# Мультипротокольные маршрутизаторы NSG

## Программное обеспечение NSG Linux

# *ui*TCP

## Система обеспечения бесперебойных соединений

Версия 0.33

Требуемая версия NSG Linux 1.0 build 4.1 или более поздняя

Обновлено 21.10.2011

Москва 2011

# АННОТАЦИЯ

Данный документ содержит руководство по настройке и применению программного обеспечения NSG *ui*TCP (автономная версия для NSG Linux 1.0 и выделенных серверов) для построения бесперебойных сетевых соединений. *ui*TCP является дополнительной компонентой программного обеспечения NSG Linux для мультипротокольных маршрутизаторов NSG. Руководство по настройке и применению собственно NSG Linux 1.0 содержится в отдельном документе NSG.

Реализация *ui*TCP в составе NSG Linux 2.0 является штатной компонентой програмного обеспечения и описана в Руководстве пользователя по NSG Linux 2.0, Часть 6.

**ВНИМАНИЕ** Продукция компании непрерывно совершенствуется, в связи с чем возможны изменения отдельных аппаратных и программных характеристик по сравнению с настоящим описанием.

Замечания и комментарии по документации NSG принимаются по адресу: doc@nsg.net.ru.



ООО «Эн–Эс–Джи»
Россия 105187 Москва
ул. Вольная, д.35
Тел./факс: (+7–495) 727–19–59 (многоканальный)

http://www.nsg.ru/
mailto:info@nsg.net.ru
mailto:sales@nsg.net.ru
mailto:support@nsg.net.ru

# § СОДЕРЖАНИЕ §

## *ui*TCP. Система обеспечения бесперебойных соединений

# §1. Введение

## §1.1. Назначение и общая характеристика системы

Технология ***ui*TCP** (Un-Interruptible TCP) предназначена для организации бесперебойных сеансов работы прикладного программного обеспечения при разрыве, восстановлении и переустановлении сетевых соединений. Технология разработана ООО "Эн–Эс–Джи" и представляет собой комплекс программных механизмов, основанный на существующих стандартах и технологиях построения сетей IP.

***ui*TCP** обеспечивает работу критически ответственных приложений, в том числе:
— программного обеспечения банкоматов;
— Интернет-приложений, требующих аутентификации пользователя в каждом сеансе работы;
— приложений для корпоративной информационной среды.

Классическая реализация сетевых технологий подразумевает, что при разрыве соединения на 1–3 уровнях протокольного стека (вплоть до уровня IP) происходит разрыв TCP-соединений и всех сеансов работы протоколов вышестоящих уровней, а затем повторное установление последовательно на всех уровнях, начиная с физического. Для датаграммных протоколов возможна потеря данных во время отсутствия связи, а после её восстановления — полное переустановление сеанса работы вышестоящего протокола (например, туннеля IPsec или PPTP, если после восстановления связи устройству назначен новый IP-адрес). В ряде специфических приложений такая ситуация может приводить к крайне нежелательным последствиям, таким как полная перезагрузка банкомата, потеря всей работы пользователя в аварийно завершённом сеансе (например, данных, введённых в формы) и т.п. Технология ***ui*TCP** позволяет поддерживать непрерывный обмен данными при неоднократных переключениях на нижележащих уровнях, в том числе:
— при переходе на резервные маршруты и обратно;
— при восстановлении соединения с иными IP-адресами.

При этом потеря и восстановление связи на 1–3 уровнях производится прозрачно для программного обеспечения вышестоящих уровней и отражается на его работе только в виде увеличенных задержек.

Механизм ***ui*TCP** способен работать через любые сети и каналы связи, с любыми IP-адресами клиента (статическими или динамическими, глобальными или приватными). Он универсально применим ко всем типам сетей, со всеми типами среды передачи и скоростями, которые поддерживаются маршрутизаторами NSG, в том числе:

— ЛС Ethernet сторонних организаций;
— городские сети Fiber Ethernet;
— системы широкополосного доступа (ADSL, кабельные модемы и др.) с локальным интерфейсом Ethernet;
— сотовые сети всех существующих стандартов (GSM, CDMA, 3G);
— коммутируемые модемные линии;
— традиционные синхронные каналы WAN: serial, *x*DSL, E1 и др.

Помимо этого, сотовые интерфейсы GSM и 3G могут представлять до 4 альтернативных каналов связи в одном аппаратном модуле:

— через двух различных операторов (для модулей с двумя SIM-картами);
— в пакетном (GPRS) и канальном (CSD, он же GSM data) режиме передачи данных.

На клиентской стороне ***ui*TCP** одинаково пригоден для подключения терминального оборудования любого типа, в т.ч. пользовательских ПК, банкоматов IP-over-Ethernet и X.25, киосков самообслуживания с протоколом IP-over-PPP, POS-терминалов без встроенных сетевых протоколов и т.п. При этом на удалённой площадке может располагаться как одно устройство, так и целый офис, батарея POS-терминалов, или локальный модемный пул.

Программный комплекс ***ui*TCP** может использоваться на всех продуктах NSG под управлением NSG Linux. В частности, для построения крупномасштабных систем используются высокопроизводительные коммуникационные шлюзы NSG–1000/GW.

Автономная версия *ui*TCP является дополнительной компонентой программного обеспечения NSG Linux 1.0 и предназначена для мультипротокольных маршрутизаторов под управлением NSG Linux 1.0, а также выделенных серверов. Настройка этой версии описана в данном документе.

Реализация *ui*TCP в составе NSG Linux 2.0 является штатной компонентой програмного обеспечения и описана в Руководстве пользователя по NSG Linux 2.0, Часть 6.

## §1.2. Основные функциональные возможности *ui*TCP

Основные функции и механизмы *ui*TCP включают в себя:

— Поддержание постоянного сеанса обмена прикладными данными независимо от состояния каналов связи и переключений между ними.

— Неограниченное число каналов связи связи произвольных типов, выбираемых в порядке их приоритета или по кругу.

— Постоянный мониторинг наличия связи между устройствами.

— Переключение с основного канала на резервный(-е), с GPRS на CSD и с основного GSM-оператора на резервного, с сохранением прежнего IP-адреса или его изменением.

— Автоматическое возвращение на основной (или более приоритетный) канал при восстановлении его работоспособности, выполняемое в периоды неактивности пользователя.

— Передачу различных типов IP-трафика, в том числе:
  — Данных из прикладных TCP-соединений пользователя с заданным номером порта
  — Данных из прикладных UDP-потоков пользователя с заданным номером порта
  — Пакетов заданных датаграммных протоколов (UDP, IPsec и др.)
  — Произвольных IP-пакетов

— Доступность терминального оборудования, находящегося в любых типах сетей (Интернет, внутренние сети поставщиков услуг Интернет, сети сторонних организаций) с любыми IP-адресами (глобальными или приватными через NAT, динамическими или статическими).

— Установление прикладных сеансов обмена данными (как TCP, так и датаграммных) по инициативе любой из сторон.

## §1.3. Расширенные функциональные возможности *ui*TCP

Дополнительные возможности, предоставляемые *ui*TCP, включают:

— Прохождение любых типов и реализаций NAT как на выходе из сети поставщика услуг Интернет, так и на входе в сеть процессингового центра. Протокол TCP передаётся через NAT всегда (в отличие от, например, GRE или IPsec). Если поставщик услуг фильтрует трафик по определённым протоколам и портам TCP, то для работы *ui*TCP могут быть намеренно назначены как общеупотребительные номера портов TCP для стандартных служб (25, 80 и т.п.), так и уникальные специфические номера портов.

— NAT для локальных адресов на обеих сторонах. Принимая входящий сеанс обмена данными, отвечающая сторона *ui*TCP может инициировать следующую в цепочке сессию как с исходными IP-адресами и номерами портов, так и с изменёнными. На практике это означает, например, что при массовой инсталляции все банкоматы могут быть настроены совершенно одинаково, равно как и локальная сторона устройств NSG. Различение будет производиться уже на стороне сервера, где эти соединения сходятся в одну точку. Сервер различает входящие пакеты по уникальному имени клиента и может назначать им заданные IP-адрес источника, IP-адрес назначения и порт TCP назначения.

— Систему безопасности на основе SSL, функционально эквивалентную STunnel, OpenVPN, HTTPS и другим методам защиты данных на протокольном уровне, включающую в себя:
  — Асимметричное шифрование пользовательских данных (или пакетов) с длиной ключа до 2048 бит.
  — Взаимную аутентификацию сторон с использованием сертификатов X.509.

— Горячее резервирование сервера. Клиентам могут быть указаны не только резервные каналы связи, но и резервные сервера, которые могут располагаться совершенно в других сетях и помещениях, нежели основной. Более того, система предусматривает механизм принудительного "мягкого" перевода клиентов на другой сервер по мере завершения текущих транзакций. После того, как все клиенты ушли с сервера, он может быть безопасно остановлен.

— Агрегирование нескольких каналов связи в одно соединение с увеличенной пропускной способностью.

— Централизованный мониторинг и управление клиентскими устройствами *ui*TCP, в т.ч. автоматизированный сбор статистики, обновление программного обеспечения, обновление сертификатов.

— Web-управление и мониторинг текущего состояния каналов. Сервер системы оснащён Web-интерфейсом, с помощью которого дежурный оператор в банке или процессинговом центре имеет возможность наблюдать текущее состояние клиентов, статистику их работы, распределение активности между основным и резервным(и) каналами связи, установленные TCP-сессии и т.п. Он также может рестартовать отдельные интерфейсы или клиентское устройство целиком, обновлять их программное обеспечение и т.п.

— Web-интерфейс для настройки *ui*TCP. При наличии установленного туннеля через Web-интерфейс сервера может осуществляться также настройка клиента.

— Вывод журнала с различной степенью детализации в локальный файл или в SQL-совместимую базу данных.

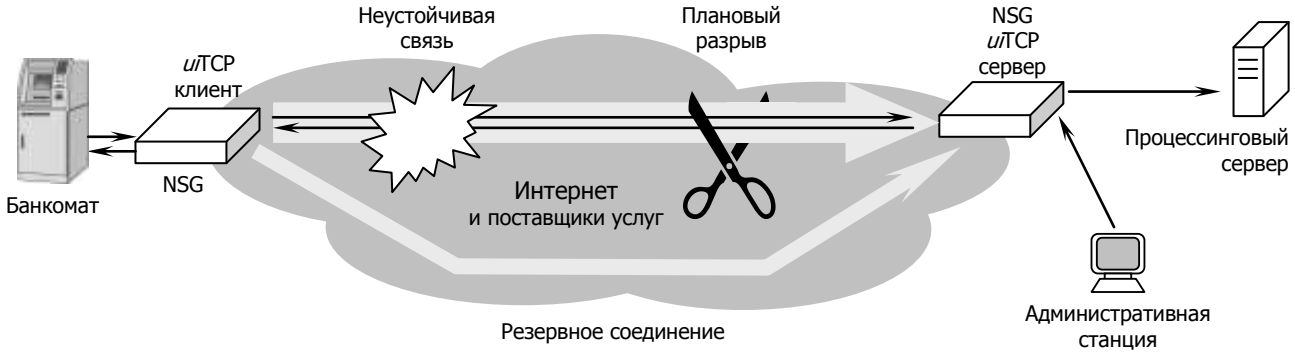
— Встроенное управление клиентами посредством SMS при отсутствии других каналов связи.

## §1.4. Принципы работы *ui*TCP

Работа *ui*TCP основана на организации собственного TCP-соединения между клиентом и сервером *ui*TCP и расширенного управления передачей данных по нему. Способы использования этого соединения зависят от характера передаваемого трафика:

**При передаче трафика TCP** *ui*TCP представляет собой TCP-прокси, или иначе говоря, подставной хост, работающий между клиентом и сервером прикладного решения.

Предположим, что в исходном случае прикладной клиент (например, банкомат) устанавливает TCP-соединение с сервером (процессинговым хостом и т.п.). При использовании *ui*TCP банкомат устанавливает TCP-сессию, вместо реального процессингового сервера, к устройству NSG, расположенному в непосредственной близости от него (как правило, в самом банкомате). Это устройство играет роль *клиента ui*TCP и может оснащаться широким ассортиментом фиксированных и сменных портов для различной среды передачи.




Программное обеспечение клиентской части отвечает за выбор работоспособного канала (из нескольких возможных) и устанавливает следующую TCP-сессию с *сервером ui*TCP. Сервер *ui*TCP устанавливается непосредственно в процессинговом центре или в головном офисе банка — там, откуда уже можно гарантировать 100% доступ к процессинговому хосту. Он принимает входящие TCP-сессии от клиентов и устанавливает заключительную сессию к хосту. Таким образом, данные прикладных протоколов передаются по эстафете из трёх TCP-сессий, где первая и третья проходят по гарантированно надёжной среде (например, по локальной сети), а бесперебойная работа второй обеспечивается собственно средствами *ui*TCP.

**При передаче трафика датаграммных протоколов** (например, UDP) устройство *ui*TCP принимает пакеты от прикладного клиента, имеющие определённый номер порта назначения (для UDP) или определённый номер протокола четвёртого уровня (для остальных протоколов, например, GRE — 47). Эти пакеты инкапсулируются в TCP и передаются на другую сторону, причём заголовок 4 уровня сохраняется во всех случаях, заголовок IP — сохраняется или не сохраняется в зависимости от сделанных настроек. На сервере заголовок TCP удаляется и пакет передаётся в сеть назначения либо в исходном виде (если заголовок IP был сохранён), либо с вновь сформированным заголовком IP. Таким образом, в данном случае *ui*TCP функционирует в качестве туннеля.

**При передаче произвольного трафика IP** *ui*TCP настраивается в виде виртуального IP-интерфейса, на который могут маршрутизироваться IP-пакеты обычным образом. Эти пакеты инкапсулируются в TCP целиком (включая заголовок IP и все заголовки вышестоящих уровней) и передаются на другую сторону. Таким образом, в этом случае *ui*TCP функционирует также в качестве туннеля.

Для общности, соединение между клиентом и сервером *ui*TCP во всех случаях будет называться *туннелем*.

## §1.5. Механизм обеспечения бесперебойности

При старте клиент *ui*TCP немедленно устанавливает со своей стороны TCP-соединение (туннель) с сервером и поддерживает его в рабочем состоянии. Внутри этого туннеля в ходе работы может устанавливаться неограниченное число пользовательских соединений, инициируемых любой из сторон.

Клиент *ui*TCP постоянно контролирует работоспособность текущего рабочего канала связи. При наличии пользовательского трафика для этого используется стандартный контрольный механизм TCP. В отсутствие данных клиент регулярно посылает пакеты *keepalive*, которые пересылаются поверх IP. В отличие от средств 1–2 уровней, этот механизм позволяет удостовериться в работоспособности всей сети вплоть до сервера *ui*TCP — т.е. диагностировать отказы не только ближайшего звена (например, GPRS-соединения), но и канала связи где-либо между вышестоящими операторами.

При отсутствии ответа на заданное число запросов *keepalive* подряд соединение считается неработоспособным и клиент переходит на следующий по приоритету или по очереди канал связи. При этом локальные TCP-сессии на обеих сторонах и сессии прикладных протоколов сохраняются. При уходе со старого

канала может быть выполнен заданный сценарий, например, рестарт модуля GPRS, перенастройка программного обеспечения для работы с другой SIM-картой, или изменения в таблице маршрутизации.

Новый или восстановленный канал связи может иметь как те же самые IP-адреса, так и новые (назначенные оператором динамически, или полученные у нового оператора). После восстановления связи клиент и сервер *ui*TCP синхронизируют свои входные и выходные очереди и восстанавливают последовательность пакетов. Все данные, отправленные уровнем TCP с одной и с другой стороны, при этом гарантированно доходят до адресата.

Если каналы связи неравнозначны, то при достаточно длительном времени неактивности клиент *ui*TCP проверяет работоспособность более приоритетных каналов и по возможности переходит на них. Однако при наличии пользовательского трафика эта проверка, по умолчанию, не начинается, т.е. система дожидается завершения текущей транзакции по каналу, работающему в данный момент.

Если восстановить соединение между клиентом и сервером не удаётся ни по одному из каналов связи за некоторое конечное время, то туннель считается разорванным, данные, оставшиеся в выходных очередях, теряются, а локальные TCP-соединения с прикладными хостами на стороне клиента и на стороне сервера разрываются. После этого туннель может быть только инициализирован заново, и все прикладные сеансы обмена данными начинаются с нуля.

## §1.6. Симметрия и асимметрия в *ui*TCP

Система *ui*TCP имеет асимметричную архитектуру "клиент-сервер", причём один сервер может обслуживать произвольное число клиентов (ограниченное только его производительностью). Однако эта асимметрия относится только к процедуре установления, поддержания, контроля и восстановления соединений между ними:

— *Клиент* инициирует создание туннеля, следит за его работоспособностью (по наличию ответов *keepalive*), в случае необходимости пытается восстановить соединение по альтернативному каналу связи, по мере возможности пытается переходить на более приоритетные каналы связи. Максимально разрешённое время отсутствия связи определяется заданными значениями таймаутов и числа попыток. Если восстановить связь с сервером за это время не удаётся, клиент считает туннель разорванным безвозвратно.

— *Сервер* ожидает входящие соединения от клиентов и следит за их активностью по наличию входящих пакетов от клиента: пользовательских данных, подтверждений TCP о приёме, или *keepalive* при отсутствии данных. При установлении туннеля сервер получает от клиента информацию о его таймаутах и числе попыток и, исходя из этого, вычисляет максимально возможное время неактивности клиента; по истечении этого времени туннель объявляется разорванным. Сервер также осуществляет служебные операции (управление и мониторинг, хранение и передачу новых версий *ui*TCP, и т.п.).

