Manuel Utilisateur de SuperBOL Studio

Version 1.0: Le Guide Ultime

L'équipe SuperBOL de Titagone-OCamlPro

26 février 2026

Table des matières

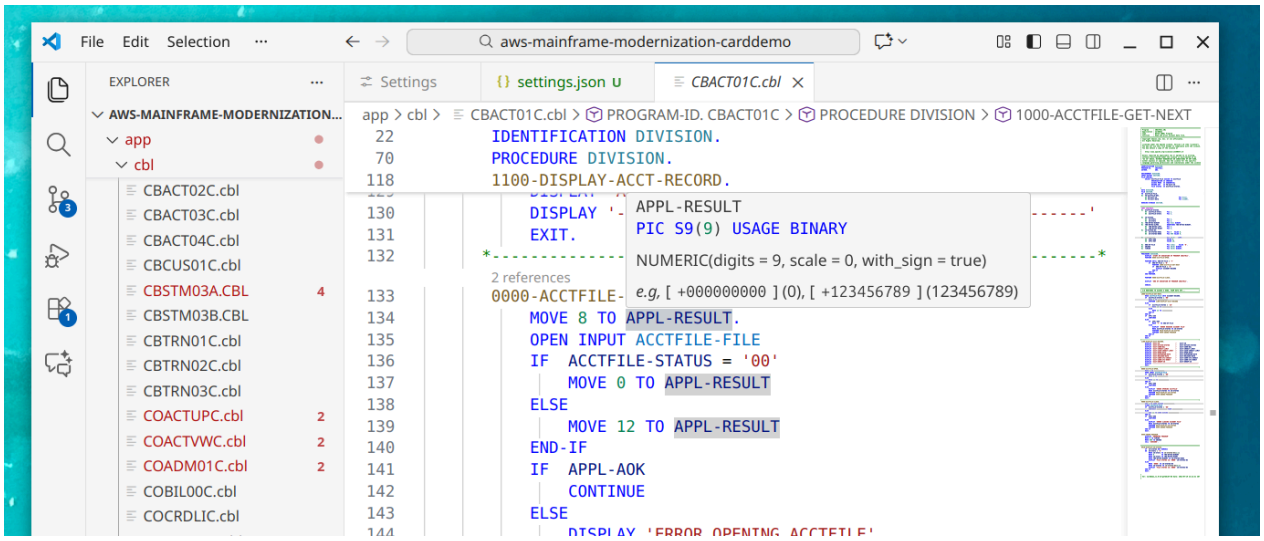
Ce manuel est disponible :

- En tant que site web HTML: en Français ou en Anglais
- En tant que fichier PDF : en Français ou en Anglais

1 Avant-propos

SuperBOL Studio est une extension de VSCode pour développer en COBOL. SuperBOL Studio utilise un serveur LSP (Language Server Protocol) pour comprendre les sources COBOL, afficher diverses informations et aider les développeurs à naviguer dans les sources.

SuperBOL Studio a été initialement conçu par Titagone-OCamlPro pour fonctionner avec le compilateur libre GnuCOBOL.



2 Introduction

Bienvenue dans ce manuel d'utilisation. Ce document explique comment installer, configurer et utiliser l'extension SuperBOL Studio pour Vscode afin de modifier les sources d'applications COBOL.

L'extension SuperBOL Studio permet aux développeurs COBOL de bénéficier d'une expérience IDE moderne pour éditer leurs programmes COBOL.

Liens relatifs à SuperBOL Studio :

- Description sur le site Web SuperBOL : https://superbol.eu/solutions/superbol-studio/
- Description sur Open-VSX : https://open-vsx.org/extension/OCamlPro/SuperBOL
- Description sur Vscode Marketplace : https://marketplace.visualstudio.com/items?itemName=OCamlPro.SuperBOL
- Sources sur Github : https://github.com/ocamlpro/superbol-studio-oss
- Autre documentation : https://ocamlpro.github.io/superbol-studio-oss/sphinx

Dans ce manuel, les captures d'écran sont souvent prises dans un thème de couleur claire, afin de faciliter l'impression de ce document, bien que de nombreux utilisateurs préfèrent utiliser VSCode dans le thème sombre par défaut. Une section du chapitre « Utilisation de Vscode » montre comment passer d'un thème de couleur à l'autre.

3 Installation

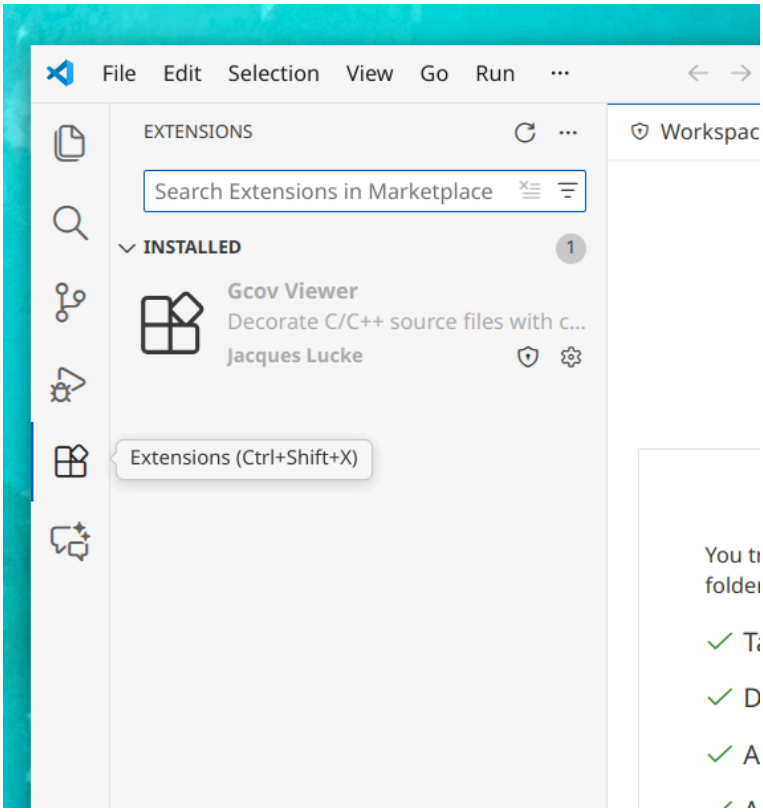
Vous pouvez installer SuperBOL Studio soit directement depuis VSCode (en utilisant les marketplaces d'extensions), soit via un fichier VSIX téléchargé.

