

# Рекомендации по тонкой настройке сервера

Настройки сервера по умолчанию в большой степени универсальны и могут нуждаться в подстройке под определенный сценарий клиента.

## Настройка REST клиента

При использовании [REST hooks](#), на каждое действие WCS сервера (присоединение клиента, публикация и воспроизведение потока, установка звонка и т.д.) создается HTTP REST соединение к бэкенд-серверу. При большом количестве одновременно публикующих клиентов или подписчиков, при настройках по умолчанию возможно исчерпание пула потоков встроенного REST клиента WCS, что, в свою очередь, может приводить к блокировкам (deadlock). При этом сервер перестает публиковать и воспроизводить потоки.

По умолчанию, максимальное количество одновременных REST соединений установлено в 200 при помощи параметра в файле [flashphoner.properties](#)

```
rest_max_connections=200
```

Для того, чтобы избежать переполнения пула потоков и блокировок, необходимо уменьшить это значение, например

```
rest_max_connections=20
```

Если [REST hooks](#) не используются, REST клиент может быть отключен при помощи настройки

```
disable_rest_requests=true
```

## Ограничение логирования

При использовании [REST hooks](#) в серверный лог записывается работа REST-клиента, работа встроенного бэкенда EchoApp, а также работа REST API сервера. Это приводит к большому количеству записей в лог и, следовательно, увеличивает нагрузку на сервер. При необходимости, объем логирования может быть уменьшен при помощи следующих параметров в файле [log4j.properties](#):

```
log4j.logger.RestClient=WARN  
log4j.logger.EchoApp=WARN
```

```
log4j.logger.RestApiRouter=WARN
```

## Настройка UDP

При стриминге медиаданные передаются UDP-пакетами. UDP-пакеты могут отбрасываться, если, например, сервер не успевает разобрать очередь пакетов, что ведет к ухудшению качества изображения, фризам. Для того, чтобы этого избежать, необходимо подстроить буферы UDP-сокетов настройками в файле [flashphoner.properties](#)

```
rtp_receive_buffer_size=131072  
rtp_send_buffer_size =131072
```

а также настроить системные очереди командой

```
ip link set txqueuelen 2000 dev eth0
```

Для того, чтобы диагностировать проблему с UDP, необходимо отследить сброс UDP-пакетов командой

```
dropwatch -l kas  
>start
```

## Настройка буферов UDP на уровне системы

Если на сервере публикуются потоки с высоким битрейтом, и для публикации и проигрывания используется UDP (например, в качестве транспорта внутри CDN), может потребоваться настройка буферов UDP на уровне системы

```
sudo sysctl -w net.core.rmem_max=26214400  
sudo sysctl -w net.core.rmem_default=26214400
```

При этом производительности сервера должно хватать для обработки всех поступающих пакетов трафика. В противном случае качество трансляции ухудшится (появятся фризы), но узким местом станет не сеть, а CPU.

## Оптимизация загрузки канала

Качество картинки при воспроизведении у пользователя зависит от битрейта: чем выше битрейт, тем выше качество. Однако, чем выше битрейт, тем больше загружается канал передачи данных, и, если полоса пропускания между сервером и клиентами ограничена, есть вероятность, что канал будет загружен полностью. Это приводит к сбросу битрейта и резкому снижению качества картинки.

В связи с этим, необходимо ограничивать битрейт для того, чтобы обеспечить достаточное качество картинки при приемлемой загрузке канала.

## Ограничение битрейта при публикации

Для снижения нагрузки на канал от публикующей стороны до сервера можно задать минимальное и максимальное значения битрейта в кбит/с в скрипте публикации при помощи JavaScript API

```
session.createStream({
  name: streamName,
  display: localVideo,
  constraints: {
    video: {
      minBitrate: 500
      maxBitrate: 1000
    }
  },
  ...
}).publish();
```

## Ограничение битрейта на сервере

Минимальное и максимальное значение битрейта в бит/с на сервере устанавливается следующими настройками в файле [flashphoner.properties](#)

```
webrtc_cc_min_bitrate=500000
webrtc_cc_max_bitrate=1000000
```

Также нужно включить декодирование потоков на сервере только по запросу, для снижения нагрузки на сервер

```
streaming_video_decoder_fast_start=false
```

## Изменение диапазона динамических портов в Linux

Динамический или эфемерный порт, — временный порт, открываемый при установке IP-соединения из определённого диапазона программного стека TCP/IP. Многие версии ядра Linux используют в качестве динамических порты 32768 — 61000. Проверить, какой именно диапазон используется на сервере, можно при помощи команды

```
sysctl net.ipv4.ip_local_port_range
```

