



ViPNet Client 5 for Linux SDK

Справочник разработчика

Версия продукта: 5.1.4

© АО «ИнфоТеКС», 2025

ФРКЕ.00380-01 33 01

Версия продукта 5.1.4

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
ItcsVpnApi.....	6
ItcsCipherApi	44
ItcsBusinessMailApi.....	47
ItcsEncryptionApi	52
Примеры	54
Обратная связь.....	55

Введение

О документе

Данный документ содержит инструкции по использованию SDK ViPNet Client 5 for Linux (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 защищает IP-трафик на компьютерах с ОС Linux путём шифрования IP-пакетов.

С помощью ViPNet Client 5 вы можете подключаться к защищённым узлам или узлам, которые туннелируются координаторами, по защищённым каналам и получать доступ к размещённым на этих узлах ресурсам: корпоративным веб-порталам, электронной почте, системе IP-телефонии и другим корпоративным сервисам.

Начало работы с ViPNet Client 5 SDK

1 Установите ViPNet Client 5. При этом будут установлены следующие файлы ViPNet Client 5 SDK:

- Заголовочные файлы:

```
/usr/include/vipnet/business_mail_api.h,  
/usr/include/vipnet/cipher_api.h,  
/usr/include/vipnet/encryption_api.h,  
/usr/include/vipnet/errors.h,  
/usr/include/vipnet/vpn_api.h,  
/usr/include/vipnet/events.h,  
/usr/include/vipnet/vpn_api_types.h,  
/usr/include/vipnet/vpn_api_version.h.
```

- Библиотека /usr/lib/vipnet/libvpn_api.so.

- Контрольные суммы этих файлов.

2 В зависимости от того, какими функциями SDK вы планируете пользоваться, подключите один или несколько заголовочных файлов в ваши исходные коды:

```
#include <vipnet/vpn_api.h> — основные функции для работы с VPN-соединением.
```

```
#include <vipnet/cipher_api.h> — функции для шифрования данных.
```

```
#include <vipnet/encryption_api.h> — функции для шифрования данных.
```

```
#include <vipnet/business_mail_api.h> — функции для работы с конвертами программы  
ViPNet Деловая почта.
```

3 При сборке проекта выполните линковку с библиотекой libvpn_api.so.

4 Проверьте, что версия SDK, с которой собирается программа ViPNet Client 5, соответствует версии SDK, поставляемой вместе с программой ViPNet Client 5 (только для тех функций, которыми планируете пользоваться):

```
if ( VPN_API_VERSION != GetVpnApiVersion() )  
{  
    return "Version mismatch";  
}  
  
if ( CIPHER_API_VERSION != GetCipherApiVersion() )  
{  
    return "Version mismatch";  
}  
  
if ( BUSINESS_MAIL_API_VERSION != GetBusinessMailApiVersion() )  
{
```

```

        return "Version mismatch";
    }

```

- 5 Получите структуры с указателями на функции SDK (только для тех функций, которыми планируете пользоваться):

```

ItcsVpnApi* api = GetVpnApi();
ItcsCipherApi* capi = GetCipherApi();
ItcsEncryptionApi* encapi = GetEncryptionApi();
ItcsBusinessMailApi* bmapi = GetBusinessMailApi();

```

Если в результате будут возвращены ненулевые указатели, то вы сможете вызывать [функции SDK](#).

Функции SDK

ItcsVpnApi

Таблица 1. *ItcsVpnApi*

Функция	Описание
<code>ItcsVpnApi* GetVpnApi()</code>	Получение указателя на структуру VPN API с проинициализированными указателями на функции
<code>ItcsVpnApi* GetVpnApiExt (VpnApiInitData * params, size_t len)</code>	Получение указателя на структуру VPN API с заданными параметрами инициализации. • [in] <code>params</code> — параметры инициализации, • [in] <code>len</code> — количество параметров
<code>uint32_t GetVpnApiVersion()</code>	Получение номера текущей версии VPN API
<code>VpnApiReturnCode (*BlockUserTraffic) (enum VpnUserTrafficBlockStatus status)</code>	Установка статуса блокировки защищённого пользовательского трафика. • [in] <code>status</code> — статус блокировки защищённого пользовательского трафика
<code>VpnApiReturnCode (*CheckActiveCoordinatorConnection) (uint32_t* nodeId)</code>	Проверка соединения с активным координатором. • [out] <code>nodeId</code> — идентификатор активного координатора
<code>VpnApiReturnCode (*CheckNodeConnection) (uint32_t nodeId)</code>	Проверка соединения с узлом. • [in] <code>nodeId</code> — идентификатор узла

Функция	Описание
<code>VpnApiReturnCode (*CollectEventlog) (const char* path, const char* psw, bool truncate)</code>	Экспорт журнала критически важных событий в ZIP-архив. [in] path — путь для сохранения файла архива, [in] psw — пароль администратора узла, [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
<code>VpnApiReturnCode (*CollectTokens) (int (*cbk) (VpnTokenHandle))</code>	Получение списка подключённых и поддерживаемых внешних устройств (токенов). [in] cbk — обработчик дескриптора токена; может вернуть не 0 для прерывания опроса
<code>VpnApiReturnCode (*DeleteKeys) ()</code>	Удаление ключей
<code>VpnApiReturnCode (*ExportRefBook) (const char* refBook, const char* filepath)</code>	Экспорт файла справочника. [in] refBook — название справочника, [in] filepath — путь для экспорта файла справочника
<code>VpnApiReturnCode (*GetActiveCoordinator) (uint32_t* id)</code>	Получение идентификатора (ID) активного координатора (сервера IP-адресов). [out] id — ID узла
<code>VpnApiReturnCode (*GetAvailableRefBooks) (VpnRefBookListInfo** refBookList)</code>	Получение информации о списке доступных справочников. [out] refBookList — информация о списке доступных справочников. Память для refBookList выделяется динамически и должна быть освобождена с помощью <code>ReleaseRefBookList(refBookList)</code>
<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] key — ключ для поиска данных, [in] keyLength — длина буфера с ключом поиска данных, [out] outData — буфер для возвращения данных, [out] outDataLength — размер динамически выделенного буфера с данными. Память для outData выделяется динамически и должна быть освобождена с помощью <code>ReleaseClientParamData(outData)</code>

Функция	Описание
<code>VpnApiReturnCode (*GetClientVersion) (char* version, size_t* size)</code>	Получение текущей версии ViPNet Client 5. • [out] <code>version</code> — буфер для передачи версии или <code>NULL</code> для получения необходимого размера буфера, • [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>	Проверка работоспособность службы <code>vipnetclient</code>. • [out] <code>status</code> — результат проверки
<code>VpnApiReturnCode (*GetLicenseExpiration) (time_t* time)</code>	Получение даты и времени истечения срока действия лицензии в секундах в формате POSIX. • [out] <code>time</code> — дата и время истечения срока действия лицензии в секундах
<code>VpnApiReturnCode (*GetLogLevel) (int32_t* logLevel)</code>	Получение текущего уровня детализации в журнале ViPNet Client 5. • [out] <code>logLevel</code> — уровень детализации в журнале
<code>VpnApiReturnCode (*GetNodeCrypto) (uint32_t id, char** keyType)</code>	Получение информации о шифраторе заданного узла. • [in] <code>id</code> — ID узла. • [out] <code>keyType</code> — указатель на строку с JSON-структурой в UTF-8. Строка с результатом должна быть освобождена после использования с помощью вызова <code>ReleaseNodeCrypto(keyType)</code>. Возвращаемая структура имеет формат: <pre>{ "keyType": "<Тип ключа: Default, Quantum, etc>", "keyStartDate": "<Время начала действия квантового ключа (time_t)>" }</pre>