3.1 Installation depuis VSCode

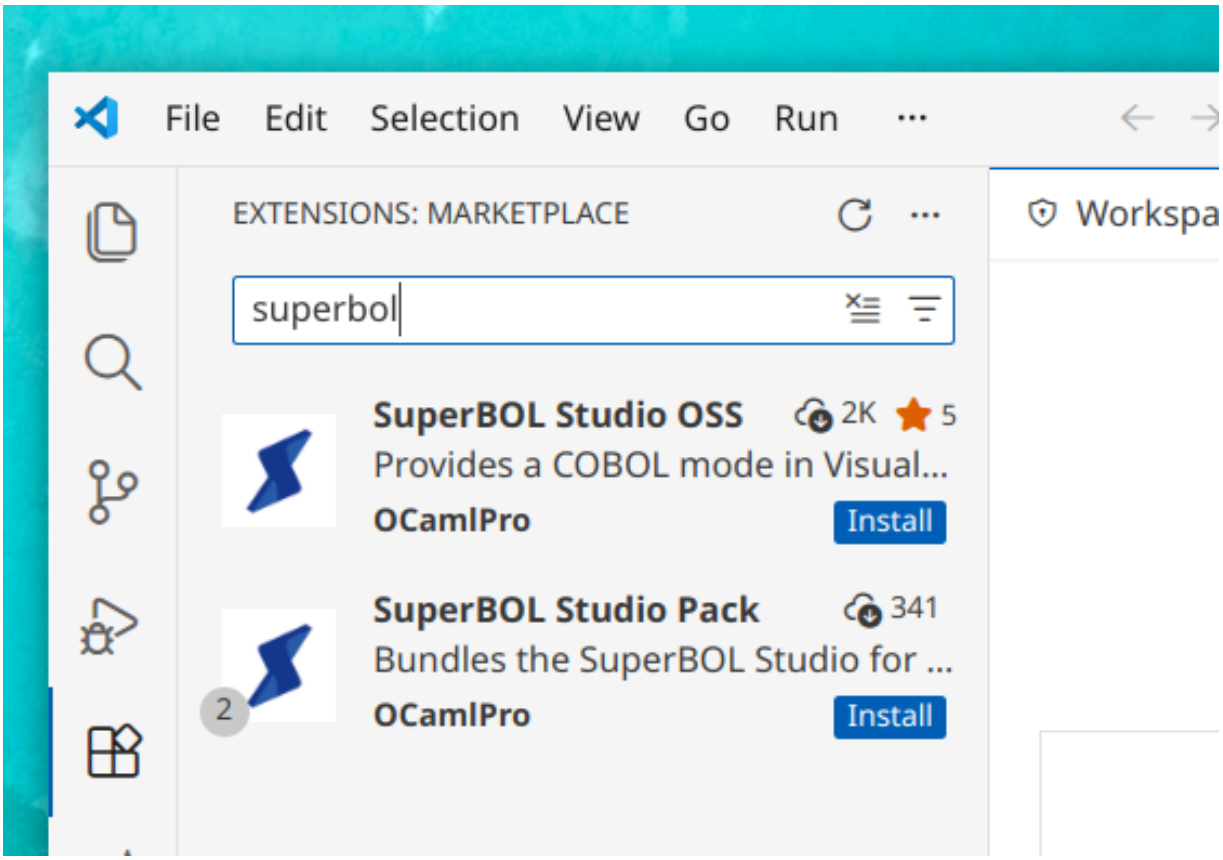
C'est la façon la plus simple d'installer SuperBOL Studio, directement depuis VSCode. De cette manière, vous n'installerez que les versions officielles de SuperBOL Studio. En fonction de votre version de VSCode, les extensions proviennent soit du Microsoft VSCode Marketplace, soit du dépôt Open-OSX, géré par la Fondation Eclipse.

Procédez comme suit :

1. Cliquez sur l'icône « Extensions » dans la barre d'activité de l'écran d'accueil. à gauche, ou appuyez sur Ctrl+Shift+X.



2. Recherchez « superbol »



3. Les premiers éléments qui devraient apparaître sont
   - SuperBOL Studio OSS : l'extension que vous voulez installer pour utiliser SuperBOL Studio
   - SuperBOL Studio Pack : une extension qui contient l'extension « SuperBOL Studio OSS », ainsi que d'autres extensions tierces.
4. Cliquez sur le bouton « Installer » de l'une de ces deux extensions.
5. L'extension s'installera d'elle-même, et le bouton « Install » se transformera en un bouton de roue de configuration.

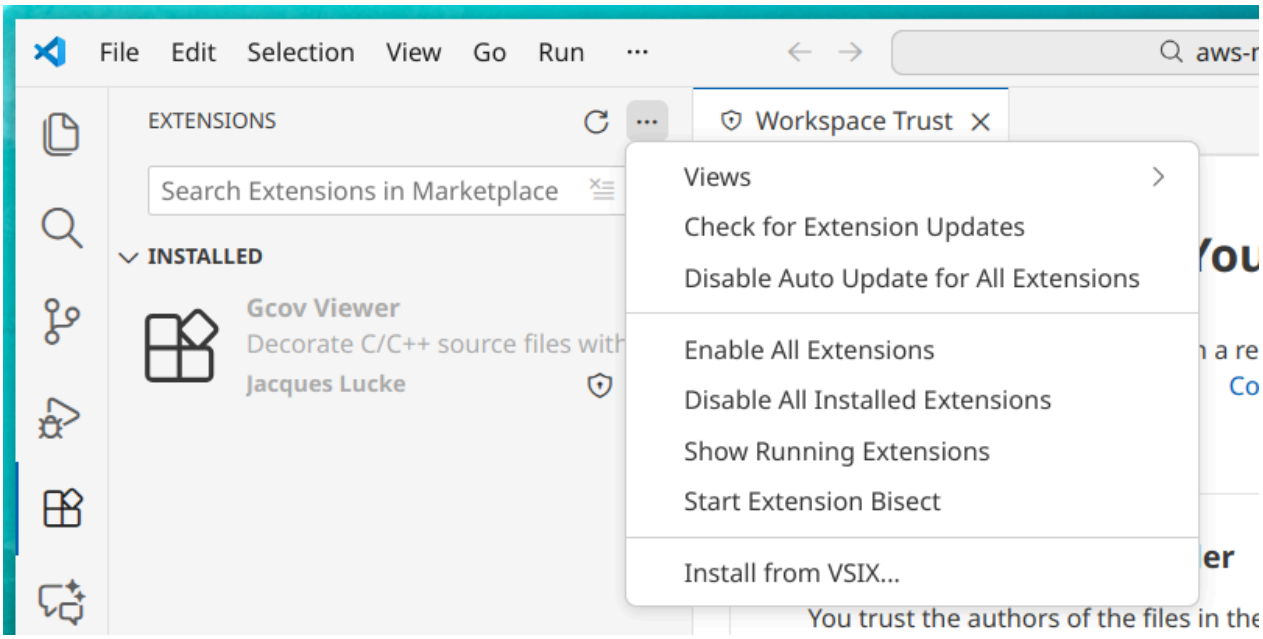
Vous trouverez d'autres instructions pour l'installation d'extensions directement dans VS-Code sur cette page.

