



ViPNet Client 5 for Windows SDK

Справочник разработчика

Версия продукта: 5.5

© АО «ИнфоТеКС», 2025

ФРКЕ.00403-01 33 01

Версия продукта 5.5

Этот документ входит в комплект поставки продукта ViPNet, и на него распространяются все условия лицензионного соглашения.

Ни одна из частей этого документа не может быть воспроизведена, опубликована, сохранена в электронной базе данных или передана в любой форме или любыми средствами, такими как электронные, механические, записывающие или иначе, для любой цели без предварительного письменного разрешения АО «ИнфоТеКС».

ViPNet® является зарегистрированным товарным знаком АО «ИнфоТеКС».

Все названия компаний и продуктов, которые являются товарными знаками или зарегистрированными товарными знаками, принадлежат соответствующим владельцам.

АО «ИнфоТеКС»

125167, г. Москва, вн. тер. г. муниципальный округ Хорошевский, ул. Викторенко, д. 9, стр. 1, помещ. 47

Телефон: +7 (495) 737-6192, 8 (800) 250-0260 — бесплатный звонок из России (кроме Москвы)

Сайт: infotecs.ru

Служба поддержки: hotline@infotecs.ru

Содержание

Введение.....	4
О документе.....	4
Соглашения документа.....	4
О программе	4
Начало работы с ViPNet Client 5 SDK	5
Функции SDK.....	6
VPN API.....	6
Cipher API.....	43
Windows API	47
Примеры	48
Обратная связь.....	49

Введение

О документе

Данный документ содержит инструкции по использованию SDK ViPNet Client 5 for Windows (далее — ViPNet Client 5) и описание функций этого SDK.

Документ предназначен для разработчиков, обеспечивающих взаимодействие ViPNet Client 5 с другими программами.

Соглашения документа

Обозначение	Описание
Название	Название элемента интерфейса: окна, вкладки, поля, кнопки, ссылки
Клавиша+Клавиша	Сочетание клавиш: нажмите первую клавишу и, не отпуская её, нажмите вторую
Меню > Команда	Последовательность элементов или действий
Код	Имя файла, путь, фрагмент кода или команда в командной строке



Примечание. В документе могут присутствовать снимки интерфейса из предыдущих версий продукта. Поэтому некоторые элементы интерфейса, которые не влияют на понимание текста, могут выглядеть не так, как в продукте.

О программе

Программа ViPNet Client 5 предназначена для защиты IP-трафика на компьютерах с Windows путём шифрования IP-пакетов.

Начало работы с ViPNet Client 5 SDK

- 1 Установите ViPNet Client 5. Подробнее см. руководство пользователя.
- 2 В зависимости от архитектуры или режима работы программы скопируйте на компьютер содержимое архива `libvpn_api.zip`:

- o Windows API
 - x86
 - o `libvpn_api.dll`
 - o `libvpn_api.lib`
 - x64
 - o `libvpn_api.dll`
 - o `libvpn_api.lib`
- o include
 - `business_mail_api.h`
 - `encryption_api.h`
 - `errors.h`
 - `vpn_api_version.h`
 - `vpn_api.h`
 - `vpn_api_types.h`
 - `cipher_api.h`
 - `events.h`

- 3 Подключите заголовочные файлы:

```
#include <vipnet/vpn_api.h>
#include <vipnet/cipher_api.h>
```

- 4 При сборке проекта выполните линковку с библиотекой `libvpn_api.lib` или во время работы загружайте библиотеку `libvpn_api.dll` с помощью функции `LoadLibrary()`. Используйте библиотеки из папки `Windows API`.

- 5 Проверьте, что версия SDK, с которой собирается ViPNet Client 5, соответствует версии SDK, поставляемой вместе с ViPNet Client 5:

```
if( VPN_API_VERSION != GetVpnApiVersion() )
{
    return "Version mismatch";
}

if( CIPHER_API_VERSION != GetCipherApiVersion() )
{
    return "Version mismatch";
}
```

6 Получите структуры с указателями на функции SDK:

```
ItcsVpnApi* api = GetVpnApi();  
ItcsCipherApi* capi = GetCipherApi();
```

Если будут возвращены ненулевые указатели, то вы сможете вызывать функции SDK.

Функции SDK

VPN API

Таблица 1. *ItcsVpnApi*

Функция	Описание
<code>ItcsVpnApi* GetVpnApi(void)</code>	Получение указателя на структуру VPN API с проинициализированными указателями на функции
<code>ItcsVpnApi* GetVpnExt(VpnApiInitData* params, size_t len)</code>	Получение указателя на структуру VPN API с заданными параметрами инициализации. • [in] params — параметры инициализации; • [in] len — количество параметров
<code>uint32_t GetVpnApiVersion(void)</code>	Получение номера текущей версии VPN API
<code>VpnApiReturnCode (*CheckActiveCoordinatorConnection) (uint32_t* nodeId)</code>	Проверка соединения с активным координатором. • [out] nodeId — идентификатор активного координатора
<code>VpnApiReturnCode (*CheckNodeConnection) (uint32_t nodeId)</code>	Проверка соединения с узлом. • [in] nodeId — идентификатор узла
<code>VpnApiReturnCode (*CollectEventlog) (const char* path, const char* psw, bool truncate)</code>	Экспорт журнала событий в ZIP-архив. В журнал vipnetadm.log записываются такие события как включение и выключение VPN-соединения, проверка лицензии. • [in] path — путь для сохранения файла архива; • [in] psw — пароль администратора сетевого узла ViPNet; • [in] truncate — признак необходимости удаления журнала после экспорта
<code>VpnApiReturnCode (*CollectReport) (const char* path)</code>	Экспорт отчёта о работе программы ViPNet Client 5 в ZIP-архив. • [in] path — путь для сохранения файла архива

Функция	Описание
<code>VpnApiReturnCode (*CollectTokens)(int (*cbk) (VpnTokenHandle))</code>	Получение списка подключённых и поддерживаемых внешних устройств (токенов). [in] <code>cbk</code> — обработчик дескриптора токена; может вернуть не 0 для прерывания опроса
<code>VpnApiReturnCode (*DeleteKeys)(void)</code>	Удаление ключей ViPNet
<code>VpnApiReturnCode (*GetActiveCoordinator)(uint32_t* id)</code>	Получение идентификатора (ID) активного координатора (сервера IP-адресов). [out] <code>id</code> — ID узла
<code>VpnApiReturnCode (*GetClientParam)(const char* key, uint32_t keyLength, char** outData, uint32_t* outDataLength)</code>	Получение конфигурационного параметра ViPNet Client 5 по ключу поиска. [in] <code>key</code> — ключ для поиска данных; [in] <code>keyLength</code> — длина буфера с ключом поиска данных; [out] <code>outData</code> — буфер для возвращения данных; [out] <code>outDataLength</code> — размер динамически выделенного буфера с данными. Память для <code>outData</code> выделяется динамически и должна быть освобождена с помощью <code>ReleaseClientParamData(outData)</code>
<code>VpnApiReturnCode (*GetClientVersion)(char* version, size_t* size)</code>	Получить текущую версию ViPNet Client 5. [out] <code>version</code> — буфер для передачи версии; [in,out] <code>size</code> — размер буфера [in], размер записанных данных с завершающим <code>'\0'</code> [out]. Память для <code>version</code> должна быть выделена и освобождена самостоятельно
<code>VpnApiReturnCode (*GetDstInfo)(const char* path, VpnDstInfo* info)</code>	Получение информации о дистрибутиве ключей. [in] <code>path</code> — путь к файлу <code>*.dst</code> или <code>*.ds5</code>; [out] <code>info</code> — информация о дистрибутиве ключей
<code>VpnApiReturnCode (*GetHealthStatus)(VpnHealthStatus* status)</code>	Проверить работоспособности службы <code>vipnetclient</code>. [out] <code>status</code> — результат проверки
<code>VpnApiReturnCode (*GetLicenseExpiration)(time_t* time)</code>	Получить дату и время истечения срока действия лицензии в секундах в формате POSIX. [out] <code>time</code> — дата и время истечения срока действия лицензии в секундах

Функция	Описание
<pre>VpnApiReturnCode (*GetLogLevel) (int32_t* logLevel)</pre>	Получение текущего уровня протоколирования событий ViPNet Client 5. [out] logLevel — уровень протоколирования Получить информацию о шифраторе заданного узла. [in] id — ID узла; [out] keyType — указатель на строку с JSON-структурой в UTF-8. Строка с результатом должна быть освобождена после использования с помощью вызова ReleaseNodeCrypto(keyType). Возвращаемая структура имеет следующий формат: <pre>{ "keyType": "<Тип ключа: Default, Quantum, etc>", "keyStartDate": "<Время начала действия квантового ключа (time_t)>" }</pre>
<pre>VpnApiReturnCode (*GetNodeInfo) (uint32_t id, VpnNodeInfo** nodeInfo)</pre>	Получение информации об узле ViPNet по его идентификатору (ID). [in] id — ID узла; [out] nodeInfo — информация об узле. Память для nodeInfo выделяется динамически и должна быть освобождена с помощью ReleaseVpnNodeInfo(nodeInfo, 1)
<pre>VpnApiReturnCode (*GetNodeInfoExt) (uint32_t id, VpnNodeInfoExt** nodeInfo)</pre>	Получение расширенной информации об узле ViPNet по его идентификатору (ID). [in] id — ID узла; [out] nodeInfo — информация об узле. Память для nodeInfo выделяется динамически и должна быть освобождена с помощью ReleaseVpnNodeInfoExt(nodeInfo)
<pre>VpnApiReturnCode (*GetNodesIds) (uint32_t** nodeIds, uint32_t* size)</pre>	Получение списка идентификаторов (ID) узлов ViPNet. [out] nodeIds — список ID узлов; [out] size — количество узлов. Память для nodeIds выделяется динамически и должна быть освобождена с помощью ReleaseVpnNodeIds(nodeIds, size)

