ВКС

Архитектура ВКС

Портал с документацией: https://biz.mail.ru/docs/on-premises/

Оглавление

Термины и определения

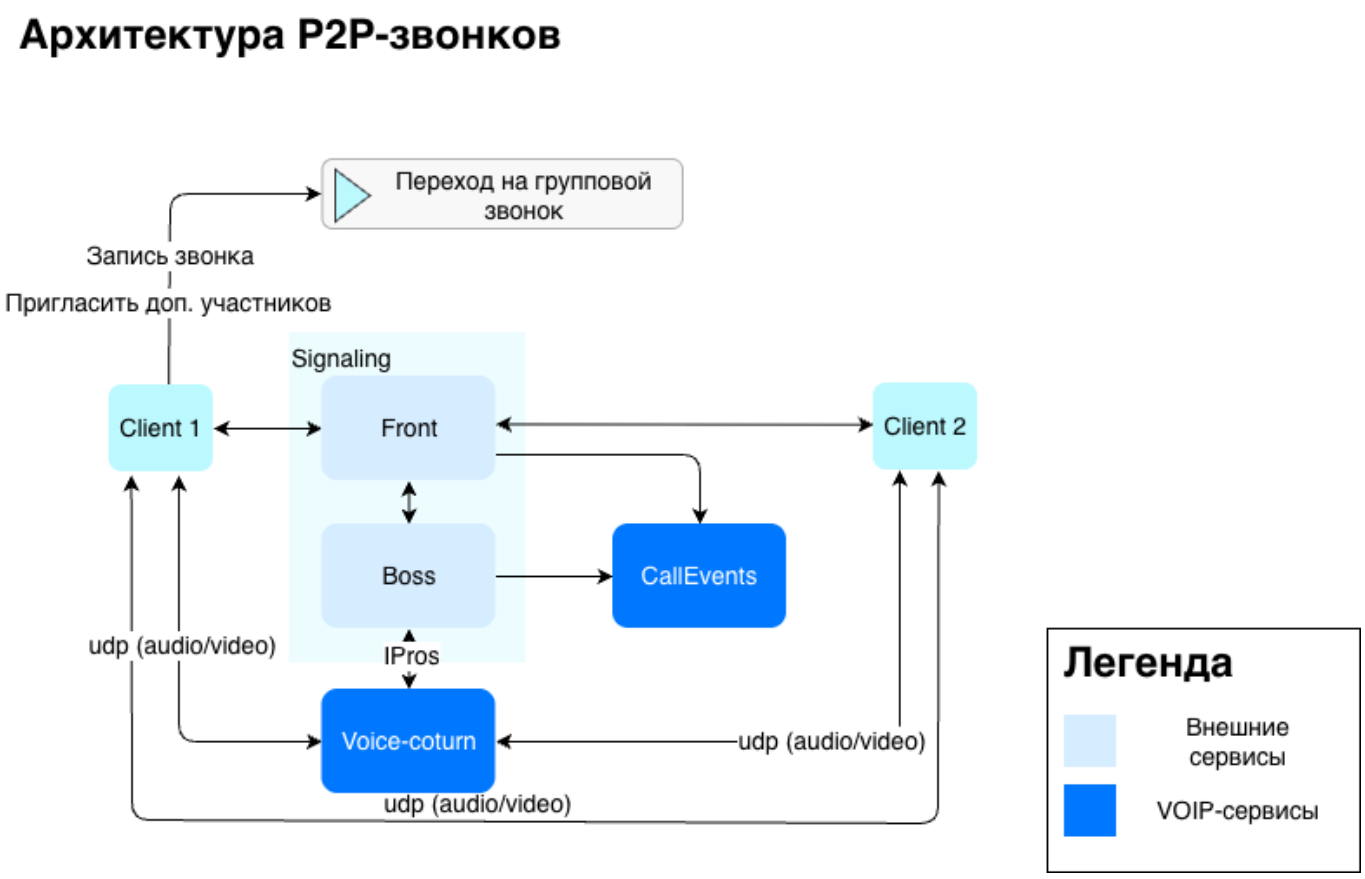
Сигналинг — процесс обмена служебной информацией между устройствами для установления, управления и завершения соединения.

