ИНТЕРКОННЕКТ

Функциональные характеристики ПО "CPaaS"

Версия 1.0

28.06.2026

ООО «Интерконнект»

Оглавление

1 Термины и определения

Абонент:	Клиент сотовой связи, по которому предполагается отслеживать сетевые события. Абонент может быть клиентом домашнего оператора или других операторов
Автоматизированная система:	система, реализующая информационную технологию выполнения установленных функций с помощью персонала и комплекса средств автоматизации
ООО:	Общество с ограниченной ответственностью
ПО:	программное обеспечение
CAMEL:	(Customized Application for Mobile Enhanced Logic) стандартизованный протокол для использования в мобильных интеллектуальных платформах
CAP:	(CAMEL Application Part) прикладной протокол CAMEL
CdPN:	(Called Party Number) номер вызываемого абонента
CgPN:	(Calling Party Number) номер вызывающего абонента
GMSC:	(Gateway Mobile Switching Center) шлюзовой центр коммутации подвижной службы сети GSM. Узел, принимающий входящий вызов и определяющий, на какой обслуживающий MSC (Serving MSC) отправлять вызов
GSM:	(Global System for Mobile Communications) глобальный стандарт цифровой мобильной сотовой связи с разделением каналов по времени и частоте
gRPC:	(Remote procedure calls) система удаленного вызова процедур
HVR:	(Home visitor register) домашний регистр местоположения
HAProxy:	серверное программное обеспечение для реализации высокой доступности и балансировки нагрузки для TCP и HTTP-приложений, посредством распределения входящих запросов на несколько обслуживающих серверов. См. https://www.haproxy.org/
HTTP:	(HyperText Transfer Protocol) протокол передачи гипертекста
IMSI:	(International Mobile Subscriber Identity) международный идентификатор мобильного абонента (индивидуальный номер абонента)
IMEI:	(International Mobile Equipment Identity) международный идентификатор оборудования абонентской радиостанции
JSON:	(JavaScript Object Notation) текстовый формат обмена данными, основанный на JavaScript
MSC:	(Mobile switching center) центр коммутации
MT:	(Mobile Terminated) входящий вызов
MO:	(Mobile Terminated) исходящий вызов
MSISDN:	(Mobile Station Integrated Services Digital Network) номер мобильной станции в формате E.164
NAI:	(Nature of address indicator) индикатор типа адреса
OSS:	(Operation system support) система поддержки операций
Protocol Buffers:	язык описания интерфейса для системы удаленного вызова процедур
SCCP:	(Signalling connection control part) часть управления сигнальными соединениями
SCP:	(Signaling control point) узел управления услугами
STP:	(Signaling transfer point) узел маршрутизации сигнальных сообщений
TCAP:	(Transaction capabilities application part) прикладная часть возможностей транзакций
TON:	(Type of number) тип номера
VLR:	(Visitors location register) визитный регистр местоположения

2 Введение

Настоящий документ содержит описание функциональных характеристик программного обеспечения и информацию, необходимую для установки и эксплуатации программного обеспечения **«CPaaS»** производства компании ООО «Интерконнект». Структура и способ изложения материала предполагают наличие у читателя рабочих знаний UNIX-подобных операционных систем, а также основ построения сетей связи общего пользования. Документ предназначен для сотрудников Министерства цифрового развития, связи и массовых коммуникаций Российской Федерации (Минцифры России), составлен в соответствии с методическими рекомендациями по подготовке заявок на включение ПО в Единый реестр российского программного обеспечения.

3 Назначение комплекса

ПО **«CPaaS»** представляет собой программную платформу для реализации подхода CPaaS (Communication Platform as a Service) - коммуникационной платформы как услуги. ПО **«CPaaS»** предназначено для использования на сетях операторов подвижной и фиксированной телефонной связи, а также на внутренних корпоративных сетях юридических лиц - от медицинских клиник и магазинов до холдингов, имеющих в распоряжении собственные выделенные сети LTE/5G.

