VTX Media Devices 1: microphones, speakers and cameras in a session

Nov 15, 2026 · VTX 1 · extension · draft d01 · MIT licence · vtx.foundation/spec/v1/mediadev-1/

§1 Status and scope

Media Devices 1 makes the microphones, speakers and cameras of the user's device appear as devices of the hosted desktop. A call or recording program in the session sees "Headset" and "FaceTime HD camera" in its settings, picks them like local devices, and the user hears and is seen through the hardware on the desk. This is RDP's audio input, audio output and camera redirection. It is an extension of VTX 1, draft d01, in conformance profile 3, and the third redirection.

The sound and pictures still travel as Media 1 streams. What Media 1 alone cannot say is which device a stream belongs to. Media Devices 1 adds the device list and a way to tie a stream to a device, and changes nothing in Media 1.

Adds	What it is
`redirect.media`	A capability: the client lists its devices and lets the server choose among them
`mediadev` channel	One channel per connection, opened by the client: the devices, and which Media 1 stream uses which device
`media_device`	An application event: a device appeared on the host, changed or went away

Out of scope. Microphone and camera permissions, the capture indicator and the codecs, which stay Media 1's. Screen capture, which is not a device. Headset buttons such as answer and hang up, which are input devices. Camera controls such as pan, tilt and zoom. Surround sound and spatial audio beyond Media 1's eight channels.

§2 Capabilities and the device channel

The client offers `redirect.media` together with Media 1's `audio`, `audio.capture`, `video` and `video.capture`; the server grants it with `{}`. After READY, the client opens one channel of type `mediadev` and lists its devices on it.

```
cap-mediadev = {
  ? "max": 1..32,            ; client: most devices it lists (default 16)
}
```

Frame	Direction	What it carries
0x10 DEVICES	Client to server	Every microphone, speaker and camera the session may use; again after every change
0x11 BIND	Server to client	Tie a Media 1 stream name to a device
0x12 UNBIND	Server to client	Let that stream use the default device again

Frame	Direction	What it carries
0x13 BOUND	Client to server	The stream now uses that device
0x14 UNBOUND	Client to server	The stream cannot use it, or no longer can: `gone`, `busy` or `unknown`

Only the client opens the channel (an odd ID), at most one per connection. Listing a device gives the server no access to it: capturing still takes a Media 1 capture stream, which the user allows with the `microphone` or `camera` permission, as before.

§3 Devices

DEVICES lists what the session may use, with what a program's settings page needs to show: a name, the kind, which device is the default and which the user prefers for calls, and what each can do. The client sends it when the channel opens and again whenever something changes: a headset is plugged in, a camera is unplugged, or the user changes a default.

```
md-device = {
  "id": md-device-id,         ; chosen by the client for this connection; never a
hardware id
  "kind": "microphone" / "speaker" / "camera",
  "name": tstr .size (1..64),
  ? "default": true,          ; the device's default of this kind
  ? "calls": true,            ; the device the user prefers for calls
  ? "channels": 1..8,         ; microphones and speakers
  ? "muted": bool,            ; a microphone muted by its own switch
  ? "processing": [* ("echo" / "noise" / "gain")],   ; microphones: what the client can
apply
  ? "max": [uint, uint],      ; cameras: largest size, pixels
  ? "fps": uint,              ; cameras: highest frame rate at that size
  ? "facing": "user" / "environment" / "external",   ; cameras
}
```

A laptop with a USB headset plugged in:

id	kind	name	default	calls	can do
`mic-1`	microphone	Built-in microphone	yes		mono; echo, noise, gain
`mic-2`	microphone	Headset		yes	mono; echo, noise, gain
`spk-1`	speaker	Built-in speakers	yes		stereo
`spk-2`	speaker	Headset		yes	stereo
`cam-1`	camera	FaceTime HD camera	yes		1920 × 1080 at 30 fps, facing the user

Rules. Device ids are unique. Each kind has at most one `default` and one `calls` device. Camera fields (`max`, `fps`, `facing`) appear only on cameras, and `processing` and `muted` only on microphones. At most `max` devices are listed.

Which devices. The user decides which devices the session may see; by default, all microphones, speakers and cameras of the device. A device left out does not exist for the session.

