MusicIP — Analyse acoustique et listes de lecture pour Lyrion

Référence : MusicIP Mixer 1.8 (32 bits) — Ubuntu Server 24.04 — Lyrion Music Server



Les noms d'hôtes, de domaines et de comptes de cette page sont des exemples à transposer. Les adresses `192.168.0.x` appartiennent à l'espace privé RFC 1918 et sont conservées telles quelles : elles illustrent la topologie sans identifier personne.

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

⚠ Une ligne de code de ce guide dépasse la largeur du PDF — la ligne `Documentation=` de l'unité systemd, qu'une URL rend insécable. Elle s'enroule dans le PDF sans aucun signe visible : la recopier depuis le `.md`, jamais depuis le PDF.

Ce que cette page couvre : faire tourner MusicIP en service sur un serveur sans écran, y faire entrer une bibliothèque musicale, et le piloter depuis un script — ce dernier point étant celui que la documentation d'origine ne couvre pas du tout.

Contexte : MusicIP écoute les pièces musicales, en calcule une empreinte acoustique (*fingerprint*), et s'en sert pour fabriquer des listes de lecture à partir d'un morceau choisi comme graine. Le greffon SugarCube de Lyrion Music Server l'interroge pour alimenter une file de lecture qui se prolonge toute seule.

Le logiciel n’est plus maintenu depuis 2008 : c’est un binaire 32 bits que personne ne corrigera. Il fonctionne très bien, à condition de savoir où sont ses angles morts — et il en a plusieurs qui échouent en silence.

Trois choses valent le détour même sans suivre ce guide de bout en bout.

La section *Une empreinte, deux endroits où elle peut vivre* explique pourquoi la même analyse peut être perdue ou survivre selon la manière dont on l’a obtenue. C’est la décision la plus structurante de tout le montage.

La section *Piloter le serveur depuis un script* documente une API que personne n’a écrite, avec les points d’entrée qui répondent `HTTP 302 OK` sans rien faire.

La section *Faire tenir le travail sur le disque* dit pourquoi une campagne d’analyse de treize heures peut disparaître à l’extinction suivante, et le geste d’une minute qui l’empêche.

> *Convention : chaque bloc indique où il s’exécute — `[nas-host]` pour le serveur, `[poste-bureau]` pour le poste de travail, `[poste de travail]` pour l’instance Traitement.*

---

1. Deux logiciels, deux rôles, deux besoins

Tout le reste découle de cette distinction, et la confondre coûte cher.

`MusicMagicServer` est le service. Il tient la base des empreintes, répond aux requêtes de Lyrion et expose une petite interface web. C’est un exécutable 32 bits natif qui ne dépend que de la bibliothèque C du système :

```
linux-gate.so.1
libpthread.so.0
libc.so.6
```

Il n’a besoin d’aucune machine Java. La liste ci-dessus est exhaustive : le service ne charge aucune JVM.

`MusicMagicMixer` est l’interface graphique. C’est un lanceur 32 bits qui exige la variable `JAVA_HOME`, refuse de démarrer sans elle, et exécute `$JAVA_HOME/bin/java` sur le fichier `mm.jar`. Ce Java charge à son tour `lib/MusicMagic.so`, une bibliothèque native 32 bits.

⚠ D’où la seule contrainte Java du montage : le Mixer exige un JRE 32 bits. Un JRE 64 bits démarre, puis échoue à charger la bibliothèque native. Rien d’autre sur la machine n’a besoin de Java pour MusicIP.

Ce que chacun sait faire, et que l’autre ne sait pas :

- seul le Mixer écrit l’empreinte dans les métadonnées du fichier audio ;
- seul le Serveur répond à Lyrion et à l’API HTTP ;
- seul le Mixer possède une commande d’enregistrement de la base sur le disque.

---

2. Une empreinte, deux endroits où elle peut vivre

Une empreinte calculée peut être rangée à deux endroits, et le choix décide de ce qui survit aux années.

Voie persistante — l’empreinte dans le fichier

Le Mixer écrit deux étiquettes dans le fichier audio lui-même : `MusicMagic Fingerprint` et `MusicMagic Data` (étiquettes `TXXX` pour MP3, étiquettes Vorbis équivalentes pour FLAC). L’option qui l’active est *Archive analysis when tracks*

are analyzed.

L’empreinte suit alors le fichier partout : elle survit à l’effacement de la base, aux renommages, aux déplacements, à une migration de serveur. Le jour où la base est reconstruite de zéro, le serveur relit les étiquettes et n’analyse rien — ce qui transforme des semaines en quelques heures.

Voie volatile — l’empreinte dans la base seulement

Une soumission directe au serveur par HTTP fait analyser le fichier sans jamais y toucher. L’empreinte n’existe alors que dans `default.m3lib`.

C’est plus rapide et ça ne demande aucune interface graphique. Mais l’empreinte devient orpheline dès que le fichier est renommé ou déplacé, et elle disparaît entièrement si la base est perdue.

 L’analyse complète d’une grande bibliothèque se compte en semaines de calcul. C’est le seul travail de ce montage qu’on ne peut pas refaire à volonté. La voie persistante est donc le choix par défaut, et la voie volatile un raccourci qu’on prend en connaissance de cause, avec une sauvegarde de la base.

Ce que la voie volatile ne peut pas faire

Le calcul d’empreinte n’accepte que du 16 bits / 44,1 kHz. Un fichier en 24/48, 24/88, 24/96 ou 24/192 soumis directement au serveur est écarté sans message : rien dans la réponse HTTP ne l’indique, le compteur n’augmente simplement pas.

En revanche, un fichier haute résolution qui porte déjà les étiquettes d’empreinte est accepté tel quel — le serveur les lit sans jamais ouvrir l’audio, et le format cesse d’être un obstacle. C’est ce qui rend la voie persistante obligatoire pour la haute résolution, et c’est l’objet de l’architecture décrite ensuite.

Choisir

Critère	Voie persistante (Mixer)	Voie volatile (HTTP direct)

Critère	Voie persistante (Mixer)	Voie volatile (HTTP direct)
Où vit l’empreinte	Étiquettes du fichier audio	`default.m3lib` seulement
Après un effacement de la base	Relue depuis les fichiers	Perdue — tout est à réanalyser
Après un renommage ou un déplacement	Relue depuis le fichier, sans réanalyse	Perdue — tout est à réanalyser
Formats couverts	MP3, FLAC 16/44, et haute résolution par rééchantillonnage	MP3 et FLAC 16/44 seulement
Instance nécessaire	Traitement, donc un Mixer en état de marche	Production seule
Vitesse	Lente — analyse par le Mixer	Rapide — une requête par dossier
Quand elle convient	Par défaut, et pour toute bibliothèque qui bouge	Bibliothèque stable, base sauvegardée

Un point que le tableau ne distingue pas, parce qu’il vaut pour les deux : un fichier renommé ou déplacé laisse dans la base une entrée qui ne désigne plus rien, et c’est `Refresh Songs` qui la retire. La différence est ce qui se passe ensuite — la voie persistante relit l’étiquette et ne recalcule rien, la voie volatile repart de l’audio.

