СИСТЕМА
УПРАВЛЕНИЯ
БАЗАМИ
ДАННЫХ

ЛИНТЕР®

ЛИНТЕР БАСТИОН

ЛИНТЕР СТАНДАРТ

Сетевые средства

НАУЧНО-ПРОИЗВОДСТВЕННОЕ ПРЕДПРИЯТИЕ

РЕЛЭКС

Товарные знаки

РЕЛЭКС™, ЛИНТЕР® являются товарными знаками, принадлежащими АО НПП «Реляционные экспертные системы» (далее по тексту – компания РЕЛЭКС). Прочие названия и обозначения продуктов в документе являются товарными знаками их производителей, продавцов или разработчиков.

Интеллектуальная собственность

Правообладателем продуктов ЛИНТЕР® является компания РЕЛЭКС (1990-2026). Все права защищены.

Данный документ является результатом интеллектуальной деятельности, права на который принадлежат компании РЕЛЭКС.

Все материалы данного документа, а также его части/разделы могут свободно размещаться на любых сетевых ресурсах при условии указания на них источника документа и активных ссылок на сайты компании РЕЛЭКС: relex.ru и linter.ru.

При использовании любого материала из данного документа несетевым/печатным изданием обязательно указание в этом издании источника материала и ссылок на сайты компании РЕЛЭКС: relex.ru и linter.ru.

Цитирование информации из данного документа в средствах массовой информации допускается при обязательном упоминании первоисточника информации и компании РЕЛЭКС.

Любое использование в коммерческих целях информации из данного документа, включая (но не ограничиваясь этим) воспроизведение, передачу, преобразование, сохранение в системе поиска информации, перевод на другой (в том числе компьютерный) язык в какой-либо форме, какими-либо средствами, электронными, механическими, магнитными, оптическими, химическими, ручными или иными, запрещено без предварительного письменного разрешения компании РЕЛЭКС.

О документе

Материал, содержащийся в данном документе, прошел доскональную проверку, но компания РЕЛЭКС не гарантирует, что документ не содержит ошибок и пропусков, поэтому оставляет за собой право в любое время вносить в документ исправления и изменения, пересматривать и обновлять содержащуюся в нем информацию.

Контактные данные

394006, Россия, г. Воронеж, ул. Бахметьева, 2Б.

Тел./факс: (473) 2-711-711, 2-778-333.

e-mail: info@linter.ru.

Техническая поддержка

С целью повышения качества программного продукта ЛИНТЕР и предоставляемых услуг в компании РЕЛЭКС действует автоматизированная система учёта и обработки пользовательских рекламаций. Обо всех обнаруженных недостатках и ошибках в программном продукте и/или документации на него просим сообщать нам в раздел Поддержка на сайте ЛИНТЕР.

Содержание

Предисловие 4
Назначение документа 4
Для кого предназначен документ 4
Необходимые предварительные знания 4
Дополнительные документы 4
Общие сведения 5
Модель сетевой архитектуры СУБД ЛИНТЕР 6
ЛИНТЕР-сервер 6
Клиентское приложение 6
Программные сетевые средства 7
Сетевой драйвер клиента 7
Сетевой драйвер сервера 7
Файл сетевой конфигурации клиента 8
Переменные окружения ОС 8
Протоколы обмена данными 8
Механизм взаимодействия клиент-сервер 9
Сетевая прозрачность 9
Независимость от протоколов 9
Прозрачность местоположения 9
Организация сетевого доступа 10
Операции соединения/отсоединения 10
Операции обмена данными 10
Аварийное прекращение соединения 11
Разъединение по тайм-ауту 11
Варианты сетевых соединений 12
Локальный доступ к БД по умолчанию 12
Локальный доступ через сетевой драйвер клиента 12
Локальный сетевой доступ 13
Удаленный сетевой доступ 14
Установка сетевых средств 15
Файл сетевой конфигурации 16
Структура файла 16
Подготовка файла 17
Протоколы обмена данными 18
TCP/IP 18
TCP/IPS 20
TLS 21
LOCAL 21
LOCALS 22
LASSP 23
BAL 24
ATCP/IP 25
ATCP/IPS 26
REZ 26
Загрузка openssl библиотек 27
Командные сетевые средства 28
Драйвер сервера 28
Назначение драйвера 28
Условия выполнения 28
Запуск драйвера 28
Общие ключи командной строки 28
Ключи командной строки ОС Linux, ЗОСРВ Нейтрино 31
Ключи командной строки ОС Windows 32

Выполнение драйвера 32
Настройка защищенного соединения со стороны сервера 33
Драйвер клиента 35
Назначение драйвера 35
Условия выполнения 35
Запуск драйвера 35
Общие ключи командной строки 36
Ключи командной строки ОС Linux, ЗОСРВ Нейтрино 41
Ключи командной строки ОС Windows 42
Настройка защищенного соединения со стороны клиента 42
Графические сетевые средства для ОС Windows 45
Графический драйвер клиента 45
Запуск графического сетевого драйвера клиента 45
Основное окно программы 45
Работа программы 47
Создание новой конфигурации 47
Загрузка файла существующей конфигурации 49
Сохранение конфигурации 49
Просмотр статистики серверов конфигурации 50
Сортировка списка серверов конфигурации 51
Добавление сервера 52
Редактирование параметров сервера 53
Удаление сервера 53
Установка сервера по умолчанию 53
Запуск клиента 53
Останов клиента 54
Соединение с сервером 54
Отсоединение от сервера 55
Запрет сервера 55
Настройка протоколирования 56
Протоколирование в память 57
Протоколирование в файл 58
Общие настройки приложения 59
Завершение работы программы 61
Графический драйвер сервера 61
Запуск графического сетевого драйвера сервера 61
Основное окно программы 61
Работа программы 62
Добавление драйвера 62
Редактирование конфигурации запуска драйвера 63
Удаление конфигурации запуска драйвера 63
Запуск драйвера при запуске приложения 63
Запуск драйвера 64
Останов драйвера 64
Отсоединение драйвера 64
Просмотр статистики 64
Настройка протоколирования 66
Настройка программы 66
Завершение работы программы 67
Тестирование конфигурации сети 68
Коды завершения сетевых средств 69
Управление сетевым доступом 70
Активизация клиента 71
Удаленный клиент 71
Локальный клиент 72

 © Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

Активизация сервера 72
Локальный сервер 72
Удаленный сервер 73
Приложение 1. Варианты сетевых конфигураций СУБД ЛИНТЕР 76
Приложение 2. Пример создания файла сетевой конфигурации 80
Приложение 3. Примеры конфигурационных файлов для запуска из сетевого суперсервиса inetd 83
Приложение 4. Работа с сертификатами 84
Приложение 5. Проверка работы сетевого клиента в среде ОС Linux, ЗОСРВ Нейтрино 106

Предисловие

Назначение документа

Документ содержит описание сетевой инфраструктуры СУБД ЛИНТЕР.

Приведены сведения о конфигурировании, настройке и порядке запуска сетевых средств для обеспечения локального и/или удаленного доступа из клиентских приложений к одной или нескольким БД с использованием различных протоколов обмена данными.

Описаны возможные способы организации взаимодействия клиентского приложения и СУБД ЛИНТЕР.

Документ предназначен для СУБД ЛИНТЕР БАСТИОН 6.0 сборка 20.8, далее по тексту СУБД ЛИНТЕР.

Для кого предназначен документ

Документ предназначен для администраторов СУБД ЛИНТЕР и разработчиков клиентских ЛИНТЕР-приложений.

Необходимые предварительные знания

Для работы с сетевыми средствами необходимо:

- знать основы функционирования локальных вычислительных сетей (ЛВС);
- уметь работать в соответствующей операционной системе на уровне продвинутого пользователя (в том числе устанавливать переменные окружения, контролировать состояние процессов ОС).

Дополнительные документы

- Администрирование комплекса средств защиты данных
- Установка СУБД ЛИНТЕР в среде ОС Linux
- Установка СУБД ЛИНТЕР в среде ОС Windows
- Сетевой администратор
- Запуск и останов СУБД ЛИНТЕР в среде ОС Linux
- Запуск и останов СУБД ЛИНТЕР в среде ОС Windows
- Репликация данных
- Система резервирования
- Справочник кодов завершения

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

Общие сведения

Сетевые средства СУБД ЛИНТЕР предназначены для обеспечения «прозрачного» доступа клиентских приложений к базам данных (БД), расположенных как на удаленных, так и на локальных компьютерах вычислительной сети. Прозрачность доступа обеспечивается с помощью следующих механизмов:

- доступ к локальным или удаленным БД выполняется посредством одинакового интерфейса;
- отсутствует необходимость знать местонахождение БД.

Активизация сетевых средств может выполняться одним из следующих способов:

- автоматически при запуске операционной системы (ОС);
- автоматически при запуске ядра СУБД ЛИНТЕР;
- вручную;
- из клиентских приложений.

Модель сетевой архитектуры СУБД ЛИНТЕР

В основу сетевой архитектуры СУБД ЛИНТЕР положена модель «клиент-сервер», которая предполагает наличие трех программных компонентов:

1) ЛИНТЕР-сервер;
2) клиентское приложение;
3) программные сетевые средства.

Допустимые схемы сетевых конфигураций клиентского компьютера и ЛИНТЕР-сервера приведены в приложении 1.

ЛИНТЕР-сервер

ЛИНТЕР-сервер отвечает за прием запросов клиентских приложений, их обработку и передачу результатов выполнения запроса клиентскому приложению. В общем случае ЛИНТЕР-сервер представляет собой ядро СУБД ЛИНТЕР и сетевой драйвер сервера. Список доступных клиентскому приложению ЛИНТЕР-серверов хранится в файле сетевой конфигурации клиентского приложения. Клиентское приложение обращается к ЛИНТЕР-серверу либо по имени компьютера, на котором установлена СУБД ЛИНТЕР (в случае доступа к конкретному ЛИНТЕР-серверу), либо без указания имени (в случае доступа к ЛИНТЕР-серверу по умолчанию).

Следует различать имя ЛИНТЕР-сервера и собственно имя БД, задаваемое при её создании с помощью утилиты `gendb`. Имя БД является её атрибутом, и для всех клиентских приложений оно одинаково, в то время как в локальной сети одна и та же БД может быть видна разным клиентским приложениям под различными логическими (ЛИНТЕР-сервер) именами.

Для соединения клиентского приложения с сервером БД сетевые средства СУБД ЛИНТЕР используют прозрачные сетевые драйверы (клиента и сервера), промышленные и собственные стандарты сетевых протоколов.

Сетевые драйверы выполняют коммуникационные задачи через общие точки входа, которые не зависят от специфики используемых сетевых протоколов (TCP/IP, TLS, разделяемая память).

Связь между клиентом и сервером обрабатывается по принципу стека.

Логической единицей обмена являются SQL-запросы и строки данных. На уровне CALL-интерфейса (интерфейс нижнего уровня) логические единицы обмена транслируются в серии команд к ядру СУБД.

Клиентское приложение

Клиентское приложение обеспечивает взаимодействие пользователя с ЛИНТЕР-сервером. Оно определяет SQL-операции, которые должен выполнить ЛИНТЕР-сервер, передает их через соответствующий программный интерфейс (программный интерфейс зависит от языка программирования, на котором разработано приложение – C/C++, Perl, PHP, Qt) и обрабатывает полученные от ЛИНТЕР-сервера результаты.

По отношению к ЛИНТЕР-серверу клиентское приложение может быть локальным или удаленным.

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

Локальное клиентское приложение функционирует на том же компьютере, что и ЛИНТЕР-сервер. В этом случае необходимость в использовании протоколов обмена данными между клиентским приложением и ЛИНТЕР-сервером может отсутствовать.

Удалённое клиентское приложение функционирует на отдельном от ЛИНТЕР-сервера компьютере. В этом случае клиентское приложение и ЛИНТЕР-сервер обмениваются информацией между собой с помощью сетевого протокола обмена данными.

Примечание
Чтобы разрешить удаленный доступ клиентских приложений к БД необходимо на ЛИНТЕР-сервере с этой БД:

1. выполнить команду:

```
grant access on unlisted station to all;
```

(разрешить доступ к этой БД со всех компьютеров);

2. либо команду на создание станции (разрешить доступ к этой БД только с конкретных компьютеров) (см. документ «Администрирование комплекса средств защиты данных»).

Программные сетевые средства

Программные сетевые средства обеспечивают обмен данными между клиентским приложением и ЛИНТЕР-сервером. В их состав входят:

- сетевой драйвер клиента;
- сетевой драйвер сервера;
- файл сетевой конфигурации клиента;
- переменные окружения ОС;
- протоколы обмена данными.

Сетевой драйвер клиента

Сетевой драйвер клиента – программное средство, устанавливаемое вместе с клиентским приложением и выполняющее следующие функции:

- прием запросов клиентских приложений к ЛИНТЕР-серверу;
- установление соответствия между именем ЛИНТЕР-сервера и его сетевым адресом;
- осуществление сетевого или локального соединения с ЛИНТЕР-сервером;
- передачу запросов клиентских приложений ЛИНТЕР-серверу через сеть или локально;
- прием результатов обработки запросов от ЛИНТЕР-сервера;
- передачу результатов обработки запросов клиентскому приложению.

Сетевой драйвер сервера

Сетевой драйвер сервера – программное средство, устанавливаемое в составе ЛИНТЕР-сервера и выполняющее следующие функции:

- осуществление локального соединения с ядром СУБД ЛИНТЕР;

- осуществление сетевого соединения с сетевыми драйверами клиентов;
- получение от сетевого драйвера клиента запросов клиентских приложений и передачу их ядру СУБД ЛИНТЕР;
- получение от ядра СУБД ЛИНТЕР результатов выполнения запросов и передачу их драйверу клиента;
- периодически отправляет тестовые пакеты драйверу клиента с целью проверки активности соединения.

Файл сетевой конфигурации клиента

Файл сетевой конфигурации клиента – текстовый файл (со стандартным именем `nodetab`), размещаемый на том же компьютере, где запускается клиентское приложение, и описывающий сетевую конфигурацию СУБД ЛИНТЕР с точки зрения этого клиентского приложения (имена доступных ЛИНТЕР-серверов, их сетевые адреса и другие параметры, необходимые сетевому драйверу клиента для установления сетевого соединения с конкретным ЛИНТЕР-сервером).

Переменные окружения ОС

Переменные окружения ОС – это переменные окружения операционной системы, используемые для настройки работы сетевых средств СУБД ЛИНТЕР и клиентских приложений.

Протоколы обмена данными

Протоколы обмена данными – правила передачи, используемые при обмене данными между сетевым драйвером клиента и сетевым драйвером сервера либо непосредственно между клиентским приложением и ядром СУБД ЛИНТЕР. Настраиваемые характеристики протоколов обмена задаются с помощью параметров протокола (например, величина тайм-аута сетевого соединения), которые хранятся в файле сетевой конфигурации `nodetab` и задаются индивидуально для каждого ЛИНТЕР-сервера. Формат представления параметров протокола определяется используемым сетевым протоколом.

 © Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

Механизм взаимодействия клиент-сервер

Сетевые средства СУБД ЛИНТЕР обеспечивают прозрачный механизм доступа клиентских приложений к ЛИНТЕР-серверам. Это означает, что клиентское приложение не обязано содержать внутри себя или запрашивать извне информацию о местоположении ЛИНТЕР-сервера в локальной сети и его сетевых параметрах доступа, кроме, возможно, самого имени ЛИНТЕР-сервера, к которому необходим доступ (В ряде случаев клиентские приложения могут работать с ЛИНТЕР-сервером, даже не зная его имени, то есть с «ЛИНТЕР-сервером по умолчанию»).

Сетевая прозрачность

Клиентские приложения, функционирующие на локальном ЛИНТЕР-сервере, могут быть развернуты в сети для доступа к аналогичному ЛИНТЕР-серверу без каких-либо изменений в тексте приложений. С точки зрения разработчика или пользователя приложения, весь обмен данными через сетевые средства невидим для пользователя приложения. Кроме того, можно изменять структуру сети для приложения, не изменяя самого приложения. Эта способность оставаться невидимым известна как сетевая прозрачность.

Независимость от протоколов

Сетевые средства СУБД ЛИНТЕР обеспечивают клиентским приложениям независимость от протоколов обмена данными. Сетевые средства могут работать с большинством сетевых протоколов, поддерживаемых операционными системами. Любое приложение, разработанное на любом компьютере, использовавшем любой протокол, может быть перенесено без изменений на другие компьютеры, использующие иные протоколы.

Прозрачность местоположения

Сетевые средства СУБД ЛИНТЕР обеспечивают прозрачность местоположения. Это означает, что доступ из клиентского приложения к ЛИНТЕР-серверу осуществляется только по имени сервера, никакой дополнительной информации о физическом местоположении сервера приложению не требуется.

Организация сетевого доступа

Операции соединения/отсоединения

Операция соединения между сетевыми драйверами клиента и сервера инициируется во время передачи любого запроса к удаленному ЛИНТЕР-серверу. Для соединения с ЛИНТЕР-сервером необходимо указать его имя в соответствующих полях программных интерфейсов или в опциях утилит. Если имя ЛИНТЕР-сервера не указывается, то выполняется соединение с ЛИНТЕР-сервером по умолчанию. ЛИНТЕР-сервер по умолчанию – это локальный ЛИНТЕР-сервер или, если локальный ЛИНТЕР-сервер отсутствует, один из ЛИНТЕР-серверов, перечисленный в файле сетевой конфигурации.

Разрыв соединения между сетевыми драйверами клиента и сервера происходит в следующие моменты:

- завершение работы всех клиентских приложений;
- завершение работы сетевого драйвера сервера или клиента;
- системная ошибка сетевого соединения (аппаратная или программная);
- завершение работы удаленного ЛИНТЕР-сервера.

При ошибках сетевого соединения или при завершении работы удаленного ЛИНТЕР-сервера клиентскому приложению передаются соответствующие коды завершения в диапазоне от 4000 до 4999.

Об ошибках сетевого соединения сообщает ОС в коде завершения соответствующего системного вызова, или же они обнаруживаются сетевыми драйверами самостоятельно по истечении тайм-аутов получения тестовых посылок.

Следует различать сетевое соединение между сетевыми драйверами клиента и сервера и соединение клиентского приложения с ЛИНТЕР-сервером. По одному сетевому соединению может быть открыто несколько соединений (каналов) к ЛИНТЕР-серверу, возможно, даже разными клиентскими приложениями.

При разрыве сетевого соединения ядро СУБД ЛИНТЕР закрывает все каналы, по которым происходила работа приложений с данного компьютера. Каналы приложения также закрываются в случае завершения работы самого приложения.

Операции обмена данными

Сетевые средства СУБД ЛИНТЕР реализуют следующий механизм обмена данными:

- клиентское приложение передает SQL-запрос пользовательскому интерфейсу, встроенному в это приложение;
- пользовательский интерфейс направляет запрос непосредственно ядру СУБД ЛИНТЕР или сетевому драйверу клиента;
- сетевой драйвер клиента передает по сети запрос сетевому драйверу сервера;
- сетевой драйвер сервера переадресует запрос ядру СУБД ЛИНТЕР;
- ядро СУБД ЛИНТЕР обрабатывает запрос и результаты его выполнения возвращает сетевому драйверу сервера;
- сетевой драйвер сервера передает по сети результаты выполнения запроса сетевому драйверу клиента;

- сетевой драйвер клиента передает полученные данные клиентскому приложению.

Запросы клиентского приложения при работе через сеть могут быть как синхронными, так и асинхронными (аналогично работе с локальным ЛИНТЕР-сервером).

Примечание
Некоторые протоколы (`LOCAL`) могут не использовать в качестве посредника сетевой драйвер сервера, а работать напрямую с ядром, используя механизмы межпроцессного обмена. Такие протоколы могут работать только в рамках одной машины. Они используются для доступа приложения к нескольким ядрам СУБД ЛИНТЕР на локальной машине.

Аварийное прекращение соединения

Аварийное прекращение соединения возникает в ситуации, когда клиентское приложение аварийно завершает свою работу, не успев проинформировать сетевые средства о необходимости закончить соединение с ЛИНТЕР-сервером. В этом случае сетевые средства сбрасывают операции клиента и сервера, что приведет к разъединению текущей операции.

Разъединение по тайм-ауту

Разъединение по тайм-ауту возникает при обнаружении зависшего соединения. Сетевой драйвер сервера периодически (с заданным при конфигурировании сетевого доступа интервалом) посылает тестовый пакет. Если не получено ни одного пакета в течение заданного времени (обычно из-за недоступности компьютера), соединение закрывается, клиентскому приложению передается соответствующий код завершения.

Варианты сетевых соединений

Локальный доступ к БД по умолчанию

Конфигурация локального доступа к БД по умолчанию приведена на рисунке 1. В этом случае клиентское приложение и ЛИНТЕР-сервер (БД и ядро СУБД ЛИНТЕР) размещаются на одном компьютере.

Конфигурационный файл `nodetab` не требуется (доступ осуществляется по умолчанию к установленному на компьютере ЛИНТЕР-серверу).

Сетевые средства (сетевые драйверы клиента и сервера) не используются.

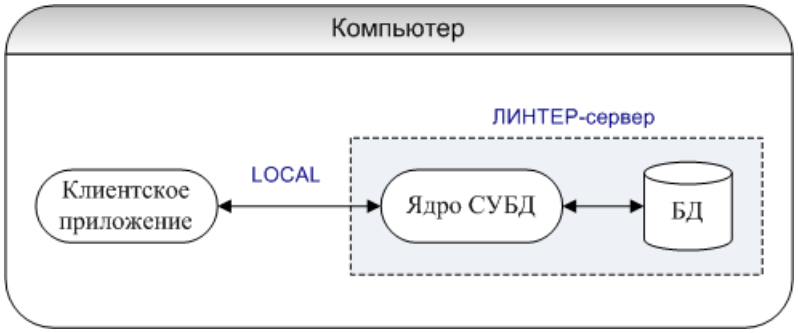


Рисунок 1. Конфигурация локального доступа по умолчанию

Локальный доступ через сетевой драйвер клиента

Конфигурация локального доступа через сетевой драйвер клиента приведена на рисунке 2. В этом случае клиентское приложение и один или несколько ЛИНТЕР-серверов (БД и экземпляры ядра СУБД ЛИНТЕР) размещаются на одном компьютере.

Используется один сетевой драйвер клиента и несколько экземпляров СУБД ЛИНТЕР.

Конфигурационный файл `nodetab` требуется.

Пример конфигурационного файла:

```
Условное имя  Протокол Адрес Порт Тайм-аут  Тайм-аут  Тайм-аут
 компьютера                        сервера   клиента  соединения
     A          LOCAL   1254
```

Примечание
В протоколе LOCAL значение `Адрес` в файле `nodetab` определяет не номер порта, а значение идентификатора механизма межпроцессного взаимодействия, аналог переменной окружения `LINTER_MBX` (см. описание протокола LOCAL).

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

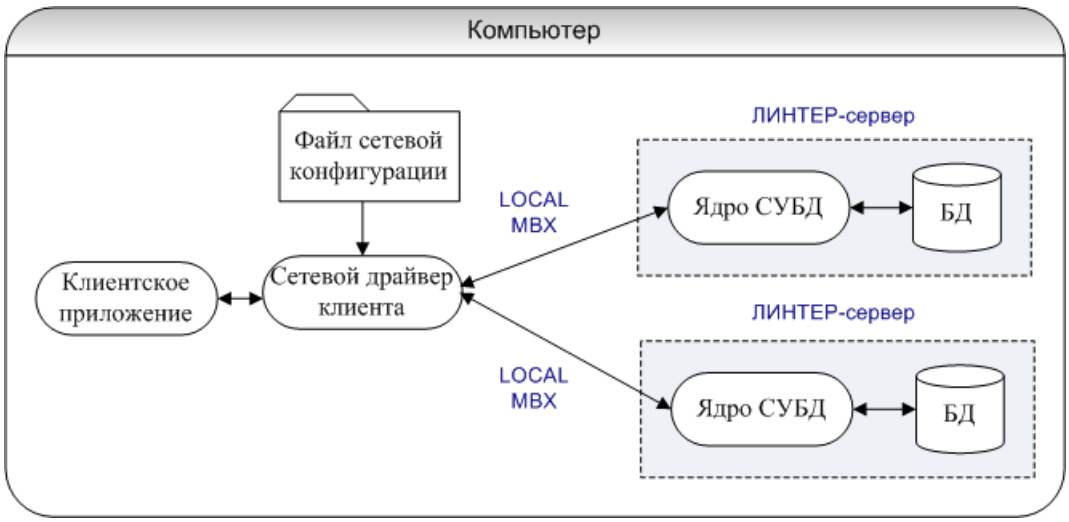


Рисунок 2. Конфигурация локального доступа через сетевой драйвер клиента

Локальный сетевой доступ

Конфигурация локального сетевого доступа приведена на рисунке 3. В этом случае клиентское приложение и один или несколько ЛИНТЕР-серверов (БД и экземпляры ядра СУБД ЛИНТЕР) размещаются на одном компьютере.

Используется один сетевой драйвер клиента и столько экземпляров сетевого драйвера сервера, сколько установлено на компьютере БД, и экземпляров СУБД ЛИНТЕР (одной и той же или разных версий).

Конфигурационный файл `nodetab` требуется.

Данная конфигурация может применяться в информационной системе, в которой предполагается масштабирование системы. В этом случае локальная связка «Сетевой драйвер сервера + ядро СУБД ЛИНТЕР» выносятся на отдельный компьютер, при этом клиентское приложение не меняется.

Пример конфигурационного файла:

```
Условное имя  Протокол Адрес     Порт Тайм-аут  Тайм-аут  Тайм-аут
 компьютера                             сервера   клиента
 соединения
      A        TCPIP   localhost
      B        TCPIPS  127.0.0.1 1061    2
```

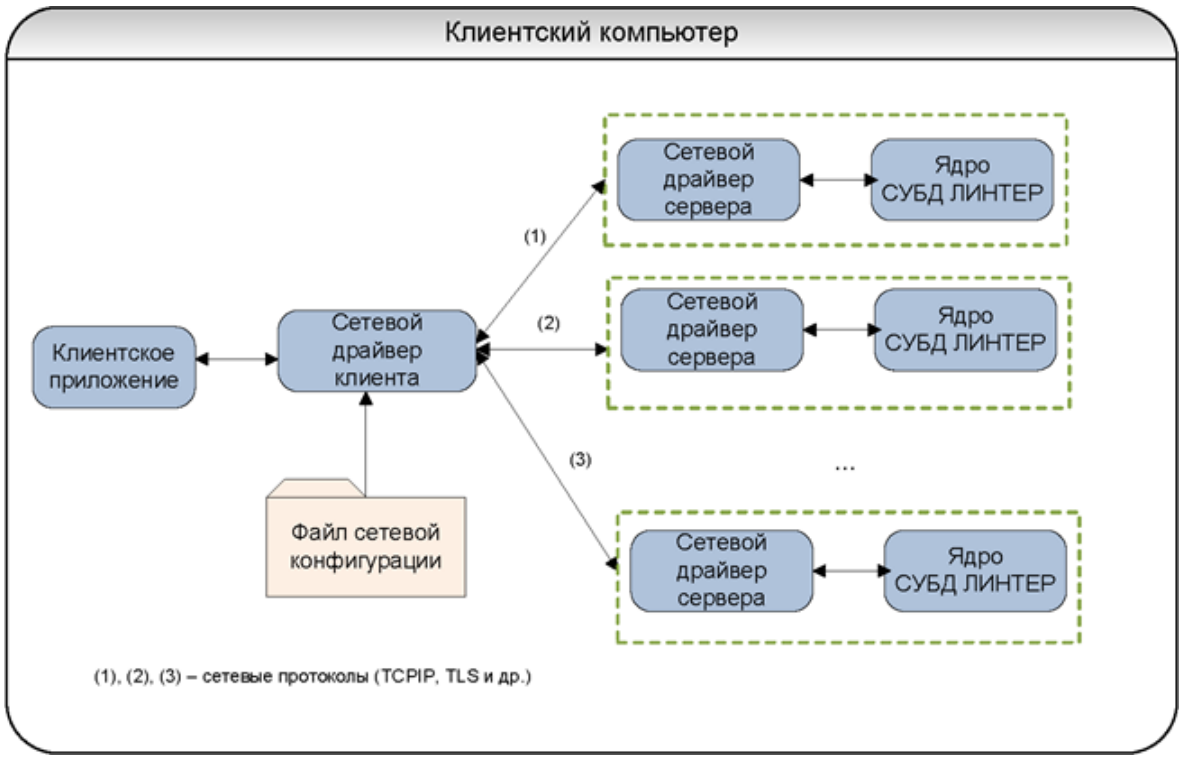


Рисунок 3. Конфигурация локального сетевого доступа

Удаленный сетевой доступ

Конфигурация удаленного сетевого доступа приведена на рисунке 4. В этом случае клиентское приложение и один или несколько ЛИНТЕР-серверов (БД и экземпляры ядра СУБД ЛИНТЕР) размещаются на разных компьютерах.

Конфигурационный файл `nodetab` требуется.

