

Статистика звонков в Asterisk



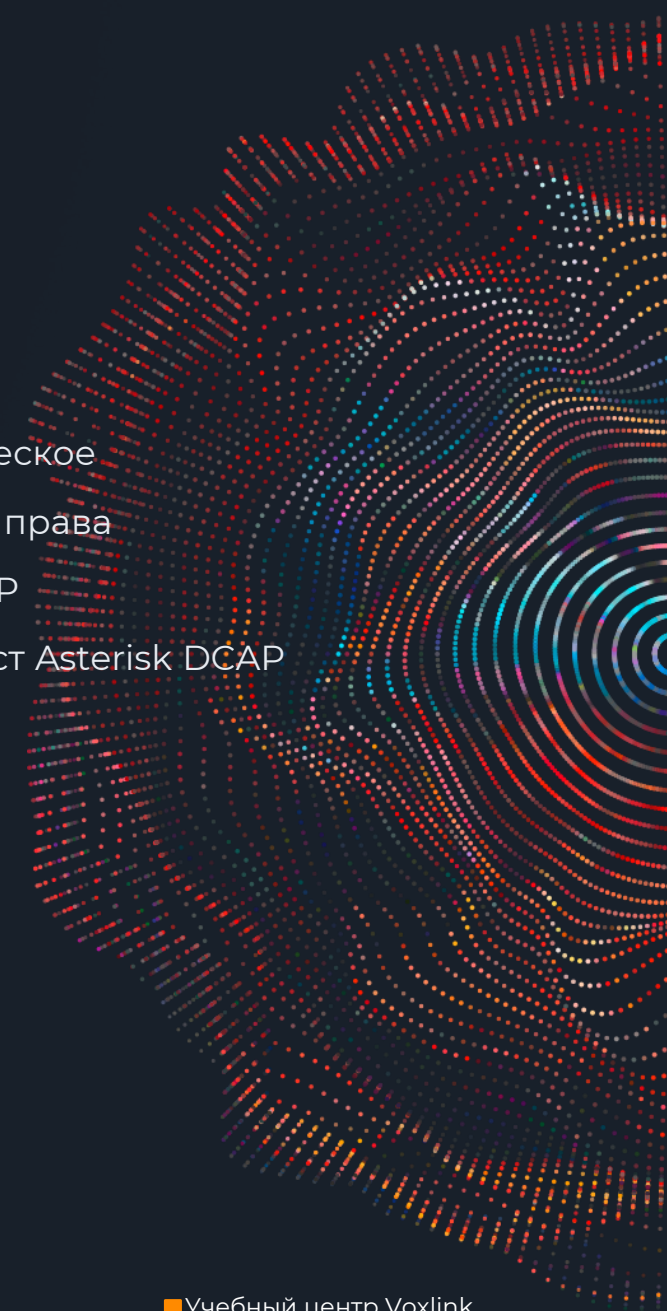
CDR · CEL · queue_log — три источника данных:
что пишут, чем отличаются, как связать между собой



**АЛЕКСАНДР
АБАБИЙ**



- Образование — высшее, техническое
- Информационных технологий и права
- Разработка и внедрение ПО VoIP
- Сертифицированный специалист Asterisk DCAP



План доклада



от формата записи — к готовой аналитике

1 Три источника

CDR, CEL, queue_log:
что пишет каждый и когда

2 Технические детали

Структура записей, поля,
бэкенды хранения

3 Настройка

cdr.conf, cel.conf,
queues.conf, realtime

4 Плюсы и минусы

Где каждый источник врёт
и почему

5 Связывание

linkedid, uniqueid —
сборка полного звонка

6 Экосистема

AMI/ARI, HEP, Prometheus,
QueueMetrics, Homer

Три источника, три уровня абстракции



один звонок — три разных описания

CDR

Call Detail Record

Итог разговора.
Одна строка на пару
«кто кому дозвонился».

Где нужен

Биллинг, отчёт
«кто кому и сколько».

CEL

Channel Event Logging

Хроника событий.
Десятки строк на каждый
канал звонка.

Где нужен

Расследование:
что именно произошло.

queue_log

Очереди Asterisk

События очереди.
Ожидание, соединение,
отказ, работа агентов.

Где нужен

Метрики call-центра:
SL, AWT, abandon.