Media Plane (плоскость передачи медиаданных) в интернет-звонках — функциональная часть сетевой архитектуры, отвечающая непосредственно за перенос оцифрованного голоса и видео между участниками. В VK WorkSpace предпочтительно осуществляется по протоколу UDP для обеспечения минимальной задержки.

SIP (Session Initiation Protocol) — основной протокол сигнализации в IP-телефонии, предназначенный для управления и завершения сеансов связи в реальном времени, включая голосовые вызовы, видеоконференции и обмен сообщениями.

P2P-звонки

Ощая схема архитектуры P2P-звонков приведена ниже:



Front — gateway для клиентских запросов.

Boss(сервис Мессенджера) — хранит пользовательские сессии, очередь событий, токен и копию контакт-листа клиента.

Voice-coturn — TURN-сервер для p2p звонков (NAT traversal). В P2P-звонках соединение производится между двумя клиентами напрямую.

CallEvents — сервис обработки событий звонков: принимает события, сохраняет их в БД для последующей статистики и отчётности. Аналитика доступна в Grafana.

> **Примечание**
>
> При старте звонка клиент №1 выполняет специальный запрос и получает внешний IP адрес и порт сервиса **Voice-coturn**. Эти параметры используются как резервный маршрут, если установить прямое соединение между клиентами невозможно.

Общее описание взаимодействия:

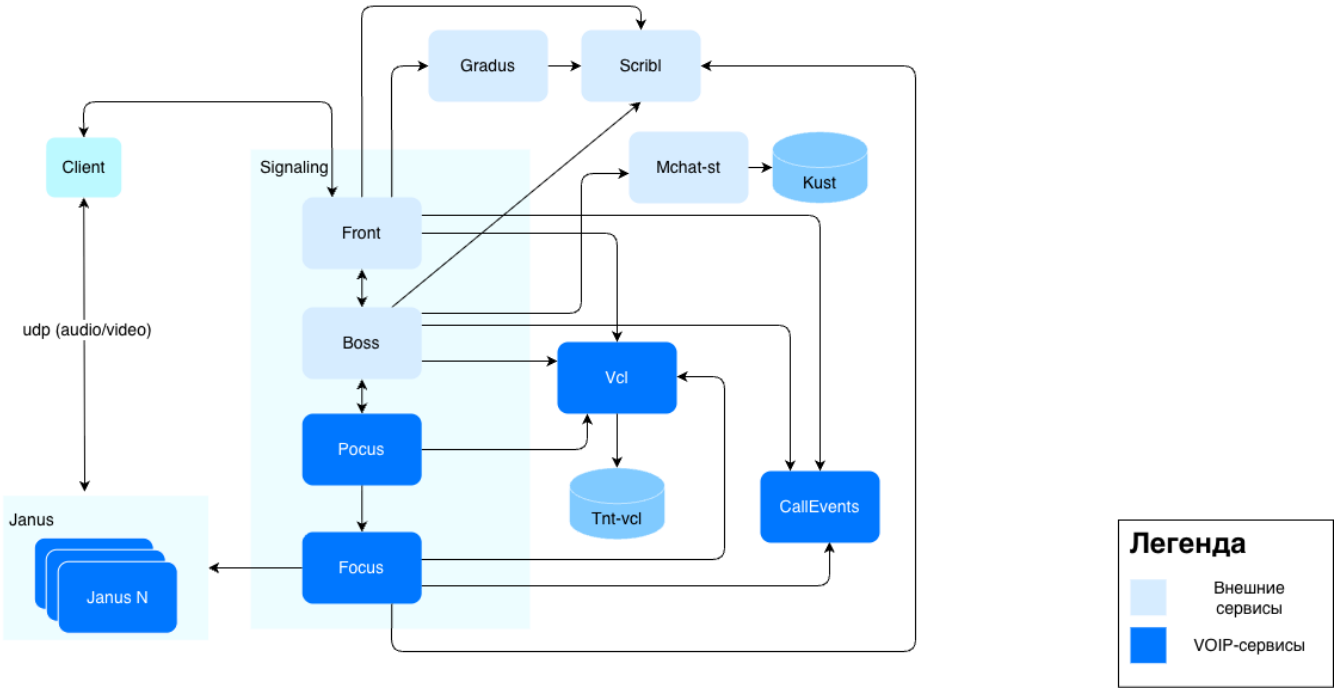
1. Клиент №1 инициирует вызов и отправляет сигнальное сообщение invite.
2. Сообщение invite поступает в сервис **Boss** на стороне клиента №1.
3. Далее сообщение доставляется в очередь **Boss** на стороне клиента №2.
4. Клиент №2 забирает сообщение (invite) с помощью механизма fetch event.
5. На клиенте №2 отображается окно входящего звонка.
6. Пользователь на клиенте №2 нажимает «Принять» — формируется сигнальное сообщение accept.
7. Сигнальное сообщение accept отправляется в сторону клиента №1 сервис **Boss**.
8. В сообщениях invite и accept передаются сетевые параметры (IP адреса и порты), необходимые для установления медиасоединения.
9. На основании этих параметров клиенты устанавливают прямое UDP соединение для обмена медиатрафиком.