3.2 Installation avec un fichier VSIX téléchargé

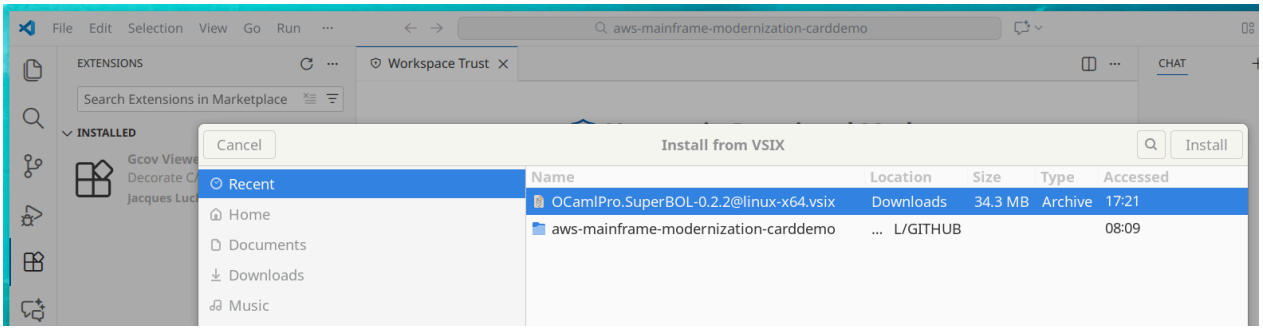
C'est la méthode à utiliser si vous voulez installer une version non officielle de SuperBOL Studio, typiquement une version de test avec des fonctionnalités bêta, ou une version compilée à partir des sources sur Github.

Procédez comme suit :

1. Vérifiez l'emplacement du fichier d'extension sur votre ordinateur. Il doit se présenter sous la forme d'un fichier unique portant l'extension `.vsix`
2. Dans VSCode, allez dans la vue « Extensions ».
3. Dans la barre latérale, cliquez sur les trois points (. . .) en haut à droite (juste au-dessus de `recherche`),



4. Sélectionnez `Installer à partir de VSIX...`, et une boîte de dialogue devrait apparaître
5. Choisissez le fichier VSIX sur votre disque, et cliquez sur Installer.



Notez que vous pouvez également utiliser cette méthode pour installer les fichiers Vsix téléchargés sur les marketplaces. Pour cela, vous devez sélectionner le binaire binaire correct pour votre plateforme :

Works With

VS Code: ^1.64.0

Target Platforms: Windows x64, Linux x64, macOS Intel, Universal

Resources

Homepage

Repository

Bugs

DOWNLOAD

- Windows x64
- Linux x64
- macOS Intel
- Universal

ad, you accept this
Use.

s

cov-viewer

Report Abuse

3.3 SuperBOL Studio OSS vs SuperBOL Studio Pack

L'extension SuperBOL Studio OSS contient les fonctionnalités de base de SuperBOL Studio. de base de SuperBOL Studio. OSS signifie « Open-Source Software », car l'extension est open-source : la plupart des sources sont disponibles publiquement sur Github (ce qui permet de reconstruire le fichier .vsix avec des fonctionnalités publiques), et les les autres sources sont fournies aux clients de SuperBOL dans le cadre du « contrat » d'abonnement.

L'extension SuperBOL Studio Pack est une « méta extension » : elle sélectionne un ensemble d'extensions à installer en même temps. Ceci est généralement utile si si vous voulez bénéficier de notre choix d'autres extensions que nous avons trouvées à utiliser avec SuperBOL Studio OSS.

Actuellement, SuperBOL Studio Pack comprend:

- SuperBOL Studio OSS, bien sûr
- tamasfe.even-better-toml : un mode pour éditer les fichiers TOML, qui peut typiquement être utilisé pour éditer le fichier `superbol.toml` pour une configuration avancée. Voir https://marketplace.visualstudio.com/items?itemName=tamasfe.even-better-toml

3.4 Liens vers les pages des extensions

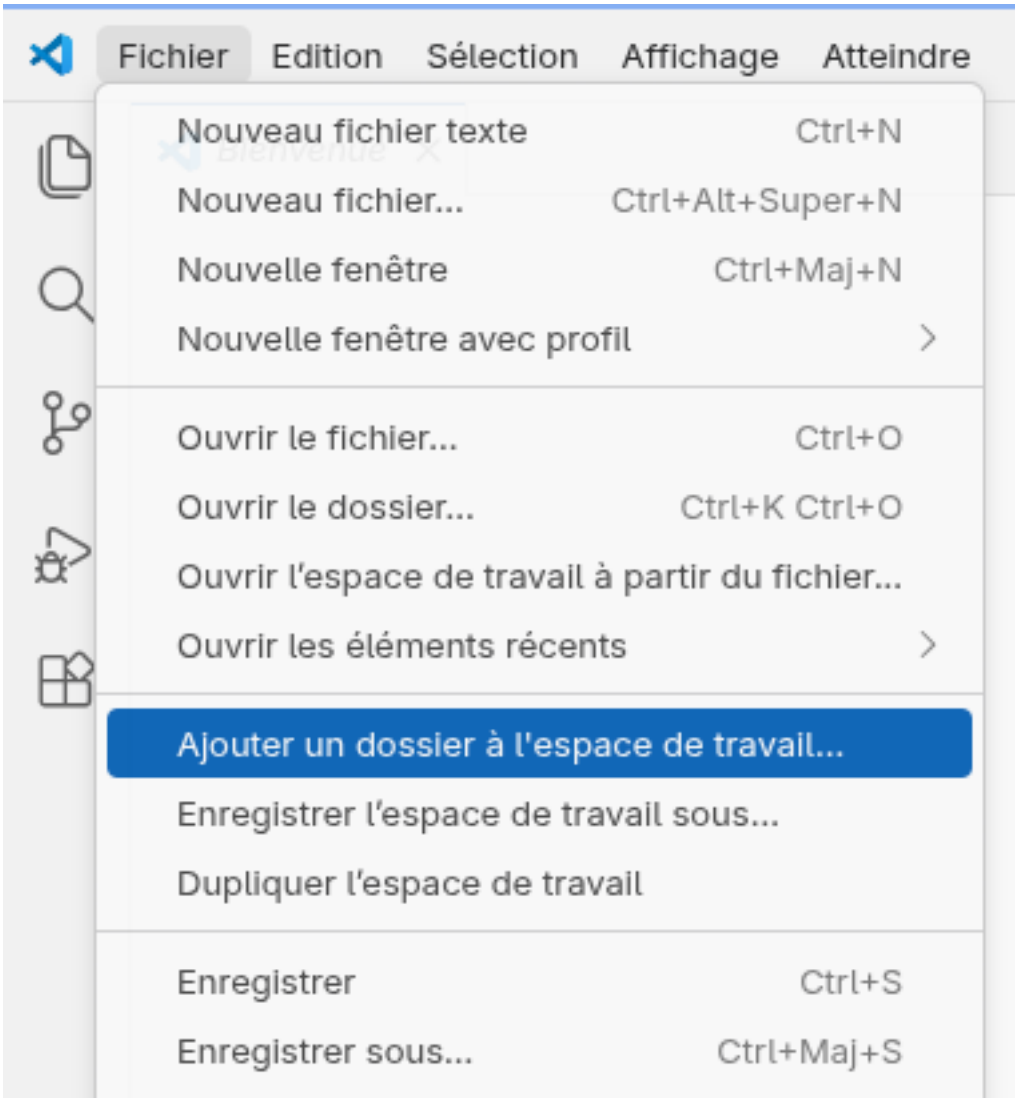
- Descriptions sur Open-VSX : https://open-vsx.org/
  - SuperBOL Studio OSS : https://open-vsx.org/extension/OCamlPro/SuperBOL
  - SuperBOL Studio Pack : https://open-vsx.org/extension/OCamlPro/SuperBOL-studio-pack
- Descriptions sur Vscode Marketplace :
  - SuperBOL Studio OSS : https://marketplace.visualstudio.com/items?itemName=OCamlPro.SuperBOL
  - SuperBOL Studio Pack : https://marketplace.visualstudio.com/items?itemName=OCamlPro.SuperBOL-studio-pack
- Sources sur Github : https://github.com/ocamlpro/superbol-studio-oss