⚠ Le Mixer est un logiciel 32 bits que personne ne maintient. Le jour où il cesse de fonctionner, la voie volatile devient la seule disponible — et seulement pour les formats natifs. C’est une raison de plus de préférer la voie persistante tant qu’elle est là.

---

3. Deux instances, deux rôles

Une seule instance ne suffit pas dès que la bibliothèque mélange des formats natifs et de la haute résolution.

L’instance Production vit sur `[nas-host]`, écoute sur `http://192.168.0.11:10002`, et sert Lyrion. Sa base `default.m3lib` accumule des années d’analyse : c’est le

fichier le plus irremplaçable du montage.

L'instance Traitement vit sur un poste de travail, écoute sur `http://localhost:10002`, et ne sert qu'à pré-étiqueter des fichiers avant leur entrée dans la bibliothèque. On la remplit, on l'exploite, on la vide. Sa base n'a aucune valeur.

⚠ Ne pas confondre l'instance Traitement avec le Mixer de Production. Le Mixer de Production s'exécute sur `[nas-host]` et sert à administrer le serveur de Production. Le pré-étiquetage, lui, doit passer par l'instance Traitement : le Mixer inscrit dans les étiquettes le chemin absolu tel qu'il le voit depuis la machine qui l'exécute, et un pré-étiquetage fait au mauvais endroit produit des chemins incohérents avec la bibliothèque de destination.

---

4. Installer l'instance Production

L'architecture 32 bits

Le logiciel est en 32 bits ; un système 64 bits doit apprendre à en servir les paquets.

Sur `[nas-host]` :

```
sudo dpkg --add-architecture i386
sudo apt update
```

Les fichiers du logiciel

Deux dossiers à restaurer depuis la sauvegarde, ou à tirer du dépôt github.com/CDrummond/musicip, qui miroite les binaires Linux :

- `.MusicMagic` va dans `/home/hostadmin` — il contient la configuration et la base ;

- `MusicIP` va dans `/home/hostadmin/logiciels` — il contient les exécutables.

 Le fichier `register.key` est la clé de licence. Il existe dans les deux dossiers et ne se retrouve nulle part ailleurs : le fournisseur a disparu. Sa présence dans la sauvegarde conditionne toute reconstruction — son contenu ne s'écrit dans aucun document.

Le JRE 32 bits, pour le Mixer

Un seul paquet suffit, et c'est le Mixer qui en a besoin :

```
sudo apt install openjdk-11-jre:i386
```

 Les variantes `-jdk` sont des outils de développement et rien ici ne compile de Java. La variante 64 bits ne sert pas davantage : c'est la machine virtuelle 32 bits qui doit ouvrir `lib/MusicMagic.so`.

Le Mixer lit `JAVA_HOME` dans son environnement. Comme il est toujours lancé depuis un interpréteur de commandes interactif, la variable se pose là où cet interpréteur la lit.

Sur `[nas-host]` :

```
sudo nano /etc/bash.bashrc
```

Ajouter à la fin du fichier ouvert ci-dessus :

```
JAVA_HOME=/usr/lib/jvm/java-11-openjdk-i386
export JAVA_HOME
PATH=$PATH:$JAVA_HOME/bin
export PATH
```

⚠ Cette variable n'atteint jamais le service. Une unité systemd ne lit pas

`/etc/bash.bashrc` — ce fichier n’est lu que par un interpréteur bash interactif. Ce n’est pas un défaut : le serveur n’a pas besoin de Java. Mais il ne faut pas compter sur ce fichier pour configurer quoi que ce soit du service.

La configuration du serveur

Sur `[nas-host]` :

```
nano /home/hostadmin/.MusicMagic/mmm.ini
```

Contenu à coller dans l’éditeur ouvert ci-dessus :

```
[mix]
mixsize=20 tracks
restrict=0 tracks
style=100
variety=6
seed=1
[server]
proxy=0
host=http://music.predixis.com
proxyport=80
authentication=0
user=
password=
[services]
key=
cache=/home/hostadmin/.MusicMagic/default.m3lib
port=10002
tivo=1
upnp=1
api=1
readonly=0
refresh=15
```

Les réglages de la section `[mix]` se modifient aussi depuis l’interface web, qui est plus commode : `variety` et `mixsize` y portent les mêmes noms.

 `host=http://music.predixis.com` désigne un service qui n’existe plus. Le serveur tente régulièrement de le joindre et le journal se remplit de `gethostbyname failed: music.predixis.com`. C’est sans conséquence sur le fonctionnement — mais il faut le savoir pour ne pas partir en chasse d’une panne imaginaire.

Le service systemd

Sur `[nas-host]` :

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

Contenu à coller dans l’éditeur ouvert ci-dessus :

```
[Unit]
Description=MusicIP MusicMagicServer — analyse acoustique et listes de lecture
Documentation=https://blog.infolaf.ca/wiki/musicip-analyse-acoustique-listes-de-l
ecture-lyrion/
After=network-online.target
Wants=network-online.target
RequiresMountsFor=/media/nas1

[Service]
Type=simple
User=hostadmin
Group=sambashare
WorkingDirectory=/home/hostadmin/logiciels/MusicIP/MusicMagicMixer
ExecStart=/home/hostadmin/logiciels/MusicIP/MusicMagicMixer/MusicMagicServer
start
ExecStop=/home/hostadmin/logiciels/MusicIP/MusicMagicMixer/MusicMagicServer stop
TimeoutStopSec=60
Restart=on-failure
RestartSec=10

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

Chaque ligne de ce bloc répare quelque chose qui casse en silence sans elle. Les quatre qui comptent le plus :

 `ExecStop=`. `MusicMagicServer` n’obéit pas à `SIGTERM`. Sans ordre d’arrêt explicite, systemd attend la totalité de son délai puis tue le processus au `SIGKILL` — à chaque extinction de la machine, qui traîne alors une minute et demie de plus. Pire, le processus tué peut survivre à l’unité qui le contenait, et un second serveur démarrer à côté du premier sur la même base. La commande d’arrêt est celle du script d’initialisation fourni par l’éditeur ; avec elle, l’arrêt prend trois secondes.

 Pas de `RemainAfterExit=yes`. Cette directive maintient l’unité en état `active` même lorsque le processus est mort. Le service ment alors sur son propre état, et toute surveillance bâtie sur `systemctl is-active` devient aveugle. `Restart=on-failure` prend le relais et relance réellement.

⚠ `WantedBy=multi-user.target`. La valeur `default.target` semble équivalente ; elle ne l’est pas. Elle désigne la cible par défaut de la machine, qui n’est pas toujours `multi-user.target` : elle peut être `graphical.target`. Le service se retrouve alors accroché à une cible graphique sur une machine sans écran, et tout démarrage qui ne l’atteint pas le laisse à l’arrêt. `systemctl get-default` dit laquelle est en vigueur.

⚠ `After=network-online.target`, et non `network.service`. Sur un système moderne, `network.service` n’existe pas : la directive ne désigne rien et ne retarde rien. Elle passe la vérification de syntaxe sans un mot, parce qu’ordonner un service par rapport à une unité absente est légal.

`RequiresMountsFor=/media/nas1` empêche le serveur de démarrer avant que la bibliothèque soit montée. `WorkingDirectory=` aligne le service sur la façon dont le logiciel est lancé à la main.

 Aucun script d’enveloppe n’est nécessaire. Un script qui se contente d’appeler le binaire nuit : c’est lui que systemd surveille, si bien que le `Main PID` affiché n’est pas celui du serveur.

Activer et contrôler

Sur `[nas-host]` :