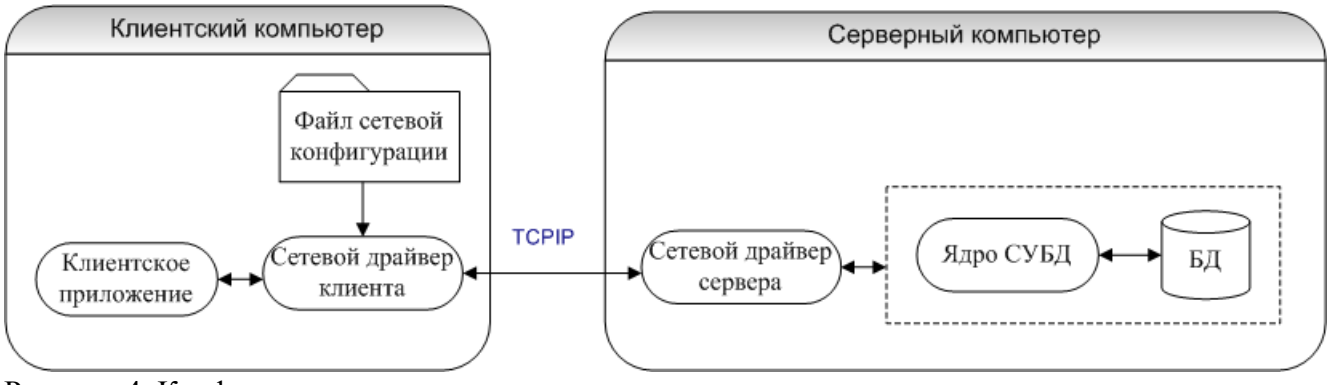


Рисунок 4. Конфигурация удаленного сетевого доступа

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

Установка сетевых средств

В среде ОС Windows установка сетевых средств выполняется выборочно в процессе установки СУБД ЛИНТЕР путем выставления флажка Сетевые драйверы в окне «Выбор компонентов для установки» (см. документ «Установка СУБД ЛИНТЕР в среде ОС Windows»).

В среде ОС типа Linux, ЗОСРВ Нейтрино сетевые средства устанавливаются всегда при установке СУБД ЛИНТЕР, независимо от того, будут ли они использоваться в дальнейшем (см. документ «Установка СУБД ЛИНТЕР в среде ОС Linux»).

Файл сетевой конфигурации

Структура файла

Для управления доступом клиентских приложений к БД используется файл сетевой конфигурации клиента – текстовый файл с именем `nodetab` (приложение 2). Строки файла описывают сетевые параметры ЛИНТЕР-серверов и, при необходимости, системы горячего резервирования СУБД ЛИНТЕР. Каждая строка разделяется на поля (рис. 5). Разделительным символом полей является пробел или символ табуляции. Каждое поле имеет свое назначение и устанавливает один из параметров сетевого соединения ЛИНТЕР-сервера.

Условное имя компьютера	Протокол	Адрес	Порт	Тайм-аут сервера	Тайм-аут клиента	Тайм-аут соединения

Рисунок 5. Поля строки файла `nodetab`

Обязательными полями являются:

- условное имя компьютера;
- протокол;
- адрес.

В строке могут присутствовать и другие поля для работы сетевых компонентов СУБД ЛИНТЕР (необязательные поля имеют значения по умолчанию):

- порт;
- тайм-аут сервера;
- тайм-аут клиента;
- тайм-аут соединения.

Формат и значение этих полей зависит от типа используемого протокола обмена данными (см. раздел «Протоколы обмена данными»).

Поле `Тайм-аут сервера` задает максимально допустимое время (в минутах), после которого сетевой драйвер сервера будет считать, что сетевое соединение с драйвером клиента разорвано. При сетевой работе драйвер клиента посылает сетевому драйверу сервера тестовые посылки несколько чаще, чем этот интервал. В случае если драйвер сервера не получит тестового пакета от драйвера клиента в течение времени, указанного в поле `Тайм-аут сервера`, соединение будет закрыто драйвером сервера с генерацией соответствующего кода завершения. Значение тайм-аута по умолчанию: 1 минута. При явном задании нулевого тайм-аута тестовые посылки драйвером клиента посылаться не будут, и в случае разрыва соединения длительность обнаружения разрыва будет определяться реализацией протокола в ОС.

Поле `Тайм-аут клиента` задает интервал времени (в сек.), через который драйвер клиента считает соединение с драйвером сервера, разорванным в случае неполучения от драйвера сервера тестового пакета. Значение по умолчанию: 0, то есть тестовые пакеты не посылаются, и состояние разрыва соединения драйвером клиента не обнаруживается.

Поле `Тайм-аут соединения` задает время (в секундах), по истечении которого будет фиксироваться ошибка, если сетевое соединение между драйверами клиента и

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

сервера установить не удалось. По умолчанию устанавливается минимальное значение (3 секунды).

Подготовка файла

В среде ОС Linux, ЗОСРВ Нейтрино файл `nodetab` создается вручную с помощью любого текстового редактора, имеющегося в операционной системе.

В среде ОС Windows он может быть подготовлен следующими способами:

- вручную;
- в процессе установки СУБД ЛИНТЕР на клиентском компьютере (см. документ «Установка СУБД ЛИНТЕР в среде ОС Windows»);
- для сетевой Windows – с помощью утилиты `linadm` (см. документ «Сетевой администратор»).

Правила ввода файла `nodetab`:

1) обязательными полями являются:
   - условное имя компьютера (не более 8 символов в латинской кодировке);
   - имя протокола;
   - адрес.
2) необязательным полям с незаполненными значениями присваиваются значения по умолчанию;
3) неправильно введенное значение (например, вместо цифрового значения буквенное) игнорируется, сообщение об ошибке не выдается, и неправильное значение заменяется значением по умолчанию;
4) каждая строка файла должна заканчиваться символом «переход на новую строку»;
5) разделителем столбцов файла могут быть пробел и знаки табуляции (вследствие этого задаваемые в таблице имена не должны содержать пробелов);
6) файл допускает ввод комментариев. Признаком начала комментария является символ #. Комментарием считаются все символы от начала комментария до конца строки файла.

Протоколы обмена данными

С функциональной точки зрения протоколы обмена можно разделить на следующие группы:

- сетевые протоколы, предназначенные для обмена данными между клиентским приложением и удаленным ЛИНТЕР-сервером. В частном случае сетевые протоколы могут использоваться для организации одновременного доступа к нескольким локальным ЛИНТЕР-серверам. В эту группу входят протоколы: TCP/IP, TCP/IPS, TLS;
- локальные протоколы, предназначенные для обмена данными между клиентским приложением и локальным ЛИНТЕР-сервером. В эту группу входят протоколы: LOCAL, LOCALS;
- протоколы резервного доступа, предназначенные для доступа к одному из нескольких ЛИНТЕР-серверов именованной группы. В эту группу входит протокол LASSP;
- протоколы балансировщика нагрузки, предназначенные для балансировки нагрузки клиентских приложений. В эту группу входит протокол BAL;
- протоколы системы репликации (тиражирования) данных, предназначенные для обмена данными между реплицируемыми серверами. Клиентскому приложению эти протоколы недоступны. В эту группу входят протоколы: ATCP/IP, ATCP/IPS;
- протокол системы резервирования: REZ.

TCP/IP

Протокол передачи поверх TCP протокола.

Описание в конфигурационном файле

Условное имя компьютера

Условное имя компьютера локальной сети, на котором запущено ядро СУБД.

Поле обязательное.

Протокол

Обозначение протокола: TCPIP.

Поле обязательное.

Адрес

Сетевой IP-адрес или имя, определяемое через службу DNS (Domain Name Service). Например, 100.101.102.103 – числовой IP-адрес, mycomp.myorg.mydomen – полное DNS-имя, mycomp – краткое DNS-имя.

Поле обязательное.

Порт

Номер порта TCP/IP удаленного сервера. Задается в десятичном или шестнадцатеричном виде.

Поле необязательное. По умолчанию используется номер порта 1060 (0x424).

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

Примечания

1. Если на удаленном компьютере запущено несколько сетевых драйверов сервера, использующих несколько портов, то необходимо явно указывать каждый порт, за исключением, может быть, одного.
2. Поле обязательно для заполнения, если сетевой драйвер сервера на удаленном компьютере «слушает» порт, отличный от порта 1060.
3. В ОС Linux для получения более наглядной информации о портах TCP/IP по команде netstat можно внести в файл `/etc/services` строку:

```
1060  tcpip  Linter
```

Тайм-аут сервера

Тайм-аут обнаружения разрыва соединения сетевым драйвером сервера (мин.).

Поле необязательное. По умолчанию используется значение 1 мин.

Примечание

При указании значения 0 тайм-аут игнорируется, и разрыв соединения диагностируется средствами ОС.

Тайм-аут клиента

Тайм-аут обнаружения разрыва соединения сетевым драйвером клиента (сек.).

Поле необязательное. По умолчанию используется значение 0 (тайм-аут определяется ОС).

Примечание

Величина этого тайм-аута должна быть больше 2 секунд.

Тайм-аут соединения

Тайм-аут ожидания установки соединения сетевым драйвером клиента (сек.).

Поле необязательное. По умолчанию используется значение 0 (тайм-аут определяется ОС).

Применимость протокола для программных платформ

Все операционные системы, поддерживающие TCP/IP.

Пример

Условное имя компьютера	Протокол	Адрес	Порт	Тайм-аут сервера, мин.	Тайм-аут клиента, сек.	Тайм-аут соединения, сек.
SERV1	TCPIP	192.168.69.226	1060	1	4	
SERV2	TCPIP	195.98.69.227	0x425			
SERV3	TCPIP	Accountant	1062	2	20	4
SERV4	TCPIP	relex.ru				

TCP/IPS

Протокол поверх TCP/IP, защищенный SSL.

Примечание
Протокол TCP/IPS является устаревшим и его не рекомендуется использовать. Вместо него рекомендуется применять аналогичный ему протокол TLS.

Описание в конфигурационном файле

Условное имя компьютера

Условное имя компьютера локальной сети, на котором запущено ядро СУБД.

Поле обязательное.

Протокол

Обозначение протокола: TCPIPS.

Поле обязательное.

Адрес

Значение идентично протоколу TCP/IP.

Порт

Значение идентично протоколу TCP/IP.

Тайм-аут сервера

Значение идентично протоколу TCP/IP.

Тайм-аут клиента

Значение идентично протоколу TCP/IP.

Тайм-аут соединения

Значение идентично протоколу TCP/IP.

Применимость протокола для программных платформ

Возможно применение в ОС Linux, ЗОСРВ Нейтрино. В некоторых поставках может не включаться. Нецелесообразно применение во встроенных системах из-за значительных накладных расходов на реализацию SSL.

Пример

Условное имя компьютера	Протокол	Адрес	Порт	Тайм-аут сервера, мин.	Тайм-аут клиента, сек.	Тайм-аут соединения, сек.
SERV1	TCPIPS	195.98.69.226	1060	1		
SERV2	TCPIPS	195.98.69.227	0x425			
SERV3	TCPIPS	Accountant	1062	2	20	4

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

TLS

Аналогичен протоколу TCP/IPS, но с использованием TLS протокола. Использует для защиты соединения TLS1 протокол с возможностью поддержки ГОСТ алгоритмов. Поддержка включается автоматически при использовании сервером и клиентом ключей ГОСТ алгоритма. По умолчанию использует AES256 алгоритм.

Позволяет выполнять аутентификацию сетевых драйверов с использованием удостоверяющего центра. Сертификаты удостоверяющих центров складываются в файлы `dbc_tcp.CA` и `dbs_tcp.CA` для `dbc_tcp` и `dbs_tcp` соответственно. В эти файлы должны складываться сертификаты удостоверяющих центров, которым доверяет клиент и сервер.

Если присутствует файл с расширением `.CRL` (`dbc_tcp.CRL`, `dbs_tcp.CRL`), то считается, что в этом файле располагается список отозванных сертификатов. Список отозванных сертификатов имеет смысл только при наличии сертификата удостоверяющего центра.

LOCAL

Протокол для доступа к нескольким локальным ЛИНТЕР-серверам из одного клиентского приложения. С помощью протокола LOCAL клиентское приложение может обращаться через сетевой драйвер клиента к нескольким ядрам СУБД на данном компьютере без участия сетевого драйвера сервера. Возможна также работа со всеми ЛИНТЕР-серверами через протокол LOCAL.

Описание в конфигурационном файле

Условное имя компьютера

Условное имя компьютера локальной сети, на котором запущено ядро СУБД.

Поле обязательное.

Протокол

Обозначение протокола: LOCAL.

Поле обязательное.

Адрес

Задает идентификатор механизма межпроцессного взаимодействия. Аналог переменной окружения `LINTER_MBX`. Осуществляет доступ к ядру СУБД, при запуске которого `LINTER_MBX` принимала точно такое же значение, как в этом поле (см. документ «Запуск и останов СУБД ЛИНТЕР в среде ОС Linux»).

Поле обязательное.

Порт

Не задается.

Тайм-аут сервера

Не задается.

Тайм-аут клиента

Не задается.

Тайм-аут соединения

Не задается.

Применяемость протокола для программных платформ

Не поддерживается ОС VxWorks и ОС РВ, на которых может выполняться только один экземпляр СУБД.

Пример

1) На компьютере запускаются два экземпляра СУБД.

Файл `nodetab`:

Условное имя компьютера	Протокол	Адрес	Порт	Тайм-аут сервера, мин.	Тайм-аут клиента, сек.	Тайм-аут соединения, сек.
SERV1	LOCAL	1234				

Запуск СУБД и сетевого драйвера клиента:

```
>linter /base=/first/base
>LINTER_MBX=1234
>export LINTER_MBX
>linter /base=/second/base
>unset LINTER_MBX
>dbc_tcp
```

2) Обращение к СУБД для БД `/first/base`

```
>inl
```

3) Обращение к СУБД для БД `/second/base`

```
>inl -n SERV1
```

LOCALS

Протокол, позволяющий клиентскому приложению обращаться к нескольким локальным ядрам СУБД ЛИНТЕР, использующим UNIX domain sockets для LOCALS.

Если для межпроцессного обмена драйвер клиента и клиентское приложение используют UNIX domain sockets, то протокол LOCALS является синонимом протокола LOCAL.

Все сказанное о протоколе LOCAL применимо к LOCALS.

Описание в конфигурационном файле

См. описание протокола LOCAL.

Применяемость протокола для программных платформ

Только в ОС Linux, ЗОСРВ Нейтрино.

 © Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

LASSP

Протокол доступа к одному из нескольких ЛИНТЕР-серверов в заданной группе.

По протоколу LASSP (Linter Automated Standby Server Protocol) осуществляется попытка соединения со всеми ЛИНТЕР-серверами, перечисленными в строке файла `nodetab` для этого протокола. При успешном соединении с первым ЛИНТЕР-сервером группы обмен данными выполняется через это соединение. При разрыве соединения, через которое в данный момент выполняется обмен данными, осуществляется попытка автоматического переключения на следующий (по порядку) ЛИНТЕР-сервер группы, принадлежащий данному протоколу LASSP. Если попытки соединения со всеми ЛИНТЕР-серверами группы оказались неудачными, клиентскому приложению возвращается соответствующий код завершения.

Описание в конфигурационном файле

Условное имя компьютера

Условное имя компьютера локальной сети, на котором запущено ядро СУБД.

Поле обязательное.

Протокол

Обозначение протокола: LASSP.

Поле обязательное.

Адрес

Список имен ЛИНТЕР-серверов, разделенных пробелами или символами табуляции. Перечисленные в данном списке имена должны быть описаны в этом же файле `nodetab` в строках, предшествующих строке с описанием протокола LASSP. Не допускается включение в группу ЛИНТЕР-серверов для одного LASSP-протокола ЛИНТЕР-серверов из других групп LASSP-протоколов: каждый ЛИНТЕР-сервер должен включаться только в одну группу.

Порт

Не задается.

Тайм-аут сервера

Не задается.

Данный протокол не имеет собственных средств ограничения времени соединения и обнаружения разрыва соединения, поэтому необходимо обязательно устанавливать индивидуальные тайм-ауты каждого из ЛИНТЕР-серверов, входящих в группу. Если хотя бы один ЛИНТЕР-сервер не будет иметь тайм-аутов, то до возврата кода завершения клиентскому приложению и обнаружения разрыва соединения может пройти большой интервал времени.

Тайм-аут клиента

Не задается. См. пункт Тайм-аут сервера данного протокола.

Тайм-аут соединения

Не задается. См. пункт Тайм-аут сервера данного протокола.

Применяемость протокола для программных платформ

В среде ОС Linux, ЗОСРВ Нейтрино и Windows.

Пример

Файл `nodetab`:

Условное имя компьютера	Протокол	Адрес	Порт	Тайм-аут сервера, мин.	Тайм-аут клиента, сек.	Тайм-аут соединения, сек.
SERV1	TCPIP	195.200.56.40	1060	1	20	10
SERV2	LOCAL	1234				
SERV3	TCPIPS	195.200.56.50	1060	1	20	10
MAIN	LASSP	SERV1 SERV2 SERV3				

BAL

Протокол для балансировки нагрузки клиентских приложений.

Описание в конфигурационном файле

```
Формат строки настройки виртуального сервера кластера:
<name> BAL <имя сервера 1> [:<приоритет сервера 1>]
<имя сервера 2> [:<приоритет сервера 2>]
...
<имя сервера N> [:<приоритет сервера N>]
```

Условное имя компьютера

Логическое имя виртуального сервера балансировщика нагрузки. В режиме балансировки нагрузки клиентское приложение должно обращаться именно к этому серверу.

Поле обязательное.

Протокол

Обозначение протокола: BAL.

Поле обязательное.

<имя сервера 1>, <имя сервера 2>, ..., <имя сервера N>

Условные имена узлов локальной сети, на которых установлены серверы кластера и между которыми должна осуществляться балансировка нагрузки. Каждый из этих узлов сети должен быть предварительно описан в этом же файле nodetab. В списке узлов каждое имя узла отделяется от другого имени пробелами или символами табуляции.

<приоритет сервера 1>, <приоритет сервера 2>, ..., <приоритет сервера N>

Приоритеты соответствующих серверов кластера. Пробелы между именем узла, символом «:» и приоритетом недопустимы. Приоритет представляет собой число в диапазоне от 1 до 16. Меньшее значение соответствует более высокому приоритету узла, то есть его большей загрузке. По умолчанию приоритет равен 8.

 © Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

Пример файла сетевой конфигурации nodetab:

```
SRV1 TCPIP 172.20.9.101 1060 1 20 20
SRV2 TCPIP 172.20.9.102 1060 1 20 20
SRV3 TCPIP 172.20.9.103 1060 1 20 20
CLSTR BAL SRV1:5 SRV2 SRV3:10
```

ATCP/IP

Протокол ATCP/IP предназначен для использования системой репликации (асинхронного тиражирования), поскольку сервер репликации и сетевой драйвер клиента имеют общий файл конфигурации.

Описание в конфигурационном файле

Условное имя компьютера

Условное имя компьютера локальной сети, на котором запущено ядро СУБД, и в БД которого должны тиражироваться данные. Это имя должно совпадать с именем сервера, создаваемым в БД ЛИНТЕР посредством запроса вида create server.

Поле обязательное.

Примечание
Количество строк с протоколом ATCP/IP в файле `nodetab` должно соответствовать количеству ЛИНТЕР-серверов, на которые выполняется тиражирование данных (см. документ «Репликация данных»).

Протокол

Обозначение протокола: ATCPIP.

Поле обязательное.

Адрес

Сетевой IPv4-адрес или имя, определяемое через службу DNS (Domain Name Service). Например, 100.101.102.103 – числовой IP-адрес, mycomp.myorg.mydomen – полное DNS-имя, mycomp – короткое DNS-имя.

Поле обязательное.

Порт

Номер TCP/IP порта удаленного сервера репликации. Задается в десятичном или шестнадцатеричном виде.

Поле необязательное. По умолчанию используется номер порта 1080 (0x438).

Номер должен совпадать с номером порта, указываемым при запуске сервера репликации на принимающей стороне.

Тайм-аут сервера

Поле не используется.

Тайм-аут клиента

Поле не используется.

Тайм-аут соединения

Поле не используется.

Применяемость протокола для программных платформ

Все операционные системы, на которых устанавливается система репликации СУБД ЛИНТЕР.

Примеры

См. документ «Репликация данных».

ATCP/IPS

Протокол TCP/IP для использования в системе репликации (асинхронного тиражирования), защищенный SSL. Полностью аналогичен протоколу ATCP/IP, за исключением преобразования данных, передающихся по сети.

Описание в конфигурационном файле

См. описание протокола ATCP/IP.

REZ

Зарезервировано для системы резервирования (см. документ «Система резервирования»).

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

Загрузка openssl библиотек

В сетевых средствах возможно использование динамических или статических `openssl` библиотек. Динамически загружаемая библиотека имеет приоритет над статически.

К динамически загружаемым библиотекам для ОС типа Windows относятся `libssl.dll` и `libcrypto.dll`. Таким образом, Windows будет пытаться загружать пары `.dll`:

- `ssleay32.dll` - `libeay32.dll`;
- `libssl.dll` - `libcrypto.dll`.

Для ОС Linux, ЗОСРВ Нейтрино будет производиться попытка загрузки `libssl.so`.

Для ОС Windows сетевые утилиты содержат в себе статически включенную библиотеку `openssl` и могут использовать ее в случае отсутствия динамически загружаемой библиотеки.

Для ОС Linux, ЗОСРВ Нейтрино необходимо наличие динамических библиотек `openssl` в дистрибутиве СУБД ЛИНТЕР. В случае их отсутствия протоколы TLS и TCPIPS не могут быть использованы.

Командные сетевые средства

Драйвер сервера

Назначение драйвера

Сетевой драйвер сервера предназначен для обслуживания запросов клиентов на ЛИНТЕР-сервере.

Условия выполнения

Для работы драйвера в ОС должен быть установлен протокол TCP/IP.

При работе драйвера сервера используются значения переменных окружения:

1) **LINTER_MBX** – переменная, определяющая «имя почтового ящика» для обмена данными между ядром СУБД ЛИНТЕР и клиентскими приложениями. Под «почтовым ящиком» следует понимать некий межпроцессный механизм обмена между ядром ЛИНТЕР и приложением. Используется для передачи данных от приложения ядру ЛИНТЕР и обратно. Изменение значения этой переменной может быть использовано для запуска нескольких ядер на одном компьютере.

   Значение `LINTER_MBX` должно быть уникальным для данного ядра СУБД ЛИНТЕР и приложений, работающих с этим ядром.
2) **DBS_LOG** – переменная, определяющая каталог файла протоколирования работы драйвера серверной части.

Запуск драйвера

Возможны следующие варианты запуска сетевого драйвера сервера:

- одновременно с запуском ядра СУБД с помощью указания ключа `/TCP` (аналогично будет выполнен запуск драйвера при установке флага «Автозапуск СУБД ЛИНТЕР после старта машины», см. «Запуск и останов СУБД ЛИНТЕР в среде ОС Linux» , «Запуск и останов СУБД ЛИНТЕР в среде ОС Windows»);
- ручной запуск исполняемого файла драйвера средствами ОС, как службы ОС;
- запуск исполняемого файла драйвера из командной строки, как консольного приложения;
- запуск в среде ОС Windows, как сервиса с помощью инструментального средства «Сетевой администратор».

Команда запуска драйвера, как консольное приложение:

```
dbs_tcp [<ключ> …]
<ключ>::= <вариант 1> | <вариант 2>
<вариант 1>::= -<код ключа>{<пробел> | = }<значение ключа>
<вариант 2>::= /<код ключа>=<значение ключа>
```

Общие ключи командной строки

Драйвер обрабатывает следующие ключи:

 © Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

`-H | -HELP | -?`

Подсказка по всем ключам драйвера.

`-P=[<сетевой адрес>:]<номер порта>`

Номер порта протокола TCP/IP. Сетевой драйвер сервера принимает соединения сетевого драйвера клиента через этот порт. По умолчанию `<номер порта>` равен 1060. Драйвер сервера можно привязывать не только к порту, но и к сетевому адресу (адресу интерфейса). В этом случае адрес должен задаваться только в числовом виде.

Примечания

1. Разрешена работа только с одним интерфейсом либо со всеми интерфейсами компьютера.
2. В случае отсутствия ключа -P делается попытка чтения порта из реестра из описания БД и только потом используется значение по умолчанию.

Пример.

```
dbs_tcp -P 127.0.0.1:1060
```

(драйвер будет работать только с локальным интерфейсом)

`-M=<идентификатор очереди> | NAME=<идентификатор очереди>`

Идентификатор очереди сообщений СУБД ЛИНТЕР. Для ОС Linux имеет числовое значение, для ЗОСРВ «Нейтрино» и ОС Windows – текстовое. Если ключ не задан, то значение идентификатора очереди берется из переменной окружения `LINTER_MBX`, а если не определена и эта переменная, то для ОС Linux по умолчанию принимается значение 20561, а для ЗОСРВ «Нейтрино» и ОС Windows – пустая строка.

`-W`

Заставляет отслеживать активность ядра СУБД ЛИНТЕР. Если ядро стало неактивным, драйвер также завершает свою работу.

`-I=<спецификация файла>`

Заставляет записывать идентификатор процесса (PID) драйвера сервера в текстовый файл. Если заданный файл существует, он будет перезаписан.

Примечание

Информация о PID драйвера в текстовом файле требуется, как правило, при обработке командных файлов (например, для того, чтобы завершить работу драйвера).

`-N`

Запрещает сообщать ядру СУБД ЛИНТЕР о разрыве сетевого соединения. При последующем восстановлении соединения драйвер продолжит работу, будто разрыва соединения не было.

`-VERSION`

Вывод на консоль информации о версии драйвера.

`-SSLONLY`

Запрет на передачу данных по сети в немаскируемом режиме (по незащищенному соединению). В режиме маскирования исключается возможность получения информации

путем прослушивания сетевого соединения. Если будет предпринята попытка переслать данные по незащищенному соединению, то это соединение будет закрыто.

`-SSLKEY=<длина ключа>`

Генерировать и запоминать ключи маскирования (пара «ключ-сертификат») сетевого сервера в файлах `dbs_tcp.key` и `dbs_tcp.crt` текущего каталога. Если ключи маскирования не были сохранены (файлы `dbs_tcp.key` и `dbs_tcp.crt` отсутствуют), то они будут генерироваться заново при каждом запуске драйвера. Рекомендуемое значение длины ключа – в диапазоне 2024-4096. В случае копирования файла `dbs_tcp.crt` на сторону клиента под другим именем идентификация сетевого сервера сетевым клиентом возможна во время установки соединения. Для этого необходимо, чтобы приведенное к верхнему регистру имя файла с расширением `.crt` на стороне клиента совпадало с именем ЛИНТЕР-сервера, с которым устанавливается соединение, из файла сетевой конфигурации `nodetab`.

Длина генерируемого ключа указывается в битах. Если значение не задано, по умолчанию генерируется ключ длиной 512 бит (минимально возможная длина).

Примечание

Файл `dbs_tcp.key` необходимо защитить средствами ОС Linux, ЗОСРВ Нейтрино от просмотра и изменения другими пользователями, тем более нельзя перемещать его на другую машину.

`-SSLAUTH`

Режим идентификации клиента. В нем при выполнении очередного (не первого) соединения от клиента принимается используемый им ключ маскирования и сравнивается с запомненным ранее. Файл маскирования клиента, сохраняемый на диске, имеет имя, соответствующее IP-адресу клиента в точечной нотации и расширение `.crt`. В случае отсутствия в рабочем каталоге сетевого сервера файла маскирования клиента, он создается автоматически с сохранением в нем полученного ключа маскирования. В дальнейшем сравнение передаваемого при соединении ключа маскирования будет производиться с ключом, запомненным в первый раз.

Файл ключа маскирования клиента `dbc_tcp.crt` может быть перемещен на ЛИНТЕР-сервер администратором СУБД и соответствующим образом переименован. Так как клиентов может быть довольно много, целесообразно производить запуск сетевого сервера из отдельного каталога. Данный режим поддерживается только в случае наличия в текущем каталоге сетевого драйвера сервера, файлов ключа и сертификата.

Примечания

1. Этот механизм не обеспечивает 100% надежность идентификации клиента, так как в принципе ключ клиента может быть получен при подмене адреса ЛИНТЕР-сервера.
2. Для обеспечения надежной идентификации необходимо использовать протокол TLS и сертификаты, подписанные удостоверяющим центром. Список сертификатов удостоверяющих центров хранится в файлах `dbc_tcp.CA` и `dbs_tcp.CA`. Список отозванных сертификатов – в `dbc_tcp.CRL` и `dbs_tcp.CRL`.

`-SSLNOCREAT`

Запрещает создавать файл ключа маскирования клиента при работе в режиме идентификации клиента. Это исключает добавление новых клиентских компьютеров без

ведома администратора СУБД. Ключ может применяться только совместно с ключом SSLAUTH.

`-T=<тайм-аут>`

Задает длительность тайм-аута (сек.) проверки ядра СУБД ЛИНТЕР. По истечении половины заданного тайм-аута ядру СУБД посылается тестовый пакет. Если до окончания заданного тайм-аута от ядра не придет ни одного сообщения, то произойдет разрыв всех сетевых соединений. Если задан только ключ -T, то по умолчанию величина тайм-аута принимается равной 60 сек. Величина тайм-аута не может быть меньше 10 сек.

`-INTERACTIVE={0|1}`

Значение параметра:

0 – отменяет интерактивный режим. В этом случае `dbs_tcp` в случае ошибки инициализации завершается немедленно;

1– устанавливает интерактивный режим. В этом случае `dbs_tcp` в случае ошибки инициализации будет выведена просьба нажать клавишу ENTER для завершения работы.

Интерактивный режим отключается автоматически при запуске `dbs_tcp` как сервис или в фоновом режиме.