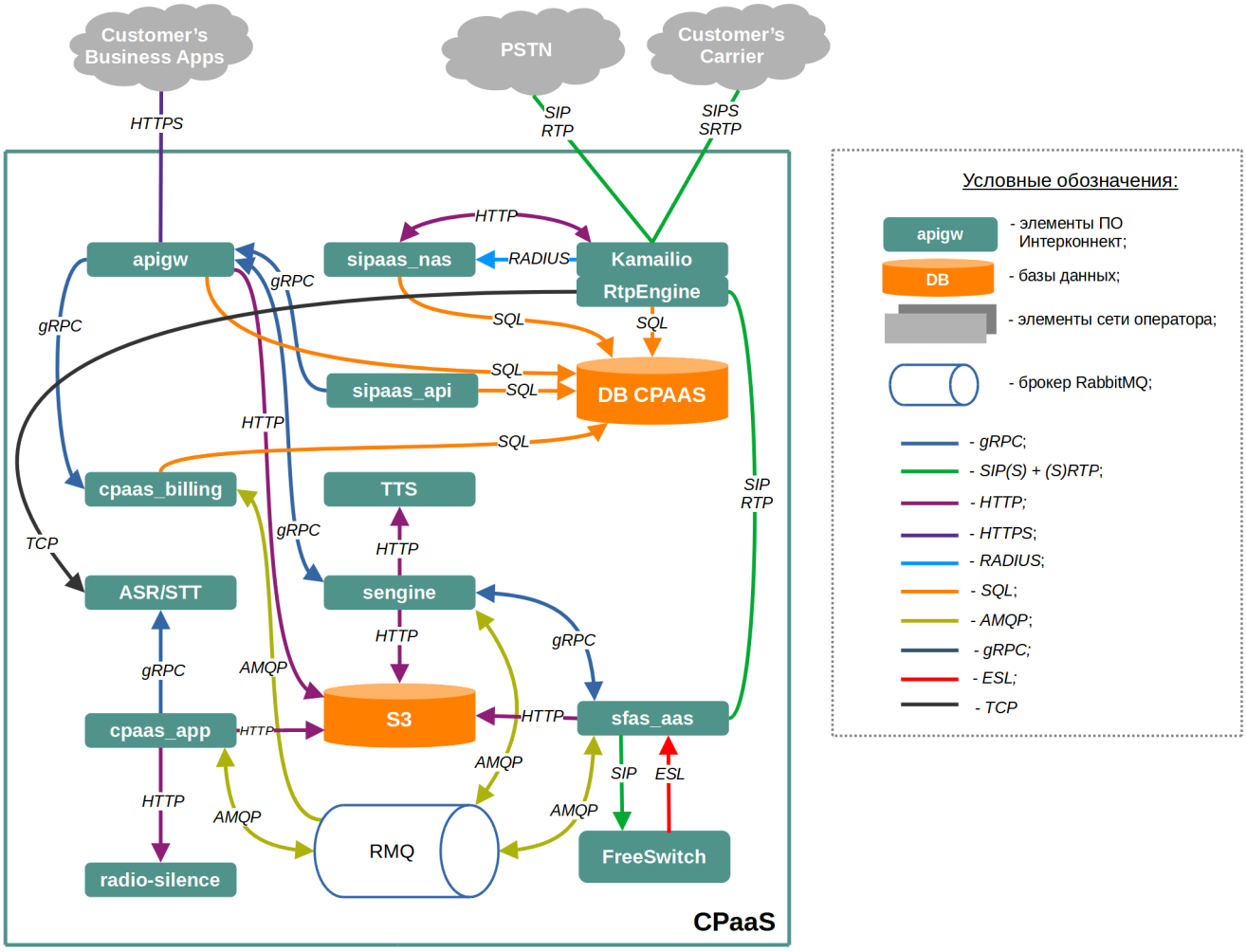
ПО **«CPaaS»** применяется на сетях мобильной связи стандартов GSM, UMTS, LTE, имеющих множество географически распределенных узлов связи.

