

# Центр Управления Гетерогенными Инфраструктурами

Система централизованного анализа и  
управления гетерогенными  
инфраструктурами — MiniCA

**Руководство администратора**

RU.CLRW.00947-01 32 01

ООО «Клируэй технолоджис»

Релиз Q3 2025

## Содержание

1.	Глоссарий.....	5
1.1.	Принятые сокращения.....	5
1.2.	Термины и определения .....	6
2.	Введение.....	7
2.1.	Идентификационные данные.....	7
2.2.	Общее описание .....	7
3.	Архитектура системы.....	8
3.1.	Описание архитектуры MiniCA .....	10
3.2.	Описание архитектуры MiniCA-web.....	11
3.3.	Описание архитектуры mOSCP .....	12
3.4.	Типовая архитектура внедрения. ....	13
3.5.	Рекомендуемые аппаратные требования к серверам (виртуальным или физическим).....	14
3.6.	Требования к АРМ оператора.....	16
3.7.	Требования к системному ПО.....	16
4.	Установка и настройка корневого MiniCA.....	19
4.1.	Установка .....	19
4.2.	Проверка OpenSSL.....	20
4.3.	Настройка PostgreSQL. ....	20
4.4.	Каталоги установки MiniCA.....	22
4.5.	Структура файлов и каталогов .....	24
4.6.	Параметры конфигурации minica .....	26
4.7.	Опции командной строки minica .....	26
4.8.	Параметры конфигурации mclient.....	27
4.9.	Описание строк конфигурации mclient .....	28
4.10.	Опции командной строки mclient.....	29
4.11.	Опции конфигурационного файла openssl. ....	30
4.12.	Создание БД и хранимых процедур .....	31
4.13.	Настройка сервиса systemd .....	34
4.14.	Установка микросервиса minica без наличия прав суперпользователя.....	36
5.	Установка и настройка выдающего MiniCA.....	39
5.1.	Установка .....	39
5.2.	Настройка PostgreSQL. ....	39
5.3.	Каталоги установки MiniCA.....	39
5.4.	Параметры конфигурации minica .....	39
5.5.	Параметры конфигурации mclient.....	40
5.6.	Создание БД и хранимых процедур .....	40
5.7.	Настройка сервиса systemd .....	40
6.	Создание сертификатов и настройка конфигурационных файлов MiniCA.....	41
6.1.	Выпуск корневого сертификата ЦС.....	41

6.2.	Выпуск сертификата выдающего ЦС: .....	43
6.3.	Создание сертификата для TLS подключения minica. ....	44
6.4.	Перенастройка службы minica на выпущенные сертификаты.....	45
6.5.	Обновление закрытой и открытой пары ключей.....	47
6.6.	Перенастройка службы postgresql на выпущенные сертификаты.....	48
6.7.	Перенастройка файла конфигурации minica.yml для выполнения требования безопасности.....	49
7.	Установка и настройка микросервиса minica-web-panel. ....	52
7.1.	Каталоги установки службы.....	54
7.2.	Параметры конфигурации minica-web-panel .....	55
7.3.	Параметры конфигурации файла client_env.js.....	57
7.4.	Настройка сервиса systemd .....	58
7.5.	Установка микросервиса minica-web-panel без наличия прав суперпользователя	59
7.6.	Создание сертификатов для minica-web-panel.....	59
	Установка и настройка сервиса Nginx. ....	62
7.7.	Настройка клиента в Keycloak.....	66
7.8.	Создание токенов .....	72
8.	Установка и настройка микросервиса mOCSP .....	74
8.1.	Каталоги установки службы.....	75
8.2.	Параметры конфигурации mOCSP.....	75
8.3.	Настройка сервиса systemd .....	79
8.4.	Создание сертификатов для mOCSP. ....	81
8.5.	Перенастройка файла конфигурации mocsp.yml для выполнения требования безопасности.....	82
9.	Настройка сервера точек распространения.....	84
9.1.	Каталоги распространения. ....	84
9.2.	Параметры конфигурации Nginx .....	84
9.3.	Настройка скриптов распространения. ....	85
9.4.	Автоматизация обновления CRL с помощью cron.....	87
9.5.	Настройка точек распространения в конфигурационном файле minica. ....	88
	Приложение А Последовательность шагов при разворачивании minica, minica web-panel, mocsp. ....	90
A.1	Установка minica .....	90
A.1.1	Создание каталогов .....	90
A.1.2	Настройка сервиса .....	91
A.1.3	Создание сертификатов.....	91
A.1.4	Выпуск сертификата выдающего ЦС .....	91
A.1.5	Создание TLS- ключа для postgresQL.....	92
A.1.6	Перенастройка службы minica на выпущенные сертификаты.....	92
A.1.7	Выполнить обновление закрытой и открытой пары ключей.....	93
A.1.8	Перенастройка minica.yml для выполнения требования безопасности.....	93
A.2	Установка и настройка PostgreSQL. ....	94
A.2.1	Редактирование конфигурационных файлов.....	94

A.2.1	Перенастроить PostgreSQL на новые сертификаты .....	95
A.2.2	Настройка БД postgresql .....	95
A.3	Установка и настройка микросервиса minica web-panel. ....	97
A.3.1	Параметры конфигурации файла client_env.js .....	98
A.3.2	Настройка сервиса systemd .....	98
A.3.3	Создание сертификатов для web-minica. ....	99
A.3.4	NGINX.....	99
A.4	Настройка микросервиса mocsp .....	100
A.4.1	Перенастройка файла конфигурации mocsp.yml .....	101
A.4.2	Настройка точек распространения .....	102
A.4.3	Настройка скриптов распространения .....	102
A.4.4	Автоматизация обновления CRL.....	103

# 1. Глоссарий

## 1.1. Принятые сокращения

Сокращение	Расшифровка
API	Application Programming Interface, Прикладной программный интерфейс компонента ИС.
CSR	Certificate Signing Request — это зашифрованный запрос на выпуск сертификата в формате PKCS10, содержащий подробную информацию его параметрах.
PKI (ИОК)	Public Key Infrastructure, инфраструктура открытых ключей (ИОК) — набор средств, распределенных служб и компонентов, используемых для поддержки крипто-задач (шифрования, аутентификации, подписи) на основе закрытого и открытого ключей.
RSA	Rivest - Shamir -Adleman — реализация криптосистемы на основе закрытого и открытого ключей.
TLS / mTLS	Transport Level Security / mutual Transport Level Security — развитие протокола SSL, криптографический протокол на основе сертификатов x509, обеспечивающий организацию безопасной сетевой связи и взаимную аутентификацию клиента и сервера.  SSL (Secure Socket Layers) — криптографический протокол, обеспечивающий организацию безопасной сетевой связи между клиентом и сервером и упрощенной проверкой подлинности сторон связи с применением сертификатов x509.
БД	База данных.
ВМ	Виртуальная машина.
MiniCA, Система,	Центр Сертификации на основе OpenSSL.
ИС	Информационная система.
ИТ, IT	Информационные технологии, Information Technology.
ОС	Операционная система.
УЦ	Удостоверяющий Центр — программный или программно-аппаратный комплекс, обеспечивающий управление жизненным циклом сертификатов x509.
ЦС (CA)	Центр Сертификации (Certification Authority) — сервер, предназначенный для выпуска и отзыва сертификатов, а также публикации CRL (COC). Часть Удостоверяющего Центра.
ЦУГИ	Центр Управления Гетерогенными Инфраструктурами — набор (система) программ для управления и мониторинга различных компонентов ИТ инфраструктуры, разработанный и лицензируемый предприятием ООО "Клируэй Текнолоджис".

## 1.2. Термины и определения

Термин	Определение
Агент ЦУГИ	Отдельное приложение Windows / Linux / Mac OS X.
Микросервис	Отдельная программа, являющаяся частью программной системы или проекта, реализующая отдельный функциональный или информационный блок и тесно связанная с другими частями системы или проекта через сетевые вызовы API.
Роль	Набор системных и объектных прав, которые могут быть выданы и отозваны как единое целое, и после добавления этой Роли, могут быть временно активированы и деактивированы во время существования сессии.
Endpoint (Эндпойнт)	Конечная точка, с помощью которой моментальные снимки метрик могут предоставляться через HTTP в различных форматах.

## 2. Введение

Настоящий документ является составной частью эксплуатационной документации на Систему Автоматизированного Управления Инфраструктурой Удостоверяющего Центра (САУИ УЦ), которая создана как составная часть Центра управления гетерогенными инфраструктурами (ЦУГИ) в ООО «Клируэй Текнолоджис».

Документ предназначен для ознакомления технического персонала с основными операциями по работе с системой.

### 2.1. Идентификационные данные

Идентификационные данные приведены в таблице 1

Таблица 1 — Идентификационные данные

Идентификационные данные ОО	
Название документа	Руководство администратора
Децимальный номер документа	RU.CLRW.00947-01 34 01
Версия документа	1.0
Автор документа, предприятие-изготовитель	ООО «Клируэй Текнолоджис»

### 2.2. Общее описание

MiniCA реализует автономный сервер центра сертификации на базе **OpenSSL**. Микросервис, который реализует программный интерфейс с использованием протокола **HTTP** (или **HTTPS**). Основные функции сервиса:

- выпуск сертификатов **X.509v3** на основе **CSR** запросов в формате **PKCS#10**;
- отзыв сертификатов в соответствии с заданной причиной;
- формирование списка отозванных сертификатов **CRL (RFC 5280)**;
- формирование разностных списков отозванных сертификатов **Delta CRL**;
- предоставление информации о статусе сертификата;
- предоставление сертификатов по их серийному номеру или **SHA1** отпечатку;
- сохранение информации о всех запросах, сертификатах, **CRL/Delta CRL** и журнала событий в базе данных.

### 3. Архитектура системы

MiniCA реализует автономный многопоточный HTTP сервер с возможностями установления TLS или mTLS (mutual TLS) в зависимости от параметров конфигурации. Сервис эксплуатирует утилиту OpenSSL для выполнения всех криптографических операций, а также операций с различными контейнерами ASN.1 (CSR запросы PKCS#10, сертификаты X.509v3, CRL/DeltaCRL).

MiniCA построен на базе модульных компонентов, написанных на языках программирования C# (.Net 8.0.x) и Go (1.21 и выше).

Ядром системы является сервис центра сертификации на базе OpenSSL. Система реализует программный интерфейс с использованием протокола HTTP (или HTTPS).

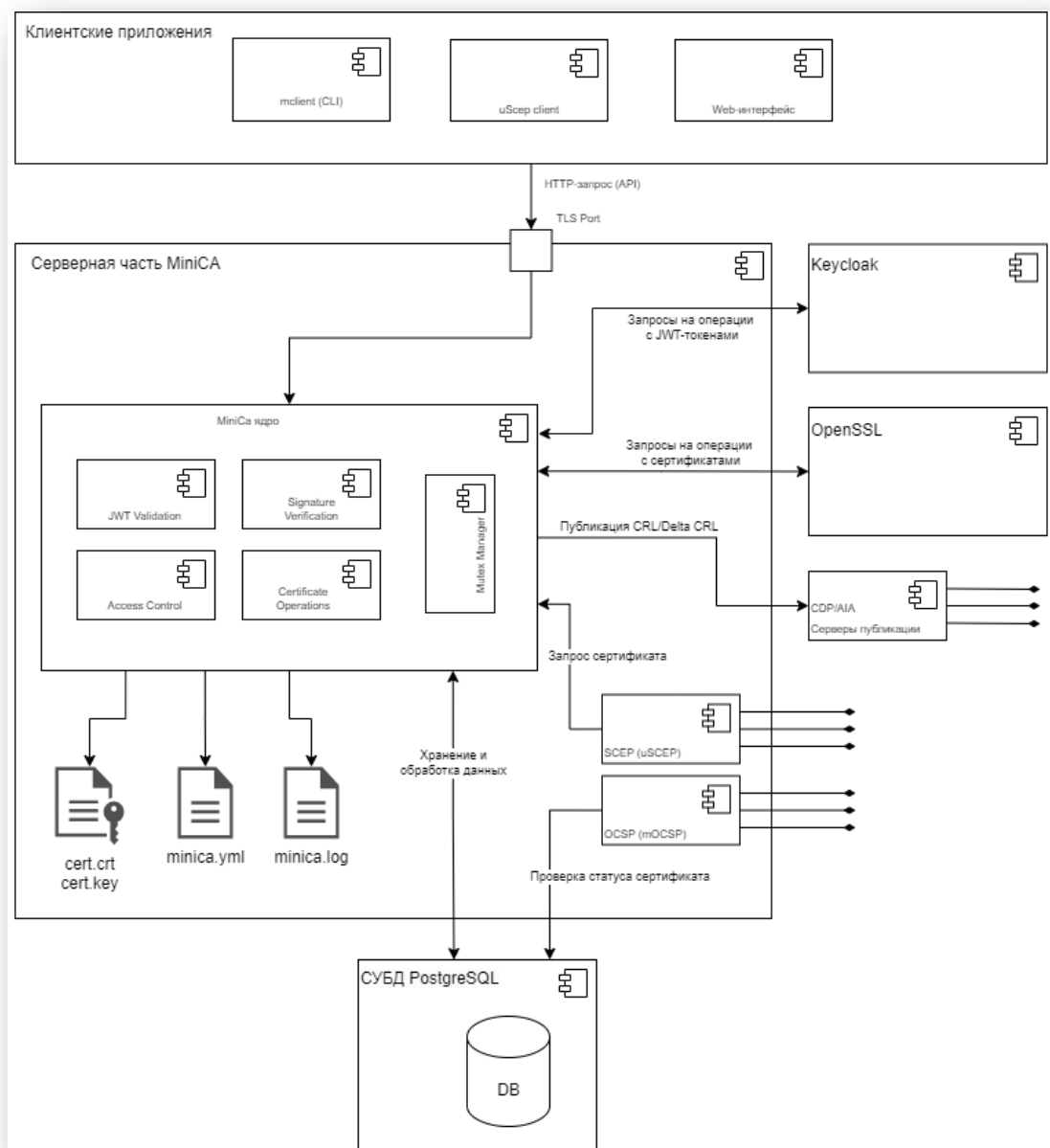


Рисунок 1 — Архитектурная схема MiniCA

Также имеется ряд дополнительных программных компонентов, которые дополняют сервис MiniCA. В таблице ниже приведено описание функций компонентов.

Таблица 2 — Компоненты MiniCA и их функции

Компонент	Тип	Необходимые сторонние компоненты	Конфигурационный файл	Функции
MiniCA	сервис (демон)	PostgreSQL, OpenSSL	MiniCA.yml	Обеспечивает выполнение всех необходимых функций по управлению жизненным циклом сертификатов.
mclient	консольное приложение		mclient.yml	Внутреннее клиентское приложение для вызова функций MiniCA с использованием интерфейса командной строки
mOCSP	сервис (демон)		mocsp.yml	Реализация сервиса OCSP с предоставлением статуса сертификатов в реальном времени
Web интерфейс	сервис (демон)	Nginx, KeyCloak	itc_Api_MiniCA_controlPanel_config.json	Обеспечение основных функций по управлению сертификатами через графический интерфейс пользователя
uSCEP	сервис (демон)		uscep.yml	Обеспечивает интерфейс для запроса сертификатов по протоколу SCEP

Работа микросервиса возможна с OpenSSL версий 1.1.1x и 3.x.

На время выполнения операций с OpenSSL применяется блокировка (мьютекс). Система может быть масштабирована использованием нескольких микросервисов MiniCA, работающих параллельно с одной базой данных.

Протокол информационного взаимодействия между сервисом и клиентом основан на передаче структур данных JSON (запрос → ответ). Все запросы к микросервису должны быть подписаны закрытым ключом, известным только клиенту. Открытый ключ для проверки подписи запросов сервис извлекает из X.509v3 сертификата, заданного конфигурацией. Опционально проверка подписи может быть отключена.

Операционная система — GNU/Linux. Целевая система Astra Linux SE 1.7 «Воронеж». Возможна эксплуатация с использованием других распространенных дистрибутивов ОС Linux (Debian/Ubuntu/Fedora) и их производных. При необходимости микросервис может быть запущен в контейнере docker (podman).

Микросервис реализует программный интерфейс с использованием протокола HTTP (или HTTPS). Основные функции микросервиса:

- выпуск сертификатов X.509v3 на основе CSR запросов в формате PKCS#10;
- отзыв сертификатов в соответствии с заданной причиной;
- формирование списка отозванных сертификатов CRL (RFC 5280);
- формирование разностных списков отозванных сертификатов DeltaCRL;
- предоставление информации о статусе сертификата;
- предоставление сертификатов по их серийному номеру или SHA1 отпечатку;
- сохранение информации о всех запросах, сертификатах, CRL/DeltaCRL и журнала событий в базе данных.

### 3.1. Описание архитектуры MiniCA

Дистрибутив состоит из двух исполняемых программ:

- MiniCA — приложение микросервиса;
- mclient — клиентское приложение для вызова функций сервиса с использованием интерфейса командной строки.

Для управления конфигурацией OpenSSL используется традиционный файл конфигурации `openssl.cnf` (имя и путь могут быть заданы в конфигурационном файле MiniCA). Параметры выпускаемых сертификатов (шаблоны/атрибуты) определяются конфигурацией OpenSSL.

Для автоматизации запуска микросервиса рекомендуется применение службы systemd, входящей в большинство современных дистрибутивов Linux.

В качестве СУБД используется PostgreSQL. СУБД может быть развернута как на локальной, так и на удаленной машине. Реквизиты доступа к базе данных указываются в параметрах конфигурации.

Конфигурация микросервиса описывается в конфигурационном файле в формате JSON. Некоторые опции могут быть заданы опциями командной строки (путь к конфигурационному файлу, опции журнала). Единственная используемая переменная окружения `MINICA_CONF` может использоваться для указания пути в конфигурационном файле микросервиса.

Диаграмма взаимодействия программных компонентов приведена на следующем рисунке:

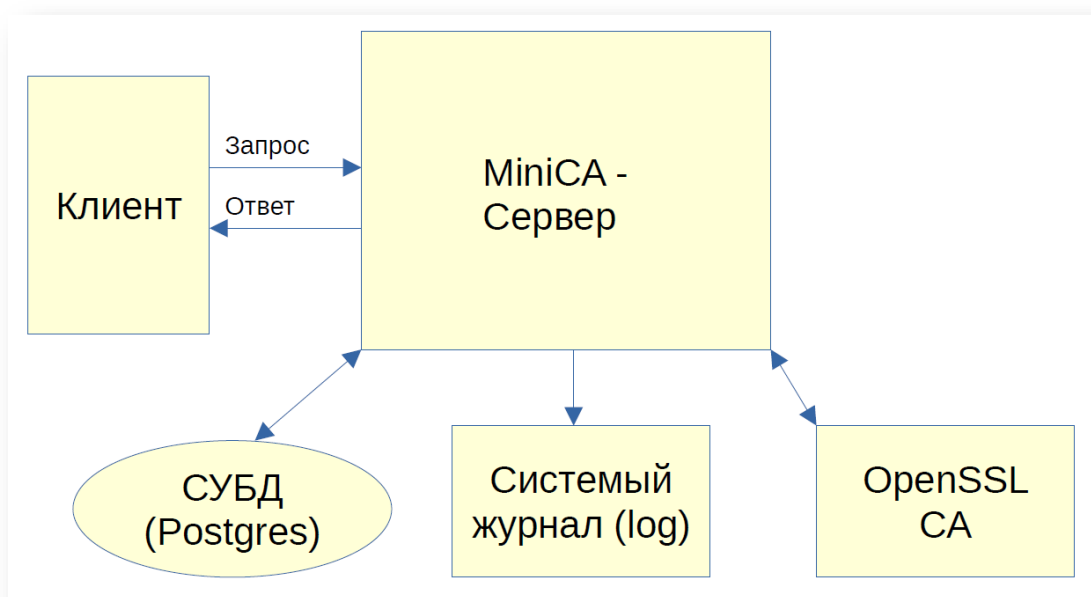


Рисунок 2 — Диаграмма взаимодействия программных компонентов

### 3.2. Описание архитектуры MiniCA-web

Графическая панель управления MiniCA-web предназначена для доступа к функциям выдающего ЦС, а также размещения CDP, OCSP и прочих компонентов.

MiniCA-web состоит из двух пакетов и одной исполняемой программы:

- minica-cp-service— приложение микросервиса;
- minica-spa — интерфейс приложения и конфигурационные файлы.
- Микросервис также использует программу Nginx для перенаправления web интерфейса.

Для автоматизации запуска микросервиса рекомендуется применение службы `systemd`, входящей в большинство современных дистрибутивов Linux.

В качестве СУБД используется PostgreSQL. Реквизиты доступа к базе данных указываются в параметрах конфигурации.