Применительно к передаче пользовательских данных, система является симметричной по отношению к серверу и клиенту. Любые сеансы обмена данными (как TCP, так и датаграммные) могут быть инициированы прикладным хостом с любой стороны туннеля и адресованы хосту на противоположной стороне. В этом отношении следует говорить не о паре "клиент — сервер", а о паре "вызывающая сторона — отвечающая сторона". Эти два взаимодействия никак не связаны друг с другом и могут как совпадать, так и быть противоположными. Например, если *ui*TCP используется для бесперебойного управления удалёнными площадками, то вызывающей стороной будет являться, скорее всего, сервер *ui*TCP (в сети центрального узла управления), а отвечающей — клиент *ui*TCP (на удалённой площадке).

## §1.7. Многотуннельные и многоканальные режимы

Одно клиентское устройство *ui*TCP может поддерживать одновременно несколько различных туннелей к одном или к нескольким серверам. Например, на одной площадке может быть расположено несколько банкоматов различных банков, каждый из которых обращается к своему процессинговому центру. Для каждого из них может быть создан отдельный туннель *ui*TCP, трафик которого изолирован от остальных.

Соединение клиента с одним и тем же сервером по нескольким доступным каналам связи, через одного или нескольких различных операторов, позволяет выполнять балансировку трафика, либо агрегацию каналов для достижения максимально возможной пропускной способности. Такой режим работы возможен только для датаграммных туннелей, однако, с точки зрения применения системы *ui*TCP, это не ограничивает общности, поскольку трафик протоколов, ориентированных на соединения (таких, как TCP), возможно передавать как произвольные IP-пакеты.

**ПРИМЕЧАНИЕ** Максимальная пропускная способность многоканального соединения выше, чем одного канала, но любом случае ниже, чем арифметическая сумма пропускной способности всех используемых каналов. Это принципиальное ограничение, связанное с внутренними алгоритмами работы *ui*TCP.

# §2. Общие сведения об установке и настройке *ui*TCP

## §2.1. Установка и обновление *ui*TCP в NSG Linux 1.0

Программное обеспечение *ui*TCP поставляется в виде дополнительного модуля (NSG plug-in) к основному программному обеспечению NSG Linux 1.0. *ui*TCP устанавливается на устройства NSG, как правило, по заказу в заводских условиях. Если *ui*TCP поставлено заказчику отдельно от оборудования (например, в порядке модернизации существующей системы передачи данных на основе оборудования NSG), то его установка выполняется в следующем порядке:

1. Убедиться, что на устройстве NSG установлена текущая требуемая версия ПО NSG Linux, или более поздняя. При необходимости — обновить NSG Linux на серверном и на всех клиентских устройствах. Процедура обновления NSG Linux описана в документе NSG: *Мультипротокольные маршрутизаторы NSG. Программное обеспечение NSG Linux. Руководство пользователя. Часть 1.*
2. Поместить дистрибутивный файл uitcp-X.Y (где X.Y — номер версии) на сервере TFTP, FTP, HTTP или NFS, доступном с данного устройства по сети.
3. Войти в командную оболочку ОС Linux (с помощью команды start-shell, либо войти в систему как root).
4. Загрузить на устройство файл uitcp-X.Y посредством tftp, ftpget, wget или nfs, соответственно, присвоить ему права исполняемого файла и запустить на исполнение. Пример:

```
cd /tmp
tftp -g 123.45.67.89 -r uitcp-1.2
chmod +x uitcp-1.2
./uitcp-1.2
```

5. Сохранить сделанные изменения:

```
savecfg
```

Установка специализированной сборки Linux и *ui*TCP на коммуникационные шлюзы с архитектурой x86 производится только специалистами компании.

**ВНИМАНИЕ** В NSG Linux 2.0 система *ui*TCP является штатной компонентой программного обеспечения и не требует установки пользователем ни при каких условиях. Настройка *ui*TCP производится в узле меню .tunnel.uitcp .

## §2.2. Запуск, остановка и рестарт служб *ui*TCP

После того, как система *ui*TCP установлена на устройство и настроена с обеих сторон, её необходимо запустить на исполнение командой:

```
/etc/uitcp/bin/uitcp&
```

и с помощью команды ps убедиться, что служба uitcp успешно стартовала.

Чтобы остановить работу *ui*TCP, следует определить её номер процесса (PID) с помощью команды ps, и удалить этот процесс с помощью команды kill.

Если настройка производится вручную, то после изменения конфигурации необходимо сохранить эти изменения (savecfg) и рестартовать службу uitcp (остановить и запустить снова).

После того, как система успешно настроена, следует включить uitcp в число процессов, запускаемых автоматически при старте системы. Для этого с помощью встроенного редактора nano отредактировать файл /etc/inittab, вставив в него следующие строки:

```
null::respawn:/etc/uitcp/bin/uitcp
null::respawn:/usr/sbin/httpd -c /etc/uitcp/bin/uitcp_web/http.uitcp.conf -h /etc/uitcp/bin/uitcp_web
```

Здесь первая строка запускает собственно *ui*TCP. Вторая запускает Web-сервер для управления системой и требуется, как правило, только на серверном устройстве.

Редактор nano имеет встроенную подсказку. Более подробная информация об использовании nano представлена в виде страниц руководств (*man pages*), доступных на любой настольной Linux-системе или в Интернет.

После этого необходимо сохранить изменения (savecfg), перезагрузить систему (reboot), снова войти в командную оболочку ОС Linux и с помощью ps убедиться, что процессы uitcp и httpd (если требуется) успешно стартовали.

Если служба uitcp включена в файл inittab, как описано выше, то при остановке она немедленно рестартует автоматически.

## §2.3. Настройки *ui*TCP и настройки основного программного обеспечения

*ui*TCP представляет собой надстройку над основным программным обеспечением NSG Linux и функционирует, используя его базовые возможности. По этой причине настройка любого устройства в системе *ui*TCP начинается с настройки соединений WAN на уровне основного ПО. В особенности это относится к клиентским устройствам и к сеансовым соединениям PPP по коммутируемым телефонным линиям, сотовым сетям GPRS и CDMA и т.п. Только после того, как обеспечено прохождение *ping* между клиентом и сервером по каждому из каналов связи, следует приступать к настойке *ui*TCP.

Исключением являются два объекта конфигурации в основном ПО клиентских устройств:

— маршруты на сервер(а) *ui*TCP
— приоритеты основной/резервной SIM-карт (для модулей GPRS/EDGE/3G с двумя SIM-картами)

*ui*TCP манипулирует этими объектами самостоятельно, чтобы направить трафик в выбранный канал связи. После первоначальной настройки основного ПО и проверки работоспособности каждого из каналов явно указанные маршруты на сервера рекомендуется удалить, чтобы избежать возможных неоднозначностей. Приоритеты SIM-карт необходимо иметь в виду, чтобы обеспечить правильное начало работы системы (подробнее см. п.3.2). Рекомендуется оставить их с настройками по умолчанию: prio main 1 aux none.

Маршруты между локальной сетью клиента и локальной сетью сервера (например, между банкоматом и процессинговым сервером, или административной рабочей станцией и банкоматом), указывать не требуется. Вместо них указывается localListener (для TCP-соединений) или socket (для датаграмм UDP и Raw IP), принимающий этот трафик из локальной сети и направляющий его в туннель.

**ВНИМАНИЕ** В некоторых случаях (особенно при ручном редактировании файла конфигурации) сделанные пользователем настройки *ui*TCP могут оказаться некорректными и система будет неработоспособна. Если при этом запуск *ui*TCP уже вставлен в файл inittab, то возможна ситуация, когда *ui*TCP полностью блокирует работу устройства и доступ к нему. Для выхода из такого положения приходится полностью удалить пользовательскую конфигурацию (как основного ПО, так и *ui*TCP) на этапе загрузки устройства.
Поэтому перед началом настройки *ui*TCP настоятельно рекомендуется сохранить резервную копию конфигурации основного ПО, полученной на этот момент, чтобы её можно было быстро восстановить. Способы резервирования и восстановления конфигурации описаны в документе NSG: *Мультипротокольные маршрутизаторы NSG. Программное обеспечение NSG Linux. Руководство пользователя. Часть 1.*

Удалённое управление клиентскими устройствами может производиться, в частности, по самому туннелю *ui*TCP. Однако если туннель не может быть установлен по каким-либо причинам (например, из-за истечения срока действия клиентского сертификата), то устройство окажется недоступно. По этой причине целесообразно предусмотреть отдельный безопасный канал для управления, например, по SSH (если устройство имеет глобальный IP-адрес) или по PPTP (относительно простой протокол, успешно проходящий через NAT из приватных сетей поставщиков услуг).

## §2.4. Обновление *ui*TCP

Для обновления *uiTCP* вручную на работающем устройстве можно использовать специальную утилиту, запускаемую из командной оболочки ОС Linux:

```
/etc/uitcp/bin/tools/uitcp-update-tftp сервер файл
```

где *сервер* — имя TFTP-сервера, на котором находится файл с новой версией, а *файл* — имя файла (вида uitcp-X.Z). Утилита загружает указанный файл, распаковывает его, сохраняет копию в предназначенной для этого директории (см. ниже параметр pluginFolder, по умолчанию /etc/uitcp/plugins) и сохраняет конфигурацию. По завершении работы необходимо рестартовать процесс uitcp (см. п.2.2).

**ВНИМАНИЕ** Директория pluginFolder автоматически не создаётся. При установке uitcp на сервере необходимо создать её вручную.

В работающей системе необходимо установить новый файл uitcp-X.Z только на сервере. Обновление *ui*TCP на клиентах выполняется автоматически. (Если обновление выполняется вручную, то необходимо также сохранить копию дистрибутивного файла uitcp-X.Z в директории pluginFolder.)

Встроенные средства обновления *ui*TCP проверяют только совпадение номеров текущей и загружаемой версий, но не сравнивают их по принципу больше/меньше. Таким образом, при необходимости возможно не только автоматическое обновление на более позднюю версию в масштабе всей системы, но и откат на более раннюю.

**ВНИМАНИЕ** Если для работы новой версии *ui*TCP требуется обновление основного ПО NSG Linux (это указывается в сопроводительной документации), то необходимо сначала обновить основное ПО на сервере и на всех клиентах, а затем обновить *ui*TCP на сервере.

**ПРИМЕЧАНИЕ** Утилита автоматического обновления *uiTCP* использует, по умолчанию, порт TCP 50003 на обеих сторонах туннеля. Данный порт может быть изменён пользователем, однако без настоятельной необходимости не рекомендуется использовать порты с номерами вида 500*xx*. Эти порты могут быть заняты различными внутренними службами NSG Linux.

## §2.5. Способы настройки *ui*TCP

Для настройки и управления *ui*TCP могут использоваться следующие инструменты:

1. Текстовый файл конфигурации — по умолчанию, /etc/uitcp/config. Может быть отредактирован пользователем:

   nano /etc/uitcp/config

   но этот способ настройки крайне нежелателен ввиду возможности совершения ошибок. После внесения изменений в конфигурацию необходимо сохранить их и удалить текущий процесс uitcp, чтобы он стартовал заново. (Либо рестартовать его вручную, если он не внесён в inittab).

2. Утилита uish. Запускается из командной оболочки ОС Linux:

   /etc/uitcp/bin/tools/uish

   и работает в текстовом режиме.

3. Web-интерфейс. Является основным инструментом для большинства пользователей. Имя и пароль для входа через Web, установленные по умолчанию — nsg nsg.

При настройке с помощью uish или Web-интерфейса достаточно выполнить команду Apply, которая применяет сделанные изменения "на лету".

**ВНИМАНИЕ** Если в системе включён Web-интерфейс, то перед вводом в эксплуатацию необходимо изменить пароль для входа в него.

Во всех трёх случаях конфигурация системы имеет вид иерархического дерева. Настройка системы состоит в настройке соответствующих ветвей и листьев этого дерева. Структура дерева, англоязычные названия объектов и значения параметров одинаковы во всех трёх случаях, независимо от выбранного пользовательского интерфейса.

В Web-интерфейсе, по усмотрению пользователя, может быть включена локализация. Кроме того, Web-интерфейс включает в себя инструменты для управления пользователями, сертификатами, системными операциями и для экспорта конфигурационных файлов на вновь устанавливаемые клиентские устройства.

Текстовый файл конфигурации имеет жестко определённый синтаксис, в котором используются: фигурные скобки ({...}) для описания иерархии объектов; разделитель "запятая" (,) для перечисления объектов одного уровня. При работе с текстовой утилитой конфигурации или Web-интерфейсом пользователю следует заботиться только о настройке объектов по существу, а соблюдение синтаксиса возложено на соответствующие программные компоненты. Поэтому ниже в данном документе эти обозначения будут иногда опускаться, чтобы не загромождать изложение.

При ручном редактировании конфигурационного файла необходимо строго следить за соблюдением этого синтаксиса, в противном случае конфигурация окажется неработоспособной; настоятельно рекомендуется соблюдать принятые отступы для упрощения проверки.

## §2.6. Общая структура конфигурационного дерева

Дерево конфигурации *ui*TCP содержит следующие основные ветви:

```
корень дерева
    mode "client" | "server"          // режим работы устройства
    clients                           // только для сервера: описания клиентов
        клиент1
        клиент2
        .........
    tunnelListener                    // только для сервера: IP-адреса и порты,
        общие параметры               // открытые со стороны сети для входящих туннелей
        tunnelListener1
        tunnelListener2
        .........
    tunnel                            // только для клиента: описание туннеля
    tunnels                           // только для клиента: описание дополнительных туннелей
    multiTunnelSocket                 // описание многоканального соединения
    ssl                               // общие настройки безопасности для всех туннелей
    nat                               // настройки NAT для соединений, входящих со стороны сети
        правило1                      // (общие для всех туннелей)
        правило2
        .........
    sys                               // общесистемные настройки
    log                               // настройки системного журнала
    show                              // команды для просмотра текущего состояния,
                                      // в конфигурации не сохраняются
```

Если устройство работает в режиме сервера, то в дереве конфигурации присутствуют два специфических узла. Узел tunnelListener определяет IP-адреса и порты, на которых сервер ожидает входящие запросы на установление туннелей. Узел clients содержит описания клиентов. Описание клиента содержит следующие объекты:

```
    имя                               // уникальное имя клиента в системе
        ssl                           // индивидуальные настройки безопасности для данного туннеля
        localListener                 // IP-адреса и TCP порты, на которых ожидаются исходящие
            localListener1            // соединения (от локальных хостов в сети сервера
            localListener2            // к удалённым хостам в сети клиента)
            .........
        socket                        // IP-адреса и сокеты (UDP или Raw IP), на которых
            socket1                   // ожидаются исходящие датаграммы (от локальных хостов
            socket2                   // в сети сервера к удалённым хостам в сети клиента)
            .........
        nat                           // настройки NAT для соединений, входящих со стороны сети
            правило1                  // (индивидуальные для данного туннеля)
            правило2
            .........
```

Число клиентов в системе программно не ограничено. Имена клиентов, определённые на сервере, должны совпадать с именами, указанными в описаниях туннелей на клиентских устройствах (см. ниже).

Если устройство работает в качестве клиента, то на нём определяется туннель в виде следующей иерархической структуры:

```
tunnel
    name                            // уникальное имя туннеля в системе
    общие параметры туннеля         // IP-адрес и порт сервера, таймауты и т.п.
    ssl                             // индивидуальные настройки безопасности для данного туннеля
    links                           // описания индивидуальных каналов связи
        link1
        link2
        .........
    localListener                   // IP-адреса и TCP порты, на которых ожидаются исходящие
        localListener1              // соединения (от локальных хостов в сети клиента
        localListener2              // к удалённым хостам в сети сервера)
        .........
    socket                          // IP-адреса и сокеты (UDP или Raw IP), на которых
        socket1                     // ожидаются исходящие датаграммы (от локальных хостов
        socket2                     // в сети клиента к удалённым хостам в сети сервера)
        .........
    nat                             // настройки NAT для соединений, входящих со стороны сети
        правило1                    // (индивидуальные для данного туннеля)
        правило2
        .........
```

Как можно видеть, объекты ssl, localListener, socket и nat имеют одинаковый смысл и для описания клиента на сервере, и для описания туннеля на клиенте. В этом проявляется симметричная сущность туннеля *ui*TCP: будучи установленным, он передаёт данные одинаково в обоих направлениях.

Несимметричными являются только параметры, определяющие процедуру установления самого туннеля. На клиенте это узел links и группа параметров туннеля, общих для всех каналов связи, а на сервере — узел tunnelListener.

Ряд параметров (ssl, nat и др.) могут быть определены как внутри некоторого конкретного объекта (клиента, туннеля, канала связи), так и на уровне выше — параллельно с описанием всей группы таких объектов. В этом случае, параметры, заданные внутри объекта, имеют приоритет. Если некоторый параметр не определён внутри объекта, тогда используется соответствующий параметр из общего набора. Допускается определять часть параметров внутри объекта, а часть — общими настройками, и сочетать их произвольным образом индивидуально для каждого из объектов. Например, для разных клиентов (на сервере) или каналов связи (на клиенте) могут использоваться разные сертификаты устройств, но общий корневой сертификат.

Если на клиенте требуется определить несколько туннелей (например, к разным серверам, или для включения их в состав многоканального соединения), то дополнительные туннели создаются в узле tunnels . Единственное формальное отличие для них состоит в том, что имя каждого туннеля является именем узла конфигурации:

```
tunnel
    имя
        общие параметры
        ssl
        links
        localListener
        socket
        nat
```

По умолчанию, большинство параметров в конфигурации отсутствуют (формально — имеют значение nil). Большинство параметров представляют собой текстовые строки и должны обязательно вводиться в кавычках (одинарных либо двойных). В том числе, все IP-адреса вводятся только в десятичной дотовой нотации, в кавычках. Без кавычек вводятся только:

— Порты TCP и UDP Задаются номерами от 0 до 65535.

— Другие числа Таймауты (в секундах), количество попыток и т.п. — целочисленные параметры.

— true и false Булевские значения.

