Централизованный сбор данных о работе серверов в БД ClickHouse

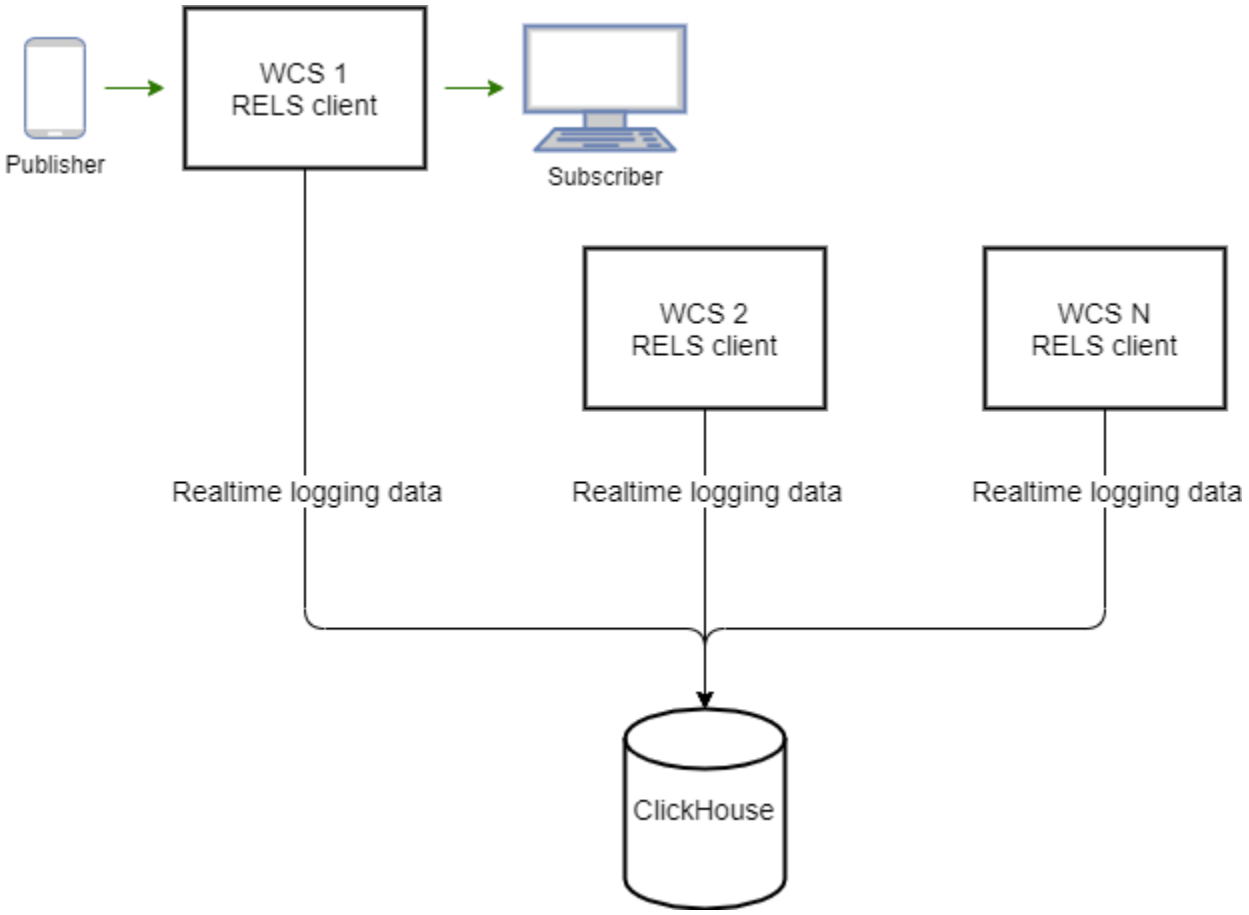
Описание

При управлении большим количеством WCS серверов, для отладки возможных проблем с вещанием потоков, возникает необходимость централизованного сбора данных о потоках, клиентских соединениях и событий CDN. Фактически, необходимо в одной точке собрать информацию, которая пишется в логи каждого сервера. При этом само по себе логирование в промышленной эксплуатации сведено к минимуму, чтобы не давать нагрузку на дисковую подсистему сервера.

Для сбора данных в реальном времени в больших объемах хорошо подходят базы данных временных рядов. На основе одной из таких БД с открытым исходным кодомClickHouse, в сборке5.2.774добавлена система удаленного сбора логов RELS (Remote Event Logging System).

Архитектура

Каждый WCS сервер отправляет данные в ClickHouse независимо, используя JDBC-драйвер и HTTP-соединение. Для оптимизации работы с БД ClickHouse, данные буферизуются и отправляются пачками по времени



Описание таблиц данных

Данные собираются в таблицы ClickHouse, перечисленные ниже. При этом, в таблицу записывается числовой идентификатор события. Для того, чтобы отобразить события в выборках в удобном для чтения виде, каждой таблице сопоставлен словарь текстовых строк, описывающих события.

Данные соединений (таблица ConnectionEvent)

Поле	Тип	Описание
timestamp	UInt64	Метка времени
ip	IPv4	Адрес сервера
sessionId	String	Идентификатор сессии
eventType	UInt64	Идентификатор типа события
eventPayload	String	Содержимое события

Данные о событиях потоков (таблица StreamEvent)

Поле	Тип	Описание
timestamp	UInt64	Метка времени
ip	IPv4	Адрес сервера
sessionId	String	Идентификатор сессии
mediaSessionId	String	Идентификатор медиасессий
streamName	String	Имя потока
eventType	UInt64	Идентификатор типа события

eventPayload	String	Содержимое события

Данные CDN(таблица CDNEvent)

Поле	Тип	Описание
timestamp	UInt64	Метка времени
ip	IPv4	Адрес сервера
nodeId	String	Идентификатор узла (строка IP адреса)
eventType	UInt64	Идентификатор типа события
eventPayload	String	Содержимое события