Функция	Описание
<pre>VpnApiReturnCode (*GetNodeInfo)(uint32_t id, VpnNodeInfo** nodeInfo)</pre>	Получение информации об узле по его идентификатору (ID). • [in] id — ID узла, • [out] nodeInfo — информация об узле. Память для nodeInfo выделяется динамически и должна быть освобождена с помощью <code>ReleaseVpnNodeInfo(nodeInfo, 1)</code>
<pre>VpnApiReturnCode (*GetNodeInfoExt)(uint32_t id, VpnNodeInfoExt** nodeInfo)</pre>	Получение расширенной информации об узле по его идентификатору (ID). • [in] id — ID узла, • [out] nodeInfo — информация об узле. Память для nodeInfo выделяется динамически и должна быть освобождена с помощью <code>ReleaseNodeInfoExt(nodeInfo)</code>
<pre>VpnApiReturnCode (*GetNodesIds)(uint32_t** nodeIds, size_t* size)</pre>	Получение списка идентификаторов (ID) узлов. • [out] nodeIds — список ID узлов, • [out] size — количество узлов. Память для nodeIds выделяется динамически и должна быть освобождена с помощью <code>ReleaseVpnNodeIds(nodeIds, size)</code>
<pre>VpnApiReturnCode (*GetNodesInfo)(VpnNodeInfo** nodesInfo, size_t* size)</pre>	Получение информации об узлах. • [out] nodesInfo — информация об узлах, • [out] size — количество узлов. Память для nodesInfo выделяется динамически и должна быть освобождена с помощью <code>ReleaseVpnNodeInfo(nodesInfo, size)</code>
<pre>VpnApiReturnCode (*GetNodesInfoByTask)(VpnNodeInfo** nodesInfo, size_t* size, VpnNodeTasksMask tasksMask)</pre>	Получение информации об узлах по маске ролей и свойств узла. • [out] nodesInfo — информация об узлах, • [out] size — количество узлов, • [in] tasksMask — битовая маска ролей и свойств узла. Память для nodesInfo выделяется динамически и должна быть освобождена с помощью <code>ReleaseVpnNodeInfo(nodesInfo, size)</code>
<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 <code>ReleaseVpnNodeInfo(nodeInfo, 1)</code>

Функция	Описание
<pre>VpnApiReturnCode (*GetOwnPrivilegeLevel)(enum VpnPrivilege* privilege)</pre>	Получение уровня полномочий своего узла. [out] privilege — уровень полномочий
<pre>VpnApiReturnCode (*GetTokenConfig)(int (*cbk)(const char*))</pre>	Получение конфигурации 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, 1)
<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)
<pre>VpnApiReturnCode (*GetUserTrafficBlockInfo) (VpnUserTrafficBlockInfo** blockInfo)</pre>	Получение информации о блокировке защищённого пользовательского трафика. • [out] blockInfo — информация о блокировке защищённого пользовательского трафика. Память для blockInfo выделяется динамически и должна быть освобождена с помощью ReleaseUserTrafficBlockInfo(blockInfo)
<pre>VpnApiReturnCode (*GetVpnStatus) (VpnStatus* vpnStatus)</pre>	Получение объекта, описывающего текущее состояние VPN-соединения. • [out] vpnStatus — текущий статус VPN-соединения
<pre>VpnApiReturnCode (*InstallKeys) (const char* dstPath, const char* psu)</pre>	Установка ключей. • [in] dstPath — путь к дистрибутиву ключей (*.dst или *.ds5), который будет установлен, • [in] psu — пароль от дистрибутива ключей

Функция	Описание
<pre>VpnApiReturnCode (*InstallKeysExt)(const char* path, const char* password, VpnDstStatus* status)</pre>	Установка ключей и настройка службы для доступа в защищённую сеть. • [in] path — путь к дистрибутиву ключей (*.dst или *.ds5), • [in] password — пароль от дистрибутива ключей (может быть пуст или NULL), • [out] status — результат проверки файла *.dst или *.ds5 и статус установки ключей
<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 — журнал. По умолчанию 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 использоваться 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. Для успешного выполнения команды VPN-соединение должно быть включено
<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 (*ReleaseNodeCrypto)(char* keyType)</pre>	Освобождение памяти, выделенной при вызове GetNodeCrypto. • [in] keyType — указатель на строку, возвращённую GetNodeCrypto
<pre>VpnApiReturnCode (*ReleaseNodeInfoExt)(VpnNodeInfoExt* nodeInfo)</pre>	Освобождение памяти, выделенной для расширенной информации об узлах. • [in] nodeInfo — информация об узлах, память которой необходимо освободить

Функция	Описание
<pre>VpnApiReturnCode (*ReleaseRefBookList) (VpnRefBookListInfo* refBookList)</pre>	Освобождение памяти, выделенной для информации о списке справочников. [in] refBookList — информация о списке справочников, 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>VpnApiReturnCode (*ReleaseUserTrafficBlockInfo) (VpnUserTrafficBlockInfo* blockInfo)</pre>	Освобождение памяти, выделенной для информации о блокировке защищённого пользовательского трафика. [in] blockInfo — информация о блокировке защищённого пользовательского трафика, память которой необходимо освободить
<pre>void (*ReleaseVpnNodeIds) (uint32_t* nodeIds, size_t size)</pre>	Освобождение памяти, выделенной для списка узлов. [in] nodeIds — массив ID узлов, который необходимо освободить, [in] size — количество узлов
<pre>void (*ReleaseVpnNodeInfo) (VpnNodeInfo* nodeInfo, size_t size)</pre>	Освобождение памяти, выделенной для информации об узлах. [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, который 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

Функция	Описание
<code>VpnApiReturnCode (*SetEventHandlers) (const char* consumer, VpnEventHandlers* events)</code>	Установка обработчиков событий службы vipnetclient. Позволяет установить индивидуальные функции для обработки каждого события, а также принимать параметры этих событий (если предусмотрено). • [in] <code>consumer</code> — уникальный идентификатор подписчика (не должен содержать пробелы), • [in] <code>events</code> — структура указателей на функции-обработчики
<code>VpnApiReturnCode (*SetLogLevel) (int32_t logLevel)</code>	Выставление уровня детализации в журнале событий ViPNet Client 5. • [in] <code>logLevel</code> — уровень детализации в журнале
<code>VpnApiReturnCode (*SetOptionAutorun) (void)</code>	Включение функции автозапуска VPN-соединения при авторизации пользователя в ОС
<code>VpnApiReturnCode (*SetOptionNoAutorun) (void)</code>	Выключение функции автозапуска VPN-соединения при авторизации пользователя в ОС
<code>VpnApiReturnCode (*SetOptionFaultTolerance) (void)</code>	Включение функции восстановления ViPNet Client 5 при сбоях. Настройка хранится в памяти процесса, использующего API, и после перезапуска теряется. По умолчанию система восстановления активируется при включении VPN-соединения
<code>VpnApiReturnCode (*SetOptionNoFaultTolerance) (void)</code>	Выключение функции восстановления ViPNet Client 5 при сбоях. Настройка хранится в памяти процесса, использующего API, и после перезапуска теряется
<code>VpnApiReturnCode (*SetPin) (VpnTokenHandle handle, const char* pin)</code>	Сохранение PIN-кода 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

