

ViPNet Client 4U for Linux SDK

Справочник разработчика

© АО «ИнфоТеКС», 2021

ФРКЕ.00239-01 33 01

Версия продукта 4.12

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»

127083, Москва, улица Мишина, д. 56, стр. 2, этаж 2, помещение IX, комната 29

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

Веб-сайт: infotecs.ru

Служба поддержки: hotline@infotecs.ru

Содержание

Введение.....	4
О документе.....	4
Соглашения документа.....	4
О программе	5
Комплект поставки.....	5
Начало работы с ViPNet Client 4U for Linux SDK	5
Функции SDK.....	6
Примеры	17
Обратная связь.....	19

Введение

О документе

Данный документ содержит инструкции по использованию SDK ViPNet Client 4U for Linux и описание функций этого SDK.

Документ предназначен для разработчиков, обеспечивающих взаимодействие программы ViPNet Client 4U for Linux с другими программами.

Соглашения документа

Ниже перечислены соглашения, принятые в этом документе для выделения информации.

Таблица 1. Обозначения, используемые в примечаниях




Обозначение	Описание
	Внимание! Указывает на обязательное для исполнения или следования действие или информацию.
	Примечание. Указывает на необязательное, но желательное для исполнения или следования действие или информацию.
	Совет. Содержит дополнительную информацию общего характера.

Таблица 2. Обозначения, используемые для выделения информации в тексте

Обозначение	Описание
Название	Название элемента интерфейса. Например, заголовок окна, название поля, кнопки или клавиши.
Клавиша+Клавиша	Сочетание клавиш. Чтобы использовать сочетание клавиш, следует нажать первую клавишу и, не отпуская ее, нажать вторую клавишу.
Меню > Подменю > Команда	Иерархическая последовательность элементов. Например, пункты меню или разделы на панели навигации.
Код	Имя файла, путь, фрагмент текстового файла (кода) или команда, выполняемая из командной строки.

О программе

Программа ViPNet Client 4U for Linux предназначена для защиты IP-трафика на компьютерах с ОС Linux путем шифрования IP-пакетов.

С помощью программы ViPNet Client 4U for Linux, установленной на компьютере, вы можете подключаться к сетевым узлам ViPNet или узлам, которые туннелируются координаторами ViPNet, по защищенным каналам и получать доступ к размещенным на этих узлах ресурсам: корпоративным веб-порталам, электронной почте, системе IP-телефонии, различным серверам и другим корпоративным сервисам.

Комплект поставки

Комплект поставки программы ViPNet Client 4U for Linux включает следующее:

- Пакеты установки в форматах DEB, RPM и IPK для консольной и графической версий программы ViPNet Client 4U for Linux.
- Библиотека ViPNet Client 4U for Linux SDK.
- Документы в формате PDF:
 - «ViPNet Client 4U for Linux. Руководство администратора».
 - «ViPNet Client 4U for Linux. Руководство пользователя».
 - «ViPNet Client 4U for Linux. Соответствие пакетов установки поддерживаемым платформам».
 - «ViPNet Client 4U for Linux. Установка на промышленные контроллеры».
 - «ViPNet Client 4U for Linux. Установка на контроллер Wago».
 - «ViPNet Client 4U for Linux. Лицензионные соглашения на компоненты сторонних производителей».
 - «ViPNet Client 4U for Linux SDK. Справочник разработчика».

Начало работы с ViPNet Client 4U for Linux SDK

Чтобы начать работу с ViPNet Client 4U for Linux SDK, выполните следующие действия:

- 1 Установите программу ViPNet Client 4U for Linux. При этом будут установлены следующие файлы ViPNet Client 4U for Linux SDK:

- Заголовочные файлы:
/usr/include/vipnet/cipher_api.h, /usr/include/vipnet/vpn_api.h.
- Библиотека /usr/lib/vipnet/libvpn_api.so.
- Контрольные суммы этих файлов.

