Центр Управления Гетерогенными Инфраструктурами

Система централизованного анализа и управления гетерогенными инфраструктурами — MiniCA

Функциональные характеристики

ООО «Клируэй Текнолоджис»

Москва 2025

Оглавление

Перечень терминов и сокращений

Сокращение	Расшифровка	Описание
API	Application Programming Interface	Прикладной программный интерфейс компонента ИС.
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.
JWT	JSON Web Token	Открытый стандарт (RFC 7519), определяющий компактный и самодостаточный способ передачи информации между сторонами в виде JSON-объекта, подписанного цифровой подписью или зашифрованного.
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.
X.509v3		Версия международного стандарта, определяющая структуру цифровых сертификатов, содержащих информацию о субъекте (пользователе, устройстве или организации), открытый ключ субъекта и дополнительные расширения, используемые для настройки поведения сертификата и улучшения функциональности. Является основой современных методов аутентификации и защиты данных в интернет-коммуникациях, включая TLS/SSL и электронную почту.
БД	База данных	БД.
ИС	Информационная система	
ОС	Операционная система	
УЦ	Удостоверяющий Центр	Удостоверяющий Центр — программный или программно-аппаратный комплекс, обеспечивающий управление жизненным циклом сертификатов x509.

Сокращение	Расшифровка	Описание
ЦС (CA)	Центр Сертификации	Центр Сертификации (Certification Authority) — сервер, предназначенный для выпуска и отзыва сертификатов, а также публикации CRL (COC). Часть Удостоверяющего Центра.
HSM	Hardware Security Module	Специализированное устройство для генерации, хранения и управления криптографическими ключами.
KRA	Key Recovery Agent	Пользователь, имеющий специальный сертификат и права для резервного копирования и восстановления криптографических ключей и цифровых сертификатов.

Определения

Термин	Определение
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.

1. Введение

Настоящий документ относится к эксплуатационной документации ПО "Система централизованного анализа и управления гетерогенными инфраструктурами — MiniCA" (далее — MiniCA). Разработчиком ПО MiniCA является ООО «Клируэй Текнолоджис».

MiniCA — это центр сертификации, который выполняет выпуск сертификатов X.509v3 на основе CSR-запросов в формате PKCS#10, используя готовые шаблоны. Система позволяет создавать и настраивать эти шаблоны, а также отзыв сертификатов с указанием причины. MiniCA формирует списки отозванных сертификатов (CRL и DeltaCRL), предоставляет статус сертификатов через протокол OCSP и обеспечивает доступ к ним по серийному номеру или SHA1-отпечатку. Все запросы, сертификаты, журналы событий и CRL/DeltaCRL сохраняются в базе данных. Управление осуществляется через веб-интерфейс и API, а также поддерживается работа с протоколом SCEP для выпуска сертификатов.

ПО MiniCA является развитием одного из модулей ПО «Система централизованного анализа и управления гетерогенными инфраструктурами (ЦУГИ)», зарегистрированной в Реестре российского ПО (Реестровая запись №13033 от 21.03.2022), обладает расширенным составом функций и позволяет применяться в отдельно устанавливаемом исполнении.

2. Функции MiniCA

Основной функцией MiniCA является управление жизненным циклом сертификатов:

1) Выпуск сертификатов:
   - Выпуск сертификатов X.509v3 на основе CSR запросов в формате PKCS#10;
   - Поддержка шаблонов сертификатов для наполнения и контроля атрибутов сертификатов;
   - Поддержка выпуска сертификатов по различным протоколам – SCEP, WSTEP и т.п.
2) Предоставление сертификата по запросу:
   - Возможность получения выпущенных сертификатов по запросу;
   - Возможность получения различных выборок и отчетов о выпущенных сертификатах.
3) Отзыв сертификата:
   - Отзыв сертификатов с указанием причины;
   - Формирование Списка Отозванных Сертификатов (COC, CRL);
   - Формирование разностных списков отозванных сертификатов DeltaCRL;
   - Предоставление информации о статусе сертификата по протоколу OCSP.
4) Вспомогательные функции:
   - Сохранение информации о всех запросах, сертификатах, CRL/DeltaCRL в базе данных;
   - Текстовый журнал событий;
   - Веб-интерфейс с поддержкой ролевой модели;
   - API-интерфейс с обеспечением авторизации и разделения ролей;
   - Поддержка функций самодиагностики;
   - Поддержка хранения ключа ЦС на аппаратных носителях HSM.
5) Выпуск сертификатов:
   - Выпуск сертификатов X.509v3 на основе CSR запросов в формате PKCS#10;
   - Поддержка шаблонов сертификатов для наполнения и контроля атрибутов сертификатов.
6) Отзыв сертификата:
   - Отзыв сертификатов с указанием причины;
   - Возвращение в действие сертификата (Unrevoke);
   - Формирование Списка Отозванных Сертификатов (COC, CRL);
   - Формирование разностных списков отозванных сертификатов DeltaCRL;
   - Предоставление информации о статусе сертификата по протоколу OCSP.
7) Предоставление информации о сертификатах по запросу:

- Возможность получения выпущенных сертификатов по запросу;
- Возможность массовой выгрузки сертификатов;
- Возможность получения различных выборок и отчетов о выпущенных сертификатах.

8) Дополнительные функции:
   - Поддержка протоколов запроса сертификатов:
     - Поддержка выпуска сертификатов по протоколу SCEP;
     - Поддержка выпуска сертификатов по протоколу MS-WSTEP;
     - Поддержка выпуска сертификатов по протоколам ACMEv2, MS-RPC, CMP, EST.
   - Интерфейс пользователя
     - Интерфейс командной строки;
     - Веб-интерфейс;
     - API-интерфейс.
   - Журналирование
     - Журнал событий MiniCA в базе данных;
     - Текстовый журнал событий.
   - Мониторинг
     - Поддержка функций самодиагностики;
     - Оповещения о событиях жизненного цикла сертификатов.
   - Безопасность
     - Разграничение полномочий пользователей к функциям UI (ролевая модель);
     - Разграничение полномочий пользователей к функциям MiniCA;
     - Защищенное хранения закрытых ключей в БД ЦС и управление ими с помощью Агента Восстановления Ключей (KRA);
     - Поддержка хранения ключа ЦС на аппаратных носителях HSM.
   - Взаимодействие с другими системами:
     - Публикация CRL на внешние серверы;
     - Публикация сертификатов и CRL в объекты LDAP Microsoft Active Directory;
     - Публикация сертификатов и CRL в объекты LDAP Samba;
     - Интеграция с системой KeyBox.
   - Функции обслуживания:
     - Резервное копирование;
     - Архивирование истекших сертификатов.

В следующих разделах более подробно описаны выполняемые функции и задачи.

3. Функции системы

3.1. Выпуск сертификатов

В рамках функции выпуска сертификатов MiniCA обеспечивает формирование сертификата на основе информации, поступающей от пользователя. Описанные ниже функции определяют способы получения и обработки входной информации для выпуска сертификатов.

Компонент Название	Краткое описание	Доступ к функции через командную строку (mclient)	Доступ к функции с помощью веб-интерфейса
Выпуск сертификатов X.509v3 на основе CSR запросов в формате PKCS#10	Подпись запроса CSR в формате PKCS#10 сертификатом ЦС. При формировании сертификата в него добавляется ряд расширений: • EKU; • CDP; • AIA • и т.п. Сертификат формируется с использованием пакета OpenSSL. Сертификат записывается в базу данных и выкладывается в указанную папку, в зависимости от настроек.	Для выпуска сертификата с помощью командной строки используется утилита mclient. В типовой команде выпуска сертификата указывается путь к файлу запроса, имя шаблона и путь к сертификату. ./mclient ca -csr test.req -template tls -out test.crt	Типовой пользовательский путь: • Нажать кнопку "Запрос сертификата"; • Вставить в окно текст запроса; • Указать шаблон; Нажать кнопку "Подтвердить".
Поддержка шаблонов сертификатов для наполнения и контроля атрибутов сертификатов	Шаблоны сертификатов необходимы для следующих целей: • Контроль состава атрибутов в запросе; • Упрощение создания запроса путем указания части атрибутов в шаблоне. Шаблоны хранятся в БД ЦС и содержат наиболее часто изменяемые	Для управления шаблонами имеются следующие команды: • Команда на создание шаблона ./mclient addtempl; • Команда на удаление шаблона ./mclient deltempl;	Использование шаблона при выпуске сертификата описано в предыдущем пункте. Для просмотра, создания или редактирования шаблонов нужно выбрать вкладку "Шаблоны". Для просмотра и редактирования выбрать нужный шаблон из списка. Для создания нового

Компонент Название	Краткое описание	Доступ к функции через командную строку (mclient)	Доступ к функции с помощью веб-интерфейса
	атрибуты. Остальные атрибуты, явно неуказанные в запросе и шаблоне, заполняются параметрами по умолчанию OpenSSL. Шаблоны имеют разрешения на использование и изменение. Шаблоны являются частью процедуры запроса и создания сертификата.	• Команда вывода параметров шаблона ./mclient gettempl; • Команда вывода списка шаблонов ./mclient templates.	шаблона нужно нажать кнопку "Создать шаблон".

3.2. Отзыв сертификата

Компонент Название	Краткое описание	Доступ к функции через командную строку (mclient)	Доступ к функции с помощью веб-интерфейса
Отзыв сертификатов с указанием причины (Revoke)	При отзыве сертификата он помечается в БД статусом Revoked. Добавляется в CRL при следующем его формировании. Отозванные сертификаты не должны участвовать в безопасном взаимодействии. Проверка статуса сертификата и соответствующая его обработка выполняется клиентом, использующим сертификат.	Отзыв осуществляется командой ./mclient revoke	Отзыв сертификата осуществляется из его карточки или прямо из списка сертификатов. При отзыве необходимо указать причину.

Компонент Название	Краткое описание	Доступ к функции через командную строку (mclient)	Доступ к функции с помощью веб-интерфейса
Возвращение в действие сертификата (Unrevoke)	Сертификаты, отозванные с причиной Certificate Hold, могут быть возвращены в действие и изъяты из файлов CRL.	Возвращение в действие сертификата осуществляется командой ./mclient repair	Для возвращения в действие нужно выбрать соответствующий сертификат в списке и нажать кнопку "Восстановить выбранные сертификаты".
Формирование Списка Отозванных Сертификатов (COC, CRL)	При формировании Списка Отозванных Сертификатов (COC, CRL) в него помещаются серийные номера всех отозванных сертификатов. CRL формируется в БД ЦС, а также может быть помещен в файл по указанному пути.	Формирование CRL выполняется командой ./mclient updcrl	На вкладке "CRL" в меню "Действия" выбрать пункт "Обновить CRL".
Формирование разностных списков отозванных сертификатов DeltaCRL	При формирование разностных списков отозванных сертификатов DeltaCRL в него помещаются серийные номера всех сертификатов, отозванных с момента публикации последнего полного CRL. Формирование DeltaCRL позволяет снизить нагрузку на сеть и вычислительные ресурсы, особенно в случае большого объема отозванных сертификатов в полном CRL.	Формирование deltaCRL выполняется командой ./mclient updeltacrl	На вкладке "CRL" в меню "Действия" выбрать пункт "Обновить DeltaCRL".

Компонент Название	Краткое описание	Доступ к функции через командную строку (mclient)	Доступ к функции с помощью веб-интерфейса
Предоставление информации о статусе сертификата по протоколу OCSP	Отдельный микросервис mOCSP принимает запросы от клиентов и возвращает в ответ статус сертификатов. Определение статуса происходит в режиме реального времени при помощи прямого обращения к БД ЦС.	Интерфейс пользователя отсутствует	Интерфейс пользователя отсутствует

3.3. Предоставление информации о сертификатах по запросу

Компонент Название	Краткое описание	Доступ к функции через командную строку (mclient)	Доступ к функции с помощью веб-интерфейса
Возможность получения выпущенных сертификатов по запросу	Пользователи могут запросить как сам выпущенный сертификат из БД ЦС, так и сопутствующую информацию по нему.	Для запроса информации по сертификатам используются команды ./mclient get ./mclient status	• В списке сертификатов выбрать нужную строку; • Зайти в карточку сертификата; • Для получения сертификата в виде файла нажать кнопку "Скачать сертификат".
Возможность массовой выгрузки сертификатов	Массовая выгрузка сертификатов используется для синхронизации БД ЦС с другими системами.	Команда для массовой выгрузки ./mclient export	• В списке сертификатов выбрать нужные строки, отметив их галочкой; • Нажать на кнопку "Скачать сертификат".

