдера (он может быть получен, например, командой SCM_CMD_GET_ALG_LIST). Например: crypto.tls_ignore_crl.1

Значение параметра сохраняется после перезагрузки.

Значение параметра «1» эквивалентно тому, что при создании TLS сессии командой «Начать TLS сессию» всегда задается флаг SCM_TLS_FLAG_IGNORE_REVOCATION. Использование этого значения допустимо только в случае, если у мастера нет возможности регулярно обновлять CRL в модуле.

## 2.29 crypto.use_black_list.<PROVIDER_ID>

Значение: использовать ли «черный список» сертификатов.

Формат: строка вида «0» или «1».

Примечание. В имени параметра вместо “<PROVIDER_ID>” указывается идентификатор криптопровайдера (он может быть получен, например, командой SCM_CMD_GET_ALG_LIST). Например: crypto.use_black_list.1

Значение параметра сохраняется после перезагрузки.

См. описание команды «Черный список».

## 2.30 crypto.use_white_list.<PROVIDER_ID>

Значение: использовать ли «белый список» сертификатов.

Формат: строка вида «0» или «1».

Примечание. В имени параметра вместо “<PROVIDER_ID>” указывается идентификатор криптопровайдера (он может быть получен, например, командой SCM_CMD_GET_ALG_LIST). Например: crypto.use_white_list.1

Значение параметра сохраняется после перезагрузки.

См. описание команды «Белый список».

### 2.31 crypto.user_cert_cdp.<PROVIDER_ID>

Значение: использовать ли CDP (CRL Distribution Point) из пользовательских сертификатов.

Формат: строка вида «0» или «1».

Примечание. В имени параметра вместо “<PROVIDER_ID>” указывается идентификатор криптопровайдера (он может быть получен, например, командой SCM_CMD_GET_ALG_LIST). Например: crypto.user_cert_cdp.1

Значение параметра сохраняется после перезагрузки.

Если значение параметра не равно 0, то при проверке пользовательских сертификатов (во время установки TLS сессии, раскодирования CMS и т.п.) из них могут извлекаться CDP для автоматического обновления CRL (см. также описание команды «Сервисное соединение»). CDP будут извлекаться, только если сертификат выдан доверенным УЦ и в флагах проверки включено использование CRL.

### 2.32 storage.total

Параметр только для чтения.

Значение: общий объем накопителя для хранилищ в модуле.

Формат: размер в байтах в строковом представлении.

Пример: 28420468736

### 2.33 storage.used

Параметр только для чтения.

Значение: использованный объем накопителя для хранилищ в модуле.

Формат: размер в байтах в строковом представлении.

Пример: 12177408