2 Подключите заголовочные файлы в ваши исходные коды:

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

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

4 Проверьте, что версия SDK, с которой собирается программа ViPNet Client 4U for Linux, соответствует версии SDK, поставляемой вместе с программой ViPNet Client 4U for Linux:

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

5 Получите структуры с указателями на функции SDK:

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

Если в результате будут возвращены ненулевые указатели, то вы сможете вызывать [функции SDK](#) (на стр. 6).

Функции SDK

Таблица 3. ItcsVpnApi

Функция	Описание
ItcsVpnApi* GetVpnApi()	Получение указателя на структуру VPN API с проинициализированными указателями на функции.
uint32_t GetVpnApiVersion()	Получение номера текущей версии VPN API.
VpnApiReturnCode(* CollectEventlog) (const char *path, const char *psw, bool truncate)	Экспорт журнала критически важных событий в ZIP-архив. <ul style="list-style-type: none"> • [in] path — путь для сохранения файла архива. • [in] psw — пароль администратора сетевого узла ViPNet.

Функция	Описание
	<ul style="list-style-type: none"> [in] truncate — признак необходимости удаления журнала после экспорта.
VpnApiReturnCode(* CollectReport) (const char *path)	<p>Экспорт отчета о работе программы ViPNet Client 4U for Linux в ZIP-архив.</p> <ul style="list-style-type: none"> [in] path — путь для сохранения файла архива.
VpnApiReturnCode(* CollectTokens) (int (*cbk) (VpnTokenHandle))	<p>Получение списка подключенных и поддерживаемых внешних устройств (токенов).</p> <ul style="list-style-type: none"> [in] cbk — обработчик дескриптора токена; может вернуть не 0 для прерывания опроса.
VpnApiReturnCode(* DeleteKeys) ()	Удаление ключей ViPNet.
VpnApiReturnCode (*GetActiveCoordinator) (uint32_t *id)	<p>Получение идентификатора (ID) активного координатора (сервера IP-адресов).</p> <ul style="list-style-type: none"> [out] id — ID узла.
VpnApiReturnCode(* GetClientParam) (const char *key, uint32_t keyLength, char **outData, uint32_t *outDataLength)	<p>Получение конфигурационного параметра ViPNet Client 4U for Linux по ключу поиска.</p> <ul style="list-style-type: none"> [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. <p>Память для outData выделяется динамически и должна быть освобождена с помощью ReleaseClientParamData(outData).</p>
VpnApiReturnCode(*GetClientVersion) (char *version, size_t *size)	<p>Получить текущую версию ViPNet Client 4U for Linux.</p> <ul style="list-style-type: none"> [out] version — буфер для передачи версии. [in,out] size — размер буфера [in], размер записанных данных [out].
VpnApiReturnCode(*GetLicenseExpiration) (time_t *time)	<p>Получить дату и время истечения срока действия лицензии в секундах в формате POSIX.</p> <ul style="list-style-type: none"> [out] time — дата и время истечения срока действия лицензии в секундах.
VpnApiReturnCode (*GetLogLevel) (int32_t *logLevel)	<p>Получение текущего уровня протоколирования событий ViPNet Client 4U for Linux.</p> <ul style="list-style-type: none"> [out] logLevel — уровень протоколирования.
VpnApiReturnCode (*GetNodeInfo) (uint32_t id, VpnNodeInfo **nodeInfo)	<p>Получение информации об узле ViPNet по его идентификатору (ID).</p> <ul style="list-style-type: none"> [in] id — ID узла. [out] nodeInfo — информация об узле.