Компонент Название	Краткое описание	Доступ к функции через командную строку (mclient)	Доступ к функции с помощью веб-интерфейса
Возможность получения различных выборок и отчетов о выпущенных сертификатах	Пользователь может сформировать выборку сертификатов с определенными параметрами и выгрузить ее в файл.	Интерфейс командной строки отсутствует.	• В списке сертификатов (а также запросов и событий) сформировать выборку при помощи инструментов поиска и фильтрации; • Нажать на кнопку выгрузки файла.

3.4. Дополнительные функции

3.4.1. Интерфейс пользователя

Компонент Название	Краткое описание	Доступ к функции через командную строку (mclient)	Доступ к функции с помощью веб-интерфейса
Интерфейс командной строки	Программа mclient, позволяющая управлять функциями MiniCA при помощи интерфейса командной строки. При помощи входных параметров, задаваемых при запуске программы		
Веб-интерфейс (контрольная панель)	Веб-интерфейс обеспечивает удобное и наглядное представление информации и интуитивно понятный доступ к функциям ЦС. Повышает комфорт работы и снижает требования к квалификации пользователя.	Интерфейс командной строки отсутствует.	Доступ к порталу веб-интерфейса с помощью браузера

Компонент Название	Краткое описание	Доступ к функции через командную строку (mclient)	Доступ к функции с помощью веб-интерфейса
API-интерфейс	API-интерфейс обеспечивает возможность подключения к ЦС при помощи сторонних приложений.	Все команды mclient используют API вызовы.	Все графические элементы управления используют API вызовы

3.4.2. Поддержка протоколов запроса сертификатов

Компонент Название	Краткое описание	Доступ к функции через командную строку (mclient)	Доступ к функции с помощью веб-интерфейса
Поддержка выпуска сертификатов по протоколу SCEP	Отдельный микросервис uSCEP принимает запросы от клиентов по протоколу SCEP и перенаправляет их в MiniCA. В ответ получает сертификат и возвращает его клиенту.	Для тестирования запросов по протоколу SCEP существует отдельная утилита uScepClient. С ее помощью можно проверить работоспособность сервиса SCEP.	Графический интерфейс пользователя отсутствует. Сторонние SCEP-клиенты имею свой собственный интерфейс.
Поддержка выпуска сертификатов по протоколу MS-WSTEP	Отдельный микросервис принимает запросы от Windows-клиентов по протоколу MS-WSTEP и перенаправляет их в MiniCA. В ответ получает сертификат и возвращает его клиенту. Необходима предварительно настроенная инфраструктура Microsoft Active Directory.	Интерфейс командной строки отсутствует.	Веб-интерфейс пользователя отсутствует. Для настройки инфраструктуры и запросов сертификатов используются оснастки, встроенные в ОС Windows.

3.4.3. Журналирование

Компонент Название	Краткое описание	Доступ к функции через командную строку (mclient)	Доступ к функции с помощью веб-интерфейса
Журнал событий MiniCA в базе данных	В БД ЦС сохраняется информация о запросах на сертификаты, а также всех действиях, выполненных MiniCA для управления жизненным циклом сертификатов.	Возможен доступ из командной строки к таблице в БД с помощью утилиты Postgres	Вкладка "Запросы" Вкладка "События"
Текстовый журнал событий	Текстовый журнал событий сохраняет все действия, выполняемые MiniCA. Уровень журналирования определяется в настройках системы. По умолчанию, файл журнала находится по пути ./logs/MiniCA.log.	стандартные команды ОС - cat, tail и т.п.	Нет графического интерфейса

3.4.4. Мониторинг

Компонент Название	Краткое описание	Доступ к функции через командную строку (mclient)	Доступ к функции с помощью веб-интерфейса
Поддержка функций самодиагностики	Диагностические команды выводят статистику по основным параметрам ЦС. Предоставляют информацию в ответ на API запросы	Для запроса диагностической информации используются команды ./mclient ping ./mclient stat	На вкладке "Настройка" представлены некоторые параметры

Компонент Название	Краткое описание	Доступ к функции через командную строку (mclient)	Доступ к функции с помощью веб-интерфейса
Оповещения о событиях жизненного цикла сертификатов	При запросе сертификата можно указать события жизненного цикла, при наступлении которых пользователь будет получать уведомления: • Выпуск сертификата; • Отзыв сертификата; • Истечение срока действия сертификата.	Интерфейс командной строки отсутствует.	В диалоге выпуска сертификата можно добавить адреса электронной почты и указать события.

3.4.5. Безопасность

Компонент Название	Краткое описание	Доступ к функции через командную строку (mclient)	Доступ к функции с помощью веб-интерфейса
Разграничение полномочий пользователей к функциям UI (ролевая модель)	Сертификаты являются доверенной сущностью в информационной инфраструктуре, применяются в том числе для аутентификации и авторизации. Поэтому разграничение доступа к функциям выпуска сертификатов и изменению конфигурации ЦС является важной частью обеспечения ИБ. Ролевая модель позволяет предоставить пользователям только те функции, которые необходимы им для выполнения рабочего процесса. Ролевая модель основана на JWT-токенах, которые предоставляет KeyCloak.	Интерфейс командной строки отсутствует.	Графический интерфейс к настройкам ролей предоставляет KeyCloak.

Компонент Название	Краткое описание	Доступ к функции через командную строку (mclient)	Доступ к функции с помощью веб-интерфейса
Разграничение полномочий пользователей к функциям MiniCA	Ролевая модель MiniCA основана на JWT токенах. Ответственность за формирование токена лежит на стороне клиента.		Графический интерфейс отсутствует.
Защищенное хранение закрытых ключей в БД ЦС и управление ими с помощью Агента Восстановления Ключей (KRA);	Хранение в БД ЦС закрытых ключей сертификатов в зашифрованном виде. Шифрование производится при помощи сертификата Агента Восстановления Ключей (KRA). Функция выгрузки сертификата с закрытым ключом в формате PFX.		
Поддержка хранения ключа ЦС на аппаратных носителях HSM	Хранение сертификата и ключа ЦС с помощью аппаратных модулей HSM значительно повышает безопасность ЦС, поскольку ключ невозможно не санкционированно извлечь даже при наличии физического доступа к модулю. Функция настраивается при установке ЦС с помощью конфигурационных файлов. Операции с ключами в дальнейшем осуществляются при помощи специализированного ПО от производителя HSM.	Функция настраивается при установке ЦС с помощью конфигурационных файлов. Интерфейс командной строки к функциям HSM предоставляет программное обеспечение производителя.	Функция не имеет веб-интерфейса, предоставляемого MiniCA

3.4.6. Взаимодействие с другими системами

Компонент Название	Краткое описание	Доступ к функции через командную строку (mclient)	Доступ к функции с помощью веб-интерфейса
Публикация CRL на внешние серверы	Публикация CRL и сертификатов ЦС по расписанию на внешние серверы, в том числе находящиеся за пределами ЛВС.	Интерфейс командной строки отсутствует.	Графический интерфейс к настройкам ролей предоставляет Keycloak
Публикация сертификатов и CRL в объекты LDAP Microsoft Active Directory	Публикация сертификатов и CRL в объекты LDAP Microsoft Active Directory по расписанию или по требованию.		Графический интерфейс отсутствует.
Публикация сертификатов и CRL в объекты LDAP Samba	Публикация сертификатов и CRL в объекты LDAP Samba по расписанию или по требованию.		
Интеграция с системой KeyBox	Сервис, позволяющий системе KeyBox управлять функциями MiniCA.	Функция настраивается при установке ЦС с помощью конфигурационных файлов. Интерфейс командной строки к функциям HSM предоставляет программное обеспечение производителя.	Функция не имеет веб-интерфейса, предоставляемого MiniCA

3.4.7. Функции обслуживания

Компонент Название	Краткое описание	Доступ к функции через командную строку (mclient)	Доступ к функции с помощью веб-интерфейса
Резервное копирование	Выполнение резервного копирования при помощи команд MiniCA.	Для выполения резервного копирования используются команды ./mclient backup	
Архивирование истекших сертификатов	Удаление из основной БД истекших сертификатов с заданной глубиной по дате.		

3.4.8. API вызовов

3.4.8.1. Проверка связи с сервисом

Метод: POST
Endpoint: {BASE}/ping
{BASE} — базовая часть URL, например «https://MiniCA.some.domain»

JSON запрос:

```
{
 «action» : «ping»,
 «time»   : «...», // текущее время в формате RFC3339
}
```

JSON ответ:

```
{
 «errno»   : 0,          // код ошибки (0 - успех)
 «error»   : «...»,      // текст ошибки (в случае успеха пустая строка)
 reply     : «pong»,     // всегда "pong"
 version   : «1.x.y-hash», // версия MiniCA
 «message» : «Welcome to MiniCA», // сообщение приветствия
 «ski»     : «7544986F0F52...»    // текущий SKI ЦС (hex)
}
```

HTTP ответ:

1) 4xx, 5xx — ошибка (см. errno и error);
2) 200 — успех (errno=0).

3.4.8.2. Запрос на выпуск сертификата

Метод: POST
Endpoint: {BASE}/ca

JSON запрос:

```
{
 «action»    : «ca»,
 «time»      : «...», // текущее время в формате RFC3339
 «csr»       : «...», // CSR запрос (PKCS#10) в формате PEM
 «requester» : «...», // тот, кто запросил сертификат (user/host)
 «serial»    : «...», // серийный номер нового сертификата HEX (или пусто)
 «days»      : 366,   // запрашиваемый срок действия сертификата (дней) или 0
 «short»     : false, // признак «короткоживущего» сертификата
 «template»  : «...», // имя заданного шаблона сертификата
 «no-aia»    : false, // отключить расширение AIA в сертификате (bool)
 «no-cdp»    : false, // отключить расширение CDP в сертификате (bool)
 «context»   : «...», // дополнительные данные
 «email»     : «...», // адрес электронной почты (для системы уведомлений)
 «flags»     : 0      // битовые флаги (опционально) для внешних систем
```

```
}
```

JSON ответ:

```
{
  «errno»        : 0,     // код ошибки (0 - успех)
  «error»        : «...», // текст ошибки (в случае успеха пустая
строка)
  «crypto-error» : «..»,  // ошибка криптопровайдера (OpenSSL)
  «id»           : 1235,  // идентификатор сертификата в базе данных
(int64)
  «serial»       : «...», // серийный номер сертификата (назначенный)
  «fingerprint»  : «...», // отпечаток сертификата (HEX)
  «requester»    : «...», // тот, кто запросил сертификат (user/host)
  «template»     : «...», // шаблон сертификата
  «short»        : true,  // признак «короткоживущего» сертификата
(если «да»)
  «crt»          : «...»  // сертификат x509v3 в формате PEM
}
```

HTTP ответ:

1) 4xx, 5xx — ошибка (см. errno и error);
2) 200 — успех, сертификат выпущен (errno=0).

3.4.8.3. Отзыв сертификата

Метод: POST
Endpoint: {BASE}/revoke

JSON запрос:

```
{
  «action»      : «revoke»,
  «time»        : «...», // текущее время в формате RFC3339
  «id»          : 1235,  // идентификатор сертификата в БД (int64) или
0
```

```
  «serial»      : «...», // серийный номер сертификата в базе данного
ЦС
  «fingerprint» : «...», // отпечаток сертификата (HEX)
  «requester»   : «...», // владелец сертификата (user/host)
  «reason»      : «...», // причина отзыва сертификата (сокращенно)
  «comment»     : «...», // подробная причина отзыва (UTF8),
комментарий
  «context»     : «...»  // дополнительные данные от ЕСАУС (JSON)
}
```

Должен быть задан или id, serial, fingerprint или requester (что-то одно).
Если указан requester, то отзывается последний по порядку в таблице БД сертификат.

