РУТОКЕН

Рутокен TLS. Описание интерфейса. Версия 2

Информация о документе

В данном документе описан интерфейс прошивки устройства версии 530 build 0.

Введение

Устройство Рутокен TLS обеспечивает поддержку работы через HTTP-запросы.

Для этого с соответствующего операционной системе раздела внутренней памяти токена должна быть запущена программа-сервер (например, "start.exe" для MS Windows), которая будет обрабатывать HTTP-запросы, формируемые ПО клиента, на IP-адресе "localhost" (127.0.0.1). В запросах используются идентификаторы сессий в URL.

Для выполнения процедуры аутентификации клиентское ПО может использовать начальный идентификатор сессии SID0. Идентификатор сессии находится в файле "sslgate.url" на разделе внутренней памяти токена.

Ниже приводится пример данного файла, где SID0 представляет собой 34 символа латиницы и цифр: "12345678901234567890123456789OABcd"

```
[InternetShortcut]

URL=http://localhost:28016/vpnkeylocal/12345678901234567890123456789OABcd/auth.shtml
```

В этом случае запросы должны формироваться на адрес "http://localhost:28016/vpnkeylocal/" или на "http://localhost:28016/vpnkeylocal/12345678901234567890123456789OABcd/".

Тонкий клиент

В тонком клиенте при запуске программы-сервера осуществляется открытие браузера по умолчанию и переход на страницу аутентификации пользователей, адрес которой может быть взят из секции «URL» файла «sslgate.url». Откроется страница аутентификации пользователя для ввода PIN-кода. После успешной аутентификации устройством генерируется случайный идентификатор авторизованной сессии администратора SID, длина которого составляет 34 символа (латиница + цифры), и происходит переход на главную страницу интерфейса управления устройством, в URL-адресе которой присутствует SID.

С главной страницы возможен переход на страницу нужной бизнес-системы, чьи реквизиты установлены на устройство. После выбора в меню главной страницы нужной бизнес-системы устройство начинает сеанс TLS-связи с веб-сервером этой бизнес-системы и браузер осуществляет переход по переданному токеном URL вида:

http://localhost:28016/?infocrypt_sid=12345678901234567890123456789012345678901234&infocrypt_prefix=aHR0cDovL2xvY2FsaG9zdDoyODAxNi8=,

где присутствует идентификатор сессии SID для работы с API: «12345678901234567890123456789012345678901234». После этого все запросы браузера по адресу http://localhost:28016/*** будут транслироваться токеном по TLS-тоннелю Веб-серверу по адресу "***".

Идентификатор SID должен использоваться в URL для формирования запросов от тонкого клиента к API токена. URL в данном случае имеет вид «http://localhost:28016/vpnkeylocal/<SID>/».

Толстый клиент

Для начала работы толстый клиент может получить SID0 из sslgate.url, с его использованием (или без него) сформировать POST-запрос к API для получения списка доступных учетных записей (см. "GET_PIN_LIST" ниже), и выполнить аутентификацию путем POST-запроса к функции API (см. LOGIN ниже). Ответ на данный запрос будет содержать SID, который следует в дальнейшем использовать для формирования POST-запросов к другим функциям API, в том числе для установления TLS-связи с бизнес-системами.

Функции веб-приложения

1. Генерирование (обработка) ключевых данных.
2. Операции создания (проверки) ЭП по ГОСТ Р 34.10-2001, ГОСТ Р 34.10-2012.

Вызов функций

Вызов функции осуществляется путем формирования POST-запроса, тело которого включает поля:

- id функции,

- имя и значение параметра 1,

…

- имя и значение параметра N

Результат выполнения функции размещается в одном из заранее предусмотренных полей ответа на запрос.

Формат запросов

В соответствии с RFC 2068:

```
Request = Request-Line *( general-header | request-header | entity-header ) CRLF

    [ message-body ]


Request-Line   = Method SP Request-URI SP HTTP-Version CRLF

Method = “POST”
```

Тело запроса должно состоять из пар: "имя поля" и "значение поля".

Пример запроса, у которого значение поля с именем "ID" равно "COMMAND_ID", значение поля с именем "FIELD2" пустое, значение поля с именем "FIELD3" равно "11234":

```
ID=COMMAND_ID&Field1=%D0%12%C0%13QWERTY%11%F2%22&Field2=&Field3=11234
```

Содержимое запросов

1. Главным полем в запросе является "ID". Это команда, показывающая, какую операцию токену следует произвести.
2. Порядок полей в запросе является произвольным, однако рекомендуется ставить поле "ID" на первое место. Также рекомендуется располагать поля с короткими значениями (числами, енумераторами и хендлами) в начале запроса, а поля, длина которых может превысить 15 Кбайт – в конце. Это относится в первую очередь к командам криптоопераций: подписания, проверки подписи, шифрования и расшифровывания. Как только токен получит имена всех обязательных полей, он может начать криптооперацию до окончания прихода всех данных, что может значительно ускорить обработку запроса.
3. Повторное вхождение одного и того же поля в запрос не допускается. Это более строгое чем в RFC правило введено для возможности начать выполнение команды до окончания прихода всех ее аргументов (длина которых может достигать десятков мегабайт), что, в свою очередь, позволяет ускорить работу до двух раз.

Кодировка запросов

1. В целях обеспечения клиенту возможности передавать на сервер токена HTTP-запросы такой длины, обработка которой Java-скриптами за одно HTTP-обращение может оказаться за гранью возможностей Java-скриптов (что бывает при операциях подписания, проверки подписи, шифрования, расшифровывания, а также загрузки обновления), сервер токена поддерживает "Transfer-Encoding: chunked" (для JS это выглядит как: SendChunked = true). Для остальных запросов, которые Java-скрипт может гарантированно обработать одним HTTP-вызовом, не следует злоупотреблять chunked-кодированием, в целях экономии машинного времени.
2. Сервер токена поддерживает "Content-Type: application/x-www-form-urlencoded" (по умолчанию). Для запросов этого типа ПО клиента должно преобразовать с помощью функции URL-encode все поля запроса. Кодировке должны быть подвергнуты только значения полей; символы "=" и "&", разделяющие имя поля и его значение следует оставить без изменения.
3. Запросы с длиной менее 15 Кбайт рекомендуется передавать именно в этом формате.
4. Существует ПО, преобразовывающее с помощью URL-encode не все поля запроса, а только имеющие (по настоящей документации) тип "STRING", и не кодирующее поля, имеющие тип "BASE64". В целях совместимости с этим ПО, такая ситуация автоматически обнаруживается и поддерживается сервером токена. Однако она не соответствует RFC и обрабатывается как исключение, что требует дополнительного машинного времени. Поэтому для коротких запросов, передаваемых в этом формате, рекомендуется кодировать все поля, как того требует RFC, а длинные запросы отправлять в формате "Content-Type: multipart/form-data" (см. ниже).
5. Сервер токена поддерживает "Content-Type: text/plain" и "Content-Type: text/html". Однако для обоих форматов требование наличия URL-кодировки полей внутри запроса сохраняются, т.к. в общем случае поля типов PEMDER или STRING могут содержать символы "=" и "&", из-за которых парсер запроса, требующий формата: "key1=value1&key2=value2", может быть дезориентирован, что приведёт к неверной интерпретации запроса.
6. Сервер токена поддерживает "Content-Type: multipart/form-data". В этом случае все поля запроса должны передаваться клиентом без дополнительной кодировки (chunked-кодирование при этом допускается).

Поля типа **BASE64** с таким типом контента должны передаваться без кодировки **BASE64**. Поля типа **STRING** с таким типом контента должны передаваться без кодировки **URL-encode.**

Этот режим рекомендуется для запросов длиной в несколько мегабайт (криптооперации в режиме 1) для достижения максимальной скорости.

Кодировка ответов

Все ответы имеют "Content-Type: text/html". Кодировка ответа всегда совпадает с кодировкой запроса.

Форматы данных, используемых в HTTP-интерфейсе

Название формата	Направление (слева ПК)	Контроль "start.exe"		Токену передается	Комментарии
		алфавит	другое		
BASE64	<< >>	Для "Content-Type" != "multipart/form-data" - символы Base64 и/или url-safe-Base64	Длина декодированных данных [от и до]	Для "Content-Type" != "multipart /form-data" - результат однократного декодирования из Base64/url-safe-Base64	Поддерживаются Base64 и url-safe-Base64 одновременно. Для "multipart/form-data" - данные клиента передаются без Base64-декодирования.
ENUM(...)	<< >>	только цифры	варианты только из явно указанных в документации.	Число типа int8_t	Ведущие нули теряются
HANDLE_	<< >>	["0"; "9"], ["A"; "Z"], ["a"; "z"]	Длина [от и до]	Строка переданных символов переданной длины	
ID_FORMAT	>>	Заглавные латинские буквы и "_"	Точное соответствие значению из таблицы	Команда CCID	Используется только для идентификаторов команд, не для аргументов.
NUMBER	<< >>	только цифры	Значение [от и до]	Число типа int32: [-(2^31); (2^31)-1]	Ведущие нули теряются
NUMBER64	<< >>	только цифры	Значение [от и до]	Число типа int64: [-(2^63); (2^63)-1]	Ведущие нули теряются
PEM	<<	Base64, "-", " "	Длина [от и до]	*	ASN.1-данные, закодированные в Base64 и обрамлённые PEM-заголовками: "-----BEGIN DATA----- ..... -----END DATA-----"

PEMDER	>>	не контролируется	Длина декодированных двоичных данных [от и до]	Конечный результат декодирования из Base64/url-safe-Base64	Допускаются данные в двоичном формате ASN.1, или в PEM /Base64 /url-safe-Base64, в т.ч. кодированные больше одного раза. Декодирование осуществляется до получения не-PEM-символов, после чего результат отправляется на токен.
PIN_STR	>>	только цифры	Длина [от и до]	Строка переданных символов переданной длины	Ведущие нули сохраняются
STRING	<< >>	не контролируется	Длина [от и до]	Результат декодирования из формата, указанного в заголовке HTTP	
STRING_A	<<	Печатные символы ASCII	Длина [от и до]	*	Используется для передачи незакодированной информации о конфигурации токена, времени и т.д

Коды возврата

В теле ответа на POST-запрос присутствует поле «retcode=”N”», где «N» - это знаковое число в десятичной системе счисления, пригодное для парсинга (atoi).

Если «N» отличается от кода «OK» (1), то остальные поля в теле могут отсутствовать или содержать невалидную информацию, если иного не указано явно в описании команды, в ответ на которую пришел такой retcode.

