InfiNet Wireless Session Gateway Controller

Руководство Системного Администратора

Дата создания**: 14 July 2004**

Оглавление

I. Введение

Данный документ является описанием системы InfiNet Wireless Session Gateway Controller (SGC) и содержит инструкции по инсталляции, настройке и эксплуатации системы. В настоящем руководстве подробно изложены способы и приёмы, которые позволяют эффективно использовать систему.

1. Для кого этот документ

Руководство предназначено для системного администратора, чьей задачей является установка, настройка и эксплуатация SGC.

2. Организация руководства

Документ состоит из следующих составных частей:

Введение

Содержит сведения о назначении и организации документа

Общее описание

Ознакомит с основными понятиями используемыми в системе SGC, даст краткое описание продукта и опишет его основные функциональные характеристики и способы использования.

Инсталляция

Ознакомит с процедурой установки продукта и требованиям к аппаратному обеспечению и операционной системе.

Настройка

Расскажет об основных действиях по конфигурированию системы, регистрации абонентов, выборов маршрута терминации, настройки связей с гейткиперами и шлюзами других провайдеров и различных тонкостях в этих вопросах. Также в этой главе особое внимание будет уделено особенностям учёта звонков и настройке взаимодействия с биллинговой системой.

Синтаксис файла конфигурации

Содержит детальное описание всех выражений, используемых для настройки и управления.

Справочная информация

Информация, описывающая форматы файлов CDR и log, а также форматы пакетов к RADIUS-серверу и другая справочная информация.

3. Список литературы

1. ITU-T Recommendation H.323, Packet-based multimedia communications systems
2. ITU-T Recommendation H.225.0, Call signalling protocols and media stream packetization for packet-based multimedia communication systems
3. ITU-T Recommendation H.245, Control protocol for multimedia communication
4. ITU-T Recommendation H.235, Security and encryption for H-series (H.323 and other H.245-based) multimedia terminals

5. ITU-T Recommendation H.450.1, Generic functional protocol for the support of supplementary services in H.323
6. ITU-T Recommendation H.450-2, Call transfer supplementary service for H.323
7. ITU-T Recommendation H.450.3, Call diversion supplementary service for H.323
8. ITU-T Recommendation H.450.4, Call hold supplementary service for H.323
9. ITU-T Recommendation H.450.5, Call park and call pickup supplementary services for H.323
10. ITU-T Recommendation H.450.6, Call waiting supplementary service for H.323
11. ITU-T Recommendation H.450.7, Message waiting indication supplementary service for H.323
12. ITU-T Recommendation H.450.8, Name identification supplementary service for H.323
13. RFC 1889 "RTP: A Transport Protocol for Real-Time Applications. Audio-Video"
14. RFC 2865, Remote Authentication Dial In User Service (RADIUS)
15. RFC 2866, RADIUS Accounting

II.Общее описание

Infinet Wireless Session Gateway Controller (SGC) выполняет функции ядра системы провайдера IP-телефонии. Он обеспечивает основные функции по обслуживанию как собственных абонентов (IP-телефонов и их шлюзов) так и по взаимодействию с оборудование других провайдеров, включая аутентификацию, гибкую маршрутизацию и учёт звонков. SGC предлагает широкий спектр возможностей по трансляции адресов и согласованию планов нумерации межоператорских систем IP-телефонии. SGC поддерживает современную версию протокола сигнализации H.323 версии 4.

1. Базовые понятия

Раздел опишет основные термины и понятия используемые в данном руководстве.

Абонент

Абонент - некая сущность, которая может совершать и/или принимать звонки и для которой в конфигурации могут быть указаны всевозможные параметры. Абонентами могут быть различные терминалы (IP телефоны, шлюзы IP телефонии или другие гейткиперы). Звонок всегда происходит между двумя абонентами: вызывающим '**caller**' и вызываемым '**callee**', причём один и тот же абонент может одновременно выступать в двух ролях (например при звонке с одного порта шлюза IP телефонии к другому порту этого же шлюза). Для описания абонента в конфигурации SGC используется директива **user** или **gatekeeper** (абонент с расширенными функциональными свойствами). Совершенно не обязательно с одним абонентом связывать одно физическое устройство. SGC позволяет одним выражением **user** описать несколько (хоть несколько тысяч) конкретных терминалов, используя гибкий синтаксис аутентификации (в том числе и в биллинговой системе).

Псевдонимы

Псевдонимы – аналог абонентских номеров, используемых в ТФОП. В H.323 данная сущность называется псевдонимом (не номером), потому что имеет более широкое понимание. Так, в H.323 псевдонимами могут быть не только номера, образованные цифрами, но и другие идентификаторы, такие как, произвольные строки, состоящие из символов, в том числе содержащие пробел или другие специальные символы. Псевдонимом может выступать e-mail адрес а также другие типы псевдонимов, описанные в стандарте. Кроме того, абонент может быть вызван (или описан) не только единичным псевдонимом, но и набором из псевдонимов. Например, некто имеет возможность позвонить, представив для обозначения терминирующей стороны любое из следующего: '311212', 'JohnDoe'. В этом случае будет выбран только тот абонент, среди псевдонимов которого встречаются оба представленных псевдонима.

Псевдонимы в пакетах установления соединения

В пакетах, инициирующих соединение встречаются два типа псевдонимов: Псевдонимы вызываемого абонента и псевдонимы вызывающего абонента. Псевдонимы вызываемого '**callee alias**' абонента это псевдонимы абонента, на который будет терминирован вызов. Псевдонимы вызывающего абонента называются '**caller alias**'. В пакете RAS уровня AdmissionRequest эти параметры находятся в полях DeastinationInfo и SrcInfo соответственно. Если речь идёт о статическом абоненте, то маршрутизация производится по полям Q.931 пакета Setup. Это поля CalledPartyNumber и DestinationAddress

для вызываемого абонента и CallingPartyNumber и SourceAddress для вызывающего. Все эти псевдонимы могут быть транслированы командой **'translate ... alias'**.

2. Функциональные характеристики

SGC обладает следующими функциональными характеристиками по управлению и мониторингу сетей IP-телефонии:

Проксирование

Возможность проксирования:

- сигнальный трафик (H.225, H.245)
- мультимедийный трафик (RTP/RTCP)

Это позволяет соединять свои внутренние IP-сети с внешним миром, полностью обезличивая проходящий трафик и позволяет с высокой точностью производить учёт.

Маршрутизация вызовов

Маршрутизация звонков происходит с учётом следующих параметров:

- псевдонимов (номеров) вызывающего и вызываемого абонентов
- принадлежности вызывающего абонента к определённой группе
- доступности шлюза места назначения
- приоритета шлюза места назначения
- времени
- балансировка нагрузки в пределах одного направления
- выбор альтернативного маршрута при отказе обслуживания более приоритетного (call hunting)
- загруженности зон и связей между ними
- загруженности оборудования

Прим. *Наличие «Smart» процедуры, FastStart и H.245 address не останавливают процесс выбора альтернативных абонентов, подробное описание процедуры «Smart» описано в разделе **Хантинг**.*

Конверсия H.323-диалектов

Позволяет SGC осуществлять соединения между устройствами разных производителей, несовместимых между собой

Адресная трансляция, преобразование псевдонимов (телефонных номеров)

Трансляция псевдонимов вызывающего и вызываемого абонентов с использованием **регулярных выражений** и именованных переменных позволяет:

- согласовывать планы нумерации провайдеров IP-телефонии с внутренним планом нумерации
- обеспечить предоставление "нужных" номеров для биллинговой системы
- осуществлять гибкую маршрутизацию
- производить аутентификацию абонентов, используя части псевдонимов, в том числе и через биллинговую систему

Зональная организация и управление

SGC позволяет детально описать инфраструктуру VOIP-сети, используя зоны и связи между ними, указывая доступную полосу пропускания из зоны в

зону, которая будет учитываться при авторизации и распределению загрузки.

Также существует возможность ограничения числа одновременных звонков у абонента в зоне или в линке между зонами.

Авторизация вызовов

Авторизация может быть осуществлена различными способами с учётом следующих параметров:

- возможность свободного доступа (без авторизации)
- на основании IP-адреса инициатора
- посредством службы удаленной аутентификации RADIUS
- на основании произвольной информации по звонку (например, часть набранного номера может выступить в роли пароля или идентификатора абонента), используя именованные переменные H.235
- передача H.235 Cisco Access Token (CAT) на RADIUS через Chap-Password

Биллинговые возможности

Возможности по учёту звонков:

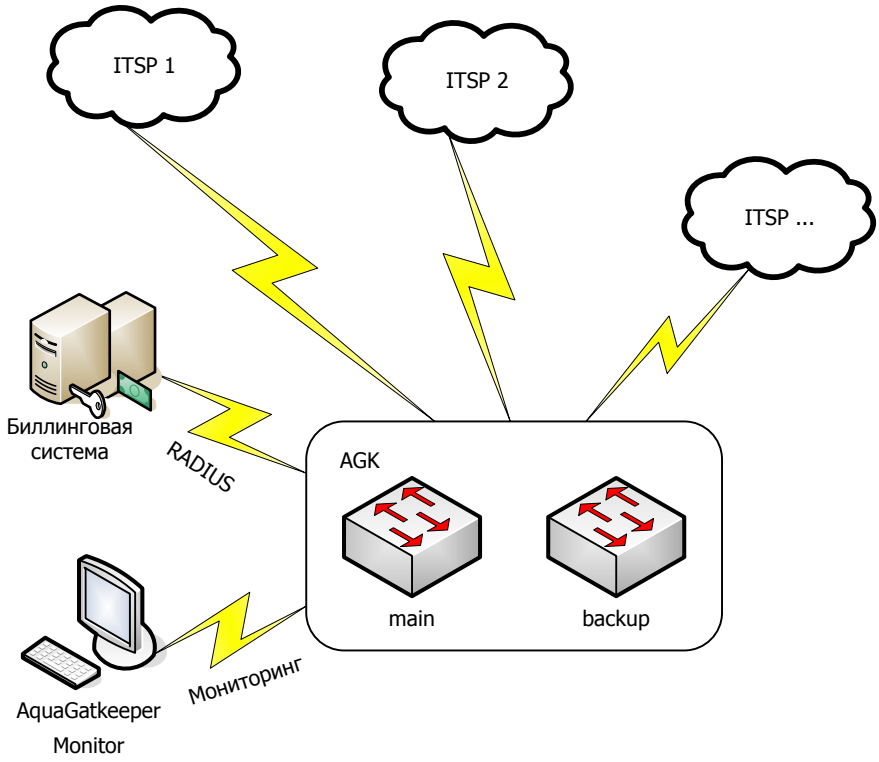
- протокол RADIUS (в том числе с использованием Cisco VSA)
- CDR-файла
- возможности управления звонками
- вывод списка текущих звонков
- вывод детальной информации о звонке
- принудительное завершение звонка

3. Примеры использования

Перечисленные функциональные характеристики открывают широкие возможности по использованию SGC как крупным провайдером IP-телефонии, так и небольшим, только начинающим развивать свой бизнес. SGC будет также полезен предприятиям, не связанным с представлением услуг IP-телефонии для организации своей внутренней IP-телефонной инфраструктуры, для объединения филиалов своего предприятия, а также для доступа к услугам провайдера IP-телефонии (ITSP).

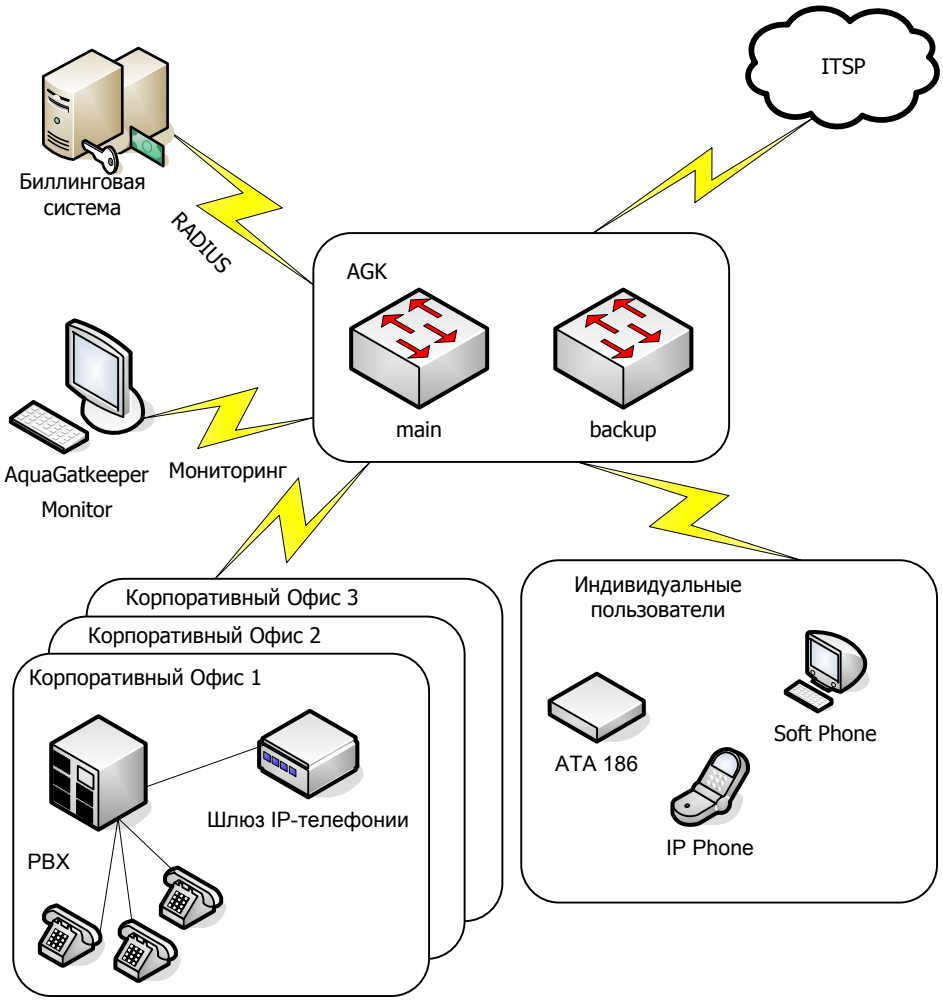
Межоператорский обмен транзитным трафиком

Высокая производительность, резервирование, гибкая маршрутизация и учёт позволяют организовать обмен VOIP-трафиком между операторами IP-телефонии.



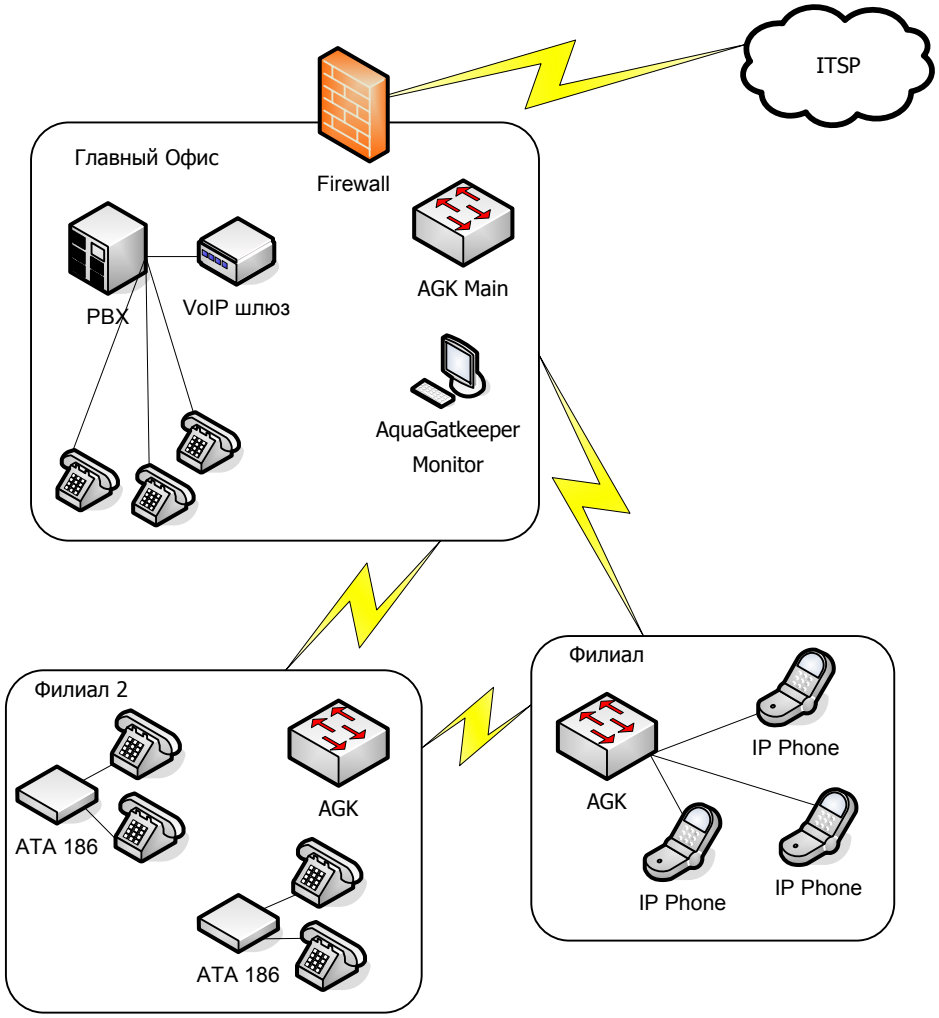
Предоставление услуг IP-телефонии

SGC позволяет организовать внутренний план нумерации для каждого корпоративного клиента, и в тоже самое время обслуживать индивидуальных абонентов.