## §2.7. Утилита uish

Утилита uish (*ui*TCP shell) предназначена для настройки *ui*TCP в текстовом консольном режиме. Программа представляет собой интерактивную командную оболочку для автоматизированного редактирования файла конфигурации.

```
# uish

NSG Device Configurator,  Copyright © 2008-2009, nsg.ru
Scheme file - /opt/uitcp/uitcpScheme.lua
Config file - /etc/uitcp/config
(type _help for usage information)

uitcp>
```

Внутри uish используется иерархическое меню, состоящее из команд двух типов:

— Встроенные служебные команды начинаются с подчерка (_) и одинаковы во всех узлах меню.
— Контекстно-зависимые команды зависят от текущего узла меню и начинаются с буквы.

Для вывода списка команд следует ввести команду _help:

```
uitcp>_help
```

Основные действия для вывода подсказок и перемещения по меню:

команда <Enter>
: Выполнить команду, например, перейти в дочерний узел меню или присвоить значение параметру. Если команда выполняет переход в дочерний узел, то приглашение uish последовательно наращивается, указывая полный путь к текущему узлу, и завершается угловой скобкой:

```
uitcp>log
uitcp.log>
```

Если команда осуществляет настройку параметра, то в качестве приглашения выводится полный путь к данному параметру и знак равенства:

```
uitcp.log>level
uitcp.log.level =
```

При вводе команд можно ограничиться первыми несколькими символами, позволяющими однозначно идентифицировать команду в пределах текущего узла меню. То же самое относится к параметрам, представленным фиксированным набором ключевых слов, например вместо debug в последнем случае достаточно ввести просто d:

```
uitcp.log.level = d
= "debug"
uitcp.log>
```

<Enter>
: Вывести список команд текущего узла.

_<Enter>
: Вывести список встроенных общих команд.

пробел <Enter>
: Перейти в вышестоящий узел меню, эквивалентно _exit.

пробел пробел <Enter>
: Перейти в корень меню, эквивалентно _end.

Встроенные команды для всех узлов меню:

_end
: Перейти в корень меню uish.

_exit
: Перейти в вышестоящий узел меню uish.

_help
: Вывести справку об основных командах.

_show
: Вывести конфигурацию текущего узла.

_new
: Добавить новый элемент в текущий узел.

_insert
: Вставить элемент в список.

_validate
: Проверить конфигурацию текущего узла. Если она некорректна, будут выведены соответствующие сообщения и подсказки.

_remove
: Удалить текущий узел.

_apply
: Применить сделанные изменения. При этом они не сохраняются, сохранять их нужно отдельно командой _write.

_write
: Сохранить полученную конфигурацию.

_quit
: Выйти из uish. Изменения, сделанные в данном сеансе, автоматически не сохраняются и не применяются.

## §2.8. Расширенные возможности установки и вызова *ui*TCP

Перед установкой *ui*TCP на устройстве может быть создан файл /etc/nsgSysConfig. Он является корневым файлом, в котором могут содержаться ссылки на другие файлы конфигурации, пути установок и другая важная информация, используемая различными компонентами *ui*TCP как при установке, так и при работе системы.. Это текстовый файл, имеющий формат:

```
return {
    ключ = значение,
    ...
}
```

В данной версии в /etc/nsgSysConfig предусмотрены следующие параметры:

uitcpConfigPath = "путь"
: Путь к директории, содержащей файлы конфигурации uitcp.

uitcpBinPath = "путь"
: Путь к директории, содержащей исполняемые файлы и библиотеки uitcp.

Например, для установки на x86-совместимый сервер рекомендуется использовать следующие пути:

```
return{
    uitcpConfigPath = "/home/user/uitcp",
    uitcpBinPath = "/opt/uitcp"
}
```

Параметры, задаваемые в файле /etc/nsgSysConfig, могут быть переопределены явным образом при вызове соответствующих программ. Если они не указаны никаким образом, то используются значения по умолчанию, заложенные непосредственно в самих исполняемых файлах.

**ВНИМАНИЕ** Для установки *ui*TCP, других дополнительных компонент и их конфигураций на клиентских устройствах без расширенной энергонезависимой памяти (USB Flash, USB HDD и т.п.) могут использоваться только директории внутри /etc.

Если *ui*TCP устанавливается на устройство с расширенной энергонезависимой памятью, позволяющей хранить файловую систему в развёрнутом виде, то запуске установщика ему может быть явно указан путь для установки бинарных файлов, например:

```
box1 ~ # ./uitcp-1.2 /usr/share/uitcp
```

Если путь явно не указан, то он берётся из параметра uitcpBinPath в файле /etc/nsgSysConfig. Если параметр либо сам файл отсутствует, то по умолчанию используется директория /etc/uitcp/bin.

После установки рекомендуется создать символические ссылки для основных утилит, например:

```
box1 ~ # ln -s /opt/uitcp/uitcp /usr/sbin/uitcp
box1 ~ # ln -s /opt/uitcp/tools/uish /usr/bin/uish
box1 ~ # ln -s /opt/uitcp/tools/uitcp-update-sftp /usr/bin/uitcp-update-sftp
```

При запуске собственно программы uitcp ей может быть явно указан файл конфигурации, например:

```
uitcp /etc/uitcp/config.last.working
```

Программа uitcp считывает конфигурацию из файла, определенного данным параметром. Если этот параметр отсутствует, то файл конфигурации ищется в директории, определенной параметром uitcpConfigPath в файле /etc/nsgSysConfig, и с именем config. (В вышеприведённом примере это будет /home/user/uitcp/config). Если этот параметр отсутствует в файле, или отсутствует сам файл /etc/nsgSysConfig, то конфигурация, по умолчанию, берётся из файла /etc/uitcp/config.

При запуске программы настройки uish ей могут быть явно указаны два параметра: файл схемы конфигурации и файл конфигурации, например:

```
uish /usr/share/uitcp/uitcp_scheme /etc/uiTCP_config
```

Если файл конфигурации не указан, то он находится так же как и для uitcp.

Если файл схемы конфигурации не указан, то он ищется в директории, определенной параметром uitcpBinPath в файле /etc/nsgSysConfig, с именем uitcpScheme.lua (В вышеприведённом примере это будет /home/user/uitcp/uitcpScheme.lua). Если этот параметр отсутствует в файле, или отсутствует сам файл /etc/nsgSysConfig, то конфигурация, по умолчанию, берётся из файла /etc/uitcp/bin/uitcp/uitcpScheme.lua.

# §3. Настройка *ui*TCP

## §3.1. Создание простейшего бесперебойного TCP-туннеля

**ВНИМАНИЕ** Данный и следующий разделы описывают конфигурационные файлы *ui*TCP на системах под управлением NSG Linux 1.0 и выделенных серверах. Указанные настройки могут быть также выполнены с помощью консольной утилиты uish.
На системах под управлением NSG Linux 2.0 настройка производится аналогичным образом с помощью Web-интерфейса или консольной утилиты nsgsh.

Для создания TCP-туннеля в самом минимальном виде необходимо выполнить следующие настройки *ui*TCP:

— На сервере: определить IP-адреса и TCP порты, по которым будут приниматься входящие запросы на установление туннелей.

— На клиенте: установить общие параметры туннеля и описать хотя бы один канал связи.

Кроме того, поскольку в данной версии защита данных с помощью SSL включена по умолчанию, дополнительно требуется либо выключить её для данного туннеля (что и делается ниже в данном примере), либо сгенерировать сертификаты X.509 и поместить их на оба устройства (см. п.4).

Соответствующие ветви в конфигурации сервера и клиента имеют вид:

Сервер:

```
{
    mode = "server",
    clients = {
        имя = {
            ssl = {
                disable = true
            },
        },
    },
    tunnelListener = {
        host = "ip-адрес",
        port = tcp-порт,
        {
            host = "ip-адрес",
            port = tcp-порт
        },
        ......
    }
}
```

В данном примере для клиента определено только имя, остальные настройки будут сделаны позже.

**ВНИМАНИЕ** Имя клиента должно содержать только латинские буквы и цифры.

Узел tunnelListener содержит, как минимум, два конечных параметра:

host — IP-адрес, по которому сервер ожидает входящие запросы на установление туннелей. Адрес должен принадлежать одному из IP-интерфейсов устройства NSG. Если в качестве адреса указана звёздочка (*), то сервер принимает запросы по любым IP-адресам, принадлежащим данному устройству NSG. Значение по умолчанию: "*".

port — Номер порта TCP, на котором ожидаются входящие соединения. Значение по умолчанию 50005.

Если требуется указать несколько сочетаний адресов и портов, например, если разные интерфейсы сервера должны принимать входящие соединения по разным TCP портам, то они добавляются в список в виде следующих элементов:

```
{
    host = "ip-адрес ",
    port = tcp-порт
}
```

Точно так же могут быть заданы несколько портов TCP на одном интерфейсе. Несмотря на формальную вложенность, все эти элементы в данном случае рассматриваются не иерархически, а параллельно с общей парой {host, port}. Например, в следующей конфигурации:

```
        tunnelListener = {
            host = "10.0.0.1",
            port = 50005,
            {
                host = "10.0.0.1",
                port = 51006
            }
        }
```

сервер будет использовать на интерфейсе 10.0.0.1 два порта TCP — 50005 и 51006.

Настройки клиента:

```
    {
        mode = "client",
        tunnel = {
            name = "имя",
            disable = true | false,
            description = "текст",
            numOfServers = число,
            servers = {
                [1] = "ip-адрес:tcp-порт",
                [2] = "ip-адрес:tcp-порт",
                .....
            },
            host = "ip-адрес",                  — устаревший
            port = tcp-порт,                    — устаревший
            connectAttempt = число,
            connectTmo = секунды,
            window = число,
            keepalive = число,
            attempt = число,
            links = {
                link1,
                link2,
                ......,
                priority = true | false
            },
            ssl = {
                disable = true
            }
        }
    }
```

Отдельные параметры в узле tunnel имеют следующий смысл:

name — Уникальное имя клиента в системе. Параметр обязательный. Должно совпадать с именем клиента, определённым на сервере.

disable = true | false — Административное отключение данного туннеля (disable = true). По умолчанию, все вновь создаваемые туннели включены.

description — Текстовое описание данного туннеля для удобства администрирования. Непосредственно на работу туннеля данный параметр не влияет.

numOfServers — Число серверов, используемых в системе. По умолчанию, клиент работает с одним сервером. Максимально допустимое число серверов — 8.

servers — Комплексный параметр, содержащий описания серверов. Каждая строка внутри данного узла является нумерованной и содержит IP-адрес сервера и номер порта TCP на нём. Если на устройстве включён и настроен клиент DNS, то сервер может быть задан в алфавитно-цифровом виде (например, my.processing.host.ru:50005). Для работы туннеля необходимо указать хотя бы один сервер.
Сервера используются по кругу в порядке возрастания номеров, причём они могут быть пронумерованы не подряд. Сервера с номерами, бóльшими чем numOfServers, игнорируются.
Глобальный узел servers в описании туннеля, по умолчанию, действует на все каналы связи. Однако он может быть переопределен индивидуально для каждого канала и для каждого режима работы этого канала (см. ниже).

host — Устаревший параметр: IP-адрес сервера. Формальное значение по умолчанию "127.0.0.1" (практического смысла не имеет).

port Устаревший параметр: номер порта TCP, на котором сервер ожидает входящие соединения. Значение по умолчанию 50005.

Параметры host и port в данной версии сохранены ради обратной совместимости с существующими конфигурациями и используются только при отсутствии параметра servers. Если параметр servers указан, то host и port игнорируются. Использовать эти параметры в новых конфигурациях не следует; в частности, они не позволяют корректно описать работу с несколькими серверами или с одним и тем же сервером, когда он имеет различные IP-адреса при доступе через разных операторов и при этом использует скрипты для реинициализации каналов связи.

connectAttempt
Число попыток соединения по каждому из каналов. Значение по умолчанию 10.

connectTmo Задержка между попытками соединения. Значение по умолчанию 20 сек.

Алгоритм установления/восстановления соединений работает следующим образом. Изначально клиент работает с первым сервером из списка. При инициализации туннеля или разрыве текущего канала связи клиент пытается установить связь с ним поочерёдно по всем каналам (в порядке приоритета или по кругу в зависимости от параметра priority, см. ниже). При этом время ожидания соединения на каждом канале составляет connectTmo/2; если за это время установить соединение не удалось, или если раньше этого времени пришёл явный отказ по какой-либо причине, то *ui*TCP переходит на следующий канал. Если список каналов исчерпан и связь установить не удалось, то система ожидает, пока истечёт время connectTmo с момента начала предыдущей попытки, после чего процедура повторяется снова. Если это время уже истекло (например, используются 3 канала, и на каждом попытка завершилась по таймауту), то следующая серия попыток начинается немедленно.

Если все возможные каналы связи перебраны connectAttempt раз без успеха, то считается, что сервер окончательно недоступен и туннель разрывается. При этом разрываются все TCP-соединения с локальными хостами, т.е. на них будет детектировано состояние *off-line*. Данные, которые не удалось отправить, теряются, и туннель уже не подлежит восстановлению — он может быть только инициализирован заново, после чего заново устанавливаются все TCP-соединения для передачи пользовательских данных.

Если число серверов больше 1, то следующая попытка инициализации туннеля предпринимается ко второму серверу из списка, и также connectAttempt раз по всем каналам связи. Затем используется третий сервер и так далее по кругу.

**ВНИМАНИЕ** После успешной инициализации туннеля к одному серверу клиент будет продолжать работать с ним неограниченное время и восстанавливать туннель до тех пор, пока это возможно. Переход на следующий сервер произойдёт в том и только в том случае, если туннель будет разорван окончательно.
Для принудительного перенаправления клиентов на другой сервер по мере завершения текущих транзакций следует использовать команду SHUTDOWN для "мягкого" выключения сервера *ui*TCP (см. п.3.9).

window Размер окна на передачу — максимальное число пакетов, которое может быть передано по туннелю, не дожидаясь подтверждения о приёме предыдущих пакетов. При установлении туннеля этот размер передаётся серверу, и сервер использует такой же размер окна для передачи данных клиенту. Значение по умолчанию 8.

keepalive Интервал посылки пакетов *keepalive* серверу в периоды отсутствия трафика. Уменьшение данного интервала приводит к более оперативной реакции на разрывы связи, увеличение — к экономии служебного трафика. Значение по умолчанию 3 сек.

**ПРИМЕЧАНИЕ** Для сетей GPRS/EDGE (2G) и других типов каналов с большим временем обращения пакетов рекомендуемое значение *keepalive* — 15 сек. Чрезмерно малое время *keepalive* может приводить к ложным срабатываниям *ui*TCP и переустановлению соединения.

attempt Предельное число пакетов *keepalive*, на которое могут быть не получены ответы от сервера. Если не получено указанное число ответов подряд, канал связи считается разорванным и клиент предпринимает попытку восстановить соединение. Уменьшение данного числа приводит к более оперативной реакции на разрывы связи, увеличение — к снижению вероятности ложных срабатываний. Значение по умолчанию 3.

**ПРИМЕЧАНИЕ** Помимо *keepalive*, *ui*TCP следит за состоянием TCP-соединения с удалённой стороной. Таким образом, детектируется также обрыв связи по всем другим критериям, приводящим к падению интерфейса и разрыву этого соединения: падению сигнала DCD, срабатыванию механизма LCP Echo на PPP-соединениях и др.

Сервер со своей стороны не использует *keepalive*, но он знает параметры клиента и может вычислить максимальное суммарное время, по истечении которого клиент окончательно разорвёт туннель:

connectAttempt × connectTmo + keepalive × (attempt + 1)

Если за это время от клиента не будет получено ни одного пакета (данных или *keepalive*), то сервер также сочтёт туннель окончательно разорванным и разорвёт все локальные TCP-соединения на своей стороне.

**ПРИМЕЧАНИЕ** Таймауты *ui*TCP должны быть согласованы с настройками прикладного программного обеспечения следующим образом: максимальное время ожидания ответа/подтверждения приёма в прикладном ПО должно быть не меньше, чем максимально допустимое время отсутствия связи, возможное при данных настройках *ui*TCP.

links Список используемых каналов связи. Внутри данного узла имеются также следующие параметры:

priority *секунды*

Приоритизация каналов связи. Если установлено любое положительное значение, то каналы имеют приоритет в порядке их перечисления в списке (первый — наивысший, т.е. этот канал является основным). В этом случае:

— При разрыве основного канала связи клиент пытается заново установить связь сначала по второму каналу, потом по третьему и т.д.

— При разрыве любого из резервных каналов клиент пытается установить связь, начиная с первого (основного) канала.

— При работе по любому из резервных каналов связи после указанного времени клиент будет пытаться разорвать связь и восстановить туннель по более приоритетному каналу. Точное поведение клиента в этом случае определяется дополнительным параметром idle-time.

Если значение данного параметра равно нулю, то время работы канала ограничивается одним только параметром idle-time.

Если значение данного параметра отсутствует(равно nil), то клиент не пытается самостоятельно переключаться с одного канала связи на другой, а при разрыве связи пытается использовать следующий канал в списке (и далее по кругу).

idle-time *секунды*

Дополнительное время работы по неприоритетному каналу связи. Данный параметр используется совместно с priority. После истечения времени priority клиент ждёт ближайшей паузы в передаче полезных данных, чтобы процедура переключения произошла с минимально возможным воздействием на работу пользователя. Например, если по каналу работает банкомат, то *ui*TCP дожидается завершения текущей транзакции. Параметр idle-time устанавливает длительность этой паузы.

Например, если idle-time = 30 и в момент истечения времени priority по каналу передаются данные, то клиент будет ждать окончания передачи и ещё 30 сек. после этого. Если же к моменту priority данные уже не передаются в течение 10 сек, то клиент будет ждать только оставшиеся 20 сек. Если idle-time явно не указано, то по умолчанию оно равно keepalive.

Если значение idle-time равно 0, то резервные каналы связи будут разрываться ровно через priority секунд, независимо от наличия трафика в этот момент.

