

# Mémo — Gestion des Formats Multilingues (Binaire & Hexadécimal)

---

## Objectif de ce mémo

Assurer un fonctionnement **clair, stable et bilingue** du système de formats du convertisseur (Brut, Blocs de 4, Blocs de 8, etc.) **sans complexité inutile**, tout en gardant le format choisi lors du changement de langue.

Ce document explique :

- Pourquoi le problème original apparaissait
  - Comment fonctionne la solution simple
  - La fonction utilitaire `trouver_cle_format()`
  - Les fonctions finales proprement structurées
  - Un schéma logique clair
  - Une section pour se rappeler de l'essentiel
- 

## 1 Le problème initial

---

Au changement de langue :

- Les options des menus déroulants étaient régénérées dans la nouvelle langue
- **Mais le format affiché était basé sur les textes eux-mêmes**, qui changent d'une langue à l'autre
- Le programme ne savait donc plus que "Blocs de 4" en français correspond à "Blocks of 4" en anglais

Résultat :

- Le format sélectionné pouvait être perdu
  - Ou un mauvais format pouvait être appliqué
  - Ou le premier lancement ne fonctionnait pas correctement
- 

## La solution adoptée (simple et robuste)

---

Nous avons choisi la méthode **la plus simple** à maintenir :

- **Comparer directement les textes traduits**, mais
- **Conserver l'ancien texte AVANT le changement de langue**, puis
- Le transformer en **clé de format** grâce à une fonction utilitaire
- Puis réappliquer le même format dans la nouvelle langue

Ainsi :

Pas de "code interne" ajouté  
Pas de callbacks compliqués  
Le programme reste facile à comprendre  
Le format reste cohérent même après changement de langue  
fonctionnement correct dès le premier lancement

---

## Fonction utilitaire : `trouver_cle_format()`

Cette petite fonction est la clé permettant de garder de la simplicité.  
Elle permet de transformer le texte affiché avant le changement de langue en une clé logique :

```
def trouver_cle_format(ancien_texte):
    cles_formats = ["brut", "blocs_2", "blocs_4", "blocs_8"]
    for cle in cles_formats:
        if anciennes_traductions.get(cle) == ancien_texte:
            return cle
    return "brut"
```

Elle travaille avec :

- `ancien_texte` = texte visible *dans l'ancienne langue*
- `anciennes_traductions` = contenu du fichier JSON de la langue actuelle

Elle renvoie :

- "`brut`"
- "`blocs_2`"
- "`blocs_4`"
- "`blocs_8`"

Grâce à elle, on sait exactement quel format logique l'utilisateur avait choisi avant de changer de langue.

