Маршрутизаторы NSG

Программное обеспечение NSG Linux 2.0

Руководство пользователя

Часть 6

Система обеспечения бесперебойных соединений

*ui*TCP

Версия программного обеспечения 2.0 build 7

Обновлено 10.11.2016

Москва 2016

АННОТАЦИЯ

Данный документ содержит руководство по настройке и применению маршрутизаторов NSG, оснащенных программным обеспечением NSG Linux 2.0. Документ имеет следующую структуру:

Часть 1. Общесистемная конфигурация.
Часть 2. Настройка физических интерфейсов, портов и сетевых интерфейсов. Обработка трафика Ethernet.
Часть 3. Обработка IP-трафика.
Часть 4. Приложения и службы IP.
Часть 5. Туннелирование и виртуальные частные сети (VPN).
Часть 6. Система обеспечения бесперебойных соединений ***ui*TCP**.
Часть 7. Основные команды и утилиты NSG Linux.

Руководства по применению маршрутизаторов под управлением NSG Linux 1.0 и базового программного обеспечения NSG, а также других продуктов NSG (модемов, мостов и т.п.), содержатся в отдельных документах.

ВНИМАНИЕ

Данное Руководство пользователя предназначено для лучшего понимания процедуры настройки в целом и описывает сущность выполняемых действий — а именно, *что* необходимо настраивать. Рассматриваемые вопросы относятся, как правило, к сущности используемой технологии и являются общими для любых её реализаций, независимо от конкретного производителя и устройства. (Исключением являются вопросы, специфичные для оборудования NSG — таких, как организация пользовательского интерфейса или настройка системы бесперебойных соединений ***ui*TCP**.)

Основной документацией по NSG Linux 2.0 является встроенная справка на борту устройства. Она описывает конкретные команды и параметры настройки — т.е. *как* настраивать функции и возможности, описанные в данном Руководстве. Для просмотра справки по каждому из параметров следует использовать кнопку в Web-интерфейсе или команду _manual (_m) в консольном интерфейсе. Если справка на вашем языке отсутствует, следует установить на устройстве русскую локаль.

ВНИМАНИЕ Продукция компании непрерывно совершенствуется, в связи с чем возможны изменения отдельных аппаратных и программных характеристик по сравнению с настоящим описанием. Сведения о последних изменениях приведены в файлах README.TXT, CHANGES, а также в документации на отдельные устройства.

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 §

Часть 6. Система обеспечения бесперебойных соединений *ui*TCP

§6.1. Введение

§6.1.1. 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 неизбежных 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 за клиентским устройством NSG может располагаться как одно устройство, так и целый офис, батарея POS-терминалов, или локальный модемный пул.

С точки зрения протокольной архитектуры, *ui*TCP представляет собой фирменную VPN 4 уровня с защитой данных при помощи SSL. Ближайшими аналогами являются STunnel, Open VPN, KerioNET; отличие состоит в том, что *ui*TCP специально ориентирована, в первую очередь, на поддержание бесперебойных соединений и гарантированную доставку данных.

Программный комплекс *ui*TCP является составной частью NSG Linux 2.0 и может использоваться на любых устройствах NSG, работающим под его управлением, как в качестве клиента, так и в качестве сервера. В частности, для построения опытных зон с небольшим числом клиентов (в пределах 30) возможно использование устройств NSG–700 в качестве сервера, а для первичного ознакомления с системой — даже построение канала между парой устройств младшего уровня NSG–600.

Для устройств под управлением NSG Linux 1.0 и выпускается отдельная версия *ui*TCP, устанавливаемая в качестве опциональной компоненты. Документация по этой версии является дополнением к документации по NSG Linux 1.0. Обе версии совместимы друг с другом, используют один и тот же код и общую схему конфигурации, за исключением отдельных аспектов, специфичных для NSG Linux 1.0 либо 2.0 (в основном, синтаксиса скриптов для рестарта каналов связи).

§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 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.

§6.1.3. Расширенные функциональные возможности *ui*TCP

Дополнительные возможности, предоставляемые *ui*TCP, включают:

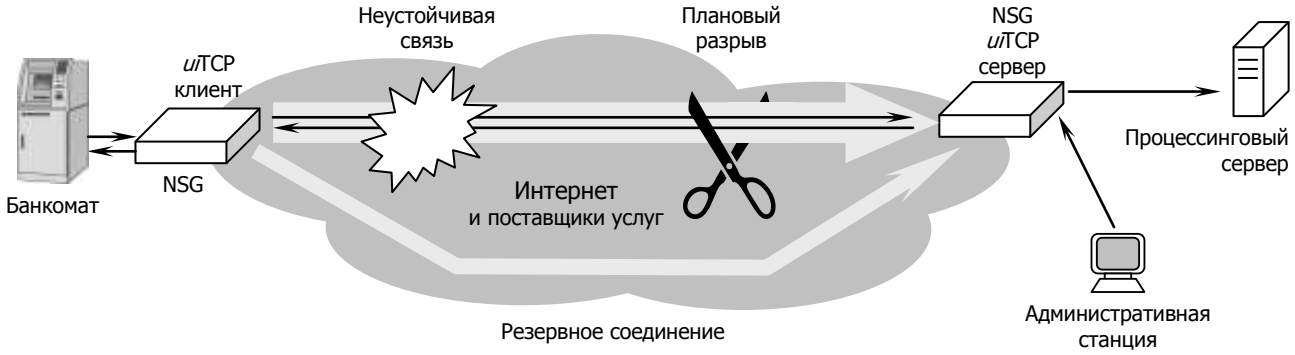
- Прохождение любых типов и реализаций NAT как на выходе из сети поставщика услуг Интернет, так и на входе в сеть процессингового центра. Протокол TCP передаётся через NAT всегда (в отличие от, например, GRE или IPsec). Если поставщик услуг фильтрует трафик по определённым протоколам и портам TCP, то 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:
  - Асимметричные ключи длиной 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 на других площадках, 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 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.

§6.1.4. 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 в качестве полнофункционального туннеля VPN Layer3-over-Layer4.

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

§6.1.5. 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.

§6.1.6. 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).

§6.1.7. Многотуннельные и многоканальные режимы

Одно клиентское устройство *ui*TCP может поддерживать одновременно несколько различных туннелей к одному 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 Агрегация имеет смысл только для каналов связи, более или менее равноценных с точки зрения пропускной способности и времени прохождения пакетов (например, для двух подключений к разным операторам 3G). При этом максимальная пропускная способность многоканального соединения в любом случае ниже, чем арифметическая сумма пропускной способности всех используемых каналов. Это принципиальное ограничение, связанное с внутренними алгоритмами работы *ui*TCP.

§6.1.8. Общие замечания о настройке и мониторинге *ui*TCP

*ui*TCP является составной частью NSG Linux 2.0 и настраивается в рамках общего конфигурационного дерева, доступного через Web-интерфейс либо консольную утилиту nsgsh . Порядок работы с этими инструментами изложен в Части 1 данного Руководства пользователя. Все настройки производятся в узле меню .tunnel.uitcp .

Для мониторинга работы *ui*TCP в графическом режиме в Web-интерфейсе имеется специальное окно, доступное по нажатию кнопки в команде .tunnel.uitcp.configure.show . При работе в консольном интерфейсе такая возможность отсутствует по определению.

При настройке *ui*TCP на клиентских устройствах следует обращать внимание на то, что этот механизм модифицирует некоторые другие элементы конфигурации, а именно:

— маршруты на сервер(ы) *ui*TCP
— приоритеты основной/резервной SIM-карт (для интерфейсов и сменных модулей GPRS/EDGE/3G)

*ui*TCP манипулирует этими объектами самостоятельно, чтобы направить трафик в выбранный канал связи. После первоначальной настройки основной части конфигурации и проверки работоспособности каждого из каналов явно указанные маршруты на серверы рекомендуется удалить, чтобы избежать возможных неоднозначностей. Приоритеты SIM-карт необходимо иметь в виду, чтобы обеспечить правильное начало работы системы (подробнее см. §6.2.3). Рекомендуется оставить их с настройками по умолчанию:

```
ppp.main.attempts = 1
ppp.aux.attempts = 0
```

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.

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).

Отдельная версия *ui*TCP для Linux 1.0 имеет встроенный механизм автоматического обновления. В NSG Linux 2.0 этот механизм не используется, поскольку *ui*TCP более не является самостоятельным продуктом. Если в одной системе используются клиенты под управлением NSG Linux 1.0 и 2.0, то первые при установлении туннеля будут обращаться к серверу по порту TCP 50003 для проверки обновлений. Это нормальное поведение и оно не должно вызывать беспокойства сетевого администратора.

ВНИМАНИЕ При использовании SSL с сертификатами X.509 необходимо обратить внимание на правильную установку системного времени. Если системные часы не установлены, то текущее время окажется раньше времени начала действия сертификата, и туннели не будут устанавливаться. Рекомендуется использовать клиента NTP для синхронизации всех устройств в системе от единого источника точного времени. (При этом сервер NTP должен быть доступен вне туннелей *ui*TCP.)
На устройствах серии NSG–600, не имеющих встроенных аппаратных часов, использование NTP является обязательным по существу.