Поле «reason» должно принимать одно из 8-ми допустимых значений:

1) «unspecified»;
2) «keyCompromise»;
3) «CACompromise»;
4) «affiliationChanged»;
5) «superseded»;
6) «cessationOfOperation»;
7) «certificateHold»;
8) «removeFromCRL».

Согласно руководству по OpenSSL, данные строки не чувствительны к регистру. Однако рекомендуется использовать указанные идентификаторы для однообразия.

JSON ответ:

```
{
  «errno»         : 0,     // код ошибки (0 - успех)
  «error»         : «...», // текст ошибки (в случае успеха пустая
строка)
  «crypto-error»  : «..»,  // ошибка криптопровайдера (OpenSSL)
  «id»            : 1235   // идентификатор сертификата в базе данных
(int64)
  «serial»        : «...», // серийный номер сертификата
  «fingerprint»   : «...», // отпечаток сертификата (HEX)
  «requester»     : «...», // владелец сертификата (user/host)
```

```
  «status»        : «...», // статус (VALID, EXPIRED, REVOKED,
UNKNOWN)
  «revoke-reason» : «...», // причина отзыва сертификата
  «template»      : «...»  // шаблон сертификата
}
```

HTTP ответ:

1) 4xx, 5xx — ошибка (см. errno и error);
2) 200 — успех, сертификат отозван (errno=0).

3.4.8.4. Запрос статуса сертификата

Метод: POST
Endpoint: {BASE}/status

JSON запрос:

```
{
  «action»      : «status»,
  «time»        : «...», // текущее время в формате RFC3339
  «id»          : 1235,  // идентификатор сертификата в БД (int64) или
0
  «serial»      : «...», // серийный номер сертификата в базе данного
ЦС
  «fingerprint» : «...», // отпечаток сертификата (HEX)
  «requester»   : «...»  // тот, кто запросил сертификат (user/host)
}
```

Должен быть задан или id, serial, fingerprint или requester (что-то одно).

Если указан requester, то возвращается статус последнего по порядку в таблице БД сертификата.

JSON ответ:

```
{
  «errno»         : 0,     // код ошибки (0 - успех)
```

```
  «error»         : «...», // текст ошибки (в случае успеха пустая
строка)
  «id»            : 1235,  // идентификатор сертификата в базе данных
(int64)
  «serial»        : «...», // серийный номер сертификата в базе
данного ЦС
  «fingerprint»   : «...», // отпечаток сертификата (HEX)
  «requester»     : «...», // тот, кто запросил сертификат (user/host)
  «status»        : «...», // статус (VALID, EXPIRED, REVOKED,
UNKNOWN)
  «revoke-reason» : «...»  // причина отзыва сертификата
}
```

HTTP ответ:

1) 4xx, 5xx — ошибка (см. errno и error);
2) 200 — успех, статус сертификата определен (errno=0).

3.4.8.5. Запрос сертификата из БД

Метод: POST
Endpoint: {BASE}/get

JSON запрос:

```
{
  «action»      : «get»,
  «time»        : «...», // текущее время в формате RFC3339
  «id»          : 1235,  // идентификатор сертификата в БД (int64) или
0
  «serial»      : «...», // серийный номер сертификата в базе данного
ЦС
  «fingerprint» : «...», // отпечаток сертификата (HEX) или пустая
строка
  «requester»   : «...»  // тот, кто запросил сертификат (user/host)
}
```

Должен быть задан или id, serial, fingerprint или requester (что-то одно).
Если указан requester, то возвращается последний по порядку в таблице БД сертификат.

JSON ответ:

```
{
  «errno»         : 0,     // код ошибки (0 - успех)
  «error»         : «...», // текст ошибки (в случае успеха пустая
строка)
  «id»            : 1235,  // идентификатор сертификата в базе данных
(int64)
  «serial»        : «...», // серийный номер сертификата в базе
данного ЦС
  «fingerprint»   : «...», // отпечаток сертификата (HEX)
  «requester»     : «...», // тот, кто запросил сертификат (user/host)
  «status»        : «...», // статус (VALID, EXPIRED, REVOKED,
UNKNOWN)
  «revoke-reason» : «...», // причина отзыва сертификата
  «template»      : «...», // имя шаблона сертификата (ASCII)
  «crt»           : «...», // сертификат в формате PEM
  «email»         : «...», // адрес электронной почты (для системы
уведомлений)
  «flags»         : 0      // битовые флаги (опционально) для внешних
систем
}
```

HTTP ответ:

1) 4xx, 5xx — ошибка (см. errno и error);
2) 200 — успех, запрашиваемый сертификат предоставлен (errno=0).

3.4.8.6. Обновить CRL

Метод: POST
Endpoint: {BASE}/updcrl

JSON запрос:

```
{
  «action»       : «updcrl»,
  «time»         : «...», // текущее время в формате RFC3339
  «days»         : 0,     // срок действия CRL (0 – использовать
умолчания)
  «last-update»  : «...», // время обновления или пустая строка
(YYMMDDHHMMSSZ)
  «next-update»  : «...», // время следующего обновления CRL или
пустая строка
  «next-publish» : «...», // время следующей публикации CRL или пустая
строка
  «aki»          : «...»  // SKI ЦС (если пусто, то последний)
}
```

Сервис проверяет метку времени (допустимое расхождение +/- 5 минут). Метка времени требуется как «соль» для возможности формирования уникальной цифровой подписи, если last-update и next-update не заданы.

JSON ответ:

```
{
  «errno»        : 0,      // код ошибки (0 – успех)
  «crypto-error» : «..»,   // ошибка криптопровайдера (OpenSSL)
  «error»        : «...»,  // текст ошибки (в случае успеха пустая
строка)
  «id»           : 1235,   // идентификатор CRL в базе данных (int64)
  «number»       : «1001», // номер CRL в нотации OpenSSL
  «last-update»  : «...»,  // время обновления CRL (RFC3339)
  «next-update»  : «...»,  // время следующего обновления CRL
(RFC3339)
  «revoked»      : 7,      // число отозванных сертификатов
  «crl»          : «...»   // CRL в формате PEM
}
```

HTTP ответ:

1) 4xx, 5xx — ошибка (см. errno и error);
2) 200 — успех, запрашиваемый CRL предоставлен (errno=0).

3.4.8.7. Запрос CRL

Метод: POST
Endpoint: {BASE}/crl

JSON запрос:

```
{
  «action» ; «crl»,
  «time»   : «...», // текущее время в формате RFC3339
  «aki»    : «...»  // SKI ЦС (если пусто, то последний)
}
```

Сервис проверяет метку времени (допустимое расхождение +/- 5 минут). Метка времени требуется как «соль» для возможности формирования уникальной цифровой подписи.

JSON ответ:

```
{
  «errno»       : 0,      // код ошибки (0 - успех)
  «error»       : «...»,  // текст ошибки (в случае успеха пустая
строка)
  «id»          : 1235,   // идентификатор CRL в базе данных (int64)
  «number»      : «1001», // номер CRL в нотации OpenSSL
  «last-update» : «...»,  // время обновления CRL (RFC3339)
  «next-update» : «...»,  // время следующего обновления CRL (RFC3339)
  «revoked»     : 3,      // число отозванных сертификатов
  «crl»         : «...»   // CRL в формате PEM
}
```

HTTP ответ:

4xx, 5xx — ошибка (см. errno и error);

200 — успех, запрашиваемый CRL предоставлен (errno=0).

3.4.8.8. Обновить DeltaCRL

Метод: POST
Endpoint: {BASE}/updeltacrl

JSON запрос:

```
{
  «action»       : «updeltacrl»,
  «time»         : «...», // текущее время в формате RFC3339
  «days»         : 0,     // срок действия DeltaCRL (0 – испол.
умолчания)
  «last-update»  : «...», // время обновления или пустая строка
(YYMMDDHHMMSSZ)
  «next-update»  : «...», // время следующего обновления CRL или
пустая строка
  «next-publish» : «...», // время следующей публикации CRL или пустая
строка
  «aki»          : «...»  // SKI ЦС (если пусто, то последний)
}
```

Сервис проверяет метку времени (допустимое расхождение +/- 5 минут). Метка времени требуется как «соль» для возможности формирования уникальной цифровой подписи, если last-update и next-update не заданы.

JSON ответ:

```
{
  «errno»        : 0,      // код ошибки (0 - успех)
  «error»        : «...»,  // текст ошибки (в случае успеха пустая
строка)
  «crypto-error» : «..»,   // ошибка криптопровайдера (OpenSSL)
  «id»           : 1235,   // идентификатор CRL в базе данных (int64)
  «number»       : «1001», // номер CRL в нотации OpenSSL
  «last-update»  : «...»,  // время обновления DeltaCRL (RFC3339)
  «next-update»  : «...»,  // время следующего обновления DeltaCRL
(RFC3339)
  «revoked»      : 7,      // число отозванных сертификатов (в
DeltaCRL)
  «delta-crl»    : «...»   // DeltaCRL в формате PEM
}
```

HTTP ответ:

1) 4xx, 5xx — ошибка (см. errno и error);
2) 200 — успех, запрашиваемый CRL предоставлен (errno=0).

3.4.8.9. Запрос DeltaCRL

Метод: POST
Endpoint: {BASE}/deltacrl

JSON запрос:

```
{
  «action» : «deltacrl»,
  «time»   : «...», // текущее время в формате RFC3339
  «aki»    : «...»  // SKI ЦС (если пусто, то последний)
}
```

Сервис проверяет метку времени (допустимое расхождение +/- 5 минут). Метка времени требуется как «соль» для возможности формирования уникальной цифровой подписи.

JSON ответ:

```
{
  «errno»       : 0,      // код ошибки (0 - успех)
  «error»       : «...»,  // текст ошибки (в случае успеха пустая
строка)
  «id»          : 1235,   // идентификатор CRL в базе данных (int64)
  «number»      : «1001», // номер CRL в нотации OpenSSL
  «last-update» : «...»,  // время обновления DeltaCRL (RFC3339)
  «next-update» : «...»,  // время следующего обновления DeltaCRL
(RFC3339)
  «revoked»     : 3,      // число отозванных сертификатов (в
DeltaCRL)
  «delta-crl»   : «...»   // DeltaCRL в формате PEM
}
```

HTTP ответ:

1) 4xx, 5xx — ошибка (см. errno и error);
2) 200 — успех, запрашиваемый CRL предоставлен (errno=0).

3.4.8.10. Поиск сертификатов в базе данных ЦС

Поиск сертификатов по серийному номеру (Serial), по SKI (Subject Key Identifier), или по отпечатку (Fingerprint), по Subject, или по CN (Common Name). Должно быть задано что-то одно.

Поле «status» может использоваться как фильтр.

Метод: POST

Endpoint: {BASE}/find

JSON запрос:

```
{
  «action» : «find»,
  «time»   : «...», // текущее время в формате RFC3339
  «status» : «...», // статус фильтр (VALID, EXPIRED, REVOKED, UNKNOWN
или «»)
  «id»     : 1235,  // идентификатор сертификата в базе данных (int64)
или 0
  «serial» : «...», // серийный номер сертификата в базе данного ЦС
или «»
 «fingerprint» : «...»,      // отпечаток сертификата (HEX) или «»
  «requester»   : «...»,          // владелец сертификата (user/host)
или «»
  «ski»    : «A4871BAE29...»,     // Subject Key Identifier или пустая
строка
  «subj»   : «/CN=servev.my.org», // Subject или пустая строка
  «cn»     : «user@my.org»,       // Common Name или пустая строка
  «offset» : 500, // смещение в SQL запросе
  «num»    : 500  // максимальное число (LIMIT) сертификатов в ответе
}
```

JSON ответ:

```
{
  «errno» : 0,     // код ошибки (0 - успех)
  «error» : «...», // текст ошибки (в случае успеха пустая строка)
  «count» : 1,     // общее число найденных сертификатов
  «certs» : [      // сведения о найденных сертификатах (не более num)
    {
      «id»            : 1235,  // идентификатор сертификата в БД
(int64)
      «serial»        : «...», // серийный номер сертификата в базе
   «fingerprint» : «...», // отпечаток сертификата (HEX)
      «requester»     : «...», // владелец сертификата (user/host)
      «status»        : «...», // статус (VALID, EXPIRED, REVOKED,
UNKNOWN)
      «revoke-reason» : «...», // причина отзыва для отозванного
сертификата
      «subj»       : «/CN=servev.my.org», // Subject
      «key-type»   : «RSA»,               // тип ключа
      «key-size»   : 2048,                // размер ключа
      «cn»         : «user@my.org»,       // Common Name
      «valid-from» : «...»,               // начало срока действия
      «valid-to»   : «...»,               // конец срока действия
      «ski»        : «A4871BAE29...»,     // Subject Key Identifier
      «bc»         : «...»,               // Basic Constraints
      «ku»         : «...»,               // Key Usage (CSV)
      «eku»        : «...»,               // Extended Key Usage (CSV)
      «san-dns»    : «...»,               // SAN DNS (CSV)
      «san-ip»    : «...»,                // SAN IP (CSV)
      «san-uri»   : «...»,                // SAN URIs (CSV)
      «san-email» : «...»,                // SAN email (CSV)
      «san-oname» : «...»,                // SAN otherName (CSV)
      «template»  : «...»,                // шаблон сертификата
(ASCII)
      «crt»       : «...»,                // сертификат x509v3 в
формате PEM
      «email»     : «...», // адрес электронной почты
```

```
      «flags»      : 0      // битовые флаги (опционально) для внешних
систем
    }
  ]
}
```

HTTP ответ:

1) 4xx, 5xx — ошибка (см. errno и error);
2) 200 — успех, сведения о найденных/не найденных сертификатах предоставлены (errno=0).