```
sudo chmod 644 /etc/systemd/system/musicip_server.service
sudo systemd-analyze verify /etc/systemd/system/musicip_server.service
sudo systemctl daemon-reload
sudo systemctl enable --now musicip_server.service
```

`systemd-analyze verify` doit rester muet. Toute ligne qu'il imprime est un défaut.

Contrôler ensuite :

```
systemctl status musicip_server.service --no-pager
```

Le résultat attendu tient en une ligne du `CGroup` : le seul processus doit être `MusicMagicServer`, et le `Main PID` doit porter son nom.

⚠ Si l'unité a déjà existé sous `WantedBy=default.target`, un `enable` seul ne déplace pas le lien d'activation : l'ancien subsiste. Il faut `disable` puis `enable` pour que le lien migre vers `multi-user.target.wants/`.

 Au démarrage, le journal signale `File does not exist: /home/hostadmin/.MusicMagic/recipes.xml`. Ce fichier est optionnel et son absence est l'état normal.

---

5. Le Mixer, affiché à distance

Un serveur sans écran n'a pas d'environnement graphique. Le Mixer s'exécute sur `[nas-host]` ; c'est sa fenêtre qui voyage jusqu'au poste de travail, par le transfert X11 de SSH.

Une entrée de configuration plutôt qu'une commande à retenir

Sur `[poste-bureau]` :

```
nano ~/.ssh/config
```

Ajouter dans l’éditeur ouvert ci-dessus :

```
Host MusicIP
    HostName 192.168.0.11
    User hostadmin
    ForwardX11 yes
    RequestTTY yes
    RemoteCommand cd /home/hostadmin/scripts && ls && exec bash -l
```

La connexion devient alors :

```
ssh MusicIP
```

L’interpréteur ouvre directement dans le dossier des scripts, avec son contenu affiché : le lanceur du Mixer est sous les yeux, sans rien à se rappeler.

 `exec bash -l` et non `bash ; exec $SHELL -l`. La seconde forme, qu’on rencontre souvent, lance un interpréteur puis en lance un autre à sa sortie : il faut quitter deux fois. `exec` remplace le processus courant, et une seule sortie suffit.

Le lanceur du Mixer

Sur `[nas-host]` :

```
nano /home/hostadmin/scripts/start_musicip_mixer.sh
```

Contenu à coller dans l’éditeur ouvert ci-dessus :

```
#!/bin/bash
cd /home/hostadmin/logiciels/MusicIP/MusicMagicMixer
./MusicMagicMixer
```

Puis le rendre exécutable :

```
chmod 755 /home/hostadmin/scripts/start_musicip_mixer.sh
```

⚠ Le changement de dossier n'est pas une commodité. Le lanceur construit sa commande Java à partir du dossier courant — `java -classpath mm.jar -Dinstall.root=<dossier courant> …`. Appelé par son chemin absolu depuis ailleurs, il désigne une racine d'installation qui n'existe pas et ne trouve ni `mm.jar` ni `lib/MusicMagic.so`.

Le binaire s'appelle sans argument : aucun verbe n'est attendu, contrairement au serveur.

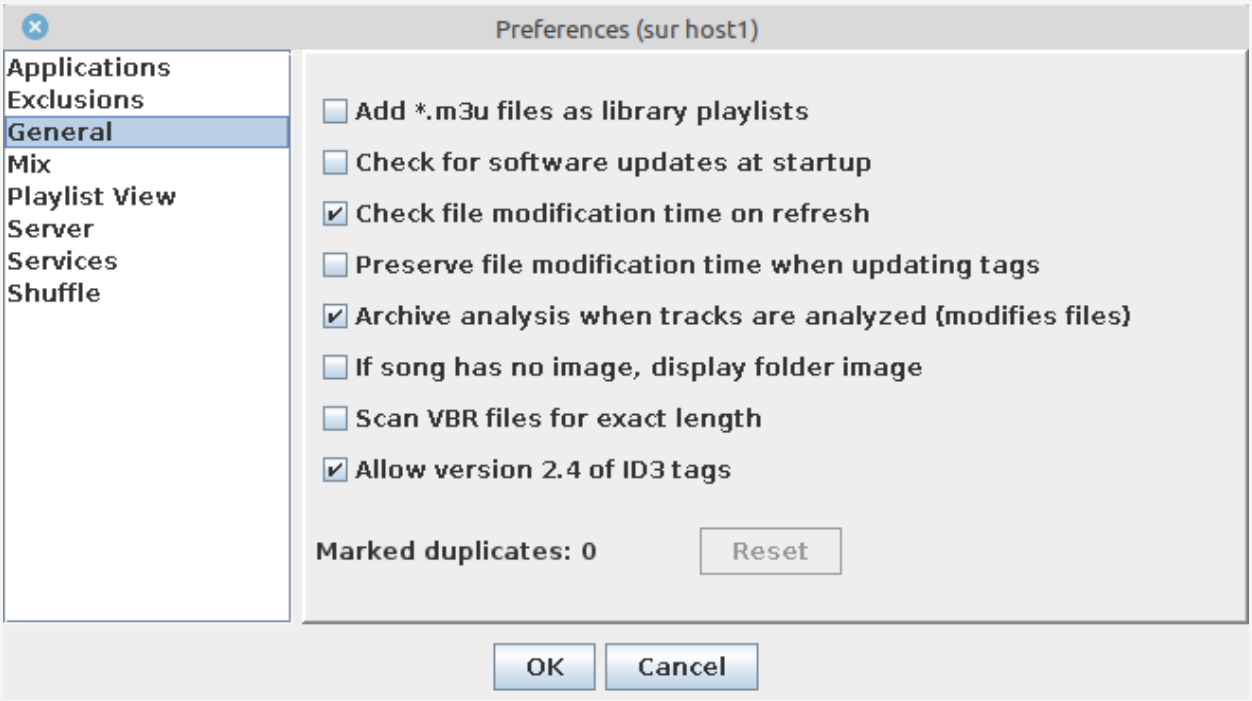
Une fois connecté par `ssh MusicIP` :