§6.2. Настройка *ui*TCP

§6.2.1. Общая структура конфигурационного дерева

Дерево конфигурации *ui*TCP содержит следующие основные ветви:

```
tunnel
: uitcp
: : configure
: : : 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                                // разовые команды для просмотра текущего состояния
: : : SHUTDOWN                            // разовая команда для планового выключения сервера
: : enable = "true" | "false"             // Включение службы uiTCP
: : log                                   // разовая команда для просмотра журнала
```

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
: uitcp
: : configure
: : : 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
: uitcp
: : configure
: : : tunnels
: : : : имя
: : : : : общие параметры туннеля
: : : : : ssl
: : : : : links
: : : : : localListener
: : : : : socket
: : : : : nat
```

§6.2.2. Создание простейшего бесперебойного TCP-туннеля

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.6.3).

Соответствующие ветви в конфигурации сервера и клиента имеют вид:

Сервер:

```
: : : mode                  = "server",
: : : clients
: : : : имя
: : : : : ssl
: : : : : : disable         = true
: : : tunnelListener
: : : : 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 виде элементов нумерованного списка:

```
: : : tunnelListener
: : : : 1
: : : : : host              = "ip-адрес"
: : : : : port              = tcp-порт
: : : : 2
: : : : : .....
```

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 основной парой {host, port}. Например, в следующей конфигурации:

```
: : : tunnelListener
: : : : host = "10.0.0.1"
: : : : port = 50005
: : : : 1
: : : : : 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"
: : : : : .....
: : : : 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 — Список серверов. Значение каждой строки 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.
Идентификатором сервера может быть только целое число, причём они могут быть пронумерованы не подряд. Формально данный список является именованным и не упорядочивается автоматически.
Серверы используются по кругу в порядке возрастания номеров. Серверы 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).

connectAttempt
: Число попыток соединения по каждому из каналов.

connectTmo — Задержка между попытками соединения, в секундах.

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 "мягкое" выключение сервера *ui*TCP с помощью команды SHUTDOWN для (см. п.6.2.10).

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 5 сек.

ПРИМЕЧАНИЕ Для сетей GPRS/EDGE (2G) и других типов каналов с большим временем обращения пакетов рекомендуемое значение keepalive — 15 сек; для сетей 3G/4G — 10 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. Последнее может складываться, в максимальном случае, из времени обнаружения отказа канала (keepalive × attempts), времени рестарта сотового модуля (35–40 сек) и времени восстановления туннеля (5–15 сек).

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.

Каждый элемент в списке каналов связи имеет следующую структуру:

```
: : : : : номер
: : : : : : disable              = true | false
: : : : : : servers
: : : : : : : 1                  = "ip-адрес:tcp-порт"
: : : : : : : 2                  = "ip-адрес:tcp-порт"
: : : : : : : .....
: : : : : : dev                  = "интерфейс"
: : : : : : gw                   = "шлюз"
: : : : : : description          = "текст"
: : : : : : csqRequest
: : : : : : : interval           = "секунды"
: : : : : : : ttydev             = "порт"
: : : : : : restart
: : : : : : : номер_скрипта
: : : : : : : : description      = "текст"
: : : : : : : : script           = "скрипт"
: : : : : : : : servers
: : : : : : : : : 1              = "ip-адрес:tcp-порт"
: : : : : : : : : 2              = "ip-адрес:tcp-порт"
: : : : : : : : : .....
: : : : : : : : timeout          = секунды
: : : : : : : .....
: : : : : : : primary            = номер_скрипта
: : : : : : : timeout            = секунды
```

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.

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.

servers 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 этот список задан, то для данного канала связи он имеет 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 соединений "точка-точка" достаточно указать имя интерфейса, например, s1. Для широковещательных интерфейсов (Ethernet, VLAN и т.п.), наоборот, следует указать IP-адрес шлюза, а имя устройства можно не указывать (если для вывода статистики 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.
Если для данного канала связи не указано ни dev, ни gw, то при попытке работы по этому каналу *ui*TCP не будет создавать специфических маршрутов на сервер, а будет пытаться использовать маршруты, уже имеющиеся на данный момент в системе (например, маршрут по умолчанию, полученный по DHCP).

ПРИМЕЧАНИЕ Для соединений по туннелям PPTP и PPPoE (в роли клиента) в качестве dev следует указывать имя туннеля, например, pppoe1.

csqRequest
Контроль уровня сигнала. Имеет смысл только для сотовых модулей, поддерживающих параллельно передачу данных и работу с AT-командами (SMS-управление и др.). 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. Полученное значение выводится сервером в Web-мониторинге клиента. Чтобы использовать данную возможность, модуль должен иметь, как правило, несколько внутренних интерфейсов. Параметры для выполнения этой процедуры:

- interval — Период опроса уровня сигнала.
- ttydev — Имя сотового порта с префиксом "port. ", например, port.3g или port.s1 . Если данный параметр не установлен, контроль уровня сигнала не производится.
- cdma — Использовать формат запроса, специфичный для модулей CDMA. Применимо только для модулей UM–EVDO/A *ver.5* и для интерфейсов CDMA устройств NSG–6*xx*D, NSG–1820D, NSG–1820HD, настроенных в режиме *usb-serial* (подробнее см. Часть 2). По умолчанию, используется формат команды для модулей GSM/UMTS (в т.ч. для модулей UM–EVDO/A *ver.7* и опции opt18xx.CDMA).
  Начиная с версии 2.0 *build 5*, параметр cdma является формальным и сохранён исключительно для совместимости с имеющимися конфигурациями. На работу устройства не влияет, тип модуля и нужный для него формат команд определяются автоматически. Возможно, будет удалён в одной из последующих версий.

Для устаревших типов модулей с одним интерфейсом (UIM–EDGE *h/w v3* и все модули *IM–xxx*) на порту должен быть включён обработчик SMS (sms-handler) в режиме ppp-cooperation yes. В качестве ttydev в этом случае допускается указывать номер порта TCP, на котором обработчик SMS ожидает соединения от других программ (sms-handler.control-tcp-port); в данном случае этот параметр необходимо вводить как алфавитно-цифровое имя, т.е. в кавычках, например: "50000".
Подробнее о механизмах работы с сотовыми модулями в режиме совместной передачи данных и исполнения AT-команд см. Часть 2.

Узел restart содержит набор скриптов, которые могут исполняться в случае, когда клиент *ui*TCP детектировал отказ данного канала связи. С помощью этого механизма можно принудительно рестартовать данный канал связи, если его отказ не детектируется и, соответственно, не обрабатывается 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.) При работе с 2-симчатыми сотовыми интерфейсами GSM и UMTS механизм restart используется для переключения на другого оператора.

В общем случае данный узел содержит несколько процедур и три общих параметра:

timeout *секунды*
Таймауты позволяют ограничить минимальное время, которое должно пройти между двумя последовательными исполнениями скриптов. В противном случае возможна ситуация, когда *ui*TCP будет постоянно рестартовать порт раньше, чем он успеет стартовать полностью. Это необходимо, в первую очередь, для сотовых интерфейсов GPRS и CDMA, которым требуется 30–35 сек. для инициализации, регистрации в сети и установления PPP-соединения. Рекомендуется, чтобы значение таймаута было, как минимум, в 3 раза больше времени рестарта интерфейса.
Таймаут может быть установлен общим для всех процедур рестарта, и/или индивидуально для каждой процедуры. Если установлены оба таймаута, то локальный (установленный для данной процедуры) имеет приоритет.

tumeout-max

Продолжительность соединения, после которой канал считается исправно работающим и рестарт при уходе с этого соединения отменяется. Актуально для 2-симчатых сотовых модулей в следующей ситуации: модуль нормально работает с приоритетным оператором длительное время, после чего разъединяется (например, по инициативе оператора). В этом случае нет оснований считать данное соединение проблемным и переключаться на резервного оператора, достаточно просто пересоединиться, используя ту же самую SIM-карту. (Рестарт модуля всё равно происходит средствами pppd, но без смены SIM-карты.)

Если попытка повторного соединения окажется неудачной, то следующая попытка будет сделана уже обычным образом, т.е. с рестартом и сменой SIM-карты (если задано).

По смыслу, данный параметр не должен быть меньше или равен минимального таймаута, а также времени priority (если указано). Рекомендуется устанавливать для него значения, как минимум, на порядок больше: от нескольких часов до суток.

При отсутствии параметра рестарт при уходе с данного канала выполняется всегда (кроме очень коротких попыток, ограниченных параметром timeout). Значение 0 даёт тот же результат и позволяет отменить рестарт локально для данного узла конфигурации, если он установлен глобально в вышестоящем узле.

primary *номер*

Номер процедуры рестарта, которая приводит к работе по приоритетному каналу связи. Используется в сочетании с параметрами links.priority и links.idle-time. Параметр предназначен для 2-симчатых сотовых интерфейсов, если требуется работать с двумя существенно неравноценными операторами. Обычно первый оператор является приоритетным, а второй — резервным; процедура рестарта 1 приводит к переходу на вторую SIM-карту, а процедура 2 — на первую. В этом случае следует указать primary = 2 и некоторое ненулевое время в links.priority + links.idle-time. После выполнения рестарта 1, по прошествии этого времени, будет принудительно выполнен рестарт 2, т.е. модем попытается возвратиться на основного оператора. Если же был выполнен рестарт 2, т.е. модем уже работает через основного оператора, то принудительный рестарт не будет выполняться и это соединение будет работать сколь угодно долго, пока не разорвётся по естественным причинам.

Описание каждой процедуры рестарта содержит следующие параметры:

description Описание состояния, в которое переходит система после выполнения данной процедуры. Значение данного поля добавляется к общему description, установленному в вышестоящем узле меню. Параметр актуален, в первую очередь, для 2-симчатых сотовых модулей, чтобы сообщить серверу, через какого оператора в данный момент этот модуль работает.

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.

servers Специфический набор серверов, устанавливаемый после выполнения данной процедуры. Описывается по тем же правилам, что и для локальный набор для данного канала связи и глобальный набор серверов и, если задан, то имеет приоритет перед ними обоими.

timeout Специфический таймаут (время моратория на последующие рестарты), устанавливаемый после выполнения данной процедуры. Аналогичен общему параметру timeout для данного канала.

ВНИМАНИЕ Параметр script определяет процедуру рестарта канала, а параметры description, servers, timeout относятся к состоянию системы после её выполнения, т.е. в ходе работы на отрезке времени от этого рестарта до следующего.

В общем случае, при начале работы по данному каналу устанавливается глобальный таймаут, гарантирующий некоторое минимально необходимое время для установления соединения. 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, величина таймаута до следующего рестарта и описание (если таковые определены). *ui*TCP тем временем продолжает работу по какому-либо другому каналу. (Если данный канал единственный, например, 2-симчатый сотовый модуль, то *ui*TCP просто ждёт, пока он восстановит работоспособность через другого оператора.) По прошествии таймаута, 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 §6.2.3.

§6.2.3. Особенности настройки соединений GPRS/3G с двумя SIM-картами

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

— Вставить в модуль или устройство две SIM-карты. Для сменных модулей — снять перемычку, запрещающую работу с резервной картой.

— Настроить на порту два набора параметров ppp для соединения с основным (main) и резервным (aux) операторами.

Переключение между операторами регулируется параметрами ppp.main.attempts и ppp.aux.attempts в меню порта. В частности, предельный вариант

```
ppp.main.attempts = 1
ppp.aux.attempts = 0
```

предписывает работу исключительно через основного оператора, противоположный вариант