Функция	Описание
<pre>VpnApiReturnCode (*GetNodesInfo) (VpnNodeInfo** nodesInfo, uint32_t* size)</pre>	Получение информации об узлах ViPNet. • [out] nodesInfo — информация об узлах ViPNet; • [out] size — количество узлов. Память для nodesInfo выделяется динамически и должна быть освобождена с помощью ReleaseVpnNodeInfo(nodesInfo, size)
<pre>VpnApiReturnCode (*GetNodesInfoByTask) (VpnNodeInfo** nodesInfo, size_t* size, uint32_t tasksMask)</pre>	Получение информации об узлах ViPNet по маске ролей и свойств узла. • [out] nodesInfo — информация об узлах ViPNet; • [out] size — количество узлов; • [in] tasksMask — битовая маска ролей и свойств узла. Память для nodesInfo выделяется динамически и должна быть освобождена с помощью ReleaseVpnNodeInfo(nodesInfo, size)
<pre>VpnApiReturnCode (*GetOwnNodeInfo) (VpnNodeInfo** nodeInfo)</pre>	Получение информации о своём узле. • [out] nodeInfo — информация об узле. Память для nodeInfo выделяется динамически и должна быть освобождена с помощью ReleaseVpnNodeInfo(nodeInfo, 1)
<pre>VpnApiReturnCode (*GetOwnPrivilegeLevel) (enum VpnPrivilege* privilege)</pre>	Получение уровня полномочий своего узла. • [out] privilege — уровень полномочий

Функция	Описание
<pre>VpnApiReturnCode (*GetTokenConfig) (int (*cbk) (const char*))</pre>	Получить конфигурацию модулей PKCS#11 и статус их использования. [in] cbk — обработчик информации, получает JSON-структуру в utf8. Возвращаемая структура имеет следующий формат: <pre>{ "global": { "paths": "<path1>:<path2>:...:<pathN>" }, "token01": { "name": "<наименование>", "file": "<имя файла модуля pkcs#11>", "type": "<числовой идентификатор типа>", "check": { "library": "<актуальный путь к модулю pkcs#11, если найден>", "load": "<true/false: загружена библиотека pkcs#11 в CSP или нет>" } }, "token02": { ... }, ... "tokenNN": { ... } }</pre>
<pre>VpnApiReturnCode (*GetTokenInfo) (VpnTokenHandle handle, VpnTokenInfo* info)</pre>	Получение параметров токена. [in] handle — дескриптор токена, информацию о котором нужно получить; [out] info — указатель на структуру, в которую будет скопирована информация. VpnTokenInfo содержит указатели на строки, эти объекты уничтожаются при вызове ReleaseTokenHandle(handle)
<pre>VpnApiReturnCode (*GetUltimateStatus) (VpnUltimateStatus* status)</pre>	Получение информации о состоянии и режиме работы службы vipnetclient. [out] status — запрошенные данные

Функция	Описание
<pre>VpnApiReturnCode (*GetUserIdsForNode) (uint32_t nodeId, uint32_t** userIds, size_t* size)</pre>	Получение списка ID пользователей, зарегистрированных на данном узле. • [in] nodeId — ID узла; • [out] userIds — список ID пользователей; • [out] size — количество пользователей. Память для userIds выделяется динамически и должна быть освобождена с помощью ReleaseVpnUserIds(userIds, size)
<pre>VpnApiReturnCode (*GetUserInfo) (uint32_t id, VpnUserInfo** userInfo)</pre>	Получение информации о пользователе по его ID. • [in] id — ID пользователя; • [out] userInfo — информация о пользователе. Память для userInfo выделяется динамически и должна быть освобождена с помощью ReleaseVpnUserInfo(userInfo, size)
<pre>VpnApiReturnCode (*GetUsersInfo) (VpnUserInfo** usersInfo, size_t* size)</pre>	Получение информации о пользователях. • [out] usersInfo — информация о пользователях; • [out] size — количество пользователей. Память для usersInfo выделяется динамически и должна быть освобождена с помощью ReleaseVpnUserInfo(userInfo, size). Структура VpnUserInfo имеет переменную длину
<pre>VpnApiReturnCode (*GetVpnStatus) (VpnStatus* vpnStatus)</pre>	Функция возвращает объект, описывающий текущее состояние VPN-соединения. • [out] vpnStatus — текущий статус VPN-соединения
<pre>VpnApiReturnCode (*InstallKeys) (const char* dstPath, const char* psw)</pre>	Установка ключей ViPNet. • [in] dstPath — путь к дистрибутиву ключей (*.dst или *.ds5), который будет установлен; • [in] psw — пароль к дистрибутиву ключей
<pre>VpnApiReturnCode (*InstallKeysExt) (const char* path, const char* password, VpnDstStatus* status)</pre>	Установка ключей ViPNet и настройка службы для доступа в сеть ViPNet. • [in] path — путь к дистрибутиву ключей (*.dst или *.ds5); • [in] password — пароль к дистрибутиву ключей (может быть пуст или NULL); • [out] status — результат проверки файла *.dst или *.ds5 и статус установки ключей

Функция	Описание
<pre>void (*LogEventHandler) (enum LoggerLevel logLevel, const char* message)</pre>	Прототип обработчика журналов. [in] logLevel — уровень детализации записей в журнале. Доступны 5 уровней детализации; [in] message — сообщение. По умолчанию установлен уровень детализации 3
<pre>VpnApiReturnCode (*MoveKeysToToken) (VpnTokenHandle handle, const char* pin)</pre>	Перенос персонального ключа на внешнее устройство (токен) и смена типа аутентификации на «Персональный ключ на внешнем устройстве». [in] handle — дескриптор токена, который должен быть использован для аутентификации; [in] pin — PIN-код пользователя токена (немаскированный). Для успешного выполнения команды VPN-соединение должно быть включено
<pre>VpnApiReturnCode (*RegulationsCheck) (const char* psw)</pre>	Выполнение регламентного контроля целостности ViPNet Client 5. [in] psw — пароль администратора сетевого узла
<pre>void (*ReleaseClientParamData) (char* data)</pre>	Освобождение памяти, выделенной для информации о конфигурационном параметре ViPNet Client 5. [in] data — буфер, который необходимо освободить
<pre>VpnApiReturnCode (*ReleaseNodeInfoExt) (VpnNodeInfoExt* nodeInfo)</pre>	Освобождение памяти, выделенной для расширенной информации об узлах ViPNet. [in] nodeInfo — информация об узлах, память которой необходимо освободить
<pre>VpnApiReturnCode (*ReleaseTokenHandle) (VpnTokenHandle handle)</pre>	Освобождение памяти, выделенной для дескриптора токена. [in] handle — дескриптор токена. Следует вызывать для каждого дескриптора, когда он перестаёт быть нужным (с учётом примечания в описании GetTokenInfo)
<pre>void (*ReleaseVpnNodeIds) (uint32_t* nodeIds, uint32_t size)</pre>	Освобождение памяти, выделенной для списка узлов ViPNet. [in] nodeIds — массив ID узлов, память которого необходимо освободить; [in] size — количество узлов

Функция	Описание
<pre>void (*ReleaseVpnNodeInfo)(VpnNodeInfo* nodeInfo, uint32_t size)</pre>	Освобождение памяти, выделенной для информации об узлах ViPNet. • [in] nodeInfo — информация об узлах, память которой необходимо освободить; • [in] size — количество узлов
<pre>void (*ReleaseVpnUserIds)(uint32_t* userIds, size_t size)</pre>	Освобождение памяти, выделенной для списка пользователей. • [in] userIds — массив ID пользователей, память которого необходимо освободить; • [in] size — количество пользователей
<pre>void (*ReleaseVpnUserInfo)(VpnUserInfo* userInfo, size_t size)</pre>	Освобождение памяти, выделенной для информации о пользователях. • [in] userInfo — информация о пользователях, память которой необходимо освободить; • [in] size — количество пользователей
<pre>VpnApiReturnCode (*SetActiveCoordinatorId)(uint32_t nodeId)</pre>	Смена активного координатора. • [in] nodeId — идентификатор нового активного координатора
<pre>VpnApiReturnCode (*SetEventHandlers)(const char* consumer, VpnEventHandlers* events)</pre>	Установить обработчики событий службы vipnetclient. Позволяет установить индивидуальные функции для обработки каждого события, а также принимать параметры этих событий (если предусмотрено). • [in] consumer — уникальный идентификатор подписчика (не должен содержать пробелы); • [in] events — структура указателей на функции-обработчики
<pre>VpnApiReturnCode (*SetExchangeFolder)(const wchar_t* folder)</pre>	Задать папку обмена для настройки RPC через TCP/IP. • [in] folder — папка обмена
<pre>VpnApiReturnCode (*SetLogLevel)(int32_t logLevel)</pre>	Выставление уровня протоколирования событий ViPNet Client 5. • [in] logLevel — уровень протоколирования
<pre>VpnApiReturnCode (*SetOptionAutorun)(void)</pre>	Включение функции автозапуска VPN-соединения при авторизации пользователя в ОС
<pre>VpnApiReturnCode (*SetOptionNoAutorun)(void)</pre>	Выключение функции автозапуска VPN-соединения при авторизации пользователя в ОС
<pre>VpnApiReturnCode (*SetOptionFaultTolerance)(void)</pre>	Включение функции восстановления ViPNet Client 5 при сбоях. По умолчанию система восстановления активирована

Функция	Описание
<code>VpnApiReturnCode (*SetOptionNoFaultTolerance) (void)</code>	Выключение функции восстановления ViPNet Client 5 при сбоях
<code>VpnApiReturnCode (*SetPin) (VpnTokenHandle handle, const char* pin)</code>	Сохранение PIN-код от токена, используемого для аутентификации. PIN-код сохраняется в маскированном виде и удаляется при выполнении <code>ReleaseTokenHandle(handle)</code>. [in] <code>handle</code> — дескриптор токена, PIN-код которого нужно сохранить; [in] <code>pin</code> — PIN-код пользователя токена (немаскированный)
<code>VpnApiReturnCode (*SetToken) (VpnTokenHandle handle, const char* pin)</code>	Задание параметров токена, используемого для аутентификации. [in] <code>handle</code> — дескриптор токена, который должен быть использован для аутентификации; [in] <code>pin</code> — PIN-код пользователя токена (немаскированный)
<code>VpnApiReturnCode (*StartVpn) (const char* psw)</code>	Включение VPN-соединения. [in] <code>psw</code> — пароль к дистрибутиву ключей
<code>VpnApiReturnCode (*StopVpn) (void)</code>	Выключение VPN-соединения
<code>VpnApiReturnCode (*SubscribeForEvents) (VpnEventHandler eventHandler)</code>	Подписка на события VPN-соединения. Поддерживается подписка только одного обработчика. Не рекомендуется отменять подписки внутри обработчика событий. [in] <code>eventHandler</code> — обработчик событий
<code>VpnApiReturnCode (*SubscribeForEventsExt) (void* data, VpnEventHandlerExt eventHandler)</code>	Подписка на события VPN-соединения для объекта <code>data</code>. Для отмены подписки необходимо использовать <code>UnsubscribeFromEventsExt</code>. Не рекомендуется отменять подписки внутри обработчика событий. [in] <code>data</code> — ненулевой указатель на произвольные данные, который будет передан обработчику событий; [in] <code>eventHandler</code> — обработчик событий
<code>VpnApiReturnCode (*UnsubscribeFromEventsExt) (void* data)</code>	Отмена подписки на события VPN-соединения для объекта <code>data</code>. [in] <code>data</code> — ненулевой указатель на произвольные данные, для которого нужно отменить подписку
<code>void (*VpnEventHandler) (enum VpnEvent event)</code>	Прототип обработчика событий. [in] <code>event</code> — событие VPN-соединения