Конфигурация микросервиса описывается в конфигурационном файле в формате JSON. Некоторые опции могут быть заданы опциями командной строки (путь к конфигурационному файлу, опции журнала). Диаграмма взаимодействия программных компонентов приведена на следующем рисунке:

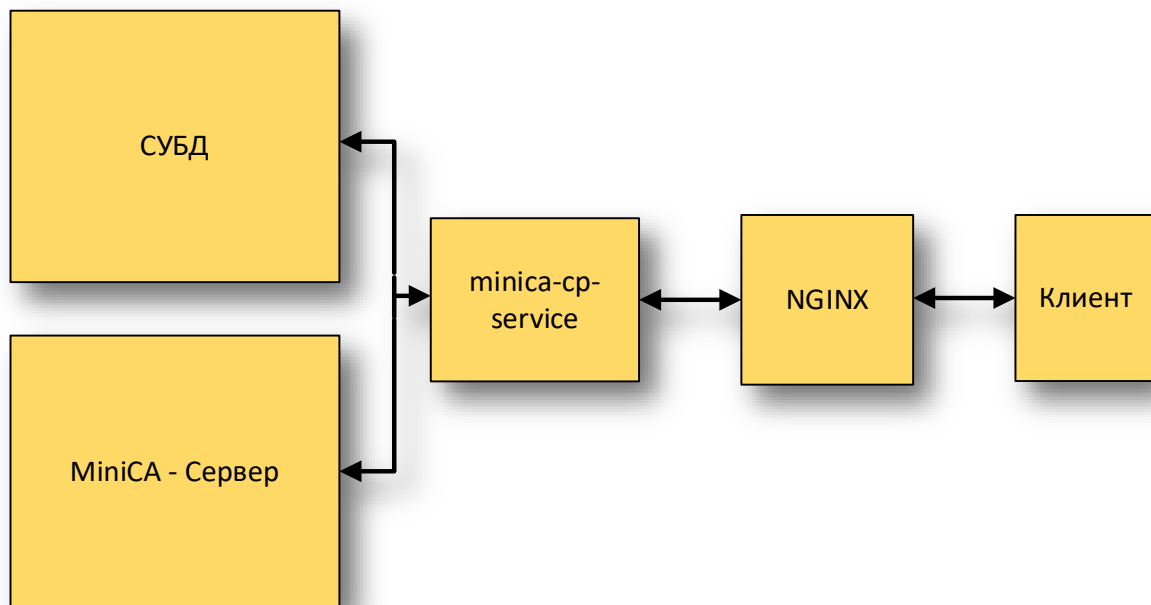


Рисунок 3 — Диаграмма взаимодействия программных компонентов MiniCA-web

### 3.3. Описание архитектуры mOSCP

mOSCP специализированный микросервис для быстрой онлайн-проверки текущего статуса цифровых сертификатов, используемый в инфраструктуре открытых ключей (PKI). Состоит из одного пакета и одной исполняемой программы:

- mocsr — приложение микросервиса;
- Для автоматизации запуска микросервиса рекомендуется применение службы `systemd`, входящей в большинство современных дистрибутивов Linux.

В качестве СУБД используется PostgreSQL. Реквизиты доступа к базе данных указываются в параметрах конфигурации.

Конфигурация микросервиса описывается в конфигурационном файле в формате JSON. Некоторые опции могут быть заданы опциями командной строки (путь к конфигурационному файлу, опции журнала). Диаграмма взаимодействия программных компонентов приведена на следующем рисунке:

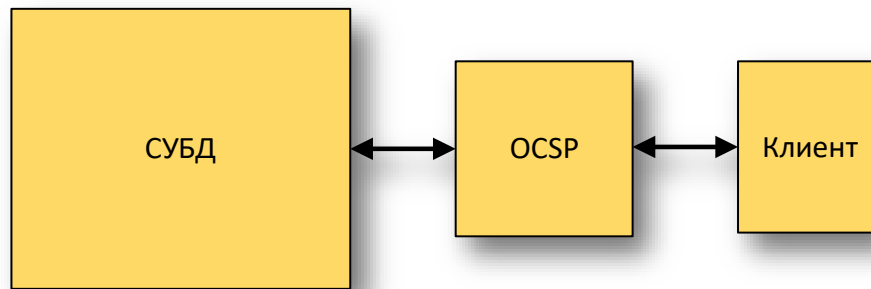


Рисунок 4 — Диаграмма взаимодействия программных компонентов MOSCP

### 3.4. Типовая архитектура внедрения

Пример установки продукта на стенде может выглядеть следующим образом:

- Экземпляр MiniCA, который выполняет функции корневого сервера;
- Экземпляр MiniCA, выпускающего сервера;
- База данных PostgreSQL;
- Точки распространения, сервер OSCP и панель web-MiniCA, на одном сервере.

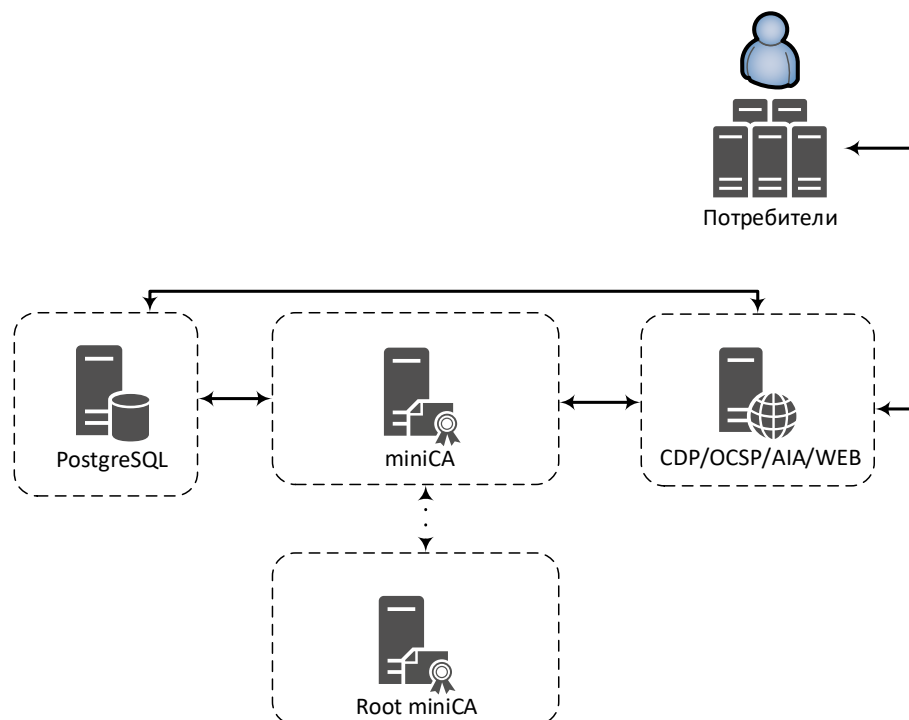


Рисунок 5 — Схема стенда MiniCA

### 3.5. Рекомендуемые аппаратные требования к серверам (виртуальным или физическим)

Таблица 3 — аппаратные требования к минимальной конфигурации

№	Наименование ТС	Кол-во серверов	CPU, кол-во ядер	CPU, тактовая частота, не менее, ГГц	RAM, Гб	SSD, Гб	ОС	Описание
1)	Сервер MiniCA + СУБД PostgreSQL	1	4	2	4	256	Astra Linux 1.7.5 Orel	Включает роли PostgreSQL, Web-консоль, AIA, CDP и OCSP. Рекомендации по применению: корневой УЦ, выдающий УЦ для небольших организаций (до 100 сертификатов в сутки).

Таблица 4 — рекомендуемые аппаратные требования к базовой конфигурации

№	Наименование ТС	Кол-во серверов	CPU, кол-во ядер	CPU, тактовая частота, не менее, ГГц	RAM, Гб	SSD, Гб	ОС	Описание
1)	Сервер MiniCA + СУБД PostgreSQL + сервер web-управления	1	4	2	8	512	Astra Linux 1.7.5 Orel	Включает роли PostgreSQL, web-консоль, AIA, CDP и OCSP. Рекомендации по применению: выдающий УЦ для организаций со средней нагрузкой (до 5000 сертификатов в сутки).

Таблица 5 — рекомендуемые аппаратные требования к типовой конфигурации

№	Наименование ТС	Кол-во серверов <sup>1</sup>	CPU, кол-во ядер	CPU, тактовая частота, не менее, ГГц	RAM, Гб	SSD, Гб	ОС	Описание
1)	Сервер MiniCA	1 или 2	8	2	16	256	Astra Linux 1.7.5 Orel	Основной сервер MiniCA. Рекомендации по применению: выдающий УЦ, для высоконагруженных систем, с использованием K8S, service mesh
2)	Сервер СУБД PostgreSQL	1 или кластер СУБД	4	2	8	512	Astra Linux 1.7.5 Orel	Сервер PostgreSQL
3)	Сервер web-управления	1 или 2	2	2	4	100	Astra Linux 1.7.5 Orel	Web-консоль, AIA, CDP и OCSP

<sup>1</sup> 2 сервера используются для распределения нагрузки и обеспечения отказоустойчивой работы компонентов MiniCA

### 3.6. Требования к АРМ оператора

Таблица 6 — Требования к АРМ оператора

№	Требования к ПК для рабочих мест	ПК	Ядер	Тактовая частота не менее, ГГц	RAM, ГБ	SSD, ГБ	Браузер	Операционная система
1)	АРМ оператора	1	2	2	4	80	Google Chrome версия не ниже 113, браузеры на Chromium (Microsoft Edge, Яндекс.Браузер).	Windows 10+

### 3.7. Требования к системному ПО

Таблица 7 — Требования к системному ПО

№	Сервер / АРМ, на который устанавливается ПО	Наименование ПО	Минимальная версия	Рекомендуемая версия	Назначение
Системное ПО, необходимое для работы прикладного ПО					
1)	Сервер web-консоль	dot.Net	8.0	8.0	Компонент системы
		nginx	Установка из репозитория		Веб сервер
		curl	Установка из репозитория		Обращение к API системы
		jq	Установка из репозитория		Для скрипта формирования

№	Сервер / АРМ, на который устанавливается ПО	Наименование ПО	Минимальная версия	Рекомендуемая версия	Назначение
					токена для обращения к API
2)	Сервер MiniCA	psql	Установка из репозитория	Не ниже версии БД	Для создания БД и таблиц
3)	Keycloak сервер	KeyCloak	19.0.3	24.0.2 и выше	Авторизация
		openjdk	11 (зависит от версии Keycloak)	17 (зависит от версии Keycloak)	Для работы keycloak
		nginx	Установка из репозитория		Веб сервер
4)	Сервер СУБД	PostgreSQL	11	15	База данных
		etcd	Установка из репозитория		Обеспечение работы кластера PostgreSQL
		patroni	Установка из репозитория		Обеспечение работы кластера PostgreSQL
Системное ПО, необходимое для первоначальной настройки и эксплуатации					
5)	АРМ администратора	Dbeaver +pg utils(pg_dump, pg_dumpall,pg_re- store,psql) +driver PostgreSQL	21.3.1	Самая последняя	Для настройки и эксплуатации БД
		WinSCP	5.13.7	Самая последняя	Доступ к файлам серверов

№	Сервер / АРМ, на который устанавливается ПО	Наименование ПО	Минимальная версия	Рекомендуемая версия	Назначение
		putty	0.70	Самая последняя	Доступ к серверам по ssh
		liquibase	4.7.1	Самая последняя	Первичное наполнение/обновление БД
		Google Chrome	113	Самая последняя	Доступ к веб-интерфейсу системы

## 4. Установка и настройка корневого MiniCA

Дистрибутив MiniCA поставляется в виде DEB-пакета для операционных систем архитектуры amd64 (x86\_64) основанных на dpkg (Debian, Ubuntu, Astra Linux). Поставка в виде RPM пакета и/или tar.gz архива может быть выполнена по требованию Заказчика.

Наименование пакета представлено как minica-X.Y.Z-hash\_amd64.deb, где:

- X.Y.Z — старший номер версии, младший номер версии, номер сборки соответственно;
- hash — хэш идентификатор коммита в репозитории исходных текстов Git;
- amd64 — целевая архитектура.

Пакет содержит исполняемый файл сервиса minica, тестовый клиент mclient, примеры файлов конфигурации и примеры ключей/сертификатов для тестирования.

В данной инструкции используется состав серверов, приведенный в Таблице Таблица 8Таблица 8.

Адреса и FQDN серверов приведены только для примера.

Таблица 8 — Данные серверов

Назначение сервера	IP тестовый	Hostname тестовый
PostgreSQL	172.24.240.13	pg1.test.local
Корневой сервер MiniCA	172.24.240.10	root-ca.test.local
Выдающий сервер MiniCA	172.24.240.11	minica.test.local
сервер MiniCA-web	172.24.240.12	minica-web.test.local
Сервер Keycloak	172.24.240.21	Keycloak.test.local
Сервер контроллер домена Active Directory		ad.test.local

Для корректного выполнения команд, указанных в инструкции в конкретной ИТ-инфраструктуре, необходимо IP адреса, FQDN и прочие названия, приведенные в инструкции, изменить на свои.

### 4.1. Установка

Для установки необходимо авторизоваться на сервере root-ca.test.local, выполняющего роль корневого ЦС. Используемые для примера в настоящей инструкции имена серверов и адреса приведены в разделе 1.1.

- 1) Скопировать на сервер установочную программу любым доступным способом.
- 2) Установить пакет при помощи терминала командой типа:

```
$ sudo dpkg -i ./minica-X.Y.Z-hash_amd64.deb
```

где секция `./minica-X.Y.Z-hash_amd64.deb` представляет собой путь до файла пакета. Необходимо указать действительный путь до файла пакета

Установка из самораспаковывающего архива:

- 1) Выдать права на выполнение:

```
$ sudo chmod 755 ./minica-sfx-2.4.4-d48e974_amd64-2025.06.20.run
```

- 2) Запустить архив:

```
$ sudo ./minica-sfx-2.4.4-d48e974_amd64-2025.06.20.run
```

## 4.2. Проверка OpenSSL

Приложения в deb-пакете (minica и mclient) не требуют установки на целевую систему дополнительных программных библиотек, поскольку не являются динамическими исполняемыми файлами.

Однако, для функционирования микросервиса в системе необходимо наличие установленной утилиты командной строки OpenSSL.

Чтобы проверить, что утилита OpenSSL установлена, в терминале необходимо выполнить команду определения текущей версии:

```
$ openssl version
```

Пример успешного вывода:

```
# OpenSSL 3.0.11 19 Sep 2023 (Library: OpenSSL 3.0.11 19 Sep 2023)
```

## 4.3. Настройка PostgreSQL.

В качестве СУБД используется PostgreSQL. СУБД может быть развернута как на локальной, так и на удаленной машине. Реквизиты доступа к базе данных указываются в параметрах конфигурации.

Внимание!

Настройка PostgreSQL, производится на корневом центре сертификации и выдающем. На корневом ЦС PostgreSQL ставится локально, на выдающем WC может устанавливаться локально или отдельно, как это указано на схеме.

Настройка:

- 1) Настроить репозиторий на сервере БД — репозиторий настраивается согласно инструкции на ОС.
- 2) Выполнить обновление репозитория и установить сервер PostgreSQL:

```
$ sudo apt update
$ sudo apt install postgresql -y
```

или

```
$ sudo dnf install postgresql -y
```

Внимание!

Названия пакетов PostgreSQL могут различаться в зависимости от операционной системы. Перед установкой рекомендуется использовать команду поиска в файловом менеджере ОС.

- 3) Открыть файл `postgresql.conf`:

```
$ sudo vim /etc/postgresql/15/main/postgresql.conf
```

Внимание!

Конфигурационные файлы PostgreSQL могут находиться по различным путям и директориям в разных ОС.

- 4) Внести изменения в строки файла `postgresql.conf`, для того чтобы PostgreSQL начал принимать внешние запросы. Должны быть указаны следующие значения параметров:

```
1: listen_addresses='*'          # (change requires restart)
2: port = 5432                   # (change requires restart)
3: max_connections = 500         # (change requires restart)
```

- 5) Внести изменения в строки файла `postgresql.conf`, для работы PostgreSQL по TLS:

```
1: ssl = on
2: ssl_ca_file = 'ca.crt'
3: ssl_cert_file = '/etc/ssl/certs/test.pem'
4: ssl_key_file = '/etc/ssl/private/test.key'
```

Внимание!

К данному пункту необходимо будет вернуться позже, когда сервер MiniCA будет работать, чтобы выпустить сертификаты для службы `postgres`.

- 6) Сохранить внесенные изменения и закрыть `postgresql.conf`.
- 7) Открыть файл конфигурации `pg_hba.conf`:

```
$ sudo vim /etc/postgresql/15/main/pg_hba.conf
```

- 8) Изменить строки в `pg_hba.conf` для настройки доступа к PostgreSQL. Должны быть указаны следующие значения параметров:

```
1: local all postgres peer
2: # TYPE DATABASE USER ADDRESS METHOD
3: # "local" is for Unix domain socket connections only
4: local all all peer
5: # IPv4 local connections:
6: #host all all 127.0.0.1/32 md5
7: host all all 0.0.0.0/0 md5
8: # IPv6 local connections:
9: host all all ::1/128 md5
10: # Allow replication connections from localhost, by a user with the
11: # replication privilege.
12: local replication all peer
13: host replication all 127.0.0.1/32 md5
14: host replication all ::1/128 md5
```

Пример настройки файла `pg_hba.conf` без дополнительных ограничений соответствует стандартным требованиям.

- 9) Сохранить внесенные изменения и закрыть `pg_hba.conf`
- 10) После изменений параметров PostgreSQL, необходимо перезапустить службу:

```
$ sudo systemctl restart postgresql
```

- 11) Проверить состояние службы командой:

```
$ sudo systemctl status postgresql
```

Внимание!

Не допускается при эксплуатации использование RSA ключей и X.509v3 сертификатов из состава дистрибутива!

#### 4.4. Каталоги установки MiniCA

При установке deb-пакета MiniCA автоматически добавляется пользователь `minica` с домашним каталогом `/app/itc/minica/`.

Для настройки сервиса на сервере `minica` необходимо выполнить следующие шаги:

- 1) При установке из самораспаковывающего архива необходимо создать директорию `/app/itc/`:

```
$ sudo mkdir -p /app/itc/
```

- 2) Переместить каталог из самораспаковывающего архива в созданную директорию и создать подкаталог для хранения журналов:

```
$ sudo mv minica /app/itc/minica  
$ sudo mkdir -p /app/itc/minica/logs/
```

- 3) Скопировать исполняемые файлы из директории `bin` в каталог `minica`:

```
$ sudo scp /app/itc/minica/bin/* /app/itc/minica/
```

Внимание:

Данный пункт не обязателен, но в представленном примере установка осуществлена при условии размещения исполняемых файлов MiniCA в корневой директории ПО

- 4) Добавить технологическую учетную запись (ТУЗ) системного пользователя `minica`.

```
$ sudo adduser minica
```

- 5) Изменить пароль (если потребуется)

```
$ sudo passwd minica
```

- 6) Создать `env` файл, в каталоге `/app/itc/minica/minica.env`:

```
$ sudo vim /app/itc/minica/minica.env
```

- 7) Задать в созданном файле переменные окружения для сервиса `minica`:

```
1: # Переменные окружения сервиса `minica`  
2: # =====
```

```
3: # Опции командной строки передаваемые через minica.service
4: # (опции имеют приоритет перед переменными окружения)
5: MINICA_OPTIONS="-log-level=debug -log-format=tint"
6: # Путь к файлу конфигурации
7: MINICA_CONF="conf/minica.yml"
8: # Переменные окружения конфигурации логера
9: #LOG_PIPE="stdout"
10: #LOG_FILE=""
11: #LOG_FILE_MODE="0644"
12: #LOG_LEVEL="flood"
13: #LOG_FORMAT="tinted"
14: #LOG_GOID="1"
15: #LOG_ID="1"
16: #LOG_SUM="1"
17: #LOG_SUM_CHAIN=""
18: #LOG_SUM_ALONE="1"
19: #LOG_TIME=""
20: #LOG_TIME_MICRO=""
21: #LOG_TIME_LOCAL="no"
22: #LOG_TIME_FORMAT="RFC3339Micro"
23: #LOG_SRC="1"
24: #LOG_SRC_PKG="1"
25: #LOG_SRC_FUNC="1"
26: #LOG_SRC_EXT="1"
27: #LOG_COLOR="1"
28: #LOG_LEVEL_OFF=""
29: #LOG_ROTATE=""
30: #LOG_ROTATE_MAX_SIZE="10"
31: #LOG_ROTATE_MAX_AGE="10"
32: #LOG_ROTATE_MAX_BACKUPS="100"
33: #LOG_ROTATE_LOCAL_TIME=""
34: #LOG_ROTATE_COMPRESS=""
```

8) Сохранить и закрыть файл

#### 4.5. Структура файлов и каталогов

- `minica` — является исполняемым файлом приложения MiniCA и обычно размещается в одном из каталогов сервера (при развертывании из пакета используется каталог `/app`):

- `/app/itc/minica/;`
- `/app/itc/minica/bin/.`
- `mclient` — файл клиента MiniCA с интерфейсом командной строки, может размещаться в произвольном месте файловой системы (обычно в одном из каталогов `bin`, доступных через переменную окружения `PATH`).

При разворачивании пакета файл размещается в каталоге `/app/itc/minica/`, при необходимости администратор может создать требуемую символическую ссылку из каталога доступного в переменной окружения `PATH` самостоятельно.

- `minica.conf` — конфигурационный файл микросервиса в формате JSON.

Рекомендуется размещать (или размещается) в каталоге `/app/itc/minica/conf` сервера (при разворачивании из пакета используется каталог `/app`). Конфигурационный файл по умолчанию может быть создан с помощью однократного запуска программы `minica` с опцией `mkconf`. Путь к конфигурационному файлу задается:

- через опцию командной строки сервиса `-conf` (в приоритете);
- через переменную окружения `MINICA_CONF`.

В конфигурационном файле сохраняются реквизиты доступа к базе данных и другая важная для работы сервиса информация, потому права доступа к файлу (в т.ч. на чтение) для широкой группы пользователей должны быть ограничены и по возможности зашифрованы штатными средствами `minica`.

- `minica.service` — файл `systemd`, размещается в каталоге `/etc/systemd/system` при ручном развёртывании сервиса, используется при запуске сервиса как системного.

При установке пакета файл размещается в стандартном каталоге `/lib/systemd/system/`.

- `minica-user.service` — файл `systemd`, размещается в домашнем каталоге пользователя `~/.config/systemd/user`.

При разворачивании пользовательского сервиса должен сохраняться под именем `minica.service`. При разворачивании из пакета файл-заготовка `minica-user.service` помещается в каталог `/app/itc/minica/`.