Описание канала связи имеет следующую структуру:

```
{
    disable = true | false,
    servers = {
        [1] = "ip-адрес:tcp-порт",
        [2] = "ip-адрес:tcp-порт",
        .....................................
    },
    host = "ip-адрес",                  — устаревший
    port = tcp-порт,                    — устаревший
    dev = "интерфейс",
    gw = "шлюз",
    description = "текст",
    dualSim = true | false,
    csqRequest = {
        interval = секунды,
        ttydev = "порт"
    },
    restart = {
        {
            description = "текст",
            script = "скрипт",
            servers = {
                [1] = "ip-адрес:tcp-порт",
                [2] = "ip-адрес:tcp-порт",
                .....................................
            },
            timeout = секунды,
        },
        .....................,
        primary = номер_скрипта,
        timeout = секунды,
```

```
        },
        ....................................,
        restartScript = {                   — устаревший
            "скрипт1",
            "скрипт2",
            .........
        }
        restartSciptTmo = секунды           — устаревший
    }
```

Назначение данных параметров:

disable = true | false
: Административное отключение данного канала связи (disable = true) — например, для целей отладки других каналов или при долговременной неработоспособности данного канала. По умолчанию, все вновь создаваемые каналы включены.

servers, host, port
: Список серверов с их IP-адресами и номерами портов TCP, к которым надлежит обращаться при работе по данному каналу связи. Указываются так же, как и аналогичные глобальные параметры в узле tunnel.
Если эти параметры заданы, то для данного канала связи они имеют безусловный приоритет перед параметрами, установленными в вышестоящем узле меню tunnel. Если глобальный список серверов не задан, то локальный список должен быть создан для каждого канала связи.
Если какой-либо из серверов отсутствует в списке для данного канала, то при работе с этим сервером данный канал не используется.

**ВНИМАНИЕ**
: Под одним номером во всех списках servers необходимо указывать один и тот же физический сервер *ui*TCP. Локальные списки предназначены только для случая, когда один и тот же сервер имеет различные IP-адреса при доступе по разным каналам связи. Например, клиент может использовать один канал доступа в Интернет общего пользования, по которому сервер виден под своим глобальным IP-адресом (указываются в меню tunnel), а другой канал через виртуальную частную сеть с приватными IP-адресами (указывается в меню данного канала).

dev, gw
: Имя выходного интерфейса и/или IP-адрес шлюза для отправки пакетов на сервер. Эти параметры используются *ui*TCP для маршрутизации. При переключении на данный канал в таблице маршрутизации создаётся запись в одном из следующих форматов:

```
ip route x.x.x.x/32 device
ip route x.x.x.x/32 gateway
```

соответственно, а запись, соответствующая прежнему каналу связи, удаляется.
Для интерфейсов "точка-точка" достаточно указать имя устройства, например, s1. Для широковещательных интерфейсов (Ethernet, VLAN и т.п.), наоборот, IP-адрес шлюза обязателен, а имя устройства можно не указывать (если указано description).
Значение параметра dev однозначно соответствует имени IP-интерфейса устройства NSG во всех случаях, в том числе и для модулей GPRS/EDGE/3G с двумя SIM-картами.

description
: Текстовое описание данного канала связи для удобства администрирования: имя оператора и т.п. Например, "GPRS" или "Скайлинк". Отражается в Web-интерфейсе и статистике работы туннеля. Непосредственно на работу туннеля данный параметр не влияет.
Если данный параметр не указан, то в Web-интерфейсе и статистике вместо него указывается имя устройства dev. Из двух этих параметров следует указывать, как минимум, один, поскольку в противном случае сервер не будет получать информацию о том, по какому каналу работает клиент в данный момент.

dualSim = true | false
: Устаревший параметр: режим учёта активной SIM-карты для модуля GPRS/EDGE/3G с двумя SIM-картами. Если данный режим включён, то в Web-интерфейсе и статистике работы туннеля отражается, через какого из сотовых операторов — основного (main) или резервного (aux) он в данный момент работает. Непосредственно на работу туннеля данный параметр не влияет. Рекомендуется вместо него использовать поля description в меню restart (см. ниже).
Подробно об особенностях использования модулей с двумя SIM-картами см. п.3.2.

csqRequest
: Контроль уровня сигнала. Имеет смысл только для сотовых модулей, поддерживающих параллельно передачу данных и работу с AT-командами (SMS-управление и др.). Чтобы использовать данную возможность, на порту должен быть включён обработчик SMS (sms-handler) в режиме ppp-cooperation yes.Если настроена данная группа параметров, то клиент *ui*TCP периодически запрашивает у своего сотового модуля уровень сигнала сети и сообщает его серверу в очередных пакетах. Параметры для выполнения этой процедуры:

    interval — Период опроса уровня сигнала.
    ttydev — Имя порта для опроса. Параметр обязательный для работы этой функции. В зависимости от типа модуля, это может быть либо имя порта, либо специальный служебный порт, который создаётся для некоторых типов сотовых модулей.

Узел restart содержит набор скриптов, которые могут исполняться в случае, когда клиент *ui*TCP детектировал отказ данного канала связи и переходит с него на какой-либо другой. Основное назначение этого механизма — принудительно рестартовать данный канал связи, если его отказ не обнаруживается встроенными средствами физического и канального уровней. (Например, оператор не отвечает, в нарушение стандарта, на запросы LCP Echo, т.е. механизм *keepalive* в PPP-соединении неприменим; в то же время пропускная способность GPRS опустилась до нуля, но формально соединение не разрывается, т.е. сигнал DCD не падает.)

script Скрипт для реинициализации канала связи. Тело скрипта заключается в двойные кавычки и может содержать любую последовательность команд, которую может исполнить обработчик командной строки Linux.

С каждым скриптом может быть ассоциирован свой локальный набор серверов и свой таймаут, объединенные вместе с ним в один блок символами {...}. Локальный набор серверов для скрипта описывается по тем же правилам, что и для канала связи и глобальный набор серверов и, если задан, то имеет приоритет перед ними обоими.

Таймауты в узле restart позволяют ограничить минимальное время, которое должно пройти между двумя исполнениями скриптов. В противном случае возможна ситуация, когда *ui*TCP будет постоянно рестартовать интерфейс раньше, чем он успеет стартовать полностью. Это необходимо, в первую очередь, для сотовых интерфейсов GPRS и CDMA, которым требуется 20–35 сек. для инициализации, регистрации в сети и установления PPP-соединения. Рекомендуется, чтобы значение таймаута было, как минимум, в 3 раза больше времени рестарта интерфейса. Значение глобального таймаута по умолчанию — 120 сек., локальные таймауты не определены.

В общем случае узел restart может содержать несколько блоков {скрипт, сервера, таймаут, описание}. При начале работы устанавливается глобальный таймаут, гарантирующий некоторое минимально необходимое время для установления соединения по данному каналу после старта системы. По истечении этого времени, если детектируется отказ канала и клиент переходит на следующий канал, то исполняется первый скрипт и устанавливаются ассоциированные с ним набор серверов и величина таймаута до следующего исполнения скрипта (если таковые определены). Значение поля description добавляется к общему description, установленному в вышестоящем узле меню. По завершении этого времени, в случае очередного возвращения на этот канал и его очередного отказа, будет исполнен второй скрипт, и т.д. по кругу. На практике эта возможность имеет смысл только в количестве 2 скриптов для управления модулем GPRS/EDGE/3G с двумя SIM-картами. Пример использования скриптов см. в п. 3.2.

restartScript Устаревший параметр: список скриптов для реинициализации канала связи.

restartSciptTmo
Устаревший параметр: минимальный интервал между исполнением скриптов реинициализации.

Параметры restartScript и restartSciptTmo в данной версии сохранены ради обратной совместимости с существующими конфигурациями и используются только при отсутствии узла restart. Если этот узел указан, то restartScript и restartSciptTmo игнорируются. Использовать эти параметры в новых конфигурациях не следует.

## §3.2. Особенности настройки соединений через модули GPRS с двумя SIM-картами

Для работы в сетях GSM/GPRS/EDGE/3G двух операторов необходимо:

— Вставить в модуль две SIM-карты и снять перемычку, запрещающую работу с резервной картой.

— Настроить на порту два сценария дозвона (chat-script) и два шаблона виртуальных соединений (virtual-template) с параметрами, соответствующими основному (main) и резервному (aux) операторам.

Переключение между операторами регулируется следующим параметром в меню порта:

prio main *X* aux *Y*

В частности, предельный вариант main 1 aux none предписывает работу исключительно через основного оператора, main none aux 1 — только через резервного. Именно такие упрощённые приоритеты следует использовать при работе *ui*TCP через модуль с двумя SIM-картами. Манипулируя ими при помощи скриптов, которые задаются в узле меню restart, клиент *ui*TCP в каждой попытке синхронизирует выбор оператора и используемый набор серверов.

Подробно о настройке сотовых соединений и, в частности, об использовании модулей с двумя SIM-картами, см. документ NSG: *Программное обеспечение NSG Linux. Руководство пользователя. Часть 3. Протоколы канального уровня. Коммутация пакетов.*

Модуль с двумя SIM-картами рассматривается в данной версии *ui*TCP как один канал связи (в отличие от ранних версий). Наиболее сложным является случай, когда у разных операторов один и тот же сервер *ui*TCP доступен по разным IP-адресам. Рассмотрим конфигурацию при следующих условиях:

— На устройстве NSG установлен единственный интерфейс для подключения к вышестоящей сети — модуль с двумя SIM-картами.

— Основной оператор (Мегафон) обеспечивает выход в VPN, в которой сервер *ui*TCP доступен по адресу 10.11.12.13. (Под VPN сотовые операторы обычно подразумевают услугу построения закрытой корпоративной IP-сети, изолированной от сетей общего пользования, поверх своей сотовой сети.)

— Резервный оператор (Билайн) предоставляет только выход в Интернет, откуда сервер доступен по адресу 123.45.67.89. Кроме того, предположим для наглядности, что у этого оператора процедура регистрации в сети может выполняться дольше обычного.

Конфигурация канала связи:

```
{
    mode = "client",
    tunnel = {
        .....................
        links = {
            {
                servers = {
                    [1] = "10.11.12.13:50005"
                },
                dev = "s1",
                description = "GPRS",
                dualSim = true,
                restart = {
                    timeout = 120,
                    {
                        script = "config-nsg port s1 main none aux 1",
                        description = "BEELINE",
                        servers = {
                            [1] = "123.45.67.89:50005"
                        },
                        timeout = 150
                    },
                    {
                        script = "config-nsg port s1 main 1 aux none",
                        description = "MEGAFON",
                    }
                }
            }
        }
    }
}
```

Дополнительно считается, что в настройках порта s1 в основном программном обеспечении установлены приоритеты операторов по умолчанию: prio main 1 aux none. В этом случае алгоритм работы клиента будет следующим:

— После старта системы клиент пытается соединиться с сервером через основного оператора по адресу 10.11.12.13 в течение 120 сек. Если в это время попытка соединения завершается неудачно (на любом уровне от скрипта дозвона до *ui*TCP) или соединение разрывается, то сотовый модуль рестартует с этой же SIM-картой.

— После истечения первых 120 сек, если соединение не устанавливается (в уже начатой попытке) или разрывается, сотовый модуль переключается на резервную SIM-карту, а *ui*TCP — на резервный адрес сервера 123.45.67.89.

— В течение следующих 150 сек клиент пытается соединиться с сервером через резервного оператора.

— По истечении 150 сек, если соединение не устанавливается или разрывается, сотовый модуль снова переключается на основную SIM-карту, а *ui*TCP — на основной адрес 10.11.12.13, и снова работает с ними не менее, чем в течение глобального таймаута 120 сек (поскольку локальный таймаут и сервер для данного скрипта не указаны).

— Время работы через одного или через другого оператора не ограничено, принудительное возвращенние на приоритетного оператора не производится.

**ПРИМЕЧАНИЕ** Скрипты для выбора основной/резервной SIM-карты фактически изменяют текущую конфигурацию устройства, устанавливая число попыток 1 и 0, соответственно. Если в этот момент сохранить конфигурацию, то в неё попадут эти изменения, и в результате после следующего рестарта устройства его поведение может не соответствовать ожидаемому.

## §3.3. Настройка TCP-соединений и NAT

Простейший туннель, созданный в п.3.1, ещё не пригоден для передачи данных. Помимо собственно создания туннеля, необходимо явным образом описать трафик, который должен передаваться через него. Для TCP-трафика это описание заменяет собой стандартные процедуры IP-маршрутизации и NAT. Описание выглядит одинаково для клиента и для сервера, поскольку установленный туннель позволяет инициировать TCP-соединения в обоих направле-ниях. Различие состоит только в том, что на клиенте эти узлы меню находятся внутри описания туннеля, а на сервере — в меню каждого клиента:

**Клиент**

```
tunnel = {
    localListener = {
        localListener1
        localListener2
        ..................
    },
    nat = {
        правило1
        правило2
        ..................
    }
}
```

**Сервер**

```
имя = {
    localListener = {
        localListener1
        localListener2
        ..................
    },
    nat = {
        правило1
        правило2
        ..................
    }
}
```

Кроме того, на сервере могут быть заданы глобальные правила NAT для всех клиентов. Эти правила применяются в случае, если полученный пакет не подпадает ни под одно правило NAT, установленное индивидуально для данного клиента.

Функциональное различие в этом плане заключается не между клиентом и сервером *ui*TCP, а между устройством NSG на стороне той локальной сети, откуда инициируется запрос на установление TCP-соединения (вызывающим устройством), и на стороне той сети, куда этот запрос адресован (отвечающим устройством). На вызывающем устройстве должны быть созданы объекты localListener — они определяют, какие пакеты готова принимать служба *ui*TCP, чтобы отправить их в туннель. На отвечающем устройстве должны быть настроены правила NAT — они определяют, как дальше отправлять пакеты, полученные из туннеля, в локальную сеть. При этом роль вызывающего и отвечающего устройства могут исполнять как клиент, так и сервер *ui*TCP в зависимости от того, с какой стороны инициируется соединение.

Настраивать передачу данных в противоположном направлении — от отвечающего хоста к вызывающему — не требуется.

**На вызывающей стороне** узлы localListener имеют следующую структуру:

```
{
    id = "идентификатор"
    host = "ip-адрес",
    port = tcp-порт
    tcpKeepaliveInterval = секунды
    tcpKeepaliveRetry = число
}
```

где параметры имеют следующий смысл:

id — Идентификатор исходящего соединения из локальной сети в туннель, для упрощённой настройки NAT на отвечающей стороне. Может использоваться вместо параметров InSrcAddr, InDstPort как критерий для выбора правила преобразования адресов.

host — IP-адрес, по которому устройство ожидает входящие запросы из локальной сети на установление TCP-соединений. Адрес должен принадлежать одному из IP-интерфейсов устройства NSG. Если в качестве адреса указана звёздочка (*), то принимаются запросы по любым IP-адресам, принадле-жащим данному устройству NSG.

**ВНИМАНИЕ** — Использовать установку host = "*" настоятельно не рекомендуется, поскольку в этом случае вход в туннель будет открыт со всех интерфейсов, в т.ч. и с интерфейса сети общего пользования. (Хотя этот интерфейс и должен быть защищён фильтрами, ни на одну защиту не следует полагаться на 100%, поскольку ошибки возможны, в т.ч., и в человеческом факторе.) Вместо этого следует явно указывать IP-адрес внутреннего интерфейса, обращённого к банкомату или к процессинговому центру, соответственно.
Возможно, в одной из следующих версий возможность указания * будет запрещена.

port — Номер порта TCP, на котором ожидаются входящие соединения.

tcpKeepaliveInterval — Интервал посылки пакетов TCP *keepalive* в сторону локального прикладного хоста — инициатора TCP-соединения. С помощью этих пакетов можно контролировать работоспособность хоста и

сетевого соединения с ним, а также эмулировать активность удалённого прикладного хоста в случае, если при отсутствии такой активности соединение разрывается по таймауту. Значение по умолчанию nil — пакеты *keepalive* не посылаются.

tcpKeepaliveRetry
: Максимально допустимое число пакетов TCP *keepalive* подряд, на которые не получены ответы. В этом случае устройство NSG разрывает локальное TCP-соединение и соответствующее ему соединение в туннеле. Значение по умолчанию nil — разрыв соединения не производится.

По умолчанию, никакие localListener не определены. Туннель может принимать входящие запросы одновременно по нескольким TCP портам (на всех или только на некоторых IP-интерфейсах), т.е. на входе одного туннеля могут стоять одновременно несколько localListener.

**ВНИМАНИЕ** Для трафика TCP система *ui*TCP работает в режиме прокси, или *подставного хоста*. Хост, инициирующий передачу данных, должен обращаться не к конечному прикладному хосту, а к устройству NSG на своей стороне. Указывать на прикладных хостах шлюз по умолчанию, как правило, не требуется, поскольку они располагаются в одной сети и одной IP-подсети с устройством NSG (если же в разных — то шлюзом будет уже другое устройство).

**На отвечающей стороне** правила NAT имеют следующую структуру:

```
{
    inId = "идентификатор"
    inSrcAddr = "ip-адрес",
    inDstPort = tcp-порт,
    outDstAddr = "ip-адрес",
    outDstPort = tcp-порт,
    outSrcAddr = "ip-адрес"
    tcpKeepaliveInterval = секунды
    tcpKeepaliveRetry = число
}
```

При установлении TCP-соединения запрос, полученный из туннеля, проверяется либо по идентификатору соединения, либо по совокупности IP-адреса источника и номера TCP порта назначения; на сервере, помимо этого, неявно присутствует ещё один критерий — имя клиента. Далее отвечающее устройство NSG формирует новый запрос на установление локального TCP-соединения и, исходя из этих критериев, может подставить в него некоторый заданный IP-адрес источника, IP-адрес назначения и порт TCP назначения (или не подставлять ничего). Именно с этими атрибутами запрос и последующие пакеты с данными отправляются дальше в локальную сеть назначения.

inId
: Идентификатор входящего соединения из туннеля в локальную сеть. Если установлен, то в качестве критерия для NAT используется идентификатор, который был установлен для данного соединения на вызывающей стороне; если нет, то inSrcAddr и inDstPort.

inSrcAddr
: IP-адрес, указанный в качестве источника в исходном пакете.

inDstPort
: TCP порт, указанный в качестве назначения в исходном пакете.