Функция	Описание
<pre>VpnApiReturnCode (*GetNodesIds) (uint32_t **nodeIds, uint32_t *size)</pre>	<p>Память для <code>nodeInfo</code> выделяется динамически и должна быть освобождена с помощью <code>ReleaseVpnNodeInfo (nodeInfo, 1)</code>.</p> <p>Получение списка идентификаторов (ID) узлов ViPNet.</p> <ul style="list-style-type: none"> [out] <code>nodeIds</code> — список ID узлов. [out] <code>size</code> — количество узлов. <p>Память для <code>nodeIds</code> выделяется динамически и должна быть освобождена с помощью <code>ReleaseVpnNodeIds (nodeIds, size)</code>.</p>
<pre>VpnApiReturnCode (*GetNodesInfo) (VpnNodeInfo **nodesInfo, uint32_t* size)</pre>	<p>Получение информации об узлах ViPNet.</p> <ul style="list-style-type: none"> [out] <code>nodesInfo</code> — информация об узлах ViPNet. [out] <code>size</code> — количество узлов. <p>Память для <code>nodesInfo</code> выделяется динамически и должна быть освобождена с помощью <code>ReleaseVpnNodeInfo (nodesInfo, size)</code>.</p>
<pre>VpnApiReturnCode (*GetNodesInfoByTask) (VpnNodeInfo **nodesInfo, size_t *size, uint32_t tasksMask)</pre>	<p>Получение информации об узлах ViPNet по маске ролей и свойств узла.</p> <ul style="list-style-type: none"> [out] <code>nodesInfo</code> — информация об узлах ViPNet. [out] <code>size</code> — количество узлов. [in] <code>taskMask</code> — битовая маска ролей и свойств узла. <p>Память для <code>nodesInfo</code> выделяется динамически и должна быть освобождена с помощью <code>ReleaseVpnNodeInfo (nodesInfo, size)</code>.</p>
<pre>VpnApiReturnCode (*GetOwnNodeInfo) (VpnNodeInfo ** nodeInfo)</pre>	<p>Получение информации о своем узле.</p> <ul style="list-style-type: none"> [out] <code>nodeInfo</code> — информация об узле. <p>Память для <code>nodeInfo</code> выделяется динамически и должна быть освобождена с помощью <code>ReleaseVpnNodeInfo (nodeInfo, 1)</code>.</p>
<pre>VpnApiReturnCode (* GetOwnPrivilegeLevel) (enum VpnPrivilege *privilege)</pre>	<p>Получение уровня полномочий своего узла.</p> <ul style="list-style-type: none"> [out] <code>privilege</code> — уровень полномочий.
<pre>VpnApiReturnCode (* GetTokenConfig) (int (*cbk) (const char *))</pre>	<p>Получить конфигурацию модулей PKCS#11 и статус их использования.</p> <ul style="list-style-type: none"> [in] <code>cbk</code> — обработчик информации, получает JSON-структуру в utf8. Возвращаемая структура имеет следующий формат: <pre>{ "global": { "paths": "<path1>:<path2>:...:<pathN>" }, "token01": {</pre>

Функция	Описание
<pre>VpnApiReturnCode(* GetTokenInfo) (VpnTokenHandle handle, VpnTokenInfo *info)</pre>	<pre> "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> <p>Получение параметров токена.</p> <ul style="list-style-type: none"> [in] handle — дескриптор токена, информацию о котором нужно получить. [out] info — указатель на структуру, в которую будет скопирована информация. <p>VpnTokenInfo содержит указатели на строки, эти объекты уничтожаются при вызове ReleaseTokenHandle.</p>
<pre>VpnApiReturnCode(* GetUserIdsForNode) (uint32_t nodeId, uint32_t **userIds, size_t *size)</pre>	<p>Получение списка ID пользователей, зарегистрированных на данном узле.</p> <ul style="list-style-type: none"> [in] nodeId — ID узла. [out] userIds — список ID пользователей. [out] size — количество пользователей. <p>Память для userIds выделяется динамически и должна быть освобождена с помощью ReleaseVpnUserIds (userIds, size).</p>
<pre>VpnApiReturnCode(* GetUserInfo) (uint32_t id, VpnUserInfo **userInfo)</pre>	<p>Получение информации о пользователе по его ID.</p> <ul style="list-style-type: none"> [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>	<p>Получение информации о пользователях.</p> <ul style="list-style-type: none"> [out] usersInfo — информация о пользователях. [out] size — количество пользователей.