Данные о метриках потоков (таблица MediaSessionEvent)

В сборке 5.2.1896 добавлен сбор данных о метриках определенных потоков

Поле	Тип	Описание
timestamp	UInt64	Метка времени
ip	IPv4	Адрес сервера
mediaSessionId	String	Идентификатор медиасессий
streamName	String	Имя потока
videoProfileId	UInt32	Идентификатор профиля видео
videoWidth	UInt32	Высота картинки
videoHeight	UInt32	Ширина картинки
videoFrameRate	UInt32	Частота кадров в секунду
videoKframes	UInt64	Количество ключевых кадров (I-frame)
videoPframes	UInt64	Количество промежуточных кадров (P-frame)
videoBframes	UInt64	Количество двунаправленных кадров (B-frames)
videoRate	UInt64	Битрейт видео, бит/с
audioRate	UInt64	Битрейт аудио, бит/с
videoSyncTime	UInt64	Значение синхронизации видео
audioSyncTime	UInt64	Значение синхронизации аудио
videoTimestamp	UInt64	Метка времени из видеопакета
audioTimestamp	UInt64	Метка времени из аудиопакета
lastKeyFrameSyncTime	UInt64	Значение синхронизации видео из последнего ключевого кадра
sendNACK	UInt64	Количество отправленных NACK
recvNACK	UInt64	Количество полученных NACK
videoFramesLost	UInt64	Количество потерянных кадров видео
audioPacketsLost	UInt64	Количество потерянных пакетов аудио
audioPlaybackSpeed	Float32	Скорость публикации/проигрывания аудио
videoPlaybackSpeed	Float32	Скорость публикации/проигрывания видео

Данные о нарезке HLS потока (таблица HlsStreamEvent)

В сборке 5.2.1917 добавлен сбор данных о событиях определенных HLS потоков

Поле	Тип	Описание

timestamp	UInt64	Метка времени
ip	IPv4	Адрес сервера
severity	UInt8	Уровень события: INFO, WARNING, ERROR
messageType	UInt16	Тип события
streamId	String	Идентификатор HLS потока (имя)
variantName	String	Имя варианта качества
segmentId	String	Идентификатор сегмента
message	String	Описание события в том виде, как залогировано

Данные о сегментах HLS потока (таблица HlsSegmenterEvent)

В сборке 5.2.1917 добавлен сбор данных о сегментах определенных HLS потоков

Поле	Тип	Описание
timestamp	UInt64	Метка времени
ip	IPv4	Адрес сервера
streamId	String	Идентификатор HLS потока (имя)
variantName	String	Имя варианта качества
segmentId	String	Идентификатор сегмента
segmentStartPts	UInt64	Начальный PTS сегмента
videoStartPts	UInt64	Начальный PTS видео дорожки
audioStartPts	UInt64	Начальный PTS аудио дорожки
videoWidth	UInt32	Ширина картинки
videoHeight	UInt32	Высота картинки
videoFrameCount	UInt32	Количество кадров видео
audioPacketCount	UInt32	Количество пакетов аудио
segmentDuration	UInt64	Длительность сегмента, мс
independent	Bool	Независимый сегмент
gap	Bool	GAP сегмент
discontinuity	Bool	DISCONTINUTY сегмент
segmentInterval	UInt64	Интервал между сегментами, мс
partial	Bool	Частичный сегмент
playbackSpeed	Float32	Скорость проигрывания

Данные о подписчиках HLS потока (таблица HlsClientEvent)

В сборке 5.2.1929добавлен сбор данных о подписчиках определенных HLS потоков

Поле	Тип	Описание
creationTime	UInt64	Время подключения подписчика
responseTime	UInt64	Время ответа на запрос подписчика
streamId	String	Идентификатор HLS потока (имя)
variantName	String	Имя варианта качества
uri	String	URI плейлиста
localIp	IPv4	Адрес сервера

remoteIp	IPv4	Адрес клиента
remotePort	UInt32	TCP порт, с которого подключился клиента
userAgent	String	Значение заголовка User-Agent, полученное от клиента
httpStatus	UInt32	Статус ответа на запрос клиента
clientId	UInt64	Идентификатор клиентской сессии

Данные о метриках микшера (таблица MixerEvent)

В сборке 5.2.1923 добавлен сбор данных о метриках определенных микшеров

Поле	Тип	Описание
timestamp	UInt64	Метка времени
mixerMediaSessionId	String	Идентификатор медиасессии микшера
mixerStreamName	String	Имя выходного потока микшера
mediaSessionId	String	Идентификатор входящего потока микшера
streamName	String	Имя входящего потока
mixerAverageTickTimeInMs	Int64	Среднее время одного такта микширования, мс
audioMixerSync	Int64	Значение синхронизации аудио в выходном потоке микшера
videoMixerSync	Int64	Значение синхронизации видео в выходном потоке микшера
nextAudioDataTime	Int64	Время следующего аудио пакета
nextVideoDataTime	Int64	Время следующего видео пакета
audioBuffered	Int64	Количество аудио пакетов в буфере входящего потока
videoBuffered	Int64	Количество видео пакетов в буфере входящего потока
audioDropsCounter	Int64	Количество отброшенных аудио пакетов входящего потока
audioDropsSizeInBytes	Int64	Объем отброшенных данных аудио в байтах
videoDropsCounter	Int64	Количество отброшенных видео пакетов входящего потока
videoDropsSizeInBytes	Int64	Объем отброшенных данных видео в байтах
videoFps	Int64	Частота кадров выходного потока микшера в секунду
audioRate	DOUBLE	Битрейт аудио выходного потока микшера, бит/с
videoRate	DOUBLE	Битрейт видео выходного потока микшера, бит/с
eventType	UInt32	Тип события микшера
eventPayload	String	Описание события микшера

Данные о восстановлении потерянных аудио пакетов (таблица AudioRecoveryEvent)