4 Configuration de l'espace de travail

SuperBOL Studio peut être utilisé à la fois sur des fichiers COBOL uniques, ou sur un projet COBOL complet, généralement appelé espace de travail dans le langage VSCode.

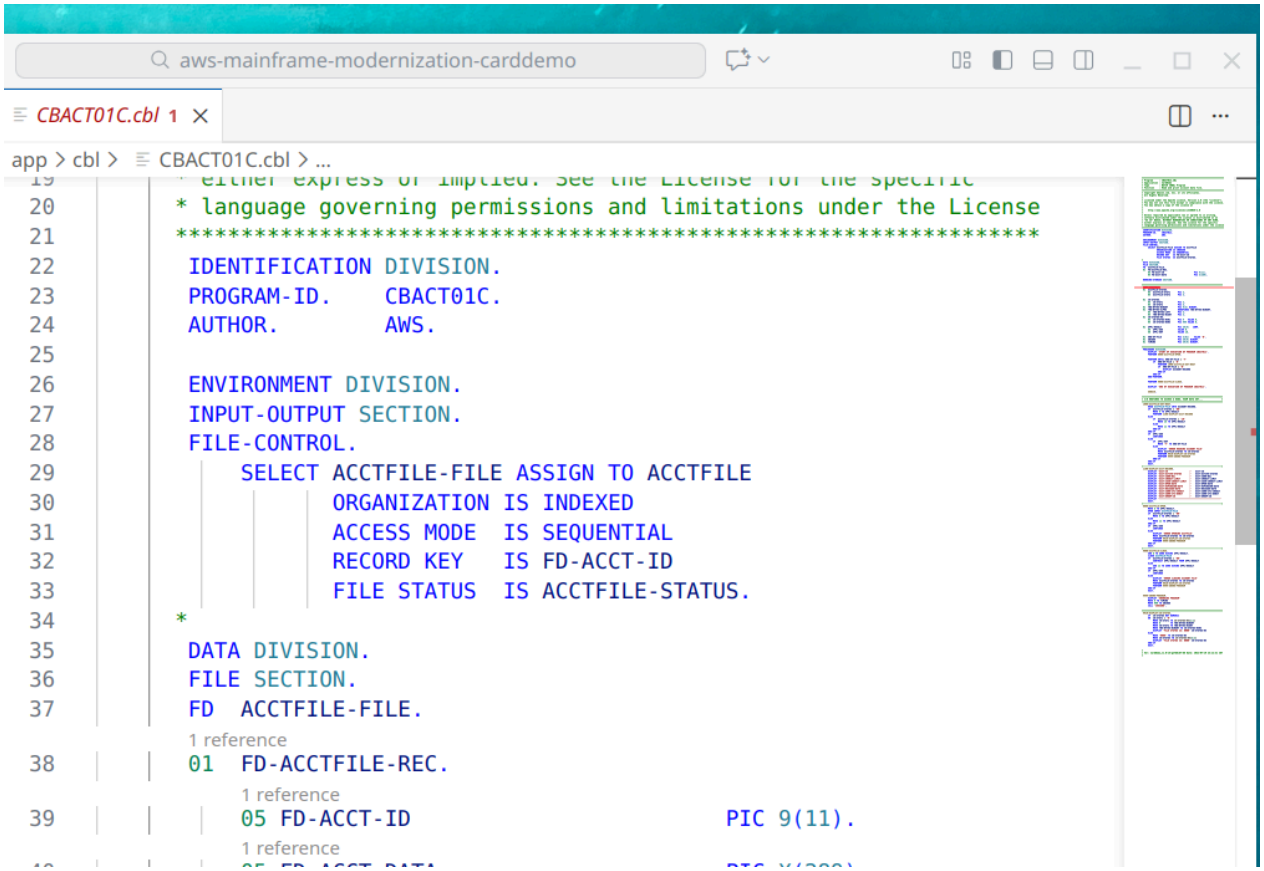
4.1 Modifier un projet existant en tant qu'espace de travail

Pour commencer à utiliser l'extension sur un projet existant, ouvrez son dossier dans VSCode (`Fichier > Ajouter un dossier à l'espace de travail...`).