Функция	Описание
<pre>void (*VpnEventHandlerExt)(void* data, enum VpnEvent event)</pre>	Прототип обработчика событий с дополнительными данными. [in] data — ненулевой указатель на произвольные данные; [in] event — событие VPN-соединения
<pre>VpnApiReturnCode (*GetConnectionId)(int32_t* conn)</pre>	Получение идентификатора запущенного ViPNet-соединения. [out] conn — идентификатор запущенного ViPNet-соединения. Не поддерживается
<pre>VpnApiReturnCode (*GetConnectionIdRoutine)(int32_t* conn)</pre>	Получение идентификатора запущенного ViPNet-соединения через DBus-RPC вызов. [out] conn — идентификатор запущенного ViPNet-соединения. Не поддерживается
<pre>VpnApiReturnCode (*BlockUserTraffic)(enum VpnUserTrafficBlockStatus status)</pre>	Изменение состояния блокировки пользовательского трафика. [in] status — новое состояние
<pre>VpnApiReturnCode (*GetUserTrafficBlockInfo)(VpnUserTrafficBlockInfo** blockInfo)</pre>	Получение информации о блокировке пользовательского трафика. [out] blockInfo — информация о блокировке пользовательского трафика. Память для blockInfo выделяется динамически и должна быть освобождена с помощью ReleaseUserTrafficBlockInfo(blockInfo)
<pre>VpnApiReturnCode (*ReleaseUserTrafficBlockInfo)(VpnUserTrafficBlockInfo* blockInfo)</pre>	Освобождение памяти, выделенной для информации о блокировке пользовательского трафика. [in] blockInfo — информация о блокировке пользовательского трафика, память которой нужно освободить
<pre>VpnApiReturnCode (*GetAvailableRefBooks)(VpnRefBookListInfo** refBookList)</pre>	Получение списка доступных справочников конфигурации. [out] refBookList — список доступных справочников конфигурации. Память для refBookList выделяется динамически и должна быть освобождена с помощью ReleaseRefBookList(refBookList)

Функция	Описание
<code>VpnApiReturnCode (*ExportRefBook) (const char* refBook, const char* filepath)</code>	Экспортировать файл справочника конфигурации. [in] <code>refBook</code> — имя справочника конфигурации; [in] <code>filepath</code> — путь для экспорта файла справочника
<code>VpnApiReturnCode (*ReleaseRefBookList) (VpnRefBookListInfo* refBookList)</code>	Освобождение памяти, выделенной для списка доступных справочников конфигурации. [in] <code>refBookList</code> — список доступных справочников конфигурации, память которого нужно освободить
<code>VpnApiReturnCode (*GetTunnelVisibility) (enum VpnTunnelVisibilityMode* mode)</code>	Получение режима видимости туннелей. [out] <code>mode</code> — режим видимости туннелей
<code>VpnApiReturnCode (*SetTunnelVisibility) (enum VpnTunnelVisibilityMode mode)</code>	Установка режима видимости туннелей. [in] <code>mode</code> — режим видимости туннелей
<code>VpnApiReturnCode (*SetOptionConnectServerGroup) (enum VpnConnectServerGroupState state)</code>	Включение и выключение настройки «Группа серверов соединений». [in] <code>state</code> — желаемое состояние настройки «Группа серверов»
<code>VpnApiReturnCode (*GetOptionConnectServerGroup) (enum VpnConnectServerGroupState* state)</code>	Получение информации о состоянии настройки «Группа серверов соединений». [out] <code>state</code> — состояние настройки «Группа серверов»
<code>VpnApiReturnCode (*ChangeCredentials) (const char* password)</code>	Смена пароля. [in] <code>password</code> — новый пароль
<code>VpnApiReturnCode (*ContinueChangeCredentials) (const char* oldPassword, const char* newPassword)</code>	Возобновление смены пароля после сбоя в GLE. [in] <code>oldPassword</code> — текущий пароль; [in] <code>newPassword</code> — новый пароль
<code>VpnApiReturnCode (*GetDmsStatus) (enum VpnDmsStatus* status)</code>	Получение информации о состоянии мастер-ключа устройства. [out] <code>status</code> — состояние мастер-ключа устройства
<code>VpnApiReturnCode (*GetExpiredDateDeferredChangeDms) (time_t* date)</code>	Получение информации о дате, на которую перенесена смена пароля. [out] <code>date</code> — дата, на которую перенесли смену пароля

Функция	Описание
<code>VpnApiReturnCode (*DeferChangeDms) (uint32_t months)</code>	Перенос смены пароля. [in] months — число месяцев, на которое нужно перенести смену пароля
<code>VpnApiReturnCode (*GetDmsLastChangeDate) (time_t* date)</code>	Получение информации о дате последней смены пароля. [out] date — дата последней смены пароля
<code>VpnApiReturnCode (*GetDmsExpiredDate) (time_t* date)</code>	Получение информации о дате истечения пароля. [out] date — дата истечения пароля
<code>VpnApiReturnCode (*SetOptionActiveCoordinatorInsist) (bool insist)</code>	Изменение режима выбора сервера соединений. [in] insist — игнорировать настройки активного координатора из управляющего ПО
<code>VpnApiReturnCode (*GetOptionActiveCoordinatorInsist) (bool* insist)</code>	Получение информации о режиме выбора сервера соединений. [out] insist — игнорируются настройки активного координатора из управляющего ПО
<code>VpnApiReturnCode (*GetUserTrafficRestrictions) (VpnUserTrafficRestrictions restrictions)</code>	Получение списка причин блокировки пользовательского трафика. [out] restrictions — список причин блокировки пользовательского трафика
<code>VpnApiReturnCode (*StartMfAuth) (void)</code>	Запуск многофакторной аутентификации
<code>VpnApiReturnCode (*GetMfAuthStatus) (VpnMfAuthStatus* status)</code>	Получение статуса многофакторной аутентификации. [out] status — статус многофакторной аутентификации
<code>VpnApiReturnCode (*GetMfAuthChallengeTips) (VpnMfAuthChallengeTipsList** tips)</code>	Получение списка доступных способов многофакторной аутентификации. [out] tips — список доступных способов многофакторной аутентификации
<code>void (*ReleaseMfAuthChallengeTips) (VpnMfAuthChallengeTipsList* tips)</code>	Освобождение памяти, выделенной для списка доступных способов многофакторной аутентификации. [in] tips — список доступных способов многофакторной аутентификации, память которого нужно освободить
<code>VpnApiReturnCode (*EnterMfAuthChallengeCredentials) (const char* login, const char* password)</code>	Ввод доменных учётных данных для многофакторной аутентификации. [in] login — имя пользователя; [in] password — пароль

Функция	Описание
<code>VpnApiReturnCode (*EnterMfAuthChallengeData)(const char* data)</code>	Ввод или выбор второго фактора для многофакторной аутентификации. [in] data — второй фактор для многофакторной аутентификации
<code>VpnApiReturnCode (*CancelMfAuth)(void)</code>	Отмена многофакторной аутентификации
<code>VpnApiReturnCode (*GetMfAuthAllowRestartIn)(uint32_t* timeout)</code>	Получение информации о времени, через которое можно повторить многофакторную аутентификацию. [out] timeout — время, через которое можно повторить многофакторную аутентификацию
<code>VpnApiReturnCode (*InstallKeysV10)(const char* path, const char* password, VpnDstCheckListInfoV10** status)</code>	Установка ключей. [in] path — путь к дистрибутиву ключей; [in] password — пароль узла; [out] status — список ошибок при установке ключей
<code>VpnApiReturnCode (*GetDstInfoV10)(const char* path, VpnDstInfoV10* info, VpnDstCheckListInfoV10** status)</code>	Получение информации о дистрибутиве ключей. [in] path — путь к дистрибутиву ключей; [out] info — информация о дистрибутиве ключей; [out] status — список ошибок при установке ключей
<code>void (*ReleaseDstCheckListInfoV10)(VpnDstCheckListInfoV10* info)</code>	Освобождение памяти, выделенной для списка ошибок при установке ключей. [in] info — список ошибок при установке ключей, память которого нужно освободить
<code>VpnApiReturnCode (*CheckUserPassword)(const char* password)</code>	Проверка пароля. [in] password — пароль узла
<code>VpnApiReturnCode (*GetLoginAvailabilityInfo)(VpnLoginAvailabilityInfo* info)</code>	Получение информации о возможности повторить попытку входа. [out] info — информации о возможности повторить попытку входа
<code>VpnApiReturnCode (*GetKeySystemVersion)(VpnKeySystemVersion* ks)</code>	Получение информации о поколении продукта. [out] ks — поколение продукта
<code>VpnApiReturnCode (*AdoptEnvironment)(void)</code>	Пересчёт контрольных сумм окружения