- `minica.env` — предназначен для задания переменных окружения:
  - `MINICA_OPTIONS`: используются в сервисных файлах `systemd` (`*.service`);
  - `MINICA_CONF`: указывает путь к основному конфигурационному файлу `minica.conf`.

При развертывании из пакета файл размещается в каталоге `/app/itc/minica/`.

- `openssl.cnf` — конфигурационный файл OpenSSL.

При развертывании из пакета файл размещается в каталоге `/app/itc/minica/conf/`.

- `mclient.yml` — конфигурационный файл `mclient`.

При развертывании из пакета файл размещается в каталоге `/app/itc/minica/conf/`.

#### 4.6. Параметры конфигурации minica

Источником конфигурации является конфигурационный файл `minica.yml`.

Для настройки конфигурации:

- 1) Открыть файл с настройками minica.

```
$ sudo vim /app/itc/minica/conf/minica.yml
```

- 2) Изменить строку подключение к БД в формате «url: postgres://имя пользователя:пароль@адрес:порт/бд»:

```
1: url: postgres://minica:1234578@localhost:5432/minica
```

- 3) Изменить строки:
  - `ca-pass` — установить пароль от ключа `root`;
  - `db-pass` — установить пароль к базе данных;
  - `allow-without-tls: false` — изменить на `true`, чтобы разрешить работу TLS.
- 4) Сохранить файл с внесенными настройками.

Настройки являются первичными для функционирования MiniCA без использования TLS и защитой данных в базе.

#### 4.7. Опции командной строки minica

Доступные общие опции (указываются до команды в командной строке):

Таблица 9 — Доступные общие опции

Опция	Значение
<code>-h --h</code>	Вывести короткую справку по опциям (флагам).
<code>-help --help help</code>	Вывести полный перечень опций и команд.

Опция	Значение
-v --version version	Вывести версию и выйти.
-conf CONFIG	Задать путь к конфигурационному файлу (JSON).
-log LEVEL	Задать уровень журналирования (flood, trace, debug, info, warn, error, critical, fatal).
-slog	Принудительно включить структурированный логгер slog (TextHandler).
-jlog	Принудительно включить структурированный логгер slog (JSONHandler).
-tlog	Принудительно включить структурированный цветной логгер (TintHandler).
-lsrc	Принудительно выводить в журнал имена файлов и номера строк.
-lpkg	Выводить имя файла и каталога (пакета).
-ltime	Выводить метку времени в журнал.
-ltimefmt	Задать формат метки времени в журнале (в го стиле ~15.04.05.999).

#### Доступные команды

Таблица 10 — Доступные команды

Команда	Значение
mkconfg [conf]	Создать конфигурационный файл по умолчанию (образец) и выйти.
showconf	Вывести файл конфигурации на stdout (JSON) и выйти.
start	Запустить сервис (действие по умолчанию).

## 4.8. Параметры конфигурации mclient

Источником конфигурации является конфигурационный файл `mclient.yml`.

- 1) Открыть файл с настройками mclient:

```
$ sudo vim /app/itc/minica/conf/mclient.yml
```

- 2) Найти строку, определяющую URL доступа к серверу MiniCA:

```
1: # URL доступа к серверу MiniCA
2: server-url: http://localhost:8080
```

3) Заменить строку подключения:

```
1: # URL доступа к серверу MiniCA
2: server-url: http://root-ca.test.local:8080
```

4) Сохранить файл после внесения изменений.

## 4.9. Описание строк конфигурации mclient

Таблица 11 — Строки конфигурации mclient

Параметр	Значение	Описание
ServerURL	https://localhost:8443	URL доступа к серверу MiniCA
tls-enable	true	Использовать TLS
mtls-enable	false	Использовать mTLS (mutual TLS)
tls-cert-ca	conf/ca.crt	X.509v3 сертификат доверенного ЦС (PEM формат)
mtls-key	conf/mtls.key	Клиентский закрытый ключ клиента mTLS (PEM формат)
mtls-cert	conf/mtls.crt	X.509v3 сертификат клиента mTLS (PEM формат)
mtls-client	mclient	Предъявляемое серверу имя mTLS клиента
sign-off	true	Отключить подпись запросов в адрес сервера
key-file	conf/itc.key	Закрытый ключ для подписи запросов к серверу
log		Параметры журналирования
file	stdout	Файл журнала или stdout/stderr
file-mode	0640	Права файла журнала при создании (если не stdout/stderr)
level	info	Уровень журналирования (от flood/trace/debug до critical/fatal)
slog	false	Использовать стандартный slog.TextHandler
json	false	Использовать стандартный slog.JsonHandler
tint	false	Использовать xlog.TintHandler на основе slog (с подсветкой вывода)
time	false	Добавлять метку времени в журнал
time-us	false	Выводить метку времени с точностью до мкс
time-tint		Go-формат метки времени xlog.TintHandler (например 15.04.05.999)

Параметр	Значение	Описание
src	false	Добавлять в журнал информацию об исходных файлах, номер строки
src-long	false	Добавлять в журнал информацию об исходных каталоге/файле
no-level	false	Исключить метку уровня журналирования
no-color	false	Отключить ANSI раскраску журнала xlog.TintHandler
prefix		Добавить префикс в традиционный журнал (не slog)
add-key		Добавить ключ ко всем сообщениям журнала
add-value		Добавить значение (к заданном выше ключу)

#### 4.10. Опции командной строки mclient

Доступные общие опции (указываются до команды в командной строке):

Таблица 12 — Опции mclient

Опция	Описание
-url URL	Базовый URL сервера
-conf CONFIG	Задать конфигурационный файл (JSON)
-key KEY_FILE	Файл с закрытым ключом RSA для подписи запросов
-log LEVEL	Задать уровень логирования (trace/debug/info/warm/error/fatal)
-slog	Принудительно включить структурированный логгер slog (TextHandler).
-jlog	Принудительно включить структурированный логгер slog (JSONHandler).
-tlog	Принудительно включить структурированный цветной логгер (TintHandler).
-lsrc	Принудительно выводить в журнал имена файлов и номера строк.
-lpkg	Выводить имя файла и каталога (пакета).
-ltime	Выводить метку времени в журнал.
-ltimefmt	Задать формат метки времени в журнале (в го стиле ~15.04.05.999).

Доступные команды (без обращения к серверу):

Таблица 13 — Доступные команды без обращения к серверу

Команда	Описание
-h --h	Вывести короткую справку по опциям (флагам)
help -help --help	Вывести полный перечень опций и команд.
version -version --version	Вывести номер версии.
mkconfg [conf]	Создать конфигурационный файл по умолчанию (образец).
showconf	Вывести файл конфигурации на stdout (JSON) и выйти.

Доступные команды (обращение к функциям сервиса по HTTP API)

Таблица 14 — Доступные команды обращения по HTTP API

Команды (HTTP API)	Описание
ping	Проверить связь с сервером
ca	Отправить CSR-запрос на выпуск сертификата
revoke	Отозвать сертификат
status	Запросить статус сертификата (по серийному номеру или отпечатку)
get	Запросить сертификат (по серийному номеру или отпечатку)
updcrl	Обновить (выпустить) CRL
crl	Запросить последний CRL
updeltacrl	Обновить (выпустить) DeltaCRL
deltacrl	Запросить последний DeltaCRL

Опции команд (указываются после команды в командной строке)

Таблица 15 — Опции команд

Опции команд	Описание
-csr FILE	Файл с CSR-запросом в PEM формате (для команды ca)
-requester STRING	Тот, кто запрашивает сертификат (пользователь/хост) (для команды ca)
-days N	Срок действия сертификата (для команды ca)
-ext STRING	Расширения сертификата (для команды ca)
-context STRING	Контекст запроса (для команды ca)
-serial SN	Серийный номер сертификата (для команд ca/revoke/get/status/delete/find)
-fingerprint HEX	Отпечаток сертификата (для команд revoke/get/status/delete/find)
-reason STRING	Причина отзыва сертификата (для команды revoke)
-comment STRING	Комментарий по отзыву сертификата (для команды revoke)
-last_update DATE	Дата обновления CRL (для команд updcrl/updeltacrl)
-next_update DATE	Следующая дата обновления CRL (для команд updcrl/updeltacrl)
-next_publish DATE	Следующая дата публикации CRL (для команд updcrl/updeltacrl)

Временные метки задаются в формате «YYMMDDHHMMSSZ».

Переменная окружения `MCLIENT_CONF` может быть использована для указания пути к файлу конфигурации, если не задана опция `-conf`.

#### 4.11. Опции конфигурационного файла openssl.

Файл конфигурации OpenSSL играет ключевую роль в работе Центра Сертификации (ЦС). По умолчанию он располагается по пути `/app/itc/minica/conf/openssl.cnf`. Перед выпуском

корневого сертификата и сертификата выдающего ЦС необходимо предварительно настроить этот файл. В данной инструкции этапы настройки файла `openssl.cnf` не будут рассмотрены.

## 4.12. Создание БД и хранимых процедур

Сервис MiniCA использует встроенные SQL-процедуры PostgreSQL. При создании таблиц базы данных (таблица запросов, таблица сертификатов, таблица CRL и таблица событий) необходимо загрузить SQL-процедуры. Ниже представлена примерная последовательность создания БД и процедур.

На сервере базы данных выполнить последовательность команд:

- 1) Создать пользователя (роль) Postgres с именем `minica`:

```
$ sudo -u postgres createuser -l -P minica
```

При выполнении команды потребуется два раза ввести пароль.

Для смены пароля в дальнейшем можно выполнить:

```
1: $ sudo -i -u postgres psql
2: ALTER ROLE minica WITH PASSWORD 'NewPasswd';
3: ALTER ROLE postgres WITH PASSWORD 'NewPasswd';
4: CREATE DATABASE minica OWNER minica;
5: GRANT ALL PRIVILEGES ON DATABASE minica TO minica;
6: \q
```

- 2) Вывести список созданных баз данных следующей командой:

```
$ sudo -u postgres psql
# \l
# \q
```

```
postgres=# \l
```

Name	Owner	Encoding	Collate	Ctype	ICU Locale	Locale Provider	Access privileges
minica	minica	UTF8	ru_RU.UTF-8	ru_RU.UTF-8		libc	=Tc/minica + minica=CtC/minica
postgres	postgres	UTF8	ru_RU.UTF-8	ru_RU.UTF-8		libc	
template0	postgres	UTF8	ru_RU.UTF-8	ru_RU.UTF-8		libc	=c/postgres + postgres=CtC/postgres
template1	postgres	UTF8	ru_RU.UTF-8	ru_RU.UTF-8		libc	=c/postgres + postgres=CtC/postgres
template_rbac	pg_database_admin	UTF8	ru_RU.UTF-8	ru_RU.UTF-8		libc	

(5 rows)

Рисунок 6 — Пример вывода

- 3) Проверить доступность базы данных:
  - Локально на сервере БД:

```
$ psql -h 127.0.0.1 -d minica -U minica -W
```

- На сервере `minica`, если установлен `psql`:

```
$ psql -h root-ca.test.local -d minica -U minica -W
```

- 4) Скопировать файл дампа базы данных на сервер БД. В случае если сервер БД не локальный.

```
$ scp /app/itc/minica/deploy/sql/minica.up.sql administrator@pg1.test.local:/home/administrator/
```

- 5) Загрузить из прилагаемого SQL-файла необходимые таблицы (функции и типы) в базу данных:

```
$ cat db/pgsql/sql/minica.up.sql | sudo -i -u postgres psql minica
```

или:

```
$ cat db/pgsql/sql/minica.up.sql | psql -h localhost -d minica -U minica -W
```

или:

```
$ psql -h localhost -U postgres -d minica -f minica.up.sql
```

В процессе загрузки таблиц потребуется ввод пароля пользователя `postgres`.

- 6) После успешного завершения операций выгрузки установить дополнительные разрешения в базе данных при помощи `psql`:

```
$ sudo -u postgres psql minica
# GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO minica;
# GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO minica;
# GRANT ALL PRIVILEGES ON TABLE csr TO minica;
# GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO minica;
```

- 7) Создать в БД `minica` группы шаблонов для сертификатов (далее они потребуются на этапе настройки):

```
$ sudo -u postgres psql minica
INSERT INTO public.templ
```

```
(id, dtime, "name", descr, days, noaia, nocdp, short, "version", deleted, "key_type", key_size, exts,
bc, ku, eku, ski, aki, ext)
VALUES(1, '2025-01-20 15:41:08.726', 'tls', 'TLS клиент/сервер', 365, false, false, false, 1, false, 'RSA',
2048, "", "", 'critical,digitalSignature,keyEncipherment', 'serverAuth,clientAuth', 'hash', 'keyid,issuer',
");
INSERT INTO public.templ
(id, dtime, "name", descr, days, noaia, nocdp, short, "version", deleted, "key_type", key_size, exts,
bc, ku, eku, ski, aki, ext)
VALUES(3, '2025-01-20 15:41:08.750', 'tls-client', 'TLS клиент', 365, false, false, false, 1, false, "", 2048,
"", "", 'critical,digitalSignature,keyEncipherment', 'emailProtection,clientAuth', 'hash', 'keyid,issuer',
");
INSERT INTO public.templ
(id, dtime, "name", descr, days, noaia, nocdp, short, "version", deleted, "key_type", key_size, exts,
bc, ku, eku, ski, aki, ext)
VALUES(4, '2025-01-20 15:41:08.761', 'ocsp-server', 'OCSP сервер', 365, true, true, false, 1, false,
'RSA', 2048, "", 'CA:FALSE', 'critical,digitalSignature', 'OCSPSigning', 'hash', 'keyid,issuer',
'noCheck=ignored');
INSERT INTO public.templ
(id, dtime, "name", descr, days, noaia, nocdp, short, "version", deleted, "key_type", key_size, exts,
bc, ku, eku, ski, aki, ext)
VALUES(5, '2025-01-20 15:41:08.773', 'user', 'сертификат пользователя', 365, false, false, false, 1,
false, "", 0, "", "", 'critical,digitalSignature,keyEncipherment', 'critical,codeSigning', 'hash', 'keyid,issuer',
");
INSERT INTO public.templ
(id, dtime, "name", descr, days, noaia, nocdp, short, "version", deleted, "key_type", key_size, exts,
bc, ku, eku, ski, aki, ext)
VALUES(6, '2025-01-20 15:41:08.782', 'tls-short', 'TLS клиент/сервер', 31, false, false, true, 1, false, "",
0, "", "", 'critical,digitalSignature,keyEncipherment', 'serverAuth,clientAuth', 'hash', 'keyid,issuer',
");
INSERT INTO public.templ
(id, dtime, "name", descr, days, noaia, nocdp, short, "version", deleted, "key_type", key_size, exts,
bc, ku, eku, ski, aki, ext)
VALUES(2, '2025-01-20 15:41:08.740', 'tls-server', 'TLS сервер', 365, false, false, false, 1, false, 'RSA',
2048, "", "", 'critical,digitalSignature,keyEncipherment', 'serverAuth', 'hash', 'keyid,issuer',
");
INSERT INTO public.templ
(id, dtime, "name", descr, "policy", priv, days, noaia, nocdp, short, "version", deleted, "key_type",
key_size, exts, bc, ku, eku, ski, aki, ext)
VALUES(7, '2025-07-21 14:26:19.551', 'subca', 'Subordinate CA', "", "", 1460, false, false, false, 1, false,
"", 0, "", 'critical,CA:true,pathlen:0', 'critical,cRLSign,digitalSignature,keyCertSign', "", 'hash',
'keyid,issuer',
");
```

Результат правильного выполнения операции:

```
INSERT 0 1
```

```
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
```

Если в процессе установки возникла необходимость пересоздать БД в PostgreSQL, то необходимо выполнить следующую последовательность шагов:

- 1) Выполнить:

```
$ DROP USER minica;
# CREATE USER minica WITH PASSWORD 'P@ssw0rd';
# DROP DATABASE minica;
# CREATE DATABASE minica OWNER minica;
# GRANT ALL PRIVILEGES ON DATABASE minica TO minica;
```

- 2) Далее загрузить файл БД в таблицу и применить заново настройки привилегий:

```
# GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO minica;
# GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO minica;
# GRANT ALL PRIVILEGES ON TABLE csr TO minica;
# GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO minica;
```

- 3) Далее загрузить шаблоны групп для minica (см. пункт 7 ранее).

#### 4.13. Настройка сервиса systemd

Микросервис minica является консольным приложением не поддерживает процедуру разделения процесса (fork). Для запуска процесса в фоновом режиме рекомендуется создать юнит systemd:

- 1) Открыть файл сервиса для редактирования:

```
$ sudo vim /lib/systemd/system/minica.service
```

- 2) Заполнить следующим образом:

```
1: [Unit]
2: Description=ITC Mini CA service
3: After=syslog.target network.target
4: [Service]
```

```
5: # Переменные окружения (MINICA_CONF, MINICA_OPTIONS)
6: EnvironmentFile=/app/itc/minica/minica.env
7: # Рабочий каталог сервиса
8: WorkingDirectory=/app/itc/minica
9: # Исполняемый файл
10: ExecStart=/app/itc/minica/minica $MINICA_OPTIONS
11: # Имя пользователя и группы от имен которых будет запущен сервис
12: User=minica
13: Group=minica
14: # Разрешить работу сервиса на привилегированных портах (<1024)
15: AmbientCapabilities=CAP_NET_BIND_SERVICE
16: # Процесс не выполняет fork
17: Type=simple
18: # Код возврата при успехе
19: SuccessExitStatus=0
20: # После завершения процесса сервис завершается
21: RemainAfterExit=no
22: # Перезапускать сервис всегда
23: Restart=always
24: # Время ожидания перед перезапуском сервиса
25: RestartSec=10
26: # MiniCA для штатного завершения обрабатывает сигнал SIGINT (Ctrl+C)
27: KillSignal=SIGINT
28: # Перенаправление stdout/stderr
29: StandardOutput=file:/app/itc/minica/logs/minica.log
30: StandardError=file:/app/itc/minica/logs/minica.log
31: [Install]
32: WantedBy=default.target
```

При выполнении системного юнита systemd возможна работа микросервиса на привилегированном TCP порту.

3) Задать права на каталог `/app/` от ТУЗ minica:

```
$ sudo chown minica:minica -R /app/
```

4) После размещения необходимых файлов и настройки файлов конфигурации необходимо выполнить:

```
$ sudo systemctl daemon-reload
```

```
$ sudo systemctl enable minica.service  
$ sudo systemctl start minica.service  
$ sudo systemctl status minica.service
```

Сервис не запустится, так как нет сертификатов CA. Подробнее в разделе 6.

- 5) Проверить журнал микросервиса minica на наличие ошибок. Просмотр журнала осуществляется командой:

```
$ sudo journalctl -u minica -o cat
```

#### 4.14. Установка микросервиса minica без наличия прав суперпользователя

В ряде случаев необходимо развернуть микросервис minica без наличия прав суперпользователя на машине. В этом случае возможен запуск микросервиса как пользовательского сервиса systemd, для чего необходимо произвести следующие действия:

- 1) Перейти в учетную запись пользователя minica:

```
$ sudo -su minica
```

- 2) Включить режим `linger`, чтобы сервисы могли работать в фоновом режиме:

```
$ loginctl enable-linger $USER
```

- 3) Открыть файл с конфигурацией сервиса:

```
$ vim /home/$USER/.config/systemd/system/minica.service
```

- 4) Заполнить файл:

```
1: Description=ITC Mini CA service  
2: After=syslog.target network.target  
3: [Service]  
4: # Переменные окружения (MINICA_CONF, MINICA_OPTIONS)  
5: EnvironmentFile=/app/itc/minica/minica.env  
6: # Рабочий каталог сервиса  
7: WorkingDirectory=/app/itc/minica  
8: # Исполняемый файл
```

```
9: ExecStart=/app/itc/minica/minica $MINICA_OPTIONS
10: # Имя пользователя и группы от имен которых будет запущен сервис
11: # Разрешить работу сервиса на привилегированных портах (<1024)
12: AmbientCapabilities=CAP_NET_BIND_SERVICE
13: # Процесс не выполняет fork
14: Type=simple
15: # Код возврата при успехе
16: SuccessExitStatus=0
17: # После завершения процесса сервис завершается
18: RemainAfterExit=no
19: # Перезапускать сервис всегда
20: Restart=always
21: # Время ожидания перед перезапуском сервиса
22: RestartSec=10
23: # MiniCA для штатного завершения обрабатывает сигнал SIGINT (Ctrl+C)
24: KillSignal=SIGINT
25: # Перенаправление stdout/stderr
26: StandardOutput=file:/app/itc/minica/logs/minica.log
27: StandardError=file:/app/itc/minica/logs/minica.log
28: [Install]
29: WantedBy=default.target
```

- 5) Добавить переменные в окружения пользователя:  
а) Выполнить:

```
$ vim .bashrc
```

- б) Вставить:

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

- б) После размещения необходимых файлов и настройки файлов конфигурации сервис может быть запущен традиционным образом:

```
$ systemctl --user enable minica.service --now
$ systemctl --user status minica.service
$ systemctl --user start minica.service
```

Сервис не запустится, так как нет сертификатов CA. Подробнее в разделе 6.

Возможно, потребуется вручную экспортировать окружения командами `export` при каждой авторизации от пользователя.

Для просмотра журнала необходимо выполнить:

```
$ sudo -su minica  
$ journalctl --user -u minica -o cat
```

## 5. Установка и настройка выдающего MiniCA

### 5.1. Установка

Для установки необходимо авторизоваться на сервере `minica.test.local`, выполняющим роль выдающего ЦС. Используемые для примера в настоящей инструкции имена серверов и адреса приведены в разделе 1.1

Установить `minica`, как это описано в разделе 4. Детальная установка пакетов и создание ТУЗ описана в пункте 0 и 4.2

### 5.2. Настройка PostgreSQL.

Установка и настройка PostgreSQL описано в разделе 4.3

### 5.3. Каталоги установки MiniCA

Настройка каталогов описано в разделе 4.4

### 5.4. Параметры конфигурации minica

Источником конфигурации является конфигурационный файл `minica.yml`.

