Центр

Управления

Гетерогенными

Инфраструктурами

Руководство администратора ClearwayCA. Версия 26.2.0

ООО «Клируэй Текнолоджис»

2026

Оглавление

1 Введение

Настоящий документ — Руководство администратора программного обеспечения "Система централизованного анализа и управления гетерогенными инфраструктурами — Clearway CA" версии 26.1.0 (далее — Clearway CA или продукт). Разработчик — ООО "Клируэй Текнолоджис".

Clearway CA — это центр сертификации (Certification Authority) на основе стандарта X.509v3. Продукт выпускает сертификаты по запросам CSR в формате PKCS#10 с использованием настраиваемых шаблонов, отзывает сертификаты с указанием причины, формирует списки отозванных сертификатов (CRL и разностные DeltaCRL), предоставляет статусы сертификатов по протоколу OCSP и поддерживает автоматизированную выдачу сертификатов устройствам по протоколу SCEP. Все запросы, сертификаты, журналы и списки отзыва хранятся в базе данных; управление выполняется через веб-интерфейс (Контрольную панель) и API. Продукт построен из отдельных микросервисов и разворачивается на серверах под управлением ОС Linux.

1.1 Назначение документа

Руководство описывает установку, настройку, приведение в безопасное состояние и сопровождение Clearway CA. Оно охватывает как автоматическую установку через штатный инсталлятор, так и ручное развертывание и настройку отдельных сервисов, а также содержит справочные сведения по сообщениям журналов, сетевым портам, ролям доступа и типовым эксплуатационным задачам.

1.2 Область применения

Документ применяется при развертывании и эксплуатации Clearway CA версии 26.1.0 в рамках инфраструктуры открытых ключей (PKI) организации: при первичной установке продукта, настройке его компонентов и СУБД, обеспечении безопасного режима работы, а также при последующем обслуживании и обновлении сервисов.

1.3 Целевая аудитория

Руководство предназначено для администраторов, отвечающих за установку, настройку, сопровождение и обеспечение безопасной работы Clearway CA. Предполагается, что читатель владеет базовыми навыками администрирования ОС Linux (в том числе управления службами через systemctl), работы с СУБД PostgreSQL, а также понимает основные принципы работы инфраструктуры открытых ключей (PKI), сертификатов X.509 и протоколов TLS/mTLS.

1.4 Основные функции

Ключевые возможности Clearway CA:

- выпуск сертификатов X.509v3 по запросам CSR (PKCS#10) на основе настраиваемых шаблонов;
- отзыв сертификатов с указанием причины и формирование списков отзыва CRL и DeltaCRL;
- предоставление статуса сертификатов по протоколу OCSP (сервис mOCSP);
- автоматизированная выдача сертификатов устройствам по протоколу SCEP (адаптер Clearway CA);
- хранение запросов, сертификатов, шаблонов, журналов и списков отзыва в базе данных PostgreSQL;

- ведение журнала событий с контролем целостности и отдельного отказоустойчивого журнала безопасности;
- резервное копирование и восстановление ключей (механизм KRA), шифрование данных сертификатов в БД;
- опциональное хранение ключа ЦС на аппаратном модуле HSM (через интерфейс PKCS#11);
- управление через веб-интерфейс (Контрольную панель) и API;
- аутентификация и авторизация — через сервер OpenID Connect с разграничением доступа по ролям.

Из протокольных адаптеров в состав штатного инсталлятора входит только SCEP. Другие протоколы (ACMEv2, MS-RPC, CMP, EST, MS-WSTEP) инсталлятором не разворачиваются.

2 Архитектура и состав системы

Clearway CA — модульная система удостоверяющего центра. Основу составляет ядро ЦС (сервис MiniCA), вокруг которого работает набор сервисов аутентификации, управления, проверки статуса и распространения сертификатов. Все прикладные сервисы запускаются как пользовательские systemd-юниты (systemctl --user) от технологической учетной записи (по умолчанию itc-svc); веб-сервисы публикуются наружу через Nginx. Хранилищем состояния служит PostgreSQL (для SCEP-адаптера — локальный SQLite).

Модульный состав:

Компонент	Назначение
Ядро ЦС (minica)	Сетевой REST-сервис удостоверяющего центра: выпуск и отзыв X.509-сертификатов по шаблонам, ведение CRL/DeltaCRL, ролевая модель на JWT. Одна программа работает и корневым, и выдающим ЦС — роль определяется конфигурацией. Конфигурация — *minica.yml*
Сервис аутентификации OpenID (openid)	OpenID Connect провайдер (форк Dex): единый вход для компонентов, аутентификация через LDAP/Active Directory, маппинг групп каталога на роли Clearway CA, выдача access/refresh-токенов
Контрольная панель (ControlPanel)	Графическая панель управления ЦС: серверная часть ASP.NET (ITC.Api.MiniCA.ControlPanel), одностраничное приложение (ITC.MiniCA.Spa) и конфигурация Nginx (обратный прокси и раздача статики). Вход по OpenID/JWT
Сервис публикации (Publication)	Сервис ITC.Api.MiniCA.Publication: публикация сертификатов и CRL на точки распространения, в том числе в каталог LDAP, по настраиваемым заданиям
OCSP-респондер (mOCSP)	Сервис mocsp: проверка статуса сертификатов в реальном времени по протоколу OCSP; статусы читаются из БД ЦС
Архиватор (Archiver)	Сервис ITC.Api.MiniCA.Archiver: перенос истекших сертификатов из рабочей БД ЦС в отдельную архивную БД по расписанию (@daily)
SCEP-адаптер (scepAdapter)	Сервис itc.acm.scepAdapter.clearwayCa: выпуск и обновление сертификатов для сетевых устройств по протоколу SCEP; аутентификация через OpenID, локальное состояние в SQLite
Точка распространения CRL и сертификатов (CDP)	Веб-сервер Nginx, раздающий по HTTP статические файлы CRL (CDP) и сертификатов ЦС (AIA) из каталогов публикации
Консольный клиент (mclient)	CLI для обращения к ядру ЦС: выпуск сертификатов, управление шаблонами, работа с JWT. Права клиента ограничены ролью в его токене

Из протокольных адаптеров в дистрибутив штатно входит только SCEP. Адаптеры ACMEv2, MS-RPC, CMP, EST установщиком не разворачиваются; WSTEP-адаптер в состав установщика не входит.

Порты и сетевые интерфейсы:

Сервис	Порт (по умолчанию)	Протокол	Публикация
Ядро ЦС (minica)	9080 / 9443	HTTP / HTTPS (mTLS)	напрямую
OpenID	443	HTTPS	напрямую (требуется setcap cap_net_bind_service)
OpenID (админ-интерфейс)	5559	HTTP	локально
Контрольная панель (backend)	8407	HTTP	за Nginx (443)
Сервис публикации	9407	HTTP	за Nginx (443)
OCSP-респондер	8888 (/ocsp)	HTTP	напрямую
Архиватор	9507	HTTP	локально
SCEP-адаптер	8440 (/api/scep)	HTTP	за Nginx (443)
CDP	80	HTTP	напрямую
PostgreSQL	5432	TCP	БД компонентов

По умолчанию TLS ядра ЦС включается на этапе установки, максимально допустимое рассогласование времени при проверке подписи запросов и токенов — max-dt-sec = 600 (окно ±10 минут), поэтому на всех узлах ЦС и на сервере СУБД необходима синхронизация времени (NTP).

Схема развертывания:

Компоненты можно размещать как на одном сервере, так и распределять по нескольким узлам. Типовые конфигурации:

- Базовая (1 сервер): ядро ЦС, PostgreSQL и необходимые сервисы (OpenID, Контрольная панель, OCSP, CDP, при необходимости SCEP) устанавливаются на одну машину. Подходит для стендов, пилотов и небольших инсталляций.
- Без резервирования (2 сервера): один сервер несет корневой ЦС (как правило, вынесенный и оффлайновый), второй — выдающий ЦС с сопутствующими сервисами и БД. Разделение снижает риски компрометации корневого ключа.

- Распределенная: каждая функциональная группа (корневой ЦС, выдающий ЦС, аутентификация OpenID, Контрольная панель, публикация, OCSP, архивирование, SCEP, CDP) выносится на отдельные серверы; БД разворачиваются на выделенных узлах PostgreSQL. Обеспечивает масштабирование и изоляцию ролей.

Развертывание выполняется веб-мастером установки, который оркестрирует установку компонентов на целевые серверы по SSH. Мастер сам на целевые серверы не устанавливается.

Технологический стек:

- Взаимодействие: HTTP(S), TLS/mTLS; на границе — Nginx как обратный прокси (443 для веб-сервисов, 80 для CDP).
- Криптография: OpenSSL (ядро ЦС — обёртка над CLI OpenSSL); опционально аппаратные носители/HSM через OpenSSL engine (pkcs11).
- Языки и платформы: ядро ЦС и OCSP-респондер — Go; прикладные сервисы (Контрольная панель, публикация, архиватор, SCEP-адаптер) — C#/.NET 8; веб-интерфейс — SPA.
- Внешние компоненты: PostgreSQL (версии 11–18) как СУБД; Nginx как веб-сервер и обратный прокси; для SCEP-адаптера — встроенный SQLite.

Взаимодействие компонентов:

Сервисы взаимодействуют по HTTP(S) API. Авторизация построена на JWT: сервисы обращаются к ядру ЦС с токеном, выпущенным на самом ЦС (JWT + приватный ключ), а пользовательские запросы проходят аутентификацию через OpenID и авторизуются по ролям (claim roles OIDC-токена). Роли назначаются членством в группах LDAP/AD (карта соответствия задается в конфигурации OpenID). Ядро ЦС дополнительно проверяет подпись входящих запросов мастер-ключом клиента.

3 Требования к среде

В разделе приведены требования к операционной системе, аппаратным ресурсам, системному ПО, СУБД, рабочим местам администратора и оператора, сетевым портам и технологической учетной записи, необходимые для установки и эксплуатации Clearway CA 26.1.0.

Операционная система:

Серверные компоненты Clearway CA устанавливаются на 64-разрядные ОС семейства Linux. Поддерживаются:

ОС	Версия
Astra Linux Special Edition (РУСБ.10015-01)	очередное обновление 1.8
РЕД ОС	8
Альт 8 СП	релиз 10

Установщик определяет семейство ОС автоматически (altlinux, redos, прочие — семейство Debian, включая Astra Linux) и выбирает менеджер пакетов (apt или dnf). На всех узлах должно быть настроено единое системное время (NTP): проверка меток времени в запросах к ЦС использует окно рассогласования max-dt-sec = 600 (±10 минут); при большем расхождении запрос отклоняется.

Аппаратные требования:

Приведенные конфигурации виртуальных ресурсов (vCPU / vRAM / vHDD) рассчитаны на разную нагрузку по числу выпускаемых сертификатов.

Минимальная конфигурация — один сервер, все компоненты Clearway CA и СУБД PostgreSQL:

Узел	Компоненты	vCPU / vRAM / vHDD	Нагрузка
Единый сервер	Все компоненты + СУБД	4 / 4 ГБ / 256 ГБ	до 100 сертификатов в сутки

Базовая конфигурация — один сервер, все компоненты и СУБД, повышенная нагрузка:

Узел	Компоненты	vCPU / vRAM / vHDD	Нагрузка
Единый сервер	Все компоненты + СУБД	4 / 8 ГБ / 512 ГБ	до 5000 сертификатов в сутки

Типовая конфигурация — компоненты разнесены по отдельным узлам:

Узел	Компоненты	vCPU / vRAM / vHDD	Системное ПО
Центр сертификации	Сервис управления сертификатами (ЦС)	2 / 2 ГБ / 80 ГБ	PostgreSQL
Аутентификация	OpenID	2 / 2 ГБ / 40 ГБ	PostgreSQL
Вспомогательные сервисы	Контрольная панель, публикация, архиватор, OCSP, SCEP, CDP	2 / 4 ГБ / 40 ГБ	PostgreSQL, nginx, SMTP-сервер

Каталог журналов ЦС (*/opt/itc/minica/logs*) рекомендуется размещать на разделе с запасом свободного места и контролировать его заполнение средствами ОС.

Системное ПО:

На серверных узлах устанавливается следующее ПО (как правило, из репозитория ОС или из состава поставки):

ПО	Минимальная версия	Назначение
.NET (ASP.NET Core Runtime)	8.0	Среда выполнения .NET-сервисов (контрольная панель, публикация, архиватор). Компоненты SCEP, OCSP и ядро ЦС поставляются как самодостаточные (self-contained) сборки и отдельной установки .NET не требуют
nginx	из репозитория	Обратный прокси и TLS-терминация для веб-сервисов (контрольная панель, публикация, SCEP, CDP)
curl	из репозитория	Обращения к API системы из скриптов
jq	из репозитория	Обработка JSON при формировании токенов и настройке сервисов
OpenSSL	из репозитория	Генерация ключей, запросов (CSR) и сертификатов на узлах ЦС
psql (клиент PostgreSQL)	не ниже версии сервера БД	Создание и обслуживание БД и таблиц

СУБД PostgreSQL:

Состояние ЦС, сервиса аутентификации и архиватора хранится в PostgreSQL.

- Поддерживаемые версии — PostgreSQL 11–18; минимально допустимая — 11.
- При установке мастер автоматически подбирает наибольшую доступную версию из диапазона (от 18 к 11); если ни одна не обнаружена, используется запасное значение 16.

- Та же логика применяется к БД сервиса аутентификации (OpenID).

Для отказоустойчивого развертывания PostgreSQL поддерживается работа в кластере на базе etcd и patroni (устанавливаются из репозитория ОС на узлах СУБД).

> Подключение сервисов к БД по умолчанию разрешено без TLS (database.allow-without-tls: true). Для защищенного состояния канал "сервис–БД" следует переводить на TLS (allow-without-tls: false, sslmode=verify-full). Подробнее см. в разделе настройки конфигурации ЦС.

АРМ администратора и оператора:

Рабочее место — ПК под управлением любой ОС, поддерживающей рекомендуемый браузер и допускающей применение мер защиты информации.

Параметр	Значение
Ядер / частота	2 / не менее 2 ГГц
RAM	4 ГБ
Диск	80 ГБ SSD
Браузер	Google Chrome версии не ниже 113 или браузер на базе Chromium (Microsoft Edge, Яндекс.Браузер)

Дополнительное ПО для АРМ администратора:

ПО	Минимальная версия	Назначение
DBeaver (+ утилиты pg_dump, pg_dumpall, pg_restore, psql, драйвер PostgreSQL)	21.3.1	Настройка и обслуживание БД
WinSCP	5.13.7	Доступ к файлам серверов
PuTTY	0.70	Доступ к серверам по SSH
Google Chrome	113	Доступ к веб-интерфейсу системы

Сетевые порты:

Для настройки правил межсетевого экрана используются перечисленные ниже порты. Веб-сервисы публикуются через Nginx (порт 443, TLS), а сами приложения слушают локальные порты и извне напрямую недоступны.

Служба / узел	Порт/протокол	Назначение
Веб-интерфейс (Nginx)	80/tcp, 443/tcp	Доступ к веб-консоли; на 443 — TLS. Порт 80 используется, в частности, точкой распространения CRL (CDP)
Центр сертификации (ЦС)	9443/tcp	Защищенный интерфейс ЦС (mTLS); обращения mclient и сервисов к ЦС
Аутентификация (OpenID)	443/tcp	HTTPS OIDC-провайдера
Аутентификация (OpenID)	5559/tcp	Служебный HTTP-интерфейс администрирования
Аутентификация (OpenID)	5432/tcp	Подключение к БД OpenID (PostgreSQL)
SCEP-адаптер	8440/tcp	Локальный порт приложения SCEP (публикуется через Nginx: *https://<fqdn>/api/scep*)
Серверы продукта (ОС)	22/tcp	Администрирование по SSH (OpenSSH)
Сервер БД	5432/tcp	Подключение к PostgreSQL для настройки и сопровождения

Внутренние порты .NET-сервисов (контрольная панель — 8407, публикация — 9407, OCSP — 8888) используются только для локального проксирования через Nginx и в правилах внешнего брандмауэра, как правило, не открываются.

Требования к учетной записи:

Все серверные сервисы Clearway CA работают от имени **технологической учетной записи-владельца** (по умолчанию itc-svc), а не от root. Каталог установки по умолчанию — */opt/itc* (для ЦС — */opt/itc/minica*), владельцем каталога должна быть эта учетная запись.

Сервисы запускаются как **пользовательские systemd-юниты** (`systemctl --user`), поэтому для учетной записи-владельца обязательны:

- включенный режим сохранения сеанса — `enable-linger` (`loginctl enable-linger <учетная запись>`), иначе пользовательские юниты не стартуют при отсутствии интерактивного входа;
- настроенное окружение для управления юнитами — переменные XDG_RUNTIME_DIR и (при необходимости) DBUS_SESSION_BUS_ADDRESS, прописываемые в *~/.bashrc*.

```
# Проверка режима linger для учетной записи-владельца
loginctl show-user itc-svc | grep Linger
```

```
# ожидается Linger=yes
# Включение при необходимости
loginctl enable-linger itc-svc
```

Наличие учетной записи, каталога установки, корректного владельца, режима linger, а также OpenSSL и Nginx проверяется установщиком на этапе контроля пререквизитов; при отсутствии предлагается автоматическое исправление.

4 Общие сведения об установке

Инсталлятор поставляется также в версии под ОС Linux — она предоставляется по запросу заказчика.

Clearway CA устанавливается через веб-инсталлятор — самодостаточное приложение (backend на ASP.NET Core, frontend на Angular), которое запускается локально и открывается в браузере по адресу https://localhost:5224/installer. Сам мастер на целевые серверы не ставится: он оркестрирует установку, а компоненты разворачивает удаленно по SSH от имени технологической учетной записи. Host и Username для подключения задаются в параметрах каждого компонента.

Ход и результаты установок (введенные значения, выпущенные сертификаты, состояние компонентов) сохраняются в базе установщика. Это позволяет продолжать начатый сценарий и переиспользовать выходы одних компонентов как входы других (например, CSR, выпущенный на одном шаге, автоматически подставляется в шаг подписания на ЦС).

Детальные пошаговые инструкции по каждому экрану мастера (перечень полей, значения по умолчанию, порядок нажатий) вынесены из настоящего руководства и поставляются вместе с инсталлятором. Ниже приведен обзор состава установки и общего порядка действий.

Группы установки:

Компоненты объединены в восемь групп — законченных сценариев развертывания. Группа выбирается на старте и определяет, какие компоненты и в каком порядке будут установлены.

Группа	Что разворачивает	Состав компонентов
Корневой ЦС	Самоподписанный корневой центр сертификации на сервисе MiniCA с базой данных	База данных ЦС (PostgreSQL) → Корневой ЦС
Выдающий ЦС	Подчиненный (выдающий) ЦС: выпуск CSR, подписание на корневом ЦС, установка сертификата и запуск	База данных ЦС → Выдающий ЦС (CSR) → Выпуск сертификата на корневом ЦС → Настройка выдающего ЦС
Аутентификация (OpenID)	OIDC-провайдер аутентификации (форк Dex) с привязкой к LDAP/AD	CSR для TLS → Выпуск сертификата → База данных OpenID → Сервис OpenID
Контрольная панель	Графическая панель управления ЦС (backend + SPA) за Nginx, вход по OpenID	CSR для TLS → Выпуск сертификата → Контрольная панель → Выпуск JWT → Подключение ЦС к панели
Сервис публикации	Сервис публикации сертификатов и CRL на точки распространения (в т.ч. LDAP) за Nginx	CSR для TLS → Выпуск сертификата → Сервис публикации → Выпуск JWT

Группа	Что разворачивает	Состав компонентов
OCSP	OCSP-респондер (mOCSP), читающий статусы из базы данных ЦС	OCSP-респондер (CSR подписанта) → Выпуск сертификата подписи → Настройка OCSP
Архиватор	Сервис переноса истекших сертификатов из рабочей базы в архивную по расписанию	Архивная база данных → Архиватор
SCEP	SCEP-адаптер выпуска сертификатов для устройств за Nginx, аутентификация через OpenID, состояние в SQLite	CSR (TLS + SCEP) → Выпуск сертификатов → SCEP-адаптер → Выпуск JWT → Настройка SCEP

Из протокольных адаптеров штатно устанавливается только SCEP. Адаптеры ACMEv2, MS-RPC, CMP, EST и WSTEP в состав инсталлятора не входят.

Отдельно вне групп поставляется веб-сервер точек распространения CDP (публикация CRL и сертификатов по HTTP на порту 80).

Общий порядок установки:

Для любой группы мастер проходит одну и ту же последовательность этапов:

1. Выбор группы — задается сценарий и, соответственно, набор и порядок компонентов.
2. Сбор параметров — по компонентам заполняются данные подключения (SSH-хост, учетная запись, каталог установки), реквизиты БД, TLS/сертификаты, параметры LDAP и т. п. Значения, полученные на предыдущих шагах, подставляются автоматически.
3. Проверка и автопочинка пререквизитов — для каждого компонента выполняются проверки готовности сервера (Check). Там, где предусмотрено исправление, кнопка настройки пререквизитов запускает автопочинку (Fix): создание учетной записи, каталога и прав, включение linger, установка недостающих пакетов и т. д.
4. Установка по SSH — на сервер загружается полезная нагрузка (бинарные файлы, конфигурации), выполняются скрипты установки от имени технологической учетной записи, запускаются пользовательские systemd-юниты.
5. Результаты — фиксируются и становятся доступны для скачивания выходные файлы: сертификаты (корневой, TLS, цепочка), CSR, пары JWT + зашифрованный ключ. Эти результаты используются как входные данные последующих компонентов и групп.