Функция	Описание
<code>VpnApiReturnCode (*StopVpn)()</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 или <code>NULL</code> для отмены подписки
<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
<code>void (*VpnEventHandlerExt) (void* data, enum VpnEvent event)</code>	Прототип обработчика событий с дополнительными данными. • [in] <code>data</code> — ненулевой указатель на произвольные данные, • [in] <code>event</code> — событие VPN-соединения
<code>VpnApiReturnCode (*SetExchangeFolder) (const wchar_t* folder)</code>	Установка директории обмена для настройки RPC через TCP/IP. • [in] <code>folder</code> — директория обмена
<code>VpnApiReturnCode (*GetConnectionId) (int32_t* conn)</code>	Получение идентификатора запущенного VPN-соединения. • [out] <code>conn</code> — идентификатор запущенного VPN-соединения
<code>VpnApiReturnCode (*GetConnectionIdRoutine) (int32_t* conn)</code>	Получение идентификатора запущенного VPN-соединения через DBus-RPC вызов. • [out] <code>conn</code> — идентификатор запущенного VPN-соединения

Функция	Описание
<code>VpnApiReturnCode (*GetTunnelVisibility) (enum VpnTunnelVisibilityMode* mode)</code>	Получение режима видимости туннелей. [out] mode — режим видимости туннелей
<code>VpnApiReturnCode (*SetTunnelVisibility) (enum VpnTunnelVisibilityMode mode)</code>	Установка режима видимости туннелей. [in] mode — режим видимости туннелей
<code>VpnApiReturnCode (*SetOptionConnectServerGroup) (enum VpnConnectServerGroupState state)</code>	Включение и выключение настройки «Группа серверов». [in] state — желаемое состояние настройки «Группа серверов»
<code>VpnApiReturnCode (*GetOptionConnectServerGroup) (enum VpnConnectServerGroupState* state)</code>	Получение информации о состоянии настройки «Группа серверов». [out] state — состояние настройки «Группа серверов»
<code>VpnApiReturnCode (*ChangeCredentials) (const char* password)</code>	Смена пароля. [in] password — новый пароль
<code>VpnApiReturnCode (*ContinueChangeCredentials) (const char* oldPassword, const char* newPassword)</code>	Возобновление смены пароля после сбоя в GLE. [in] oldPassword — текущий пароль, [in] newPassword — новый пароль
<code>VpnApiReturnCode (*GetDmsStatus) (enum VpnDmsStatus* status)</code>	Получение информации о состоянии мастер-ключа устройства. [out] status — состояние мастер-ключа устройства
<code>VpnApiReturnCode (*GetExpiredDateDeferredChangeDms) (time_t* date)</code>	Получение информации о дате, на которую перенесена смена пароля. [out] date — дата, на которую перенесли смену пароля
<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

Функция	Описание
<code>VpnApiReturnCode (*GetOptionActiveCoordinatorInsist) (bool* insist)</code>	Получение информации о режиме выбора сервера соединений. [out] <code>insist</code> — игнорировать настройки активного координатора из управляющего ПО
<code>VpnApiReturnCode (*GetUserTrafficRestrictions) (VpnUserTrafficRestrictions restrictions)</code>	Получение списка причин блокировки пользовательского трафика. [out] <code>restrictions</code> — список причин блокировки пользовательского трафика
<code>VpnApiReturnCode (*StartMfAuth) (void)</code>	Запуск многофакторной аутентификации
<code>VpnApiReturnCode (*GetMfAuthStatus) (VpnMfAuthStatus* status)</code>	Получение статуса многофакторной аутентификации. [out] <code>status</code> — статус многофакторной аутентификации
<code>VpnApiReturnCode (*GetMfAuthChallengeTips) (VpnMfAuthChallengeTipsList** tips)</code>	Получение списка доступных способов многофакторной аутентификации. [out] <code>tips</code> — список доступных способов многофакторной аутентификации. Память для списка доступных способов многофакторной аутентификации выделяется динамически и должна быть освобождена с помощью <code>ReleaseMfAuthChallengeTips(tips)</code>
<code>void (*ReleaseMfAuthChallengeTips) (VpnMfAuthChallengeTipsList* tips)</code>	Освобождение памяти, выделенной для списка доступных способов многофакторной аутентификации. [in] <code>tips</code> — 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] <code>login</code> — имя пользователя, [in] <code>password</code> — пароль
<code>VpnApiReturnCode (*EnterMfAuthChallengeData) (const char* data)</code>	Ввод или выбор второго фактора для многофакторной аутентификации. [in] <code>data</code> — второй фактор для многофакторной аутентификации
<code>VpnApiReturnCode (*CancelMfAuth) (void)</code>	Отмена многофакторной аутентификации

Функция	Описание
<code>VpnApiReturnCode (*GetMfAuthAllowRestartIn) (uint32_t* timeout)</code>	Получение информации о времени, через которое можно повторить многофакторную аутентификацию. [out] <code>timeout</code> — время в секундах, через которое можно повторить многофакторную аутентификацию
<code>VpnApiReturnCode (*InstallKeysV10) (const char* path, const char* password, VpnDstCheckListInfoV10** status)</code>	Установка ключей. [in] <code>path</code> — путь к дистрибутиву ключей, [in] <code>password</code> — пароль узла, [out] <code>status</code> — список ошибок при установке ключей. Память для списка ошибок при установке ключей выделяется динамически и должна быть освобождена с помощью <code>ReleaseDstCheckListInfoV10(status)</code>
<code>VpnApiReturnCode (*GetDstInfoV10) (const char* path, VpnDstInfoV10* info, VpnDstCheckListInfoV10** status)</code>	Получение информации о дистрибутиве ключей. [in] <code>path</code> — путь к дистрибутиву ключей, [out] <code>info</code> — информация о дистрибутиве ключей, [out] <code>status</code> — список ошибок при установке ключей. Память для списка ошибок при установке ключей выделяется динамически и должна быть освобождена с помощью <code>ReleaseDstCheckListInfoV10(status)</code>
<code>void (*ReleaseDstCheckListInfoV10) (VpnDstCheckListInfoV10* info)</code>	Освобождение памяти, выделенной для списка ошибок при установке ключей. [in] <code>info</code> — список ошибок при установке ключей, память которого нужно освободить
<code>VpnApiReturnCode (*CheckUserPassword) (const char* password)</code>	Проверка пароля. [in] <code>password</code> — пароль узла
<code>VpnApiReturnCode (*GetLoginAvailabilityInfo) (VpnLoginAvailabilityInfo* info)</code>	Получение информации о возможности повторить попытку входа. [out] <code>info</code> — информация о возможности повторить попытку входа
<code>VpnApiReturnCode (*GetKeySystemVersion) (VpnKeySystemVersion* ks)</code>	Получение информации о поколении ключей. [out] <code>ks</code> — поколение ключей
<code>VpnApiReturnCode (*AdoptEnvironment) (void)</code>	Обновление контрольных сумм среды функционирования
<code>VpnApiReturnCode (*IsTrustedAccessGranted) (bool* granted)</code>	Получение признака доверенности процесса. <code>granted</code> — признак доверенности процесса