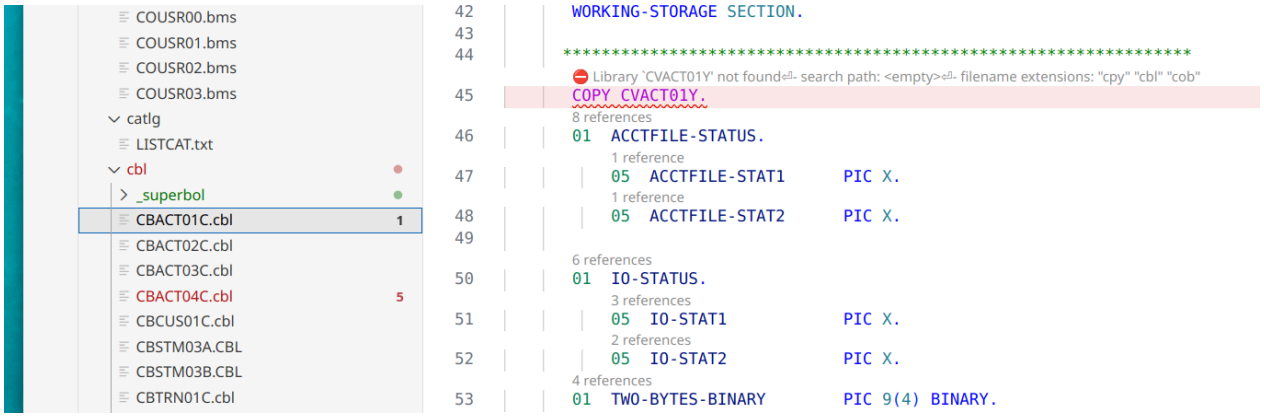
L'extension démarre automatiquement lorsque le dossier contient des fichiers avec les extensions COBOL habituelles (`.cob`, `.cbl`, `.cpy`, `.cbx`).

Une fois l'extension lancée, les sources COBOL devraient apparaître avec colorisation :



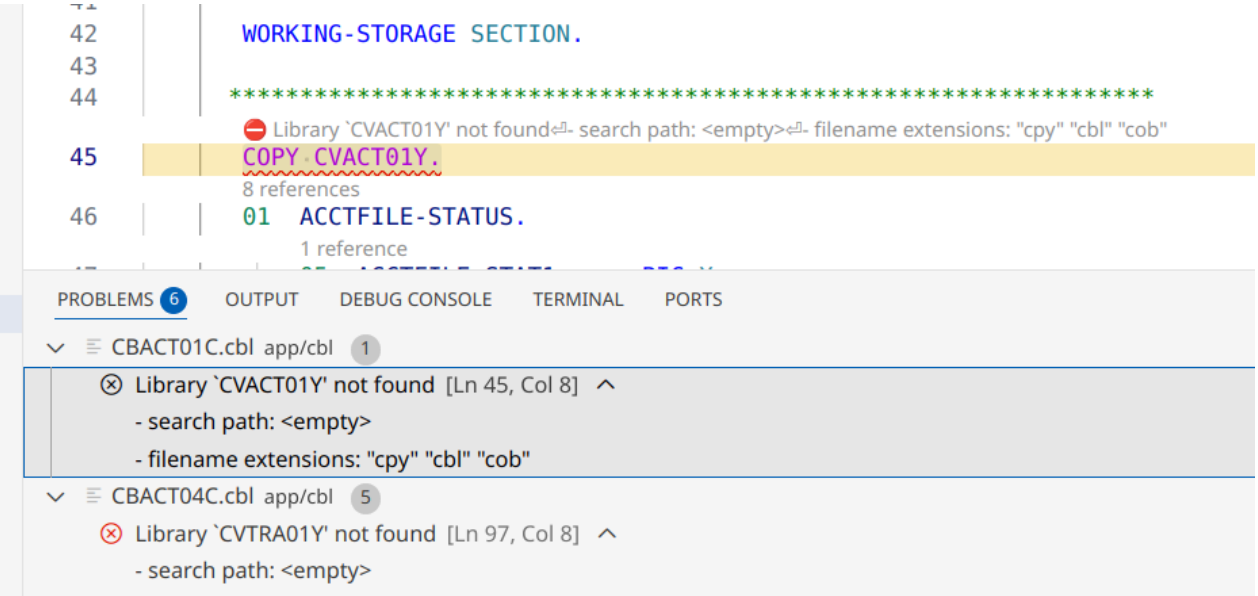
L'extension tentera d'analyser les sources et de résoudre les instructions COPY. En cas d'échec, elle affichera les problèmes :

- le nombre de problèmes trouvés après le nom du fichier dans l'explorateur de fichiers
- ces problèmes sont affichés directement dans les sources



Notez que, dans notre cas, nous utilisons l'extension VSCode « Error Lens » pour intégrer les erreurs dans le code source. Il existe de nombreuses extensions similaires pour améliorer la visualisation des problèmes.

Vous pouvez également passer directement au premier problème, en utilisant Ctrl+Maj+M :



Ces problèmes proviennent généralement d'une mauvaise configuration de l'espace de travail pour le dialecte COBOL utilisé par ce projet. Nous devons donc le configurer pour que l'extension fonctionne correctement. Nous verrons cela dans la section suivante.

4.2 Configuration de l'extension pour le projet COBOL

Ouvrez les paramètres (`Fichier > Préférences > Paramètres`, ou Ctrl+,), et commencez à taper « superbol... ». Vous verrez un écran qui ressemble à.. :