Общие пререквизиты:

Перед развертыванием компонента мастер проверяет (и при возможности автоматически устраняет) следующие условия на целевом сервере:

Пререквизит	Проверка	Автопочинка
Технологическая учетная запись	наличие пользователя (по умолчанию itc-svc)	создание учетной записи

Пререквизит	Проверка	Автопочинка
Каталог установки и права	наличие каталога (по умолчанию */opt/itc*, для ЦС — */opt/itc/minica*) и корректный владелец	создание каталога, chown
Запуск пользовательских служб	включен enable-linger для учетной записи	включение linger
Nginx	служба Nginx активна (для веб-компонентов)	установка Nginx
OpenSSL	наличие openssl (для компонентов ЦС)	установка пакета
Доступ к базе данных	подключение к PostgreSQL и наличие прав	зависит от компонента

Большинство сервисов работают как пользовательские systemd-юниты (`systemctl --user`), поэтому для технологической учетной записи обязателен enable-linger и настройка окружения (XDG_RUNTIME_DIR и при необходимости DBUS_SESSION_BUS_ADDRESS). Веб-компоненты (Контрольная панель, Сервис публикации, SCEP, CDP) публикуются через Nginx как reverse-proxy на локальный порт приложения. Поддерживаются версии PostgreSQL 11–18; по умолчанию установщик подбирает наибольшую доступную.

5 Ручное развертывание сервисов (без инсталлятора)

Настоящий раздел описывает развертывание и настройку сервисов Clearway CA вручную, без веб-инсталлятора. Ручное развертывание применяют, когда использование инсталлятора невозможно или нежелательно. Порядок действий соответствует автоматической установке: сначала разворачиваются база данных и ядро ЦС, затем — вспомогательные сервисы. Все прикладные сервисы работают как пользовательские systemd-юниты от технологической учетной записи (по умолчанию itc-svc). Подробное описание параметров конфигураций приведено в разделах "Настройка ядра ЦС" и "Настройка вспомогательных сервисов".

Команды используют значения по умолчанию (учетная запись itc-svc, каталог */opt/itc*). При иных значениях подставьте свои. Предварительно ознакомьтесь с разделами Требования к среде и Обеспечение безопасной среды.

Общая подготовка сервера:

Действия выполняются на каждом сервере ЦС (корневого и/или выдающего). Команды с sudo выполняются от учетной записи с административными правами; остальные — от технологической учетной записи-владельца (по умолчанию itc-svc). Значения по умолчанию: учетная запись itc-svc, каталог установки */opt/itc/minica*. При иных значениях подставьте свои.

1. Создайте технологическую учетную запись-владельца. От её имени работает служба ЦС и ей принадлежит каталог установки.

```
# Debian/Ubuntu/Astra (apt):
sudo adduser --disabled-password --gecos "" itc-svc
# AltLinux/RedOS:
sudo adduser itc-svc
```

2. Создайте каталог установки и назначьте для него права:

```
sudo mkdir -p /opt/itc/minica
sudo chown itc-svc:itc-svc /opt/itc/minica
# проверка:
stat -c "%U:%G" /opt/itc/minica
# ожидается: itc-svc:itc-svc
```

3. Включите enable-linger. Это позволяет пользовательским службам работать после выхода учетной записи из системы (обязательно для `systemctl --user`).

```
sudo loginctl enable-linger itc-svc
# проверка:
loginctl show-user itc-svc | grep --color=never Linger
# ожидается: Linger=yes
```

4. Настройте окружение пользовательских служб. Настройка выполняется от имени itc-svc (`su - itc-svc`) и прописывает XDG_RUNTIME_DIR и адрес шины D-Bus в *~/.bashrc* — без них не работает `systemctl --user.`

```
echo "export XDG_RUNTIME_DIR=/run/user/$UID" >> ~/.bashrc
export XDG_RUNTIME_DIR=/run/user/$UID
echo "export DBUS_SESSION_BUS_ADDRESS=unix:path=${XDG_RUNTIME_DIR}/bus" >> ~/.bashrc
export DBUS_SESSION_BUS_ADDRESS="unix:path=${XDG_RUNTIME_DIR}/bus"
```

5. Установите OpenSSL. Требуется ядру ЦС для генерации ключей и работы с сертификатами.

```
sudo apt install -y openssl    # apt
sudo dnf install -y openssl    # dnf
command -v openssl             # проверка
```

6. Установите Nginx. Ядру ЦС напрямую не требуется, но необходим компонентам, публикующим данные по HTTP/HTTPS (точки распространения CDP/AIA, публикация, Контрольная панель, SCEP-адаптер). Устанавливайте, если сервер совмещает такие компоненты:

```
sudo apt install -y nginx       # apt
sudo dnf install -y nginx       # dnf
# RedOS дополнительно (SELinux):
sudo setsebool -P httpd_can_network_connect 1
sudo systemctl enable --now nginx
```

5.1 Развертывание базы данных ЦС

БД ЦС хранит сведения о выпущенных сертификатах, шаблонах, ключах JWT и журнал событий. Разворачивается на сервере СУБД. Поддерживаются PostgreSQL 11–18 (по умолчанию установщик выбирает наибольшую доступную версию). По умолчанию: учетная запись-владелец и имя БД — cwica (для сервера архивирования — cwica_archive).

1. Установите PostgreSQL (если не установлен):

```
# apt (Debian/Ubuntu/Astra) — кластер инициализируется автоматически:
sudo apt install -y postgresql-16 postgresql-contrib-16
# dnf (RedOS) — требуется ручная инициализация:
sudo dnf install -y postgresql16-server postgresql16-contrib
sudo /usr/pgsql-16/bin/postgresql-16-setup initdb
sudo systemctl enable --now postgresql-16
# проверка готовности:
pg_isready -q && echo active || echo inactive
```

2. Создайте учетную запись-владельца и базу данных (выполняется от имени системного пользователя postgres. Замените <ПАРОЛЬ> на заданный пароль):

```
sudo -u postgres psql -c "CREATE USER \"cwica\" WITH PASSWORD '<ПАРОЛЬ>';"
sudo -u postgres psql -c "CREATE DATABASE cwica OWNER \"cwica\";"
sudo -u postgres psql -c "GRANT CONNECT, CREATE ON DATABASE \"cwica\" TO \"cwica\";"
sudo -u postgres psql -d cwica -c "GRANT USAGE, CREATE ON SCHEMA public TO \"cwica\";"
```

3. Загрузите схему. Для этого скопируйте файл схемы *minica.up.sql* из дистрибутива на сервер СУБД (например, в */tmp*) и примените его к БД cwica:

```
PGPASSWORD='<ПАРОЛЬ>' psql -U cwica -h localhost -d cwica -f /tmp/minica.up.sql
```

4. Убедитесь, что БД доступна с сервера ЦС по сети (порт 5432, правило брандмауэра):

```
PGPASSWORD='<ПАРОЛЬ>' psql -U cwica -h <db-host> -p 5432 -d cwica -tAc "SELECT 1"   #
ожидается: 1
```

5.2 Развертывание ядра ЦС (minica)

Выполняется на сервере ЦС от имени технологической учетной записи itc-svc (после разделов Общая подготовка сервера и Развертывание базы данных ЦС). Каталог установки — */opt/itc/minica*.

1. Скопируйте файлы дистрибутива (артефакты ядра) в каталог установки согласно таблице:

Файл дистрибутива	Размещение на сервере	Назначение
minica	*/opt/itc/minica/minica*	исполняемый файл службы ЦС
mclient	*/opt/itc/minica/mclient*	клиент управления ЦС
mjwt	*/opt/itc/minica/bin/mjwt*	утилита выпуска JWT
minica.yml	*/opt/itc/minica/conf/minica.yml*	конфигурация ЦС
mclient.yml	*/opt/itc/minica/conf/mclient.yml*	конфигурация mclient
openssl.cnf	*/opt/itc/minica/conf/openssl.cnf*	базовая конфигурация OpenSSL
minica.env	*/opt/itc/minica/minica.env*	переменные окружения службы (MINICA_CONF)
minica.service	*/home/itc-svc/.config/systemd/user/minica.service*	пользовательский systemd-юнит
make-jwt-key.sh	*/opt/itc/minica/script/make-jwt-key.sh*	генерация ключа подписи JWT

Файл дистрибутива	Размещение на сервере	Назначение
make-mclient-key.sh	*/opt/itc/minica/script/make-mclient-key.sh*	генерация ключевой пары mclient
make-mclient-jwt.sh	*/opt/itc/minica/script/make-mclient-jwt.sh*	выпуск мастер-JWT

2. Создайте директории и назначьте права на исполнение:

```
mkdir -p /opt/itc/minica/{conf,bin,script,logs,ca/certs,ca/crl}
mkdir -p ~/.config/systemd/user
chmod +x /opt/itc/minica/minica /opt/itc/minica/mclient /opt/itc/minica/bin/mjwt \
         /opt/itc/minica/script/make-jwt-key.sh \
         /opt/itc/minica/script/make-mclient-key.sh \
         /opt/itc/minica/script/make-mclient-jwt.sh
```

3. Подставьте пути установки в конфигурационные файлы. Файлы дистрибутива содержат служебный placeholder $PATH (путь установки), который при ручном развертывании нужно заменить на фактический каталог.

```
sed -i 's#\$PATH#/opt/itc/minica#g' \
  /opt/itc/minica/minica.env /opt/itc/minica/conf/minica.yml \
  /opt/itc/minica/conf/mclient.yml \
  /home/itc-svc/.config/systemd/user/minica.service
```

4. В *conf/minica.yml* задайте параметры подключения к БД (секция database). Спецсимволы пароля в URL укажите в процентном кодировании.

```
database:
  type: postgres
  url: postgres://cwica:<ПАРОЛЬ>@<db-host>:5432/cwica
  # По умолчанию true. Для запрета подключения к БД без TLS укажите false
  # и добавьте в url опцию sslmode=verify-full.
  allow-without-tls: true
```

5. Создайте самоподписанный сертификат для корневого ЦС. Атрибут Subject укажите в формате OpenSSL (через /).

```
openssl genrsa -out /opt/itc/minica/ca/ca.key 4096
openssl req -x509 -new -nodes -key /opt/itc/minica/ca/ca.key -sha256 -days 2920 \
  -out /opt/itc/minica/ca/ca.crt -subj "/CN=RootCA/O=Company/C=RU"
cp /opt/itc/minica/ca/ca.crt /opt/itc/minica/ca/chain.pem
```

В этом примере для корневого ЦС создается самоподписанный сертификат сроком 8 лет (2920 дней).

6. Вариант — для выдающего (подчиненного) ЦС вместо самоподписи сформируйте запрос, который подписывается на корневом ЦС по шаблону subca, после чего собирается цепочка "корневой ЦС + выдающий ЦС":

```
openssl genrsa -out /opt/itc/minica/ca/ca.key 4096
openssl req -new -key /opt/itc/minica/ca/ca.key -out /opt/itc/minica/ca/ca.csr \
  -subj "/CN=Issuing CA/O=Company/C=RU"
# передать ca.csr на корневой ЦС и подписать по шаблону subca:
/opt/itc/minica/mclient ca -csr /tmp/ca.csr -out /tmp/sub-ca.crt -template subca
# вернуть подписанный сертификат и собрать цепочку:
cp /tmp/sub-ca.crt /opt/itc/minica/ca/ca.crt
cat /tmp/root-ca.crt /opt/itc/minica/ca/ca.crt > /opt/itc/minica/ca/chain.pem
```

7. Создайте ключи и мастер-JWT для mclient.
Указанные в командах ниже скрипты формируют ключ подписи JWT, ключевую пару mclient и первичный (мастер) JWT. Тип ключей — ED25519; мастер-JWT выпускается с ролью admin (полный доступ к API), бессрочный. Запускайте скрипты строго в указанном порядке из каталога установки:

```
/opt/itc/minica/script/make-jwt-key.sh       # -> conf/jwt.key, conf/jwt-keys/<SKI>.pem
/opt/itc/minica/script/make-mclient-key.sh   # -> conf/mclient.key (0600), conf/
mclient.pem
/opt/itc/minica/script/make-mclient-jwt.sh   # -> conf/mclient.jwt (name=master,
role=admin)
```

8. Запустите службы. Служба ЦС — пользовательский systemd-юнит minica.service.

```
systemctl --user daemon-reload
systemctl --user enable minica.service
systemctl --user start minica.service
```

9. Выполните проверку, перед вызовом mclient укажите путь к его конфигурации:

```
export MCLIENT_CONF=/opt/itc/minica/conf/mclient.yml
systemctl --user status minica.service
/opt/itc/minica/mclient ping        # ожидается успешный ответ службы
```

10. Создайте базовые шаблоны.
Для корневого ЦС создаются шаблоны tls, subca, ocsp; для выдающего ЦС — tls и ocsp. Шаблон tls необходим до выпуска TLS-сертификата на следующем шаге.

```
/opt/itc/minica/mclient addtempl -requester mclient -name "tls" -desc "tls server&client"
 \
  -days "365" -key-type "RSA" -key-size "2048" -exts "tls_client_cert" \
```

```
  -ku "critical,digitalSignature,keyEncipherment" -eku "serverAuth,clientAuth" \
  -ski "hash" -aki "keyid,issuer"
/opt/itc/minica/mclient addtempl -requester mclient -name "subca" -desc "Subordinate CA"
 \
  -days "1460" -key-type "RSA" -key-size "2048" -bc "critical,CA:true,pathlen:0" \
  -ku "critical,cRLSign,digitalSignature,keyCertSign" -ski "hash" -aki "keyid,issuer"
/opt/itc/minica/mclient addtempl -requester mclient -name "ocsp" -desc "OCSP Signing" \
  -days "365" -key-type "RSA" -key-size "2048" -bc "CA:FALSE" \
  -ku "critical,digitalSignature" -eku "OCSPSigning" -ski "hash" -aki "keyid,issuer"
```

11. Включите TLS.
    По умолчанию ЦС слушает HTTP на порту 9080. Для перехода на защищенный канал выпускается TLS-сертификат ЦС (на собственное имя <host> по шаблону tls), затем в конфигурациях включается TLS и служба перезапускается (HTTPS-порт 9443):

```
# 1) Запрос и выпуск TLS-сертификата ЦС:
openssl req -new -keyout /opt/itc/minica/conf/tls.key -out /opt/itc/minica/conf/tls.req \
  -sha256 -nodes -subj "/CN=<host>" -addext "subjectAltName = DNS:<host>" \
  -addext "extendedKeyUsage = serverAuth, clientAuth"
/opt/itc/minica/mclient ca -csr /opt/itc/minica/conf/tls.req \
  -out /opt/itc/minica/conf/tls.crt -template tls
# 2) Включить TLS (tls.enable: false -> true) и задать server-url в conf/minica.yml и
 conf/mclient.yml
# 3) Перезапуск и проверка по HTTPS:
systemctl --user restart minica.service
/opt/itc/minica/mclient ping
```

> ⚠ Приватные ключи (*ca/ca.key, conf/mclient.key, conf/tls.key*) и мастер-JWT дают полный доступ к ЦС. Ограничьте права на файлы (см. раздел Обеспечение безопасной среды) и не передавайте их по незащищенным каналам.

5.3 Развертывание сервиса аутентификации OpenID

Сервис OpenID — провайдер OpenID Connect (форк Dex), обеспечивает единый вход пользователей во все компоненты, аутентифицирует их через каталог LDAP/Active Directory и сопоставляет группы каталога ролям Clearway CA. Состояние (сессии, refresh-токены, ключи подписи) хранится в собственной БД PostgreSQL. Сервис слушает HTTPS на порту 443 и служебный HTTP администрирования на порту 5559.

Предусловия:

- каталог */opt/itc* создан и принадлежит itc-svc;
- включен linger;
- настроено окружение `systemctl --user`;
- установлен LDAP-клиент (ldap-utils для apt, openldap-clients для dnf).

По настройке предусловий см. раздел Общая подготовка сервера.

Пути в инструкции ниже приведены для значений по умолчанию: PATH=*/opt/itc*, USER_NAME=itc-svc.

1. Создайте базу данных OpenID. Схему сервис создает сам при первом запуске — нужны только владелец и пустая БД (по умолчанию пользователь cwi, БД openid). На сервере PostgreSQL от postgres выполните команды:

```
sudo -u postgres psql -c "CREATE USER \"cwi\" WITH PASSWORD '<ПАРОЛЬ>';"
sudo -u postgres psql -c "CREATE DATABASE openid OWNER \"cwi\";"
sudo -u postgres psql -c "GRANT CONNECT, CREATE ON DATABASE \"openid\" TO \"cwi\";"
sudo -u postgres psql -d openid -c "GRANT USAGE, CREATE ON SCHEMA public TO \"cwi\";"
```

2. Создайте TLS-сертификат сервиса. Для этого на сервере OpenID сгенерируйте ключ и CSR, подпишите его на ЦС по шаблону tls:

```
mkdir -p /opt/itc/itc/certs
openssl req -new -keyout /opt/itc/itc/certs/openid-tls.key -out /opt/itc/itc/certs/
openid-tls.csr \
  -sha256 -nodes -subj "/CN=openid.example.local" \
  -addext "subjectAltName = DNS:openid.example.local"
# на сервере ЦС:
/opt/itc/minica/mclient ca -csr /tmp/openid-tls.csr -out /tmp/openid-tls.crt -template
tls
```

3. Верните сертификат на сервер OpenID в */opt/itc/itc/certs/openid-tls.crt*.
4. Разложите файлы сервиса по директориям:
   - Исполняемый файл *openid* → */opt/itc/openid/openid*.
   - systemd-юнит → */home/itc-svc/.config/systemd/user/openid.service*.
   - Файл конфигурации → */opt/itc/openid/config.yaml*.
5. В config.yaml поместите слдеующее содержимое, заменив плейсхолдеры своими данными. Блок ldapGroupRolesMap поставляется готовым — соответствует матрице ролей.

```
issuer: https://openid.example.local          # канонический URL сервиса (claim "iss")
storage:                                       # состояние сервиса — в своей БД
 PostgreSQL
  type: postgres
  config:
    host: "db.example.local"
    port: 5432
    database: "openid"
    user: "cwi"
    password: "<ПАРОЛЬ>"
web:
  https: "openid.example.local:443"            # 443 — привилегированный порт (см. setcap
ниже)
  tlsCert: "/opt/itc/itc/certs/openid-tls.crt"
  tlsKey: "/opt/itc/itc/certs/openid-tls.key"
adminServer:
  http: "openid.example.local:5559"            # служебный порт администрирования
```

```
connectors:
- type: ldap
  id: ldap
  name: LDAP
  config:
    host: "dc.example.local:389"              # сервер каталога (Active Directory)
    bindDN: "CN=svc-openid,OU=Users,DC=example,DC=local"
    bindPW: "<ПАРОЛЬ>"
    userSearch:
      baseDN: "ou=Users,ou=CWI,dc=example,dc=local"
      username: "sAMAccountName"
      idAttr: "sAMAccountName"
    groupSearch:
      baseDN: "ou=Groups,ou=CWI,dc=example,dc=local"
      nameAttr: "cn"
    ldapGroupRolesMap:                        # сопоставление групп каталога ролям
 Clearway CA
      - ldapGroup: ITC_MAM_Administrators     # администратор ЦС — полный набор ролей
        roles: [rleMiniCAMaster, rleMiniCAOperator, rleMiniCAAuditor,
                rleMiniCACertList-r, rleMiniCAEventList-r, rleMiniCATemplList-r,
rleMiniCAReqList-r,
                rleMiniCACertReq-w, rleMiniCACertRev-w, rleMiniCACertUnRev-w,
rleMiniCAReqApprove-w,
                rleMiniCATempl-w, rleMiniCACRL-r, rleMiniCACRL-w, rleMiniCAConfig-r,
rleMiniCAConfig-w]
      - ldapGroup: ITC_MAM_Users              # оператор
        roles: [rleMiniCACertList-r, rleMiniCAEventList-r, rleMiniCATemplList-r,
rleMiniCAReqList-r,
                rleMiniCACertReq-w, rleMiniCACertRev-w, rleMiniCACertUnRev-w,
rleMiniCATempl-w,
                rleMiniCACRL-r, rleMiniCAConfig-r]
      - ldapGroup: ITC_CWCA_User              # аудитор — только чтение
        roles: [rleMiniCACertList-r, rleMiniCATemplList-r, rleMiniCAReqList-r,
                rleMiniCAEventList-r, rleMiniCACRL-r, rleMiniCAConfig-r]
expiry:
  signingKeysAlgorithm: "RS256"               # алгоритм подписи токенов (по умолчанию)
  refreshTokens:
    sessionLimit: 1                           # ограничение числа одновременных сессий
staticClients:                                # клиенты, которым OpenID выдает токены
  - name: clearway
    id: itc-clearway                          # этот id указывается в client_env.js
панели
    secret: <СЕКРЕТ>
    redirectURIs: [https://cp.example.local]
```

Если имена групп каталога LDAP отличаются от ITC_MAM_Administrators / ITC_MAM_Users / ITC_CWCA_User, измените значения ldapGroup, оставив перечни ролей без изменения.

6. Запустите сервис. Порту 443 (<1024) требуется разрешение на привязку — оно выдается исполняемому файлу через setcap:

```
chmod +x /opt/itc/openid/openid
sudo setcap 'cap_net_bind_service=+ep' /opt/itc/openid/openid   # право слушать 443 без
 root
systemctl --user daemon-reload
systemctl --user enable openid.service
systemctl --user start openid.service
```

7. На брандмауэре откройте порты 443 (вход клиентов), 5559 (администрирование), исходящие к 5432 (БД) и 389 (LDAP).

5.4 Развертывание Контрольной панели

Контрольная панель — веб-интерфейс управления ЦС. Технически это два приложения за Nginx: backend ITC.Api.MiniCA.ControlPanel (порт 8407, локально) и статическое SPA (одностраничное веб-приложение) ITC.MiniCA.Spa. Nginx публикует панель по HTTPS на 443, проксирует API на backend и раздает SPA. Вход пользователей — через сервис OpenID.

