Icecast2 — Ices2

Guide de reconstruction. Il décrit comment monter un serveur de diffusion audio sur un réseau domestique, et comment lui fournir des sources.

Le fichier `.md` de cette page : ouvrir le `.md`

Ce que cette page couvre : installer un serveur Icecast2 sur un réseau local, le configurer, et lui envoyer du son avec `ices2`. Deux types de sources sont possibles ; celle qui sert d'exemple ici est une carte son — tout ce qui entre par une entrée audio ressort en flux réseau. La seconde, un tuner radio logiciel, fait l'objet d'une page distincte à laquelle celle-ci renvoie.

Contexte : Ubuntu Server 24.04, `icecast2` et `ices2` des dépôts de la distribution, serveur et sources sur la même machine — un serveur de fichiers qui porte aussi ce rôle. Deux points de montage sont servis : un flux de radio FM et un flux de carte son. Les adresses, noms d'hôtes et mots de passe sont des exemples.

Deux choses valent le détour même sans monter la chaîne complète. Le fait que les `reconnectattempts` d'ices2 ne couvrent pas la première connexion — ce qui rend l'ordre de démarrage décisif et disqualifie cron pour ce travail : section *Le service*. Et le rechargement des métadonnées par signal, qui change le titre affiché chez les auditeurs sans couper la diffusion : section *Les métadonnées*.

Convention : chaque bloc de commandes s'exécute sur le serveur, sauf mention contraire. Un bloc unique se copie tel quel ; des blocs séparés signifient qu'une édition ou une décision intervient entre eux.

Ce que fait cette chaîne

Icecast est un serveur de diffusion. Il ne produit aucun son : il en reçoit

d’un ou plusieurs clients dits *sources*, et le redistribue à ses auditeurs sous forme de flux HTTP. Chaque flux porte un nom — son *point de montage* — et s’écoute à une adresse ordinaire, du genre `http://192.168.0.11:8000/Chromecast.ogg`, que n’importe quel lecteur sait ouvrir.

`ices2` est l’un de ces clients sources. Il prend du son d’un côté, l’encode en Ogg Vorbis, et le pousse vers un point de montage d’Icecast.

Les deux flux servis ici :

Flux	Source	Rôle
`/Chromecast.ogg`	capture d’une carte son USB, sur laquelle est branché un Chromecast Audio	Envoyer l’audio d’un téléphone vers le serveur de musique de la maison, donc vers toutes les pièces
`/Radio_FM.ogg`	un tuner radio logiciel — voir la page Nooelec NESDR SMART	Repli d’une liste de lecture d’alarme : si le flux web du réveil tombe, le lecteur bascule sur cette entrée

Pourquoi passer par un serveur plutôt que de brancher un câble. Un câble relie une source à un amplificateur. Le serveur, lui, rend la même source disponible à tous les lecteurs du réseau simultanément, sans les recâbler — et il la rend disponible en permanence, ce qui est la seule façon qu’un repli fonctionne.

⚠ Un repli ne vaut que s’il est déjà en ondes. Le lecteur bascule en une fraction de seconde ; il n’attend pas qu’on démarre la source. C’est ce qui impose des services permanents plutôt que des outils lancés à la demande — et c’est le fil conducteur de tout ce qui suit.

Le serveur Icecast2

Installer

```
sudo apt install icecast2
```

L’installation pose quelques questions — nom d’hôte, mots de passe. On peut répondre n’importe quoi : tout est repris ensuite dans le fichier de configuration, qui fait autorité.

Noter la version installée, elle situe ce qui suit :

```
dpkg -l icecast2 ices2 | grep ^ii
```

Les quatre mots de passe, et ce que chacun protège

C’est le point où l’on se trompe le plus souvent, parce que les quatre se ressemblent et que trois d’entre eux ne servent jamais dans une installation domestique.

Mot de passe	Qui le présente	Ce qu’il ouvre
`<source-password>`	les clients sources — ici `ices2`	Le droit de pousser un flux vers un point de montage. C’est le seul dont on se sert quotidiennement.
`<relay-password>`	un autre serveur Icecast	Le droit de retransmettre ce serveur. Inutilisé ici.
`<admin-password>`	un navigateur, sur `/admin/`	L’interface d’administration : voir les montages, les auditeurs, couper une source.
`<password>` de `<authentication>`	idem, avec `<admin-user>`	Le couple identifiant / mot de passe de cette même interface.

⚠ Le `source-password` circule en clair dans chaque connexion de source. Sur un réseau local fermé c’est acceptable ; il ne doit pour autant pas être un mot de passe réutilisé ailleurs, et il figure en clair dans les fichiers de configuration d’`ices2`, dont les permissions comptent donc (voir plus bas).