«N»	Мнемоника ответа	Пояснения
0	FUNCTION_FAILED	Функция не выполнена (как правило синоним: «Ошибка связи с устройством»)
1	OK	Функция успешно выполнена
2	ARGUMENTS_BAD	Указаны неверные аргументы
3	GEC_CMS_SIGNCOUNT	Недостаточное число подписей контейнера корневого сертификата
4	GEC_NOVALIDSIGN	Не удалось проверить подлинность подписи
5	GEC_PARSEERROR	Сертификат не распознан. Возможно, выбранный вами файл поврежден или не содержит сертификат
6	GEC_NTRUSTEDRT	Попытка загрузки корневого сертификата без безопасного контейнера
7	GEC_RATTRCHECK	Ошибка классификации корневого сертификата
8	GEC_CATTRCHECK	Ошибка классификации сертификата УЦ
9	GEC_SATTRCHECK	Ошибка классификации сертификата ЭП
10	GEC_TATTRCHECK	Ошибка классификации сертификата TLS
11	GEC_UNCLASSF	Класс сертификата определить не удалось
12	GEC _ DUPLICATE	Дубликат (сертификата)
13	GEC_FILEERROR	Ошибка ФС при сохранении сертификата
14	GEC _ ERROR	Прочая ошибка при обработке входящего сертификата.
15	PARSE_ERROR	Ошибка парсинга формата CMS
20	UA_INVALID_ARGUMENT	Некорректный аргумент
21	UA_NOT_ENOUGH_STORAGE	В хранилище объектов недостаточно места
22	UA_FILE_WRITE_ERROR	Ошибка записи объекта
24	UA_USER_ALREADY_LOGGED_IN	Пользователь уже выполнил вход
25	UA_USER_BLOCKED	Пользователь заблокирован
26	UA _ RND _ NOT	Ошибка инициализации ДСЧ

27	UA_FAILED_PUK_TRIES	Израсходованы попытки ввода PUK
28	UA_FAILED_PIN_TRIES	Израсходованы попытки ввода PIN
29	UA_INCORRECT_PIN	Введен некорректный PIN
30	PIN_INCORRECT	Неверный PIN-код
31	USER_ALREADY_LOGGED_IN	Пользователь уже выполнил вход
32	USER_NOT_LOGGED_IN	Пользователь не выполнил вход
33	OPERATION_NOT_INITIALIZED	Операция не проинициализирована
34	OPERATION_ACTIVE	Операция активна
35	OBJECT_HANDLE_INVALID	Введен некорректный идентификатор объекта
36	KEY_HANDLE_INVALID	Введен некорректный идентификатор ключа
37	KEY_SIZE_RANGE	Некорректный размер ключа
38	KEY_TYPE_INCONSISTENT	Некорректный тип ключа
39	DATA_INVALID	Введены некорректные данные
40	DATA_LEN_RANGE	Введены данные некорректной длины
41	PIN_EXPIRED	Истек срок действия PIN
42	NOT_IN_INIT_MODE	Устройство не в режиме инициализации
43	NO_RNGINITDATA	Отсутствуют данные инициализации ДСЧ
44	NOT_IN_ACTIVE_MODE	Устройство не в активном режиме
45	UA_EXPIRED	'Истек таймаут сессии
47	INVALID_BS_ID	Введен некорректный идентификатор бизнес-системы
48	PUK_INCORRECT	Введен некорректный PUK
51	TMPL_NOT_APPLICABLE	Шаблон не применяется на данном устройстве
52	TMPL_CMS_ERROR	Ошибка формата CMS при разборе шаблона
53	TMPL_SIGN_ERROR	Ошибка подписи под шаблоном
54	TMPL_FMT_ERROR	Ошибка формата при разборе документа.
55	TMPL_DOCPARSE_ERROR	Ошибка формата при разборе документа
56	TMPL_SYNTAX_ERROR	Синтаксическая ошибка в документе
57	TMPL_MARKUP_ERROR	Ошибка задания разметки в HTML в шаблоне
58	TMPL_NO_TEMPLATE	Невозможно отобразить документ т.к. не был загружен шаблон.
59	TMPL_FIELDS_ERROR	Ошибка формата шаблона
60	TMPL_NOT_MATCH	Версия документа и шаблона не совместимы.

61	TMPL_NOT_ENOUGH_MEMORY	Недостаточно памяти для визуализации всех документов
62	TMPL_XML_ERROR	Ошибка парсинга XML-шаблона
63	TMPL_XML_OVERQUOTE	Превышение квоты документов на подписание.
88	CHECK_USER_KEYS_FAILED	Ошибка при проверке ключей пользователя
89	UNKWN_POST0_ID	Неизвестный идентификатор функции с SID0
90	INVALID_SID	Введен некорректный SID
94	UNKNOWN_EEROR_SID2	Неизвестная ошибка при указании SID2
95	FUNCID_ABSENT_SID2	Неизвестная команда
96	TIMEOUT_SID2	Таймаут сессии
97	OPERATION_NOT_COMPLETE	Операция (подписания; проверки подписи; шифрования) не выполнена
200	CCIDSSL_ERR_PROXYCON	Ошибка соединения с HTTP прокси. Проверьте настройки локального прокси.
210	CCIDSSL_ERR_PROXYAUTH	Ошибка аутентификации на локальном HTTP прокси. Проверьте ваши логин и пароль.
220	CCIDSSL_ERR_PROXYGENERIC	Ошибка работы с HTTP прокси.
700	CORE_FAULT_00	Ошибка ядра #0.
705	MALLOC_ERROR	Ошибка выделения памяти (наследная).
720	FS_WRONG_FILEID	Файл не найден.
722	FS_IO_WRITE_ERROR	Ошибка записи файла на токен.
723	FS_IO_READ_ERROR	Ошибка чтения файла с токена.
724	FS_PERMISSION_ERROR	Недостаточно полномочий для файловой операции на токене.
725	FS_NOT_WR_MODE	Недопустимая файловая операция.
726	FS_ZOP_ERROR	Ошибка защищённой памяти.
727	FS_INIT_ERROR	Ошибка инициализации файловой системы.
728	FS_INTERNAL_ERROR	Внутренняя ошибка файловой системы.
780	CO_HANDLE_INVALID	Неверный хендл криптооперации.
781	CO_NO_FREE_CONTENT	Нет свободных слотов для криптооперации.
782	CO_UI_IS_BUSY	Интерфейс занят другой интерактивной операцией.
783	CO_REJECTED_BY_USER	Операция отменена пользователем.
784	CO_USER_NOT_READY	Ожидание выбора пользователя.
785	CO_USER_WAIT_TIME_OUT	Истёк таймаут согласия пользователя на операцию.

786	CO_NO_RECEPIENT	Не удалось найти указанный сертификат в списке получателей зашифрованного документа.
787	CO_RECEPIENT_NOT_SPECIFIED	Не указан сертификат, в адрес которого выполняется шифрование.
790	UPD_OLD_VERSION	Попытка загрузить более старую версию прошивки.
791	UPD_WRONG_SIGNATURE	Неверный заголовок в файле прошивки.
792	UPD_ERR_BL_SIGN	Подпись под обновлением неверна.
793	UPD_FINISH	Загрузка обновления завершена.
802	UPD_OLD_ERR	Прочая ошибка обновления.
820	UA_USER_DOESN_T_EXIST	Учётная запись с таким номером отсутствует на устройстве.
821	UA_USER_SUSPEND	PIN пользователя заблокирован. (ещё можно восстановить с помощью PUK)
822	UA_CHANGE_PIN_DIVERGENCE	Введенные PIN не совпадают.
823	UA_CHANGE_PIN_INCORRECT	Введен неверный текущий PIN. при смене PIN
824	UA_CHANGE_PUK_DIVERGENCE	Введенные PUK не совпадают.
825	UA_CHANGE_PUK_INCORRECT	Введен неверный текущий PUK. при смене PUK
830	UA_TLS_CERT_WARNING	Ошибка загрузки личного сертификата TLS. Будет использован сертификат первичного подключения. Не фатально, для TLS будет использоваться первичный
831	UA_TLS_CERT_ERROR	TLS-сертификат не загружен. Для TLS будет использоваться первичный
832	UA_ENTER_PUK_PREMATURE	Преждевременная попытка использования PUK-кода.
833	UA_USER_ACTIVE	Команда не предназначена для активных учётных записей.
850	GEC_REVOKED	Сертификат отозван.
851	GEC_CORRUPTED_CRL	Испорчен файл со списком отозванных сертификатов.
852	GEC_EXPIRED	Сертификат просрочен.
855	GEC_OBSOLETE_CRL	На устройстве установлен более новый список отзыва сертификатов.
856	GEC_WRONGUSAGE	Неверное использование сертификата (например, TLS для ЭП итд).
863	CREATEOBJ_INVALIDDN	Структура с данными пользователя не соответствует формату ASN.1
864	CREATEOBJ_INVALIATTRS	Структура с атрибутами пользователя не соответствует формату ASN.1

865	GEC_NO_FREE_PAIR_CELLS	Нет свободных слотов для создания пары 'Секретный ключ - Сертификат'
866	GEC_NO_FREE_USERCRT_CELLS	Превышение допустимого числа личных сертификатов на один запрос.
867	GEC_RQST_NOT_FOUND	Не найден запрос для входящего сертификата.
900	FUNCTION_NOT_IMPLEMENTED	Функция не реализована в этой версии прошивки.
1030	CCIDSSL_CANT_OPEN_SOCKET_ON_TlsS	Не удаётся открыть сокет на TLS-сервере.
1102	CCIDSSL_ACCESSDENY	Бизнес-система не выбрана, или вход на токен не произведён.
1103	CCIDSSL_OUTOFSTREAMS	Нет свободных TLS-потоков.
1104	CCIDSSL_WANTMOREDATA	Недостаточно данных с TLS-сервера.
1105	CCIDSSL_WANTOUTDATA	Токену есть что отправить на TLS-сервер.
1106	CCIDSSL_SSLERROR	Ошибка разбора пакета TLS.
1107	CCIDSSL_SSLCLOSE	TLS-сервер закрыл соединение.
1108	CCIDSSL_PROTOERR	Ошибка протокола TLS.
1109	CCIDSSL_NOCIPHER	Сервер не имеет совместимых наборов шифров.
1110	CCIDSSL_NOCACERT	Не удается построить цепочку доверия до сертификата сервера.
1111	CCIDSSL_NOOWN	Сертификат сервера отсутствует или повреждён.
1112	CCIDSSL_NOCERT	Ошибка получения сертификата сервера.
1113	CCIDSSL_BADPEM	Поврежден контейнер сертификата сервера.
1114	CCIDSSL_BADSIGN	Подпись сертификата сервера не верна.
1115	CCIDSSL_CERTEXPIRED	Срок действия сертификата сервера истёк.
1116	CCIDSSL_CERTREVOKED	Сертификат сервера находится в списке отзыва.
1117	CCIDSSL_NOTTRUSTED	Не удалось проверить подлинность сертификата сервера.
1118	CCIDSSL_UNOCERT	Сервер не получил сертификат клиента.
1119	CCIDSSL_UBADPEM	Сервер получил поврежденный контейнер сертификата клиента.
1120	CCIDSSL_UBADCERT	Сервер получил повреждённый сертификат клиента.
1121	CCIDSSL_UCERTEXPIRED	Сервер получил сертификат клиента с истекшим сроком действия, или истек срок действия CRL издателя.
1122	CCIDSSL_UCERTREVOKED	Сервер обнаружил сертификат клиента в списке отзыва.
1123	CCIDSSL_UNOTTRUSTED	Серверу не удалось проверить подлинность сертификата клиента.

1124	CCIDSSL_UNRECGNAME	Запрашиваемая прикладная система неизвестна серверу.
1125	CCIDSSL_RHSREQ	Запрос рехендшейка от сервера.
1126	CCIDSSL_BROKENKEY	Несовпадение ключей клиента и сервера.
1127	CCIDSSL_NOMOREOWNCERTS	При SwitchCert сертификаты клиента закончились
1128	CCIDSSL_OK_WITHKEYEXPORT	Успешное завершение хендшейка для СОФТ-TLS соединения.
1129	CCIDSSL_BROKEN_SOFTKEYS	Неправильные ключи переданы токеном на ПК при установке СОФТ-TLS соединения
1200	CRYPTO_INVAL	Неверные параметры вычисления хеша/ЭП
1201	CRYPTO_FAIL	Ошибка криптоядра
1300	MAKE_CERT_CHAIN_ERROR	Не удалось построить цепочку сертификатов

Логины

Логин

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	LOGIN	id функции
user	NUMBER	1 байт	номер учетной записи
pin	PIN_STR	6 байт	PIN-код

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID0>/ HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id=LOGIN&user="1"&pin="123456"
```

Выход:

Параметр	Тип	Ограничение	Назначение
sid2	HANDLE	34 байта	идентификатор сессии (опц., наслед., значение равно предыдущему полю "sid")
user	NUMBER	1 байт	номер залогинившейся учетной записи
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

sid2="hagstey2u7dksiu3746xga8kagstev5209"&user="1"&retcode="1"
```

Примечания

- Если при вызове команды «LOGIN» на устройстве уже была открыта сессия этого же или другого пользователя, то его сессия автоматически прекращается.
- При успешном вызове команды "LOGIN" устанавливается использование по умолчанию первой "толстой" бизнес-системы из списка бизнес-систем. То есть, любой запрос по локальному адресу, не содержащему "vpnkeylocal", будет переадресован на установленную к использованию бизнес-систему. Эта особенность не рекомендуется к использованию, т.к. не факт, что первой в списке окажется как раз нужная программе бизнес-система. Правильнее выбирать бизнес-систему после логина явным образом.

