CDN 2.0

CDN based on WCS servers can be organized in two ways:

1. Static CDN with pre-configured nodes. Changing CDN configuration requires restarting the server(s), because in this scheme the server(s) are the source of streams in the network. Such CDN 1.0 can be organized based on the load balancing function.
2. Dynamic CDN with dynamically changing nodes. To include/exclude a node to/from such CDN, restarting of this node only is required.

In this section we describe dynamic CDN based on Web Call Server 5.1.

Overview

Dynamic CDN operation principle

Distributed dynamic content delivery network (CDN) based on WCS 5.1 operates as follows:
Upon startup of the server, a special CDN module starts. The module sends a request to the server specified as the CDN entry point to retrieve from it the list of other servers in the CDN and the list of available streams. If the entry point is not specified or if the specified server is not available, the module waits for a notification from another server in the CDN (for example, if that server was already added to the list of CDN servers on another server, or if it was specified as the entry point). As a result, each active CDN server at any given moment has the up-to-date list of all other CDN servers.
All interaction between nodes of CDN is performed via Websocket.

Server roles in CDN

A server can perform one of two roles

1. Origin: works as a source of streams for other servers in CDN. Can distribute both streams published on it and streams fetched from other servers.
2. Edge: can fetch streams from Origin servers, but do not distribute its own (local) streams to other servers.

Streams are not distributed by force. An Edge server fetches a specific stream from Origin only upon user request from a browser or a mobile application.

Operation flowchart
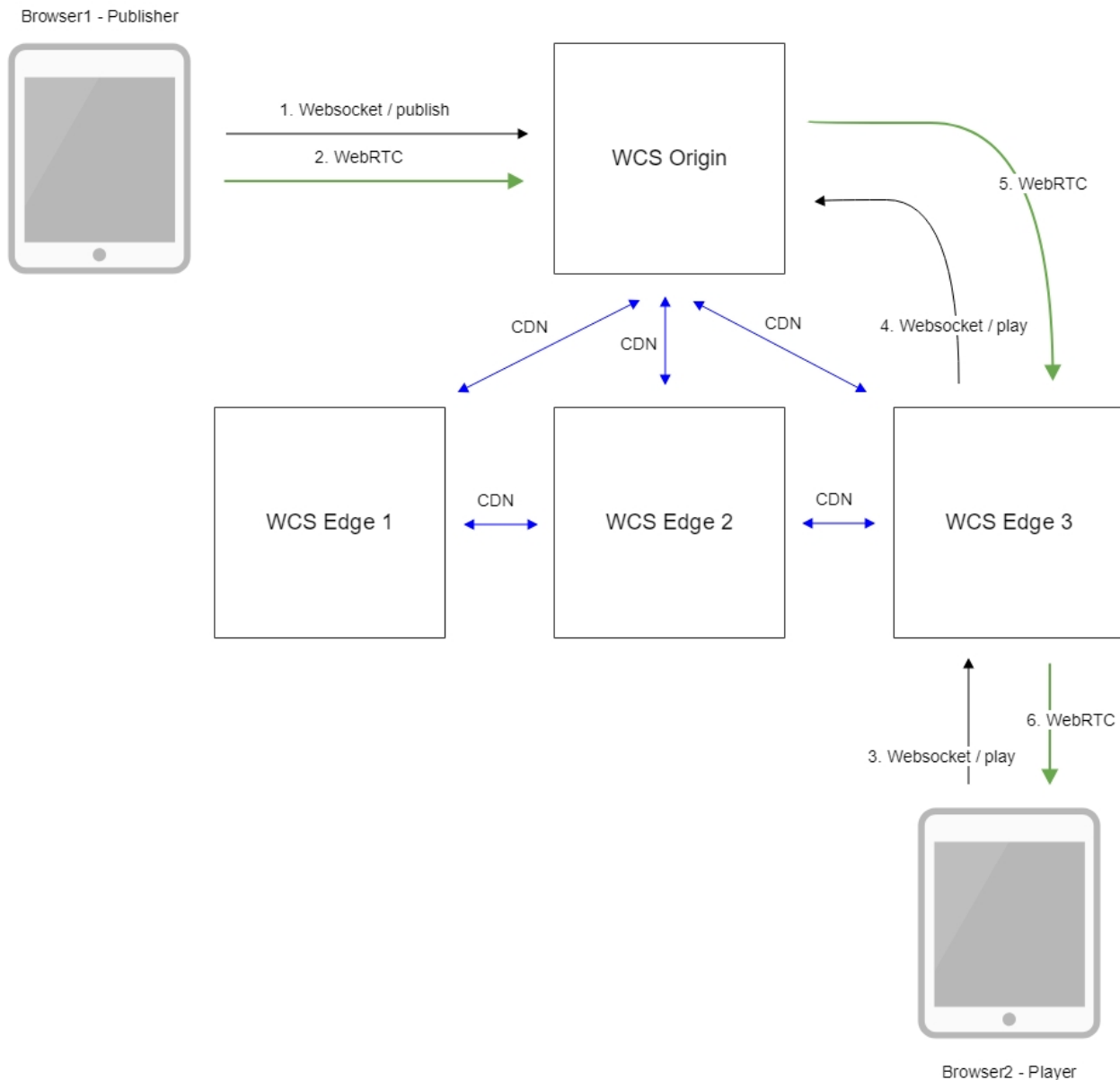
1. The browser connects to the Origin server via the Websocket protocol and sends the publish command.
2. The browser captures the microphone and the camera and sends the WebRTC stream to the server.
3. The second browser establishes a connection to the Edge3 server via Websocket and sends the play command.
4. The Edge3 server requests the stream from the Origin server.
5. The Edge3 server receives the WebRTC stream.
6. The second browser receives the WebRTC stream and plays this stream on the page.