Предусловия:

- установлен Nginx;
- развернут сервис OpenID;
- включен linger;
- настроено окружение `systemctl --user`.

1. Создайте TLS-сертификат для Nginx. Для этого сгенерируйте CSR и подпишите на ЦС по шаблону tls:

```
mkdir -p /opt/itc/itc/certs
openssl req -new -keyout /opt/itc/itc/certs/nginx-tls.key -out /opt/itc/itc/certs/nginx-
tls.csr \
  -sha256 -nodes -subj "/CN=cp.example.local" -addext "subjectAltName =
DNS:cp.example.local"
# на сервере ЦС:
/opt/itc/minica/mclient ca -csr /tmp/nginx-tls.csr -out /tmp/nginx-tls.crt -template tls
```

2. Выпустите JWT для контрольной панели. Панель авторизуется на ЦС по JWT (роль admin). Для этого на сервере ЦС выполните команды:

```
export MCLIENT_CONF=/opt/itc/minica/conf/mclient.yml
NAME=control-panel
openssl genpkey -algorithm RSA > /tmp/$NAME.key
openssl req -new -x509 -key /tmp/$NAME.key -out /tmp/$NAME.crt -subj "/CN=$NAME"
/opt/itc/minica/mclient mkjwt -requester "mclient" -jwt-name "$NAME" -jwt-role "admin" \
  -jwt-key "/tmp/$NAME.crt" -out "/tmp/$NAME.jwt"
openssl pkcs8 -topk8 -in /tmp/$NAME.key -out /tmp/$NAME.encrypted.key -v2 aes-256-cbc
-passout pass:"<ПАРОЛЬ>"
```

3. Скопируйте *control-panel.jwt* и *control-panel.encrypted.key* на сервер панели. Временные файлы на ЦС удалите.
4. Распакуйте backend и SPA:

```
mkdir -p /opt/itc/itc.minica.controlPanel /opt/itc/itc.minica.spa /opt/itc/itc/
{config,env,nginx,logs}
tar -xzf /tmp/ITC.Api.MiniCA.ControlPanel.tar.gz -C /opt/itc/itc.minica.controlPanel/
tar -xzf /tmp/ITC.MiniCA.Spa.tar.gz              -C /opt/itc/itc.minica.spa/
```

5. Настройте backend — */opt/itc/itc/config/itc_api_miniCA_controlPanel_config.json*:

```
{
  "OpenId": { "Authority": "https://openid.example.local", "IgnoreCertificateErrors":
true },
  "Services": { "OpenId": "http://openid.example.local:5559", "PublicationCrlUrl":
"http://localhost:9407" }
}
```

6. Настройте клиент (SPA) — */opt/itc/itc/env/client_env.js* (clientId совпадает с id клиента в staticClients OpenID):

```
(function () {
  window.itcenv = { oid: {
    issuer: 'https://openid.example.local', clientId: 'itc-clearway', secret: null,
    rolesClaim: 'roles', useSilentRefresh: false } }; // useSilentRefresh тихое
обновление, для OpenID значение false, для Keycloak при проблемах обновления access
токена true.
})();
```

7. Настройте Nginx: общий server-блок */opt/itc/itc/nginx/default* (TLS 443, подключает все **.conf*) + блок панели */opt/itc/itc/nginx/controlPanel.conf* (location / → раздача SPA; location */api/minica* → proxy_pass http://localhost:8407/api). Подключите ссылкой и перезагрузите:

```
# Debian/Astra, РЕД ОС:  sudo ln -sf /opt/itc/itc/nginx/default /etc/nginx/conf.d/
itc.conf
# ALT Linux:             sudo ln -sf /opt/itc/itc/nginx/default /etc/nginx/sites-
enabled.d/itc.conf
sudo systemctl reload nginx
```

8. Запустите сервис:

```
systemctl --user daemon-reload
systemctl --user enable controlPanel.service
systemctl --user start controlPanel.service
```

9. Подключите ЦС к панели. Дя этого разместите на сервере панели три файла ЦС (в примере sub-ca): *sub-ca.crt* (TLS ЦС), *sub-ca.jwt*, *sub-ca.encrypted.key,* в директорию */opt/itc/itc/certs/*. Расшифруйте ключ и внесите запись о ЦС (требуется jq):

```
openssl pkcs8 -in /opt/itc/itc/certs/sub-ca.encrypted.key -out /opt/itc/itc/certs/sub-ca.key -passin pass:"<ПАРОЛЬ>"
rm /opt/itc/itc/certs/sub-ca.encrypted.key
# добавить блок MiniCa.sub-ca (Url на порт 9443, пути к учетным данным) в конфиг backend через jq, затем:
systemctl --user restart controlPanel.service
```

Повторите этот шаг для каждого ЦС, которым управляет панель.

После настройки панель доступна по адресу: *https://cp.example.local*.

5.5 Развертывание сервиса публикации

Сервис публикации (ITC.Api.MiniCA.Publication) — фоновый .NET-сервис (пользовательский systemd, порт 9407), публикующий сертификаты и CRL на внешние точки распространения (CDP/AIA) и в каталог LDAP по расписанию. Публикуется через Nginx (TLS 443, путь */api/publication*); входящие запросы авторизуются через OpenID/JWT.

Предусловия:

- создана технологическая учетная запись itc-svc;
- создана директория */opt/itc*;
- включен linger;
- настроено окружение `systemctl --user`;
- установлен Nginx;
- создан TLS-сертификат (создание CSR → подпись на ЦС по шаблону `tls` → размещение *nginx-tls.crt/nginx-tls.key* в */opt/itc/itc/certs/)*.

По настройке предусловий см. раздел Общая подготовка сервера.

1. Получите JWT для доступа к ЦС (роль crluser для получения CRL либо exporter для выгрузки сертификатов). Для этого на сервере ЦС выполните команды:

```
export MCLIENT_CONF=/opt/itc/minica/conf/mclient.yml
openssl genpkey -algorithm RSA > /tmp/publication-service.key
openssl req -new -x509 -key /tmp/publication-service.key -out /tmp/publication-service.crt -subj "/CN=publication-service"
/opt/itc/minica/mclient mkjwt -requester "mclient" -jwt-name "publication-service" -jwt-role "crluser" \
  -jwt-key "/tmp/publication-service.crt" -out "/tmp/publication-service.jwt"
openssl pkcs8 -topk8 -in /tmp/publication-service.key -out /tmp/publication-service.encrypted.key \
  -v2 aes-256-cbc -passout pass:"<ПАРОЛЬ>"
```

2. В конфигурационном файле *itc_svc_env.conf* укажите переменную окружения: `ITCRoot="/opt/itc"`.
3. Создайте необходимые директории и распакуйте дистрибутив сервиса:

```
mkdir -p /opt/itc/itc.minica.publication /opt/itc/itc/{config,nginx,logs}
tar -xzf /tmp/ITC.Api.MiniCA.Publication.tar.gz -C /opt/itc/itc.minica.publication/
```

4. Запустите юнит сервиса:

```
systemctl --user daemon-reload && systemctl --user enable --now publication.service
```

5. (Опционально) Отредактируйте файл конфигурации сервиса */opt/itc/itc/config/itc_api_miniCA_publication_config.json*:

```
{
  "OpenId": { "authority": "https://<fqdn-OpenID>", "ignoreCertificateErrors": true },
  "PublishingCrlTasks": [],
  "PublishingCertificationTasks": []
}
```

Этот файл содержит адрес OpenID и списки заданий на публикацию списков отзыва (точка CDP) и сертификатов ЦС (точка AIA). По умолчанию списки пусты. Они наполняются через контрольную панель или путем редактирования файла.

Чтобы добавить задание непосредственно в файл:

a. Укажите источник (ЦС), приемник (внешняя точка CDP/AIA либо каталог LDAP — Active Directory или Samba AD) и расписание.
   Пример задания `PublishingCrlTasks` (публикация CRL для точки CDP):

```
"PublishingCrlTasks": [
  {
    "caName": "MiniCA-Root",
    "sourceType": "LocalCA",
    "sourcePath": "/opt/itc/itc.minica.publication/crl/root.crl",
    "destinationType": "HTTP",
    "destinationPath": "/opt/itc/itc/nginx/root.crl",
    "schedule": "0 */6 * * *"
  },
  {
    "caName": "MiniCA-Issuing",
    "sourceType": "LocalCA",
    "sourcePath": "/opt/itc/itc.minica.publication/crl/issuing.crl",
    "destinationType": "LDAP",
    "destinationPath": "ldap://dc.company.local/CN=MiniCA-Issuing,CN=CDP,CN=Public
Key Services,CN=Services,CN=Configuration,DC=company,DC=local",
    "schedule": "0 */12 * * *"
```

```
  }
]
```

Пример задания `PublishingCertificationTasks` (публикация сертификата ЦС для точки AIA):

```
"PublishingCertificationTasks": [
  {
    "caName": "MiniCA-Root",
    "sourceType": "LocalCA",
    "sourcePath": "/opt/itc/itc.minica.publication/certs/root.crt",
    "destinationType": "HTTP",
    "destinationPath": "/opt/itc/itc/nginx/root.crt",
    "schedule": "0 3 * * *"
  }
]
```

b. Настройте Nginx для HTTP-публикации, создав симлинк `default`:

```
# Для большинства дистрибутивов:
ln -s /opt/itc/itc/nginx/default /etc/nginx/conf.d/itc.conf

# Для ALT Linux:
ln -s /opt/itc/itc/nginx/default /etc/nginx/sites-enabled.d/itc.conf

# Перечитать конфигурацию Nginx:
sudo systemctl reload nginx
```

c. Перезапустите сервис.

```
systemctl --user restart publication.service
```

5.6 Развертывание OCSP-респондера (mOCSP)

Респондер mOCSP (исполняемый файл *mocsp*) отвечает на OCSP-запросы в реальном времени. В зависимости от параметра `status-source` сервис определяет статус сертификата по данным БД ЦС, по актуальному файлу CRL либо сначала обращается к БД и при ошибке запроса использует CRL. Слушает порт 8888, путь — */ocsp* (без Nginx).

1. Разложите файлы сервиса по директориям:
   a. *mocsp* → */opt/itc/ocsp/mocsp*;
   b. *mocsp.env* → */opt/itc/ocsp/conf/mocsp.env*;
   c. *mocsp.yml* → */opt/itc/ocsp/conf/mocsp.yml*;

d. юнит mocsp.service → *~/.config/systemd/user/*;
e. сертификат ЦС → */opt/itc/ocsp/conf/ca.crt*.

2. Сделайте файл */opt/itc/ocsp/mocsp* исполняемым:

```
chmod +x /opt/itc/ocsp/mocsp
```

3. Создайте сертификат, которым OCSP-респондер будет подписывать ответы о статусе сертификатов:
   a. Сгенерируйте закрытый ключ:

```
openssl genrsa -out /opt/itc/ocsp/conf/ocsp.key 4096
```

   b. Создайте запрос на сертификат CSR:

```
openssl req -new \
  -key /opt/itc/ocsp/conf/ocsp.key \
  -out /opt/itc/ocsp/conf/ocsp.csr \
  -sha256 -nodes \
  -subj "/CN=<fqdn-ocsp>"
```

   c. Выпустите сертификат на ЦС по шаблону `ocsp` (не `tls`). Для этого на сервере ЦС выполните команду:

```
/opt/itc/minica/mclient ca \
  -csr /tmp/ocsp.csr \
  -out /tmp/ocsp.crt \
  -template ocsp
```

4. Укажите параметры в файле конфигурации *mocsp.yml* — источник сведений о статусе сертификата, пути к файлам ключа и сертификатов, а также параметры БД и/или CRL:

```
listen-port: 8888
url-path: /ocsp
key-file: /opt/itc/ocsp/conf/ocsp.key      # ключ подписи OCSP
cert-file: /opt/itc/ocsp/conf/ocsp.crt     # сертификат подписи OCSP
ca-file: /opt/itc/ocsp/conf/ca.crt         # сертификат ЦС
status-source: db-only                     # Для режимов crl-only и db-first необходимо
дополнительно настроить секцию crl, как описано в разделе "OCSP-респондер mOCSP".
database:
  type: postgres
  url: postgres://<пользователь>:<пароль>@<хост-БД>:5432/<база-ЦС>
  allow-without-tls: true
```

5. Запустите и проверьте сервис:

```
systemctl --user daemon-reload && systemctl --user enable --now mocsp.service
openssl ocsp -issuer /opt/itc/ocsp/conf/ca.crt -cert <проверяемый>.crt \
  -url http://<fqdn-ocsp>:8888/ocsp -resp_text -no_nonce
```

5.7 Развертывание Архиватора

Архиватор (сервис ITC.Api.MiniCA.Archiver) — .NET-сервис (порт 9507), по расписанию @daily переносящий истекшие сертификаты из рабочей БД ЦС в отдельную архивную БД.

1. Создайте архивную БД (имя БД по умолчанию — cwica_archive, схема — та же, что у рабочей БД):

```
sudo -u postgres psql -c "CREATE DATABASE cwica_archive OWNER cwica;"
sudo -u postgres psql -d cwica_archive -f /tmp/minica.up.sql
```

2. Распакуйте дистрибутив сервиса *ITC.Api.MiniCA.Archiver.tar.gz* в директорию */opt/itc/itc.minica.archiver/*.
3. В файле конфигурации */opt/itc/itc/config/itc_api_miniCA_archiver_config.json* укажите свои данные вместо плейсхолдеров:

```
  "Tasks": [{
    "Key": "ClearwayCA", "Cron": "@daily",
    "Source": "Host=<хост-рабочей-
БД>;Port=5432;Database=cwica;Username=cwica;Password=<пароль>",
    "Target": "Host=<хост-архивной-
БД>;Port=5432;Database=cwica_archive;Username=cwica;Password=<пароль>"
  }]
}
```

4. Запустите юнит сервиса:

```
systemctl --user daemon-reload && systemctl --user enable --now archiver.service
```

5.8 Развертывание адаптера SCEP

Адаптер SCEP (сервис itc.acm.scepAdapter.clearwayCa) — .NET-сервис (порт 8440), обеспечивающий выпуск и обновление сертификатов по протоколу SCEP для устройств. Публикуется через Nginx (TLS 443, путь — */api/scep*); внешний адрес — *https://<fqdn>/api/scep*. Аутентификация запросов — через OpenID (не Kerberos). Локальное состояние — в SQLite (*/opt/itc/itc/sqlite/scep.db*, не PostgreSQL). Приложение self-contained — установка dotnet не требуется.

1. Создайте TLS- и SCEP-сертификаты:
   a. Создайте для сертификатов директорию */opt/itc/itc/certs/*.
   b. Сгенерируйте две ключевые пары: для TLS и для SCEP.

c. Подпишите оба CSR на ЦС по шаблону `tls`. Будут выпущены сертификаты *nginx-tls.crt* и *scep.crt*.

d. Сформируйте цепочку сертификатов в файле *scep-chain.pem* — сам сертификат SCEP плюс сертификаты вышестоящих ЦС.
Пример:

```
# На сервере ЦС:
cat /tmp/scep.crt /opt/itc/ocsp/conf/ca.crt > /opt/itc/itc/certs/scep-chain.pem
```

e. Разместите *nginx-tls.crt*, *scep.crt* и цепочку *scep-chain.pem* в директории */opt/itc/itc/certs/*.

2. Получите JWT для доступа к ЦС (роль центра регистрации causer): на сервере ЦС — так же, как для сервиса публикации, но с параметром `-jwt-role "causer"`.

3. В конфигурационном файле *itc_svc_env.conf* укажите переменную окружения: `ITCRoot="/opt/itc"`, если это не было сделано ранее.

4. Создайте необходимые директории и распакуйте дистрибутив сервиса:

```
mkdir -p /opt/itc/itc/sqlite /opt/itc/itc.acm.scepAdapter.clearwayCa /opt/itc/itc/
{config,nginx,logs}
tar -xzf /tmp/itc.acm.scepAdapter.clearwayCa.tar.gz -C /opt/itc/
itc.acm.scepAdapter.clearwayCa/
```

5. В файле конфигурации сервиса */opt/itc/itc/config/itc_acm_scepAdapter_clearwayCa_config.json* укажите параметры провайдера sqlite и аутентификация OpenID:

```
{
  "Database": { "Provider": "sqlite", "ConnectionName": "DefaultConnection" },
  "ConnectionStrings": { "DefaultConnection": "Data Source=/opt/itc/itc/sqlite/scep.db" }
,
  "CaChainPath": "/opt/itc/itc/certs/scep-chain.pem",
  "ScepCertPath": "/opt/itc/itc/certs/scep.crt",
  "ScepKeyPath": "/opt/itc/itc/certs/scep.key",
  "SupportsRenewal": true,
  "AuthenticationConfig": {
    "AuthenticationType": "openId",
    "OpenId": { "Authority": "https://<fqdn-OpenID>", "IgnoreCertificateErrors": true }
  },

  "ClearwayCaClientConfig": { "CaTemplate": "tls", "CaCluster": "", "MiniCa": {} }
}
```

Юнит сервиса scepAdapter.service принимает HTTP-запросы на порту 8440 и использует окружение из *itc_svc_env.conf*, без переменных KRB5_*:

```
ExecStart=/opt/itc/itc.acm.scepAdapter.clearwayCa/ITC.Api.Acm.Scep.Adapter.ClearwayCa --
urls="http://0.0.0.0:8440"
```

6. Запустите юнит SCEP-адаптера:

```
systemctl --user daemon-reload && systemctl --user enable --now scepAdapter.service
```

7. Выполните привязку SCEP-адаптера к ЦС:
   a. Разместите файлы *sub-ca.crt*, *sub-ca.jwt*, *sub-ca.encrypted.key* в директории */opt/itc/itc/certs/*.
   b. Расшифруйте ключ:

```
openssl rsa \
  -in /opt/itc/itc/certs/sub-ca.encrypted.key \
  -out /opt/itc/itc/certs/sub-ca.key
```

   c. Внесите URL ЦС на порт 9443 и пути к учетным данным в секцию "ClearwayCaClientConfig.MiniCa" в файле конфигурации */opt/itc/itc/config/itc_acm_scepAdapter_clearwayCa_config.json*:
   Пример:

```
{
  "ClearwayCaClientConfig": {
    "CaTemplate": "tls",
    "CaCluster": "",
    "MiniCa": {
      "Url": "https://<fqdn-ca>:9443",
      "CaCertPath": "/opt/itc/itc/certs/sub-ca.crt",
      "JwtPath": "/opt/itc/itc/certs/sub-ca.jwt",
      "KeyPath": "/opt/itc/itc/certs/sub-ca.key"
    }
  }
}
```

   d. Перезапустите сервис:

```
systemctl --user restart scepAdapter.service
```

8. Настройте Nginx для HTTP-публикации, создав симлинк `default`:

```
# Для большинства дистрибутивов:
ln -s /opt/itc/itc/nginx/default /etc/nginx/conf.d/itc.conf

# Для ALT Linux:
```

```
ln -s /opt/itc/itc/nginx/default /etc/nginx/sites-enabled.d/itc.conf
```

9. В файле конфигурации *scep.conf* настройте проксирование `/api/scep` → *http://localhost:8440/api*:

```
location /api/scep {
    proxy_pass http://localhost:8440/api;
}
```

10. Перезапустите веб-сервер Nginx:

```
sudo systemctl reload nginx
```

После этого адаптер доступен устройствам по адресу *https://<fqdn>/api/scep*.

6 Настройка ядра ЦС (minica.yml)

Ядро Удостоверяющего центра — сетевой сервис MiniCA (minica.service). Всё его поведение задается единственным конфигурационным файлом:

```
/opt/itc/minica/conf/minica.yml
```

Путь к файлу содержится в переменной MINICA_CONF (файл */opt/itc/minica/minica.env*). Файл представляет собой шаблон: при установке в него подставляются вычисляемые значения (имя ЦС из субъекта сертификата, CDP_SERVER_NAME, параметры подключения к БД), поэтому большинство ключей уже заполнено. Ниже разобраны основные секции. Изменения вступают в силу после перезапуска службы (см. подраздел Управление службой).

> ⚠ Файл содержит и задает пути к закрытым ключам ЦС. Он должен принадлежать технологической учетной записи (itc-svc) с правами 0600. Не редактируйте файл под root и не оставляйте резервных копий в общедоступных каталогах. Меры по защите файлов ЦС описаны в разделе Обеспечение безопасной среды.

6.1 Сеть и HTTP/HTTPS-сервер

Сервис принимает запросы Центра регистрации и служебных компонентов по HTTP или HTTPS. Штатный рабочий порт — 9443 (HTTPS).

Параметр	Назначение	Значение по умолчанию
listen-ip	IP-адрес, на котором сервис принимает соединения; пустая строка — все интерфейсы	"" (все интерфейсы)
http-port	Порт HTTP (используется, когда TLS выключен)	9080
https-port	Порт HTTPS (используется, когда TLS включен)	9443
timeout-ms	Таймаут обработки запроса, мс	60000

> ℹ Отдельного параметра server-url нет: адрес сервиса формируется из listen-ip, порта и признака TLS. Клиентские компоненты обращаются к ЦС по адресу вида *https://<host>:9443*.

6.2 TLS и взаимная аутентификация (mTLS)

По умолчанию TLS выключен — это состояние для первичной установки. В рабочей конфигурации TLS включается компонентом настройки издающего ЦС. Для доверенной эксплуатации рекомендуется дополнительно включить взаимную аутентификацию (mTLS).