Интерактивный режим включается автоматически при запуске `dbs_tcp` в отдельном окне.

`-LOG [уровень трассировки]`

Задает уровень выдаваемой трассировочной информации в файл `dbs_tcp.log`.

Уровни детализации по умолчанию: ERROR, SYSERROR, INIT, INFO, CLIENT.

Все доступные уровни детализации: TRACE, ERROR, CONNECT, SYSERROR, INIT, INFO, CLIENT, DEFAULT, NODETAB, RETCODE, SYSINFO, PACKET, DBG, LINTER.

Значение 0xFFFFFFFF соответствует максимальному уровню детализации.

Задание уровней производится аналогично заданию уровней для управляющей программы системы резервирования (см. документ «Система резервирования», раздел «Протоколирование работы сервера резервирования (debug)»).

`-DEBUG [уровень трассировки]`

Задает уровень выдаваемой отладочной информации в файл `dbs_tcp.log`.

Уровни детализации по умолчанию: ERROR, SYSERROR, INIT, INFO, CLIENT, LINTER, SYSINFO.

Все доступные уровни детализации: TRACE, ERROR, CONNECT, SYSERROR, INIT, INFO, CLIENT, DEFAULT, NODETAB, RETCODE, SYSINFO, PACKET, DBG, LINTER.

Значение 0xFFFFFFFF соответствует максимальному уровню детализации.

Задание уровней производится аналогично заданию уровней для управляющей программы системы резервирования (см. документ «Система резервирования», раздел «Протоколирование работы сервера резервирования (debug)»).

Ключи командной строки ОС Linux, ЗОСРВ Нейтрино

Драйвер обрабатывает следующие специфичные для ОС Linux, ЗОСРВ Нейтрино ключи:

`-D`

Флаг автоматического запуска драйвера из системной сетевой программы `inetd`. Ключ задается в файле настройки `inetd.conf`. Структура файла `inetd.conf` приведена в приложении 3.

`-K[=<сигнал>]`

Заставляет драйвер посылать указанный <сигнал> родительскому процессу по окончании своей инициализации. Значение <сигнала> должно быть целым положительным числом. В случае отсутствия аргумента по умолчанию посылается сигнал SIGTERM.

`-C`

Запрещает переводить процесс драйвера в фоновый режим. Драйвер в этом случае не освобождает консоль до своего завершения.

`-S`

Заставляет использовать дочерний процесс при каждом новом соединении.

`-WD`

Устанавливает режим контроля над работой драйвера. В этом режиме создается специальный («следящий») процесс, который контролирует основной процесс `dbs_tcp`. При зависании основного процесса производится его автоматический перезапуск. Обнаружение зависания происходит с запаздыванием в 10 сек. после того, как оно фактически произошло.

После окончания работы основного процесса `dbs_tcp` «следящий» процесс также завершает свою работу в течение 10 сек.

При завершении «следящего» процесса основной процесс тоже завершается.

Ключи командной строки ОС Windows

Драйвер обрабатывает следующие специфичные для ОС Windows ключи:

`-K=<имя события>`

Задает событие, которое будет установлено по окончании инициализации.

`-PPID=<идентификатор процесса>`

Задает идентификатор процесса, за которым следит `dbs_tcp`. При завершении этого процесса завершится и `dbs_tcp`.

`-SERVICE`

Запуск `dbs_tcp`, как сервиса. См. пункт «Запуск драйвера».

`-MUTEX=<имя>`

Создает мутекс с заданным именем. Мутекс может использоваться для проверки существования процесса. Ключ должен использоваться одновременно с ключом -service.

Выполнение драйвера

Драйвер работает в автоматическом режиме.

 © Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

Настройка защищенного соединения со стороны сервера

Чтобы установить защищенное соединение между сетевым клиентом и сетевым ЛИНТЕР-сервером необходимо в файле сетевой конфигурации `nodetab` вместо протокола обмена TCP/IP указать протокол TCP/IPS или TLS. Все остальные поля файла `nodetab` для протокола TCP/IPS и TLS имеют ту же функциональность, что и для протокола TCP/IP. Если используется протокол защищённого соединения, то сначала будет установлено обычное соединение по протоколу TCP/IP, а затем по специальному запросу сетевого драйвера клиента это соединение переведется в защищенный режим. По умолчанию сетевой драйвер сервера одновременно может работать как с обычными TCP/IP протоколами, так и с защищенными.

Драйвер сервера (`dbs_tcp`) будет стараться загружать и использовать системные библиотеки `openssl`. При невозможности этого будут использованы библиотеки из соответствующих исполняемых файлов, или защищенные протоколы будут отключены, если вариант сборки соответствующих исполняемых файлов не содержит статически встроенных библиотек `openssl`.

В файле `nodetab` можно задавать две строки с различными именами ЛИНТЕР-сервера и протоколами соединений, но с идентичными значениями сетевого адреса и порта. В этом случае возможно попеременное как защищенное, так и незащищенное соединение с одним и тем же сетевым ЛИНТЕР-сервером. На стороне сетевого сервера для обеспечения работоспособности данного режима нет необходимости предпринимать какие-либо дополнительные действия.

По умолчанию для защищенного режима используются ключи длиной 512 бит.

Ключ и сертификат по умолчанию автоматически генерируются при запуске сетевого драйвера ЛИНТЕР-сервера (во время генерации на экране отображаются точки) и сохраняются в оперативной памяти до завершения работы сетевого драйвера сервера. В случае наличия в текущем каталоге запуска сетевого драйвера сервера файлов `dbs_tcp.key` и `dbs_tcp.crt` (см. пункт «Запуск драйвера») стадия генерации ключей пропускается, ключ и сертификат будут взяты из указанных файлов. Файлы ключа и сертификата должны соответствовать друг другу.

Файлы ключа и сертификата генерируются с использованием соответствующей утилиты генерации ключей операционной системы (`openssl`). Сертификаты могут быть подписаны удостоверяющим центром.

Сетевой ЛИНТЕР-сервер также позволяет сгенерировать пару «ключ-сертификат». Для этого сетевой драйвер сервера необходимо запустить с ключом `-SSLKEY` (одновременно можно задать длину ключа в битах). Продолжительность генерации ключа и сертификата лежит в пределах от нескольких секунд до нескольких минут (в зависимости от заданной длины ключа и мощности процессора). После окончания генерации ключ и сертификат будут сохранены в текущем каталоге в файлах `dbs_tcp.key` и `dbs_tcp.crt` соответственно. При последующем запуске из этого же каталога сетевого драйвера сервера будут использованы ключи и сертификат, запомненные в указанных файлах. Файлы `dbs_tcp.key` и `dbs_tcp.crt` будут доступны для чтения и записи только их владельцу. Можно увеличить длину используемого ключа до 1024 или 2048 бит для повышения степени защиты.

Созданный таким образом сертификат является самоподписанным.

Используя ключ `-SSLONLY`, можно отказаться от обмена с сетевым ЛИНТЕР-сервером по незащищенному каналу. В этом случае все попытки обращения к сетевому ЛИНТЕР-

серверу без предварительного перевода соединения в защищенный режим приведут к разрыву соединения. При выборе ключа достаточной длины это обеспечивает практически полную защиту сетевого соединения от постороннего доступа, необходимо только тщательно контролировать доступ к файлам сертификата и (особенно) ключа.

Если файл сертификата будет:

- скопирован с серверного компьютера на клиентский в текущий каталог сетевого драйвера клиента;
- переименован таким образом, чтобы его имя (в верхнем регистре) совпало с именем ЛИНТЕР-сервера, определенным в файле `nodetab`;
- иметь расширение `.crt`

то сетевой драйвер клиента при установке защищенного соединения сравнит сохраненный сертификат и сертификат сервера и, в случае их несоответствия, закроет соединение. Это обеспечит невозможность соединения при подмене адреса сетевого ЛИНТЕР-сервера. Злоумышленник также должен будет получить файл ключа, хранящийся на сервере, поэтому этот файл должен быть защищен от доступа посторонних лиц средствами операционной системы. В данном режиме также невозможно будет установить соединение с сетевым ЛИНТЕР-сервером, который сгенерирует ключ и сертификат при запуске, или ключ которого не будет соответствовать сертификату, хранимому на стороне клиента. Клиент, не имеющий сертификата, сохраняет возможность открыть защищенное соединение с ЛИНТЕР-сервером. Если имеется файл `dbc_tcp.CA`, то будет осуществлена проверка сертификата удостоверяющими центрами из этого списка. Существует возможность проверки у клиента наличия соответствующего сертификата (для этого сетевой драйвер сервера необходимо запустить с ключами `SSLAUTH` и `SSLONLY`).

Проверка выполняется следующим образом:

- если задан ключ `SSLAUTH` и существуют файлы ключа и сертификата (`dbs_tcp.key` и `dbs_tcp.crt` соответственно) в текущем каталоге, то сетевой драйвер сервера (`dbs_tcp`) при установке с ним сетевого соединения запрашивает сертификат сетевого драйвера клиента;
- сетевой драйвер клиента читает свой ключ и сертификат из файлов `dbs_tcp.key` и `dbs_tcp.crt` текущего каталога драйвера сервера;
- сетевой драйвер клиента передает свой сертификат драйверу сервера по его запросу;
- драйвер сервера пытается прочитать сертификат данного клиента из файла, имеющего имя, соответствующее сетевому адресу данного клиента в точечной нотации (например, для адреса клиента `127.0.0.1` имя файла должно быть `127.0.0.1.crt`);
- если сертификат сетевого клиента отсутствует или не соответствует образцу, хранимому на сервере, защищенное сетевое соединение установлено не будет;
- в случае если соединение устанавливается в первый раз, и данный драйвер клиента имеет файлы ключа и сертификата, то соединение будет разрешено. При этом автоматически будет создан файл сертификата, соответствующий сетевому адресу данного клиента;
- если другой клиент попытается соединиться с того же сетевого адреса, или кто-то подменит сетевой адрес, то выявится несовпадение сертификатов, и такое соединение будет отвергнуто.

Если существуют файлы сертификатов удостоверяющих центров `dbs_tcp.CA`, то дополнительно будет проверена подпись сертификата `dbs_tcp.crt` удостоверяющим

центром, что гарантирует подлинность сертификата. Также производится проверка отозванных сертификатов, сохраненных в файле `dbs_tcp.CRL`.

Если необходимо исключить подключение к ЛИНТЕР-серверу неизвестных клиентов, то следует добавить ключ `SSLNOCREAT` при запуске сетевого драйвера сервера. Это исключит возможность соединения с ЛИНТЕР-сервером тех клиентов, для которых у сетевого драйвера сервера нет файлов сертификата. Таким образом, администратор СУБД путем запрета создания файлов сертификатов может обеспечить защищенные соединения с ограниченным кругом клиентов. Файлы сертификатов служат «ключами» открытия соединения и должны тщательно оберегаться как на серверной, так и на клиентской стороне.

Драйвер клиента

Назначение драйвера

Драйвер клиента предназначен для обслуживания запросов клиентских приложений к локальному/удаленному ЛИНТЕР-серверу.

Условия выполнения

Для работы драйвера в ОС должен быть установлен протокол TCP/IP.

При работе драйвер сервера использует следующие переменные окружения (если они определены):

1) **PATH** – системная переменная окружения, используется для поиска файла сетевой конфигурации (`nodetab`);
2) **NET_MBX** – переменная аналогична по своему назначению переменной `LINTER_MBX`. `NET_MBX` определяет «имя почтового ящика» для обмена данными между приложением и сетевым драйвером клиента;
3) **DBC_LOG** – переменная, определяющая уровень трассировки, имя и каталог файла протоколирования работы драйвера клиента. Уровень трассировки вводится в шестнадцатеричном формате. Для полной трассировки значение уровня трассировки – 0xFFFFFFFF. Например: DBC_LOG=0xFFFFFFFF:dbc_tcp.log

Запуск драйвера

Возможны следующие варианты запуска сетевого драйвера клиента:

- ручной запуск исполняемого файла драйвера средствами ОС, как службы ОС;
- запуск исполняемого файла драйвера из командной строки, как консольного приложения;
- запуск в среде ОС Windows, как сервиса с помощью инструментального средства «Сетевой администратор».

Команда запуска драйвера:

```
dbc_tcp [<ключ> …]
<ключ>::= <вариант 1> | <вариант 2>
<вариант 1>::= -<код ключа>{<пробел> | = }<значение ключа>
<вариант 2>::= /<код ключа>=<значение ключа>
```

Драйвер клиента должен запускаться на компьютере, где будет функционировать клиентское приложение. Драйвер нужно запускать обязательно до запуска приложения.

Общие ключи командной строки

`-H | -HELP | ?`

Подсказка по всем ключам драйвера.

Пример.

```
dbc_tcp -h
[-H][-HELP][-?] - this help screen
[-N Path]
[-NODETAB Path] - path to \"nodetab\" file - configuration file of
 dbc_tcp
[-S][-DEFAULT]  - set default server name
[-M ExchangeId] - set interprocess communication identifier
…
```

`-S|-DEFAULT=<условное имя компьютера>`

Задает условное имя компьютера по умолчанию с установленной на нем СУБД ЛИНТЕР. Если ключ не задан, то компьютером по умолчанию считается первый из компьютеров, определенный в файле сетевой конфигурации `nodetab`.

Примечание
Ключ DEFAULT является синонимом ключа S.

Примеры.

```
dbc_tcp -S Serv1
dbc_tcp -S=Serv2
dbc_tcp -default "Bank"
```

`-N|-NODETAB=<спецификация файла nodetab>`

Задает местоположение файла `nodetab` на клиентском компьютере.

Поиск драйвером файла конфигурации выполняется по следующему алгоритму:

- если в командной строке запуска драйвера задан ключ `/N`, то используется заданный им файл конфигурации;
- если значение ключа реестра, указывающего на файл конфигурации, не пусто, то используется этот файл конфигурации (в ОС Windows). Ключ HKEY_CURRENT_USER \Software\Relex, Inc.\Linter Network\Client\5.x\ "Nodetab path" при запуске как приложения или HKEY_LOCAL_MACHINE\Software\Relex, Inc.\Linter Network\Client\5.x\ "Nodetab path" при запуске как сервиса;
- в текущем каталоге драйвера клиента ищется файл с именем `nodetab`;
- в каталоге исполняемого модуля драйвера клиента ищется файл с именем `nodetab`;
- в пути, указанном в переменной окружения HOME, ищется файл с именем `nodetab`;
- в каталогах, задаваемых переменной окружения PATH, ищется файл с именем `nodetab`.

Пример.

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

```
>dbc_tcp -n /path/to/file/nodetab
```

где /path/to/file/nodetab – путь к файлу nodetab.

-NS <строки файла nodetab>

Значение опции имеет тот же формат, что и строка файла nodetab. Для разделения записей необходимо использовать последовательность из 2 символов \n. Это позволяет указывать драйверу сетевую конфигурацию без создания файла конфигурации nodetab.

Пример

```
1)  dbc_tcp -ns "AAA TCPIP localhost"
задает 1 узел AAA

2)  dbc_tcp -ns "`echo -e "AAA TCPIP localhost\nBBB TCPIP olegn
 1060 1 20 20"`"
задает 2 узла AAA и BBB

3) dbc_tcp -ns "S1 TCPIP 127.0.0.1 1060 1 30 30 \n S2 TCPIP
 172.17.2.25 1060 1 30 30"
```

-M=<параметр> | -NAME=<параметр>

Задает идентификатор межпроцессного обмена (аналог переменной окружения NET_MBX).

-A | -AUTOCHANGE

Задает режим автоматического выбора другого ЛИНТЕР-сервера по умолчанию с уведомлением об этом клиентского приложения.

Если попытка соединения с явно указанным ЛИНТЕР-сервером по умолчанию (ключ -S или -DEFAULT) была неудачной, то клиентскому приложению возвращается соответствующий код завершения (ошибка установления соединения или истечение тайм-аута соединения), после чего драйвер автоматически выберет в качестве ЛИНТЕР-сервера по умолчанию следующий ЛИНТЕР-сервер из файла сетевой конфигурации nodetab. При последующем запросе приложения к ЛИНТЕР-серверу по умолчанию будет осуществлена попытка соединиться с вновь выбранным ЛИНТЕР-сервером. Смена ЛИНТЕР-сервера по умолчанию будет производиться до осуществления удачного сетевого соединения с одним из ЛИНТЕР-серверов из файла nodetab.

В случае если кандидатом на ЛИНТЕР-сервер по умолчанию был выбран последний ЛИНТЕР-сервер из файла nodetab, и соединение с ним также оказалось неудачным, то следующим будет назначен первый ЛИНТЕР-сервер файла nodetab.

Примечание
Если клиентское приложение при установке соединения с БД посылает несколько запросов (как, например, утилита inl – команды OPEN и DESC), то в случае невозможности соединения с ЛИНТЕР-сервером по умолчанию в режиме AUTOCHANGE эти запросы будут посланы разным ЛИНТЕР-серверам.

-G | -AUTOCONNECT

Задает режим автоматического выбора другого ЛИНТЕР-сервера по умолчанию без уведомления об этом клиентского приложения.

Если попытка соединения с явно указанным ЛИНТЕР-сервером по умолчанию (ключ -S или -DEFAULT) была неудачной в течение тайм-аута, то клиентскому приложению не будет возвращен соответствующий код завершения (ошибка установления соединения или истечение тайм-аута соединения). Драйвер автоматически выберет в качестве нового ЛИНТЕР-сервера по умолчанию следующий ЛИНТЕР-сервер из файла сетевой конфигурации nodetab и инициирует соединение с ним. Если соединение с очередным ЛИНТЕР-сервером окажется неудачным, процесс продолжится до исчерпания списка ЛИНТЕР-серверов в файле nodetab.

Клиентское приложение получит соответствующий код завершения только в случае невозможности установить соединение со всеми ЛИНТЕР-серверами, перечисленными в файле nodetab. Если значение переменной окружения DBC_LOG установлено в 1, то в стандартный поток ошибок будет выводиться информация о процессах установления и разрыва соединений, о смене соединения по умолчанию.

Примечание

Параметры AUTOCONNECT и AUTOCHANGE являются взаимно исключающими. При одновременном задании обоих параметров будет использован AUTOCONNECT.

`-AUTOCONNECTTIMEOUT=<тайм-аут циклического соединения>`

Задает тайм-аут циклического соединения, то есть интервал времени (сек.), в течение которого драйвер будет самостоятельно пытаться установить соединение с хотя бы одним ЛИНТЕР-сервером, перечисленным в файле nodetab.

Ключ AUTOCONNECTTIMEOUT имеет смысл только при совместном использовании с ключом AUTOCONNECT.

В режиме AUTOCONNECT (без ключа AUTOCONNECTTIMEOUT) клиентское приложение, использующее ЛИНТЕР-сервер по умолчанию, получит код завершения неудачного соединения с ЛИНТЕР-сервером сразу после каждой неудачной попытки установить соединение с ЛИНТЕР-серверами, перечисленными в файле nodetab.

В случае использования ключа AUTOCONNECT одновременно с ключом AUTOCONNECTTIMEOUT код завершения неудачного соединения с ЛИНТЕР-сервером будет возвращен клиентскому приложению только после указанного в ключе интервала времени, в течение которого драйвер будет производить периодические попытки соединиться с каким-либо ЛИНТЕР-сервером из перечисленных в файле nodetab.

Если в течение заданного интервала не будет установлено ни одно соединение, драйвер вернет клиентскому приложению код завершения 4006 (или 1001, если задан ключ ERR1001).

Если установить тайм-аут циклического соединения небольшим, то будет произведен только один цикл соединений, и задержка будет определяться суммарным временем попыток соединения с каждым из ЛИНТЕР-серверов (так же, как и при работе без ключа AUTOCONNECTTIMEOUT).

Если тайм-аут для конкретного соединения с каким-либо ЛИНТЕР-сервером (в файле nodetab) задать небольшим, то при разрыве сетевого соединения общее время от передачи клиентским приложением запроса сетевому серверу до возврата ошибки приложению будет определяться ключом AUTOCONNECTTIMEOUT или суммой времен тайм-аутов соединений всех ЛИНТЕР-серверов файла nodetab. Если же тайм-аут соединения в nodetab не задан, то скорость реакции на неудачное соединение будет определяться особенностями сетевого протокола ОС, но все равно будет суммой времен, потраченных на попытку

соединения с каждым из ЛИНТЕР-серверов. Также необходимо учитывать, что тайм-аут не может быть менее двух секунд (реально минимальный тайм-аут устанавливается в интервале от 2 до 4 сек.). Это относится ко всем тайм-аутам `nodetab` и к ключу `AUTOCONNECTTIMEOUT`.

Таким образом, при задании минимальных тайм-аутов соединения (как в `nodetab`, так и в ключе `AUTOCONNECTTIMEOUT`) в 2 сек. и ключа `-G` реальный тайм-аут будет равен 2 – 4 сек., умноженным на количество ЛИНТЕР-серверов в файле `nodetab`. Задать тайм-аут 0 сек. нельзя, так как это значение зарезервировано для исключения тайм-аута клиента и сервера.

`-CONNECTONLOAD`

Заставляет драйвер начать немедленное установление соединения со всеми ЛИНТЕР-серверами сразу после запуска (даже если запросов на соединение от клиентских приложений не поступало) и держать открытыми установленные ранее соединения. Первый из ЛИНТЕР-серверов, с которым будет установлено соединение, назначается сервером по умолчанию.

В обычном режиме установление соединения с ЛИНТЕР-сервером (по умолчанию или указанным явно) инициируется только после получения от клиентского приложения запроса на соединение, и, соответственно, все ранее установленные соединения закрываются, если от клиентских приложений в течение примерно 5 мин. не поступило ни одного запроса к СУБД.

Если задан ключ `CONNECTONLOAD`, то сетевые соединения будут держаться открытыми до выгрузки драйвера.

`-CONNERROR`

Заставляет драйвер возвращать немедленно код завершения в случае отсутствия соединения с заданным ЛИНТЕР-сервером. В обычном режиме драйвер ожидает установления соединения и приема ответа от ЛИНТЕР-сервера или ошибки соединения, либо истечения тайм-аута соединения прежде, чем вернуть соответствующий код завершения клиентскому приложению. Если задан ключ `CONNERROR`, то код завершения будет возвращен клиентскому приложению немедленно. Это может быть полезно, например, в режиме автоматического соединения с ЛИНТЕР-сервером по умолчанию, когда приложение может ждать ответа достаточно долго. Если же запрос от клиентского приложения поступил после установления соединения, то работа с приложением будет проходить в обычном режиме.

`-P|-PIDFILE=<спецификация файла>`

Записывает идентификатор процесса (PID) драйвера клиента в текстовый файл.

Если заданный файл существует, он будет перезаписан.

Файл будет уничтожен при нормальном завершении работы драйвера.



Примечание
Информация о PID драйвера в текстовом файле требуется, как правило, при обработке командных файлов (например, для того, чтобы завершить работу драйвера).

`-PPID <значение PID>`

Задает слежение за функционированием процесса с указанным PID. В среде ОС Linux, ЗОСРВ Нейтрино также указывает PID процесса, которому будет посылаться сигнал об успешной инициализации драйвера (см. опцию PKILL).

Примечание
В ОС Linux, ЗОСРВ Нейтрино изменяет PID процесса для опции PCHECK.

`-PING`

Проверка активности удаленного ЛИНТЕР-сервера по умолчанию.

Драйвер клиента выводит на консоль сообщение об активности или неактивности ЛИНТЕР-сервера по умолчанию и завершает свою работу.

При активном ЛИНТЕР-сервере по умолчанию код возврата равен 0, в противном случае возвращается код завершения, который получило бы клиентское приложение в данной ситуации. Коды возврата могут быть использованы для проверки активности удаленного ЛИНТЕР-сервера в командном файле.

Для проверки активности произвольного ЛИНТЕР-сервера данный ключ должен применяться совместно с ключом `S`.

Примечание
Если в файле конфигурации `nodetab` для протокола TCP/IP тайм-аут соединения не задан, то он будет определяться операционной системой и может оказаться очень большим, что может повлиять на длительность работы драйвера, запущенного с ключом `PING`.

`-VERSION`

Вывод на консоль информации о версии драйвера клиента.

`-INTERACTIVE={0|1}`

Управляет режимом взаимодействия с драйвером:

- 0: интерактивный режим выключен и в случае ошибки инициализации драйвер завершается немедленно;
- 1: интерактивный режим включен и в случае ошибки инициализации выводится просьба нажать клавишу ENTER для завершения работы драйвера.

Примечание
В ОС Windows интерактивный режим отключается автоматически при запуске `dbc_tcp` как сервиса или в фоновом режиме и интерактивный режим автоматически включается при запуске `dbc_tcp` в отдельном окне.

`-LOG [уровень трассировки]`

Задает уровень выдаваемой трассировочной информации в файл `dbc_tcp.log`.

Все доступные уровни детализации: TRACE, ERROR, CONNECT, SYSERROR, INIT, INFO, CLIENT, DEFAULT, NODETAB, RETCODE, SYSINFO, PACKET, DBG.

Уровни детализации по умолчанию: ERROR, CONNECT, SYSERROR, INFO, INIT, CLIENT, DEFAULT, NODETAB, RETCODE.

Значение 0xFFFFFFFF соответствует максимальному уровню детализации.

 © Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

Задание уровней производится аналогично заданию уровней для управляющей программы системы резервирования (см. документ «Система резервирования», раздел «Протоколирование работы сервера резервирования (debug)»).

`-DEBUG [уровень трассировки]`

Задает уровень выдаваемой отладочной информации в файл `dbc_tcp.log`.

Все доступные уровни детализации: TRACE, ERROR, CONNECT, SYSERROR, INIT, INFO, CLIENT, DEFAULT, NODETAB, RETCODE, SYSINFO, PACKET, DBG.

Уровни детализации по умолчанию: ERROR, CONNECT, SYSERROR, INFO, INIT, CLIENT, DEFAULT, NODETAB, RETCODE, SYSINFO.

Значение 0xFFFFFFFF соответствует максимальному уровню детализации.

Задание уровней производится аналогично заданию уровней для управляющей программы системы резервирования (см. документ «Система резервирования», раздел «Протоколирование работы сервера резервирования (debug)»).

Ключи командной строки ОС Linux, ЗОСРВ Нейтрино

`-K|-PKILL[=<сигнал>]`

Заставляет драйвер посылать указанный <сигнал> родительскому процессу по окончании своей инициализации. Значение <сигнала> должно быть целым положительным числом. В случае отсутствия значения аргумента по умолчанию посылается сигнал SIGTERM.

`-C|-PCHECK`

Заставляет драйвер периодически проверять существование родительского процесса и прекращать свою работу после его завершения. Максимальный промежуток времени между завершением работы родительского процесса и завершением работы самого драйвера в этом случае составляет 10 сек.

`-NODAEMON`

Запрещает переводить процесс драйвера в фоновый режим. Драйвер в этом случае не освобождает консоль до своего завершения.

`-ERR1001`

Заставляет драйвер в режиме AUTOCONNECT возвращать код завершения 1001 («ЛИНТЕР-сервер не найден») в случае, если все попытки соединения со всеми ЛИНТЕР-серверами из файла `nodetab` закончились неудачно.

В обычном режиме в этом случае возвращается ошибка соединения.

Пример

```
dbc_tcp -ppid 1241 -interactive 0
```

`-U`

Завершение работы драйвера (драйверу передается сигнал TERM, инициирующий завершение работы).

Примечания

1. Только для ОС Linux, ЗОСРВ Нейтрино.

2. Если в файле конфигурации nodetab для протокола TCP/IP тайм-аут соединения не задан, то он будет определяться операционной системой и может оказаться очень большим, что может повлиять на длительность работы драйвера, запущенного с ключом PING.

Драйвер работает в автоматическом режиме. В процессе работы драйвер использует управляющую информацию из указанного при запуске файла сетевой конфигурации nodetab.

Ключи командной строки ОС Windows

`-service`

Ключ регистрозависимый.

Если приложение запускается как сервис, то ключ должен быть указан, если как обычное приложение – не должен добавляться. Запуск приложения как сервиса означает, что приложение регистрируется в ОС как сервис и может управляться как сервис, в этом случае приложение рапортует менеджеру сервисов о своем состоянии.

`-MUTEX=<имя мутекса>`

Ключ регистрозависимый.

При старте приложения как сервиса (с ключом -service) будет создан мутекс с указанным именем. При завершении процесса этот мутекс уничтожается в случае, если он не открыт другими процессами. Это позволяет отслеживать состояние сервиса другими процессами.

`-K|-PKILL=<сигнал>`

Значение <сигнал> – это имя события, которое драйвер dbc_tcp установит по окончании своей успешной инициализации.

Примечание
Событие должно существовать до запуска dbc_tcp.

Настройка защищенного соединения со стороны клиента

При указании в файле сетевой конфигурации nodetab протокола TCPIPS или TLS соединение будет осуществляться в защищенном режиме. Этот режим исключает получение информации, хранящейся в БД ЛИНТЕР, путем прослушивания сетевого соединения. Все остальные параметры соединения аналогичны соответствующим параметрам протокола TCPIP.

Драйвер клиента dbc_tcp будет стараться загружать и использовать системные библиотеки openssl. При невозможности этого будут использованы библиотеки из соответствующих исполняемых файлов, или защищенные протоколы будут отключены, если вариант сборки соответствующих исполняемых файлов не содержит статически встроенных библиотек openssl.