```
ppp.main.attempts = 0
ppp.aux.attempts = 1
```

— только через резервного. Именно такие упрощённые конфигурации использует *ui*TCP, манипулируя ими при помощи команды restart в меню порта. Вызывая её с помощью скриптов, клиент *ui*TCP в каждой попытке синхронизирует выбор оператора и используемый набор серверов.

Модуль с двумя SIM-картами рассматривается в *ui*TCP как *один* канал связи. 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–700 установлен единственный интерфейс для подключения к вышестоящей сети — модуль UIM–3G 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.

Считается, что в настройках порта s1 установлено число попыток по умолчанию: 1 для основной SIM-карты и 0 для резервной, как указано выше. Конфигурация канала связи:

```
: : : mode                    = "client",
: : : tunnel
.....
: : : : links
: : : : : 1
: : : : : : servers
: : : : : : : 1               = "10.11.12.13:50005"
: : : : : : dev               = "s1"
: : : : : : description       = "3G_"
: : : : : : restart
: : : : : : : 1
: : : : : : : : script        = "nsgsh -q .port.s1.restart.aux"
: : : : : : : : description   = "BEELINE"
: : : : : : : : servers
: : : : : : : : : 1           = "123.45.67.89:50005"
: : : : : : : : timeout       = 150
: : : : : : : 2
: : : : : : : : script        = "nsgsh -q .port.s1.restart.main"
: : : : : : : : description   = "MEGAFON "
: : : : : : : timeout         = 120
```

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 возвращение на приоритетного оператора в данном примере не производится.

ВНИМАНИЕ Существенным предположением в данном случае является то, что система начинает работу по основному каналу связи. По этой причине первая процедура рестарта должна выполнять переход с основного оператора на резервного, а вторая — с резервного на основного, и никак не наоборот.

ПРИМЕЧАНИЕ Команды restart.main и restart.aux в меню порта 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.

§6.2.4. Настройка TCP-соединений и NAT

Простейший туннель, созданный в п.6.2.2, 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 направлениях. Различие состоит только в том, что на клиенте эти узлы меню находятся внутри описания туннеля, а на сервере — в меню каждого клиента:

Клиент	**Сервер**
: : : tunnel	: : : clients
: : : : localListener	: : : : *имя_клиента*
: : : : : *localListener1*	: : : : : localListener
: : : : : *localListener2*	: : : : : : *localListener1*
.....	: : : : : : *localListener2*
: : : : nat	.....
: : : : : *правило1*	: : : : : nat
: : : : : *правило2*	: : : : : : *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.

ПРИМЕЧАНИЕ Название NAT для данного узла сложилось исторически и не вполне соответствует сущности выполняемой процедуры. При работе туннеля в режиме прокси заголовок IP-уровня не передаётся, поэтому на принимающей стороне выполняется, по существу, не подмена тех или иных полей IP, а формирование этого заголовка заново. Подменяться может только номер порта назначения в заголовке TCP.

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

На вызывающей стороне узлы localListener имеют следующую структуру:

```
: : : : localListener
: : : : : номер
: : : : : : 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, принадлежащим 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. По умолчанию, *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. По умолчанию, разрыв соединения не производится.

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:

```
: : : : nat
: : : : : номер
: : : : : : 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 значение 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.

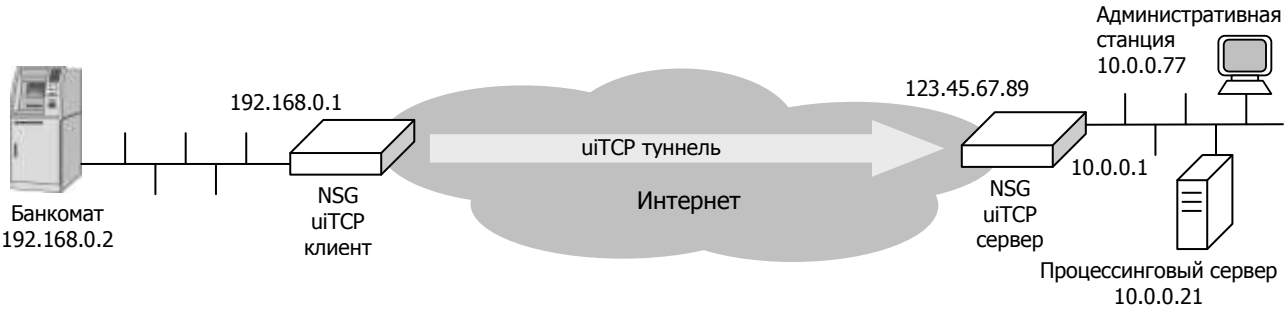
outDstPort TCP порт, подставляемый в пакет в качестве порта назначения. Если данный параметр отсутствует, то сохраняется TCP порт назначения, указанный в оригинальном пакете.

outSrcAddr IP-адрес, подставляемый в пакет в качестве адреса источника. Этот же адрес присваивается в качестве вторичного (*alias*) локальному интерфейсу lo устройства NSG. (При необходимости можно выбрать для этой цели другой интерфейс с помощью параметра sys.ifaceForSrcAddr .)
Если данный параметр отсутствует, то в качестве источника указывается IP-адрес того интерфейса, через который пакет уходит в локальную сеть.

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*"

ВНИМАНИЕ Динамическое добавление и удаление адресов на интерфейсах, выполняемое *ui*TCP, может конфликтовать с работой других подсистем. Например, если включён демон динамической маршрутизации BIRD, то он выбирает для своих целей один из адресов интерфейса в качестве первичного (primary); подробнее см. Часть 3. При этом, если удаляется адрес, который был выбран BIRD в качестве первичного, то BIRD рестартует интерфейс и выбирает из оставшихся адресов новый первичный. То же самое происходит, если добавляется адрес, который должен был бы быть выбран BIRD в качестве первичного по умолчанию. Рестарты интерфейса нарушают нормальную работу *ui*TCP и нежелательны для системы в целом. Чтобы избежать их, следует назначить интерфейсу формальный адрес, который не будет динамически назначаться и удаляться, и указать именно его в качестве первичного для BIRD. (Или выбрать первичный адрес из числа реальных адресов интерфейса, сконфигурированных статически.)

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. По умолчанию, пакеты *keepalive* не посылаются.
Кроме того, этот же параметр ограничивает время ожидания ответа от конечного адресата. Если за это время установить TCP–соединение с ним не удалось, разрывается также входящее соединение со стороны исходного хоста — инициатора транзакции.

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. По умолчанию, 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
: : : : : 1
: : : : : : host        = "192.168.0.1"
: : : : : : port        = 10001
: : : : nat
: : : : : 1
: : : : : : inSrcAddr   = "10.0.0.77"
: : : : : : inDstPort   = 20001
: : : : : : outDstAddr  = "192.168.0.1"
: : : : : : outDstPort  = 80
: : : : : 2
: : : : : : inSrcAddr   = "10.0.0.77"
: : : : : : inDstPort   = 30001,
: : : : : : outDstAdd   = "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
: : : : : : 1
: : : : : : : outDstAddr    = "10.0.0.21"
: : : : : : : outDstPort= 25001
: : : : : localListener
: : : : : : 1
: : : : : : : host      = "10.0.0.1"
: : : : : : : port      = 20001
: : : : : : 2
: : : : : : : 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).

§6.2.5. 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	**Сервер**	
	: : : clients	
: : : tunnel	: : : : *имя_клиента*	
: : : : socket	: : : : : socket	
: : : : : *имя_сокета*	: : : : : : *имя_сокета*	
: : : : : : disable = true | false	: : : : : : : disable = true | false	
: : : : : : type = "udp" | "raw" | "net"	: : : : : : : type = "udp" | "raw" | "net"	
: : : : : : *параметры udp*	: : : : : : : *параметры udp*	В зависимости от типа сокета
: : : : : : *параметры raw-ip*	: : : : : : : *параметры raw-ip*	
: : : : : : *параметры вирт.интерфейса*	: : : : : : : *параметры вирт.интерфейса*	

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 имя в виде числа от 1 до 64. Иные типы имён не допускаются. Всего в одном туннеле может быть открыто до 64 сокетов. Имя 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 вид:

```
: : : : : : udp
: : : : : : : peerAddress      = "ip-адрес"
: : : : : : : peerPort         = udp-порт
: : : : : : : socketAddress    = "ip-адрес "
: : : : : : : socketPort       = udp-порт
```

где параметры имеют следующий смысл:

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             = "имя"
: : : : : : : ifAddress
: : : : : : : : anycast          = "ip-адрес"
: : : : : : : : broadcast        = "ip-адрес"
: : : : : : : : peer             = "ip-адрес"
: : : : : : : : prefix           = "ip-префикс"
: : : : : : : : alias1
: : : : : : : : .....
```

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.имя_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.

Чтобы назначить IP-адрес этому интерфейсу, можно либо использовать параметры prefix, anycast, broadcast, peer в данном узле, либо создать в системе псевдо-интерфейс с таким же именем:

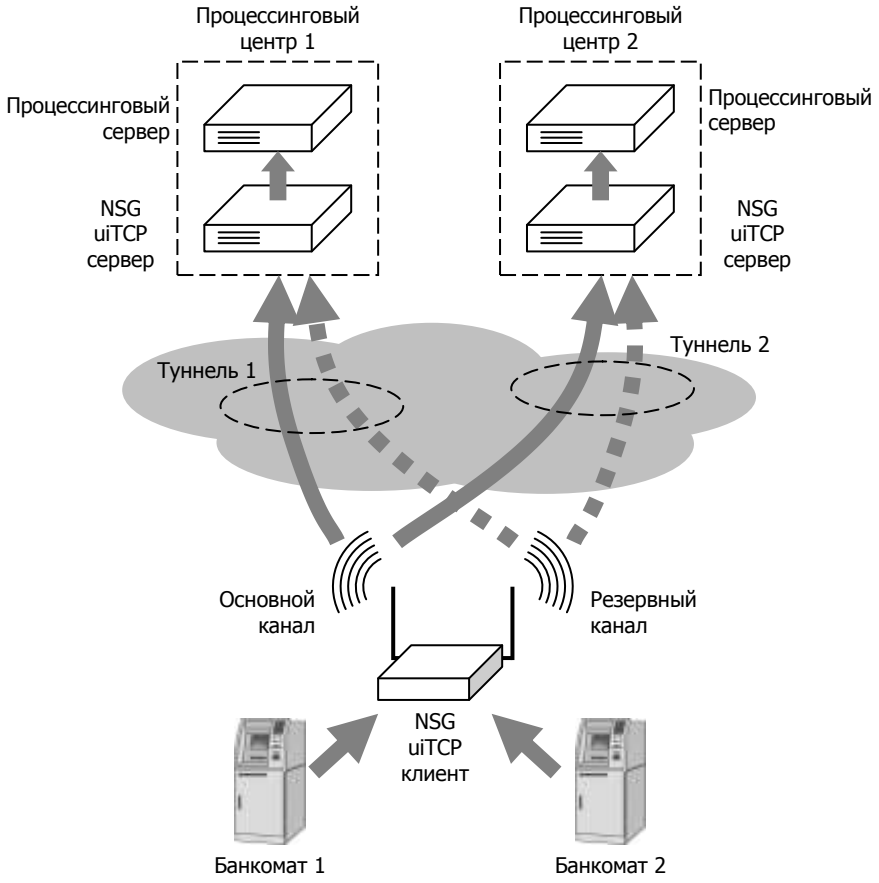
```
pseudo-interface
: ifName
: : ifAddress
: : : prefix
.....
```

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 дополнительных узлов как в том, так и в другом случае. Все остальные параметры, присущие виртуальному IP-интерфейсу *ui*TCP, могут быть установлены в узле .pseudo-interfaces.*имя* (см. Часть 3).