Параметр	Назначение	Значение по умолчанию
tls.enable	Включение TLS (переводит сервис на https-port)	false
tls.verify	Включение mTLS — проверки клиентского сертификата	false
tls.key	Файл закрытого ключа сервера	*conf/tls.key*
tls.cert	Файл сертификата сервера	*conf/tls.crt*
tls.ca-cert	Сертификат УЦ для проверки клиентов (mTLS)	*conf/ca.crt*
tls.ca-certs	Дополнительные доверенные сертификаты (mTLS)	[]
tls.insecure	Не проверять сертификат УЦ (только для отладки)	false

```
tls:
  enable: true    # рабочий режим — HTTPS на порту 9443
  verify: true    # взаимная аутентификация клиентов (mTLS)
  key:  /opt/itc/minica/conf/tls.key
  cert: /opt/itc/minica/conf/tls.crt
  ca-cert: /opt/itc/minica/conf/ca.crt
```

> ⚠ Не оставляйте `tls.insecure: true` в промышленной эксплуатации — при этом сертификаты не проверяются. Рекомендации по защищенной конфигурации канала приведены в разделе Обеспечение безопасной среды.

6.3 Контроль рассогласования времени (max-dt-sec)

Подпись каждого запроса и JWT-токена проверяется с учетом метки времени. Параметр max-dt-sec задает максимально допустимую разницу между временем узла-отправителя и временем ЦС.

Параметр	Назначение	Значение по умолчанию
max-dt-sec	Допустимое рассогласование времени при проверке подписи, с	600 (окно ±10 минут)

При расхождении часов больше заданного окна запрос отклоняется. Это защищает от повторного воспроизведения ранее перехваченных запросов.

> ⚠ Синхронизируйте системное время (NTP) на **всех** узлах ЦС и на сервере СУБД. Без синхронизации времени легитимные запросы будут отклоняться при выходе за окно ±10 минут.

6.4 Ключ и сертификат ЦС

Собственно ключевая пара и сертификат Удостоверяющего центра, а также рабочие каталоги OpenSSL.

Параметр	Назначение	Значение по умолчанию
ca.name	Имя секции CA в *openssl.cnf* (роль ЦС: издающий/корневой)	CA_default
ca.key	Файл закрытого ключа ЦС	*ca/ca.key*
ca.cert	Файл сертификата ЦС	*ca/ca.crt*
ca.old-keys	Каталог для ранее использовавшихся ключей ЦС (ротация)	*ca/old-keys*
ca.start-sn	Стартовый серийный номер выпускаемых сертификатов	1000000000000000000000
ca.start-crl	Стартовый серийный номер CRL	1000000000000000000000
ca.rand-sn	Использовать случайные серийные номера вместо порядковых	false
chain-file	Файл цепочки сертификатов (PEM)	*ca/chain.pem*

> ⚠ Перед выпуском ГОСТ-сертификатов установите пакет на сервере, где установлен центр сертификации (ЦС):

Astra Linux

```
sudo apt install libgost-astra
```

Хранение ключа ЦС в HSM/токене (PKCS#11)

Ключ ЦС может храниться на аппаратном носителе (HSM, Rutoken). Отдельного параметра для этого нет. Хранение на аппаратном носителе включается следующим образом:

1. В параметре `ca.key` вместо пути к файлу укажите PKCS#11-URI:

   ```
   ca:
     key: "pkcs11:token=ClearwayCA;object=clearway_ca_key;pin-value=<PIN>"
   ```

   Здесь значения `token`, `object`, `pin-value` даны как пример. Измените значения под свой носитель.

2. Активируйте соответствующий движок в *openssl.cnf*. Для этого в *openssl.cnf* раскомментируйте строки подключения движка:

   ```
   openssl_conf = openssl_init
   [openssl_init]
   engines = engine_section
   [engine_section]
   pkcs11 = pkcs11_section
   ```

   Секция [pkcs11_section] в поставляемом *openssl.cnf* уже подготовлена (движок pkcs11, путь к модулю библиотеки токена).

6.5 Шаблоны, поля CDP/AIA и сроки действия

Точки распространения CRL (CDP), доступа к сертификату издателя (AIA) и адрес свежайшего CRL вычисляются из переменной CDP_SERVER_NAME и имени ЦС при установке. Если CDP_SERVER_NAME не задана, соответствующие списки остаются пустыми.

Параметр	Назначение	Значение по умолчанию
ca.exts	Секция расширений *openssl.cnf*, используемая при выпуске	usr_cert
ca.exts-crl	Секция расширений *openssl.cnf*, используемая при выпуске CRL	crl_ext
ca.cdp	Расширение CDP (точка распространения CRL)	http://<CDP_SERVER_NAME>/cdp/<CN>.crl

Параметр	Назначение	Значение по умолчанию
ca.aia	Расширение AIA (доступ к сертификату издателя)	caIssuers;URI:http://<CDP_SERVER_NAME>/aia/<CN>.crt
ca.freshest-crl	Точка распространения DeltaCRL (суффикс +)	http://<CDP_SERVER_NAME>/cdp/<CN>+.crl
ca.days.cert	Срок действия выпускаемых сертификатов, дней	1460 (≈4 года)
ca.days.crl	Срок действия CRL, дней	365
ca.days.delta-crl	Срок действия DeltaCRL, дней (0 — как у CRL)	0
ca.overlap	Смещение начала действия сертификата в прошлое, мин	10
ca.overlap-crl	Разница между nextUpdate и nextPublish для CRL, ч	0
ca.no-limit-ca	Не ограничивать срок целевых сертификатов сроком сертификата ЦС	false

При выпуске отдельного сертификата эти значения могут переопределяться шаблоном (шаблоны хранятся в БД, кэш — database.templ-cache-minutes). Значения в *minica.yml* применяются, когда шаблон не задает собственных.

6.6 Шифрование данных ЦС (ca.encryption)

Конфиденциальные данные ЦС шифруются на **отдельной** ключевой паре, не совпадающей с ключом подписи ЦС.

Параметр	Назначение	Значение по умолчанию
ca.encryption.key	Закрытый ключ пары шифрования данных	*ca/encr.key*
ca.encryption.cert	Сертификат пары шифрования данных	*ca/encr.crt*

> ⚠ Не указывайте здесь ключ подписи ЦС (*ca/ca.key*). Для шифрования используется самостоятельная пара *encr.key/encr.crt*.

6.7 Восстановление ключей субъектов (KRA)

Для возможности восстановления закрытых ключей субъектов их резервные копии шифруются на сертификате агента восстановления ключей (Key Recovery Agent).

Параметр	Назначение	Значение по умолчанию
ca.kra.cert	Сертификат агента восстановления ключей (KRA)	*ca/kra.crt*

6.8 База данных

Состояние ЦС (реестры сертификатов, CRL, шаблоны, JWT) хранится в PostgreSQL (поддерживаются версии 11–18).

Параметр	Назначение	Значение по умолчанию
database.type	Тип СУБД	postgres
database.url	URL подключения к СУБД	postgres://<user>:<pass>@<host>:<port>/<db>
database.templ-cache-minutes	Время кэширования шаблонов, мин	3
database.allow-without-tls	Разрешить подключение к БД без TLS	true (для безопасности — false)

Параметр `allow-without-tls` управляет **каналом к СУБД**, а не подключениями клиентов к самому ЦС (за это отвечает секция tls:). По умолчанию установщик задает значение true. Для доверенной эксплуатации установите false — тогда при отсутствии TLS на стороне БД сервис завершит работу, не открыв незащищенного соединения. Для реальной проверки сертификата сервера БД в строке подключения дополнительно укажите `sslmode=verify-full`:

```
database:
  type: postgres
  url: postgres://minica:<pass>@db.example:5432/minica?sslmode=verify-full
  allow-without-tls: false   # запретить подключение к БД без TLS
```

6.9 Сервисные роли JWT (ролевая модель)

Доступ служебных компонентов к функциям API ЦС ограничивается ролевой моделью на JWT. Роль — именованный набор атомарных прав (всего в API 24 функции). В поставке определено 7 ролей.

Роль	Назначение
admin	Полный доступ ко всем функциям API
causer	Центр регистрации: выпуск, отзыв и проверка сертификатов
crluser	Клиент CDP: обновление и получение CRL/DeltaCRL
archuser	Архивариус: удаление устаревших данных из БД
exporter	Экспортер: чтение любых данных из БД
templadmin	Администратор шаблонов сертификатов
jwtadmin	Администратор JWT-токенов

Ключи и режим проверки токенов задаются в секции jwt:

Параметр	Назначение	Значение по умолчанию
jwt.no-check	Отключить проверку JWT (тогда подпись запросов проверяется только мастер-ключом)	false
jwt.key	Файл ключа подписи JWT	*conf/jwt.key*
jwt.keys	Каталог ключей проверки подписи JWT (имена файлов — SKI, HEX 20 байт)	*conf/jwt-keys*
jwt.cache-minutes	Время кэширования ключей проверки, мин	5
master-key	Мастер-ключ проверки подписи запросов (PEM)	*conf/mclient.pem*
sign-off	Отключить проверку подписи запросов	false

⚠ Не устанавливайте `jwt.no-check: true` и `sign-off: true` в промышленной эксплуатации — это отключает контроль полномочий и подлинности запросов.

6.10 Управление службой

MiniCA работает как пользовательский systemd-юнит технологической учетной записи (itc-svc), а не как системный сервис. Управление ведется с флагом `--user` от имени этой учетной записи, а автозапуск без активной сессии обеспечивается включенным linger.

```
# от имени технологической учетной записи (itc-svc)
systemctl --user daemon-reload
systemctl --user enable  minica.service      # автозапуск
systemctl --user restart minica.service      # применить изменения minica.yml
systemctl --user status  minica.service      # состояние службы
```

Чтобы служба запускалась при загрузке узла и продолжала работать после выхода из SSH-сессии, для учетной записи должен быть включен linger:

```
sudo loginctl enable-linger itc-svc
```

Путь к конфигурации задается переменной MINICA_CONF в *opt/itc/minica/minica.env*; значение переменной по умолчанию — */opt/itc/minica/conf/minica.yml*. После правки любого параметра конфигурации перезапустите службу командой:

```
systemctl --user restart minica.service
```

Параметры журналирования (секции log и secLog, контроль целостности записей) вынесены в раздел Журналирование и мониторинг, а меры по защите файлов, ключей и каналов ЦС — в раздел Обеспечение безопасной среды. В настоящем разделе они не дублируются.

7 Настройка вспомогательных сервисов

Помимо ядра центра сертификации (сервис minica) в состав Clearway CA входит набор вспомогательных сервисов: сервис аутентификации OpenID, Контрольная панель, Сервис публикации, OCSP-респондер mOCSP, Архиватор, SCEP-адаптер и точка распространения CRL (CDP). Все сервисы взаимодействуют по HTTP/HTTPS и могут размещаться как на одном, так и на разных узлах.

Общие принципы, действующие для всех вспомогательных сервисов:

- Сервисы работают как пользовательские юниты systemd от имени технологической учетной записи itc-svc (по умолчанию), поэтому управление ведется через `systemctl --user`, а для учетной записи включен режим `enable-linger`.
- Веб-сервисы (Контрольная панель, Сервис публикации, SCEP, CDP) публикуются наружу через Nginx как обратный прокси: TLS на порту 443 (для CDP — HTTP на порту 80), проксирование на локальный порт приложения.
- Каталог установки по умолчанию — */opt/itc* (для ядра ЦС — */opt/itc/minica*, для mOCSP — */opt/itc/ocsp*, для CDP — */opt/itc/web*).
- Аутентификация пользователей и сервис-клиентов выполняется через сервис OpenID (OIDC), авторизация сервисов при обращении к ЦС — по JWT.

Управление любым сервисом выполняется от имени технологической учетной записи:

```
su - itc-svc
systemctl --user enable --now <юнит>     # включить автозапуск и запустить
systemctl --user restart <юнит>          # перезапустить (после правки конфига)
systemctl --user status  <юнит>          # состояние
```

Сводка портов и юнитов вспомогательных сервисов:

Сервис	Юнит systemd	Порт приложения	Публикация (Nginx)	Конфигурационный файл (по умолчанию)
Сервис аутентификации OpenID	openid.service	443 (HTTPS), 5559 (admin HTTP)	— (слушает 443 напрямую)	*/opt/itc/openid/config.yaml*
Контрольная панель	controlPanel.service	8407	HTTPS 443 → /api/minica	*/opt/itc/itc/config/itc_api_miniCA_controlPanel_config.json*
Сервис публикации	publication.service	9407	HTTPS 443 → /api/publication	*/opt/itc/itc/config/itc_api_miniCA_publication_config.json*
OCSP-респондер mOCSP	mocsp.service	8888 (/ocsp)	—	*/opt/itc/ocsp/conf/mocsp.yml*
Архиватор	archiver.service	9507	—	*/opt/itc/itc/config/itc_api_miniCA_archiver_config.json*

Сервис	Юнит systemd	Порт приложения	Публикация (Nginx)	Конфигурационный файл (по умолчанию)
SCEP-адаптер	scepAdapter.service	8440	HTTPS 443 → /api/scep	*/opt/itc/itc/config/itc_acm_scepAdapter_clearwayCa_config.json*
CDP (раздача CRL/AIA)	— (Nginx)	—	HTTP 80 → /cdp, /aia	*/opt/itc/web/nginx.conf*

> ⚠ Все пути указаны для значений по умолчанию (PATH=*/opt/itc*, учетная запись itc-svc). Если при установке заданы иные значения, скорректируйте пути и имя учетной записи во всех приведенных командах.

7.1 Сервис аутентификации OpenID

Назначение: OIDC-провайдер (форк Dex), аутентифицирует пользователей и сервис-клиентов через LDAP/Active Directory, сопоставляет группы AD ролям Clearway CA и выдает клиентам access- и refresh-токены (JWT). Состояние (запросы авторизации, сессии, ключи подписи) хранится в PostgreSQL.

Конфигурационный файл: */opt/itc/openid/config.yaml*. Юнит — openid.service (ExecStart: openid serve config.yaml). Порт 443 требует прав на привязку к привилегированному порту; они выдаются исполняемому файлу через `setcap cap_net_bind_service.`

Ключевые параметры:

Параметр	Назначение	Значение по умолчанию
issuer	Базовый URL издателя токенов (OIDC issuer)	*https://<fqdn>*
storage.type / storage.config	Тип и параметры хранилища состояния	postgres (host, port, database, user, password; ssl.mode: disable)
web.https	Адрес и порт приема HTTPS-запросов OIDC	<host>:443
web.tlsCert / web.tlsKey	Сертификат и ключ TLS сервиса	*/opt/itc/itc/certs/openid-tls.crt* / *.key*
adminServer.http	Служебный (административный) HTTP-интерфейс	<host>:5559
connectors[ldap]	Коннектор к LDAP/AD: host, bindDN, bindPW, insecureNoSSL, периодический импорт (import.timer: 1h, объекты person/groups)	—

Параметр	Назначение	Значение по умолчанию
userSearch / groupSearch	Правила поиска пользователей и групп (baseDN, filter, атрибуты sAMAccountName, userPrincipalName, displayName, cn)	—
ldapGroupRolesMap	Сопоставление групп AD ролям Clearway CA (ITC_MAM_Administrators, ITC_MAM_Users, ITC_CWCA_User)	задано в конфиге
expiry.idTokens	Время жизни access-токена	2m
expiry.authRequests	Время на прохождение авторизации	10m
expiry.refreshTokens	Параметры refresh-токенов: validIfNotUsedFor, sessionLimit, sessionLimitPolicy (reject/ terminate_oldest), absoluteLifetime	—
staticClients[]	Зарегистрированные OIDC-клиенты: id, secret, redirectURIs	клиент itc-clearway

Сопоставление групп AD ролям (ldapGroupRolesMap) здесь задает только соответствие "группа → набор ролей". Подробный состав ролей и прав, а также порядок назначения ролей пользователям приведены в разделе Управление доступом и ролевая модель.

Управление службой:

```
su - itc-svc
systemctl --user restart openid.service
systemctl --user status  openid.service
```

7.2 Контрольная панель

Назначение: веб-интерфейс управления центрами сертификации, предоставляет функции просмотра и отзыва сертификатов, управления шаблонами, публикации CRL. Состоит из серверной части (backend ASP.NET) и статического веб-приложения (SPA), которые публикуются через Nginx. Вход выполняется через OpenID; к центру сертификации панель подключается по защищенному каналу с авторизацией по JWT.

Конфигурационные файлы:

Файл	Назначение
/opt/itc/itc/config/ itc_api_miniCA_controlPanel_config.json	Конфигурация backend: параметры OpenID, зарегистрированные ЦС, адреса зависимых сервисов
/opt/itc/itc/env/client_env.js	Параметры OpenID для браузера (SPA): issuer, clientId, secret, rolesClaim
/opt/itc/itc/nginx/controlPanel.conf	Location-блоки Nginx: раздача SPA и проксирование */api/minica → http://localhost:8407/api*

Ключевые параметры backend:

Параметр	Назначение	Значение по умолчанию
OpenId.Authority	URL сервера OpenID (AUTH_URL)	из настроек OpenID
OpenId.IgnoreCertificateErrors	Игнорировать ошибки проверки сертификата OIDC-провайдера	true
JwtRoleClaim	Имя claim с ролями в токене	null (используется roles)
Services.OpenId	Адрес служебного интерфейса OpenID	*http://<auth>:5559*
Services.PublicationCrlUrl	Адрес сервиса публикации CRL	*http://localhost:9407*
Регистрация ЦС (MiniCa.<имя-ЦС>)	Подключение к ЦС: URL (*https://<host>:9443*) и параметры авторизации (Auth.Tls, Auth.Sign, Auth.Jwt)	добавляется при настройке подключения ЦС

Подключение конкретного ЦС к панели выполняется после установки: в конфигурацию backend добавляется блок с адресом ЦС и учетными данными — TLS-сертификатом, расшифрованным ключом и JWT-токеном сервиса.

Управление службой:

```
su - itc-svc
systemctl --user restart controlPanel.service
sudo systemctl reload nginx        # после изменения Nginx-конфигурации
```

7.3 Сервис публикации

Назначение: публикует выпущенные сертификаты и списки отзыва (CRL/DeltaCRL) во внешние точки размещения — на внешние веб-серверы (CDP/AIA) и в каталоги LDAP (Active Directory, Samba) — по

заданиям, выполняемым по расписанию. Публикуется через Nginx (*/api/publication*) с авторизацией через OpenID/JWT.

Конфигурационный файл: */opt/itc/itc/config/itc_api_miniCA_publication_config.json*. Юнит — publication.service (порт 9407). Nginx-блок — */opt/itc/itc/nginx/publication.conf* (*/api/publication → http://localhost:9407/api*, таймаут 600 с).

Ключевые параметры:

Параметр	Назначение	Значение по умолчанию
OpenId.authority	URL сервера OpenID	из настроек OpenID
OpenId.ignoreCertificateErrors	Игнорировать ошибки проверки сертификата OIDC-провайдера	true
OpenId.jwtRoleClaim	Имя claim с ролями	null (roles)
PublishingCrlTasks	Список заданий публикации CRL/ DeltaCRL	[] (задается администратором)
PublishingCertificationTasks	Список заданий публикации сертификатов	[] (задается администратором)

После установки списки заданий пусты. Задания публикации (точка назначения — внешний сервер или LDAP, публикуемые объекты, расписание) администратор настраивает самостоятельно — через Контрольную панель либо правкой конфигурационного файла с последующим перезапуском сервиса.

Управление службой:

```
su - itc-svc
systemctl --user restart publication.service
```

7.4 OCSP-респондер mOCSP

Назначение: отвечает на запросы о статусе сертификатов по протоколу OCSP (RFC 6960). Принимает OCSP-запросы по HTTP (GET/POST), в реальном времени определяет статус сертификата по выбранному источнику — БД центра сертификации, актуальному файлу CRL либо БД с переходом на CRL при ошибке запроса — и возвращает подписанный ответ (*good*, *revoked* или *unknown*). Ответы подписываются отдельным сертификатом подписи OCSP.

Конфигурационный файл: */opt/itc/ocsp/conf/mocsp.yml*. Юнит — mocsp.service.

Ключевые параметры:

Параметр	Назначение	Значение по умолчанию
status-source	Источник сведений о статусе сертификата: db-only, db-first или crl-only	db-only

Параметр	Назначение	Значение по умолчанию
crl.crl-path	Точный путь к основному файлу CRL	Не задан; обязателен для db-first и crl-only
crl.delta-crl-path	Путь к необязательному Delta CRL	Не задан
crl.reload-enabled	Автоматически перезагружать CRL после изменения файла	true
crl.reload-timeout	Максимальная продолжительность одной попытки перезагрузки	30s
crl.time-skew	Допуск рассогласования времени при проверке периода действия CRL	5s
crl.fail-on-expired	Запрещать использование просроченного CRL	true
crl.source.watch.method	Способ обнаружения изменения CRL: inotify или poll	inotify
crl.source.watch.poll-interval	Интервал проверки файла при методе poll	30s
crl.source.watch.debounce	Задержка перед перезагрузкой после файлового события inotify	300ms
listen-ip	Адрес прослушивания (пусто — все интерфейсы)	""
listen-port	Порт OCSP-сервера	8888
url-path	Базовый путь URL респондера	*/ocsp*
timeout-ms	Таймаут обработки запроса, мс	60000
key-file / cert-file	Ключ и сертификат подписи OCSP-ответов	*conf/ocsp.key / conf/ocsp.crt*
ca-file	Сертификат ЦС, чьи сертификаты обслуживаются	*conf/ca.crt*
hash	Алгоритм хэширования	SHA256
add-cert	Включать сертификат подписанта в ответ	true

Параметр	Назначение	Значение по умолчанию
nmin	Интервал формирования поля NextUpdate, мин	60
database.type / database.url	Тип и строка подключения к БД ЦС (PostgreSQL) используются в режимах db-only и db-first	postgres
database.allow-without-tls	Разрешить подключение к БД без TLS	true
tls.enable / tls.verify	TLS/mTLS самого респондера	false / false

Сертификат подписи OCSP-ответов (*ocsp.crt*) выпускается центром сертификации по шаблону ocsp и устанавливается в каталог *conf/* на этапе настройки OCSP. Без него сервис не может подписывать ответы.