> **Примечание**
>
> Каким образом будет производиться соединение, напрямую или через **Voice-coturn**, зависит от условий, выбирается максимально подходящий вариант соединения.

Групповые звонки

Схема групповых звонков:

Групповой звонок



Front — gateway для клиентских запросов.

Boss — хранит пользовательские сессии, очередь событий, токен и копию контакт-листа клиента.

Pocus — прокси-сервис шардинга и фейловера инстансов Focus.

Focus — сигналинг-сервер групповых звонков. Отвечает за хранение списка комнат, участников комнат, обслуживает процесс их добавления/удаления.

Janus — медиасервер.

Gradus — сервис для хранения статусов пользователей (состояние «В звонке»).

Scribl — сервис подписок, используется для подписки на состояние звонка в чате.

Mchat-st — сервис отправки сообщений о звонках.

Vcl — вспомогательный сервис для звонков по ссылке. Создаёт и хранит ссылки на звонок, проверяет их наличие.

CallEvents — принимает звонковые события и сохраняет в БД для статистики по звонкам. Аналитика доступна в Grafana.

Главное отличие от предыдущей схемы в том, что соединение производится не напрямую, а через медиасервер (инстансы сервисов **Janus**). Сигналинг обрабатывается сервисом **Focus**, который отвечает за хранение состояния группового звонка (комнаты). Он является stateful-сервисом, точкой правды о состоянии звонка. В случае перезагрузки инстанса **Focus** все групповые звонки на этом инстансе завершатся с ошибкой. Также **Focus** назначает, каким медиасервером (**Janus**) будет обслуживаться конкретный звонок.

Когда клиент хочет подключиться к групповому звонку, он отправляет сигнальное сообщение, в поле `destination` указывается контакт специального вида, который заканчивается на @focus. В сервисе **Boss**,

куда приходит это сообщение, есть логика, которая перенаправляет его через сервис **Pocus** в нужный инстанс **Focus**. Это позволяет закрепить конкретный звонок за инстансом **Focus**. У сервиса **Pocus** есть БД Tarantool, отвечающая за привязку идентификаторов звонков к инстансам **Focus**, однако на схеме она не представлена.

Первый клиент, подключившийся к комнате, создаёт запись о её существовании; после выхода последнего пользователя запись уничтожается. То есть данные хранятся только в то время, когда в звонке кто-то есть.