Логин 1

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	LOGIN1	id функции
user	NUMBER	1 байт	номер учетной записи
pin	PIN_STR	6 байт	PIN-код

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID0>/ HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id=LOGIN1&user="1"&pin="123456"
```

Выход:

Параметр	Тип	Ограничение	Назначение
sid2	HANDLE	34 байта	идентификатор сессии
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

sid2="hagstey2u7dksiu3746xga8kagstev5209"&retcode="1"
```

Примечания

Функция полностью совместима по входу и выходу с «LOGIN», однако имеет два отличия:

1. Если при вызове команды на устройстве уже была открыта сессия этого же или другого пользователя, то она не прерывается, а в ответ на команду выдается соответствующий код ошибки.
2. При успешном вызове команды бизнес-системы не устанавливается к использованию автоматически. Для работы по TLS следует указать нужную бизнес-систему явным образом (функция «SET_BS_USE»).

Работа с ключевой парой

Генерация ключевой пары и квалифицированного запроса на сертификат PKCS10

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	CREATE_PAIR_EX_ID	id функции	
dn	BASE64	1536 байт бинарных данных (размер base64 для их кодировки не регламентирован)	Владелец сертификата (поле subject из запроса/ будущего сертификата) в виде собранной ASN.1 структуры.	
attr	BASE64	1536 байт бинарных данных (размер base64 для их кодировки не регламентирован)	Атрибуты расширенного запроса на сертификат в виде фрагмента ASN.1 структуры типа Extension.	
attr2	BASE64	1536 байт бинарных данных (размер base64 для их кодировки не регламентирован)	Прочие атрибуты запроса в виде фрагмента ASN.1 структуры типа Attribute.	Опционально
req_type	ENUM	1 байт	1 - личный ЭП; 2 - личный TLS;	Требуется SID
pk_alg	ENUM	1 байт	Формат криптоопераций: 2 - ГОСТ-2001 3 - ГОСТ-2012-256 4 - ГОСТ-2012-512	
hash_alg	ENUM	1 байт	Стандарт подсчёта хеша: 1 - ГОСТ-3411_1994_256 2 - ГОСТ-3411_2012_256 3 - ГОСТ-3411_2012_512	Опционально По умолч.: "1"
enc_alg	ENUM	1 байт	Алгоритмы шифрования: 1 - ГОСТ 28147_1989, CRYPTOPRO_PARAM_A_256, 2 - ГОСТ 28147_1989, PARAM_Z_256	Опционально По умолч.: "1".

paramset	ENUM	1 байт	Наборы параметров эллиптической кривой: 1 - SET_A, 2 - SET_B, 3 - SET_C, 4 - SET_XchA (стандартный для СКЗИ Крипто Про, Верба), 5 - SET_XchB, 6 - SET_A_TK26_256, 7 - SET_A_TK26_512, 8 - SET_B_TK26_512, 9 - SET_C_TK26_512	Опционально По умолч.: "2".
sig_alg	ENUM	1 байт	2(GostR3411_GostR3410)	Игнорируется
ow	ENUM	1 байт	1 - если в хранилище нет места, то автоматически удалить самый старый из прошлых ключей; 2 - не удалять; если в хранилище нет места, то вернуть код ошибки	
charset	ENUM	1 байт	3 (UTF-8)	Игнорируется

Пример:

```
POST http://localhost:28016/vpnkeylocal/< SID>/ HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id=CREATE_PAIR_EX_ID&dn=MIIBkjENMAsGA1UEAwwESFNCQzEVMBMGA1UEBAwM0KLRj9C%2F0Lr
QuNC9MSowKAYDVQQqDCHQlNC20Y3RhCDQktC10L3QtdC00LjQutGC0L7QstC40YcxCzAJBgNVBAYT
AlJVMRUwEwYDVQQHDAzQntC80YHQutCw0Y8xGDAWBgNVBAgMDzU1INCe0LzRgdC60LDRjzErMCkGA
1UECQwi0J3QsNCx0LXRgNC10LbQvdCw0Y8g0YPQu9C40YbQsCwgMTENMAsGA1UECgwESFNCQzEkMC
IGA1UECwwb0J7RgtC00LXQuyDQvtC%2F0LXRgNCw0YbQuNC5MTAwLgYDVQQMDCfQodGC0LDRgNGI0
LjQuSDQvtC%2F0LXRgNCw0YbQuNC%2B0L3QuNGB0YIxGjAYBggqhQMDgQMBARIMMDA3NzEwMzUzNj
A2MRgwFgYFKoUDZAESDTEwMjc3MzkyMDc0NjIxFjAUBgUqhQNkAxILMTEyMjMzNDQ1OTUxHjAcBgk
qhkiG9w0BCQEWD29AbG9jYWxob3N0LmNvbQ%3D%3D&attr=MCYGByqFAwN7AwEEGwwZQTJKUDAwMD
lz0KLRj9C%2F0LrQuNC90JTQkjAOBgNVHQ8BAf8EBAMCBv8wFAYHKoUDA3sDBAQJBgcqhQMDewUJ&
attr2=MA4GByqFAwN7BAkxAwIBAw%3D%3D&pk_alg=2&req_type=1&sig_alg=2&ow=1&charset=3
```

Пояснение

dn – имя владельца сертификата, на который формируется запрос. Сущность Name определяется пунктом 9.1.3 спецификации X.501 (т.е. значение поля Issuer сертификата в виде «как есть»). Переданное аргументом dn имя будет без изменений вставлено в запрос сертификат и далее в сам сертификат на УЦ. Аргумент на токене проверяется исключительно на валидность ASN.1 структуры.

attr – произвольное число сущностей типа Extension как определено RFC5280, последовательно помещенных в передаваемый массив. Аргумент на токене проверяется на валидность ASN.1 структуры.

attr2 – произвольное число сущностей типа Attribute как определено RFC5280, последовательно помещенных в передаваемый массив.

В дополнение к переданным атрибутам токеном будут добавлены дополнительные системные атрибуты.

Производится контроль по определенным атрибутам. Они указаны в таблице:

OID	Назначение OID	Требование в OID
1.2.643.100.111	Наименование используемого средства ЭП	Должен отсутствовать (значение ставится токеном автоматически)
1.2.643.100.113.*	Класс криптосредства	Должен отсутствовать (значение ставится токеном автоматически)
1.2.643.3.123.3.5	Серийный номер токена	Должен отсутствовать (значение ставится токеном автоматически)

Выход:

Параметр	Тип	Ограничение	Назначение
obj_id	HANDLE	8 байт	id объекта
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

obj_id="abcdefgh"&retcode="1"
```

Удаление ключевой пары и соответствующих ей сертификатов или одного личного сертификата

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	DELETE_OBJ_ID	id функции
obj_id	HANDLE	8 байт	id объекта

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/  HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id="DELETE_OBJ_ID"&obj_id="abcdefgh"
```

Выход:

Параметр	Тип	Ограничение	Назначение
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

retcode="1"
```

Пояснения

При выборе хендла сертификата - удаляется только сертификат. При выборе хендла PKCS10-запроса - удаляются ключевая пара, PKCS10-запрос и все связанные сертификаты.

Установка CRL

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	SET_CRL_D_ID	id функции
data	PEMDER	15360 байтов бинарных данных (размер base64, требуемый для их кодировки, не регламентирован)	данные CRL в PEM /DER/Base64

Пример:

```
POST http://localhost:28016/vpnkeylocal/< SID>/ HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id=SET_CRL_D_ID&data="<crl_pem_data>"
```

Выход:

Параметр	Тип	Ограничение	Назначение
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

retcode="1"
```

Примечание

Сертификат, на котором проверяется подпись устанавливаемого CRL, должен удовлетворять определенным требованиям. Они указаны в таблице.

OID	Назначение OID	Требование в OID
2.5.29.15	KeyUsage	Должен присутствовать и иметь установленным cRLSign.
2.5.29.37	ExtendedKeyUsage	Если присутствует, то должен содержать OID "Any extended key usage" (2.5.29.37.0), либо "AnyUsage" (1.3.6.1.5.5.7.3.0). Если является критичным, то должен содержать только значения из списка: serverAuth (1.3.6.1.5.5.7.3.1), clientAuth (1.3.6.1.5.5.7.3.2), codeSigning (1.3.6.1.5.5.7.3.3), emailProtection (1.3.6.1.5.5.7.3.4), timeStamping (1.3.6.1.5.5.7.3.8), OCSPSigning (1.3.6.1.5.5.7.3.9), AnyUsage (1.3.6.1.5.5.7.3.0), AnyExtendedKeyUsage (2.5.29.37.0). Подробнее: https://tools.ietf.org/html/rfc5280

Установить обновление

Начало установки частей обновления

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	INIT_UPD_ID	id функции

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID >/ HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id=INIT_UPD_ID
```

Выход:

Параметр	Тип	Ограничение	Назначение
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

retcode="1"
```

Установка части обновления (поле)

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	SET_UPD_D_ID	id функции	
data	PEMDER	1024000 байтов бинарных данных (размер base64, требуемый для их кодировки, не регламентирован)	данные обновления в base64	
blocknum	NUMBER	2^32 - 1	номер блока	опциональное, если присутствует, то проверяется. Нумерация начинается с 1

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id=SET_UPD_D_ID&data="<upddata>"
```

Выход:

Параметр	Тип	Ограничение	Назначение
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

retcode="1"
```

Завершение установки частей обновления

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	FIN_UPD_ID	id функции

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id=FIN_UPD_ID
```

Выход:

Параметр	Тип	Ограничение	Назначение
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

retcode="1"
```

Работа с сертификатами

Получение списка объектов (сертификатов ЭП, TLS, корневых и УЦ)

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	GET_OBJ_LIST_ID	id функции
obj_type	ENUM	2 байта	0 - сертификаты ЭП; 1 - сертификаты TLS; 2 - корневые сертификаты; 3 - запрос ЭП; 4 - запрос TLS; 5 - сертификаты УЦ; 6 - первичные TLS; 7 - транспортные ЭП;

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id=GET_OBJ_LIST_ID&obj_type=1
```

Выход:

Параметр	Тип	Ограничение	Назначение
data	STRING_A	4096 байт	список id, сепаратор;
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

data="abcdefga;abcdefgh;abcdefgk"&retcode="1"
```

Получение сертификата X.509 (запроса на сертификат PKCS10)

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	GET_OBJ_CERT_D_ID	id функции
obj_id	HANDLE	8 байт	id объекта

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id=GET_OBJ_CERT_D_ID&obj_id="abcdefgh"
```

Выход:

Параметр	Тип	Ограничение	Назначение
data	PEM		данные запроса (сертификата) в формате PEM
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

data="<cert_data>"&retcode="1"
```

Пояснение

При вызове функции GET_OBJ_CERT_D_ID с хендлом PKCS10-запроса, PKCS10-запрос подписывается на ключе транспортного сертификата и возвращается в формате CMS signed data в поле data.

Поиск объекта по полям сертификата

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	GET_CERT_ID	id функции	
serial	BASE64	xx байт	серийный номер в base64	Опционально
c	STRING	xx байт	поле УЦ [Country]	Игнорируется
st	STRING	xx байт	поле УЦ [STate]	Игнорируется
l	STRING	xx байт	поле УЦ [Location]	Игнорируется
org	STRING	xx байт	поле УЦ [ORGanization]	Игнорируется
ou	STRING	xx байт	поле УЦ [OrgUnit]	Игнорируется
cn	STRING	xx байт	поле УЦ [CommonName]	Игнорируется
ea	STRING	xx байт	поле УЦ [E-mAil]	Игнорируется
openkey	BASE64	xx байт	открытый ключ в base64	Опционально
g	STRING	xx байт	поле УЦ []	Игнорируется
inn	STRING	xx байт	поле УЦ [ИНН]	Игнорируется
ogrn	STRING	xx байт	поле УЦ [ОГРН]	Игнорируется
snils	STRING	xx байт	поле УЦ [СНИЛС]	Игнорируется
t	STRING	xx байт	поле УЦ [должность]	Игнорируется
obj_type	NUMBER	0..2^32-1	Типы объектов, среди которых искать (возможны комбинации по схеме «или») 16-корневые сертификаты 32-сертификаты УЦ 64-сертификаты ЭП 128-сертификаты TLS 512-запросы ЭП 1024-запросы TLS	Опц.; при отсутствии или значении меньше шестнадцати ищет среди всех типов по возрастанию

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id="GET_CERT_ID"&serial="MDBDQTEyMzQ1Ng%3D%3D"&st="Moscow"&org=
"SberteX"&ou="OAO"&ea="mail@sbertekx.ru"
```

Выход:

Параметр	Тип	Ограничение	Назначение
data	HANDLE	8 байт	id объекта
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

data="abcdefga"&retcode="1"
```

Примечание 1: Если какое-либо поле в запросе не задано, то оно игнорируется при поиске, и в выборке присутствуют объекты с любым значением этого поля. Если оно задано как "", то в выборке присутствуют только те объекты, у которых оно пустое. Если подходящих объектов несколько, возвращен будет хэндл только одного из них.