Функция	Описание
<code>VpnApiReturnCode (*IsTrustedAccessGranted) (bool* granted)</code>	Получение признака доверенности процесса. [out] granted — признак доверенности процесса
<code>VpnApiReturnCode (*GetIgnorePrivateDnsServers) (bool* isIgnore)</code>	Получение признака «Не использовать защищённые DNS-серверы для запросов открытой сети». [out] isIgnore — игнорировать DNS-запросы открытой сети
<code>VpnApiReturnCode (*SetIgnorePrivateDnsServers) (bool isIgnore)</code>	Установка признака «Не использовать защищённые DNS для запросов открытой сети» [in] isIgnore — игнорировать DNS-запросы открытой сети
<code>VpnApiReturnCode (*ReplaceUserVpnconfig) (const char* userVpnConfig)</code>	Применение файла настроек. [in] userVpnConfig — файл настроек
<code>VpnApiReturnCode (*CollectProfiles) (bool (*cbk) (ProfileHandle hdl))</code>	Получение списка доступных профилей. [in] cbk — обработчик дескриптора профиля. При возвращении false перебор списка прекращается
<code>VpnApiReturnCode (*GetProfileInfo) (ProfileHandle hdl, ProfileInfo* info)</code>	Получение информации о профиле по его дескриптору. [in] hdl — дескриптор профиля; [out] info — информация о профиле
<code>VpnApiReturnCode (*ActivateProfile) (ProfileHandle hdl)</code>	Сделать профиль активным. [in] hdl — дескриптор профиля, который нужно сделать активным
<code>VpnApiReturnCode (*DeleteProfile) (ProfileHandle hdl)</code>	Удалить профиль. [in] hdl — дескриптор профиля, который нужно удалить
<code>VpnApiReturnCode (*FreeProfile) (ProfileHandle hdl)</code>	Освободить память, связанную с указанным профилем. [in] hdl — дескриптор профиля
<code>VpnApiReturnCode (*SetProfileCustomName) (const char* name)</code>	Задать имя активного профиля. [in] name — новое имя активного профиля

Функция	Описание
<pre>VpnApiReturnCode (*AddCustomAdminLogEntries)(const char* name, const char* programname, const char* entries)</pre>	Записать в журнал аудита набор сообщений. • [in] name — имя журнала аудита; • [in] programname — имя программы; • [in] entries — набор записей, разделённых символом перевода строки
<pre>VpnApiReturnCode (*ImportEnvelope)(const char** files, int size)</pre>	Импортировать транспортный конверт. • [in] files — набор путей к конвертам в кодировке UTF-8; • [in] size — размер списка конвертов
<pre>typedef void (*GetStringFn)(void* context, const char* str)</pre>	Прототип функции получения строки. • [in,out] context — определяемый пользователем контекст для взаимодействия со строкой; • [in] str — получаемая строка
<pre>VpnApiReturnCode (*GetClientVersionEx)(void* context, GetStringFn get_version_fn)</pre>	Получить текущую версию ViPNet Client 5. • [in,out] context — определяемый пользователем контекст для взаимодействия с полученной версией; • [in] get_version_fn — прототип функции получения строки
<pre>VpnApiReturnCode (*GetMfAuthLastLogin)(void* context, GetStringFn get_login_fn)</pre>	Получить сохранённое имя пользователя для многофакторной аутентификации. • [in,out] context — определяемый пользователем контекст для взаимодействия с полученным именем пользователя; • [in] get_login_fn — прототип функции получения строки
<pre>VpnApiReturnCode (*SetMfAuthLastLogin)(const char* login)</pre>	Сохранить имя пользователя для многофакторной аутентификации. • [in] login — имя пользователя для многофакторной аутентификации
<pre>VpnApiReturnCode (*GetNodeDeviceInfo)(uint32_t id, VpnNodeDeviceInfo* info)</pre>	Получение информации об устройстве по идентификатору узла. • [in] id — идентификатор узла; • [out] info — информация об устройстве

Таблица 2. Ключи для инициализации с помощью функции *GetVpnApiExt*

Ключ	Описание
<code>const char* const initKeyCheckSupportedFunctions = "check_supported_functions"</code>	Ключ для работы с режимом проверки поддержки функций. Требуется для тестирования ViPNet Client 5
<code>const char* const initKeyAppGroupName = "app_group_name"</code>	Ключ для задания имени группы, специфичен для платформы Apple
<code>const char* const initKeyAppName = "app_name"</code>	Ключ для задания имени приложения, специфичен для платформы Apple
<code>const char* const initKeyLogHandler = "log_handler"</code>	Ключ для передачи обработчика, перенаправляющего журналы в вызывающий код
<code>const char* const initKeyLogHandlerLevel = "log_level"</code>	Ключ для передачи уровня детализации журналов из вызывающего кода
<code>const char* const initKeyWorkDirs = "WorkDirs"</code>	Ключ для изменения рабочих директорий в ViPNet Client 5, специфичен для Windows
<code>const char* const initKeyRpcPort = "RpcPort"</code>	Ключ для указания номера порта подключения к ViPNet Client 5, специфичен для Windows

Таблица 3. Ключи для получения параметров с помощью функции *GetClientParam*

Ключ	Описание
<code>const char* const paramItcslicXmlData = "data/itcslic.xml"</code>	Ключ для получения содержимого файла лицензии <code>itcslic.xml</code> . Работает только для клиентов с ключами <code>*.dst</code> . Для клиентов с ключами <code>*.ds5</code> используйте функцию <code>ExportRefBook</code>
<code>const char* const paramKeyNetworkNumber = "network.number"</code>	Ключ для получения номера сети ViPNet
<code>const char* const paramNodeDeviceId = "node.deviceId"</code>	Ключ для получения идентификатора (ID) устройства, который используется для отправки данных в систему мониторинга ViPNet NVS
<code>const char* const paramNvsAuthenticationId = "nvs.info/authentication.id"</code>	Ключ для получения идентификатора авторизации в системе мониторинга ViPNet NVS
<code>const char* const paramNvsAuthenticationSecret = "nvs.info/authentication.secret"</code>	Ключ для получения секрета авторизации в системе мониторинга ViPNet NVS
<code>const char* const paramNvsConnectionOpenUrls = "nvs.info/connection.openUrls"</code>	Ключ для получения ссылки для доступа к системе мониторинга ViPNet NVS по открытой сети
<code>const char* const paramNvsConnectionVipnetUrls = "nvs.info/connection.vipnetUrls"</code>	Ключ для получения ссылки для доступа к системе мониторинга ViPNet NVS по защищённой сети

Ключ	Описание
<code>const char* const paramNvsInfoData = "data/nvs.info"</code>	Ключ для получения содержимого файла <code>nvs.info</code> с параметрами подключения к ViPNet NVS. Работает только для клиентов с ключами <code>*.dst</code> . Для клиентов с ключами <code>*.ds5</code> используйте функцию <code>ExportRefBook</code>
<code>const char* const paramSecurityEncryptionMode = "node.security.encryption.mode"</code>	Ключ для получения используемого типа шифрования
<code>const char* const paramSystemNetworkMonitoringOpenUrls = "system.monitoring.openUrls"</code>	Ключ для получения ссылки для доступа к системе мониторинга по открытой сети
<code>const char* const paramSystemNetworkMonitoringVipnetUrls = "system.monitoring.vipnetUrls"</code>	Ключ для получения ссылки для доступа к системе мониторинга по защищённой сети

Таблица 4. *VpnApiReturnCode*

Поле	Описание
<code>uint32_t code</code>	Код возврата функций (0 — успешное завершение)
<code>const char* message</code>	Сообщение об ошибке (пустое при успешном завершении). Не требует освобождения памяти. Сообщение хранится до следующего вызова команды

Таблица 5. *VpnErrors*

Поле	Описание
<code>Success</code>	0
<code>ErrInvalidParameter</code>	0x00020002
<code>ErrUnmergeContext</code>	0x00030000
<code>ErrKeysIntegrityFailed</code>	0x00030040
<code>ErrIplircfgFailed</code>	0x00030200
<code>ErrKeysNotFoundInFile</code>	0x00030400
<code>ErrKeysUserNotRegistered</code>	0x00030800
<code>ErrDstContext</code>	0x00040000
<code>ErrDstContextMask</code>	0x0004ffff
<code>ErrPasswordContext</code>	0x00050000
<code>ErrPasswordIsWrong</code>	0x00050001

Поле	Описание
ErrNoPasswordAttemptsLeft	0x00050002
ErrPasswordIsExpired	0x00050003
ErrCriticalAuthFailure	0x00050004
ErrWrongLoginState	0x00050005
ErrLoginCounterFileError	0x00050006
ErrFinalPasswordAttempt	0x00050008
ErrStartContext	0x00060000
ErrStartKeyNotFound	0x00060008
ErrStartIplircfgWrongPin	0x000600d9
ErrStartIplircfgUserPinLocked	0x000600e2
ErrClientContext	0x00070000
ErrVpnWorks	0x00070001
ErrVpnDoesntWork	0x00070002
ErrGetNodeList	0x00070003
ErrVpnAlreadyStarted	0x00070005
ErrGetNodeInfo	0x00070006
ErrCoordinatorSetFailed	0x00070007
ErrCheckConnection	0x00070008
ErrBinaryCheckFailed	0x00070009
ErrServiceContext	0x00090000
ErrServiceUserUndefined	0x00090003
ErrServiceActionCanceled	0x00090004
ErrServiceIntegrityBroken	0x00090005
ErrServiceDoesNotRespond	0x00090007
ErrSetCapabilities	0x00090081
ErrConfigureAutorun	0x00090083
ErrServiceStop	0x00090085
ErrServiceAlreadyStarted	0x00090086

Поле	Описание
ErrServiceCanceledPassword	0x00090089
ErrDemo	0x00090090
ErrRpc	0x00090091
ErrRpcIsBusy	0x00090092
ErrServiceUnknown	0x000900ff
ErrReportContext	0x000c0000
ErrPathIsDir	0x000c0001
ErrFileExists	0x000c0002
ErrPathDoesNotExist	0x000c0003
ErrPathIsNotAvailable	0x000c0004
ErrReportCreation	0x000c0005
ErrAuthContext	0x000d0000
ErrIncorrectPinOrPswd	0x000d0001
ErrLogonRequired	0x000d0002
ErrUpdateCredentials	0x000d0003
ErrContainerExists	0x000d0004
ErrWrongPin	0x000d0005
ErrDeviceLocked	0x000d0006
ErrUnsupportedMode	0x000d0007
ErrMemoryLack	0x000d0008
ErrNoDevice	0x000d0009
ErrKeysNotFoundOnDevice	0x000d000a
ErrDeviceKeysAreCorrupted	0x000d000b
ErrSystemResourceAccessDenied	0x000d000c
ErrAuthInternal	0x000d000d
ErrProfileVpnOn	0x00130001
ErrProfileUnbindFailed	0x00130002
ErrProfileBindFailed	0x00130003