Если в каталоге запуска драйвера присутствуют файлы dbc_tcp.key (ключ) и dbc_tcp.crt (сертификат), то происходит сравнение открытого ключа маскирования

сервера, считанного из файла сертификата и принятого по сети от сетевого драйвера сервера СУБД ЛИНТЕР. В случае несовпадения значений соединение не устанавливается. Если файл сертификата сервера в текущем каталоге не найден, он будет создан автоматически. Файл сертификата сервера имеет имя, совпадающее с именем ЛИНТЕР-сервера из файла `nodetab`, и расширение `.crt`. Имя ЛИНТЕР-сервера в файле приводится к верхнему регистру. Администратор СУБД может перенести файл `dbs_tcp.crt` на клиентскую сторону и соответствующим образом переименовать его до установления первого соединения с сервером. Наличие файлов `dbc_tcp.crt` и `dbc_tcp.key` (сертификата и ключа маскирования клиента) позволяет сетевому драйверу сервера получить сертификат клиента и произвести сравнение с запомненным ранее сертификатом.

Создать файлы ключей маскирования клиента можно, запустив `dbs_tcp` с ключом `-SSLKEY` и переименовав полученные файлы `dbs_tcp.key` и `dbs_tcp.crt` в `dbc_tcp.key` и `dbc_tcp.crt` соответственно. При этом файлы должны быть защищены средствами ОС. В этом случае создается самоподписанные сертификаты.

Средствами ОС могут быть созданы сертификаты с заданным алгоритмом. Эти сертификаты могут быть подписаны удостоверяющим центром. Проверка подписи удостоверяющего центра включается при наличии файла `dbc_tcp.CA` – списка сертификатов удостоверяющих центов. Отозванные сертификаты должны храниться в файле `dbc_tcp.CRL` (приложение 4).

При инициализации защищенного соединения сначала устанавливается обычное соединение по протоколу TCP/IP, затем оно переводится в защищенный режим. Существуют три режима защищенного соединения:

1) без аутентификации сервера;
2) аутентификация с сохранением ключей в автоматическом режиме;
3) аутентификация с сохранением ключей в ручном режиме.

Если драйвер клиента запускается стандартным образом, без манипулирования файлами `dbc_tcp.key` и `dbc_tcp.crt`, аутентификация сервера не проводится, но защищенное соединение будет установлено.

Если перенести сертификаты с серверного компьютера на клиентский и сохранить их в каталоге запуска драйвера клиента с именем, аналогичным имени ЛИНТЕР-сервера из файла `nodetab`, и с расширением `.crt`, то при установке соединения будет проверено соответствие запомненного сертификата клиента и сертификата сервера. В случае их несоответствия соединение установлено не будет.

Серверные сертификаты могут быть также получены от удостоверяющего цента. Для включения проверки подписи удостоверяющего цента должен присутствовать файл списка сертификатов удостоверяющих центров `dbc_tcp.CA`.

Если в каталоге запуска драйвера клиента создать файлы ключа и сертификата клиента с именами `dbc_tcp.key` и `dbc_tcp.crt` соответственно, это позволит:

- передать сертификат клиента серверу для аутентификации клиента сервером;
- включить режим автоматического сохранения сертификата сервера в текущем каталоге при установке соединения с данным ЛИНТЕР-сервером в первый раз.

При установке последующих соединений с данным ЛИНТЕР-сервером будет производиться сверка сертификатов с запомненным при первом соединении.

Файлы ключей и сертификатов должны быть защищены от доступа посторонних лиц. Это достигается сменой прав доступа на 0600 средствами ОС Linux, ЗОСРВ Нейтрино. Сертификат и ключ могут быть созданы утилитой `openssl` (в состав дистрибутива СУБД ЛИНТЕР не входит) с именами `dbc_tcp.crt` и `dbc_tcp.key` соответственно. Также самоподписанный сертификат и ключ можно создать утилитой `dbs_tcp` с опцией `SSLKEY`. В этом случае файлы должны быть переименованы из `dbs_tcp.crt` в `dbc_tcp.crt` и `dbs_tcp.key` в `dbc_tcp.key`.

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

Графические сетевые средства для ОС Windows

Графический драйвер клиента

Примечание
Поддержка программы остановлена, использовать не рекомендуется.

Запуск графического сетевого драйвера клиента

Запуск можно выполнить одним из способов:

- выполнить команду **Пуск => Программы => СУБД ЛИНТЕР => Сетевой клиент**;
- с помощью стандартных средств запуска программ операционной системы запустить на выполнение файл `dbcgui` в подкаталоге `\bin` установочного каталога СУБД ЛИНТЕР.

При первоначальном запуске программы будет выведено окно выбора загружаемой по умолчанию конфигурации (рис. 6).

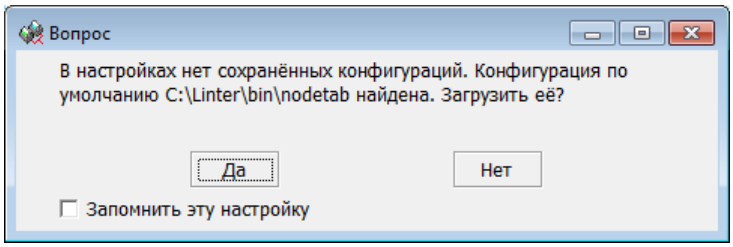


Рисунок 6. Окно выбора загружаемой по умолчанию конфигурации

Основное окно программы

Основное окно программы приведено на рисунке 7.

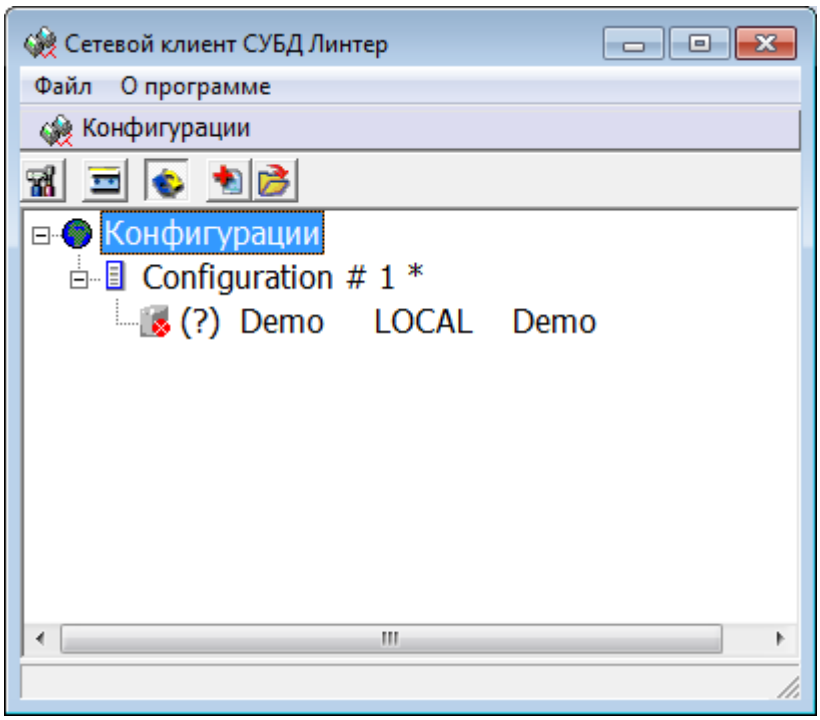


Рисунок 7. Основное окно программы Сетевой клиент

Состав основного окна программы:

- главное меню;
- панель инструментов;
- область отображения конфигураций в древовидной структуре.

Древовидная структура конфигураций содержит следующие типы узлов:

- корневой элемент конфигураций;
- выбранная конфигурация (загруженный файл сетевой конфигурации `nodetab`);
- выбранный сервер.

Состав операций главного меню и панели инструментов зависит от типа выбранного узла.

В таблице 1 приведен перечень кнопок панели инструментов.

Таблица 1. Кнопки панели инструментов

Кнопка	Назначение
	Настройки шрифтов и форматирования текста программы
	Включение протоколирования
	Обновление указанного поддерева
	Добавление узла указанного типа
	Открытие файла конфигурации
	Запуск программы при запуске приложения

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

Кнопка	Назначение
	Запуск драйвера клиента
	Останов драйвера клиента
	Сохранение измененной конфигурации
	Сохранение новой или измененной конфигурации
	Удаление указанного узла дерева конфигураций
	Просмотр статистики работы серверов конфигурации
	Сортировка списка серверов указанной конфигурации в прямом порядке
	Сортировка списка серверов указанной конфигурации в обратном порядке
	Соединение с сервером
	Разъединение с сервером
	Запрещение/разрешение сервера

Работа программы

Создание новой конфигурации

Окно создания конфигурации можно вызвать одним из способов:

- в главном меню выполнить **Файл=>Создать конфигурацию**;
- в контекстном меню узла типа **Конфигурации** выбрать пункт **Создать конфигурацию**;
- одновременно нажать клавиши <Ctrl>+<N>;
- выбрав узел типа **Конфигурации**, на панели инструментов нажать кнопку .

Окно конфигурации таблицы узлов состоит из двух вкладок:

- вкладка **Серверы** (рис. 8);
- вкладка **Параметры** (рис. 9);

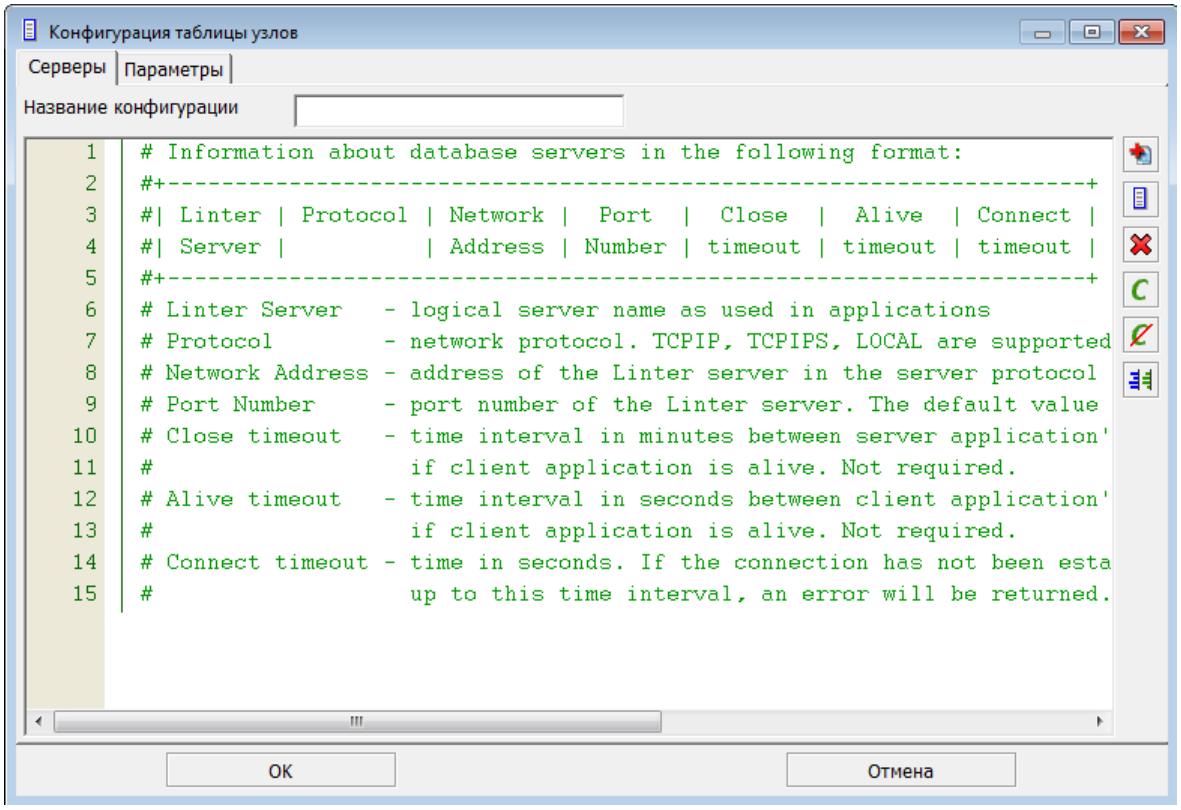


Рисунок 8. Окно конфигурации таблицы узлов, вкладка Серверы

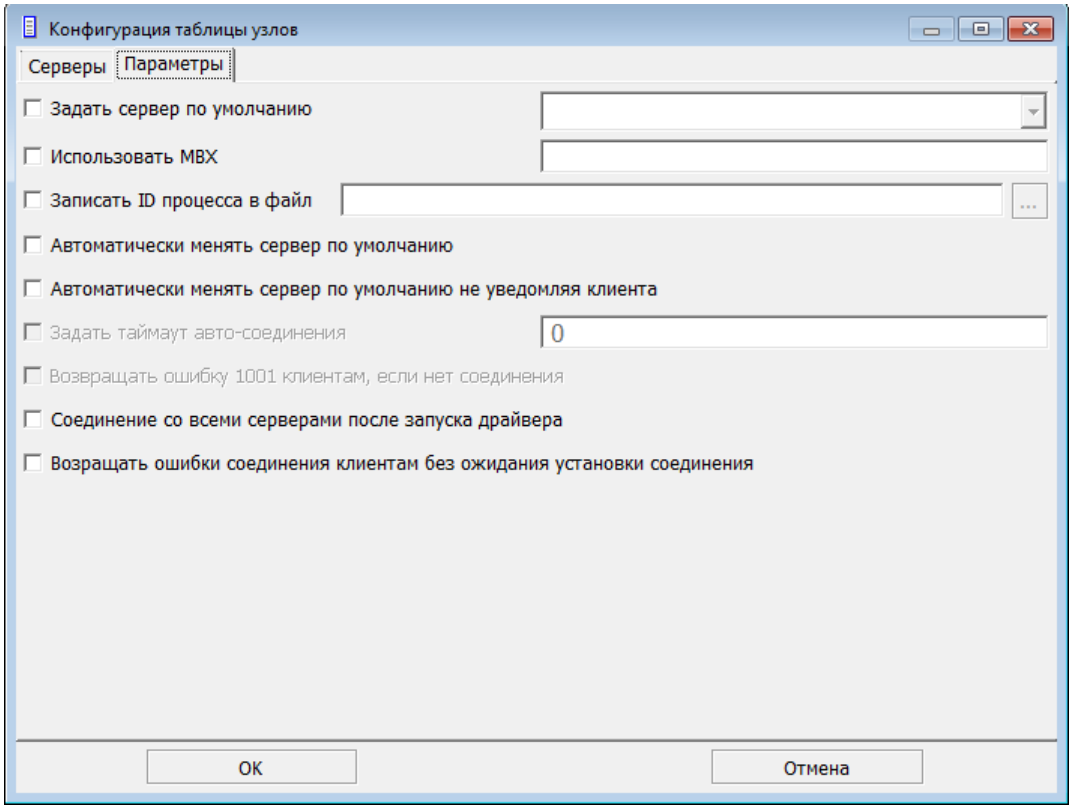


Рисунок 9. Окно конфигурации таблицы узлов, вкладка Параметры

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

Вкладка **Серверы** содержит следующие элементы:

- поле ввода названия конфигурации (для сохранения ее в файл);
- область редактирования конфигурации;
- панель инструментов создания и редактирования конфигурации.

В таблице 2 приведен перечень кнопок панели инструментов создания и редактирования конфигурации.

Таблица 2. Кнопки панели инструментов создания и редактирования конфигурации

Кнопка	Назначение
	Добавление сервера
	Редактирование параметров сервера
	Удаление сервера
	Закомментировать указанную запись сервера
	Раскомментировать текущую запись сервера
	Просмотр статистики работы сервера

Создание и редактирование сетевой конфигурации возможно двумя способами:

- непосредственное указание параметров серверов в окне конфигурации;
- с помощью панели инструментов.

По окончании ввода параметров конфигурации для создания новой конфигурации нажать кнопку **ОК**, для выхода из диалога создания конфигурации без создания – **Отмена**.

Загрузка файла существующей конфигурации

Для загрузки существующего файла сетевой конфигурации необходимо:

1) вызвать окно выбора загружаемого файла конфигурации.

   Открыть окно выбора загружаемого файла конфигурации можно одним из способов:
   - в главном меню выполнить **Файл=>Открыть файл**;
   - в контекстном меню узла типа **Конфигурации** выбрать пункт **Открыть файл**;
   - одновременно нажать клавиши <Ctrl>+<O>;
   - на панели инструментов нажать кнопку .
2) выбрать требуемый файл и нажать кнопку **Открыть**.

В дереве элементов узла типа **Конфигурации** будет создано поддерево загруженной конфигурации.

Сохранение конфигурации

Для сохранения конфигурации в файл необходимо:

1) выбрать сохраняемую конфигурацию;

2) выполнить одно из действий:
   - в главном меню выполнить **Таблица узлов=>Сохранить (Сохранить как …)**;
   - в контекстном меню узла типа **Конфигурация** выбрать пункт **Сохранить (Сохранить как …)**;
   - на панели инструментов нажать кнопку или .

Примечание
Если сохраняется существующая конфигурация – выбрать пункт **Сохранить**, если производится сохранение созданной конфигурации – **Сохранить как …**

Просмотр статистики серверов конфигурации

Для просмотра статистики работы серверов необходимо:

1) выбрать необходимую конфигурацию;
2) выполнить одно из действий:
   - в главном меню выполнить **Таблица узлов=>Статистика**;
   - в контекстном меню узла типа **Конфигурация** выбрать пункт **Статистика**;
   - на панели инструментов нажать кнопку .

Появится окно статистики работы серверов конфигурации (рис. 10).

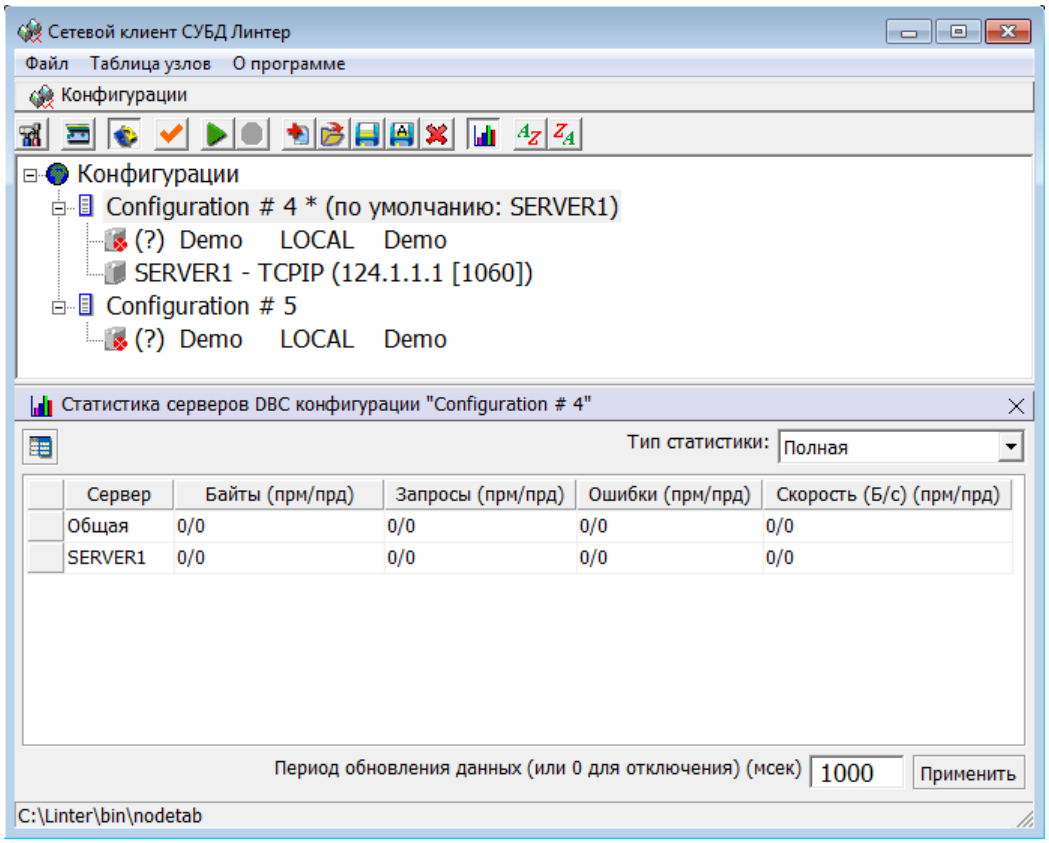


Рисунок 10. Окно статистики работы серверов конфигурации

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

Основными элементами окна статистики являются:

- выпадающий список выбора типа отображаемой статистики: полная или текущая;
- кнопка выбора отображаемых параметров (рис. 10);
- область отображения статистических данных работы серверов.
- отображаемые параметры (рис. 11):
  1) имя сервера;
  2) количество переданной и принятой информации;
  3) запросы;
  4) ошибки;
  5) скорость.
- поле установки периода обновления данных;
- строка отображения расположения файла сетевой конфигурации.

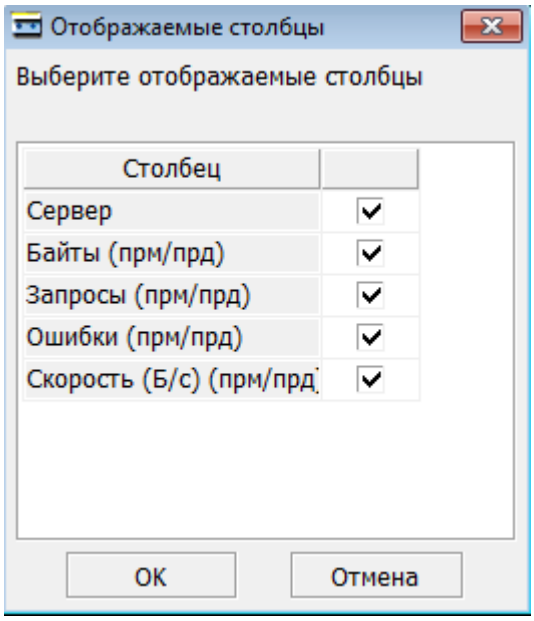


Рисунок 11. Отображаемые столбцы

Сортировка списка серверов конфигурации

Возможны следующие варианты сортировки серверов:

- в прямом порядке;
- в обратном порядке;
- отсутствует.

Для сортировки серверов конфигурации необходимо:

1) выбрать необходимую конфигурацию;
2) выполнить одно из действий:
   - в главном меню выполнить **Таблица узлов=>Сортировка**;

- в контекстном меню узла типа **Конфигурация** выбрать пункт **Сортировка**;
- на панели инструментов узла типа **Конфигурация** нажать требуемую кнопку  или ;
- выбрать вариант сортировки.

Добавление сервера

Для добавления сервера необходимо:

1) выбрав необходимую конфигурацию, вызвать окно ввода параметров сервера.

   Окно ввода параметров сервера можно одним из способов:
   - в главном меню выполнить **Таблица узлов=>Добавить сервер**;
   - в контекстном меню узла типа **Конфигурация** выбрать пункт **Добавить сервер**;
   - на панели инструментов узла типа **Конфигурация** нажать кнопку .
2) в появившемся окне ввода и редактирования параметров сервера (рис. 12) ввести необходимые параметры.

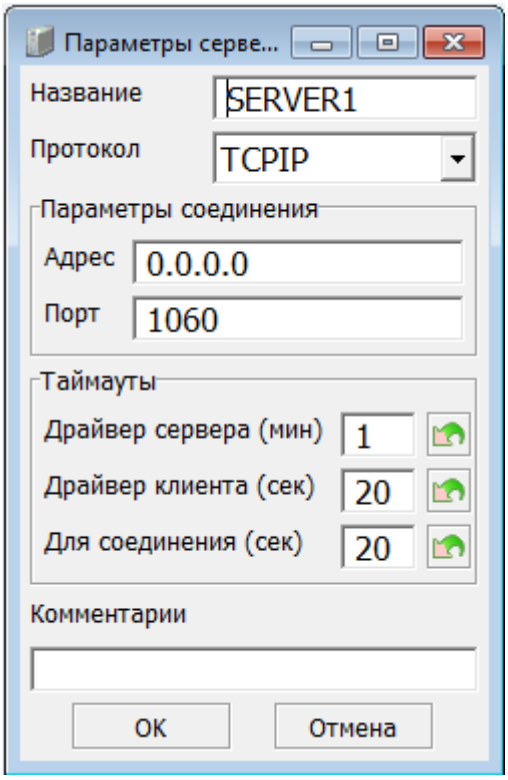


Рисунок 12. Окно параметров сервера

В окне параметров сервера в зависимости от типа протокола необходимо ввести требуемые параметры:

- название сервера (не более 8 символов);
- протокол обмена;
- сетевой адрес или сетевое имя;

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

- таймаут драйвера сервера;
- таймаут драйвера клиента;
- таймаут для соединения.

3) для добавления нового сервера и выхода из диалогового окна нажать кнопку **ОК**, для выхода из диалогового окна – **Отмена**.

Редактирование параметров сервера

Для редактирования параметров сервера необходимо:

1) выбрав редактируемый сервер в древовидной структуре конфигурации, вызвать окно редактирования параметров сервера.

   Активизировать окно редактирования параметров сервера можно одним из способов:

   - в главном меню выполнить **Сервер=>Редактировать сервер**;
   - в контекстном меню узла типа **Конфигурация** выбрать пункт **Редактировать сервер**.

2) в появившемся окне редактирования (рис. 12) ввести требуемые параметры аналогично пункту «Добавление сервера».

Удаление сервера

Для удаления сервера из конфигурации необходимо:

1) выбрать удаляемый сервер в древовидной структуре серверов;
2) удалить выбранный сервер из конфигурации одним из способов:
   - в главном меню выполнить **Сервер=>Удалить сервер**;
   - в контекстном меню узла типа **Сервер** выбрать пункт **Удалить сервер**;
   - на панели инструментов нажать кнопку .

Установка сервера по умолчанию

Для установки сервера по умолчанию необходимо:

1) выбрать требуемый сервер в древовидной структуре конфигурации;
2) выполнить одно из действий:
   - в главном меню выполнить **Сервер=>Установить по умолчанию**;
   - в контекстном меню узла типа **Сервер** выбрать пункт **Установить по умолчанию**.

Запуск клиента

Для запуска сетевого клиента с указанной конфигурацией необходимо:

1) выбрать требуемую конфигурацию;
2) выполнить одно из действий:
   - в главном меню выполнить **Таблица узлов=>Запустить клиент**;

- в контекстном меню узла типа **Таблица узлов** выбрать пункт **Запустить клиент**;
- на панели инструментов узла типа **Конфигурация** нажать кнопку .

После запуска сетевого клиента узел выбранной загруженной конфигурации будет отображен зеленым цветом и будет загружен сервер, установленный по умолчанию (рис. 13).

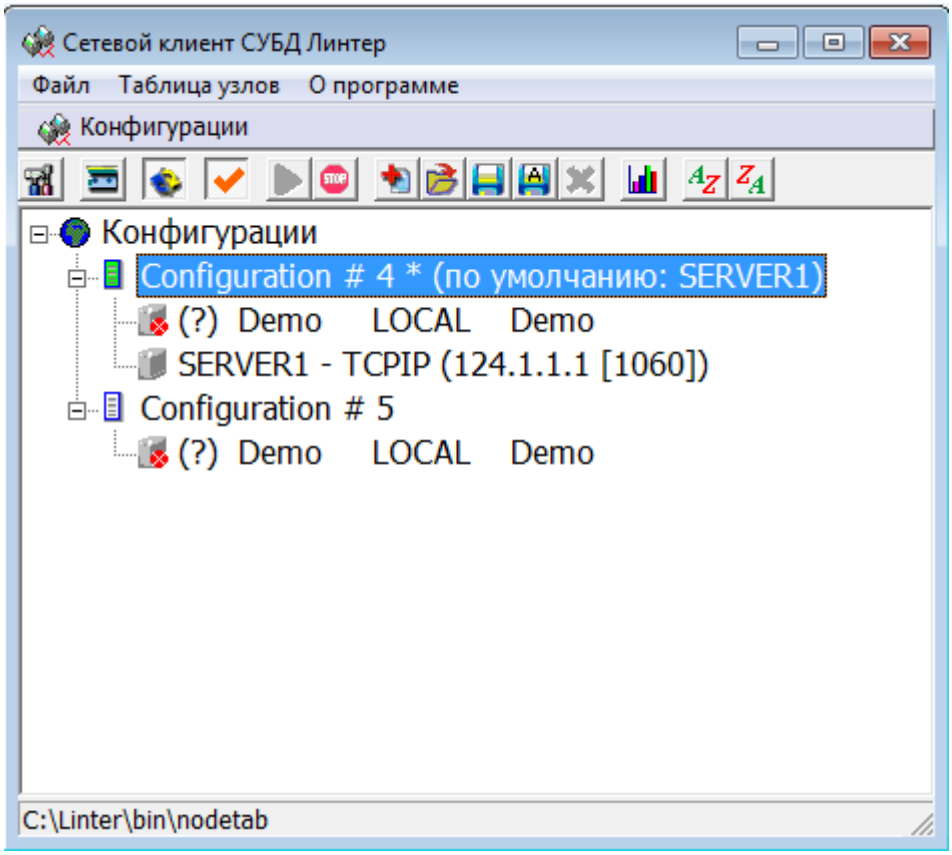


Рисунок 13. Окно программы, Сервер по умолчанию

Останов клиента

Для останова сетевого клиента необходимо:

1) выбрать активную конфигурацию;
2) выполнить одно из действий:
   - в главном меню выполнить **Таблица узлов=>Остановить клиент**;
   - в контекстном меню узла типа **Таблица узлов** выбрать пункт **Остановить клиент**;
   - на панели инструментов узла типа **Конфигурация** нажать кнопку .