В сборке 5.2.1969 добавлен сбор данных о восстановлении потерянных аудио пакетов

Поле	Тип	Описание
timestamp	UInt64	Метка времени
mediaSessionId	String	Идентификатор медиасессии
type	UInt32	Тип события
rtpTimestamp	UInt64	Метка времени из RTP потока

Данные о метриках буфера для входящих RTMP потоков (таблица RtmpInBufferEvent)

В сборке 5.2.1978 добавлен сбор данных о метриках буфера входящих RTMP потоков

Поле	Тип	Описание
timestamp	UInt64	Метка времени
streamClockTime	UInt64	Метка времени по часам потока
mediaSessionId	String	Идентификатор медиасессии
streamName	String	Имя потока
nextAudioDataTime	Int64	Метка времени следующего пакета аудио данных
nextVideoDataTime	Int64	Метка времени следующего пакета видео данных
audioBuffered	Int64	Количество данных аудио в буфере
videoBuffered	Int64	Количество данных видео в буфере
maximumAllowedBuffer	Int64	Максимальная емкость буфера
bufferingCounter	Int64	Счетчик буферизаций
lastAudioDataTime	Int64	Время последнего пакета аудио данных
lastVideoDataTime	Int64	Время последнего пакета видео данных
bufferState	UInt32	Состояние буфера

Данные об отправленных REST хуках (таблица RestHooksEvent)

В сборке 5.2.2005 добавлен сбор данных об отправленных REST хуках

Поле	Тип	Описание
sendTime	UInt64	Время отправки REST хука
responseTime	UInt64	Время ответа на REST хук
appKey	String	Идентификатор приложения, отправляющего REST хук
method	String	Имя отправляемого хука
sessionId	String	Идентификатор клиентской сессии
requestUrl	String	URL REST хука
requestBody	String	Тело запроса (по умолчанию не логируется)
requestBodyMD5	String	MD5 хэш тела запроса
requestContentLength	UInt64	Длина тела запроса
responseBody	String	Тело ответа (по умолчанию не логируется)
responseBodyMD5	String	MD5 хэш тела ответа
responseContentLength	UInt64	Длина тела ответа
responseStatus	UInt32	Статус ответа
error	String	Сообщение об ошибке

Настройка

Установка и настройка ClickHouse

Требования к серверу

- CPU не менее 4 физических ядер, частотой не менее 3 ГГц, напримерIntel(R) Xeon(R) CPU E3-1246 v3 @ 3.50GHz
- RAM не менее 32 Гб
- HDD не менее 2 Тб

Установка ClickHouse из rpm пакетов (CentOS, Red Hat etc)

1. Подключите официальный репозиторий

```
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo https://packages.clickhouse.com/rpm/clickhouse.repo
```

2. Установите ClickHouse

```
sudo yum install -y clickhouse-server clickhouse-client
```

3. Запустите ClickHouse

```
sudo systemctl enable clickhouse-server
sudo systemctl start clickhouse-server
```

Установка ClickHouse из deb пакетов (Debian, Ubuntu etc)

1. Подключите официальный репозиторий

```
sudo apt-get install -y apt-transport-https ca-certificates dirmngr
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 8919F6BD2B48D754
echo "deb https://packages.clickhouse.com/deb stable main" | sudo tee /etc/apt/sources.list.d/clickhouse.list
sudo apt-get update
```

2. Установите ClickHouse

```
sudo apt-get install -y clickhouse-server clickhouse-client
```

3. Запустите ClickHouse

```
sudo systemctl enable clickhouse-server
sudo systemctl start clickhouse-server
```

Настройка ClickHouse для сборок WCS до 5.2.1999

1. Для того, чтобы прослушивать входящие запросы на всех интерфесах сервера, раскомментируйте строку в файле/etc/clickhouse-server /config.xml

```
    <listen_host>::</listen_host>
```

2. Для того, чтобы создать пользователя, укажите для пользователя default в файле/etc/clickhouse-server/users.xml параметр

```
            <access_management>1</access_management>
```

3. Перезапустите ClickHouse

```
systemctl restart clickhouse-server
```

4. Создайте базу данных wcs и таблицы в ней

```
cat wcs_clickhouse.sql | clickhouse-client -mn
```