Stream identification principles in CDN

A stream name is unique for every WCS server (and identifies the stream unambiguously), but not for the CDN in a whole. Hence, there are two limitations:

1. If a stream is published on the server, and there is another stream with the same name on some other Origin server, the local (non-CDN) stream will be played on that server.
2. When a CDN stream is played on the server, a stream with the same name cannot be published on that server. The same name can be used for publishing only when the playback stops and the pull agent is deleted after the activity checking timeout expires (by default it is 1 minute).

Configuration

Parameters of the settings file

To configure CDN, the following main settings in the flashphoner.properties file are used (see the complete list in the settings file description):

Parameter	Default value	Type	Description
cdn_enabled	false	Boolean	Enable/disable CDN
cdn_ip	Empty	String	Server address in CDN
cdn_nodes_resolve_ip	false	Boolean	Resolve server names to IP addresses
cdn_point_of_entry	Empty	String	Address of the server, the entry point to CDN for the given server. If the server is Origin, this parameter should not be set to this servers' IP address or hostname
cdn_port	8084	Int	Port number for CDN
cdn_role	edge	String	Role of the server in CDN: origin - source of streams for other servers in CDN edge - can receive streams from other servers

Configuration examples

Minimum configuration example

Two servers: Origin (origin.flasphoner.com) and Edge (edge.flasphoner.com)

Origin settings:

```
cdn_enabled=true
cdn_ip=origin.flashponer.com
cdn_nodes_resolve_ip=true
cdn_role=origin
```

Edge settings:

```
cdn_enabled=true
cdn_ip=edge.flashponer.com
cdn_nodes_resolve_ip=true
cdn_point_of_entry=origin.flashponer.com
cdn_role=edge
```

TwoOriginserversconfigurationexample

Four servers: Origin1 (origin1.flasphoner.com), Origin2 (origin2.flasphoner.com) and Edge1 (edge1.flasphoner.com), Edge2 (edge2.flasphoner.com)