Выбор источника сведений о статусе сертификата:

Параметр status-source определяет источник сведений, на основании которого mOCSP формирует статусы *good*, *revoked* и *unknown*.

Значение	Назначение	Использование БД	Использование CRL
db-only	Получение статуса только из БД ЦС. Это режим по умолчанию	Всегда	Не используется
crl-only	Постоянное получение статуса из актуального CRL	Не используется	Всегда
db-first	Получение статуса из БД с переходом на CRL при технической ошибке запроса к БД	Основной источник	Резервный источник

Значение необходимо указывать строчными буквами точно в приведенном виде. Неизвестное или ошибочно написанное значение интерпретируется сервисом как db-only.

В режиме db-first переход на CRL выполняется, если запрос сертификата к БД завершился ошибкой. Если БД доступна, но сертификат в ней не найден, сервис возвращает статус *unknown* и к CRL не обращается.

Подготовка и настройка CRL:

CRL (Certificate Revocation List) — подписанный удостоверяющим центром список отозванных сертификатов.

Администратор должен обеспечить получение и регулярное обновление актуального CRL обслуживаемого ЦС в локальном файле, указанном в параметре `crl.crl-path`. Сервис mOCSP не

загружает CRL из точки распространения, а читает указанный локальный файл. Файл должен быть доступен для чтения технологической учетной записи.

mOCSP принимает CRL в одном из следующих форматов:

- DER;
- PEM с типом блока X509 CRL.

Сертификат ЦС, указанный в параметре `ca-file`, используется для проверки издателя и подписи CRL. Файл CRL должен содержать поля `ThisUpdate` и `NextUpdate`. При обработке каждого OCSP-запроса сервис проверяет время вступления CRL в действие и срок его действия с учетом допуска `time-skew`. CRL, который еще не вступил в действие, отклоняется. При `fail-on-expired: true` просроченный CRL также отклоняется; при `fail-on-expired: false` его использование допускается.

Пример для постоянного использования CRL:

```
status-source: crl-only

ca-file: /opt/itc/ocsp/conf/ca.crt

crl:
  # Рекомендуемый локальный путь; разместите здесь CRL обслуживаемого ЦС.
  crl-path: /opt/itc/ocsp/conf/ca.crl

  # Оставьте пустым, если Delta CRL не используется.
  delta-crl-path: ""

  reload-enabled: true
  reload-timeout: 30s

  time-skew: 5s
  fail-on-expired: true

  source:
    watch:
      method: inotify
      poll-interval: 30s
      debounce: 300ms
```

Для резервного использования CRL при ошибке обращения к БД применяется та же секция `crl`, но источник задается следующим образом:

```
status-source: db-first
```

В режиме db-first секция `database` должна содержать действующие параметры подключения к БД ЦС. Значение `crl-path` берется из принятой процедуры доставки CRL обслуживаемого ЦС. Если указан `delta-crl-path`, соответствующий файл становится обязательным и должен загружаться вместе с основным CRL.

Значения `reload-timeout`, `time-skew`, `poll-interval` и `debounce` записываются с единицей измерения, например, 300ms, 5s, 30s или 1m. Допуск `time-skew` должен соответствовать принятой политике синхронизации времени.

Метод `inotify` предназначен для Linux и отслеживает изменение либо атомарную замену файла по точному пути. На другой операционной системе необходимо выбрать `poll`.

Если задан `delta-crl-path`, Delta CRL проверяется и загружается вместе с основным CRL. При определении статуса запись из Delta CRL имеет приоритет.

Если серийный номер отсутствует и в Delta CRL, и в основном CRL, mOCSP возвращает статус *good*. В режиме `crl-only` сервис не обращается к БД и не проверяет, был ли сертификат с таким серийным номером действительно выпущен. Поэтому статус *good* в этом режиме означает отсутствие записи об отзыве в загруженных CRL. Запись с причиной отзыва removeFromCRL также интерпретируется как статус *good*.

Управление службой:

```
su - itc-svc
systemctl --user restart mocsp.service
```

7.5 Архиватор

Назначение: по расписанию переносит данные из рабочей БД центра сертификации (истекшие сертификаты и связанные записи) в отдельную архивную БД, разгружая рабочую базу. Архивная БД по умолчанию называется cwica_archive.

Конфигурационный файл: */opt/itc/itc/config/itc_api_miniCA_archiver_config.jso*n. Юнит — archiver.service (порт 9507).

Ключевые параметры:

Параметр	Назначение	Значение по умолчанию
Tasks[].Key	Идентификатор задания архивации	ClearwayCA
Tasks[].Cron	Расписание запуска (cron-выражение)	@daily (ежедневно)
Tasks[].Source	Строка подключения к рабочей БД ЦС (Host/Port/Database/Username/Password)	из параметров рабочей БД
Tasks[].Target	Строка подключения к архивной БД (cwica_archive)	из параметров архивной БД

Управление службой:

```
su - itc-svc
systemctl --user restart archiver.service
```

7.6 SCEP-адаптер

Назначение: реализует выпуск и обновление сертификатов для сетевых устройств по протоколу SCEP. Приложение itc.acm.scepAdapter.clearwayCa (.NET) публикуется через Nginx; внешний адрес для устройств — *https://<fqdn>/api/scep*. Аутентификация запросов выполняется через OpenID, состояние адаптера хранится в локальной БД SQLite.

Каталоги и юнит:

- Каталог приложения: */opt/itc/itc.acm.scepAdapter.clearwayCa*;
- Юнит systemd: scepAdapter.service (файл */home/itc-svc/.config/systemd/user/scepAdapter.service*), порт приложения 8440;
- Конфигурационный файл: */opt/itc/itc/config/itc_acm_scepAdapter_clearwayCa_config.json*;
- Nginx-блок: */opt/itc/itc/nginx/scep.conf* (location */api/scep* → http://localhost:8440/api).

Ключевые параметры:

Параметр	Назначение	Значение по умолчанию
Database.Provider	Провайдер БД адаптера	sqlite
ConnectionStrings.DefaultConnection	Строка подключения к БД	Data Source=/opt/itc/itc/sqlite/scep.db
CaChainPath	Цепочка сертификатов ЦС	*/opt/itc/itc/certs/scep-chain.pem*
ScepCertPath / ScepKeyPath	Сертификат и ключ подписи SCEP	*.../certs/scep.crt / scep.key*
SupportedAlgorithms	Поддерживаемые алгоритмы подписи	SHA256withRSA, SHA384withECDSA
SupportsRenewal	Поддержка обновления сертификатов	true
AuthenticationConfig.AuthenticationType	Способ аутентификации запросов	openId
AuthenticationConfig.OpenId.Authority	URL сервера OpenID (AUTH_URL)	из настроек OpenID
AuthenticationConfig.OpenId.IgnoreCertificateErrors	Игнорировать ошибки проверки сертификата OIDC-провайдера	true

Параметр	Назначение	Значение по умолчанию
ClearwayCaClientConfig.CaTemplate	Шаблон выпуска сертификатов через ЦС	tls

Пример конфигурации адаптера:

```
{
    "Database": { "Provider": "sqlite", "ConnectionName": "DefaultConnection" },
    "ConnectionStrings": {
        "DefaultConnection": "Data Source=/opt/itc/itc/sqlite/scep.db"
    },
    "CaChainPath": "/opt/itc/itc/certs/scep-chain.pem",
    "ScepCertPath": "/opt/itc/itc/certs/scep.crt",
    "ScepKeyPath":  "/opt/itc/itc/certs/scep.key",
    "AuthenticationConfig": {
        "AuthenticationType": "openId",
        "OpenId": {
            "Authority": "https://<fqdn-сервера-openid>",
            "IgnoreCertificateErrors": true
        }
    },
    "ClearwayCaClientConfig": { "CaTemplate": "tls" }
}
```

Из протокольных адаптеров в состав установщика штатно входит только SCEP. По умолчанию SCEP-адаптер использует БД SQLite (*/opt/itc/itc/sqlite/scep.db*) и аутентификацию OpenID; настройки Kerberos или PostgreSQL для этого сервиса не применяются.

Управление службой:

```
su - itc-svc
systemctl --user restart scepAdapter.service
sudo systemctl reload nginx        # после изменения Nginx-конфигурации
```

7.7 Точка распространения CRL и сертификатов (CDP)

Назначение: веб-сервер (Nginx), раздающий по HTTP списки отзыва (CRL/DeltaCRL) и сертификаты ЦС для построения цепочки доверия (AIA). Именно эти адреса ЦС прописывает в поля CDP и AIA выпускаемых сертификатов, поэтому компонент критичен для проверки статуса и цепочки сертификатов внешними потребителями.

Конфигурационный файл: */opt/itc/web/nginx.conf*. Корневой каталог публикации — */opt/itc/web*; публикуемые файлы размещаются в подкаталогах *cdp/* (списки отзыва) и *aia/* (сертификаты ЦС). Раздача ведется по HTTP на порту 80.

Адреса раздачи:

Ресурс	URL
Список отзыва (CRL)	*http://<fqdn>/cdp/<CN>.crl*
Список отзыва (DeltaCRL)	*http://<fqdn>/cdp/<CN>-delta.crl*
Сертификат ЦС (AIA)	*http://<fqdn>/aia/<CN>.crt*

где <CN> — общее имя (CN) соответствующего центра сертификации. Актуальные файлы CRL и сертификатов в каталоги *cdp/* и *aia/* помещает Сервис публикации по заданиям публикации.

Управление службой:

CDP обслуживается системным Nginx (не пользовательским юнитом):

```
sudo systemctl reload nginx
```

8 Управление доступом (роли и права)

Доступ к функциям Clearway CA определяется ролевой моделью. Каждому пользователю при входе через сервис идентификации и аутентификации (OpenID) выдается набор прав; каждое право разрешает одну операцию — либо чтение данных определенного типа (суффикс -r), либо изменение (суффикс -w). Пользователь видит в веб-интерфейсе только те разделы и кнопки, которые соответствуют его правам; свой перечень прав пользователь может посмотреть на странице "Профиль пользователя" Контрольной панели.

Права не назначаются пользователям поштучно. Пользователь включается в группу каталога Active Directory / LDAP, а группе сопоставлен фиксированный набор прав. Сопоставление задается в конфигурации сервиса OpenID (см. подраздел Назначение ролей через группы AD/LDAP).

8.1 Перечень прав

В поставке определено 13 прав с идентификаторами вида rleMiniCA<Право>. Идентификатор передается в значении claim roles OIDC-токена и потребляется Контрольной панелью и ядром ЦС.

Право	Назначение
rleMiniCACertList-r	Просмотр списка выпущенных сертификатов
rleMiniCAEventList-r	Просмотр журнала событий
rleMiniCATemplList-r	Просмотр списка шаблонов сертификатов
rleMiniCAReqList-r	Просмотр списка запросов на выпуск
rleMiniCACRL-r	Просмотр списков отзыва (CRL/DeltaCRL)
rleMiniCAConfig-r	Просмотр конфигурации ЦС
rleMiniCACertReq-w	Создание запроса на выпуск сертификата
rleMiniCACertRev-w	Отзыв сертификата
rleMiniCACertUnRev-w	Снятие отзыва сертификата (восстановление)
rleMiniCACRL-w	Формирование и публикация списков отзыва
rleMiniCAConfig-w	Изменение конфигурации ЦС
rleMiniCATempl-w	Создание и изменение шаблонов сертификатов
rleMiniCAReqApprove-w	Одобрение (подтверждение) запросов на выпуск

8.2 Составные роли-маркеры

Дополнительно в токен могут включаться три составных маркера. Это отдельные значения claim roles, которыми помечается администратор ЦС; они задаются только для группы администраторов.

Маркер	Назначение
rleMiniCAOperator	Признак оператора ЦС
rleMiniCAAuditor	Признак аудитора
rleMiniCAMaster	Признак администратора с полными полномочиями

Маркеры rleMiniCAOperator, rleMiniCAAuditor, rleMiniCAMaster не расширяют перечень операций сами по себе — фактический доступ определяется набором прав rleMiniCA*-r/-w. Маркеры служат признаком роли учетной записи.

8.3 Стандартные роли и матрица доступа

В поставке предопределены три роли, каждая из которых соответствует одной группе каталога.

Роль	Группа AD/LDAP	Права
Администратор ЦС	ITC_MAM_Administrators	Все 13 прав rleMiniCA* + маркеры rleMiniCAOperator, rleMiniCAAuditor, rleMiniCAMaster
Пользователь (оператор)	ITC_MAM_Users	10 прав: CertList-r, EventList-r, TemplList-r, ReqList-r, CRL-r, Config-r, CertReq-w, CertRev-w, CertUnRev-w, Templ-w
Аудитор	ITC_CWCA_User	6 прав только на чтение: CertList-r, EventList-r, TemplList-r, ReqList-r, CRL-r, Config-r

Матрица доступа по отдельным правам:

Право	Администратор ЦС	Пользователь	Аудитор
CertList-r	✓	✓	✓
EventList-r	✓	✓	✓
TemplList-r	✓	✓	✓

Право	Администратор ЦС	Пользователь	Аудитор
ReqList-r	✓	✓	✓
CRL-r	✓	✓	✓
Config-r	✓	✓	✓
CertReq-w	✓	✓	–
CertRev-w	✓	✓	–
CertUnRev-w	✓	✓	–
Templ-w	✓	✓	–
CRL-w	✓	–	–
Config-w	✓	–	–
ReqApprove-w	✓	–	–

Отличия ролей:

- **Аудитор** — только чтение; не может изменять данные, выпускать или отзывать сертификаты.
- **Пользователь (оператор)** — ведет жизненный цикл сертификатов (создание запросов, отзыв, снятие отзыва) и правит шаблоны, но не управляет списками отзыва (CRL-w), не меняет конфигурацию ЦС (Config-w) и не одобряет запросы (ReqApprove-w).
- **Администратор ЦС** — полный набор прав, включая управление CRL, изменение конфигурации и одобрение запросов на выпуск.

8.4 Назначение ролей через группы AD/LDAP

Сопоставление групп каталога и наборов прав задано в карте ldapGroupRolesMap конфигурационного файла сервиса OpenID (config.yaml). Имена групп в поставке зафиксированы (ITC_MAM_Administrators, ITC_MAM_Users, ITC_CWCA_User). Чтобы назначить пользователю роль, включите его учетную запись в соответствующую группу каталога средствами AD/LDAP.

Фрагмент карты ldapGroupRolesMap (роль "Аудитор" — только чтение):

```
ldapGroupRolesMap:
  - ldapGroup: ITC_MAM_Administrators   # Администратор ЦС — полный набор + маркеры
    roles:
      - rleMiniCAMaster
      - rleMiniCAOperator
      - rleMiniCAAuditor
      # ... все 13 прав rleMiniCA*-r/-w
```

```
- ldapGroup: ITC_MAM_Users            # Пользователь (оператор) — 10 прав
  roles:
    - rleMiniCACertList-r
    - rleMiniCACertReq-w
    # ...
- ldapGroup: ITC_CWCA_User            # Аудитор — 6 прав только на чтение
  roles:
    - rleMiniCACertList-r
    - rleMiniCAEventList-r
    - rleMiniCATemplList-r
    - rleMiniCAReqList-r
    - rleMiniCACRL-r
    - rleMiniCAConfig-r
```

Чтобы изменить состав прав роли или использовать иные имена групп каталога, отредактируйте карту ldapGroupRolesMap в *config.yaml* сервиса OpenID и перезапустите сервис. Импорт групп пользователя из каталога выполняется при входе.

> ⚠ Изменение карты ldapGroupRolesMap напрямую влияет на права всех пользователей соответствующих групп. Перед правкой убедитесь, что имена групп совпадают с фактическими группами каталога, иначе пользователи войдут без прав.

8.5 Выпуск сертификатов с одобрением

Для отдельных шаблонов можно включить обязательное одобрение выпуска. Если у шаблона установлен признак `requires_approval`, запрос по такому шаблону не выпускается сразу, а переходит в состояние ожидания подтверждения.

Порядок работы:

1. Пользователь с правом rleMiniCACertReq-w создает запрос по шаблону с `requires_approval`. Запрос ставится в очередь на одобрение.
2. Пользователь с правом rleMiniCAReqApprove-w (в стандартной модели — Администратор ЦС) одобряет или отклоняет запрос.
3. Сертификат выпускается только после одобрения. Событие фиксируется в журнале значением CA_APPROVE.

> ℹ Право rleMiniCAReqApprove-w в стандартной поставке входит только в роль "Администратор ЦС". Чтобы одобрять запросы могла отдельная роль, добавьте это право нужной группе в карте ldapGroupRolesMap.

9 Обеспечение безопасной среды

После установки Clearway CA необходимо привести продукт и среду его функционирования в безопасное состояние (харденинг/hardening). В этом разделе описано, что и зачем настроить: защиту сетевых соединений центра сертификации (ЦС), защиту канала к базе данных (БД), ограничение прав на файлы, хранение ключа ЦС в аппаратном модуле, шифрование данных сертификатов в БД, механизм резервного копирования ключей (KRA), синхронизацию времени и разграничение доступа.

Все параметры ЦС задаются в конфигурационном файле */opt/itc/minica/conf/minica.yml*. После изменения файла перезапустите сервис:

```
systemctl --user restart minica.service
```

9.1 Защита соединений центра сертификации (TLS/mTLS)

По умолчанию ЦС принимает подключения без TLS (HTTP, порт 9080). Для безопасного состояния включите TLS, а для взаимной аутентификации клиентов — mTLS. При включенном TLS ЦС слушает порт 9443 (HTTPS).

Задайте в секции tls-файла *minica.yml*:

```
tls:
  enable: true              # включить TLS (HTTPS, порт 9443)
  verify: true              # включить mTLS (проверка клиентского сертификата)
  key:  $PATH/conf/tls.key  # закрытый ключ сервера
  cert: $PATH/conf/tls.crt  # сертификат сервера
  ca-cert: $PATH/conf/ca.crt # сертификат УЦ для проверки клиентов (mTLS)
  ca-certs: []              # доверенные сертификаты (mTLS)
  insecure: false           # не отключать проверку сертификата CA
```

Параметр	Назначение	Безопасное значение
tls.enable	Включение TLS на сокете ЦС; трафик переходит на порт 9443	true
tls.verify	Взаимная аутентификация (mTLS): клиент обязан предъявить сертификат, доверенный по ca-cert/ca-certs	true
tls.insecure	Отключение проверки сертификата CA	false

> ⚠ Параметр `tls.insecure: true` отключает проверку сертификата и допустим только для отладки. В рабочей конфигурации он должен быть false.

9.2 Запрет подключения к БД без TLS

Канал "ЦС → СУБД" защищается отдельно от TLS самого сервиса. За него отвечает параметр `allow-without-tls` в секции `database` (не путать с секцией `tls`, которая защищает подключения клиентов к ЦС).

По умолчанию установщик задает `allow-without-tls: true` (допускается незашифрованное подключение к БД). Для безопасного состояния установите false — тогда при отсутствии TLS на стороне СУБД сервис завершит работу и не будет передавать данные по открытому каналу:

```
database:
  url: postgres://<user>:<password>@<host>:<port>/<db>
  allow-without-tls: false   # запретить подключение к БД без TLS
```

Чтобы сертификат сервера БД действительно проверялся (а не принимался любой), в строке подключения `database.url` укажите режим `sslmode=verify-full`. Без него, даже при включенном TLS, сервис доверяет любому предъявленному сертификату.

Дополнительно на стороне СУБД PostgreSQL должны быть включены `ssl = on`, заданы серверный сертификат и ключ, а минимальная версия протокола — не ниже TLS 1.2.

Пример postgresql.conf для версии PostgreSQL 11–18.

```
ssl = on
ssl_cert_file = '/etc/postgresql/tls/server.crt'
ssl_key_file = '/etc/postgresql/tls/server.key'
ssl_min_protocol_version = 'TLSv1.2'
```

Сервис поддерживает PostgreSQL 11–18, однако в PostgreSQL 11 параметр ssl_min_protocol_version отсутствует. Для этой версии ограничение TLS 1.2 и выше настраивается отдельно.

После изменений перечитайте конфигурацию PostgreSQL, проверьте журнал на ошибки и перезапустите ЦС.

Проверка сохранённого значения на сервере ЦС:

```
grep -n "allow-without-tls" /opt/itc/minica/conf/minica.yml
```

В секции database должно быть allow-without-tls: false.

9.3 Ограничение прав доступа к файлам

Каталог установки ЦС — */opt/itc/minica*. Права на файлы приводятся к минимально необходимым с помощью прилагаемой утилиты *change_right.sh*, которая проверяет и выставляет права по описанию из файла *.right*.

1. Запустите утилиту (утилита вызывается с путем к сервису и файлом прав):

```
chmod +x change_right.sh
./change_right.sh /opt/itc/minica minica.right
```

2. Проверьте фактически выставленные права:

```
cd /opt/itc/minica
find . -type f -exec stat -c "%a %n" {} \;
```

3. По результатам проверки устраните избыточные права (например, 777). Особое внимание уделите закрытым ключам (*conf/tls.key, ca/*.key, conf/jwt.key, conf/mclient.key*) — они должны быть доступны только служебной учетной записи itc-svc.

9.4 Хранение ключа ЦС на HSM/Rutoken (PKCS#11)

Хранение закрытого ключа ЦС в аппаратном модуле (HSM, Rutoken) повышает защищенность: ключ не может быть несанкционированно извлечен даже при физическом доступе к носителю. Все криптооперации с ключом выполняет модуль через интерфейс PKCS#11.