Функция	Описание
	<ul style="list-style-type: none"> Память для <code>usersInfo</code> выделяется динамически и должна быть освобождена с помощью <code>ReleaseVpnUserInfo (userInfo, size)</code>. Структура <code>VpnUserInfo</code> имеет переменную длину.
<code>VpnApiReturnCode (*GetVpnStatus) (VpnStatus *vpnStatus)</code>	<p>Функция возвращает объект, описывающий текущее состояние VPN-соединения.</p> <ul style="list-style-type: none"> [out] <code>vpnStatus</code> — текущий статус VPN-соединения.
<code>VpnApiReturnCode(* InstallKeys) (const char *dstPath, const char *psw)</code>	<p>Установка ключей ViPNet.</p> <ul style="list-style-type: none"> [in] <code>dstPath</code> — путь к дистрибутиву ключей (*.dst), который будет установлен. [in] <code>psw</code> — пароль к дистрибутиву ключей.
<code>VpnApiReturnCode(* MoveKeysToToken) (VpnTokenHandle handle, const char *pin)</code>	<p>Перенос персонального ключа на внешнее устройство (токен) и смена типа аутентификации на «Персональный ключ на внешнем устройстве».</p> <ul style="list-style-type: none"> [in] <code>handle</code> — дескриптор токена, который должен быть использован для аутентификации. [in] <code>pin</code> — ПИН-код пользователя токена (немаскированный). <p>Для успешного выполнения команды VPN-соединение должно быть включено.</p>
<code>VpnApiReturnCode(* RegulationsCheck) (const char *psw)</code>	<p>Выполнение регламентного контроля целостности ViPNet Client 4U for Linux.</p> <ul style="list-style-type: none"> [in] <code>psw</code> — пароль администратора сетевого узла.
<code>void (* ReleaseClientParamData) (char *data)</code>	<p>Освобождение памяти, выделенной для информации о конфигурационном параметре ViPNet Client 4U for Linux.</p> <ul style="list-style-type: none"> [in] <code>data</code> — буфер, который необходимо освободить.
<code>VpnApiReturnCode(* ReleaseTokenHandle) (VpnTokenHandle handle)</code>	<p>Освобождение памяти, выделенной для дескриптора токена.</p> <ul style="list-style-type: none"> [in] <code>handle</code> — дескриптор токена. <p>Следует вызывать для каждого дескриптора, когда он перестаёт быть нужным (с учётом примечания в описании <code>GetTokenInfo</code>).</p>
<code>void (*ReleaseVpnNodeIds) (uint32_t *nodeIds, uint32_t size)</code>	<p>Освобождение памяти, выделенной для списка узлов ViPNet.</p> <ul style="list-style-type: none"> [in] <code>nodeIds</code> — массив ID узлов, который необходимо освободить. [in] <code>size</code> — количество узлов.
<code>void (*ReleaseVpnNodeInfo) (VpnNodeInfo</code>	<p>Освобождение памяти, выделенной для информации</p>

Функция	Описание
<code>*nodeInfo, uint32_t size)</code>	об узлах ViPNet. <ul style="list-style-type: none"> [in] <code>nodeInfo</code> — информация об узлах, память которой необходимо освободить. [in] <code>size</code> — количество узлов.
<code>void (* ReleaseVpnUserIds) (uint32_t *userIds, size_t size)</code>	Освобождение памяти, выделенной для списка пользователей. <ul style="list-style-type: none"> [in] <code>userIds</code> — массив ID пользователей, который необходимо освободить. [in] <code>size</code> — количество пользователей.
<code>void (* ReleaseVpnUserInfo) (VpnUserInfo *userInfo, size_t size)</code>	Освобождение памяти, выделенной для информации о пользователях. <ul style="list-style-type: none"> [in] <code>userInfo</code> — информация о пользователях, память которой необходимо освободить. [in] <code>size</code> — количество пользователей.
<code>VpnApiReturnCode (*SetLogLevel) (int32_t logLevel)</code>	Выставление уровня протоколирования событий ViPNet Client 4U for Linux. <ul style="list-style-type: none"> [in] <code>logLevel</code> — уровень протоколирования.
<code>VpnApiReturnCode(* SetToken) (VpnTokenHandle handle, const char *pin)</code>	Задание параметров токена, используемого для аутентификации. <ul style="list-style-type: none"> [in] <code>handle</code> — дескриптор токена, который должен быть использован для аутентификации. [in] <code>pin</code> — ПИН-код пользователя токена (немаскированный).
<code>VpnApiReturnCode(*StartVpn) (const char *psw)</code>	Включение VPN-соединения. <ul style="list-style-type: none"> [in] <code>psw</code> — пароль к дистрибутиву ключей.
<code>VpnApiReturnCode(*StopVpn) ()</code>	Выключение VPN-соединения.
<code>VpnApiReturnCode(*SubscribeForEvents) (VpnEventHandler eventHandler)</code>	Подписка на события VPN-соединения. <ul style="list-style-type: none"> [in] <code>eventHandler</code> — обработчик событий.
<code>void (*VpnEventHandler) (enum VpnEvent event)</code>	Прототип обработчика событий. <ul style="list-style-type: none"> [in] <code>event</code> — событие VPN-соединения.