§4 At a glance

> **Diagram: a video call in the session · three Media 1 streams bound to three devices.** Drawn in the PDF version, `15-media-devices-1.pdf`.

The call program sees ordinary devices on the host. Behind each one is a Media 1 stream that BIND has tied to one device on the desk: the microphone and camera stream towards the host, the call's sound streams back to the headset. Echo is removed on the device, where the headset's speaker and microphone share a clock.

§5 Binding streams to devices

A Media 1 stream has a name. BIND ties that name to a device, before the stream's channel opens or while it is open; the client answers with BOUND or UNBOUND. A stream nobody bound uses the device's default, exactly as in Media 1.

```
BIND  {"stream": "call-mic", "device": "mic-2"}
BOUND {"stream": "call-mic", "device": "mic-2"}
OPEN  {"channel": 30, "type": "audio", "params": {"stream": "call-mic", "dir":
"capture", "source": "microphone",
       "codecs": ["opus"], "channels": 1, "processing": ["echo", "noise"]}}
```

Kinds match. A stream binds only to a device of its own kind:

Device	Stream
Microphone	An audio channel with `dir` capture and `source` microphone
Camera	A video channel with `dir` capture and `source` camera
Speaker	An audio channel with `dir` play

Screen capture and video play bind to no device. A server that binds across kinds, or opens a bound stream of another kind, breaks the protocol.

Switching. When the user picks another microphone in a program's settings, the server sends BIND again with the open stream's name. The client moves the stream to the new device without closing the channel; if the format changes, a new Media 1 CONFIG follows.

Unplugging. When a bound device goes away, the client moves its streams to the default device of the kind and sends UNBOUND with `gone` for each, before the DEVICES that no longer lists it. A device held alone by another program on the device gets `busy`, and a BIND for a device that left the list meanwhile gets `unknown`. In every case the stream keeps running on the default, so a call never goes silent because a headset was pulled out.

§6 Echo, noise and delay

Echo cancellation happens on the device. Only the device has the speaker's sound and the microphone's sound on one clock. When a capture stream asks for `echo` (Media 1's `processing`), the client cancels the echo of everything it plays for the session from the microphone's signal. A canceller on the host cannot do this well: the network delay between the sound it sends and the echo that comes back changes from moment to moment.

So the host tells programs that the microphone is already clean. Its virtual microphone carries the platform's mark for an echo-cancelled source, and a call program then leaves its own canceller off instead of running a second one on top. Noise suppression and gain work the same way: the client applies what the stream asks for, and a program that adds its own does no harm.

Delay. Calls use Media 1's `live` mode on QUIC datagrams. The host's virtual devices report the delay that Media 1's FEEDBACK measures, so programs can keep sound and picture in step and size their own buffers.

Cameras. The client encodes at the size and frame rate the stream asks for (Media 1's `max`, `fps` and VIEW), not at the camera's largest, so a 1080p camera feeding a small call tile costs little.

§7 With other extensions, and without the capability

Extension	With media devices
Media 1	Unchanged: the streams, codecs, permissions and the capture indicator are Media 1's. Media Devices 1 only says which device a stream uses
Viewers 1	Only the owner's connections list devices, as with folders and printers: the session's microphone is the owner's. Viewers talk to each other through Viewers 1's own calls
Brokering 1	Each host connection has its own device channel; programs on every host see the same devices
Windows 1, Displays 1	Unrelated: a call program's window can sit on any display, and its camera is still the camera on the desk

Without the capability. Media 1 alone gives the session one microphone, one speaker and one camera: the device's defaults. A call works, but the user cannot pick the headset from inside the program, and pulling the headset out moves the sound to wherever the device sends it.

§8 Application interface

The server gives the host one virtual device for each listed device, named after it, and keeps them in step with DEVICES. Nothing records or plays until a program uses one: when a program opens the virtual microphone, the server binds a stream to the real one and opens the Media 1 capture; when the program stops, the server closes the capture, and the device's indicator goes off.