Считается, что данное TCP-соединение подпадает под действие данного правила NAT, если значенит id либо оба значения inSrcAddr, inDstPort в нём равны критериям, указанным в конфигурации. Если правило содержит только один критерий inSrcAddr или inDstPort, то под него подпадают все пакеты, проходящие по этому критерию, независимо от значения другого. Если правило не содержит ни одного из вышеперечисленных критериев, то оно действует на все входящие соединения.

TCP-запрос, полученный из туннеля, проверяется по правилам NAT в порядке их следования в конфигурации. На сервере, если пакет не подпадает ни под одно правило NAT для данного туннеля, то он дополнительно проверяется по глобальным правилам NAT. Если пакет не соответствует ни одному правилу, он уничтожается.

outDstAddr
: IP-адрес, подставляемый в пакет в качестве адреса назначения. Параметр обязательный.

outDstPort
: TCP порт, подставляемый в пакет в качестве порта назначения. Если данный параметр отсутствует, то сохраняется TCP порт назначения, указанный в оригинальном пакете.

outSrcAddr
: IP-адрес, подставляемый в пакет в качестве адреса источника. Этот же адрес приваивается в качестве вторичного (*alias*) локальному интерфейсу lo устройства NSG.
Если данный параметр отсутствует, то в качестве источника указывается IP-адрес того интерфейса, через который пакет уходит в локальную сеть.

**ВНИМАНИЕ** Для того, чтобы туннель мог передавать пользовательские данные, необходимо настроить, как минимум:

- Один localListener на вызывающей стороне; в противном случае устройство NSG не будет принимать никакие пакеты из локальной сети.
- Одно правило NAT, под которое будут подпадать эти пакеты, на отвечающей стороне;

это правило должно содержать, как минимум, IP-адрес вызываемого прикладного хоста, поскольку исходный IP-адрес назначения — локальный адрес вызывающего устройства NSG — уже не имеет никакого смысла в отвечающей сети. Минимальное правило, действующее на все пакеты, имеет вид:

```
{
    outDstAddr = "A.B.C.D"
}
```

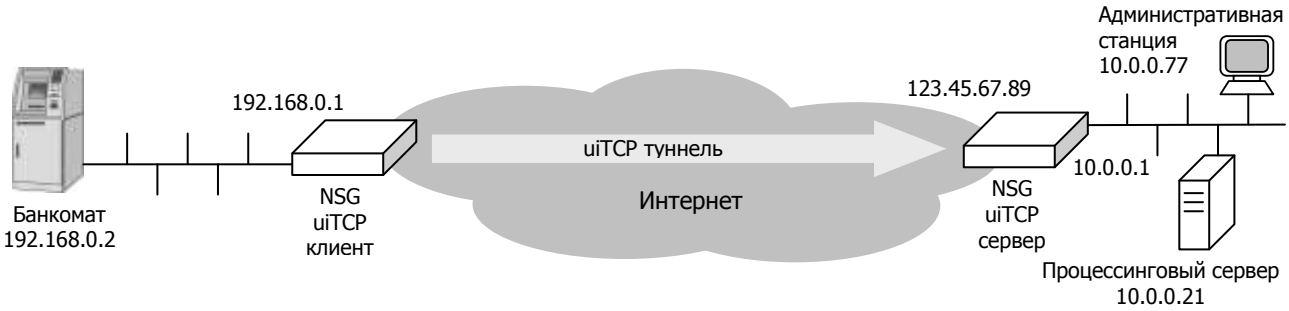
tcpKeepaliveInterval

Интервал посылки пакетов TCP *keepalive* в сторону локального прикладного хоста — конечного адресата TCP-соединения. С помощью этих пакетов можно контролировать работоспособность хоста и сетевого соединения с ним, а также эмулировать активность удалённого прикладного хоста (инициатора соединения) в случае, если при отсутствии такой активности соединение разрывается по таймауту. Значение по умолчанию nil — пакеты *keepalive* не посылаются.

tcpKeepaliveRetry

Максимально допустимое число пакетов TCP *keepalive* подряд, на которые не получены ответы. В этом случае устройство NSG разрывает локальное TCP-соединение и соответствующее ему соединение в туннеле. Значение по умолчанию nil — разрыв соединения не производится.

**Пример.** Имеется банкомат, установленный по адресу ул.Ленина, 1. Сервер *ui*TCP доступен через Интернет по адресу 123.45.67.89. (Адрес клиента априори неизвестен, поскольку он может меняться динамически при каждом переустановлении соединения через каждый канал связи.) В локальной сети банка имеется процессинговый сервер, к которому обращается банкомат, и административная станция, которая обращается к банкомату и к клиентскому устройству *ui*TCP.




Конфигурация localListener и NAT на клиенте:

```
{
    mode = client,
    tunnel = {
        name = "ATM0001",
        description = "ul.Lenina, 1"
        servers = {
            [1] = "123.45.67.89:50005"
        },
        ...........
        localListener = {
            {
                host = "192.168.0.1",
                port = 10001
            }
        },
        nat = {
            {
                inSrcAddr = "10.0.0.77",
                inDstPort = 20001,
                outDstAddr = "192.168.0.1",
                outDstPort = 80
            },
            {
                inSrcAddr = "10.0.0.77",
                inDstPort = 30001,
                outDstAddr = "127.0.0.1",
                outDstPort = 23
            }
        }
    }
}
```

Конфигурация localListener и NAT на сервере:

```
{
    mode = server,
    tunnelListener = {
        host = "123.45.67.89",
        port = 50005,
    }
    ...........
    clients = {
        ATM0001 = {
            nat = {
                {
                    outDstAddr = "10.0.0.21",
                    outDstPort = 25001
                }
            },
            localListener = {
                {
                    host = "10.0.0.1",
                    port = 20001
                },
                {
                    host = "10.0.0.1",
                    port = 30001
                }
            }
        }
    }
}
```

В данной конфигурации через туннель могут устанавливаться три TCP-соединения:

— Банкомат обращается к клиенту *ui*TCP по адресу 192.68.0.1 и порту TCP 10001. Для него это устройство выполняет роль процессингового сервера. Клиент передаёт данные в туннель. В центральном офисе сервер *ui*TCP устанавливает для этих данных TCP-соединение с реальным процессинговым сервером по адресу 10.0.0.21 и порту TCP 25001, при этом в качестве IP-адреса источника указывает свой собственный адрес 10.0.0.1.

— Административная станция устанавливает соединение с сервером *ui*TCP по адресу 10.0.0.1 и порту TCP 20001 и попадает в Web-интерфейс банкомата по порту TCP 80.

— Административная станция устанавливает соединение с сервером *ui*TCP по адресу 10.0.0.1 и порту TCP 30001 и попадает в клиентское устройство *ui*TCP по Telnet.

Как можно заметить, в данной конфигурации ни IP-адрес банкомата, ни номера портов TCP, по которым он обращается к процессингу или доступен для администрирования, не используются на клиентской стороне. Все уникальные настройки привязаны исключительно к имени клиента. Это означает, что при массовой инсталляции все банкоматы и все локальные интерфейсы клиентских устройств *ui*TCP могут быть настроены одинаково. Все уникальные атрибуты — номер TCP порта, выделенного данному банкомату на процессинге, и номера портов для администрирования данного банкомата и данного клиентского устройства — централизованно назначаются на сервере *ui*TCP.

Возможна также схема, при которой каждый банкомат (или каждая удалённая площадка — банкомат и клиентское устройство) имеет уникальный IP-адрес вида 10.*x.y.z*, под которым он известен процессинговому серверу и административной станции. При этом, если указанный адрес не принадлежит ни одному из интерфейсов устройства NSG, он автоматически назначается локальному интерфейсу (lo).

## §3.4. Настройка датаграммных соединений

Датаграммные соединения позволяют передавать через бесперебойные соединения трафик произвольных протоколов, работающих поверх IP, либо непосредственно IP-трафик. Для организации датаграммных соединений через бесперебойный туннель *ui*TCP возможны следующие варианты:

— Передача пакетов UDP с заданными номерами портов UDP источника и назначения.

— Передача пакетов 4 уровня модели OSI с заданным типом протокола (GRE, IPsec, ICMP и др.), либо целиком IP-пакетов (включая заголовок 3 уровня).

— Передача произвольного IP-трафика. В этом режиме в системе создаётся виртуальный IP-интерфейс, на который может быть направлен трафик стандартными средствами IP-маршрутизации, так же как и на физический интерфейс или в туннель какого-либо из стандартных типов.

Для передачи датаграммного трафика на клиенте и на сервере *ui*TCP организуются объекты, называемые *сокетами* (букв. "розетками"). Как и для TCP-соединений, на клиенте описания сокетов находятся внутри описания туннеля, а на сервере — в меню каждого клиента:

**Клиент**

```
tunnel = {
    socket = {
        имя_сокета = {
            disable = true | false
            type = "udp" | "raw" | "net"
            параметры udp
            параметры raw-ip
            параметры вирт.интерфейса
        },
        .....
    },
}
```

**Сервер**

```
имя_клиента = {
    socket = {
        имя_сокета = {
            disable = true | false
            type = "udp" | "raw" | "net"
            параметры udp
            параметры raw-ip
            параметры вирт.интерфейса
        },
        .....
    },
}
```

} В зависимости от типа сокета

Как и для TCP-соединений, для датаграммных потоков также может производиться преобразование IP-адресов и портов UDP на стороне конечного получателя датаграммы.

Узел socket содержит следующие параметры:

*имя_сокета* Каждый сокет имеет номер от 1 до 64. Иные типы имён не допускаются. Всего в одном туннеле может быть открыто до 64 сокетов. Имя (номер) сокета на клиенте и соответствующего ему сокета на сервере должны совпадать.

disable = true | false
Отключение сокета (с сохранением его настроек). По умолчанию, все вновь создаваемые сокеты включены.

type = "udp" | "raw" | "net"

Один из трёх возможных типов сокета:

udp Передача UDP трафика с заданными номерами портов.
raw Передача произвольного трафика IP или 4 уровня.
net Виртуальный IP-интерфейс.

По умолчанию, для вновь создаваемых сокетов устанавливается тип UDP.

**Настройка UDP-сокетов**

Узел настроек для сокета UDP имеет следующий вид:

```
udp = {
    peerAddress = "ip-адрес",
    peerPort = udp-порт,
    socketAddress = "ip-адрес ",
    socketPort = udp-порт
}
```

peerAddress IP-адрес локального хоста, с которым предполагается обмен датаграммами. Параметр обязательный.

peerPort Номер порта UDP локального хоста, на который будут посылаться датаграммы, полученные из бесперебойного туннеля. Параметр обязательный.

socketAddress

IP-адрес, по которому устройство ожидает входящие UDP-датаграммы от локального хоста и с которого отправляются датаграммы этому хосту. Адрес должен принадлежать одному из IP-интерфейсов устройства NSG. Если в качестве адреса указана звёздочка (*), то принимаются входящие пакеты по любым IP-адресам, принадлежащим данному устройству NSG, а в исходящих пакетах в качестве источника указывается IP-адрес того интерфейса устройства NSG, через который отправляются пакеты.

socketPort Номер порта UDP, на котором устройство NSG ожидает датаграммы. Параметр обязательный.

Таким образом, сокет принимает от хоста-источника пакеты со следующими атрибутами:

source IP address = peerAddress
destination IP address = socketAddress
destination UDP port = socketPort

Данные из этих пакетов передаются по бесперебойному туннелю *ui*TCP на сокет с таким же номером, организованный на удалённом устройстве NSG. Остальные пакеты сокетом игнорируются.

Если сокет UDP получает данные по туннелю от удалённой стороны, то эти данные отправляются в локальную сеть в виде пакета со следующими атрибутами:

source IP address = socketAddress или адрес выходного интерфейса
destination IP address = peerAddress
destination UDP port = peerPort

**Настройка сокетов общего вида**

Узел настроек для сокета, обрабатывающего произвольный IP-трафик, имеет следующий вид:

```
raw = {
    ipHdrIncl = true | false,
    peerAddress = "ip-адрес",
    protocol = номер
    socketAddress = "ip-адрес "
}
```

ipHdrIncl = true | false

Передача пакета целиком (включая заголовок IP) либо только начиная с заголовка 4 уровня модели OSI. По умолчанию, IP-заголовок не передаётся.

**ВНИМАНИЕ** Для сокета общего вида параметр ipHdrIncl должен быть настроен одинаково на обеих сторонах туннеля. В противном случае сокет будет неработоспособным.

peerAddress
socketAddress

Для входящих пакетов из локальной сети, используются так же, как и для UDP-сокета. Для исходящих пакетов в локальную сеть, если в них не передаётся IP-заголовок, эти адреса используются в качестве адреса назначения и источника, соответственно.

protocol Номер протокола 4 уровня модели OSI в соответствии с документом IANA:
http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml
Некоторые наиболее употребительные протоколы:

| | | | |
|---|---|---|---|
| 1 | ICMP | 50 | ESP |
| 6 | TCP | 51 | AH |
| 17 | UDP | 115 | L2TP |
| 47 | GRE | | |

**ПРИМЕЧАНИЕ** Некоторые технологии используют несколько различных протоколов 4 уровня. В частности, PPTP использует TCP и GRE. IPsec использует протоколы ESP, AH и UDP с номером порта 500 и обязательно требует передачи IP-заголовка полностью.

**ВНИМАНИЕ** Использовать установку socketAddress = "*" настоятельно не рекомендуется для обоих типов сокетов, поскольку в этом случае вход в туннель будет открыт со всех интерфейсов, в т.ч. и с интерфейса сети общего пользования. (Хотя этот интерфейс и должен быть защищён фильтрами, ни на одну защиту не следует полагаться на 100%, поскольку ошибки возможны, в т.ч., и в человеческом факторе.) Вместо этого следует явно указывать IP-адрес внутреннего интерфейса, обращённого к банкомату или к процессинговому центру, соответственно.
Возможно, в одной из следующих версий возможность указания * будет запрещена.

**Настройка виртуальных интерфейсов**

Узел настроек для сокета, работающего в режиме виртуального IP-интерфейса, имеет следующий вид:

```
net = {
    ifName = nil
    ifAddress = {
        anycast = "ip-адрес",
        broadcast = "ip-адрес",
        peer = "ip-адрес",
        prefix = "ip-префикс",
        {
            anycast = "ip-адрес",
            broadcast = "ip-адрес",
            peer = "ip-адрес",
            prefix = "ip-префикс"
        },
        ...
    },
}
```

При такой настройке *ui*TCP в системе создаётся виртуальный IP-интерфейс (псевдо-интерфейс), доступный для маршрутизации обычными средствами основной командной оболочки. Интерфейс получает имя, указанное параметром ifName либо, если этот параметр не задан, имя следующего вида:

*имя_туннеля.номер_сокета*

**ВНИМАНИЕ** Если имя туннеля имеет длину более 10 символов, то остальные символы отбрасываются. При настройке системы, особенно на сервере *ui*TCP, необходимо следить за тем, чтобы имена туннелей различались в первых 10 символах, в противном случае второй сокет не будет инициализироваться.

IP-адрес этому интерфейсу назначается при помощи параметров prefix, anycast, broadcast, peer, аналогичных общей команде назначения IP-адресов в основной командной оболочке:

pseudo-interface *ifName*

ip address *ip-префикс* [peer *ip-адрес*] [broadcast *ip-адрес*] [anycast *ip-адрес*]

Обязательным в обоих случаях является только IP-префикс (совокупность адреса и длины маски), остальные три параметра опциональны. Если для интерфейса требуются вторичные IP-адреса (*aliases*), то их можно назначить с помощью вложенных групп:

```
{
    anycast = "ip-адрес",
    broadcast = "ip-адрес",
    peer = "ip-адрес",
    prefix = "ip-префикс"
},
```

Данные группы являются исключением из общей иерархической структуры командного меню *ui*TCP, поскольку все создаваемые ими IP-адреса равнозначны и не имеют иерархии.

Все остальные параметры, присущие виртуальному IP-интерфейсу *ui*TCP, также могут быть назначены при помощи команды pseudo-intreface основной командной оболочки (см. *Мультипротокольные маршрутизаторы NSG. Программное обеспечение NSG Linux. Руководство пользователя. Часть 4.*)

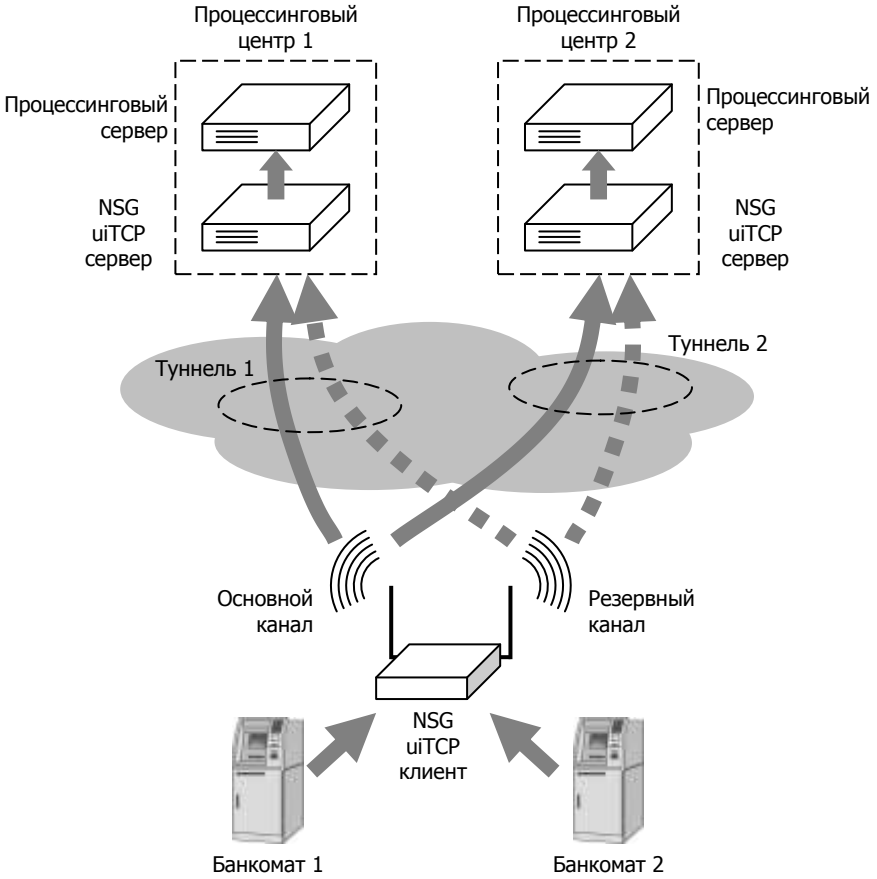
## §3.5. Настройка множественных туннелей

Один клиент *ui*TCP может поддерживать одновременно несколько туннелей к одному или к разным серверам. Поддержка множественных туннелей позволяет решить 2 задачи:

— Одновременную передачу данных по всем имеющимся каналам связи для достижения максимально возможной пропускной способности туннеля (см. п.3.6).