Все три пишутся независимо и включаются отдельными модулями — общего «журнала звонков» в Asterisk нет.

01.



CDR — Call Detail Record

Что это

Итоговая запись о соединении между двумя каналами.

Когда пишется

При разрушении канала-источника, после hangup-логики.

Главный вопрос

«Кто кому звонил, дозвонился ли, сколько длилось?»

CDR: структура записи



стандартный набор полей, одинаковый для всех бэкендов

`calldate / start`

начало звонка

`answer / end`

ответ и завершение

`clid, src, dst`

CallerID, кто, кому

`dcontext`

контекст набора

`channel`

канал источника

`dstchannel`

канал назначения

`lastapp / lastdata`

последнее приложение

`duration`

всего секунд (start→end)

`billsec`

секунд разговора (answer→end)

`disposition`

ANSWERED / NO ANSWER / BUSY

`amaflags`

BILLING / DOCUMENTATION / OMIT

`accountcode`

код учёта, peeraccount — второй

`uniqueid / linkedid`

id канала и звонка

`userfield`

своё поле — 255 символов

`duration` ≠ `billsec`: разница — время дозвона. `disposition` берётся из состояния канала, а не из SIP-кода ответа.

CDR: настройка



cdr.conf + бэкенд хранения

```
; /etc/asterisk/cdr.conf
[general]
enable = yes
unanswered = yes      ; писать неотвеченные
congestion = yes
endbeforehexten = yes ; без h-extension
batch = yes           ; буферизация записи
size = 100
time = 300
safeshutdown = yes
```

```
; cdr_adaptive_odbc.conf — поля на лету
[first]
connection = asterisk
table = cdr
alias start => calldate
; колонка таблицы = имя CDR-переменной
; exten> Set(CDR(myfield)=value)
```

Бэкенды

cdr_adaptive_odbc

рекомендуемый: схема таблицы = схема CDR

cdr_odbc / cdr_pgsql

фиксированный набор колонок

cdr_custom

свой CSV-шаблон, любые переменные

cdr_manager

событие CDR в AMI — для интеграций

cdr_beanstalkd / cdr_syslog

очередь / централизованный лог

cdr_csv, cdr_radius, cdr_sqlite3

локально и legacy

CDR: плюсы и минусы



почему его используют все — и почему этого мало

Плюсы

- Компактно: 1–3 строки на звонок
- Понятен без обработки — можно читать глазами
- Готовые отчёты во FreePBX и любой биллинг
- Легко расширяется: CDR(userfield), adaptive ODBC
- Минимальная нагрузка, особенно в batch-режиме

Минусы

- Нет промежуточных состояний: IVR, удержание, переводы
- Перевод звонка ломает картину — биллинг «уезжает»
- Local-каналы и очереди дают несколько записей на звонок
- Нет надёжной связи записей между собой без linkedid
- disposition упрощает реальность: BUSY и FAILED сливаются

CDR: подводные камни



что чаще всего ломает отчёты на проде

! Переводы

При attended transfer CDR перепишет dst на конечного абонента. Исходное направление восстанавливается только по CEL.

! Local-каналы

Dial(Local/...) даёт лишнюю пару записей. Лечится /h и Set(CDR_PROP(disable)=1) на транзитном плече.

! ForkCDR

Порождает вторую запись — удобно для тарификации по этапам, но легко получить двойной биллинг.

! h-extension

endbeforehexten=no добавит время работы hangup-логики в billsec.

! Очереди

Каждая попытка дозвона до агента — отдельная запись, если не отключить unanswered.

! Batch-режим

kill -9 Asterisk = потерянные CDR из буфера. Для биллинга — batch=no или дубль в AMI.

02.

CEL — Channel Event Logging

Что это

Поток событий жизненного цикла каждого канала.

Когда пишется

Немедленно, в момент наступления события.

Главный вопрос

«Что именно произошло со звонком и в каком порядке?»