Origin1 settings:

```
cdn_enabled=true
cdn_ip=origin1.flashponer.com
cdn_nodes_resolve_ip=true
cdn_role=origin
```

Origin2 settings:

```
cdn_enabled=true
cdn_ip=origin2.flashponer.com
cdn_point_of_entry=origin1.flashponer.com
cdn_nodes_resolve_ip=true
cdn_role=origin
```

Edge1 settings:

```
cdn_enabled=true
cdn_ip=edge1.flashphoner.com
cdn_point_of_entry=origin1.flashponer.com
cdn_nodes_resolve_ip=true
cdn_role=edge
```

Edge2 settings:

```
cdn_enabled=true
cdn_ip=edge2.flashphoner.com
cdn_point_of_entry=origin1.flashponer.com
cdn_nodes_resolve_ip=true
cdn_role=edge
```

Using CLI

Obtaining information about the current status of CDN from the WCScommand lineis performed with these commands:

Command	Description	Example result
show cdn-nodes	Displays the list of node servers in CDN: ACTIVE - the server is running, responds to queries and/or sends notifications PASSIVE - the server is stopped or unavailable	```IIp State -------------------------------- edge1.flashphoner.com ACTIVE edge2.flashphoner.com ACTIVE origin2.flashphoner.com PASSIVE```
show cdn-routes	Shows the list of active streams in CDN	```Route Stream ----------------------------------- 1-origin2.flashphoner.com-2 stream1 0-origin2.flashphoner.com-0 stream2 2-origin1.flashphoner.com-1 stream1```

Using REST API

A REST-query should be HTTP/HTTPS POST request as follows:

- HTTP:http://test.flashphoner.com:8081/rest-api/cdn/show_routes
- HTTPS:https://test.flashphoner.com:8444/rest-api/cdn/show_routes

Where:

- test.flashphoner.com - is the address of the WCS server
- 8081 - is the standard REST / HTTP port of the WCS server
- 8444 - is the standard HTTPS port
- rest-api - is the required part of the URL
- /cdn/show_routes - REST-method to use

REST queries and response statuses

REST query	REST query example	REST response example	Response states	Description
/cdn/show_routes		```{ "1-origin2.flashphoner.com-2": "stream1", "0-origin2.flashphoner.com-0": "stream2", "2-origin1.flashphoner.com-1": "stream1" }```	200 – OK 500 – Internal Server Error	Show active CDN routes

Parameters

Description	Example
Route identifier	`1-origin2.flashphoner.com-2`
Stream name in CDN	`stream1`

Removing stopped servers from CDN nodes list

By default, when server is stopped or unavailable, it will be displayed in CDN nodes list in PASSIVE state until it will be started again or all the CDN will be stopped. The time interval may be set if necessary to remove inactive node from nodes list. This interval should be set in seconds using the following parameter in flashphoner.properties file, for example

```
cdn_nodes_timeout=60
```

In this case inactive nodes will be removed from CDN nodes list after 60 seconds of inactivity.

Choosing audio codecs for stream forwarding through CDN

Forwarding audio through CDN

When forwarding audio through CDN, a set of available codecs is formed with the

- codec used to publish stream (as preferred), and
- codecs supported on Origin and Edge servers (`codecs` setting in flashphoner.properties file)

excluding codecs listed in `codecs_exclude_cdn` setting on Edge.

By default, if AAC is not excluded, the following AAC sample rates are defined in SDP: 48, 44.1, 32, 24, 22.05, 16, 12, 8 kHz.