Примечание 2: По историческим причинам, не все заявленные в этой таблице поля участвуют в поиске. Из соображений совместимости с работающими в промышленной среде решениями, некоторые поля при поиске игнорируются.

Установка сертификата

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	SET_CERT_D_ID	id функции
data	PEMDER	15360 байтов бинарных данных (размер base64, требуемый для их кодировки, не регламентирован)	данные сертификата в формате PEM/DER /Base64

Пример:

```
POST http://localhost:28016/vpnkeylocal/< SID>/  HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id=SET_CERT_D_ID&data="<certdata>"
```

Выход:

Параметр	Тип	Ограничение	Назначение
obj_id	HANDLE	8 байт	id объекта
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

obj_id="abcdefgh"&retcode="1"
```

Пояснения

- При установке сертификата возвращается его идентификатор.
- При наличии соответствующего объекта (ключевой пары) сертификат привязывается к нему.
- Для успешной установки и возможности использования сертификатов должны быть выполнены определенные требования. Они указаны в таблице.

OID	Назначение OID	Требование в OID
2.5.29.15	KeyUsage	У корневых и УЦ-сертификатов должен присутствовать и иметь установленным keyCertSign и (или) cRLSign, но обязательно хотя бы один из них. У остальных если присутствует, то должен иметь установленным биты: - у ЭП-сертификатов: или digitalSignature, или nonRepudiation, или оба, но обязательно хотя бы один из них; - у TLS-сертификатов: digitalSignature.

2.5.29.37	ExtendedKeyUsage	Если присутствует, то должен содержать OID "Any extended key usage" (2.5.29.37.0), либо "AnyUsage" (1.3.6.1.5.5.7.3.0), либо: - для ЭП-сертификатов: "emailProtection" (1.3.6.1.5.5.7.3.4). - для TLS-сертификатов: "clientAuth" (1.3.6.1.5.5.7.3.2). Если является критичным, то должен содержать только значения из списка: serverAuth (1.3.6.1.5.5.7.3.1), clientAuth (1.3.6.1.5.5.7.3.2), codeSigning (1.3.6.1.5.5.7.3.3), emailProtection (1.3.6.1.5.5.7.3.4), timeStamping (1.3.6.1.5.5.7.3.8), OCSPSigning (1.3.6.1.5.5.7.3.9), AnyUsage (1.3.6.1.5.5.7.3.0), AnyExtendedKeyUsage (2.5.29.37.0). Подробнее: https://tools.ietf.org/html/rfc5280

Установка текущего используемого объекта (не рекомендуется к использованию)

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	SET_ACT_OBJ_ID	id функции
obj_id	HANDLE	8 байт	id объекта

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/  HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id=SET_ACT_OBJ_ID&obj_id="abcdefgh"
```

Выход:

Параметр	Тип	Ограничение	Назначение
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

retcode="1"
```

Примечание

В текущей реализации данная функция не выполняет никаких действий, а только возвращает retcode.

Пояснение

Функция была необходима для прошивок, в функциях которых не задавались напрямую используемые сертификаты, а использовались сертификаты по умолчанию.

Информация о системе

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	GET_SYS_INFO_ID	id функции	
recalc_hash	ENUM	0, 1	пересчитать ли хеш прошивки и записанного на токене "start.exe"	Опц., умолч.:" 0"

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id=GET_SYS_INFO_ID&recalc_hash=0
```

Выход:

Параметр	Тип	Ограничение	Назначение
devcfg	STRING_A	128 байт	информация о конфигурации
rnginitdate	STRING_A	19 байт	HH:MM:SS dd.mm.YYYY (опционально)
serial	STRING_A	12 байт	серийный номер
buildid	NUMBER	8 байт	номер сборки
curuser	NUMBER	1..5	текущая учетная запись (1- 5)
fw_hash_st	STRING_A	64 байта	записанное на токене контрольное значение микропрограммы
fw_hash_calc	STRING_A	64 байта	пересчитанное контрольное значение микропрограммы (возвращается только если при вызове был установлен параметр "recalc_hash")
sw_hash_st	STRING_A	64 байта	записанное на токене контрольное значение записанного на токене "start.exe"
sw_hash_calc	STRING_A	64 байта	пересчитанное контрольное значение записанного на токене "start.exe" (возвращается только если при вызове был установлен параметр "recalc_hash")
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

devcfg="RT0_T128K0000_C1_VT527NT05"&rnginitdate="00:00:00
01.01.2017"&serial="TLS0923502801"&buildid="0"&curuser="1"
&fw_hash_st="2D7D61BDC88423A69686095371DDC2C7A53F3FF3A96
BC0645BD6532204C0017D 0"&sw_hash_st="2D7D61BDC88423A69686
095371DDC2C7A53F3FF3A96BC0645BD6532204C0017D"&retcode="1"
```

Примечание

Строка «devcfg» строится согласно следующему шаблону:

<Идентификатор разработчика (напр."RT" для Рутокен)>_<Сведения об аппаратной части токена, не несущие полезной нагрузки для программы-клиента и гарантированно не содержащие подстроки "_VT" >_VT<Версия прошивки>[<Имя Модуля 1><Версия Модуля 1>][<Имя модуля 2><Версия Модуля 2>]...[<Имя Модуля N><Версия Модуля N>], где N не больше 8.

- Версия прошивки представляет из себя число, состоящее цифр без знака в количестве от 1 до 5. Они заканчиваются концом строки, либо буквой, являющейся первой буквой имени одного из модулей прошивки.
- "Модуль" - это одна или несколько дополнительных программно-аппаратных функций (свойств) данного токена, информирующих прикладную программу о ее новых возможностях или ограничениях при работе с токеном.
- Имя модуля состоит строго из двух латинских букв (с учётом регистра) и от одной до пяти цифр версии (возможно, в этом количестве присутствуют ведущие нули). Цифры заканчиваются концом строки, либо буквой, являющейся первой буквой имени очередного модуля прошивки.
- Строка конфигурации может содержать информацию о модулях, количеством до восьми.
- Имена модулей не сортированы.
- Порядок следования информации о модулях не имеет значения.
- От версии к версии прошивки не гарантируется сохранение порядка следования информации о модулях.

На момент написания настоящей документации, в различных прошивках использовались следующие модули в различных комбинациях:

- BU - кнопка
- NT - новые технологии

Получение URL для переходя из бизнес-системы

С помощью этой команды можно получить URL-адрес страницы администрирования токена, на который можно вернуться по окончании работы с бизнес-системой.

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	GET_SID_URL_ID	id функции	
urlnum	NUMBER	1 байт	номер SID в URL	При вводе «0» функция возвращает URL-адрес страницы аутентификации пользователей, содержащий или нет SID0 (например, « http://localhost:28016/vpnkeylocal/»). При вводе «1» функция возвращает URL-адрес главной страницы администрирования токена, содержащий SID. При переходе при логауте из бизнес-системы в первом случае выполняется разлогинивание пользователя, во втором - нет

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id="GET_SID_URL_ID"&urlnum="1"
```

Выход:

Параметр	Тип	Ограничение	Назначение
data	STRING	128 байт	URL перехода
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

data="http://localhost:28016/vpnkeylocal/<SID>/main.html"&retcode="1"
```

Расширение API для работы с толстым клиентом

Получение списка доступных учетных записей

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	GET_PIN_LIST	id функции

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID0 или SID или без SID>/ HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id="GET_PIN_LIST"
```

Выход:

Параметр	Тип	Ограничение	Назначение
pin	STRING_A	8 байт	номер учетной записи
user	STRING_A	8 байт	имя учетной записи
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

pin="PIN 1"&user="1"&pin="PIN 2"&user="2"&retcode="1"
```

Примечание 1: в случае, если действие PIN-кода приостановлено (т.е. требуется восстановление PIN-кода на странице администрирования или при помощи команды «CH_PIN_BY_PUK_ID»), в начале имени присутствует слово «BLOCKED».

Например:

```
pin="PIN 1"&user="1"&pin="BLOCKED_PIN 2"&user="2"&retcode="1"
```

Примечание 2: если заблокирован PUK, то соответствующий PIN должен выводиться с префиксом «SUSPEND».

Например:

```
pin="PIN 1"&user="1"&pin="SUSPEND_PIN 2"&user="2"&retcode="1"
```

Получение списка доступных бизнес-систем

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	GET_BS_LIST	id функции	
rutoken_visible	ENUM	0,1	0 - недоступные для перехода 1 - доступные для перехода	Опционально

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID >/ HTTP/1.1

Content-Length: xxx\r\n\r\n

id=GET_BS_LIST&rutoken_visible=1
```

Выход:

Параметр	Тип	Ограничение	Назначение
path	STRING		путь
name	STRING_A	128 байт	имя системы
bsid	NUMBER	1 байт	идентификатор системы
retcode	ENUM	2 байт	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

path="/ic/"&name="SBBOL1"&bsid="0"&path="/ic/"&name="SBBOL2"&bsid="1"&retcode="1"
```

Примечание

При вызове данного метода проверяется подпись списка бизнес-систем. Сертификат, на котором проверяется подпись списка бизнес-систем, должен иметь определенные элементы. Они указаны в таблице.

OID	Назначение OID	Требование в OID
2.5.29.15	KeyUsage	Если присутствует, то должен иметь установленным биты: digitalSignature, или nonRepudiation, или оба, но обязательно хотя бы один из них.
2.5.29.37	ExtendedKeyUsage	Если присутствует, то должен содержать OID "Any extended key usage" (2.5.29.37.0), либо "AnyUsage" (1.3.6.1.5.5.7.3.0). Если является критичным, то должен содержать только значения из списка: serverAuth (1.3.6.1.5.5.7.3.1), clientAuth (1.3.6.1.5.5.7.3.2), codeSigning (1.3.6.1.5.5.7.3.3), emailProtection (1.3.6.1.5.5.7.3.4), timeStamping (1.3.6.1.5.5.7.3.8), OCSPSigning (1.3.6.1.5.5.7.3.9), AnyUsage (1.3.6.1.5.5.7.3.0), AnyExtendedKeyUsage (2.5.29.37.0). Подробнее: https://tools.ietf.org/html/rfc5280

Определение (установка) используемой бизнес-системы

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	SET_BS_USE	id функции
bsid	NUMBER	1 байт	идентификатор системы

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Length: xxx\r\n\r\n

id=SET_BS_USE&bsid="0"
```

Выход:

Параметр	Тип	Ограничение	Назначение
sid2	HANDLE	34 байта	идентификатор сессии
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

sid2="<SID>"&retcode="1"
```

Расширение API для работы с резервной бизнес-системой

Открытие сессии с резервной бизнес-системой

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	START_NEW_SYSTEM	id функции
name	STRING_A	128	Fully qualified DNS hostname

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id="START_NEW_SYSTEM"&name="rutoken.ru"
```

Выход:

Параметр	Тип	Ограничение	Назначение
url	STRING	http://localhost:28016/auxch/?infocrypt_sid=<SID>	URL
retcode	ENUM	2 байта	Код возврата функции

Пример:

```
HTTP/1.0 302 FOUND\r\n

>Location: http://localhost:28016/auxch/?infocrypt_sid=<SID>
```

ЭП в CMS

Для получения ЭП в формате CMS необходимо выполнить следующие действия:

1. Проинициализировать устройство (INIT_SIGN_H_ID). Устройство вернет хендл криптооперации - последовательность из 8 букв и цифр. Ее нужно будет использовать при вызове каждой очередной функции данной криптооперации.
2. Внести данные для подписи (SET_SIGN_DATA_H_ID – 1 или более вызовов).
3. Рассчитать подпись (CALC_SIGN_H_ID). В случае некорректного задания данных статус принимает значение FAILED. В случае использования устройства с возможностью интерактивного подтверждения расчета ЭП статус принимает значение WAITING до того момента, пока пользователь не нажмет клавишу подтверждения. В случае отрицательного решения пользователя статус принимает значение REJECTED. При положительном решении (или использовании токена без кнопки) – статус принимает значение COMPLETE.
4. Проверить статус устройства в контексте ЭП (GET_CTX_INFO_H_ID). Данную операцию необходимо выполнять в цикле, если статус принял значение WAITING, до момента, пока он не примет отличное значение.
5. Получить данные CMS (GET_SIGN_CMS_H_ID). Операция должна быть выполнена и вернет CMS с ЭП, если статус принял значение COMPLETE. В случае вызова этой функции до того, как статус принял значение COMPLETE, устройство вернет ошибку "Операция не завершена". В любом случае, после отработки этой функция хендл операции станет невалидным.