— Балансировку нагрузки между различными каналами.

— Обслуживание нескольких клиентских хостов, взаимодействующих с различными сетями на удалённой стороне. Например, на одной площадке может быть расположено несколько банкоматов различных банков, каждый из которых обращается к своему процессинговому центру (см. рис). Для каждого из них может быть создан отдельный туннель *ui*TCP, трафик которого изолирован от остальных.




Для описания множественных туннелей на клиенте предназначен узел меню tunnels, имеющий следующую структуру:

```
tunnels = {
    имя_туннеля1 = {
        .....
    },
    имя_туннеля2 = {
        .....
    },
    .....
}
```

Описание каждого из туннелей содержит те же узлы и параметры, что и описание одиночного туннеля в меню tunnel, за исключением следующих особенностей:

— Параметр name внутри описания туннеля отсутствует, вместо этого имя туннеля предваряет его описание.
— Узел restart следует использовать только в одном из туннелей, во избежание "игры в 4 руки".
— Загрузка новых версий *ui*TCP в данной версии производится только по туннелю, описанному в узле tunnel.

Узлы tunnel и tunnels могут использоваться совместно. Как частный случай, единственный туннель может быть описан в узле tunnels вместо tunnel.

Каждый из туннелей может использовать или не использовать любой из каналов связи, имеющихся на устройстве, в любом порядке. Например, если есть два качественно различных канала связи, скажем, Ethernet (приоритетный) и GPRS или dial-up, и три неравноценных прикладных хоста, то в нормальном режиме все три клиента могут работать через Ethernet, а если он отключается, то два переходят на GPRS, а третий перестаёт работать вообще:

```
tunnels = {
    TUNNEL1= {
        priority = true
        links = {
            gw = "123.45.67.89"
            ..................
        },
        {
            dev = s1
            ..................
        }
    },
    TUNNEL2= {
        priority = true
        links = {
            gw = "123.45.67.89"
            ..................
        },
        {
            dev = s1
            ..................
        }
    },
    TUNNEL3= {
        priority = true
        links = {
            gw = "123.45.67.89"
            ..................
        }
    }
}
```

С другой стороны, варьируя набор и очерёдность каналов в описании туннелей, можно обеспечить балансировку нагрузки между несколькими примерно равноценными каналами связи. В нижеприведённом примере при нормальной работе обоих каналов связи каждый туннель работает по своему каналу, а при отказе одного канала оба туннеля работают через оставшийся:

```
tunnels = {
    TUNNEL1= {
        priority = true
        links = {
            dev = s1
            ..................
        },
        {
            dev = s2
            ..................
        }
    },
    TUNNEL2= {
        priority = true
        links = {
            dev = s2
            ..................
        },
        {
            dev = s1
            ..................
        }
    }
}
```

## §3.6. Настройка многоканальных соединений

Многоканальный режим передачи данных возможен только для датаграммных соединений. Однако, с точки зрения применения системы *ui*TCP, это не ограничивает общности, поскольку для передачи трафика TCP можно использовать виртуальные интерфейсы (тип net).

Узел настройки многоканальных сокетов находится в корневом меню как на клиенте, так и на сервере, и имеет структуру, в основном схожую с настройками обыкновенных сокетов:

```
multiTunnelSocket = {
    имя_сокета = {
        disable = true | false
        qdisc = "roundRobin" | "fullWindow"
        type = "udp" | "raw" | "net"
        параметры udp
        параметры raw-ip                    } В зависимости
        параметры вирт.интерфейса           } от типа сокета
    },
    .....
}
```

Единственным специфическим параметром многоканального сокета является qdisc:

qdisc — Дисциплина управления выходными очередями. В данной версии поддерживаются две дисциплины:

- roundRobin — Поступающие пакеты раскладываются в выходные очереди "по кругу"; Рекомендуется во всех случаях.
- fullWindow — В каждый канал связи посылается подряд число пакетов, равное его размеру окна (windowSize); после этого посылается аналогичное число пакетов во второй канал и т.д. по кругу. Это экспериментальная дисциплина, использовать её на практике не рекомендуется.

Если выходная очередь какого-либо канала связи заполнена (по причине его низкого быстродействия, или отказа, т.е. нулевой пропускной способности), то он пропускается и пакет направляется в следующий канал. В случае пакетов TCP или другого протокола с гарантированной доставкой, если пакет будет потерян или поставлен в очередь к неработоспособному каналу связи, то принимающий хост обнаружит это и затребует повторную передачу. При этом в следующий раз пакет пойдёт уже по другому каналу, поскольку очередь этого канала будет оставаться заполненной до восстановления его работоспособности.

Управление входными очередями в многоканальном сокете не производится. Полученные пакеты передаются локальному хосту по мере их поступления. Восстановление очерёдности пакетов возлагается на его прикладное программное обеспечение.

Имя многоканального сокета, как и обыкновенного, должно быть целым числом от 1 до 64. Однако в данном случае, в отличие от одноканального сокета, это имя используется только для идентификации сокета внутри одного устройства и может никак не коррелировать с именем того же сокета на удалённой стороне.

Для построения многоканального сокета используется одноименный параметр. На клиентском устройстве он находится внутри каждого описания туннеля (как в узле tunnel, так и внутри узла tunnels). На серверном устройстве — в каждом узле client. (В данном случае одному клиентскому устройству будут соответствовать несколько узлов client на сервере.)

multiTunnelSocket

Включение туннеля в состав многоканального сокета. Значением параметра является имя сокета. Каждый туннель может быть включён только в один многоканальный сокет.

**ВНИМАНИЕ** Для работы многоканальных сокетов необходимы следующие условия:

- — Все туннели, используемые для одного сокета, должны устанавливаться к одному и тому же серверу.
- — Один из туннелей, включённых в состав сокета, должен быть обязательно описан в узле tunnel.
- — Список туннелей, включённых в описание сокета на одной и на другой стороне, должен совпадать.

Если в состав сокета не включено ни одного туннеля, то передача данных через такой сокет невозможна.

## §3.7. Системный журнал и трассировщик туннелей

Для контроля за работой *ui*TCP предназначен следующий узел меню:

```
log = {
    level = "debug" | "info" | "warn" | "error" | "fatal",
    logFile = файл,
    logFileMaxSize = байты,
    trace = байты
    параметры SQL
}
```

Значения параметров:

level — Уровень сообщений, выводимых в журнал: от только фатальных событий ("fatal") до всех событий ("debug"). Значение по умолчанию — info.

logFile — Файл журнала. Значение по умолчанию — "/var/log/uitcp".

logFileMaxSize — Максимальный размер файла журнала. Допустимые значения от 1024 до 1073741824 байт (1 Кб — 1 Гб), значение по умолчанию — 1048576 байт (1 Мб). По достижении указанного размера файл, указанный в *logFile*, переименовывается в *logFile*.old (старый файл с таким именем, если он уже существует, удаляется) и открывается новый файл. Таким образом, файлы журнала могут занимать на устройстве не более чем 2 × logFileMaxSize байт.

**ВНИМАНИЕ** На бездисковых устройствах (NSG–600, NSG–700, NSG–900 и т.п.) директория /var/log является временной и размещается на виртуальном диске в оперативной памяти устройства. При перезагрузке устройства всё её содержимое утрачивается.
Директория /etc имеет ограниченный размер и не предназначена для хранения log-файлов (в противном случае она не сможет быть сохранена в энергонезависимой памяти).
Если требуется сохранять файлы журнала при перезагрузке системы (например, для поиска причины самопроизвольных перезагрузок), то необходимо установить в систему устройство для хранения данных (USB Flash, USB HDD), смонтировать его средствами ОС Linux и разместить на нём файл журнала.

trace — Вывод трассы туннеля. Возможные значения — от 0 до 1024. Если значение данного параметра не равно нулю, то указанное число байт из каждого пакета (принимаемого или отправляемого) выводится в журнал. При значении 0 или отсутствии данного параметра трасса не выводится.

Просматривать файл журнала удобно командой tail, в том числе в реальном времени, например:

```
tail -f /var/log/uitcp
```

(Опция -f выводит новые записи по мере их поступления. Для выхода следует нажать CTRL–C.) В частности, после минимальной настройки туннеля, описанной в предыдущем параграфе, можно таким образом убедиться, что туннель установлен и работает.

Настройки для отправки записей журнала во внешнюю базу данных рассмотрены в следующем параграфе.

## §3.8. Хранение журнала во внешней базе данных

Помимо локального файла, записи журнала *ui*TCP могут храниться во внешней SQL-совместимой базе данных. Для отправки записей используется следующий узел в меню log:

```
sql = {
    database = "PostgreSQL" | "ODBC" | "MySQL" | "SQLite" | "Oracle" | "ADO",
    hostname = ip-адрес,
    port = число,
    username = строка,
    password = строка,
    sourcename = строка,
    tablename = строка,
    fields = {
        date = строка,
        hostname = строка,
        tunnelname = строка,
        level = строка,
        message = строка
    }
},
```

**ВНИМАНИЕ** Для работы с базами данных на устройстве NSG должен быть установлен клиент соответствующего типа БД для Linux. Как правило, он устанавливается только на x86-совместимых серверах. В настоящее время на практике опробована работа с БД MySQL, остальные варианты следует рассматривать как экспериментальные.

Для хранения записей *ui*TCP в базе данных необходимо предварительно создать таблицу с набором необходимых полей. Сервер *ui*TCP может передавать в БД до пяти полей: текущую дату, своё имя, имя туннеля, уровень сообщения и собственно сообщение.

Значения параметров:

database Тип базы данных.

hostname IP-адрес сервера БД (или его имя, если на устройстве включена служба DNS, или данное имя содержится в файле /etc/hosts).

port Номер порта TCP на сервере БД. Если номер порта не указан, клиент БД использует номер порта, установленный в нём по умолчанию.

username Имя пользователя для доступа к БД.

password Пароль пользователя для доступа к БД.

sourcename Имя базы данных, предназначенной для хранения записей *ui*TCP.

tablename Имя таблицы в базе данных, предназначенной для хранения записей *ui*TCP.

fields Описания полей таблицы. Значением каждого из пяти параметров, входящих в данный узел, является заголовок соответствующего столбца таблицы. (Если столбец с таким заголовком не существует, то сервер БД воспримет это как ошибку.)
Если поле отсутствует в конфигурации *ui*TCP (имеет значение nil), но присутствует в таблице, то оно заполняется значением по умолчанию согласно описанию таблицы.
Например, если в конфигурации *ui*TCP имеется непустой параметр date, то в БД отсылается текущее время сервера *ui*TCP. Если этот параметр отсутствует, то, как правило, в таблицу заносится текущее время сервера БД.

На случай, если сервер БД недоступен, в данной версии *ui*TCP предусмотрен следующий алгоритм: запись выводится в локальный файл и следующая попытка установить соединение с БД будет предпринята не ранее чем через 1 секунду; если в течение этого времени появляются новые записи, они также выводятся в локальный файл. Далее, если соединиться снова не удалось, то следующая попытка будет предпринята через 2 сек., потом через 4, 8, 16 сек. и т.д. до максимального интервала 512 сек. Увеличение интервала в 2 раза производится только после попытки вывода очередной записи. Таким образом, даже если работа сервера БД будет восстановлена, то может пройти до 512 сек., прежде чем сервер *ui*TCP попытается отправить следующую появившуюся запись в БД.

**Пример конфигурации:**

```
log = {
    level = "debug",
    sql = {
        database = "PostgreSQL",
        hostname = "10.0.0.10",
        username = "uitcp",
        password = "uitcp",
        sourcename = "uitcp",
        tablename = "log",
        fields = {
            date = nil,
            hostname = "server",
            tunnelname = "tunnel",
            level = "level",
            message = "message"
        },
    },
}
```

Предполагается, что соответствующая база данных корректно создана и описана. Вывод, который в этом случае можно увидеть с помощью клиента MySQL, приведён ниже.

```
mysql> show tables;
+-----------------+
| Tables_in_uitcp |
+-----------------+
| log             |
+-----------------+
1 row in set (0.00 sec)

mysql> describe log;
+---------+--------------+------+-----+-------------------+-------+
| Field   | Type         | Null | Key | Default           | Extra |
+---------+--------------+------+-----+-------------------+-------+
| date    | timestamp    | NO   |     | CURRENT_TIMESTAMP |       |
| level   | varchar(8)   | YES  |     | NULL              |       |
| server  | varchar(32)  | YES  |     | NULL              |       |
| tunnel  | varchar(32)  | YES  |     | NULL              |       |
| message | varchar(256) | YES  |     | NULL              |       |
+---------+--------------+------+-----+-------------------+-------+
5 rows in set (0.00 sec)

mysql> select * from log;
+---------------------+--------+--------+--------+-----------------------------------------------------------+
| date                | level  | server | tunnel | message                                                   |
+---------------------+--------+--------+--------+-----------------------------------------------------------+
| 2009-05-08 16:49:04 | INFO   | igor   |        | ============== uiTCP-work started ==============         |
| 2009-05-08 16:49:07 | INFO   | igor   |        | Server mode                                               |
| 2009-05-08 16:49:07 | INFO   | igor   |        | Waiting connection from talker on 0.0.0.0:50005           |
| 2009-05-08 16:49:07 | INFO   | igor   |        | Update task is created: 2                                 |
| 2009-05-08 16:49:07 | INFO   | igor   |        | Waiting connection from talker on 0.0.0.0:50006           |
| 2009-05-08 16:49:07 | INFO   | igor   |        | Start main loop                                           |
| 2009-05-08 16:49:21 | INFO   | box122 |        | interrupted!                                              |
| 2009-05-08 16:49:21 | INFO   | box122 |        | ============== uiTCP-0.19 exitted ==============         |
| 2009-05-08 16:49:21 | INFO   | box122 |        | ============== uiTCP-0.19 started ==============         |
| 2009-05-08 16:49:24 | INFO   | box122 |        | Server mode                                               |
| 2009-05-08 16:49:24 | INFO   | box122 |        | Waiting connection from talker on 0.0.0.0:50100           |
| 2009-05-08 16:49:24 | INFO   | box122 |        | Update task is created: 2                                 |
| 2009-05-08 16:49:24 | INFO   | box122 |        | Waiting connection from talker on 0.0.0.0:50006           |
| 2009-05-08 16:49:24 | INFO   | box122 |        | Start main loop                                           |
| 2009-05-08 16:49:31 | INFO   | box122 |        | Accept connection 10.0.10.107:2217>>10.0.54.122:50100     |
| 2009-05-08 16:49:31 | INFO   | box122 |        | Can't resume tunnel "ATM-1", no such tunnel               |
| 2009-05-08 16:49:31 | INFO   | box122 |        | Waiting connection from talker on 0.0.0.0:50023           |
| 2009-05-08 16:49:31 | INFO   | box122 | ATM-1  | New tunnel initiated from 10.0.10.107                     |
| 2009-05-08 16:49:31 | INFO   | box122 | ATM-1  | Start SSL handshake                                       |
| 2009-05-08 16:49:32 | INFO   | box122 | ATM-1  | SSL handshake successfully done                           |
+---------------------+--------+--------+--------+-----------------------------------------------------------+
```

```
mysql> select * from log where server = 'box122' and tunnel = 'ATM-1';
+---------------------+-------+--------+--------+---------------------------------------+
| date                | level | server | tunnel | message                               |
+---------------------+-------+--------+--------+---------------------------------------+
| 2009-05-08 16:36:33 | INFO  | box122 | ATM-1  | New tunnel initiated from 192.168.1.10 |
| 2009-05-08 16:36:33 | INFO  | box122 | ATM-1  | Start SSL handshake                   |
| 2009-05-08 16:36:34 | INFO  | box122 | ATM-1  | SSL handshake successfully done       |
| 2009-05-08 16:49:31 | INFO  | box122 | ATM-1  | New tunnel initiated from 10.0.10.107  |
| 2009-05-08 16:49:31 | INFO  | box122 | ATM-1  | Start SSL handshake                   |
| 2009-05-08 16:49:32 | INFO  | box122 | ATM-1  | SSL handshake successfully done       |
+---------------------+-------+--------+--------+---------------------------------------+
6 rows in set (0.00 sec)

mysql> select * from log where date >= '2009-05-08 16:49:07' and date <= '2009-05-08 16:49:31';
+---------------------+-------+--------+--------+-----------------------------------------------------------+
| date                | level | server | tunnel | message                                                   |
+---------------------+-------+--------+--------+-----------------------------------------------------------+
| 2009-05-08 16:49:07 | INFO  | igor   |        | Server mode                                               |
| 2009-05-08 16:49:07 | INFO  | igor   |        | Waiting connection from talker on 0.0.0.0:50005           |
| 2009-05-08 16:49:07 | INFO  | igor   |        | Update task is created: 2                                 |
| 2009-05-08 16:49:07 | INFO  | igor   |        | Waiting connection from talker on 0.0.0.0:50006           |
| 2009-05-08 16:49:07 | INFO  | igor   |        | Start main loop                                           |
| 2009-05-08 16:49:21 | INFO  | box122 |        | interrupted!                                              |
| 2009-05-08 16:49:21 | INFO  | box122 |        | ============== uiTCP-0.19 exitted ==============       |
| 2009-05-08 16:49:21 | INFO  | box122 |        | ============== uiTCP-0.19 started ==============       |
| 2009-05-08 16:49:24 | INFO  | box122 |        | Server mode                                               |
| 2009-05-08 16:49:24 | INFO  | box122 |        | Waiting connection from talker on 0.0.0.0:50100           |
| 2009-05-08 16:49:24 | INFO  | box122 |        | Update task is created: 2                                 |
| 2009-05-08 16:49:24 | INFO  | box122 |        | Waiting connection from talker on 0.0.0.0:50006           |
| 2009-05-08 16:49:24 | INFO  | box122 |        | Start main loop                                           |
| 2009-05-08 16:49:31 | INFO  | box122 |        | Accept connection 10.0.10.107:2217>>10.0.54.122:50100     |
| 2009-05-08 16:49:31 | INFO  | box122 |        | Can't resume tunnel "ATM-1", no such tunnel               |
| 2009-05-08 16:49:31 | INFO  | box122 |        | Waiting connection from talker on 0.0.0.0:50023           |
| 2009-05-08 16:49:31 | INFO  | box122 | ATM-1  | New tunnel initiated from 10.0.10.107                     |
| 2009-05-08 16:49:31 | INFO  | box122 | ATM-1  | Start SSL handshake                                       |
+---------------------+-------+--------+--------+-----------------------------------------------------------+
18 rows in set (0.01 sec)
```

```
mysql> select server, tunnel, message from log where date >= '2009-05-08 16:49:07';
+--------+--------+-----------------------------------------------------+
| server | tunnel | message                                             |
+--------+--------+-----------------------------------------------------+
| igor   |        | Server mode                                         |
| igor   |        | Waiting connection from talker on 0.0.0.0:50005     |
| igor   |        | Update task is created: 2                           |
| igor   |        | Waiting connection from talker on 0.0.0.0:50006     |
| igor   |        | Start main loop                                     |
| box122 |        | interrupted!                                        |
| box122 |        | ============== uiTCP-0.19 exitted ==============    |
| box122 |        | ============== uiTCP-0.19 started ==============    |
| box122 |        | Server mode                                         |
| box122 |        | Waiting connection from talker on 0.0.0.0:50100     |
| box122 |        | Update task is created: 2                           |
| box122 |        | Waiting connection from talker on 0.0.0.0:50006     |
| box122 |        | Start main loop                                     |
| box122 |        | Accept connection 10.0.10.107:2217>>10.0.54.122:50100 |
| box122 |        | Can't resume tunnel "ATM-1", no such tunnel         |
| box122 |        | Waiting connection from talker on 0.0.0.0:50023     |
| box122 | ATM-1  | New tunnel initiated from 10.0.10.107               |
| box122 | ATM-1  | Start SSL handshake                                 |
| box122 | ATM-1  | SSL handshake successfully done                     |
+--------+--------+-----------------------------------------------------+
19 rows in set (0.00 sec)
```