wcs_clickhouse.sql

```
CREATE DATABASE IF NOT EXISTS wcs;

DROP DICTIONARY IF EXISTS wcs.DictionaryStreamEvents;

DROP DICTIONARY IF EXISTS wcs.DictionaryConnectionEvents;

DROP DICTIONARY IF EXISTS wcs.DictionaryCDNEvents;

DROP DICTIONARY IF EXISTS wcs.DictionaryHlsStreamEventType;

DROP DICTIONARY IF EXISTS wcs.DictionaryHlsStreamEventSeverity;

DROP DICTIONARY IF EXISTS wcs.DictionaryMixerEvents;

DROP DICTIONARY IF EXISTS wcs.DictionaryBufferStateTypes;

DROP TABLE IF EXISTS wcs.StreamEvent;

DROP TABLE IF EXISTS wcs.ConnectionEvent;

DROP TABLE IF EXISTS wcs.CDNEvent;

DROP TABLE IF EXISTS wcs.StreamEventTypes;

DROP TABLE IF EXISTS wcs.ConnectionEventTypes;

DROP TABLE IF EXISTS wcs.CDNEventTypes;

DROP TABLE IF EXISTS wcs.MediaSessionEvents;

DROP TABLE IF EXISTS wcs.HlsStreamEvents;

DROP TABLE IF EXISTS wcs.HlsSegmenterEvents;

DROP TABLE IF EXISTS wcs.HlsStreamEventSeverity;

DROP TABLE IF EXISTS wcs.HlsStreamEventType;

DROP TABLE IF EXISTS wcs.HlsClientEvents;

DROP TABLE IF EXISTS wcs.MixerEvent;

DROP TABLE IF EXISTS wcs.MixerEventTypes;

DROP TABLE IF EXISTS wcs.RtmpInBufferEvent;

DROP TABLE IF EXISTS wcs.AudioRecoveryEvent;

DROP TABLE IF EXISTS wcs.BufferStateTypes;

CREATE TABLE wcs.ConnectionEventTypes
(
    `id` UInt32,
    `type` String
)
ENGINE = MergeTree()
ORDER BY id
SETTINGS index_granularity = 8192;

INSERT INTO wcs.ConnectionEventTypes VALUES (0, 'CONNECTED'), (1, 'DISCONNECTED');

CREATE TABLE wcs.StreamEventTypes
(
    `id` UInt32,
    `type` String
)
ENGINE = MergeTree()
ORDER BY id
SETTINGS index_granularity = 8192;

INSERT INTO wcs.StreamEventTypes VALUES (0,'CREATED'),(1,'LOCAL_SDP_CREATED'),(2,'REMOTE_SDP_RECEIVED'),
```

```
(3,'ICE_STARTED'),(4,'ICE_COMPLETE'),(5,'DTLS_STARTED'),(6,'DTLS_COMPLETE'),(7,'INITIALIZED'),(8,'DISPOSING'),
(9,'DISPOSED'),(10,'AUDIO_RECEIVED'),(11,'VIDEO_RECEIVED'),(12,'VIDEO_KFRAME_RECEIVED'),
(13,'AUDIO_RTCP_RECEIVED'),(14,'VIDEO_RTCP_RECEIVED'),(15,'RESOLUTION_RECEIVED'),(16,'VIDEO_ENCODER_CREATED'),
(17,'AUDIO_ENCODER_CREATED'),(18,'VIDEO_ENCODER_DISPOSED'),(19,'AUDIO_ENCODER_DISPOSED'),(20,'TERMINATED'),
(21,'AUDIO_SENT'),(22,'VIDEO_SENT'),(23,'VIDEO_JITTER_BUFFER_STALL'),(24,'SENT_PLI'),(25,'RECEIVED_PLI'),
(26,'SYNC_BUFFER_FULL'),(27,'SYNC_FORCE_FAILED'),(28,'SYNC_SHIFT'),(29,'SYNC_DEVIATION'),(30,'VIDEO_STATS'),
(31,'RECORD');

CREATE TABLE wcs.CDNEventTypes
(
    `id` UInt32,
    `type` String
)
ENGINE = MergeTree()
ORDER BY id
SETTINGS index_granularity = 8192;

INSERT INTO wcs.CDNEventTypes VALUES (0, 'STATE'), (1, 'CDN_STATE'), (2, 'VERSION'), (3, 'ACL_REFRESH'), (4,
'ACL_UPDATE');

CREATE DICTIONARY wcs.DictionaryStreamEvents (
    `id` UInt16,
    `type` String DEFAULT ''
)
PRIMARY KEY id
SOURCE(CLICKHOUSE(
 host 'localhost'
 port 9000
 user 'default'
 password ''
 db 'wcs'
 table 'StreamEventTypes'
))
LAYOUT(FLAT())
LIFETIME(300);

CREATE DICTIONARY wcs.DictionaryConnectionEvents (
    `id` UInt16,
    `type` String DEFAULT ''
)
PRIMARY KEY id
SOURCE(CLICKHOUSE(
 host 'localhost'
 port 9000
 user 'default'
 password ''
 db 'wcs'
 table 'ConnectionEventTypes'
))
LAYOUT(FLAT())
LIFETIME(300);

CREATE DICTIONARY wcs.DictionaryCDNEvents (
    `id` UInt16,
    `type` String DEFAULT ''
)
PRIMARY KEY id
SOURCE(CLICKHOUSE(
 host 'localhost'
 port 9000
 user 'default'
 password ''
 db 'wcs'
 table 'CDNEventTypes'
))
LAYOUT(FLAT())
LIFETIME(300);

CREATE TABLE wcs.StreamEvent
(
    `timestamp` UInt64,
```

```
    `ip` IPv4,
    `sessionId` String,
    `mediaSessionId` String,
    `streamName` String,
    `eventType` UInt64,
    `eventPayload` String
)
ENGINE = MergeTree()
ORDER BY (sessionId, mediaSessionId, streamName)
SETTINGS index_granularity = 8192;

CREATE TABLE wcs.ConnectionEvent
(
    `timestamp` UInt64,
    `ip` IPv4,
    `sessionId` String,
    `eventType` UInt64,
    `eventPayload` String
)
ENGINE = MergeTree()
ORDER BY (timestamp, sessionId)
SETTINGS index_granularity = 8192;

CREATE TABLE wcs.CDNEvent
(
    `timestamp` UInt64,
    `ip` IPv4,
    `nodeId` String,
    `eventType` UInt64,
    `eventPayload` String
)
ENGINE = MergeTree()
ORDER BY (nodeId, eventType)
SETTINGS index_granularity = 8192;

CREATE TABLE wcs.MediaSessionEvents
(
    `timestamp` UInt64,
    `ip` IPv4,
    `mediaSessionId` String,
    `streamName` String,
    `videoProfileId` UInt32,
    `videoWidth` UInt32,
    `videoHeight` UInt32,
    `videoFrameRate` UInt32,
    `videoKframes` UInt64,
    `videoPframes` UInt64,
    `videoBframes` UInt64,
    `videoRate` UInt64,
    `audioRate` UInt64,
    `videoSyncTime` UInt64,
    `audioSyncTime` UInt64,
    `videoTimestamp` UInt64,
    `audioTimestamp` UInt64,
    `lastKeyFrameSyncTime` UInt64,
    `sendNACK` UInt64,
    `recvNACK` UInt64,
    `videoFramesLost` UInt64,
    `audioPacketsLost` UInt64,
    `audioPlaybackSpeed` Float32,
    `videoPlaybackSpeed` Float32
)
ENGINE = MergeTree()
ORDER BY (mediaSessionId, streamName)
SETTINGS index_granularity = 8192;

CREATE TABLE wcs.HlsSegmenterEvents
(
    `timestamp` UInt64,
    `ip` IPv4,
    `streamId` String,
```

```
    `variantName` String,