Host	Microphones and speakers	Cameras
Linux	PipeWire sources and sinks, `Headset (via VTX)`	PipeWire camera nodes, with a V4L2 loopback device for older programs

Host	Microphones and speakers	Cameras
Windows	Audio endpoints of the server's virtual audio driver	A virtual camera through Media Foundation
macOS	Devices of an audio server plug-in	A camera extension

The host marks the default and the calls device as the platform does, such as Windows' default communications device, so a call program picks the headset by itself. A virtual microphone carries the platform's mark for an echo-cancelled source (Section 6).

The desktop shell and the session's settings learn about devices through `media_device` events:

```
{"kind":"media_device","action":"added","id":"mic-2","type":"microphone","name":"Headset","node":"Head
(via VTX)","calls":true}
{"kind":"media_device","action":"changed","id":"mic-2","muted":true}
{"kind":"media_device","action":"removed","id":"mic-2"}
```

§9 Security and privacy

A microphone and a camera are the most sensitive things a session can reach on the device. Media Devices 1 changes which device is used, never whether one may be used.

- **Media 1's permissions still decide.** Capturing takes the `microphone` or `camera` permission, as in Media 1, and the trusted zone shows a capture indicator while a stream runs. With devices listed, the indicator also names the device, "Headset" or "FaceTime HD camera".
- **On only while used.** The server opens a capture only while a program on the host uses the virtual device, and closes it when the program stops. Listing devices records nothing.
- **Little about the hardware.** DEVICES carries the device's name and what it can do, never a serial number, a hardware id or the USB path. Ids are made up per connection, and the user can rename devices or leave them out (Section 3).
- **The device's own switches win.** A microphone muted by its own switch reports `muted`, and the client sends silence while it is; the session cannot unmute it.
- **Only the owner.** Only the owner's connections list devices, so a viewer's microphone never becomes the session's (Section 7).
- **Limits.** At most `max` devices (32 at the most). Media 1's limits on channels and bandwidth apply to the streams.

§10 Definitions and checks

The definitions live in the bundle under `ext/mediadev-1/`:

File	What it holds
`vtx-mediadev-1.cddl`	The capability and the `mediadev` channel: devices, BIND, UNBIND, BOUND and UNBOUND. It loads after v1 and Media 1, whose stream names it uses
`vtx-app-events-mediadev-1.schema.json`	Server lines to an application, plus the `media_device` event. Media Devices 1 adds no scene op and no application line

File	What it holds
`examples/valid/`	The call trace (headset and camera bound, Media 1 channels opened, the headset pulled out, the microphone picked again, the call ended) and the media_device events
`examples/invalid/`	Broken device lists, media_device events and traces, each with the check that must catch it

`tools/gen_ext_mediadev.py` writes the schema, built on the Printers 1 generator; `tools/make_mediadev.py` writes the examples. Both are deterministic.

The validator's Section 23 runs 46 checks, and every one passes:

Part	Checks	What they cover
Setup	6	The schema is valid; the CDDL parses with the core and Media 1, and with every earlier extension; no rule is defined twice; earlier schemas reject `media_device`; no earlier CDDL has BIND
CDDL and device lists	17	5 valid samples accepted and 7 malformed ones rejected; 5 device lists pass the CDDL and fail a rule of Section 3
Traces	17	The valid trace replayed in full, with every Media 1 OPEN checked against what its stream is bound to; 16 broken ones, each caught by the right check, among them a play stream bound to a microphone, a screen capture bound to the camera, a headset dropped while streams are still bound to it, and BOUND for a device that is no longer listed
Application interface	6	4 valid media_device events, 2 rejected

The whole bundle runs 2,543 checks, all passing.

§11 Wire summary

Everything Media Devices 1 adds, for the registry and for implementers:

Kind	Addition
Capability	`redirect.media`: from the client, `max` (1 to 32; default 16); the server grants `{}`
Channel type	`mediadev`, opened by the client, at most one per connection: 0x10 DEVICES, 0x11 BIND, 0x12 UNBIND, 0x13 BOUND, 0x14 UNBOUND
Device kinds	`microphone`, `speaker`, `camera`
UNBOUND reasons	`gone`, `busy`, `unknown`
Application event	`{"kind":"media_device","action":"added", "changed" or "removed","id":…,"type":…,"name":…,"node":…,"default":…,"calls":…,"muted":…}`

No permission is added and no frame of the core or of an earlier extension changes meaning: Media 1's `microphone` and `camera` permissions, streams and codecs are used as they are.