Организация защищённой приватной сети

SGC позволяет компании объединить несколько своих филиалов и иметь одну защищённую точку входа.



III.Инсталляция

SGC выпускается в нескольких модификациях, работающих под управлением разных операционных систем:

- WANFlex
- FreeBSD 32bit и 64bit
- Linux
- Windows

функционально ничем не отличающихся друг от друга.

1. WANFlex

Для запуска и конфигурирования процесса SGC, нужно настроить операционную систему WANFlex в соответствии с прилагаемой документацией, затем осуществить запуск, дальнейшее конфигурирование осуществляется с использованием SGCMonitor.

Шаги

- Указать IP-адрес на интерфейсе eth0
- Создать конфигурация гейткипера
- Запустить гейткипер
- Произвести конфигурацию гейткипера используя SGCMonitor

Пример

```
console>ifconfig eth0 192.168.0.1/24 up
console>gatekeeper create
console>gatekeeper start
console>configuration save
```

Устройство готово к работе и дальнейшей конфигурации.

2. Unix

Система SGC поставляется в виде сжатого архива, например файл SGC122FreeBSD.tgz содержит SGC версии 1.22 для операционной системы FreeBSD.

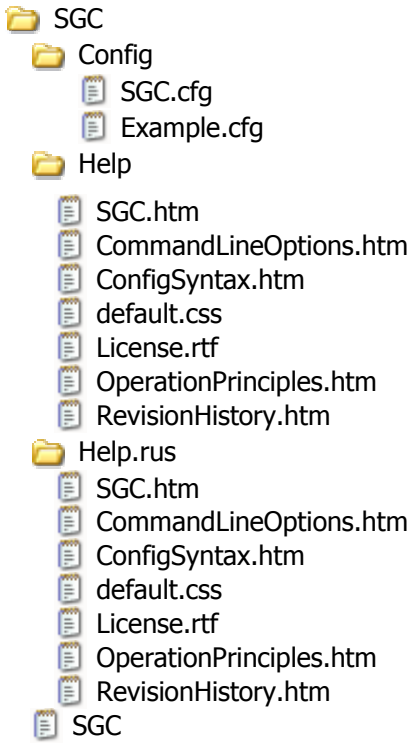
Для установки дистрибутива необходимо извлечь файлы находящиеся в архиве с помощью команды tar.

Пример

```
tar –xzf SGC122FreeBSD.tgz
```

все файлы распакуются в каталог с названием SGC, который будет иметь следующую структуру

Структура



Файл	Описание
SGC	исполняемый модуль
SGC.cfg	Конфигурационный файл

Запуск

SGC может работать в двух режимах:

- Консольный (console mode)
- Демон (Unix daemon mode)

Консольный

Этот режим не является основным и служит только для отладки системы. В этом режиме SGC выводит лог своей работы непосредственно на консоль.

Для запуска в консольном режиме необходимо выполнить команду:

```
./SGC --conf=Config/SGC.cfg
```

Демон

Основной режим работы SGC (его удобно применять также и при старте операционной системы из rc скриптов). В этом режиме при запуске процесс SGC открепляется от консоли и переходит в состояние Unix daemon при этом весь лог программы выводится в файл, указанный в конфигурации с помощью директивы **log file <filename>**; а события информирующие об ошибках и требующие особого внимания в системный журнал Операционной системы (syslog).

Для запуска в режиме daemon, выполнить команду:

```
./SGC --daemon --conf=Config/SGC.cfg
```

Теперь Ваша система готова к эксплуатации и дальнейшей настройке.

Прим. *Для удобной настройки конфигурации и мониторинга работы SGC можно установить SGCMonitor из пакета дистрибутива под Windows.*

3. Windows

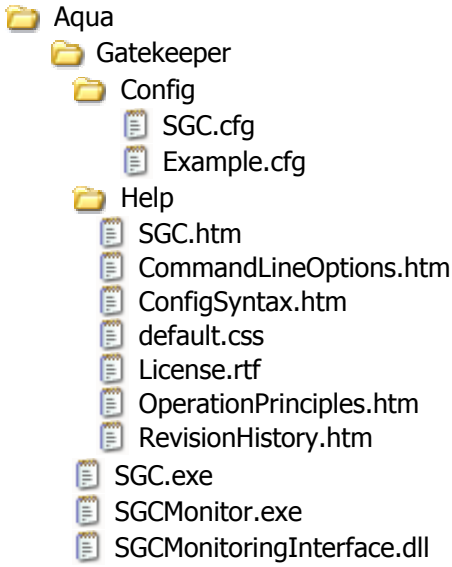
Установка для операционной системы Windows полностью автоматизирована и происходит с помощью встроенной компоненты Windows Installer.

SGC поставляется в виде файла «Windows Installer Package», например файл SGC122.msi содержит SGC и SGCMonitor версии 1.22 для операционной системы Windows.

Для установки получите контекстное меню и выберите пункт «Установка/Install». Далее следуя инструкциям «Wizard» произведите установку.

«Wizard» позволяет выбрать каталог установки и компоненты системы, Вы можете выбрать установить только SGC и/или Монитор. После завершения работы программы инсталлятора система готова к запуску и конфигурированию.

Структура



Файл	Описание
SGC.exe	исполняемый модуль
SGC.cfg	конфигурационный файл
SGCMonitor.exe	исполняемы модуль программы Монитор

Запуск

SGC может работать в двух режимах:

- Приложения (console mode)
- Сервис (Service)

Приложения

Этот режим не является основным и служит только для отладки системы, в этом режиме SGC свой лог работы выводит непосредственно в окне консоли.

Для запуска в режиме приложения, выполнить команду:

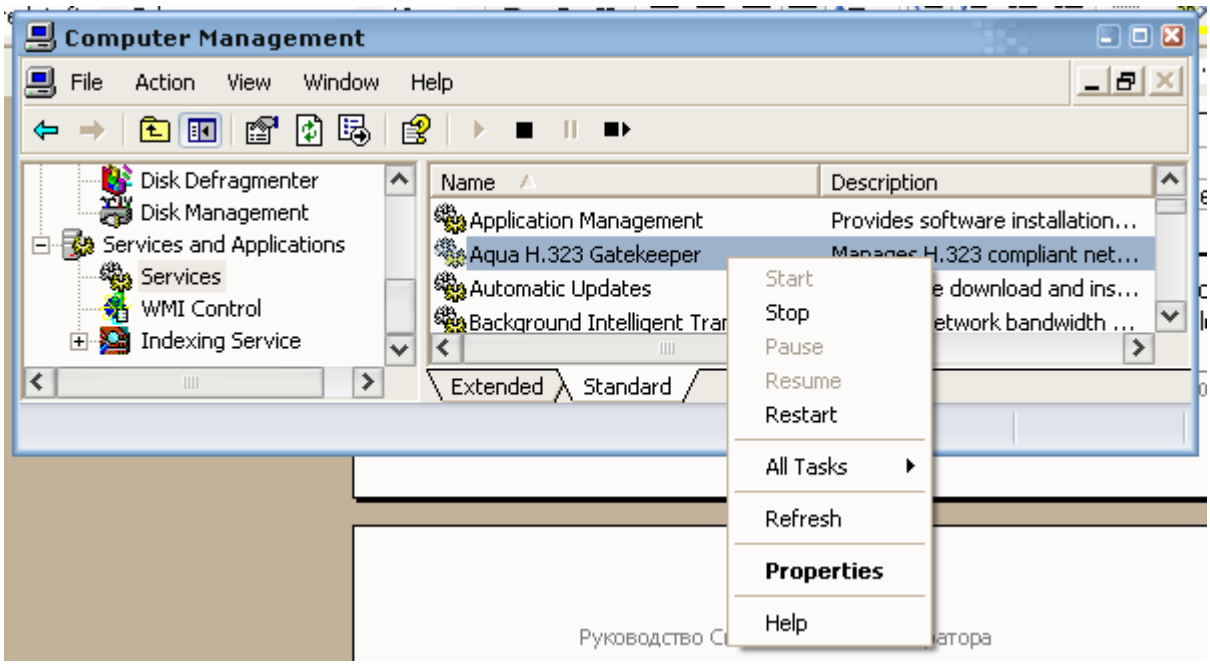
```
SGC.exe --conf=Config/SGC.cfg
```

Сервис

Основной режим работы SGC его удобно применять также и для автоматического запуска при старте операционной системы, в этом режиме SGC не использует никаких окон при этом весь лог программы выводится в файл указанный в конфигурации с помощью директивы **log file <filename>;** а события информирующие об ошибках и требующие особого внимания в системный журнал Операционной системы (EventLog).

Для запуска в режиме сервис, у вас есть два варианта

1. Используя системную утилиту «Computer Manager» выберите пункт Services и далее выбрав сервис «» из контекстного меню укажите Start.



2. Из командной строки выполнить команду

```
SGC.exe --start
```

или

```
net start SGC
```

Теперь Ваша система готова к эксплуатации и дальнейшей настройке.

IV. Настройка

Данная глава построена по принципу от простого к сложному, в начале будут описаны необходимые действия по глобальным настройкам системы и о системе вывода вспомогательной отладочной информации, далее подробно будут описаны способы регистрации абонентов, потом будет рассказано о том гак SGC выбирает осуществляет выбор вызываемого абонента, о трансляции псевдонимов и согласовании планов нумерации различных систем в конце будет рассказано о том как объединить несколько SGC в одну сеть с примерами и возможными вариантами.

Перед тем как продолжить изучение данной главы, авторы рекомендуют ещё раз внимательно ознакомится с **базовыми понятиями** и терминологией используемой в этом руководстве.

1. Начальная конфигурация

Стартовый конфигурационный файл (поставляемый с дистрибутивом), позволяет SGC регистрировать любых абонентов с любыми псевдонимами. Зарегистрированный абонент будет доступен по псевдонимам (номерам) которые он представил для регистрации. Такой файл поставляется только ради примера и в 99.99% случаев не приемлем для нормальной работы, так как у вас нет никакой возможности осуществлять контроль за тем какие абоненты регистрируются и куда будут терминироваться звонки, просто этот файл показывает что SGC имеет тривиальную функциональность описанную стандартом H.323.

После того, как абонентское оборудование зарегистрируется на, SGC, можно начинать звонить, однако, первое, что нужно сделать - это убрать из конфигурации строку, которая разрешает регистрироваться кому угодно и звонить куда угодно.

```
/*
  user Anyone login "(.+)" set UserAlias add "\1",
               alias UserAlias;
*/
```

В секции **system** необходимо установить IP адрес, который SGC будет использовать для работы. Иначе SGC будет использовать IP-адрес, который указан в DNS для этого компьютера, что не всегда правильно, так как на компьютере может быть несколько интерфейсов и/или несколько IP адресов на одном интерфейсе (IP alias). В этом случае Вы полностью контролируете ситуацию, указав нужный Вам адрес. Эта методика позволит также запустить несколько копий SGC, каждая из которых будет работать на своём IP-адресе.

В этой же секции нужно указать идентификатор гейткипера. Этот идентификатор имеет важное значение для работы абонентов по протоколу RAS, так как SGC будет проверять совпадает ли значение, передаваемое абонентами в RAS пакетах с указанным в конфигурации.

Пример

```
system
{
  identifier 'AGK';
  address 192.168.0.1;
}
```

Хочется также отметить что синтаксис файла конфигурации позволяет описывать ‘сложные выражения’ такие как **system**, **log**, **user**, **group**, и другие двум способами, об этом подробно написано в главе ‘Синтаксис файла конфигурации’ - **Выражения**

Пример

```
system identifier 'AGK', address 192.168.0.1;
```

Далее для того чтобы понимать, контролировать, диагностировать работу и неполадки системы, необходимо произвести настройку подсистемы вывода отладочной информации и протокола работы и используя секцию **log**, в которой указывается **файл** для вывода информации и необходимо указать какую информацию нужно выводить в журнал.

Пример

```
log
{
  file '/home/gatekeeper/logs/gatekeeper.log';
  RAS messages yes;
  Q931 messages yes;
  H245 messages yes;
  radius messages yes;

  messages dump no;

  registrations          yes;
  connection status      yes;
  chosen route           yes;
  available bandwidth    yes;
  bandwidth changes      yes;
}
```

Для того чтобы изменения в конфигурации вошли в силу необходимо выполнить одно из ниже перечисленных действий:

- Reload Configuration (SGC Monitor)
- Послать сигнал процессу USR1, например с помощью команды kill (Unix)

Перечисленные команды не прерывают открытые соединения, если не были удалены пользователи (**user**, **gatekeeper**), зоны и связи (**zone**, **link**) участвующие в соединении и не была превышена разрешённая полоса пропускания (**bandwidth**, **connections**). Все вновь открывающиеся соединения будут работать по новым правилам.

Как уже говорилось ранее мониторинг системы, изменение и проверку синтаксиса конфигурации удобнее всего осуществлять с помощью SGC Monitor, для ограничения доступа к SGC Вы можете задать пароль и/или ограничить набор адресов с которых можно подключаться для управления. Протокол общения Monitor и Gatekeeper открыт и его **описание** приведено в главе Справочная Информация.

2. Методы подключения абонентов

Любой звонок в SGC может быть осуществлён только между зарегистрированными в системе абонентами, если SGC не смог идентифицировать абонента как зарегистрированного, то на любой запрос полученный от такого абонента будет послан отказ.

SGC поддерживает два типа подключения абонентов: RAS и статическое подключения. Для обоих типов поддерживается полная схема маршрутизации вызовов, контроль полосы полезных данных, хантинг. Однако, RAS подключение предпочтительней, так как при использовании этого протокола достигается большая гибкость и скорость предоставления сервиса.

RAS

RAS подключение – это базовый тип подключения абонентов гейткипера, описанный в стандарте ITU-T H.323.

Для лучшего понимания ситуации рассмотрим процесс процедуры регистрации по протоколу RAS подробно. Терминал (IP телефон, VoIP шлюз, гейткипер, другое устройство или программа), желающий подключится к гейткиперу должен послать пакет RegistrationRequest (RRQ) гейткиперу на его RAS UDP порт (по умолчанию порт имеет значение 1719), указав свой Call Signal (CS) адрес и RAS адрес а также перечислив свои login aliases (псевдонимы), которые он хочет зарегистрировать. Login aliases (псевдонимы для регистрации) это ключевой момент именно по их набору SGC и определяет что это за абонент и находит его в конфигурации и только потом начинают срабатывать другие проверки IP адрес регистрируемый абонентом (см. выражение **address**), H.235 информация переданная абонентом (см. выражение **h235 authentication**).