Подключение выглядит следующим образом: клиент отправляет специальное сигнальное сообщение ROOM_JOIN, **Focus** его обрабатывает, при необходимости создаёт комнату и возвращает ROOM_JOIN_ACK клиенту. В нём передаётся IP сервиса **Janus**, который будет обслуживать данную комнату. Клиент подключается по UDP к конкретному **Janus** для передачи своего аудио и видео и получения аудио и видео от других участников.

Медиасервер микширует (сводит) аудио у клиентов. Видео клиент получает в зависимости от того, что у него на экране. Например, в групповом звонке может быть несколько сотен пользователей, но на экран помещается только 10. Через сигналинг клиент запрашивает, каких пользователей он хочет видеть и в каком разрешении. Информация через **Focus** попадает в **Janus**, и **Janus** начинает транслировать эти видеопотоки клиенту.

Основная задача сервиса **Vcl** — хранение информации о ссылках на звонок. При подключении проверяется валидность ссылки с помощью БД Tarantool **tnt-Vcl**.

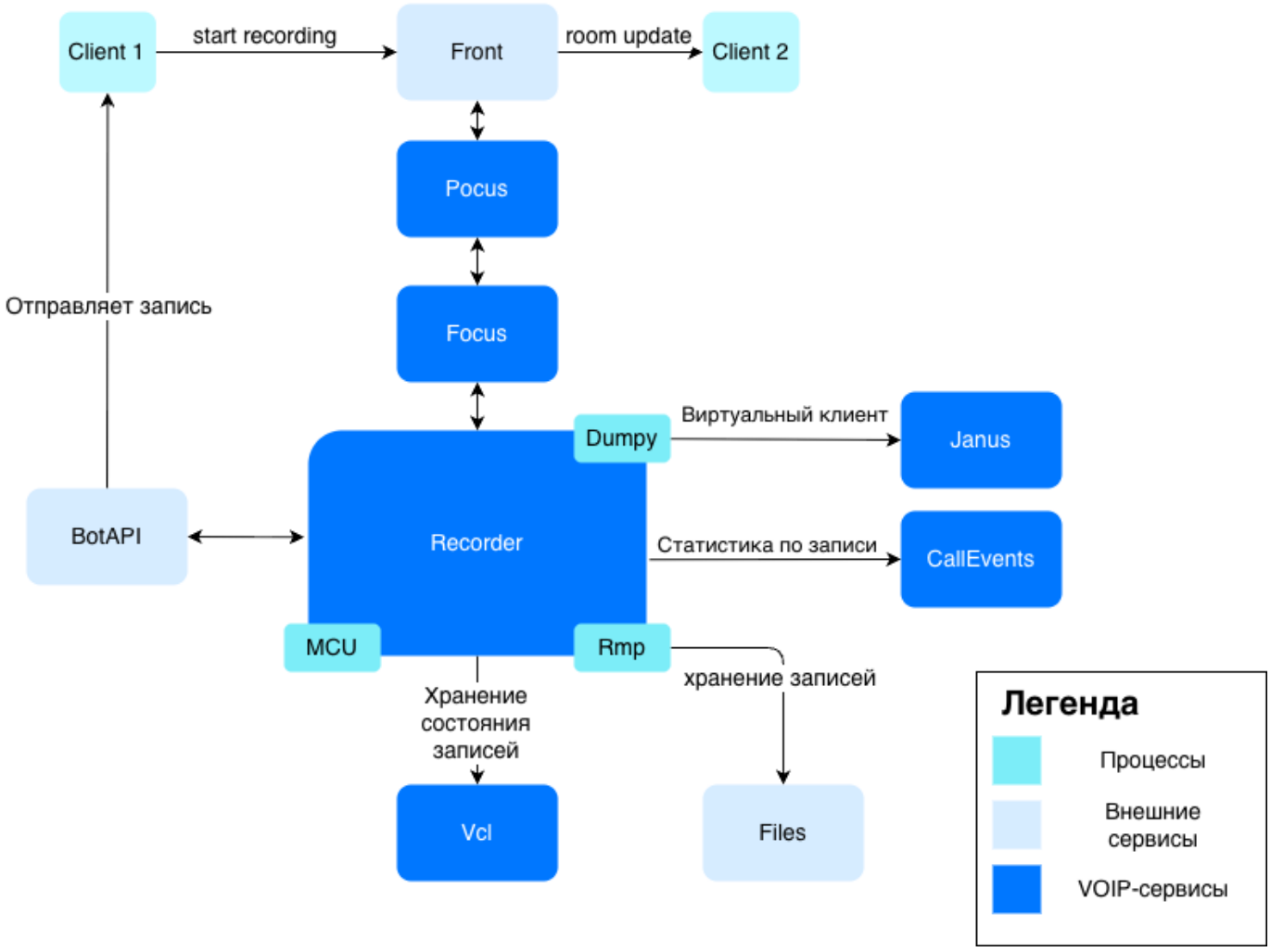
Есть механизм подписок на состояние звонка, его обеспечивает сервис **Scribl**. Существует такая функциональность, как звонок в групповой чат. Благодаря сервису подписок пользователи увидят, что в чате происходит групповой звонок. При вступлении в чат каждый пользователь автоматически подписывается на статусы звонков в чате через **Scribl**, а **Scribl** получает через **Focus** информацию о звонке: сколько там человек и есть ли вообще звонок.