```
./start_musicip_mixer.sh
```

Les préférences du Mixer

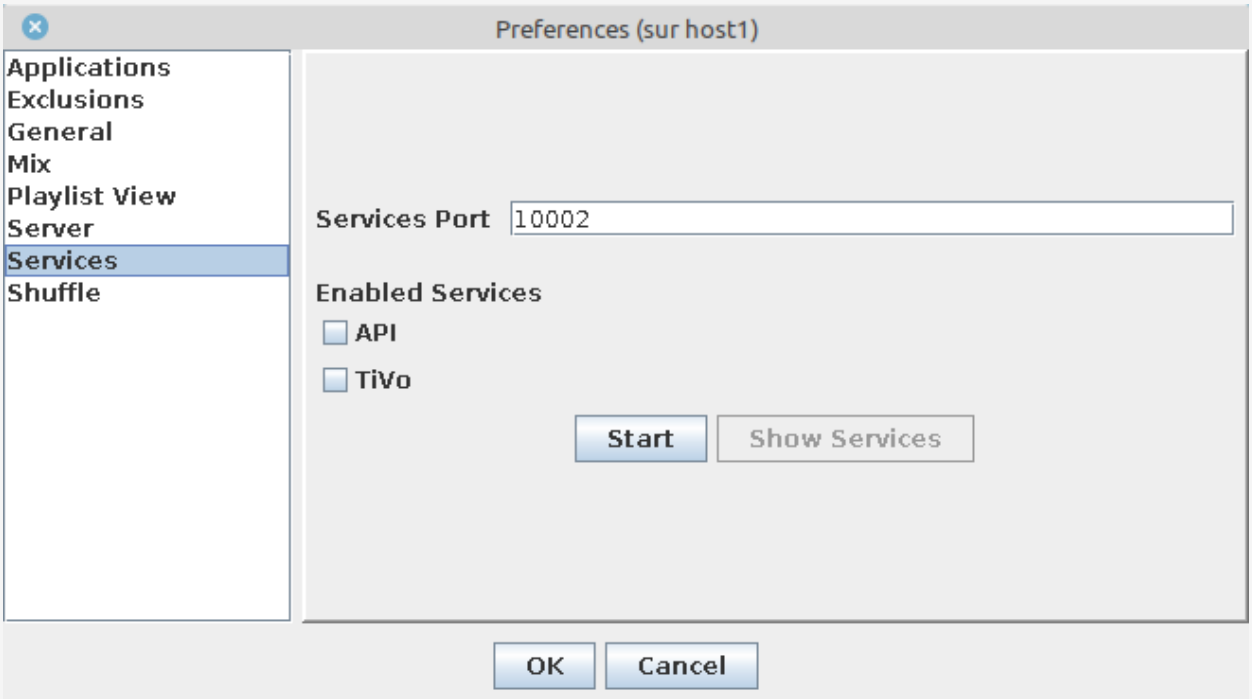
Dans Fichier → Preferences → General :

- décocher *Check for software updates on startup* — le serveur de mise à jour n'existe plus ;
- cocher *Archive analysis when tracks are analyzed* — c'est l'option qui écrit l'empreinte dans le fichier, donc celle qui rend l'analyse récupérable ;
- cocher *Allow version 2.4 of ID3 tags*.



Dans Fichier → Preferences → Services :

- décocher *API* et *TiVo*. Le Mixer et le serveur se disputeraient sinon les mêmes ports, le Mixer exposant ses propres services.



Le contrôle tient en une commande, Mixer ouvert, sur `[nas-host]` :

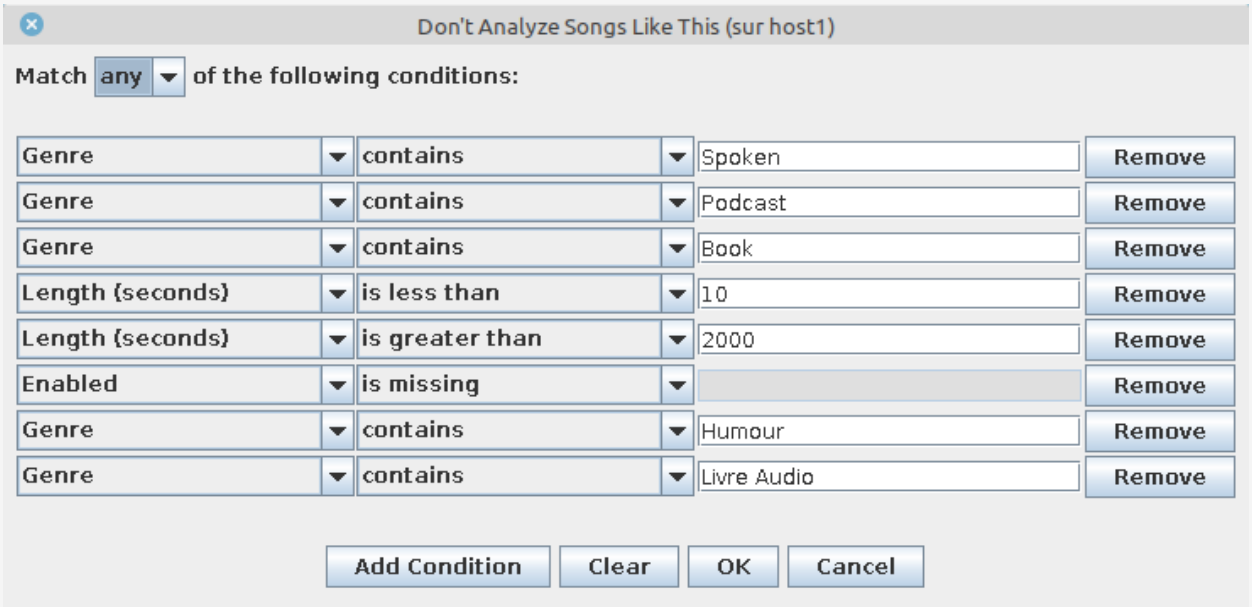
```
ss -lnt | grep :10002
```

Une seule ligne d'écoute : le Mixer n'expose rien et les deux logiciels cohabitent. Deux lignes, ou un Mixer qui refuse de démarrer en se plaignant du port, et les cases n'ont pas été décochées.

Le filtre d'exclusion, qui agit côté serveur

Dans Fichier → Preferences → Exclusions se règle un filtre dont la portée dépasse largement le Mixer : il s'applique à toutes les voies de soumission, y compris l'API HTTP et les scripts tiers. C'est un filet de second niveau qui protège la base.

Un exemple éprouvé :



> *Don't analyze songs where genre contains « ASMR » or genre contains « Podcast » or genre contains « Balado » or length is less than 10 seconds or length is greater than 4800 seconds or enabled is missing or genre contains « Livre Audio » or genre contains « Test audio »*

⚠ La durée minimale interne du serveur est d’environ dix secondes, quoi qu’on règle ici. Tout fichier plus court repasse en `active=no` de lui-même. Une valeur plus haute dans les préférences agit sur le Mixer, jamais sur ce comportement du serveur.

Un outil de soumission a intérêt à reprendre les mêmes critères de son côté : ça évite de fabriquer du trafic pour des fichiers qui seront rejetés, tout en gardant le filtre serveur comme filet.

La première audiothèque

Vider l’audiothèque du Mixer si elle contient déjà quelque chose — l’opération ne touche pas aux fichiers — puis ajouter le répertoire des médias avec le chemin exact qu’emploie Lyrion :