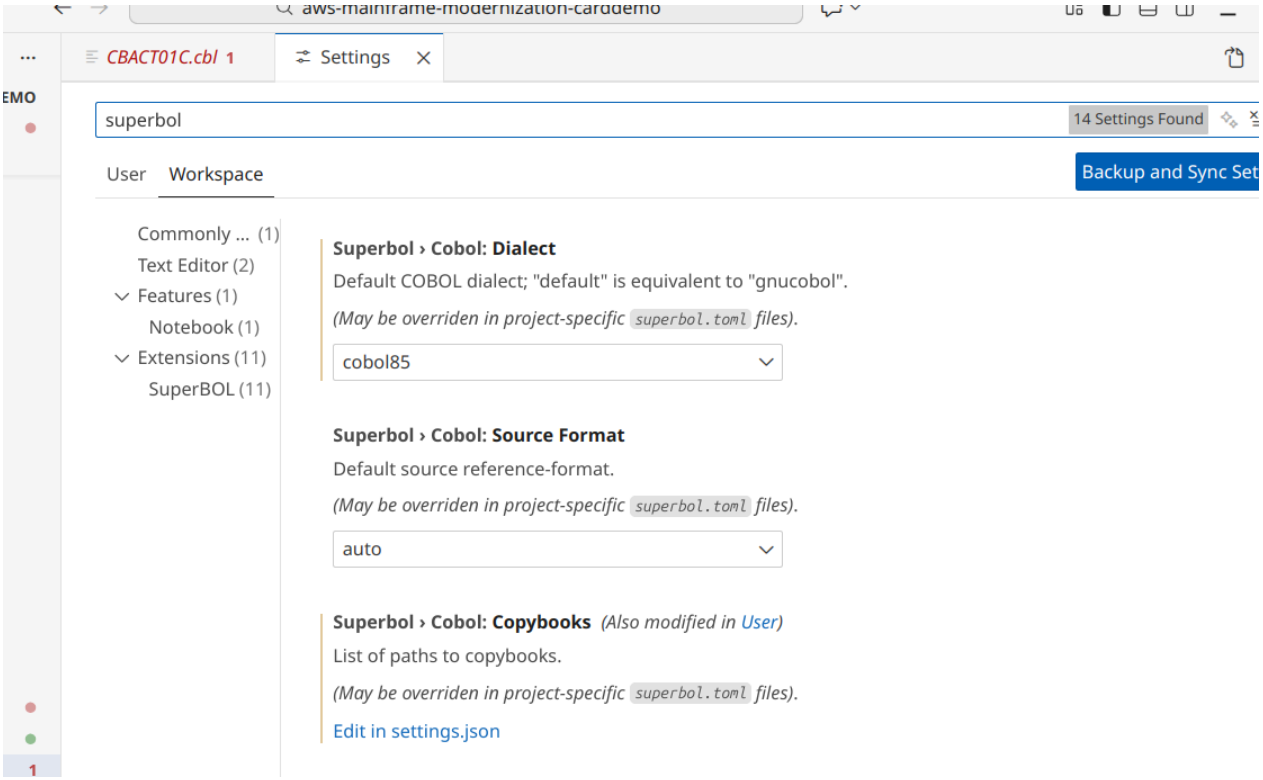


Figure 1. – Paramètres SuperBOL

Attention, il existe plusieurs façons de configurer les paramètres :

- Paramètres **Utilisateur** : il s'agit des paramètres par défaut lorsque l'espace de travail n'est pas configuré. Sur l'image ci-dessus, ils se trouvent dans l'onglet de gauche « Utilisateur ». Ils sont stockés dans le profil de l'utilisateur, typiquement dans `$HOME/.config/Code/User/settings.json`.
- Paramètres **Espace de travail** : ce sont les paramètres spécifiques à cet espace de travail. **Ce sont ceux que vous voulez probablement configurer maintenant**. Sur l'image ci-dessus, ils se trouvent dans l'onglet de droite « Espace de travail ». Ils sont stockés dans un fichier `.vscode/settings.json` à la racine de l'espace de travail.
- De plus, SuperBOL Studio peut utiliser son propre fichier de configuration pour certains paramètres, situé dans un fichier **superbol.toml** à la racine du projet. Ce fichier est généralement utile si vous prévoyez d'utiliser les outils de SuperBOL en ligne de commande dans ce projet. VSCode vous avertit parfois si vous essayez de configurer des paramètres en interne dans VSCode alors qu'il existe un fichier `superbol.toml`.

Dans notre exemple, le format de la source a été correctement déduit. Si ce n'est pas votre cas, vous pouvez modifier l'option **Source Format** :

- Le format de source de référence par défaut `"superbol.cobol.sourceFormat"`. Lorsque `auto` est sélectionné, ce qui est le cas par défaut, SuperBOL (et GnuCOBOL) essaiera automatiquement de deviner si la source est en format `free` ou `fixed`. Les autres formats de source doivent être configurés explicitement.

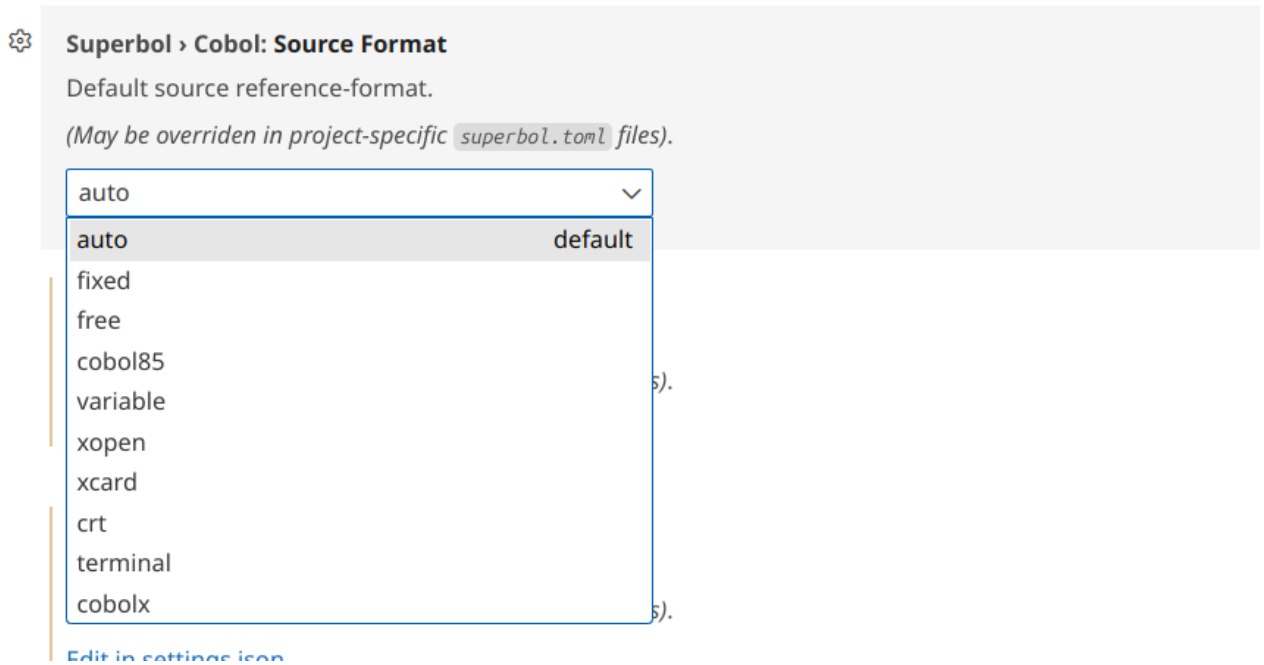
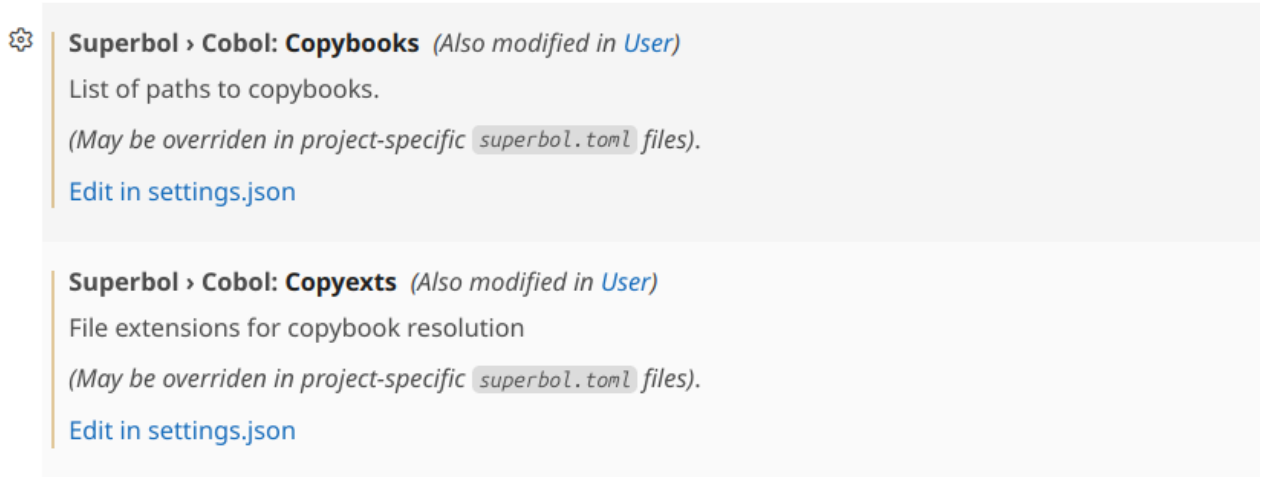