§6.2.6. 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.6.2.7).
— 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:

```
uitcp
: configure
: : 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".

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
: : : : : 1
: : : : : : gw           = "123.45.67.89"
: : : : : 2
: : : : : : dev          = "s1"
.................
: : : TUNNEL2
: : : : priority          = true
: : : : links
: : : : : 1
: : : : : : gw           = "123.45.67.89"
: : : : : 2
: : : : : : dev          = "s1"
.................
: : : TUNNEL3
: : : : links
: : : : : 1
: : : : : : 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
: : : TUNNEL
: : : : priority          = true
: : : : links
: : : : : 1
: : : : : : dev          = "s1"
: : : : : 2
: : : : : : dev          = "s2"
.................
: : : TUNNEL2
: : : : priority = true
: : : : links
: : : : : 1
: : : : : : dev          = "s2"
: : : : : 2
: : : : : : dev          = "s1"
.................
```

§6.2.7. 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 узле .tunnel.uitcp.configure как на клиенте, так и на сервере, и имеет структуру, в основном схожую с настройками обыкновенных сокетов:

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

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 (window); 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.

§6.2.8. Системный журнал и трассировщик туннелей

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

```
tunnel
: uitcp
: : configure
: : : log
: : : : level                = "debug" | "info" | "warn" | "error" | "fatal"
: : : : logFile              = файл
: : : : logFileMaxSize       = байты
: : : : trace                = байты
: : : : параметры SQL
```

Значения параметров для ведения локального журнала:

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

logFile — Путь и имя файла для журнала *ui*TCP.

logFileMaxSize — Максимальный размер файла журнала. Допустимые значения от 1024 до 1073741824 байт (1 Кб … 1 Гб). 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 можно командой .tunnel.uitcp.log . В командной оболочке Linux удобно это делать 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.

§6.2.9. 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. В данной версии NSG Linux он реализован только на x86-совместимых серверах (серия NSG–1000). 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 включен клиент 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 *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:

```
tunnel
: uitcp
: : configure
: : : log
: : : : level = "debug"
: : : : : sql
: : : : : : database = "PostgreSQL"
: : : : : : hostname = "10.0.0.10"
: : : : : : username = "uitcp"
: : : : : : password = "uitcp"
: : : : : : sourcename = "uitcp"
: : : : : : tablename = "log"
: : : : : : : fields
: : : : : : : 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)
```

§6.2.10. Общесистемные настройки

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

.tunnel.uitcp.configure.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.).

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 .tunnel.uitcp.configure.sys предназначен для хранения общесистемных служебных настроек. В данной версии *ui*TCP этот узел содержит параметры:

pluginFolder Директория, в которую будут помещаться дистрибутивные файлы uitcp-X.Y для раздачи клиентам. Использование данного параметра имеет смысл только на сервере. Собственно в NSG Linux 2.0 данный параметр не используется, сохранён для совместимости с отдельной версией *ui*TCP для NSG Linux 1.0.

ifaceForSrcAddr
Интерфейс, которому будут присваиваться все IP-адреса источника, указываемые в параметрах nat.N.outSrcAddr . Присваивать их какому-либо локальному интерфейсу устройства необходимо для того, чтобы оно могло получать ответные пакеты. По умолчанию, для этой цели используется псевдо-интерфейс lo, однако в некоторых случаях он не подходит. В этом случае следует выбрать какой-либо другой физический (но обязательно — постоянно поднятый, такой как eth0 в NSG–700, eth1 в NSG–1800) или псевдо-интерфейс. Рекомендуется использовать псевдо-интерфейс dummy0 .

monitoring-servers
Настройка централизованного мониторинга удалённых серверов *ui*TCP. Подробно о данной функции см. §6.2.13.

Узел show содержит команды для просмотра статуса и статистики туннелей. Данный узел не входит в состав конфигурации и не сохраняется в конфигурационном файле.

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. Собственно в NSG Linux 2.0 данная команда не имеет смысла, сохранёна для совместимости с отдельной версией *ui*TCP для NSG Linux 1.0.

show clearstat
Сбросить статистику туннеля с указанным именем. Параметр команды не должен быть пустым. Кроме имени конкретного туннеля, допускается специальное значение true (без кавычек) — в этом случае будет сброшена статистика всех туннелей.

§6.2.11. Просмотр списка клиентов по группам

В больших системах, если список клиентов уже не помещается в экран Web-мониторинга у дежурного оператора, возникает потребность некоторым образом разбить его на группы и выводить эти группы иерархическим образом: в левой части экрана сначала выводится список групп в свёрнутом виде, после чего любую группу можно развернуть щелчком по ней самой или по знаку "+" перед ней.

Явно указать принадлежность конкретного клиента к конкретной группе можно с помощью параметра group в описании клиента на сервере. Отдельно создавать группу не требуется — все группы, перечисленные в этих параметрах для разных клиентов, создаются автоматически. Клиенты, для которых группа не указана, по умолчанию включаются в default group .

Дальнейший выбор представления возможен уже на самой странице Web-мониторинга с помощью выпадающего меню group . Помимо вышеупомянутого явного способа, оно предлагает несколько шаблонов для разбиения существующего списка клиентов на группы без изменения их конфигурации. Основные пункты данного меню (в будущем возможно добавление других шаблонов):

none — Выводить плоский список клиентов без разделения на группы.

group — Выводить список согласно явно указанным группам в описаниях клиентов; клиенты, для которых группа не указана, включаются в default group .

state — Разбить клиентов на три группы normal, suspended и dead сообразно их текущему состоянию.

name1(*[punct]*)
: Разбить клиентов на группы по следующему принципу: все клиенты, имеющие одинаковое начало имени, вплоть до первого знака препинания (точки, запятой, дефиса и т.п.), включаются в одну группу. Имена, состоящие исключительно из букв и цифр, в данном случае читаются до конца (конец строки тоже есть знак препинания), таким образом, они целиком составляют имя группы, и для каждого такого клиента создаётся уникальная группа.

name2([alpha]*)
: Разбить клиентов на группы по начальной части имени — до первого символа, не являющегося буквой.

name3([digit]*)
: Разбить клиентов на группы по начальной части имени — до первого символа, не являющегося цифрой.

alphabet
: Разбить клиентов на группы по первой букве имени.

cert expiration
: Разбить клиентов на 3 группы по состоянию сертификата: действителен, менее 2 недель до окончания срока действия и недействителен.

Данные шаблоны предполагают, что имена существующих клиентов начинаются с некоторого общего префикса, который может означать, например, название населённого пункта или код подразделения: MSK001, MSK002, … *vs* SPB001, SPB002, …, или 1234.001, 1234.002, … *vs* 5678.001, 5678.002, …, и т.п.

§6.2.12. Создание отдельного пользователя для Web-мониторинга

Если мониторинг системы предполагается поручить отдельному пользователю, которому не следует иметь доступ к настройкам, то для него возможно создать дополнительный вход следующим образом:

```
system
: users
: : texnik
: : : scheme
: : : : @link = ".tunnel.uitcp.show._html"
```

и установить отдельный пароль. При входе под этим именем пользователь будет попадать сразу на страницу мониторинга. Подробнее о создании пользователей с индивидуальными правами в системе см. Часть 1.

§6.2.13. Централизованный мониторинг нескольких серверов

Мониторинг распределённой системы *ui*TCP, состоящей из нескольких серверов, имеет свою сложность. Поскольку заранее неизвестно, на каком сервере окажется какой клиент в некоторый момент времени, то мониторинг на одном сервере не может дать полной картины: клиент, отсутствующий в данный момент на данном сервере, может работать через какой-то другой сервер, а может быть недоступен вообще. Данный механизм позволяет контролировать состояние всей системы *ui*TCP с одного сервера, в одном окне Web-броузера.

Список удалённых серверов, за которыми необходимо вести наблюдение с данного устройства, создаётся в узле .tunnel.uitcp.configure.sys.monitoring-servers . Каждый элемент списка определяется именем, которое будет выводиться на странице мониторинга в списке клиентов в виде

клиент@имяСервера

(например, MASTER или SLAVE). Таким образом, оператор может видеть, какой из клиентов с каким сервером работает в данный момент.

Значением элемента списка может быть IP-адрес удалённого сервера, либо его доменное имя, либо ключевое слово local для наблюдения за самим данным устройством NSG. Если список серверов пустой, то мониторинг производится только локально на данном сервере, при этом в списке выводятся только имена клиентов.

ВНИМАНИЕ Для мониторинга удалённых серверов используется доступ по SSH. Предварительно необходимо сгенерировать на данном устройстве ключ SSH и передать его на все удалённые серверы. Подробно о настройке клиента SSH для выполнения операций на удалённых устройствах см. Часть 4.

§6.2.14. Мониторинг событий и датчиков

Общий механизм обработки событий (*event-handler*) позволяет предпринимать заданные действия при возникновении заданных событий. Событием является переход некоторого датчика из одного состояния в другое, а датчиками могут служить различные физические и виртуальные объекты. В числе наблюдаемых параметров может быть состояние интерфейсов, состояние службы *netping*, показания внешних датчиков физических параметров, уровень сигнала беспроводной сети, и т.п. Подробнее об обработчике событий см. Часть 4, о настройке беспроводных портов как виртуальных датчиков — Часть 2.