```
/media/nas1/Audio/LMS/
```

 Le chemin doit être identique des deux côtés. Lyrion et MusicIP s’échangent des chemins de fichiers, pas des identifiants : la moindre divergence, un lien symbolique résolu d’un côté et pas de l’autre, et les deux bases cessent de se reconnaître.

Faire Fichier → Save Cache, laisser le Mixer parcourir le répertoire — c’est très long la première fois — puis quitter le Mixer une fois le parcours terminé.

Revenir enfin sur `http://192.168.0.11:10002/server` et rafraîchir l’affichage : le bouton Reload Cache apparaît, signe que le fichier sur disque a changé sous le serveur. L’appuyer.

 Ce bouton n’est visible que lorsque la base a changé sur le disque. Son absence n’est pas un défaut : elle signifie que le serveur et le fichier sont d’accord.

---

6. Piloter le serveur depuis un script

Le serveur expose une petite API que la documentation d'origine ne décrit pas. Ce qui suit vient de l'observation des requêtes émises par l'interface web.

Ce qui répond

Ajouter un dossier — le seul point d'entrée d'ajout qui fasse quelque chose :

```
curl -sS "http://192.168.0.11:10002/server/add?root=/media/nas1/Audio/LMS/Flac"
```

Il parcourt le dossier, ignore ce que la base connaît déjà, lit les étiquettes d'empreinte des fichiers qui en portent, et analyse l'audio des autres. Il répond `HTTP 302` vers `/server`. Il est idempotent : le même dossier peut être soumis plusieurs fois sans dommage.

Lister les pièces connues :

```
curl -sS "http://192.168.0.11:10002/api/songs?extended"
```

⚠ Ce n'est pas du JSON. C'est du texte, un champ par ligne, les enregistrements séparés par des lignes vides : `name`, `artist`, `album`, `file`, `genre`, `active yes|no`, `seconds`, `bytes`, `year`, `bitrate`. Un analyseur JSON échoue immédiatement, ce qui est la bonne nouvelle — l'erreur est bruyante.

Lire les compteurs — la page d'administration, dont on extrait *Total songs*, *Mixable songs* et *Songs to validate* :

```
curl -sS "http://192.168.0.11:10002/server"
```

Lancer la validation : `POST` sur `/server/validate` avec `action=Start Validation`.

Le binaire contient aussi les chemins `/api/mix`, `/api/analysis`, `/api/duplicates`, `/api/version` et `/server/cancel`. Leur présence ne prouve pas qu'ils répondent comme on l'espère — ils sont à éprouver avant d'être employés.

Ce qui ne répond pas, et comment ça se manifeste

 `/server/add?file=<fichier>` répond `HTTP 302` et ne fait rien. Le code appelant a toutes les raisons de croire qu'il a réussi. C'est le piège le plus coûteux de cette API : il faut `root=` et un dossier, jamais `file=` et un fichier. Il n'existe aucune soumission fichier par fichier — on groupe par dossier parent et on émet une requête par dossier.

`POST /server/refresh` est sans effet. Le bouton *Refresh Songs* de l'interface web passe par un mécanisme interne que l'API n'expose pas.

`/server/delete`, `/server/remove`, `/server/purge`, `/api/delete`, `/api/remove` répondent `HTTP 500`. Ils n'existent pas.

⚠ Sur ce serveur, `HTTP 500` signifie « adresse inconnue », pas « paramètre invalide ». Toute URL qu'il ne connaît pas produit un 500. Un script qui interprète ce code comme une erreur de paramètre cherchera longtemps.

Le modèle à deux phases

Une soumission n'est pas une opération unique. Le serveur enchaîne deux phases distinctes, et les confondre fait perdre le travail.

Phase A — le catalogue. Quelques secondes à une vingtaine. Les chemins entrent dans la base et le compteur *Total songs* monte, puis se stabilise. On la détecte en interrogeant ce compteur toutes les quatre secondes et en sortant après quatre lectures identiques d'affilée.

Phase B — la validation. De l'ordre de 260 à 280 secondes, constantes. Le serveur calcule l'état mixable des pièces nouvellement cataloguées. La durée ne dépend pas du nombre de fichiers : une pièce et douze pièces prennent le même temps. On la détecte en cherchant la chaîne `Validating tracks` dans le HTML de la page `/server`.

 Une nouvelle soumission pendant la phase B fait abandonner le travail en cours. Le serveur passe au nouveau parcours et ce qui était en validation reste inachevé. Une campagne lancée en rafale produit alors le résultat déroutant d’un seul fichier ajouté par dossier : seule la première pièce de chaque dossier a eu le temps d’être traitée. Le Mixer graphique se comporte exactement pareil — il se bloque sur le premier répertoire et n’enchaîne pas.

⚠ `/api/getStatus` est un faux ami. Il renvoie `idle` pendant la phase B. Un script qui s’y fie croit le serveur disponible, enchaîne la soumission suivante, et détruit ainsi le travail en cours. La seule détection fiable de la fin de phase B est la disparition de `Validating tracks` dans la page `/server`.

L’algorithme qui tient

Pour chaque dossier, après l’envoi de la requête d’ajout :

1. attendre trois secondes ;
2. interroger *Total songs* toutes les quatre secondes ; sortir après quatre lectures stables ;
3. interroger `/server` toutes les cinq secondes ; sortir quand `Validating tracks` a disparu ;
4. journaliser le nombre ajouté et les durées des deux phases ;
5. plafonner à dix minutes par dossier, avec un avertissement, sans interrompre la campagne.

Le coût total est élevé et incompressible : à environ 280 secondes par dossier, une bibliothèque répartie sur 165 dossiers demande une douzaine d’heures. Aucun moyen connu ne l’accélère. En contrepartie, la progression est visible et aucun fichier ne se perd en route ; et les campagnes suivantes ne portent que sur les nouveautés.

---

7. Faire tenir le travail sur le disque

 Le serveur n’écrit pas la base ; le Mixer l’écrit. `default.m3lib` reste inchangé pendant que le serveur tourne, et un arrêt même propre ne l’écrit pas. Le seul écrivain est le Mixer, par Fichier → Save Cache.

Le serveur calcule pourtant bien des empreintes — c’est ce que fait une soumission HTTP sur un fichier sans étiquettes. Mais ce qu’il calcule vit en mémoire. La conséquence est brutale et silencieuse : une campagne de douze heures peut repartir au redémarrage suivant sans qu’aucun message ne le signale.

Le geste est donc obligatoire, pas facultatif, après toute campagne d’ajout :

1. relever la date de la base, sur `[nas-host]` :