В случае если гейткипер не нашёл в своей конфигурации абонента, у которого все переданные псевдонимы (login alias) подошли или таких абонентов несколько в регистрации будет отказано и в ответ будет послан пакет RegistrationReject (RRJ) с указанной причиной отказа. Если же все перечисленные проверки прошли, то гейткипер подтвердит регистрацию, отослав пакет RegistrationConfirm (RCF), выделив ему уникальный идентификатор (EndpointIdentifier) и указав через какое время абонент должен будет подтвердить свою регистрацию (TimeToLive, см. **registration validity**). После этого абонент считается зарегистрированным в системе и может принимать и инициировать звонки.

При данном типе подключения требуется при помощи выражений '**login**', перечислить псевдонимы, представляемые абонентом для регистрации. Если Вы не уверены, какие псевдонимы абонент представляет для регистрации, то можно включить печать RAS-сообщений (при помощи команды '**log RAS messages**') и посмотреть сообщение RRQ. Необходимо обратить внимание на список TerminalAlias – это как раз и есть те самые псевдонимы (login alias), Каждый из указанных в этом списке псевдонимов должен совпасть хотя бы с одним из выражений **login** в конфигурации у данного абонента (user).

Пример

```
RAS, got message from 192.168.0.2:3954
Message RegistrationRequest {
  RequestSeqNum: 3 (0x3)
  ProtocolIdentifier: { 0, 0, 8, 2250, 0, 4 }
  DiscoveryComplete: true
  CallSignalAddress count(1) {
    #0 IpAddress {
      Ip: 4 octets [C0 A8 00 02] (192.168.0.2)
      Port: 1720 (0x6b8)
    }
  }
  RasAddress count(1) {
    #0 IpAddress {
      Ip: 4 octets [C0 A8 00 02] (192.186.0.2)
      Port: 3954 (0xf72)
    }
  }
  TerminalType {
    Vendor {
      Vendor {
        T35CountryCode: 181 (0xb5)
        T35Extension: 0 (0x0)
        ManufacturerCode: 0 (0x0)
```

```
      }
      ProductId: 'AquA IPhone'
      VersionId: '1.0.1'
    }
    Terminal {
    }
    Mc: false
    UndefinedNode: false
  }
  TerminalAlias count(2) {
    #0 DialedDigits: '1001'
    #1 H323_ID: 'john'
  }
  GatekeeperIdentifier: 'AGK'
  EndpointVendor {
    Vendor {
      T35CountryCode: 181 (0xb5)
      T35Extension: 0 (0x0)
      ManufacturerCode: 0 (0x0)
    }
    ProductId: 'AquA IPhone'
    VersionId: '1.0.1'
  }
  TimeToLive: 60 (0x3c)
  KeepAlive: false
  WillSupplyUUIEs: false
  MaintainConnection: false
  SupportsAltGK
}
```

В таблице приведены ключевые параметры в пакете:

Параметр	Значение	Описание
CallSignalAddress	192.168.0.2:1720	IP-адрес, который устройство регистрирует на гейткипере
TerminalAlias	1001, john	Список псевдонимов, по которому гейткипер определит что это за абонент
GatekeeperIdentifier	AGK	Идентификатор гейткипера к которому желает подключится абонент, должен соответствовать указанному в конфигурации **идентификатору**.

Для успешной регистрации в конфигурации должен быть описан соответствующий абонент (**user**)

Пример

```
user JohnDoe
{
  login 'john';
  login '1001';
  address 192.168.0.2 allow;
}
```

или так, **address** можно не указывать в этом случае абонент сможет зарегистрироваться с любого адреса.

Пример

```
user JohnDoe
{
  login 'john|1001';
}
```

После получения RRQ и успешной регистрации, гейткипер выдаст пакет RCF в котором укажет EndpointIdentifier и TimeToLive.

```
User registered:  JohnDoe at 192.168.0.2:1720
RAS, send message to 192.168.0.2:3954
Message RegistrationConfirm {
  RequestSeqNum: 3 (0x3)
  ProtocolIdentifier: { 0, 0, 8, 2250, 0, 4 }
  CallSignalAddress count(1) {
    #0 IpAddress {
      Ip: 4 octets [C0 A8 00 01] (192.168.0.1)
      Port: 1720 (0x6b8)
    }
  }
  TerminalAlias count(1) {
    #0 DialedDigits: '1001'
    #1 H323_ID: 'john'
  }
  EndpointIdentifier: '81ebc4'
  AlternateGatekeeper: <empty>
  TimeToLive: 60 (0x3c)
  WillRespondToIRR: true
  MaintainConnection: false
}
```

Теперь ваш абонент может осуществлять звонки. Для приёма звонков (терминации), необходимо с помощью выражения **alias** задать псевдонимы, звонки на которые должны терминироваться на нём, более подробно об этом далее в пункте '**административное присвоение псевдонимов**'.

Пример

```
user JohnDoe
{
  login 'john|1001';
  alias '1001';
}
```

На абонента JohnDoe можно будет позвонить, набрав 1001 после его успешной регистрации.

Статическое

Статическое подключение не подразумевает регистрации абонента, так же как и любой другой RAS-процедуры. Статический абонент будет зарегистрирован в системе автоматически при старте системы. Объявление статического абонента производится при помощи директивы '**static**'.

Пример

```
user GW3
  static 192.168.1.1;
```

Если данный абонент предусматривает терминацию, то требуется объявить набор псевдонимов, по которым он будет доступен, точно так же как мы делали это для RAS абонента. Более подробно об этом см. в пункте '**административное присвоение псевдонимов**'.

Пример

```
user GW3
{
  static 192.168.1.1;
  alias '22\d{4}';
}
```

На абонента GW3 можно будет позвонить сразу после старта системы, набрав «22» и ещё 4 любых цифры.

Выражение **address** для статически регистрируемых абонентов имеет несколько другое значение. Если для RAS-абонентов это адрес и/или

диапазон адресов с которых данный абонент может регистрироваться в системе, то для статических абонентов это адрес и/или набор адресов, звонки с которых будут относиться к данному абоненту, используя CS порт гейткипера (получение пакета Q.931:Setup).

Гейткипер отвергнет звонок, полученный на его CS адрес, если не сможет соотнести его ни к одному абоненту.

Так как выражение «**static**» включает/подразумевает в себя выражение «**address**», то необходимо перед использованием static исключить данный IP-адрес если есть другие абоненты звонки которых могут придти с того же адреса.

Пример

```
// Incoming calls only from SomeNetwork:192.168.100.0-255
user SomeNetwork
{
  address 192.168.100.0-255;
  static; // необходимо для статической регистрации
}

// Terminate calls only to 192.168.100.1
user SomeNetworkGateway
{
  address 192.168.100.1 false;
  static 192.168.100.1;
  alias '7095.+';
}
```

Или наоборот, исключить адрес шлюза терминации из абонента принимающего звонки из указанного диапазона адресов, включающего в себя адрес этого шлюза.

Пример

```
// Incoming calls only from SomeNetwork:192.168.100.0-255
user SomeNetwork
{
  address 192.168.100.1 false; // SomeNetworkGateway's IP
  address 192.168.100.0-255;
  static; // необходимо для статической регистрации
}

// Receive and Terminate calls at 192.168.100.1
user SomeNetworkGateway
{
  static 192.168.100.1;
  alias '7095.+';
}
```

Выражение «**address** 192.168.100.1 **false**;» необходимо. Как уже упоминалось, выражение «**static** 192.168.100.1;» включает в себя «**address** 192.168.100.1;». Таким образом, при поступлении звонка с IP-адреса 192.168.100.1 у SGC будет два абонента: «SomeNetwork» и «SomeNetworkGateway», которым принадлежит этот IP адрес и звонок будет отвергнут, так как гейткиперу необходим только один вызывающий абонент для учёта (billing). Эту неоднозначность и помогает разрешить выражение «**address** 192.168.100.1 **false**;».

Преимущества RAS

Подключение абонентов с использованием протокола RAS имеет ряд преимуществ перед статическим подключением:

- Контроль доступности абонента, если абонент не зарегистрирован (отключен, нет связи, сломался). В этом случае SGC не будет тратить время на попытки вызвать несуществующее устройство, что

позволит ему сразу переключиться на другой альтернативный маршрут и значительно уменьшит время установления соединения и/или позволит дать отрицательный ответ вызывающему абоненту. (см. **registration validity**)

- Точный контроль занимаемой полосы пропускания. Протокол RAS предусматривает процедуру, в соответствии с которой абонент сообщает реальную занимаемую полосу в каналах связи. Это позволяет планировать/контролировать/управлять VOIP сетью. (см. **zone bandwidth**, **link bandwidth**)
- Контроль за установленными/повисшими соединениями. С помощью протокола RAS гейткипер имеет возможность заставить абонента посылать специальные пакеты InfoRequestResponse (IRR) в указанный администратором промежуток времени на протяжении всего звонка. Отсутствие таких пакетов говорит о том, что соединение не активно и гейткипер завершит его. (см. **connection validity**)
- В протоколе RAS предусмотрено использование альтернативных гейткиперов, что позволяет организовать распределение нагрузки на разные гейткиперы и их взаимозаменяемость при неисправностях. (см. **alternate gatekeeper**)
- Для RAS-абонентов можно производить учёт и аутентификацию их регистраций в биллинговой системе, ещё до того как они попытаются позвонить. (см. **radius accounting login**)
- RAS-абоненты при регистрации могут создавать переменные, значение которых можно использовать позже в процессе работы, например это может быть пароль для RADIUS сервера. А с помощью login alias можно создать псевдонимы, по которым этот абонент будет доступен. (см. **make login alias**)

3. Административное присвоение псевдонимов

В SGC имеется возможность присваивать абонентам любые псевдонимы, вне зависимости от того, какие псевдонимы регистрирует абонент. Для этого существует директива '**alias**'. Укажите командой '**login**' в записи пользователя все псевдонимы, которые он регистрирует. Эти псевдонимы будут использованы только для того, чтобы отождествить конкретного абонента с данной пользовательской записью (см. **Подключение абонентов RAS**). Зарегистрированный абонент не будет доступен по данным псевдонимам. Для того, чтобы обозначить псевдонимы, по которым будет доступен данный пользователь, перечислите их, используя директиву '**alias**'.

Пример:

```
user UserOne
{
  login 'login1';
  alias '101';
}

user UserTwo
{
  login 'login2';
  alias '102';
}
```

В данном примере абонент, зарегистрировавшийся под псевдонимом 'login1' будет доступен по псевдониму '101', а абонент, зарегистрировавшийся под псевдонимом 'login2' будет доступен по псевдониму '102'.

Всё вышеперечисленное относится и к статически зарегистрированным абонентам.

Пример:

```
user UserOne
{
  static 192.168.0.10;
  alias '101';
}

user UserTwo
{
  static 192.168.0.11;
  alias '102';
}
```

В случае если абонент должен терминировать звонки на несколько номеров (например четырёх портовый FXS шлюз), то есть возможность использовать несколько выражений **alias** в описании абонента.

Пример

```
user fxs
{
  login 'fxs';
  alias '101';
  alias '102';
  alias '103';
  alias '104';
}
```

SGC будет терминировать звонки на зарегистрированного абонента «fxs» если псевдоним вызываемого абонента (**callee alias**) будет одним из: 101, 102, 103, 104;

Так как в выражении **alias**, псевдоним задаётся «**регулярным выражением**», то существует несколько вариантов описания такого абонента.

Пример

```
user fxs
{
  login 'fxs';
  alias '101|102|103|104';
}
```

или

```
user fxs
{
  login 'fxs';
  alias '10[1-4]';
}
```

Соответственно, используя «**регулярные выражения**» есть возможность описывать целые направления одним шаблоном, это необходимо для описания ваших шлюзов VOIP и провайдеров, терминирующих трафик

Пример

```
user gateway1
{
  static 1.1.1.1;

  alias '7095.+';          // Moskow
  alias '7343212.+' false; // exclede Ekaterinburg 212
  alias '7343.+';          // Ekaterinbug
}

user gateway2
{
  static 1.1.1.2;
```

```
  alias '7343212.+';
}
```

*Прим. Данный пример показывает не только возможность использования «**регулярных выражений**» для описания направлений но и полезную особенность выражения alias <bool>, позволяющую исключить указанное направление.*

Указанный порядок важен, так как SGC останавливается на первом совпавшем alias.

Правильно

Пример

```
alias '7343212.+' false; // exclede Ekaterinburg 212
alias '7343.+'; // Ekaterinbug
```

Не правильно

Пример

```
alias '7343.+'; // Ekaterinbug
alias '7343212.+' false; // exclede Ekaterinburg 212
```

«**alias** '7343212.+' **false**;» не обработается, так как «**alias** '7343.+';» подразумевает «7343212.+».

4. Аутентификация абонентов и учёт звонков на RADIUS

SGC позволяет производить аутентификацию и учёт непосредственно в вашей биллинговой системе, используя протокол RADIUS.

Настройка параметров RADIUS

Параметры RADIUS-сервера необходимо указать в секции **aaa**. Наиболее значимые и достаточные: **address** и **password**. Если ваш сервер RADIUS для работы использует стандартные порты UDP:1812 и UDP:1813, то их в конфигурации можно не указывать, SGC будет использовать их по умолчанию. Также есть возможность указать в какой временной зоне SGC будет указывать время в пакетах accounting, используя выражения: **time zone name** и **time zone offset**. Не маловажным является возможность регистрировать не только состоявшиеся звонки но и те, которые по какой-нибуть причине не смогли установится, за это поведение отвечает выражение **log failed calls**.

Для отладки информации передаваемой между SGC – NAS и сервером RADIUS можно использовать выражение «**log radius messages true**», именно так и получены пакеты приведённые в этой главе.

Пример

```
aaa
{
  address 192.168.1.1;
  authentication port 1812;
  accounting port 1813;
  log failed calls true;
  time zone name 'YEKST';
  time zone offset local;
}
```

Далее можно индивидуально для каждого абонента указывать, что для него нужно делать. Отдельно настраивается аутентификация (**radius authentication**) и учёт (**radius accounting**).

Аутентификация - authentication

Существует несколько режимов аутентификации абонентов: **Регистрация**, **Инициирование звонка**, **Ответ на звонок**.

Регистрация

Только для RAS абонентов, отправляется запрос Access-Request атрибут Service-Type имеет значение «Login».

Управляется выражением **radius authentication** в режиме login.

Пример

```
user abonent
{
  login 'abonent';
  radius name 'user';
  radius password 'password';
  radius authentication login;
}
```

При получении пакета RegistrationRequest (RRQ) и отождествления псевдонимов регистрации (login alias) с абонентом «user abonent» (см. **Регистрация RAS абонентов**), SGC отправит пакет Access-Request на сервер RADIUS.

Пример

```
RADIUS, send message to 192.168.1.1:1812
{
  Code: Access-Request
  Identifier: 1
  Length : 79
  Authenticator: FC 79 00 00 39 74 00 00 F5 77 00 00 18 0F
  Attributes {
    NAS-IP-Address: 192.168.0.1
    NAS-Identifier: "AGK"
    User-Name: "user"
    User-Password: 29 97 18 85 F6 7B A6 3C B9 CB D3 3C 81
    Service-Type: Login
  }
}
```

И только после получения пакета Access-Accept от сервера RADIUS, система разрешит регистрацию абонента, отослав пакет RegistrationConfirm. В любом другом случае в регистрации будет отказано.

Инициирование звонка

Аутентификация инициирования звонка осуществляется как для абонентов зарегистрированных как по RAS так и статически, отправляется запрос Access-Request атрибут Service-Type имеет значение «Call Check».

Управляется выражением **radius authentication** в режиме «caller».

Пример

```
user abonent
{
  login 'abonent';
  radius name 'user';
  radius password 'password';
  radius authentication caller;
}
```

При получении пакета AdmissionRequest (ARQ) от RAS-абонента или Q931:Setup от статического абонента, SGC отправит Access-Request на сервер RADIUS, система разрешит звонок только при получении

положительного пакета от RADIUS. В ответном пакете Access-Accept, сервер RADIUS может указать максимальную продолжительность звонка в атрибутах Session-Timeout и/или Cisco VSA h323-credit-time, SGC прервёт звонок после истечения указанного таймаута.