Соединение с сервером

Для соединения с сервером необходимо:

1) выбрать сервер активной конфигурации;

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

2) выполнить одно из действий:
   - в главном меню выполнить **Сервер=>Соединить**;
   - в контекстном меню узла типа **Сервер** выбрать пункт **Соединить**;
   - на панели инструментов узла типа **Сервер** нажать кнопку .

После соединения с указанным сервером иконка сервера будет отображена зеленым цветом (рис. 14).

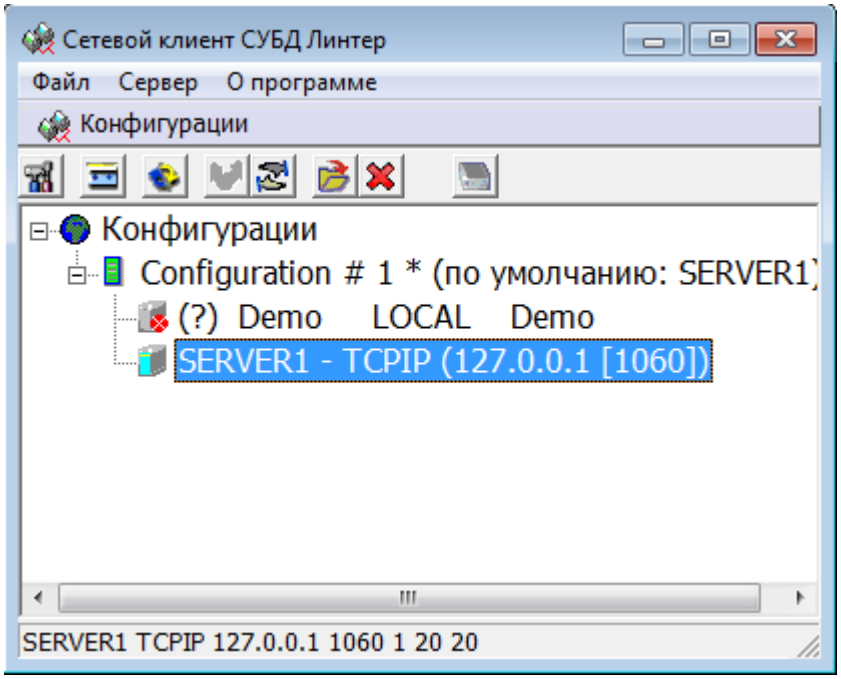


Рисунок 14. Окно программы после запуска сервера

Отсоединение от сервера

Для отсоединения от сервера необходимо:

1) выбрать активный сервер;
2) выполнить одно из действий:
   - в главном меню выполнить **Сервер=>Отсоединить**;
   - в контекстном меню узла типа **Сервер** выбрать пункт **Отсоединить**;
   - на панели инструментов узла типа **Сервер** нажать кнопку .

Запрет сервера

Для запрета сервера необходимо выполнить одно из действий:

1) в главном меню выполнить **Сервер=>Запретить сервер**;
2) в контекстном меню узла типа **Сервер** выбрать пункт **Запретить сервер**;
3) на панели инструментов узла типа **Сервер** нажать кнопку .

Для отмены запрещения сервера выполнить одно из вышеуказанных действий повторно.

Настройка протоколирования

Для вызова окна протоколирования необходимо выполнить одно из действий:

- в главном меню выполнить **Файл=>Протоколирование**;
- на панели инструментов нажать кнопку .

Появится окно настройки и отображения протоколируемых событий (рис. 15).

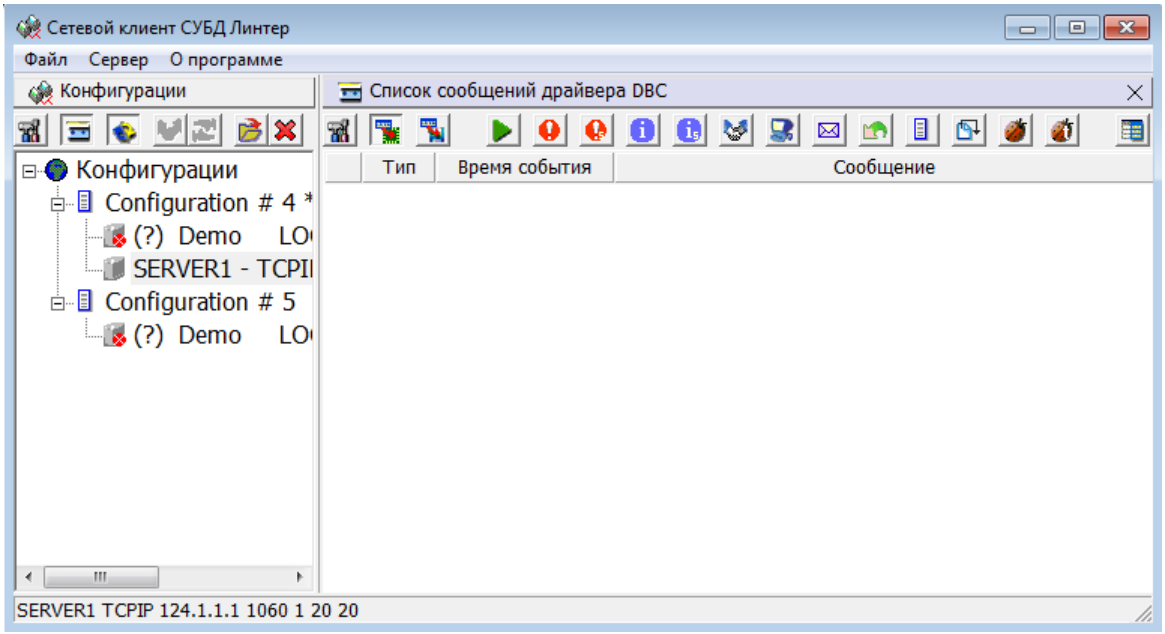


Рисунок 15. Окно настройки и отображения протоколируемых событий

Состав меню окна протоколируемых событий:

- кнопка включения протоколирования в память ();
- кнопка включения протоколирования в файл ();
- панель кнопок выбора протоколируемых событий;
- кнопка вызова окна настроек столбцов отображаемых событий ();
- кнопка вызова окна прочих настроек протоколирования ();
- область отображения событий.

В таблице 3 приведен перечень видов протоколируемых событий.

Таблица 3. Виды протоколируемых событий

Кнопка	Назначение
	Сообщения инициализации
	Сообщения об ошибках
	Сообщения о системных ошибках
	Информационные сообщения
	Информационные сообщения о системе
	Сообщения соединения
	Сообщения клиентов

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

Кнопка	Назначение
	Пакетные сообщения
	Сообщения умолчаний
	Сообщения о конфигурационном файле
	Сообщения о кодах возврата
	Отладочные сообщения
	Сообщения трассировки

Протоколирование в память

Для настройки протоколирования в память необходимо:

1) нажать на панели инструментов области протоколирования кнопку .

   Появится окно настроек протоколирования (рис. 16).

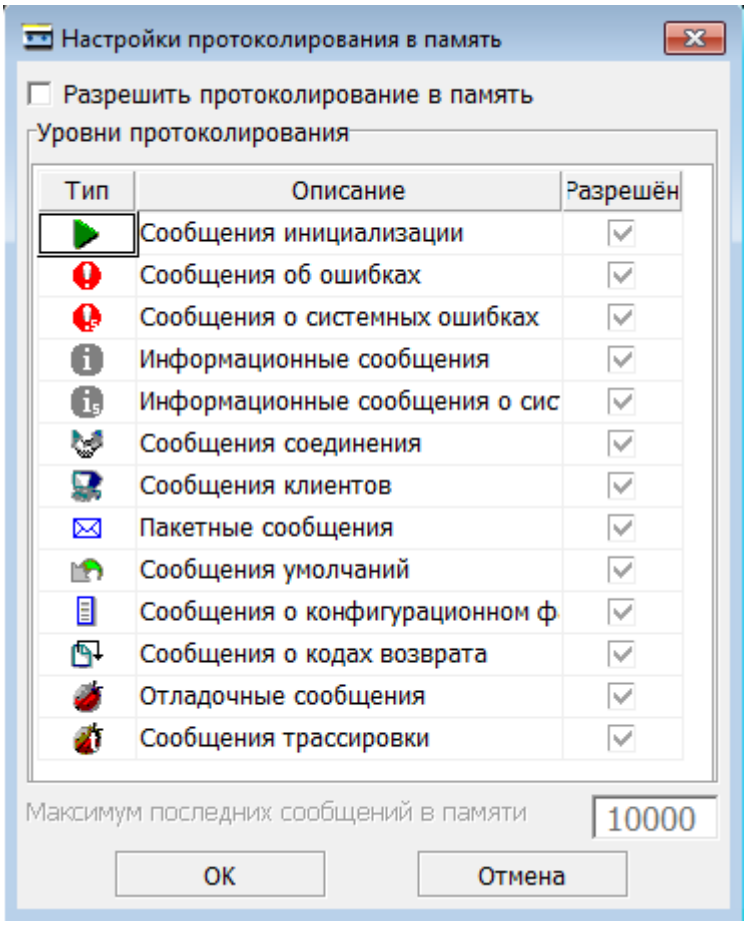


Рисунок 16. Окно настроек протоколирования в память

2) в появившемся окне установить флажок **Разрешить протоколирование в память**;
3) установить флажки напротив типов протоколируемых событий;
4) при необходимости установить максимальное количество последних сообщений в памяти (по умолчанию – 10 000 сообщений);

5) для начала протоколирования в память и выхода их диалогового окна нажать кнопку ОК, для выхода из диалогового окна – Отмена.

Также в контекстом меню области отображения протоколируемых событий доступны операции:

- очистить – очистить область вывода сообщений;
- установить максимум записей – установка максимального количества отображаемых событий протоколирования;
- экспорт лога из памяти в файл – сохранение сообщений протоколирования в файл.

Протоколирование в файл

Для настройки протоколирования в файл необходимо:

1) нажать на панели инструментов области протоколирования кнопку .

Появится окно настроек протоколирования (рис. 17).

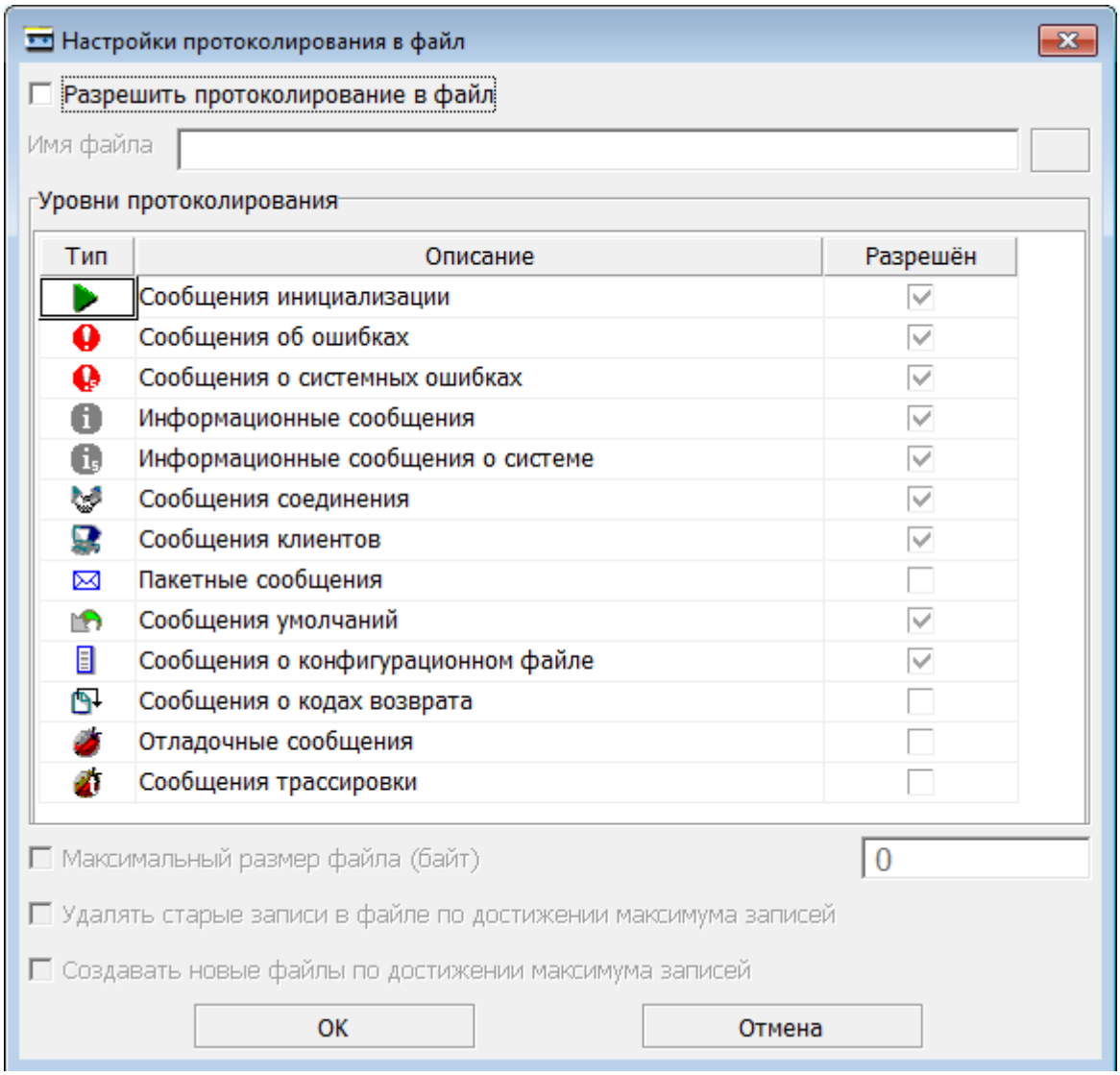


Рисунок 17. Окно настроек протоколирования в файл

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

2) в появившемся окне установить флажок **Разрешить протоколирование в файл**;
3) ввести имя или выбрать существующий файл;
4) установить флажки напротив протоколируемых событий;
5) при необходимости установить:
   - флажок и значение **Максимального размера файла**;
   - флажок **Удаление старых записей в файле при достижении максимума записей**;
   - флажок **Создавать новые файлы по достижении максимума записей**.
6) для начала протоколирования в файл и выхода их диалогового окна нажать кнопку ОК, для выхода из диалогового окна – Отмена.

Общие настройки приложения

Окно настроек программы можно вызвать одним из способов:

- в главном меню выполнить **Файл=>Настройки**;
- нажать клавишу <F10>;
- на панели инструментов нажать кнопку .

Общие настройки программы состоят из трех вкладок:

- вкладка **Общие** – предназначена для настройки шрифтов (рис. 18);

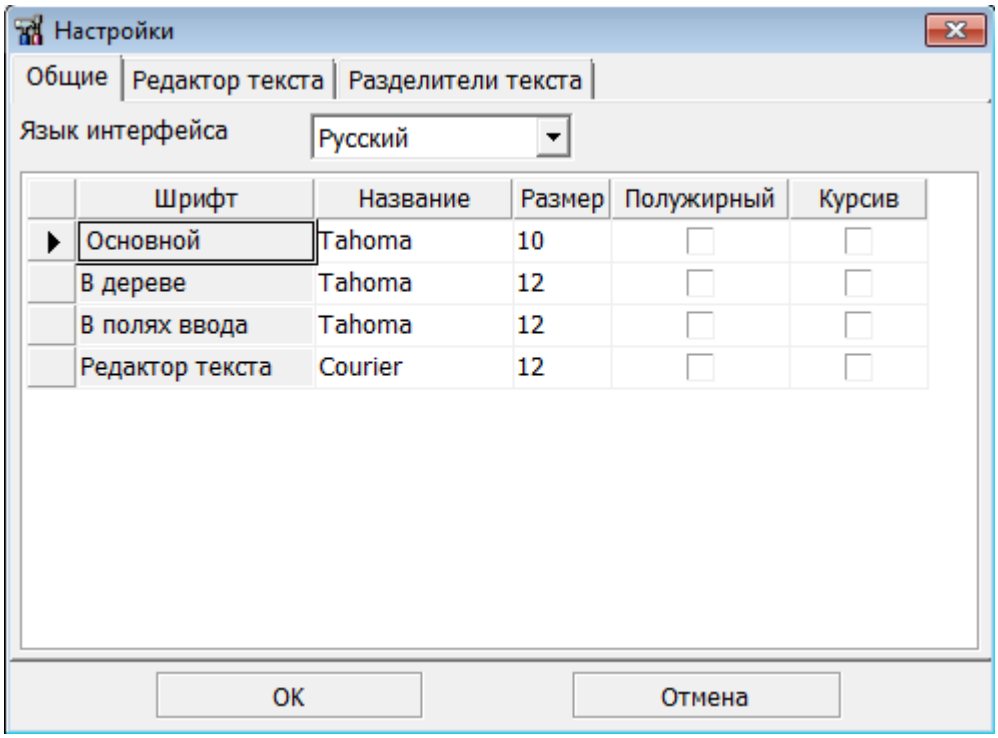


Рисунок 18. Настройки, вкладка Общие

- вкладка **Редактор текста** (рис. 19);

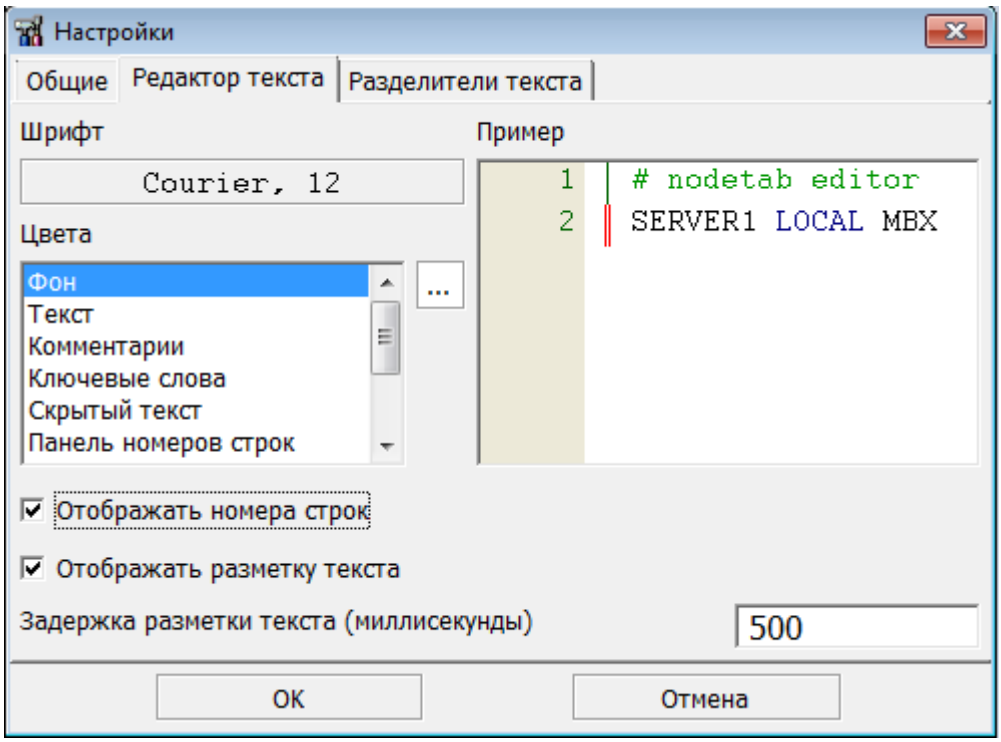


Рисунок 19. Настройки, вкладка Редактор текста

- вкладка **Разделители текста** (рис. 20).

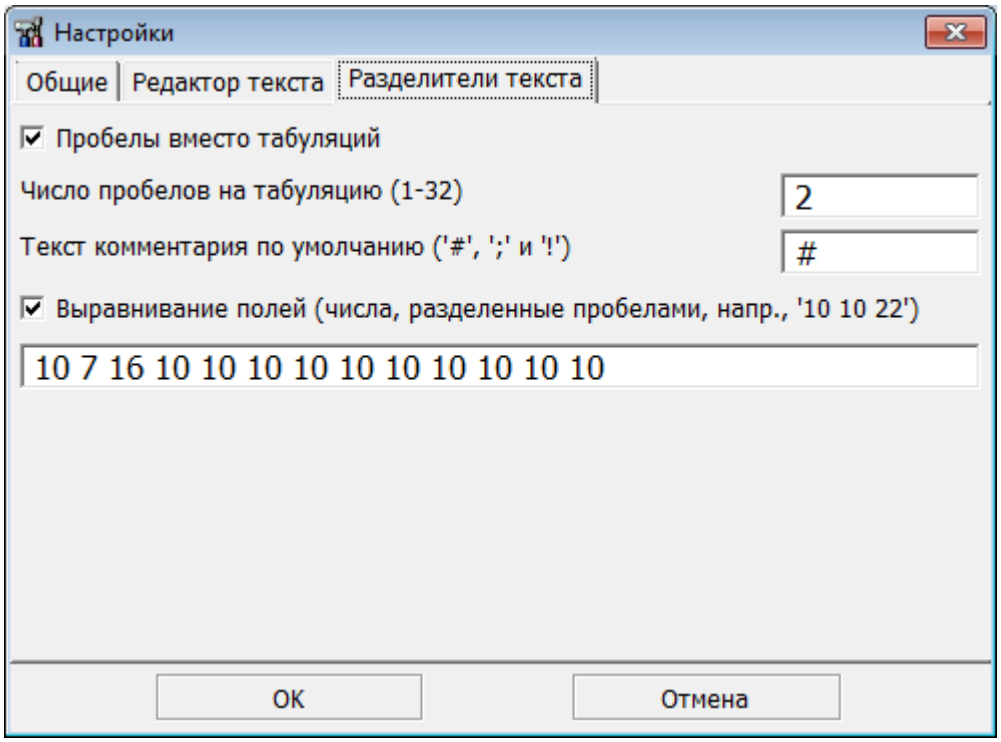


Рисунок 20. Настройки, вкладка Разделители текста

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

Завершение работы программы

Для завершения работы программы необходимо в главном меню выполнить **Файл=>Выход**.

Графический драйвер сервера

Примечание
Поддержка программы остановлена, использовать не рекомендуется.

Запуск графического сетевого драйвера сервера

Запуск можно выполнить одним из способов:

- выполнить команду **Пуск => Программы => СУБД ЛИНТЕР => Сетевой сервер**;
- с помощью стандартных средств запуска программ операционной системы запустить на выполнение файл `dbsgui` в подкаталоге `\bin` установочного каталога СУБД ЛИНТЕР.

Основное окно программы

Основное окно программы приведено на рисунке 21.

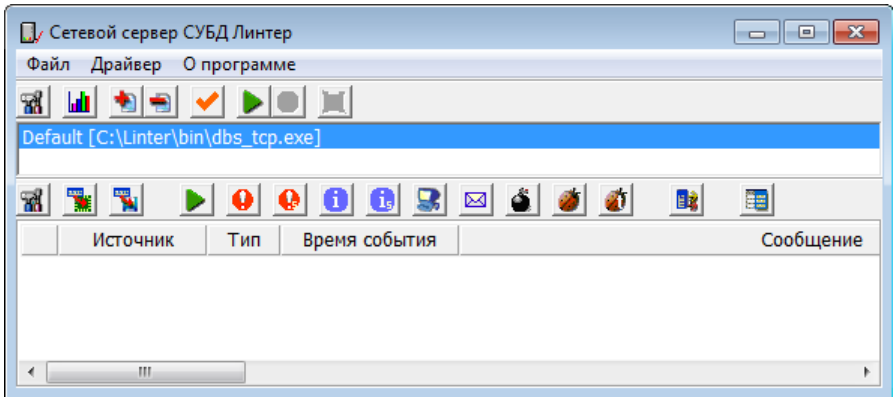


Рисунок 21. Основное окно программы Сетевой сервер

Состав основного окна программы:

- главное меню;
- панель инструментов;
- область отображения доступных конфигураций запуска сетевого драйвера;
- область настройки и отображения сообщений протоколирования.

В таблице 4 приведен перечень кнопок панели инструментов программы Сетевой сервер.

Таблица 4. Кнопки панели инструментов программы Сетевой сервер

Кнопка	Назначение
	Настройки программы
	Просмотр статистики работы сетевого сервера

Кнопка	Назначение
	Добавление драйвера
	Удаление драйвера
	Запуск программы при запуске приложения
	Запуск драйвера
	Останов драйвера
	Отсоединение драйвера

Работа программы

Добавление драйвера

Для добавления конфигурации запуска сетевого сервера необходимо:

1) вызвать окно ввода параметров запуска сетевого сервера.

   Выполнить одно из действий:

   - в главном меню выполнить **Драйвер=>Добавить драйвер**;
   - в контекстном меню области конфигураций сетевых драйверов выбрать пункт **Добавить драйвер**;
   - на панели инструментов нажать кнопку .

2) в появившемся окне параметров запуска драйвера (рис. 22) установить необходимые значения.

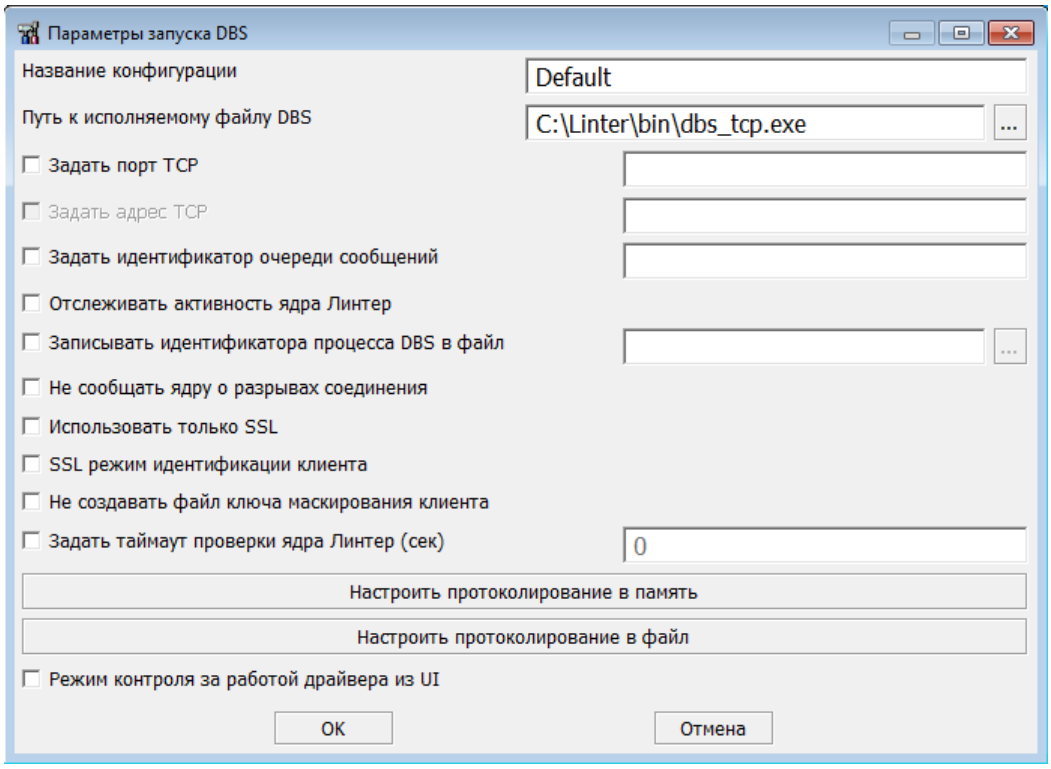


Рисунок 22. Параметры запуска драйвера

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

3) для создания новой конфигурации запуска и выхода из диалогового окна нажать кнопку ОК, для выхода из диалогового окна без создания конфигурации нажать кнопку Отмена.

Редактирование конфигурации запуска драйвера

Для редактирования конфигурации запуска сетевого драйвера сервера необходимо:

1) выбрать в области отображения доступных конфигураций запуска сетевого драйвера редактируемую конфигурацию;
2) вызвать окно редактирования параметров конфигурации.

   Вызвать окно можно одним из способов:

   - в контекстном меню выбранной конфигурации выбрать пункт **Параметры запуска**;
   - два раза кликнув левой кнопкой мыши на редактируемой конфигурации.
3) в появившемся окне редактирования параметров конфигурации (рис. 22) произвести коррекцию параметров;
4) для сохранения изменений конфигурации и выхода из диалогового окна нажать кнопку ОК, для выхода из диалогового окна без сохранения изменений конфигурации нажать кнопку Отмена.

Удаление конфигурации запуска драйвера

Для удаления конфигурации запуска сетевого драйвера сервера необходимо:

1) выбрать в области отображения доступных конфигураций запуска сетевого драйвера удаляемую конфигурацию;
2) выполнить одно из действий:
   - в контекстном меню выбранной конфигурации выбрать пункт **Драйвер => Удалить драйвер**;
   - в контекстном меню выбранной конфигурации выбрать пункт **Удалить драйвер**;
   - на панели инструментов нажать кнопку .

Запуск драйвера при запуске приложения

Для запуска сетевого драйвера с указанными параметрами при запуске приложения необходимо:

1) выбрать в области отображения доступных конфигураций запуска сетевого драйвера необходимую конфигурацию;
2) выполнить одно из действий:
   - в контекстном меню выбранной конфигурации выбрать пункт **Драйвер => Запускать драйвер при запуске приложения**;
   - в контекстном меню выбранной конфигурации выбрать пункт **Запускать драйвер при запуске приложения**;
   - на панели инструментов нажать кнопку .