К сертификату, который будет использоваться для подписания, предъявляются определенные требования. Они указаны в таблице.

OID	Назначение OID	Требование в OID
2.5.29.15	KeyUsage	Если присутствует, то должен иметь установленным биты digitalSignature и/или nonRepudiation.
2.5.29.37	ExtendedKeyUsage	Если присутствует, то должен содержать OID "Any extended key usage" (2.5.29.37.0), либо "AnyUsage" (1.3.6.1.5.5.7.3.0), либо "emailProtection" (1.3.6.1.5.5.7.3.4) Если является критичным, то должен содержать только значения из списка: serverAuth (1.3.6.1.5.5.7.3.1), clientAuth (1.3.6.1.5.5.7.3.2), codeSigning (1.3.6.1.5.5.7.3.3), emailProtection (1.3.6.1.5.5.7.3.4), timeStamping (1.3.6.1.5.5.7.3.8), OCSPSigning (1.3.6.1.5.5.7.3.9), AnyUsage (1.3.6.1.5.5.7.3.0), AnyExtendedKeyUsage (2.5.29.37.0). Подробнее: https://tools.ietf.org/html/rfc5280

При формировании CMS, в нее будут добавлены атрибуты. Они указаны в таблице.

OID	Назначение OID	Возможные значения
1.3.6.1.4.1.13242.5.2.1	Тип подтверждения	■ 1.3.6.1.4.1.13242.5.2.1.1 – не подтвержден; ■ 1.3.6.1.4.1.13242.5.2.1.2 – подтвержден кнопкой;
1.3.6.1.4.1.13242.5.2.2	Тип криптосредства	1.3.6.1.4.1.13242.5.2.2.1 – токен подписывает хеш документа, подсчитанный снаружи;
1.3.6.1.4.1.13242.5.2.3	Тип токена	■ 1.3.6.1.4.1.13242.5.2.3.1 – обычный токен; ■ 1.3.6.1.4.1.13242.5.2.3.2 – токен с кнопкой;
1.3.6.1.4.1.13242.5.2.4	Серийный номер токена	
1.3.6.1.4.1.13242.5.2.5	Номер PIN-кода	

Инициализация ЭП

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	INIT_SIGN_H_ID	id функции	
datasize	NUMBER64	2^64 - 1	размер данных для ЭП	
hascert	NUMBER	1 байт	максимальное количество сертификатов из цепочки доверия, которое вложить в CMS	"-1" - вложить все сертификаты кроме корневого
hasdata	ENUM	1 байт	0 - не вкладывать в CMS подписанные данные; 1 - вложить в CMS подписанные данные	
mode	ENUM	1, 2	Стандарт криптооперации	Игнорируется
obj_id	HANDLE	8 байт	id сертификата для подписи	Опционально, рекомендуется.
name	BASE64	128 символов UTF8 (размер BASE64 для их кодировки не регламентирован)	Для экранных токенов. Имя подписываемого документа для отображения на экран.	Опционально, игнорируется.

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id="INIT_SIGN_H_ID"&datasize="66144"&hascert="1"&hasdata="0"&obj_id="zxcvbnma"
```

Выход:

Параметр	Тип	Ограничение	Назначение
retcode	ENUM	2 байта	код возврата функции
ctx_handle	HANDLE	8 байт	хендл контекста операции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

retcode="1"&ctx_handle="abcdefgh"
```

Примечание

Для работы по ГОСТ 2011 и выше необходимо указать хендл сертификата на этапе вызова команды "INIT_SIGN_H_ID", т.к. параметры криптооперации берутся из сертификата. Если сертификат не будет указан при вызове INIT_SIGN_H_ID, то его будет необходимо указать в конце (не рекомендуется).

Загрузка порции данных для ЭП

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	SET_SIGN_DATA_H_ID	id функции	
data	BASE64	16777216 байтов бинарных данных (размер base64, требуемый для их кодировки, не регламентирован)	данные для подписи	
blocknum	NUMBER	2^32 - 1	номер посылки	опциональное, если присутствует, то проверяется. Нумерация начинается с 1
ctx_handle	HANDLE	8 байт	хендл контекста операции	

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Length: xxx

id="SET_SIGN_DATA_H_ID"&data="<data>"&blocknum="8"&ctx_handle="abcdefgh"
```

Выход:

Параметр	Тип	Ограничение	Назначение
data_length	NUMBER64	2^64 - 1	суммарное количество данных, переданных для подписи (включая переданное данным вызовом)
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

data_length="1234"&retcode="1"
```

Вычисление ЭП

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	CALC_SIGN_H_ID	id функции	
ctx_handle	HANDLE	8 байт	хендл контекста операции	
obj_id	HANDLE	8 байт	id сертификата для подписи	Опционально

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Length: xxx

id=CALC_SIGN_H_ID&ctx_handle="abcdefgh"
```

Выход:

Параметр	Тип	Ограничение	Назначение
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

retcode="1"
```

Примечание

Поле "obj_id" не рекомендуется использовать с командой "CALC_SIGN_H_ID", т.к. рекомендуется использовать его на этапе вызова команды "INIT_SIGN_H_ID". Подробнее смотрите примечание к "INIT_SIGN_H_ID".

Получение ЭП в формате CMS

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	GET_SIGN_CMS_H_ID	id функции
ctx_handle	HANDLE	8 байт	хендл контекста операции

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Length: xxx

id=GET_SIGN_CMS_H_ID&ctx_handle="abcdefgh"
```

Выход:

Параметр	Тип	Ограничение	Назначение
head	BASE64	16384 байт	данные начала CMS в base64
suffix	BASE64	16384 байт	данные суффикса CMS в base64
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

head="<head_data>"&suffix="<suffix_data>"&retcode="1"
```

Пояснение

Если при инициализации подписания было указано, что выходной файл должен содержать подписанные данные, то клиент должен сам сформировать выходной cms-файл следующим образом: сначала записать возвращённое этой функцией поле “head”, потом сами подписанные данные (чтобы не гонять их через токен туда-обратно), потом возвращённое этой функцией поле “suffix”.

Если при инициализации не собирались включать подписанные данные в cms-файл, то клиент должен просто записать сначала поле “head”, потом поле “suffix”.

Поле “suffix” может быть пустым.

Информация о контексте

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	GET_CTX_INFO_H_ID	id функции
ctx_handle	HANDLE	8 байт	хендл контекста операции

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Length: xxx

id=GET_CTX_INFO_H_ID&ctx_handle="abcdefgh"
```

Выход:

Параметр	Тип	Ограничение	Назначение
status	ENUM	0 - READY, 1 - ACTIVE, 2 - COMPLETE, 3 - WAITING, 4 - FAILED, 5 - REJECTED	состояние контекста устройства
data_length	NUMBER64	2^64 - 1	количество данных, переданных для подписи
sign_num	NUMBER	2^32 - 1	число уже рассчитанных за время данной сессии ЭП
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

status="1"&data_length="1234"&sign_num="7"&retcode="1"
```

Подписание пакетов документов

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	SIGN_PACKET	id функции
obj_id	HANDLE	8 байт	id сертификата для подписи
spec	PEMDER	16000 байтов бинарных данных. (размер base64 для их кодировки не регламентирован)	CMS с XML-правилами обработки подписываемого пакета
vis_required	ENUM	0, 1	1 = Необходимо выдать подписи с типом "визуализировано", исключив автоматическое переключение типа на "подтверждено кнопкой" при нехватке памяти на визуализацию исходных данных.
mode	ENUM	1, 2	Стандарт криптооперации
docs_count	NUMBER	1..2^32 - 1	Количество документов в пакете
doc1... doc4294967295	BASE64	не регламентировано	Документы на подпись

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Length: xxx

id=SIGN_PACKET&obj_id="asdf2534"&spec="77u/PD94bWwgdmVyc2lvbj0iMS4w
IiBlbmNvZGluZz1VVEYtOCI/Pg08VGV..."&docs_count="10"&doc1="77u/QVRUU
klCVV..."&doc2="..."&doc...
```

Выход:

Параметр	Тип	Ограничение	Назначение
count	NUMBER	не регламентировано	Количество CMS-ов с подписями
cms1... cms4294967295	BASE64	не регламентировано	Последовательность полей «cms...=<...>», содержащих полные подписи загруженных для визуализации документов с теми же номерами, с которыми они были загружены.
retcode	ENUM	2 байт	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

count="10"&cms1="<cms1_data>"&cms2="<cms2_data>"&...&retcode="1"
```

Пояснение 0. В качестве параметра «spec» должны быть использованы специальные шаблоны в формате XML, описанные в документе «СББОЛ.VPNkeyTLS. Токен с экраном.Шаблоны.БТ.docx».

Пояснение 1. Блок документов должен иметь следующий формат: «doc<номер документа>=<base64-закодированный документ>&doc<номер следующего документа>=<base64-закодированный следующий документ>итд...». Документы в блоке должны быть отсортированы по порядку и не перемежаться с другими параметрами.

Номера, переданные в именах входных параметров «data...», будут в именах выходных параметров "cms... data". Например, "cms15_data", полученный из функции на выходе, будет подписью для данных, переданных на вход функции в блоке «doc15».

При формировании CMS, в нее будут добавлены дополнительные атрибуты. Они указаны в таблице.

OID	Назначение OID	Возможные значения
1.3.6.1.4.1.13242.5.2.1	Тип подтверждения	■ 1.3.6.1.4.1.13242.5.2.1.1 – не подтвержден; ■ 1.3.6.1.4.1.13242.5.2.1.2 – подтвержден кнопкой;
1.3.6.1.4.1.13242.5.2.2	Тип криптосредства	1.3.6.1.4.1.13242.5.2.2.1 – токен подписывает хеш документа, подсчитанный снаружи;
1.3.6.1.4.1.13242.5.2.3	Тип токена	■ 1.3.6.1.4.1.13242.5.2.3.1 – обычный токен; ■ 1.3.6.1.4.1.13242.5.2.3.2 – токен с кнопкой;
1.3.6.1.4.1.13242.5.2.4	Серийный номер токена	
1.3.6.1.4.1.13242.5.2.5	Номер PIN-кода	
1.3.6.1.4.1.13242.5.2.6	Значение хеш-функции шаблона	Зависит от переданного параметра spec
1.3.6.1.4.1.13242.5.2.8	Количество дайджестов в пакете, в составе которого получен CMS	Зависит от переданного параметра docs_count

Проверка ЭП в CMS

Инициализация проверки ЭП

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	INIT_CHECK_H_ID	id функции	
mode	ENUM	1, 2	Стандарт криптооперации	Опционально, игнорируется
cms_data	PEMDER	15360 байтов бинарных данных (размер base64, требуемый для их кодировки, не регламентирован)	данные CMS ЭП для проверки.	

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Length: xxx

id=INIT_CHECK_H_ID
```

Выход:

Параметр	Тип	Ограничение	Назначение
ctx_handle	HANDLE	8 байт	хендл контекста операции
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

ctx_handle="abcdefgh"&retcode="1"
```

Примечание

Если одновременно имеется и отдельно указанный сертификат, и сертификат внутри CMS, то последний для проверки не используется.