---

## Fonction `changer_langue()` — Version finale et propre

Voici la version commentée, lisible et pédagogique :

```
def changer_langue(nouvelle_langue):

    global langue_actuelle, textes_langues
    global zone_texte_aide, panneau_aide, bouton_fermer_aide
    global zone_texte_contexte, panneau_contexte, bouton_fermer_contexte

    # Sauvegarder la langue actuelle et les textes visibles des formats
```

```
ancienne_langue = langue_actuelle
ancienne_valeur_binaire = format_binaire_var.get()
ancienne_valeur_hexa = format_hexadecimal_var.get()

# Charger les traductions de l'ancienne langue
anciennes_traductions =
charger_traductions(f"./Langues/lang_{ancienne_langue}.json")

# Fonction utilitaire
def trouver_cle_format(ancien_texte):
    cles_formats = ["brut", "blocs_2", "blocs_4", "blocs_8"]
    for cle in cles_formats:
        if anciennes_traductions.get(cle) == ancien_texte:
            return cle
    return "brut"

# Déduire la clé logique à partir de l'ancien texte
cle_format_binaire = trouver_cle_format(ancienne_valeur_binaire)
cle_format_hexa = trouver_cle_format(ancienne_valeur_hexa)

# Changer la langue actuelle
langue_actuelle = nouvelle_langue

# Charger la nouvelle traduction
textes_langues = charger_traductions(f"./Langues/lang_{langue_actuelle}.json")

# Retraduire le message éventuel dans erreur_label
texte_actuel = erreur_label.cget("text")
anciennes_traductions =
charger_traductions(f"./Langues/lang_{ancienne_langue}.json")
for cle, texte in anciennes_traductions.items():
    if texte_actuel == texte and cle in textes_langues:
        erreur_label.config(text=textes_langues[cle])
        break

# Mise à jour générale de l'interface (labels, menus, etc.)
mettre_a_jour_interface()
mettre_a_jour_boutons_radio()
construire_menus()
fenetre.update_idletasks()
fenetre.geometry("")

# Réappliquer les formats dans la nouvelle langue
format_binaire_var.set(textes_langues[cle_format_binaire])
format_hexadecimal_var.set(textes_langues[cle_format_hexa])
appliquer_format_binaire()
appliquer_format_hexadecimal()

# Gestion des panneaux Aide et Contexte
if zone_texte_aide is not None:
    charger_fichier_aide()
if panneau_aide is not None:
    panneau_aide.config(text=textes_langues["titre_aide"])
if bouton_fermer_aide is not None:
```

```
bouton_fermer_aide.config(text=textes_langues["fermer"])

if zone_texte_contexte is not None:
    charger_fichier_contexte()
if panneau_contexte is not None:
    panneau_contexte.config(text=textes_langues["titre_contexte"])
if bouton_fermer_contexte is not None:
    bouton_fermer_contexte.config(text=textes_langues["fermer"])
```

---

## Fonctions finales des formats

---

### Binaire

```
def appliquer_format_binaire(*args):
    val = format_binaire_var.get()
    texte = binaire_brut_var.get()

    if val == textes_langues["brut"]:
        resultat_binaire.config(text=texte)
    elif val == textes_langues["blocs_4"]:
        resultat_binaire.config(text=grouper_par_blocs(texte, 4))
    elif val == textes_langues["blocs_8"]:
        resultat_binaire.config(text=grouper_par_blocs(texte, 8))

    ajuster_label(resultat_binaire)
```

### Hexa

```
def appliquer_format_hexadecimal(*args):
    val = format_hexadecimal_var.get()
    texte = hexadecimal_brut_var.get()

    if val == textes_langues["brut"]:
        resultat_hexadecimal.config(text=texte)
    elif val == textes_langues["blocs_2"]:
        resultat_hexadecimal.config(text=grouper_par_blocs(texte, 2))
    elif val == textes_langues["blocs_4"]:
        resultat_hexadecimal.config(text=grouper_par_blocs(texte, 4))
    elif val == textes_langues["blocs_8"]:
        resultat_hexadecimal.config(text=grouper_par_blocs(texte, 8))

    ajuster_label(resultat_hexadecimal)
```

# Schéma logique ultra simple

```
[Avant changement de langue]
|
▼
Je lis le texte : "Blocs de 4"
|
▼
Je retrouve sa clé logique : "blocs_4"
|
▼
Je charge la nouvelle langue
|
▼
Je recrée les menus
|
▼
Je remets format_binaire_var = "Blocks of 4"
|
▼
J'applique appliquer_format_binaire()
|
▼
Résultat correct dans la nouvelle langue
```

## Pour mémoire

Tu veux te souvenir de l'essentiel ?

- Le texte visible change selon la langue.**
- La clé logique (brut, blocs\_4...) ne change jamais.**
- Au changement de langue :**

1. Sauvegarde le texte actuel
2. Trouve sa clé avec `trouver_cle_format()`
3. Recharge la langue
4. Recrée les menus
5. Réaffiche le bon texte traduit
6. Réapplique le format

C'est tout.

Et ça marche **à chaque fois**, dans toutes les langues.