Поле	Описание
ErrProfileAliasEmpty	0x00130004
ErrProfileAliasTooLong	0x00130005
ErrProfileAliasDuplicate	0x00130006
ErrNoProfileToSetAlias	0x00130007
ErrProfileAliasFailed	0x00130008
ErrProfileNotFound	0x00130009
ErrProfileIsActive	0x0013000a
ErrProfileRemoveFailed	0x0013000b
ErrNoProfileToDelete	0x0013000c
ErrInternal	0xffffffff

Таблица 6. VpnDstInfo

Поле	Описание
VpnDstStatus errors	Результат проверки файла дистрибутива ключей *.dst или *.ds5
time_t licenseExpired	Дата истечения лицензии
time_t licenseIssued	Дата выдачи *.dst или *.ds5
size_t logonDevice	Требуется токен для аутентификации
uint32_t nodeId	Идентификатор узла ViPNet (ViPNet Id)
char nodeName[NODE_NAME_MAX]	Название узла ViPNet
size_t password	Для расшифровки резервных ключей требуется пароль
enum VpnPrivilege privileges	Полномочия пользователя

Таблица 7. VpnDstStatus

Поле	Описание
size_t CheckDstLicenceExpired : 1	Файл *.dst или *.ds5 содержит истекшую лицензию
size_t CheckDstMobileTaskNotFound : 1	В файле *.dst или *.ds5 не найдена роль «VPN-клиент» для собственного узла
size_t CheckDstUnmergeError : 1	Системная ошибка распаковки дистрибутива ключей

Поле	Описание
<code>size_t CheckDstMalformedDst : 1</code>	Файл *.dst или *.ds5 повреждён и не может быть распакован
<code>size_t CheckDstTooManyWs : 1</code>	В дистрибутиве ключей слишком много связей узлов
<code>size_t CheckDstTooManyTunnels : 1</code>	В дистрибутиве ключей слишком много туннелей
<code>size_t CheckDstWrongPassword : 1</code>	Указан неверный пароль к дистрибутиву ключей
<code>size_t CheckDstWrongParams : 1</code>	Функции распаковки переданы неверные параметры
<code>size_t CheckDstTooBig : 1</code>	Превышен максимально допустимый размер файла *.dst или *.ds5
<code>size_t CheckDstNoAccessIp : 1</code>	Отсутствуют IP-адреса доступа для активного координатора
<code>size_t CannotReadDnsTxt : 1</code>	Невозможно прочесть или обработать dns.txt
<code>size_t DnsNoServerFound : 1</code>	Не заданы DNS-серверы
<code>size_t DnsTooManyServers : 1</code>	Превышено максимально допустимое количество записей DNS-серверов
<code>size_t DnsZoneNotFound : 1</code>	Не заданы доменные зоны
<code>size_t DnsTooManyZones : 1</code>	Превышено максимально допустимое количество записей доменных зон
<code>size_t DnsAutoStartRuleNotFound : 1</code>	Не задан параметр настройки автозапуска
<code>size_t DnsConnectDemandNotFound : 1</code>	Не задан параметр настройки Connect On Demand
<code>size_t ConfigNoSupported : 1</code>	Конфигурационный файл не поддерживается
<code>size_t DnsFileTooBig : 1</code>	Превышен максимально допустимый размер файла *.vpnconfig
<code>size_t ConfigureValueIsEmpty : 1</code>	В конфигурационном файле не найден ни один параметр
<code>size_t DnsConnectDemandNotAllowed : 1</code>	У пользователя недостаточно полномочий для включения настройки Connect On Demand
<code>size_t CheckAbTooManyDns : 1</code>	Превышено максимально допустимое количество доменных зон
<code>size_t UnexpectedAlgServiceName : 1</code>	Недопустимое имя службы в настройках alg
<code>size_t InvalidAlgServiceConfiguration : 1</code>	Невалидная конфигурация службы alg
<code>size_t DnsZoneIncludesForbiddenAddr : 1</code>	Доменные зоны включают в себя недопустимый адрес
<code>size_t CheckDNSNameNumber : 1</code>	Количество доменных имён координатора не должно быть более 5

Поле	Описание
size_t CheckDstPrivilegesLevelIsLow : 1	Недостаточный уровень полномочий для данного действия
size_t DnsInternalSystemError : 1	Внутренняя системная ошибка
size_t UnmergeDstBlocked : 1	Разворачивание *.dst или *.ds5 заблокировано из-за превышения количества попыток ввода пароля
size_t CheckDstAdPinBlocked : 1	Разворачивание *.dst или *.ds5 заблокировано из-за блокировки устройства аутентификации
size_t CheckDstAdPinFailed : 1	Неверный PIN-код
size_t CheckDstAdNotFound : 1	Указанное устройство аутентификации не найдено
Если значения всех полей 0, то проверка дистрибутива ключей прошла успешно	

Таблица 8. VpnHealthStatus

Поле	Описание
size_t cryptography	Результат проверки криптобиблиотеки CSP (1 — работоспособность нарушена)
size_t environment	Результат проверки целостности исполняемых файлов (1 — целостность нарушена)
size_t integrity	Результат проверки целостности конфигурационных файлов (1 — целостность нарушена)
size_t operating_environment	Результат проверки целостности окружения (1 — целостность нарушена)

Таблица 9. VpnStatus

Поле	Описание
bool isKeysInstalled	Проверка, установлены ли ключи ViPNet
bool isVpnEnabled	Проверка, включено ли VPN-соединение

Таблица 10. VpnNodeInfo

Поле	Описание
uint32_t id	Идентификатор узла ViPNet (ViPNet Id)
uint32_t ip	IPv4-адрес узла
char* name	Название узла ViPNet
uint32_t tasksMask	Битовая маска ролей и свойств узла

Таблица 11. *VpnNodeInfoExt*

Поле	Описание
<code>uint32_t coordinatorId</code>	Идентификатор активного координатора для узла (0 — неизвестен)
<code>bool directLink</code>	Связь с узлом напрямую без участия координатора
<code>bool haveKeys</code>	Имеются симметричные ключи обмена с узлом
<code>uint32_t id</code>	Идентификатор узла ViPNet (ViPNet Id)
<code>uint32_t ip</code>	IPv4-адрес узла
<code>const char* name</code>	Название узла ViPNet
<code>uint16_t networkId</code>	Идентификатор сети ViPNet
<code>VpnNodeTasksMask tasks</code>	Битовая маска ролей и свойств узла
<code>uint16_t tcpTunnelPort</code>	Порт доступа для TCP-туннеля на координаторе (0 — неизвестен)
<pre>struct { uint32_t endIp uint32_t startIp } * tunnels</pre>	Массив диапазонов IPv4-адресов, туннелируемых этим узлом (если узел — координатор)
<code>size_t tunnelsSize</code>	Размер массива диапазонов туннелируемых IPv4-адресов
<pre>struct { uint32_t endIp uint32_t startIp } * tunnelsVirtual</pre>	Массив диапазонов виртуальных адресов, туннелируемых этим узлом (если узел — координатор)
<code>size_t tunnelsVirtualSize</code>	Размер массива диапазонов туннелируемых виртуальных адресов

Таблица 12. *VpnUserInfo*

Поле	Описание
<code>uint32_t id</code>	Идентификатор (ID) пользователя
<code>char* name</code>	Имя пользователя.
<code>uint32_t NumNodes</code>	Количество связанных с пользователем узлов
<code>uint32_t relatedNodes</code>	Список связанных с пользователем узлов

Таблица 13. *VpnUltimateStatus*

Поле	Описание
<code>size_t autorunDesktop</code>	Флаг настройки автозапуска при авторизации в графическом интерфейсе. 1 — включена; 0 — отключена
<code>size_t autorunShell</code>	Флаг настройки автозапуска при авторизации в консоли. 1 — включена; 0 — отключена
<code>char cipher[16]</code>	Наименование криптографического алгоритма, на котором работает узел
<code>size_t configUpdated</code>	Флаг валидности конфигурационных файлов и базы данных после обновления ПО. 1 — валидны; 0 — невалидны
<code>uint32_t coordinatorId</code>	Идентификатор активного координатора
<code>uint32_t coordinatorIp</code>	IPv4-адрес активного координатора
<code>char coordinatorName[NODE_NAME_MAX]</code>	Название активного координатора
<code>int dnsState</code>	Текущий статус подсистемы DNS. 0 — включена; 1 — выключена; 2 — выключена (отсутствует утилита iptables); 3 — выключена (выключен loopback-интерфейс)
<code>int faultTolerance</code>	Количество попыток восстановления. 0 — настройка отключена; 1 — настройка включена, количество не ограничено
<code>size_t integrityConfigs</code>	Результат проверки контрольных сумм конфигурационных файлов. 1 — целостность не нарушена; 0 — целостность нарушена
<code>size_t integrityHostlinks</code>	Результат проверки контрольных сумм справочников. 1 — целостность не нарушена; 0 — целостность нарушена

Поле	Описание
<code>size_t integrityLicense</code>	Результат проверки контрольных сумм лицензии. 1 — целостность не нарушена; 0 — целостность нарушена
<code>time_t keysIssued</code>	Дата и время выдачи дистрибутива ключей
<code>int keysStatus</code>	Текущий статус ключевой конфигурации. 0 — нет; 1 — проверена; 2 — недействительна
<code>time_t licenseExpired</code>	Дата и время истечения лицензии
<code>time_t licenseIssued</code>	Дата и время выпуска лицензии
<code>int licenseStatus</code>	Статус проверки лицензии. 0 — действительна; 1 — повреждена; 2 — истекла; 3 — ошибочная дата; 4 — отсутствует; 5-12 — действительна, но истекает через 1-7 дней; 13 — неограниченная
<code>size_t loginCounterAdmin</code>	Флаг аутентификаций администратора. 1 — разрешена; 0 — заблокирована
<code>size_t loginCounterUser</code>	Флаг аутентификаций пользователя (по паролю). 1 — разрешена; 0 — заблокирована
<code>size_t loginType</code>	Тип аутентификации. 0 — не определён 1 — пароль; 2 — пароль на устройстве; 4 — ключ на устройстве