Данные для проверки ЭП

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	SET_CHECK_DATA_H_ID	id функции	
data	BASE64	16777216 байтов бинарных данных (размер base64, требуемый для их кодировки, не регламентирован)	данные для проверки подписи в base64	
blocknum	NUMBER	2^32 - 1	номер посылки	опциональное, если присутствует, то проверяется. Нумерация начинается с 1
ctx_handle	HANDLE	8 байт	хендл контекста операции	

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Length: xxx

id=SET_CHECK_DATA_H_ID&data="<data>"&blocknum="8"&ctx_handle="abcdefgh"
```

Выход:

Параметр	Тип	Ограничение	Назначение
data_length	NUMBER64	2^64 - 1	суммарное количество данных, переданных для проверки подписи (включает то, что передано данным вызовом)
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

data_length="1234"&retcode="1"
```

Проверка ЭП

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	CHECK_SIGN_H_ID	id функции	
ctx_handle	HANDLE	8 байт	хендл контекста операции	
cms_data	BASE64	15360 байтов бинарных данных (размер base64, требуемый для их кодировки, не регламентирован)	данные CMS ЭП для проверки в base64	Опционально

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID1>/ HTTP/1.1

Content-Length: xxx

id=CHECK_SIGN_H_ID&ctx_handle="abcdefgh"
```

Выход:

Параметр	Тип	Ограничение	Назначение
retcode	ENUM	2 байт	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

retcode="1"
```

Пояснение

Аргументы "cert_data" и "cms_data" рекомендуется передавать не команде "CHECK_SIGN_H_ID", а команде "INIT_CHECK_H_ID".

Шифрование данных в CMS

Для шифрования в формате CMS необходимо выполнить следующие действия:

1. Проинициализировать устройство (INIT_ENCIPHER_ID).
2. Внести данные сторонних сертификатов (ADD_RECIPIENT_Y_ID – 0 или более вызовов).
3. Внести данные внутренних сертификатов (ADD_RECIPIENT_X_ID – 0 или более вызовов). Необходимо внести данные как минимум одного стороннего или внутреннего сертификата.
4. Внести данные для шифрования (ENCIPHER_ID – 1 или более вызовов).
5. Получить данные начала и окончания CMS (GET_ENCIPHER_CMS_ID).
6. Сформировать CMS с использованием полученных начала и окончания, а также зашифрованных данных в соответствии со схемой:

Шифрованные данные в формате CMS = PEM_head + base64(head + Recipient_info + suffix + зашифрованные данные) + PEM_tail

Для выполнения операции шифрования в адрес какого-либо сертификата необходимо, чтобы сертификат удовлетворял требованиям. Они указаны в таблице.

OID	Назначение OID	Требование в OID
2.5.29.15	KeyUsage	Если присутствует, то должен иметь установленным биты keyEncipherment и (или) keyAgreement
2.5.29.37	ExtendedKeyUsage	Если присутствует, то должен содержать ОИД "Any extended key usage" (2.5.29.37.0), либо "AnyUsage" (1.3.6.1.5.5.7.3.0), либо "emailProtection" (1.3.6.1.5.5.7.3.4) Если является критичным, то должен содержать только значения из списка: serverAuth (1.3.6.1.5.5.7.3.1), clientAuth (1.3.6.1.5.5.7.3.2), codeSigning (1.3.6.1.5.5.7.3.3), emailProtection (1.3.6.1.5.5.7.3.4), timeStamping (1.3.6.1.5.5.7.3.8), OCSPSigning (1.3.6.1.5.5.7.3.9), AnyUsage (1.3.6.1.5.5.7.3.0), AnyExtendedKeyUsage (2.5.29.37.0). Подробнее: https://tools.ietf.org/html/rfc5280

Инициализация шифрования

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	INIT_ENCIPHER_ID	id функции	
datasize	NUMBER64	2^64 - 1	размер данных для шифрования	
mode	ENUM	1, 2	Стандарт криптооперации	Опционально, игнорируется

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Length: xxx

id=INIT_ENCIPHER_ID&datasize="2048"
```

Выход:

Параметр	Тип	Ограничение	Назначение
ctx_handle	HANDLE	8 байт	хендл контекста операции
retcode	ENUM	2 байт	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

ctx_handle="abcdefgh"&retcode="1"
```

Добавление стороннего сертификата получателя

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	ADD_RECIPIENT_Y_ID	id функции
data	BASE64	15360 байтов бинарных данных (размер base64, требуемый для их кодировки, не регламентирован)	данные стороннего сертификата в base64
ctx_handle	HANDLE	8 байт	хендл контекста операции

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Length: xxx

id=ADD_RECIPIENT_Y_ID&data="<cert_data>"&ctx_handle="abcdefgh"
```

Выход:

Параметр	Тип	Ограничение	Назначение
data	BASE64	15360 байтов бинарных данных (размер base64, требуемый для их кодировки, не регламентирован)	часть данных для формирования CMS, содержит информацию о получателе
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

data="<recipient_info>"&retcode="1"
```

Добавление собственного сертификата

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	ADD_RECIPIENT_X_ID	id функции
obj_id	HANDLE	8 байт	id собственного сертификата
ctx_handle	HANDLE	8 байт	хендл контекста операции

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Length: xxx

id=ADD_RECIPIENT_X_ID&obj_id="<cert_obj_id>"&ctx_handle="abcdefgh"
```

Выход:

Параметр	Тип	Ограничение	Назначение
data	BASE64	15360 байтов бинарных данных (размер base64, требуемый для их кодировки, не регламентирован)	часть данных для формирования CMS, содержит информацию о получателе
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

data="<recipient_info>"&retcode="1"
```

Шифрование данных

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	ENCIPHER_ID	id функции
data	BASE64	16777216 байтов бинарных данных (размер base64, требуемый для их кодировки, не регламентирован)	данные для шифрования в base64
ctx_handle	HANDLE	8 байт	хендл контекста операции

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Length: xxx

id=ENCIPHER_ID&data="<data_to_encipher>"&ctx_handle="abcdefgh"
```

Выход:

Параметр	Тип	Ограничение	Назначение
enciphered_blob	BASE64	16777216 байтов бинарных данных (размер base64, требуемый для их кодировки, не регламентирован)	часть зашифрованных данных для CMS
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

enciphered_blob="<data_blob>"&retcode="1"
```

Получение части для CMS

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	GET_ENCIPHER_CMS_ID	id функции
ctx_handle	HANDLE	8 байт	хендл контекста операции

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Length: xxx

id=GET_ENCIPHER_CMS_ID&ctx_handle="abcdefgh"
```

Выход:

Параметр	Тип	Ограничение	Назначение
head	BASE64	2048 байт	данные начала CMS в base64
suffix	BASE64	2048 байт	данные суффикса CMS в base64
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

head="<head_data>"&suffix="<suffix_data>"&retcode="1"
```

Расшифрование данных в CMS

Для расшифрования в формате CMS необходимо выполнить следующие действия:

1. Проинициализировать устройство (INIT_DECIPHER_ID) путем передачи части с заголовком.
2. Внести данные для расшифрования (DECIPHER_ID - 1 или более вызовов) и получить в ответ расшифрованные данные

Инициализация расшифрования

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	INIT_DECIPHER_ID	id функции	
head	BASE64	15360 байт бинарных данных (размер base64, требуемый для их кодировки, не регламентирован)	данные начала CMS в base64	
mode	ENUM	1, 2	Стандарт криптооперации	Опционально, игнорируется

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Length: xxx

id=INIT_DECIPHER_ID&head="<cms_head_containing_blob>"&obj_id="zxcvbnmm"
```

Выход:

Параметр	Тип	Ограничение	Назначение
ctx_handle	HANDLE	8 байт	хендл контекста операции
body_displ	NUMBER	2^32 - 1	размер (смещение) начала CMS в чистом виде (до кодирования base64)
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

ctx_handle="abcdefgh"&body_displ="1456"&retcode="1"
```

Примечание

body_displ в текущей реализации всегда равен 0.

Расшифрование данных

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	DECIPHER_ID	id функции
data	BASE64	16777216 байтов бинарных данных; (размер base64, требуемый для их кодировки, не ограничен)	данные для расшифрования в base64
ctx_handle	HANDLE	8 байт	хендл контекста операции

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id=DECIPHER_ID&data="<data_to_decipher>"&ctx_handle="abcdefgh"
```

Выход:

Параметр	Тип	Ограничение	Назначение
deciphered_blob	BASE64	Соответствует размеру поданного на вход поля "data"	часть расшифрованных данных
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

deciphered_blob="<data_blob>"&retcode="1"
```

Установка конфигурации бизнес-систем

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	SET_MC_D_ID	id функции
data	PEMDER	15360 байтов бинарных данных (размер base64, требуемый для их кодировки, не регламентирован)	данные конфигурации в CMS

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id=SET_MC_D_ID&data="<mc_cms_data>"
```

Выход:

Параметр	Тип	Ограничение	Назначение
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

retcode="1"
```

Смена кодов

Смена PIN по PUK коду

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	CH_PIN_BY_PUK_ID	id функции	
user	NUMBER	1 байт	номер учетной записи	
puk	PIN_STR	12 байт	данные PUK-кода	
pin	PIN_STR	6 байт	данные нового PIN-кода	
pin2	PIN_STR	6 байт	повтор нового PIN-кода	опционально, если подан, то проверяется

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID (опц.)>/ HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id="CH_PIN_BY_PUK_ID"&user="1"&puk="597346123456"&pin="123456"
```

Выход:

Параметр	Тип	Ограничение	Назначение
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

retcode="1"
```

Смена PIN по PIN коду

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	CH_PIN_BY_PIN_ID	id функции	
user	NUMBER	1 байт	номер учетной записи	
pin_old	PIN_STR	6 байт	данные старого PIN-кода	
pin_new	PIN_STR	6 байт	данные нового PIN-кода	
pin_new2	PIN_STR	6 байт	повтор нового PIN-кода	опционально, если подан, то проверяется

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id="CH_PIN_BY_PIN_ID"&user="1"&pin_old="123456"&pin_new="098765"
```

Выход:

Параметр	Тип	Ограничение	Назначение
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

retcode="1"
```

Расширения Рутокен

Установка параметров прокси

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	RUTOKEN_SET_PROXY_SETTINGS_ID	id функции	
enabled	ENUM	0 - не использовать; 1 - использовать	Использование прокси	По умолч. 0
host	STRING		ip хоста	Не может быть пустым при enabled=1
port	NUMBER	Значение в промежутке [0; 65535]	Порт	По умолч. 0
auth	ENUM	0 - аутентификация отсутствует 1 - аутентификация используется	Аутентификация пользователей на прокси	По умолч. 0
username	STRING		Логин для прокси	
password	STRING		Пароль для прокси	

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Length: xxx

id=RUTOKEN_SET_PROXY_SETTINGS_ID&enabled=1&host=127.0.0.1&port=2&auth=0&username=&password=
```

Выход:

Параметр	Тип	Ограничение	Назначение
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

retcode="1"
```

Примечание 1: Для использования команды необходимо предъявить SID. Прокси настраивается для каждого профиля отдельно.

Примечание 2: Команды установки соединения с бизнес-системой (SET_BS_USE, START_NEW_SYSTEM) вернут ошибку CCIDSSL_ERR_PROXYGENERIC в случаях, если:

- на токене поврежден файл, хранящий настройки прокси;
- при обновлении изменился формат хранения данных о настройках прокси на токене.

Соединение с бизнес-системой не будет установлено, если в настройках прокси включено, но прокси-сервер недоступен.

Примечание 3: Общий объем параметров host, username и password не должен превышать 4 Кб.

Получение текущего параметра прокси

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	RUTOKEN_GET_PROXY_SETTINGS_ID	id функции	

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Length: xxx

id=RUTOKEN_GET_PROXY_SETTINGS_ID
```

Выход:

Параметр	Тип	Ограничение	Назначение
enabled	ENUM	0 - не использовать; 1 - использовать	Использование прокси
host	STRING		ip хоста
port	NUMBER	Значение в промежутке [0; 65535]	Порт
auth	ENUM	0 - аутентификация отсутствует 1 - аутентификация используется	Аутентификация пользователей на прокси
username	STRING		Логин для прокси
retcode	ENUM	2 байта	Код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

enabled="0"&host=""&port="0"&auth="0"&username=""&retcode="1"
```

Примечание 1: Для использования команды необходимо предъявить SID.