Функция	Описание
<code>VpnApiReturnCode (*GetIgnorePrivateDnsServers) (bool* isIgnore)</code>	Получение признака «Не использовать защищённые DNS-серверы для запросов открытой сети». <code>isIgnore</code> — игнорировать DNS-запросы открытой сети
<code>VpnApiReturnCode (*SetIgnorePrivateDnsServers) (bool isIgnore)</code>	Установка признака «Не использовать защищённые DNS для запросов открытой сети» <code>isIgnore</code> — игнорировать DNS-запросы открытой сети
<code>VpnApiReturnCode (*ReplaceUserVpnconfig) (const char* userVpnconfig)</code>	Применение файла настроек. <code>userVpnConfig</code> — файл настроек
<code>VpnApiReturnCode (*CollectProfiles) (bool (*cbk) (ProfileHandle hdl))</code>	Получение списка доступных профилей. <code>cbk</code> — обработчик дескриптора профиля. При возвращении <code>false</code> перебор списка прекращается
<code>VpnApiReturnCode (*GetProfileInfo) (ProfileHandle hdl, ProfileInfo* info)</code>	Получение информации о профиле по его дескриптору. <code>hdl</code> — дескриптор профиля, <code>info</code> — информация о профиле
<code>VpnApiReturnCode (*ActivateProfile) (ProfileHandle hdl)</code>	Сделать профиль активным. <code>hdl</code> — дескриптор профиля, который нужно сделать активным
<code>VpnApiReturnCode (*DeleteProfile) (ProfileHandle hdl)</code>	Удаление профиля. <code>hdl</code> — дескриптор профиля, который нужно удалить
<code>VpnApiReturnCode (*FreeProfile) (ProfileHandle hdl)</code>	Освобождение памяти, связанной с указанным профилем. <code>hdl</code> — дескриптор профиля
<code>VpnApiReturnCode (*SetProfileCustomName) (const char* name)</code>	Задание имени активного профиля. <code>name</code> — новое имя активного профиля
<code>VpnApiReturnCode (*AddCustomAdminLogEntries) (const char* name, const char* programname, const char* entries)</code>	Запись в журнал аудита набора сообщений. <code>name</code> — имя журнала аудита, <code>programname</code> — имя программы, <code>entries</code> — набор записей, разделённых символом перевода строки

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

Ключ	Описание
<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 programName = "program_name"</code>	Ключ для задания имени службы ViPNet, специфичен для платформы Linux
<code>const char* const initKeyWorkDirs = "WorkDirs"</code>	Ключ для задания пути к рабочим директориям.
<code>const char* const initKeyRpcPort = "RpcPort"</code>	Ключ для задания порта RPC-сервера
<code>const char* const initTrustedProcess = "trusted_process"</code>	Ключ для передачи атрибута доверенного процесса

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

Ключ	Описание
<code>const char* const paramNvsConnectionVipnetUrls = "nvs.info/connection.vipnetUrls"</code>	Ключ для получения ссылки для доступа к системе мониторинга ViPNet NVS по защищённой сети
<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 paramSystemNetworkMonitoringVipnetUrl s = "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. *VpnDstInfo*

Поле	Описание
<code>VpnDstStatus errors</code>	Результат проверки файла дистрибутива ключей <code>*.dst</code> или <code>*.ds5</code>
<code>time_t licenseExpired</code>	Дата истечения лицензии
<code>time_t licenseIssued</code>	Дата выдачи <code>*.dst</code> или <code>*.ds5</code>
<code>size_t logonDevice : 1</code>	Требуется токен для аутентификации
<code>uint32_t nodeId</code>	Идентификатор узла (ViPNet ID)
<code>char nodeName[NODE_NAME_MAX]</code>	Название узла
<code>size_t password : 1</code>	Для расшифровки резервных ключей требуется пароль
<code>enum VpnPrivilege privileges</code>	Полномочия пользователя

Таблица 6. *VpnDstStatus*

Поле	Описание
<code>size_t CheckDstLicenceExpired : 1</code>	Файл *.dst или *.ds5 содержит истекшую лицензию
<code>size_t CheckDstMobileTaskNotFound : 1</code>	В файле *.dst или *.ds5 не найдена роль «Client for Linux» для собственного узла
<code>size_t CheckDstUnmergeError : 1</code>	Системная ошибка распаковки дистрибутива ключей
<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 от дистрибутива ключей
<code>size_t CheckDstWrongParams : 1</code>	В 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

Поле	Описание
size_t InvalidAlgServiceConfiguration : 1	Невалидная конфигурация службы alg
size_t DnsZoneIncludesForbiddenAddr : 1	Доменные зоны включают в себя недопустимый адрес
size_t CheckDNSNameNumber : 1	Количество доменных имён координатора не должно быть более 5
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 7. *VpnHealthStatus*

Поле	Описание
size_t cryptography : 1	Результат проверки криптобиблиотеки CSP (1 — работоспособность не нарушена)
size_t environment : 1	Результат проверки целостности окружения (1 — целостность не нарушена)
size_t integrity : 1	Результат проверки целостности исполняемых файлов (1 — целостность не нарушена)
size_t operating_environment : 1	Результат проверки среды функционирования (1 — проверка пройдена)

Таблица 8. *VpnStatus*

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

Таблица 9. *VpnNodeInfo*

Поле	Описание
<code>uint32_t id</code>	Идентификатор узла (ViPNet ID)
<code>uint32_t ip</code>	IPv4-адрес узла
<code>char* name</code>	Название узла
<code>VpnNodeTasksMask tasksMask</code>	Битовая маска ролей и свойств узла

Таблица 10. *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 ID)
<code>uint32_t ip</code>	IPv4-адрес узла
<code>const char* name</code>	Название узла
<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 startIp uint32_t endIp } * 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 startIp uint32_t endIp } * tunnelsVirtual</pre>	Массив диапазонов виртуальных адресов, туннелируемых этим узлом (если узел — координатор)
<code>size_t tunnelsVirtualSize</code>	Размер массива диапазонов туннелируемых виртуальных адресов

Таблица 11. *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 12. *VpnUltimateStatus*

Поле	Описание
<code>size_t autorunDesktop : 1</code>	Флаг настройки автозапуска при авторизации в графическом интерфейсе. • 1 — включена, • 0 — отключена
<code>size_t autorunShell : 1</code>	Флаг настройки автозапуска при авторизации в командной строке. • 1 — включена, • 0 — отключена
<code>char cipher[16]</code>	Наименование криптографического алгоритма, на котором работает узел
<code>size_t configUpdated : 1</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)

Поле	Описание
<code>int faultTolerance</code>	Количество попыток восстановления. • 0 — настройка отключена, • N — количество доступных попыток перезапуска (уменьшается с каждым перезапуском), • 0x7FFFFFFF — количество попыток не ограничено, • -1 — ошибка системы восстановления
<code>size_t integrityConfigs : 1</code>	Результат проверки контрольных сумм конфигурационных файлов. • 1 — целостность не нарушена, • 0 — целостность нарушена
<code>size_t integrityHostlinks : 1</code>	Результат проверки контрольных сумм справочников. • 1 — целостность не нарушена, • 0 — целостность нарушена
<code>size_t integrityLicense : 1</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 — действительна, но истекает сегодня, • 6-12 — действительна, но истекает через 1-7 дней, • 13 — неограниченная
<code>size_t loginCounterAdmin : 1</code>	Флаг аутентификации администратора. • 1 — разрешена, • 0 — заблокирована