For example, if Edge requests from Origin RTMP stteam published with AAC 48 kHZ, audio SDP will be following

```
m=audio 31006 RTP/SAVPF 102 111 8 18 100 9 103 104 105 106 107 108 109 110
c=IN IP4 192.168.1.5
a=mid:1
a=rtpmap:102 mpeg4-generic/48000/2
a=rtpmap:111 opus/48000/2
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=fmtp:18 annexb=no
a=rtpmap:100 speex/16000
a=rtpmap:9 G722/8000
a=rtpmap:103 mpeg4-generic/44100/2
a=rtpmap:104 mpeg4-generic/32000/2
a=rtpmap:105 mpeg4-generic/24000/2
a=rtpmap:106 mpeg4-generic/22050/2
a=rtpmap:107 mpeg4-generic/16000/2
a=rtpmap:108 mpeg4-generic/12000/2
a=rtpmap:109 mpeg4-generic/11025/2
a=rtpmap:110 mpeg4-generic/8000/2
```

PCMU is not available by default and will be enabled only when PCMA is excluded:

```
codecs_exclude_cdn=alaw
```

Custom SDP settings for RTMP (flash_handler_publish.sdp and flash_handler_play.sdp) does not affect SDP for sound forwarding through CDN.

With the default settings, transcoding will not occur, for example, in the following cases.:

- WebRTC (Opus) stream published on Origin is played as WebRTC on Edge (Opus)
- RTMP (AAC, any of sample rates listed above) stream published on Origin is played as RTMP on Edge (AAC with the same sample rate without resampling)
- RTMP (Speex) stream published on Origin is played as RTMP on Edge (Speex)

If there is no codec used to publish a stream in the subscribers codecs set, then transcoding will be on Edge server:

- WebRTC (Opus) stream published on Origin is played asRTMP on Edge (AAC): Opus-AAC transcoding on Edge
- RTMP (AACstream published on Origin is played asWebRTC on Edge (Opus): AAC-Opustranscoding on Edge
- RTMP (Speex) stream published on Origin is played asWebRTC on Edge (Opus): Speex-Opustranscoding on Edge
- RTMP (Speex) stream published on Origin is played asRTMP on Edge (AAC): Speex-AACtranscoding on Edge

If the subscribers codecs set includes no only AAC, then Speex for example may be used to playback RTMP as RTMP without transcoding.
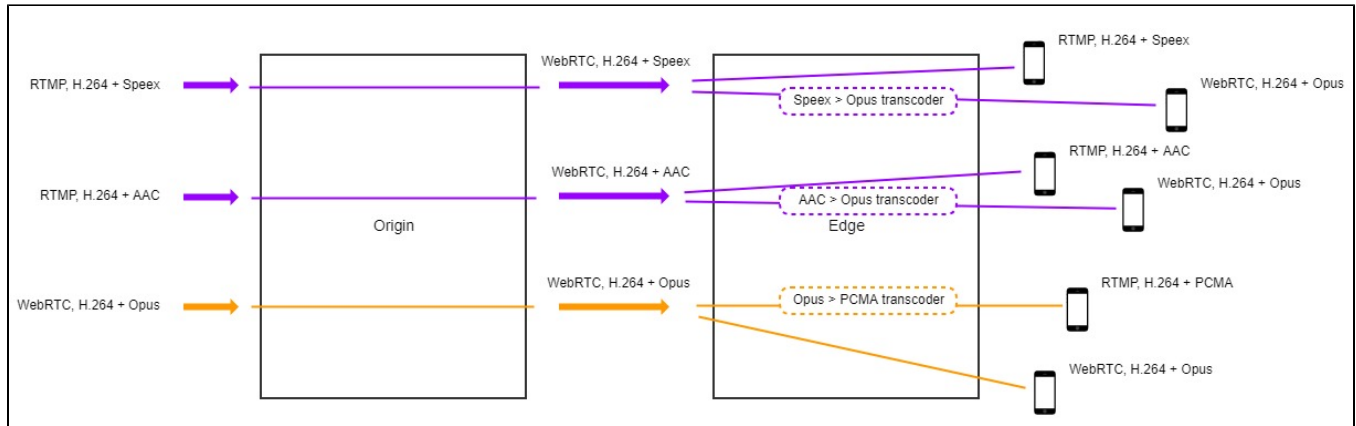
If the subscribers codecs set is limited to AAC only (flash_handler_play.sdp on Edge includes AAC only), then transcoding will be on Edge if another codec is used to publish RTMP stream:
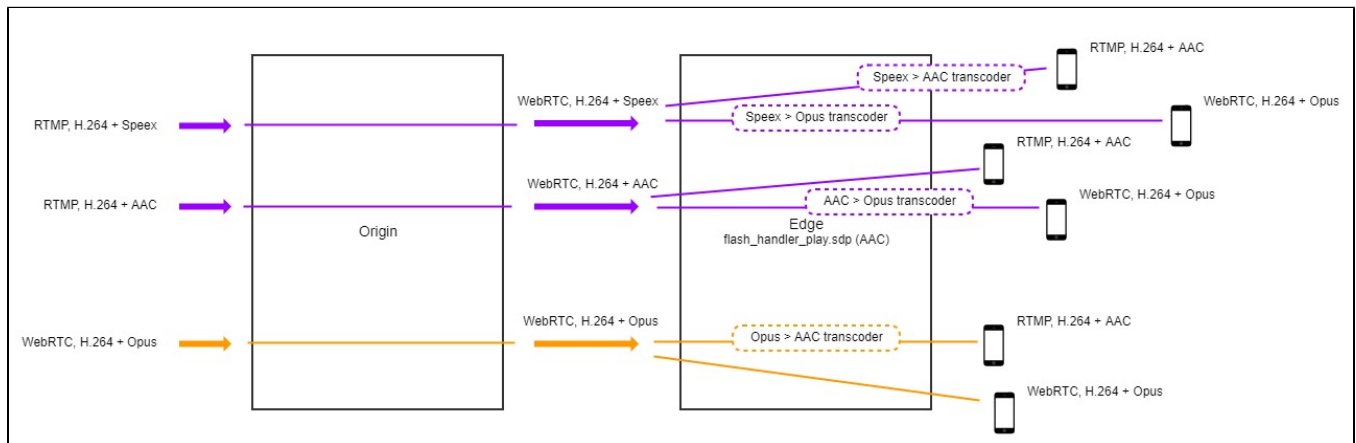