Figure 2. – Paramètres SuperBOL

Dans notre cas, l'extension se plaint de ne pas trouver certains copybook. Deux paramètres sont disponibles pour contrôler la manière dont l'extension recherche les copybooks :



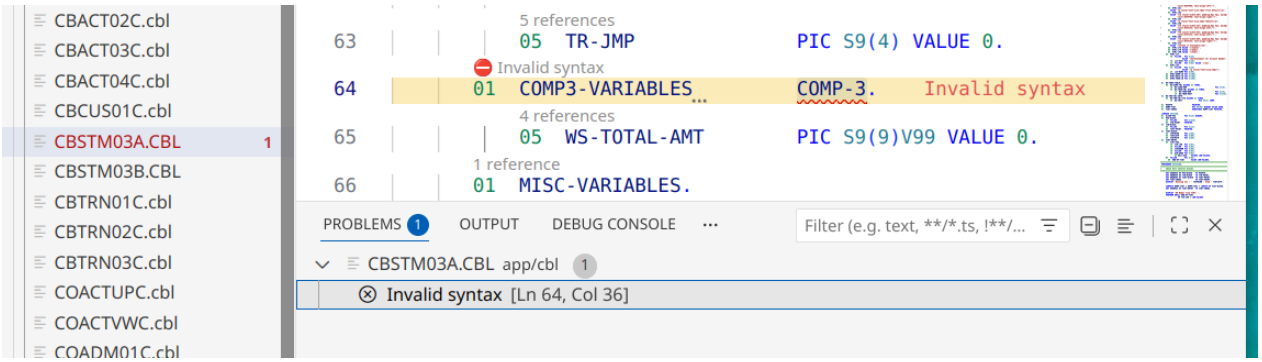
Les deux options doivent être configurées en JSON en modifiant le fichier `settings.json`, vous devez donc cliquer sur « Modifier dans settings.json ».

Figure 3. – Modification des chemins d'accès aux copybooks dans `.vscode/settings.json`

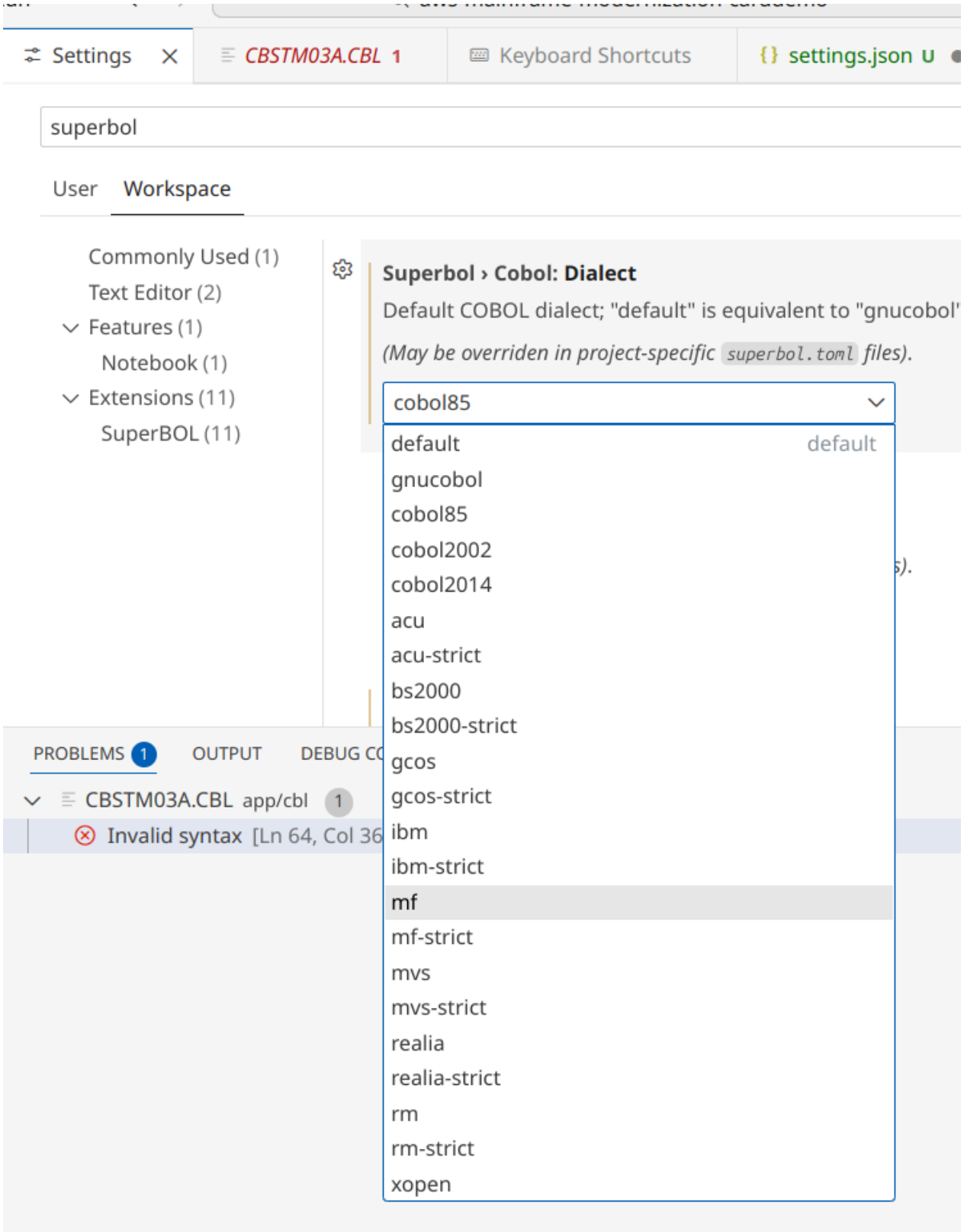
- Le chemin vers les copybooks **« superbol.cobol.copybooks »** :
  Il s'agit d'une liste, où chaque entrée décrit un élément du chemin de recherche où vos vos copybooks seront recherchés. Chaque entrée doit contenir un champ **« dir »**, et un champ facultatif **« file-relative »**.
  - Si le répertoire que vous souhaitez spécifier (où se trouvent les copybooks) est un chemin commençant à la racine de l'espace de travail, ou un chemin absolu, vous pouvez ignorer le champ `file-relative`, ou le mettre à `false`.
  - Si le répertoire que vous voulez spécifier est un chemin relatif au répertoire contenant la source COBOL spécifique l'instruction `COPY`, vous devez définir le champ `file-relative`. à `true`.
- Les extensions du livre de copie **« superbol.cobol.copyexts »** :
  Pour configurer ce paramètre, vous devrez sélectionner `Editer dans settings.json`. Il s'agit d'une liste, où chaque entrée décrit une extension de fichier si le nom du copybook ne peut pas être trouvé tel quel (par exemple `COPY "mycpy.lib"`). Dans SuperBOL, l'option `default` correspond à l'option GnuCOBOL par défaut, qui est `["cpy", "cbl", "cob"]` (recherche en majuscules d'abord, puis en minuscules).