Ключ на токене задается тем же параметром `ca.key`, значением которого выступает PKCS#11-URI (отдельного параметра `ca-uri` в продукте нет):

```
ca:
  key: "pkcs11:token=<метка_токена>;object=<метка_ключа>;type=private;pin-value=<pin>"
  cert: $PATH/ca/ca.crt
```

Чтобы OpenSSL мог обращаться к токену, в файле */opt/itc/minica/conf/openssl.cnf* активируйте движок PKCS#11 — раскомментируйте подключение секции движков и саму секцию [engine_section] (секция [pkcs11_section] в файле уже раскомментирована):

```
[ default_conf ]
ssl_conf = ssl_sect
engines = engine_section          # раскомментировать
[ engine_section ]
pkcs11 = pkcs11_section            # раскомментировать (RSA/ECDSA + Rutoken)
[ pkcs11_section ]
engine_id = pkcs11
dynamic_path = /usr/lib/x86_64-linux-gnu/engines-1.1/libpkcs11.so
MODULE_PATH  = /usr/lib/librtpkcs11ecp.so   # библиотека PKCS#11 (пример для Rutoken)
default_algorithms = ALL
```

Выше значения token, object и pin-value в URI, а также MODULE_PATH приведены как пример и зависят от используемого модуля. Метку токена и метку ключа определяет администратор при инициализации носителя средствами модуля (pkcs11-tool, утилиты производителя).

9.5 Шифрование данных сертификатов в БД

Механизм защищает данные сертификата при хранении в БД: если в шаблоне сертификата включен соответствующий признак, сертификат шифруется перед сохранением и хранится в БД в зашифрованном виде, а при скачивании автоматически расшифровывается. Это снижает риск раскрытия данных при прямом доступе к БД.

Для шифрования используется самостоятельная ключевая пара — её задает секция `ca.encryption`. Применять для шифрования ключ подписи ЦС (*ca.key/ca.crt*) недопустимо:

```
ca:
  encryption:
    key:  $PATH/ca/encr.key   # закрытый ключ для расшифрования данных
    cert: $PATH/ca/encr.crt   # сертификат для шифрования данных
```

Параметр	Назначение
ca.encryption.key	Закрытый ключ, которым расшифровываются данные сертификата при чтении из БД
ca.encryption.cert	Сертификат (открытый ключ), которым шифруются данные перед сохранением в БД

Шифрование включается признаком в шаблоне сертификата, а не глобально. Шифрование при записи и расшифрование при чтении могут снижать производительность.

9.6 Механизм KRA (резервное копирование и восстановление ключей)

KRA (Key Recovery Agent) обеспечивает резервное копирование и восстановление закрытых ключей пользователей. Закрытый ключ пользователя шифруется открытым сертификатом KRA перед сохранением; восстановление выполняет владелец закрытого KRA-ключа.

В конфигурации ЦС указывается только сертификат KRA — файл *kra.crt*. Закрытый KRA-ключ в продукте не хранится: он находится у ответственного лица и применяется им самостоятельно для расшифрования скачанной резервной копии:

```
ca:
  kra:
    cert: /opt/itc/minica/ca/kra.crt   # открытый сертификат KRA
```

Пример подготовки ключевой пары KRA (сертификат затем выпускается средствами ЦС по сформированному запросу):

```
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:4096 -out kra.key
openssl req -new -key kra.key -out kra.csr -subj "/CN=KRA/O=MyOrg/C=RU"
```

> ⚠ Закрытый ключ KRA (*kra.key*) — единственное средство восстановления резервных копий пользовательских ключей. Храните его вне ЦС в защищенном месте; его утрата делает восстановление невозможным.

9.7 Синхронизация времени и защита от повторов

ЦС проверяет метки времени (формат RFC3339) в подписанных запросах клиентов и отклоняет запрос, если рассогласование между временем клиента и временем сервера превышает допустимое. Это защищает от воспроизведения (replay) ранее перехваченных запросов.

Допустимое окно задается параметром `max-dt-sec` (в секундах). Значение по умолчанию — 600 с, то есть ±10 минут:

```
max-dt-sec: 600
```

Чтобы штатные запросы не отклонялись, на всех узлах, где размещены компоненты Clearway CA, и на сервере СУБД должна быть включена синхронизация времени по протоколу NTP с единым доверенным источником:

```
timedatectl set-ntp true
timedatectl status
```

В выводе команды `timedatectl status` убедитесь, что синхронизация активна ("System clock synchronized: yes", "NTP service: active"). Рассогласование часов между узлами и клиентами должно быть заведомо меньше `max-dt-sec`; при превышении обращения к системе отклоняются с ошибкой проверки времени.

9.8 Разграничение доступа

Ограничение полномочий пользователей и сервисов — обязательная часть безопасного состояния. Доступ к операциям ЦС разграничивается по ролям: роли назначаются членством в группах каталога (AD), а сервисные компоненты обращаются к ЦС по JWT с ограниченным набором прав. Порядок настройки ролей, групп и матрица прав приведены в разделе Управление доступом (роли и права).

10 Журналирование и мониторинг

Clearway CA ведет два вида журналов:

- **Журнал событий в базе данных ЦС** — таблица evt в СУБД хранит записи об обращениях к HTTP API центра сертификации (выпуск, отзыв, проверка статуса, получение сертификата, работа с CRL/DeltaCRL, управление шаблонами и токенами, выпуск с одобрением и др.). Записи журнала событий доступны через веб-интерфейс и API.
- **Текстовые журналы сервисов** — файлы событий, которые каждый микросервис ведет локально. Для ядра ЦС это журнал событий сервиса (*minica.log*) и отдельный журнал событий безопасности (*security.log*).

Параметры текстового журналирования задаются в конфигурационном файле ядра ЦС */opt/itc/minica/conf/minica.yml* (секции `log`, `rotate` и `secLog`). Аналогичные секции `log/rotate` присутствуют в конфигурациях остальных сервисов (например, *mocsp.yml* для OCSP-респондера).

После изменения параметров журналирования перезапустите сервис:

```
sudo systemctl restart minica.service
```

10.1 Параметры журналирования (секции log и rotate)

Clearway CA поддерживает два способа хранения текстовых журналов:

- вывод в стандартные потоки stdout/stderr с последующей обработкой системными службами journald и logrotate (сроки и объем хранения определяются настройками ОС);
- запись в автономный файл со структурированными записями в текстовом формате (json или logfmt); объем хранения регулируется встроенной ротацией (секция `rotate`).

По умолчанию события сервиса записываются в */opt/itc/minica/logs/minica.log*, события безопасности — в отдельный файл */opt/itc/minica/logs/security.log*.

Параметр	Назначение	По умолчанию
log.pipe	Способ вывода: в стандартные потоки (stdout/stderr) либо запись в файл (log.file)	stdout
log.file	Путь к файлу журнала при файловом выводе	*/opt/itc/minica/logs/minica.log*
log.format	Формат записи событий: json или logfmt	json
log.level	Минимальный уровень регистрируемых событий: flood, trace, debug, info, notice, warn, error, crit, fatal	info
log.fileMode	Права доступа к создаваемому файлу журнала (восьмеричная запись)	0640

Параметр	Назначение	По умолчанию
rotate.enable	Включение ротации автономного файла журнала	true
rotate.maxSize	Размер файла (МБ), при достижении которого выполняется ротация	10
rotate.maxAge	Срок хранения ротированных файлов (дней)	365
rotate.maxBackups	Количество хранимых резервных копий файла журнала	100

Пример секций журналирования:

```
log:
  pipe: "stdout"                          # stdout/stderr — потоки; "" вместе с file — запись
в файл
  file: "/opt/itc/minica/logs/minica.log" # путь к файлу журнала
  format: "json"                          # json | logfmt
  level: "info"                           # flood | trace | debug | info | notice | warn |
error | crit | fatal
  fileMode: "0640"                        # права доступа к файлу журнала
  rotate:
    enable: true                          # включить ротацию
    maxSize: 10                           # размер файла для ротации, МБ
    maxAge: 365                           # срок хранения, дней
    maxBackups: 100                       # число хранимых резервных копий
```

Просмотр журналов:

При выводе в стандартные потоки журнал доступен через journald:

```
sudo journalctl -u minica.service
```

При автономной записи журнал просматривается напрямую:

```
sudo tail -f /opt/itc/minica/logs/minica.log
```

10.2 Контроль целостности записей журнала

К каждой записи журнала могут добавляться два служебных поля:

- `logId` — уникальный идентификатор записи в формате UUID v7 (RFC 9562). Обеспечивает однозначную идентификацию и упорядоченность записей по времени.

- `logSum` — контрольная сумма записи по алгоритму CRC16. Несовпадение суммы при проверке означает искажение содержимого записи.

Дополнительно поддерживается цепочный режим (`sumChain`): контрольная сумма каждой записи вычисляется с учетом суммы предыдущей записи, в результате чего записи образуют связанную последовательность. Любое изъятие, добавление или перестановка записей нарушает эту цепочку и выявляется при проверке — это позволяет обнаруживать систематическую потерю или подмену записей, незаметную при независимом расчете сумм.

Параметр	Назначение	По умолчанию
idOff	Отключает добавление поля logId (UUID v7). При true идентификатор не формируется	false (logId формируется)
sumOff	Отключает добавление контрольной суммы logSum (CRC16)	false (logSum формируется)
sumChain	Цепочный режим CRC16 — сумма записи учитывает сумму предыдущей записи	true (для ядра ЦС)
sumAlone	Сохранять контрольную сумму отдельным атрибутом logSum (иначе — в младших битах logId). Используется в OCSP-респондере (*mocsp.yml*)	false

Пример записи в формате JSON (значения условные):

```
{
  "time": "2026-07-09T12:00:00Z",
  "level": "info",
  "source": "minica",
  "msg": "certificate issued",
  "logId": "0190e2a1-7c3d-7f8a-b2c1-9f4e6d5a1b20",
  "logSum": "a1b2"
}
```

Цепочный режим рекомендуется включать для журналов аудита с повышенными требованиями к защите от подделки. Для сохранности предыдущих записей включайте каталог */opt/itc/minica/logs* в общий регламент резервного копирования; при выводе в потоки хранение обеспечивается средствами journald/logrotate.

10.3 Отказоустойчивый журнал событий безопасности (secLog)

События безопасности ведутся в отдельном файле */opt/itc/minica/logs/security.log*. Механизм активируется заданием пути в параметре `secLog.path` и обеспечивает надежную регистрацию событий безопасности даже при нехватке дискового пространства.

Контроль занятости диска (watchdog): периодически проверяет свободное место в каталоге журналов и заранее предупреждает о его исчерпании:

Параметр	Назначение	По умолчанию
watchdogEnabled	Включение проактивного контроля занятости диска	false
watchdogProbeInterval	Период проверки, секунд	20
watchdogWarnPercent	Порог предупреждения (Warning) по занятости диска, %	85
watchdogCritPercent	Критический порог (Critical) по занятости диска, %	95

Уведомления: при сбоях записи журнала и достижении критических порогов система оповещает администратора по двум независимым каналам.

Параметры:

Параметр	Назначение	По умолчанию
notifyUiEnabled / notifyUiEndpoint	Уведомления в веб-интерфейсе Clearway CA (адрес приемника)	false
notifyEmailEnabled	Уведомления по электронной почте (SMTP)	false
notifyEmailSmtp	Параметры SMTP-сервера в формате host:port;username;password;recipient;authType	""
notifyEmailFrom	Адрес отправителя (иначе берется из username)	""
notifyEmailTls / notifyEmailSkipVerify	Использование TLS и пропуск проверки сертификата SMTP-сервера	false / true
notifyMinInterval	Минимальный интервал повторных уведомлений (анти-спам), секунд	30

Кольцевой буфер в ОЗУ: при недоступности диска события безопасности не теряются, а временно удерживаются в оперативной памяти и выгружаются в файл после восстановления записи.

Параметры:

Параметр	Назначение	По умолчанию
bufferMaxEvents	Максимальное число событий, удерживаемых в ОЗУ при сбое диска	50000
bufferMaxBytes	Ограничение объёма буфера, байт	52428800 (50 МБ)
preallocSizeMb	Жёсткое резервирование места под основной файл журнала при старте (fallocate), МБ; 0 — отключить	100

Фильтры регистрируемых событий: определяют, какие события безопасности подлежат регистрации. Списки поддерживают как отдельные идентификаторы (1005), так и диапазоны (1000-1100).

Параметры:

Параметр	Назначение	По умолчанию
allowedEventIds	Разрешающий список идентификаторов событий; пустой список — регистрируются все события	[]
deniedEventIds	Запрещающий список идентификаторов; пустой список — исключений нет	[]
allowedCategories / deniedCategories	Разрешающий и запрещающий списки категорий событий	[]
criticalCategories	Категории, регистрируемые даже при деградации логгера (нехватке места на диске)	[]

⚠ Отказоустойчивый журнал безопасности (secLog) реализован только в ядре ЦС (MiniCA). В OCSP-респондере (*mocsp.yml*) доступен лишь контроль целостности записей (idOff/sumOff/sumChain/sumAlone), без отдельного *security.log*, watchdog и буферизации.

10.4 Мониторинг и самодиагностика

Проверка доступности ЦС. Чтобы проверить работоспособность ядра ЦС, выполните команду клиента mclient:

```
cd /opt/itc/minica
./mclient ping
```

Успешный ответ подтверждает, что сервис запущен и принимает запросы. Команда `mclient ping` используется в том числе при развертывании как признак готовности ЦС.

Сводную статистику работы ЦС возвращает команда `mclient stat`.

Состояние службы: микросервисы работают как пользовательские юниты systemd (от учетной записи сервиса, с включенным enable-linger). Чтобы просмотреть текущее состояние службы, выполните команду:

```
systemctl --user status minica.service
```

Оповещения о событиях жизненного цикла: помимо канала событий безопасности, Clearway CA отправляет по электронной почте уведомления о событиях жизненного цикла сертификатов — выпуске, отзыве и приближении срока истечения действия сертификата. Оба канала (жизненный цикл сертификатов и события безопасности) однонаправленные: система только отправляет сообщения на внешний почтовый сервер. Для их работы требуется настроенный и доступный SMTP-сервер; если он не настроен или недоступен, уведомления не отправляются.

11 Эксплуатация и обслуживание

В этом разделе описаны штатные операции обслуживания Clearway CA: управление службами, обновление продукта, резервное копирование, архивирование истекших сертификатов и периодический контроль целостности исполняемых файлов.

11.1 Управление службами

Все службы Clearway CA работают как пользовательские systemd-юниты от имени сервисной учетной записи (по умолчанию itc-svc) и управляются через `systemctl --user`. Это позволяет обходиться без прав root при повседневном обслуживании.

Служба (юнит)	Назначение	Порт
minica.service	Ядро ЦС (MiniCA): выпуск сертификатов, ведение БД, CRL	9080 (клиент), 9443 (mTLS)
openid.service	Провайдер OpenID Connect: аутентификация через LDAP/AD, выдача токенов	443 (HTTPS), 5559 (admin HTTP)
controlPanel.service	Контрольная панель (веб-интерфейс администратора)	8407 (публикуется через Nginx на 443)
publication.service	Публикация сертификатов и CRL на точки распространения	9407
mocsp.service	Служба OCSP-ответчика	8888 (/ocsp)
archiver.service	Архиватор: перенос истекших сертификатов в архивную БД	9507
scepAdapter.service	SCEP-адаптер для выпуска сертификатов устройствам	8440 (публикуется через Nginx на 443, /api/scep)

Состав установленных служб зависит от роли узла: на корневом или подчиненном ЦС присутствует minica.service; вспомогательные службы (openid, controlPanel, publication, mocsp, archiver, scepAdapter) устанавливаются на тех узлах, где выбраны соответствующие группы установки.

Требования к окружению:

Пользовательские юниты активны без интерактивного сеанса только при выполнении двух условий, которые обеспечиваются установщиком, но их следует контролировать при обслуживании:

- Для сервисной учетной записи включен linger (продление сеанса), иначе службы останавливаются при выходе пользователя из системы:

```
sudo loginctl enable-linger itc-svc
loginctl show-user itc-svc | grep Linger   # ожидается Linger=yes
```

- В окружении сервисной учетной записи заданы переменные XDG_RUNTIME_DIR и DBUS_SESSION_BUS_ADDRESS (прописаны в *~/.bashrc*). Без них команды `systemctl --user` завершаются ошибкой "Failed to connect to bus":

```
export XDG_RUNTIME_DIR="/run/user/$(id -u)"
export DBUS_SESSION_BUS_ADDRESS="unix:path=${XDG_RUNTIME_DIR}/bus"
```

Основные команды:

Все команды выполняются от имени сервисной учетной записи (su - itc-svc). После изменения любого юнита или его конфигурации перечитайте определения служб:

```
systemctl --user daemon-reload
```

Управление отдельной службой (на примере minica.service; аналогично для остальных юнитов из таблицы выше):

```
systemctl --user enable  minica.service    # включить автозапуск
systemctl --user start   minica.service    # запустить
systemctl --user stop    minica.service    # остановить
systemctl --user restart minica.service    # перезапустить
systemctl --user status  minica.service    # состояние и последние строки журнала
```

Проверка активности службы (используется, например, перед донастройкой Контрольной панели и SCEP-адаптера):

```
systemctl --user is-active controlPanel.service   # 1 = активна
systemctl --user is-active scepAdapter.service
```

⚠ Изменения параметров в конфигурационных файлах (например, *minica.yml*, *mocsp.yml*, JSON-конфиги .NET-служб) вступают в силу только после перезапуска соответствующей службы (`systemctl --user restart <юнит>`).

11.2 Обновление продукта

Обновление Clearway CA выполняется штатным установщиком из подписанного дистрибутива. Различают:

- Полное обновление — переход на новую версию продукта (замена бинарных файлов служб и, при необходимости, миграция схемы БД).
- Частичное обновление — установка исправлений отдельных компонентов без изменения остальных.

Перед установкой обязательна проверка электронной подписи дистрибутива: она подтверждает подлинность и целостность полученного пакета. Дистрибутив, не прошедший проверку подписи, устанавливать запрещено.

Порядок обновления:

1. Выполнить резервное копирование БД, ключей и конфигураций (см. ниже).
2. Проверить электронную подпись дистрибутива.
3. Остановить обновляемые службы (systemctl --user stop <юнит>).
4. Запустить установщик и дождаться завершения обновления компонентов.
5. Выполнить systemctl --user daemon-reload, запустить службы и проверить их состояние (status, is-active).

Администратору следует отслеживать сообщения производителя о новых версиях и исправлениях безопасности, а также информацию об уязвимостях используемого системного окружения (ОС, PostgreSQL, Nginx, OpenSSL) и своевременно применять обновления.

11.3 Резервное копирование

Резервное копирование выполняется штатными средствами PostgreSQL и файловым копированием критичных каталогов. Рекомендуется автоматизировать процедуру (cron) и хранить копии на отдельном носителе.

Базы данных:

Для рабочей БД ЦС (по умолчанию cwica) и архивной БД (cwica_archive) используются утилиты PostgreSQL. Логическая выгрузка отдельной БД:

```
pg_dump -h <host> -U cwica -Fc cwica > /backup/cwica_$(date +%F).dump
```

Выгрузка всего кластера, включая роли и права (например, вместе с БД провайдера OpenID):

```
pg_dumpall -h <host> -U postgres > /backup/cluster_$(date +%F).sql
```

Восстановление из копии, созданной pg_dump в пользовательском формате:

```
pg_restore -h <host> -U cwica -d cwica --clean /backup/cwica_2026-07-10.dump
```

⚠ Резервные копии БД содержат сведения о выпущенных сертификатах и служебные данные. Храните их в защищенном месте с ограниченным доступом.

Каталоги ключей и конфигураций:

Помимо БД резервируются файлы, которые не восстанавливаются из дистрибутива:

Что копировать	Путь (по умолчанию)
Ключи и сертификаты ЦС, включая пару шифрования и сертификат KRA	/opt/itc/minica/ca/ (в т.ч. ca.key, ca.crt, encr.key, encr.crt, kra.crt)
Конфигурация ЦС	/opt/itc/minica/conf/minica.yml
Конфигурации служб OCSP, SCEP, Публикации, Панели, Архиватора	/opt/itc/**/conf/, /opt/itc/itc/config/*.json
Состояние SCEP-адаптера (SQLite)	/opt/itc/itc/sqlite/scep.db
Определения systemd-юнитов	/home/itc-svc/.config/systemd/user/*.service

Пример копирования каталога ЦС:

```
tar czf /backup/minica_conf_$(date +%F).tgz /opt/itc/minica/ca /opt/itc/minica/conf
```

⚠ Закрытые ключи ЦС (ca.key, encr.key) — самый критичный актив. Резервные копии ключей должны храниться в зашифрованном виде и отдельно от резервных копий БД.

Журналы:

Журналы служб (по умолчанию каталоги вида /opt/itc/minica/logs/, logs/mocsp.log и т. п.) включаются в состав резервных копий для последующего анализа. Ротацию и число хранимых файлов задают параметры секции журналирования соответствующего конфигурационного файла.

11.4 Архивирование истекших сертификатов

Перенос истекших сертификатов из рабочей БД ЦС в отдельную архивную БД выполняет служба Архиватор (archiver.service). Это снижает объём рабочей БД и ускоряет операции ЦС, сохраняя историю выпущенных сертификатов для последующего обращения.

Параметр	Значение по умолчанию
Служба	archiver.service (ITC.Api.MiniCA.Archiver, порт 9507)
Расписание	@daily (ежедневный прогон)
Источник (Source)	рабочая БД ЦС (cwica)
Приёмник (Target)	архивная БД (cwica_archive)

Расписание задается параметром Tasks[0].Cron в конфигурации itc/config/itc_api_miniCA_archiver_config.json; строки подключения к рабочей и архивной БД формируются из параметров Source/Target. После изменения конфигурации перезапустите службу:

```
systemctl --user restart archiver.service
systemctl --user status  archiver.service
```

Архивная БД разворачивается тем же компонентом БД, что и рабочая, и подлежит такому же резервному копированию (pg_dump).