Запуск драйвера

Для запуска сетевого драйвера с указанными параметрами необходимо:

1) выбрать в области отображения доступных конфигураций запуска сетевого драйвера необходимую конфигурацию;
2) выполнить одно из действий:
   - в контекстном меню выбранной конфигурации выбрать пункт **Драйвер => Запускать драйвер**;
   - в контекстном меню выбранной конфигурации выбрать пункт **Запускать драйвер**;
   - на панели инструментов нажать кнопку .

Останов драйвера

Для останова сетевого драйвера необходимо:

1) выбрать в области отображения доступных конфигураций запуска сетевого драйвера необходимую конфигурацию;
2) выполнить одно из действий:
   - в контекстном меню выбранной конфигурации выбрать пункт **Драйвер => Остановить драйвер**;
   - в контекстном меню выбранной конфигурации выбрать пункт **Остановить драйвер**;
   - на панели инструментов нажать кнопку .

Отсоединение драйвера

Для отсоединения сетевого драйвера необходимо:

1) выбрать в области отображения доступных конфигураций запуска сетевого драйвера необходимую конфигурацию;
2) выполнить одно из действий:
   - в контекстном меню выбранной конфигурации выбрать пункт **Драйвер => Отсоединить драйвер**;
   - в контекстном меню выбранной конфигурации выбрать пункт **Отсоединить драйвер**;
   - на панели инструментов нажать кнопку .

Просмотр статистики

Для просмотра статистики работы сетевого драйвера необходимо выполнить одно из действий:

- в контекстном меню выбранной конфигурации выбрать пункт **Файл => Разрешить статистику**;
- на панели инструментов нажать кнопку .

Появится новая область отображения статистики работы сервера главного окна программы (рис. 23).

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

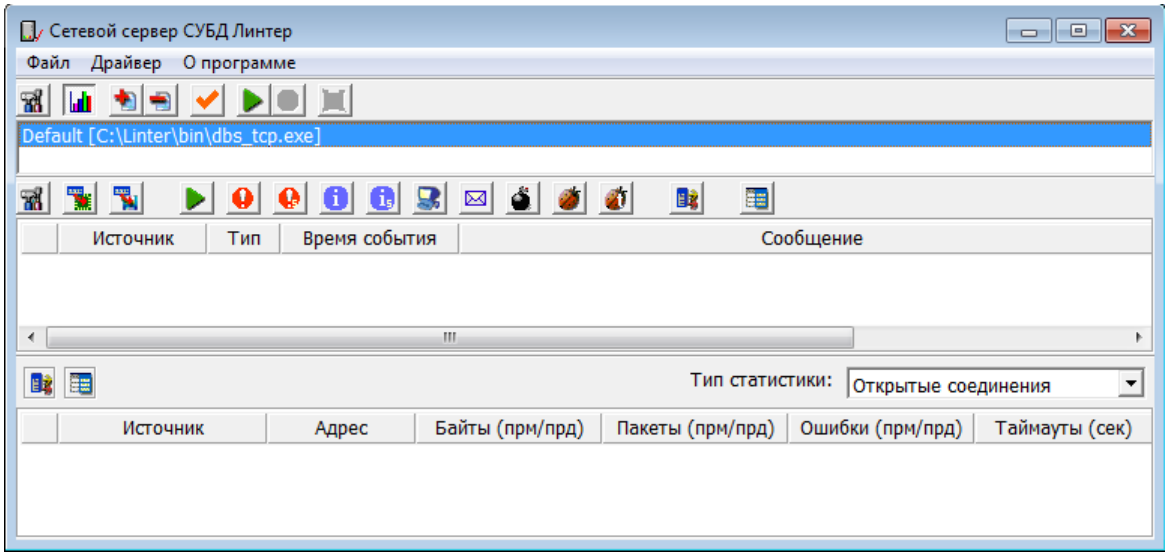


Рисунок 23. Основное окно программы с областью настройки и отображения статистики

Область отображения статистики состоит из следующих элементов:

- список выбора типа статистики: открытые соединения, работа с ядром и история соединений;
- кнопка вызова окна настройки фильтров сообщений ( );
- кнопка вызова окна настройки списка отображаемых столбцов ( );
- область отображения статистических данных.

Окно настройки фильтров представлено на рисунке 24.

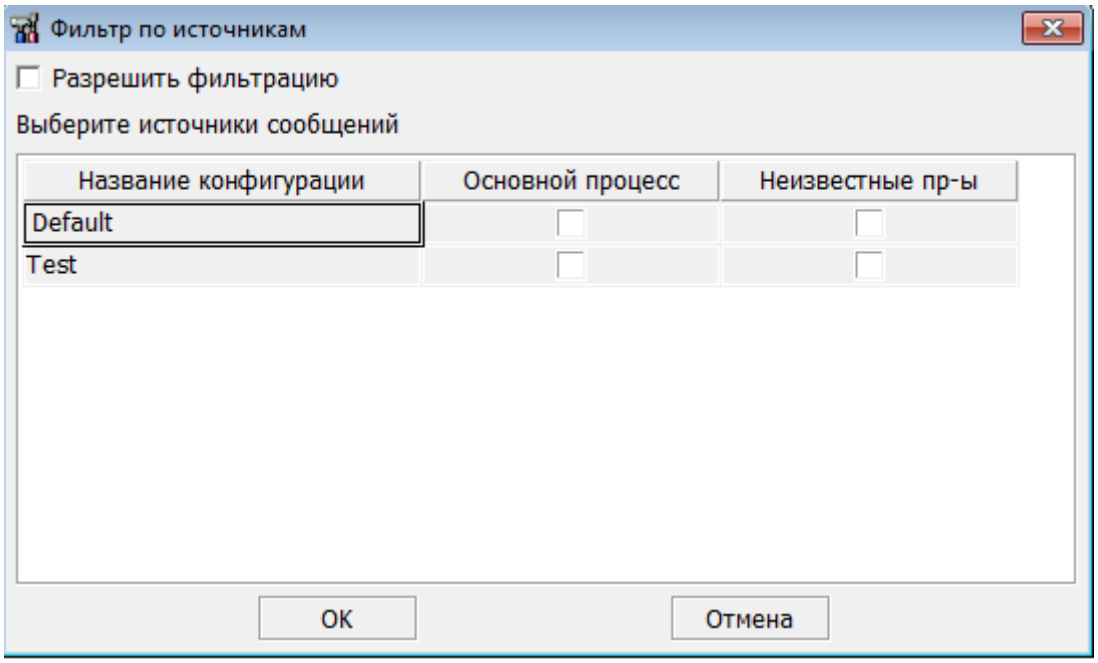


Рисунок 24. Окно настройки фильтров

Окно настройки отображаемых столбцов статистики представлено на рисунке 25.

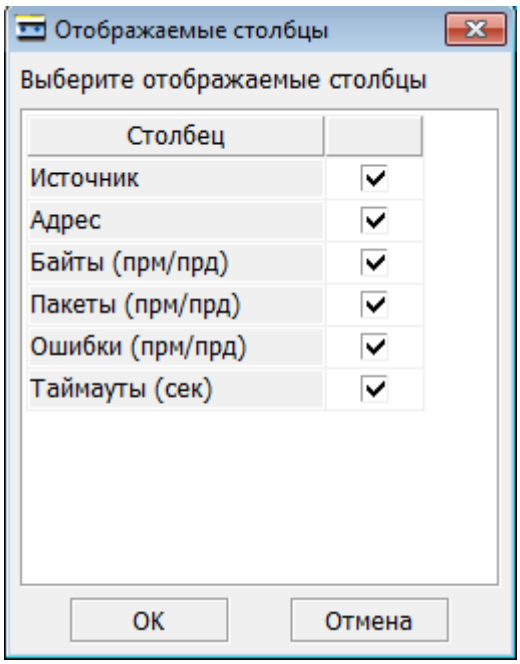


Рисунок 25. Окно настройки столбцов статистики

Настройка протоколирования

Диалоговое окно настройки протоколирования можно вызвать либо из области настройки и отображения сообщений протоколирования, либо с помощью кнопок «Настроить протоколирование в память» и «Настроить протоколирование в файл» диалоговых окон создания и редактирования конфигурации запуска сервера.

Настройка протоколирования производится аналогично пункту «Настройка протоколирования» графического сетевого драйвера клиента.

Настройка программы

Окно настроек программы вызвать одним из способов:

- в главном меню выполнить **Файл=>Настройки**;
- нажать клавишу <F10>;
- на панели инструментов нажать кнопку .

Окно настроек программы состоит из двух вкладок:

- общие – настройки шрифтов (рис. 26);

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

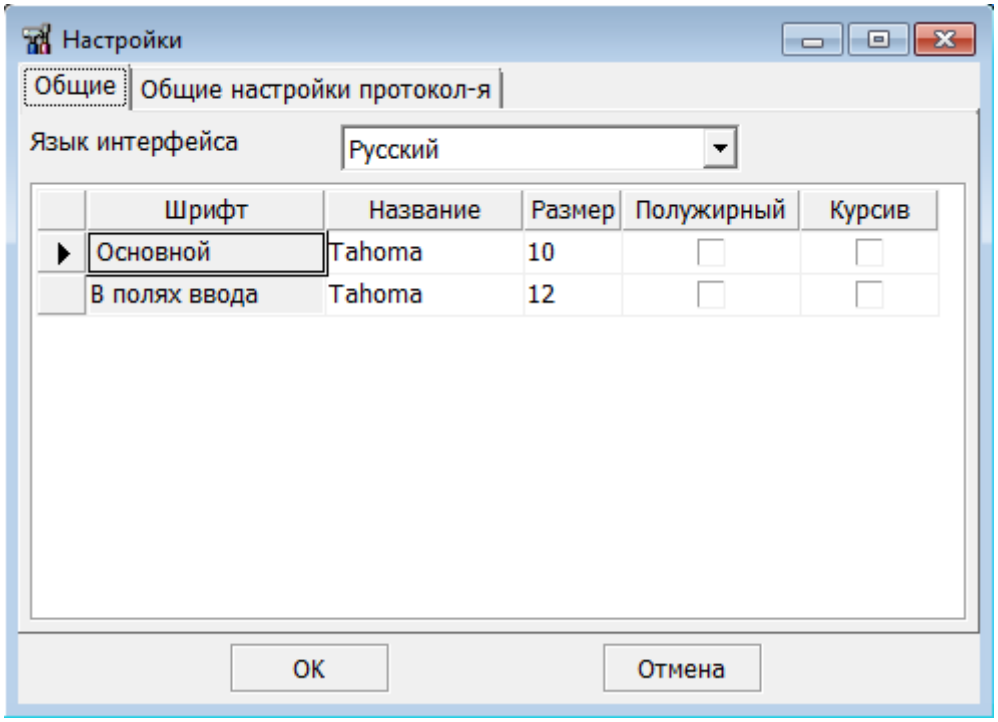


Рисунок 26. Настройки, вкладка Общие

- настройки протоколирования (рис. 27).

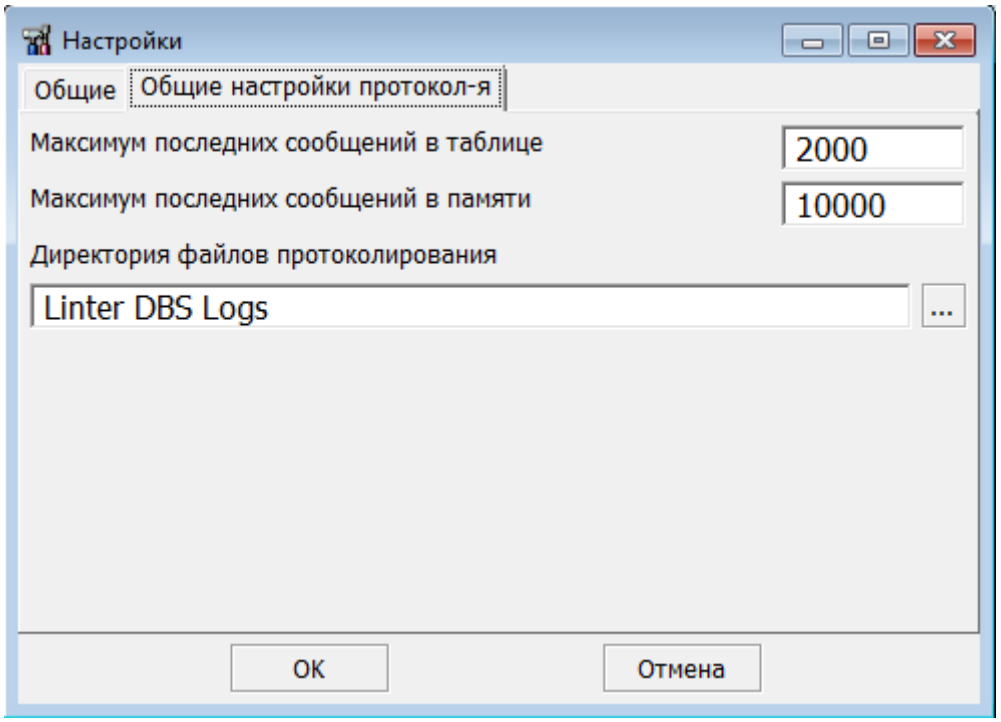


Рисунок 27. Настройки, вкладка Общие настройки протоколирования

Завершение работы программы

Для завершения работы программы необходимо в главном меню выполнить **Файл=>Выход**.

Тестирование конфигурации сети

После создания или модификации файла сетевой конфигурации рекомендуется проверить корректность введенных сетевых параметров.

Для тестирования файла `nodetab` в среде ОС Linux, ЗОСРВ Нейтрино следует:

1) активировать любой ЛИНТЕР-сервер, указанный в файле `nodetab` (запустить на нем драйвер сервера и ядро СУБД ЛИНТЕР);
2) запустить на клиентском компьютере драйвер клиента:

```
dbc_tcp /S <имя сервера> /ping
```

где `<имя сервера>` – имя тестируемого ЛИНТЕР-сервера;

3) повторить операции 1, 2 для всех остальных ЛИНТЕР-серверов файла `nodetab`.

Пример проверки работы сетевого клиента в среде ОС Linux, ЗОСРВ Нейтрино демонстрируется в приложении 5.

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

Коды завершения сетевых средств

Все коды завершения, генерируемые сетевыми драйверами сервера и клиента, передаются клиентскому приложению точно также, как результат обработки клиентского SQL-запроса (см. документ «Справочник кодов завершения»).

Управление сетевым доступом

В этом разделе описана процедура управления сетевым доступом на примере типовой сетевой конфигурации СУБД ЛИНТЕР (рис. 28). Описание дается применительно к среде ОС Linux, ЗОСРВ Нейтрино.

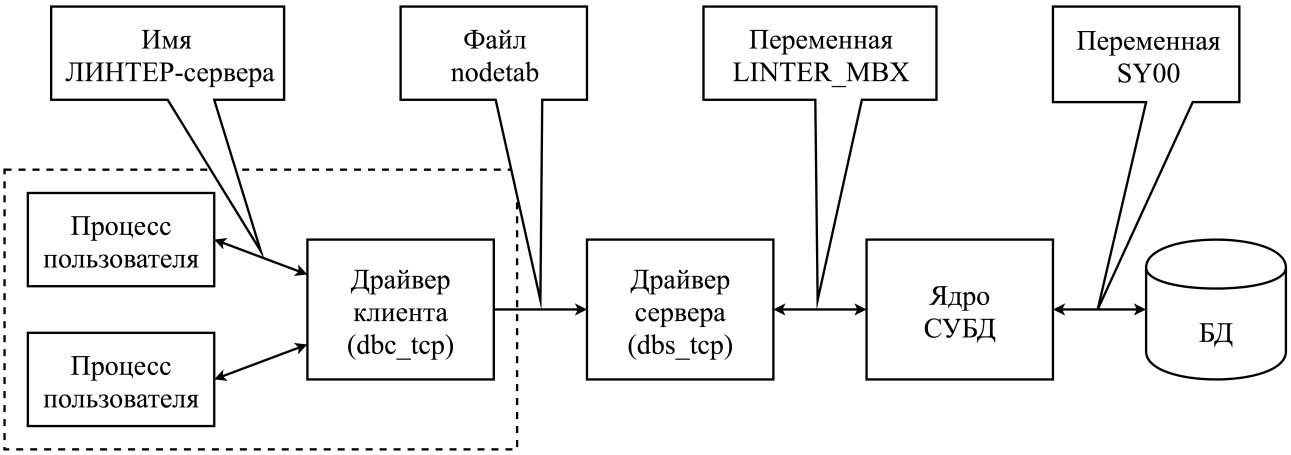


Рисунок 28. Схема доступа к удаленной БД

Примечание
В среде ОС Windows управление сетевым доступом выполняется аналогично с соответствующими поправками (другие имена ключей, идентификаторы межпроцессного обмена).

Для обеспечения сетевого доступа к БД администратор СУБД должен выполнить следующие действия:

1) определить БД на данном ЛИНТЕР-сервере, для которых должен быть разрешен удаленный (сетевой) доступ;
2) для каждой такой БД выполнить процедуру настройки переменных окружения, используемых СУБД ЛИНТЕР в процессе функционирования, и запустить экземпляр ядра СУБД. Ядро СУБД использует переменную SY00 для определения местонахождения файлов БД и переменную `LINTER_MBX` для взаимодействия с сетевым драйвером сервера:
   - указываем местоположение БД: например `SY00=/linter/db/bank; export SY00;`
   - задаем идентификатор межзадачного обмена, например:

```
export LINTER_MBX=30000;
```

   - запускаем на выполнение экземпляр СУБД ЛИНТЕР.

Для первой БД можно не указывать переменную окружения `LINTER_MBX`, а использовать значения по умолчанию.

Примечание
Кроме `SY00` могут быть использованы и другие переменные окружения настройки запуска ядра СУБД ЛИНТЕР.

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

3) для обеспечения взаимодействия запущенного экземпляра ядра СУБД ЛИНТЕР с удаленными клиентами запустить сетевой драйвер сервера с ключами /M, /P.

   Ключ /M определяет идентификатор межзадачного обмена, через который драйвер будет взаимодействовать с ядром, поэтому его значение должно совпадать со значением переменной LINTER_MBX при запуске соответствующего ядра.

   Ключ /P задает номер порта, через который должно осуществляться взаимодействие сетевого драйвера сервера и клиента (dbs_tcp, dbc_tcp) между собой.

   По файлу nodetab определяем, что выбранному ЛИНТЕР-серверу соответствует номер порта 1060, следовательно, /P=1060. Или, наоборот, задаем номер порта и затем редактируем nodetab.
4) запустить сетевой драйвер сервера с установленными параметрами:

```
dbs_tcp /M=30000 /P=1060
```

   Полученная схема взаимодействия сетевого и клиентского драйверов показана на рисунке 29.

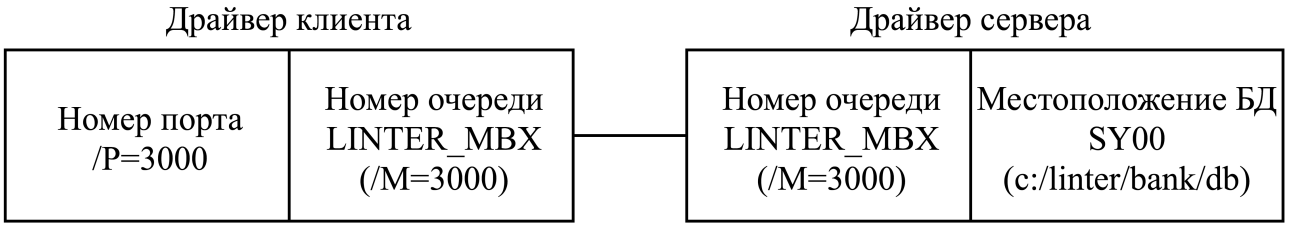


Рисунок 29. Схема взаимодействия сетевого и клиентского драйверов

5) повторить первые четыре пункта с необходимыми модификациями для запуска всех БД.

Если на сервере установлена только одна БД, то при запуске сетевого драйвера сервера ключи /P, /M можно не задавать (драйвер и ядро СУБД будут использовать значение по умолчанию). При этом номер порта в файле nodetab для соответствующего сервера также должен иметь значение, равное значению порта по умолчанию для этого типа протокола.

Активизация клиента

Удаленный клиент

Для удаленного доступа к конкретной БД необходимо:

1) запустить на клиентском компьютере сетевой драйвер клиента. Он может обслуживать несколько одновременно функционирующих на компьютере приложений, поэтому должен запускаться только один раз.

   В процессе соединения драйвер использует управляющую информацию из файла nodetab. По умолчанию драйвер предполагает, что nodetab находится в том же каталоге, что и сам драйвер.

   Если это не так, или на компьютере имеется несколько версий файла nodetab, то следует использовать ключ /N, задающий конкретный полный путь для поиска файла nodetab.

Примечание
Файл `nodetab` на пользовательском компьютере не обязан содержать список всех ЛИНТЕР-серверов локальной сети. Так, если с данного компьютера всегда осуществляется соединение лишь с одной конкретной БД, файл `nodetab` может состоять только из одной строки, таким образом, файлы `nodetab` на разных компьютерах сети могут различаться. Это предоставляет администратору СУБД дополнительные возможности для ограничения доступа пользователей к БД путем расширения или сужения списка доступных ЛИНТЕР-серверов в каждом локальном файле `nodetab`.

2) запустить клиентское приложение и в процессе соединения указать имя соответствующего ЛИНТЕР-сервера. Дальнейший процесс протекает в следующей последовательности:
   - сетевой драйвер клиента получает имя ЛИНТЕР-сервера, с которым приложение предполагает установить соединение;
   - в файле `nodetab`, указанном при запуске, драйвер клиента ищет ЛИНТЕР-сервер с заданным именем;
   - если имя ЛИНТЕР-сервера не найдено, происходит отказ в соединении, в противном случае определяются сетевой адрес компьютера и номер порта, выделенный на сервере для взаимодействия с клиентским приложением. Если на указанном сервере запущен сетевой драйвер сервера с указанным номером порта, то все запросы данного клиентского приложения направляются этому экземпляру сетевого драйвера сервера, который, в свою очередь, будет передавать их связанному с ним ядру СУБД.

Локальный клиент

Соединение с локальным ядром СУБД ЛИНТЕР осуществляется аналогично сетевому, но по протоколу LOCAL.

Также соединение может быть установлено вообще без использования сетевых средств с указанием только переменной окружения `LINTER_MBX` или, вообще, без указания каких либо переменных. В последнем случае будет осуществлено соединение с локальным ядром с идентификатором обмена по умолчанию.

Активизация сервера

Локальный сервер

Вариант 1: на сервере одна БД

Выполнить следующие действия:

1) если переменная окружения `SY00` установлена, выполнить команду `linter`.
2) если переменная SY00 не установлена, выполнить команду

```
linter /base=<каталог БД> /local
```

Вариант 2: на сервере несколько локальных БД

В этом случае работа с локальными ЛИНТЕР-серверами выполняется аналогично работе с удаленными ЛИНТЕР-серверами (см. пункт «Удаленный сервер»).

 © Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

Удаленный сервер

Порядок запуска СУБД ЛИНТЕР зависит от того, к скольким ЛИНТЕР-серверам необходимо обеспечить одновременный доступ.

Вариант 1: на сервере одна БД

Выполнить следующие команды:

1) `linter` (запуск ядра СУБД);
2) `dbs_tcp` (запуск сетевого драйвера сервера).

Вариант 2: на сервере несколько БД

Для иллюстрации запуска БД в качестве примера используется следующий фрагмент файла `nodetab`:

Условное имя компьютера	Протокол	Адрес	Номер порта	Местоположение БД
BANK	TCPIP	100.101.67.90		# …/db/bank
Sale	TCPIP	100.101.67.90	1061	# …/db/com/sale
Person	TCPIP	100.101.67.90	1062	# …/db/com/person
Plan	TCPIP	100.101.67.90	1063	# /usr/linter/db

БД могут запускаться в любой очередности.

Запуск БД Sale

1) Указать местоположение БД:

```
SY00=<префикс>/db/com/sale
export SY00
```

где:

<префикс> – начальный путь к каталогу БД, например:

`/usr/linter` или `/home/user`

2) Выбрать из файла `nodetab` номер порта (1061), закрепленный за этой БД. Данное значение соответствует ключу `/P` сетевого драйвера сервера (`/P=1061`).
3) Задать номер очереди сообщений, например, 30100:

```
LINTER_MBX=30100
export LINTER_MBX
```

Данное значение соответствует ключу `/M` сетевого драйвера сервера (`/M=30100`).

4) Запустить на выполнение экземпляр СУБД ЛИНТЕР:

```
linter
```

5) Запустить сетевой драйвер сервера для этого экземпляра СУБД с ключами `/M`, `/P`:

```
dbs_tcp /M=30100 /P=1061
```

Запуск БД BANK

Примечание
В файле `nodetab` БД BANK описана как БД по умолчанию, то есть номер порта для нее не указан.

1) Указать местоположение БД:

```
SY00=<префикс>/db/bank
export SY00
```

2) Выбрать из файла `nodetab` закрепленный за этой БД номер порта, а так как он явно не задан, используется значение по умолчанию (1060). Это значение задается в ключе `/P` сетевого драйвера сервера (`/P=1060`).

3) Номер очереди не задаем, а используем значение по умолчанию (30000). Оно соответствует ключу `/M` сетевого драйвера сервера (`/M=30000`).

4) Запустить на выполнение экземпляр СУБД ЛИНТЕР:

```
linter
```

5) Запустить сетевой драйвер сервера для этого экземпляра СУБД с ключами `/M`, `/P` по умолчанию:

```
dbs_tcp
```

Примечание
Только одна БД может быть запущена таким образом (то есть по умолчанию).

Запуск БД Person

1) Указать местоположение БД:

```
SY00= <префикс>/db/com/person
export SY00
```

2) Выбрать из файла `nodetab` закрепленный за этой БД номер порта (1062). Это значение задается в ключе `/P` сетевого драйвера сервера (`/P=1062`).

3) Задать номер очереди сообщений, например, 30200:

```
LINTER_MBX=30200
export LINTER_MBX
```

Это значение соответствует ключу `/M` сетевого драйвера сервера (`/M=30200`).

4) Запустить на выполнение экземпляр СУБД ЛИНТЕР:

```
linter
```

5) Запустить сетевой драйвер сервера для этого экземпляра СУБД с ключами `/M`, `/P`:

```
dbs_tcp /M=30200 /P=1062
```

Аналогично выполняется запуск и всех остальных БД сервера.

 © Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

Примечание
Убедиться, что все необходимые для функционирования СУБД ЛИНТЕР компоненты загружены, можно с помощью команды `ps -ax`.

В списке процессов ОС должны присутствовать:

- linter;
- sql;
- tsp;
- intsrt;
- dbs_tcp.

Количество процессов должно соответствовать количеству запущенных экземпляров БД.

Примечания

1. Некоторых процессов (`dbs_tcp`, `intsrt`) может быть более чем 1 на каждый экземпляр СУБД.
2. Некоторые процессы могут не запускаться (`sql`, `tsp`). Пропуска запуска процессов настраивается при запуске СУБД ЛИНТЕР ключами командной строки.

Приложение 1
Варианты сетевых конфигураций СУБД ЛИНТЕР

Ниже приведены схемы сетевых конфигураций клиентского компьютера и ЛИНТЕР-сервера.

1) Несколько клиентских компьютеров – один ЛИНТЕР-сервер (рис. П1.1).

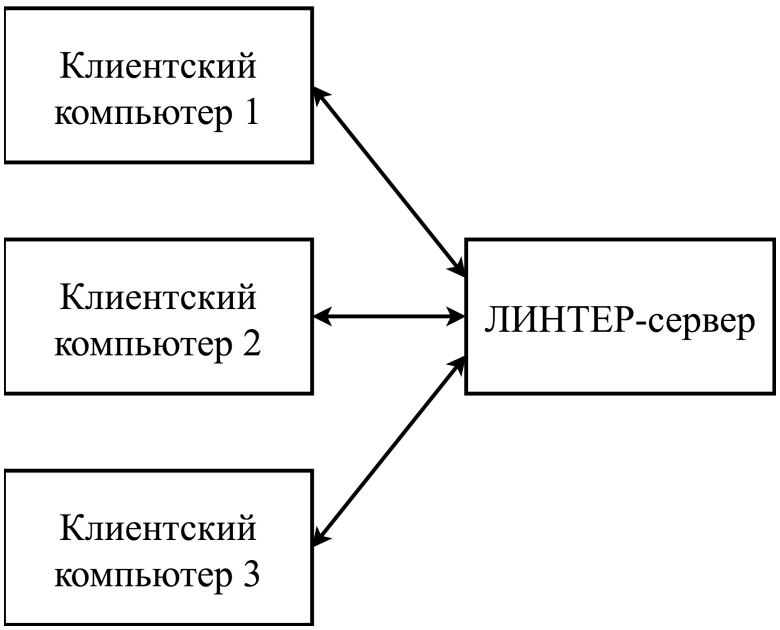


Рисунок П1.1. Сетевая конфигурация с одним ЛИНТЕР-сервером и несколькими клиентами

В данной конфигурации файлы `nodetab` на каждом клиентском компьютере должны ссылаться на один и тот же сетевой адрес ЛИНТЕР-сервера (который задается по-разному для различных типов протоколов), все остальные параметры могут быть заданы индивидуально.