ПО **«CPaaS»** позволяет:

1. Оказывать услуги телефонной связи и технологически связанные с ними сервисы: запись разговоров, VoLTE-этикетки.
2. Предоставлять функциональность:
    - STT (Speech-to-Text) — распознавание и транскрибирование аудиофайлов;
    - Streaming ASR (Automatic Speech Recognition) — онлайн/потоковое распознавание фраз/слов для построения IVR-решений и интеллектуальных голосовых ассистентов;
    - TTS (Text-to-Speech) — генерация голосовых фраз из текста;
    - прочие сервисы, которые доступны у отдельных поставщиков, работающих в области искусственного интеллекта или машинного обучения (AI/ML).
3. Строить более высокоуровневые продукты на базе готовых примитивов стандартизированных API - Voice API и SMS API:
    - Скрытие реальных номеров при общении клиентов компании (например, телефонные звонки по онлайн-объявлениям);
    - Построение решений по аутентификации/верификации абонентов: двухфакторная аутентификация - 2FA, одноразовые пароли - OTP и т.п.;
    - Low/No Code решения - визуальные конструкторы IVR и т.п.;
    - Виртуальные Ассистенты;
    - Контакт-центры.
4. Организовать среду для разработки (SDK) клиентом дополнительных сервисов и сценариев на следующих языках программирования: C#, Java, JavaScript (Node.js), PHP, Python, Ruby, Go.
5. Масштабировать **ПО «CPaaS»** по мере развития сети оператора связи или клиента.
6. Осуществлять мониторинг работоспособности платформы в режиме реального времени.
7. Плавно интегрироваться в существующие каналы связи и бизнес-приложения оператора связи или клиента.

Часть функций или услуг может предоставляться существующими программными системами или подразделениями на сети оператора связи или организации - ценность объединения перечисленного функционала в ПО **«CPaaS»** состоит в возможности его представления в рамках целостного и единообразного API.

3.1 Архитектура системы



Архитектура ПО **«CPaaS»**

3.2 Состав ПО комплекса

В состав ПО **«CPaaS»** входят следующие программные компоненты:

- **apigw** — API Gateway, точка входа для запросов бизнес-приложений заказчика по HTTPS (Voice API/SMS API); взаимодействует с базой данных **DB CPAAS** по SQL и с сервисами **cpaas_billing** и **sengine** по gRPC.
- **Kamailio/RtpEngine** — SIP-сервер и медиарелей; принимает сигнальный (SIP/SIPS) и медийный (RTP/SRTP) трафик от сетей PSTN и оператора связи (Customer's Carrier), взаимодействует с **sipaas_nas** по RADIUS, записывает данные в **DB CPAAS** по SQL и передает SIP/RTP трафик сервису **sfas_aas**.
- **sipaas_nas** — компонент учета сетевого доступа (NAS) SIP-подсистемы; обменивается данными с **apigw** по HTTP, с Kamailio по RADIUS и с **DB CPAAS** по SQL.
- **sipaas_api** — API-компонент SIP-подсистемы, обеспечивающий чтение и запись данных в **DB CPAAS** по SQL.

- **DB CPAAS (PostgreSQL)** — реляционная СУБД, центральное хранилище конфигурационных, справочных данных и данных о вызовах; используется компонентами **apigw**, **cpaas_billing**, **sipaas_nas**, **sipaas_api** и **Kamailio/RtpEngine**.
- **cpaas_billing** — модуль биллинга; взаимодействует с **apigw** по gRPC, с **DB CPAAS** по SQL.
- **sengine** — сервисный движок обработки сценариев; взаимодействует с **apigw** по gRPC, с сервисом **TTS** и хранилищем **S3** по HTTP, а также со сервисом **sfas_aas** по AMQP/gRPC.
- **TTS** — сервис синтеза речи (Text-to-Speech), предоставляющий **sengine** аудиофайлы голосовых сообщений по HTTP.
- **S3** — объектное хранилище, совместимое с S3, предназначенное для хранения медиафайлов (голосовые приветствия, аудиозаписи разговоров, файлы сценариев); используется компонентами **cpaas_app**, **cpaas_billing**, **sengine** и **sfas_aas**.
- **cpaas_app** — клиентское приложение платформы; взаимодействует с сервисом распознавания речи **ASR/STT** по gRPC, с хранилищем **S3** по HTTP и с сервисом **radio-silence** по HTTP.
- **VOSK server** — сервер распознавания речи (STT), развертывается в составе комплекса в нескольких экземплярах; используется компонентами **cpaas_app** (по gRPC) и **sfas_aas** (по протоколу IFA/gRPC).
- **radio-silence** — сервис определения тишины в аудиопотоке, взаимодействует с **cpaas_app** по HTTP.
- **sfas_aas** — сервис распознавания и обработки речи как услуги; взаимодействует с хранилищем **S3** по HTTP, с брокером сообщений **RMQ** по AMQP, с **sengine** по gRPC, с **Kamailio/RtpEngine** по SIP/RTP.
- **RMQ (RabbitMQ)** — брокер сообщений, обеспечивающий асинхронное взаимодействие компонентов **cpaas_app** и **sfas_aas** по протоколу AMQP.