Le fichier de configuration

`/etc/icecast2/icecast.xml`. Le fichier livré par Debian est un long document dont la plus grande partie est en commentaire : des exemples de relais, de points de montage avancés, d’authentification par URL, tous inactifs. Il faut le savoir

avant de le lire, sous peine de croire configuré ce qui ne l’est pas.

Ce qu’il y a réellement à changer, en tout et pour tout : deux choses.

1. Les mots de passe de la section `<authentication>`.

2. Le `<hostname>`, à mettre à l’adresse du serveur sur le réseau.

Tout le reste peut rester tel quel. Voici la configuration active — commentaires retirés, mots de passe remplacés par `MASQUE` — pour vérification :

```
<icecast>
    <location>Earth</location>
    <admin>icemaster@localhost</admin>

    <limits>
        <clients>100</clients>
        <sources>2</sources>
        <queue-size>524288</queue-size>
        <client-timeout>30</client-timeout>
        <header-timeout>15</header-timeout>
        <source-timeout>10</source-timeout>
        <burst-on-connect>1</burst-on-connect>
        <burst-size>65535</burst-size>
    </limits>

    <authentication>
        <source-password>MASQUE</source-password>
        <relay-password>MASQUE</relay-password>
        <admin-user>admin</admin-user>
        <admin-password>MASQUE</admin-password>
    </authentication>

    <hostname>192.168.0.11</hostname>

    <listen-socket>
```

```
        <port>8000</port>
    </listen-socket>

    <http-headers>
        <header name="Access-Control-Allow-Origin" value="*" />
    </http-headers>

    <fileserve>1</fileserve>

    <paths>
        <basedir>/usr/share/icecast2</basedir>
        <logdir>/var/log/icecast2</logdir>
        <webroot>/usr/share/icecast2/web</webroot>
        <adminroot>/usr/share/icecast2/admin</adminroot>
        <alias source="/" destination="/status.xsl"/>
    </paths>

    <logging>
        <accesslog>access.log</accesslog>
        <errorlog>error.log</errorlog>
        <loglevel>3</loglevel>
        <logsize>10000</logsize>
    </logging>

    <security>
        <chroot>0</chroot>
    </security>
</icecast>
```

⚠ Ce bloc est une vue, pas un fichier à copier tel quel. Il montre ce qui agit ; le fichier réel garde ses commentaires, qui sont sa documentation. Éditer le fichier livré, ne pas le remplacer par cet extrait.

Prendre une copie datée avant toute modification, et la nommer par ce qu’elle précède :

```
sudo cp -p /etc/icecast2/icecast.xml /etc/icecast2/icecast.xml.avant-hostname
```

 Le `-p` conserve la date de l’original : la copie porte alors la date de

l’état sauvegardé, pas celle de la sauvegarde. C’est ce qu’on veut savoir en la retrouvant deux ans plus tard.

Ce que ces valeurs veulent dire

`<hostname>192.168.0.11</hostname>` — l’adresse par laquelle le serveur se désigne lui-même. ⚠ Ce n’est pas cosmétique, et c’est la seule valeur qu’il faille vraiment penser à changer : elle est reprise dans toutes les URL qu’Icecast publie — la page d’état, les listes de flux, les fichiers `.m3u` qu’il génère. Laissée à `localhost`, elle produit des adresses qui ne fonctionnent que depuis la machine elle-même, ce qui déroute tout client qui suit un de ces liens depuis le réseau. Y mettre l’adresse par laquelle le réseau joint la machine.

`<clients>100</clients>` et `<sources>2</sources>` — nombre maximum d’auditeurs simultanés, et de sources. Ce sont les valeurs par défaut de Debian, et le hasard veut que `2` corresponde exactement au nombre de flux servis ici. ⚠ Une troisième source serait donc refusée — y compris le temps d’une bascule, où l’ancienne et la nouvelle se recouvrent une seconde. Penser à relever ce compte avant d’ajouter un flux, pas après avoir cherché pourquoi il est rejeté.

`<queue-size>524288</queue-size>` — la file d’attente par auditeur, en octets. Un auditeur trop lent prend du retard ; quand il dépasse cette file, Icecast le déconnecte plutôt que de laisser la mémoire enfler.