CEL: ТИПЫ СОБЫТИЙ



events = ALL или точечный список в cel.conf



Жизнь канала

CHAN_START
CHAN_END
ANSWER
HANGUP
LINKEDID_END



Мосты

BRIDGE_ENTER
BRIDGE_EXIT
PARK_START
PARK_END



Переводы

BLINDTRANSFER
ATTENDEDTRANSFER
PICKUP
FORWARD



Приложения

APP_START
APP_END
USER_DEFINED
LOCAL_OPTIMIZE

```
; /etc/asterisk/cel.conf  
[general]  
enable=yes | apps=dial,park,queue | events=ALL | dateformat=%F %T ; + cel_odbc.conf / cel_custom.conf
```

CEL: поля записи и хранение



ключевое отличие — `uniqueid` и `linkedid` в каждой строке

<code>eventtype, eventtime</code>	тип события и метка времени
<code>channame</code>	канал, породивший событие
<code>uniqueid</code>	идентификатор этого канала
<code>linkedid</code>	идентификатор всего звонка
<code>cid_name, cid_num</code>	CallerID, набранный номер
<code>exten, context</code>	точка в диалплане
<code>appname, appdata</code>	приложение и аргументы
<code>peer</code>	вторая сторона для BRIDGE
<code>extra</code>	JSON: детали перевода, cause

```
; cel_odbc.conf
[first]
connection=asterisk
table=cel
usegmtime=no

; cel_custom.conf
[mappings]
Master.csv => ${CSV_QUOTE(${eventtype})},
               ${CSV_QUOTE(${CHANNEL(linkedid)})}
```

Объём

Простой звонок 1⇌1: 10–14 строк.
Звонок через очередь с 5 агентами
и переводом: 40–80 строк.
Планируйте партиционирование
и ретеншн заранее.

CEL: как выглядит звонок



входящий → очередь → агент → перевод

10:00:01	 CHAN_START	PJSIP/trunk-0001	linkedid = 1712...01
10:00:02	 APP_START	PJSIP/trunk-0001	Queue(support)
10:00:05	 CHAN_START	PJSIP/101-0002	дозвон до агента
10:00:12	 ANSWER	PJSIP/101-0002	агент снял трубку
10:00:12	 BRIDGE_ENTER	оба канала	peer = вторая сторона
10:03:40	 ATTENDEDTRANSFER	PJSIP/101-0002	extra: JSON с деталями
10:05:02	 BRIDGE_EXIT / HANGUP	все каналы	cause в extra
10:05:02	 LINKEDID_END	—	звонок завершён целиком

Один linkedid связывает всё: и транк, и все попытки дозвона, и канал после перевода.

CEL: плюсы и минусы



полная правда о звонке — но её надо ещё собрать

Плюсы

- Полная хронология: видно каждое действие канала
- linkedid в каждой строке — звонок собирается однозначно
- Корректно отражает переводы, подхваты, парковку
- Позволяет считать время в IVR, в очереди, на удержании
- extra в JSON: причины разрыва, детали переводов

Минусы

- Нет готовых отчётов — всё пишется руками
- Объем в 10–40 раз больше CDR: нужен ретеншн и партиции
- Логика агрегации нетривиальна и зависит от версии
- Часть событий устарела: код на 11 не заведётся на 20
- Ошибка с Local-каналами резко раздувает объём

03.

queue_log — журнал очередей

Что это

Плоский лог событий очередей: ожидание, соединение, отказ.

Когда пишется

app_queue пишет сам, независимо от CDR и CEL.

Главный вопрос

«Как работает call-центр: SL, ожидание, потери?»

queue_log: анатомия строки



/var/log/asterisk/queue_log — разделитель «|»

```
1712048400|1712048395.17|support|SIP/101|CONNECT|12|1712048395.18|7
```

- | | | |
|---|---------------------------|--|
| 1 | epoch | unixtime события |
| 2 | callid | uniqueid канала клиента — стыковка с CDR и CEL |
| 3 | queuename | имя очереди или NONE для системных событий |
| 4 | agent | интерфейс или membername |
| 5 | event | тип события |
| 6 | arg1 / arg2 / arg3 | holdtime, calltime, position... |

Почему это важно

Формат не менялся со времён Asterisk 1.4 — поэтому все парсеры и отчётные системы совместимы между собой.

callid — это uniqueid канала клиента. Именно он стыкует queue_log с CDR и CEL.