Поле	Описание
<code>size_t loginCounterUser : 1</code>	Флаг аутентификации пользователя (по паролю). • 1 — разрешена, • 0 — заблокирована
<code>size_t loginType : 3</code>	Тип аутентификации. • 0 — не определён, • 1 — пароль, • 2 — пароль на устройстве, • 4 — ключ на устройстве
<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 ID)
<code>char nodeName[NODE_NAME_MAX]</code>	Название узла
<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 в сети ViPNet
<code>char workdir[PATH_MAX]</code>	Путь к файлам текущей конфигурации

Таблица 13. *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
<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 14. *VpnNodeTasksMask*

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

Таблица 15. *VpnTokenHandle*

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

Таблица 16. *VpnEventHandlers*

Поле
<code>VIPNET_EVENT(vipnetDaemonStarted, void)</code>
<code>VIPNET_EVENT(vipnetDaemonStopped, void)</code>
<code>VIPNET_EVENT(vipnetStarted, void)</code>

Поле
VIPNET_EVENT(vipnetStopped, void)
VIPNET_EVENT(vipnetIntegrityCheck, int)
VIPNET_EVENT(vipnetUpdateReceived, void)
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, void)
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, int)
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)

Поле
VIPNET_EVENT(vipnetProfileBound, bool)
VIPNET_EVENT(vipnetProfileDelete, void)

Таблица 17. VpnEvent

Поле
Event_VpnServiceStarted
Event_VpnServiceStopped
Event_KeysInstalled
Event_KeysUninstalled
Event_VpnStarted
Event_VpnStopped
Event_NodesUpdated
Event_VpnBeginStarting
Event_VpnBeginStopping

Таблица 18. 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

Поле	Описание
Task_NativeCoordinator	0x4000
Task_BusinessMail	0x8000

Таблица 19. 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

Поле	Описание
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
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-VPNМ
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

Поле	Описание
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 20. VpnPrivilege

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

Таблица 21. LogLevel

Поле	Описание
LOG_CRIT	0, критические ошибки
LOG_ERR	1, ошибки
LOG_NOTICE	2, уведомления
LOG_INFO	3, информационные сообщения
LOG_DEBUG	4, отладочная информация
LOG_DEEP	5, остальные уровни используются для детального вывода
LOG_DEEPER	6
LOG_GARBAGE	10
LOG_PARANOID	11

Таблица 22. VpnNodeTasksMaskAll

Поле	Описание
VPN_NODE_TASKS_MASK_ALL	UINT32_MAX

Таблица 23. *VpnUserTrafficBlockStatus*

Поле	Описание
UserTrafficNotBlocked	0, защищённый пользовательский трафик не заблокирован
UserTrafficBlocked	1, защищённый пользовательский трафик заблокирован

Таблица 24. *VpnUserTrafficBlockInfo*

Поле	Описание
enum VpnUserTrafficBlockStatus status	Статус блокировки защищённого пользовательского трафика
const char* processName	Имя процесса, инициировавшего блокировку

Таблица 25. *VpnRefBookInfo*

Поле	Описание
const char* name	Имя справочника

Таблица 26. *VpnRefBookListInfo*

Поле	Описание
VpnRefBookInfo* list	Массив справочников
uint32_t size	Размер массива справочников

Таблица 27. *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
uint32_t password : 1	Задана ли аутентификация по паролю

Таблица 28. *VpnDstCheckListInfoV10*

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

Таблица 29. *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 при распаковке ключей
<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 прочитать 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 <code>On Demand Connect</code>
<code>DstCheckCode_ConfigNotSupported = 17</code>	Конфигурационный файл повреждён
<code>DstCheckCode_DnsFileTooBig = 18</code>	Превышен размер конфигурационного файла
<code>DstCheckCode_ConfigureValueIsEmpty = 19</code>	В конфигурационном файле отсутствуют параметры
<code>DstCheckCode_DnsConnectDemandNotAllowed = 20</code>	У пользователя недостаточно полномочий для включения настройки <code>On Demand Connect</code>
<code>DstCheckCode_AbTooManyDns = 21</code>	Превышено количество DNS-серверов

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

Таблица 30. *VpnKeySystemVersion*

Поле	Описание
<code>KeySystemInvalid = 0</code>	Поколение ключей не определено
<code>KeySystem4 = 4</code>	Ключи 4-го поколения
<code>KeySystem5 = 5</code>	Ключи 5-го поколения

Таблица 31. *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 32. *VpnMfAuthChallengeTipsList*

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

Таблица 33. *VpnMfAuthChallengeTipInfo*

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

Таблица 34. *VpnMfAuthRequestStatus*

Поле	Описание
<code>MFARequestStatus_Idle = 0</code>	Запрос не сделан
<code>MFARequestStatus_Pending = 1</code>	Запрос отправлен, ожидание ответа
<code>MFARequestStatus_Continue = 2</code>	Получен ответ «Продолжить»
<code>MFARequestStatus_Success = 3</code>	Получен ответ «Успех»
<code>MFARequestStatus_Fail = 4</code>	Получен ответ «Не удалось»
<code>MFARequestStatus_Timeout = 5</code>	Истекло время запроса

Поле	Описание
MFARequestStatus_Cancelled = 6	Запрос отменен
MFARequestStatus_Error = 7	Ошибка запроса
MFARequestStatus_Unknown = 8	Статус запроса неизвестен

Таблица 35. 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 ввода учётных данных
MFAStatus_ChallengeSelect = 4	Ожидание выбора способа аутентификации
MFAStatus_ChallengeInput = 5	Ожидание ввода второго фактора
MFAStatus_ChallengeMessage = 6	Ожидание от пользователя действий, описанных в сообщении
MFAStatus_NotAuthenticatedCooldown = 7	Ожидание после неудачной попытки
MFAStatus_NotInitialized = 8	Не запущен модуль многофакторной аутентификации
MFAStatus_Invalid = ~0	Неверный статус аутентификации

Таблица 36. VpnUserTrafficRestrictions

Поле	Описание
VpnUserTrafficBlockStatus VpnUserTrafficRestrictions[UserTrafficBlockReason_Count]	Список причин блокировки пользовательского трафика

Таблица 37. 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

Поле	Описание
UserTrafficBlockReason_Feature_Ztna = 5	Трафик заблокирован из-за ViPNet EPP
UserTrafficBlockReason_Feature_Others = 6	Трафик заблокирован из-за настроек в справочнике node.info
UserTrafficBlockReason_Environment = 7	Нарушена целостность среды функционирования
UserTrafficBlockReason_Count = 8	Количество причин блокировки трафика

Таблица 38. 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 39. 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 40. *VpnDstFileState*

Поле	Описание
<code>DstNotCheckedOrNotInstalled = 0</code>	Ключи не установлены или не прошли проверку
<code>DstIsChecked = 1</code>	Ключи установлены и проверены
<code>DstIsInvalid = 2</code>	Ключи недействительны из-за смены мастер-ключа

Таблица 41. *VpnLicenseStatus*

Поле	Описание
<code>LicenseDataError = 1</code>	Лицензия повреждена
<code>LicenseDataExpired = 2</code>	Лицензия истекла
<code>LicenseByPeriod = 3</code>	Лицензия содержит ошибочную дату
<code>LicenseNotFound = 4</code>	Лицензия отсутствует
<code>LicenseUnlimited = 13</code>	Лицензия неограниченная