Пример

```
RADIUS, send message to 192.168.1.1:1812
{
  Code: Access-Request
  Identifier: 2
  Length : 304
  Authenticator: 8E700000A72C0000803E00003D1F0000
  Attributes {
    NAS-IP-Address: 192.168.0.1
    NAS-Identifier: "AGK"
    User-Name: "user"
    User-Password: 2D221478A5A10B8C71A772A9E13E814F
    Service-Type: Call check
    Calling-StationId: "abonent"
    Called-StationId: "1010"
    Vendor-Specific 9 Cisco: 26 "h323-call-origin=answer"
    Vendor-Specific 9 Cisco: 27 "h323-call-type=VoIP"
    Vendor-Specific 9 Cisco: 24 "h323-conf-id=C0F754BE..."
    Vendor-Specific 9 Cisco: 23 "h323-remote-
address=192.168.1.10"
    Vendor-Specific 9 Cisco: 25 "h323-setup-
time=07:33:00.364 UTC Mon May 31 2004"
  }
}

RADIUS, got message from 192.168.1.1:1812
{
  Code: Access-Accept
  Identifier: 2
  Length : 108
  Authenticator: 9D697A885FC619262B02E05108878027
  Attributes {
    Vendor-Specific 9 Cisco: 102 "h323-credit-time=3600"
    Vendor-Specific 9 Cisco: 101 "h323-credit-
amount=400.00"
    Vendor-Specific 9 Cisco: 103 "h323-return-code=0"
  }
}
```

Ответ на звонок

SGC обладает возможностью запрашивать разрешение у RADIUS не только для абонента который производит звонок, но и для абонентов которые отвечают на звонок, это может быть очень удобно если ваша биллинговая система поддерживает такой режим.

Управляется выражением **radius authentication** в режиме «callee».

Пример

```
user abonent
{
  login 'abonent';
  radius name 'user';
  radius password 'password';
  radius authentication callee;
}
```

Принцип функционирования и атрибуты используемые в пакетах те же самые, что и в случае **инициирования звонка** абонентом с одной лишь разницей Cisco VSA h323-call-origin имеет значение «originate».

Имя пользователя и пароль

Для всех видов аутентификации на сервере RADIUS, необходимо указать имя пользователя и пароль, данные параметры необходимы для правильной работы. В показанных выше примерах параметры задаются с использованием выражений: **radius name**, **radius password**.

Параметрами могут выступать:

- символьные строки
- имена переменных
- H.235 информация от абонента (сохраняется в специальных переменных)

Несколько абонентов

Используя переменные значение которых подставляются из регистрационной информации (псевдонимы регистрации) и/или из информации о звонке (псевдонимы вызываемого и/или вызывающего абонента), можно описать одним выражением «user» сразу несколько абонентов, что безусловно очень удобно и гибко при наличии «продвинутой» биллинговой системы.

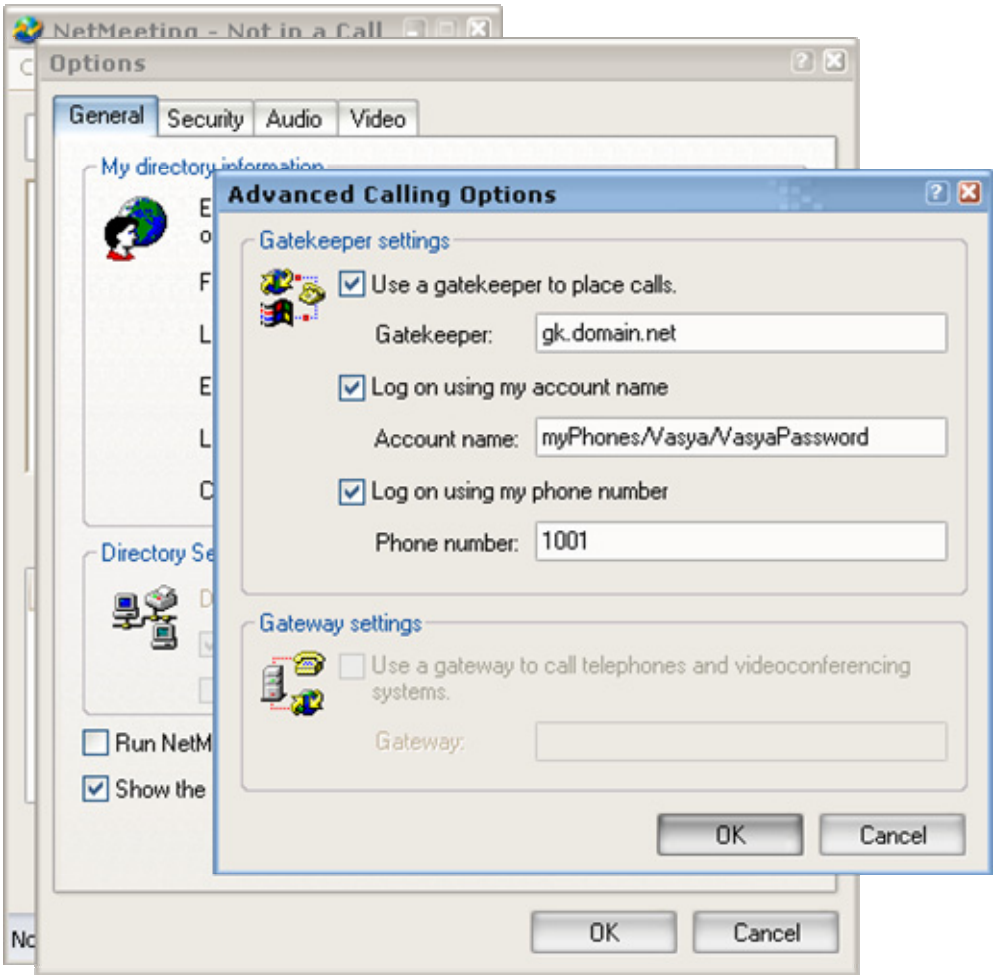
Пример

```
user MyPhones
{
  login 'myPhones.+|\d{4}';
  make login 'myPhones/(.+)/(.+)' set MyName to '\1'
                                   set MyPass to '\2';

  make login '(\d{4})' set MyAlias add '\1';
  alias MyAlias;
  radius name MyName;
  radius password MyPass;
  radius authentication login|caller;
  radius accounting call;
}
```

Приведенный пример наглядно показывает как из «login alias» выбирается информация для использования на RADIUS и определяются псевдонимы для терминации звонков «alias», при этом все процедуры регистрации будет выполнять ваша биллинговая система, останется только правильно настроить абонента.

Пример



Учёт – accounting

Учёт звонков также происходит на сервере RADIUS и контролируется выражением **radius accounting**, задаётся для каждого абонента отдельно.

Каждый звонок для SGC состоит из двух «плеч» - call legs:

- Answer – входящий звонок от вызывающего абонента, в пакетах RADIUS помечается с помощью использования Cisco VSA «h323-call-origin=answer»
- Originate – исходящий звонок к вызываемому абоненту, в пакетах RADIUS помечается с помощью использования Cisco VSA «h323-call-origin=originate»

Соответственно есть возможность включать/отключать посылку accounting информации отдельно для каждого «плеча» звонка указывая режимы radius accounting:

- caller – answer call leg
- callee – originate call leg
- call – answer и originate call legs

Пример

```
user callerAbonent
{
  login 'Abonent1';
  radius accounting caller;
}

user calleeAbonent
{
```

```
  login 'Abonent2';
  radius accounting callee;
}
```

В данном примере в случае звонка от «callerAbonent» к «calleeAbonent», SGC от править четыре пакета accounting Start/Stop для каждого «плеча» звонка. Если же звонок пойдет в обратном направлении от «calleeAbonent» к «callerAbonent» на сервер RADIUS вообще не будет передано ни одного пакета. Поэтому для того чтобы учётная информация передавалась в любом случае и в полном объёме, необходимо у обоих абонентов участвующих в соединении указать «**radius accounting call**».

Пример

```
user Abonent1
{
  login 'Abonent1';
  radius accounting call;
}

user Abonent2
{
  login 'Abonent2';
  radius accounting call;
}
```

Удобнее всего указать «**radius accounting call**» в группе everyone, тогда для любого абонента будет вестись учёт, и отпадёт необходимость для каждого абонента описанного в конфигурации указывать, что для него нужно вести учёт.

Пример

```
group everyone
  radius accounting call;

user Abonent1 login 'Abonent1';
user Abonent2 login 'Abonent2';
```

5. Трансляция псевдонимов

Трансляция псевдонимов, одна из важнейших функциональных характеристик SGC, имеет применение в основном для:

- согласования планов нумерации
- блокировка отдельных направлений
- упрощение плана набора, для абонентов
- организации многоканальных номеров
- извлечения полезной информации, например для сервера RADIUS

Варианты трансляции

SGC имеет возможность обрабатывать как псевдонимы вызываемого абонента (**callee alias**) так и псевдонимы вызывающего абонента (**caller alias**). Трансляция может осуществляться как на входе при получении псевдонима от абонента (outgoing) так и на выходе при передачи псевдонимов к абоненту (incoming).

Для описания трансляции псевдонимов используется выражение «**translate ... alias ... to**».

Прим. Направление (direction) указывается от лица абонента

- outgoing – псевдонимы передаются от абонента к SGC;
- incoming – псевдонимы передаются от SGC к абоненту.

согласования планов нумерации

Согласования планов нумерации провайдеров IP-телефонии, к которым подключена ваша VOIP-сеть с внутренним планом нумерации вашей системы и биллинга.

Обычно в системе принят некий общий план набора, т.е. для каждой конечной терминирующей точки, точки за границами системы, существует уникальный псевдоним, подчиняющийся определённым общим правилам его образования. Этим достигается исключение пересечения псевдонимов из разных сетей в данной системе.

Ваш внутренний план нумерации не только позволяет производить определить на каком абоненте необходимо терминировать трафик, но именно псевдонимы во внутреннем плане нумерации будут отдаваться в вашу биллинговую систему.

Допустим вы транзитом пропускаете трафик от провайдера А к провайдеру В и наоборот, а также часть этого трафика для вашего города терминируете на вашем шлюзе и имеется несколько IP-абонентов.

Ваш внутренний план нумерации для России в формате 7<междугородний код><номер>, всего 11 цифр.

Точка терминации	Описание

ProviderA	Абонент - шлюз VOIP «ProviderA», предоставляющий терминацию на Москву, принимает номера в формате (7 цифр телефонного номера в Москве) и вы принимаете от него звонки в формате 8 + (10 цифр <междугородний код><номер телефона>). Требуется трансляция псевдонимов вызываемого абонента как при отправке номера на него (отрезать 7095)
	`translate incoming callee`
	`  alias '7095(\d{7})' to '\1';`
	так и при приёме вместо 8 необходимо добавить 7
	`translate outgoing callee`
	`  alias '8(\d{10})' to '7\1';`

ProviderB	Абонент – шлюз VOIP «ProviderB», представляющий терминацию на всю Россию, в том числе и на Москву, но по большей цене чем «ProviderA». Именно поэтому стоят разные **cost** на это направление.
	Шлюз принимает и передаёт номера в формате 10 цифр <междугородний код> + <номер телефона>, необходимо отрезать 7 от номера передаваемого на этот шлюз,
	`translate incoming callee`
	`  alias '7(\d{10})' to '\1';`
	и добавить 7 при приёме звонка от него.
	`translate outgoing callee`
	`  alias '(\d{10})' to '7\1';`

Gateway	Это ваш собственный шлюз «Gateway», терминирующий звонки на ваш город (для примера Екатеринбург), здесь всё очень похоже на ProviderA с той лишь разницей, что международный код 343.
	`translate outgoing callee`
	`  alias '8(\d{10})' to '7\1';`
	`translate incoming callee`
	`  alias '7343(\d{7})' to '\1';`

AbonentN№	IP-телефоны/шлюзы ваших пользователей, для них доступен набор номера в форматах:

- 7 цифр номера Екатеринбурга
- 10 цифр междугороднего номера
- 8 + (10 цифр междугороднего номера)

```
translate outgoing callee
  alias '(\d{7})' to '7343\1';
translate outgoing callee
  alias '(\d{10})' to '7\1';
translate outgoing callee
  alias '8(\d{10})' to '7\1';
```

На абонента номер будет приходить в формате 7 цифр, что позволит на конечном устройстве указывать семизначный номер на порту.

```
translate incoming callee
  alias '7343(\d{7})' to '\1';
```

Чтобы не описывать для каждого абонента однотипные трансляции псевдонимов мы используем группу «Ektb».

Пример

```
// --- Учёт для всех абонентов ---------------------------
group everyone
{
  radius accounting call;
  dial Self deny; // Запретить абонентам звонить самим себе
  dial allow; // Разрешить использовать все другие
 }

// --- Терминация на Москву -------------------------------
user ProviderA
{
  static 192.168.10.1;

  alias '7095\d{7}' cost 10;
  translate outgoing callee alias '8(\d{10})' to '7\1';
  translate incoming callee alias '7095(\d{7})' to '\1';
}

// --- Терминация на Россию -------------------------------
user ProviderB
{
  static 192.168.20.1;

  alias '7343\d{7}' false; // на 343 только «Gateway»
  alias '7095\d{7}' cost 100;
  alias '7\d{10}';
  translate outgoing callee alias '(\d{10})' to '7\1';
  translate incoming callee alias '7(\d{10})' to '\1';
}

// --- Локальный шлюз в Екатеринбург ----------------------
user Gateway
{
  login 'myGateway';


  alias '7343\d{7}';
  translate outgoing callee alias '(\d{7})' to '7343\1';
  translate outgoing callee alias '8(\d{10})' to '7\1';
  translate incoming callee alias '7343(\d{7})' to '\1';
}

// --- Трансляции для Локалных абонентов ------------------
group Ektb
{
```

```
  translate outgoing callee alias '(\d{7})' to '7343\1';
  translate outgoing callee alias '(\d{10})' to '7\1';
  translate outgoing callee alias '8(\d{10})' to '7\1';

  translate incoming callee alias '7343(\d{7})' to '\1';
}


// --- Абоненты ----------------------------------------
user Abonent1
{
  login 'abonent1';
  alias '73431111111';
  group member Ektb;
}

user Abonent2_2FXS
{
  login 'abonent2|222222[1-2]';
  alias '7343222222[1-2]'; // local:2222221, 2222222
  dial Self allow; // Разрешить звонки, с порта на порт
  group member Ektb;
}
```

блокировка отдельных направлений

Трансляцию псевдонимов можно также использовать для блокировки отдельных номеров и/или направлений, если транслировать псевдоним в пустую строку, то звонок по такому «пустому псевдониму» не куда не пойдёт.

Например у вас есть абонент которому можно совершать только локальные звонки и звонки в Москву, мы будем транслировать все другие псевдонимы для блокировки данных направлений. Конечно можно было бы воспользоваться выражением «**dial** ProviderB **deny**;», но в этом случае для этого абонента ушла бы возможность звонит на Москву через «ProviderB», что не приемлемо.

Пример

```
user Abonent1
{
  login 'abonent1';
  alias '73431111111';
  translate outgoing callee alias '\d{7}' to '7343\0';
  translate outgoing callee alias '095\d{7}' to '7\0';
  translate outgoing callee alias '8095\d{7}' to '7095\0';
  translate outgoing callee alias '.+' to '';
  group member Ektb;
}
```

Прим. *Пришлось повторить принятую в группе «Ektb» трансляцию принимаемых от абонента псевдонимов, так как SGC останавливается на первом совпавшем «регулярном выражении», но мы смогли унаследовать из группы трансляцию псевдонимов передаваемых абоненту.*

В случае если устройство имеет более одного порта, например «Abonent2_2FXS», есть уникальная возможность блокировать направление только для одного порта и/или разные направления для разных портов. Используя псевдонимы вызывающего абонента, при звонке с первого порта это «2222221», со второго «2222222». Ограничим второй порт.