queue_log: ключевые события



и что лежит в arg1–arg3

ENTERQUEUE	url callerid позиция	клиент встал в очередь
CONNECT	holdtime uniqueid агента ringtime	соединён с агентом
COMPLETECALLER	holdtime calltime позиция	разговор завершил клиент
COMPLETEAGENT	holdtime calltime позиция	разговор завершил агент
ABANDON	позиция исх. позиция ожидание	клиент бросил трубку
EXITWITHTIMEOUT	позиция исх. позиция ожидание	вышел по таймауту
EXITWITHKEY / EXITEMPTY	клавиша, позиция / время	вышел по DTMF или нет агентов
RINGNOANSWER	время звонка, мс	агент не ответил
ADDMEMBER / REMOVEMEMBER	—	агент вошёл / вышел
PAUSE / UNPAUSE / PAUSEALL	причина паузы	перерыв оператора
TRANSFER	цель контекст holdtime calltime	перевод из очереди

Полный список зависит от версии: смотрите `app_queue.c` и `doc/queuelog.txt` в исходниках вашей сборки.

queue_log: настройка и метрики



файл, realtime или AMI

```
; queues.conf
[general]
persistentmembers = yes
log_membername_as_agent = yes ; имя вместо SIP/101
monitor-type = MixMonitor
eventwhencalled = yes ; события в AMI
eventmemberstatus = yes
```

```
; extconfig.conf - queue_log в БД
[settings]
queue_log => odbc,asterisk,queue_log

; CLI
asterisk -rx 'queue show support'
asterisk -rx 'queue reset stats support'
```

Что считается из queue_log

Service Level

доля CONNECT с holdtime ≤ порога от ENTERQUEUE

AWT / ожидание

среднее arg1 (holdtime) по CONNECT

Abandon rate

$ABANDON \div ENTERQUEUE$

АНТ / разговор

среднее arg2 (calltime) по COMPLETE*

Загрузка агента

$calltime \div (\text{время в очереди} - PAUSE)$

Дисциплина

RINGNOANSWER, PAUSE по причинам

queue_log: плюсы и минусы



единственный штатный источник метрик call-центра

Плюсы

- Единственный источник данных по ожиданию и потерям
- Стабильный формат — совместим со всеми парсерами
- Есть готовые продукты: QueueMetrics, Asternic
- Легко парсится и на bash, и в realtime-БД
- Свои события через QueueLog() прямо из диалплана

Минусы

- Только очереди: IVR и прямые номера не видны
- Плоский текст, схемы нет — семантика arg1–arg3 плавает
- Смысл событий менялся между версиями Asterisk
- Нет длительности разговора после перевода из очереди
- Ротация файла легко ломает исторические отчёты

04.

Собираем полную картину

Сравнение

Какой источник на какой вопрос отвечает.

Хранилище

MySQL или PostgreSQL, схема и обслуживание.

Связывание

linkedid как единый идентификатор звонка.

Сравнение трёх источников



выбирайте по вопросу, на который отвечаете

Критерий	CDR	CEL	queue_log
Модель данных	агрегат	поток событий	доменный лог
Строк на звонок	1–3	10–80	3–8
Видит переводы	искажённо	да	частично
Время в IVR / очереди	нет	да	только очередь
Работа агентов	нет	косвенно	да
Готовые отчёты	да	нет	да
Нагрузка на БД	низкая	высокая	средняя
Пригоден для биллинга	да	да, сложнее	нет

Правильный ответ — использовать все три источника одновременно и связывать их по linkedid.

Хранилище: MySQL/MariaDB или PostgreSQL



оба поддерживаются штатно — разница вылезает на объёме

Критерий	MySQL / MariaDB	PostgreSQL
Модули Asterisk	cdr_odbc, cdr_adaptive_odbc, cel_odbc (cdr_mysql — addons, заброшен)	cdr_pgsql, cel_pgsql — в основном дереве, схему таблицы читают сами
Realtime	res_config_odbc	res_config_pgsql или тот же ODBC
Партиционирование	RANGE(TO_DAYS(...)), ключ обязан входить в PRIMARY KEY	Declarative (11+), pg_partman, честный partition pruning
JSON в CEL.extra	тип JSON, generated columns	jsonb + GIN — заметно удобнее
Аналитика	оконные функции с 8.0 / MariaDB 10.2	CTE, LATERAL, зрелый планировщик
Обслуживание	VACUUM не нужен, но InnoDB фрагментируется после чисток	autovacuum обязателен после массового удаления CEL
Время и таймзоны	DATETIME без зоны — грабли	timestamptz — корректно из коробки
Экосистема	FreePBX, QueueMetrics, Asternic — сразу	требует ручной адаптации отчётов

FreePBX и готовые продукты → MariaDB. Своя аналитика с большим CEL → PostgreSQL. Свыше ~10 млн CDR в месяц — витрины в ClickHouse.