Примеры.

Одинаковая сетевая конфигурация:

```
SERVER   TCPIP   195.98.69.227   0x424
```

Индивидуальная для каждого клиентского компьютера сетевая конфигурация:

```
Для компьютера 1
SERVER1  TCPIP  195.98.69.227  1060   2   20

Для компьютера 2
SERVER2  TCPIP  195.98.69.227  1062   1   30
```

2) Один клиентский компьютер – несколько ЛИНТЕР-серверов (рис. П1.2).

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

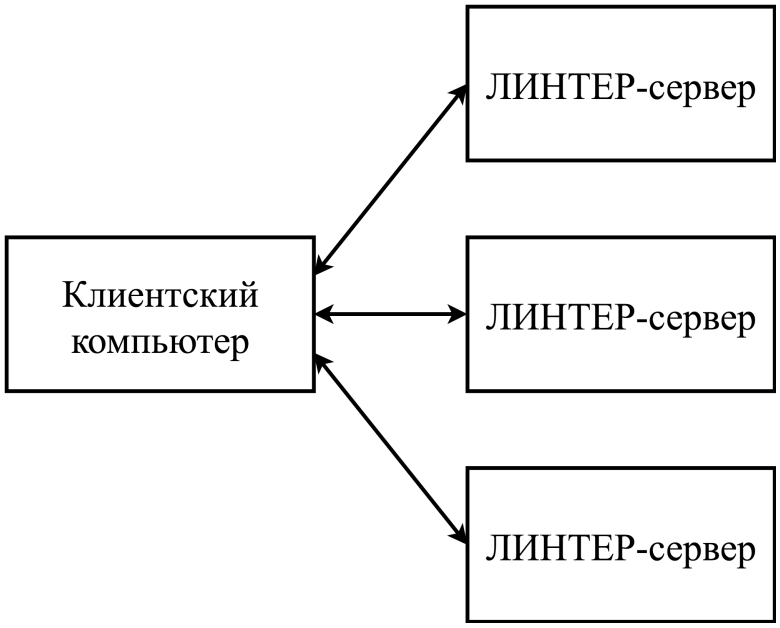


Рисунок П1.2. Сетевая конфигурация с одним клиентом и несколькими ЛИНТЕР-серверами

Пример.

Файл `nodetab` имеет вид:

```
SERVER1  TCPIP  195.98.69.200  1060  2  10  2
SERVER2  TCPIP  linter.ru    1062  1  10  5
SERVER3  TCPIP  195.98.69.201  1063
```

3) Несколько клиентских компьютеров – несколько ЛИНТЕР-серверов (рис. П1.3).

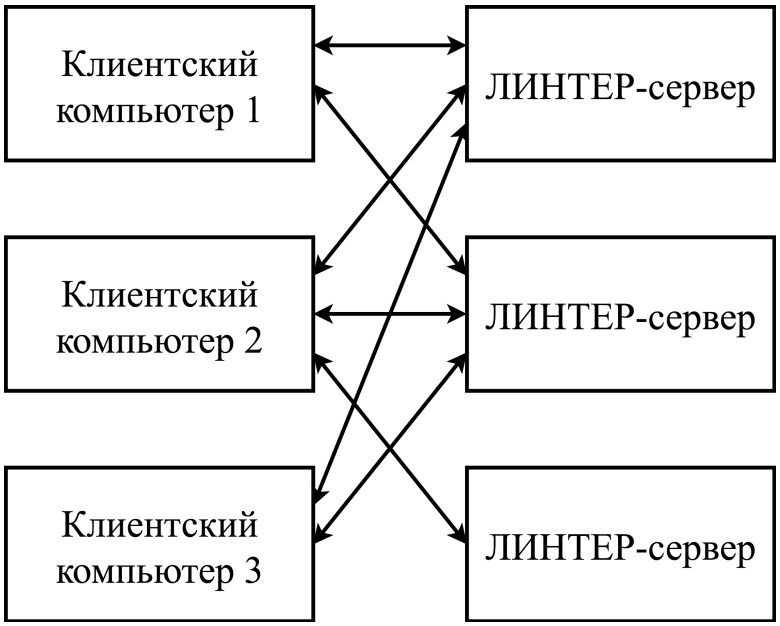


Рисунок П1.3. Сетевая конфигурация с несколькими клиентами и ЛИНТЕР-серверами

Пример.

Файлы `nodetab` имеют вид:

```
Для компьютера 1
SERVER1  TCPIP   195.98.69.227         1060  2  0
SERVER2  TCPIPS  195.98.69.200         1060  2

Для компьютера 2
SERVER2   TCPIP    195.98.69.200     1062  1  20
SERVER1   TCPIPS   195.98.69.227     1060  2  20
SERVER3   TCPIP    195.98.69.100     1060  2  0

Для компьютера 3
SERVER1  TCPIP   195.98.69.227            1060   2
```

4) Несколько клиентских компьютеров – несколько ЛИНТЕР-серверов (рис. П1.4).

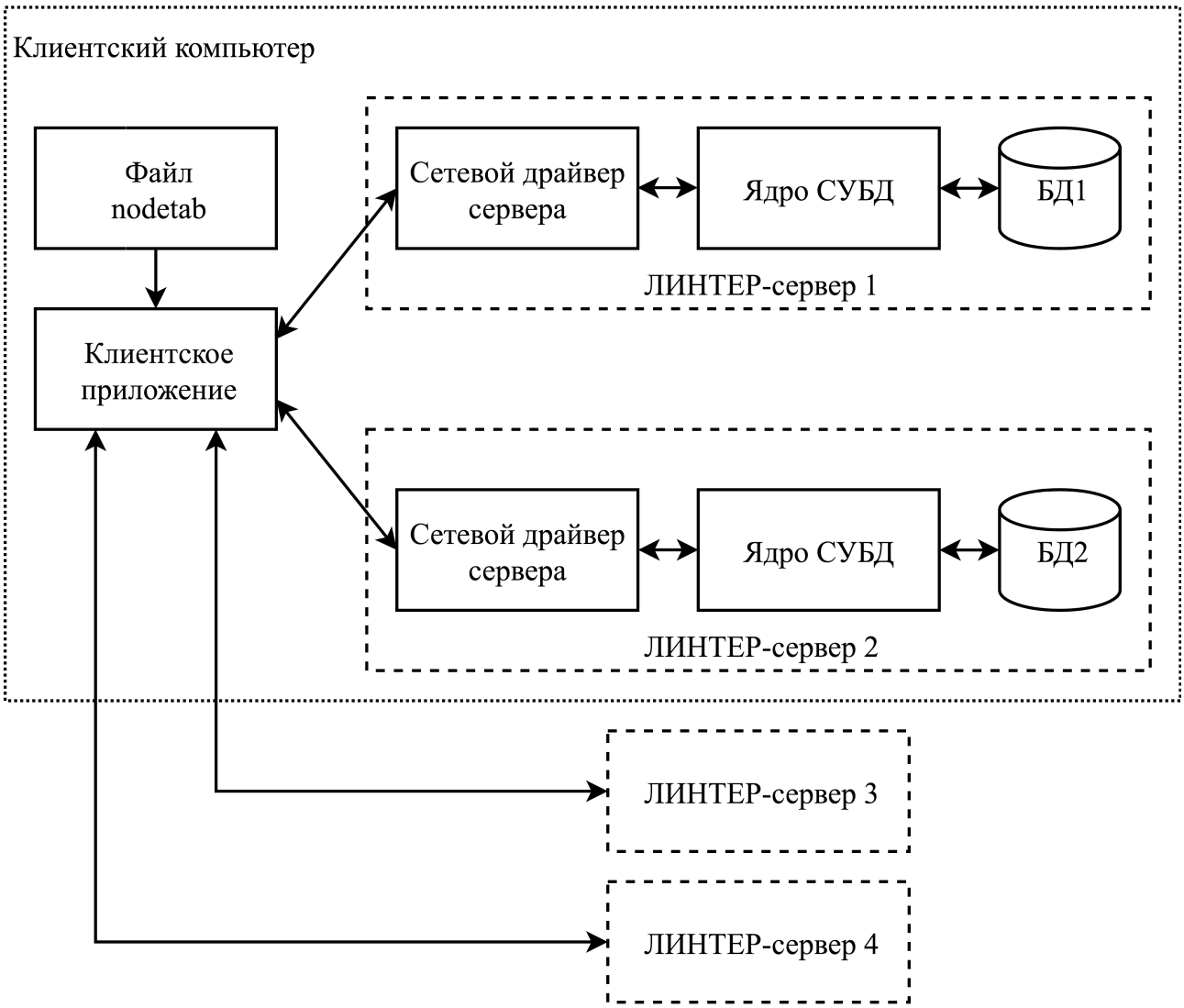


Рисунок П1.4. Сетевая конфигурация с несколькими клиентскими компьютерами и несколькими ЛИНТЕР-серверами

Пример.

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

Файлы `nodetab` имеют вид:

```
SERVER1    LOCAL
SERVER2    TCPIPS  195.98.69.200            1060  2
SERVER3    TCPIP   195.98.69.227            1062  1  20
```

5) Локальный ЛИНТЕР-сервер (без использования сетевых средств) (рис. П1.5).

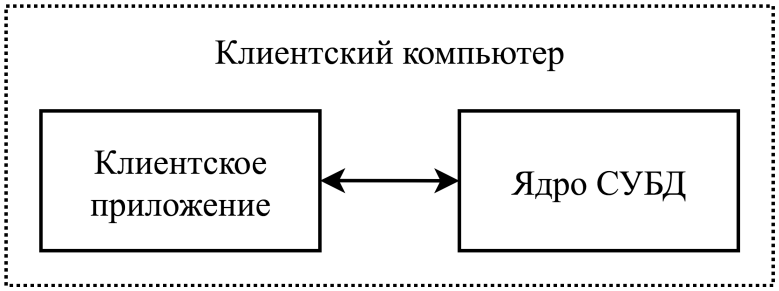


Рисунок П1.5. Сетевая конфигурация локального ЛИНТЕР-сервера

Приложение 2
Пример создания файла сетевой конфигурации

Для иллюстрации процесса создания файла сетевой конфигурации СУБД ЛИНТЕР будет использоваться фрагмент локальной вычислительной сети, представленный на рисунке П2.1.

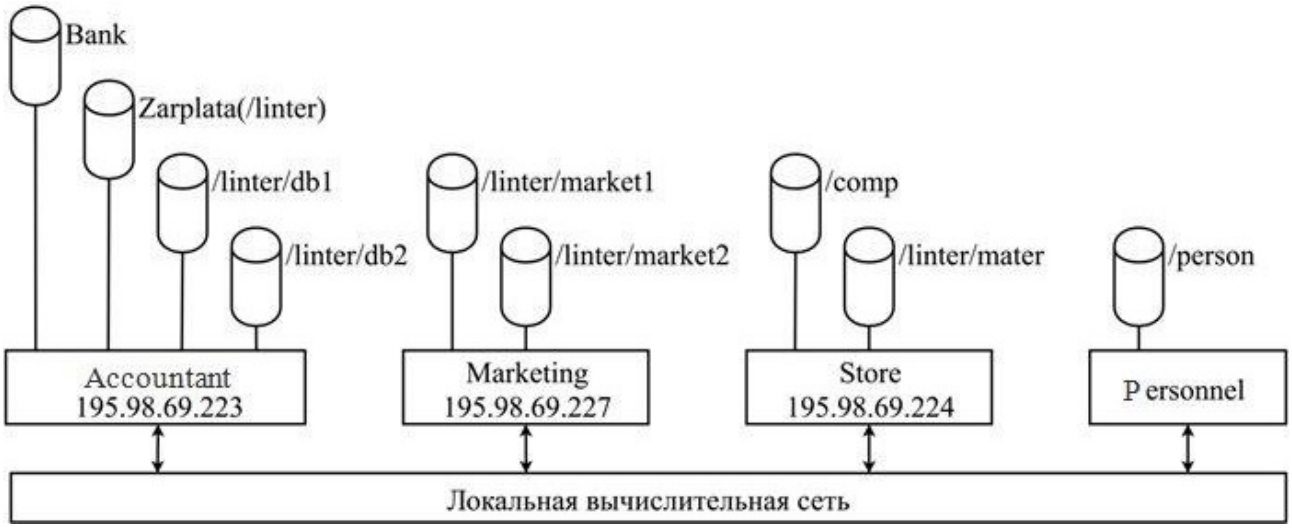


Рисунок П2.1. Фрагмент локальной вычислительной сети

Для создания файла сетевой конфигурации необходимо:

1) определить, к скольким БД одновременно на каждом компьютере будет разрешен сетевой доступ (то есть доступ с другого компьютера). На этой стадии не решается вопрос, к каким конкретно БД, а определяется только их суммарное количество. Например, если для компьютера `Accountant` (рис. П2.1), где установлено четыре БД, мы решим, что в любой момент времени только к двум из них будет разрешен сетевой доступ, это значит, что сетевой доступ будет обеспечен к любым двум БД из перечисленных четырех, к любой из оставшихся БД доступ будет возможен только в локальном режиме. Количество одновременных сетевых доступов и определяет количество задаваемых в таблице `nodetab` портов для этого компьютера.
2) получить у администратора локальной сети сетевые адреса установленных (или планируемых к установке) компьютеров;

   Для протокола TCP/IP (TCP/IPS TLS) это могут быть сетевой IP-адрес или имя, определяемое через службу DNS (Domain Name Service). Например, 100.101.102.103 – числовой IP-адрес, mycomp.myorg.mydomen – каноническое DNS-имя, mycomp – сопредельное DNS-имя.
3) внести сетевые адреса в поле `Адрес` строк файла `nodetab`;
4) выяснить у администратора сети тип установленного протокола связи для доступа к компьютеру;
5) внести обозначение используемых протоколов в поле `Протокол` файла `nodetab`.

После выполнения пунктов 1-4 файл `nodetab` может иметь следующий вид:

Условное имя компьютера	Протокол	Адрес	Порт	Тайм-аут сервера	Тайм-аут клиента	Тайм-аут соединения
	TCPIP	Accountant				

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

Условное имя компьютера	Протокол	Адрес	Порт	Тайм-аут сервера	Тайм-аут клиента	Тайм-аут соединения
	TCPIP	195.98.69.227				
	TCPIPS	195.98.69.226				

[1)]для ОС Windows.

[2)]для ОС Linux, ЗОСРВ Нейтрино.

6) для каждой строки `nodetab` создать столько копий этой строки, сколько портов определено для данного компьютера.

   Пусть, например, для компьютера `Accountant` (рис. П2.1) разрешено три одновременных сетевых доступа к БД, для компьютеров `Marketing` и `Store` – по два сетевых доступа к БД и для компьютера `Personnel` – один сетевой доступ к БД. Тогда файл `nodetab` на данном этапе может иметь следующий вид:

Условное имя компьютера	Протокол	Адрес	Порт	Тайм-аут сервера	Тайм-аут клиента	Тайм-аут соединения
	TCPIP	Accountant				
	TCPIP	195.98.69.227				
	TCPIP	Marketing				
	LOCAL	Personnel				
	TCPIP	Accountant				
	TCPIP	Accountant				

Примечание

При дублировании строки для компьютера 2 в столбце `Адрес` вместо первоначального физического адреса 195.98.69.227 подставлено сетевое имя `Marketing`, тем не менее эти две строки совершенно идентичны с точки зрения сетевых служб. Дублирующие строки для компьютера 1 поставлены в конец файла – это говорит о том, что записи в файле могут располагаться в произвольном порядке.

7) для каждой записи файла `nodetab` указать номер порта, через который будет происходить взаимодействие клиентского приложения с СУБД ЛИНТЕР на этом компьютере. В пределах одного компьютера все номера портов должны быть уникальными.

Примечания

1. Если на данном компьютере будет использоваться несколько портов для доступа к ЛИНТЕР-серверам, то для исключения конфликтных ситуаций при работе сети допускается использовать только один номер по умолчанию.
2. В ОС Linux, ЗОСРВ Нейтрино для получения более наглядной информации о портах TCP/IP по команде netstat лучше внести в файл `/etc/services` строку:

```
1060   tcpip   Linter
```

После выполнения пункта 7 файл `nodetab` должен иметь следующий вид:

Условное имя компьютера	Протокол	Адрес	Порт	Тайм-аут сервера	Тайм-аут клиента	Тайм-аут соединения
	TCPIP	Accountant	1)			
	TCPIP	195.98.69.227	1)			
	TCPIP	Marketing	0x425			
	TCPIPS	195.98.69.226	0x426			
	LOCAL	Personnel				
	TCPIP	Accountant	1061			
	TCPIP	Accountant	1062			

1) Событие не аудируется.

8) ввести, при необходимости, в столбцы `Тайм-аут сервера`, `Тайм-аут клиента`, `Тайм-аут соединения` отличные от значения по умолчанию значения тайм-аутов.

Таким образом, видно, что комбинация значений `Протокол`, `Адрес` и `Порт` в файле `nodetab` задает неповторяющееся (уникальное) значение в пределах строк файла, то есть данному набору значений можно поставить в соответствие обобщенное имя и в дальнейшем, при необходимости, ссылаться на него. Это имя называется «ЛИНТЕР-сервер» – логическое имя для обозначения сетевого имени компьютера и соответствующего сетевого порта на этом компьютере. В качестве имени ЛИНТЕР-сервера следует использовать названия, несущие информацию о содержании БД, например, `Bank`, `Sale`. После заполнения поля ЛИНТЕР-сервер файл `nodetab` приобретет свой окончательный вид:

Условное имя компьютера	Протокол	Адрес	Порт	Тайм-аут сервера	Тайм-аут клиента	Тайм-аут соединения
Bank	TCPIP	Accountant				
Tender	TCPIP	195.98.69.227	0x424			
Sale	TCPIP	Marketing				
Series	TCPIPS	195.98.69.226	1060			
Material	TCPIPS	195.98.69.225	1060			
Cadres	LOCAL	Personnel				
Zarplata	TCPIP	Accountant	1061			
Postavki	TCPIP	Accountant	1062			

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

Приложение 3
Примеры конфигурационных файлов для запуска из сетевого суперсервиса inetd

Пример конфигурационного файла сервиса inetd `inetd.conf`:

```
linter stream tcp nowait root /usr/linter/bin/dbs_tcp dbs_tcp /d
```

Пример конфигурационного файла сервиса xinetd `xinetd.conf`:

```
service linter { disable = no socket_type = stream wait = no
 protocol = tcp server = /usr/linter/bin/dbs_tcp server_args = /d
 port = 1060 user = root }
```

В приложении:

- для сервиса inetd в файл `inetd.conf` необходимо внести строку:

```
linter stream tcp nowait root /usr/linter/bin/dbs_tcp dbs_tcp /d
```

  где linter – имя сервиса, описание которого должно быть добавлено в файл `/etc/services`: linter 1060/tcp

  После аргумента `/d` могут быть добавлены и другие дополнительные аргументы запуска `dbs_tcp`.

- для сервиса xinetd в каталог `xinetd.conf` необходимо внести файл с именем `linter` и содержанием:

```
service linter
{ disable = no socket_type = stream wait = no protocol = tcp
 server = /usr/linter/bin/dbs_tcp server_args = /D /M=1234 port =
 1060 user = root }
```

  В поле `server_args` могут быть добавлены и другие дополнительные аргументы, однако аргумент `/d` является обязательным. В данном примере указывается, что `dbs_tcp` работает с ядром СУБД на МВХ со значением 1234.

Приложение 4
Работа с сертификатами

В приложении приведен скрипт, иллюстрирующий возможности работы сетевых средств с сертификатами и ключами.

В скрипте демонстрируется выполнение следующих функций:

1) удаление старых файлов и каталогов;
2) создание структуры каталогов УЦ (Удостоверяющего Центра, (Certification authority, CA));
3) создание самоподписного сертификата и ключа корневого УЦ:
   - создание запроса сертификата и ключа без пароля;
   - подписание своего сертификата своим ключом;
   - копирование сертификата с выкидыванием текстовой информации.

   где:

   RootCACert.pem – корневой сертификат.

   ./demoCA/private/./cakey.pem – ключ.
4) создание промежуточного УЦ:
   - создание запроса сертификата и ключа;
   - подписание сертификата промежуточного УЦ корневым УЦ;
   - копирование с выкидыванием текстовой информации;
   - добавление в сертификат промежуточного УЦ сертификата корневого УЦ;
   - переименование структуры каталогов корневого УЦ в demoCARoot;
   - создание структуры каталогов промежуточного УЦ;
   - копирование сертификата и ключа в нужные каталоги дерева.

   где:

   cacert.pem – сертификат промежуточного УЦ.

   IntermCA.key – ключ промежуточного УЦ.
5) создание сертификатов 1 клиента:
   - создание ключа и запроса на сертификат;
   - создание самоподписного сертификата;
   - подписание сертификата УЦ;
   - удаление из сертификата текста.

   где:

   client1.key – ключ.

   client1SS.crt – самоподписной сертификат.

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

client1.crt – сертификат, подписанный промежуточным УЦ.

6) создание сертификатов 2 клиента:
   - аналогично первому, но с цифрой 2.

   где:

   client2.key – ключ.

   client2SS.crt – самоподписной сертификат.

   client2.crt – сертификат, подписанный промежуточным УЦ.
7) создание сертификатов сервера.

   где:

   server.key – ключ.

   serverSS.crt – самоподписной сертификат.

   server.crt – сертификат, подписанный промежуточным УЦ.
8) создание сертификатов 2 сервера.

   где:

   server1.key – ключ.

   server1SS.crt – самоподписной сертификат.

   server1.crt – сертификат, подписанный промежуточным УЦ.
9) создание каталогов сетевого клиента и сетевого сервера.

   Предполагается что один из каталогов является каталогом валидного клиента или сервера, а второй – нарушителя.

   Создание `nodetab` для работы незащищенного соединения, защищенного SSL23 и TLS с именами MYOPEN MYSSL и MYTLS соответственно в каталогах клиентов.

   где:

   dbs_tcp – каталог сервера.

   dbs_tcp1 – каталог нарушителя сервера.

   dbc_tcp1 – каталог клиента.

   dbc_tcp2 – каталог нарушителя.
10) создание каталога БД и собственно БД. Предварительно убиваются существующие ядра СУБД ЛИНТЕР и сетевые драйвера.
11) проверка простого зашифрованного соединения:
    - запуск сетевых драйверов в соответствующих пустых каталогах;
    - проверка удачного соединения по всем 3 протоколам;

- останов драйверов.

Все соединения успешны, поскольку нет работы с сертификатами и контроля клиента и сервера соответственно.

12) проверка отсечения работы по незащищенному каналу:
    - запуск сетевых драйверов `dbs_tcp` с опцией запрещения работы по незащищенному соединению;
    - попытка соединения по 3 протоколам;
    - останов драйверов.

    Соединение по незащищенному каналу неуспешно.
13) копирование самоподписных сертификатов и ключей в соответствующие каталоги запуска.

    Сертификаты используются для идентификации клиента и сервера. Также сертификаты определяют протокол соединения. Например, ГОСТ-сертификаты предполагают ГОСТ-алгоритм. Чтобы работало везде используется DSA-сертификаты. В каталоге сервера ключи и сертификаты должны иметь имена `dbs_tcp.key` и `dbs_tcp.crt`, в каталоге клиента – `dbc_tcp.key` и `dbc_tcp.crt` соответственно.
14) запуск `dbc` и `dbs`:
    - `dbs` запускается с дополнительным ключом аутентификации клиента;
    - незащищенное соединение неуспешно;
    - защищенные соединения успешны;
    - повторные защищенные соединения также успешны;
    - при повторном соединении производится сверка сертификатов.
15) попытка подмены адреса клиента:
    - перезапуск клиента в другом каталоге. Поскольку ключи и сертификаты заменены – это эквивалентно попытке работы другого клиента с этого же адреса;
    - поскольку сравнение сертификатов прошло неуспешно, все 3 соединения неуспешны:
        - незащищенное – поскольку запрещено;
        - защищенные – из-за разницы в сертификатах.
16) подмена адреса сервера:
    - останов всех сетевых серверов;
    - запуск dbc в правильном каталоге, а dbs в каталоге с другими сертификатами. Результат, как в предыдущем пункте, поскольку не совпадают сохраненные сертификаты клиентов;
    - останов сетевых драйверов.
17) запрет работы без сертификатов:
    - удаление всех сохраненных сертификатов;
    - запуск dbs с соответствующей дополнительной опцией;
    - несмотря на присутствие своих правильных сертификатов, соединения неуспешны из-за отсутствия сертификатов;
    - копирование нужного сертификата в каталог `dbs_tcp` восстанавливает возможность соединения.

 © Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

18) удаление всех старых сертификатов, как самоподписных, так и сохраненных:
    - копирование сертификатов, подписанных УЦ, в соответствующие каталоги;
    - повторение проверок п.п. 14, 15, 16 для сертификатов УЦ.
19) проверка отсутствия соединения с сертификатами, не подписанными УЦ.

    Эта проверка включается при наличии сертификата(ов) УЦ в текущем каталоге. Они должны называться `dbc_tcp.CA` и `dbs_tcp.CA` соответственно для клиента и сервера. В то же время для соединения используются самоподписные сертификаты. В случае неподписанных сертификатов и активации проверки подписи УЦ SSL соединение проходит успешно, а TLS нет, поскольку сертификат подписан не тем УЦ, которому доверяют клиент и сервер.

    Сохранение и сверка сертификатов продолжает работать (для SSL).
20) проверка, соединения с сертификатами, подписанными УЦ.

    Восстановление подписанных УЦ сертификатов в соответствующих каталогах. И SSL и TLS соединение успешно.

    Сверка сертификатов продолжает работать верно – подмена адреса не проходит.
21) проверка списка отозванных сертификатов (CRL):
    - отзываем сертификат у клиента;
    - формируем список отозванных сертификатов;
    - копируем его в каталог сервера с именем `dbs_tcp.CRL`;
    - после этого попытка соединения по TLS протоколу неуспешна, потому что сертификат клиента недействителен (присутствует в списке CRL).
22) выдача клиенту нового сертификата и ключа, чтобы они были валидными. Замена ключей на валидные.
23) проверка восстановления ключей клиента. Соединения проходят успешно.
24) отзыв сертификата сервера. Список отозванных сертификатов копируется клиенту.
25) после отозвания сертификата сервера установка соединения с ним по TLS протоколу неуспешна. SSL продолжает работать как было.

    Скрипт может быть вызван без аргументов для работы с промежуточным УЦ.

    Аргумент no_interm используется для работы без промежуточного УЦ.