## §3.9. Общесистемные настройки

Для "мягкого" выключения сервера в плановом режиме (например, на техобслуживание или для установки новой версии программного обеспечения) предназначен следующий параметр в корневом меню:

```
SHUTDOWN = true | false
```

При установленном значении true сервер не отвечает на пакеты keepalive и на запросы на установление новых туннелей. Таким образом, клиенты, подключённые в данный момент к серверу, вынуждаются переключиться на резервный сервер по мере завершения текущих транзакций. После того, как все клиенты перейдут на резервный сервер, данный сервер может быть остановлен с минимальным ущербом для работы системы в целом (без аварийно завешённых транзакций и т.п.).

При настройке с помощью uish данный параметр вступает в силу без команды _apply. Для восстановления работоспособности сервера следует установить значение SHUTDOWN = false.

**ПРИМЕЧАНИЕ** Завершение текущей транзакции детектируется клиентом *ui*TCP как периоды достаточно длительного отсутствия пользовательского трафика. Таким образом, если прикладные хосты непрерывно обмениваются какими-либо данными (например, собственными пакетами *keepalive* прикладного уровня) с достаточно малыми интервалами, то такая транзакция, с точки зрения *ui*TCP, продолжается бесконечно долго, и "мягко" разорвана быть не может. Такие клиенты могут быть отключены только непосредственным выключением сервера или рестартом uitcp на нём; передаваемые в этот момент данные будут утеряны.

Узел меню sys предназначен для хранения общесистемных служебных настроек:

```
sys {
    pluginFolder = "директория"
}
```

В данной версии *ui*TCP этот узел содержит единственный параметр:

pluginFolder Директория, в которую будут помещаться дистрибутивные файлы uitcp-X.Y для раздачи клиентам. Использование данного параметра имеет смысл только на сервере. Значение по умолчанию "/etc/uitcp/plugins".

Узел show содержит команды для просмотра статуса и статистики туннелей. Данный узел не входит в состав конфигурации и не сохраняется в конфигурационном файле. Эти команды доступны только при использовании uish или Web-интерфейса:

show connections
: Вывести список туннелей, определённых на устройстве, и их состояние.

show stat = "имя"
: Вывести статистику для туннеля с указанным именем.

show certificate
: Вывести информацию о сроках действия всех сертификатов X.509, используемых *ui*TCP на данном устройстве.

show version
: Вывести версию *ui*TCP.

# §4. Настройка безопасности

## §4.1. Общие сведения о безопасности в *ui*TCP

Безопасность обмена данными по туннелю *ui*TCP обеспечивается с помощью библиотеки OpenSSL. Система предусматривает аутентификацию сторон (одно- или двустороннюю) с использованием сертификатов X.509 и асимметричное шифрование с длиной ключа до 2048 бит. Таким образом, возможности *ui*TCP в этом отношении эквивалентны SSH, STunnel, OpenVPN, HTTPS и другим протоколам, использующим SSL. Подробная информация об организации SSL-соединений доступна в соответствующей литературе и на Web-ресурсах, посвящённых SSL.

Для взаимной аутентификации и защиты передаваемых данных каждая из сторон должна располагать следующими обязательными атрибутами, которые передаются непосредственно в библиотечные функции openssl:

— Корневым сертификатом, с помощью которого устанавливается подлинность удалённой стороны.
— Собственным сертификатом, который передаётся удалённой стороне по её требованию; сертификат содержит в себе открытый ключ.
— Собственным закрытым ключом и, при необходимости, паролем к нему.
— Списком действительных либо недействительных (отозванных) сертификатов, либо набором этих сертификатов полностью — в зависимости от алгоритма их проверки. По существу, требуется только на сервере (для контроля скомпрометированных клиентов), поскольку компрометация сервера — событие экстраординарное и безусловно требующее ручной перенастройки всей системы безопасности. В данной реализации *ui*TCP используется проверка по набору действительных сертификатов. В случае компрометации клиента его сертификат просто удаляется.

Ключи и сертификаты генерируются стандартными средствами OpenSSL и могут быть созданы как на одном из устройств NSG, так и на стороннем устройстве. Ключи и сертификаты представляют собой файлы в формате .pem, который является стандартом де-факто. Созданные сертификаты (как правило, на сервере *ui*TCP или на выделенном сервере безопасности) переносятся на клиентские устройства посредством USB Flash или локальной сети.

Для ускорения поиска среди большого числа сертификатов они записываются также в виде *индексных файлов*. Индексный файл содержит тот же самый сертификат, но имя файла формируется специальным образом: оно представляет собой хэш от поля Subject этого сертификата (с добавлением счётчика, если хэши для двух сертификатов совпадают).

Сервер *ui*TCP может использовать как один набор из сертификата и закрытого ключа, так и индивидуальные наборы для каждого клиента или группы клиентов. Корневые сертификат и ключ, на основе которых генерируются прочие сертификаты и ключи, также могут быть общими или индивидуальными.

Список действительных сертификатов хранится в директории, указанной параметром capath (см. ниже), в виде набора индексных файлов.

Настройка SSL на клиенте и на сервере производится аналогичным образом, различаются только узлы конфигурационного дерева, в которых размещаются эти параметры. На клиенте узел ssl входит в состав узла tunnel, на сервере — в состав описания данного клиента. Кроме того, на сервере могут быть определены глобальные настройки SSL, имеющие силу для всех клиентов. Каждый из параметров, независимо от остальных, может быть описан на сервере как локально (для определённого клиента), так и глобально. Если параметр описан дважды, то приоритетным является значение, указанное индивидуально для клиента.

**ВНИМАНИЕ** Сертификаты имеют конечный срок действия. Во избежание потери связи с клиентами необходимо заблаговременно заменять сертификаты, срок действия которых истекает.

**ВНИМАНИЕ** Для работы SSL необходимо, чтобы на всех устройствах было корректно установлено системное время. Настоятельно рекомендуется синхронизировать все устройства от единого сервера NTP, доступного по открытому каналу или, как минимум, без использования сертификатов.

**ПРИМЕЧАНИЕ** На клиентском устройстве также возможна установка глобальных параметров безопасности (в корне конфигурационного дерева), однако это не имеет смысла, поскольку туннель единственный. Во избежание неоднозначностей и ошибок конфигурации, рекомендуется настраивать SSL только внутри узла tunnel.

## §4.2. Создание и индексирование сертификатов

Генерацию ключей и сертификатов (если они не предоставлятся сторонним удостоверяющим центром) рекомендуется выполнять на сервере *ui*TCP с помощью набора утилит cert-*, предназначенных для этой цели (см. п.5.6). Файлы, необходимые для каждого клиента, упаковываются в отдельный архив для удобства переноса.

Для работы SSL необходимо на сервере иметь файлы root.pem, server.pem и serverkey.pem, а на клиенте — root.pem, client.pem и clientkey.pem. Рекомендуется расположить все вышеперечисленные файлы *.pem в директории /etc/uitcp/certs, принятой по умолчанию — как на клиенте, так и на сервере. Корневой сертификат и сертификат сервера могут быть общими для всех клиентов. Если для каких-либо клиентов требуются специфические файлы на сервере, то их рекомендуется размещать в директории /etc/uitcp/certs/*имя_клиента* .

Если устройство использует несколько корневых сертификатов, то можно либо включить их все в файл root.pem, либо разместить в директории capath (см. ниже) и выполнить скрипт rehash. Он создаёт индексные ссылки на сертификаты с дóлжными именами. Скрипт хранится, по умолчанию, на клиенте в директории /etc/uitcp/bin/tools/, на сервере — /opt/uitcp/tools/ .

Если включена проверка клиентов по списку действительных сертификатов, то на сервере требуется также создать индексные файлы или индексные ссылки на клиентские сертификаты. Для этого следует поместить все клиентские сертификаты в директорию capath и выполнить скрипт rehash. При компрометации сертификата или истечении его срока действия следует удалить сертификат; мёртвая ссылка на него удалится автоматически при следующем исполнении rehash.

Если ключи и сертификаты генерируются сторонним центром сертификации, то их необходимо поместить на сервер и на клиентов *ui*TCP, соответственно, и указать соответствующие файлы и директории в разделе SSL настроек *ui*TCP. Рекомендуется использовать имена файлов и директорий и схему размещения файлов, принятую в *ui*TCP по умолчанию; в этом случае никаких специальных настроек SSL не требуется, что существенно упрощает настройку системы.

## §4.3. Параметры SSL

Набор параметров безопасности в *ui*TCP дублирует стандартные опции OpenSSL. Имена параметров, по возможности, совпадают с именами соответствующих опций, или аналогичны им, поэтому настройка безопасности не представляет особых сложностей для специалиста, знакомого с SSL.

Для сервера *ui*TCP узел ssl может находиться внутри описания каждого клиента (локальные настройки) либо в корне конфигурационного дерева (глобальные настройки). Для клиента данный узел находится внутри узла tunnel. Во всех случаях узел содержит один и тот же набор параметров:

```
ssl = {
    cafile = "файл",
    capath = "путь",
    certcheck = true | false,
    certificate = "файл",
    ciphers = "список",
    depth = число,
    disable = true | false,
    key = "файл",
    options = "список",
    password = "пароль",
    path = "путь",
    protocol = "tlsv1" | "sslv3" | "sslv23",
    verify = true | false
}
```

которые можно разделить на 4 группы.

**а) Общие параметры SSL**

disable = true | false

Выключение и включение защиты данных с помощью SSL. Значение true отключает как аутентификацию сторон, так и шифрование; в этом случае устанавливается незащищённый туннель, и все остальные параметры SSL не имеют смысла. Параметр должен иметь одинаковое значение на обеих сторонах, в противном случае никакое соединение не будет установлено.

path — Директория, в которой следует искать все файлы .pem, которые не указаны в конфигурации явным образом. Значение path для отдельного клиента или туннеля может быть пустым; глобальное значение всегда непустое и по умолчанию равно "etc/uitcp/certs/".

protocol = "tlsv1" | "sslv3" | "sslv23"

Выбор протокола: TLS *ver.1*, только SSL *ver.3*, либо SSL *ver.2* или *3*. По умолчанию используется только SSL *ver.3*, как наиболее безопасный. Какая-либо необходимость в использовании TLS или SSL *ver.2* в настоящее время отсутствует, поскольку *ui*TCP представляет собой единую систему. Данный параметр, скорее, можно считать зарезервированным на случай появления последующих версий и модификаций протокола SSL.

ciphers = *список*

Список поддерживаемых шифров. Формат списка описан в документе:
*http://www.openssl.org/docs/apps/ciphers.html* .
По умолчанию, разрешены все шифры, соответствующие выбранной версии протокола SSL.

options = { *опция1*, *опция2*, ... }

Дополнительные опции SSL. Поддерживаются следующие стандартные опции из группы SSL_OP_*XXX*:

| | |
|---|---|
| "all" | "no_sslv3" |
| "cipher_server_preference" | "no_tlsv1" |
| "dont_insert_empty_fragments" | "pkcs1_check_1" |
| "ephemeral_rsa" | "pkcs1_check_2" |
| "netscape_ca_dn_bug" | "single_dh_use" |
| "netscape_challenge_bug" | "ssleay_080_client_dh_bug" |
| "microsoft_big_sslv3_buffer" | "sslref2_reuse_cert_type_bug" |
| "microsoft_sess_id_bug" | "tls_block_padding_bug" |
| "msie_sslv2_rsa_padding" | "tls_d5_bug" |
| "netscape_demo_cipher_change_bug" | "tls_rollback_bug" |
| "netscape_reuse_cipher_change_bug" | "cookie_exchange" |
| "no_session_resumption_on_renegotiation" | "no_query_mtu" |
| "no_sslv2" | "single_ecdh_use" |

Подробное описание данных опций доступно по адресу:
*http://www.openssl.org/docs/ssl/SSL_CTX_set_options.html* .
Рекомендуется использовать следующий минимальный набор опций:
options = {"all", "no_sslv2"}

**б) Параметры удостоверяющего центра (корневого сертификата)**

cafile — Полный путь и имя файла, содержащего один или несколько корневых сертификатов. Фактическое имя файла, которое передаётся openssl, выбирается следующим образом, в порядке приоритета:
— значение, установленное для конкретного клиента или для туннеля;
— *local_path*/*имя_клиента*/root.pem;
— *local_path*/root.pem;
— глобальное значение (для сервера);
— *global_path*/*имя_клиента*/root.pem (для сервера);
— *global_path*/root.pem (для сервера).

capath — Директория, в которой следует искать корневые сертификаты. Для поиска файлы сертификатов должны быть проиндексированы скриптом rehash. Фактическое значение capath, которое передаётся openssl, выбирается следующим образом, в порядке приоритета:
— значение, установленное для конкретного клиента или для туннеля;
— *local_path*/*имя_клиента*/capath;
— *local_path*/capath;
— глобальное значение (для сервера);
— *global_path*/*имя_клиента*/capath (для сервера);
— *global_path*/capath (для сервера).
Поскольку глобальный путь всегда непустой, то глобальный параметр capath также всегда определен и по умолчанию равен "etc/uitcp/certs/capath".

При получении сертификата от удалённой стороны корневой сертификат, на который он ссылается, сравнивается последовательно со всеми сертификатами в файле cafile и в директории cafile, вычисленных по вышеописанным правилам.

Если ни одного совпадения не найдено, устройство отказывает удалённой стороне в аутентификации.

depth — Глубина проверки цепочки вложенных сертификатов — целое число. По умолчанию, глубина не ограничена.

**в) Описание данного устройства в SSL-соединении**

certificate — Полный путь и имя файла, содержащего сертификат данного устройства. Этот сертификат передаётся для аутентификации на удалённой стороне.
Фактическое имя файла, которое передаётся openssl, всегда непустое и выбирается следующим образом, в порядке приоритета:
а) для клиента:
— значение, установленное для туннеля явным образом;
— *path*|*имя_клиента*|client.pem;
— *path*|client.pem;
б) для сервера:
— значение, установленное для конкретного клиента явным образом;
— *local_path*|*имя_клиента*|server.pem;
— *local_path*|server.pem;
— глобальное значение (для сервера);
— *global_path*|*имя_клиента*|server.pem;
— *global_path*|server.pem.

password — Пароль для доступа к закрытому ключу в случае, если ключ зашифрован.

key — Полный путь и имя файла, содержащего закрытый ключ данного устройства. Этот ключ используется для шифрования в паре с открытым ключом, который передаётся в сертификате.
Фактическое имя файла, которое передаётся openssl, всегда непустое и выбирается следующим образом, в порядке приоритета:
а) для клиента:
— значение, установленное для туннеля явным образом;
— *path*|*имя_клиента*|clientkey.pem;
— *path*|clientkey.pem;
б) для сервера:
— значение, установленное для конкретного клиента явным образом;
— *local_path*|*имя_клиента*|serverkey.pem;
— *local_path*|serverkey.pem;
— глобальное значение (для сервера);
— *global_path*|*имя_клиента*|serverkey.pem;
— *global_path*|serverkey.pem.