Для настройки конфигурации:

- 1) Открыть файл с настройками `minica`.

```
$ sudo vim /app/itc/minica/conf/minica.yml
```

- 2) Изменить строку подключение к БД в формате «url: postgres://имя пользователя:пароль@адрес:порт/бд»:

```
3: url: postgres://minica:1234578@pg1.test.local:5432/minica
```

- 3) Изменить строки:
  - `ca-pass` — установить пароль от ключа `root`;
  - `db-pass` — установить пароль к базе данных;
  - `allow-without-tls: false` — изменить на `true`, чтобы разрешить работу TLS.
- 4) Сохранить файл с внесенными настройками.

Настройки являются первичными для функционирования `minica` без использования TLS и защитой данных в базе.

## 5.5. Параметры конфигурации mclient

Источником конфигурации является конфигурационный файл `mclient.yml`.

- 1) Открыть файл с настройками mclient:

```
$ sudo vim /app/itc/minica/conf/mclient.yml
```

- 2) Найти строку, определяющую URL доступа к серверу MiniCA:

```
4: # URL доступа к серверу MiniCA  
5: server-url: http://localhost:8080
```

- 3) Заменить строку подключения:

```
6: # URL доступа к серверу MiniCA  
7: server-url: http://minica.test.local:8080
```

- 4) Сохранить файл после внесения изменений.

## 5.6. Создание БД и хранимых процедур

Выполнить настройку как показано в пункте 3.7, вместо локальной базы данных указывать `pg1.test.local`.

## 5.7. Настройка сервиса systemd

Выполнить настройку как показано в пункте 3.8

## 6. Создание сертификатов и настройка конфигурационных файлов MiniCA.

В ряде случаев Заказчик предоставляет сертификат выдающего ЦС, который используется в дальнейшем при настройке `minica`, конфигурационных файлов `minica.yml` и `mclient.yml`. Далее представлен пример полной цепочки выпуска сертификатов от корневого до выдающего ЦС.

### 6.1. Выпуск корневого сертификата ЦС

- 1) Перейти в директорию на корневом сервере minica:

```
$ cd /app/itc/minica
```

- 2) Для создания корневого ключа, выполнить команду:

```
$ sudo openssl genpkey -algorithm RSA -aes-256-cbc -out ca/ca.key -pkeyopt  
rsa_keygen_bits:4096
```

- 3) Задать пароль ключу:

- пароль должен иметь требование к безопасности;
- пароль нужно хранить от посторонних лиц в надежном месте.

- 4) Выполнить команду для создания самоподписанного корневого сертификата с помощью ключа:

```
$ sudo openssl req -x509 -new -key ca/ca.key -sha256 -days 3650 -out ca/ca.crt -subj  
"/C=RU/ST=Moscow/L=Moscow/O=test/OU=IT/CN=root ca cert"
```

Потребуется ввести:

- пароль от ключа;
- срок действия ключа (в днях);
- сведения о ключе, используя атрибуты из таблицы 16.

Таблица 16 — Атрибуты ключа

Атрибут	Расшифровка	Пример значения
C	Country (Страна)	RU
ST	State/Province (Регион/Область)	Moscow

Атрибут	Расшифровка	Пример значения
L	Locality (Город/Населённый пункт)	Moscow
O	Organization (Организация)	test
OU	Organizational Unit (Подразделение)	IT
CN	Common Name (Общее имя)	test.local
DC	Domain Component (компонент домена)	test
emailAddress	Email субъекта	123@test.local
UID	User ID (идентификатор пользователя)	1000
T	Title (должность)	root
SERIALNUMBER	Серийный номер сертификата	100000000
CN + DC	Часто используются вместе для LDAP-путей	CN=User1,DC=test,DC=local

- 5) Скопировать корневые сертификаты в директорию:

```
$ sudo scp /app/itc/minica/ca/ca.* /app/itc/minica/conf/
```

- 6) Убедиться, что корневые сертификаты находятся в каталогах:

- `ca/;`
- `conf/.`

- 7) Заново задать права на каталог `/app/` от ТУЗ `minica`:

```
$ sudo chown minica:minica -R /app/
```

#### Внимание:

Права доступа необходимо перенастраивать после каждого изменения в директории `minica`, поскольку инсталляция ПО может действовать под учетной записью администратора. Чтобы избежать подобных проблем, настройки рекомендуется выполнять от учетной записи `minica`.

- 8) Запустить службу `minica`:

```
$ sudo systemctl restart minica.service
$ sudo systemctl status minica.service
```

- 9) Проверить журнал микросервиса `minica` на наличие ошибок. Просмотр подробного журнала событий:

```
$ sudo tail -f /app/itc/minica/logs/minica.log
```

На данном моменте служба будет работать в режиме без TLS (legacy). Чтобы перевести работу `minica` в штатный режим необходимо:

- выпустить сертификат выдающего ЦС и сертификат TLS;
- включить защиту паролей;
- зашифровать ключ сертификата выдающего ЦС.

## 6.2. Выпуск сертификата выдающего ЦС:

- 1) Выполнить команду для создания ключа:

```
$ sudo openssl genrsa -aes256 -out /app/itc/minica/ca/subCA.key 4096
```

- 2) Задать пароль ключу:

- пароль должен соответствовать требованиям к безопасности;
- пароль нужно хранить от посторонних лиц в надежном месте.

- 3) Выполнить команду генерации запроса для выдающего сертификата:

```
$ sudo openssl req -config conf/openssl.cnf -new -sha256 -key /app/itc/minica/ca/subCA.key -out /app/itc/minica/ca/certs/subCA.csr
```

- Ввести пароль от ключа
- Задать атрибуты ключу.
- Произвести настройки параметров запроса в файле `conf/openssl.cnf`.

- 4) Выполнить команду создания сертификата выдающего ЦС с помощью запроса:

```
$ sudo /app/itc/minica/mclient ca -csr /app/itc/minica/ca/certs/subCA.csr -out /app/itc/minica/ca/subCA.crt -template subca -days 1825
```

- 5) Создать цепочку сертификатов, выполнить команду:

```
$ sudo bash -c 'cat ca/ca.crt ca/subCA.crt > ca/subCA-chain.crt'
```

- Скопировать выдающий сертификат с корневого сервера на выдающий;

```
$ sudo scp ca/subCA.crt minica@minica.test.local:/app/itc/minica/ca/ca.crt  
$ sudo scp ca/subCA.key minica@minica.test.local:/app/itc/minica/ca/ca.key  
$ sudo scp ca/subCA-chain.crt minica@minica.test.local:/app/itc/minica/ca/
```

- Авторизоваться на выдающем сервере и скопировать в папку `conf` сертификаты.

```
$ ssh minica@minica.test.local  
$ cd /app/itc/minica/
```

- Проверить права сертификатов командой ls -l.
- Задать права командой:

```
$ sudo chown minica:minica -R /app/
```

### 6.3. Создание сертификата для TLS подключения minica.

Выполнить на корневом или выдающем ЦС, в зависимости для кого выпускается ключ.

- 1) Создание TLS-ключа для корневого ЦС:

```
$ sudo openssl req -new -keyout /app/itc/minica/ca/certs/tls.key -out  
/app/itc/minica/ca/certs/tls.req -sha256 -nodes -subj "/CN=root-ca.test.local" -addext  
"subjectAltName = DNS:root-ca.test.local" -addext "extendedKeyUsage = serverAuth, clientAuth"
```

- 2) Создание TLS-ключа для выдающего ЦС:

```
$ sudo openssl req -new -keyout /app/itc/minica/ca/certs/tls.key -out  
/app/itc/minica/ca/certs/tls.req -sha256 -nodes -subj "/CN=minica.test.local" -addext  
"subjectAltName = DNS:minica.test.local" -addext "extendedKeyUsage = serverAuth, clientAuth"
```

- 3) Создание TLS-сертификата с помощью запроса (выполнить команду) на обоих ЦС:

```
$ sudo /app/itc/minica/mclient ca -csr /app/itc/minica/ca/certs/tls.req -out  
/app/itc/minica/ca/certs/tls.crt -template tls
```

- 4) Создание TLS-ключа для PostgreSQL с помощью запроса для корневого ЦС:

```
$ sudo openssl req -new -keyout /app/itc/minica/ca/certs/postgres.key -out  
/app/itc/minica/ca/certs/postgres.req -sha256 -nodes -subj "/CN=root-ca.test.local" -addext  
"subjectAltName = DNS:root-ca.test.local" -addext "extendedKeyUsage = serverAuth, clientAuth"
```

- 5) Создание TLS-ключа для PostgreSQL с помощью запроса для выдающего ЦС

```
$ sudo openssl req -new -keyout /app/itc/minica/ca/certs/postgres.key -out  
/app/itc/minica/ca/certs/postgres.req -sha256 -nodes -subj "/CN=pg1.test.local" -addext  
"subjectAltName = DNS:pg1.test.local" -addext "extendedKeyUsage = serverAuth, clientAuth"
```

- 6) Создание TLS- сертификата для postgresSQL с помощью запроса (выполнить команду) на обоих ЦС:

```
$ sudo /app/itc/minica/mclient ca -csr app/itc/minica/ca/certs/postgres.req -out  
/app/itc/minica/ca/certs/postgres.crt -template tls
```

#### 6.4. Перенастройка службы minica на выпущенные сертификаты.

После выпуска сертификатов требуется выполнить на обоих ЦС:

- настроить службу по TLS;
  - указать сертификат TLS;
  - перевыпустить закрытый и открытый ключ minica;
- 1) Перейти в директорию minica на обоих ЦС:

```
$ cd /app/itc/minica/
```

- 2) Скопировать TLS--сертификат для minica в папку с конфигурацией.

```
$ sudo scp ca/certs/tls.* conf/
```

- 3) Открыть файл конфигурации `minica`:

```
$ sudo vim /app/itc/minica/conf/minica.yml
```

- a) Изменить строки:

```
1:  tls:  
2:    enable: false # включить TLS
```

на

```
1:  tls:  
2:    enable: true # включить TLS
```

- б) Проверить правильность имен и правильный путь к TLS-сертификатам и ключам

```
1: key: conf/tls.key # файл tls
2: cert: conf/tls.crt # файл tls
3: ca-cert: conf/ca.crt # файл с сертификатом CA
```

в) Проверить порт HTTPS

```
1: https-port: 8443 # HTTPS port (если TLS вкл.)
```

Порт по умолчанию установлен по умолчанию. Если на сервере minica разворачивается веб-панель, требуется сменить порт.

г) Сменить строку:

```
1: allow-without-tls: true
```

на

```
1: allow-without-tls: false
```

4) Сохранить внесенные изменения и закрыть файл.

5) Открыть файл конфигурации `mclient.yml`.

```
$ sudo vim /app/itc/minica/conf/mclient.yml
```

а) Изменить строку:

```
1: server-url: http://localhost:8080
```

На

```
2: server-url: https://root-ca.test.local:8443
```

или

```
3: server-url: https://minica.test.local:8443
```

б) Изменить строки:

```
1: tls:
2:   enable: false # включить TLS
```

```
3: verify: false # включить mTLS
4: key: conf/mtls.key # файл с ключевой парой сервера
5: cert: conf/mtls.crt # файл с сертификатом сервера
6: ca-cert: conf/ca.crt # файл с сертификатом CA
```

на

```
1: tls:
2: enable: true # включить TLS
3: verify: false # включить mTLS
4: key: conf/tls.key # файл с ключевой парой сервера
5: cert: conf/tls.crt # файл с сертификатом сервера
6: ca-cert: conf/ca.crt # файл с сертификатом CA
```

- 6) Сохранить внесённые данные и закрыть файл.
- 7) Настроить службу `postgresql` на TLS--подключение.
- 8) Обновить права доступа на каталоги и перезапустить службу:

```
$ sudo chown minica:minica -R /app/
$ sudo systemctl restart minica.service
```

## 6.5. Обновление закрытой и открытой пары ключей

Выполнить на обоих ЦС

- 1) Перейти в директорию `minica`:

```
$ cd /app/itc/minica
```

- 2) Открыть файл конфигурации `config-jwt.env`:

```
$ vim config-jwt.env
```

- 3) Установить пароль:

```
1: JWT_KEY_PASS="12345678"
```

- 4) Сохранить внесённые данные и закрыть файл конфигурации.
- 5) Запустить скрипты настройки:

```
$ cd /app/itc/minica/script/  
$ sudo ./make-jwt-key.sh  
$ sudo ./make-mclient-key.sh  
$ sudo ./make-mclient-jwt.sh
```

Внимание: В данном примере использованы скрипты для автоматического перевыпуска закрытого и открытого ключа minica и создание jwt-токена.

## 6.6. Перенастройка службы postgresql на выпущенные сертификаты.

- 1) Скопировать TLS-сертификат PostgreSQL в директорию конфигурации БД локально.

```
$ sudo scp ca/certs/postgres.* /etc/postgresql/15/main/  
$ sudo scp ca/ca.crt /etc/postgresql/15/main/
```

- 2) Скопировать TLS-сертификат PostgreSQL в директорию конфигурации БД.

```
$ sudo scp ca/certs/postgres.*  
/etc/postgresql/15/main/administrator@pg1.test.local:/home/administrator/  
$ sudo scp ca/ca.crt administrator@pg1.test.local:/home/administrator/
```

- 3) Переместить сертификат и ключ на сервер в директорию конфигурации БД (для удалённой БД).

```
$ sudo mv postgres.* /etc/postgresql/15/main/  
$ sudo mv ca.crt /etc/postgresql/15/main/
```

- 4) Выдать права на сертификат и ключи от пользователя postgres(на обоих БД):

```
$ sudo chown postgres:postgres /etc/postgresql/15/main/postgres.*
```

- 5) Открыть файл конфигурации postgresql.conf(на обоих БД):

```
$ sudo vim /etc/postgresql/15/main/postgresql.conf
```

- 6) Внести данные:

```
1: ssl = on
2: ssl_ca_file = '/etc/postgresql/15/main/ca.crt'
3: ssl_cert_file = '/etc/postgresql/15/main/postgres.crt'
4: ssl_key_file = '/etc/postgresql/15/main/postgres.key'
5: ssl_ciphers = 'HIGH:MEDIUM:+3DES:!aNULL' # allowed SSL ciphers
```

- 7) Сохранить внесенные данные и закрыть файл конфигурации.
- 8) Перезапустить службу postgresql (на обоих БД):

```
$ sudo systemctl restart postgresql
```

## 6.7. Перенастройка файла конфигурации minica.yml для выполнения требования безопасности.

Для обеспечения требований безопасности к ключу и паролям в конфигурационных файлах minica используется приложение Maskpass, поставляемое совместно с дистрибутивом minica.

Все действия выполняются как на коревом сервере, так и выдающем.

Для перенастройки необходимо выполнить следующие действия:

- 1) Отключить запись команд в историю терминала:

```
$ set +H
```

- 2) Перейти в директорию minica

```
$ cd /app/itc/minica/
```

- 3) Выполнить команду для шифрования приватного ключа:

```
$ sudo openssl pkcs8 -topk8 -v2 aes-256-cbc -passin pass:12345678 -in ca/ca.key -out  
ca.key.encrypted
```

Ввести дважды пароль, который был ранее задан в командной строке (т.е в примере 12345678).

- 4) Скопировать зашифрованный файл `ca.key` в каталоги:

```
$ sudo scp ca.key.encrypted ca/ca.key
$ sudo scp ca.key.encrypted conf/ca.key
```

- 5) При помощи Maskpass создать зашифрованную комбинацию пароля командой:

```
$ ./maskpass -p 12345678
```

В результате выполнения команды будет получен вывод:

```
yxtde0FYcvVTLwPAqQUvasXCQL3xGqlgpwbsVA5Vx2Xz0wMs
```

- 6) Открыть файл конфигурации

```
$ sudo vim /app/itc/minica/conf/minica.yml
```

- 7) Изменить строку, место пароля поставить полученную зашифрованную комбинацию пароля:

```
1: ca-pass: yxtde0FYcvVTLwPAqQUvasXCQL3xGqlgpwbsVA5Vx2Xz0wMs #12345678
```

- 8) Сохранить внесенные данные и закрыть файл конфигурации.  
9) Зашифровать пароль для входа в БД postgresql ключевой фразой:

```
$ sudo ./maskpass -p 12345678 -k password-fraza
```

В результате выполнения команды будет получен вывод:

```
eTsEl_4QwtmRAqv3nm-KAn1aFr9QzYIJQccNxliHwbEvs089
```

- 10) Открыть файл конфигурации minica.yml:

```
$ sudo vim /app/itc/minica/conf/minica.yml
```

- 11) Изменить строку — вместо пароля поставить зашифрованную комбинацию пароля.

```
1: url: postgres://minica:eTsEl\_4QwtmRAqv3nm-KAn1aFr9QzYIJQccNxliHwbEvs089@pg1.test.local:5432/minica
```

- 12) Сохранить внесенные данные и закрыть файл конфигурации.  
13) Применить Maskpass для шифрования парольную фразы для БД:

```
$ sudo ./maskpass -p password-fraza
```

В результате выполнения команды будет получен вывод:

```
utna4dVpB_BC5XgjkNL4EonxvtG6Bj3y8-NbecW_36whUD3zDO7QIUoj
```

14) Открыть файл конфигурации `minica.yml`:

```
$ sudo vim /app/itc/minica/conf/minica.yml
```

15) Изменить строку — вместо пароля поставить зашифрованную комбинацию пароля:

```
1: db-pass: utna4dVpB_BC5XgjkNL4EonxvtG6Bj3y8-NbecW_36whUD3zDO7QIUoj # 1234
```

16) Перезапустить службу `minica`:

```
$ sudo systemctl restart minica
```

## 7. Установка и настройка микросервиса minica-web-panel

Дистрибутив MiniCA-web-panel поставляется в виде DEB пакета для операционных систем архитектуры amd64 (x86\_64) основанных на dpkg (Debian, Ubuntu, Astra Linux).

Поставка в виде RPM-пакета, архива tar.gz и/или самораспаковывающего архива SFX может быть выполнена по требованию Заказчика.

Файл с именем пакета имеет наименование X.Y.Z-hash\_amd64.deb, где:

- X.Y.Z — старший номер версии, младший номер версии, номер сборки соответственно;
- hash — хэш-идентификатор комита в репозитории исходных текстов Git;
- amd64 — целевая архитектура.

Пакеты содержат:

- исполняемые файлы сервиса — minica-cp-web и minica-cp-service;
- примеры файлов конфигурации.

Установка пакета производится через терминал путём ввода команды:

```
sudo dpkg -i к/файлу/minica-cp-web-2.3.0-hash_amd64.deb к/файлу/minica-cp-service-2.3.0-hash_amd64.deb
```

где вместо «к/файлу/ minica-X.Y.Z-hash\_amd64.deb» требуется указать фактический путь до файла пакета.

Установка MiniCA из самораспаковывающегося архива осуществляется следующим образом:

- 1) Предоставить исполняемые права архиву:

```
sudo chmod 755 minica-cp-web-2.3.0-rc1.run minica-cp-service-2.3.0-rc1.run
```

- 2) Запустить архивы для распаковки содержимого:

```
sudo ./minica-cp-web-2.3.0-rc1.run  
sudo ./minica-cp-service-2.3.0-rc1.run
```

Для установки из ZIP-архива:

- 1) Установить права на запуск извлечения:

```
sudo chmod 755 unzip
```

- 2) Распаковать содержимое архивов:

```
sudo unzip minica-cp-web-2.3.0-rc1.zip  
sudo unzip minica-cp-service-2.3.0-rc1.zip
```

Приложения `minica-cp-web` и `minica-cp-service` требуют наличия пакета Nginx и группы пакетов DotNet6. Чтобы убедиться, что требуемые компоненты установлены, рекомендуется проверить их версию в терминале операционной системы.

Пример проверки:

- 1) Проверка наличия и версии пакета Nginx на Debian-подобных системах:

```
$ dpkg -l | grep nginx
```

# Результат примера:

```
1: ii nginx          1.26.0-1      amd64  
2: ii nginx-common    1.26.0-1      all
```

- 2) Проверка наличия и версии пакета Nginx на Red Hat-подобных системах:

```
$ rpm -qa | grep nginx
```

Результат примера:

```
3: nginx          1.26.0-1
```

- 3) Проверка наличия пакета DotNet6 на Debian-подобных системах:

```
$ dpkg -l | grep dotnet6
```

Если вывод пуст — значит пакет не установлен.

При отсутствии установленных пакетов, их необходимо установить с помощью менеджера пакетов операционной системы:

```
$ sudo apt update
```

```
$ sudo apt install nginx dotnet6 -y
```

## 7.1. Каталоги установки службы

При установке пакета автоматически добавляется пользователь `minica` и ему назначается домашний каталог `/app/itc/`.