    `segmentId` String,
    `segmentStartPts` UInt64,
    `videoStartPts` UInt64,
    `audioStartPts` UInt64,
    `videoWidth` UInt32,
    `videoHeight` UInt32,
    `videoFrameCount` UInt32,
    `audioPacketCount` UInt32,
    `segmentDuration` UInt64,
    `independent` Bool,
    `gap` Bool,
    `discontinuity` Bool,
    `segmentInterval` UInt64,
    `partial` Bool,
    `playbackSpeed` Float32
)
ENGINE = MergeTree()
ORDER BY (streamId)
SETTINGS index_granularity = 8192;

CREATE TABLE wcs.HlsStreamEventSeverity
(
    `id` UInt8,
    `type` String
)
ENGINE = MergeTree()
ORDER BY id
SETTINGS index_granularity = 8192;

INSERT INTO wcs.HlsStreamEventSeverity VALUES (0, 'INFO'), (1, 'WARNING'), (2, 'ERROR');

CREATE TABLE wcs.HlsStreamEventType
(
    `id` UInt16,
    `type` String
)
ENGINE = MergeTree()
ORDER BY id
SETTINGS index_granularity = 8192;

INSERT INTO wcs.HlsStreamEventType
VALUES (0, 'PLAYBACK_SPEED'), (1, 'FPS_CHANGED'), (2, 'GAP'), (3, 'RESOLUTION_CHANGED'), (4, 'DISCONTINUITY'),
(5, 'TASK_SKIPPED'), (6, 'NO_KYE_FRAME'), (7, 'NO_VIDEO'), (8, 'NO_AUDIO'), (9, 'SEGMENT_INTERVAL'), (10,
'OTHER');

CREATE DICTIONARY wcs.DictionaryHlsStreamEventSeverity
(
   `id` UInt8,
   `type` String DEFAULT ''
)
PRIMARY KEY id
SOURCE(CLICKHOUSE(
 host 'localhost'
 port 9000
 user 'default'
 password ''
 db 'wcs'
 table 'HlsStreamEventSeverity'
))
LAYOUT(FLAT())
LIFETIME(300);

CREATE DICTIONARY wcs.DictionaryHlsStreamEventType
(
   `id` UInt16,
   `type` String DEFAULT ''
)
PRIMARY KEY id
SOURCE(CLICKHOUSE(
 host 'localhost'
```

```
 port 9000
 user 'default'
 password ''
 db 'wcs'
 table 'HlsStreamEventType'
))
LAYOUT(FLAT())
LIFETIME(300);

CREATE TABLE wcs.HlsStreamEvents
(
    `timestamp` UInt64,
    `ip` IPv4,
    `severity` UInt8,
    `messageType` UInt16,
    `streamId` String,
    `variantName` String,
    `segmentId` String,
    `message` String
)
ENGINE = MergeTree()
ORDER BY (streamId)
SETTINGS index_granularity = 8192;

CREATE TABLE wcs.HlsClientEvents
(
    `creationTime` UInt64,
    `responseTime` UInt64,
    `streamId` String,
    `variantName` String,
    `uri` String,
    `localIp` IPv4,
    `remoteIp` IPv4,
    `remotePort` UInt32,
    `userAgent` String,
    `httpStatus` UInt32,
    `clientId` UInt64
)
ENGINE = MergeTree()
ORDER BY (creationTime)
SETTINGS index_granularity = 8192;

CREATE TABLE wcs.MixerEvent
(
    `timestamp` UInt64,
    `mixerMediaSessionId` String,
    `mixerStreamName` String,
    `mediaSessionId` String,
    `streamName` String,
    `mixerAverageTickTimeInMs` Int64,
    `audioMixerSync` Int64,
    `videoMixerSync` Int64,
    `nextAudioDataTime` Int64,
    `nextVideoDataTime` Int64,
    `audioBuffered` Int64,
    `videoBuffered` Int64,
    `audioDropsCounter` Int64,
    `audioDropsSizeInBytes` Int64,
    `videoDropsCounter` Int64,
    `videoDropsSizeInBytes` Int64,
    `videoFps` Int64,
    `audioRate` DOUBLE,
    `videoRate` DOUBLE,
    `eventType` UInt32,
    `eventPayload` String
)
ENGINE = MergeTree()
ORDER BY timestamp
SETTINGS index_granularity = 8192;

CREATE TABLE wcs.MixerEventTypes
```

```
(
    `id` UInt32,
    `type` String
)
ENGINE = MergeTree()
ORDER BY id
SETTINGS index_granularity = 8192;

INSERT INTO wcs.MixerEventTypes VALUES (0, 'nullEvent'), (1, 'dropBallastAudio'), (2, 'dropBallastVideo'), (3,
'audioNotBuffered'), (4, 'videoNotBuffered'), (5, 'audioBufferExhausted'), (6, 'videoBufferExhausted'), (7,
'alignStreamFailed'), (8, 'alignStreamDropAudio'), (9, 'alignStreamDropVideo'), (10, 'rateOutOfBoundsAudio'),
(11, 'rateOutOfBoundsVideo');

CREATE DICTIONARY wcs.DictionaryMixerEvents (
    `id` UInt16,
    `type` String DEFAULT ''
)
PRIMARY KEY id
SOURCE(CLICKHOUSE(
 host 'localhost'
 port 9000
 user 'default'
 password ''
 db 'wcs'
 table 'MixerEventTypes'
))
LAYOUT(FLAT())
LIFETIME(300);

CREATE TABLE wcs.RtmpInBufferEvent
(
    `timestamp` UInt64,
    `streamClockTime` UInt64,
    `mediaSessionId` String,
    `streamName` String,
    `nextAudioDataTime` Int64,
    `nextVideoDataTime` Int64,
    `audioBuffered` Int64,
    `videoBuffered` Int64,
    `maximumAllowedBuffer` Int64,
    `bufferingCounter` Int64,
    `lastAudioDataTime` Int64,
    `lastVideoDataTime` Int64,
    `bufferState` UInt32
)
ENGINE = MergeTree()
ORDER BY timestamp
SETTINGS index_granularity = 8192;

CREATE TABLE wcs.BufferStateTypes
(
    `id` UInt32,
    `type` String
)
ENGINE = MergeTree()
ORDER BY id
SETTINGS index_granularity = 8192;

INSERT INTO wcs.BufferStateTypes VALUES (0, 'BUFFERING'), (1, 'HOLD'), (2, 'TERMINATED'), (3, 'OVERFLOW'), (4,
'PASSTHROUGH');

CREATE DICTIONARY wcs.DictionaryBufferStateTypes (
    `id` UInt16,
    `type` String DEFAULT ''
)
PRIMARY KEY id
SOURCE(CLICKHOUSE(
 host 'localhost'
 port 9000
 user 'default'
 password ''
```

```
 db 'wcs'
 table 'BufferStateTypes'
))
LAYOUT(FLAT())
LIFETIME(300);

CREATE TABLE wcs.AudioRecoveryEvent
(
    `timestamp` UInt64,
    `mediaSessionId` String,
    `type` UInt32,
    `rtpTimestamp` UInt64
)
ENGINE = MergeTree()
ORDER BY rtpTimestamp
SETTINGS index_granularity = 8192;
```