3.4.8.11. Удаление сертификата и соответствующего CSR из БД

Метод: POST
Endpoint: {BASE}/delete

JSON запрос:

```
{
  «action»      : «delete»,
  «time»        : «...», // текущее время в формате RFC3339
  «id»          : 1235,  // идентификатор сертификата в базе данных
(int64)
  «serial»      : «...», // серийный номер сертификата в базе данного
ЦС
 «fingerprint» : «...», // отпечаток сертификата (HEX)
  «context»     : «...»  // дополнительные данные от ЕСАУС (JSON)
}
```

Должен быть задан id, serial, fingerprint или requester (что-то одно).
Если указан requester, то удаляется последний по порядку в таблице БД сертификат.

JSON ответ:

```
{
  «errno»         : 0,     // код ошибки (0 - успех)
```

```
  «error»         : «...», // текст ошибки (в случае успеха пустая
строка)
  «id»            : 1235,  // идентификатор сертификата в базе данных
(int64)
  «serial»        : «...», // серийный номер сертификата в базе
данного ЦС
  «fingerprint»   : «...», // отпечаток сертификата (HEX)
  «requester»     : «...», // тот, кто запросил сертификат (user/host)
  «status»        : «...», // статус (VALID, EXPIRED, REVOKED,
UNKNOWN)
  «revoke-reason» : «...»  // причина отзыва сертификата
  «crt»           : «...»  // сертификат x509v3 в формате PEM
}
```

HTTP ответ:

1) 4xx, 5xx — ошибка (см. errno и error);
2) 200 — успех, сертификат и CSR запросы из БД удалены (errno=0).

3.4.8.12. Экспорт группы сертификатов из БД

Метод: POST
Endpoint: {BASE}/export

JSON запрос:

```
{
  «action» : «export»,
  «time»   : «...», // текущее время в формате RFC3339
  «status» : «...», // статус фильтр (VALID, EXPIRED, REVOKED, UNKNOWN
или «»)
  «id»     : 1235,  // начальный идентификатор сертификата в БД
(int64)
  «offset» : 500,   // смещение в SQL запросе
  «num»    : 500    // максимальное число (LIMIT) сертификатов в
запросе
}
```

JSON ответ:

```
{
  «errno»  : 0,     // код ошибки (0 - успех)
  «error»  : «...», // текст ошибки (в случае успеха пустая строка)
  «certs» : [       // сертификаты (массив)
    {
      «id»            : 1235,  // идентификатор сертификата в БД
(int64)
      «serial»        : «...», // серийный номер сертификата в базе
   «fingerprint»  : «...», // отпечаток сертификата (HEX)
      «requehster»    : «...», // владелец сертификата (user/host)
      «status»        : «...», // статус (VALID, EXPIRED, REVOKED,
UNKNOWN)
      «revoke-reason» : «...», // причина отзыва для отозванного
сертификата
      «subj»       : «/CN=servev.my.org», // Subject
      «key-type»   : «RSA»,               // тип ключа
      «key-size»   : 2048,                // размер ключа
      «cn»         : «user@my.org»,       // Common Name
      «valid-from» : «...»,               // начало срока действия
      «valid-to»   : «...»,               // конец срока действия
      «ski»        : «A4871BAE29...»,     // Subject Key Identifier
      «bc»         : «...»,               // Basic Constraints
      «ku»         : «...»,               // Key Usage (CSV)
      «eku»        : «...»,               // Ext. Key Usage (CSV)
      «san-dns»    : «...»,               // SAN DNS (CSV)
      «san-ip»     : «...»,               // SAN IP (CSV)
      «san-uri»    : «...»,               // SAN URI (CSV)
      «san-email»  : «...»,               // SAN email (CSV)
      «san-oname»  : «...»,               // SAN other name (CSV)
      «template»   : «...»,               // шаблон сертификата
      «crt»        : «...»                // сертификат x509v3 в
формате PEM
    }
  ]
}
```

HTTP ответ:

1) 4xx, 5xx — ошибка (см. errno и error);
2) 200 — успех, сертификаты из БД предоставлены (errno=0).

3.4.8.13. Предоставление информации о шаблонах УЦ

Метод: POST
Endpoint: {BASE}/templates

JSON запрос:

```
{
  «action» : «templates»,
  «time»   : «...»  // текущее время в формате RFC3339
}
```

JSON ответ:

```
{
  «errno»  : 0,     // код ошибки (0 - успех)
  «error»  : «...», // текст ошибки (в случае успеха пустая строка)
  «templates» : [   // шаблоны (массив)
    {
      «id»       : 123,           // идентификатор шаблона в БД
(int64)
      «name»     : «user»,        // имя/идентификатор шаблона
(ASCII)
      «version»  : 1,             // версия шаблона
      «desc»     : «пользователь», // описание шаблона (UTF8)
      «policies» : [...],         // политики для Subject (JSON)
      «days»     : 31,            // максимальный срок действия
(умолчание)
      «no-aia»   : false,         // отключить расширение AIA
      «no-cdp»   : false,         // отключить расширение CDP
      «key-type» : «RSA»,         // допустимый тип ключа или «»
```

```
      «key-size» : 2048,           // минимально допустимый размер
ключа
      «exts»     : «user_cert»,    // секция конфигурации в
"openssl.cnf"
      «ku»       : «...»,          // key usage (CSV)
      «eku»      : «...»           // extended key usage (CSV)
    },
    {
      «id»       : 124,          // идентификатор шаблона (int64)
      «name»     : «tls-short»,  // имя/идентификатор шаблона (ASCII)
      «version»  : 2,            // версия шаблона
      «desc»     : «TLS сервер», // описание шаблона (UTF8)
      «policy»   : {...},        // политика для Subject (JSON)
      «days»     : 20,           // максимальный срок действия (или
умолчание)
      «no-aia»   : false,        // отключить расширение AIA
      «no-cdp»   : false,        // отключить расширение CDP
      «short»    : true,         // признак «короткоживущего»
сертификата
      «key-type» : «»,           // допустимый тип ключа или «»
      «key-size» : 2048,         // минимально допустимый размер ключа
      «exts»     : «tls_cert»,   // секция конфигурации в файле
"openssl.cnf"
      «ku»       : «...»,        // key usage (CSV)
      «eku»      : «...»         // extended key usage (CSV)
    }
  ]
}
```

Шаблоны предоставляются из таблицы БД (Примечание: до версии 1.6.x информация о шаблонах извлекалась из файла конфигурации MiniCA чуть в другом виде).

HTTP ответ:

1) 4xx, 5xx — ошибка (см. errno и error);
2) 200 — успех, шаблоны предоставлены (errno=0).

3.4.8.14. Предоставить CSR данного сертификата

Метод: POST
Endpoint: {BASE}/csr

JSON запрос:

```
{
  «action»      : «csr»,
  «time»        : «...», // текущее время в формате RFC3339
  «id»          : 1235,  // идентификатор сертификата в базе данных
(int64)
  «serial»      : «...», // серийный номер сертификата в базе данного
ЦС
  «fingerprint» : «...», // отпечаток сертификата (HEX) или пустая
строка
  «requester»   : «...»  // тот, кто запросил сертификат (user/host)
}
```

Должен быть задан или id, serial, fingerprint или requester (что-то одно).

Если указан requester, то возвращается запрос на последний по порядку в таблице БД сертификат.

JSON ответ:

```
{
  «errno»         : 0,     // код ошибки (0 - успех)
  «error»         : «...», // текст ошибки (в случае успеха пустая
строка)
  «id»            : 1235,  // идентификатор сертификата в базе данных
(int64)
  «serial»        : «...», // серийный номер сертификата в базе
данного ЦС
  «fingerprint»   : «...», // отпечаток сертификата (HEX)
  «requester»     : «...», // владелец сертификата (user/host)
  «csr»           : «...»  // CSR запрос (PKCS#10) в формате PEM
}
```

HTTP ответ:

1) 4xx, 5xx — ошибка (см. errno и error);
2) 200 — успех, CSR из БД предоставлен (errno=0).

3.4.8.15. Реабилитация ранее отозванного сертификата

Реабилитировать можно сертификат, который отозван только с причиной «certificateHold».
Метод: POST
Endpoint: {BASE}/repair

JSON запрос:

```
{
  «action»      : «repair»,
  «time»        : «...», // текущее время в формате RFC3339
  «id»          : 1235,  // идентификатор сертификата в базе данных
(int64)
  «serial»      : «...», // серийный номер сертификата в базе данного
ЦС
  «fingerprint» : «...», // отпечаток сертификата (HEX) или пустая
строка
  «requester»   : «...»  // тот, кто запросил сертификат (user/host)
  «comment»     : «...», // подробная реабилитации (UTF8), комментарий
  «context»     : «...»  // дополнительные данные от ЕСАУС (JSON)
}
```

Должен быть задан или id, serial, fingerprint или requester (что-то одно).
Если указан requester, то реабилитируется последний по порядку в таблице БД сертификат.

JSON ответ:

```
{
  «errno»        : 0,     // код ошибки (0 - успех)
  «error»        : «...», // текст ошибки (в случае успеха пустая
строка)
  «crypto-error» : «...», // ошибка криптопровайдера (OpenSSL)
```

```
  «id»           : 12356  // идентификатор сертификата в базе данных
(int64)
  «serial»       : «...», // серийный номер сертификата
  «fingerprint»  : «...», // отпечаток сертификата (HEX)
  «requester»    : «...», // владелец сертификата (user/host)
  «template»     : «...»  // шаблон сертификата
}
```

HTTP ответ:

1) 4xx, 5xx — ошибка (см. errno и error);
2) 200 — успех, сертификат восстановлен (errno=0).

3.4.8.16. Запрос статистики использования СУБД

Метод: POST
Endpoint: {BASE}/stat

JSON запрос:

```
{
  «action» : «stat»,
  «time»   : «...», // текущее время в формате RFC3339
}
```

JSON ответ:

```
{
  «errno»       : 0,     // код ошибки (0 - успех)
  «error»       : «...», // текст ошибки (в случае успеха пустая
строка)
  «valid-cnt»   : 100,   // число действующих сертификатов в БД
  «revoked-cnt» : 10,    // число отозванных сертификатов в БД
  «expired-cnt» : 50,    // число просроченных сертификатов в БД
  «unknown-cnt» : 0, // число сертификатов с неверным статусом (должно
быть 0)
  «csr-cnt»     : 200,   // число CSR запросов в БД
```

```
  «crl-cnt»     : 70,    // число CRL/DeltaCRL в БД
  «templ-cnt»   : 7,     // число записей о шаблонах в БД
  «JWT-cnt»     : 3,     // число записей о JWT в БД
  «evt-cnt»     : 200    // число событий в БД
}
```

HTTP ответ:

1) 4xx, 5xx — ошибка (см. errno и error);
2) 200 — успех, данные статистики предоставлены (errno=0).

Примечание: запрос может выполняется несколько секунд!