Поле	Описание
<code>int logLevel</code>	Уровень детализации событий в журнале. 0 — критические ошибки; 1 — ошибки; 2 — уведомления; 3 — информационные сообщения; 4 — отладочная информация; 5, 6, 10, 11 — детальная информация
<code>uint16_t networkId</code>	Идентификатор сети ViPNet
<code>char networkName[NET_NAME_MAX]</code>	Название сети ViPNet
<code>uint32_t nodeId</code>	Идентификатор узла ViPNet (ViPNet Id)
<code>char nodeName[NODE_NAME_MAX]</code>	Название узла ViPNet
<code>pid_t pid</code>	PID службы vipnetclient
<code>enum VpnPrivilege privileges</code>	Полномочия пользователя
<code>VpnNodeTasksExt tasks</code>	Роли узла
<code>char user[LOGIN_NAME_MAX]</code>	Имя пользователя ОС, для которого запущена служба vipnetclient
<code>char version[20]</code>	Версия ПО ViPNet Client 5 (MJR.MNR.SUB-BUILD)
<code>int vpnStatus</code>	Текущий режим VPN-соединения. 0 — выключено; 1 — включено; 2 — рассинхронизация с активным координатором; 3 — соединение проверено
<code>time_t vpnTime</code>	Время сети ViPNet
<code>char workdir[PATH_MAX]</code>	Путь к файлам текущей конфигурации

Таблица 14. *VpnTokenInfo*

Поле	Описание
<code>uint32_t isFinal : 1</code>	Осталась последняя попытка аутентификации с помощью токена
<code>uint32_t isLocked : 1</code>	Токен заблокирован из-за превышения количества попыток аутентификации

Поле	Описание
<code>uint32_t isLow : 1</code>	Была как минимум одна неудачная попытка аутентификации с помощью токена
<code>const char* label</code>	Метка устройства (устанавливается при инициализации)
<code>const char* manufacturer</code>	Идентификатор (ID) производителя устройства
<code>uint32_t maxPinLen</code>	Максимальная длина PIN-кода
<code>uint32_t minPinLen</code>	Минимальная длина PIN-кода
<code>const char* model</code>	Модель устройства
<code>const char* serialNumber</code>	Серийный номер устройства
<code>uint32_t hasPin : 1</code>	PIN-код пользователя установлен
<code>uint32_t isInitialized : 1</code>	Токен проинициализирован
<code>uint32_t reserved : 25</code>	Поле не используется
<code>uint32_t writeProtected : 1</code>	Токен защищён от записи
<code>uint32_t isDefault : 1</code>	Этот токен требуется для авторизации в сети ViPNet

Таблица 15. *VpnNodeTasksMask*

Поле	Описание
<code>uint32_t VpnNodeTasksMask</code>	Битовая маска ролей и свойств узла из <i>VpnNodeTask</i>

Таблица 16. *VpnTokenHandle*

Поле	Описание
<code>void* VpnTokenHandle</code>	Дескриптор токена

Таблица 17. *VpnEventHandlers*

Поле
<code>VIPNET_EVENT(vipnetDaemonStarted, void)</code>
<code>VIPNET_EVENT(vipnetDaemonStopped, void)</code>
<code>VIPNET_EVENT(vipnetStarted, void)</code>
<code>VIPNET_EVENT(vipnetStopped, void)</code>
<code>VIPNET_EVENT(vipnetIntegrityCheck, int)</code>
<code>VIPNET_EVENT(vipnetUpdateReceived, void)</code>

Поле
VIPNET_EVENT(vipnetConnectionSuccessful, int, bool)
VIPNET_EVENT(vipnetConnectionFailed, int, bool)
VIPNET_EVENT(vipnetLicenseCheck, unsigned)
VIPNET_EVENT(vipnetKeysInstalled, bool)
VIPNET_EVENT(vipnetNewDstRequired, void)
VIPNET_EVENT(vipnetPasswordRequired, void)
VIPNET_EVENT(vipnetServerTimeDiffDetected, time_t)
VIPNET_EVENT(vipnetNetworkStatusChanged, unsigned)
VIPNET_EVENT(vipnetCryptographyCheck, bool)
VIPNET_EVENT(vipnetReportSaved)
VIPNET_EVENT(vipnetChangeLogonType, void)
VIPNET_EVENT(vipnetLogLevelChanged, int)
VIPNET_EVENT(vipnetBeginStarting, void)
VIPNET_EVENT(vipnetBeginStopping, void)
VIPNET_EVENT(vipnetDefaultTokenAttached, void)
VIPNET_EVENT(vipnetDefaultTokenDetached, void)
VIPNET_EVENT(vipnetUserTrafficStatusChanged, unsigned)
VIPNET_EVENT(vipnetSettingsChanged, void)
VIPNET_EVENT(vipnetMfAuthStarted, void)
VIPNET_EVENT(vipnetMfAuthStartFailed, void)
VIPNET_EVENT(vipnetMfAuthStatusChanged, int, int)
VIPNET_EVENT(vipnetEnvironmentCheck, bool)
VIPNET_EVENT(vipnetDaemonStartFailed, void)
VIPNET_EVENT(vipnetChangeUserPassword, void)
VIPNET_EVENT(vipnetPasswordAttemptsOver, void)
VIPNET_EVENT(vipnetProfileBound, bool)
VIPNET_EVENT(vipnetProfileDelete, void)

Таблица 18. VpnEvent

Поле
Event_VpnServiceStarted
Event_VpnServiceStopped
Event_KeysInstalled
Event_KeysUninstalled
Event_VpnStarted
Event_VpnStopped
Event_NodesUpdated
Event_VpnBeginStarting
Event_VpnBeginStopping

Таблица 19. VpnNodeTask

Поле	Описание
Task_Undefined	0x0
Task_TrafficProtection	0x1
Task_Coordinator	0x2
Task_ActiveCoordinator	0x4
Task_Client	0x8
Task_MobileClient	0x10
Task_LinuxClient	0x20
Task_Connect	0x40
Task_QssServer	0x80
Task_QssPoint	0x100
Task_QssPhone	0x200
Task_GroupChatServer	0x400
Task_PushServer	0x800
Task_ConnectHub	0x1000
Task_ConnectFileExchange	0x2000
Task_NativeCoordinator	0x4000
Task_BusinessMail	0x8000

Таблица 20. *VpnNodeTasksExt*

Поле	Описание
size_t TASK_MAIL : 1	Деловая почта
size_t TASK_NCC : 1	Network Control Center (ЦУС)
size_t TASK_POLICY_MANAGER : 1	Policy Manager (Центр управления политиками безопасности)
size_t TASK_MONITORING_SYSTEM : 1	Система мониторинга
size_t TASK_MFTP : 1	MFTP
size_t TASK_IP : 1	Защита трафика
size_t TASK_IP_SERVER : 1	Сервер IP-адресов
size_t TASK_IP_PROXY : 1	Сервер IP-адресов
size_t TASK_PUBLIC_INTERNET : 1	Сервер IP-адресов с функцией сервера Открытого Интернета
size_t TASK_CRYPTOSERVICE : 1	CryptoService
size_t TASK_LINUX_FAILOVER : 1	Failover
size_t TASK_VIPNET_SDK : 1	ViPNet SDK
size_t TASK_LINUX_WEB_GATE : 1	Web-шлюз
size_t TASK_VIPNET_U : 1	Coordinator KB1000
size_t TASK_TERMINAL_NAVIGATOR : 1	Терминал Навигатор
size_t TASK_TERMINAL_SPECTR : 1	Терминал Спектр
size_t TASK_TERMINAL_AREAL : 1	Терминал Ареал-Сервис
size_t TASK_ISU_MAGISTRAL : 1	ИСУ Магистраль
size_t TASK_ISU_NAVIGATOR : 1	ИСУ Навигатор
size_t TASK_ISU_SPECTR : 1	ИСУ Спектр
size_t TASK_MONITORING : 1	АРМ мониторинга
size_t TASK_ISU_VIDEOTON : 1	ИСУ Видеотон
size_t TASK_TERMINAL_UFS : 1	Блок сопряжения УФС
size_t TASK_SGA_SERVER : 1	Сервер SGA
size_t TASK_SGA_CLIENT : 1	Клиент SGA
size_t TASK_TLS_SERVER : 1	Сервер TLS
size_t TASK_HW100_A : 1	Coordinator HW100-A
size_t TASK_HW100_B : 1	Coordinator HW100-B
size_t TASK_HW100_C : 1	Coordinator HW100-C

Поле	Описание
size_t TASK_HW1000 : 1	Coordinator HW1000
size_t TASK_HW2000 : 1	Coordinator HW2000
size_t TASK_SDK_LINUX : 1	CryptoService Linux
size_t TASK_TERMINAL : 1	ViPNet Terminal
size_t TASK_HWVPNM : 1	Coordinator HW-VPNM
size_t TASK_MONITORING_AGENT : 1	Агент мониторинга СБ РФ
size_t TASK_CLIENT_MOBILE : 1	VPN-клиент для мобильных устройств
size_t TASK_VSG : 1	ViPNet Security Gateway (Coordinator HW/VA)
size_t TASK_HWVA : 1	Coordinator HW-VA
size_t TASK_HW10 : 1	Coordinator HW10
size_t TASK_HWMCM : 1	Coordinator HW-MCM
size_t TASK_HW100_D : 1	Coordinator HW100D
size_t TASK_CONNECT : 1	Connect
size_t TASK_HW50_A : 1	Coordinator HW50A
size_t TASK_HW50_B : 1	Coordinator HW50B
size_t TASK_HW500 : 1	Coordinator HW500
size_t TASK_HW1000_C : 1	Coordinator HW1000C
size_t TASK_HW1000_D : 1	Coordinator HW1000D
size_t TASK_HW5000 : 1	Coordinator HW5000
size_t TASK_IG10 : 1	Coordinator IG10
size_t TASK_SHW1000 : 1	StateWatcher1000
size_t TASK_SHW2000 : 1	StateWatcher2000
size_t TASK_SHWVA : 1	StateWatcher VA
size_t TASK_IG10_A0 : 1	Coordinator IG10A0
size_t TASK_IG10_A1 : 1	Coordinator IG10A1
size_t TASK_IG10_B0 : 1	Coordinator G10B0
size_t TASK_IG10_B1 : 1	Coordinator IG10B1
size_t TASK_LINUX_CLIENT : 1	Client for Linux
size_t TASK_CONNECT_HUB : 1	VPN-клиент для сервера Connect
size_t TASK_CONNECT_FILE_EXCHANGE : 1	Connect для обмена файлами