5.Создайте пользователя wcs и дайте ему права на таблицы в базе данных wcs

```
cat wcs_clickhouse_users.sql | clickhouse-client -mn
```

wcs_clickhouse_users.sql

```
CREATE USER IF NOT EXISTS wcs IDENTIFIED BY 'wcs';
GRANT ALL ON wcs.* TO wcs WITH GRANT OPTION;
```

6.Отключите управление пользователями для пользователя default, указав в файле/etc/clickhouse-server/users.xml параметр

```
            <access_management>0</access_management>
```

7. ПерезапуститеClickHouse

```
systemctl restart clickhouse-server
```

Настройка ClickHouse для сборок WCS 5.2.1999 и новее

В сборке 5.2.1999 добавлено автоматическое создание БД при подключении определенной версии WCS с определенного IP адреса. Данные о всех БД хранятся в таблицеWcsMetadata.metadata

```
SELECT *
FROM WcsMetadata.metadata

dbNameiphostnameversionNumberversionHash
 test1flashphonercom_192168065_521999  192.168.0.65   test1.flashphoner.com  5.2.1999
93e39647113e0121dabc4283ef700814c355568f
 test2flashphonercom_192168039_521999  192.168.0.39   test2.flashphoner.com  5.2.1999
93e39647113e0121dabc4283ef700814c355568f
```

Поэтому для настройки ClickHouse необходимо только создать пользователя и дать ему права на создание Баз данных, таблиц и вставку данных в них

1. Для того, чтобы прослушивать входящие запросы на всех интерфесах сервера, раскомментируйте строку в файле/etc/clickhouse-server /config.xml

```
    <listen_host>::</listen_host>
```

2. Для того, чтобы создать пользователя, укажите для пользователя default в файле /etc/clickhouse-server/users.xml параметры

```
            <access_management>1</access_management>
            <named_collection_control>1</named_collection_control>
            <show_named_collections>1</show_named_collections>
            <show_named_collections_secrets>1</show_named_collections_secrets>
```

3. Перезапустите ClickHouse

```
systemctl restart clickhouse-server
```

4. Создайте пользователя wcs и дайте ему необходимые права

```
cat wcs_clickhouse_users.sql | clickhouse-client -mn
```

wcs_clickhouse_users.sql

```
CREATE USER IF NOT EXISTS wcs IDENTIFIED BY 'wcs';
SET allow_introspection_functions = 1;
GRANT ALL ON *.* TO wcs WITH GRANT OPTION;
```

5. Отключите управление пользователями для пользователя default, указав в файле /etc/clickhouse-server/users.xml параметр

```
            <access_management>0</access_management>
```

6. Перезапустите ClickHouse

```
systemctl restart clickhouse-server
```

Настройка WCS

Сбор данных в БД ClickHouse включается настройкой, в которой перечисляются типы отправляемых данных

```
rels_enabled=CONNECTION,STREAM,CDN,MEDIA_SESSION
```

Доступны следующие типы данных:

Тип	Описание
CONNECTION	События клиенсткой сессии
STREAM	События потока
CDN	События CDN
MEDIA_SESSION	События медиа сессии
HLS_SEGMENTER	Данные о нарезке HLS потоков
HLS_STREAM	События HLS потоков
HLS_CLIENT	Статистика HLS клиентов
MIXER	События микшера
AUDIO_RECOVERY	Статистика потерь и восстановления аудиопакетов
RTMP_IN_BUFFER	Статистика буфера входящих RTMP потоков
REST_HOOKS	Статистика отправленных REST хуков
REST_HOOKS_BODY	Статистика отправленных REST хуков, включая тело запроса и тело ответа

Настройка подключения к ClickHouse до сборки WCS 5.2.1999

Адрес сервера ClickHouse, база данных и протокол задаются настройками

```
rels_client_type=HTTP
rels_database_address=http://clickhouseserver:8123/wcs?user=wcs&password=wcs
```

По умолчанию используется рекомендованный HTTP протокол. Однако, при необходимости можно переключиться на использование JDBC драйвера

```
rels_client_type=JDBC
rels_database_address=jdbc:clickhouse://clickhouseserver:8123/wcs?user=wcs&password=wcs
```