`<burst-on-connect>1</burst-on-connect>` et `<burst-size>65535</burst-size>` — à la connexion, Icecast envoie d’un coup les 64 kio déjà en mémoire. Le lecteur remplit son propre tampon immédiatement au lieu d’attendre le temps réel : c’est ce qui fait démarrer un flux en une seconde plutôt qu’en cinq. Le prix est une latence permanente : 64 kio à 192 kb/s représentent environ 2,7 secondes de retard entre la source et l’auditeur. Sans importance pour une radio ; gênant pour qui voudrait synchroniser le son avec autre chose.

Les trois `timeout` — `client` 30 s, `header` 15 s, `source` 10 s. Le dernier est le plus intéressant : si une source cesse d’envoyer pendant dix secondes, Icecast considère le montage abandonné et le ferme. C’est ce qui fait qu’un flux dont la source est morte disparaît proprement de la liste, au lieu de rester affiché

en silence.

`<listen-socket>` sur le port 8000 — le seul port ouvert. Les blocs 8080 et 8443 du fichier Debian sont laissés en commentaire.

🔍 Vérifier ce qu'Icecast écoute vraiment plutôt que de le lire dans le fichier. Un `grep` ligne par ligne montre les ports commentés sans leurs `<!--` et fait croire à trois sockets ouvertes. La question se tranche depuis le système :

```
sudo ss -tlnp | grep icecast
```

Attendu : une seule ligne, `0.0.0.0:8000`.

`<http-headers>` avec `Access-Control-Allow-Origin: *` — réglage par défaut de Debian, qui autorise n'importe quelle page web à lire les flux et la page d'état en JSON. Pratique pour intégrer un lecteur dans une page ; à retirer si le serveur devait être exposé au-delà du réseau local.

`<alias source="/" destination="/status.xsl"/>` — c'est ce qui fait que la racine du serveur affiche la page d'état avec la liste des montages, plutôt qu'une erreur.

`<loglevel>3</loglevel>` — niveau *info*. Le 4 est le débogage : il fait enfler les journaux sans rien apprendre en régime normal.

`<chroot>0</chroot>` — Icecast ne s'enferme pas dans une racine restreinte. C'est le réglage de Debian, qui laisse le confinement au service systemd.

⚠️ Un piège de lecture, rencontré sur cette machine. Le fichier contient `<shoutcast-mount>/webadmin</shoutcast-mount>` — une valeur manifestement choisie à la main, là où Debian écrit `/stream`. Elle ne sert à rien : la ligne est à l'intérieur d'un bloc en commentaire. C'est l'exemple qui a été personnalisé, pas la configuration — et rien ne le signale, puisqu'un réglage inactif ne produit ni erreur ni effet.

C’est le risque propre à un fichier dont l’essentiel est commenté. La parade est mécanique : produire de temps en temps la vue « configuration active » ci-dessus, où seul apparaît ce qui agit réellement.

```
sudo grep -v -e '^\s*<!--' -e '^\s*$' /etc/icecast2/icecast.xml
```

⚠ Ce filtre est grossier et le montre : il retire la ligne qui ouvre un commentaire mais laisse celles du milieu et le `-->` de fin. Les blocs commentés restent donc visibles — repérables à leur `-->` orphelin. C’est suffisant pour lire, insuffisant pour conclure : ce qui tranche vraiment reste `ss` pour les ports, et le comportement du serveur pour le reste.

Les permissions

```
ls -l /etc/icecast2/icecast.xml
```

`-rw-r----- root icecast` — le fichier porte les quatre mots de passe, et seul le compte du service peut le lire. C’est le réglage posé par le paquet ; ne pas l’élargir.

⚠ Conséquence pratique : toute lecture du fichier demande `sudo`, y compris un simple `grep`. Un « permission refusée » sur ce fichier n’est pas un défaut à corriger.

Le service — la panne à ne pas reproduire

Installer Icecast ne l’active pas au démarrage. C’est le défaut qui a fait échouer cette chaîne toutes les nuits pendant des mois sans que rien ne le signale : au redémarrage, les sources se lançaient, ne trouvaient pas de serveur, et abandonnaient.

```
sudo systemctl enable --now icecast2
systemctl is-enabled icecast2
```

⚠️ Les `reconnectattempts` d'`ices2` n'auraient rien changé. Ils ne régissent que les *re*connexions — après qu'une première ait réussi. Sur un échec initial, `ices2` abandonne immédiatement, sans réessayer. Seul l'ordre de démarrage compte, et c'est précisément ce que cron ne sait pas exprimer : une tâche cron se lance à une heure, pas après un service. C'est l'argument décisif en faveur d'unités systemd pour les sources, développé plus bas.