To exclude codecs while stream forwarding trough CDN, to escape transcoding on Edge server,codecs_exclude_cdnsetting should be used on Edge. Then the only remaining codec will be used to forward a stream trough CDN, and if stream is published with another codec transcoding will be on Origin server.

For example, if supposed to be connected to Edge

1. RTMP subscribers with AAC only: all the codecs should be excluded except mpeg4-generic:

```
codecs_exclude_cdn=opus,alaw,ulaw,g729,speex16,g722,telephone-event,flv
```
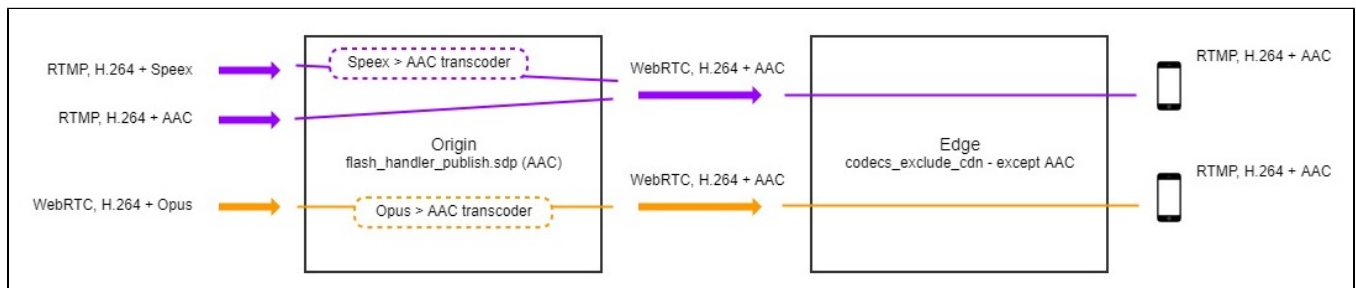