В числе возможных действий, предпринимаемых в качестве реакции на заданное событие, имеется специфическое для устройств NSG — отправка TCP-уведомления в текстовом формате на сервер централизованного мониторинга. Данные уведомления могут также анализироваться системой *ui*TCP и выводиться в её мониторинге (на странице конкретного клиента). Для этого необходимо выполнение следующих условий:

— Данное TCP-соединение должно передаваться через туннель *ui*TCP в режиме TCP-прокси.

— На этом же сервере должен *ui*TCP быть запущен сервер мониторинга (.services.event-server), необходимый для терминирования TCP-соединений. Он желателен также и по существу, поскольку позволяет хранить предыдущие показания датчиков (в мониторинге *ui*TCP выводится, по очевидным причинам, только одно крайнее показание) и представлять их в удобной форме.

— Имя демона уведомлений (description, а при его отсутствии — само имя узла в event-actions.tcpsender) должно совпадать с именем туннеля.

§6.3. Настройка безопасности

§6.3.1. 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 защиту данных с помощью асимметричного ключа длиной 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.

§6.3.2. Создание и индексирование сертификатов

Генерацию ключей и сертификатов (если они не предоставляются сторонним удостоверяющим центром) рекомендуется выполнять на сервере *ui*TCP с помощью набора утилит cert-*, предназначенных для этой цели (см. п.6.3.4). 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 до́лжными именами.

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.

ПРИМЕЧАНИЕ При импорте сохранённой конфигурации, содержащей сертификаты X.509, необходимо после импорта выполнить операцию rehash .

§6.3.3. 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.

Глобальные настройки SSL для *ui*TCP находятся в узле .tunnel.uitcp.configure . Локальные настройки (для отдельно взятого туннеля) — внутри описания каждого клиента (на сервере) либо в узлах tunnel, tunnels.*имя* (на клиенте). 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 защиту данных; 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_2" | "tlsv1_1" | "tlsv1" | "sslv3" | "sslv23"

По умолчанию используется SSL *ver.3* исключительно как вынужденная мера для совместимости с некоторыми унаследованными конфигурациями, пользователи которых ещё не переключились на TLS. Протокол SSL в настоящее время имеет доказанные уязвимости и не может считаться безопасным. Рекомендации IETF RFC–7568 и PCI DSS 3.1 предписывают использование исключительно TLS.

Версия должна быть установлена одинаковой на обеих сторонах.

ciphers = *список*

Список поддерживаемых алгоритмов защиты данных. Формат списка описан в документе:
http://www.openssl.org/docs/apps/ciphers.html .

По умолчанию, разрешены все алгоритмы, 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:
"all", "no_sslv2", "no_sslv3"

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;
— *global_path*/*имя_клиента*/root.pem;
— *global_path*/root.pem.

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;
— *global_path*/*имя_клиента*/capath;
— *global_path*/capath.

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
— глобальное значение
— *global_path*/*имя_клиента*/client.pem
— *global_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
— *global_path*/*имя_клиента*/server.pem
— *global_path*/server.pem

password Пароль для доступа к закрытому ключу в случае, если ключ защищён паролем.

key Полный путь и имя файла, содержащего закрытый ключ данного устройства. Этот ключ используется для защиты данных 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
— глобальное значение
— *global_path*/*имя_клиента*/clientkey.pem
— *global_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
— *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:
"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.

§6.3.4. Централизованное управление ключами и сертификатами в системе

В данной версии NSG Linux 2.0 управление сертификатами осуществляется с помощью скриптов и утилит, вызываемых из командной оболочки Linux. (Механизм управления сертификатами через Web-интерфейс планируется в последующих версиях.) Для генерации самоподписанных ключей и сертификатов в системе *ui*TCP имеются три скрипта cert-root, cert-server и cert-client и вспомогательный скрипт rehash . Перед их первым использованием необходимо отредактировать файлы конфигурации сертификатов, которые находятся в директории /etc/uitcp/certs.conf (указать имя организации и т.п.).

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

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

Скрипт 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:

```
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 и выполнить скрипт rehash .

ПРИМЕЧАНИЕ В случае переноса полной конфигурации сервера (директория /etc целиком, включая сертификаты) на новую систему, например, после переустановки ПО через сервисный режим или замены физической машины, после копирования сертификатов также необходимо выполнить rehash .

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

Как можно видеть, все три скрипта 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.

Все три скрипта могут вызываться со следующими опциональными параметрами:

-d *дни* Срок действия сертификата (по умолчанию — 365).

-p *путь* Путь для сохранения сертификата.

-c *файл* Путь/имя файла конфигурации для соответствующего типа сертификатов.

§6.3.5. 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.

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. Перейти в рабочую директорию /etc/uitcp/certs (все пути ниже отсчитываются от этой директории) и удалить старый корневой сертификат:

   ```
   # cd /etc/uitcp/certs
   # rm _root/root.pem
   ```

2. Создать новый корневой ключ и сертификат:

   ```
   # cert-root
   ```

3. Удалить старый сертификат сервера:

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

   Это копия сертификата, необходимая для создания клиентских сертификатов. На сервере в это время продолжает работать старый сертификат server.pem, находящийся непосредственно в данной директории.

4. Создать новый сертификат и ключ сервера:

   ```
   # cert-server
   ```

   Новый сертификат и ключ сервера создается во вспомогательной папке _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 *имя*/cert-client.name.tgz

6. Создать новый сертификат и ключ клиента:

   # cert-client *имя*

   При этом в CA-файл *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 выполнить следующую команду:

   .tunnel.uitcp.configure.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 ./_server/* ./
   ```

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 выполнить команду secrets в следующем формате:

.tunnel.uitcp.configure.clients.*имя*.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 /etc/uitcp/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 /etc/uitcp/certs/*имя*/cert-client и имеют имена стандартные имена client.pem, clientkey.pem, root.pem, то в качестве параметра команды 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.

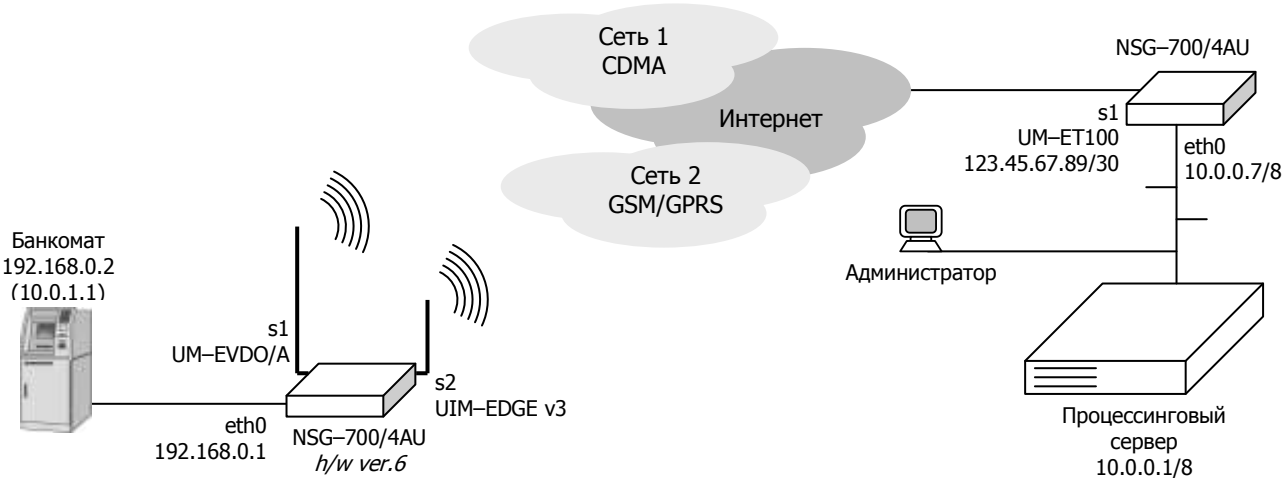
Приложение 6–А. Быстрый старт

6–А.1. Типовые конфигурации

Будем предполагать, что поставлена задача подключить один банкомат к процессинговому серверу, используя двух операторов. Оба оператора предоставляют только базовую услугу — доступ в Интернет с динамическими приватными IP-адресами, наиболее сложную для целей построения корпоративных сетей.

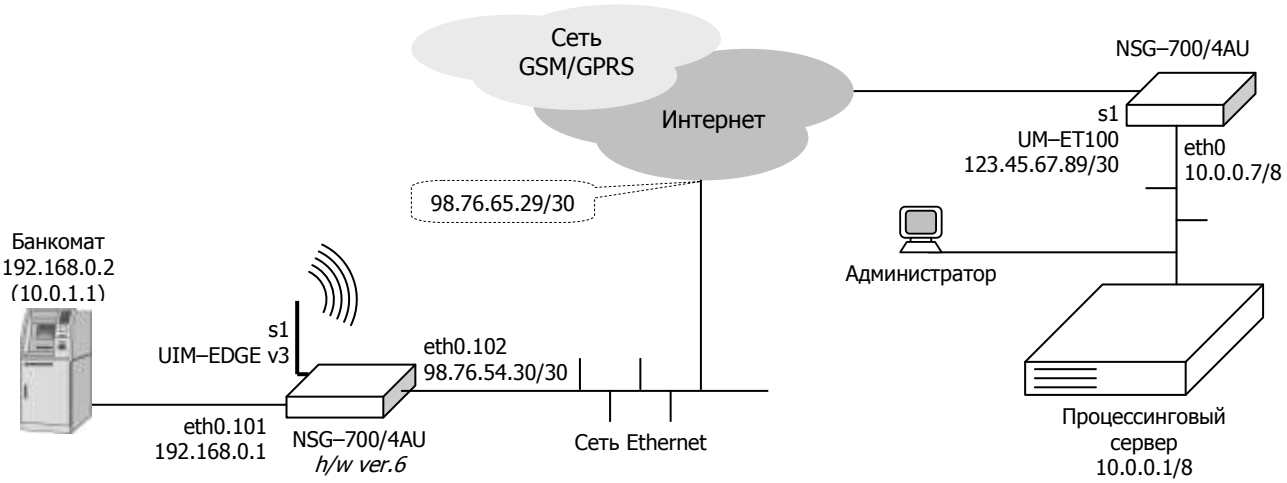
В качестве клиентского устройства используется NSG–700/4AU. В качестве сервера для реального построения крупномасштабных систем (десятки, сотни клиентов) следует использовать специализированные коммуникационные серверы NSG–1000/GW. Однако для целей тестирования и оценки возможностей данной технологии, а также для небольших систем (до 30 клиентов) в качестве сервера может использоваться второе устройство NSG–700/4AU.