Схема таблиц и обслуживание



подключение и партиции по месяцам

MySQL / MariaDB

```
; res_odbc.conf
[asterisk]
enabled => yes
dsn => asterisk-mysql
pre-connect => yes
sanitysql => select 1
forcecommit => yes
idlecheck => 3600

-- calldate обязан входить в PRIMARY KEY
ALTER TABLE cdr
  PARTITION BY RANGE (TO_DAYS(calldate)) (
    PARTITION p202607 VALUES LESS THAN
      (TO_DAYS('2026-08-01')),
    PARTITION pmax VALUES LESS THAN MAXVALUE);
```

PostgreSQL

```
; cdr_pgsql.conf
[global]
hostname=127.0.0.1
dbname=asterisk
table=cdr
user=asterisk
appname=asterisk-cdr
; модуль сам читает схему таблицы

-- декларативные партиции
CREATE TABLE cdr (...)
  PARTITION BY RANGE (calldate);
CREATE TABLE cdr_2026_07 PARTITION OF cdr
  FOR VALUES FROM ('2026-07-01')
  TO ('2026-08-01');
```

1 Пул соединений

CEL на нагрузке даёт лавину вставок.
Ограничьте пул в res_odbc или поставьте pgbouncer.

2 Чистка истории

Никогда DELETE по CEL. Только DROP PARTITION или TRUNCATE — иначе таблица встанет на часы.

3 Индексы

Обязательны linkedid, uniqueid, calldate.
Каждый лишний индекс на CEL — минус к вставке.

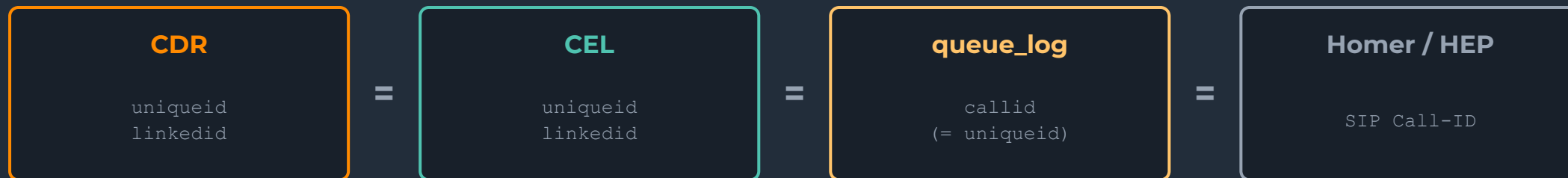
Как связать всё вместе



linkedid — идентификатор звонка, uniqueid — идентификатор канала

linkedid

присваивается первому каналу звонка и наследуется всеми последующими — транком, Local, каналом агента, каналом после перевода



1 Запишите linkedid явно

Set(CDR(linkedid)=\${CHANNEL(linkedid)}) — не во всех сборках он есть в таблице по умолчанию.

2 Проставьте SIP Call-ID

Set(CDR(userfield)=\${CHANNEL(pjsip,call-id)}) — мост к Homer и трассировке сигнализации.

3 Общий ключ для CRM

Единый идентификатор звонка = linkedid. Именно его отдавайте наружу в вебхуках.

Практика: собираем звонок целиком



один linkedid — три источника

```
-- 1. Путь звонка по CEL
SELECT eventtime, eventtype, channame, exten, peer
FROM   cel
WHERE  linkedid = '1712048395.17'
ORDER BY eventtime, id;

-- 2. Все CDR этого звонка
SELECT src, dst, disposition, duration, billsec
FROM   cdr
WHERE  linkedid = '1712048395.17';

-- 3. Что было в очереди
SELECT time, queue_name, agent, event,
       data1 AS holdtime, data2 AS calltime
FROM   queue_log
WHERE  callid = '1712048395.17';
```

```
-- Витрина: очередь + результат
SELECT q.callid,
       MIN(CASE WHEN q.event='ENTERQUEUE'
                THEN q.time END)          AS entered,
       MAX(CASE WHEN q.event='CONNECT'
                THEN q.data1 END)         AS holdtime,
       MAX(CASE WHEN q.event LIKE 'COMPLETE%'
                THEN q.data2 END)         AS talktime,
       MAX(q.agent)                       AS agent,
       MAX(c.disposition)                 AS cdr_result
FROM   queue_log q
LEFT JOIN cdr c ON c.linkedid = q.callid
WHERE  q.queue_name = 'support'
       AND q.time >= NOW() - INTERVAL 1 DAY
GROUP BY q.callid;
```