Пример

```
user Abonent2_2FXS
{
```

```
  login 'abonent2|222222[1-2]';
  alias '7343222222[1-2]'; // local:2222221, 2222222

  // Подготовим переменные для первого порта
  make outgoing caller alias '2222221'
    set LocalNumber to '7343\\1'
    set MSKNumber to '7\\1'
    set Russia to '7\\1';

  // Подготовим переменные для второго порта
  make outgoing caller alias '2222222'
    set LocalNumber to '7343\\1'
    set MSKNumber to '7\\1'
    set Russia to '';

  translate outgoing callee
    alias '(\d{7})' to LocalNumber;
  translate outgoing callee
    alias '(095\d{7})' to MSKNumber;
  translate outgoing callee
    alias '810(.+)' to Russia;

  dial Self allow; // Разрешить звонки, с порта на порт
  group member Ektb;
}
```

организации многоканальных номеров

Например не смотря на то что устройство ожидает получения конкретного номера для того чтобы отправить звонок на определённый порт, вы через трансляцию номеров можете легко транслировать его в ожидаемый для абонента.

Пример

```
user abonent1
{
  login 'ab1';
  alias '1001'; // ожидаемый псевдоним
  alias '000'; // общий с абонентом abonent2

  translate incoming callee alias '.+' to '1001';
}

user abonent2
{
  login 'ab2';
  alias '1002'; // ожидаемый псевдоним
  alias '000'; // общий с абонентом abonent1

  translate incoming callee alias '.+' to '1002';
}
```

извлечение полезной информации

Информацию передаваемую абонентом в псевдонимах вызываемого (callee alias) и/или вызывающего абонента (caller alias), можно использовать для предоставления в RADIUS сервер например для карточной платформы.

Примем, что абонент «gw1» должен перед номером на который он хочет позвонить добавить три цифры номера карты и затем ещё три цифры PIN кода а затем собственно номер по которому он желает позвонить, это позволит вам организовать карточную платформу на шлюзах без поддержки IVR.

Пример

```
user gw1
{
  static 192.168.0.2;

  make outgoing callee alias '(\d{3})(\d{3})(.+)'
    set Card to '\1'
    set PIN to '\2'
    set CalleeAlias to '\3';

  translate outgoing callee alias '.+' to CalleeAlias;

  radius name Card;
  radius password PIN;
  radius authentication caller;
}
```

6. Проксирование трафика

Гейткипер имеет возможность проксирования сигнального трафика и трафика медиа данных. Впрочем, проксирование может быть полностью отключено. По умолчанию гейткипер проксирует только сигнальный трафик. Уровень проксирования задаётся пользователю выражением «**proxy level**». Для соединения будет выбран наивысший из двух уровней, вызывающего и вызываемого абонентов.

Если стоит задача точного учёта времени соединения, то не рекомендуется использовать гейткипер в режиме без проксирования, так как данные о времени соединения будут не точными. Для точного определения времени соединения достаточно режима проксирования сигнализации.

Полное проксирование «proxy level full», позволит вам не только контролировать сигнализацию проходящую через гейткипер, но и скрыть «обезличить» трафик уходящий на вашего провайдера, а также обеспечить безопасность вашей сети и вообще и использовать «серые» IP-адреса для своих абонентов.

Вообще полное проксирование можно включить только для связи с внешним миром, что оптимально позволит совместить безопасность, производительность и точность учёта.

Пример

```
system identifier 'SGC', address <real IP>;

group external
  proxy level full;

user ExternalGW
{
  static <real IP>;
  group member external;
}

user abonent1
{
  static 192.168.0.1;
}

user abonent1
{
  static 192.168.0.2;
}
```

В привёденном примере локальные абоненты имеют «серые» IP-адреса, и не имеют прямого доступа к реальным адресам (кроме одного адреса SGC), голосовой трафик между абонентами будет идти напрямую, но в случае связи с «ExternalGW» включится полное проксирование.

7. Выбор точки терминации

В системе может существовать несколько абонентов с одинаковыми псевдонимами. В данном случае все эти абоненты поступают в процедуру балансировки для того, чтобы определить единственного фактически терминирующего абонента.

Пример

```
user gw1
{
  static 192.168.0.10;
  alias '7095.+';
  alias '7343.+';
}

user gw2
{
  static 192.168.0.11;
  alias '7095.+';
  alias '790284.+';
}
```

В данном примере при звонке по номеру начинающемуся с «7095», у SGC есть две альтернативы куда он может терминировать звонок: «gw1», «gw2», выбор точки терминации будет осуществлён в порядке описанном в **процедуре балансировки**.

Балансировка

Абоненты сбалансируются по следующей схеме:

- Выигрывает абонент с меньшей ценой.
- Если цены равны, выигрывает абонент с большей доступной пропускной способностью.
- Если пропускная способность одна и та же для обоих абонентов, выигрывает абонент с меньшей загрузкой.
- Если загрузка на абонентах одинакова, балансировку выигрывает первый случайный.

Прим. Загрузка абонента вычисляется по формуле Load = Current Connections Count+1/Max Connections Count.

«Max Connections Count» задаётся с помощью выражения «**connections**» в описании абонента.

Цена может быть назначена как на абонента целиком выражением «**cost**» так и на каждый «**alias**» отдельно.

Пример

```
user abonent1
{
  alias '7095.+' cost 1;
  alias '7343.+';
  alias '1000';
  cost 10;
}

user abonent2
{
  alias '7095.+';
```

```
  alias '7343.+' cost 1;
  alias '1000';
  cost 100;
}
```

В случае поступления звонка с номером начинающимся на «7095» звонок уйдёт на «abonent1», если номер начинается на «7343» то на «abonent2» в случае если псевдоним будет «1000», то звонок уйдёт на «abonent1».

Доступная полоса пропускания определяется как разница между доступной полосой пропускания, задаваемой выражением «**bandwidth**» в «**zone**» и «**link**» между зонами в которых расположены абоненты и фактически занятой полосой всеми соединениями проходящими через эти зоны. Полоса пропускания занимаемая одним соединением для абонентов RAS берётся из информации от самих абонентов в пакетах RAS: AdmissionRequest (ARQ), BandwidthRequest (BRQ) установления соединения (поле bandwidth), для статически зарегистрированных абонентов полоса занимаемая на одно соединение может быть определена с помощью выражения «**bandwidth**» в описании абонента «**user**».

Пример

```
zone A bandwidth 1G;
zone B bandwidth 500K;


user A1
{
  location A;
  login 'AUser';
}

user A2
{
  location A;
  static 192.168.0.2;
  bandwidth 16K;
  alias '1000';
}

user B1
{
  location B;
  static 192.168.1.1;
  alias '1000';
}
```

При звонке абонента «A1» по номеру «1000» скорее всего выберется абонент «A2» чем абонент «B1», так как свободной полосы пропускания в зоне B значительно меньше, хотя всё зависит от текущей загрузки – фактически занятой полосы пропускания. Фактически занятую полосу пропускания и полосу пропускания занимаемую одним соединением можно наблюдать в системном журнале и/или в Мониторе.

8. Хантинг

Для вызывающего абонента может быть включен последовательный перебор вызываемых абонентов. Перебор будет происходить в порядке, установленном **процедурой балансировки** и до тех пор пока не произойдёт удачное соединение с каким либо абонентом, либо абонентов для перебора больше не останется. Уровень перебора устанавливается командой «**hunt level**».

Уровни перебора

В SGC реализовано несколько режимов перебора альтернативных абонентов:

- **none** - В этом режиме перебор производиться не будет. Если по каким либо причинам (удалённый абонент занят или удалённое оборудование неисправно, и.т.д.) разговор состояться не может, вызывающий абонент не будет переключен на другую альтернативу и звонящий услышит сигнал ошибки.
- **neutral** - Вызывающий абонент будет переключен на альтернативный маршрут в случае возникновения ошибки во время соединения. Однако, если к этому моменту каналы данных уже установлены, переключение произведено не будет. Гейткипер не будет пытаться отсрочить образование каналов данных, поэтому данный режим назван "нейтральным".
- **soft** - В этом режиме гейткипер будет пытаться отсрочить создание каналов данных перемещая информацию FastStart и H245Address в сообщение Alerting. Что позволит абонентам услышать сигнал КПВ (RingBack tone) если он передаётся непосредственно в голосовом канале.
- **hard** - В этом режиме гейткипер будет пытаться отсрочить создание каналов данных перемещая информацию FastStart и H245Address в сообщение Connect.

Прим. Заметьте, что в любом режиме абоненты всё же имеют возможность открыть каналы данных используя H.245 tunneling, тем самым делая дальнейший перебор невозможным.

Приведём как можно эффективно использовать перебор на примере Биржи VOIP «Тарио». Специфика обработки звонков биржей предполагает совершить несколько попыток соединения на Биржу VOIP для удачной терминации звонка. Как описано на их сайте.

Пример

```
group voipexch_gr
{
  proxy level full;

  address  212.53.35.0-255 false;
  static 212.53.35.34;

  h245 tunneling true;
  fast start true;

  // MSK
  alias "7095.+";
  alias "8095.+";
  alias "7499.+";
  alias "8499.+";

  // Smolensk
  alias "70812.+";
  alias "80812.+";

  translate incoming callee alias "710(.+)" to "\1";
  translate incoming callee alias "810(.+)" to "\1";
  translate incoming callee alias "8(.+)" to "7\1";
}

// все входящие от тарио
user voipexch
  address 212.53.35.0-255 true,
  group member voipexch_gr;
```

```
user voipexch2 group member voipexch_gr;
user voipexch3 group member voipexch_gr;
user voipexch4 group member voipexch_gr;
user voipexch5 group member voipexch_gr;
user voipexch6 group member voipexch_gr;
```

В данном примере будет сделано до 6 попыток терминировать звонок на «**static** 212.53.35.34».

9. Объединение гейткиперов

Несколько гейткиперов могут быть объединены в сеть. В ниже приведённом примере гейткиперы зарегистрируются друг у друга и будут посылать запросы на разрешение псевдонимов. При удачном разрешении псевдонима звонок будет маршрутизирован на удалённый гейткипер.

Есть несколько вариантов подключения SGC к другим гейткиперам, равно как и несколько способов регистрации других гейткиперов на SGC, от способа регистрации зависит и поведение. Для описания всевозможных способов подключения используется выражение «**gatekeeper**».

Выражение «**gatekeeper**», включает в себя полную функциональность и смысл директивы «**user**», что просто расширяет возможности SGC по работе с H.323 Терминалами типа Gatekeeper.

Варианты подключения

Существует два варианта подключения

- С регистрацией по RAS
- Без регистрации по RAS
- Статический абонент

Регистрация по RAS

Используя этот способ регистрации, вы указываете SGC работать по полной схеме протокола RAS, поведение SGC со стороны удалённого гейткипера будет выглядеть как поведение обычного RAS абонента:

- Регистрация, SGC при подключении будет отсылать пакет RegistrationRequest, в случае успешной регистрации поддерживается используется значение полей: TimeToLive (для подтверждения регистрации) и AlternateGatekeeper
- Для звонка SGC, сначала будет посылать пакет LocationRequest и в случае подтверждения отправит запрос AdmissionRequest после получения AdmissionConfirm, SGC будет следовать всем инструкциям от гейткипера: Резервирование полосы пакеты: BandwidthRequest и подтверждения активного соединения InfoRequestResponse.
- По завершению звонка отсылается пакет DisengageRequest
- При остановке, произойдёт отключение SGC от гейткипера используя пакет UnregistrationRequest.

Такой способ подключения даёт возможность использовать аутентификацию с паролем по протоколу H.235.

Пример

```
gatekeeper RemoteGK
{
  identifier 'remote@domain.net';
  ras address 192.168.1.1;
  register alias 'SGC';

  static;
```

```
  alias '.+';
}
```

Или тоже самое, но с использованием H.235, указанием периода подтверждения регистрации. Также в примере разрешено использовать список альтернативных гейткиперов передаваемых с «RemoteGK»

Пример

```
system identifier 'SGC', address 192.168.0.1;

gatekeeper RemoteGK
{
  identifier 'remote@domain.net';
  ras address 192.168.1.1;
  register alias 'SGC';
  regitsration period 120;
  link H235 name 'SGC';
  link H235 password 'xxx';
  link H235 authentication true;
  link alternate gatekeeper true;

  static;
  alias '.+';
}
```

Без регистрации по RAS

В случае использования удаленного гейткипера без регистрации на нём по протоколу RAS, достаточно указать адрес RAS и идентификатор.

В этом варианте SGC будет регистрироваться, а для звонка сперва уйдет пакет LocationRequest и в случае подтверждения сразу Q.931:Setup. Такой способ подтверждения всё равно имеет преимущества перед описанием удалённого гейткипера как простого статического «user», так как при наличии нескольких альтернатив пакеты LocationRequest отсылаются одновременно на нескольким гейткиперам и пир получении первого положительного ответа сразу отправиться Setup, что значительно быстрее чем перебором по очереди отправлять Setup на различных (может быть мёртвых абонентов).

Пример

```
system identifier 'SGC', address 192.168.0.1;

gatekeeper RemoteGK
{
  identifier 'remote@domain.net';
  ras address 192.168.1.1;

  static;
  alias '.+';
}
```

Статический абонент

Такой способ описания мы неоднократно описывали в данном руководстве, удаленный гейткипер описывается как обычный статический абонент и общение с ним ничем не отличается. Конечно удалённый гейткипер должен поддерживать статических абонентов.

Пример

```
system identifier 'SGC', address 192.168.0.1;

user RemoteGK
{
  static 192.168.1.1;
```

```
  alias '7095.+';
}
```

Варианты регистрации гейткипера

Уже упоминалось что выражение «**gatekeeper**», является расширением «**user**», поэтому для регистрации гейткипера на SGC у вас есть все те же возможности что и для «рядового» абонента, все варианты регистрации приведены в разделе «**Методы подключения абонентов**». Приведу только примеры для каждого способа.

RAS

Данный способ описания соответствует привёдённой **регистрации по RAS** с H.235 аутентификацией.

Пример

```
system
  identifier 'remote@domain.net',
  address 192.168.1.1;

gatekeeper SGC
{
  login 'SGC';
  registration validity 120;
  connection validity 60;

  h235 name 'SGC';
  h235 password 'xxx';
  h235 authentication true;
}
```

Без регистрации, статическое

Соответствует приведенной конфигурации «**Без регистрации по RAS**»

Пример

```
system
  identifier 'remote@domain.net',
  address 192.168.1.1;

gatekeeper SGC
{
  ras address 192.168.0.1;
  static 192.168.0.1;
}
```

Обычный статический абонент

Данный способ соответствует подключению «**статический абонент**».

Пример

```
system
  identifier 'remote@domain.net',
  address 192.168.1.1;

user SGC
{
  static 192.168.0.1
}
```

Регистрации двух SGC друг на друге

В данном разделе подведём итог информации о связи гейткиперов друг с другом, в примере будет приведена полная конфигурация двух SGC с их

абонентами, взаимная регистрация будет осуществлена по протоколу RAS и аутентификацией по H.235, один SGC будет обслуживать направление на Москву и IP-телефоны в офисе, второй будет находится в Екатеринбурге и предоставлять доступ к IP-абонентам филиала и выход в город

Конфигурация первого SGC, Москва

```
system
{
  identifier 'MSK';
  address 192.168.0.1;
}

gatekeeper EKTB
{
  identifier 'EKTB';
  ras address 192.168.1.1;
  register alias 'MSK';

  link H235 name 'MSK';
  link H235 password 'MSK';
  link H235 authentication true;

  login 'EKTB';
  registration validity 120;
  connection validity 60;

  h235 name 'EKTB';
  h235 password 'EKTB';
  h235 authentication true;

  alias '.+';
  dial self deny;
  dial yes;
}

user moskow
{
  static 192.168.0.2;
  alias '7095.+';
}

user abonent1
{
  login 'abonent1';
  alias '1001';
}

user abonent2
{
  login 'abonent2';
  alias '1002
}
```

Конфигурация второго SGC, филиал в Екатеринбурге

```
system
{
  identifier 'EKTB';
  address 192.168.1.1;
}

gatekeeper MSK
{
  identifier 'MSK';
```

```
  ras address 192.168.0.1;
  register alias 'EKTB';

  link H235 name 'EKTB';
  link H235 password 'EKTB';
  link H235 authentication true;

  login 'MSK';
  registration validity 120;
  connection validity 60;

  h235 name 'MSK';
  h235 password 'MSK';
  h235 authentication true;

  alias '.+';
  dial self deny;
  dial yes;
}

user ekaterinburg
{
  static 192.168.0.2;
  alias '7095.+';
}

user abonent1
{
  login 'abonent1';
  alias '1003
}

user abonent2
{
  login 'abonent2';
  alias '1004;
}
```

V.Синтаксис файла конфигурации

Конфигурационный файл состоит из отдельных выражений. В положении символов и использовании знака «пробел» не существует каких либо ограничений. Текст, помеченный как комментарий, не интерпретируется как полезная информация.