Конфигурация а). Основной является сеть CDMA (Скайлинк), резервной — GRPS (для определённости, Мегафон). Соответственно, в качестве клиента используется шасси NSG–700/4AU *h/w ver.6* с модулями UM–EVDO/A *h/w ver.5* (разъём s1) и UIM–EDGE *h/w ver.3* (разъём s2).



Конфигурация б). Основным оператором является МТС, резервным — Билайн. Используется шасси NSG–700/4AU (аппаратная версия несущественна) с одним 2-симчатым модулем UIM–EDGE *h/w ver.3* (разъём s1).

Конфигурация в). Основное соединение по наземному каналу Ethernet, резервное — Мегафон. Используется шасси NSG–700/4AU (версия несущественна) с модулем UIM–EDGE *h/w ver.3* (разъём s1).



В качестве сервера используется устройство NSG–700/4AU (аппаратная версия несущественна). Чтобы удовлетворить самым строгим требованиям безопасности, в шасси установлен дополнительный физический порт Ethernet — модуль UM–ET100. Встроенным портом eth0 устройство включено в локальную сеть процессингового центра.

Банкомат устанавливает TCP-соединение к процессинговому серверу в локальной сети банка. На процессинге он зарегистрирован под IP-адресом 10.0.1.1. Кроме того, предполагается (хотя это не обязательно), что каждый банкомат устанавливает соединение на уникальный TCP порт сервера, для данного банкомата — 20001.

Помимо этого, в сети банка имеется администратор(ы), которому требуется устанавливать соединения со своей стороны к банкомату и к клиентскому устройству NSG.

6–A.2. Настройка соединений в сетях общего пользования

а) Вариант CDMA + GPRS

Конфигурация сервера:

```
ip
: route
: : 1
: : : gateway                 = "123.45.67.90"
: : : network                 = "0.0.0.0/0"
port
: eth0
: : ifAddress
: : : prefix                  = "10.0.0.7/8"
: s1
: : type                      = "eth"
: : ifAddress
: : : prefix                  = "123.45.67.89/30"
system
: hostname                    = "uitcp_srv"
```

Основная конфигурация клиентского устройства. SIM-карта Мегафон установлена в основное (верхнее) гнездо на модуле UIM–EDGE, PIN-код с обеих карт снят, все перемычки (аппаратного рестарта и 1-/2-симчатого режима) на обоих модулях установлены. Курсивом показаны существенные настройки, установленные по умолчанию.

```
port
: eth0
: : ifAddress
: : : prefix                  = "192.168.0.1/24"
: s1
: : type                      = "cdma"
: : adm-state                 = "up"
: : encapsulation             = "ppp"
: : ppp
: : : main
: : : : chat
: : : : : timeout             = 40
: : : : ipcp
: : : : : accept-address      = true
: : : : : accept-peer-address = true
: : : : lcp-echo-failure      = 3
: : : : lcp-echo-interval     = 10
: : : : sent-username         = "mobile"
: : : : sent-password         = "internet"
: s2
: : type                      = "edge"
: : adm-state                 = "up"
: : encapsulation             = "ppp"
: : ppp
: : : main
: : : : chat
: : : : : APN                 = "internet"
: : : : ipcp
: : : : : accept-address      = true
: : : : : accept-peer-address = true
: : : : lcp-echo-failure      = 3
: : : : lcp-echo-interval     = 10
: : : : sent-username         = "gdata"
: : : : sent-password         = "gdata"
system
: hostname                    = "ATM001"
```

Проверка доступности сервера через Скайлинк и Интернет:

```
ip
: route
: : 1
: : : device                    = "s1"
: : : network                   = "123.45.67.89/32"
tools
: ping
: : host                        = "123.45.67.89"
: : launch
```

Проверка доступности сервера через Мегафон и Интернет: заменить device = "s1" на "s2", повторить пинг.

Для работы *ui*TCP этот маршрут не нужен, поэтому после проверки его следует удалить.

Включение SSH для управления устройствами (на обоих устройствах). Помимо прямого назначения, SSH можно использовать как удобный инструмент для передачи файлов с одного устройства на другое; дополнительно для этой цели необходимо установить пароль для пользователя root.

```
services
: ssh
: : keygen                      = "yes"
: : enable                      = true
```

Настройка сервера NTP (на обоих устройствах). Корректная установка системного времени критически важна для работы *ui*TCP, поскольку сертификаты X.509 имеют ограниченный срок действия (как сверху, так и снизу). Рекомендуется синхронизировать все устройства в системе от одного сервера. В данном примере используется ntp.psn.ru [194.149.67.129].

На клиенте

```
ip
: route
: : 1
: : : device        = "s1"
: : : network       = "194.149.67.129/32"
: : 2
: : : device        = "s2"
: : : metric        = 2
: : : network       = "194.149.67.129/32"
system
: ntp
: : enable          = true
: : host            = "194.149.67.129"
```

На сервере

(*маршрут уже известен — это маршрут по умолчанию*)

```
system
: ntp
: : enable          = true
: : host            = "194.149.67.129"
```

Настройка сервера DHCP на клиентском устройстве. Он позволяет заранее прописать сетевые настройки банкомата в типовой конфигурации устройства NSG. Для удобства массовой инсталляции все удалённые площадки имеют одинаковую конфигурацию: устройство NSG — 192.168.0.1, банкомат — 192.168.0.2.

```
services
: dhcp
: : lan
: : : adm-state                 = "up"
: : : interface                 = "eth0"
: : : ip-address-pool
: : : : from                    = "192.168.0.2"
: : : : to                      = "192.168.0.2"
: : : mask                      = "255.255.255.0"
```

На банкомате, по умолчанию, включена динамическая конфигурация параметров TCP/IP; таким образом, их можно не настраивать вообще. Кроме того, как будет показано ниже, банкомат будет всегда обращаться к процессинговому серверу по адресу 192.168.0.1 (т.е. к своему устройству NSG, работающему как TCP proxy) и по порту TCP 20000. Подстановка уникального IP-адреса банкомата и/или номера порта TCP будет выполняться на сервере *ui*TCP.

Таким образом, в данной задаче уникальными параметрами на всей площадке оказываются, как будет видно из дальнейшего, только имя клиентского устройства NSG в системе *ui*TCP, его ключ и сертификат X.509. Все остальные параметры одинаковы для всех клиентов и могут быть загружены в них в виде готовой типовой конфигурации.

б) Вариант 2×GPRS

Конфигурация сервера идентична предыдущему случаю, поскольку она не зависит от способа подключения клиентов. Это же относится к настройке SSH, клиента NTP (кроме маршрутов на NTP сервер) и сервера DHCP на клиентском устройстве. Основная конфигурация клиентского устройства приведена ниже. SIM-карта МТС установлена в верхнее гнездо, Билайн — в нижнее. Перемычка аппаратного рестарта установлена, перемычка одно-/двухсимчатого режима снята.

```
port
: eth0
: : ifAddress
: : : prefix                    = "192.168.0.1/24"
: s1
: : type                        = "edge"
: : adm-state                   = "up"
: : encapsulation               = "ppp"
: : ppp
: : : main
: : : : attempts                = 1
: : : : chat
: : : : : APN                   = "internet.mts.ru"
: : : : ipcp
: : : : : accept-address        = true
: : : : : accept-peer-address   = true
: : : : lcp-echo-failure        = 3
: : : : lcp-echo-interval       = 10
: : : : sent-username           = "mts"
: : : : sent-password           = "mts"
: : : aux
: : : : attempts                = 0
: : : : chat
: : : : : APN                   = "internet.beeline.ru"
: : : : ipcp
: : : : : accept-address        = true
: : : : : accept-peer-address   = true
: : : : lcp-echo-failure        = 3
: : : : lcp-echo-interval       = 10
: : : : sent-username           = "beeline"
: : : : sent-password           = "beeline"
system
: hostname                      = "ATM001"
```

Проверка доступности сервера через сотовые сети и Интернет:

```
ip
: route
: : 1
: : : device                    = "s1"
: : : network                   = "123.45.67.89/32"
tools
: ping
: : host                        = "123.45.67.89"
: : launch
```

Чтобы проверить работу через того или другого оператора, следует выполнить команду port.s1.restart = "main" или "aux", соответственно. (Данная команда изменяет значения параметров main.attempts и aux.attempts, поэтому после её применения необходимо перед сохранением конфигурации проверять их и при необходимости восстанавливать.)

Для работы *ui*TCP этот маршрут не нужен, поэтому после проверки его следует удалить.

в) Вариант Ethernet + GPRS

Конфигурация сервера идентична предыдущим двум случаям, поскольку она не зависит от способа подключения клиентов. Это же относится к настройке SSH, клиента NTP и сервера DHCP на клиентском устройстве. Основная конфигурация клиентского устройства приведена ниже. SIM-карта Мегафон установлена в верхнее гнездо, все перемычки на модуле установлены.

```
ethernet
: ethernet-switch
: : phy0
: : : vlan-groups
: : : : 1                    = 101
: : : : 2                    = 102
: : : vlan-tagged            = true
: : phy1
: : : vlan-group             = 101
: : phy2
: : : vlan-group             = 102
: : vlan-mode                = true
port
: eth0
: : vlan
: : : eth0.101
: : : : ifAddress
: : : : : prefix             = "192.168.0.1/24"
: : : eth0.102
: : : : ifAddress
: : : : : prefix             = "98.76.54.30/30"
: s1
: : type                     = "edge"
: : adm-state                = "up"
: : encapsulation            = "ppp"
: : ppp
: : : main
: : : : chat
: : : : : APN                = "internet"
: : : : ipcp
: : : : : accept-address     = true
: : : : : accept-peer-address = true
: : : : lcp-echo-failure     = 3
: : : : lcp-echo-interval    = 10
: : : : sent-username        = "gdata"
: : : : sent-password        = "gdata"
system
: hostname                   = "ATM001"
```

Проверка доступности сервера через Ethernet и через сотовую сеть, соответственно:

```
ip
: route
: : 1
: : : gateway     = "98.76.54.29"
: : : network     = "123.45.67.89/32"
tools
: ping
: : host          = "123.45.67.89"
: : launch
```

```
ip
: route
: : 1
: : : device      = "s1"
: : : network     = "123.45.67.89/32"
tools
: ping
: : host          = "123.45.67.89"
: : launch
```

Для работы *ui*TCP эти маршруты не нужны, поэтому после проверки их следует удалить.