При установке из самораспаковывающего архива или просто архива:

- 1) Создать директорию `/app/itc/itc/`:

```
$ sudo mkdir -p /app/itc/itc/config/  
$ sudo mkdir -p /app/itc/minica-spa-service  
$ sudo mkdir -p /app/itc/minica-SP-service  
$ sudo mkdir -p /app/itc/itc/logs  
$ sudo mkdir -p /app/itc/nginx/tls  
$ sudo mkdir -p /app/itc/itc/certs
```

- 2) Переместить файлы из самораспаковывающего архива в директорию:

```
$ sudo mv MiniCA CP 2.4.0-rc6/* /app/itc/minica-SP-service/  
$ sudo mv minica-spa-2.4.0-rc6/* /app/itc/minica-spa-service
```

- 3) Создать директорию для журналирования:

```
$ sudo mkdir -p /app/itc/mocsp/logs/
```

- 4) Добавить системного пользователя `minica` (ТУЗ).

```
$ sudo adduser minica
```

- 5) Изменить пароль (если потребуется)

```
$ sudo passwd minica
```

Файл `ITC.Api.MiniCA.ControlPanel` является исполняемым файлом приложения `minica-SP-service` и обычно размещается в одном из каталогов: `/app/itc/minica-SP-service/ITC.Api.MiniCA.ControlPanel`, для его работоспособности требуется установка зависимостей пакета `dotnet6`. В системе должен предварительно настроен быть репозиторий для установки пакетов.

Файл `itc_api_minica_controlPanel_config.json` — конфигурационный файл микросервиса в формате JSON. Рекомендуется размещать в каталоге `/app/itc/itc/config/` сервера.

В конфигурационном файле хранится важная информация, включая реквизиты доступа к базе данных, поэтому доступ к нему (включая возможность чтения) для широкого круга пользователей должен быть ограничен.

Файл `minica.spa.service` — файл `systemd` находится в одном из двух мест размещения:

- При ручном развертывании: `/etc/systemd/system/`;
- При автоматизированной установке пакета: `/lib/systemd/system/`.

Файл `itc_svc_env.conf` содержит переменные окружения для ITCRoot и используется в `*.service` файлах `systemd`. Он также служит для указания пути к рабочему каталогу.

## 7.2. Параметры конфигурации minica-web-panel

Источником конфигурации является конфигурационный файл в формате JSON — `itc_api_minica_controlPanel_config.json`.

Пример конфигурационного файла:

```
1: {{
2:   "ConnectionStrings": {
3:     "MiniCA":
4:       "Host=pg1.test.local;Port=5432;Database=minica;Username=minica;Password=12345678"
5:   },
6:   "OpenId": {
7:     "authority": "https://keycloak.test.local/realms/master",
8:     "ignoreCertificateErrors": true,
9:     "jwtRoleClaim": "itc_roles"
10:  },
11: "BaseUri": "https://minica.test ",
12: "Mailing": {
13:   "TemplatesPath": "/app/itc/itc/templates",
14:   "TemplateBody": "itc.mail.mca.body.html",
15:   "CertificateExpirationDays": 500,
16:   "Boxes": [
17:     {
18:       "Key": "itc.mail.mca",
19:       "Host": "mail.test.local",
```

```
19:     "Port": "465",
20:     "Email": "jari@ test.local",
21:     "SecureSocket": "StartTls",
22:     "UserName": "test\\Jari",
23:     "PasswordKey": "ITCMailOutboxPasswordKey_jari"
24:   }
25: },
26: "Templates": [
27:   {
28:     "Key": "itc.mca.cpa.certificate.enrolled",
29:     "Box": "itc.mail.mca",
30:     "Path": "itc.mail.mca.certificate.enrolled.html",
31:     "Subject": "Выпуск сертификата."
32:   },
33:   {
34:     "Key": "itc.mca.cpa.certificate.revoked",
35:     "Box": "itc.mail.mca",
36:     "Path": "itc.mail.mca.certificate.revoked.html",
37:     "Subject": "Отзыв сертификата."
38:   },
39:   {
40:     "Key": "itc.mca.cpa.certificate.expiring",
41:     "Box": "itc.mail.mca",
42:     "Path": "itc.mail.mca.certificate.expiring.html",
43:     "Subject": "Истекающие сертификаты."
44:   }
45: ]
46: },
47: "MiniCaConfigs": {
48:   "MiniCA": {
49:     "Url": "https://minica.test.local:8443",
50:     "JwtInFile": {
51:       "FilePath": "/app/itc/itc/certs/panel.jwt"
52:     },
53:     "CertificatesInFile": {
54:       "ItcAuthorizeCert": {
55:         "CertPath": "/app/itc/itc/certs/panel.crt",
56:         "KeyPath": "/app/itc/itc/certs/panel.key"
57:       },
```

```
58:     "TlsCaAuthorizeCert": {  
59:         "CertPath": "/app/itc/itc/certs/ca.crt"  
60:     }  
61: }  
62: }  
63: }  
64: }
```

Для того, чтобы создать конфигурационный файл `itc_api_minica_controlPanel_config.json`, необходимо:

- 1) Выполнить:

```
$ sudo vim /app/itc/itc/config/itc_api_minica_controlPanel_config.json
```

- 2) Внести данные, сохранить и закрыть файл.

### 7.3. Параметры конфигурации файла `client_env.js`

Источником конфигурации является конфигурационный файл в формате JS — `client_env.js`.

Файл расположен в директории `/app/itc/minica-spa-service/assets/env/client_env.js` и служит для указания подключения к Keycloak системе для авторизации доменных пользователей в системе ЦУГИ.

Пример конфигурационного файла:

```
1: (function () {  
2:     window.itcenv = {  
3:         oid: {  
4:             issuer: 'https://keycloak.test.local/realms/master', # Указать keycloak  
5:             clientId: 'minica-client', # Указать клиент как создано в keycloak  
6:         },  
7:     };  
8: })();
```

Для настройки подключения:

- 1) Указать Keycloak в двух конфигурационных файлах:

```
$ sudo vim /app/itc/minica-spa-service/assets/env/client_env.js
```

```
$ sudo vim /app/itc/itc/env/client_env.js
```

- 2) Создать файл в директории:

```
$ sudo vim /app/itc/itc_svc_env.conf  
ITCRoot="/app/itc"
```

- 3) Выдать права исполняемому файлу:

```
$ sudo chmod 755 /app/itc/minica-SP-service/ITC.Api.MiniCA.ControlPanel
```

## 7.4. Настройка сервиса systemd

Микросервис `minica` является консольным приложением, вызов `fork` не применяется. Для запуска процесса в фоновом режиме рекомендуется создание юнита `systemd`. Ниже приведен пример файла `minica.spa.service` для реализации системного сервиса:

- 1) Создать конфигурационный файл сервиса:

```
$ sudo vim /lib/systemd/system/minica.spa.service
```

- 2) Содержание конфигурационного файла:

```
1: [Unit]  
2: Description=minica-SP-service  
3: [Service]  
4: WorkingDirectory=/app/itc/minica-SP-service  
5: ExecStart=/app/itc/minica-SP-service/ITC.Api.MiniCA.ControlPanel --  
   urls="http://0.0.0.0:8452"  
6: Restart=always  
7: RestartSec=10  
8: KillSignal=SIGINT  
9: SyslogIdentifier=minica-SP-service  
10: EnvironmentFile=/app/itc/itc_svc_env.conf  
11: User=minica  
12: Group=minica  
13: [Install]  
14: WantedBy=default.target
```

- 3) Сохранить данные в конфигурационном файле и закрыть.

- 4) Перезагрузить конфигурацию системных служб:

```
$ sudo systemctl daemon-reload
```

- 5) Включить сервис в автозагрузку:

```
$ sudo systemctl enable minica.spa.service
```

- 6) Запустить сервис:

```
$ sudo systemctl start minica.spa.service
```

Чтобы проверить статус службы и выявить возможные ошибки, используются следующие команды:

- Проверка статуса службы:

```
$ sudo systemctl status minica.spa.service
```

- Просмотр ошибок из журнала:

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

- Просмотр файла лога:

```
$ sudo tail -f /app/itc/itc/logs/itc_api_minica_controlPanel/file.log
```

Сервис не запустится, так как отсутствуют сертификаты. Подробнее о создании сертификатов в пункте 7.6

## 7.5. Установка микросервиса minica-web-panel без наличия прав суперпользователя

Пример создания микросервиса без наличия прав суперпользователя, приведен в пункте 4.14 данной инструкции.

## 7.6. Создание сертификатов для minica-web-panel.

- 1) Перейти на сервер ЦС (minica) и выпустить сертификаты для minica-web-panel.
- 2) Создать TLS-ключ для minica-web-panel с помощью запроса командой:

```
$ sudo openssl req -new -keyout /app/itc/minica/ca/certs/panel.key -out  
/app/itc/minica/ca/certs/panel.req -sha256 -nodes -subj "/CN=minica-web.test.local" -addext  
"subjectAltName = DNS:minica-web.test.local" -addext "extendedKeyUsage = serverAuth, clientAuth"
```

- 3) Создать TLS-сертификата для web-panel с помощью запроса командой:

```
$ sudo /app/itc/minica/mclient ca -csr /app/itc/minica/ca/certs/panel.req -out  
/app/itc/minica/ca/certs/panel.crt -template tls
```

- 4) Скопировать сертификаты для minica-web-panel в директорию:

```
$ sudo scp /app/itc/minica/ca/certs/panel.* /app/itc/minica/conf/
```

- 5) Создать `.env` файл:

```
$ sudo vim panel.env  
MCLIENT="/app/itc/minica/mclient"
```

- 6) Создать скрипт запроса:

```
$ sudo touch jwt-panel.sh  
$ sudo vim jwt-panel.sh
```

- 7) Внести в скрипт данные:

```
1: #!/bin/bash  
2: . "panel.env"  
3: export MCLIENT_CONF  
4: RQ="mclient"  
5: AUD=""  
6: NAME="panel"  
7: ROLE="admin"  
8: PERM=""  
9: PRIV=""  
10:  
11: NBF=`date --rfc-3339=seconds | sed 's/ /T/'`  
12: EXP=""  
13:  
14: PUB_KEY="/app/itc/minica/conf/panel.crt"  
15:
```

```
16: [ "$RQ" ] && RQ="-requester $RQ"
17: [ "$AUD" ] && AUD="-jwt-aud $AUD"
18: [ "$NAME" ] && NAME="-jwt-name $NAME"
19: [ "$ROLE" ] && ROLE="-jwt-role $ROLE"
20: [ "$PERM" ] && PERM="-jwt-perm $PERM"
21: [ "$PRIV" ] && PRIV="-jwt-priv $PRIV"
22: [ "$NBF" ] && NBF="-jwt-nbf $NBF"
23: [ "$EXP" ] && EXP="-jwt-exp $NBF"
24:
25: JWT="/app/itc/minica/conf/panel.jwt"
26:
27: "$MCLIENT" mkjwt $RQ $AUD $NAME $ROLE $PERM $PRIV $NBF $EXP -jwt-key
"$PUB_KEY" -out "$JWT"
```

8) Выдать права на исполнение:

```
$ sudo chmod 755 jwt-panel.sh
```

9) Запустить скрипт:

```
$ sudo ./jwt-panel.sh
```

10) Полученный токен и сертификаты скопировать на сервер minica-web-panel:

```
$ sudo scp /app/itc/minica/conf/panel.* administrator@minica-web.test.local:/home/administrator/
$ sudo scp /app/itc/minica/conf/ca.crt administrator@minica-web.test.local:/home/administrator/
```

11) На сервере minica-web (panel) переместить файлы в папку `/certs`:

```
$ sudo mv panel.* /app/itc/itc/certs/
$ sudo mv ca.crt /app/itc/itc/certs/
```

12) Изменить владельца файлов:

```
$ sudo chown minica:minica -R /app/
```

13) Запустить сервис:

```
$ sudo systemctl start minica.spa.service
```

Чтобы проверить статус сервиса и выявить возможные ошибки, используются следующие команды:

- Проверка статуса:

```
$ sudo systemctl status minica.spa.service
```

- Просмотр ошибок из журнала:

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

- Просмотр файла лога:

```
$ sudo tail -f /app/itc/itc/logs/itc_api_minica_controlPanel/file.log
```

## Установка и настройка сервиса Nginx.

Программа minica-web-panel зависит от заранее установленного пакета Nginx, который обеспечивает взаимодействие между клиентом и панелью управления.

Для настройки Nginx для взаимодействия в minica-web-panel:

- 1) Создать директорию для хранения TLS-сертификатов:

```
$ sudo mkdir -p /app/itc/nginx/tls/
```

- 2) Создать конфигурационный файл Nginx:

```
$ sudo vim /app/itc/nginx/nginx.conf
```

- 3) Заполнить данные по примеру:

```
1: server {
2:     listen 80;
3:     listen [::]:80;
4:
5:     server_name minica-web.test.local;
6:
7:     root /app/inetpub;
8:
9:     # Явно указываем индексный файл (если он нужен)
```

```
10:   index root-test.crl; # Замените на реальное имя файла
11:
12:   # Добавляем MIME-тип для .crl
13:   types {
14:       application/pkix-crl crl;
15:   }
16:
17:   location / {
18:       # Разрешаем листинг директорий (если нужно)
19:       autoindex on; # Опционально
20:
21:       # Ищем сначала файл, затем директорию
22:       try_files $uri $uri/ =404;
23:   }
24: }
25: server {
26:     listen 8443 ssl;
27:     listen 443 default_server ssl;
28:     ssl_certificate /app/itc/nginx/tls/tls.crt;
29:     ssl_certificate_key /app/itc/nginx/tls/tls.key;
30:
31:
32:     location / {
33:         root /app/itc/minica-spa-service;
34:         index index.html;
35:         try_files $uri $uri/ /index.html?$args;
36:     }
37:
38:     location /viewtemplates {
39:         alias /app/itc/itc.templates;
40:     }
41:
42:     location =/favicon.ico {
43:         alias /app/itc/minica-spa-service/assets/env/favicon.ico;
44:     }
45:
46:     location /assets/env {
47:         alias /app/itc/minica-spa-service/assets/env/;
48:     }
```

```
49:
50:
51:     location /api/minica {
52:         proxy_pass http://127.0.0.1:8452/api;
53:         proxy_http_version 1.1;
54:         proxy_cache_bypass $http_upgrade;
55:         proxy_set_header Upgrade $http_upgrade;
56:         proxy_set_header Connection keep-alive;
57:         proxy_set_header Host $host;
58:         proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
59:         proxy_set_header X-Forwarded-Proto $scheme;
60:     }
61: }
```

Настройка конфигурационного файла Nginx производится одновременно для точек распространения.

- 4) Открыть корневой конфигурационный файл Nginx:

```
$ sudo vim /etc/nginx/nginx.conf
```

- 5) Заполнить данные по примеру:

```
1: user www-data;
2: worker_processes auto;
3: worker_cpu_affinity auto;
4: pid /run/nginx.pid;
5: error_log /var/log/nginx/error.log;
6: include /etc/nginx/modules-enabled/*.conf;
7:
8: events {
9:     worker_connections 768;
10: }
11: http {
12:
13:     ##
14:     # Basic Settings
15:     sendfile on;
16:     tcp_nopush on;
17:     types_hash_max_size 2048;
18:     include /etc/nginx/mime.types;
```

```
19:     default_type application/octet-stream;
20:
21:     ##
22:     # SSL Settings
23:     ##
24:
25:     ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Dropping SSLv3, ref: POODLE
26:     ssl_prefer_server_ciphers on;
27:     access_log /var/log/nginx/access.log;
28:     gzip on;
29:
30:     include /etc/nginx/conf.d/*.conf;
31:     include /app/itc/nginx/nginx.conf; # Указать путь на конфигурационный файл
    nginx
32: }
```

6) Скопировать сертификаты для службы Nginx:

```
$ sudo scp /app/itc/itc/certs/panel.key /app/itc/nginx/tls/tls.key
$ sudo scp /app/itc/itc/certs/panel.crt /app/itc/nginx/tls/tls.crt
```

Сертификаты подойдут от minica-web panel, так как сертификаты выполняют одну и ту же функцию, с одним и тем-же DNS.

7) Задать повторно на папку права от пользователя minica:

```
$ sudo chown minica:minica -R /app/
```

8) Далее перезапустить службу Nginx и проверить ее статус:

```
$ sudo systemctl restart nginx
$ sudo systemctl status nginx
```

В случае ошибок посмотреть журнал и файлы логов:

```
$ sudo journalctl -u nginx.service
$ sudo tail -f /var/log/nginx/access.log
$ sudo tail -f /var/log/nginx/error.log
```

## 7.7. Настройка клиента в Keycloak.

Для функционирования веб-панели MiniCA требуется наличие заранее настроенного сервера Keycloak, на котором выполнена интеграция федерации учетных записей. Сервер Keycloak служит в ЦУГИ для аутентификации и авторизации, предоставляя возможность единого входа (SSO) и управления доступом к приложениям и сервисам. Это упрощает реализацию безопасности, отделяя управление пользователями от логики приложения.

- 1) Авторизоваться в Keycloak и перейти в вкладку «Client».
- 2) Нажать кнопку «Create Client»:

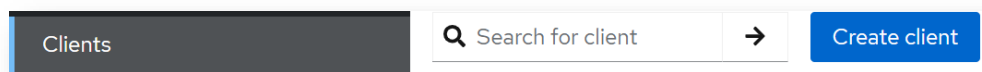


Рисунок 7 — Кнопка «Create Client»:

- 3) Заполнить поля:
  - ID Client;
  - Name;
  - Description.

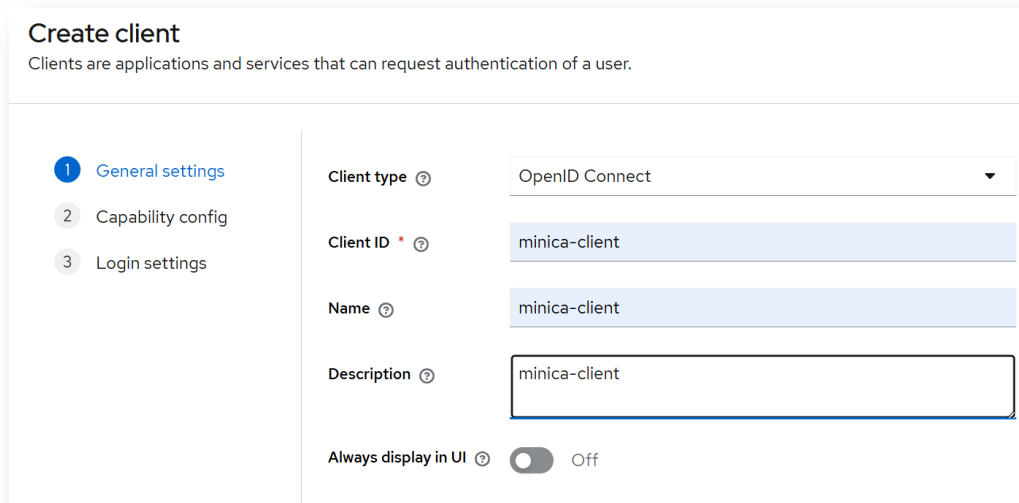
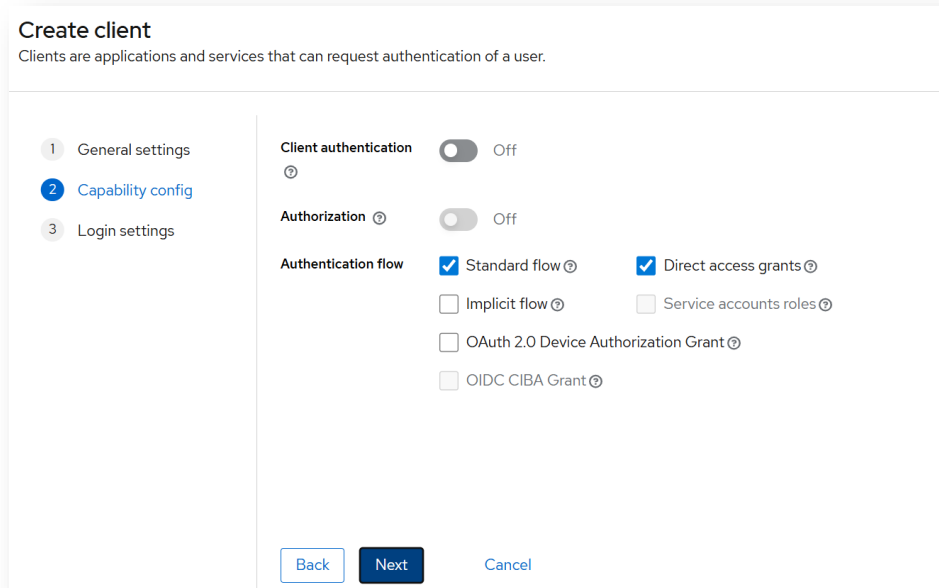


Рисунок 8 — Создание клиента в Keycloak

- 4) Нажать «Next» и перейти на следующую вкладку «Capability config».