3.4.8.17. Получить шаблон

Метод: POST
Endpoint: {BASE}/gettempl

JSON запрос:

```
{
  «action»  : «gettempl»,
  «time»    : «...»,        // текущее время в формате RFC3339
  «id»      : 123           // идентификатор шаблона в БД или 0
  «name»    : «tls-server», // имя (идентификатор) шаблона (ASCII) или
«»
  «ver»     : 2,            // версия шаблона (0 - предоставить
последний)
}
```

Если указан «id», то «name» и «version» игнорируются. Если указан name, id=0, version=0, то возвращается не удаленный шаблон в последней версии.

JSON ответ:

```
{
  «errno»    : 0,              // код ошибки (0 - успех)
```

```
  «error»    : «...»,          // текст ошибки (в случае успеха пустая
строка)
  «update-time» : «...»,       // время обновления записи о шаблоне
(RFC3339)
  «id»       : 123,            // идентификатор шаблона в БД (int64)
  «name»     : «user»,         // имя/идентификатор шаблона (ASCII)
  «ver»      : 3,              // версия шаблона
  «desc»     : «пользователь», // описание шаблона (UTF8)
  «policies» : [...],          // политики для Subject (JSON)
  «priv»     : «...»,          // перечень привилегий JWT (CSV)
  «days»     : 31,             // максимальный срок действия (или
умолчание)
  «no-aia»   : false,          // отключить расширение AIA в
сертификате (bool)
  «no-cdp»   : false,          // отключить расширение CDP в
сертификате (bool)
  «short»    : false,          // признак «короткоживущего»
сертификата
  «key-type» : «»,             // допустимый тип ключа или «»
  «key-size» : 2048,           // минимально допустимый размер ключа
  «deleted»  : false,          // признак удаленного шаблона
  «exts»     : «user_cert»,    // секция конфигурации в файле OpenSSL
  «bc»       : «...»,          // опции Basic Constraints (CSV)
  «ku»       : «...»,          // опции Key Usage (CSV)
  «eku»      : «...»,          // опции Extended Key Usage (CSV)
  «ski»      : «...»,          // опции Subject Key Identifier (CSV)
  «aki»      : «...»,          // опции Authority Key Identifier (CSV)
  «ext»      : «...»           // произвольные опции OpenSSL
(key=value)
}
```

3.4.8.18. Добавить/обновить шаблон

Метод: POST
Endpoint: {BASE}/addtempl

JSON запрос:

```
{
  «action»   : «addtempl»,
  «time»     : «...»,        // текущее время в формате RFC3339
  «id»       : 123,          // идентификатор шаблона в БД или 0
  «name»     : «tls-server», // имя/идентификатор шаблона (ASCII) или
«»
  «ver»      : 0,            // версия шаблона (0 – использовать
следующий)
  «desc»     : «TLS сервер», // описание шаблона (UTF8)
  «policies» : [...],        // политики для Subject (JSON)
  «priv»     : «...»,        // перечень привилегий JWT (CSV)
  «days»     : 365,          // максимальный срок действия (или
умолчание)
  «no-aia»   : false,        // отключить расширение AIA в сертификате
(bool)
  «no-cdp»   : false,        // отключить расширение CDP в сертификате
(bool)
  «short»    : false,        // признак «короткоживущего» сертификата
  «key-type» : «»,           // допустимый тип ключа или «»
  «key-size» : 2048,         // минимально допустимый размер ключа
  «deleted»  : false,        // признак удаленного шаблона
  «exts»     : «user_cert»,  // секция конфигурации в файле OpenSSL
  «bc»       : «...»,        // опции Basic Constraints (CSV)
  «ku»       : «...»,        // опции Key Usage (CSV)
  «eku»      : «...»,        // опции Extended Key Usage (CSV)
  «ski»      : «...»,        // опции Subject Key Identifier (CSV)
  «aki»      : «...»,        // опции Authority Key Identifier (CSV)
  «ext»      : «...»         // произвольные опции OpenSSL (key=value)
}
```

Если указан «id», то редактируется именно заданный шаблон.

В противном случае редактируется шаблон заданный через «name» и «version».

Если version=0, то обновляется последний шаблон с увеличением версии.

Если шаблон не найден, то он добавляется в БД.

JSON ответ:

```
{
```

```
  «errno»   : 0,            // код ошибки (0 - успех)
  «error»   : «...»,        // текст ошибки (в случае успеха пустая
строка)
  «id»      : 123,          // идентификатор шаблона (int64)
  «name»    : «tls-server», // имя/идентификатор шаблона (ASCII)
  «ver»     : 4,            // следующая (актуальная) версия шаблона
  «deleted» : false         // признак удаленного шаблона
}
```

3.4.8.19. Удалить шаблон

Метод: POST
Endpoint: {BASE}/deltempl

JSON запрос:

```
{
  «action» : «deltempl»,
  «time»   : «...»,        // текущее время в формате RFC3339
  «id»     : 123,          // идентификатор шаблона в БД или 0
  «name»   : «tls-server», // имя (идентификатор) шаблона (ASCII)
  «ver»    : 3             // версия шаблона (0 - удалить последний)
}
```

Если указан «id», то «name» и «version» игнорируются. Если указан name, id=0, version=0, то удаляется (помечается) шаблон последней версии (по name). Если задан id=0, name и version не 0, то удаляется (помечается) соответствующая версия.

JSON ответ:

```
{
  «errno»   : 0,            // код ошибки (0 - успех)
  «error»   : «...»,        // текст ошибки (в случае успеха пустая
строка)
  «id»      : 123,          // идентификатор шаблона (int64)
  «name»    : «tls-server», // имя/идентификатор шаблона (ASCII)
  «ver»     : 4,            // следующая (актуальная) версия шаблона
```

```
}
```

3.4.8.20. Выпустить JWT токен авторизации

Метод: POST
Endpoint: {BASE}/mkJWT

JSON запрос:

```
{
  «action»     : «mkJWT»,
  «time»       : «...»,        // текущее время в формате RFC3339
  «requester»  : «...»,        // клиент запросивший JWT (user/host)
  «aud»        : «MiniCA.server.su», // массив FQDN сервисов MiniCA
(CSV)
  «JWT-name»   : «user/host»,  // опциональное имя субъекта JWT
  «role»       : «admin»,      // требуемые пользовательские роли
(CSV)
  «perm»       : «ca,revoke»,  // требуемые атомарные роли (CSV)
  «nbf»        : «...»,        // срок действия с (RFC3339) или «»
  «exp»        : «...»,        // срок действия по (RFC3339) или «»
  «key»        : «{{PEM}}»     // публичный ключ (PEM)
}
```

При выпуске токена анализируются права (пользовательские и атомарные) того клиента (субъекта), который запросил новый токен. Новый токен не может быть наделен большими правами, чем тот токен от имени которого поступил запрос. Аналогично со сроком действия - срок действия токена не может быть больше срока действия токена первого субъекта.

Фактически только «суперадминистратор» владеющий токеном без ограниченного срока действия с полными правами может выпускать произвольные токены.

Сам субъект может перевыпустить JWT токен для себя с продлением срока действия только, если его токен наделен соответствующей ролью (renewJWT).

В случае успеха в ответе возвращается сам токен, а также результирующий срок действия и назначенные права.

Полный список атомарных ролей: ping, ca, revoke, status, get, updcrl, crl, updeltacrl, deltacrl, find, delete, export, templates, csr, repair, stat, gettempl, addtempl, deltempl, mkJWT, revokeJWT, repairJWT, delJWT, getJWT, renewJWT.

Пользовательские роли и соответствующие им атомарные роли:

1) **admin**: имеет все возможные права;
2) **causer**: ping, ca, revoke, status, get, repair, csr, gettempl, templates;
3) **crluser**: ping, updcrl, crl, updeltacrl, deltacrl;
4) **archuser**: ping, status, get, crl, deltacrl, find, delete, export, templates, stat, gettempl, csr, getJWT;
5) **exporter**: ping, status, get, crl, deltacrl, find, export, csr, templates, gettempl;
6) **templadmin**: ping, templates, gettempl, addtenpl, deltempl;
7) **JWTadmin**: ping, mkJWT, revokeJWT, repairJWT, delJWT, getJWT, reviveJWT;

Пользовательские роли определяются параметрами конфигурации сервиса (определены в JSON/YAML конфигурационном файле).

JSON ответ:

```
{
  «errno»    : 0,     // код ошибки (0 - успех)
  «error»    : «...», // текст ошибки (в случае успеха пустая строка)
  «aud»      : «MiniCA.server.su», // массив FQDN сервисов MiniCA
(CSV)
  «JWT-name» : «user/host»,  // опциональное имя того, кто запросил
JWT
  «iss»      : «...», // SKI ключа подписи JWT (HEX), 20 байт (поле
iss JWT)
  «sub»      : «...», // SKI ключа клиентами (HEX), 20 байт (поле sub
JWT)
  «jat»      : «...», // время выпуска JWT токена (RFC3339)
  «nbf»      : «...», // установленный срок действия с (RFC3339) или
«»
  «exp»      : «...», // установленный срок действия по (RFC3339) или
«»
  «role»     : «...», // назначенные пользовательские роли (CSV)
  «perm»     : «...», // назначенные атомарные роли (CSV)
  «priv»     : «...», // привилегии по использованию шаблонов (CSV)
  «jti»      : «...», // UUIDv7 идентификатор JWT токена (поле jti
JWT)
  «JWT»      : «...»  // JWT токен в соответствии с RFC 7519
}
```

3.4.8.21. Отозвать JWT токен авторизации

Метод: POST
Endpoint: {BASE}/revokeJWT

JSON запрос:

```
{
  «action» : «revokeJWT»,
  «JWT»    : «...» // JWT токен в соответствии с RFC 7519
}
```

JSON ответ:

```
{
  «errno»  : 0,     // код ошибки (0 - успех)
  «error»  : «...», // текст ошибки (в случае успеха пустая строка)
  «sub»    : «...», // SKI ключа клиентами (HEX), 20 байт (поле sub
JWT)
  «jti»    : «...»  // UUIDv7 идентификатор JWT токена (поле jti JWT)
}
```

Отзываемый токен помечается в БД как отозванный, но из таблицы JWT не удаляется. Для удаления может использоваться HTTP функция «delJWT».

3.4.8.22. Реабилитировать отозванный JWT токен авторизации

Метод: POST
Endpoint: {BASE}/repairJWT

JSON запрос:

```
{
  «action» : «repairJWT»,
  «JWT»    : «...» // JWT токен в соответствии с RFC 7519
}
```

JSON ответ:

```
{
  «errno»  : 0,     // код ошибки (0 - успех)
  «error»  : «...», // текст ошибки (в случае успеха пустая строка)
  «sub»    : «...», // SKI ключа клиентами (HEX), 20 байт (поле sub
JWT)
  «jti»    : «...»  // UUIDv7 идентификатор JWT токена (поле jti JWT)
}
```

Отозванный токен может быть реабилитирован с помощью данного метода.

3.4.8.23. Удалить JWT токен авторизации из БД

Метод: POST
Endpoint: {BASE}/delJWT

JSON запрос:

```
{
  «action» : «revokeJWT»,
  «JWT»    : «...» // JWT токен в соответствии с RFC 7519
}
```

JSON ответ:

```
{
  «errno»  : 0,     // код ошибки (0 - успех)
  «error»  : «...», // текст ошибки (в случае успеха пустая строка)
  «sub»    : «...», // SKI ключа клиентами (HEX), 20 байт (поле sub
JWT)
  «jti»    : «...»  // UUIDv7 идентификатор JWT токена (поле jti JWT)
}
```

Данная функция удаляет JWT токен из БД, соответствующий клиент теряет возможность авторизованного доступа к сервису MiniCA.

3.4.8.24. Запросить JWT токен авторизации из БД

Метод: POST
Endpoint: {BASE}/getJWT

JSON запрос:

```
{
  «action» : «getJWT»,
  «sub»    : «...», // SKI ключа клиентами (HEX), 20 байт (поле sub
JWT)
  «jti»    : «...», // UUIDv7 идентификатор JWT токена (поле jti JWT)
}
```

В запросе должен быть указан «sub» (поиск токена по хэшу ключа клиента — поле «sub» JWT) И/ИЛИ «jti» (поиск токена по UUIDv7 идентификатору токена — поле «jti» JWT). Можно задать только одно поле или оба.