6–A.3. Генерация сертификатов X.509

Поскольку в практических применениях *ui*TCP необходима, как минимум, надёжная аутентификация клиентов, целесообразно включить в данную минимальную инсталляцию настройку SSL. Это, в свою очередь, требует установки взаимосвязанных сертификатов и ключей X.509 на серверные и на клиентские устройства. Сертификаты могут быть выданы сторонним удостоверяющим центром и помещены в соответствующие директории в виде файлов формата *.pem . Однако в данной тестовой инсталляции будет продемонстрировано создание самоподписанных сертификатов при помощи встроенных инструментов *ui*TCP.

На сервере необходимо создать директории, используемые *ui*TCP по умолчанию, и скопировать образцы конфигурации сертификатов:

```
root@uitcp_srv # mkdir  /etc/uitcp/certs
root@uitcp_srv # mkdir  /etc/uitcp/certs/capath
root@uitcp_srv # cp  -r  /etc/uitcp/bin/certs.conf  /etc/uitcp/
```

Образцы конфигураций, скопированные в директорию /etc/uitcp/certs.conf, следует отредактировать должным образом при помощи редактора nano. Как минимум, необходимо в файлах root.cnf и server.cnf в секции [req_distinguished_name] в поле commonName вписать реальный IP-адрес (123.45.67.89) или доменное имя сервера.

ВНИМАНИЕ Перед созданием сертификатов необходимо убедиться, что системное время на сервере установлено правильно.

После этого можно приступать непосредственно к созданию сертификатов:

```
root@uitcp_srv tools # cert-root
root@uitcp_srv tools # cert-server
root@uitcp_srv tools # cert-client ATM001
```

В результате на сервере создаются необходимые сертификаты и ключи: корневой, самого сервера, и клиента под именем ATM001. Для последующих клиентов необходимо повторять только последний скрипт с новым именем клиента. В заключение необходимо составить локальный список клиентских сертификатов с хэшами в качестве имён, для проверки их действительности:

```
root@uitcp_srv tools # rehash
```

Корневой сертификат и сертификаты сервера готовы к использованию. Набор сертификатов и ключей, необходимых клиенту, упакован в один архив /etc/uitcp/certs/*имя_клиента*/cert-client.*имя_клиента*.tgz . Его необходимо переместить на клиентское устройство любым безопасным способом (на USB Flash, через промежуточный FTP или TFTP сервер в локальной сети и т.п.) и распаковать в нужную директорию. В нижеприведённом примере файл копируется на клиента по SSH (предполагается, что клиент временно подключён к локальной сети банка):

```
root@ATM001 # mkdir  /etc/uitcp/certs
root@ATM001 # cd  /etc/uitcp/certs
root@ATM001 certs # ifconfig eth0 10.0.1.123/8
root@ATM001 certs # ssh  root@10.0.0.7  "cat  /etc/uitcp/certs/ATM001/cert-client.ATM001.tgz" |
cat > cert-client.ATM001.tgz
root@ATM001 certs # tar  -xzf  ./cert-client.ATM001.tgz
root@ATM001 certs # rm  cert-client.ATM001.tgz
```

6–A.4. Создание туннеля *ui*TCP

а) Вариант CDMA + GPRS

Теперь можно приступать непосредственно к настройке *ui*TCP. Первый этап — создание отказоустойчивого туннеля между клиентским и серверным устройствами.

Настройка SSL на обоих устройствах. Рекомендуемые опции для максимальной безопасности установлены по умолчанию; все сертификаты также уже находятся в файлах и директориях, используемых по умолчанию. Таким образом, специальная настройка SSL не требуется. Показана настройка по умолчанию:

```
tunnel
: uitcp
: : configure
: : : ssl
: : : : options
: : : : : all                    = true
: : : : : no_sslv2               = true
: : : : verify
: : : : : fail_if_no_peer_cert   = true
: : : : : peer                   = true
```

Настройка журналов для отладки (на обоих устройствах):

```
: : : log
: : : : level                   = "debug"
: : : : logFileMaxSize          = 10485760
```

Далее на сервере нужно установить режим сервера и создать запись для клиента, хотя никаких настроек по существу пока делать не будем:

```
: : : mode                      = "server"
: : : clients
: : : : ATM001
: : : : : description           = "test ATM"
```

На сервере uitcp ждёт входящих соединений, по умолчанию, на порту TCP 50005 (при необходимости можно настроить другой порт, и избирательно указать интерфейсы). Теперь можно запустить просмотр системного журнала, чтобы следить за попытками клиента подключиться. Удобно делать это в реальном времени в командной оболочке Linux:

```
root@uitcp_srv # tail -f /var/log/uitcp
2010-02-15 00:15:30 INFO :============= uiTCP-0.23 beginning =============
2010-02-15 00:15:31 INFO :Reopen log file. Next output may be in other file
2010-02-15 00:15:31 INFO :============== uiTCP-0.23 started ==============
2010-02-15 00:15:34 INFO :Server mode
2010-02-15 00:15:34 INFO :Waiting connection from talker on 0.0.0.0:50005
2010-02-15 00:15:34 INFO :Start main loop
```

(для завершения вывода нужно нажать CTRL-C).

Все операции по контролю работоспособности туннеля, переходу на резервный канал, возвращению на основной и т.п. в *ui*TCP возлагаются на клиента, поэтому основная часть настроек производится на нём:

```
tunnel
: uitcp
: : configure
: : : mode                      = "client"
: : : tunnel
: : : : name                    = "ATM001"
: : : : description             = "test ATM"
: : : : servers
: : : : : 1                     = "123.45.67.89:50005"
: : : : keepalive               = 10
: : : : links
: : : : : 1
: : : : : : description         = "SKYLINK"
: : : : : : dev                 = "s1"
: : : : : : restart
: : : : : : : 1
: : : : : : : : script          = "nsgsh -q .port.s1.restart"
: : : : : : : : timeout         = 180
: : : : : 2
: : : : : : description         = "MEFAGON"
: : : : : : dev = "s2"
: : : : : : restart
: : : : : : : 1
: : : : : : : : script          = "nsgsh -q .port.s2.restart"
: : : : : : timeout             = 120
: : : : : priority              = 300
```

Теперь в логе сервера должны быть видны попытки соединения со стороны клиента:

```
2010-02-15 00:29:09 INFO :Accept connection X.X.X.X:Y>>123.45.67.89:50005
2010-02-15 00:29:09 WARN :[test ATM]Can't resume tunnel, no such tunnel
2010-02-15 00:29:10 INFO :[test ATM]New tunnel initiated from X.X.X.X:Y
2010-02-15 00:29:10 INFO :[test ATM]Certificate /etc/uitcp/certs/server.pem expires in 365 days
2010-02-15 00:29:10 INFO :[test ATM]Start SSL handshake
2010-02-15 00:29:12 INFO :[test ATM]SSL handshake successfully done
```

Если соединение успешно, то можно переходить к следующему этапу.

б) Вариант 2×GPRS

Настройка SSL, журналов и сервера выполняется так же, как в предыдущем случае. Конфигурация туннеля на клиентском устройстве:

```
tunnel
: uitcp
: : configure
: : : mode                    = "client"
: : : tunnel
: : : : name                  = "ATM001"
: : : : description           = "test ATM"
: : : : servers
: : : : : 1                   = "123.45.67.89:50005"
: : : : keepalive             = 10
: : : : links
: : : : : 1
: : : : : : description       = "GPRS"
: : : : : : dev               = "s1"
: : : : : : restart
: : : : : : : 1
: : : : : : : : description   = "BEELINE"
: : : : : : : : script        = "nsgsh -q .port.s1.restart.aux"
: : : : : : : 2
: : : : : : : : description   = "MTS"
: : : : : : : : script        = "nsgsh -q .port.s1.restart.main"
: : : : : : : timeout         = 120
```

Особенность данной конфигурации состоит в скриптах, используемых для управления сотовым модулем. При разрыве соединения с МТС (верхняя SIM-карта) порт переконфигурируется для работы только с нижней SIM-картой и начинает работать через Билайн; при разрыве соединения с Билайн — наоборот. Приоритет между двумя каналами связи выключен, поскольку услуги обоих операторов предполагаются примерно равными по стоимости и по качеству.

ПРИМЕЧАНИЕ Данные скрипты изменяют значения параметров main.attempts и aux.attempts, поэтому после необходимо перед сохранением конфигурации проверять их и при необходимости восстанавливать.

в) Вариант Ethernet + GPRS

Настройка SSL, журналов и сервера выполняется так же, как в предыдущих двух случаях. Конфигурация туннеля на клиентском устройстве:

```
tunnel
: uitcp
: : configure
: : : mode                    = "client"
: : : tunnel
: : : : name                  = "ATM001"
: : : : description           = "test ATM"
: : : : servers
: : : : : 1                   = "123.45.67.89:50005"
: : : : keepalive             = 10
: : : : links
: : : : : 1
: : : : : : description       = "Ethernet"
: : : : : : gw                = "98.76.54.29"
: : : : : 2
: : : : : : description       = "MEFAGON"
: : : : : : dev = "s2"
: : : : : : restart
: : : : : : : 1
: : : : : : : : script        = "nsgsh -q .port.s2.restart"
: : : : : : : timeout         = 120
: : : : : priority            = 300
```

В конфигурации основного канала связи указан не интерфейс, а IP-адрес следующего шлюза, как это необходимо для Ethernet.

Рестартовать порт Ethernet бессмысленно, поскольку проблема находится, как правило, где-то дальше по каналу связи. (А в данном случае это и не возможно, поскольку все внешние порты коммутируются на разные VLAN на одном физическом порту процессора, и рестартовать можно только их все вместе.)

6–A.5. Настройка TCP-соединений в туннеле

Заключительный этап настройки *ui*TCP — создание TCP-соединений в построенном туннеле. Эта часть конфигурации уже никак не зависит от числа и типа используемых каналов связи и является общей для всех трёх типовых задач.

Соединения могут устанавливаться как в направлении от клиента к серверу, так и наоборот, поэтому здесь важно уже не отношение клиент/сервер (они относятся только к функционированию туннеля в целом), а отношение вызывающая/отвечающая сторона. На вызывающей стороне создаётся TCP-сокет, принимающий локальные соединения из сети источника. На отвечающей стороне настраиваются правила NAT, определяющие, с какими IP-адресами и номерами портов TCP пакеты будут препровождаться в сеть назначения.