11.5 Периодический контроль целостности исполняемых файлов

Для своевременного выявления несанкционированного изменения или повреждения программных компонентов рекомендуется периодически контролировать целостность исполняемых файлов служб путём сверки их контрольных сумм с эталонными значениями, зафиксированными сразу после установки (или обновления) продукта.

Зафиксируйте эталонные контрольные суммы исполняемых файлов после установки:

```
sha256sum /opt/itc/minica/minica \
          /opt/itc/ocsp/mocsp \
          /opt/itc/openid/openid > /opt/itc/checksums.sha256
```

Периодическая проверка (например, по расписанию cron):

```
sha256sum -c /opt/itc/checksums.sha256
```

Любое расхождение контрольной суммы (строка со статусом FAILED) означает изменение исполняемого файла и требует расследования: сверки с дистрибутивом, при необходимости —

восстановления файла из проверенного источника и повторной установки из подписанного дистрибутива.

> Эталонный файл контрольных сумм следует хранить отдельно от контролируемых файлов (например, вместе с резервными копиями) и обновлять после каждого штатного обновления продукта.

12 Ротация ключа ЦС

Ротация ключа ЦС — замена ключевой пары и сертификата центра сертификации. После неё центр продолжает обслуживать не только вновь выпускаемые, но и ранее выпущенные сертификаты. Раздел описывает, какие файлы необходимо сохранить до замены ключа, как настроить ядро ЦС и OCSP-респондер для одновременной работы с прежним и новым ЦС и как проверить результат.

После ротации требуется обеспечить:

- отзыв сертификатов, выпущенных прежним ЦС;
- выпуск CRL и Delta CRL для этих сертификатов;
- ответы на OCSP-запросы по сертификатам как прежнего, так и нового ЦС.

Что меняется при ротации:

Связь сертификата с ключом, которым он выпущен, определяется двумя идентификаторами:

- **SKI** (Subject Key Identifier) — идентификатор ключа владельца сертификата; для сертификата ЦС это идентификатор ключа самого ЦС;
- **AKI** (Authority Key Identifier) — идентификатор ключа ЦС, которым выпущен целевой сертификат.

SKI сертификата ЦС = идентификатор ключа ЦС.

AKI целевого сертификата = SKI выпустившего его ЦС.

Если заменён только сертификат ЦС, а ключевая пара осталась прежней, SKI не меняется: прежние и новые сертификаты относятся к одному ключу ЦС. Настройка каталога ca.old-keys в этом случае не требуется.

Если заменена ключевая пара, новый сертификат ЦС получает новый SKI. Фактически это новый ЦС, даже если Subject и Common Name остались прежними. Образуются два поколения сертификатов:

- Прежние сертификаты -> AKI прежнего ЦС -> прежний ключ ЦС;
- Новые сертификаты -> AKI нового ЦС -> новый ключ ЦС.

Прежние сертификаты не перестают действовать после ротации. Пока их срок не истек, необходимо поддерживать их отзыв, выпуск CRL и ответы OCSP.

12.1 Краткий порядок действий

До замены ключа ЦС:

1. Определите SKI прежнего сертификата ЦС.
2. Скопируйте прежние ca.key и ca.crt в каталог ca/old-keys, переименовав их по SKI.
3. Сохраните прежние ocsp.key, ocsp.crt и сертификат прежнего ЦС.
4. Уточните OCSP URL, записанный в уже выпущенных сертификатах.

После замены ключа:

1. Установите новые ca.key и ca.crt.
2. Перезапустите службу minica.
3. Выпустите ключ и сертификат OCSP-ответчика для нового ЦС.
4. Запустите отдельный экземпляр mOCSP для прежнего ЦС.
5. Запустите отдельный экземпляр mOCSP для нового ЦС.
6. Проверьте отзыв, CRL и OCSP отдельно для прежнего и нового сертификатов.

Каталог ca.old-keys используется только ядром ЦС (minica). Сервис mOCSP этот каталог не читает, поэтому прежние ключи ЦС из него он не получает.

12.2 Параметры конфигурации ядра ЦС

За выбор ключа ЦС и доступ к прежним ключам отвечают следующие параметры minica.yml:

```
ca-pass: ""
openssl:
  pkcs11:
    enable: false
    ca-uri: ""

ca:
  key: ca/ca.key
  cert: ca/ca.crt
  old-keys: ca/old-keys
```

Параметр	Назначение	Значение по умолчанию
ca.key	Закрытый ключ текущего ЦС	*ca/ca.key*
ca.cert	Сертификат текущего ЦС	*ca/ca.crt*
ca.old-keys	Каталог ключей и сертификатов предыдущих ЦС	*ca/old-keys*
ca-pass	Пароль файлового ключа ЦС	пусто
openssl.pkcs11.enable	Использовать ключ ЦС через PKCS#11	false
openssl.pkcs11.ca-uri	URI единственного ключа ЦС в PKCS#11	пусто

Ядро ЦС не создает архив прежних ключей автоматически. Скопировать файлы в ca/old-keys должен администратор — до того, как они будут заменены.

12.3 Сохранение прежней ключевой пары

Прежнюю ключевую пару необходимо сохранить до её замены: без неё отзыв ранее выпущенных сертификатов и выпуск CRL для них станут невозможны. Подготовка выполняется в три шага.

12.3.1 Шаг 1. Определение SKI прежнего сертификата ЦС

Значение SKI выводится командой:

```
openssl x509 \
  -in ca/ca.crt \
  -noout \
  -ext subjectKeyIdentifier
```

Пример результата:

```
X509v3 Subject Key Identifier:
    AA:BB:CC:DD:EE:FF:00:11:22:33:44:55:66:77:88:99:AA:BB:CC:DD
```

Имя файла в каталоге *ca/old-keys* должно содержать то же значение:

- без двоеточий;
- без пробелов;
- в верхнем регистре.

Для приведенного примера получится: AABBCCDDEEFF00112233445566778899AABBCCDD.

12.3.2 Шаг 2. Сохранение прежней пары в каталоге old-keys

Создайте каталог, если он еще не существует:

```
mkdir -p ca/old-keys
```

Скопируйте текущие файлы, задав имена по SKI:

```
cp ca/ca.key \
  ca/old-keys/AABBCCDDEEFF00112233445566778899AABBCCDD.key

cp ca/ca.crt \
  ca/old-keys/AABBCCDDEEFF00112233445566778899AABBCCDD.crt
```

Итоговая структура каталога:

```
ca/
├── ca.key
├── ca.crt
└── old-keys/
    ├── AABBCCDDEEFF00112233445566778899AABBCCDD.key
    └── AABBCCDDEEFF00112233445566778899AABBCCDD.crt
```

> ⚠ Доступ к закрытым ключам в каталоге ca/old-keys ограничивайте так же, как к действующему ключу ЦС: владелец — технологическая учетная запись, права 0600 (см. подраздел "Ограничение прав доступа к файлам"). Не помещайте закрытые ключи в системы контроля версий.

12.3.3 Шаг 3. Установка нового ключа и сертификата ЦС

После сохранения прежней пары замените файлы ca/ca.key и ca/ca.crt на ключ и сертификат нового ЦС, после чего перезапустите службу:

```
systemctl --user restart minica.service
```

При запуске сервис читает текущий ca.cert, извлекает его SKI и считает этот ЦС текущим.

12.4 Выбор ключа по значению AKI

Значение AKI, полученное в API-запросе, нормализуется: удаляются пробелы по краям и двоеточия, символы приводятся к верхнему регистру. Далее действует правило:

1. AKI не задан → текущие ca.key и ca.crt.
2. AKI = текущий SKI → текущие ca.key и ca.crt.
3. AKI != текущий SKI → old-keys/<AKI>.key.
4. old-keys/<AKI>.crt.

Например, для AKI AABB...CCDD будут использованы файлы:

- ca/old-keys/AABB...CCDD.key;
- ca/old-keys/AABB...CCDD.crt.

Если хотя бы один из файлов не найден, операция завершается ошибкой ERR_UNKNOWN_AKI.

12.5 Операции, использующие прежние ключи

Отзыв сертификата: при отзыве ядро ЦС читает AKI сертификата из базы данных. Если AKI совпадает с текущим SKI ЦС, используется действующая пара; если AKI относится к предыдущему ЦС — пара из каталога old-keys. Выбранные ключ и сертификат передаются OpenSSL при выполнении openssl ca -revoke. Передавать AKI в запросе клиенту API не требуется: ядро ЦС получает его из записи сертификата.

Полный CRL: эндпоинт /updcrl принимает необязательный параметр AKI: если он не передан, создается CRL текущего ЦС, если передан — CRL указанного ЦС. Для прежнего AKI ядро ЦС находит пару в каталоге old-keys, выбирает из базы отозванные сертификаты с этим AKI, формирует CRL, подписывает его прежним ключом ЦС и сохраняет в базе с соответствующим AKI.

Delta CRL: эндпоинт /updeltacrl работает аналогично: выбирает ЦС по AKI, находит последний полный CRL этого ЦС, затем последующий CRL или Delta CRL, отбирает новые записи об отзыве с тем же AKI и подписывает Delta CRL выбранным ключом ЦС.

Получение готового CRL: эндпоинты /crl и /deltacrl прежние ключи не открывают: они возвращают уже сформированный CRL из базы данных по значению AKI.

Выпуск сертификатов: каталог old-keys не позволяет выпускать новые сертификаты от имени прежнего ЦС. Операции выпуска всегда используют текущие ca.key и ca.cert.

12.6 Ограничения механизма old-keys

Пароль прежнего ключа: значение ca-pass применяется и к текущему ключу, и к файлам из каталога old-keys. Отдельный пароль для каждого прежнего ЦС конфигурацией не предусмотрен. Если прежние ключи зашифрованы разными паролями, штатная схема работать не будет — потребуется унифицировать способ доступа к ключам.

Хранение ключа в PKCS#11: при такой конфигурации:

```
openssl:
  pkcs11:
    enable: true
    ca-uri: <URI текущего ключа>
```

OpenSSL использует ключ из единственного ca-uri, а путь old-keys/<AKI>.key, выбранный ядром ЦС, как файловый ключ не передается. Механизм old-keys рассчитан прежде всего на ключи в файлах PEM: одновременный выбор нескольких прежних ключей в HSM по значению AKI не реализован.

12.7 Настройка mOCSP для прежнего и нового ЦС

В mOCSP аналога каталога old-keys нет. Один процесс mOCSP загружает:

- один закрытый ключ OCSP-ответчика;
- один сертификат OCSP-ответчика;
- один сертификат издающего ЦС.

Поэтому один процесс обслуживает ровно один ЦС, и для прежнего и нового ЦС потребуются отдельные экземпляры службы.

12.7.1 Параметры конфигурации mOCSP

Обслуживаемый ЦС и ключ подписи ответа задаются в mocsp.yml:

```
key-file: conf/ocsp.key
cert-file: conf/ocsp.crt
ca-file: conf/ca.crt
add-cert: true
hash: SHA256
nmin: 60
update-expired: true
no-trim-zeros: false
```

Параметр	Назначение	Значение по умолчанию
key-file	Закрытый RSA-ключ подписи OCSP-ответа	conf/ocsp.key
cert-file	Сертификат OCSP-ответчика	conf/ocsp.crt
ca-file	Сертификат ЦС, для которого отвечает процесс	conf/ca.crt
add-cert	Включать сертификат ответчика в OCSP-ответ	true
hash	Хеш по умолчанию; на выбор ЦС не влияет	SHA256
nmin	Время до NextUpdate, мин	60
update-expired	Обновлять в БД статус просроченного сертификата	true
no-trim-zeros	Не удалять ведущие нули серийного номера	false

При запуске mOCSP вычисляет для файла ca-file значения IssuerKeyHash и IssuerNameHash. OCSP-запрос содержит такие же значения для ЦС, выпустившего проверяемый сертификат. Если хеши запроса не совпадают с хешами настроенного ca-file, mOCSP возвращает unauthorized.

Процесс mOCSP, настроенный на новый ЦС, не сможет обслужить запрос по сертификату прежнего ЦС — и наоборот. Именно это и определяет необходимость отдельных экземпляров службы.

12.7.2 Чем подписывается OCSP-ответ

OCSP-ответ подписывается ключом из key-file и не подписывается файлом ca/old-keys/<SKI>.key: каталог old-keys ядра ЦС сервису mOCSP неизвестен.

Для прежнего ЦС потребуются:

- сертификат прежнего ЦС;
- сертификат прежнего OCSP-ответчика;
- соответствующий ему закрытый RSA-ключ ответчика.

Сертификат делегированного ответчика ocsp.crt должен:

- соответствовать ключу ocsp.key;
- быть выдан нужным ЦС;
- быть пригоден для подписи OCSP-ответов;

- содержать необходимые расширения, включая назначение OCSPSigning.

Порядок обработки запроса по сертификату прежнего ЦС:

```
Прежний целевой сертификат
        |
        | AKI прежнего ЦС
        v
OCSP-запрос с хешами прежнего ЦС
        |
        v
mOCSP прежнего ЦС
        |
        +-- ca-file   = прежний ca.crt
        +-- cert-file = прежний ocsp.crt
        +-- key-file  = прежний ocsp.key
        |
        v
OCSP-ответ, подписанный прежним ключом ответчика
```

Закрытый ключ прежнего ЦС нужен ядру ЦС для подписи CRL, но для обычной делегированной подписи ответа mOCSP он не требуется. Подписывать OCSP-ответ ключом самого ЦС возможно только тогда, когда его сертификат и закрытый ключ явно указаны в cert-file и key-file; рекомендуемая схема — отдельный делегированный OCSP-ответчик.

12.7.3 Отдельные экземпляры mOCSP

Для прежнего и нового ЦС задаются разные конфигурации. Прежний ЦС:

```
url-path: /
key-file: conf/old/ocsp.key
cert-file: conf/old/ocsp.crt
ca-file: conf/old/ca.crt
add-cert: true
nmin: 60
```

Новый ЦС:

```
listen-port: 8888
url-path: /
key-file: conf/current/ocsp.key
cert-file: conf/current/ocsp.crt
ca-file: conf/current/ca.crt
add-cert: true
nmin: 60
```

Оба процесса могут работать с одной базой данных ЦС. Различаться должны их порты, IP-адреса и URL либо правила внешней маршрутизации. После замены key-file, cert-file или ca-file процесс mOCSP необходимо перезапустить.

12.7.4 Учет OCSP URL в выпущенных сертификатах

Адрес OCSP-ответчика записывается в расширение AIA целевого сертификата при его выпуске, и у уже выпущенного сертификата изменить его нельзя. Поэтому до отключения прежнего OCSP-ответчика проверьте, к какому адресу обращаются ранее выпущенные сертификаты.

Надёжная схема разводит поколения сертификатов по разным адресам:

1. Сертификаты прежнего ЦС -> прежний OCSP URL -> mOCSP прежнего ЦС.
2. Сертификаты нового ЦС -> новый OCSP URL -> mOCSP нового ЦС

> ⚠ Если прежние и новые сертификаты обращаются к одному URL, запросы нельзя распределять между экземплярами mOCSP произвольно: экземпляр с неподходящим ca-file вернёт unauthorized. В этом случае внешний компонент должен маршрутизировать запрос к нужному ответчику с учетом издателя проверяемого сертификата.

12.8 Полная процедура ротации

Ниже сведены все шаги ротации в порядке выполнения — от подготовки резервных копий до контрольных проверок.

До ротации:

1. Создайте резервные копии ключей и сертификатов ЦС, а также комплекта OCSP.
2. Определите SKI прежнего ca.crt.
3. Сохраните прежние ca.key и ca.crt как ca/old-keys/<SKI>.key и ca/old-keys/<SKI>.crt.
4. Сохраните прежние ocsp.key и ocsp.crt.
5. Проверьте срок действия прежнего ocsp.crt.
6. Проверьте OCSP URL в ранее выпущенных целевых сертификатах.
7. Убедитесь, что прежний ЦС и комплект OCSP можно восстановить из резервной копии.

Во время ротации:

1. Замените текущие ca.key и ca.crt.
2. Перезапустите службу minica.
3. Проверьте, что команда ./mclient ping возвращает новый SKI.
4. Выпустите сертификат OCSP-ответчика от нового ЦС.
5. Настройте отдельный экземпляр mOCSP для нового ЦС.
6. Сохраните работающий экземпляр mOCSP для прежнего ЦС.

После ротации:

1. Выпустите тестовый сертификат новым ЦС.
2. Проверьте OCSP для нового сертификата.
3. Проверьте OCSP для сертификата прежнего ЦС.

4. Отзовите тестовый сертификат прежнего ЦС.
5. Сформируйте CRL прежнего ЦС, передав его AKI.
6. Проверьте подпись прежнего CRL сертификатом прежнего ЦС.
7. Убедитесь, что OCSP показывает статус revoked.
8. Повторите проверки для нового ЦС.

12.9 Проверка результата ротации

Проверки выполняются отдельно для сертификатов прежнего и нового ЦС: корректная работа одного поколения не подтверждает работу другого.

12.9.1 Проверка OCSP

Для сертификата прежнего ЦС:

```
openssl ocsp \
  -issuer conf/old/ca.crt \
  -cert conf/old/test.crt \
  -url http://old-ocsp.example.test/ \
  -resp_text
```

Для сертификата нового ЦС:

```
openssl ocsp \
  -issuer conf/current/ca.crt \
  -cert conf/current/test.crt \
  -url http://ocsp.example.test/ \
  -resp_text
```

В ответе проверьте:

- статус ответа — successful;
- статус сертификата — ожидаемый: good, revoked или unknown;
- корректность подписи ответа;
- соответствие ответа нужному издателю;
- наличие сертификата ответчика, если задано add-cert: true.

12.9.2 Проверка CRL

Сформированный CRL прежнего ЦС сохраните в файл и проверьте его реквизиты:

```
openssl crl \
  -in old-ca.crl \
  -noout \
  -issuer \
  -lastupdate \
```

```
  -nextupdate \
  -crlnumber
```

Подпись проверяется сертификатом прежнего ЦС из каталога old-keys:

```
openssl crl \
  -in old-ca.crl \
  -noout \
  -verify \
  -CAfile ca/old-keys/AABBCCDDEEFF00112233445566778899AABBCCDD.crt
```

12.10 Файлы конфигурации

Примеры конфигураций, задействованных при ротации:

- conf/minica.yml — рабочая конфигурация ядра ЦС;
- conf/minica.example.yml — расширенный пример конфигурации ядра ЦС;
- conf/mocsp.yml — рабочая конфигурация mOCSP;
- conf/mocsp.example.yml — расширенный пример конфигурации mOCSP.

13 Диагностика и устранение неисправностей

Раздел описывает штатные средства диагностики Clearway CA, коды ответов API и типовые неисправности с порядком их устранения. Все примеры приведены для установки по умолчанию: базовый каталог ЦС — */opt/itc/minic*a, файл конфигурации — */opt/itc/minica/conf/minica.yml*, сервисы запускаются как пользовательские systemd-юниты (systemctl --user) от учетной записи обслуживания (по умолчанию itc-svc).

13.1 Средства диагностики

13.1.1 Проверка доступности ЦС (mclient)

Основной инструмент проверки — консольная утилита mclient в каталоге ЦС. Запускать её следует от учетной записи обслуживания, которой доступны ключ и токен клиента (*conf/mclient.key*, *conf/mclient.jwt*):

```
su - itc-svc
cd /opt/itc/minica
# Проверка связи с ЦС (аутентификация по JWT, проверка подписи и времени)
./mclient ping
# Статистика ЦС: число выпущенных, отозванных и истекших сертификатов
./mclient stat
```

Успешный ping подтверждает, что сервис minica запущен, слушает свой порт, а токен и время клиента приняты. Ошибка ping локализует проблему до одного из уровней: сеть/порт, TLS, подпись JWT или рассогласование времени (см. таблицы кодов и типовых проблем ниже).

13.1.2 Статус служб и журналы

```
# Статус конкретного сервиса (пользовательский юнит)
systemctl --user status minica.service
# Полный журнал сервиса
journalctl --user -u minica.service
# "Хвост" журнала в реальном времени
journalctl --user -u minica.service -f -o cat
# События с момента последней загрузки, только ошибки и выше
journalctl --user -u minica.service -b -p err
```

Имена юнитов основных компонентов: minica.service (ЦС), mocsp.service (OCSP-ответчик), controlPanel.service (Контрольная панель), publication.service (Сервис публикации), archiver.service (Архиватор), scepAdapter.service (SCEP-адаптер).

Прикладной журнал ЦС дополнительно пишется в файл */opt/itc/minica/logs/minica.log*. Уровень детализации задается параметром log.level в minica.yml (flood, trace, debug, info, notice, warn, error, crit, fatal; по умолчанию info). Для разбора инцидента временно поднимите уровень до debug или trace и перезапустите сервис.

13.2 Коды ответов API

Ответ ЦС на вызов API содержит числовой код результата. 0 означает успех, ненулевые значения — конкретную причину отказа. Код результата сопровождается соответствующим статусом HTTP.