    Аргумент clean используется для очистки результатов работы.

```
#!/bin/sh
KEY_LEN=512
arg=$1 #clean no_interm

if [ "$arg" = "-h" -o "$arg" = "--help" -o "$arg" = "?" ]; then
  echo "usage: keys.sh [clean] [no_interm]
  clean - remove all test files
  no_interm - use only one root CA whithout intermediate CA"
fi

#1
```

```
rm -rf ./demoCA
rm -rf ./demoCARoot
rm -rf ./dbc_tcp1
rm -rf ./dbc_tcp2
rm -rf ./dbs_tcp
rm -rf ./dbs_tcp1
rm -rf db
rm -f client1*
rm -f client2*
rm -f server*
rm -f RootCACert*
rm -f IntermCA*
rm -f cacert.pem
rm -f CRL.pem
rm -f RootCRL.pem

if [ "$arg" = "clean" ]; then
  exit
fi #clean

#2
mkdir -p ./demoCA
mkdir -p ./demoCA/certs
mkdir -p ./demoCA/crl
mkdir -p ./demoCA/newcerts
mkdir -p ./demoCA/private
chmod og-rwx ./demoCA/private
touch ./demoCA/index.txt

#3
#creating self signed sertificate of root CA
#creating key and request
openssl  req  -new -keyout ./demoCA/private/./cakey.pem -newkey
 rsa:$KEY_LEN
 -nodes \
 -days 100 -set_serial 100 -out ./demoCA/./careq.pem \
 -subj /C=RU/ST=Voronezh\ region/L=Voronezh/O=Relex\ inc/OU=System
\
  develop/CN=RootCA\/emailAddress=RootCA@relex.ru
echo "00" > ./demoCA/serial
#self sign the sertificate
openssl ca -out ./demoCA/./cacert.pem -days 100 -batch -keyfile \
./demoCA/private/./cakey.pem -selfsign -extensions v3_ca -infiles
 ./demoCA/./careq.pem
#copy the certificate without text info
openssl x509 -in ./demoCA/./cacert.pem -out ./RootCACert.pem
```

 © Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

```
#4
if [ "$arg" != "no_interm" ]; then
#creating intermediate CA
#creating request for the sertificate
openssl req -keyout IntermCA.key -newkey rsa:$KEY_LEN -nodes -days
 100
 -set_serial 1000 \
-out IntermCA.req -subj /C=RU/ST=Voronezh\ region/L=Voronezh/
O=Relex\
 inc/OU=System\ develop/CN=IntermCA\/
emailAddress=IntermCA@relex.ru
#sign the request by root CA
openssl ca -batch -days 100 -policy policy_anything -extensions
 v3_ca -out
 IntermCA.crt -in IntermCA.req
openssl x509 -in IntermCA.crt -out ./cacert.pem
#add Root certificate to intermediate certificate
cat RootCACert.pem >> cacert.pem
rm -f IntermCA.req
#rename root storage
mv ./demoCA ./demoCARoot
#create intermediate storage
mkdir -p ./demoCA
mkdir -p ./demoCA/certs
mkdir -p ./demoCA/crl
mkdir -p ./demoCA/newcerts
mkdir -p ./demoCA/private
chmod og-rwx ./demoCA/private
touch ./demoCA/index.txt
cp cacert.pem ./demoCA/./cacert.pem
cp IntermCA.key ./demoCA/private/./cakey.pem
else # no_intermediate
mv RootCACert.pem cacert.pem
fi #no_intermediate
echo "10" > ./demoCA/serial

#5
#creating client request client 1
openssl req -keyout client1.key -newkey rsa:$KEY_LEN -nodes -days
 100
 -set_serial 01 \
-out client1.req -subj /C=RU/ST=Voronezh\ region/L=Voronezh/
O=Relex\
 inc/OU=System\ develop/CN=Client1\/emailAddress=c1@relex.ru
#creating self signed sertificate for the client
```

```
openssl x509 -req -days 365 -in client1.req -signkey client1.key -
out
 client1SS.crt
#sign the request of client 1
openssl ca -batch -days 100 -policy policy_anything -out
 client1_.crt -in
 client1.req
openssl x509 -in client1_.crt -out client1.crt
rm -f client1.req client1_.crt

#6
#creating client request client 2
openssl req -keyout client2.key -newkey rsa:$KEY_LEN -nodes -days
 100
 -set_serial 02 \
-out client2.req -subj /C=RU/ST=Voronezh\ region/L=Voronezh/
O=Relex\
 inc/OU=System\ develop/CN=Client2\/emailAddress=c2@relex.ru
#creating self signed sertificate for the client
openssl x509 -req -days 365 -in client2.req -signkey client2.key -
out
 client2SS.crt
#sign the request of client 2
openssl ca  -batch -days 100 -policy policy_anything -out
 client2_.crt -in
 client2.req
openssl x509 -in client2_.crt -out client2.crt
rm -f client2_.crt client2.req

#7
#creating server request
openssl req -keyout server.key -newkey rsa:$KEY_LEN -nodes -days
 100 -set_serial
 20 \
-out server.req -subj /C=RU/ST=Voronezh\ region/L=Voronezh/O=Relex
\
 inc/OU=System\ develop/CN=Server\/emailAddress=s@relex.ru
#creating self signed sertificate for the server
openssl x509 -req -days 365 -in server.req -signkey server.key -
out serverSS.crt
#sign the server request
openssl ca  -batch -days 100 -policy policy_anything -out
 server_.crt -in
 server.req
openssl x509 -in server_.crt -out server.crt
rm -f server.req server_.crt
```

 © Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

```
#8
#creating server request
openssl req -keyout server1.key -newkey rsa:$KEY_LEN -nodes -days
 100
 -set_serial 21 \
-out server1.req -subj /C=RU/ST=Voronezh\ region/L=Voronezh/
O=Relex\
 inc/OU=System\ develop/CN=Server\/emailAddress=s1@relex.ru
#creating self signed sertificate for the server
openssl x509 -req -days 365 -in server1.req -signkey server1.key -
out
 server1SS.crt
#sign the request of server
openssl ca  -batch -days 100 -policy policy_anything -out
 server1_.crt -in
 server1.req
openssl x509 -in server1_.crt -out server1.crt
rm -f server1_.crt server1.req

#9
mkdir dbs_tcp
mkdir dbs_tcp1
mkdir dbc_tcp1
mkdir dbc_tcp2
cd dbc_tcp1
echo "MYOPEN TCPIP localhost 1060 1 20 20
MYSSL TCPIPS localhost 1060 1 20 20
MYTLS TLS  localhost 1060 1 20 20" > nodetab

cp nodetab ../dbc_tcp2
cd ..

#10
killall linter
killall dbc_tcp
killall dbs_tcp
sleep 2
rm -rf db
mkdir db
export SY00=`pwd`/db
echo "cre da;" | gendb
linter
sleep 3
pwd
```

```
#11
#check simple connect
cd dbs_tcp
dbs_tcp -debug 0xFFFFFFFF
sleep 3
cd ..
cd dbc_tcp1
dbc_tcp -debug 0xFFFFFFFF
sleep 3
cd ..
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYOPEN || {
 echo "failed to open"
 exit
  }
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYSSL || {
 echo "failed to ssl simple connect"
 exit
  }
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYTLS || {
  echo "failed to tls simple connect"
  exit
  }
killall dbc_tcp
killall dbs_tcp
sleep 3

#12
#check ssl only flag
cd dbs_tcp
dbs_tcp -SSLONLY -debug 0xFFFFFFFF
sleep 3
cd ..
cd dbc_tcp1
dbc_tcp -debug 0xFFFFFFFF
sleep 3
cd ..
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYOPEN && {
  echo "failed to open success"
  exit
  }
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYSSL || {
```


© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

```
  echo "failed to ssl sslonly"
  exit
  }
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYTLS || {
  echo "failed to tls sslonly"
  exit
  }
killall dbc_tcp
killall dbs_tcp
sleep 3


#13
#coping the self signed sertificates to check work with its
cp serverSS.crt dbs_tcp/dbs_tcp.crt
cp server.key dbs_tcp/dbs_tcp.key
cp server1SS.crt dbs_tcp1/dbs_tcp.crt
cp server1.key dbs_tcp1/dbs_tcp.key
cp client1SS.crt dbc_tcp1/dbc_tcp.crt
cp client1.key dbc_tcp1/dbc_tcp.key
cp client2SS.crt dbc_tcp2/dbc_tcp.crt
cp client2.key dbc_tcp2/dbc_tcp.key

#14
#checking sertificate compare
#rm -f dbs_tcp/*.log dbc_tcp1/*.log
cd dbs_tcp
dbs_tcp -SSLONLY -SSLAUTH -debug 0xFFFFFFFF
sleep 3
cd ..
cd dbc_tcp1
dbc_tcp -debug 0xFFFFFFFF
sleep 3
cd ..
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYOPEN && {
  echo "failed to open success"
  exit
  }
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYSSL || {
  echo "failed to ssl certif"
  exit
  }
```

```
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYTLS || {
  echo "failed to tls certif"
  exit
  }
killall dbc_tcp
sleep 3
cd dbc_tcp1
dbc_tcp -debug 0xFFFFFFFF
sleep 3
cd ..
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYSSL || {
  echo "failed to ssl certif"
  exit
  }
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYTLS || {
  echo "failed to tls certif"
  exit
  }

#15
#try the client from the same address with other sertificate
killall dbc_tcp
sleep 3
cd dbc_tcp2
dbc_tcp -debug 0xFFFFFFFF
sleep 3
cd ..
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYOPEN && {
  echo "failed to open unprotected"
  exit
  }
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYSSL && {
  echo "failed to ssl alien sert"
  exit
  }
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYTLS && {
  echo "failed to tls alien sert"
  exit
  }
```

 © Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

```
#16
#try the server with other sertificate
killall dbc_tcp
killall dbs_tcp
sleep 3
cd dbc_tcp1
#rm *.log
dbc_tcp -debug=0xFFFFFFFF
cd ..
cd dbs_tcp1
dbs_tcp -debug=0xFFFFFFFF #no authentification. Can receive all
 connections !!
cd ..
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYSSL && {
  echo "failed to ssl alien srv sert"
  exit
  }
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYTLS && {
  echo "failed to tls alien srv sert"
  exit
  }
killall dbc_tcp
killall dbs_tcp
sleep 3

#17
#checking crtificate without creation
rm -f dbs_tcp/127.0.0.1.crt
rm -f dbc_tcp1/MYSSL.crt
rm -f dbc_tcp1/MYTLS.crt
cd dbc_tcp1
dbc_tcp -debug=0xFFFFFFFF
cd ..
cd dbs_tcp
dbs_tcp -debug=0xFFFFFFFF -SSLAUTH -SSLNOCREAT
cd ..
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYTLS && {
  echo "failed to tls no cert"
  exit
  }
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYSSL && {
  echo "failed to ssl no cert"
```

```
  exit
  }
cp -f client1SS.crt dbs_tcp/127.0.0.1.crt
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYSSL || {
  echo "failed to ssl no cert ok"
  exit
  }
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYTLS || {
  echo "failed to tls no cert ok"
  exit
  }
killall dbs_tcp
killall dbc_tcp
sleep 3


#18
#checking of all OLD (before 01.12) has done
#checking the same with CA signed sertificates
#coping the CA signed sertificates to check work with its
rm -f dbs_tcp/*.crt #removing old self signed cert and saved cert
rm -f dbc_tcp1/*.crt
rm -f dbs_tcp1/*.crt
rm -f dbc_tcp2/*.crt
cp -f server.crt dbs_tcp/dbs_tcp.crt
cp -f server1.crt dbs_tcp1/dbs_tcp.crt
cp -f client1.crt dbc_tcp1/dbc_tcp.crt
cp -f client2.crt dbc_tcp2/dbc_tcp.crt
#checking sertificate compare
#rm -f dbs_tcp/*.log dbc_tcp1/*.log
cd dbs_tcp
dbs_tcp -SSLONLY -SSLAUTH -debug 0xFFFFFFFF
sleep 3
cd ..
cd dbc_tcp1
dbc_tcp -debug 0xFFFFFFFF
sleep 3
cd ..
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYOPEN && {
  echo "CA failed to open success"
  exit
  }
```

 © Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

```
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYSSL || {
  echo "CA failed to ssl certif"
  exit
  }
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYTLS || {
  echo "CA failed to tls certif"
  exit
  }
killall dbc_tcp
sleep 3
cd dbc_tcp1
dbc_tcp -debug 0xFFFFFFFF
sleep 3
cd ..
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYSSL || {
  echo "CA failed to ssl certif rep"
  exit
  }
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYTLS || {
  echo "CA failed to tls certif rep"
  exit
  }
#try the client from the same address with other sertificate
killall dbc_tcp
sleep 3
cd dbc_tcp2
dbc_tcp -debug 0xFFFFFFFF
sleep 3
cd ..
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYOPEN && {
  echo "CA failed to open unprotected"
  exit
  }
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYSSL && {
  echo "CA failed to ssl alien sert"
  exit
  }
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYTLS && {
  echo "CA failed to tls alien sert"
```

```
  exit
  }
#try the server with other sertificate
killall dbc_tcp
killall dbs_tcp
sleep 3
cd dbc_tcp1
#rm *.log
dbc_tcp -debug=0xFFFFFFFF
cd ..
cd dbs_tcp1
dbs_tcp -debug=0xFFFFFFFF #no authentification. Can receive all
 connections !!
cd ..
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYSSL && {
  echo "CA failed to ssl alien srv sert"
  exit
  }
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYTLS && {
  echo "CA failed to tls alien srv sert"
  exit
  }
killall dbc_tcp
killall dbs_tcp
sleep 3

#19
#checking woiking TLS with CA
#resore back the self signed sertificates
rm -f dbs_tcp/*.crt #removing old self signed cert and saved cert
rm -f dbc_tcp1/*.crt
rm -f dbs_tcp1/*.crt
rm -f dbc_tcp2/*.crt
cp -f serverSS.crt dbs_tcp/dbs_tcp.crt
cp -f server1SS.crt dbs_tcp1/dbs_tcp.crt
cp -f client1SS.crt dbc_tcp1/dbc_tcp.crt
cp -f client2SS.crt dbc_tcp2/dbc_tcp.crt
#copyng the CA certificate
cp cacert.pem dbs_tcp/dbs_tcp.CA
cp cacert.pem dbs_tcp1/dbs_tcp.CA
cp cacert.pem dbc_tcp1/dbc_tcp.CA
cp cacert.pem dbc_tcp2/dbc_tcp.CA
#rm -f dbs_tcp/*.log dbc_tcp1/*.log
cd dbs_tcp
```

 © Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

```
dbs_tcp -SSLONLY -SSLAUTH -debug 0xFFFFFFFF
sleep 3
cd ..
cd dbc_tcp1
dbc_tcp -debug 0xFFFFFFFF
sleep 3
cd ..
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYOPEN && {
  echo "wCA failed to open success"
  exit
  }
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYSSL || {
  echo "wCA failed to ssl certif"
  exit
  }
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYTLS && {
  echo "wCA failed to tls certif"
  exit
  }
#try the client from the same address with other sertificate
killall dbc_tcp
sleep 3
cd dbc_tcp2
dbc_tcp -debug 0xFFFFFFFF
sleep 3
cd ..
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYOPEN && {
 echo "wCA failed to open unprotected"
 exit
 }
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYSSL && {
 echo "wCA failed to ssl alien sert"
 exit
 }
#must be failed because the sertificate is not signed by the CA
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYTLS && {
 echo "wCA failed to tls alien sert"
 exit
 }
#try the server with other sertificate
```

```
killall dbc_tcp
killall dbs_tcp
sleep 3
cd dbc_tcp1
#rm *.log
dbc_tcp -debug=0xFFFFFFFF
cd ..
cd dbs_tcp1
dbs_tcp -debug=0xFFFFFFFF #no authentification. Can receive all
 connections !!
cd ..
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYSSL && {
  echo "CA failed to ssl alien srv sert"
  exit
  }
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYTLS && {
  echo "CA failed to tls alien srv sert"
  exit
  }
killall dbc_tcp
killall dbs_tcp
sleep 3

#20
#restore back the CA signed sertificates
rm -f dbs_tcp/*.crt #removing old self signed cert and saved cert
rm -f dbc_tcp1/*.crt
rm -f dbs_tcp1/*.crt
rm -f dbc_tcp2/*.crt
cp -f server.crt dbs_tcp/dbs_tcp.crt
cp -f server1.crt dbs_tcp1/dbs_tcp.crt
cp -f client1.crt dbc_tcp1/dbc_tcp.crt
cp -f client2.crt dbc_tcp2/dbc_tcp.crt
rm -f dbs_tcp/*.log dbc_tcp1/*.log
cd dbs_tcp
dbs_tcp -SSLONLY -SSLAUTH -debug 0xFFFFFFFF
sleep 3
cd ..
cd dbc_tcp1
dbc_tcp -debug 0xFFFFFFFF
sleep 3
cd ..
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYOPEN && {
```

 © Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

```
  echo "waCA failed to open success"
  exit
  }
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYSSL || {
  echo "waCA failed to ssl certif"
  exit
  }
#the sertificate is signed so the connect should be success
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYTLS || {
  echo "waCA failed to tls certif"
  exit
  }
#try the client from the same address with other sertificate
killall dbc_tcp
sleep 3
cd dbc_tcp2
dbc_tcp -debug 0xFFFFFFFF
sleep 3
cd ..
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYOPEN && {
  echo "waCA failed to open unprotected"
  exit
  }
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYSSL && {
  echo "waCA failed to ssl alien sert"
  exit
  }
#must be failed because the sertificate is not signed by the CA
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYTLS && {
  echo "waCA failed to tls alien sert"
  exit
  }
#try the server with other sertificate
killall dbc_tcp
killall dbs_tcp
sleep 3
cd dbc_tcp1
#rm *.log
dbc_tcp -debug=0xFFFFFFFF
cd ..
cd dbs_tcp1
```

```
dbs_tcp -debug=0xFFFFFFFF #no authentification. Can receive all
 connections !!
cd ..
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYSSL && {
  echo "waCA failed to ssl alien srv sert"
  exit
  }
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYTLS && {
  echo "waCA failed to tls alien srv sert"
  exit
  }
killall dbc_tcp
killall dbs_tcp
sleep 3

#21
#creating CRL for the sertificate
echo "10" >./demoCA/crlnumber
openssl ca -revoke client1.crt
openssl ca -gencrl -out CRL.pem
if [ "$arg" != "no_interm" ]; then
mv demoCA demoCAinter
mv demoCARoot demoCA
echo "00" >./demoCA/crlnumber
openssl ca -gencrl -out RootCRL.pem
cat RootCRL.pem >> CRL.pem #concatenate both CRL
mv demoCA demoCARoot
mv demoCAinter demoCA
fi #no_interm
cp CRL.pem dbs_tcp/dbs_tcp.CRL
#rm -f dbs_tcp/*.log dbc_tcp1/*.log
cd dbs_tcp
dbs_tcp -SSLONLY -SSLAUTH -debug 0xFFFFFFFF
sleep 3
cd ..
cd dbc_tcp1
dbc_tcp -debug 0xFFFFFFFF
sleep 3
cd ..
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYOPEN && {
  echo "cwaCA failed to open success"
  exit
  }
```

 © Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

```
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYSSL || {
  echo "cwaCA failed to ssl certif"
  exit
  }
#now it should failed because the client sertificate was revoked
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYTLS && {
  echo "cwaCA failed to tls certif"
  exit
  }
killall dbc_tcp
killall dbs_tcp
sleep 3

#22
#creating new client request for client 1
rm -f client1.key client1.req client1.crt client1SS.crt
openssl req -keyout client1.key -newkey rsa:$KEY_LEN -nodes -days
 100
 -set_serial 01 \
 -out client1.req -subj /C=RU/ST=Voronezh\ region/L=Voronezh/
O=Relex\
 inc/OU=System\ develop/CN=Client1_new\/emailAddress=c1@relex.ru
#creating self signed sertificate for the client
openssl x509 -req -days 365 -in client1.req -signkey client1.key -
out
 client1SS.crt
#sign the request of client 1
openssl ca -batch -days 100 -policy policy_anything -out
 client1_.crt -in
 client1.req
openssl x509 -in client1_.crt -out client1.crt
rm -f client1.req client1_.crt
#replace client keys
cp -f client1.key dbc_tcp1/dbc_tcp.key
cp -f client1.crt dbc_tcp1/dbc_tcp.crt
rm -f dbs_tcp/127.0.0.1.crt
#cp -f client1.crt dbs_tcp/127.0.0.1.crt

#23
rm -f dbs_tcp/*.log* dbc_tcp1/*.log*
cd dbs_tcp
dbs_tcp -SSLONLY -SSLAUTH -debug 0xFFFFFFFF
sleep 3
cd ..
```

```
cd dbc_tcp1
dbc_tcp -debug 0xFFFFFFFF
sleep 3
cd ..
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYSSL || {
  echo "ssl after restoring crt"
  exit
  }
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYTLS || {
  echo "tls after restoring crt"
  exit
  }
killall dbc_tcp
killall dbs_tcp
sleep 3

#24
# revoke server sertificate
rm -f CRL.pem
openssl ca -revoke server.crt
openssl ca -gencrl -out CRL.pem
if [ "$arg" != "no_interm" ]; then
mv demoCA demoCAinter
mv demoCARoot demoCA
openssl ca -gencrl -out RootCRL.pem
cat RootCRL.pem >> CRL.pem #concatenate both CRL
mv demoCA demoCARoot
mv demoCAinter demoCA
fi #no_interm
cp CRL.pem dbc_tcp1/dbc_tcp.CRL

#25
rm -f dbs_tcp/*.log* dbc_tcp1/*.log*
cd dbs_tcp
dbs_tcp -SSLONLY -SSLAUTH -debug 0xFFFFFFFF
sleep 3
cd ..
cd dbc_tcp1
dbc_tcp -debug 0xFFFFFFFF
sleep 3
cd ..
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYSSL || {
  echo "ssl after revoke serv crt"
```

 © Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

```
  exit
  }
echo 'select count(*) from $$$USR;' | inl -u SYSTEM/MANAGER8 -n
 MYTLS && {
  echo "tls after revoke srv crt"
  exit
  }
killall dbs_tcp
killall dbc_tcp
killall linter
sleep 3
echo success
```

Приложение 5

Проверка работы сетевого клиента в среде ОС Linux, ЗОСРВ Нейтрино

Условия проверки

1) Для проверки используется два компьютера, один из которых является сервером, другой – клиентом.
2) На обоих компьютерах установлены ОС типа Linux, ЗОСРВ Нейтрино (не обязательно одинаковые).
3) На серверном компьютере установлены:
   - ядро СУБД ЛИНТЕР;
   - демонстрационная БД;
   - утилита `inl`;
   - сетевой драйвер сервера `dbs_tcp`;
   - сетевой драйвер клиента `dbc_tcp`.
4) Переменная окружения SY00 указывает на каталог демонстрационной БД.
5) К списку каталогов переменной окружения PATH добавлен каталог исполняемых файлов СУБД ЛИНТЕР.
6) На клиентской машине установлены:
   - утилита `inl`;
   - сетевой драйвер клиент `dbc_tcp`.
7) Если не оговорено особо, то в том же каталоге, где расположен драйвер `dbc_tcp`, должен находиться конфигурационный файл `nodetab` с описанием узлов сети:

   ```
   RTCP  TCPIP  <remote host>
   RTCPS TCPIPS <remote host>
   L     LOCAL  1254
   ```

   где `<remote host>` должно быть заменено на реальный TCP/IP адрес серверного компьютера.
8) Путь к каталогу, содержащему драйвер `dbc_tcp` и утилиту `inl`, должен быть добавлен к переменной окружения PATH.
9) После выполнения очередного пункта проверки клиентский и серверный компьютеры должны быть приведены в исходное состояние, а именно:
   - файл `nodetab` к описанному выше;
   - процессы ядра СУБД ЛИНТЕР сетевого драйвера сервера и сетевого драйвера клиента, утилиты `inl` должны быть остановлены;
   - должны быть удалены или приведены к первоначальному состоянию все файлы, созданные или изменённые в процессе предыдущей проверки.
10) Под «попыткой установления соединения с узлом А» следует понимать запуск утилиты `inl` с ключами:

© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

```
>inl -u SYSTEM/MANAGER8 -n A
```

Проверка запуска

1) На клиентском компьютере выполнить команду:

```
>dbc_tcp
```

2) Проверить появление процесса `dbc_tcp` в списке процессов:

```
>ps -e | grep dbc_tcp
```

Проверка завершения работы

1) Запустить процесс сетевого клиента на клиентском компьютере.
2) Определить PID процесса сетевого клиента, просмотрев список всех процессов:

```
>ps -e | grep dbc_tcp
```

3) Послать процессу сетевого клиента сигнал TERM:

```
>kill PID_OF_DBC_TCP
```

   где `PID_OF_DBC_TCP` – идентификатор процесса сетевого сервера `dbc_tcp`.
4) Просмотрев список процессов, убедится в завершении процесса `dbc_tcp`.
5) Снова запустить процесс сетевого клиента на клиентском компьютере и убедиться в активности процесса `dbc_tcp`.
6) Выполнить команду завершения работы сетевого клиента:

```
>dbc_tcp -U
```

   Команда должна завершиться сообщением о посылке сигнала SIGTERM сетевому клиенту. В списке не должно остаться процесса с именем `dbc_tcp`.

Примечание
Команда завершения работы отсутствует в ЗОСРВ «Нейтрино».

Проверка протоколов, декларированных в файле nodetab

1) Создать файл `nodetab` со следующими узлами:

```
N1 TCPIP <remote node>
N2 TCPIPS <remote node>
N3 LOCAL 1234
N4 LOCALS 2501
N5 ATCPIP
N6 ATCPIPS
N7 REZ
N8 ABCDE
```

   `<remote node>` должно быть заменено на реальное доменное имя серверного компьютера.
2) Запустить драйвер `dbc_tcp` с указанием полного пути к созданному файлу `nodetab` из командной строки. При запуске будет выдана информация о неподдерживаемом протоколе ABCDE.

3) Последовательно осуществить попытку соединения с узлами N1-8:

- при попытке соединении с узлами N1–N4 будет возвращен код завершения 4006 «Ошибка создания сетевого соединения»;
- при попытке соединении с узлами N5–N7 будет возвращен код завершения 4066 «Сетевой протокол запрещен или не поддерживается»;
- при попытке соединении с узлом N8 будет возвращен код завершения 4050 «Неизвестное имя сервера».

Проверка IP-адреса в файле nodetab

1) На клиентском компьютере в отдельном каталоге создать файл `nodetab`:

```
N1 TCPIP 127.0.0.1
N2 TCPIP localhost
N3 TCPIP <remote ip>
N4 TCPIP abcd.abcd.abcd
N5 TCPIPS 127.0.0.1
N6 TCPIPS localhost
N7 TCPIPS <remote ip>
N8 TCPIPS abcd.abcd.abcd
```

   где `<remote ip>` должен быть заменен на числовой IP-адрес серверного компьютера.

2) Запустить драйвер `dbc_tcp` с указанием в командной строке пути к файлу `nodetab`.

3) При запуске должна быть напечатана информация о некорректности адреса в описании узла N4 и N8.

4) При попытке соединении с узлами N4, N8 будет возвращен код завершения 4050 «Неизвестное имя сервера».

5) При попытке соединении с узлами N1–N5 будет возвращен код завершения 4006 «Ошибка создания сетевого соединения»;

6) При попытке соединении с узлами N1–N3, N5–N8 будет возвращен код завершения 4006 «Ошибка создания сетевого соединения».

Проверка IP-соединения

1) На серверном компьютере запустить ядро СУБД ЛИНТЕР и сетевой драйвер сервера.

2) На клиентском компьютере запустить сетевой драйвер клиента.

3) Осуществить попытку соединения с узлами RTCP (описание файла `nodetab` для этой проверки приведено в пункте «Условия проверки»):

```
>inl -u SYSTEM/MANAGER8 -n RTCP
```

4) Должно быть получено приглашение:

```
>SQL
```

5) Выполнить запрос:

```
select count(*) from auto;
```

   На терминал должен быть выведен результат запроса.

 © Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

Проверка тайм-аута клиента

1) На серверном компьютере запустить ядро СУБД ЛИНТЕР и сетевой драйвер сервера.
2) На клиентском компьютере создать файл nodetab:

```
RTCP  TCPIP  <remote host> 1060 1 20
RTCPS TCPIPS <remote host> 1060 1 20
```

   где <remote host> должно быть заменено на реальный TCP/IP адрес серверного компьютера.
3) Запустить драйвер dbc_tcp с данным файлом nodetab.
4) Открыть соединение из утилиты inl к узлу RTCP.
5) Разорвать сеть на 30 секунд.
6) Восстановить работу сети.
7) Подать любой запрос из утилиты inl. Должен вернуться код завершения 1069 «Неверный номер канала».
8) Повторить действия для узла RTCPS.

Проверка тайм-аута соединения

1) На серверном компьютере запустить ядро СУБД ЛИНТЕР и сетевой драйвер сервера.
2) На клиентском компьютере создать файл nodetab:

```
RTCP  TCPIP  <alien host> 1060 1 20 10
RTCPS TCPIPS <alien host> 1060 1 20 10
```

   где <alien host> – сетевой адрес реального компьютера, расположенного за межсетевым экраном. Межсетевой экран должен быть настроен так, чтобы не пропускает пакеты от компьютера с адресом <alien host> к клиентскому компьютеру.
3) Запустить драйвер dbc_tcp с данным nodetab.
4) Осуществить попытку соединения к узлу RTCP с помощью утилиты inl.
5) Не более чем через 10(+4) сек. утилита inl должна выдать сообщение о невозможности определения версии БД. По истечении не более 10(+4) сек. – вернуть код завершения 4058 «Превышен интервал ожидания установления соединения» или 4006 «Ошибка создания сетевого соединения».
6) Повторить проверку для узла RTCPS.

Проверка соединения через локальный протокол

1) На серверном компьютере установить переменную окружения LINTER_MBX=1254.
2) Запустить ядро СУБД ЛИНТЕР.
3) Сбросить переменную окружения LINTER_MBX.
4) Проверить подключение к локальному ядру. Попытка открытия соединения к локальному ядру должна закончиться неудачно.
5) Создать файл nodetab:

```
N1 LOCAL  1254
```

```
N2 LOCALS 1254
```

6) На серверном компьютере запустить сетевой драйвер клиента.
7) Осуществить с помощью утилиты `inl` соединение к узлам N1, N2.
8) Соединение должно быть успешно установлено с узлами N1 и N2.
9) Выполнить запрос (при успешном соединении):

```
select count(*) from auto;
```

На терминал должен быть выведен результат выполнения запроса.

Проверка установки сервера по умолчанию

1) На серверном компьютере запустить ядро СУБД ЛИНТЕР и сетевой драйвер сервер.
2) На клиентском компьютере создать файл `nodetab`:

```
N1 TCPIP 12.12.12.12
N2 remote host
```

где <remote host> должно быть заменено на реальный TCP/IP адрес серверного компьютера.

3) Запустить драйвер `dbc_tcp` с ключом `-S` и данным `nodetab`:

```
>dbc_tcp -N nodetab -S N2
```

При запуске на терминал должно быть выведено сообщение о сервере по умолчанию N2.

4) Осуществить соединение к серверу по умолчанию:

```
>inl -u SYSTEM/MANAGER8
```

Соединение должно быть успешно установлено.

5) Выполнить запрос:

```
select count(*) from auto;*
```

На терминал должен быть выведен результат выполнения запроса.

6) Повторить шаги 1-5 заново, запустив драйвер `dbc_tcp` с ключом `-DEFAULT` вместо `-S`.

Проверка запуска с идентификатором межпроцессного обмена, отличным от идентификатора по умолчанию

1) На серверном компьютере запустить ядро СУБД ЛИНТЕР и сетевой драйвер сервера.
2) На клиентском компьютере запустить драйвер клиента `dbc_tcp` с ключом `-M`:

```
>dbc_tcp -M 1255
```

3) Установить переменную окружения `NET_MBX=1255`.
4) Произвести попытку соединения с узлом RTCP. Соединение должно быть успешно установлено.
5) Выполнить запрос:

```
select count(*) from auto;
```


© Сетевые средства. АО НПП «РЕЛЭКС», 1990-2026

На терминал должен быть выведен результат запроса.