**г) Аутентификация удалённой стороны данным устройством**

verify = { *опция1*, *опция2*, ... }
Опции аутентификации удалённой стороны на данном устройстве SSL. Поддерживаются следующие стандартные опции из группы SSL_VERIFY_*XXX*:

| | |
|---|---|
| "none" | "client_once" |
| "peer" | "fail_if_no_peer_sert" |

Подробное описание данных опций доступно по адресу:
*http://www.openssl.org/docs/ssl/SSL_CTX_set_verify.html* .
Рекомендуется использовать следующий набор опций:
verify = { "peer", "fail_if_no_peer_sert" }

certcheck = true | false
Проверка сертификата, полученного от удалённой стороны, по списку действительных. Действительные сертификаты ищутся в директории capath. Если полученный сертификат не совпадает ни с одним из имеющихся в данной директории, соединение отвергается.

# §5. Особенности использования *ui*TCP на серверах

## §5.1. Особенности программного обеспечения серверов *ui*TCP

Программное обеспечение NSG Linux специализированных серверов *ui*TCP отличается от программного обеспечения маршрутизаторов по следующим основным пунктам:

1. На прикладные сервера не устанавливается файловая система NSG, содержащая средства и службы маршрутизации, коммутации и мультипротокольной инкапсуляции пакетов и командную оболочку vtysh. Устанавливается только базовая ОС Linux (для x86 серверов — на основе дистрибутива Gentoo Linux).
2. На серверах, оснащённых накопителем данных большого объёма (HDD или SSD), программное обеспечение устанавливается в развёрнутом виде на этот накопитель.
3. Управление собственно сервером осуществляется средствами ОС Linux; в нормальной ситуации оно заключается в выполнении нескольких простейших базовых настроек для работы сервера в сети IP.
4. Для установки *ui*TCP могут использоваться любые директории в файловой системе. В директории /etc рекомендуется хранить, как это принято в ОС Linux, только настройки. Пути для установки различных компонент *ui*TCP могут быть указаны в специальном файле /etc/nsgSysConfig или явным образом при вызове программ.

## §5.2. Вход и общая настройка сервера

Для входа на сервер через консольный порт используются, по умолчанию, следующие реквизиты:

```
login: root
password: nsg
```

Знакомство с Linux:

```
servbox ~ # less /README
servbox ~ # less http://www.gentoo.org/doc/en/handbook/handbook-x86.xml
```

Назначение имени хоста и имени домена. Данные параметры хранятся в текстовых файлах /etc/hosts и /etc/conf.d/hostname, соответственно. Просмотреть их можно при помощи команды cat:

```
servbox ~ # cat /etc/hosts

127.0.0.1    box2.at.place1 box2 localhost
::1          localhost

servbox ~ # cat /etc/conf.d/hostname

# /etc/conf.d/hostname
# Set to the hostname of this machine
HOSTNAME=" servbox"
```

Для редактирования файлов конфигурации можно использовать текстовый редактор nano.

Настройка сетевых интерфейсов. Данные параметры хранятся в файле /etc/conf.d/net.

```
servbox ~ # cat /etc/conf.d/net

# This blank configuration will automatically use DHCP for any net.*
# scripts in /etc/init.d.  To create a more complete configuration,
# please review /etc/conf.d/net.example and save your configuration
# in /etc/conf.d/net (this file :]!).

config_eth0=( "192.168.0.3/24" )
routes_eth0=( "default via 192.168.0.1" )
dns_servers_eth0=( "4.2.2.1 4.2.2.2 208.67.222.222 208.67.220.220" )
metric_eth0=( "1" )

config_eth1=( "10.0.54.113/8" "123.145.167.189/28")
routes_eth1=( "default via 10.0.0.30" )
dns_servers_eth1=( "10.0.0.128 4.2.2.1 4.2.2.2 208.67.222.222 208.67.220.220" )
metric_eth1=( "0" )
```

На сервер можно зайти через Telnet или SSH, например:

```
ssh root@10.0.54.113
Password: nsg
```

Подробно об основных командах ОС Linux для управления устройствами NSG см. документ NSG: *Мультипротокольные маршрутизаторы NSG. Программное обеспечение NSG Linux. Руководство пользователя. Часть 6.*

## §5.3. Настройка дополнительных служб

Параметры службы сетевого времени содержатся в файле /etc/conf.d/ntp-client:

```
servbox ~ # cat /etc/conf.d/ntp-client
# /etc/conf.d/ntp-client
# Command to run to set the clock initially
# Most people should just leave this line alone ...
# however, if you know what you're doing, and you
# want to use ntpd to set the clock, change this to 'ntpd'
NTPCLIENT_CMD="ntpdate"
# Options to pass to the above command
# This default setting should work fine but you should
# change the default 'pool.ntp.org' to something closer
# to your machine.  See http://www.pool.ntp.org/ or
# try running 'netselect -s 3 pool.ntp.org'.
NTPCLIENT_OPTS="-s -b -u \
0.gentoo.pool.ntp.org 1.gentoo.pool.ntp.org \
2.gentoo.pool.ntp.org 3.gentoo.pool.ntp.org"
```

Настройка брандмауэра, если таковой необходим, производится стандартными средствами iptables.

Смена пароля для пользователя root производится стандартной командной passwd.

## §5.4. Обновление основного программного обеспечения

Для обновления программного обеспечения Linux на x86-совместимых серверах следует использовать систему управления пакетами Portage, стандартную для Gentoo Linux. Описание системы приведено в документах на официальном сайте http://www.gentoo.org/:

*Working with Gentoo*
*Working with Portage*

Для выполнения обновления необходимо обеспечить доступ к репозиториям, указанным в конфигурационном файле /etc/make.conf по порту rsync (TCP 873).

В случае, если после установки новой версии того или иного пакета требуется разрешить конфликты между существующей конфигурацией от предыдущей версии и конфигурацией по умолчанию от новой версии, следует руководствоваться официальными документами по Gentoo Linux, в частности: *SSH Configuration Guide for Gentoo Infrastructure Servers*.

## §5.5. Особенности конфигурирования *ui*TCP на серверах

На x86-совместимых серверах *ui*TCP, оснащённых накопителем HDD или SSD, система устанавливается в следующие директории:

Бинарные файлы — /opt/uitcp
Файлы конфигурации — /etc/uitcp

Для удобства сделаны символьные ссылки:

```
ln -s /opt/uitcp/uitcp /usr/sbin/uitcp
ln -s /opt/uitcp/tools/uish /usr/bin/uish
ln -s /opt/uitcp/tools/uitcp-update-sftp /usr/bin/uitcp-update-sftp
```

Автозапуск uitcp прописан в файле /etc/inittab, последняя строка:

```
null::respawn:/usr/sbin/uitcp
```

Для перезапуска uitcp надо удалить этот процесс:

```
servbox ~ # ps -A
  PID    TTY        TIME       CMD
    1    ?          00:00:00   init
...
 4001    ?          00:00:07   uitcp
 6974    ?          00:00:00   sshd
 6977    pts/0      00:00:00   bash
 7438    pts/0      00:00:00   ps
servbox ~ # kill 4001
```

или

```
servbox ~ # killall uitcp
```

Для обновления *ui*TCP используется протокол SFTP вместо TFTP:

```
box1 ~ # uitcp-update-sftp basile@10.0.1.2 /soft/uitcp-1.2

Connecting to 10.0.1.2...
Password:
Fetching /soft/uitcp-1.2 to uitcp-1.2
/soft/uitcp-0.15                    100%   44KB  43.9KB/s   00:00
plugin will install to /opt/uitcp
Installation finished
```

Утилита uitcp-update-sftp воспринимает три параметра:

uitcp-update-sftp user@host *имя_файла хранилище путь_установки*

где:

*имя_файла* Полный путь и имя файла с новой версией плагина uitcp-X.Y на сервере SFTP. Параметр обязательный. Если путь не указан или указан не полностью от корня (без начального /), тогда запрошенный файл ищется на сервере относительно домашней директории данного пользователя. Например,
uitcp-update-sftp basile@10.0.1.2 /soft/uitcp-1.2 — ищет файл /soft/uitcp-1.2
uitcp-update-sftp basile@10.0.1.2 soft/uitcp-1.2 — ищет файл /home/basile/soft/uitcp-1.2
uitcp-update-sftp basile@10.0.1.2 uitcp-1.2 — ищет файл /home/basile/uitcp-1.2

*хранилище* Имя директории на сервере *ui*TCP, куда будет помещен плагин для последующей раздачи клиентам. Если этот параметр отсутствует, то плагин помещается в директорию plugins внутри директории, определенной параметром uitcpConfigPath в файле /etc/nsgSysConfig (в примере из п.2.8 это будет /home/user/uitcp/plugins). Если этот параметр отсутствует в файле, или отсутствует сам файл /etc/nsgSysConfig, то плагин, по умолчанию, помещается в директорию /etc/uitcp/plugins.

*путь_установки*
Передается как параметр при запуске плагина uitcp-X.Y.

**ПРИМЕЧАНИЕ** При первичной установке системы необходимо вручную создать директорию, которая будет использоваться в качестве хранилища (по умолчанию, /etc/uitcp/plugins).

## §5.6. Централизованное управление ключами и сертификатами в системе

На сервере *ui*TCP имеются три утилиты для управления сертификатами cert-root, cert-server и cert-client. Они находятся в директории /opt/uitcp/tools/ (или иной, указанной при установке системы). Для удобства сделаны символьные линки

```
ln -s /opt/uitcp/tools/cert-root /usr/sbin
ln -s /opt/uitcp/tools/cert-server /usr/sbin
ln -s /opt/uitcp/tools/cert-client /usr/sbin
```

Перед первым использованием этих утилит необходимо выполнить следующие действия:

— Создать директорию /etc/uitcp/certs.conf и создать в ней файлы конфигурации сертификатов. В простейшем случае можно скопировать в неё образцы сертификатов из /opt/uitcp/certs.conf и отредактировать их должным образом (указать имя организации и т.п.).

— Создать директории /etc/uitcp/certs и /etc/uitcp/certs/capath .

Если при установке системы вместо /etc/uitcp был настроен иной путь для хранения конфигурационных файлов, то вышеупомянутые директории изменяются соответствующим образом.

Пример:

```
cp /opt/uitcp/certs.conf /etc/uitcp
mkdir /etc/uitcp/certs
mkdir /etc/uitcp/certs/capath
```

Утилита cert-root создаёт файлы root.pem и rootkey.pem и помещает их в директорию /etc/uitcp/certs/_root/ .

Утилита cert-server создаёт ключ и сертификат сервера serverkey.pem и server.pem, подписанные корневым сертификатом, и помещает их в директорию /etc/uitcp/certs/ .

Утилита cert-client вызывается с ключом, которым является имя будущего клиента, например:

```
./cert-client ATM00123
```

и выполняет следующие действия:

— Создаёт на сервере директорию /etc/uitcp/certs/*имя_клиента* .

— Создаёт ключ и сертификат клиента clientkey.pem и client.pem, подписанные корневым сертификатом. При генерации каждого клиентского сертификата в него подставляется уникальное имя клиента (вместо заданного в файле конфигурации).

— Помещает в эту же директорию файл корневого сертификата root.pem.

— Создаёт в этой же директории архив с именем cert-client.*имя_клиента*.tgz, содержащий набор файлов, необходимый для клиента. Этот файл необходимо перенести на клиентское устройство и разархивировать в директорию /etc/uitcp/certs на нём. Пример:

```
mkdir  /etc/uitcp/certs
cd  /etc/uitcp/certs
sftp  root@10.0.1.2  /etc/uitcp/certs/ATM00123/cert-client.ATM00123.tgz
tar  -xzf  ./cert-client.ATM00123.tgz
rm  cert-client.ATM00123.tgz
```

**ВНИМАНИЕ** По соображениям безопасности, недопустима передача файла с сертификатами в открытом виде через Интернет. Переносить сертификаты возможно:
— на USB Flash
— по временному прямому соединению кросс-кабелем Ethernet между сервером и клиентским устройством (рекомендуется)
— по защищенной локальной сети организации
— по защищённому соединению с работающим клиентом (SSH, SFTP, *ui*TCP и др.)

— Перемещает файл client.pem в директорию /etc/uitcp/capath под именем clientcert.*имя_клиента*.pem и пересоздает индексные ссылки на все файлы клиентских сертификатов. Для отзыва сертификата конкретного клиента следует удалить указанный файл.

Файл clientkey.pem на стороне сервера не используется и после выполнения процедуры удаляется.

Как можно видеть, все три утилиты устанавливают итоговые файлы и директории согласно соглашениям, принятым по умолчанию, поэтому, если создавать сертификаты только с их помощью, то в конфигурации uitcp можно не настраивать параметры SSL вообще.

## §5.7. Централизованное обновление ключей и сертификатов

Замена сертификатов и ключей может производиться централизованно с сервера *ui*TCP. При этом на момент обновления сервер и клиент должны иметь работоспособные старые сертификаты, т.е. замена должна обязательно производиться до истечения срока их действия.

Путь к папке с сертификатами на сервере (например, etc/uitcp/certs — в зависимости от типа устройства и пользовательской конфигурации) далее обозначается как /CERTS.

**а) Самоподписанные сертификаты.**

Если ключи и сертификаты генерируются на сервере *ui*TCP и используются имена и пути, установленные по умолчанию, то порядок выполнения процедуры следующий:

1. Удалить старый корневой сертификат:

   ```
   # rm /CERTS/_root/root.pem
   ```

2. Создать новый корневой ключ и сертификат:

   ```
   # cert-root
   ```

3. Удалить старый сертификат сервера:

   ```
   # rm /CERTS/_server/server.pem
   ```

   Это копия сертификата, необходимая для создания клиентских сертификатов. На сервере в это время продолжает работать старый сертификат /CERTS/server.pem.

4. Создать новый сертификат и ключ сервера:

   ```
   # cert-server
   ```

   Новый сертификат и ключ сервера создается во вспомогательной папке /CERTS/_server/ и пока не используется, так как на клиента пока еще не послан новый корневой сертификат.

5. Удалить старый архив с ключами и сертификатами из папки клиента:

   ```
   # rm /CERTS/имя/cert-client.name.tgz
   ```

6. Создать новый сертификат и ключ клиента:

   # cert-client *имя*

   При этом в CA-файл /CERTS/*имя*/root.pem будет добавлен новый сертификат для проверки клиента. Старый при этом не стирается, т.е. с этого момента сервер может принимать и старый, и новый сертификаты клиента.

7. Передать новые ключ, сертификат и CA файл на клиента. Для этого необходимо в программе uish войти в меню clients.*имя*.tools и выполнить команду secrets = default:

   uitcp.clients.*имя*.tools.secrets = default

   По этой команде новые ключ, сертификат и CA-файл будут переданы на клиента и записаны в файлы в соответствии с конфигурацией на клиенте. Старый CA-файл на клиенте не стирается, т.е. с этого момента клиент может принимать и старый, и новый сертификаты сервера.

8. Повторить пп. 5–7 для всех клиентов.

9. После того, как обновление сертификатов произведено на всех клиентах, можно установить новый сертификат и ключ на сервере. Для этого нужно скопировать их из вспомогательной папки в рабочую:

   # cp /CERTS/_server/* /CERTS

**б) Готовые ключи и сертификаты.**

Если используются ключи и сертификаты, выданные сторонним удостоверяющим центром, то порядок выполнения процедуры следующий:

1. Добавить новый CA сертификат в соответствии с конфигурацией:
   — либо добавить новый CA файл в директорию capath и выполнить команду rehash;
   — либо добавить сертификат в cafile (например, с помощью текстового редактора).
   Необходимо, чтобы на сервере одновременно существовали старый и новый CA сертификаты на период замены сертификатов на клиентах.

2. Поместить новые ключ, сертификат и CA-файл на сервер, например, в рабочую директорию клиента.

3. Передать новые ключ, сертификат и CA-файл на клиента. Передать новые ключ, сертификат и CA файл на клиента. Для этого необходимо в программе uish войти в меню clients.*имя*.tools и выполнить команду secrets в следующем формате:

   uitcp.clients.name.tools.secrets = {c="cert",k="key",r="root"}

   где cert — имя файла с новым сертификатом
   key — имя файла с новым ключом
   root — имя нового CA-файла с сертификатом

   По этой команде новые ключ, сертификат и CA-файл будут переданы на клиента и записаны в файлы в соответствии с конфигурацией на клиенте. Старый CA-файл на клиенте не стирается, т.е. с этого момента клиент может принимать и старый, и новый сертификаты сервера.

   Имена файлов должны быть записаны в виде полного пути от корня файловой системы, например:

   {c="/new/ctrt/folder/cl007_cert.pem",k="/new/ctrt/folder/cl007_key.pem",r="/new/ctrt/folder/CAxxx.pem"}

   Если файлы помещены в стандартную папку /CERTS/*имя*/cert-client, то можно задавать только имена файлов, например:

   {c="cl007_cert.pem",k="cl007_key.pem",r="CAxxx.pem"}

   Если файлы помещены в папку /CERTS/*имя*/cert-client и имеют имена стандартные имена client.pem, clientkey.pem, root.pem, то в качестве параметра можно задать

   default

   Сертификат и ключ клиента могут быть в одном файле, например:

   {c="cl007.pem",k="cl007.pem",r="CAxxx.pem"}

4. Повторить пп. 2–3 для всех клиентов.

5. После того, как обновление сертификатов произведено на всех клиентах, можно установить новый сертификат и ключ на сервере. Для этого нужно скопировать их в рабочую директорию в соответствии с конфигурацией сервера.

Рекомендуется сохранять старые ключ и сертификат сервера до успешного завершения процедуры на случай, если новые сертификаты или ключи не установятся на каких-либо клиентах.