Если этот диапазон пересекается со стандартными [портами](#), используемыми WCS, его необходимо изменить при помощи команды

```
sysctl -w net.ipv4.ip_local_port_range="59999 63000"
```

## Изменение максимального количества открытых файлов

### Устаревший способ (сборки до [5.2.762](#))

В скрипте запуска webcallserver, расположенном в подкаталоге bin в каталоге установки WCS, например

```
/usr/local/FlashphonerWebCallServer/bin/webcallserver
```

в функции start() указано значение максимального количества открытых файлов

```
function start() {  
    ...  
    echo -n "$PRODUCT: starting"  
  
    ulimit -n 20000  
    if [[ "$1" == "standalone" ]]; then  
        ...  
    fi  
    ...  
}
```

По умолчанию, данное значение установлено в 20000, но при необходимости его можно увеличить, учитывая ограничения используемой операционной системы.

### Использование переменных среды (начиная со сборки [5.2.762](#))

Начиная со сборки [5.2.762](#), максимальное количество открытых файлов может быть настроено при помощи переменной

```
WCS_FD_LIMIT=20000
```

в файле `setenv.sh`. При обновлении с предыдущей сборки, желаемое значение необходимо добавить в этот файл вручную, например

```
export WCS_FD_LIMIT=100000
```

В отличие от скрипта запуска `webcallserver`, файл `setenv.sh` при последующих обновлениях не перезаписывается, таким образом, нет необходимости при каждом обновлении восстанавливать эту настройку.

## Использование параметров сервиса при запуске от непривилегированного пользователя (начиная со сборки [5.2.801](#))

Начиная со сборки [5.2.801](#), WCS запускается от пользователя 'flashphoner' для большей безопасности. В этом случае изменение максимального количества открытых файлов возможно в параметрах сервиса

```
sudo nano /etc/systemd/system/webcallserver.service
```

Количество открытых файлов задается параметром `LimitNOFILE`, например

```
[Service]
User=flashphoner
Group=flashphoner
LimitNOFILE=100000
...
```

## Команда для изменения максимального количества открытых файлов

В сборке [5.2.1255](#) добавлена команда для изменения максимального количества открытых файлов

```
sudo ./webcallserver set-fd-limit 100000
```

При этом WCS будет остановлен перед внесением изменений в настройки и автоматически запущен после внесения изменений.

Если указано меньшее количество, чем значение по умолчанию (20000), будет выведено сообщение об ошибке, и изменения не будут применены.

## Шифрование в отдельном потоке для каждой клиентской сессии

По умолчанию, шифрованием трафика для всех клиентских сессий занимается один процессорный поток. Это приводит к тому, что на маломощных серверах при большом количестве подписчиков такой поток нагружает одно процессорное ядро, после чего сервер не успевает рассылать медиапакеты подписчикам, и начинается деградация воспроизводимых потоков, снижение FPS и фризы.

Чтобы распределить нагрузку равномерно по ядрам процессора, необходимо включить шифрование трафика в отдельном процессорном потоке для каждой из клиентских сессий при помощи следующих настроек

```
rtp_paced_sender=true
rtp_paced_sender_initial_rate=200000
```

```
rtp_paced_sender_increase_interval=50  
rtp_paced_sender_k_up=0.9
```

и перезапустить WCS.

## Оптимизация доставки потока подписчикам

При большом количестве подписчиков на один поток (от 100 и более) качество воспроизведения потока может падать: низкий FPS, фризы. При этом нагрузочной способности сервера и канала может быть достаточно. В таких случаях рекомендуется включить распределение доставки потока подписчикам по процессорным потокам при помощи настройки

```
streaming_distributor_subgroup_enabled=true
```

При этом клиентские аудио и видео сессии распределяются по группам.

Максимальное количество видео сессий в группе задается настройкой

```
streaming_distributor_subgroup_size=50
```

Максимальное количество аудио сессий в группе задается настройкой

```
streaming_distributor_audio_subgroup_size=500
```

Размеры очередей пакетов на группу и максимальное время ожидания фрейма для отправки (в миллисекундах) задаются настройками

```
streaming_distributor_subgroup_queue_size=300  
streaming_distributor_subgroup_queue_max_waiting_time=5000
```

для видео и

```
streaming_distributor_audio_subgroup_queue_size=300  
streaming_distributor_audio_subgroup_queue_max_waiting_time=5000
```

для аудио соответственно.