1. Комментарии

Существует два типа комментариев - блочные и строковые. Блочный комментарий начинается с пары символов '**/***' и заканчивается парой '***/**'. Строчный комментарий начинается с '**//**' и заканчивается с текущей строкой.

Пример

```
/* Это комментарий */
// Это тоже комментарий
```

2. Выражения

Каждое выражение начинается зарезервированным словом, которое указывает на тип выражения. Существует два типа выражений: простые и сложные. Простые выражения осуществляют контроль над одним свойством.

Пример

```
login 'copah';
```

Сложные выражения похожи на простые. Они включают в себя одно или несколько простых выражений, заключенных в фигурные скобки. Сложные выражения имеют простую форму при которой простые составляющие разделяются запятой и следуют сразу после объявления сложного выражения.

Пример

```
user John
{
  location LAN;
  login "John";
  alias "John";
}

user Anyone
  group member Basic,
  login '(.+)' set UN to '\1',alias UN;
```

Порядок, в котором представляются выражения, имеет важное значение. Если существует более одного выражения определенного типа и все они контролируют одно и то же свойство, то только первое активное выражение будет интерпретировано как действительное. Некоторые выражения могут быть ограничены по действию во времени. Такие выражения имеют факультативное ключевое слово '**at**' (См. подвыражение '**at**').

Пример

```
group Group1
{
  proxy level none;
  registration validity 70;
}

user User1
{
  registration validity unlimited;
  group member Group1;
```

```
  proxy level signalling;
  connections 4 at 9:00-18:00;
  connections 2 at 13:00-14:00;
  connections 1;
}
```

В этом примере время регистрации пользователя не будет ограничено потому, что первое выражение соответствующего типа объявляет именно такое состояние. Ограничение регистрации, описанное в группе, не имеет силы. И наоборот, '**proxy level** ' пользователя не имеет эффекта потому, что объявление группы аннулирует это объявление.

С 9:00 до 18:00 абонент будет иметь возможность произвести до 4-х соединений, и только одно соединение все остальное время. Второе правило (connections) не имеет эффекта потому, что время его действия перекрывается первым выражением, имеющим более высокий приоритет.

3. Ключевые (зарезервированные) слова

Ключевые слова регистро-независимые. Администратор может использовать любой регистр в ключевых словах и даже смешивать их. Например, '**file**' и '**File**' - это одно и то же ключевое слово. Большинство ключевых слов используется для указания типа выражения и его факультативов. Эта тема будет рассмотрена позже. Другие ключевые слова используются для указания состояния свойств.

Пример

Boolean: **true**, **false**
Numeric: **unlimited**

4. Типы

string

Строка. Текст, заключенный в апострофы или кавычки. Везде, где применимы переменные, параметр типа string может быть представлен переменной.

regexp

Строка, в которой допустимо использование регулярных выражений. Везде, где применимы переменные, параметр типа regexp может быть представлен переменной.

uint

Беззнаковое целое число или ключевое слово '**unlimited**' для определения бесконечно большой характеристики.

bool

Булево значение. '**true**' или '**false**' ключевое слово.

identifier

Идентификатор. Текст, начинающийся с буквы без пробелов. Идентификатор используется внутри конфигурационного файла для обозначения различных объектов, таких как: зоны, соединения, пользователи, переменные, и.т.д.

ip address

Числовой IP адрес или Интернет имя заключенное в апострофы или кавычки.

Пример

```
123.48.132.56:12345
'canopus.aqua.comptek.ru:12345'
```

address range

Единичный IP-адрес или два адреса, формирующие диапазон.

Пример

Запись	Описание
19.38.12.45	Единичный IP адрес.
19.38.12.45-19.38.12.49	диапазон.
19.38.12.45-49	упрощённый диапазон.

direction

Зарезервированное слово '**incoming**' или '**outgoing**' или пусто (если разрешено конструкцией) для обоих направлений.

side

Зарезервированное слово '**caller**' или '**callee**' или пусто (если разрешено конструкцией) для обеих сторон.

bandwidth

Полоса пропускания в битах в секунду. Должна быть кратной 100. Возможно использование модификаторов '**K**', '**M**', '**G**' для записи полосы в килобитах, мегабитах и гигабитах соответственно. Возможные варианты: 128K, 6.4K, 10M, 0.5M

time

Дата и время, записанные в форме: Mon 31/12/2000 23:59:59. В случае групповых выражений, любая из отдельных величин может быть заменена на символ '**.**'. Некоторые поля могут быть опущены для того, чтобы укоротить запись. Возможные варианты:

- Mon 1/12/1999 12:00:00
- Mon 1/12/1999 12:00
- 1/12/1999 12:00:00
- 1/12/1999 12:00
- 12:00:00
- 12:00
- Mon

period

Два указателя времени, разделенных знаком '**-**', образующие период.

Пример

Выражение	Описание
Mon-Wed	с Понедельника по Среду еженедельно.
12:00-17:59	с 12:00:00 до 17:59:59 ежедневно.
1/./. 00:00-10/./. 23:59	с 1 дня 00:00:00 до 10 дня 23:59:59 ежемесячно.
./1/. 00:00-./2/. 23:59	с 1 Января по 31 Февраля ежегодно.

5. Основные опции

system { ... }

Сложное выражение, содержащее основные **опции системы**.

monitor { ... }

Выражение контроля опций монитора. Смотри раздел **опции монитора**.

log { ... }

Сложное выражение, содержащее **опции контроля за логом**.

aaa { ... }

Содержит информацию о RADUS-сервере и **AAA-опции**.

zone <*identifier*> { ... }

Объявление информационной зоны. Обычно это сегмент сети, который несет поток данных и независим от других зон по его пропускной способности. Используется для ограничения телефонного трафика через определенный участок сети и других QoS. Гейткипер не допустит образования соединений через зону с исчерпанной пропускной способностью. См. раздел "**Опции зоны и соединения**" с описанием выражений, которые могут быть включены в данное сложное выражение.

link [*identifier*] between <*identifier zone*> and <*identifier zone*> { ... }

Это выражение имеет ту же функциональность, что и '**zone**', с дополнительной возможностью соединить две зоны для повторения топологии реальной сети. См. раздел "**Опции зоны и соединения**" с описанием выражений, которые могут быть включены в данное сложное выражение.

group <*identifier*> { ... }

Сложное выражение, используемое для объявления группы и ее поведения. Это выражение может включать любые выражения из сложных выражений '**user**' и '**gatekeeper**'. См. разделы "**Опции пользователя**" и "**Опции гейткипера**". Вы можете создать группу с зарезервированным именем '**everyone**', пользователи объявленные позже автоматически становятся членами этой группы.

user <*identifier*> { ... }

Это выражение имеет почти тот же смысл, что и '**group**', за одним исключением. Оно объявляет конкретного пользователя, который может быть зарегистрирован в системе. См. раздел "**Опции пользователя**" с описанием выражений, которые могут быть включены в данное сложное выражение.

gatekeeper <*identifier*> { ... }

Служит для объявления пользователя с расширенными возможностями (типа гейткипер). Это выражение может содержать простые выражения пользователя и группы, а так же добавочные, собственные выражения.

Основное отличие между выражениями '**user**' и '**gatekeeper**' состоит в том, что гейткипер не будет сразу производить звонок при совпадении номера.

Звонку будет предшествовать сообщение Location Request (LRQ) к удаленному гейткиперу (или гейткиперам, если более одного подошли под метрику маршрута) для того, чтобы обнаружить конкретного вызываемого абонента. Как абонент, удаленный гейткипер должен зарегистрироваться для того, чтобы показать свое активное состояние. В случае, если удаленный гейткипер не поддерживает регистрацию, следует использовать выражение '**static**'. Для детальной информации по специфическим выражениям См. раздел "**Опции гейткипера**".

6. Опции системы (сложное выражение system)

identifier <*string*>

Идентификатор гейткипера. Используется как основной идентификационный элемент гейткипера. Это имя может быть использовано абонентами в процессах обнаружения гейткипера и регистрации на нём в том случае, если абонент желает зарегистрироваться на гейткипере с определенным именем. В этом случае все запросы обнаружения и регистрации с неподходящим идентификатором гейткипера будут отвергнуты гейткипером. Однако, запросы, не содержащие идентификатор гейткипера, будут обработаны.

address <*ip address*>

Выходной интерфейс, через который должен работать гейткипер. Важно правильно установить этот параметр на машинах, имеющих более одного интерфейса для того, чтобы гейткипер мог использовать правильный выход в сеть H.323. Это правило не актуально на машинах с единственным интерфейсом. Однако, если хост, на котором запущен гейткипер, имеет несколько интерфейсов (например, выполняет роль маршрутизатора), то, в случае, если это выражение будет опущено, в качестве адреса гейткипером будет выбран адрес одного из интерфейсов по случайному закону, что может привести к проблемам.

RAS port <*uint*>
default value: 1719

Номер порта, используемого для RAS коммуникации с гейткипером. Если выражение опущено, будет использован стандартный порт 1719.

call signal port <*uint*>
default value: 1720

Номер порта, который будет использован для сигнального уровня. Сигнальный порт используется гейткипером для вызова другими устройствами, не поддерживающими функции регистрации у гейткипера. Если это выражение опущено, будет использован стандартный порт 1720.

7. Опции монитора (сложное выражение monitor)

port <*uint*>
default value: 2040

Номер порта, используемого монитором.

password <*string*>

Пароль для доступа к монитору. Примечание: Этот пароль используется для шифрования конфигурации во время передачи по сети с использованием монитор протокола. В случае, если пароль не установлен — шифрование производиться не будет.

address <*address range*> [*bool*]
address <*bool*>

Определяет право соединения с сервисом монитора с определённого адреса или набора адресов. Опущенный адрес подразумевает все возможные адреса. Опущенный параметр "bool" подразумевает '**true**'.

8. Опции лога (сложное выражение log)

file <*string*>

Путь к файлу журнала, который будет содержать копию консольного вывода. Идентично ключу командной строки '**consolefile**'.

[*direction*] ras messages <*bool*>

Контролирует печать RAS-сообщений. В соответствии с использованным направлением входящие или исходящие сообщения будут печататься в лог. Опция контролирует все RAS-сообщения если направление отсутствует.

[*direction*] q931 messages <*bool*>

Контролирует печать Q.931-сообщений.

[*direction*] h245 messages <*bool*>

Контролирует печать H.245-сообщений.

[*direction*] radius messages <*bool*>

Контролирует печать RADIUS-сообщений.

messages dump <*bool*>

Печать дампа всех входящих сообщений.

registrations <*bool*>

Печать регистраций.

connection status <*bool*>

Печать информации о этапах развития соединения.

chosen route <*bool*>

Печать выбранного маршрута при разрешении строки набора.

available bandwidth <*bool*>

Печать полосы доступной для звонка.

bandwidth changes <*bool*>

Печать всех изменений используемой полосы.

9. Опции AAA (сложное выражение AAA)

file <*string*>

Путь к файлу, который будет содержать информацию о соединениях, времени их начала и продолжительности (CDR файл). Если путь опущен, то файл будет создан в каталоге с файлом конфигурации. CDR файл не будет создан если опущена опция целиком.

address <*ip address*>

Адрес RADIUS сервера.

authentication port <*uint*>
default value: 1812

Порт идентификации. Будет использоваться стандартный (1812) если опция опущена.

accounting port <*uint*>
default value: 1813

Порт учета. Будет использоваться стандартный (1813) если опция опущена.

password <*string*>

Секрет RADIUS сервера.

log failed calls <*bool*>
default value: false

Регистрировать неудачные звонки в CDR файле и на RADIUS сервере.

time zone name <*string*>
default value: "UTC"

Определяет имя временной зоны используемой в RADIUS пакетах.

time zone offset <*offset*>
default value: GMT

Смещение временной зоны для AAA записей. Допустимые значения: **GMT**, **local**.

10.Опции зоны и соединения (сложные выражения zone & link)

bandwidth <*bandwidth*>
default value: unlimited

Суммарная пропускная способность зоны. Гейткипер не допустит образования соединений через зону с исчерпанной пропускной способностью.

connections <uint>
default value: unlimited

Количество соединений которое может быть обслужено зоной. Гейткипер не допустит образования большего количества соединений для данной зоны.

11.Опции пользователя (сложное выражение user)

proxy level <level>
default value: signalling

Устанавливает указанный уровень прокси. Допустимые значения: '**none**', '**signalling**' и '**full**'.

- **none** - Абонент имеет возможность вызывать и быть вызванным другим абонентом напрямую (используя тип вызова "direct"). Однако, в этом режиме, абонент может запросить тип вызова "routed" (при включённой опции '**proxy level choice**') который соответствует '**signalling**' уровню проксирования. Не рекомендуется использовать эту опцию потому, что наличие звонка может быть определено только с использованием сообщений IRR. Поэтому, информация о продолжительности звонка может быть неадекватной.
- **signalling** - Гейткипер будет проксировать оба ("call signal" и "control") сигнальных уровня. Все вызовы типа "direct" будут ремаршрутизированы на гейткипер и тип вызова будет заменен на "routed".
- **full** - В дополнение к уровню проксирования '**signalling**', гейткипер будет проксировать все RTP каналы несущие полезные данные.

В случае если абоненты участвующие в соединении имеют различные уровни проксирования, будет принят старший. Используя эту особенность вы можете установить уровень проксирования '**signalling**' для абонентов внутри сети, в то время как внешние шлюзы используют уровень '**full**' для того, чтобы иметь возможность обойти firewall.

proxy level choice <bool>
default value: false

Управляет возможностью изменения абонентом установленного уровня проксирования. Стандарт H.323 предполагает, что абонент имеет возможность выбрать "direct" или "gatekeeper routed" режим. Следовательно с включённой опцией '**proxy level choice**' абонент может переключиться с установленного уровня проксирования '**signalling**' или '**full**' в '**none**', или остаться в оригинальном режиме. В случае если абонент сконфигурирован с уровнем проксирования '**none**', он имеет возможность переключиться в '**signalling**' или остаться в '**none**' режиме.

hunt level <level>
default value: neutral

Устанавливает уровень хантинга, если абонент является вызывающей стороной. Допустимые значения: '**none**', '**neutral**', '**soft**', '**hard**'.

- **none** - В этом режиме хантинг производиться не будет. Если по каким либо причинам (удалённый абонент занят или удалённое оборудование неисправно, и.т.д.) разговор состояться не может, вызывающий абонент не будет переключен на другую альтернативу и звонящий услышит сигнал ошибки.
- **neutral** - Вызывающий абонент будет переключен на альтернативный маршрут в случае возникновения ошибки во время соединения. Однако, если к этому моменту каналы данных уже установлены, переключение произведено не будет. Гейткипер не

будет пытаться отсрочить образование каналов данных, поэтому данный режим назван "нейтральным".

- **soft** - В этом режиме гейткипер будет пытаться отсрочить создание каналов данных перемещая информацию FastStart и H245Address в сообщение Alerting.
- **hard** - В этом режиме гейткипер будет пытаться отсрочить создание каналов данных перемещая информацию FastStart и H245Address в сообщение Connect.