Примечание 2: Команда вернет ошибку CCIDSSL_ERR_PROXYGENERIC в случаях, если:

- на токене поврежден файл, хранящий настройки прокси;
- при обновлении изменился формат хранения данных о настройках прокси на токене.

Получение дополнительной информации о запросе

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	RUTOKEN_GET_REQ_INFO_ID	id функции	
obj_id	HANDLE	8 байт	id объекта	

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID >/ HTTP/1.1

Content-Length: xxx

id=RUTOKEN_GET_REQ_INFO_ID&obj_id=R66cn182
```

Выход:

Параметр	Тип	Ограничение	Назначение
timestamp	NUMBER64		дата и время создания запроса
biid	STRING	32 байта	идентификатор ключа Бикрипт, опц.
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

timestamp="1564652375"&retcode="1"
```

Примечание: Необходимо предъявить SID.

Установка таймаута

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	RUTOKEN_SET_TIMEOUT	id функции	
seconds	NUMBER	Значение в промежутке [60; 9999]	таймаут в секундах	

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Length: xxx

id=RUTOKEN_SET_TIMEOUT&seconds=540
```

Выход:

Параметр	Тип	Ограничение	Назначение
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

retcode="1"
```

Примечание: Таймаут устанавливается для каждого хранилища персональных данных отдельно.

Получение установленного таймаута

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	RUTOKEN_GET_TIMEOUT	id функции	

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Length: xxx

id=RUTOKEN_GET_TIMEOUT
```

Выход:

Параметр	Тип	Ограничение	Назначение
seconds	NUMBER	значение в промежутке [60; 9999]	таймаут в секундах
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

seconds="2760"&retcode="1"
```

Выбор класс СКЗИ (КС1/КС2)

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	RUTOKEN_SET_DS_TOOL_CLASS	id функции	
ds_tool_class	ENUM	1 - для КС1 (1.2.643.100.113.1); 2 - для КС2 (1.2.643.100.113.2)	класс СКЗИ	по умолч. 1

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Length: xxx

id=RUTOKEN_SET_DS_TOOL_CLASS&ds_tool_class=1
```

Выход:

Параметр	Тип	Ограничение	Назначение
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

retcode="1"
```

Примечание 1: Для использования команды необходимо предъявить SID.

Примечание 2: Текущий класс СКЗИ устанавливается для каждого хранилища персональных данных отдельно.

Получение класса СКЗИ (КС1/КС2)

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	RUTOKEN_GET_DS_TOOL_CLASS	id функции	

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Length: xxx

id=RUTOKEN_GET_DS_TOOL_CLASS
```

Выход:

Параметр	Тип	Ограничение	Назначение
ds_tool_class	ENUM	1 - для КС1 (1.2.643.100.113.1); 2 - для КС2 (1.2.643.100.113.2)	класс СКЗИ
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

ds_tool_class="1"&retcode="1"
```

Примечание: Для использования команды необходимо предъявить SID.

Получение контрольной суммы сертифицированных модулей

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	RUTOKEN_GET_CHECKSUMS_ID	id функции	
source	ENUM	0 - эталонная контрольная сумма 1 - рассчитанная контрольная сумма	Источник контрольной суммы	

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Length: xxx

id=RUTOKEN_GET_CHECKSUMS_ID&source=0
```

Выход:

Параметр	Тип	Ограничение	Назначение
binary_name	STRING_A	255 байт	имя сертифицированного модуля
checksum	STRING_A	128 байт	контрольная сумма модуля в формате hex строки
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

binary_name="module1"&checksum="<checksum1>"&binary_name="module2"&checksum="<checksum2>"&retcode="1"
```

Установка уровня логирования

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	RUTOKEN_CONTROL_SET_LOG_LEVEL	id функции	
level	LOG_LEVEL_ENUM	2 байта	уровень логирования	

Пример:

```
POST http://localhost:28016/vpnkeylocal/  HTTP/1.1

Content-Length: xxx

id=RUTOKEN_CONTROL_SET_LOG_LEVEL&level=1
```

Выход:

Параметр	Тип	Ограничение	Назначение
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

retcode="1"
```

Примечание 1: Тип LOG_LEVEL_ENUM:

LOG_LEVEL_ENUM: ENUM {

1 -- none

2 -- fatal

3 -- error

4 -- warning

5 -- info

6 -- debug

7 -- trace

}

Примечание 2: Применимо только для операционных систем Linux.

Получение уровня логирования

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	RUTOKEN_CONTROL_ GET_LOG_LEVEL	id функции	

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/  HTTP/1.1

Content-Length: xxx

id=RUTOKEN_CONTROL_GET_LOG_LEVEL
```

Выход:

Параметр	Тип	Ограничение	Назначение
level	LOG_LEVEL_ENUM	2 байта	уровень логирования
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

level=1&retcode="1"
```

Примечание 1: Тип LOG_LEVEL_ENUM:

LOG_LEVEL_ENUM: ENUM {

1 -- none

2 -- fatal

3 -- error

4 -- warning

5 -- info

6 -- debug

7 -- trace

}

Примечание 2: Применимо только для операционных систем Linux.

Получение пути к журналам

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	RUTOKEN_CONTROL_GET_CURRENT_LOG_PATH	id функции	
type	ENUM	1 - файл; 2 - директория	Тип пути	

Пример:

```
POST http://localhost:28016/vpnkeylocal/ HTTP/1.1

Content-Length: xxx

id=RUTOKEN_CONTROL_GET_CURRENT_LOG_PATH&type=1
```

Выход:

Параметр	Тип	Ограничение	Назначение
path	STRING		Путь к файловому объекту
retcode	ENUM	2 байта	Код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

path="<path>"&retcode="1"
```

Примечание: Применимо только для операционных систем Linux.

Завершение приложения

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	RUTOKEN_CONTROL_QUIT	id функции	

Пример:

```
POST http://localhost:28016/vpnkeylocal/ HTTP/1.1

Content-Length: xxx

id= RUTOKEN_CONTROL_QUIT
```

Выход:

Параметр	Тип	Ограничение	Назначение
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

retcode="1"
```

Наследные (неподдерживаемые и нерекомендуемые к использованию функции)

Генерация пары ключей и запроса на сертификат PKCS10

Вход:

Параметр	Тип	Ограничение*	Назначение	Дополнительно
id	ID_FORMAT	CREATE_PAIR_ID	id функции	
cn	STRING	128 байт	ФИО	
org	STRING	128 байт	название организации	
orgunit	STRING	64 байта	название подразделения	
email	STRING	32 байта	адрес эл.почты	
country	STRING	2 байта	страна	
rooccu	STRING	60 байт	должность	
bs_id	ENUM	1 байт	0 (Бизнес Онлайн)	
pk_alg	ENUM	1 байт	1 (RSA) /2(ГОСТ 34.10-2001)	
req_type	ENUM	1 байт	1 (ЭП)/2(TLS)	
sig_alg	ENUM	1 байт	1 (MD5_RSA) \ 2 (GostR341194_GostR34102001)	
label	STRING_A	8 байт	прикладное имя ключа	не используется
ow	ENUM	1 байт	1 (удалить неактивный)\ 2 (не удалять)	
biid	STRING	32 байт	идентификатор ключа Бикрипт	
charset	ENUM	1 байт	1 (ASCII)\2(UTF-16) \3(UTF-8)	

Серым цветом обозначены неподдерживаемые на текущий момент параметры.

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/  HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id=CREATE_PAIR_ID&cn=F_I_O&org=ORGname&orgunit=DEPTname&email=em@a.il&country=RU&
rooccu=officer&pk_alg=1&req_type=0&sign_alg=0&label=<прикладное имя ключа>&ow=1&bi
id=<A8DE0012sИвановИИ>&charset=1
```

Выход:

Параметр	Тип	Ограничение	Назначение
obj_id	HANDLE	8 байт	id объекта
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

obj_id=”abcdefgh”&retcode=”1”
```

Информация о контексте

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	GET_CTX_INFO_ID	id функции

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id=GET_CTX_INFO_ID
```

Выход:

Параметр	Тип	Ограничение	Назначение
status	ENUM	1 байт: READY(0), ACTIVE(1), WAITING(2), COMPLETE (3), FAILED(4), REJECTED(5)	состояние контекста устройства
data_length	NUMBER	11 байт	количество данных, переданных для подписи
sign_num	NUMBER	11 байт	число уже рассчитанных за время данной сессии ЭП
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

status=”1”&data_length=”1234”&sign_num=”7”&retcode=”1”
```

Инициализация ЭП

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	INIT_SIGN_ID	id функции

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id=INIT_SIGN_ID
```

Выход:

Параметр	Тип	Ограничение	Назначение
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

retcode=”1”
```

Данные для ЭП

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	SET_SIGN_DATA_ID	id функции	
data	BASE64	15360 байтов бинарных данных (размер base64, требуемый для их кодировки, не регламентирован)	данные для подписи в base64	
blocknum	NUMBER	11 байт	номер посылки	опциональное, если присутствует, то проверяется. Нумерация начинается с 1

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id=SET_SIGN_DATA_ID&data=<data2sign>&blocknum=8
```

Выход:

Параметр	Тип	Ограничение	Назначение
data_length	NUMBER	11 байт	суммарное количество данных после декодирования из base64, переданных для подписи (включая переданные данным вызовом)
accepted_b64_length	NUMBER	11 байт	количество данных, переданных для подписи в данном вызове, в base64 , которые удалось декодировать
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

data_length=”1234”& accepted_b64_length=”1586”&retcode=”1”
```

Вычисление ЭП

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	CALC_SIGN_ID	id функции	
obj_id	HANDLE	8 байт	id сертификата для подписи	
ccdata	BASE64		данные контрольной суммы в base64	опциональное, если присутствует, то проверяется.

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Length: xxx

id=CALC_SIGN_ID&obj_id=abcdefgh&ccdata=<data_cc_b64>
```

Выход:

Параметр	Тип	Ограничение	Назначение
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

retcode=”1”
```

Получение ЭП

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	GET_SIGN_D_ID	id функции

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Length: xxx

id=GET_SIGN_D_ID
```

Выход:

Параметр	Тип	Ограничение	Назначение
data	BASE64		данные подписи в base64
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

data=”<sign_data>”&retcode=”1”
```

Инициализация проверки ЭП

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	INIT_CHECK_ID	id функции

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Length: xxx

id=INIT_CHECK_ID
```

Выход:

Параметр	Тип	Ограничение	Назначение
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK

Content-Length: xxx

retcode=”1”
```

Данные для проверки ЭП

Вход:

Параметр	Тип	Ограничение	Назначение	Дополнительно
id	ID_FORMAT	SET_CHECK_DATA_ID	id функции	
data	BASE64	15360 байтов бинарных данных (размер base64, требуемый для их кодировки, не регламентирован)	данные для проверки подписи в base64	
blocknum	NUMBER	11 байт	номер посылки	опциональное, если присутствует, то проверяется. Нумерация начинается с 1

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id=SET_CHECK_DATA_ID&data=<data2sign>&blocknum=8
```

Выход:

Параметр	Тип	Ограничение	Назначение
data_length	NUMBER64	11 байт	суммарное количество данных, переданных для проверки подписи (включая переданное данным вызовом)
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

data_length=”1234”&retcode=”1”
```

Проверка ЭП

Вход:

Параметр	Тип	Ограничение	Назначение
id	ID_FORMAT	CHECK_SIGN_ID	id функции
cert_data	PEMDER	15360 байтов бинарных данных (размер base64, требуемый для их кодировки, не регламентирован)	данные сертификата в base64
sign_data	BASE64	88 байт	данные ЭП для проверки в base64

Пример:

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1\r\n

Content-Length: xxx\r\n\r\n

id=CHECK_SIGN_ID&cert_data=<certificate data>&sign_data=<signature data>
```

Выход:

Параметр	Тип	Ограничение	Назначение
data	ENUM ('0'/'1' - атавизм)	1 байт	данные проверки
retcode	ENUM	2 байта	код возврата функции

Пример:

```
HTTP/1.0 200 OK\r\n

Content-Length: xxx\r\n\r\n

id_data=”<id_data>”&retcode=”1”
```

Сценарии работы с токенами Рутокен TLS

Установление сессии на токена

а) Определить SID0 (входит в состав URL в файле sslgate.url на флеш-диске токена):

```
[InternetShortcut]

URL=http://localhost:28016/vpnkeylocal/<SID0>/
```

б) Вызвать метод API для получения списка доступных PIN-кодов:

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID0 или без SID>/ HTTP/1.1

Content-Type: text/plain

Content-Length: 15

id=GET_PIN_LIST
```

{пример запроса; окончание}

{пример ответа; начало}

```
pin="PIN 1''&user=''1"&pin="PIN 2''&user=''2"&retcode="1"
```

{пример ответа; окончание}

в) Вызвать метод API для открытия сессии, указав необходимое имя и PIN-код:

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID0 или без SID>/ HTTP/1.1

Content-Type: text/plain

Content-Length: 27

id=LOGIN1&user=1&pin=597346
```

{пример запроса; окончание}

{пример ответа; начало}

```
sid2="<SID>"&retcode="1"
```

{пример ответа; окончание}

г) Полученный SID необходимо использовать для следующих сценариев.