JSON ответ:

```
{
  «errno»       : 0,     // код ошибки (0 - успех)
  «error»       : «...», // текст ошибки (в случае успеха пустая
строка)
  «requester»   : «...», // клиент запросивший JWT (user/host)
  «JWT-revoked» : true,  // признак того, что токен отозван
  «aud»         : «MiniCA.server.su», // массив FQDN сервисов MiniCA
(CSV)
  «JWT-name»    : «user/host»,  // опциональное имя того, кто запросил
JWT
  «iss»         : «...», // SKI ключа подписи JWT (HEX), 20 байт (поле
iss JWT)
  «sub»         : «...», // SKI ключа клиентами (HEX), 20 байт (поле
sub JWT)
  «jat»         : «...», // время выпуска JWT токена (RFC3339)
  «nbf»         : «...», // установленный срок действия с (RFC3339)
или «»
```

```
  «exp»         : «...», // установленный срок действия по (RFC3339)
или «»
  «role»        : «...», // назначенные пользовательские роли (CSV)
  «perm»        : «...», // назначенные атомарные роли (CSV)
  «priv»        : «...», // привилегии по использованию шаблонов (CSV)
  «jti»         : «...», // UUIDv7 идентификатор JWT токена (поле jti
JWT)
  «JWT»         : «...»  // JWT токен в соответствии с RFC 7519
}
```

3.4.8.25. Обновить JWT токен авторизации (с новым сроком/новым ключом)

Метод: POST
Endpoint: {BASE}/renewJWT

JSON запрос:

```
{
  «action»    : «renewJWT»,
  «time»      : «...»,        // текущее время в формате RFC3339
  «requester» : «...»,        // клиент запросивший JWT (user/host)
  «nbf»       : «...»,        // срок действия с (RFC3339) или «»
  «exp»       : «...»,        // срок действия по (RFC3339) или «»
  «key»       : «{{PEM}}»     // публичный ключ (PEM) или пустая
строка
}
```

При перевыпуске токена права (пользовательские и атомарные, name и aud) наследуются от старого токена. Если задан новый публичный ключ, то используется он, если не задан, то сохраняется старый.

JSON ответ:

```
{
  «errno»    : 0,     // код ошибки (0 - успех)
  «error»    : «...», // текст ошибки (в случае успеха пустая строка)
  «aud»      : «MiniCA.server.su», // массив FQDN сервисов MiniCA
(CSV)
```

```
  «JWT-name» : «user/host»,  // опциональное имя того, кто запросил
JWT
  «iss»      : «...», // SKI ключа подписи JWT (HEX), 20 байт (поле
iss JWT)
  «sub»      : «...», // SKI ключа клиентами (HEX), 20 байт (поле sub
JWT)
  «jat»      : «...», // время выпуска JWT токена (RFC3339)
  «nbf»      : «...», // установленный срок действия с (RFC3339) или
«»
  «exp»      : «...», // установленный срок действия по (RFC3339) или
«»
  «role»     : «...», // назначенные пользовательские роли (CSV)
  «perm»     : «...», // назначенные атомарные роли (CSV)
  «priv»     : «...», // привилегии по использованию шаблонов (CSV)
  «jti»      : «...», // UUIDv7 идентификатор JWT токена (поле jti
JWT)
  «JWT»      : «...»  // JWT токен в соответствии с RFC 7519
}
```

3.4.8.26. Обновить просроченный JWT токен авторизации

Данная функция оформлена как отдельный HTTP вызов для того, чтобы можно было разграничивать права доступа к данной функции стандартным механизмом атомарных и пользовательских прав, указываемых в JWT.

Метод: POST

Endpoint: {BASE}/reviveJWT

JSON запрос:

```
{
  «action»    : «reviveJWT»,
  «time»      : «...»,        // текущее время в формате RFC3339
  «requester» : «...»,        // клиент запросивший JWT (user/host)
  «nbf»       : «...»,        // срок действия с (RFC3339) или «»
  «exp»       : «...»,        // срок действия по (RFC3339) или «»
  «key»       : «{{PEM}}»     // публичный ключ (PEM) или пустая
строка
```

```
}
```

При пере выпуске просроченного токена права (пользовательские и атомарные, name и aud) наследуются от старого токена. Если задан новый публичный ключ, то используется он, если не задан, то сохраняется старый.

JSON ответ:

```
{
  «errno»    : 0,     // код ошибки (0 - успех)
  «error»    : «...», // текст ошибки (в случае успеха пустая строка)
  «aud»      : «MiniCA.server.su», // массив FQDN сервисов MiniCA
(CSV)
  «JWT-name» : «user/host»,  // опциональное имя того, кто запросил
JWT
  «iss»      : «...», // SKI ключа подписи JWT (HEX), 20 байт (поле
iss JWT)
  «sub»      : «...», // SKI ключа клиентами (HEX), 20 байт (поле sub
JWT)
  «jat»      : «...», // время выпуска JWT токена (RFC3339)
  «nbf»      : «...», // установленный срок действия с (RFC3339) или
«»
  «exp»      : «...», // установленный срок действия по (RFC3339) или
«»
  «role»     : «...», // назначенные пользовательские роли (CSV)
  «perm»     : «...», // назначенные атомарные роли (CSV)
  «priv»     : «...», // привилегии по использованию шаблонов (CSV)
  «jti»      : «...», // UUIDv7 идентификатор JWT токена (поле jti
JWT)
  «JWT»      : «...»  // JWT токен в соответствии с RFC 7519
}
```

3.4.8.27. Предоставить цепочку сертификатов

Данная сервисная функция предоставляет указанную при конфигурации сервиса цепочку сертификатов. Обычно цепочка сертификатов предоставляет последовательность X.509v3 сертификатов в PEM формате:

1) сертификат ЦС;
2) цепочка промежуточных сертификатов (если есть);
3) сертификат (самоподписанный) корневого УЦ.

Можно сказать, что сертификаты в цепочке представлены в «обратном хронологическом порядке» их выпуска. Сервис предоставляет данную цепочку как «статический контент», т. е. клиент получает тот набор данных, что указаны в конфигурационном файле сервиса при его настройке.

Метод: POST
Endpoint: {BASE}/chain

JSON запрос:

```
{
  «action»    : «chain»,
  «time»      : «...», // текущее время в формате RFC3339
  «requester» : «...», // клиент запросивший данные (user/host)
}
```

JSON ответ:

```
{
  «errno» : 0,      // код ошибки (0 - успех)
  «error» : «...»,  // текст ошибки (в случае успеха пустая строка)
  «cn-ca» : «CA 3», // CommonName из сертификата УЦ
  «chain» : «...»   // цепочка сертификатов в PEM формате
}
```

Коды ошибок API:
0: «success» - успех операции
5: «unauthorized (bad JWT)» - неверный JWT токен
10: «bad sign» - некорректная цифровая подпись запроса
15: «no privileges for template» - в JWT нет привилегий для выбора шаблона УЦ
20: «bad CSR» - запрос на выпуск сертификата содержит ошибку
25: «subject doesn»t match the policy» - поле Subject не соответствует политикам
30: «certificate not found» - сертификат не найден в базе центра сертификации
40: «certificate already expired» - срок действия сертификата уже истек
50: «certificate already revoked» - сертификат уже отозван
60: «certificate not revoked» - сертификат не отзывался

70: «CRL not found» - последний CRL не найден в базе центра сертификации
80: «DeltaCRL not found» - DeltaCRL не найден в базе центра сертификации
90: «CSR not found» - CSR не найден в базе центра сертификации
100: «bad time of request» - не корректное время запроса
110: «openssl error» - ошибка выполнения OpenSSL
120: «openssl timeout error» - ошибка выполнения OpenSSL — таймаут
130: «internal (file) error» - внутренняя ошибка выполнения
140: «unknown AKI» - не найден SKI соответствующего ЦС
150: «template not found» - шаблон не найден
160: «template already deleted» - шаблон уже удален
170: «bad key type» - недопустимый тип ключа (в CSR)
180: «key size too short» - ключ слишком короткий (в CSR)
300: «JWT not found» - JWT токен не найден
310: «JWT already revoked» - JWT токен уже отозван
320: «JWT not revoked» - JWT токен не отозван
400: «bad request» - не корректный запрос
404: «not found» - ресурс не найден (HTTP 404 Not Found)
500: «database error» - ошибка СУБД
520: «unknown error» - неизвестная ошибка

Используемые коды состояния HTTP
200 - OK (успех)
400 - Bad request (не корректный запрос)
401 - Unauthorized (не авторизован, неверный JWT)
404 - Not Found (не верный путь запроса, данные не найдены)
405 - Method Not Allowed (метод не допустим)
408 - Request Timeout (OpenSSL timeout)
423 - Locked (неверная подпись)
500 - Internal Server Error (внутренняя ошибка)
501 - Not Implemented (метод пока не реализован)
503 - Service Unavailable (ошибка СУБД)
520 - Unknown Error (неизвестная ошибка)

3.5. Входные и выходные данные

3.5.1. Входные данные

3.5.1.1. Механизм подписи запросов и авторизация

Клиент микросервиса должен в стандартном HTTP заголовке «Authorization Bearer» предъявлять специальный JWT токен, который подписан ключом самого сервиса MiniCA. JWT токен может использовать алгоритмы подписи: HS256, RS256, ES256 или EdDSA. Если используются асимметричные алгоритмы подписи, то токен может быть проверен не только самим сервером. В поле «iss» токен должен содержать SKI (Subject Key Identifier) — т.е. хеш открытого (или в случае с HMAC секретного) ключа сервера, выпустившего JWT. Поле «sub» токена должно содержать SKI (хэш) открытого ключа клиента, который используется для подписи самих запросов. Формат JWT токена описан ниже.

Все запросы, которые обрабатывает микросервис должны быть подписаны закрытым ключом клиента. Отрытый мастер ключ для проверки подписи извлекается из сертификата X.509v3, указанного в конфигурации микросервиса (сертификат является контейнером для открытого ключа и никак не проверяется) или извлекается из таблицы клиентов СУБД. Запросы, описанные ниже, передаются в форме JSON структур. Подпись формируется для SHA-256 хеш суммы сериализованных в JSON структур. Сами данные сериализации JSON структуры и подпись в свою очередь передаются в формате base64 внутри следующей JSON структуры:

```
{
  «data» : «...», // JSON структура запроса в кодировке base64
  «sign» : «...»  // цифровая подпись запроса на основе SHA-256 и RSA
в base64
}
```

Здесь и далее указаны комментарии в нарушение спецификации JSON только в рамках данного документа. Указанные ниже JSON структуры запросов упаковываются в представленный контейнер в поле «data».

3.5.1.2. JWT токены авторизации

JWT токен клиента (см. RFC7519) имеет следующий (символический) формат:

```
Заголовок:
{
```

```
  «typ»: «JWT»,  // стандартный тип JWT токена
  «alg»: «RS256» // алгоритм подписи (HS256/RS256/ES256/EdDSA)
}
Полезная нагрузка:
{
  «iss»: «AKI», // идентификатор (хеш) открытого ключа ЦС (HEX), 20
байт
  «sub»: «SKI», // идентификатор (хеш) открытого ключа клиента (HEX),
20 байт
  «name»: «requester», // опциональный идентификатор клиента
  «aud»: [«MiniCA.server.su»], // массив FQDN сервисов MiniCA
  «exp»: 1767225600, // срок действия JWT (Unix time, UTC), если есть
  «nbf»: 1735668000, // время (Unix time) до которого JWT не
действует, если есть
  «iat»: 1738260000, // время, когда JWT выпущен (Unix time, UTC)
  «jti»: «uuid», // идентификатор токена (UUIDv7)
  «role»: «admin,exporter,» // перечень пользовательских ролей (CSV)
  «perm»: «ca,revoke,...», // перечень атомарных прав (CSV)
  «priv»: «user,tls,...» // перечень привилегий на выпуск сертификатов
(CSV)
}
Подпись: HMAC/RSA/ECDSA/Ed25519
```

1) iss=AKI — это Subject Key Identifier в терминах RFC 5280 для открытого ключа сервиса, который выпустил JWT, 20 байт в шестнадцатеричном формате (вычисляется в соответствии с RFC 7093).
2) sub=SKI — это Subject Key Identifier для открытого ключа автоматизируемого клиента (собственника токена), 20 байт в шестнадцатеричном формате (вычисляется в соответствии с RFC 7093).
3) aud — необязательное поле, ограничивавшее применяемость JWT только для заданных серверов MiniCA (по FQDN).
4) exp — не обязательное поле (если не указано, то токен бессрочный)
5) nbf — не обязательное поле, указывающее время до которого JWT не действует
6) iat — время выпуска JWT (обязательное поле)
7) jti — обязательное поле (UUIDv7) для кеширования JWT токенов на стороне сервера
8) role — CSV строка с перечнем допустимых стандартных ролей
9) perm — CSV строка с перечнем разрешенных для данного клиента функций API MiniCA

10) priv — CSV строка с перечнем привилегий для выпуска сертификатов по шаблонам

Порядок использования поля priv в JWT и соответствующего значения в шаблоне.