Сложные кейсы: кто что покажет



три сценария, в которых расходятся показания источников

Attended transfer

● CDR

dst подменён, исходное
направление потеряно

● CEL

ATTENDEDTRANSFER + extra с
деталью сторон

● queue_log

TRANSFER, дальше разговор не
виден

Перелив между очередями

● CDR

одна запись, обе очереди
неразличимы

● CEL

APP_START(Queue) дважды — видно
обе

● queue_log

два ENTERQUEUE,
EXITWITHTIMEOUT в первой

Конференция / парковка

● CDR

по записи на участника, связи нет

● CEL

BRIDGE_ENTER/EXIT,
PARK_START/END

● queue_log

не отражается вообще

05.

Экосистема и инструменты

Транспорт

AMI, ARI, AGI, HEP, очереди сообщений.

Метрики

res_prometheus, Grafana, ClickHouse.

Готовое

QueueMetrics, Asternic, Homer, FreePBX.

Что ещё есть для статистики



источники и транспорт помимо трёх основных

AMI

События Newchannel, Bridge, DialEnd, QueueCallerJoin в реальном времени. Основа онлайн-панелей и интеграций с CRM.

AGI / EAGI

Точечная запись событий из диалплана в свою БД: этапы IVR, результаты скриптов.

res_prometheus

Штатный экспортёр метрик Asterisk 17+: каналы, endpoint'ы, мосты → Prometheus + Grafana.

CDR(userfield) / USER_DEFINED

Свои маркеры прямо в штатных таблицах — самый дешёвый способ обогатить данные.

ARI / Stasis

Полный контроль звонка и события по WebSocket. Своя логика вместо диалплана — и своя, точная статистика.

HEP / Homer

res_hep, res_hep_pjsip, res_hep_rtcp — захват сигнализации. Отвечает, почему не соединилось.

Очереди сообщений

cdr_beanstalkd, cel_beanstalkd, свой мост AMI → RabbitMQ/Kafka для потоковой аналитики.

MixMonitor + RTCP

Записи разговоров и MOS/jitter как отдельное измерение качества.

Готовые решения



если не хочется писать отчёты руками

FreePBX CDR Reports

CDR

Базовый веб-отчёт и прослушивание записей из коробки.

Asternic Call Center Stats

queue_log

Отчёты по очередям, есть бесплатная версия и модуль FreePBX.

Grafana + ClickHouse

CDR + CEL

Свои витрины на больших объёмах: быстрые агрегаты.

QueueMetrics

queue_log

Отраслевой стандарт для call-центра: SL, agent page, wallboard.

Homer / HEPIC

HEP

Трассировка SIP, поиск причин неуспешных вызовов.

iSymphony / vqPlus

AMI

Онлайн-мониторинг и супервизия операторов.

Чек-лист для продакшена



с чего начать завтра



Включите все три

CDR для биллинга, CEL для разбора, queue_log для call-центра. Места это стоит немного.



Сохраняйте linkedid

Явно пишите его в CDR и отдавайте наружу как идентификатор звонка.



Планируйте объём

CEL: партии по дате, ретеншн 30–90 дней, архив в холодное хранилище.



Не считайте на лету

Стройте витрины по расписанию, а не тяжёлыми JOIN'ами в момент отчёта.



Проверьте на кейсах

Перевод, перелив между очередями, конференция — прогоните вручную и сверьте.



Свяжите с сигнализацией

SIP Call-ID в userfield + Homer: причины неуспеха видны сразу.

Спасибо за внимание!



CDR отвечает «сколько», CEL — «что произошло»,
queue_log — «как работает call-центр».
Вместе, через linkedid, они дают полную картину звонка.

Контакты

asterisker.ru info@asterisker.ru Telegram
@asterisker

Вопросы?