Заметьте, что в любом режиме абоненты всё же имеют возможность открыть каналы данных используя H.245 туннелинг, тем самым делая дальнейший хантинг невозможным.

fast start <*bool*> default value: true

Контролирует "fast connect" процедуру для данного абонента. Некоторые абонентские устройства (например Cisco ATA186) могут быть приведены в нерабочее состояние "fast start" последовательностью, вы можете обойти это запретив "fast connect" процедуру для таких абонентов.

h245 tunneling <*bool*> default value: true

Контролирует H.245 туннелинг для данного абонента.

cost <*uint*> [*at*] default value: unlimited

Стоимость маршрута. Если существует более одного абонента с одним и тем же псевдонимом, для звонка будет выбран наименее дорогой. Это выражение может содержать суб-выражение '**at**'.

connections <*uint*> [*at*] default value: unlimited

Объявляет максимальное количество соединений, одновременно открытых к пользователю. Это значение используется в расчете загрузки абонента в процедуре уравновешивания загрузки (load balancing). Если существует более одного абонента с одним и тем же псевдонимом и ценой, будет выбран наименее нагруженный. В любом случае гейткипер не допустит большего количества соединений с пользователем, чем указано в этой команде.

location <*identifier* ***zone***>

Зона, которой принадлежит абонент. Используется для QoS и ограничения трафика. Если выражение не определено, абонент находится вне какой-либо зоны.

bandwidth <*bandwidth*> default value: 128k

Двунаправленная, кратная 100, полоса пропускания для статических абонентов. Например, если ваш абонент использует 64 kbit aLaw кодек для обоих направлений, то полоса должна быть записана как 128k.

login <*regexp*> [*bool*] [*set list*]
login <*bool*> [*set list*]

Псевдоним (alias) с которым будет регистрироваться абонент. Опущенный параметр "regexp" подразумевает все возможные псевдонимы. Опущенный параметр "bool" подразумевает '**true**'. Используя подвыражение '**set**' вы можете извлечь различные части псевдонима.

Пример

```
login 'user(.+)goinin' set UserAlias to '\1';
alias UserAlias;
```

Пользователь, регистрирующийся с псевдонимом 'userJohngoinin' будет зарегистрирован с установками текущего пользователя и будет доступен по псевдониму 'John'.

alias <*regexp*> [*bool*] [cost <*uint*>] [translate to <*regexp*>] [*set list*] [*at*]
alias <*bool*> [cost <*uint*>] [*set list*] [*at*]

Определяет набор действительных псевдонимов, по которым может быть вызван абонент. Параметр "bool" указывает на принадлежность данного псевдонима пользователю. Если параметр "bool" установлен в значение '**false**', данный псевдоним исключается из набора действительных.

Пример

```
alias '81234' false;
alias '8\d{5}' true;
```

В данном примере абонент будет доступен по всем пятизначным номерам начинающимся с цифры 8 кроме номера 81234. Опущенный параметр "bool" подразумевает состояние '**true**'. Опущенный параметр "regexp string" подразумевает все возможные псевдонимы.

Подвыражение '**translate to**' используется для трансляции принятой строки набора при представлении её абоненту, подобно тому как это делается в директиве '**translate alias**'.

Подвыражение '**cost**' позволяет установить цену маршрута на индивидуальный псевдоним. В случае если подвыражение опущено, будет использоваться цена маршрута на абонента.

translate [*direction*] [*side*] alias <*regexp*> to <*regexp*> [*at*]

Трансляция псевдонима для дальнейшей обработки и представления. Параметры "direction" и "side" соответственно контролируют трансляцию "входящих/исходящих" (incoming/outgoing) псевдонимов и псевдонимов "вызывающего/вызываемого" (caller/callee). "Входящий" (incoming) псевдоним будет транслирован только после того как он совпадёт с объявленным псевдонимом абонента. Псевдоним модифицирует только первая совпавшая трансляция. Параметры "direction" и "side" могу быть опущены для того чтобы произвести трансляцию всех типов псевдонимов в опущенном классе. К примеру, если псевдонимы вызывающей и вызываемой стороны относятся к одному плану нумерации, для того чтобы согласоваться с внешним планом нумерации Вы можете записать следующие правила трансляции для шлюза:

```
translate incoming alias '(\d{3})' to '+73432451\1';
translate outgoing alias '+73432451(\d{3})' to '\1';
```

Так же, очень полезно установить исходящий псевдоним конкретному единичному пользователю:

```
translate outgoing caller alias '.*' to '212';
```

в то время как его действительный псевдоним сконфигурирован как

```
alias '212';
```

dial [identifier] <bool> [at]

Определяет право пользователя вызывать другого пользователя или пользователя из группы. Опущенный идентификатор (identifier) подразумевает всех пользователей в системе. По умолчанию пользователь имеет право вызывать любого абонента в системе. Если же пользователь или его группа содержат одно или больше выражений '**dial**' вызовы к пользователям, не объявленным данными выражениями, будут отвергнуты. Вместо имени пользователя вы можете использовать зарезервированное слово '**self**'.

address <address range> [bool]
address <bool>

Определяет право пользователя регистрироваться в систему с определённого адреса или диапазона адресов. Опущенный диапазон адресов подразумевает все возможные значения адресов. Опущенное булево значение подразумевает '**true**'. Однако хотя бы один параметр должен присутствовать. По умолчанию пользователь имеет право регистрироваться с любого адреса, что эквивалентно единственному выражению '**address true**'. Если же пользователь или его группа содержат одно или больше выражений '**address**', попытки регистрации со всех остальных адресов будут отвергнуты. В случае если абонент статический (помечен директивой '**static**'), выражения '**address**' определяют набор допустимых адресов для прямого (call signal) вызова гейткипера. Так же, выражение '**static**' автоматически производит директиву '**address**' с данным адресом и флагом доступа установленным в '**true**'.

static [ip address]
static [bool]

Автоматически зарегистрировать абонента при старте системы с данным сигнальным (call signal) адресом. Предусмотрено для терминалов, которые не могут общаться с гейткипером. Все RAS регистрации на данную учётную запись пользователя будут отклонены. Так же, автоматически производит директиву '**address**' с данным адресом и флагом доступа установленным в '**true**'. При опущенном параметре "ip address" звонки на/от данного абонента невозможны. Однако, для того чтобы разрешить пользователю производить звонки вы можете вставить директиву '**address**'.

Example

```
address 19.38.12.7;
static;
```

В этом примере статический абонент имеет возможность производить звонки но не имеет возможности принимать их.

Example

```
address 19.38.12.7 false;
static 19.38.12.7;
```

В этом примере абонент имеет возможность принимать звонки но не имеет возможности производить звонки.

registration validity <*uint seconds*>
default value: unlimited

Ограничить действительность регистрации абонента до "uint" секунд. Абонент должен поддерживать возможность перерегистрации для того, чтобы поддерживать состояние активности. Если абонент не перерегистрируется в указанный промежуток времени, он будет отсоединён от системы. Используйте ключевое слово '**unlimited**' для того, чтобы отменить ранее объявленный (например, в группе пользователя) период.

connection validity <*uint seconds*>
default value: 60

Ограничить действительность соединения до "uint" секунд. Абонент должен поддерживать активное состояние соединения периодически посылая IRR сообщение. IRR период устанавливается гейткипером и он в три раза меньше чем параметр '**connection validity**'. Гейткипер не будет запрашивать у абонента IRR сообщений если '**connection validity**' установлено в '**unlimited**'. Эта возможность может помочь подключить абонентов не рассылающих IRR сообщения во время звонка, несмотря на то что эта функция объявлена обязательной в H.323 стандарте. Настоятельно не рекомендуется использовать '**connection validity unlimited;**' с '**proxy level none;**' потому, что в этом случае для гейткипера не существует надежного способа узнать что соединение было прервано. Это может вызвать недостоверность данных биллинга и "повисшие" соединения.

max [*direction*] ringback duration <*uint seconds*>
default value: unlimited

Лимит (в секундах) времени ожидания ответа. Будет взято наименьшее из двух (incoming для терминирующей стороны и outgoing для вызывающей) значений.

max [*direction*] connection duration <*uint seconds*>
default value: unlimited

Лимит (в секундах) максимальной продолжительности соединения. Будет взято наименьшее из двух (incoming для терминирующей стороны и outgoing для вызывающей) значений.

radius authentication <*mode*>
default value: none

Устанавливает режим идентификации на RADIUS сервере. Допустимые значения: '**none**', '**login**', '**caller**', '**callee**', '**call**' и '**full**'. Опции могут быть объединены знаком вертикальная черта (|).

- **none** - Идентификация производиться не будет.
- **login** - Гейткипер будет запрашивать разрешение на регистрацию пользователя у RADIUS сервера.
- **caller** - Гейткипер будет запрашивать разрешение на установление соединения в случае если абонент является вызывающей стороной.
- **callee** - Гейткипер будет запрашивать разрешение на установление соединения в случае если абонент является вызываемой стороной.
- call - Идентично 'caller | callee'.
- **full** - Идентично '**login** | **call**'.

Идентификационный запрос на регистрацию имеет поле "Service-Type" установленное в "Login". Идентификационный запрос на установление соединения имеет поле "Service-Type" установленное в "Call Check".

radius accounting <*mode*>
default value: none

Устанавливает режим учета на RADIUS сервере. Допустимые значения: '**none**', '**login**', '**caller**', '**callee**', '**call**' и '**full**'. Опции могут быть объединены знаком вертикальная черта (|).

- **none** - Учет для абонента производиться не будет.
- **login** - Гейткипер будет регистрировать время пребывания пользователя в системе.
- **caller** - Гейткипер будет регистрировать продолжительность звонков в случае если абонент является вызывающей стороной.
- **callee** - Гейткипер будет регистрировать продолжительность звонков в случае если абонент является вызываемой стороной.
- call - Идентично 'caller | callee'.
- **full** - Идентично '**login** | **call**'.

Запрос на учет регистрации имеет поле "Service-Type" установленное в "Login". Запрос на учет соединения не имеет поля "Service-Type".

radius name <*string*>
default value: UserName

Установить имя пользователя для RADIUS операций. Так же можно использовать встроенные переменные UserName H235Name которые содержат, соответственно, имя пользователя из конфигурации и имя пользователя из предоставленной H.235 информации.

radius password <*string*>

Установить пароль пользователя для RADIUS операций. Пароль нулевой длинны подразумевает, что пароль не используется. Это значение по умолчанию. В случае если использовано выражение '**radius password** H235Password', гейткипер извлечёт пароль из доставленного абонентом информационного уровня H.235. Этот режим поддерживает только Cisco Access Token.

h235 authentication <*bool*>

Включить/выключить H.235 авторизацию RAS и Q.931 сообщений. Поддерживаемые методы: CAT, Password with hashing (MD5).

h235 name <*string*>
default value: UserName

Установить имя которое должен представить пользователь для авторизации сообщений.

h235 password <*string*>

Установить пароль который должен представить пользователь для авторизации сообщений.

display <*string*>
default value: UserDisplay

Присваивает заданное значение полю Q.931 display. Оригинальное значение поля display помещается в переменную UserDisplay автоматически. Вы всегда можете восстановить оригинальный режим дав команду '**display** UserDisplay'.

connection dupe <bool> default value: false

Разрешить или запретить пользователю создавать соединения с повторяющимся идентификатором соединения. Обычно эта функция должна быть запрещена. Однако, некоторые абоненты, такие как ремаршрутизирующие автоответчики могут открыть соединение с изменённым адресом номером но с тем же идентификатором соединения. Обычно, такой звонок будет запрещён детектором петель, таким образом, в данном случае, вы должны включить функцию '**connection dupe**' для данного абонента. Используйте эту возможность с особой осторожностью, абонент не должен выступать с одной стороны (например, со стороны вызывающего) дважды в разных соединениях с идентичным идентификатором соединения. Это вызвано тем, что у гейткипера не будет возможности различать такие соединения.

set <identifier:variable> to <string>

Устанавливает данной переменной строковое значение. Как и с остальными выражениями, только первое выражение '**set**' сработает над определённой переменной.

make login <regexp> <set list>

Побудить систему создать переменную по какому-либо псевдониму. Заметьте, что это выражение не влияет на процесс регистрации, оно включается в работу только после того как псевдоним прошёл выражение '**login**'.

Пример

```
group everyone
      make login '.*' set UserAlias to '\0',
      alias UserAlias;

user User1 login '201';
user User2 login '202';
```

В этом примере User1 будет зарегистрирован с только псевдонимом "201". И он будет иметь действительный псевдоним "201".

make [direction] [side] alias <regexp> <set list> [at]

Побудить систему создать переменную по псевдониму звонка. Пропущенные direction и/или side параметры подразумевают все псевдонимы всех типов в опущенном классе.

alternate gatekeeper <ip address> [string] [gatekeeper ...] alternate gatekeeper none

Объявляет список альтернативных гейткиперов для пользователя. Этот список будет предоставлен абоненту в случае если он поддерживает процедуры альтернативного гейткипера обозначая это включением поля SupportsAltGK в RRQ сообщение. Выражение '**alternate gatekeeper none**' выключает эту функцию. Это значение по умолчанию.

12.Опции гейткипера (сложное выражение gatekeeper)

ras address <*ip address*>

Объявляет адрес уровня RAS удаленного гейткипера. Используется совместно с опцией '**static**', в том случае, если наш гейткипер не имеет возможности обнаружить этот адрес, используя другие источники.

identifier <*string*>

Идентификатор удаленного гейткипера. Наш гейткипер будет использовать его во всех запросах, где это возможно. Так же, если выражение '**ras address**' опущено, гейткипер попытается обнаружить удаленный гейткипер при помощи сообщения Gatekeeper Request (GRQ). Примечание: GRQ является широковещательным сообщением и может быть не пропущено через маршрутизатор.

register alias <*string*> [alias ...]
register alias none

Побуждает гейткипер зарегистрироваться на удаленном гейткипере для того, чтобы показать свое активное состояние. Метод регистрации точно такой же, как у абонента. Удалённый гейткипер может интерпретировать наш гейткипер как обычного абонента в том случае, если он не анализирует тип регистрируемого терминала.

link h235 name <*string*>
default value: UserName

Устанавливает H.235 имя для соединения с удалённым гейткипером.

link h235 password <*string*>

Устанавливает H.235 пароль для соединения с удалённым гейткипером.

link h235 authentication <*bool*>
default value: false

Включает/выключает H.235 аутентикацию на соединении с удалённым гейткипером.

registration period <*uint seconds*>

Побуждает гейткипер повторять процедуру регистрации в установленный период ("uint" секунд). Эта функция используется для подтверждения активного состояния.

link alternate gatekeeper <*bool*>
default value: false

Разрешает процедуры альтернативного гейткипера на данном соединении с удалённым гейткипером. При разрешённой опции будет выставлен флаг SupportsAltGK, проанализирован список альтернативных гейткиперов представленный удалённым гейткипером и включены соответствующие процедуры переключения между альтернативными гейткиперами.

13.Подвыражения

at <period>

Устанавливает период действительности выражения.

set <identifier variable> to <string>

Присваивает строковое значение данной переменной.

Пример

```
set UserLogin to 'user(.+)';
login UserLogin set UserAlias to '\1';
alias UserAlias;
```

set <identifier variable> add <string>

Добавляет строку к указанной переменной. Если переменная уже содержит строку не нулевой длинны то строка будет предварена знаком вертикальная черта (|). Эта функция создана специально для обработки нескольких псевдонимов в процессе регистрации абонента. Если в строке, которую содержит переменная, есть скобки, то подстрока будет вставлена перед первой закрывающей скобкой.

Пример

```
set UserAlias to 'common|(2())';
login 'gw5(\d{3})' set UserAlias add '\1';
alias UserAlias;
```

В этом примере, пользователь представленный псевдонимами "gw5100" и "gw5101", будет иметь результирующий псевдоним "common|(2(100|101))".

set list

Список подвыражений '**set** ... **to**' и '**set** ... **add**'.

Пример

```
alias '(\d{3})(\d{3}).*' set Account to '\1'
                         set PIN to '\2';
```

VI.Справочная информация

1. Формат файла CDR

Имя файла CDR указывается в секции **aaa** выражением **file** и имеет следующий формат.

Каждая строка относится к отдельному звонку, информационные поля имеют разделитель «,» ниже в таблице указаны поля в порядке их присутствия в строке файла CDR.