Взаимодействие с внешними системами (бизнес-приложения заказчика, сети PSTN и оператора связи) осуществляется по протоколам HTTPS и SIP(S)/(S)RTP.

4 Системные требования

Аппаратные требования ПО **«CPaaS»** складываются из требований к узлам кластера Kubernetes, на котором развертывается основная часть компонентов ПО, и требований к отдельному хосту для компонентов **sfas_aas**, **Kamailio** и **RtpEngine**, разворачиваемых вне кластера Kubernetes.

4.1 Требования к кластеру Kubernetes

Минимальная отказоустойчивая конфигурация кластера Kubernetes включает 3 ноды со следующими характеристиками каждая:

Таблица 1. Минимальные требования к узлу кластера Kubernetes

Характеристика	Значение
Количество узлов (нод)	3
Количество ядер центрального процессора (на узел)	Не менее 2
Тактовая частота центрального процессора	Не менее 2.00GHz
Объем оперативной памяти (на узел)	Не менее 4Gb
Операционная система	РЕД ОС и другие ОС семейства Linux

Примечание:

При развертывании кластера в однонодовой конфигурации (без отказоустойчивости) суммарные требования к единственному узлу составляют не менее 6 vCPU и не менее 12Gb оперативной памяти — то есть сумма ресурсов трех нод минимальной конфигурации.

4.2 Требования к хосту компонентов sfas / Kamailio / RtpEngine

Компоненты **sfas_aas**, **Kamailio** и **RtpEngine**, отвечающие за прием и обработку SIP/RTP-трафика, разворачиваются на отдельном хосте вне кластера Kubernetes.

Таблица 2. Минимальные требования к хосту sfas / Kamailio / RtpEngine

Характеристика	Значение
Количество ядер центрального процессора	Не менее 2
Тактовая частота центрального процессора	Не менее 2.00GHz
Объем оперативной памяти	Не менее 4Gb
Операционная система	РЕД ОС и другие ОС семейства Linux

Указанные конфигурации являются минимальными и могут быть увеличены в зависимости от ожидаемой нагрузки (количества единовременных вызовов, объема обрабатываемого голосового трафика и подключаемых сервисов AI/ML).

5 Установка CPaaS

Для выполнения установки войдите в систему с правами, достаточными для выполнения установки ПО. Если работа ведется в графическом режиме, то вызовите окно терминала, и все дальнейшие действия должны выполняться в командной строке терминала.

5.1 Описание установочного пакета

В состав установочного пакета входят:

- Helm-чарты (или HelmChart-манифесты) инфраструктурного уровня: Kyverno, MetalLB, cert-manager, CSI S3, CNPG Operator (с плагином резервного копирования), операторы RabbitMQ (cluster/topology), Traefik;
- Helm-чарты прикладного уровня CPaaS: **cpaas-app** (основное приложение), **billing** (биллинг), **api-gw** (API Gateway), **sengine** (сервисный движок сценариев), **copula-webui** (веб-консоль управления);
- вспомогательные чарты инициализации базы данных: миграция схемы и наполнение справочных данных (тарифы и цены на услуги);
- образы контейнеров из корпоративного реестра;
- набор Kubernetes-манифестов для конфигурации операторов, топологии RabbitMQ (виртуальные хосты, обменники, очереди, правила) и прочих ресурсов.