**Create client**  
Clients are applications and services that can request authentication of a user.

- 1 General settings
- 2 Capability config**
- 3 Login settings

**Client authentication** ☐ Off

**Authorization** ☐ Off

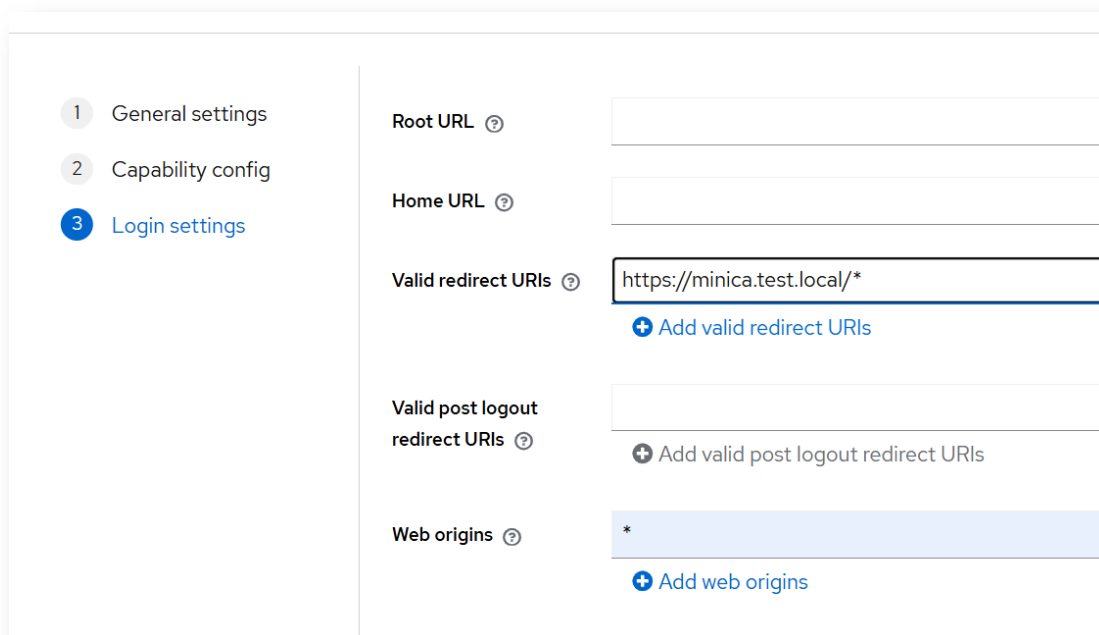
**Authentication flow**

- ☒ Standard flow
- ☒ Direct access grants
- ☐ Implicit flow
- ☐ Service accounts roles
- ☐ OAuth 2.0 Device Authorization Grant
- ☐ OIDC CIBA Grant

[Back](#) [Next](#) [Cancel](#)

Рисунок 9 — Вкладка «Capability config»

- 5) Оставить вкладку без изменения и нажать «Next»:



- 1 General settings
- 2 Capability config
- 3 Login settings**

**Root URL**

**Home URL**

**Valid redirect URIs** [Add valid redirect URIs](#)

**Valid post logout redirect URIs** [Add valid post logout redirect URIs](#)

**Web origins** [Add web origins](#)

Рисунок 10 — Вкладка «Login settings»

- 6) Оставить вкладку без изменений и нажать «Save», чтобы создать клиент.
- 7) Перейти в созданный клиент, выбрать вкладку «Roles». Данные роли отвечают за разграничения доступа к веб-панели.
- 8) Добавить роли:

Таблица 17 — Роли для добавления

Роль	Значение
rleMiniCACertReq-w	Запрос сертификата.
rleMiniCACertRev-w	Отзыв сертификата.
rleMiniCACertUnRev-w	Возврат сертификата.
rleMiniCATempl-w	Редактирование шаблона.
rleMiniCAConfig-w	Редактирование конфигурации ЦС.
rleMiniCACRL-w	Изменение периода публикации CRL, включение DeltaCRL, ручная публикация по кнопке.
rleMiniCACertList-r	Просмотр списка и карточек сертификатов.
rleMiniCATemplList-r	Просмотр списка и карточек шаблонов.
rleMiniCAReqList-r	Просмотр списка запросов.
rleMiniCAEventList-r	Просмотр журнала событий.
rleMiniCACRL-r	Просмотр карточки конфигурации CRL.
rleMiniCAConfig-r	Просмотр конфигурации.
rleMiniCAReqApprove-w	Подтверждение запроса

Данные роли отвечают за разграничения доступа к веб-панели.

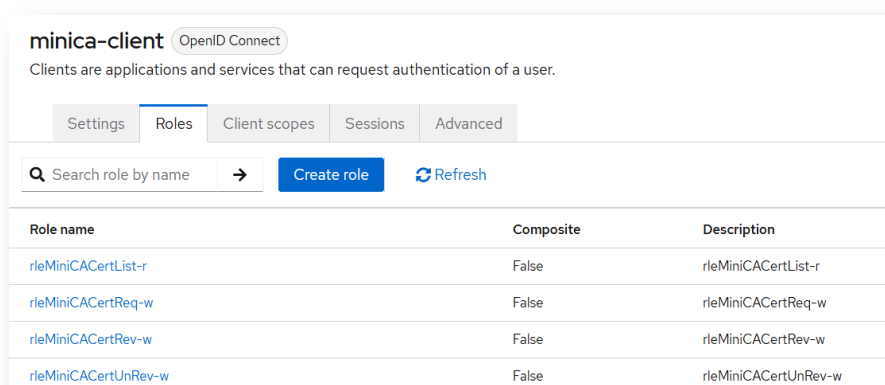
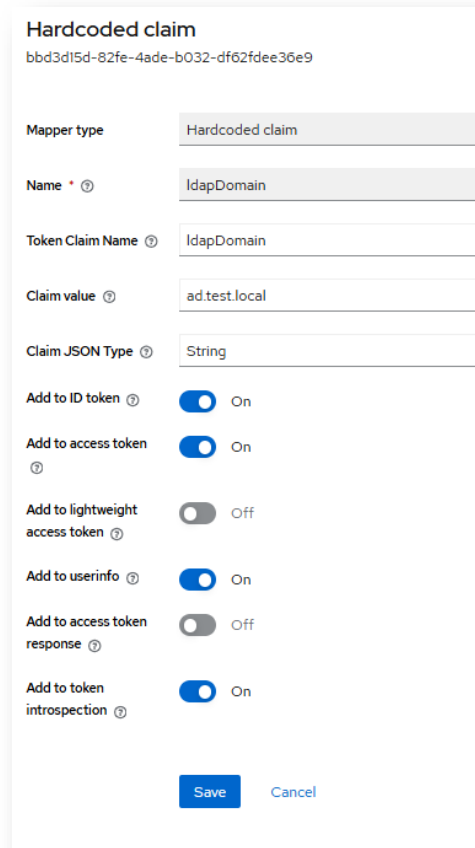


Рисунок 11 — Добавленные роли

9) Перейти вкладку «Client Scores» и создать ресурсы:

- **IdapDomain**. Заполнить поля и активировать опции:
  - Mapper type: Hardcoded claim;
  - Name: IdapDomain;
  - Token Claim Name: IdapDomain;
  - Claim value: ad.test.local;
  - Claim JSON Type: String;
  - Add to ID token: Включено;
  - Add to access token: Включено;

- Add to lightweight access token: Выключено;
- Add to userinfo: Включено;
- Add to access token response: Выключено;
- Add to token introspection: Включено.



**Hardcoded claim**  
bbd3d15d-82fe-4ade-b032-df62fdee36e9

Mapper type: Hardcoded claim

Name \* ⓘ: ldapDomain

Token Claim Name ⓘ: ldapDomain

Claim value ⓘ: ad.test.local

Claim JSON Type ⓘ: String

Add to ID token ⓘ: ☒ On

Add to access token ⓘ: ☒ On

Add to lightweight access token ⓘ: ☐ Off

Add to userinfo ⓘ: ☒ On

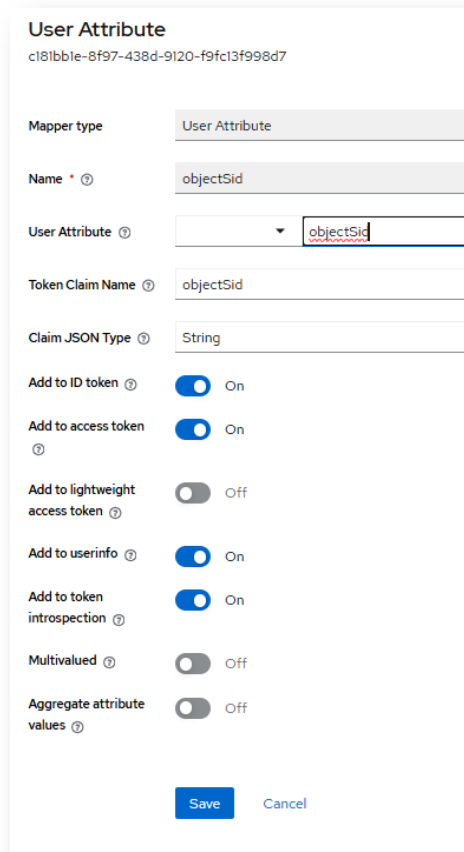
Add to access token response ⓘ: ☐ Off

Add to token introspection ⓘ: ☒ On

**Save** Cancel

Рисунок 12 — Ресурс ldapDomain

- objectSid. Заполнить поля и активировать опции:
  - Mapper type: User Attribute;
  - Name: objectSid;
  - User Attribute: objectSid;
  - Token Claim Name: objectSid;
  - Claim JSON Type: String;
  - Add to ID token: Включено;
  - Add to access token: Включено;
  - Add to lightweight access token: Выключено;
  - Add to userinfo: Выключено;
  - Add to token introspection: Включено;
  - Multivalued: Выключено;
  - Aggregate attribute values: Выключено.



**User Attribute**  
c181bb1e-8f97-438d-9120-f9fc13f998d7

Mapper type: User Attribute

Name: objectSid

User Attribute: objectSid

Token Claim Name: objectSid

Claim JSON Type: String

Add to ID token: On

Add to access token: On

Add to lightweight access token: Off

Add to userinfo: On

Add to token introspection: On

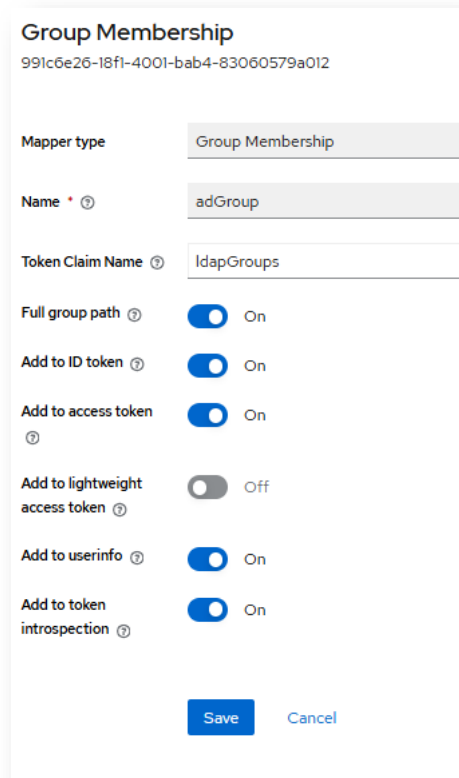
Multivalued: Off

Aggregate attribute values: Off

Save Cancel

Рисунок 13 — Ресурс objectSid

- **adGroup**. Заполнить поля и активировать опции:
  - Group Membership;
  - Mapper type: Group Membership;
  - Name: adGroup;
  - Token Claim Name: ldapGroups;
  - Full group path: Включено;
  - Add to ID token: Включено;
  - Add to access token: Включено;
  - Add to lightweight access token: Выключено;
  - Add to userinfo: Включено;
  - Add to token introspection: Включено.



**Group Membership**  
 991c6e26-18f1-4001-bab4-83060579a012

Mapper type: Group Membership

Name \* ⓘ: adGroup

Token Claim Name ⓘ: IdapGroups

Full group path ⓘ: ☒ On

Add to ID token ⓘ: ☒ On

Add to access token ⓘ: ☒ On

Add to lightweight access token ⓘ: ☐ Off

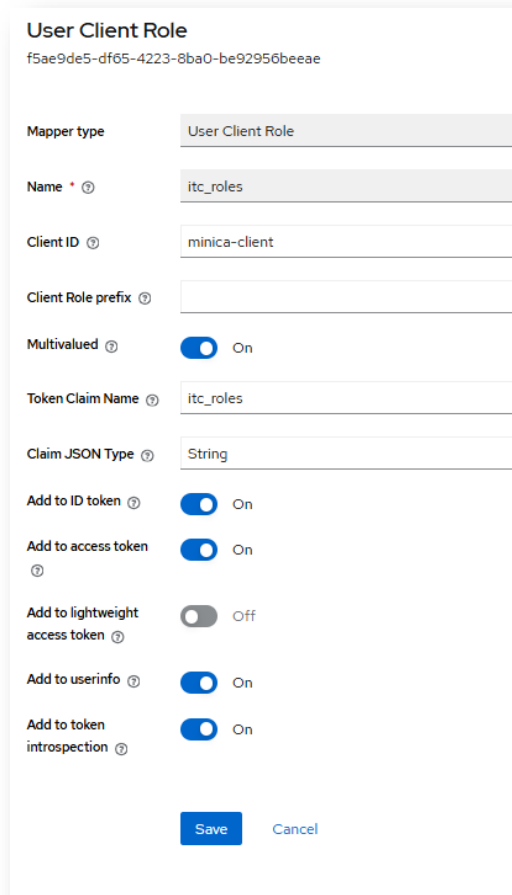
Add to userinfo ⓘ: ☒ On

Add to token introspection ⓘ: ☒ On

**Save** Cancel

Рисунок 14 — Ресурс adGroup

- `itc_roles`. Заполнить поля и активировать опции:
  - Mapper type: User Client Role;
  - Name: `itc_roles`;
  - Client ID: `minica-client`;
  - Client Role prefix: Оставить пустым;
  - Multivalued: Включено;
  - Token Claim Name: `itc_roles`;
  - Claim JSON Type: String;
  - Add to ID token: Включено;
  - Add to access token: Включено;
  - Add to lightweight access token: Выключено;
  - Add to userinfo: Включено;
  - Add to token introspection: Включено.



**User Client Role**  
f5ae9de5-df65-4223-8ba0-be92956beeeae

Mapper type: User Client Role

Name \* ⓘ: itc\_roles

Client ID ⓘ: minica-client

Client Role prefix ⓘ:

Multivalued ⓘ: ☒ On

Token Claim Name ⓘ: itc\_roles

Claim JSON Type ⓘ: String

Add to ID token ⓘ: ☒ On

Add to access token ⓘ: ☒ On

Add to lightweight access token ⓘ: ☐ Off

Add to userinfo ⓘ: ☒ On

Add to token introspection ⓘ: ☒ On

[Save](#) [Cancel](#)

Рисунок 15 — Ресурс `itc_roles`

## 7.8. Создание токенов

После завершения настройки клиента, необходимо пользователям и группам, которые будут иметь доступ к web-panel minica и выдать соответствующие права доступа.

Права доступа для группы и пользователя настраиваются одинаково.

- 1) Перейти вкладку «Users» и нажать на строку пользователя, чтобы выбрать.

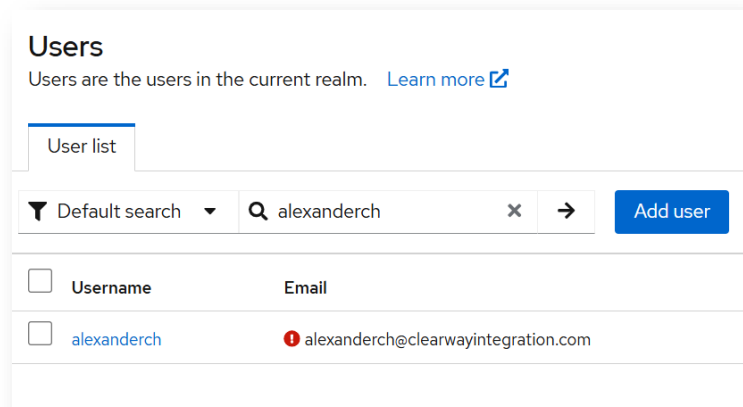


Рисунок 16 — Выбор пользователя

- 2) В карточке пользователя перейти вкладку «Role mapping».
- 3) Нажать на кнопку «Assign role».
- 4) В фильтре выбора набрать «minica-client», выбрать все роли и нажать «Assign».

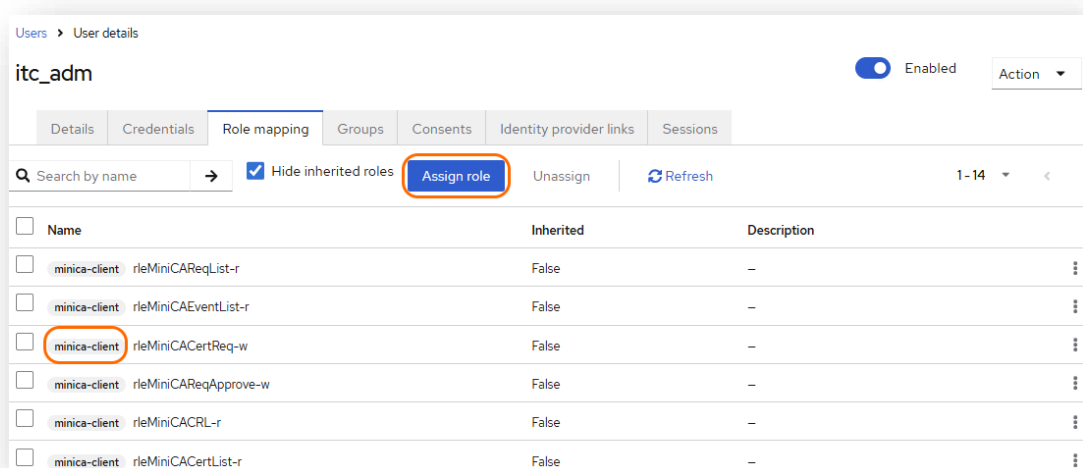


Рисунок 17 — Выбор ролей пользователя

Выбор всех ролей автоматически делает пользователя Администратором minica-web-panel.

Настройки в Keycloak обеспечивают доступ к ресурсам и данным.

Настройки прав пользователей помогают разграничивать доступ к этим ресурсам.

## 8. Установка и настройка микросервиса mOCSP

Дистрибутив mOCSP поставляется в виде DEB пакета для операционных систем архитектуры amd64 (x86\_64) основанных на dpkg (Debian, Ubuntu, Astra Linux).

Поставка в виде RPM-пакета, архива .tar.gz и/или самораспаковывающего архива может быть выполнена по требованию Заказчика.

Пакета имеет наименование формата `mocsp-sfx-X.Y.Z-hash_amd64.deb`, где:

- X.Y.Z — номер версии (старший, младший, номер сборки)
- hash — хэш идентификатор (коммит в репозитории Git)
- amd64 — целевая архитектура.

Пакеты содержат исполняемый файл сервиса `minica-cp-web` и `minica-cp-service`, примеры файлов конфигурации.

Установка пакета производится через терминал путём ввода команды:

```
sudo dpkg -i ./mocsp-sfx-X.Y.Z-hash_amd64.deb
```

где `./mocsp-sfx-X.Y.Z-hash_amd64.deb` требуется заменить на фактический путь до файла пакета.

Установка из самораспаковывающего архива:

- 1) Выдать права на выполнение:

```
sudo chmod 755 mocsp-sfx-X.Y.Z-hash_amd64.run
```

- 2) Запустить архив:

```
sudo ./ mocsp-sfx-X.Y.Z-hash_amd64.run
```

Установка из архива:

- 1) Выдать права на выполнение:

```
sudo chmod 755 unzip mocsp-sfx-X.Y.Z-hash_amd64.zip
```

- 2) Разархивировать архив:

```
sudo unzip mocsp-sfx-X.Y.Z-hash_amd64.zip
```

Внимание! В данном примере сервер mOCSP устанавливается на сервер minica-web-panel.

## 8.1. Каталоги установки службы

При установке пакета автоматически добавляется пользователь `minica` и которому присваивается домашний каталог `/app/itc/mocsp`.

При установке из самораспаковывающего архива или просто архива:

- 1) Создать директорию `/app/itc/mocsp`

```
$ sudo mkdir -p /app/itc/mocsp
```

- 2) Переместить файлы из самораспаковывающего архива в директорию:

```
$ sudo mv mocsp-sfx-X.Y.Z-hash_amd64/* /app/itc/mocsp/
```

- 3) Скопировать исполняемые файлы в корневую директорию:

```
$ sudo scp /app/itc/mocsp/bin/* /app/itc/mocsp/
```

- 4) Создать директорию для журналирования:

```
$ sudo mkdir -p /app/itc/mocsp/logs/
```

Внимание!

Требуется убедиться, что пользователь `minica` существует. Если нет, необходимо создать.

Файл `mocsp` является исполняемым файлом приложения и обычно размещается в директории `/app/itc/mocsp/mocsp`.

Файл `conf/mocsp.yml` — конфигурационный файл микросервиса в формате `yml`. Размещается по пути `/app/itc/mocsp/conf/mocsp.yml`. В конфигурационном файле сохраняются реквизиты доступа к базе данных и другая важная для работы сервиса информация, потому права доступа к файлу (в т.ч. на чтение) для широкой группы пользователей должны быть ограничены.

Файл `mocsp.service` — файл `systemd`, размещается в каталоге `/etc/systemd/system` при ручном развертывании сервиса, используется при запуске сервиса как системного. При установке пакета файл размещается в стандартном каталоге `/lib/systemd/system/`.

## 8.2. Параметры конфигурации mOCSP

Источником конфигурации является конфигурационный файл в формате `JSON (yaml)`.