Таблица 42. *VpnConnectServerGroupState*

Поле	Описание
<code>ConnectServerGroupState_Off = 0</code>	Настройка «Группа Серверов» выключена
<code>ConnectServerGroupState_On = 1</code>	Настройка «Группа Серверов» включена

Таблица 43. *VpnTunnelVisibilityMode*

Поле	Описание
<code>VpnTunnelVisibilityMode_Real = 0</code>	Реальные туннели
<code>VpnTunnelVisibilityMode_Virtual = 1</code>	Виртуальные туннели

Таблица 44. *VpnWorkDirs*

Поле	Описание
<code>const char* workdir</code>	Рабочая директория
<code>const char* exchangedir</code>	Директория обмена
<code>const char* businessmaildir</code>	Директория деловой почты
<code>const char* transportdir</code>	Транспортная директория

Таблица 45. *VpnApilnitData*

Поле	Описание
<code>const char* key</code>	Ключ для инициализации
<code>void* value</code>	Значение для инициализации

Таблица 46. *VpnErrors*

Поле	Описание
<code>Success = 0x00000000</code>	Успех
<code>ErrUnmergeContext = 0x00030000</code>	Контекст ошибок при разворачивании дистрибутива ключей
<code>ErrKeysIntegrityFailed = 0x00030040</code>	Нарушение целостности файлов конфигурации
<code>ErrIplircfgFailed = 0x00030200</code>	Не пройдена проверка файлов конфигурации
<code>ErrKeysNotFoundInFile = 0x00030400</code>	Не найден дистрибутив ключей
<code>ErrKeysUserNotRegistered = 0x00030800</code>	На устройстве нет персонального ключа выбранного узла
<code>ErrDstContext = 0x00040000</code>	Контекст ошибок при проверке дистрибутива ключей
<code>ErrDstContextMask = 0x0004ffff</code>	Маска ошибок при проверке дистрибутива ключей
<code>ErrPasswordContext = 0x00050000</code>	Контекст ошибок при проверке пароля
<code>ErrPasswordIsWrong = 0x00050001</code>	Неверный пароль
<code>ErrNoPasswordAttemptsLeft = 0x00050002</code>	Превышено количество попыток ввода пароля
<code>ErrPasswordIsExpired = 0x00050003</code>	Пароль истёк
<code>ErrCriticalAuthFailure = 0x00050004</code>	Критическая ошибка аутентификации
<code>ErrWrongLoginState = 0x00050005</code>	Ошибка проверки пароля, если пользователь не аутентифицирован
<code>ErrLoginCounterFileError = 0x00050006</code>	Ошибка чтения, записи или целостности файла-счётчика попыток аутентификации
<code>ErrFinalPasswordAttempt = 0x00050008</code>	Осталась последняя попытка ввода пароля
<code>ErrStartContext = 0x00060000</code>	Контекст ошибок при старте
<code>ErrStartKeyNotFound = 0x00060008</code>	Ключ не найден
<code>ErrStartIplircfgWrongPin = 0x000600d9</code>	Неверный PIN-код
<code>ErrStartIplircfgUserPinLocked = 0x000600e2</code>	Пользовательский PIN-код заблокирован
<code>ErrClientContext = 0x00070000</code>	Контекст ошибок клиента
<code>ErrVpnWorks = 0x00070001</code>	VPN работает, когда не должен

Поле	Описание
ErrVpnDoesntWork = 0x00070002	VPN не работает, когда должен
ErrGetNodeList = 0x00070003	Не удалось получить список узлов
ErrVpnAlreadyStarted = 0x00070005	VPN уже запущен для другого пользователя
ErrGetNodeInfo = 0x00070006	Не удалось получить информацию об узле
ErrCoordinatorSetFailed = 0x00070007	Не удалось установить активный координатор
ErrCheckConnection = 0x00070008	Не удалось проверить соединение
ErrBinaryCheckFailed = 0x00070009	Нарушение целостности бинарных файлов
ErrServiceContext = 0x00090000	Контекст ошибок сервиса
ErrServiceUserUndefined = 0x00090003	Пользователь VPN не существует
ErrServiceActionCanceled = 0x00090004	Пользователь отменил действие
ErrServiceIntegrityBroken = 0x00090005	Нарушена целостность конфигурационных или исполняемых файлов
ErrServiceDoesNotRespond = 0x00090007	Сервис не отвечает
ErrSetCapabilities = 0x00090081	Ошибка при настройке возможностей
ErrConfigureAutorun = 0x00090083	Ошибка при настройке автозапуска
ErrServiceStop = 0x00090085	Ошибка при попытке остановки
ErrServiceAlreadyStarted = 0x00090086	Ошибка повторного запуска с одинаковой конфигурацией
ErrServiceCanceledPassword = 0x00090089	Пользователь отменил ввод пароля в интерфейсе
ErrDemo = 0x00090090	Срок действия демо-сборки истёк
ErrRpc = 0x00090091	Ошибка RPC
ErrRpcIsBusy = 0x00090092	RPC-сервер занят
ErrServiceUnknown = 0x000900ff	Неизвестная ошибка
ErrReportContext = 0x000c0000	Контекст ошибок при создании отчёта
ErrPathIsDir = 0x000c0001	Имя отчёта совпадает с именем каталога
ErrFileExists = 0x000c0002	Отчёт с таким именем уже существует в этом каталоге
ErrPathDoesNotExist = 0x000c0003	Родительская директория не существует
ErrPathIsNotAvailable = 0x000c0004	Родительская директория недоступна
ErrReportCreation = 0x000c0005	Ошибка при создании отчёта
ErrAuthContext = 0x000d0000	Контекст ошибок при аутентификации
ErrIncorrectPinOrPswd = 0x000d0001	Некорректные параметры PIN-кода или пароля

Поле	Описание
<code>ErrLogonRequired = 0x000d0002</code>	Войдите для выполнения операции
<code>ErrUpdateCredentials = 0x000d0003</code>	Не удалось обновить данные аутентификации
<code>ErrContainerExists = 0x000d0004</code>	На внешнем устройстве уже есть ключи
<code>ErrWrongPin = 0x000d0005</code>	Неправильный PIN-код
<code>ErrDeviceLocked = 0x000d0006</code>	Устройство аутентификации заблокировано
<code>ErrUnsupportedMode = 0x000d0007</code>	Не поддерживается устройство аутентификации или его режим работы
<code>ErrMemoryLack = 0x000d0008</code>	На устройстве аутентификации недостаточно памяти
<code>ErrNoDevice = 0x000d0009</code>	Внешнее устройство не подключено
<code>ErrKeysNotFoundOnDevice = 0x000d000a</code>	На внешнем устройстве нет персонального ключа
<code>ErrDeviceKeysAreCorrupted = 0x000d000b</code>	Нарушена целостность ключей
<code>ErrSystemResourceAccessDenied = 0x000d000c</code>	Системные ресурсы недоступны
<code>ErrAuthInternal = 0x000d000d</code>	Внутренние ошибки аутентификации
<code>ErrProfileVpnOn = 0x00130001</code>	Не удалось выключить VPN-соединение при смене профиля
<code>ErrProfileUnbindFailed = 0x00130002</code>	Не удалось деактивировать профиль
<code>ErrProfileBindFailed = 0x00130003</code>	Не удалось активировать профиль
<code>ErrProfileAliasEmpty = 0x00130004</code>	Не задано имя профиля
<code>ErrProfileAliasTooLong = 0x00130005</code>	Имя профиля длиннее 120 символов
<code>ErrProfileAliasDuplicate = 0x00130006</code>	Профиль с таким именем уже существует
<code>ErrNoProfileToSetAlias = 0x00130007</code>	Не выбран профиль, который нужно переименовать
<code>ErrProfileAliasFailed = 0x00130008</code>	Не удалось сменить имя профиля
<code>ErrProfileNotFound = 0x00130009</code>	Профиль не найден
<code>ErrProfileIsActive = 0x0013000a</code>	Профиль активен
<code>ErrProfileRemoveFailed = 0x0013000b</code>	Не удалось удалить профиль
<code>ErrNoProfileToDelete = 0x0013000c</code>	Нет активного профиля, который нужно удалить
<code>ErrInternal = 0xffffffff</code>	Внутренняя ошибка