Le journal du serveur :

```
sudo journalctl -u icecast2 -n 30 --no-pager
```

Le client `ices2`

Ce qu'il sait faire

```
sudo apt install ices2
```

`ices2` lit un fichier XML qui décrit d'où vient le son, comment l'encoder et où le pousser. Il connaît deux entrées :

Module	Entrée	Usage
`alsa`	une carte son du système	Tout ce qui se branche sur une entrée ligne ou micro — l'exemple de cette page
`stdinpcm`	l'entrée standard, en PCM brut	Un autre programme produit le son et l'envoie dans un tuyau — le cas du tuner radio

🎧 La diffusion est en Ogg Vorbis, et ce n'est pas un choix. `ices2` n'encode qu'en Vorbis. Le MP3 relève d'`ices0`, une version antérieure et distincte. Le Vorbis n'est pas un handicap : il est lu par VLC, par les lecteurs en réseau, par les navigateurs, et il est de meilleure qualité que le MP3 à débit égal.

Anatomie d’un fichier de configuration

Tous les fichiers `ices2` de la machine suivent la même charpente. La connaître dispense de relire chacun.

```
<?xml version="1.0"?>
<ices>
    <background>0</background>          <!-- rester au premier plan -->
    <loglevel>3</loglevel>
    <consolelog>1</consolelog>          <!-- écrire aussi sur la sortie -->
    <pidfile>…</pidfile>

    <stream>
        <metadata>…</metadata>          <!-- titre, artiste, genre affichés -->

        <input>
            <module>alsa|stdinpcm</module>
            <param name="…">…</param>   <!-- carte, fréquence, canaux -->
        </input>

        <instance>
            <hostname>localhost</hostname>
            <port>8000</port>
            <password>MASQUE</password> <!-- le source-password d'Icecast -->
            <mount>/Nom.ogg</mount>
            <reconnectdelay>2</reconnectdelay>
            <reconnectattempts>5</reconnectattempts>
            <maxqueuelength>80</maxqueuelength>
            <encode>…</encode>          <!-- débit, fréquence, canaux -->
        </instance>
    </stream>
</ices>
```

Trois réglages décident du bon fonctionnement sous systemd, et ce sont exactement les trois qu’il faut corriger sur un fichier hérité d’un lancement par script :

Réglage	Valeur	Pourquoi
`<background>`	`0`	Indispensable. À `1`, `ices2` se détache et rend la main ; systemd croit le service terminé et le relance en boucle.
`<loglevel>`	`3`	Le niveau 4 est le débogage : il noie le journal système sous des messages par seconde, sans rien apprendre en régime normal.
`<consolelog>`	`1`	La sortie part vers journald, dont la rotation est déjà réglée — plutôt que vers un fichier qui grossit sans surveillance.

□□ `<pidfile>` ne sert plus à rien sous systemd, qui suit ses processus lui-même. Le laisser est sans conséquence ; s'en servir pour arrêter le service, en revanche, est le défaut d'origine de cette installation — des scripts relisaient un fichier de PID absent et lançaient des `kill` sans argument.

⚠□ `<logpath>` et `<logfile>` n'ont pas leur place ici, et l'explication vaut d'être connue. `<consolelog>1</consolelog>` ne *double* pas le journal : il remplace l'écriture dans le fichier par l'écriture sur la sortie standard. Déclarer un fichier de journal en plus ne produit donc rien du tout — le fichier n'est jamais écrit.

C'est une combinaison qui trompe à la lecture : on voit un `<logfile>`, on croit qu'un journal s'accumule quelque part, et on va le chercher. Sur cette machine, un `ices.log` de 2 Mo traînait ainsi, figé depuis le jour du passage à systemd — c'était l'héritage de l'époque où `ices2` était lancé par un script, sans `consolelog`.

Sous systemd, les deux lignes sont donc retirées : le journal va à `journalctl`, séparé par service et déjà soumis à rotation.

□□ Le `<pidfile>` reste, lui — il ne trompe personne et ne coûte rien. Mais il ne sert plus non plus : c'est systemd qui suit ses processus.

`<hostname>localhost</hostname>` dans le fichier de la source, et non l'adresse du réseau. La source tourne sur la même machine que le serveur ; la connexion ne sort pas de la machine, et le mot de passe qu'elle transporte non plus.

⚠ Ces fichiers contiennent le `source-password` en clair. Ils doivent donc être lisibles par le compte du service, et par personne d'autre :

```
sudo chown root:hostadmin /etc/ices2_*.xml
```

```
sudo chmod 640 /etc/ices2_*.xml
```

```
ls -l /etc/ices2_*.xml
```

Attendu : `-rw-r----- root hostadmin`. Le service tourne sous `hostadmin`, qui garde l'accès par le groupe ; les autres comptes le perdent.