```
ls -l --time-style=full-iso /home/hostadmin/.MusicMagic/default.m3lib
```

1. ouvrir le Mixer et faire Fichier → Save Cache ;
2. relever la date à nouveau avec la même commande.

Si la date n’a pas changé, le travail n’est pas sur le disque — et c’est le seul contrôle qui le dise. Une minute de vérification contre douze heures de calcul.

Réconcilier le Mixer et le serveur

Les deux programmes tiennent chacun leur propre état en mémoire et ne se parlent pas directement : ils ne se rejoignent que par le fichier de base. C’est la raison d’être du couple *Save Cache / Reload Cache*, et c’est pourquoi leurs compteurs peuvent diverger sans que rien n’aille mal.

L’ordre à suivre après une campagne d’analyse :

1. attendre que le Mixer ait terminé — la mention d’analyse disparaît de sa barre d’état ;
2. Fichier → Save Cache dans le Mixer ;
3. vérifier que la date du fichier a bougé ;
4. dans la page `/server`, appuyer sur Reload Cache — le bouton n’apparaît que lorsque le fichier a changé sous le serveur, et disparaît une fois les deux en accord ;
5. si *Songs to validate* reste non nul, Start Validation.

 Ne pas appuyer sur *Reload Cache* pendant que le Mixer travaille. Le serveur adopterait un instantané partiel, et le compte des pièces mixables reculerait sans explication.

⚠ La validation s’exécute dans le serveur, qui n’écrit pas le cache. Son résultat ne survit donc probablement pas à un redémarrage. Le contrôle est gratuit : après le redémarrage suivant, regarder si *Songs to validate* est remonté. Si c’est le cas, un *Save Cache* depuis le Mixer doit suivre chaque validation.

Sauvegarder la base

La base concentre des années d’analyse et rien ne la reconstruit en un temps raisonnable. Elle mérite une copie datée dans la rotation de sauvegardes, au même titre qu’une base de données :

```
/home/hostadmin/.MusicMagic/default.m3lib
```

Une copie par jour, quelques jours de rétention. Le fichier pèse quelques dizaines de mégaoctets pour une bibliothèque de l’ordre de 80 000 pièces.

8. Entretien

Les permissions, et la cause qui les défait

Les écritures du Mixer changent le propriétaire et le groupe des fichiers analysés. Lyrion, qui lit la bibliothèque par les droits de groupe, voit alors ses pièces disparaître de sa base au fil de l'analyse.

 Le bit `setgid` sur les dossiers traite la cause plutôt que le symptôme. Un dossier en `2775` impose son propre groupe à tout fichier créé dedans, quel que soit le groupe principal du compte qui écrit. Le groupe cesse de se perdre, et la remise en état périodique devient inutile.

Ce qui décide de l'accès, c'est l'appartenance au groupe — pas le bit « autres ». Le compte qui fait tourner Lyrion doit être membre de `sambashare` ; à vérifier avant de se demander quel mode poser :

```
id squeezeboxserver
```

Remettre une arborescence d'aplomb

Le traitement porte sur tout le domaine musical — `/media/nas1/Audio` — et non dossier par dossier. Les fichiers naissent alors corrects partout où ils naissent, et peu importe ensuite qu'on les copie ou qu'on les déplace.

⚠ Ne pas l'étendre au volume entier. Un disque qui porte aussi des sauvegardes exige l'inverse : leur valeur tient à ce qu'elles conservent propriétaires et modes d'origine, y compris des secrets en `600`. Un traitement uniforme les détruirait et les exposerait à travers les partages.

 Deux choses doivent être écartées, et une commande récursive naïve les manque. Les corbeilles `.Trash-1000`, dont le mode `700` est voulu. Et un éventuel miroir mp3fs : un montage FUSE qui transcode la bibliothèque à la volée est en lecture seule, et le parcourir déclenche un calcul de taille — donc un

transcodage — sur chaque fichier virtuel. `-xdev` empêche `find` de franchir une frontière de système de fichiers, ce qui l'écarte sans avoir à le nommer ; seul son point de montage se prune par son nom.

D'abord la sauvegarde des droits, seul retour arrière sur une arborescence de cette taille :

```
cd /media/nas1/Audio
sudo sh -c "find /media/nas1/Audio -xdev -print0 \
    | xargs -0 getfacl -p > /root/acl-audio.txt"
```

La redirection passe par `sh -c` parce qu'un `sudo commande > /root/fichier` échoue : c'est le shell appelant, sans droits sur `/root`, qui traite le `>`.

Puis le traitement, dans cet ordre — propriétaire, modes, ACL :

```
sudo find . -xdev -name '.Trash*' -prune -o -name sub_mp3-128 -prune -o \
     -exec chown -h hostadmin:sambashare {} +
sudo find . -xdev -name '.Trash*' -prune -o -name sub_mp3-128 -prune -o \
     -type d -exec chmod 2775 {} +
sudo find . -xdev -name '.Trash*' -prune -o -name sub_mp3-128 -prune -o \
     -type f -exec chmod 664 {} +
sudo find . -xdev -name '.Trash*' -prune -o -name sub_mp3-128 -prune -o \
     -type d -exec setfacl -m g:sambashare:rwX -m d:g:sambashare:rwX {} +
sudo find . -xdev -name '.Trash*' -prune -o -name sub_mp3-128 -prune -o \
     -type f -exec setfacl -m g::rw,g:sambashare:rw {} +
```

⚠ Le `-h` de `chown` n'est pas décoratif. Les dossiers que lit Lyrion ne contiennent que des liens symboliques ; sans lui, `chown` modifierait leurs cibles au lieu des liens.

 La dernière ligne ne peut pas être un `chmod`, et c'est contre-intuitif. Sur un fichier porteur d'ACL, `chmod` agit sur le masque — jamais sur l'entrée `group::`. Un `chmod 664` laisse donc intact un `group::rwx` hérité d'un ancien `chmod -R 774`, et le masque remonte à `rwx` au premier `setfacl` suivant. Seul `setfacl` atteint cette entrée. Pour la même raison on n'écrit pas `rwX` ici : le `X` majuscule

rendrait le `x` qu'on cherche à retirer.

Restauration complète — propriétaires, modes et ACL :