При завершении звонка **Boss** отправляет информацию об этом в сервис **Mchat-st**, а он, в свою очередь отправляет,уведомление в чат.

Запись звонка

Схема записи звонков выглядит следующим образом:

Архитектура записей



Front — gateway для клиентских запросов.

Boss — хранит пользовательские сессии, очередь событий, токен и копию контакт-листа клиента.

Pocus — прокси, сервис шардинга и фейловера инстансов **Focus**.

CallEvents — принимает звонковые события и сохраняет в БД для статистики по звонкам. Аналитика доступна в Grafana.

Janus — медиасервер.

BotAPI — сервис автоматической отправки модератору встречи файла с записью звонка.

Vcl — вспомогательный сервис для звонков по ссылке. Создает и хранит ссылки на звонок, проверяет их наличие.

Recorder — сервис, управляющий записью конференций. Принимает запросы на управление записью (start/stop/pause/continue) от **Focus** по протоколу IPROS.

Dumpy — процесс сервиса **Recorder**. Виртуальный клиент, который подключается в звонок для записи.

Mcu — процесс сервиса **Recorder**. Микширует несколько стримов (потоков аудио и/или видео).

Rmp — процесс сервиса **Recorder**. Конвертирует RTP-трафик в формат MP4.

Files — сервис для работы с файлами, аватарами, стикерами.

Ключевой компонент подсистемы записи — сервис **Recorder**. Внутри него запускается несколько процессов, в том числе **Dumpy** — «виртуальный клиент», который подключается к звонку и пишет все медиапотоки комнаты.

Начало записи:

1. Когда пользователь нажимает кнопку записи, сигнальное сообщение STAR_RECORDING которое отправляется в соответствующий инстанс **Focus**.
2. **Focus**, обрабатывает запрос и вызывает к сервис **Recorder**, после чего Recorder запускает процесс **Dumpy**.
3. **Dumpy** подключается к звонку и медиасерверу **Janus**, обрабатывающему текущий звонок.
4. Запись всех стримов ведется в `RTP dump file` в сжатом виде.