Пример конфигурационного файла с описанием:

```
1: # Параметры TLS сервера
2: tls:
3:   enable: false # включить TLS
4:   verify: false # включить mTLS
5:   key: conf/tls.key # файл с ключевой парой сервера
6:   cert: conf/tls.crt # файл с сертификатом сервера
7:   ca-cert: conf/ca.crt # файл с сертификатом CA (для mTLS)
8:   ca-certs: [] # доверенные сертификаты (для mTLS)
9:   insecure: false # не проверять сертификат CA
10:
11: listen-ip: "" # адрес, который слушает сервер
12: listen-port: 8888 # порт OCSP сервера
13: url-path: / # базовый URL
14: timeout-ms: 60000 # таймаут
15:
16: key-file: conf/ocsp.key # ключ OCSP сервера
17: cert-file: conf/ocsp.crt # сертификат OCSP сервера
18: ca-file: conf/ca.crt # сертификат CA
19: hash: SHA256 # используемых хэш
20: add-cert: true # добавить сертификат в ответ OCSP сервера
21: nmin: 60 # время в минутах для формирования NextUpdate или 0
22: update-expired: true # обновлять статус просроченных сертификатов в БД
23:
24: # Опции журналирования
25: log:
26:   level: info # уровень журналирования
27:   flood/trace/debug/info/notice/warn/error/crit/fatal
28:   pipe: "stdout" # stdout/stderr/null или ""
29:   file: "" # файл или ""
30:   file-mode: "0640" # права на файл журнала
31:   format: "tint" # формат журнала (std/text/json/tint)
32:   gold: false # добавить в журнал идентификатор горютины
33:   idOff: false # отключить UUID идентификатор для каждой записи (logId)
34:   sumOff: false # отключить вычисление контрольной суммы
35:   sumFull: false # включить полный алгоритм вычисления контрольной суммы по
36:   всем атрибутам
37:   sumChain: false # вычислять контрольную сумму с учетом суммы предыдущей
38:   записи
39:   sumAlone: false # сохранять в журнале контрольную сумму в виде отдельного
```

```
атрибута (logSum)
37: timeOff: false # не выводить метку времени
38: timeLocal: true # выводить локальное время (по умолчанию в UTC)
39: timeMicro: false # выводить временную метку с микросекундами
40: timeFormat: "" # формат метки времени при истолковании tint
41: src: true # выводить ссылки на исходные тексты
42: srcPkg: false # выводить каталог в ссылках на исходные тексты
43: srcFunc: false # выводить имена функций
44: srcExt: false # выводить расширение ".go"
45: srcFields: # дополнительные атрибуты в блоке "source" (map)
46:   id: mocsp
47: color: false # включить подсветку (для format=tint)
48: levelOff: false # не выводить уровень журналирования
49: prefix: "" # префикс для классического журнала
50: addKey: "" # ключ для всех записей
51: addValue: null # значение для всех записей
52:
53: # Настройки ротации
54: rotate:
55:   enable: true # включить ротацию журнала
56:   maxSize: 10 # максимальный размер файла журнала [МБ]
57:   maxAge: 10 # время хранения старых журналов [дней]
58:   maxBackups: 100 # максимальное число файлов старых журналов
59:   localTime: true # использовать локальное время
60:   compress: true # включить сжатие старых журналов (gzip)
61:
62: # Секретный пароль для расшифровки пароля доступа к БД или "".
63: # Данная парольная фраза может быть замаскирована.
64: db-pass: 4nwoMF8jlizZZqmLtHZ3pFNlaXO5emttU8jCira9AXU # 1234
65:
66: # Настройка СУБД
67: database:
68:   type: postgres # тип СУБД
69:
70: # URL доступа к СУБД Postgres.
71: # Для обеспечения проверки сертификата TTL необходимо
72: # использовать опцию подключения sslmode=verify-full.
73: # В противном случае сервис будет доверять любому TLS сертификату
74: # без каких-либо проверок.
```

```
75: #
76: # Пароль в составе URL може быть зашифрован с помощью утилиты
77: # maskpass с использованием парольной фразы db-pass
78: # (которая в свою очередь может быть замаскирована)
79: url:
    postgres://minica:Lcp8gWO0sXwp00vJSlenGP8ab7uTE8UZI5M4oNLsEembvB_9@loca
    lhost:5432/minica
80:
81: # Разрешить подключение к БД без TLS.
82: # По умолчанию, если сервер БД не использует TLS, то сервис завершает работу.
83: allow-without-tls: false
84:
85: # Не удалять незначащие нули серийных номеров сертификатов при поиске в БД
86: no-trim-zeros: false
```

Необходимо изменить поля конфигурационного файла `mocsp.yml`:

1) Открыть файл:

```
$ sudo vim /app/itc/mocsp/conf/mocsp.yml
```

2) Сменить значение:

```
1: tls:
2:   enable: false # включить TLS
```

на значение

```
1: tls:
2:   enable: true # включить TLS
```

3) Установить пароль подключения БД:

```
1: db-pass: 12345678 # 1234
```

4) Задать параметры БД:

```
1: url: postgres://minica:12345678@pg1.test.ru:5432/minica
```

5) Сохранить внесенные изменения и закрыть файл.

### 8.3. Настройка сервиса systemd

Микросервис mOCSP является консольным приложением, вызов fork не применяется. Для запуска процесса в фоновом режиме рекомендуется создание юнита systemd. П

Пример создания файла `mocsp.service` для реализации системного сервиса:

- 1) Создать файл сервиса:

```
$ sudo vim /lib/systemd/system/mocsp.service
```

- 2) Заполнить данными:

```
1: [Unit]
2: Description=ITC Mini CA OCSP service
3: [Service]
4: # Переменные окружения (MOCSP_CONF, MOCSP_OPTIONS)
5: EnvironmentFile=/app/itc/mocsp/mocsp.env
6: # Рабочий каталог сервиса
7: WorkingDirectory=/app/itc/mocsp
8: # Исполняемый файл
9: ExecStart=/app/itc/mocsp/mocsp $MOCSP_OPTIONS
10: # Имя пользователя и группы от имен которых будет запущен сервис
11: User=minica
12: Group=minica
13: # Разрешить работу сервиса на привилегированных портах (<1024)
14: AmbientCapabilities=CAP_NET_BIND_SERVICE
15: # Процесс не выполняет fork
16: Type=simple
17: # Код возврата при успехе
18: SuccessExitStatus=0
19: # После завершения процесса сервис завершается
20: RemainAfterExit=no
21: # Перезапускать сервис всегда
22: Restart=always
23: # Время ожидания перед перезапуском сервиса
24: RestartSec=10
25: # MiniCA для штатного завершения обрабатывает сигнал SIGINT (Ctrl+C)
26: KillSignal=SIGINT
27: # Перенаправление stdout/stderr
28: #StandardOutput=null
```

```
29: StandardOutput=file:/app/itc/mocsp/logs/mocsp.log
30: StandardError=file:/app/itc/mocsp/logs/mocsp.log
31: [Install]
32: WantedBy=default.target
```

- 3) Сохранить изменения и закрыть файл.

При выполнении системного юнита systemd возможна работа микросервиса на привилегированном TCP порту.

Создание файла окружения:

- 1) Создать файл `mocsp.env` в корневой директории `/mocsp`:

```
$ sudo vim /app/itc/mocsp/mocsp.env
```

- 2) Заполнить файл данными:

```
1: # Переменные окружения сервиса `minica`
2: # =====
3: # Опции командной строки передаваемые через minica.service
4: # (опции имеют приоритет перед переменными окружения)
5: MOCSP_OPTIONS="-log-level=debug -log-format=tint"
6: # Путь к файлу конфигурации
7: MOCSP_CONF="conf/mocsp.yml"
```

- 3) Сохранить внесенные данные и закрыть файл.
- 4) Перезагрузить системный демон:

```
$ sudo systemctl daemon-reload
```

- 5) Включить автозапуск при старте системы:

```
$ sudo systemctl enable mocsp.service
```

Для просмотра журнала используется команда:

```
$ sudo journalctl -u mocsp.service -o cat
```

## 8.4. Создание сертификатов для mOCSP.

Создание сертификата и ключа для службы `mocsp`, которая будет настроена на сервере `web-minica-panel` согласно схеме на рисунке 4.

- 1) Перейти на сервер ЦС (`minica`) и выпустить сертификаты.
- 2) Создать TLS-ключ для службы `mocsp` с помощью запроса командой:

```
$ sudo openssl req -new -keyout /app/itc/minica/ca/certs/ocsp.key -out  
/app/itc/minica/ca/certs/ocsp.req -sha256 -nodes -subj "/CN=minica-web.test.local" -addext  
"subjectAltName = DNS:minica-web.test.local " -addext "extendedKeyUsage = serverAuth,  
clientAuth"
```

- 3) Создать TLS-сертификат для `mocsp` с помощью запроса командой:

```
$ sudo /app/itc/minica/mclient ca -csr /app/itc/minica/ca/certs/ocsp.req -out  
/app/itc/minica/ca/certs/ocsp.crt -template tls
```

- 4) Полученные сертификаты скопировать на сервер `minica-web-panel`:

```
$ sudo scp /app/itc/minica/ca/certs/tls.* administrator@minica-  
web.test.local:/home/administrator/  
$ sudo scp /app/itc/minica/ca/certs/ocsp.* administrator@minica-  
web.test.local:/home/administrator/  
$ sudo scp /app/itc/minica/ca/ca.crt administrator@minica-web.test.local:/home/administrator/
```

- 5) На сервере `minica-web-panel` переместить файлы в папку `/conf` и назначить права:

```
$ sudo mv ocsp.* /app/itc/mocsp/conf/  
$ sudo mv ca.crt /app/itc/mocsp/conf/  
$ sudo mv tls.* /app/itc/mocsp/conf/  
$ sudo chown minica:minica -R /app/
```

- 6) Запустить службу `mocsp`:

```
$ sudo systemctl start mocsp.service
```

Просмотр журнала:

```
$ sudo journalctl -u mocsp.service -o cat
```

- 7) Проверить работу микросервиса командой:

```
$ telnet 172.24.240.12 8888
```

## 8.5. Перенастройка файла конфигурации mocsp.yml для выполнения требования безопасности.

Для обеспечения требований безопасности к ключу и паролям в конфигурационных файлах mOCSF используется приложение Maskpass, поставляемое совместно с дистрибутивом.

### Примечание!

Зашифрованные пароли можно взять с конфигурационного файла minica.yml

Для перенастройки необходимо выполнить следующие действия:

- 1) Зашифровать пароль для входа в БД PostgreSQL ключевой фразой при помощи Markpass:

```
$ sudo ./maskpass -p 12345678 -k password-fraza
```

В результате выполнения команды будет получен вывод:

```
eTsEl_4QwtmRAqv3nm-KAn1aFr9QzYIJQccNxliHwbEvs089
```

- 2) Открыть файл конфигурации `mocsp.yml` для редактирования:

```
$ sudo vim /app/itc/mocsp/conf/mocsp.yml
```

- 3) Заменить пароль для входа зашифрованной комбинацией пароля:

```
1: url: postgres://minica:eTsEl\_4QwtmRAqv3nm-KAn1aFr9QzYIJQccNxliHwbEvs089@pg1.test.local:5432/minica
```

- 4) Сохранить внесенные изменения и закрыть файл.  
5) Зашифровать парольную фразу для БД при помощи Markpass:

```
$ sudo ./maskpass -p password-fraza
```

В результате выполнения команды будет получен вывод:

```
utna4dVpB_BC5XgjkNL4EonxvtG6Bj3y8-NbecW_36whUD3zDO7QIUoj
```

- 6) Снова открыть файл конфигурации `mocsp.yml` для редактирования:

```
$ sudo vim /app/itc/mocsp/conf/mocsp.yml
```

- 7) Заменить пароль для БД зашифрованной комбинацией пароля.

```
1: db-pass: utna4dVpB_BC5XgjkNL4EonxvtG6Bj3y8-NbecW_36whUD3zDO7QIUoj # 1234
```

- 8) Сохранить внесенные изменения и закрыть файл.  
9) Перезапустить службу mOCSP:

```
$ sudo systemctl restart mocsp.service
```

## 9. Настройка сервера точек распространения

### 9.1. Каталоги распространения.

Каталоги создаются на серверах точек распространения с предварительно установленным и настроенным пакетом Nginx. Настройка конфигурационного файла Nginx описана в пункте 0 данной инструкции, также в данном пункте указана настройка точек распространения.

Для функционирования автоматической передачи данных CRL необходимо установиться дополнительный пакет `sshpas`:

```
$ sudo apt install sshpass
```

- 1) Создать каталоги:
  - Каталог AIA – для хранения корневых и промежуточных сертификатов:

```
$ sudo mkdir -p /app/inetpub/pki/aia/
```

- Каталог CDP – для размещения списков отозванных сертификатов (CRL и DeltaCRL):

```
$ sudo mkdir -p /app/inetpub/pki/cdp/
```

- 2) Установить необходимые права доступа и владельца для каталогов:

```
$ sudo chmod 755 /app/inetpub/pki/aia/  
$ sudo chmod 755 /app/inetpub/pki/cdp/  
$ sudo chown minica:minica -R /app/inetpub
```

### 9.2. Параметры конфигурации Nginx

Источником конфигурации является конфигурационный файл в формате `.conf`.

Часть с описанием конфигурационного файла Nginx для настройки точек распространения, в качестве примера:

```
1: server {  
2:     listen 80;  
3:     listen [::]:80;  
4:
```

```
5:  server_name crt1.test.local;
6:
7:  root /app/inetpub;
8:
9:  # Явно указываем индексный файл (если он нужен)
10:  index root-test.crt; # Замените на реальное имя файла
11:
12:  # Добавляем MIME-тип для .crt
13:  types {
14:      application/pkix-crt crt;
15:  }
16:
17:  location / {
18:      # Разрешаем листинг директорий (если нужно)
19:      autoindex on; # Опционально
20:
21:      # Ищем сначала файл, затем директорию
22:      try_files $uri $uri/ =404;
23:  }
24: }
```

Внимание! После изменения конфигурационного файла Nginx необходимо перезапустить службу.

### 9.3. Настройка скриптов распространения.

Примечание.  
Настройка для корневого или выдающего ЦС одинакова.

- 1)
- 2) Перейти на выдающий сервер ЦС.
- 3) Перейти в рабочую директорию:

```
$ cd /app/itc/minica/
```

- 4) Создать скрипт для публикации CRL:

```
$ sudo vim pubcrl.sh
```

Внести данные в файл:

```
1: #!/bin/bash
2: sudo -u minica /app/itc/minica/mclient updcr1 -out
   /app/itc/minica/ca/crl/TESTSubCA.crl
3: sshpass -p "password" scp /app/itc/minica/ca/crl/*.crl minica@minica-
   web.test.local:/app/inetpub/pki/cdp/
```

- 5) Сохранить и закрыть.
- 6) Создать скрипт для публикации DeltaCRL:

```
$ sudo vim pubdeltacr1.sh
```

- 7) Внести данные в файл:

```
1: #!/bin/bash
2: sudo -u minica /app/itc/minica/mclient updeltacr1 -out
   /app/itc/minica/ca/crl/TESTSubCA+.crl
3: sshpass -p "password" scp /app/itc/minica/ca/crl/*+*.crl minica@minica-
   web.test.local:/app/inetpub/pki/cdp/
```

- 8) Сохранить и закрыть

Для корректной работы скриптов в службе SSH требуется отключить проверку запроса на подключение к новым узлам:

- 1) Открыть конфигурационный файл:

```
$ sudo vim /etc/ssh/ssh_config
```

- 2) Сменить строку:

```
1: # StrictHostKeyChecking ask
```

на

```
1: # StrictHostKeyChecking no
```

- 3) Сохранить и закрыть файл.
- 4) Перезапустить службу:

```
$ sudo systemctl restart sshd
```

## 9.4. Автоматизация обновления CRL с помощью cron

Чтобы списки отозванных сертификатов (CRL и DeltaCRL) обновлялись автоматически по расписанию, необходимо настроить планировщик задач `cron`.

- 1) Создать файл конфигурации:

```
$ sudo vim /etc/cron.d/minica-crl
```

- 2) Внести данные в файл:

```
1: SHELL=/bin/sh
2: PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
3:
4: sun,mon,tue,wed,thu,fri,sat
5: # | | | | |
6: # * * * * * user-name command to be executed
7: 0 22 * * 7 minica /app/itc/minica/pubcrl.sh
8: 0 21 * * * minica /app/itc/minica/pubdeltacrl.sh
```

- 3) Сохранить и закрыть файл.
- 4) Перезапустить службу `cron`:

```
$ sudo systemctl restart cron
```

Проверка статуса приложения

```
$ sudo systemctl status cron
```

В данном примере, согласно конфигурации, будет происходить:

- Запуск скрипта `/app/itc/minica/pubcrl.sh` для публикации полного списка CRL каждое воскресенье в 22:00;
- Запуск скрипта `/app/itc/minica/pubdeltacrl.sh` для публикации дельта-списка DeltaCRL ежедневно в 21:00;
- Выполнение задач от имени пользователя `minica`.

Для получения подробной информации по настройке `crontab` можно воспользоваться командой:

```
$ man crontab
```

## 9.5. Настройка точек распространения в конфигурационном файле minica.

Чтобы в выпускаемых сертификатах содержалась актуальная информация о точках распространения списков отозванных сертификатов (CRL) и доступе к информации об УЦ (AIA), необходимо отредактировать конфигурационный файл `minica.yml`.

- 1) Открыть файл конфигурации:

```
$ sudo vim /app/itc/minica/conf/minica.yml
```

- 2) Внести в файл необходимые URL-адреса и настроить сроки действия:

```
1: san-ospace: [] # дополнительные расширения SAN otherName
2: cdp:
3:   - "http://web-minica.test.local/pki/cdp/TESTSubCA.crl"
4:   - "http://web-minica.test.local/pki/cdp/TESTSubCA.crl" # Если два CRL
5: aia: ["OCSP;URI:http://web-minica.test.local:8888/ocsp", "caissuers;URI:http:// web-
minica.test.local/pki/aia/ca.crt"]
6: freshest-crl:
7:   - "http://web-minica.test.local/pki/cdp/TESTSubCA+.crl"
8:   - "http://web-minica.test.local/pki/cdp/TESTSubCA+.crl" # Если два CRL
9: overlap-crl: 4 # разница между nextUpdate и nextPublish для CRL в часах или 0
10: days: # заданные сроки действия сертификатов, CRL и DeltaCRL
11:   cert: 0 # сроки действия сертификатов (0 - использовать конфигурацию
OpenSSL)
12:   crl: 7 # сроки действия CRL (0 - использовать конфигурацию OpenSSL)
13:   delta-crl: 1 # сроки действия DeltaCRL (0 - использовать конфигурацию OpenSSL
для CRL)
14: overlap: 1440 # смещение начала срока действия сертификата в прошлое
[минут]
15: no-limit-ca: false # не ограничивать срок действия целевых сертификатов
сертифика-том ЦС
16: short-days: 30 # максимальный срок действия "короткоживущего" сертификата
[дней]
```

В примере настроено:

- Два сервера CRL и два сервера DeltaCRL;
- Сервер AIA, который указывает на службу OCSP для проверки статуса сертификата в реальном времени и на расположение сертификата самого УЦ;
- Период перекрытия DeltaCRL составляет 4 часа, CRL — сутки;
- Срок действия полного CRL установлен в 7 дней, а DeltaCRL — в 1 день.

## Приложение А Последовательность шагов при развертывании minica, minica web-panel, mocsp

### А.1 Установка minica

```
$ sudo dpkg -i к/файлу/minica-X.Y.Z-hash_amd64.deb
```

или

```
$ sudo chmod 755 ./minica-sfx-2.4.4-d48e974_amd64-2025.06.20.run  
$ sudo ./minica-sfx-2.4.4-d48e974_amd64-2025.06.20.run
```

#### А.1.1 Создание каталогов

```
$ sudo mkdir -p /app/itc/  
$ sudo mkdir -p /app/itc/minica/logs/
```

Переместить СПО:

```
$ sudo mv minica/* /app/itc/minica/  
$ sudo scp /app/itc/minica/bin/* /app/itc/minica/
```

Создание ТУЗ:

```
$ sudo adduser minica  
$ sudo passwd minica
```

Создание файла конфигурации

```
$ sudo vim /app/itc/minica/minica.env
```

Изменить конфигурационный файл:

```
$ sudo vim /app/itc/minica/conf/minica.yml
```

Открыть файл с настройками mclient. Изменить строку подключение.

```
$ sudo vim /app/itc/minica/conf/mclient.yml
```

### A.1.2 Настройка сервиса

```
$ sudo vim /lib/systemd/system/minica.service
```

Задать права на каталог `/app/` от ТУЗ minica:

```
$ sudo chown minica:minica -R /app/  
$ sudo systemctl daemon-reload  
$ sudo systemctl enable minica.service  
$ sudo systemctl start minica.service № Запуститься с ошибкой  
$ sudo systemctl status minica.service
```

### A.1.3 Создание сертификатов

Перейти в директорию:

```
$ cd /app/itc/minica
```

Для создания корневого ключа, выполнить команду:

```
$ sudo openssl genpkey -algorithm RSA -aes-256-cbc -out ca/ca.key -pkeyopt  
rsa_keygen_bits:4096  
$ sudo openssl req -x509 -new -key ca/ca.key -sha256 -days 3650 -out ca/ca.crt -subj  
"/C=RU/ST=Moscow/L=Moscow/O=test/OU=IT/CN=root ca cert"  
$ sudo scp /app/itc/minica/ca/ca.* /app/itc/minica/conf/  
$ sudo chown minica:minica -R /app/  
$ sudo systemctl start minica.service  
$ sudo systemctl status minica.service  
$ sudo tail -f /app/itc/minica/logs/minica.log
```

### A.1.4 Выпуск сертификата выдающего ЦС

```
$ sudo openssl genrsa -aes256 -out /app/itc/minica/ca/subCA.key 4096  
$ sudo openssl req -config conf/openssl.cnf -new -sha256 -key /app/itc/minica/ca/subCA.key -out  
/app/itc/minica/ca/certs/subCA.csr  
$ sudo /app/itc/minica/mclient ca -csr /app/itc/minica/ca/certs/subCA.csr -out  
/app/itc/minica/ca/subCA.crt -template subca -days 1825  
$ sudo bash -c 'cat ca/ca.crt ca/subCA.crt > ca/subCA-chain.crt'
```

Создать TLS-ключ для ЦС:

```
$ sudo openssl req -new -keyout /app/itc/minica/ca/certs/tls.key -out  
/app/itc/minica/ca/certs/tls.req -sha256 -nodes -subj "/CN=minica.test.local" -addext  
"subjectAltName = DNS:minica.test.local" -addext "extendedKeyUsage = serverAuth, clientAuth"  
$ sudo /app/itc/minica/mclient ca -csr /app/itc/minica/ca/certs/tls.req -out  
/app/itc/minica/ca/certs/tls.crt -template tls
```