Таблица 4. *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</code>	Зашифрование данных для узла.

Функция	Описание
<pre>inSize, uint8_t *out, size_t *outSize)</pre>	<ul style="list-style-type: none"> [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. <p>Размер входного буфера должен быть не меньше 16 байт, выходного — не меньше, чем размер входного плюс 12 байт.</p>
<pre>CipherApiReturnCode(* EncryptBlobEx) (uint32_t cipherSuite, uint32_t targetNodeId, uint8_t *in, size_t inSize, uint8_t *out, size_t *outSize)</pre>	<p>Зашифрование данных для узла с использованием указанного набора криптографических алгоритмов.</p> <ul style="list-style-type: none"> [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. <p>Размер входного буфера должен быть не меньше 16 байт, выходного — не меньше, чем размер входного плюс 16 байт.</p>
<pre>CipherApiReturnCode(* DecryptBlob) (uint32_t sourceNodeId, uint8_t *in, size_t inSize, uint8_t *out, size_t *outSize)</pre>	<p>Расшифрование данных от узла.</p> <ul style="list-style-type: none"> [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. <p>Размер входного буфера должен быть не меньше 28 байт, выходного — не меньше, чем размер входного минус 12 байт.</p>
<pre>CipherApiReturnCode(* DecryptBlobEx) (uint32_t cipherSuite, uint32_t sourceNodeId, uint8_t *in, size_t inSize, uint8_t *out, size_t *outSize)</pre>	<p>Расшифрование данных от узла с использованием указанного набора криптографических алгоритмов.</p> <ul style="list-style-type: none"> [in] cipherSuite — набор криптографических

Функция	Описание
	<p>алгоритмов.</p> <ul style="list-style-type: none"> • [in] <code>sourceNodeId</code> — ID узла, для которого производится расшифрование. • [in] <code>in</code> — буфер с входными данными. • [in] <code>inSize</code> — размер буфера с входными данными. • [out] <code>out</code> — буфер с расшифрованными данными. • [in,out] <code>outSize</code> — размер буфера с расшифрованными данными. <p>Размер входного буфера должен быть не меньше 28 байт, выходного — не меньше, чем размер входного минус 16 байт.</p>
<pre>CipherApiReturnCode(* IsCipherSuiteSupported) (uint32_t cipherSuite)</pre>	<p>Проверка, поддерживается ли указанный набор криптографических алгоритмов.</p> <ul style="list-style-type: none"> • [in] <code>cipherSuite</code> — набор криптографических алгоритмов.

Таблица 5. Ключи для получения параметров с помощью функции `GetClientParam`

Ключ	Описание
<pre>const char* const paramKeyNetworkNumber = "network.number"</pre>	Ключ для получения номера сети ViPNet.
<pre>const char* const paramNodeDeviceId = "node.deviceId"</pre>	Ключ для получения идентификатора (ID) устройства, который используется для отправки данных в систему мониторинга ViPNet NVS.
<pre>const char* const paramNvsAuthenticationId = "nvs.info/authentication.id"</pre>	Ключ для получения идентификатора авторизации в системе мониторинга ViPNet NVS.
<pre>const char* const paramNvsAuthenticationSecret = "nvs.info/authentication.secret"</pre>	Ключ для получения секрета авторизации в системе мониторинга ViPNet NVS.
<pre>const char* const paramNvsConnectionOpenUrls = "nvs.info/connection.openUrls"</pre>	Ключ для получения ссылки для доступа к системе мониторинга ViPNet NVS по открытой сети.
<pre>const char* const paramNvsConnectionVipnetUrls = "nvs.info/connection.vipnetUrls"</pre>	Ключ для получения ссылки для доступа к системе мониторинга ViPNet NVS по защищенной сети.
<pre>const char* const paramSecurityEncryptionMode = "node.security.encryption.mode"</pre>	Ключ для получения используемого типа шифрования.
<pre>const char* const paramSystemNetworkMonitoringOpenUrls = "system.monitoring.openUrls"</pre>	Ключ для получения ссылки для доступа к системе мониторинга по открытой сети.