2. WebRTC subscribers with Opus only: all the codecs should be excluded exceptopus:

```
codecs_exclude_cdn=mpeg4-generic,alaw,ulaw,g729,speex16,g722,telephone-event,flv
```
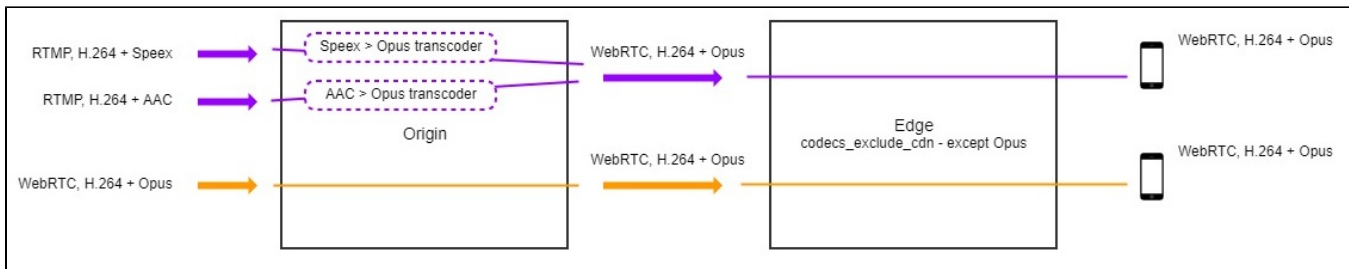

Custom SDP setting to publish and play RTMP stream in CDN

flash_handler_publish.sdp

flash_handler_publish.sdp is used as publish SDP when subscriber connects to stream before WCS get audio and video from publishing client (i.e. before the real codec used by publisher is known).

In this case codecs priority will be determined in order set in flash_handler_publish.sdp. If there is no flash_handler_publish.sdp, then the default priority audio codec will be Speex.

For example, with default settings (without flash_handler_publish.sdp), when RTMP (AAC) stream published on Origin is played as WebRTC (Opus) on Edge through CDN

- if subscriber connects after receiving audio from publisher (i.e. it is known that AAC is used): AAC-Opus transcoding on Edge
- if subscriber connects before receiving audio from publisher (i.e. default Speex is used): AAC-Speex transcoding on Origin and Speex-Opus transcoding on Edge

If Origin has flash_handler_publish.sdp, in which AAC is set as first for example, then AAC will be used including the case when subscriber connects beforereceiving audio from publisher. Thus, if publication codec is known in advance then additional transcoding can be escaped by setting a codec in flash_handler_publish.sdp.

flash_handler_publish.sdp is not used to restrict publication codecs: if any codec is excluded from flash_handler_publish.sdp, it can still be used for publishing.

flash_handler_play.sdp

flash_handler_play.sdp is used for subscriber SDP only, and not affect codecs restrictions when frowarding stream through CDN. If some code is excluded from flash_handler_play.sdp, it will not be used to play RTMP stream by subscriber.

Choosing audio codec to play a stream

If there is a codec used to publish the stream in subscrbers codecs set then this codec will be used to play the stream (regardless of its priority). Otherwise, the subscribers priority codec will be used for playback, and transcoding will occur.

Read and write timeout settings in CDN for RTMP streams delivery

If CDN is used mostly for publishing and playing RTMP streams, and Keep Alive packets is disabled on CDN servers for some reason (for example, publishers and players do not support Keep Alives), it is necessary to setup read and write timeouts to control state of RTMP connections:

1. Read timeout can be set on Origin server that is used only to publish streams, which will not be played directly from the Origin

```
keep_alive.algorithm=NONE
rtmp.server_read_socket_timeout=120
```

2. Write timeout can be set on Edge server that is used only to play CDN streams

```
keep_alive.algorithm=NONE
rtmp.server_write_socket_timeout=120
```

3. Read and write timeout can be used both on Origin and Edge servers

```
keep_alive.algorithm=NONE
rtmp.server_socket_timeout=120
```

Quick manual on testing

1. For testing we use:

- two WCS servers;
- the Two Way Streaming web application to publish the stream;
- the Player web application to play the stream.