```
sudo setfacl --restore=/root/acl-audio.txt
```

Ce que les ACL ajoutent aux modes

Les deux lignes `setfacl` ne font pas la même chose. La première pose l'accès sur ce qui existe déjà. La seconde, avec son préfixe `d:`, pose l'ACL par défaut — elle ne vit que sur les dossiers et décide de ce dont héritent les fichiers créés ensuite. Sans elle, chaque nouvel album arrive nu.

Un mode d'apparence absurde s'explique presque toujours par le masque : un fichier affiché `674` n'a pas un groupe plus permissif que son propriétaire, il a un masque `rwx` et un propriétaire en `rw-`. La position du groupe qu'affiche `ls` est le masque dès qu'une ACL est posée.

 Ces commandes visent `Musique`, où résident les fichiers, et non `LMS`, qui n'en est qu'un ensemble de liens symboliques. Le détail des permissions, des partages et des ACL est dans INCUS — Hôte et conteneur & serveur NAS.

Contrôler plutôt que supposer

L'épreuve qui tranche est la création d'un fichier : si le `setgid` agit, il sort au groupe du dossier et non au groupe principal du compte.

```
cd /media/nas1/Audio/Musique
touch ./essai.tmp; stat -c '%a %U:%G %n' ./essai.tmp; rm -f ./essai.tmp
```

Le fichier doit porter `664` et le groupe `sambashare`. Et pour la vue d'ensemble :

```
find . -type d -printf '%m\n' | sort | uniq -c | sort -rn | head -4
find . -type f -printf '%m\n' | sort | uniq -c | sort -rn | head -4
```

Des dossiers en `2775` et des fichiers en `664`, sans dispersion : c’est le résultat attendu.

Les chemins fantômes

Renommages, déplacements et suppressions laissent dans la base des chemins qui ne désignent plus rien. Le bouton Refresh Songs de l’interface web les purge — en quelques minutes, même par centaines.

⚠ Ce bouton n’est pas scriptable. Il passe par un mécanisme interne que l’API n’expose pas ; c’est l’unique geste de ménage du montage qui exige un navigateur. À faire avant une grande campagne d’ajout.

Ce ménage améliore la qualité de la base, mais pas la vitesse : la phase B dure ses 280 secondes quelle que soit la propreté de la base.

Le journal du serveur

Le serveur écrit dans `/home/hostadmin/.MusicMagic/log.txt`, qui n’est soumis à aucune rotation et grossit indéfiniment — plusieurs dizaines de mégaoctets après quelques campagnes d’analyse.

Sur `[nas-host]` :

```
sudo nano /etc/logrotate.d/musicip
```

Contenu à coller dans l’éditeur ouvert ci-dessus :

```
/home/hostadmin/.MusicMagic/log.txt {
    size 10M
    rotate 3
    compress
    delaycompress
    missingok
    notifempty
```

```
    copytruncate
    su hostadmin hostadmin
}
```

⚠ `copytruncate` est indispensable ici. Le serveur garde le fichier ouvert en permanence et ne sait pas rouvrir son journal sur demande. Une rotation par renommage le ferait écrire dans un fichier devenu invisible, et le journal resterait vide indéfiniment.

Le déclenchement se fait à la taille et non au calendrier : ce journal ne bouge pas pendant des mois, puis prend des dizaines de mégaoctets en une campagne d'analyse. Une rotation mensuelle ferait tourner onze fichiers vides par an et laisserait quand même enfler celui de la campagne.

`su hostadmin hostadmin` est requis parce que le dossier parent n'appartient pas à `root` : sans cette ligne, logrotate écarte le fichier en se plaignant de ses permissions — dans son propre journal, où personne ne va voir.

Vérifier sans rien modifier :

```
sudo logrotate -d /etc/logrotate.d/musicip
```

La sortie doit montrer `switching euid from 0 to 1000`, preuve que la bascule de compte a eu lieu, puis `copying …` suivi de `truncating …`. Toute plainte sur les permissions du dossier parent signale que la ligne `su` manque ou porte le mauvais compte.

---

9. Intégrer de nouveaux albums

Album natif, empreinte persistante — le cas recommandé

1. `[poste de travail]` — soumettre le dossier au Mixer de l'instance

   Traitement, qui analyse et écrit les étiquettes dans les fichiers.
2. Vider l'instance Traitement.
3. Déposer les fichiers étiquetés dans la bibliothèque, sous `/media/nas1/Audio/LMS/`.
4. Soumettre le dossier à Production par `?root=`. L'ajout est rapide : les étiquettes sont lues, rien n'est réanalysé.
5. Faire *Save Cache* et vérifier la date de la base.

Album natif, empreinte volatile — le raccourci

Acceptable si la bibliothèque bouge peu et que la base est sauvegardée.

1. Déposer les fichiers sous `/media/nas1/Audio/LMS/`.
2. Soumettre le dossier à Production par `?root=`.
3. Attendre la fin des deux phases.
4. Faire *Save Cache* et vérifier la date de la base.

Album haute résolution — la voie persistante est obligatoire

1. `[poste de travail]` — fabriquer un rééchantillonnage 16/44 du dossier, avec `sox` ou `ffmpeg`, en préservant toutes les étiquettes textuelles.
2. Le faire analyser par le Mixer de l'instance Traitement, qui écrit les étiquettes dans les copies rééchantillonnées.
3. Reporter les étiquettes `fingerprint` et `analysis` des copies vers les fichiers d'origine — `metaflac` pour du FLAC. (Le cas ne se pose pas en MP3, qui est toujours en 16 bits.)
4. Vider l'instance Traitement.
5. Déposer les fichiers d'origine dans la bibliothèque.
6. Soumettre le dossier à Production par `?root=`. Le fichier hors norme est accepté, puisque le serveur lit ses étiquettes sans ouvrir l'audio.

7. Faire *Save Cache* et vérifier la date de la base.

Les scripts de rééchantillonnage et de report d'étiquettes sont sur la page Scripts bash et Zenity.

---

10. Côté Lyrion Music Server

Trois greffons interrogent le serveur MusicIP, et ils coexistent sans se gêner. Chacun a sa propre page sous *Réglages → Extensions*, et ses propres réglages de mélange.

 Le service s'appelle `lyrionmusicserver.service`, ses fichiers vivent sous `/var/lib/squeezeboxserver/`. Les deux noms coexistent : chercher un service `squeezeboxserver` ne donne rien, et chercher les préférences sous `lyrion` non plus.

Lequel est livré, lesquels sont à télécharger

 La distinction décide de ce qui survit à un incident. MusicIP est livré avec le serveur, sous son nom de code `MusicMagic` : il vit dans `/usr/share/perl5/Slim/Plugin/MusicMagic`, il est réinstallé à chaque montée de version et ne peut pas disparaître. ⚠ Le chercher sous « MusicIP » ne donne rien — le fichier de préférences porte ce nom, le code porte l'autre. SugarCube et MIPMixer sont des greffons tiers, téléchargés depuis leurs dépôts et posés dans le cache des extensions — c'est-à-dire à l'endroit exact qu'un vidage du cache efface, et dont le contenu n'est plus téléchargeable si le dépôt amont ferme.

La sauvegarde du serveur couvre ce répertoire pour cette raison. L'installation des greffons, leurs dépôts et leur mise à jour automatique sont décrits sur Lyrion Music Server.

Le greffon MusicIP

C’est le pont entre le serveur de musique et le serveur d’empreintes. Ses réglages vivent dans `/var/lib/squeezeboxserver/prefs/plugin/musicip.prefs` :