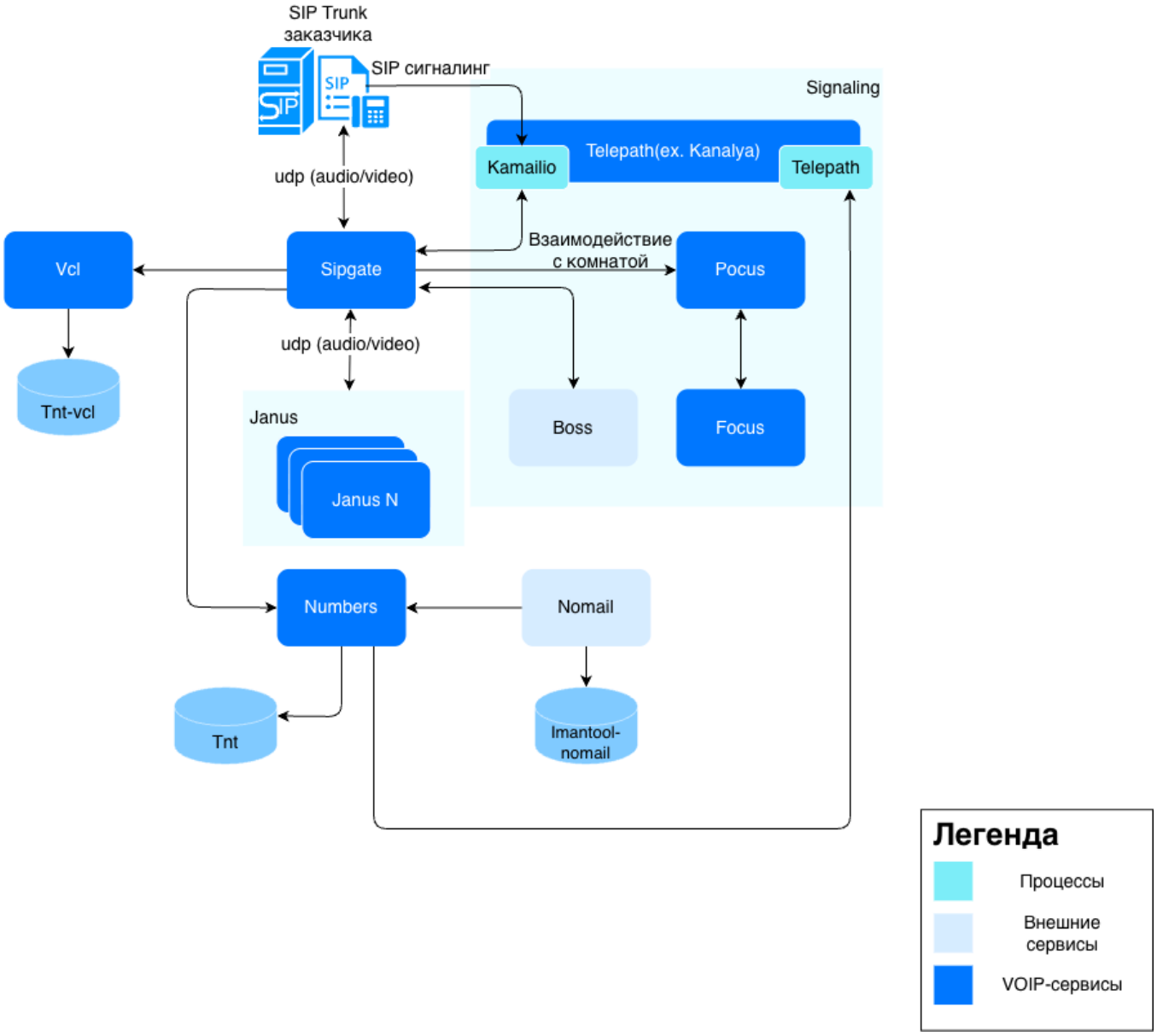
После завершения записи:

1. **Recorder** запускает процесс **Rmp**, который конвертирует `RTP dump file` в формат MP4.
2. Дополнительно используется процесс **Mcu**, который составляет «единую картину» из отдельных медиапотоков. По завершению записи:
3. Итоговый файл загружается в сервис **Files**, формируется ссылка на скачивание.
4. Ссылка отправляется инициатору записи через **BotAPI**.
5. Для локальной обработки **Recorder** требуется достаточный объём памяти, после загрузки файла в **Files** временные данные очищаются.
6. Статистика по записи сохраняется в сервисе **CallEvents**.