Компоненты **sfas_aas**, **Kamailio** и **RtpEngine** в состав установочного пакета Kubernetes не входят и устанавливаются отдельно, на выделенном хосте вне кластера (см. раздел «Требования к хосту компонентов sfas / Kamailio / RtpEngine»); их сетевые адреса указываются в параметрах конфигурации **api-gw** и **sengine**.

> *Примечание:*
>
> ПО **«CPaaS»** при поставке Заказчику не является коробочным решением, установка, как правило, осуществляется по заранее оговоренной и согласованной процедуре с задействованием имеющейся инфраструктуры Заказчика. Также в зависимости от требований конкретного Заказчика набор функционала ПО **«CPaaS»** может сильно варьироваться. Описанный в данном документе функционал необходимо рассматривать как условно-базовый. Инсталляция и настройка ПО осуществляется квалифицированными сотрудниками ООО «Интерконнект» при содействии технических специалистов Заказчика. Настройка и конфигурирование, сопровождение эксплуатации решения также осуществляется сотрудниками Службы Поддержки ООО «Интерконнект».

5.2 Установка на Kubernetes

Для развертывания ПО **«CPaaS»** в промышленной среде используется установка в Kubernetes с использованием MetalLB в качестве балансировщика нагрузки. Подготовка и установка выполняются в несколько этапов.

1. Подготовка кластера Kubernetes:
    - создаются namespace `infrastructure` и `cpaas`;
    - выбирается или настраивается default storage class;
    - в оба namespace добавляется секрет для доступа к container registry (`sifox-imagepull-secret`);

- устанавливается инструмент автоматизированного развертывания Helm-чартов (например, helm-controller) и Kyverno с политикой автоматического добавления секрета доступа к registry во все поды инфраструктурных чартов;
- устанавливаются вспомогательные компоненты: MetalLB, cert-manager, CSI S3, CNPG Operator, операторы RabbitMQ (cluster/topology), Traefik.

2. Настройка сетевой и управляющей инфраструктуры:

   - для MetalLB настраивается пул внешних IP-адресов из свободной подсети;
   - устанавливается и настраивается класс Ingress Traefik для namespace `cpaas`.

3. Развертывание базовых сервисов ПО **«CPaaS»**:

   - создаются секреты с учетными данными для PostgreSQL;
   - разворачивается кластер PostgreSQL (`cpaas-rdbms`) в namespace `cpaas` и ожидается достижение состояния `healthy`; при необходимости внешнего доступа создаются сервисы LoadBalancer;
   - развертывается кластер RabbitMQ (`cpaas-rmq`) и применяется топология (виртуальный хост, пользователь, права, обменники, очереди и правила).

4. Инициализация базы данных:

   - выполняется миграция схемы базы данных;
   - выполняется наполнение справочных данных — тарифы и цены на услуги для биллинга.

5. Развертывание прикладных компонентов CPaaS:

   - последовательно устанавливаются Helm-чарты **cpaas-app**, **billing**, **api-gw**, **sengine** и **copula-webui**;
   - в конфигурации **api-gw** и **sengine** указываются адреса внешних узлов **sipaas** и **sfas_aas** соответственно.

6. Проверка результата:

   - проверяется состояние всех подов и кластеров PostgreSQL/RabbitMQ;
   - убеждаются, что сервисы доступны по назначенным адресам и портам;
   - проверяется отсутствие ошибок в журналах развернутых компонентов.

Такой подход позволяет развернуть ПО **«CPaaS»** как часть контейнерной инфраструктуры и использовать стандартные механизмы масштабирования, отказоустойчивости и управления ресурсами Kubernetes.

5.3 Установка компонентов вне кластера Kubernetes

Компоненты **sfas_aas**, **Kamailio** и **RtpEngine**, отвечающие за прием и обработку сигнального (SIP) и медийного (RTP) трафика, в состав Kubernetes-инсталляции не входят и устанавливаются отдельно, на выделенном хосте вне кластера (аппаратные требования к такому хосту приведены в разделе «Требования к хосту компонентов sfas / Kamailio / RtpEngine»).