1) Если в JWT поле priv пустое, то это "master JWT" и его владелец может выпускать сертификат по любым шаблонам;
2) Если в шаблоне не указаны требуемые привилегии, то данный шаблон может использовать каждый (у кого есть токен, разрешающий выпуск сертификатов);
3) Если в шаблоне указаны требуемые привилегии (CSV), а в JWT имеется не пустой списков привилегий (CSV), то должно быть хотя бы одно пересечение списков требований (в шаблоне) и назначений (в JWT).

Поле role позволяет компактным образом вложить в JWT список стандартных ролей доступа. Допустимы следующие роли:

- admin — суперпользователь, которому разрешено все;
- causer — сервис, которому разрешены выпуск/отзыв/запрос сертификатов;
- crluser — сервис, которому разрешен выпуск/запрос CRL/DeltaCRL
- archuser — сервис, которому разрешены операции с сертификатами/CSR в БД (в т.ч. удаление);
- exporter — сервис, которому разрешен экспорт/поиск сертификатов/CRL/запросов из БД (доступ к БД в режиме «только-чтение»);
- templamin — администратор шаблонов;
- JWTadmin — администратор JWT.

Список ролей может быть дополнен (сокращен), пользовательские роли описываются параметром конфигурации сервиса (описаны в JSON/YAML конфигурационном файле). Роли могут не использоваться, если используются атомарные права.

Роли могу использоваться совместно с атомарными правами.

Поле perm может содержать список идентификаторов, указывающих на допустимость вызова соответствующих функций HTTP API MiniCA:

- ping — проверка доступности и запрос версии;
- ca — выпуск сертификата;
- revoke — отзыв сертификата;
- status — запрос статуса сертификата;
- get — получение сертификата из БД;
- updcrl — выпуск нового CRL;
- crl — запрос CRL из БД;

- updeltacrl — выпуск нового DeltaCRL;
- deltacrl — запрос DeltaCRL из БД;
- find — поиск сертификата в БД;
- delete — удаление сертификата из БД;
- export — экспорт группы сертификатов из БД;
- templates — получение перечня всех шаблонов;
- csr — выгрузка CSR запроса (PKCS#10) из БД;
- repair — реабилитация отозванного сертификата;
- stat — запрос статистики использования БД;
- gettempl — запрос шаблона;
- addtemp — добавление/обновление шаблона;
- delteml — удаление шаблона;
- mkJWT — создание/обновление JWT токена доступа;
- revokeJWT — отзыв JWT токена доступа;
- repairJWT — реабилитация отозванного JWT токена доступа;
- delJWT — удаление JWT токена из БД;
- getJWT — запрос JWT токена из БД по UUID или SKI клиента;
- renewJWT — право перевыпуска JWT токена «для себя» с продлением срока действия;
- reviveJWT — право перевыпуска просроченного JWT токена «для себя» с продлением срока действия;
- chain — право запроса цепочки сертификатов;

Важно, что поле «iss» может использоваться для определения ключа подписи самого JWT токена. Таким образом одно «плечо» MiniCA может проверить JWT токен выданных другим «плечом» при наличии соответствующего секретного (при использовании HMAC) или открытого ключа, указанного в параметрах конфигурации сервиса.

Поле «sub» позволяет идентифицировать открытый ключ клиента (найти его сам ключ в кеше или в СУБД) и использовать его для проверки подписи самого запроса.

3.5.1.3. «Короткоживущие» сертификаты

Короткоживущие сертификаты определяются по одному из следующих признаков:

1) В шаблоне сертификата указан признак short=true в JSON конфигурационном файле ЦС;
2) В HTTP (JSON) запросе на выпуск сертификата указан признак short=true;

3) Срок действия сертификата в запросе/шаблоне менее чем заданный параметр short-days в конфигурации ЦС.

При выпуске «короткоживущих» сертификатов ЦС не сохраняет CSR, сертификат и события в БД. Другими словами, состояние БД при выпуске «короткоживущих» сертификатов не изменяется.

Серийный номер для «короткоживущих» сертификатов всегда устанавливается случайным.

3.5.1.4. Шаблоны

Шаблоны позволяют выпускать различные типы сертификатов под разные нужды (обычно это TLS сертификаты для сертификатов, mTLS сертификаты клиентов, сертификаты сервисов OCSP и др.).

Шаблоны определяют состав KU (key usage), EKU (Extended Key Usage) и других атрибутов X.509v3 сертификатов, срок действия сертификатов (максимальный и он же используемый по умолчанию).

Для сохранения шаблонов УЦ используется отдельная таблица БД и отдельный HTTP API для доступа к шаблонам.

Особое место занимает поле policy, которое определяет политику для допустимых в CSR (PKCS#10) запросе значений поля Subject. При «пустой» политики допускается выпуск сертификатов с произвольным Subject.

Форма поля policy описан ниже в символическом JSON виде.

```
[ {
    «dn»: «C», // имя атрибута (CN/L/ST/OU/C/STREET/DC/UID) – RFC2253
    «oid»: «2.5.4.6», // OID идентификатор атрибута (не обязательно)
    «desc»: «Страна», // описание атрибута/политики
    «required»: true/false, // данный атрибут является
обязательным/нет
    «value»: «RU», // требуемое значение атрибута или пустая строка
    «glob»: «*», // допустимое значение (glob выражение)
    «regexp»: «^RU$» // допустимое значение (регулярное выражение)
  },
  {
    «san» : «dns», // Идентификатор SAN («dns», «email», «ip», «uri»)
    «desc»: «допустимый FQDN SAN», // описание атрибута/политики
    «required»: true/false, // данный атрибут является
обязательным/нет
```

```
    «value»: «», // требуемое значение атрибута или пустая строка
    «glob»: «*.super.org» // допустимое значение (glob выражение)
    «regexp»: «.*\.super\.org» // допустимое значение (регулярное
выражение)
  }
]
```

3.5.2. Выходные данные

3.5.2.1. Данные сертификата

В состав сертификата входят следующие сведения:

1) Серийный номер сертификата (Serial Number).
2) DN (Distinguished Name) субъекта (Subject).
3) DN издателя (Issuer).
4) Алгоритм подписи (Signature Algorithm).
5) Срок действия (Not Before, Not After).
6) Публичный ключ (Public Key).
7) Расширения сертификата (Certificate Extensions) - включаются все расширения, например, Key Usage, Extended Key Usage, Subject Alternative Name (SAN) и др.
8) Идентификатор запроса на сертификат (Certificate Request ID) - для отслеживания связи сертификата с первоначальным запросом.
9) Политики сертификации (Certificate Policies) - идентификаторы политик, которым соответствует сертификат.
10) Формат сертификата (например, X.509 DER, PEM).
11) Идентификатор алгоритма хеширования (Hash Algorithm).
12) Отпечаток сертификата (Certificate Fingerprint) - SHA-256, SHA-1 и др. (опционально).
13) URL OCSP (Online Certificate Status Protocol) (опционально) - для проверки статуса отзыва сертификата.
14) Информация о отозвании (Revocation Information) - дата и причина отзыва (если сертификат отозван).

3.5.2.2. Информация о процессе выпуска

В состав информации о процессе выпуска сертификата входят следующие сведения:

1) ID пользователя, выполнившего выпуск (User ID)
2) Временная метка выпуска (Timestamp)

3) ID запроса (Request ID)
4) ID политики выпуска (Policy ID)
5) Источник запроса (Source - например, веб-интерфейс, API, автоматизированный процесс)

3.5.2.3. Статус операции

1) Успех (Success) - сертификат успешно сохранен в базу данных.
2) Ошибка (Failure) - произошла ошибка при сохранении.
3) Код ошибки (Error Code) - уникальный идентификатор ошибки для диагностики.
4) Сообщение об ошибке (Error Message) - описание ошибки.

3.5.2.4. Идентификатор записи в базе данных

1) Уникальный идентификатор сохраненного сертификата в базе данных. Может быть автоинкрементным ID или UUID.

3.6. Алгоритм работы

3.6.1. Алгоритм типовой операции

MiniCA работает как web-сервис и принимает на вход HTTP-запрос. При этом выполняется ряд проверок. Ниже приведен порядок обработки типового запроса к MiniCA:

1) Клиент отправляет HTTP-запрос на микросервис MiniCA. Запрос содержит два компонента: данные в виде base64-кода и цифровую подпись этих данных.
2) Микросервис проверяет заголовок Authorization, где содержится JWT-токен. Проверяется:
    1) Алгоритм подписи токена (HS256, RS256, ES256 или EdDSA);
    2) Значения полей `iss` и `sub`:
        - `iss` содержит уникальный идентификатор ключа сервера;
        - `sub` содержит идентификатор открытого ключа клиента.
1) После успешной проверки токена, микросервис декодирует данные и цифру (`signature`) из запроса. Данные представляют собой JSON-объект, закодированный в base64.
2) Проверяется цифровая подпись данных — MiniCA вычисляет SHA-256-хеш распакованного JSON-объекта и сравнивает его с цифровой подписью, приложенной клиентом.

3) Если подпись совпадает, проходит дополнительная проверка прав доступа клиента, исходя из ролевой модели или списка разрешенных операций.
4) Производится непосредственное выполнение запрошенной операции с сертификатами: создание, обновление, аннулирование и т.п.
5) Применяется блокировка (мьютекс) для исключения возможных проблем параллельного доступа к общим данным (конфликты записи/чтения) и предотвращения ситуации, когда разные потоки одновременно пытаются изменить одни и те же ресурсы.
6) Результат выполненной операции сохраняется в базе данных PostgreSQL.
7) Формируется ответ клиенту. Ответ состоит из двух частей:
    1) Основной объект данных (результат операции), закодированный в base64;
    2) Новая цифровая подпись этого объекта, созданная сервером.
8) Ответ отправляется клиенту, завершая весь цикл обработки запроса.

3.6.2. Типовые операции

1) Получение данных сертификата и процесса выпуска:
    - принимает данные сертификата и информацию о процессе выпуска в качестве входных параметров.
2) Валидация данных:
    - проверяет целостность и корректность входных данных. Проверяет наличие обязательных полей, правильность форматов и соответствие заданным ограничениям.
3) Формирование записи для базы данных:
    - преобразует входные данные в формат, совместимый со структурой базы данных.
4) Сохранение данных в базу данных:
    - выполняет операцию записи данных в базу данных. Обрабатывает возможные исключения (например, ошибки подключения к базе данных, нарушение ограничений целостности данных).
5) Логирование:
    - записывает информацию о процессе сохранения (успех/ошибка, время выполнения, параметры) в журналы аудита.
6) Возврат статуса:
    - возвращает статус операции, код ошибки (если есть) и идентификатор записи в базе данных.

3.7. База данных

Структура базы данных включает следующие таблицы:

1) CSR, таблица запросов на сертификаты.
2) CERT, таблица сертификатов.
3) CLR, таблица CRL.
4) TEMPL, таблица шаблонов.
5) JWT, таблица JWT токенов
6) EVT, таблица событий.