Настройка подключения к ClickHouse в сборках WCS 5.2.1999 и новее

Адрес сервера ClickHouse, пользователь и протокол задаются настройками

```
rels_client_type=HTTP
rels_database_address=clickhouseserver:8123
rels_database_properties=user=wcs&password=wcs
```

По умолчанию используется рекомендованный HTTP протокол. Однако, при необходимости можно переключиться на использование JDBC драйвера

```
rels_client_type=JDBC
```

Остановка сбора данных без перезапуска WCS

При необходимости, передача данных с конкретного WCS сервера в ClickHouse может быть остановлена без перезапуска WCS. Для этого:

1. Отключите сбор данных в настройках сервера

```
rels_enabled=
```

2. Перезагрузите настройки сервера из интерфейса командной строки

```
reload node-settings
```

Изменение адреса сервера ClickHousebез перезапуска WCS

Адрес сервера ClickHouse может быть изменен без перезапуска WCS. Для этого:

1. Измените адрес в настройках сервера

```
rels_database_address=jdbc:clickhouse://newclickhouseserver:8123/wcs?user=wcs&password=wcs
```

2. Отключите сбор данных в настройках сервера

```
#rels_enabled=CONNECTION,STREAM,CDN,MEDIA_SESSION
rels_enabled=
```

3. Перезагрузите настройки сервера из интерфейса командной строки

```
reload node-settings
```

4. Включите сбор данных в настройках сервера

```
rels_enabled=CONNECTION,STREAM,CDN,MEDIA_SESSION
```

5. Перезагрузите настройки сервера из интерфейса командной строки

```
reload node-settings
```

Количество процессорных потоков, используемых клиентом ClickHouse

В сборке 5.2.2005 добавлена настройка количества процессорных потоков, используемых клиентом ClickHouse. По умолчанию, клиент будет использовать 2 потока, которые будет запускать для отправки данных и останавливать после отправки

```
rels_database_thread_pool_size=2
```

Если установить значение этого параметра в 0

```
rels_database_thread_pool_size=0
```

клиент ClickHouse создаст постоянный пул процессорных потоков размером

```
(CPU count * 2) + 1
```

Сжатие данных при отправке

В сборке 5.2.2005 добавлена настройка, позволяющая включить сжатие данных при отправке. По умолчанию, сжатие данных отключено

```
rels_enable_compression=false
```

При отправке большого объема данных можно включить сжатие для снижения трафика между WCS и ClickHouse, но в этом случае возможно проявление проблемы клиента ClickHouse, и часть строковых данных может не быть записана.

Управление сбором данных по REST API

Типы данных CONNECTION, STREAM, CDN, HLS_STREAM, REST_HOOKS собираются безусловно, для всех клиентских сессий и всех опубликованных потоков. Все остальные типы данных собираются только по запросу, поскольку объем отсылаемых данных может быть очень велик.

Сбор данных для определенного потока включается по REST API.

REST-запрос должен быть HTTP/HTTPS POST запросом в таком виде:

- HTTP:http://streaming.flashphoner.com:8081/rest-api/rels/startup
- HTTPS:https://streaming.flashphoner.com:8444/rest-api/rels/startup

Здесь:

- streaming.flashphoner.com- адрес WCS-сервера
- 8081 - стандартный REST / HTTP порт WCS-сервера
- 8444- стандартный HTTPS порт
- rest-api- обязательный префикс
- /rels/startup- используемый REST-вызов

REST методы и статусы ответа

/rels/startup

Запустить сбор определенных типов данных для определенного потока

Request example

```
POST /rest-api/rels/startup HTTP/1.1
Host: localhost:8081
Content-Type: application/json

{
    "mediaSession": {
        "frequency":100,
        "ids": [
            "d7d6b6e4-b137-461a-8a32-d6abf2b8666e",
            "39cbf770-128a-11ef-b839-d1a1f53f8bd2"
        ]
    }
}
```

Response example

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json
```

Return codes

Code	Reason
200	OK
400	Bad request
404	Not found
500	Internal server error

/rels/find_all

Получить статистику по отправляемым данным

Request example

```
POST /rest-api/rels/find_all HTTP/1.1
Host: localhost:8081
Content-Type: application/json
```

Response example

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json

[
  {
    "type": "CONNECTION",
    "sentBytes": 1989,
    "bitrateKbps": 0,
    "sentEvents": 6,
    "queueEvents": 0
  },
  {
    "type": "STREAM",
    "sentBytes": 70766,
    "bitrateKbps": 0,
    "sentEvents": 73,
    "queueEvents": 1
  },
  {
    "type": "CDN",
    "sentBytes": 0,
    "bitrateKbps": 0,
    "sentEvents": 0,
    "queueEvents": 0
  },
  {
    "type": "MEDIA_SESSION",
    "ids": [
      "d7d6b6e4-b137-461a-8a32-d6abf2b8666e",
      "39cbf770-128a-11ef-b839-d1a1f53f8bd2"
    ],
    "sentBytes": 205794,
    "bitrateKbps": 143,
    "sentEvents": 999,
    "queueEvents": 119
  }
]
```

Return codes

Code	Reason
200	OK
404	Not found
500	Internal server error

/rels/terminate

Остановить сбор определенных типов данных для определенного потока

Request example

```
POST /rest-api/rels/terminate HTTP/1.1
Host: localhost:8081
Content-Type: application/json

{
    "mediaSession": {
        "ids": [
            "d7d6b6e4-b137-461a-8a32-d6abf2b8666e"
        ]
    }
}
```

Response example

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json
```

Return codes

Code	Reason
200	OK
400	Bad request
404	Not found
500	Internal server error

/rels/terminate_all

Остановить сбор определенных типов данных для всех потоков

Request example

```
POST /rest-api/rels/terminate_all HTTP/1.1
Host: localhost:8081
Content-Type: application/json

{
    "types": [
        "MEDIA_SESSION"
    ]
}
```

Response example

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json

{
  "MEDIA_SESSION": [
    "39cbf770-128a-11ef-b839-d1a1f53f8bd2"
  ]
}
```