Установка выполняется с использованием предоставляемого набора ansible-скриптов, после предварительной корректировки переменных и шаблонов ansible в соответствии с целевым окружением:

```
# Установка FreeSWITCH и sfas_aas
ansible-playbook --vault-password-file=.vault.pass -u <имя пользователя> -v -i hosts
cpaas-not-k8s.yml

# Установка Kamailio, RtpEngine и дополнительных компонентов
ansible-playbook --vault-password-file=.vault.pass -u <имя пользователя> -v -i hosts
kamailio.yml
```

где `<имя пользователя>` — учётная запись на целевых хостах с доступом к повышению привилегий (`sudo`), а `.vault.pass` — файл с GPG-ключом для расшифровки конфиденциальных данных, зашифрованных механизмом Ansible Vault.

Интеграция этих компонентов с остальной частью платформы, развернутой в Kubernetes, выполняется через параметры конфигурации соответствующих Helm-чартов:

- в конфигурации **api-gw** (`grpcClients.sipaas.host`) указывается адрес узла, на котором развернут **Kamailio/sipaas**;
- в конфигурации **sengine** (`Sengine.sfas.channels[].ip`) указывается адрес узла, на котором развернут **sfas_aas**;
- обмен данными о записях разговоров и вызовах между **sfas_aas** и остальными компонентами платформы выполняется через отдельную топологию очередей брокера сообщений RabbitMQ (`q_sfas_filer_sfas-0` — передача записей разговоров, `q_cdr_ext` — передача данных о вызовах для биллинга).

6 Настройка ПО «CPaaS»

Настоящий раздел описывает прикладную настройку ПО **«CPaaS»** после установки: создание аккаунта, подключение номеров и настройку SIP-транков для исходящих и входящих вызовов. Параметры развёртывания компонентов (Helm-чарты, Kubernetes-манифесты) описаны в разделе «Установка CPaaS».

6.1 Создание аккаунта через веб-консоль

1. Открыть в браузере административный раздел веб-консоли по адресу `http://<WEBUI_HOST>:<PORT>/adm/`.
2. В открывшейся форме ввести API-ключ, заданный при установке компонента **copula-webui**.
3. В поле «Friendly Name» указать произвольное имя создаваемого аккаунта.
4. Нажать «Создать» — будет сгенерирована ссылка на форму регистрации.
5. Перейти по ссылке и завершить регистрацию, указав адрес электронной почты и пароль.

6.2 Настройка посредством API

Настройка аккаунта выполняется напрямую через базу данных и API.

Идентификатор аккаунта (`Account SID`) отображается в веб-консоли в разделе «Профиль». Токен аутентификации (`Auth Token`), используемый для авторизации запросов к API, можно получить из базы данных (таблица `cpaas.accounts`); значение токена конфиденциально и не должно передаваться или храниться в открытом виде.

Для работы с исходящими вызовами в справочники базы данных добавляется поддерживаемая страна и номер, с которым платформа регистрируется у оператора связи (таблицы `cpaas.countries` и `cpaas.available_phone_numbers`).

Адрес обращения к API (`APIGW_ENDPOINT`) зависит от способа публикации сервиса **api-gw** — через `Service` типа `LoadBalancer` (порт, как правило, `4000`) либо через `Ingress` (порт `80`). Для выполнения запросов к API используются переменные окружения:

```
export APIGW_ENDPOINT="http://<LB_VIP>"
export APIGW_AUTH="<ACCOUNT_SID>:<AUTH_TOKEN>"
```

Проверка доступа (получение списка подключённых номеров):

```
curl -X GET "$APIGW_ENDPOINT/2010-04-01/Accounts/<ACCOUNT_SID>/IncomingPhoneNumbers" \
  -H 'accept: application/json' \
  -u $APIGW_AUTH | jq '.'
```

6.3 Настройка SIP-транка для исходящих вызовов

1. Создать ресурс **ConnectionPolicy** — параметры регистрации на SIP-домене оператора связи (адрес домена, номер, пароль, realm).
2. Создать ресурс **ConnectionPolicyTarget** — SIP-адрес узла оператора связи, на который отправляются вызовы.
3. Создать ресурс **BYOC Trunk**, связанный с созданной политикой подключения.
4. Добавить SIP-домен, используемый для исходящих вызовов через созданный транк.
5. Убедиться, что созданный транк связан с доменом (проверка через таблицу `cpaas.trunks`).