Инициатор	Код ошибки	Ошибка	Описание	Уровень
Аутентификация	5	ERR_UNAUTHORIZED- неверный JWT	JWT отсутствует, недействителен, просрочен, отозван или не подходит для запрошенного ресурса	Warn
Аутентификация	10	ERR_BAD_SIGN- неверная подпись	Криптографическая подпись запроса не прошла проверку	Warn
Авторизация	15	ERR_NO_PRIVILEGES- нет привилегий для шаблона	В JWT нет разрешения на использование выбранного шаблона УЦ	Warn
PKI / CSR	20	ERR_BAD_CSR - неверный CSR	Запрос на выпуск сертификата повреждён или содержит недопустимые данные	Warn
PKI / Политика	25	ERR_BAD_SUBJECT- Subject не соответствует политике	Поле Subject запроса не удовлетворяет политике шаблона	Warn
PKI / Сертификаты	30	ERR_CERT_NOT_FOUND- сертификат не найден	Сертификат отсутствует в базе УЦ	Warn
PKI / Сертификаты	40	ERR_CERT_EXPIRED- срок действия сертификата истек	Операция невозможна, поскольку срок действия сертификата уже истек	Warn
PKI / Сертификаты	50	ERR_CERT_REVOKED- сертификат уже отозван	Выполнена повторная попытка отозвать сертификат	Warn
PKI / Сертификаты	60	ERR_CERT_NOT_REVOKED- сертификат не отозван	Восстановление невозможно: сертификат не находится в отозванном состоянии	Warn
PKI / CRL	70	ERR_CRL_NOT_FOUND- CRL не найден	Полный список отозванных сертификатов отсутствует	Warn
PKI / CRL	80	ERR_DCRL_NOT_FOUND- Delta CRL не найден	Разностный список отозванных сертификатов отсутствует	Warn

Инициатор	Код ошибки	Ошибка	Описание	Уровень
PKI / CSR	90	ERR_CSR_NOT_FOUND- CSR не найден	Запрос на выпуск сертификата отсутствует в базе УЦ	Warn
Валидация запроса	100	ERR_BAD_REQ_TIME- неверное время запроса	Время запроса выходит за допустимый интервал	Warn
Криптография / OpenSSL	110	ERR_OPENSSL- ошибка OpenSSL	Команда или криптографическая операция OpenSSL завершилась ошибкой	Error
Криптография / OpenSSL	120	ERR_OPENSSL_TO- тайм-аут OpenSSL	Превышено время ожидания операции OpenSSL	Error
Система	130	ERR_INTERNAL- внутренняя ошибка	Внутренняя операция сервиса, например работа с файлами, завершилась ошибкой	Error
PKI / УЦ	140	ERR_UNKNOWN_AKI- неизвестный AKI	Не найден УЦ, чей SKI соответствует указанному AKI	Warn
Шаблоны	150	ERR_TEMPL_NOT_FOUND - шаблон не найден	Запрошенный шаблон сертификата отсутствует	Warn
Шаблоны	160	ERR_TEMPL_DELETED- шаблон уже удалён	Выполнена повторная попытка удалить шаблон	Warn
PKI / CSR	170	ERR_BAD_KEY_TYPE- недопустимый тип ключа	Тип открытого ключа CSR не разрешён выбранным шаблоном	Warn
PKI / CSR	180	ERR_SHORT_KEY_SIZE- недостаточная длина ключа	Длина открытого ключа CSR меньше установленного шаблоном минимума	Warn
Лицензирование	190	ERR_DEMO_VERSION_LIMIT- ограничение демоверсии	Достигнут лимит количества сертификатов для демонстрационной версии	Notice
JWT	300	ERR_JWT_NOT_FOUND- JWT не найден	JWT отсутствует в базе данных	Warn
JWT	310	ERR_JWT_REVOKED- JWT уже отозван	Выполнена повторная попытка отозвать JWT	Warn

Инициатор	Код ошибки	Ошибка	Описание	Уровень
JWT	320	ERR_JWT_NOT_REVOKED- JWT не отозван	Восстановление невозможно: JWT не находится в отозванном состоянии	Warn
Валидация запроса	400	ERR_BAD_REQUEST- неверный запрос	Переданы некорректные, неполные или неподдерживаемые параметры запроса	Warn
HTTP API	404	ERR_NOT_FOUND- ресурс не найден	Запрошенный ресурс или маршрут API отсутствует	Warn
База данных	500	ERR_DATABASE- ошибка базы данных	Ошибка обращения к СУБД или выполнения операции с данными	Error
Система	520	ERR_UNKNOWN- неизвестная ошибка	Ошибка не сопоставлена с более точным кодом	Error
PKI / PEM	540	ERR_BAD_PEM- некорректный PEM	Не удалось разобрать данные в формате PEM	Warn
PKI / PEM	580	ERR_BAD_PEM_TYPE- некорректный тип PEM	Тип PEM-блока не соответствует ожидаемому	Warn
HTTP API	600	ERR_REQUSTER_NOT_FOUND- инициатор запроса не найден	В запросе отсутствует обязательный заголовок инициатора	Warn
Шаблоны	700	ERR_TEMPL_NODB_APPR- несовместимые параметры шаблона	В шаблоне одновременно включены несовместимые режимы db и approval	Warn
Криптография / CryptoPro	800	ERR_CRYPTOPTO - ошибка CryptoPro	Операция криптографического провайдера CryptoPro завершилась ошибкой	Error
Ограничение запросов для демо весии	900	ERR_RATE_LIMIT- слишком много запросов в демо версии	Превышен допустимый лимит запросов	Warn

Помимо кода результата, диагностику упрощают строковые сообщения в журнале. Категория (инициатор) указывает узел отказа: Конфигурация, СУБД, Криптография, HTTP клиент, Клиент (JWT), PKI операции, Валидация. Например, bad request time — рассинхронизация времени; bad signature —

подпись токена; JWT expired/JWT already revoked — проблема токена; cert already revoked — повторный отзыв; certificate not found — отсутствие сертификата; can't access to database/database error — недоступность БД.

13.3 Типовые проблемы и их устранение

mOCSP не получает статус из CRL:

В режиме crl-only БД не используется, поэтому проверка её доступности не входит в диагностику этого режима. В режиме db-first необходимо проверять как ошибку запроса к БД, так и готовность резервного CRL.

Наблюдаемый результат	Возможная причина	Что проверить	Исправный результат
В OCSP-ответе указан tryLater	CRL не загружен, еще не вступил в действие либо просрочен при fail-on-expired: true	crl-path, права чтения, ThisUpdate, NextUpdate, системное время и fail-on-expired	CRL загружен и действует; запрос возвращает ожидаемый статус сертификата
После изменения файла используется прежний CRL	Автоперезагрузка отключена либо новый файл не прошёл чтение, проверку подписи или разбор	Параметр reload-enabled и сообщения журнала. Для метода poll: crl reload failed, old crl will be used. Для метода inotify: crl reload by inotify failed, old crl cache will be used.	Для метода poll в журнале есть сообщение crl reload completed; для метода inotify — crl reload by inotify completed
На системе, отличной от Linux, файл не перезагружается	Выбран метод inotify	Установить crl.source.watch.method: poll и проверить poll-interval	Изменение файла обнаруживается при очередной проверке
В режиме db-first CRL не используется для отсутствующей записи БД	БД вернула результат unknown без технической ошибки	Проверить результат запроса к БД; переход на CRL выполняется только при ошибке запроса	При технической ошибке БД в журнале появляется сообщение о переходе на CRL
Сервис использует только БД	Неизвестное или ошибочно написанное значение status-source преобразовано в db-only	Проверить точное значение: db-only, db-first или crl-only	Используется выбранный администратором источник

Для просмотра журнала mOCSP выполните:

```
journalctl --user -u mocsp.service -n 100 -o cat
```

Сервис не запускается:

Проверьте по шагам:

```
# 1. Статус и последние строки журнала
systemctl --user status minica.service
journalctl --user -u minica.service -n 50 -o cat
```

Не включен linger:

Пользовательские юниты работают без активной сессии только при включенном linger. Проверьте и включите:

```
loginctl show-user itc-svc | grep Linger      # ожидается Linger=yes
sudo loginctl enable-linger itc-svc
```

Окружение пользовательской шины:

Команды

```
systemctl --user/journalctl --user
```

требуют корректных XDG_RUNTIME_DIR и DBUS_SESSION_BUS_ADDRESS. Выполняйте их из полноценной сессии учетной записи обслуживания (su - itc-svc), а не через sudo без входа.

Файл конфигурации:

Убедитесь, что путь к конфигу верен и файл читается. Актуальное имя — minica.yml (не minica.yml):

```
ls -l /opt/itc/minica/conf/minica.yml
grep -n "max-dt-sec\|database" /opt/itc/minica/conf/minica.yml
```

Сообщения "can't parse database URL", "bad CA configuration" в журнале указывают на ошибку в конфигурации.

Права на каталог и ключи:

Каталог ЦС и ключевые файлы должны принадлежать учетной записи обслуживания; приватные ключи — с правами 0600.

Ошибка проверки времени (код 100, bad request time):

Запрос отклоняется, если разница между временем клиента и сервера превышает окно max-dt-sec. По умолчанию окно — 600 с (±10 минут).

```
# /opt/itc/minica/conf/minica.yml
max-dt-sec: 600   # окно рассогласования времени, сек
```

Действия:

- убедитесь, что на всех узлах ЦС и на сервере СУБД работает синхронизация времени по NTP:

```
timedatectl status
# ожидается: System clock synchronized: yes; NTP service: active
```

- настройте службу синхронизации (systemd-timesyncd, chrony или ntpd) согласно документации ОС.

Изменение max-dt-sec вступает в силу после перезапуска сервиса:

```
systemctl --user restart minica.service
```

> ⚠ Увеличение max-dt-sec расширяет окно приёма запросов и ослабляет защиту от повторного воспроизведения. Предпочтительно устранять причину — расхождение часов, а не наращивать окно.

Нет доступа к базе данных (код 500, can't access to database):

1. Доступность и строка подключения: проверьте, что сервер СУБД доступен, а строка подключения в секции database: файла minica.yml корректна (хост, порт 5432, имя БД, учетные данные). Поддерживаются PostgreSQL 11–18.
2. TLS-подключение к БД: по умолчанию сервис требует TLS-соединения с БД. Поведение регулируется параметром allow-without-tls внутри секции database:

```
timedatectl status
# ожидается: System clock synchronized: yes; NTP service: active
```

3. allow-without-tls: false (безопасное состояние) — при отсутствии TLS на сервере БД сервис завершает работу. Для проверки сертификата сервера БД используйте режим sslmode=verify-full и корректный корневой сертификат.
4. allow-without-tls: true (значение установщика по умолчанию) — подключение допускается и без TLS.
5. Сообщения create Postgres DAO, can't get cert stat, database error в журнале подтверждают проблему на стороне БД.

> ℹ Параметр allow-without-tls управляет только каналом "сервис → БД". Приём защищенных подключений к самому ЦС настраивается в секции tls: (enable: true, verify: true для взаимной аутентификации, порт 9443).

Проблемы TLS и сертификатов:

Симптомы: отказ ping, ошибки установления соединения, коды 5/10.

Проверьте срок действия и цепочку сертификатов компонента:

```
openssl x509 -in /opt/itc/minica/ca/certs/tls.crt -noout -dates -subject -issuer
```

Также:

1. Убедитесь, что задействована самостоятельная пара шифрования ca.encryption (*ca/encr.key*, *ca/encr.crt*), а не ключ подписи ЦС, и что путь к сертификату KRA указан как ca/kra.crt.
2. Для взаимного TLS проверьте, что клиентский сертификат выпущен доверенным ЦС и не отозван (сверьте по CRL/OCSP).
3. После замены сертификатов или правки секции tls: перезапустите сервис.

Переполнение диска под журналы (watchdog журнала безопасности):

Журнал безопасности ЦС (log.secLog) снабжён механизмом проактивного мониторинга свободного места (watchdog). При его включении сервис контролирует занятость раздела с журналом и выдает уведомления в интерфейс/по e-mail при достижении порогов:

```
# /opt/itc/minica/conf/minica.yml
log:
  secLog:
    path: "$PATH/logs/security.log"
    watchdogEnabled: true
    watchdogProbeInterval: 20   # период проверки, сек
    watchdogWarnPercent: 85     # предупреждение при 85% занятости диска
    watchdogCritPercent: 95     # критический порог при 95%
```

Действия при срабатывании:

1. Определите занятость раздела: df -h /opt/itc.
2. Освободите место: архивируйте и вынесите старые журналы, настройте ротацию.
3. При достижении критического порога не отключайте механизм — устраните нехватку места, иначе возможна потеря событий безопасности.

> ⚠ Отказоустойчивый журнал безопасности буферизует события в ОЗУ. При длительном переполнении диска и исчерпании буфера часть событий может не записаться на диск. Реагируйте на предупреждение (85%) до достижения критического порога (95%).

13.4 Быстрый чек-лист диагностики

1. ./mclient ping — связь, аутентификация и время в норме?
2. systemctl --user status <юнит> — сервис активен, linger включен?
3. journalctl --user -u <юнит> -p err — какие ошибки и какой инициатор (категория)?
4. Код ответа API — сопоставьте с таблицей кодов, определите узел отказа.
5. timedatectl status — синхронизация времени активна?
6. Доступность БД и параметры секции database: (TLS, строка подключения).
7. df -h /opt/itc — достаточно ли места под журналы.

14 Приложения

Справочные материалы для быстрого доступа к ключевым параметрам конфигурации, сетевым портам, ролевой модели и командам консольного клиента. Значения приведены для установки по умолчанию; переменная $PATH в путях соответствует корневому каталогу ЦС (по умолчанию /opt/itc/minica).

14.1 Приложение А. Ключевые параметры minica.yml

Основные параметры файла конфигурации сервиса ЦС conf/minica.yml. Полное описание секций приведено в разделах настройки; ниже — сводка наиболее востребованных при эксплуатации параметров.

Параметр	Назначение	Значение по умолчанию
tls.enable	Включение TLS сервера ЦС (установщик переводит в true при выпуске серверного сертификата)	false
tls.verify	Включение взаимной аутентификации TLS (mTLS)	false
tls.key / tls.cert	Ключ и сертификат сервера ЦС	$PATH/conf/tls.key, $PATH/conf/tls.crt
http-port	HTTP-порт API (при выключенном TLS)	9080
https-port	HTTPS-порт API (при включенном TLS)	9443
timeout-ms	Таймаут обработки запроса, мс	60000
max-dt-sec	Максимальное рассогласование времени при проверке подписи запроса, с (окно ±10 минут)	600
log.level	Уровень журналирования (flood…fatal)	info
log.format	Формат журнала (json, logfmt, tinted, default)	json
log.file	Файл журнала сервиса	$PATH/logs/minica.log
log.rotate.maxSize / .maxAge	Ротация журнала: размер (МБ) и срок хранения (дней)	10 / 365
log.sumChain	Контроль целостности: цепочка контрольных сумм записей журнала	true

Параметр	Назначение	Значение по умолчанию
log.secLog.path	Файл отказоустойчивого журнала безопасности (при задании пути механизм активируется)	$PATH/logs/security.log
ca.key / ca.cert	Ключевая пара и сертификат подписи ЦС	$PATH/ca/ca.key, $PATH/ca/ca.crt
ca.days.cert	Срок действия выпускаемых сертификатов, дней	1460
ca.days.crl	Срок действия CRL, дней	365
ca.start-sn / ca.start-crl	Стартовые серийные номера сертификатов и CRL	100000000000000000000
ca.overlap	Смещение начала срока действия сертификата в прошлое, мин	10
ca.encryption.key / .cert	Отдельная пара ключей для шифрования (не совпадает с ключом подписи ЦС)	$PATH/ca/encr.key, $PATH/ca/encr.crt
ca.kra.cert	Сертификат агента восстановления ключей (KRA)	$PATH/ca/kra.crt
jwt.key	Ключ подписи JWT-токенов	$PATH/conf/jwt.key
jwt.cache-minutes	Время хранения ключей проверки JWT в кеше, мин	5
master-key	Мастер-ключ проверки подписи запросов	$PATH/conf/mclient.pem
sign-off	Отключить проверку подписи запросов	false
openssl.cnf	Базовый конфигурационный файл OpenSSL	$PATH/conf/openssl.cnf
chain-file	Файл цепочки сертификатов (PEM)	$PATH/ca/chain.pem
database.type	Тип СУБД	postgres
database.url	Строка подключения к БД (формируется из параметров подключения)	postgres://...@host:5432/...
database.allow-without-tls	Разрешить подключение к БД без TLS (для безопасного состояния — false)	true

Параметр	Назначение	Значение по умолчанию
database.templ-cache-minutes	Время хранения шаблонов в кеше, мин	3

Поддерживаемые версии PostgreSQL — 11–18. Параметр database.allow-without-tls управляет каналом "сервис → СУБД", а не подключениями к самому ЦС; для проверки сертификата сервера БД используйте в строке подключения sslmode=verify-full.

14.2 Приложение Б. Сетевые порты

Порты, используемые компонентами Clearway CA (значения по умолчанию). Веб-сервисы публикуются через Nginx (TLS 443) как обратный прокси на локальный порт приложения; сами прикладные порты слушаются на localhost и наружу не выставляются.

Компонент	Порт	Назначение
ЦС (MiniCA)	9443/TCP	HTTPS/mTLS API управления ЦС (клиент mclient, служебные компоненты)
ЦС (MiniCA)	9080/TCP	HTTP API (когда TLS выключен)
СУБД PostgreSQL (ЦС, OpenID, архив)	5432/TCP	Доступ сервисов к базам данных
OpenID (OIDC-провайдер)	443/TCP	HTTPS: аутентификация пользователей, выдача токенов
OpenID (административный)	5559/TCP	Служебный HTTP-интерфейс администрирования
Служба каталогов LDAP/AD	389/636 TCP	Подключение OpenID к каталогу (LDAP / LDAPS)
Nginx (Контрольная панель, Сервис публикации, SCEP)	443/TCP	HTTPS-фронтенд (обратный прокси к веб-сервисам)
CDP (точки распространения)	80/TCP	HTTP-раздача CRL и сертификатов
Контрольная панель (backend)	8407/TCP	Локальный порт приложения (за Nginx)
Сервис публикации	9407/TCP	Локальный порт приложения (за Nginx)
SCEP-адаптер	8440/TCP	Локальный порт приложения (за Nginx, /api/scep)

Компонент	Порт	Назначение
Архиватор	9507/TCP	Локальный порт приложения (перенос данных в архивную БД)
OCSP-респондер (mOCSP)	8888/TCP	Приём и обработка OCSP-запросов

14.3 Приложение B. Матрица ролей и прав

Роли назначаются членством в группах службы каталогов (Active Directory) и задаются картой соответствия ldapGroupRolesMap в конфигурации OpenID. Имена групп по умолчанию: Администратор — ITC_MAM_Administrators, Оператор/Пользователь — ITC_MAM_Users, Аудитор — ITC_CWCA_User. Каждое право — атомарный маркер вида rleMiniCA<Право> в claim roles OIDC-токена. Суффикс -r означает чтение (read), -w — изменение (write).

Право	Администратор	Оператор/Пользователь	Аудитор
CertList-r — просмотр списка сертификатов	+	+	+
CertReq-w — запрос и выпуск сертификата	+	+	—
CertRev-w — отзыв сертификата	+	+	—
CertUnRev-w — восстановление (снятие отзыва) сертификата	+	+	—
ReqList-r — просмотр списка запросов	+	+	+
ReqApprove-w — подтверждение (одобрение) запроса	+	—	—
TemplList-r — просмотр списка шаблонов	+	+	+
Templ-w — создание и изменение шаблонов	+	+	—
CRL-r — получение и просмотр CRL	+	+	+
CRL-w — обновление и публикация CRL	+	—	—
Config-r — просмотр конфигурации	+	+	+

Право	Администратор	Оператор/Пользователь	Аудитор
Config-w — изменение конфигурации	+	–	–
EventList-r — просмотр журнала событий	+	+	+
Operator — маркер роли "Оператор"	+	–	–
Auditor — маркер роли "Аудитор"	+	–	–
Master — маркер полного доступа	+	–	–

Администратору назначается полный набор прав (16 маркеров), Оператору/Пользователю — 10, Аудитору — 6 (только чтение). Маркеры Operator, Auditor, Master — составные роли-признаки, входящие только в набор администратора.

14.4 Приложение Г. Основные команды mclient

Консольный клиент mclient предназначен для управления ЦС из командной строки. Запуск — из каталога установки ЦС от имени технологической учетной записи (по умолчанию itc-svc). Адрес сервера и учетные данные задаются глобальными флагами -url <URL>, -jwt <файл>, -key <файл> / -pass <пароль> либо через конфигурацию (-conf, переменная MCLIENT_CONF) и переменные окружения (MINICA_URL, MCLIENT_KEY, MCLIENT_PASS, MCLIENT_JWT). Локальные команды: help, version, showconf, mkconf.

Команда	Назначение
ping	Проверка доступности сервера ЦС
stat	Получение статистики по сертификатам
ca	Выпуск сертификата по CSR (флаги -csr, -template, -days, -out)
revoke	Отзыв сертификата с указанием причины (-reason) и комментария
repair	Восстановление отозванного сертификата (возврат в статус "действителен")
status	Получение статуса сертификата (действителен / отозван / истек / неизвестен)
get	Получение сертификата из БД в формате PEM

Команда	Назначение
csr	Получение CSR, связанного с сертификатом
find	Поиск сертификатов по критериям с постраничной выборкой (-num, -offset, -status)
export	Экспорт сертификатов по фильтрам в каталог
delete	Удаление сертификата и связанного CSR из БД (без восстановления)
chain	Получение полной цепочки сертификатов до корневого ЦС
updcrl	Принудительное обновление полного CRL
crl	Получение актуального полного CRL в файл
updeltacrl	Обновление разностного списка отзыва (Delta CRL)
deltacrl	Получение актуального Delta CRL в файл
templates	Получение списка шаблонов сертификатов
gettempl	Получение полной конфигурации шаблона по имени и версии
addtempl	Создание нового шаблона выпуска
deltempl	Удаление шаблона
mkjwt	Выпуск и подпись JWT-токена (-jwt-name, -jwt-role, -jwt-key, -out)
getjwt	Получение информации о JWT-токене
revokejwt	Отзыв JWT-токена по sub или jti
repairjwt	Восстановление ранее отозванного JWT-токена
renewjwt	Продление срока действия JWT-токена
revivejwt	Повторная активация истекшего JWT-токена
deljwt	Удаление JWT-токена из БД

⚠ Команды delete (сертификата) и deljwt удаляют данные из БД без возможности восстановления. Команда ca с флагом -nodb выпускает сертификат без сохранения в БД — такой сертификат не будет учитываться при отзыве и построении CRL.