Таблица 21. *VpnPrivilege*

Поле	Описание
VpnPrivilegeUnknown	-1
VpnPrivilegeMin	0
VpnPrivilegeMiddle	1
VpnPrivilegeMax	2
VpnPrivilegeSpecial	3

Таблица 22. *VpnNodeTasksMaskAll*

Поле	Описание
VPN_NODE_TASKS_MASK_ALL	UINT32_MAX

Таблица 23. *VpnUserTrafficBlockStatus*

Поле	Описание
UserTrafficNotBlocked	0
UserTrafficBlocked	1

Таблица 24. *VpnUserTrafficBlockInfo*

Поле	Описание
enum VpnUserTrafficBlockStatus status	Статус блокировки пользовательского трафика
const char* processName	Имя процесса, изменившего текущий статус блокировки пользовательского трафика

Таблица 25. *VpnDstInfoV10*

Поле	Описание
char nodeName[NODE_NAME_MAX]	Имя узла
uint32_t nodeId	Идентификатор узла
enum VpnPrivilege privileges	Уровень полномочий
uint64_t licenseExpired	Дата истечения лицензии
uint64_t licenseIssued	Дата выпуска дистрибутива ключей
enum VpnKeySystemVersion ksVersion	Поколение продукта
uint32_t logonDevice : 1	Задана ли аутентификация с помощью внешнего устройства

Поле	Описание
<code>uint32_t password : 1</code>	Задана ли аутентификация по паролю

Таблица 26. *VpnDstCheckListInfoV10*

Поле	Описание
<code>VpnDstCheckCodeV10* list</code>	Список ошибок при установке ключей
<code>uint32_t size</code>	Размер списка ошибок при установке ключей

Таблица 27. *VpnDstCheckCodeV10*

Поле	Описание
<code>DstCheckCode_LicenseExpired = 0</code>	В дистрибутиве ключей истекла лицензия
<code>DstCheckCode_MobileTaskNotFound = 1</code>	В дистрибутиве ключей не задана роль для мобильного клиента
<code>DstCheckCode_UnmergeError = 2</code>	Системная ошибка
<code>DstCheckCode_Malformed = 3</code>	Дистрибутив ключей скомпрометирован
<code>DstCheckCode_TooManyWs = 4</code>	Превышено количество связей в справочнике
<code>DstCheckCode_TooManyTunnels = 5</code>	Превышено количество туннелей в справочнике
<code>DstCheckCode_WrongPassword = 6</code>	Неверный пароль
<code>DstCheckCode_WrongParams = 7</code>	В функцию распаковки переданы недопустимые параметры
<code>DstCheckCode_TooBig = 8</code>	Превышен размер дистрибутива ключей
<code>DstCheckCode_NoAccessIp = 9</code>	Отсутствует IP-адрес доступа активного координатора
<code>DstCheckCode_CannotReadDnsTxt = 10</code>	Невозможно прочесть или обработать файл <code>dns.txt</code>
<code>DstCheckCode_DnsNoServerFound = 11</code>	Отсутствуют DNS-серверы
<code>DstCheckCode_DnsTooManyServers = 12</code>	Превышено количество DNS-серверов
<code>DstCheckCode_DnsZoneNotFound = 13</code>	Отсутствуют DNS-зоны
<code>DstCheckCode_DnsTooManyZones = 14</code>	Превышено количество DNS-зон
<code>DstCheckCode_DnsAutoStartRuleNotFound = 15</code>	Не задан параметр автостарта
<code>DstCheckCode_DnsConnectDemandNotFound = 16</code>	Не задан параметр настройки On Demand Connect
<code>DstCheckCode_ConfigNotSupported = 17</code>	Конфигурационный файл повреждён
<code>DstCheckCode_DnsFileTooBig = 18</code>	Превышен размер конфигурационного файла

Поле	Описание
<code>DstCheckCode_ConfigureValueIsEmpty = 19</code>	В конфигурационном файле отсутствуют параметры
<code>DstCheckCode_DnsConnectDemandNotAllowed = 20</code>	У пользователя недостаточно полномочий для включения настройки On Demand Connect
<code>DstCheckCode_AbTooManyDns = 21</code>	Превышено количество DNS-серверов
<code>DstCheckCode_UnexpectedAlgServiceName = 22</code>	Неверное имя службы в секции [ALG]
<code>DstCheckCode_InvalidAlgServiceConfiguration = 23</code>	Неподдерживаемый протокол в секции [ALG]
<code>DstCheckCode_DnsZoneIncludesForbiddenAddr = 24</code>	Неверные адреса в DNS-зонах
<code>DstCheckCode_DnsTooManyNames = 25</code>	Количество доменных имён координатора превышает 5
<code>DstCheckCode_LowPrivilegesLevel = 26</code>	Недостаточный уровень полномочий для завершения операции
<code>DstCheckCode_DnsInternalSystemError = 27</code>	Системная ошибка
<code>DstCheckCode_UnmergeBlocked = 28</code>	Превышено количество попыток ввода пароля
<code>DstCheckCode_AdPinBlocked = 29</code>	Устройство повреждено или заблокировано
<code>DstCheckCode_AdPinFailed = 30</code>	Неверный PIN-код
<code>DstCheckCode_AdNotFound = 31</code>	Внешнее устройство не найдено
<code>DstCheckCode_InvalidCustomYaml = 32</code>	Ошибка в расширенных настройках клиента
<code>DstCheckCode_DnsZoneTooLong = 33</code>	Слишком длинные имена DNS-зон
<code>DstCheckCode_LicenseNotSupported = 34</code>	Неподходящая лицензия
<code>DstCheckCode_IntegrityViolation = 35</code>	Нарушена целостность справочников или ключей
<code>DstCheckCode_CryptographyNotSupported = 36</code>	Не поддерживается криптография дистрибутива ключей
<code>DstCheckCode_LocalTimeWrong = 37</code>	Время создания дистрибутива ключей больше времени на устройстве
<code>DstCheckCode_TooManyAuthServers = 38</code>	В связях узла должен быть только один узел с именем Client Auth Server
<code>DstCheckCode_ProfilesKeysDuplicate = 39</code>	Такой ID узла уже существует
<code>DstCheckCode_LicenseIncorrectVersion = 40</code>	Версия продукта отличается от версии в лицензии
<code>DstCheckCode_ProfilesLimit = 41</code>	Создано максимальное количество профилей
<code>DstCheckCode_KeyVersionNotSupportedKS5 = 42</code>	В эту версию ViPNet Client 5 можно установить только ключи *.ds5

Поле	Описание
<code>DstCheckCode_WrongStateVpnEnabled = 43</code>	Нужно выключить VPN-соединение, чтобы добавить профиль

Таблица 28. *VpnKeySystemVersion*

Поле	Описание
<code>KeySystemInvalid = 0</code>	Поколение продукта не определено
<code>KeySystem4 = 4</code>	4 поколение
<code>KeySystem5 = 5</code>	5 поколение

Таблица 29. *VpnLoginAvailabilityInfo*

Поле	Описание
<code>uint64_t retryTime</code>	Системное время в секундах, через которое разрешён вход
<code>bool isAvailable</code>	Разрешён ли вход сейчас

Таблица 30. *VpnMfAuthChallengeTipsList*

Поле	Описание
<code>VpnMfAuthChallengeTipInfo* list</code>	Список доступных способов многофакторной аутентификации
<code>uint32_t size</code>	Размер списка доступных способов многофакторной аутентификации

Таблица 31. *VpnMfAuthChallengeTipInfo*

Поле	Описание
<code>const char* data</code>	Данные списка доступных способов многофакторной аутентификации

Таблица 32. *VpnMfAuthRequestStatus*

Поле	Описание
<code>MFRequestStatus_Idle = 0</code>	Запрос не сделан
<code>MFRequestStatus_Pending = 1</code>	Запрос отправлен, ожидание ответа
<code>MFRequestStatus_Continue = 2</code>	Получен ответ «Продолжить»

Поле	Описание
MFARquestStatus_Success = 3	Получен ответ «Успех»
MFARquestStatus_Fail = 4	Получен ответ «Не удалось»
MFARquestStatus_Timeout = 5	Истекло время запроса
MFARquestStatus_Cancelled = 6	Запрос отменен
MFARquestStatus_Error = 7	Ошибка запроса
MFARquestStatus_Unknown = 8	Статус запроса неизвестен.

Таблица 33. VpnMfAuthStatus

Поле	Описание
MFAStatus_NotAuthenticated = 0	Требуется аутентификация
MFAStatus_Authenticated = 1	Аутентификация выполнена или не требуется
MFAStatus_PendingResponse = 2	Ожидание ответа сервера
MFAStatus_ChallengeLogin = 3	Ожидание входа
MFAStatus_ChallengeSelect = 4	Ожидание выбора способа аутентификации
MFAStatus_ChallengeInput = 5	Ожидание ввода второго фактора
MFAStatus_ChallengeMessage = 6	Получение информации о втором факторе
MFAStatus_NotAuthenticatedCooldown = 7	Ожидание после неудачной попытки
MFAStatus_NotInitialized = 8	Не удалось запустить модуль многофакторной аутентификации
MFAStatus_Invalid = ~0	Неверный статус аутентификации

Таблица 34. VpnUserTrafficBlockReason

Поле	Описание
UserTrafficBlockReason_License = 0	Лицензия недействительна
UserTrafficBlockReason_SessionLock = 1	Трафик заблокирован из-за завершения пользовательской сессии
UserTrafficBlockReason_SdkCommand = 2	Трафик заблокирован по команде VPN SDK
UserTrafficBlockReason_MfAuth = 3	Не пройдена многофакторная аутентификация
UserTrafficBlockReason_Feature_MfAuth = 4	Не пройдена многофакторная аутентификация
UserTrafficBlockReason_Feature_Ztna = 5	Трафик заблокирован из-за ViPNet EPP
UserTrafficBlockReason_Feature_Others = 6	Трафик заблокирован из-за настроек в справочнике node.info

Поле	Описание
UserTrafficBlockReason_Environment = 7	Нарушена целостность среды функционирования
UserTrafficBlockReason_Count = 8	Количество причин блокировки трафика