Учитывая специфику массовых инсталляций, действительные адреса и порты, используемые для работы с конкретной площадкой, удобно прописывать на сервере *ui*TCP.

Настройка туннеля от банкомата к процессинговому серверу. Напомним, что на процессинге данный банкомат зарегистрирован под IP-адресом 10.0.1.1 и должен обращаться на порт TCP 20001; в то же время в типовой конфигурации всех банкоматов записаны адрес 192.168.0.1 и порт 20000.

На клиенте:

```
tunnel
: uitcp
: : configure
: : : tunnel
: : : : localListener
: : : : : 1
: : : : : : host          = "192.168.0.1"
: : : : : : port          = 20000
```

На сервере:

```
tunnel
: uitcp
: : configure
: : : clients
: : : : ATM001
: : : : : nat
: : : : : : 1
: : : : : : : inDstPort     = 20000
: : : : : : : outDstAddr    = "10.0.0.1"
: : : : : : : outDstPort    = 20001
: : : : : : : outSrcAddr    = "10.0.1.1"
```

Настройка соединения от административной станции к банкомату. Станция обращается к серверу NSG по адресу 10.0.0.7 и номеру порта TCP 30001 (для других площадок — 30002 и т.д.) в локальной сети банка; это соединение принимается сервером *ui*TCP и транслируется на удалённую площадку. Для краткости приведены только результирующие фрагменты конфигураций:

На клиенте:

```
tunnel
: uitcp
: : configure
: : : tunnel
: : : : nat
: : : : : 1
: : : : : : inDstPort     = 30001
: : : : : : outDstAddr    = "192.168.0.2"
: : : : : : outDstPort    = 23
```

На сервере:

```
tunnel
: uitcp
: : configure
: : : clients
: : : : ATM001
: : : : : localListener
: : : : : : 1
: : : : : : : host          = "10.0.0.7"
: : : : : : : port          = 30001
```

Настройка соединения (Telnet, SSH, HTTP, HTTPS — для данной настройки несущественно) от административной станции к устройству NSG. Станция обращается по адресу 10.0.0.7 и номеру порта TCP 40001 (40002 и т.д.) и попадает на клиентское устройство:

На клиенте:

```
tunnel
: uitcp
: : configure
: : : tunnel
: : : : nat
: : : : : 2
: : : : : : inDstPort     = 40001
: : : : : : outDstAddr    = "127.0.0.1"
: : : : : : outDstPort    = 23
```

На сервере:

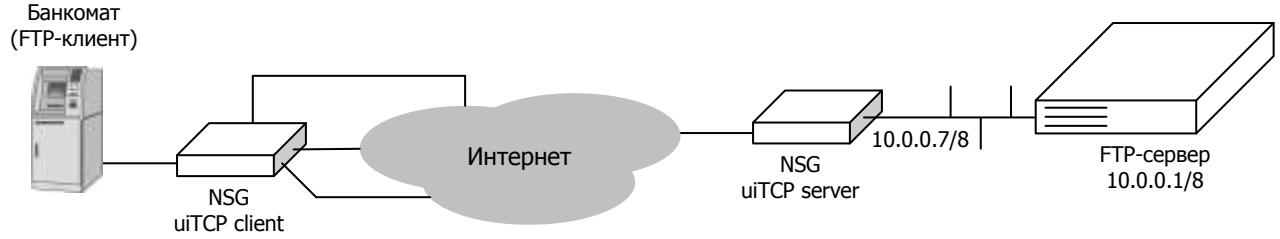
```
tunnel
: uitcp
: : configure
: : : clients
: : : : ATM001
: : : : : localListener
: : : : : : 2
: : : : : : : host          = "10.0.0.7"
: : : : : : : port          = 40001
```

6–A.6. Настройка сокетов типа net в туннеле

Датаграммные соединения в туннеле, реализуемые при помощи сокетов типа net, являются полными аналогами соединений WAN "точка-точка". По ним может передаваться любой IP-трафик без ограничения, в частности, протоколы, задействующие несколько взаимосвязанных TCP-портов (FTP, Amanda и т.п.), IPSec, ФПСУ и др.

IP-интерфейсам, которые являются концами этих соединений, должны быть назначены отдельные IP-адреса (как правило, с маской /30 или ненумерованные), и через них необходимо создать маршруты к удалённым сетям в ту и в другую сторону. Поскольку настройка обратных маршрутов (из центра в клиентскую LAN) на всём протяжении не всегда доступна или, как минимум, трудоёмка, то вместо неё можно использовать NAT на выходе в вышестоящую сеть — что и рекомендуется делать, для уменьшения числа индивидуальных настроек и снижения вероятности человеческих ошибок. Он же позволяет унифицировать конфигурацию LAN на всех площадках.

Пример конфигурации соединений для работы клиента FTP на банкомате с сервером на центральной площадке. (Детали, относящиеся к построению собственно туннеля, опущены.)



На клиенте:

```
ip
: route
: : 1
: : : network          = "10.0.0.0/8"
: : : gateway          = "10.0.0.7"
: nat
: : POSTROUTING
: : : 1
: : : : out-interface  = "ATM001.1"
: : : : target         = "MASQUERADE"
: : HELPERS
: : : FTP
: : : : enable = true
.....................................
tunnel
: uitcp
: : configure
: : : tunnel
: : : : name       = "ATM001"
: : : : socket
: : : : : 1
: : : : : : type         = "net"
: : : : : : net
: : : : : : : ifAddress
: : : : : : : : prefix   = "172.31.0.1/31"
: : : : : : : : peer     = "10.0.0.7"
.....................................
```

На сервере:

```
ip
: nat
: : POSTROUTING
: : : 1
: : : : out-interface  = "eth1"
: : : : target         = "SNAT"
: : : : to-source      = "10.0.0.7"
: : HELPERS
: : : FTP
: : : : enable = true
port
: eth1
: : ifAddress
: : : prefix           = "10.0.0.7/8"
.....................................
tunnel
: uitcp
: : configure
: : : mode             = "server"
: : : clients
: : : : ATM001
: : : : : socket
: : : : : : 1
: : : : : : : type       = "net"
: : : : : : : net
: : : : : : : : ifAddress
: : : : : : : : : prefix = "10.0.0.7/32"
: : : : : : : : : peer   = "172.31.0.1"
.....................................
```

В данном случае на стороне сервера используется ненумерованный (/32) интерфейс с адресом 10.0.0.7, "позаимствованным" у локального интерфейса eth1 и одинаковым для всех клиентов. Противоположная сторона соединения указана явно. На клиентской стороне соединения используется уникальный "собственный" адрес с маской /31 — максимальной, позволяющей определять такие адреса.

NAT на выходе из клиентского маршрутизатора в туннель и из сервера в локальную сеть центрального узла позволяет не прописывать маршруты от сервера FTP к банкоматам. Все соединения к серверу FTP будут приходить с адреса сервера *ui*TCP 10.0.0.7, поэтому необходимо и достаточно, чтобы на нём был известен только маршрут на этот адрес (через шлюз по умолчанию, или, в данном случае, оба они находятся в одной IP-подсети).

На клиентской стороне следовало бы также использовать SNAT, поскольку адрес интерфейса известен заранее; однако в данном случае сознательно использован MASQUERADING — всё с той же целью сокращения уникальных настроек на каждом клиенте. Из этих же соображений маршрут в сеть 10.0.0.0/8 указан единообразно через IP-адрес серверного конца соединения, а не через уникальный dev = ATM001.1 .

6–A.7. Настройка фильтров

Для безопасной работы любых устройств, подключённых к Интернет, необходимо настроить на всех внешних интерфейсах фильтры, отсекающие все посторонние пакеты. Пример настройки для вышеприведённых конфигураций:

На клиенте:

```
ip
: filter
: : INPUT
: : : 1
: : : : in-interface        = "s1"
: : : : source              = "123.45.67.89"
: : : : target              = "ACCEPT"
: : : 2
: : : : in-interface        = "s2"
: : : : source              = "123.45.67.89"
: : : : target              = "ACCEPT"
: : : default-target        = "DROP"
: : FORWARD
: : : default-target        = "DROP"
```

На сервере:

```
ip
: filter
: : INPUT
: : : 1
: : : : in-interface        = "s1"
: : : : source              = "194.149.67.129"
: : : : destination         = "123.45.67.89"
: : : : target              = "ACCEPT"
: : : 2
: : : : in-interface        = "s1"
: : : : protocol            = "tcp"
: : : : destination         = "123.45.67.89"
: : : : destination-port    = 50005
: : : : target              = "ACCEPT"
: : : default-target        = "DROP"
: : FORWARD
: : : default-target        = "DROP"
```

ПРИМЕЧАНИЕ Данные примеры разрешающих фильтров относятся исключительно к *ui*TCP и запрещают любой иной трафик через порты, подключённые к сетям общего пользования. Для удалённого управления и других задач необходимо открыть соответствующие порты, например, 443 (HTTPS) и 22 (SSH).

6–A.8. Клонирование конфигурации

Для переноса полученной конфигурации на другие клиентские устройства рекомендуется следующая последовательность действий:

1. Сохранить полученную конфигурацию клиентского устройства по HTTP при помощи узла .system.configurations . Следует выгрузить с устройства целиком директорию /etc (вместе со всеми настройками основного ПО, файлами и настройками *ui*TCP, файлами сертификатов и стартовыми файлами) в сжатом виде.
2. Восстановить заводскую конфигурацию второго клиентского устройства и загрузить на него сохранённую конфигурацию (в этом же узле).
3. Изменить на втором устройстве имя клиента на какое-нибудь нейтральное (например, XXX), чтобы не возникало конфликтов при одновременной работе двух клиентов под одним именем. Удалить файлы сертификатов:

   ```
   rm -r /etc/uitcp/certs/*
   ```

4. Сохранить полученную конфигурацию. Эта конфигурация готова к загрузке на третье и все последующие клиентские устройства.
5. Установить новое имя клиента (например, ATM002). Сгенерировать на сервере набор сертификатов для него, загрузить на клиента и установить. Рестартовать устройство. Эти же уникальные настройки выполняются на третьем и последующих клиентах.

ПРИМЕЧАНИЕ В зависимости от конфигурации, у каждого клиента могут быть также другие уникальные параметры: статические IP-адреса, имя/пароль для аутентификации у сотового оператора (в частности, для получения статического адреса в Скайлинк) и т.п.