```
mix_size: '20'
mix_style: '200'
mix_variety: '1'
mix_genre: '1'
mix_type: 0
mix_filter: 0
reject_size: 0
reject_type: 0
scan_interval: 3600
port: 10002
player_settings: 0
playlist_prefix: ''
playlist_suffix: ''
```

`mix_size` est le nombre de pièces d’un mélange, `scan_interval` l’intervalle en secondes entre deux interrogations du serveur MusicIP, et `port` le port sur lequel celui-ci répond.

⚠ Il n’y a pas de champ d’adresse. Ce greffon suppose que le serveur MusicIP tourne sur la même machine que le serveur de musique. Les deux autres greffons, eux, ont un champ d’hôte — d’où une asymétrie qui surprend quand on cherche où changer l’adresse.

SugarCube

C’est lui qui prolonge la file de lecture toute seule, en demandant à MusicIP des pièces proches de ce qui vient d’être joué. Les réglages qui agissent, dans `SugarCube.prefs` :

```
miphosturl: localhost
sugarport: '10002'
sugarmipsize: '10'
```

```
sugardelay: '1'
sugarhisweight: '85'
sugarlvweight: '84'
sqlitetimeout: '30'
rating_10scale: 0
sc_mipexport_enabled: 0
sc_mipexport_time: 04:17
```

 Le fichier est bien plus gros que cette liste — un millier de lignes. Le reste se partage entre les réglages par platine, un par lecteur, et une trentaine de clés `traf_` vides : les restes d'un module d'annonces de trafic routier que la version actuelle a retiré, et que le fichier de préférences traîne sans que rien ne les relise. Ce sont les dix lignes ci-dessus qui portent le fonctionnement.

SugarCube tient sa propre base, `sugarcube.db`, à côté de ses préférences. Elle vit dans `prefs/`, donc la sauvegarde du serveur la couvre.

D'où vient ce greffon, et ce que raconte sa clé de licence

SugarCube a changé de mains deux fois : écrit par Spicefly, repris par la communauté, puis modifié dans un second dépôt. C'est cette dernière version qui est installée ici, et elle est publiée sous licence GNU GPL v3 — il n'y a rien à acheter ni à activer.

⚠ Sa clé `scubelicense` affiche pourtant un message d'expiration, et c'est un fossile. Elle est héritée de la version d'origine, qui était payante ; le code actuel se contente de la lire pour l'afficher sur une page de réglages, sans jamais l'évaluer. Le serveur de licences qu'elle nomme n'existe d'ailleurs plus.

 Conséquence : le greffon *License Manager* ne sert plus à rien, et le message disparaît en le retirant. C'est un cas d'école du message alarmant qui ne signale aucune panne — et qui fait chercher un réglage introuvable, parce qu'il n'y a rien à régler.

 Le greffon TrackStat, que la version d’origine réclamait pour les statistiques, n’est plus nécessaire : notes et nombres d’écoutes se lisent directement dans la base de Lyrion. L’affichage s’active dans *Réglages* → *Avancé* → *SugarCube*.

MIPMixer

Un second pont vers le même serveur MusicIP, indépendant du premier. Son intérêt est ailleurs : il sait éviter les répétitions sur une longue file.

```
host: localhost
port: 10002
no_repeat_track: 100
no_repeat_artist: 15
no_repeat_album: 25
filter_genres: 1
filter_xmas: 1
mix_style: 20
mix_variety: 0
mix_type: 0
mix_filter: 0
min_duration: 0
max_duration: 0
```

Les trois `no_repeat_` se comptent en pièces : cent titres avant qu’une même piste puisse revenir, quinze avant un même artiste, vingt-cinq avant un même album.

⚠ Le piège : un même réglage, trois échelles

Le style de mélange existe dans les trois configurations sous des noms voisins — et les valeurs ne sont pas comparables. Ici : `style=100` dans `mmm.ini`, `mix_style: 200` dans le greffon MusicIP, `mix_style: 20` dans MIPMixer.

Ce ne sont pas trois réglages désaccordés qu’il faudrait harmoniser : chaque programme a sa propre échelle. Une valeur ne se recopie pas d’une configuration à l’autre, elle se règle à l’écoute dans chacune.

11. Les pièges, rassemblés

Ce qui casse en silence, sans message d’erreur :

- un ajout par `?file=` au lieu de `?root=` — réponse `302`, aucun effet ;
- un fichier hors 16/44 soumis sans étiquettes préalables — simplement absent du compte ;
- une soumission pendant la phase B — le travail en cours est abandonné ;
- `/api/getStatus` renvoyant `idle` en pleine validation ;
- une campagne non suivie d’un *Save Cache* — perdue au redémarrage ;
- `RemainAfterExit=yes` dans l’unité — un serveur mort déclaré vivant ;
- `After=network.service` — une unité qui n’existe pas, ordonnancement nul ;
- `WantedBy=default.target` sur une machine dont la cible est graphique ;
- un chemin de bibliothèque qui diffère entre Lyrion et MusicIP ;
- un JRE 64 bits pour le Mixer — démarre, puis échoue sur la bibliothèque native.

12. Références

- Téléchargement de MusicIP : github.com/CDrummond/musicip
- Téléchargement de SugarCube : github.com/HB64/lms-sugarcube
- Scripts de rééchantillonnage : Scripts bash et Zenity
- Le serveur, ses greffons et ses dépôts : Lyrion Music Server
- Permissions, partages et ACL : INCUS — Hôte et conteneur & serveur NAS