Таблица 35. VpnDmsStatus

Поле	Описание
None = 0	Неизвестен или не задан
NeedUpdate = 1	Требуется обновление
ProcessUpdating = 2	Обновление в процессе
Stable = 3	Успешно обновлён
UpdateFailed = 4	Не удалось обновить

Таблица 36. ProfileInfo

Поле	Описание
uint32_t nodeId	Идентификатор узла
time_t timestamp	Время добавления профиля
uint8_t privilege	Уровень полномочий
uint8_t keysystem	Поколение продукта
uint16_t networkId	Идентификатор сети
bool isActive	Профиль активен
bool useDevice	Используется аутентификация с помощью внешнего устройства
const char* profileName	Имя профиля
const char* networkName	Имя сети
void* auxiliaryData	Зарезервировано

Таблица 37. VpnNodeDeviceInfo

Поле	Описание
char deviceName[NODE_NAME_MAX]	Имя устройства
char osVersion[OS_VERSION_MAX]	Версия ОС
char appVersion[APP_VERSION_MAX]	Версия ViPNet Client 5

Таблица 38. *VpnApiVersion*

Поле	Описание
VPN_API_VERSION	14

Cipher API

Таблица 39. *ItcsCipherApi*

Функция	Описание
<code>ItcsCipherApi* GetCipherApi(void)</code>	Получение указателя на структуру Cipher API с проинициализированными указателями на функции
<code>uint32_t GetCipherApiVersion(void)</code>	Получение номера текущей версии Cipher API
<code>CipherApiReturnCode (*EncryptBlob) (uint32_t targetNodeId, uint8_t* in, size_t inSize, uint8_t* out, size_t* outSize)</code>	Зашифрование данных для узла. • [in] <code>targetNodeId</code> — ID узла, для которого производится шифрование; • [in] <code>in</code> — буфер с входными данными; • [in] <code>inSize</code> — размер буфера с входными данными; • [out] <code>out</code> — буфер с зашифрованными данными; • [in,out] <code>outSize</code> — размер буфера с зашифрованными данными. Размер входного буфера должен быть не меньше 16 байт, выходного — не меньше, чем размер входного плюс 12 байт

Функция	Описание
<pre> CipherApiReturnCode (*EncryptBlobEx) (uint32_t cipherSuite, uint32_t targetNodeId, uint8_t* in, size_t inSize, uint8_t* out, size_t* outSize) </pre>	Зашифрование данных для узла с использованием указанного набора криптографических алгоритмов. • [in] cipherSuite — набор криптографических алгоритмов; • [in] targetNodeId — ID узла, для которого производится шифрование; • [in] in — буфер с входными данными; • [in] inSize — размер буфера с входными данными; • [out] out — буфер с зашифрованными данными; • [in,out] outSize — размер буфера с зашифрованными данными. Размер входного буфера должен быть не меньше 16 байт, выходного — не меньше, чем размер входного плюс 16 байт
<pre> CipherApiReturnCode (*DecryptBlob) (uint32_t sourceNodeId, uint8_t* in, size_t inSize, uint8_t* out, size_t* outSize) </pre>	Расшифрование данных от узла. • [in] sourceNodeId — ID узла, для которого производится расшифрование; • [in] in — буфер с входными данными; • [in] inSize — размер буфера с входными данными; • [out] out — буфер с расшифрованными данными; • [in,out] outSize — размер буфера с расшифрованными данными. Размер входного буфера должен быть не меньше 28 байт, выходного — не меньше, чем размер входного минус 12 байт

Функция	Описание
<pre>CipherApiReturnCode (*DecryptBlobEx) (uint32_t cipherSuite, uint32_t sourceNodeId, uint8_t *in, size_t inSize, uint8_t *out, size_t *outSize)</pre>	Расшифрование данных от узла с использованием указанного набора криптографических алгоритмов. • [in] cipherSuite — набор криптографических алгоритмов; • [in] sourceNodeId — ID узла, для которого производится расшифрование; • [in] in — буфер с входными данными; • [in] inSize — размер буфера с входными данными; • [out] out — буфер с расшифрованными данными; • [in,out] outSize — размер буфера с расшифрованными данными. Размер входного буфера должен быть не меньше 28 байт, выходного — не меньше, чем размер входного минус 16 байт
<pre>CipherApiReturnCode (*IsCipherSuiteSupported) (uint32_t cipherSuite)</pre>	Проверка, поддерживается ли указанный набор криптографических алгоритмов. • [in] cipherSuite — набор криптографических алгоритмов
<pre>CipherApiReturnCode (*GetEncryptedBufferMinSize) (uint32_t nodeId, uint32_t inBufferSize, uint32_t* outBufferSize)</pre>	Получение минимального размера зашифрованного буфера для указанного узла. • [in] nodeId — ID узла; • [in] inBufferSize — размер буфера с входными данными; • [in,out] outBufferSize — минимальный размер буфера с зашифрованными данными
<pre>CipherApiReturnCode (*GetDecryptedBufferMinSize) (uint32_t nodeId, uint32_t inBufferSize, uint32_t* outBufferSize)</pre>	Возвращает минимальный размер расшифрованного буфера для указанного узла. • [in] nodeId — ID узла; • [in] inBufferSize — размер буфера с входными данными; • [in,out] outBufferSize — минимальный размер буфера с расшифрованными данными

Таблица 40. *CipherApiReturnCode*

Поле	Описание
<code>uint32_t code</code>	Код возврата функций (0 — успешное завершение)
<code>const char* message</code>	Сообщение об ошибке (пустое при успешном завершении). Не требует освобождения памяти. Сообщение хранится до следующего вызова команды

Таблица 41. *CipherSuite*

Поле	Описание
<code>InvalidCipherSuite</code>	<code>0xcd00dead</code>

Таблица 42. *CryptoResult*

Поле	Описание
<code>0</code>	<code>SUCCESS</code>
<code>0x000A0001</code>	<code>BUFFER_TOO_SMALL</code>
<code>0x000A0002</code>	<code>BUFFER_IS_NULLPTR</code>
<code>0x000A0003</code>	<code>ENCRYPT_ERROR</code>
<code>0x000A0004</code>	<code>DECRYPT_ERROR</code>
<code>0x000A0005</code>	<code>KEY_NOT_FOUND_ERROR</code>
<code>0x000A0006</code>	<code>BUFFER_TOO_LARGE</code>
<code>0x000A0007</code>	<code>CIPHER_SUITE_NOT_AVAILABLE</code>
<code>0x000A000A</code>	<code>NOT_AVAILABLE</code>
<code>0xFFFFFFFF</code>	<code>INTERNAL_ERROR</code>

Таблица 43. *CipherApiVersion*

Поле	Описание
<code>CIPHER_API_VERSION</code>	<code>3</code>

Windows API

Таблица 44. WindowsVpnAPI

Функция	Описание
<code>WindowsVpnApi* GetWindowsVpnApi(void)</code>	Получение указателя на структуру WindowsVpnApi с проинициализированными указателями на функции
<code>uint32_t GetWindowsVpnApiVersion(void)</code>	Получение номера текущей версии WindowsVpnApi.
<code>WindowsApiReturnCode (*StorageUpgrade)(const char* oldLocation, const char* newLocation, bool* successful)</code>	Миграция профилей из текущего расположения в новое. • [in] <code>oldLocation</code> — текущее расположение профилей; • [in] <code>newLocation</code> — новое расположение профилей; • [out] <code>successful</code> — результат миграции

Таблица 45. WindowsApiReturnCode

Поле	Описание
<code>uint32_t code</code>	Код возврата функций (0 — успешное завершение)
<code>const char* message</code>	Сообщение об ошибке (пустое при успешном завершении). Не требует освобождения памяти. Сообщение хранится до следующего вызова команды

Таблица 46. WindowsVpnApiVersion

Поле	Описание
<code>WINDOWS_VPN_API_VERSION</code>	1

Примеры

```
// Подключение заголовочных файлов
#include <vipnet/vpn_api.h>
#include <vipnet/cipher_api.h>

int main()
{
    // Проверка соответствия версии SDK, с которой собирается программа, и версии SDK,
    // поставляемой с программой
    if( VPN_API_VERSION != GetVpnApiVersion() )
    {
        return 1;
    }

    if( CIPHER_API_VERSION != GetCipherApiVersion() )
    {
        return 1;
    }

    // Получение структуры с указателями на функции SDK
    ItcsVpnApi* api = GetVpnApi();
    ItcsCipherApi* capi = GetCipherApi();

    // Установка папки обмена
    // C:\users\user замените на путь к профилю того пользователя, который развернул
    // ключи
    api->setExchangeFolder(
        L"C:\\Users\\user\\AppData\\Local\\Publishers\\f7q4ak84jrfw6\\group.com.vipnet.exchange" );

    // Получение статуса работы клиента
    VpnStatus status;
    VpnApiReturnCode code = api->getVpnStatus( &status );
    if( code.code != 0 )
    {
        return 2;
    }

    // Зашифрование строки на ключах узла с id 0x1111
    uint8_t out[1000] = { 0 };
```

```

uint8_t in[] = "to be encrypted";
size_t outLen = 1000;
capi->encryptBlob( 0x1111, in, sizeof( in ), out, &outLen );

// Получение списка узлов
size_t size = 0;
uint32_t *nodeIds;
VpnApiReturnCode code = api->getNodesIds( &nodeIds, &size );
if( code.code != 0 || !nodeIds || !size )
{
    return 3;
}
api->releaseVpnNodeIds( nodeIds, size );
}

```

Обратная связь

Контактная информация

- Единый многоканальный телефон:
+7 (495) 737-6192,
8 (800) 250-0-260 — бесплатный звонок из России (кроме Москвы).
- Служба поддержки: hotline@infotecs.ru.
Форма для обращения в службу поддержки через сайт.
Телеграм-канал поддержки: t.me/vhd21
Телефон для клиентов с расширенной поддержкой: +7 (495) 737-6196.
- Отдел продаж: soft@infotecs.ru.

Дополнительная информация на сайте ИнфоТеКС

- [О продуктах ViPNet.](#)
- [О решениях ViPNet.](#)
- [Часто задаваемые вопросы.](#)
- [Форум пользователей продуктов ViPNet.](#)

Если вы обнаружили уязвимости в продуктах компании, сообщите о них по адресу security-notifications@infotecs.ru. Распространение информации об уязвимостях продуктов компании ИнфоТеКС регулируется [политикой ответственного разглашения](#).