Configuring WCS

2. For the test, configure CDN in the minimum configuration: one Origin and one Edge server

Origin settings:

```
cdn_enabled=true
cdn_nodes_resolve_ip=true
cdn_ip=origin.flashphoner.com
cdn_role=origin
```

Edge settings:

```
cdn_enabled=true
cdn_nodes_resolve_ip=true
cdn_point_of_entry=origin.flashphoner.com
cdn_ip=edge.flashphoner.com
cdn_role=edge
```

Here origin.flashphoner.com and edge.flashphoner.com - are examples names of the WCS servers.

Restart Origin and Edge servers. Enter the command line interface of the Edge server, authorize and enter this command

```
show cdn-nodes
```

The result of the command is as follows:

```
> show cdn-nodes
Ip                      State
-------------------------
origin.flashphoner.com ACTIVE
```

Running a broadcast form a web camera to the Origin server

3. Open the Two Way Streaming web application on the Origin server. Click the Connect button and specify the name of the broadcast stream testCDN:

Two-way Streaming

Local

Player

| testCDN | Publish |

| d730 | Play | Available |

| wss://p11.flashphoner.com:8443 | Disconnect |

ESTABLISHED

4. Click the Publish button. The stream is published from the web camera:

Two-way Streaming

Local

Player

testCDN | Stop

d730 | Play | Available

PUBLISHING

wss://p11.flashphoner.com:8443 | Disconnect

ESTABLISHED

Playing the stream on the Edge server

5. Open the Player web application on the Edge server. Specify the name of the stream broadcast to the Origin server, testCDN:

WCS URL | wss://p12.flashphoner.cor

Stream | testCDN

Volume

Full Screen

Start

6. Click the Start button. Playing of the testCDN stream starts.

Player

| WCS URL | wss://p12.flashphoner.cor |
| Stream | testCDN |

7. In the command line interface of the Edge server enter the command

```
show cdn-routes
```

The result of executing the command is:

```
> show cdn-routes
Route                     Stream
-------------------------------
0-origin.flashphoner.com-0 testCDN
```

Stopping broadcasting from the web camera to the Origin server

8. Click the Stop button on the stream publishing page. Publishing stops:

On the play page of the Edge server playing of the stream also stops and the message "Stopped by publisher stop" displays.

Known limits

1. It is strongly not recommended to publish streams with same name to two Origin servers in the same CDN.

2. A stream published to one of Origin servers should be played on the same Origin server or any Edge server, but should not be played from another Origin server in the same CDN.

Known issues

1. Changing codec settings on the Edge server can lead to increased load on the server as a result of enabled transcoding.

Symptoms: when a large number of streams are broadcast, the load on the CPU of the Edge server increases
Solution:configure codec settingson Origin and Edge servers so to reduce excessive transcoding.

For instance, if the codec setting on the Origin server specifies the following set of codecs:

```
codecs=opus,mpeg4-generic,alaw,ulaw,g729,g722,telephone-event,h264,vp8
```

and the setting on the Edge server is

```
codecs=opus,speex16,mpeg4-generic,g729,g722,h264,vp8
```

then, when a stream is broadcast through the Edge server, audio transcoding from Speex16 or AAC is activated. With a large number of streams, the load to the CPU of the server can greatly increase.

2. Stream playback on Edge server may be terminated when Keep Alive is disabled on Origin server

Symptoms: when Keep Alive is disbled on Origin server

```
keep_alive.algorithm=NONE
```

stream playback on Edge server stops even with the following setting

```
wcs_agent_session_use_keep_alive_timeout=false
```

Solution: turn off WebSocket read timeout on Edge server with the following setting

```
ws_read_socket_timeout=false
```