### **A.1.5 Создание TLS- ключа для PostgreSQL**

```
$ sudo openssl req -new -keyout /app/itc/minica/ca/certs/postgres.key -out  
/app/itc/minica/ca/certs/postgres.req -sha256 -nodes -subj "/CN=pg1.test.local" -addext  
"subjectAltName = DNS:pg1.test.local" -addext "extendedKeyUsage = serverAuth, clientAuth"  
$ sudo /app/itc/minica/mclient ca -csr /app/itc/minica/ca/certs/postgres.req -out  
/app/itc/minica/ca/certs/postgres.crt -template tls
```

### **A.1.6 Перенастройка службы minica на выпущенные сертификаты.**

```
$ cd /app/itc/minica/  
$ sudo mv /app/itc/minica/ca/ca.crt /app/itc/minica/ca/rootca.crt  
$ sudo mv /app/itc/minica/ca/ca.key /app/itc/minica/ca/rootca.key  
$ sudo rm /app/itc/minica/conf/ca.crt /app/itc/minica/conf/ca.key  
$ sudo mv ca/subCA.crt ca/ca.crt  
$ sudo mv ca/subCA.key ca/ca.key  
$ sudo scp ca/ca.key ca/ca.crt conf/
```

Скопировать TLS- сертификат для minica в папку с конфигурацией.

```
$ sudo scp ca/certs/tls.* conf/
```

Открыть файл конфигурации minica и изменить строчки:

```
$ sudo vim /app/itc/minica/conf/minica.yml
```

Открыть файл конфигурации minica и изменить строчки:

```
$ sudo vim /app/itc/minica/conf/mclient.yml  
$ sudo chown minica:minica -R /app/  
$ sudo systemctl restart minica.service
```

### A.1.7 Выполнить обновление закрытой и открытой пары ключей

```
$ cd /app/itc/minica
```

Открыть файл config-jwt.env:

```
$ vim config-jwt.env
```

Задать пароль:

```
1: JWT_KEY_PASS="12345678"
```

Запустить скрипты настройки:

```
$ cd /app/itc/minica/script/  
$ sudo ./make-jwt-key.sh  
$ sudo ./make-mclient-key.sh  
$ sudo ./make-mclient-jwt.sh
```

### A.1.8 Перенастройка minica.yaml для выполнения требования безопасности

```
set +H  
$ cd /app/itc/minica/  
$ sudo openssl pkcs8 -topk8 -v2 aes-256-cbc -passin pass:12345678 -in ca/ca.key -out  
ca.key.encrypted  
$ sudo scp ca.key.encrypted ca/ca.key  
$ sudo scp ca.key.encrypted conf/ca.key  
./maskpass -p 12345678  
yxtde0FYcvVTLwPAqQUvasXCQL3xGqlgpwbsVA5Vx2Xz0wMs  
$ sudo vim /app/itc/minica/conf/minica.yml  
ca-pass: yxtde0FYcvVTLwPAqQUvasXCQL3xGqlgpwbsVA5Vx2Xz0wMs  
$ sudo ./maskpass -p 12345678 -k password-fraza  
eTsEl_4QwtmRAqv3nm-KAn1aFr9QzYIJQccNxliHwbEvs089  
$ sudo vim /app/itc/minica/conf/minica.yml
```

Зашифровать парольную фразу для БД:

```
$ sudo ./maskpass -p password-fraza  
utna4dVpB_BC5XgjkNL4EonxvtG6Bj3y8-NbecW_36whUD3zDO7QIUoj  
$ sudo vim /app/itc/minica/conf/minica.yml
```

```
db-pass: utna4dVpB_BC5XgjkNL4EonxvtG6Bj3y8-NbecW_36whUD3zDO7QIUoj # 1234
```

Перезапустить службу minica

```
$ sudo systemctl restart minica
```

## A.2 Установка и настройка PostgreSQL

```
$ sudo apt update
$ sudo apt install postgresql -y
```

### A.2.1 Редактирование конфигурационных файлов

```
$ sudo vim /etc/postgresql/15/main/postgresql.conf
```

```
1: listen_addresses='*'          # (change requires restart)
2: port = 5432                   # (change requires restart)
3: max_connections = 500         # (change requires restart)
4: ssl = on
5: ssl_ca_file = 'ca.crt'
6: ssl_cert_file = '/etc/ssl/certs/test.pem'
7: ssl_key_file = '/etc/ssl/private/test.key'
```

```
$ sudo vim /etc/postgresql/15/main/pg_hba.conf
```

```
1: local all postgres peer
2: # TYPE DATABASE USER ADDRESS METHOD
3: # "local" is for Unix domain socket connections only
4: local all all peer
5: # IPv4 local connections:
6: #host all all 127.0.0.1/32 md5
7: host all all 0.0.0.0/0 md5
8: # IPv6 local connections:
9: host all all ::1/128 md5
10: # Allow replication connections from localhost, by a user with the
11: # replication privilege.
12: local replication all peer
13: host replication all 127.0.0.1/32 md5
14: host replication all ::1/128 md5
```

Перезапуск службы

```
$ sudo systemctl restart postgresql
```

### A.2.1 Перенастроить PostgreSQL на новые сертификаты

Скопировать TLS- сертификат для Postgres в папку с конфигурацией БД.

```
$ sudo scp ca/certs/postgres.* administrator@pg1.test.local:/home/administrator/  
$ sudo scp ca/ca.crt administrator@pg1.test.local:/home/administrator/  
$ mv postgres.* /etc/postgresql/15/main/  
$ mv ca.crt /etc/postgresql/15/main/  
$ sudo chown postgres:postgres /etc/postgresql/15/main/postgres.*  
$ sudo vim /etc/postgresql/15/main/postgresql.conf  
$ sudo systemctl status postgresql
```

### A.2.2 Настройка БД postgresql

```
$ sudo -u postgres createuser -l -P minica  
$ sudo -i -u postgres psql  
ALTER ROLE minica WITH PASSWORD 'NewPasswd';  
ALTER ROLE postgres WITH PASSWORD 'NewPasswd';  
CREATE DATABASE minica OWNER minica;  
GRANT ALL PRIVILEGES ON DATABASE minica TO minica;  
\q
```

1) Проверка:

```
$ sudo -u postgres psql  
# \l  
# \q
```

2) Локально на сервере БД:

```
psql -h 127.0.0.1 -d minica -U minica -W
```

3) Проверить на minica, если предварительно установлен `psql`

4) Скопировать БД.

```
scp /app/itc/minica/deploy/sql/minica.up.sql administrator@pg1.test.local:/home/administrator/
```

## 5) Развернуть БД

```
psql -h localhost -U postgres -d minica -f minica.up.sql
```

## 6) Выполнить далее:

```
$ sudo -u postgres psql minica
# GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO minica;
# GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO minica;
# GRANT ALL PRIVILEGES ON TABLE csr TO minica;
# GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO minica;
\q
```

## 7) Добавить группы:

```
$ sudo -u postgres psql minica
```

```
1: INSERT INTO public.templ
2: (id, dtime, "name", descr, days, noaia, nocdp, short, "version", deleted, "key_type",
   key_size, exts, bc, ku, eku, ski, aki, ext)
3: VALUES(1, '2025-01-20 15:41:08.726', 'tls', 'TLS клиент/сервер', 365, false, false,
   false, 1, false, 'RSA', 2048, "", "", 'critical,digitalSignature,keyEncipherment',
   'serverAuth,clientAuth', 'hash', 'keyid,issuer', '');
4: INSERT INTO public.templ
5: (id, dtime, "name", descr, days, noaia, nocdp, short, "version", deleted, "key_type",
   key_size, exts, bc, ku, eku, ski, aki, ext)
6: VALUES(3, '2025-01-20 15:41:08.750', 'tls-client', 'TLS клиент', 365, false, false, false,
   1, false, "", 2048, "", "", 'critical,digitalSignature,keyEncipherment',
   'emailProtection,clientAuth', 'hash', 'keyid,issuer', '');
7: INSERT INTO public.templ
8: (id, dtime, "name", descr, days, noaia, nocdp, short, "version", deleted, "key_type",
   key_size, exts, bc, ku, eku, ski, aki, ext)
9: VALUES(4, '2025-01-20 15:41:08.761', 'ocsp-server', 'OCSP сервер', 365, true, true,
   false, 1, false, 'RSA', 2048, "", 'CA:FALSE', 'critical,digitalSignature', 'OCSPSigning',
   'hash', 'keyid,issuer', 'noCheck=ignored');
10: INSERT INTO public.templ
11: (id, dtime, "name", descr, days, noaia, nocdp, short, "version", deleted, "key_type",
   key_size, exts, bc, ku, eku, ski, aki, ext)
12: VALUES(5, '2025-01-20 15:41:08.773', 'user', 'сертификат пользователя', 365, false,
   false, false, 1, false, "", 0, "", "", 'critical,digitalSignature,keyEncipherment',
```

```
'critical,codeSigning', 'hash', 'keyid,issuer', '');  
13: INSERT INTO public.templ  
14: (id, dtime, "name", descr, days, noaia, nocdp, short, "version", deleted, "key_type",  
    key_size, exts, bc, ku, eku, ski, aki, ext)  
15: VALUES(6, '2025-01-20 15:41:08.782', 'tls-short', 'TLS клиент/сервер', 31, false, false,  
    true, 1, false, "", 0, "", 'critical,digitalSignature,keyEncipherment',  
    'serverAuth,clientAuth', 'hash', 'keyid,issuer', '');  
16: INSERT INTO public.templ  
17: (id, dtime, "name", descr, days, noaia, nocdp, short, "version", deleted, "key_type",  
    key_size, exts, bc, ku, eku, ski, aki, ext)  
18: VALUES(2, '2025-01-20 15:41:08.740', 'tls-server', 'TLS сервер', 365, false, false, false,  
    1, false, 'RSA', 2048, "", 'critical,digitalSignature,keyEncipherment', 'serverAuth',  
    'hash', 'keyid,issuer', '');  
19: INSERT INTO public.templ  
20: (id, dtime, "name", descr, "policy", priv, days, noaia, nocdp, short, "version", deleted,  
    "key_type", key_size, exts, bc, ku, eku, ski, aki, ext)  
21: VALUES(7, '2025-07-21 14:26:19.551', 'subca', 'Subordinate CA', "", 1460, false, false,  
    false, 1, false, "", 0, "", 'critical,CA:true,pathlen:0',  
    'critical,cRLSign,digitalSignature,keyCertSign', "", 'hash', 'keyid,issuer', '');
```

### A.3 Установка и настройка микросервиса minica web-panel

```
$ sudo dpkg -i к/файлу/minica-cp-web-2.3.0-hash_amd64.deb к/файлу/minica-cp-service-2.3.0-  
hash_amd64.deb
```

или

```
$ sudo chmod 755 unzip minica-cp-web-2.3.0-rc1.zip unzip minica-cp-service-2.3.0-rc1.zip  
$ sudo unzip minica-cp-web-2.3.0-rc1.zip  
$ sudo unzip minica-cp-service-2.3.0-rc1.zip  
$ sudo apt update  
$ sudo apt install nginx dotnet6 -y
```

Создать каталоги

```
$ sudo mkdir -p /app/itc/itc/config/  
$ sudo mkdir -p /app/itc/minica-spa-service  
$ sudo mkdir -p /app/itc/minica-SP-service  
$ sudo mkdir -p /app/itc/itc/logs  
$ sudo mkdir -p /app/itc/nginx/tls
```

```
$ sudo mkdir -p /app/itc/itc/certs
```

Переместить:

```
$ sudo mv MiniCA CP 2.4.0-rc6/* /app/itc/minica-SP-service/  
$ sudo mv minica-spa-2.4.0-rc6/* /app/itc/minica-spa-service  
$ sudo mkdir -p /app/itc/mocsp/logs/  
$ sudo adduser minica  
$ sudo passwd minica  
$ sudo vim /app/itc/itc/config/itc_api_minica_controlPanel_config.json
```

### A.3.1 Параметры конфигурации файла `client_env.js`

Отредактировать:

```
$ vim /app/itc/minica-spa-service/assets/env/client_env.js  
$ sudo vim /app/itc/minica-spa-service/assets/env/client_env.js  
$ sudo vim /app/itc/itc/env/client_env.j
```

Создать файл в директории:

```
$ sudo vim /app/itc/itc_svc_env.conf  
ITCRoot="/app/itc"
```

Выдать права исполняемому файлу:

```
$ sudo chmod 755 /app/itc/minica-SP-service/ITC.Api.Minica.ControlPanel
```

```
# web-minica
```

### A.3.2 Настройка сервиса `systemd`

Создать сервис:

```
$ sudo vim /lib/systemd/system/minica.spa.service
```

Запуск службы, проверка статуса и включение в автозагрузку службы:

```
$ sudo systemctl daemon-reload  
$ sudo systemctl enable minica.spa.service
```

Проверить ошибки службы:

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

```
$ sudo tail -f /app/itc/itc/logs/itc_api_minica_controlPanel/file.log
```

### A.3.3 Создание сертификатов для web-minica

```
$ sudo openssl req -new -keyout /app/itc/minica/ca/certs/panel.key -out  
/app/itc/minica/ca/certs/panel.req -sha256 -nodes -subj "/CN=minica-web.test.local" -addext  
"subjectAltName = DNS:minica-web.test.local" -addext "extendedKeyUsage = serverAuth, clientAuth"  
$ sudo /app/itc/minica/mclient ca -csr /app/itc/minica/ca/certs/panel.req -out  
/app/itc/minica/ca/certs/panel.crt -template tls  
$ sudo scp /app/itc/minica/ca/certs/panel.* /app/itc/minica/conf/
```

Создать .env файл:

```
$ sudo vim panel.env  
$ sudo touch jwt-panel.sh  
$ sudo vim jwt-panel.sh  
$ sudo chmod 755 jwt-panel.sh  
$ sudo ./jwt-panel.sh
```

Полученный токен и сертификаты скопировать на сервер minica-web-panel:

```
$ sudo scp /app/itc/minica/conf/panel.* administrator@minica-web.test.local:/home/administrator/  
$ sudo scp /app/itc/minica/conf/ca.crt administrator@minica-web.test.local:/home/administrator/
```

На сервере minica-web (panel) переместить файлы в папку certs

```
$ sudo mv panel.* /app/itc/itc/certs/  
$ sudo mv ca.crt /app/itc/itc/certs/  
$ sudo chown minica:minica -R /app/
```

Проверить запуск службы:

```
$ sudo systemctl start minica.spa.service  
$ sudo systemctl status minica.spa.service  
$ sudo journalctl -u minica.spa.service  
$ sudo tail -f /app/itc/itc/logs/itc_api_minica_controlPanel/file.log
```

### A.3.4 NGINX

```
$ sudo mkdir -p /app/itc/nginx/tls/  
$ sudo vim /app/itc/nginx/nginx.conf
```

```
$ sudo vim /etc/nginx/nginx.conf
$ sudo scp /app/itc/itc/certs/panel.key /app/itc/nginx/tls/tls.key
$ sudo scp /app/itc/itc/certs/panel.crt /app/itc/nginx/tls/tls.crt
```

Задать повторно на папку права от пользователя minica.

```
$ sudo chown minica:minica -R /app/
```

Далее перезапустить службу nginx и проверить ее статус:

```
$ sudo systemctl restart nginx
$ sudo systemctl status nginx
```

Проверить ошибки службы:

```
$ sudo journalctl -u nginx.service
$ sudo tail -f /var/log/nginx/access.log
$ sudo tail -f /var/log/nginx/error.log
```

## A.4 Настройка микросервиса mocsp

```
$ sudo dpkg -i к/файлу/mocsp-sfx-2.4.4-d48e974_amd64-2025.06.20.deb
```

или

```
$ sudo chmod 755 unzip mocsp-sfx-2.4.4-d48e974_amd64-2025.06.20.zip
$ sudo unzip mocsp-sfx-2.4.4-d48e974_amd64-2025.06.20.zip
$ sudo mkdir -p /app/itc/mocsp
$ sudo mkdir -p /app/itc/mocsp/logs/
$ sudo mv mocsp-sfx-2.4.4-d48e974/* /app/itc/mocsp/
```

Скопировать исполняемые файлы в корень

```
$ sudo scp /app/itc/mocsp/bin/* /app/itc/mocsp/
$ sudo vim /app/itc/mocsp/conf/mocsp.yml
$ sudo vim /lib/systemd/system/mocsp.service
$ sudo vim /app/itc/mocsp/mocsp.env
$ sudo systemctl daemon-reload
$ sudo systemctl enable mocsp.service
```

Создание TLS- ключа для службы mocsp с помощью запроса, выполнить команду:

```
$ sudo openssl req -new -keyout /app/itc/minica/ca/certs/ocsp.key -out  
/app/itc/minica/ca/certs/ocsp.req -sha256 -nodes -subj "/CN=minica-web.test.local" -addext  
"subjectAltName = DNS:minica-web.test.local " -addext "extendedKeyUsage = serverAuth,  
clientAuth"
```

Создание TLS- сертификата для mocsp с помощью запроса, выполнить команду:

```
$ sudo /app/itc/minica/mclient ca -csr /app/itc/minica/ca/certs/ocsp.req -out  
/app/itc/minica/ca/certs/ocsp.crt -template tls
```

Полученный сертификаты скопировать на сервер minica-web-panel:

```
$ sudo scp /app/itc/minica/ca/certs/tls.* administrator@minica-  
web.test.local:/home/administrator/  
$ sudo scp /app/itc/minica/ca/certs/ocsp.* administrator@minica-  
web.test.local:/home/administrator/  
$ sudo scp /app/itc/minica/ca/ca.crt administrator@minica-web.test.local:/home/administrator/
```

На сервере minica-web-panel переместить файлы в папку /conf

```
$ sudo mv ocsp.* /app/itc/mocsp/conf/  
$ sudo mv ca.crt /app/itc/mocsp/conf/  
$ sudo mv tls.* /app/itc/mocsp/conf/  
$ sudo chown minica:minica -R /app/
```

Проверить запуск службы:

```
$ sudo systemctl start mocsp.service  
$ sudo journalctl -u mocsp.service -o cat  
Проверить работу микросервиса командой:  
telnet 172.24.240.12 8888
```

#### **A.4.1 Перенастройка файла конфигурации mocsp.yml**

```
$ sudo ./maskpass -p 12345678 -k password-fraza  
eTsEl_4QwtmRAqv3nm-KAn1aFr9QzYIJQccNxliHwbEvs089  
$ sudo vim /app/itc/mocsp/conf/mocsp.yml
```

Зашифровать парольную фразу для БД:

```
$ sudo ./maskpass -p password-fraza  
utna4dVpB_BC5XgjkNL4EonxvtG6Bj3y8-NbecW_36whUD3zDO7QIUoj
```

```
$ sudo vim /app/itc/mocsp/conf/mocsp.yml
db-pass: utna4dVpB_BC5XgjkNL4EonxvtG6Bj3y8-NbecW_36whUD3zDO7QIUoj # 1234
Перезапустить службу mocsp
$ sudo systemctl restart mocsp.service
```

#### **A.4.2 Настройка точек распространения**

```
$ sudo apt install sshpass
```

Создать каталоги:

```
$ sudo mkdir -p /app/inetpub/pki/aia/ # Доступ к корневым и выдающим сертификата
$ sudo mkdir -p /app/inetpub/pki/cdp/ # Точки распространения CRL and DeltaCRL
```

Выдать права на каталоги:

```
$ sudo chmod 755 /app/inetpub/pki/aia/
$ sudo chmod 755 /app/inetpub/pki/cdp/
$ sudo chown minica:minica -R /app/inetpub
$ cd /app/itc/minica/
$ sudo vim pubcrl.sh
```

```
1: #!/bin/bash
2: $ sudo -u minica /app/itc/minica/mclient updcr1 -out
   /app/itc/minica/ca/crl/TESTSubCA.crl
3: sshpass -p "password" scp /app/itc/minica/ca/crl/*.crl minica@minica-
   web.test.local:/app/inetpub/pki/cdp/
```

```
$ sudo vim pubdeltacr1.sh
```

```
1: #!/bin/bash
2: $ sudo -u minica /app/itc/minica/mclient updeltacr1 -out
   /app/itc/minica/ca/crl/TESTSubCA+.crl
3: sshpass -p "password" scp /app/itc/minica/ca/crl/*+*.crl minica@minica-
   web.test.local:/app/inetpub/pki/cdp/
```

#### **A.4.3 Настройка скриптов распространения**

Примечание! Настройка для корневого или выдающего ЦС одинакова.

Чтобы скрипты отработывали корректно в службе ssh надо отключить проверку запроса на подключение к новым узлам. Открыть конфигурационный файл:

```
$ sudo vim /etc/ssh/ssh_config
```

Заменить строку

```
1: # StrictHostKeyChecking ask
```

на

```
1: StrictHostKeyChecking no
```

Выйти с сохранение и перезапустить службу:

```
$ sudo systemctl restart sshd
```

#### A.4.4 Автоматизация обновления CRL

Настроить планировщиком задач cron на скрипты, согласно регламентам обновления списков отозванных сертификатов CRL на объекте.

Создать файл в каталоге:

```
$ sudo vim /etc/cron.d/minica-crl
```

```
1: SHELL=/bin/sh
2: PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
3: sun,mon,tue,wed,thu,fri,sat
4: # | | | | |
5: # * * * * * user-name command to be executed
6: 0 22 * * 7 minica /app/itc/minica/pubcrl.sh
7: 0 21 * * * minica /app/itc/minica/pubdeltacrl.sh
```

Сохранить и перезапустить службу cron:

```
$ sudo systemctl restart cron
$ sudo systemctl status cron
```