ItcsCipherApi

Таблица 47. *ItcsCipherApi*

Функция	Описание
<code>ItcsCipherApi* GetCipherApi()</code>	Получение указателя на структуру CIPHER API с проинициализированными указателями на функции
<code>uint32_t GetCipherApiVersion()</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. Функция устарела, рекомендуется использовать <code>EncryptBlobEx()</code>. • [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
<code>CipherApiReturnCode (*EncryptBlobEx) (uint32_t cipherSuite, uint32_t targetNodeId, uint8_t* in, size_t inSize, uint8_t* out, size_t* outSize)</code>	Зашифрование данных для узла с использованием указанного набора криптографических алгоритмов. • [in] <code>cipherSuite</code> — набор криптографических алгоритмов. • [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 22 байт

Функция	Описание
<pre> CipherApiReturnCode (*DecryptBlob) (uint32_t sourceNodeId, uint8_t* in, size_t inSize, uint8_t* out, size_t* outSize) </pre>	Расшифрование данных от узла. Функция устарела, рекомендуется использовать DecryptBlobEx(). [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
<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 38 байт, выходного — не меньше, чем размер входного минус 22 байт
<pre> CipherApiReturnCode (*IsCipherSuiteSupported) (uint32_t cipherSuite) </pre>	Проверка, поддерживается ли указанный набор криптографических алгоритмов. [in] cipherSuite — набор криптографических алгоритмов

Функция	Описание
<pre>CipherApiReturnCode (*GetEncryptedBufferMinSize)(uint32_t nodeId, uint32_t inBufferSize, uint32_t* outBufferSize)</pre>	Получение размера буфера с зашифрованными данными для узла. • [in] nodeId — ID узла, для которого нужно получить размер буфера. • [in] inBufferSize — размер буфера с входными данными. • [out] outBufferSize — размер буфера с зашифрованными данными. Узел используется для получения набора криптографических алгоритмов
<pre>CipherApiReturnCode (*GetDecryptedBufferMinSize)(uint32_t nodeId, uint32_t inBufferSize, uint32_t* outBufferSize)</pre>	Получение размера буфера с расшифрованными данными для узла. • [in] nodeId — ID узла, для которого нужно получить размер буфера. • [in] inBufferSize — размер буфера с входными данными. • [out] outBufferSize — размер буфера с расшифрованными данными. Узел используется для получения набора криптографических алгоритмов

Таблица 48. *CipherApiReturnCode*

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

Таблица 49. *CipherSuite*

Поле	Описание
InvalidCipherSuite	0xcd00dead

Таблица 50. *CryptoResult*

Поле	Описание
SUCCESS	0
BUFFER_TOO_SMALL	0x000A0001

Поле	Описание
BUFFER_IS_NULLPTR	0x000A0002
ENCRYPT_ERROR	0x000A0003
DECRYPT_ERROR	0x000A0004
KEY_NOT_FOUND_ERROR	0x000A0005
BUFFER_TOO_LARGE	0x000A0006
CIPHER_SUITE_NOT_AVAILABLE	0x000A0007
NOT_AVAILABLE	0x000A000A

Таблица 51. *CipherApiVersion*

Поле	Описание
CIPHER_API_VERSION	3

ItcsBusinessMailApi

Таблица 52. *ItcsBusinessMailApi*

Функция	Описание
<code>ItcsBusinessMailApi* GetBusinessMailApi()</code>	Получение указателя на структуру Business Mail API с проинициализированными указателями на функции
<code>uint32_t GetBusinessMailApiVersion()</code>	Получение номера текущей версии Business Mail API
<code>BusinessMailApiReturnCode (*GetBusinessMailEnvelopes) (enum VpnBusinessMailEnvelopeType type, char*** envelopes, size_t* size)</code>	Получить список конвертов, предназначенных для программы ViPNet Деловая почта. • [in] type — тип конверта. • [out] envelopes — список конвертов. • [out] size — количество конвертов. Память для envelopes выделяется динамически и должна быть освобождена самостоятельно
<code>BusinessMailApiReturnCode (*RemoveBusinessMailEnvelope) (const char* envelope)</code>	Удалить конверт для программы ViPNet Деловая почта. • [in] envelope — имя конверта

Функция	Описание
<pre>BusinessMailApiReturnCode (*UnpackBusinessMailLetter)(const char* envelope, const char* attachmentsDir, VpnBusinessMailLetterData** bmData)</pre>	Распаковать письмо, предназначенное для программы ViPNet Деловая почта. • [in] envelope — имя конверта. • [in] attachmentsDir — путь к каталогу, куда будут скопированы вложения письма, если они есть. • [out] bmData — содержимое письма. Память для bmData выделяется динамически и должна быть освобождена с помощью ReleaseVpnBusinessMailLetterData(bmData)
<pre>BusinessMailApiReturnCode (*UnpackBusinessMailReceipt)(const char* envelope, VpnBusinessMailReceiptData** bmData)</pre>	Распаковать квитанцию, предназначенную для программы ViPNet Деловая почта. • [in] envelope — имя конверта. • [out] bmData — содержимое квитанции. Память для bmData выделяется динамически и должна быть освобождена с помощью ReleaseVpnBusinessMailReceiptData(bmData).
<pre>BusinessMailApiReturnCode (*PackBusinessMailLetter)(VpnBusinessMailLetterData* bmData, char*** envelopes, size_t* size)</pre>	Сформировать и отправить письмо, предназначенное для программы ViPNet Деловая почта. • [in] bmData — содержимое письма. • [out] envelopes — имена конвертов (отсортированы в порядке следования получателей). • [out] size — количество конвертов. Память для envelopes выделяется динамически и должна быть освобождена самостоятельно
<pre>BusinessMailApiReturnCode (*PackBusinessMailReceipt)(VpnBusinessMailReceiptData* bmData, char** envelope)</pre>	Сформировать и отправить квитанцию, предназначенную для программы ViPNet Деловая почта. • [in] bmData — содержимое квитанции. • [out] envelope — имя конверта. Память для envelope выделяется динамически и должна быть освобождена самостоятельно
<pre>BusinessMailApiReturnCode (*ReleaseVpnBusinessMailLetterData)(VpnBusinessMailLetterData* bmData)</pre>	Освободить память, выделенную для содержимого письма. • [in] bmData — содержимое письма
<pre>BusinessMailApiReturnCode (*ReleaseVpnBusinessMailReceiptData)(VpnBusinessMailReceiptData* bmData)</pre>	Освободить память, выделенную для содержимого квитанции. • [in] bmData — содержимое квитанции