6.4 Настройка для входящих вызовов

1. Добавить SIP-домен для входящих вызовов — в качестве имени домена указывается внешний адрес интерфейса SBC-узла, принимающего SIP-трафик от оператора связи.
2. Создать ресурс **SipIpAccessControlList** (список контроля доступа по IP-адресам).
3. Привязать список контроля доступа к SIP-домену (**SipIpAccessControlListMapping**).
4. Добавить в список контроля доступа IP-адрес, с которого оператор связи отправляет SIP-сообщения.

Указанные операции выполняются через REST API компонента **apigw** (ресурсы `ConnectionPolicies`, `ByocTrunks`, `SIP/Domains`, `SIP/IpAccessControlLists`); конкретные идентификаторы ресурсов, адреса и учётные данные операторов связи индивидуальны для каждой инсталляции и не входят в состав настоящего документа.

6.5 Запуск, перезапуск и завершение работы

Запуск, перезапуск и остановка ПО **«CPaaS»** осуществляются стандартными командами Kubernetes с использованием **kubectl**. Для запуска сервиса необходимо выполнить команды:

```
# 1. Запустить PostgreSQL (снять с гибернации)
kubectl annotate cluster cpaas-rdbms -n cpaas cnpg.io/hibernate-

# 2. Дождаться готовности БД
kubectl -n cpaas wait --for=condition=Ready cluster/cpaas-rdbms --timeout=300s

# 3. Запустить RabbitMQ
kubectl scale rabbitmqcluster cpaas-rmq -n cpaas --replicas=1

# 4. Дождаться готовности RabbitMQ
kubectl -n cpaas wait --for=condition=Ready rabbitmqcluster/cpaas-rmq --timeout=300s

# 5. Запустить приложение **cpaas-app**
kubectl scale deployment cpaas-app -n cpaas --replicas=1
```

Для перезапуска сервиса необходимо выполнить команды:

```
# Перезапустить приложение **cpaas-app**
kubectl rollout restart deployment cpaas-app -n cpaas

# Перезапустить RabbitMQ
kubectl delete pod -l app.kubernetes.io/name=cpaas-rmq -n cpaas

# Перезапустить PostgreSQL (с сохранением данных)
kubectl -n cpaas cnpg restart cpaas-rdbms
```

Для остановки сервиса необходимо выполнить команды:

```
# 1. Остановить приложение **cpaas-app**
kubectl scale deployment cpaas-app -n cpaas --replicas=0

# 2. Остановить RabbitMQ
kubectl scale rabbitmqcluster cpaas-rmq -n cpaas --replicas=0

# 3. Перевести PostgreSQL в гибернацию
kubectl annotate cluster cpaas-rdbms -n cpaas --overwrite cnpg.io/hibernate=true
```

6.6 Обновление и удаление

Обновление и удаление ПО **«CPaaS»** выполняются с помощью штатных механизмов Kubernetes и Helm.

Обновление конфигурации:

```
helm upgrade -f values.yaml
```

Обновление версии:

```
helm upgrade --version X -f values.yaml
```

Откатить версию:

```
helm rollback
```

Для удаления ПО **«CPaaS»** выполните команду:

```
kubectl delete namespace cpaas
```

Для удаления инфраструктуры выполните команду:

```
helm uninstall + kubectl delete ns
```

6.7 Проверка работоспособности

Проверка работоспособности ПО **«CPaaS»** осуществляется путем проверки текущего состояния всех компонентов.

Для проверки текущего состояния всех компонентов необходимо обладать соответствующими правами. Проверить текущего состояния всех компонентов можно путем исполнения команд:

```
# Все поды платформы
kubectl get pods -n cpaas

# Все поды инфраструктуры
kubectl get pods -n infrastructure

# Статус кластеров
kubectl -n cpaas get cluster,rabbitmqcluster

# Логи всех подов cpaas-app (одной командой)
kubectl -n cpaas logs -l app=cpaas-app --tail=50
```