Ключ	Описание
<code>const char* const paramSystemNetworkMonitoringVipnetUrl s = "system.monitoring.vipnetUrls"</code>	Ключ для получения ссылки для доступа к системе мониторинга по защищенной сети.

Таблица 6. *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 7. *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 8. *VpnStatus*

Поле	Описание
<code>bool isKeysInstalled</code>	Проверка, установлены ли ключи ViPNet.
<code>bool isVpnEnabled</code>	Проверка, включено ли VPN-соединение.

Таблица 9. *VpnNodeInfo*

Поле	Описание
<code>uint32_t id</code>	Идентификатор узла ViPNet (ViPNet Id).
<code>uint32_t ip</code>	IPv4-адрес узла в виде строки.
<code>char* name</code>	Название узла ViPNet.
<code>uint32_t tasksMask</code>	Битовая маска ролей и свойств узла.

Таблица 10. *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 11. *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* manufacturerID_</code>	Идентификатор (ID) производителя устройства.
<code>uint32_t maxPinLen_</code>	Максимальная длина ПИН-кода.
<code>uint32_t minPinLen_</code>	Минимальная длина ПИН-кода.
<code>const char* model_</code>	Модель устройства.
<code>const char* serialNumber_</code>	Серийный номер устройства.

Таблица 12. *VpnNodeTasksMask*

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

Таблица 13. *VpnTokenHandle*

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

Таблица 14. *VpnEvent*

Поле
Event_VpnServiceStarted
Event_VpnServiceStopped
Event_KeysInstalled
Event_KeysUninstalled
Event_VpnStarted
Event_VpnStopped
Event_NodesUpdated

Таблица 15. *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

Таблица 16. *VpnPrivilege*

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

Таблица 17. *VpnNodeTasksMaskAll*

Поле	Описание
VPN_NODE_TASKS_MASK_ALL	UINT32_MAX

Таблица 18. *CipherSuite*

Поле	Описание
csG28147_CFB	0
csG3412_MAGMA	1
csITCS_AES256	4
csG28147_CTR	5
csTFIPS	7
csG3412_KUZNYECHIK	14

Таблица 19. *CryptoResult*

Поле	Описание
0	SUCCESS
0x000A0001	BUFFER_TOO_SMALL
0x000A0002	BUFFER_IS_NULLPTR
0x000A0003	ENCRYPT_ERROR
0x000A0004	DECRYPT_ERROR
0x000A0005	KEY_NOT_FOUND_ERROR
0x000A0006	BUFFER_TOO_LARGE
0x000A0007	CIPHER_SUITE_NOT_AVAILABLE
0x000A000A	NOT_AVAILABLE

Таблица 20. *CipherApiVersion*

Поле	Описание
CIPHER_API_VERSION	2

Примеры

В данном разделе приведены примеры обращения к функциям SDK ViPNet Client 4U for Linux.

```

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

// 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();

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

// Зашифровать строку 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);

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

```

Обратная связь

Дополнительная информация

Сведения о продуктах и решениях ViPNet, распространенные вопросы и другая полезная информация собраны на сайте ИнфоТеКС:

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

Контактная информация

Если у вас есть вопросы, свяжитесь со специалистами ИнфоТеКС:

- 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.](#)
Телефон для клиентов с расширенной поддержкой: +7 (495) 737-6196.
- Отдел продаж: soft@infotecs.ru.

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](#).