Функция	Описание
<pre>BusinessMailApiReturnCode (*UnpackBusinessMailLetterV2)(const char* envelope, const char* attachmentsDir, VpnBusinessMailLetterDataV2** bmData);</pre>	Распаковать письмо с конвертом версии 2 (с флагами вложений), предназначенное для программы ViPNet Деловая почта. [in] envelope — имя конверта. [in] attachmentsDir — путь к каталогу, куда будут скопированы вложения письма, если они есть. [out] bmData — содержимое письма. Память для bmData выделяется динамически и должна быть освобождена с помощью ReleaseVpnBusinessMailLetterDataV2 (bmData)
<pre>BusinessMailApiReturnCode (*PackBusinessMailLetterV2)(VpnBusinessMailLetterDataV2* bmData, char*** envelopes, uint32_t* size);</pre>	Сформировать и отправить письмо с конвертом версии 2 (с флагами вложений), предназначенное для программы ViPNet Деловая почта. [in] bmData — содержимое письма. [out] envelopes — имена конвертов, отсортированные по получателям. [out] size — количество конвертов. Память для envelopes выделяется динамически и должна быть освобождена самостоятельно
<pre>BusinessMailApiReturnCode (*ReleaseVpnBusinessMailLetterDataV2)(VpnBusinessMailLetterDataV2* bmData);</pre>	Освободить память, выделенную для содержимого письма версии 2 (с флагами вложений)

Таблица 53. BusinessMailApiReturnCode

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

Таблица 54. VpnBusinessMailEnvelopeType

Поле	Описание
BmLetter	0 (Письмо)
BmReceipt	1 (Квитанция)

Таблица 55. *VpnBusinessMailRecipientType*

Поле	Описание
BmRecipientTo	0 (Получатель из поля «Кому»)
BmRecipientCopy	1 (Получатель из поля «Копия»)
BmRecipientHidden	2 (Получатель из поля «Скрытая копия»)

Таблица 56. *VpnBusinessMailLetterData*

Поле	Описание
uint32_t sender	Идентификатор узла отправителя
uint32_t* recipientIds	Идентификаторы узлов всех получателей письма. Для входящих писем идентификатор вашего узла идёт первым в списке получателей
enum VpnBusinessMailRecipientType* recipientTypes	Тип получателей письма
size_t recipientsCount	Количество получателей письма
const char* regNumber	Регистрационный номер письма
const char* subject	Тема письма
const char* message	Текст письма
time_t creationTime	Дата создания письма
time_t sendTime	Дата отправки письма
const char** attachmentPaths	Пути к файлам вложений
const char** attachmentNames	Имена файлов вложений
size_t attachmentsCount	Количество файлов вложений

Таблица 57. *VpnBusinessMailReceiptData*

Поле	Описание
uint32_t sender	Идентификатор узла отправителя
uint32_t recipient	Идентификатор узла получателя
uint8_t type	Тип квитанции
const char* letterName	Имя конверта
time_t creationTime	Дата создания квитанции

Таблица 58. *BusinessMailApiVersion*

Поле	Описание
BUSINESS_MAIL_API_VERSION	2

Таблица 59. *VpnBusinessMailLetterDataV2*

Поле	Описание
uint32_t sender	Идентификатор узла отправителя
uint32_t* recipientIds	Идентификаторы узлов всех получателей письма. Для входящих писем идентификатор вашего узла идёт первым в списке получателей
enum VpnBusinessMailRecipientType* recipientTypes	Тип получателей письма
size_t recipientsCount	Количество получателей письма
const char* regNumber	Регистрационный номер письма
const char* subject	Тема письма
const char* message	Текст письма
time_t creationTime	Дата создания письма
time_t sendTime	Дата отправки письма
const char** attachmentPaths	Пути к файлам вложений
const char** attachmentNames	Имена файлов вложений
size_t attachmentsCount	Количество файлов вложений
uint32_t* attachmentFlags	Флаги вложений

Таблица 60. *VpnBusinessMailRecipientType*

Поле	Описание
BmAttachmentNoFlags	0 (Вложение не имеет специальных флагов)
BmAttachmentSignature	0x08 (Вложение является PKCS#7-подписью)
BmAttachmentSigned	0x20 (Вложение подписано откреплённой PKCS#7-подписью)

ItcsEncryptionApi

Таблица 61. *ItcsEncryptionApi*

Функция	Описание
<code>ItcsEncryptionApi* GetEncryptionApi()</code>	Получение указателя на структуру Encryption API с проинициализированными указателями на функции
<code>uint32_t GetEncryptionApiVersion()</code>	Получение номера текущей версии Encryption API
<code>EncryptionApiReturnCode (*EncryptFile) (uint32_t nodeId, const char* filePath, VpnRawData** encryptionData)</code>	Шифрование файла на ключах узла. • [in] nodeId — ID узла, для которого производится шифрование. • [in] filePath — путь к файлу, который требуется зашифровать. • [out] encryptionData — криптографическая информация, выработанная при шифровании файла. Память для криптографической информации выделяется динамически и должна быть освобождена с помощью <code>ReleaseRawData(encryptionData)</code>
<code>EncryptionApiReturnCode (*DecryptFile) (uint32_t nodeId, const char* filePath, const VpnRawData* encryptionData)</code>	Расшифрование файла на ключах узла. • [in] nodeId — ID узла, для которого производится расшифрование. • [in] filePath — путь к файлу, который требуется расшифровать. • [in] encryptionData — криптографическая информация, выработанная при шифровании файла
<code>EncryptionApiReturnCode (*EncryptBlob) (uint32_t nodeId, const VpnRawData* sourceBlob, VpnRawData** destinationBlob, VpnRawData** encryptionData)</code>	Шифрование данных на ключах узла. • [in] nodeId — ID узла, для которого производится шифрование. • [in] sourceBlob — данные, которые требуется зашифровать. • [out] destinationBlob — зашифрованные данные. • [out] encryptionData — криптографическая информация, выработанная при шифровании данных. Память для зашифрованных данных выделяется динамически и должна быть освобождена с помощью <code>ReleaseRawData(destinationBlob)</code>. Память для криптографической информации выделяется динамически и должна быть освобождена с помощью <code>ReleaseRawData(encryptionData)</code>

Функция	Описание
<pre>EncryptionApiReturnCode (*DecryptBlob) (uint32_t nodeId, const VpnRawData* sourceBlob, VpnRawData** destinationBlob, const VpnRawData* encryptionData)</pre>	Расшифрование данных на ключах узла. • [in] nodeId — ID узла, для которого производится расшифрование. • [in] sourceBlob — данные, которые требуется расшифровать. • [out] destinationBlob — расшифрованные данные. • [in] encryptionData — криптографическая информация, выработанная при шифровании данных. Память для расшифрованных данных выделяется динамически и должна быть освобождена с помощью ReleaseRawData(destinationBlob)
<pre>EncryptionApiReturnCode (*ReleaseRawData) (VpnRawData* data)</pre>	Освобождение памяти, выделенной для массива данных. • [in] data — массив данных, память которого нужно освободить

Таблица 62. EncryptionApiReturnCode

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

Таблица 63. VpnRawData

Поле	Описание
const uint8_t* bytes	Массив данных
size_t size	Размер массива данных

Примеры

```
// Подключение заголовочных файлов
#include <vipnet/vpn_api.h>
#include <vipnet/cipher_api.h>
#include <vipnet/business_mail_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;
    }
    if ( BUSINESS_MAIL_API_VERSION != GetBusinessMailApiVersion() )
    {
        return 1;
    }

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

    // 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](#).