Return codes

Code	Reason
200	OK
400	Bad request
404	Not found
500	Internal server error

Параметры

Параметр	Описание	Пример
mediaSession	Объект описания сбора данных MEDIA_SESSION	`"mediaSession": {"frequency":100, "ids": ["12345678-0000-1111"]}`

frequency	Частота сбора данных в миллисекундах	100
ids	Список идентификаторов медиасессий потоков, для которых собираются данные	`["12345678-0000-1111","12345678-3333-4444"]`
hlsSegmenter	Объект описания сбора данных HLS_SEGMENTER	"hlsSegmenter": `{"ids":["stream1"]}`
hlsClient	Объект описания сбора данных HLS_CLIENT	"hlsClient": `{"ids":["stream1"]}`
mixer	Объект описания сбора данных MIXER	"mixer": `{"ids":["12345678-5555-6666"]}`
audioRecovery	Объект описания сбора данных AUDIO_RECOVERY	"audioRecovery": `{"ids":["12345678-7777-8888"]}`
rtmpInBuffer	Объект описания сбора данных RTMP_IN_BUFFER	"rtmpInBuffer": `{"ids":["12345678-9999-AAAA"]}`

Автоматический сбор данных по условиям

В сборке 5.2.2005 добавлена возможность настроить автоматический сбор некоторых типов данных по условиям. Если опубликованный на сервере медиа поток подпадает под заданные условия, начинается отправка данных указанного типа для этого потока.

Поддерживаются следующие типы данных:

- MEDIA_SESSION
- MIXER
- HLS_SEGMENTER

Условия задаются в виде фильтров в файле `/usr/local/FlashphonerWebCallServer/conf/rels_trap.json`

```
{
    "mediaSession":
    [
        {
            "ips": ["127.0.0.1"],
            "frequency": "1000ms"
        },
        {
            "ips": ["192.168.0.0/24", "192.168.2.0/24"],
            "streamName": ".*\\-screen"
        },
        {
            "ips": ["192.168.0.101/32"],
            "streamName": "test.*",
            "frequency": "100th"
        }
    ],

    "mixer":
    [
        {
            "name": "conference\\-.*",
            "streamName": "user.*",
            "frequency": "1000ms"
        }
    ],

    "hlsSegmenter":
    [
        {
            "streamId": "test.*"
        }
    ]
}
```

Здесь:

- mediaSession - блок фильтров для отправки данных MEDIA_SESSION
    - ips - список IP адресов публикующих клиентов
    - streamName - маска имени публикуемого потока в виде регулярного выражения
    - frequency - частота сбора данных для потока, подпадающего под фильтр
- mixer - блок фильтров для отправки данных MIXER

        - name - маска имени микшера в виде регулярного выражения
        - streamName - маска имени входящего потока микшера в виде регулярного выражения
        - frequency - частота сбора данных для микшера, подпадающего под фильтр
    - hlsSegmenter - блок фильтров для отправки данных HLS_SEGMENTER
        - streamId - маска имени HLS потока в виде регулярного выражения

В списке IP адресов для фильтрации публикующих клиентов могут быть указаны как точные IP адреса, так и маски адресов в CIDR формате

```
{
    "mediaSession":
    [
        {
            "ips": ["127.0.0.1", "192.168.0.0/24"],
            ...
        },
        ...
    ],
    ...
}
```

Если в условиях фильтрации типа данных MEDIA_SESSION указаны и список адресов, и имя публикуемого потока, то данные будут собираться только для потока, имя которого и адрес публикующего клиента подпадают под заданные условия

```
{
    "mediaSession":
    [
        ...,
        {
            "ips": ["192.168.2.0/24"],
            "streamName": ".*\\-screen",
            ...
        },
        ...
    ],
    ...
}
```

Если в условиях фильтрации типа данных MIXER указаны и имя микшера, и имя входящего потока микшера, то сбор данных для этого микшера начнется только после добавления в него потока, имя которого соответствует заданному условию

```
{
    ...,

    "mixer":
    [
        {
            "name": "conference\\-.*",
            "streamName": "user.*",
            "frequency": "1000ms"
        }
    ],

    ...
}
```

Параметр `frequency` определяет частоту сбора данных для этого типа:

- `"frequency": "100ms"` - данные собираются каждые 100 мс
- `"frequency": "100th"` - данные собираются каждые 100 полученных пакетов

Изменения в файле `rels_trap.json` не требуют перезапуска сервера, но применяются к новым публикациям, микшерам и HLS потокам.

Выборки иформации из БД

Выборки информации из БД производятся при помощи SQL запросов в клиенте ClickHouse

Примеры запросов для сборок WCS до 5.2.1999

```
select timestamp,ip,sessionId,mediaSessionId,streamName,dictGetString('wcs.DictionaryStreamEvents','type',
eventType) as eventType from wcs.StreamEvent where streamName = 'test'
```

```
select timestamp,ip,sessionId,dictGetString('wcs.DictionaryConnectionEvents','type', eventType) as eventType
from wcs.ConnectionEvent
```

```
select timestamp,ip,nodeId,dictGetString('wcs.DictionaryCDNEvents','type', eventType) as eventType,eventPayload
from wcs.CDNEvent
```

Примеры запросов для сборок WCS 5.2.1999 и новее

```
select timestamp,ip,sessionId,mediaSessionId,streamName,dictGetString('wcs.DictionaryStreamEvents','type',
eventType) as eventType from test1flashphonercom_192168065_521999.StreamEvent where streamName = 'test'
```

```
select timestamp,ip,sessionId,dictGetString('wcs.DictionaryConnectionEvents','type', eventType) as eventType
from test1flashphonercom_192168065_521999.ConnectionEvent
```

```
select timestamp,ip,nodeId,dictGetString('wcs.DictionaryCDNEvents','type', eventType) as eventType,eventPayload
from test1flashphonercom_192168065_521999.CDNEvent
```