№	Описание	Пример
1	Дата записи в лог, всегда в GMT+0	03/06/2004 12:13:33
2	Идентификатор вызывающего абонента	abonent1
3	Адрес Call Signalling с которого был осуществлен звонок	192.168.0.3:3175
4	Идентификатор вызываемого абонента	abonent2
5	Адрес Call Signalling на который терминировался звонок	192.168.0.4:1720
6	Дата начала разговора, зависит от **aaa time zone offset**	03/06/2004 17:58:34
7	Дата окончания разговора, зависит от **aaa time zone offset**	03/06/2004 18:13:33
8	Продолжительность разговора	0/0:14:59.365
9	Код завершения звонка, ITU-T Q.850	16
10	Псевдонимы вызывающего абонента, разделённые символом «&» если их несколько	1001 & abonent1
11	Псевдонимы вызываемого абонента, разделённые символом «&» если их несколько	73431111111

2. Протокол SGCMonitor

Открывается TCP соединение на порт 2040 и все пакеты передаются используя TPKT обрамление. Используя выражение **log monitor messages yes**; можно наблюдать передаваемые пакеты непосредственно в журнале системы.

```
--
-- Copyright (c) 1999-2003 Aqua Project Group
-- ASN.1 Gatekeeper monitor protocol version 1.00
--


MonitorProto DEFINITIONS AUTOMATIC TAGS ::=
BEGIN


-------------------------------------------------------------
-- Base types
-------------------------------------------------------------

IpAddress ::= SEQUENCE
{
  ip        OCTET STRING( SIZE( 4 ) ),
  port            INTEGER( 0..65535 )
}

Date ::= SEQUENCE
{
  year            INTEGER,
  month           INTEGER( 1..12 ),
  day       INTEGER( 1..31 )
```

```
}

Time ::= SEQUENCE
{
  hour           INTEGER( 0..23 ),
  minute    INTEGER( 0..59 ),
  second    INTEGER( 0..59 )
}

DateTime ::= SEQUENCE
{
  date           Date,
  time           Time
}

MessageIdentifier    ::= INTEGER( 0..65535 )
GatekeeperIdentifier ::= BMPString( SIZE( 1..128 ) )
EndpointIdentifier   ::= BMPString( SIZE( 1..128 ) )
ZoneIdentifier       ::= OCTET STRING( SIZE( 1..128 ) )
ConferenceIdentifier ::= OCTET STRING( SIZE( 16 ) )
Bandwidth   ::= INTEGER( 0..4294967295 ) -- in 100s of its
CallReferenceValue   ::= INTEGER( 0..65535 )

SegmentType ::= ENUMERATED
{
  comprehensive,
  initial,
  intermediate,
  final
}

CompletionCode ::= SEQUENCE
{
  errorText             OCTET STRING OPTIONAL
}

EndpointType ::= CHOICE
{
  gatekeeper           NULL,
  gateway         NULL,
  mcu             NULL,
  terminal        NULL
}

ProxyLevel ::= CHOICE
{
  none                 NULL,
  signalling           NULL,
  full                 NULL
}

ChallengeValue ::= OCTET STRING( SIZE( 8 ) )
ChallengeResponse ::= OCTET STRING( SIZE( 16 ) )

Challenge ::= SEQUENCE
{
  value                ChallengeValue OPTIONAL
}

RequestDataClass ::= ENUMERATED
{
  gatekeeper,
  zone,
  endpoint,
```

```
  connection,
  configuration,
  log,
  ...
}

RequestData ::= SEQUENCE
{
  dataClass       RequestDataClass,
  fetchCurrentData     BOOLEAN,
  notifyChanges         BOOLEAN,
  ...
}

RequestResultError ::= SEQUENCE
{
  code                  INTEGER( 0..127,... ),
  text                  OCTET STRING OPTIONAL
}

RequestResult ::= CHOICE
{
  ok              NULL,
  okDataFollowOn  NULL,
  error                 RequestResultError
}

-------------------------------------------------------
-- Gatekeeper
-------------------------------------------------------

GatekeeperInfo ::= SEQUENCE
{
  identifier            GatekeeperIdentifier,
  version         OCTET STRING,
  osInfo          OCTET STRING,
  rasAddress            IpAddress OPTIONAL,
  csAddress       IpAddress OPTIONAL,
  ...
}

-------------------------------------------------------------
-- Zone
-------------------------------------------------------------

Zone ::= SEQUENCE
{
  identifier            ZoneIdentifier,
  bandwidth       Bandwidth,
  occupiedBandwidth     Bandwidth,
  connections           INTEGER,
  occupiedConnections   INTEGER,
  links                 SEQUENCE OF ZoneIdentifier,
  ...
}

-------------------------------------------------------------
-- Endpoints
-------------------------------------------------------------

Endpoint ::= SEQUENCE
{
  identifier            EndpointIdentifier,
  userName        OCTET STRING,
```

```
  loginAliases SEQUENCE OF BMPString( SIZE( 1..256  ),
  aliases         SEQUENCE OF BMPString( SIZE( 1..256 ) ),
  rasAddress            IpAddress OPTIONAL,
  csAddress       IpAddress,
  endpointType          EndpointType,
  zone                   ZoneIdentifier OPTIONAL,
  static          BOOLEAN,
  ...
}

-------------------------------------------------------------
-- Connections
-------------------------------------------------------------

Connection ::= SEQUENCE
{
  identifier             ConferenceIdentifier,
  callerIdentifier      EndpointIdentifier OPTIONAL,
  callerCSAddress IpAddress OPTIONAL,
  calleeIdentifier      EndpointIdentifier OPTIONAL,
  calleeCSAddress IpAddress OPTIONAL,
  occupiedBandwidth     Bandwidth,
  conversationStart     DateTime OPTIONAL,
  conversationStop      DateTime OPTIONAL,
  zones                 SEQUENCE OF ZoneIdentifier OPTIONAL,
  callerAliases         SEQUENCE OF OCTET STRING OPTIONAL,
  calleeAliases         SEQUENCE OF OCTET STRING OPTIONAL,
  callerDisplay         OCTET STRING OPTIONAL,
  calleeDisplay         OCTET STRING OPTIONAL,
  proxyLevel            ProxyLevel,
  ...
}

-------------------------------------------------------------
-- Main
-------------------------------------------------------------

ServerRequest ::= SEQUENCE
{
  identifier            MessageIdentifier,
  body                  CHOICE
  {
    challenge           Challenge,
    ...
  }
}

ServerResponse ::= SEQUENCE
{
  identifier            MessageIdentifier,
  segmentType           SegmentType DEFAULT comprehensive,
  body                  CHOICE
  {
    requestResult RequestResult,
    gatekeeperInfo      GatekeeperInfo,
    configurationData   OCTET STRING,
    logData       OCTET STRING,
    zones         SEQUENCE OF Zone,
    endpoints           SEQUENCE OF Endpoint,
    connections         SEQUENCE OF Connection,
    ...
  }
}
```

```
ServerNotification ::= CHOICE
{
  gatekeeperInfo  GatekeeperInfo,
  logData         OCTET STRING,
  zonesAdded            SEQUENCE OF Zone,
  zonesChanged          SEQUENCE OF Zone,
  zonesRemoved          SEQUENCE OF ZoneIdentifier,
  endpointsAdded  SEQUENCE OF Endpoint,
  endpointsChanged      SEQUENCE OF Endpoint,
  endpointsRemoved      SEQUENCE OF EndpointIdentifier,
  connectionsAdded      SEQUENCE OF Connection,
  connectionsChanged    SEQUENCE OF Connection,
  connectionsRemoved    SEQUENCE OF ConferenceIdentifier
}

MonitorServerMessage ::= CHOICE
{
  request         ServerRequest,
  response        ServerResponse,
  notification          ServerNotification
}

ClientRequest ::= SEQUENCE
{
  identifier            MessageIdentifier,
  segmentType           SegmentType DEFAULT comprehensive,
  body                  CHOICE
  {
    requestData         RequestData,
    setConfigData OCTET STRING,
    restart       BMPString( SIZE( 1..25 ) ), -- Place "Yes,
please!" here.
    reloadConfiguration BMPString( SIZE( 1..25 ) ), -- Place
"Yes, please!" here.
    removeEndpoints     SEQUENCE OF EndpointIdentifier,
    removeConnections   SEQUENCE OF ConferenceIdentifier,
    ...
  }
}

ClientResponse ::= SEQUENCE
{
  identifier            MessageIdentifier,
  body                  CHOICE
  {
    challengeResponse   ChallengeResponse,
    ...
  }
}

MonitorClientMessage ::= CHOICE
{
  request         ClientRequest,
  response        ClientResponse,
  ...
}

END -- of ASN
```

3. Основные регулярные выражения POSIX

Язык регулярных выражений это условная запись служащая для описания текстовых шаблонов. Обычно регулярные выражения участвуют в

процедуре сравнения их со строками для того чтобы выяснить совпадает ли строка с шаблоном, или для поиска подстроки в строке.

Эта глава описывает POSIX запись регулярных выражений. Это не официальное, не точное определение регулярных выражений POSIX -- это интуитивное и возможно более доступное описание.

Введение в регулярные выражения

В простейшем случае регулярное выражение это простая строка из символов которая должна совпасть однозначным образом. К примеру, шаблон:

```
regexp
```

совпадает со строкой "regexp" и никакими другими.

Некоторые символы в регулярных выражениях имеют специальное значение. Они не сравниваются посимвольно как это было в предыдущем примере, а напротив описывают более общий шаблон. Для примера, символ * используется для указания что предыдущий элемент регулярного выражения может повторяться 0, 1 или больше раз. В шаблоне:

```
smooo*th
```

* указывает что предыдущий символ o может повториться 0 или больше раз. Поэтому шаблон совпадает с:

```
smooth
smoooth
smooooth
smoooooth
...
```

Конечно же вы можете иметь потребность написать шаблон в котором специальный символ * должен совпасть символьно -- т.е. вы не хотите чтобы * указывала на допустимое повторение, но совпадала символьно. Для того чтобы реализовать это требуется выделить специальный символ обратным слешем. Шаблон:

```
smoo\*th
```

совпадает со строкой:

```
smoo*th
```

и ни с какими другими.

Другие параграфы так же описывают различные специальные символы которые могут встречаться в регулярных выражениях.

Символьные регулярные выражения

Символьное регулярное выражение это строка не содержащая специальных символов. Символьное регулярное выражение совпадает только с точно такой же строкой, и никакой другой. К примеру:

```
literally
```

совпадает с

```
literally
```

и больше ни с чем.

Как правило, символ пробела, цифры и буквы не являются специальными символами. Некоторые символы препинания являются специальными, некоторые нет (краткая справка в конце этой главы позволяет удобно определять какие символы являются специальными а какие нет).

Наборы символов

Этот параграф описывает специальные символы . и [.

. совпадает с любым символом исключая символ NULL. К примеру:

```
p.ck
```

совпадает с

```
pick
pack
puck
pbck
pcck
p.ck
...
```

[открывает набор символов. Набор символов схож с . в том что он совпадает не с каким либо конкретным символом а с любым из приведённого набора. [отличается от . тем что при помощи [вы можете объявить совершенно определённый набор символов.

Набор символов может принимать три формы.

В первой форме набор символов выглядит следующим образом:

[<набор-символов>] -- каждый символ из набора.

Во второй форме набор символов обозначен специальным символом отрицания и выглядит:

[^<набор-символов>] -- каждый символ *не* из набора.

<набор-символов> перечисление набора символов. Оно может быть выражено строкой из индивидуальных символов:

```
[aeiou]
```

или как ряд символов:

```
[0-9]
```

Два этих формата могут быть смешаны:

```
[A-za-z0-9_$]
```

Специальные символы (такие как *) не являются специальными внутри набора символов. -, как показано выше, является специальным, за исключением, когда он находится первым в наборе.

Набор из четырех символов:

```
[-+*/]
```

Третья форма набора символов использует предопределённые классы символов:

\<класс-символов> -- каждый символ описанный данным классом.

Поддерживаемые классы символов:

Символ	Описание
\w	набор букв и цифр
\d	десятичные цифры
\l	строчные буквы
\s	whitespace characters
\u	прописные буквы

Набор класса может быть инвертирован изменением символа класса из строчного в прописной.

Наборы символов могут быть использованы в любом месте регулярного выражения, там же где используются простые символы.

Подвыражения

Подвыражение это регулярное выражение заключённое в ( ). Подвыражение может быть использовано в любом месте регулярного выражения.

Подвыражения полезны для группировки конструкций регулярного выражения. Для примера, символ повтора, *, обычно применяется только для предыдущего символа. Вспомним:

```
smooo*th
```

совпадает с

```
smooth
smoooth
...
```

Используя подвыражение мы можем применить * к более длинной строке:

```
banan(an)*a
```

совпадает с

```
banana
bananana
banananana
...
```

Также подвыражения имеют специальное значение в ссылках и подстановках.

Повторяющиеся подвыражения

Оператор повтора *, применяется к предыдущему символу, набору символов, подвыражению или к ссылке. * указывает что предыдущий элемент может совпасть 0 или более раз:

```
bana(na)*
```

совпадает с

```
bana
banana
bananana
banananana
...
```

+ похож на * за исключением что предыдущий элемент должен совпасть как минимум один раз. И так:

```
bana(na)*
```

совпадает с

```
bana
```

но не совпадает с

```
bana(na)+
```

Оба регулярных выражения совпадают с

```
banana
bananana
banananana
...
```

Таким образом, bana(na)+ является короткой записью для banana(na)*.

Факультативные подвыражения

? указывает на то что предыдущий символ, набор символов или подвыражение факультативно. Оно может совпасть или быть пропущено:

```
CSNY?
```

совпадает с обоими строками

```
CSN
CSNY
```

Количественные подвыражения

Выражение интервала, {m,n} где m и n неотрицательные целые с n >= m, применяется к предыдущему символу, набору символов, подвыражению или ссылке. И указывает на то что предыдущий элемент должен совпасть как минимум m раз но не более n раз.

К примеру:

```
c([ad]){1,4}r
```

совпадает с

```
car
cdr
caar
cdar
...
caaar
cdaar
...
cadddr
cddddr
```

Альтернативные подвыражения

Альтернативные подвыражения имеют форму:

```
regexp-1|regexp-2|regexp-3|...
```

Выражение совпадает по любому regexp-n. Для примера:

```
Crosby, Stills, (and Nash|Nash, and Young)
```

совпадает с

```
Crosby, Stills, and Nash
Crosby, Stills, Nash, and Young
```

Ссылки, извлечения и подстановки

Ссылка записывается в форме \n. Где n десятичное число отличное от нуля. Для того чтобы ссылка была действительной, в регулярном выражении должно быть как минимум n подвыражений заключённых в скобки.

Ссылка посимвольно совпадает с тем что совпало с соответствующим подвыражением. Например,

```
(.*)-\1
```

совпадает

```
go-go
ha-ha
wakka-wakka
...
```

В некоторых случаях подвыражения используются для подстановки используя схему нумерации ссылок. Для примера, команда:

```
translate alias '90244(\d{,5})' to '90248\1'
```

сначала совпадает со строкой:

```
9024412345
```

подвыражение 1 совпадает с "12345". Команда заменяет совпавшую строку на "90248\1". И после подстановки получается следующее:

```
9024812345
```

Краткая справка по синтаксису регулярных выражений

abcd -- совпадает со строкой посимвольно.

. -- совпадает с любым символом кроме символа NULL.

[a-z_?], ^[a-z_?], \d and \D -- наборы символов.

(подвыражение) -- группировка выражения в подвыражение.

Следующие специальные символы и последовательности могут быть применены к символу, набору символов, подвыражению или к ссылке:

Символ	Описание
*	предыдущий элемент повторяется 0 или более раз.
+	предыдущий элемент повторяется 1 или более раз.
?	предыдущий элемент повторяется 0 или 1 раз.
{m,n}	предыдущий элемент повторяется как минимум m но не более n раз.
regexp-1|regexp-2|..	совпадает с любым из regexp-n.

Специальные символы, такие как . или * или ? или + могут быть преобразованы в простые символы добавлением перед ними \.