Установление сессии на токене и канала TLS с поддержкой мультисистемности

а) Логин на токен произведен, SID известен. Получить список доступных бизнес-систем:

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1
Content-Type: text/plain

Content-Length: 32

id=GET_BS_LIST&rutoken_visible=1
```

{пример запроса; окончание}

{пример ответа; начало}

```
path="/ic/"&name="SBBOL1"&bsid="0"&path="/ic/"&name="SBBOL2"&bsid="1"&path="/ic/"&name="SBBOL3"&bsid="2"&retc
```

{пример ответа; окончание}

б) Из полученного списка бизнес-систем необходимо выбрать для дальнейшей работы и указать ее идентификатор:

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Type: text/plain

Content-Length: 20

id=SET_BS_USE&bsid=0
```

{пример запроса; окончание}

{пример ответа; начало}

```
sid2="<SID>"&retcode="1"
```

{пример ответа; окончание}

Формирование ключевой пары

а) Выполнить действия по сценарию «Установление сессии на токене», получить в результате SID.

б) Вызвать метод API для формирования ключевой пары и выдачи ее идентификатора (хендла):

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1
Content-Type: text/plain
Content-Length: 188

id=CREATE_PAIR_EX_ID&dn=MB8xEDAOBgNVBAMMB3Rlc3R
yZXExCzAJBgNVBAYTAlJV&attr=MA4GA1UdDwEB%2FwQEAwI
E8A%3D%3D&attr2=&req_type=1&pk_alg=3&hash_alg=2&
enc_alg=1&paramset=2&sig_alg=2&ow=2&charset=3
```

{пример запроса; окончание}

{пример ответа; начало}

```
obj_id="zuyDOqP0"&retcode="1"
```

{пример ответа; окончание}

в) Вызвать метод API для получения данных запроса на сертификат:

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1
Content-Type: text/plain
Content-Length: 36
id=GET_OBJ_CERT_D_ID&obj_id=zuyDOqP0
```

{пример запроса; окончание}

В ответ выдается текст запроса в PEM:

{пример ответа; начало}

```
data="<new_cert_request_data>"&retcode="1"
```

{пример ответа; окончание}

г) Полученный запрос можно использовать для генерации сертификата.

Установка сертификата к уже сформированному ключу

а) Выполнить действия по сценарию «Установление сессии на токене», получить в результате SID.

б) Вызвать метод API для установки сертификата на токен:

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Type: text/plain

Content-Length: 1562

id=SET_CERT_D_ID&data=<new_cert_data_PEM>
```

{пример запроса; окончание}

В ответ токен возвращает идентификатор установленного сертификата:

{пример ответа; начало}

```
obj_id="qEpE0NTZ"&retcode="1"
```

{пример ответа; окончание}

Формирование ЭП в CMS

а) Выполнить действия по сценарию «Установление сессии на токене», получить в результате SID

б) На токене должен быть установлен сертификат ЭП в результате выполнения предыдущих сценариев. Вызвать метод API для поиска личного сертификата ЭП и получения его хендла:

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Type: text/plain

Content-Length: 29

id=GET_OBJ_LIST_ID&obj_type=0
```

{пример запроса; окончание}

{пример ответа; начало}

```
data="qEpE0NTZ"&retcode="1"
```

{пример ответа; окончание}

Примечание: если таких сертификатов несколько, то data содержит в себе несколько хендлов через сепаратор.

в) Вызвать метод API для инициализации расчета ЭП:

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Type: text/plain

Content-Length: 74

id=INIT_SIGN_H_ID&datasize=4024&hascert=0&mode=1&hasdata=0&obj_id=qEpE0NTZ
```

{пример запроса; окончание}

{пример ответа; начало}

```
ctx_handle="hYfFctW8"&retcode="1"
```

{пример ответа; окончание}

г) Вызвать (возможно более одного раза) метод API для подачи данных в base64 для расчета ЭП:

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Type: text/plain

Content-Length: 5430

id=SET_SIGN_DATA_H_ID&data=<data_to_sign>&blocknum=1&ctx_handle=hYfFctW8
```

{пример запроса; окончание}

В ответ возвращается суммарный объем уже переданных на расчет данных:

{пример ответа; начало}

```
data_length="4024"&retcode="1"
```

{пример ответа; окончание}

д) Вызвать метод API для расчета ЭП, указав идентификатор сертификата:

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Type: text/plain

Content-Length: 37

id=CALC_SIGN_H_ID&ctx_handle=hYfFctW8
```

{пример запроса; окончание}

{пример ответа; начало}

```
retcode="1"
```

{пример ответа; окончание}

е) Вызвать (возможно более одного раза) метод API для получения информации о контексте. Контекст определяет, возможно ли уже получить данные ЭП (статус COMPLETE=2), или требуется подождать (статус WAITING=3).

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Type: text/plain

Content-Length: 40

id=GET_CTX_INFO_H_ID&ctx_handle=hYfFctW8
```

{пример запроса; окончание}

В ответ возвращается статус, суммарный объем уже переданных на расчет данных, количество рассчитанных ЭП:

{пример ответа; начало}

```
status="2"&data_length="26210"&sign_num="2"&retcode="1"
```

{пример ответа; окончание}

ж) По достижении статуса COMPLETE вызвать метод API для получения данных ЭП:

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Type: text/plain

Content-Length: 40

id=GET_SIGN_CMS_H_ID&ctx_handle=hYfFctW8
```

{пример запроса; окончание}

В ответ возвращаются данные ЭП в base64:

{пример ответа; начало}

```
head="<head_base64>"&suffix=""&retcode="1"
```

{пример ответа; окончание}

Проверка ЭП в CMS

- Выполнить действия по сценарию «Установление сессии на Устройстве», получить в результате SID.
- Вызвать метод API для инициализации проверки ЭП:

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Type: text/plain;

Content-Length: 787

id=INIT_CHECK_H_ID&mode=1&cms_data=<cms_data>
```

{пример запроса; окончание}

{пример ответа; начало}

```
ctx_handle="YsVGEQah"&retcode="1"
```

{пример ответа; окончание}

- Вызвать (возможно более одного раза, если одним вызовом переданы не все данные для подписания) метод API для подачи данных для проверки ЭП:

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Type: text/plain;

Content-Length: 5431

id=SET_CHECK_DATA_H_ID&data=<data>&blocknum=1&ctx_handle=YsVGEQah
```

{пример запроса; окончание}

В ответ возвращается суммарный объем уже переданных на проверку данных:

{пример ответа; начало}

```
data_length="4024"&retcode="1"
```

{пример ответа; окончание}

- Вызвать метод API для проверки ЭП:

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Type: text/plain;
```

```
  Content-Length: 38

  id=CHECK_SIGN_H_ID&ctx_handle= YsVGEQah
```

{пример запроса; окончание}

{пример ответа; начало}

```
  retcode="1"
```

{пример ответа; окончание}

Шифрование данных

а) Выполнить действия по сценарию «Установление сессии на токене», получить в результате SID

б) На токене должен быть установлен сертификат ЭП в результате выполнения предыдущих сценариев. Вызвать метод API для поиска личного сертификата ЭП и получения его хендла:

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Type: text/plain

Content-Length: 29

id=GET_OBJ_LIST_ID&obj_type=0
```

{пример запроса; окончание}

{пример ответа; начало===}

```
data="YBu4X4JG"&retcode="1"
```

{пример ответа; окончание}

Примечание: если таких сертификатов несколько, то data содержит в себе несколько хендлов через сепаратор.

в) Вызвать метод API для получения сертификата по его хэндлу:

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Type: text/plain

Content-Length: 36

id=GET_OBJ_CERT_D_ID&obj_id=YBu4X4JG
```

{пример запроса; окончание}

{пример ответа; начало}

```
data="<cert_data>"&retcode="1"
```

{пример ответа; окончание}

г) Вызвать метод API для инициализации шифрования:

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Type: text/plain
```

```
Content-Length: 39

id=INIT_ENCIPHER_ID&datasize=498&mode=1
```

{пример запроса; окончание}

{пример ответа; начало}

```
ctx_handle="ySH8zAXW"&retcode="1"
```

{пример ответа; окончание}

г) Вызвать метод API для добавления собственного сертификата в CMS:

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Type: text/plain

Content-Length: 57

id=ADD_RECIPIENT_X_ID&obj_id=YBu4X4JG&ctx_handle=ySH8zAXW
```

{пример запроса; окончание}

{пример ответа; начало}

```
data="<recipient_data>"&retcode="1"
```

{пример ответа; окончание}

д) Вызвать (возможно, более одного раза) метод API для шифрования данных:

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Type: text/plain

Content-Length: 78

id=ENCIPHER_ID&data=<data>&ctx_handle=ySH8zAXW
```

{пример запроса; окончание}

В ответ возвращаются зашифрованные данные в CMS:

{пример ответа; начало}

```
enciphered_blob="<enciphered_blob>"&retcode="1"
```

{пример ответа; окончание}

е) Вызвать (возможно, более одного раза) метод API для получения CMS:

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Type: text/plain

Content-Length: 42

id=GET_ENCIPHER_CMS_ID&ctx_handle=ySH8zAXW
```

{пример запроса; окончание}

{пример ответа; начало}

```
head=""&suffix=""&retcode="1"
```

{пример ответа; окончание}

Расшифрование данных

а) Выполнить действия по сценарию «Установление сессии на Устройстве», получить в результате SID.

б) На токене должен быть установлен сертификат ЭП в результате выполнения предыдущих сценариев. Вызвать метод API для поиска личного сертификата ЭП и получения его хендла:

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Type: text/plain

Content-Length: 29

id=GET_OBJ_LIST_ID&obj_type=0
```

{пример запроса; окончание}

{пример ответа; начало}

```
data="YBu4X4JG"&retcode="1"
```

{пример ответа; окончание}

Примечание: если таких сертификатов несколько, то data содержит в себе несколько хендлов через сепаратор.

в) Вызвать метод API для получения сертификата по его хэндлу:

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Type: text/plain

Content-Length: 36

id=GET_OBJ_CERT_D_ID&obj_id=YBu4X4JG
```

{пример запроса; окончание}

{пример ответа; начало}

```
data="<cert_data>"&retcode="1"
```

{пример ответа; окончание}

г) Вызвать (возможно, более одного раза) метод API для инициализации расшифрования:

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Type: text/plain

Content-Length: 1358

id=INIT_DECIPHER_ID&head=<head_data>&mode=1
```

{пример запроса; окончание}

{пример ответа; начало}

```
ctx_handle="pm79bha5"&body_displ="0"&retcode="1"
```

{пример ответа; окончание}

д) Вызвать (возможно, более одного раза) метод API для операции расшифрования:

{пример запроса; начало}

```
POST http://localhost:28016/vpnkeylocal/<SID>/ HTTP/1.1

Content-Type: text/plain

Content-Length: 1366

id=DECIPHER_ID&data=<encrypted_data>&ctx_handle=pm79bha5
```

{пример запроса; окончание}

{пример ответа; начало}

```
deciphered_blob="<data>"&retcode="1"
```

{пример ответа; окончание}