3.7.1. Структура таблицы CSR

Наименование поля	Тип данных	Описание
id	BIGSERIAL	Уникальный идентификатор запроса в БД, первичный ключ
dtime	TIMESTAMPTZ, NOT NULL, DEFAULT NOW()	Время получения запроса
requester	VARCHAR, NOT NULL	Тот, кто запросил сертификат (user/host)
subj	VARCHAR, NOT NULL	Subject
cn	VARCHAR, NOT NULL	Common Name
san_dns	VARCHAR, NOT NULL	“Subject Alternative Name” DNS (CSV)
san_ip	VARCHAR, NOT NULL	“Subject Alternative Name” IP (CSV)
san_uri	VARCHAR, NOT NULL	“Subject Alternative Name” URI (CSV)
san_email	VARCHAR, NOT NULL	“Subject Alternative Name” email (CSV)
san_oname	VARCHAR, NOT NULL	“Subject Alternative Name” otherName (CSV)
cert_id	BIGINT, NOT NULL	Выпущенный сертификат (или 0)
key_type	VARCHAR, NOT NULL, DEFAULT ‘’	Тип ключа
key_size	INT, NOT NULL, DEFAULT 0	Размер ключа
pem	VARCHAR, NOT NULL	CSR в формате PEM
status	status_csr, NOT NULL	Статус (satisfied, rejected, erroneous)
template	VARCHAR, NOT NULL	Идентификатор шаблона (ASCII)

3.7.2. Структура таблицы Cert

Наименование поля	Тип данных	Описание
id	BIGSERIAL	Уникальный идентификатор сертификата в БД, первичный ключ
dtime	TIMESTAMPTZ, NOT NULL, DEFAULT NOW ()	Время фактического создания сертификата
requester	VARCHAR, NOT NULL	Владелец сертификата (user/host)
update_time	TIMESTAMPTZ, NOT NULL, DEFAULT NOW ()	Время обновления (в т.ч. отзыва)
start_sn	VARCHAR, NOT NULL	Начальный серийный номер сертификата для плеча (HEX)
sn	VARCHAR, UNIQUE, NOT NULL	Серийный номер сертификата (HEX)
template	VARCHAR, NOT NULL	Использованный шаблон сертификата при его выпуске
template_id	BIGINT, NOT NULL	Идентификатор шаблона в БД
fingerprint	VARCHAR, NOT NULL	Отпечаток сертификата (HEX)
subj	VARCHAR, NOT NULL	Subject
cn	VARCHAR, NOT NULL	Common Name
valid_from	TIMESTAMPTZ, NOT NULL	Начало срока действия
valid_to	TIMESTAMPTZ, NOT NULL	Конец срока действия
ski	VARCHAR, NOT NULL	Subject Key Identifier (HEX)
aki	VARCHAR, NOT NULL	Authority Key Identifier (HEX)
bc	VARCHAR, NOT NULL	Basic Constraints
ku	VARCHAR, NOT NULL	Key Usage (CSV)
eku	VARCHAR, NOT NULL	Extended Key Usage (CSV)
san_dns	VARCHAR, NOT NULL	“Subject Alternative Name” DNS (CSV)
san_ip	VARCHAR, NOT NULL	“Subject Alternative Name” IP (CSV)
san_uri	VARCHAR, NOT NULL	“Subject Alternative name” URI (CSV)
san_email	VARCHAR, NOT NULL	“Subject Alternative name” email (CSV)
san_oname	VARCHAR, NOT NULL	“Subject Alternative name” otherName (CSV)
aia	VARCHAR, NOT NULL	Расширение AIA (Authority Information Access)

Наименование поля	Тип данных	Описание
cdp	VARCHAR, NOT NULL	Расширение CDP (CRL Distribution Points)
csr_id	BIGINT, NOT NULL	Идентификатор CSR в БД на основе которого выпущен сертификат
key_type	VARCHAR, NOT NULL, DEFAULT ‘’	Тип ключа
key_size	INT, NOT NULL, DEFAULT 0	Размер ключа
pem	VARCHAR, NOT NULL	Сертификат в формате PEM
status	status_cert, NOT NULL	Статус (unknown, valid, expired, revoked)
revoke_reason	VARCHAR, NOT NULL	Причина отзыва сертификата
index_txt	VARCHAR, NOT NULL	Строка из файла “index.txt” OpenSSL CA соответствующая сертификату
index_bak	VARCHAR, NOT NULL	Строка из файла “index.txt” до отзыва сертификата
kra_id	VARCHAR, NOT NULL, DEFAULT ‘’	Идентификатор ключа для KRA
kra_key	VARCHAR, NOT NULL, DEFAULT ‘’	Контейнер с ключом клиента для KRA
email	VARCHAR, NOT NULL, DEFAULT ‘’	E-mail для системы уведомлений
flags	BIGINT, NOT NULL, DEFAULT 0	Битовые флаги для внешних систем

3.7.3. Структура таблицы CLR

Наименование поля	Тип данных	Описание
id	BIGSERIAL	Уникальный идентификатор CRL/DeltacRL в БД, первичный ключ
num	VARCHAR, NOT NULL	Сквозной номер CRL/DeltaCRL
aki	VARCHAR, NOT NULL	Authority Key Identifier (HEX)
dtime	TIMESTAMPTZ, NOT NULL, DEFAULT NOW ()	Время выпуска CRL/DeltaCRL
last_update	TIMESTAMPTZ, NOT NULL	Время последнего обновления
next_update	TIMESTAMPTZ, NOT NULL	Следующее время обновления
revoked	INT, NOT NULL, DEFAULT 0	Число отозванных сертификатов

Наименование поля	Тип данных	Описание
pem	VARCHAR, NOT NULL	CRL/DeltaCRL в формате PEM
delta_crl	BOOLEAN, NOT NULL, DEFAULT FALSE	Признак DeltaCRL

3.7.4. Структура таблицы Templ

Наименование поля	Тип данных	Описание
id	BIGSERIAL	Уникальный идентификатор события в БД, первичный ключ
dtime	TIMESTAMPTZ, NOT NULL, DEFAULT NOW()	Время обновления записи
name	VARCHAR, NOT NULL	ASCII идентификатор шаблона
descr	VARCHAR, NOT NULL	UTF8 описание шаблона
policy	VARCHAR, NOT NULL, DEFAULT ‘’	Политики Subject (JSON) (v2.1.x)
priv	VARCHAR, NOT NULL, DEFAULT ‘’	Перечень привилегий JWT для которых разрешен шаблон (CSV)
days	INT, NOT NULL	Максимальный срок действия сертификата
noaia	BOOLEAN, NOT NULL	Признак отключения расширения AIA
nocdp	BOOLEAN, NOT NULL	Признак отключения расширения CDP
short	BOOLEAN, NOT NULL	Признак “короткоживущего” сертификата
version	INT, NOT NULL	Версия шаблона
deleted	BOOLEAN, NOT NULL, DEFAULT FALSE	Признак удаления
key_type	VARCHAR, NOT NULL, DEFAULT ‘’	Допустимый тип ключа (“” - не задан)
key_size	INT, NOT NULL, DEFAULT 0	Минимально допустимый размер ключа (0 - не задан)
exts	VARCHAR, NOT NULL	Секция конфигурации в файле OpenSSL
bc	VARCHAR, NOT NULL	basicConstraints
ku	VARCHAR, NOT NULL	keyUsage
eku	VARCHAR, NOT NULL	extendedKeyUsage
ski	VARCHAR, NOT NULL	subjectKeyIdentifier
aki	VARCHAR, NOT NULL	authorityKeyIdentifier

Наименование поля	Тип данных	Описание
ext	VARCHAR, NOT NULL	Произвольный текст (многострочный)

3.7.5. Структура таблицы JWT

Наименование поля	Тип данных	Описание
id	BIGSERIAL	Уникальный идентификатор записи о JWT токене в БД, первичный ключ
start_sn	VARCHAR, NOT NULL	Начальный серийный номер сертификата для выпускающего плеча
requester	VARCHAR, NOT NULL	Клиент запросивший JWT (user/host)
updated_at	TIMESTAMPTZ, NOT NULL, DEFAULT NOW()	Время обновления записи о JWT (в т.ч. время отзыва)
jat	TIMESTAMPTZ, NOT NULL	Время фактического создания JWT (поле jat JWT)
nbf	TIMESTAMPTZ, NOT NULL	Начало срока действия JWT (на основе поля nbf JWT)
exp	TIMESTAMPTZ, NOT NULL	Конец срока действия JWT (на основе поля exp JWT)
name	VARCHAR, NOT NULL	Субъект для которого выпущен JWT (поле name JWT)
aud	VARCHAR, NOT NULL	Перечень доступных FQDN сервисов (CSV)
jti	VARCHAR, NOT NULL	UUIDv7 идентификатор JWT токена (поле jti JWT)
iss	VARCHAR, NOT NULL	AKI — хэш ключа проверки подписи JWT, 20 байт, HEX (поле iss JWT)
sub	VARCHAR, NOT NULL	SKI — хэш открытого ключа клиента, 20 байт, HEX (поле sub JWT)
role	VARCHAR, NOT NULL	Пользовательские роли (CSV)
perm	VARCHAR, NOT NULL	Атомарные права (CSV)
priv	VARCHAR, NOT NULL, DEFAULT ‘’	Привилегии по использованию шаблонов (CSV) (v2.1.x)
revoked	BOOLEAN, NOT NULL, DEFAULT FALSE	Признак блокировки клиента

Наименование поля	Тип данных	Описание
JWT	VARCHAR, NOT NULL	ASCII представление JWT (три части разделенные символом “.”)
key	VARCHAR, NOT NULL	ASN.1 представление открытого ключа клиента в PEM формате

3.7.6. Структура таблицы EVT

Наименование поля	Тип данных	Описание
id	BIGSERIAL	Уникальный идентификатор события в БД, первичный ключ
dtime	TIMESTAMPTZ, NOT NULL, DEFAULT NOW()	Время на момент формирования ответа и события
status	status_evt, NOT NULL	Тип запроса/события
errno	INT, NOT NULL, DEFAULT 0	Код ошибки WEB API (0 — успех)
csr_id	BIGINT, NOT NULL, DEFAULT 0	Идентификатор связанного с событием запроса
cert_id	BIGINT, NOT NULL, DEFAULT 0	Идентификатор связанного с событием сертификата
crl_id	BIGINT, NOT NULL, DEFAULT 0	Идентификатор связанного с событием CRL
templ_id	BIGINT, NOT NULL, DEFAULT 0	Идентификатор связанного с событием шаблона
JWT_id	BIGINT, NOT NULL, DEFAULT 0	Идентификатор связанного с событием JWT токена (v2.0.x+)
client_id	BIGINT, NOT NULL, DEFAULT 0	Идентификатор JWT токена действующего лица (v2.0.x+)
crypto_err	VARCHAR, NOT NULL	Вывод потока ошибок криптопрвайдера (OpenSSL)
request	VARCHAR, NOT NULL	Исходный JSON HTTP запрос

Возможные значения поля Status:

1) «CA», запрос на выпуск сертификата.
2) «REVOKE», отзыв сертификата.

3) «STATUS», запрос статуса сертификата.
4) «GET», запрос сертификата из БД.
5) «CRL», запрос CRL.
6) «DELTA_CRL», запрос DeltaCRL.
7) «UPDATE_CRL», формирование CRL.
8) «UPDATE_DELTA_CRL», формирование DeltaCRL.
9) «FIND», поиск сертификата в базе ЦС.
10) «DELETE», удаление сертификата из БД.
11) «EXPORT», экспорт сертификатов.
12) «CSR», запрос CSR по сертификату.
13) «REPAIR», восстановление отозванного сертификата.
14) «ADD_TEMPL», добавлен/обновлен шаблон.
15) «DEL_TEMPL», удален шаблон.
16) «PING», операция "ping" (зарезервировано, но не используется).
17) «TEMPLATES», запрос шаблонов.
18) «STAT», запрос статистики использования БД.
19) «GET_TEMPL», запрос шаблона.
20) «MAKE_JWT», создание нового JWT токена.
21) «REVOKE_JWT», отзыв JWT токена.
22) «REPAIR_JWT», реабилитация отозванного JWT токена.
23) «DEL_JWT», удаление JWT токена из БД.
24) «GET_JWT», запрос JWT токена из БД.
25) «RENEW_JWT», перевыпуск JWT токена с новым сроком действия.
26) «REVIVE_JWT», перевыпуск просроченного JWT токена.
27) «CHAIN», запрос цепочки сертификат.