SIP-звонки

Схема SIP-звонков выглядит так:

SIP



Front — gateway для клиентских запросов.

Boss — хранит пользовательские сессии, очередь событий, токен и копию контакт-листа клиента.

Pocus — прокси, сервис шардинга и фейловера инстансов **Focus**.

Focus — сигналинг-сервер групповых звонков. Отвечает за хранение списка комнат, участников комнат, обслуживает процесс их добавления/удаления.

Vcl — вспомогательный сервис для звонков по ссылке. Создает и хранит ссылки на звонок, проверяет их наличие.

Janus — медиасервер.

Telepath — вспомогательный сервис для SIP-интеграции, отвечающий за регистрацию номеров в **Kamailio**.

Kamailio — SIP-сервер.

Sipgate — основной сервис для SIP-интеграции. Служит для подключения SIP-клиентов к конференциям Мессенджера VK WorkSpace. Принимает/совершает телефонные вызовы, подключает SIP-клиентов к конференциям в качестве гостей, передает аудио от SIP-клиента в медиасервер конференции с возможностью транскодирования.

Numbers — предоставляет номер телефона пользователя для совершения исходящего вызова и SN пользователя для входящего вызова.

Nomail — передает изменения из Keycloak в поле SIP в **Numbers**.

В этом разделе речь пойдет про интеграцию с SIP-инфраструктурой, которая размещена на стороне заказчика.

Ключевым для интеграции компонентом является **Sipgate**. Это gateway, который, с одной стороны, может представляться как SIP-устройство, а с другой — как клиент для звонков сервиса ВКС. Он имеет возможность переносить медиа из одного вида в другой, а также работать с сигналингом двух видов. Он может кподключаться как SIP, так и в ВКС к сервисам сигналинга и сервисам **Janus**.

Для преобразования сигналинга с использованием различных правил существует сервис **Telepath**. В этом сервисе работает процесс **Kamailio**. Преобразование сигналинга производится по правилам, описанным в конфигурационном файле **Kamailio**. Файл можно настраивать под конкретного клиента. Далее **Kamailio** приходит в **Sipgate**, и происходит соеднинение с SIP-инфраструктурой заказчика. К групповому звонку в сервисе ВКС **Sipgate** подключается как участник (это видно другим участникам). Соединение от **Sipgate** доходит до сервиса **Focus**, обслуживающего конкретный звонок. Также ведется подключение к медиасерверу **Janus**, который обрабатывает звонок.

Сервис **Vcl** используется для хранения PIN и пароля для подключения к конференции. Есть функциональность, реализующая ввод пользователями DTMF-кода (ввод с помощью тонального набора) для доступа к конференции. За этими кодами **Sipgate** и обращается к сервису **Vcl**.

Сервис **Numbers** занимается преобразованием номера телефона в идентификатор пользователя и обратно. Он получает информацию из Keycloak, а затем через сервис **Nomail** передает информацию о том, какой SIP-индентификатор соответствует данному номеру телефона, в сервис **Numbers**. Сервис **Numbers** хранит данные в своей БД Tarantool. Эта информация позволяет совершать исходящие звонки по SIP пользователям из Active Directory. Для осуществления вызова севис **Sipgate** обратится к **Numbers** и узнает, какой SIP URI соответствует данному пользователю, а затем отправит в SIP-инфраструктуру заказчика через **Kamailio**.

Когда поступает входящий звонок, сервис **Sipgate** обратится к **Numbers** для проверки идентификатора пользователя, а затем подключится к нужному инстансу **Focus**.

Таким образом, SIP-звонки обрабатываются по схеме групповых, а не по схеме P2P-звонков. Всегда есть комната (**Focus**), через которую производится подключение, даже если звонок один на один.

3 сентября 2026 г.