 C'est un geste à faire explicitement, parce que rien ne le fait pour vous. Le paquet ne crée pas ces fichiers — on les écrit soi-même, et l'umask les laisse en `644`, c'est-à-dire lisibles par tout compte de la machine. L'asymétrie avec `icecast.xml`, que le paquet installe en `640 root:icecast`, est instructive : le serveur protège ses mots de passe, ses clients les exposent — non par choix, mais par défaut d'attention.

Les métadonnées

Ce sont les informations que les lecteurs affichent : titre, artiste, genre. Le mécanisme est le même pour toutes les sources ; il n'est expliqué qu'ici, et la page consacrée à la radio y renvoie.

Il y a deux endroits distincts, qu'on confond aisément.

Le bloc `<metadata>` du XML décrit le flux lui-même, et ne change pas :

```
<metadata>
    <name>Diffusion Chromecast</name>
    <genre>streaming</genre>
    <artist>Service de diffusion en continu</artist>
    <description>Service de diffusion en continu</description>
```

```
</metadata>
```

Un fichier texte séparé porte ce qui change en cours de diffusion — le morceau en train de jouer. Une ligne par champ :

```
title=98.5 FM
artist=Animateurs en direct
album=Voir l'horaire du diffuseur
genre=Actualités
url=https://www.985fm.ca/
```

□□ Cet exemple est celui de la source radio, traitée sur l’autre page — d’où des valeurs qui décrivent une station.

Le fichier est désigné dans le bloc `<input>` :

```
<param name="metadatafilename">/home/hostadmin/ices2/metadata_98_5</param>
```

Le rechargement se fait par signal, sans couper la diffusion :

```
sudo pkill -USR1 -f ices2_98_5_config.xml
```

□□ `SIGUSR1` relit le fichier de métadonnées et pousse les nouvelles valeurs aux auditeurs connectés. Le flux ne s’interrompt pas — c’est ce qui permettrait d’afficher le titre en cours sur une diffusion qui dure des heures. Pour une station de radio, dont le programme n’est pas connu de la machine, les métadonnées restent fixes et le signal ne sert pas ; il vaut d’être connu pour une source dont on connaît le contenu.

⚠□ Ne pas confondre avec un redémarrage du service. `systemctl restart` coupe le flux, ce qui déconnecte les auditeurs — et, sur le montage de repli, laisse une fenêtre pendant laquelle l’alarme n’aurait rien à jouer.

Une source complète : la carte son

C’est l’exemple de référence de cette page. Le montage : un Chromecast Audio branché sur l’entrée d’un dongle audio USB, dont la sortie est diffusée. Résultat pratique — ce que joue un téléphone Android arrive dans toutes les pièces de la maison, par le serveur de musique.

Le matériel

Un dongle USB à puce Texas Instruments PCM2902 — le composant le plus répandu de sa catégorie, vendu sous des dizaines de marques. Il présente une entrée et une sortie analogiques.

```
lsusb | grep -i audio
arecord -l
```

Attendu :

```
Bus 001 Device 004: ID 08bb:2902 Texas Instruments PCM2902 Audio Codec
card 2: CODEC [USB Audio CODEC], device 0: USB Audio [USB Audio]
```

`CODEC` est le nom ALSA de la carte, et c’est lui qu’on écrit dans la configuration — pas son numéro. Le numéro change d’un démarrage à l’autre selon l’ordre d’énumération USB ; le nom, non. `hw:CARD=CODEC,DEV=0` est donc stable là où `hw:2,0` ne l’est pas.

⚠ Ce dongle n’a aucun réglage de capture. `amixer -c CODEC` ne montre qu’un contrôle `PCM` de lecture. Le gain d’entrée du PCM2902 est figé dans le matériel : il n’y a rien à démuter ni à monter, et cette piste est à écarter d’emblée en cas de silence.

⚠ La carte son dépend du paquet `linux-modules-extra` du noyau en cours. `snd-usb-audio` n’est pas dans le paquet de base. Sur cette machine, un noyau épinglé

dont le `modules-extra` avait été retiré par `autoremove` a fait disparaître le son entièrement — `/proc/asound/` inexistant, `arecord -l` vide — alors que le dongle était bien branché. Si `arecord -l` ne trouve aucune carte, vérifier ce paquet avant de soupçonner le matériel :

```
dpkg -l | grep linux-modules-extra-$(uname -r)
```

Le fichier de configuration

`/etc/ices2_chromecast_config.xml` :

```
<?xml version="1.0"?>
<ices>
    <background>0</background>
    <loglevel>3</loglevel>
    <consolelog>1</consolelog>
    <pidfile>/home/hostadmin/ices2/ices_chromecast.pid</pidfile>

    <stream>
        <metadata>
            <name>Diffusion Chromecast</name>
            <genre>streaming</genre>
            <artist>Service de diffusion en continu</artist>
            <description>Service de diffusion en continu</description>
        </metadata>

        <input>
            <module>alsa</module>
            <param name="rate">44100</param>
            <param name="channels">2</param>
            <param name="device">hw:CARD=CODEC,DEV=0</param>
            <param name="metadata">1</param>
            <param
name="metadatafilename">/home/hostadmin/ices2/metadata_chromecast</param>
        </input>

        <instance>
            <hostname>localhost</hostname>
            <port>8000</port>
```

```
            <password>MASQUE</password>
            <mount>/Chromecast.ogg</mount>
            <reconnectdelay>2</reconnectdelay>
            <reconnectattempts>5</reconnectattempts>
            <maxqueuelength>80</maxqueuelength>

            <encode>
                <nominal-bitrate>192000</nominal-bitrate>
                <samplerate>44100</samplerate>
                <channels>2</channels>
            </encode>
        </instance>
    </stream>
</ices>
```

44 100 Hz et deux canaux — la fréquence du disque compact, en stéréo, parce que la source est de la musique.

⚠ Les fréquences et le nombre de canaux doivent être identiques à l'entrée et à l'encodeur. Ce n'est pas une convention de propreté : `ices2` ne sait pas rééchantillonner, et son propre fichier d'exemple le dit — la restriction devait être levée « dans le futur », ce qui n'est pas arrivé. Une discordance donne un son trop lent ou trop rapide, sans le moindre message d'erreur.

`<param name="metadata">1</param>` — active la lecture du fichier de métadonnées décrit plus haut. Sans cette ligne, le `metadatafilename` est ignoré en silence.

`<maxqueuelength>80</maxqueuelength>` — la file interne d'`ices2`, en paquets. Quand le serveur n'absorbe plus assez vite, `ices2` vide la file et continue plutôt que de prendre un retard croissant : on perd une fraction de seconde de son, on ne dérive pas.

Le service

`/etc/systemd/system/ices2-chromecast.service` :

```
[Unit]
Description=Diffusion Chromecast vers Icecast
```

```
Requires=icecast2.service
After=icecast2.service sound.target
StartLimitIntervalSec=0

[Service]
Type=simple
User=hostadmin
Group=hostadmin
SupplementaryGroups=audio
ExecStart=/usr/bin/ices2 /etc/ices2_chromecast_config.xml
Restart=always
RestartSec=10

[Install]
WantedBy=multi-user.target
```

Ligne par ligne, ce qui n’est pas décoratif :

⚠ `Requires=` et `After=icecast2.service` — les deux, et pas l’un des deux. `Requires=` dit que le serveur doit tourner ; `After=` dit qu’il doit tourner d’abord. Sans `After=`, systemd est libre de les lancer en parallèle, et la source retombe sur l’échec de première connexion qui ne se rattrape pas.

⚠ `After=… sound.target` — attendre en plus que le sous-système audio soit prêt. Une source ALSA lancée avant que sa carte n’existe échoue sur un périphérique introuvable.

⚠ `SupplementaryGroups=audio` est indispensable. Les périphériques de `/dev/snd/` appartiennent au groupe `audio`, dont le compte de service ne fait pas partie. Systemd lui accorde cette appartenance le temps du service seulement, sans toucher au compte lui-même — ce qui est préférable à un `usermod` qui la donnerait à toutes ses sessions.

`StartLimitIntervalSec=0` — retirer le garde-fou qui, après cinq échecs rapprochés, refuse de relancer. Pour un service dont on veut qu’il retente indéfiniment, ce garde-fou est nuisible : il abandonne exactement dans le cas qu’on cherche à couvrir, une panne prolongée du serveur.

`Restart=always` et `RestartSec=10` — relancer quoi qu'il arrive, dix secondes plus tard.

`Type=simple` — cohérent avec `<background>0</background>` du XML : systemd suit le processus qu'il a lancé.

```
sudo systemctl daemon-reload
sudo systemctl enable --now ices2-chromecast
systemctl status ices2-chromecast --no-pager
```

Attendu dans le journal : `Connected to server: localhost:8000/Chromecast.ogg`.

Vérifier qu'un flux est bien servi

Depuis le serveur

```
curl -s --max-time 4 -o /tmp/essai.ogg http://192.168.0.11:8000/Chromecast.ogg ;
ls -l /tmp/essai.ogg
```

Attendu : un fichier de l'ordre de 100 à 200 kio pour quatre secondes à 192 kb/s.

⚠ `curl -I` ne teste pas un flux Icecast. La requête HEAD reçoit un `400 Bad Request` quel que soit l'état du serveur. Il faut un vrai GET, borné dans le temps.

 La taille prouve que la chaîne tourne, pas qu'il y a du son. `ices2` encode le silence exactement comme la musique, à peu près au même débit. Un fichier de bonne taille dit que la capture et l'encodage fonctionnent — pour savoir s'il porte du son, il faut l'écouter.

Depuis un poste du réseau

Ouvrir `http://192.168.0.11:8000/Chromecast.ogg` dans VLC, ou

`http://192.168.0.11:8000/` dans un navigateur — cette page liste les points de montage actifs avec leur nombre d'auditeurs.

Depuis l'interface d'administration

`http://192.168.0.11:8000/admin/`, avec `<admin-user>` et son mot de passe. On y voit chaque montage, sa source connectée, ses auditeurs, et on peut y couper une source récalcitrante sans toucher au serveur.

Depuis un serveur de musique

Un flux Icecast s'ajoute comme n'importe quelle radio web : l'URL du point de montage suffit, il n'y a ni greffon ni format particulier à prévoir. C'est ce qui rend le montage intéressant — le serveur de musique n'a pas à savoir d'où vient le son.



Capture d'époque, conservée pour le principe. Elle date de la première version de ce montage ; l'interface a pu changer depuis, et ce n'est pas la façon dont les flux sont écoutés ici aujourd'hui. Ce qu'elle montre reste vrai : une URL, et rien d'autre.

Lire les journaux

Tout passe par `journalctl`, et par rien d'autre : `<consolelog>1</consolelog>` envoie la sortie d'`ices2` à systemd, qui la range par service, la date, et la fait tourner sans qu'on s'en occupe.

Les dernières lignes d'un service :

```
journalctl -u ices2-chromecast -n 30 --no-pager
```

Suivre en direct, pendant qu’on manipule autre chose :

```
journalctl -u ices2-chromecast -f
```

Depuis une date, ou une période :

```
journalctl -u ices2-chromecast --since "aujourd'hui"
```

```
journalctl -u ices2-chromecast --since "2026-09-04 04:00" --until "2026-09-04
05:00"
```

Les erreurs seulement, tous services confondus, quand on ne sait pas encore où regarder :

```
journalctl -p err --since "il y a 2 heures" --no-pager
```

Les trois services d’un coup, pour voir l’ordre dans lequel ils se sont lancés — c’est ainsi qu’on constate un démarrage dans le mauvais ordre :

```
journalctl -u icecast2 -u ices2-chromecast -u rtl-fm-98-5 --since "il y a 15 min"
--no-pager
```

À quoi ressemble un démarrage sain

C’est la référence la plus utile de cette page : la connaître permet de repérer d’un coup d’œil ce qui manque.

```
Started ices2-chromecast.service - Diffusion Chromecast vers Icecast.
INFO ices-core/main IceS 2.0.3 started...
INFO input-alsa/alsa_open_module Opened audio device hw:CARD=CODEC,DEV=0
```

```
INFO input-alsa/alsa_open_module using 2 channel(s), 44100 Hz, buffer 500 ms
INFO input-alsa/alsa_open_module Starting metadata update thread
INFO metadata/metadata_thread_signal Updating metadata
INFO encode/encode_initialise Encoder initialising in VBR mode: 2 channels, 44100
Hz, nominal 192000
INFO stream/ices_instance_stream Connected to server:
localhost:8000/Chromecast.ogg
```

Quatre étapes s’y lisent dans l’ordre, et chacune peut être la dernière :

Ligne	Ce qu’elle prouve	Si elle manque
`Opened audio device`	La carte son existe et s’ouvre	Périphérique absent, occupé, ou droits manquants
`using 2 channel(s), 44100 Hz`	Le matériel accepte le format demandé	Discordance entre le XML et ce que la carte sait faire
`Encoder initialising`	L’encodage Vorbis démarre	Fréquence ou canaux incohérents entre entrée et `<encode>`
`Connected to server`	Le serveur a accepté la source	Icecast absent, mot de passe refusé, ou montage déjà pris

⚠ `Failed initial connect to localhost:8000` est le message à reconnaître entre tous. Il veut dire qu’`ices2` a démarré avant Icecast — et qu’il n’a pas réessayé. C’est la panne décrite plus haut, et sa cause est toujours la même : une dépendance de démarrage absente ou un lancement par cron.

Sur une source radio, une ligne s’ajoute en tête : `Tuned to …`, écrite par `rtl_fm` et non par `ices2` — les deux programmes partagent le même journal puisqu’ils sont dans le même service.

Les journaux d’Icecast, qui eux sont des fichiers

Le serveur, lui, écrit vraiment dans des fichiers, déclarés dans sa section `<logging>` :

```
sudo ls -lh /var/log/icecast2/
```

```
sudo tail -n 30 /var/log/icecast2/error.log
```

`access.log` enregistre chaque auditeur qui se connecte ; `error.log`, tout le reste. 🗒️ C’est `access.log` qui tranche la question « le lecteur essaie-t-il seulement ? » — si une tentative de lecture n’y figure pas, le problème est du côté du lecteur, pas du serveur. Ce raisonnement a déjà servi ici : voir la panne transitoire décrite plus bas.

Le service lui-même reste par ailleurs visible dans `journalctl` :

```
journalctl -u icecast2 -n 30 --no-pager
```

Dépannage — « le flux tourne mais je n’entends rien »

C’est le symptôme le plus fréquent, et le plus trompeur : tout paraît sain à chaque étape, parce que rien ne distingue le silence du son dans une chaîne numérique. La chaîne complète compte six maillons :

> *application du téléphone → Chromecast → sortie analogique → dongle de capture → `ices2` → Icecast → serveur de musique → lecteur*

L’ordre de diagnostic, du moins coûteux au plus coûteux.

1. Essayer une autre application source. Gratuit, instantané, et c’est le maillon qu’on oublie parce qu’il est hors de la machine.

⚠️ Cas réel : la cause était l’application. Amazon Music sur Android affichait une diffusion en cours sans rien émettre vers le Chromecast. VLC sur le même téléphone a fonctionné immédiatement. Avant de soupçonner le matériel, changer d’application.

2. Couper le problème en deux à Icecast. Faire jouer l'autre flux — celui de la radio — sur le même lecteur. Ce flux porte du son certain.

- L'autre flux joue, celui-ci non → le problème est en amont d'Icecast.

- Ni l'un ni l'autre → le problème est en aval : serveur de musique ou lecteur.

3. Écouter le flux directement depuis un poste, sans passer par le serveur de musique : `http://192.168.0.11:8000/Chromecast.ogg` dans VLC. Silence ici = la capture ne reçoit rien.

4. Vérifier le mélangeur — `amixer -c CODEC`, en lecture seule, sans déranger `ices2`. Sur un PCM2902 il n'y a aucun contrôle de capture : piste à écarter d'emblée (voir plus haut).

5. Mesurer le signal à l'entrée avec un vumètre. L'accès `hw:` étant exclusif, il faut d'abord libérer la carte :

```
sudo systemctl stop ices2-chromecast
```

```
arecord -D hw:CARD=CODEC,DEV=0 -f cd -d 15 -V stereo /dev/null
```

```
sudo systemctl start ices2-chromecast
```

Des barres plates pendant quinze secondes : aucun signal n'entre. Le problème est physique — câble, sortie du Chromecast, ou dongle.

Autres cas rencontrés

⚠ Un lecteur qui refuse un flux parfait, après un redémarrage du serveur. Le

flux répondait normalement — 171 kio en quatre secondes, les deux montages déclarés avec leurs sources — mais un lecteur refusait de le jouer, et aucune tentative de lecture n'apparaissait dans le journal d'Icecast : le lecteur n'essayait même pas. Rétabli seul. Circonstance particulière : la machine avait redémarré deux minutes plus tôt. Hypothèse la plus sobre — connexion morte ou échec mis en cache côté lecteur — mais rien ne la confirme.

Règle pratique qui en découle : ne pas redémarrer le serveur dans la fenêtre où l'on compte sur le repli d'alarme, et si ça arrive, vérifier que le flux joue avant de compter dessus.

⚠ Redescendre les niveaux de journalisation après un diagnostic. Côté serveur de musique, un niveau `DEBUG` sur la diffusion écrit une ligne par seconde et par lecteur — 86 400 lignes par jour. Utile une heure, ruineux ensuite.

L'autre source : la radio FM

Le second point de montage, `/Radio_FM.ogg`, est alimenté par un tuner radio logiciel — une clé USB RTL-SDR et `rtl_fm`, dont la sortie entre dans `ices2` par un tuyau plutôt que par une carte son. Le principe est celui de cette page ; les différences — le module `stdinpcm`, la mono à 48 kHz, l'exclusion mutuelle entre stations qui partagent un unique récepteur — sont traitées dans la page *Nooelec NESDR SMART*.