N'oubliez pas d'enregistrer le fichier (avec Ctrl+S). Dès qu'il est sauvegardé, l'extension redémarre le LSP pour analyser de nouveau les sources les sources COBOL.

Maintenant, l'extension nous donne un autre type d'erreurs, parce qu'elle n'a pas pu comprendre la syntaxe :



- Le dialecte COBOL utilisé dans le projet `"superbol.cobol.dialect"` :
  Dans SuperBOL, le dialecte par défaut correspond au dialecte par défaut de GnuCOBOL, qui supporte de nombreuses fonctionnalités de dialectes tels que `COBOL2014`, `IBM`, `MicroFocus` (`mf`), ou `GCOS`


Settings
CBSTM03A.CBL 1
Keyboard Shortcuts
settings.json U
superbol
User
Workspace
Commonly Used (1)
Text Editor (2)
Features (1)
Notebook (1)
Extensions (11)
SuperBOL (11)
Superbol › Cobol: Dialect
Default COBOL dialect; "default" is equivalent to "gnucobol"
(May be overriden in project-specific superbol.toml files).
cobol85
default
default
gnucobol
cobol85
cobol2002
cobol2014
acu
acu-strict
bs2000
bs2000-strict
gcos
gcos-strict
ibm
ibm-strict
mf
mf-strict
mvs
mvs-strict
realia
realia-strict
rm
rm-strict
xopen
PROBLEMS 1
OUTPUT
CBSTM03A.CBL app/cbl 1
Invalid syntax [Ln 64, Col 36

4.3 Diagnostic syntaxique

> **i** Note
>
> Les vérifications syntaxiques effectuées par SuperBOL Studio couvrent actuellement le dialecte `COBOL85` et certaines constructions plus récentes supportés par GnuCOBOL. Le rapport de ces diagnostics est actuellement désactivé pour les dialectes autres que `COBOL85` afin d'éviter d'induire en erreur les développeurs avec de faux diagnostics d'erreurs de syntaxe.
>
> Les diagnostics peuvent être réactivés pour tous les dialectes en selectionnant l'option « Force Syntax Diagnostics » dans les paramètres de configuration de SuperBOL.

4.4 Collaborer avec d'autres développeurs

À ce stade, les paramètres de votre projet sont stockés et gérés par VSCode. Cependant, vous pouvez envisager de collaborer avec des développeurs qui n'utilisent pas cet éditeur. Par exemple, ils pourraient vouloir utiliser SuperBOL avec GNU/Emacs. Dans ce cas, nous vous conseillons de laisser SuperBOL Studio stocker la configuration dans un fichier `superbol.toml` qui sera situé à la racine du projet.

Vous pouvez demander à l'extension d'écrire la configuration de votre projet actuel dans un `superbol.toml` en entrant dans la palette de commandes (`Vue > Palette de commandes...`, ou appuyez sur Ctrl+Maj+P), puis en sélectionnant la commande commande `SuperBOL : Write Project Configuration`.

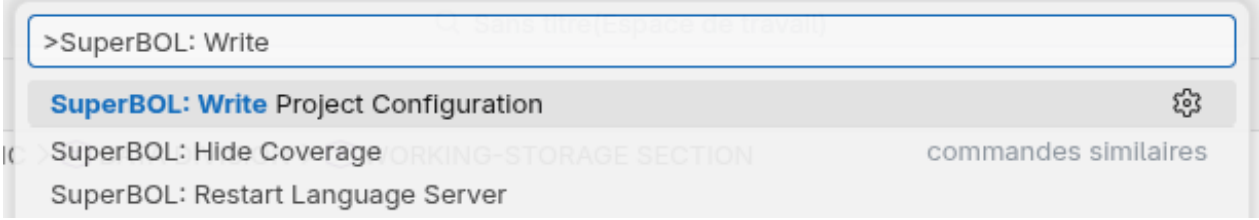


Figure 4. – SuperBOL: Write Project Configuration

Ceci sauvegardera un fichier `superbol.toml` à la racine de chaque répertoire de projet actuellement ouvert. Un tel fichier ne contiendra aucun paramètre specifique à l'utilisateur, vous pouvez donc le mettre sous contrôle de source en toute sécurité. Des extensions dédiées à l'édition de fichiers TOML, telles que `tamasfe.even-better-toml`, fournissent le même niveau d'assistance que lorsque vous éditez `.vscode/settings.json`.

> **Astuce**
>
> Installer le `OCamlPro.SuperBOL-studio-pack` pour obtenir SuperBOL Studio et `tamasfe.even-better-toml`. au total.

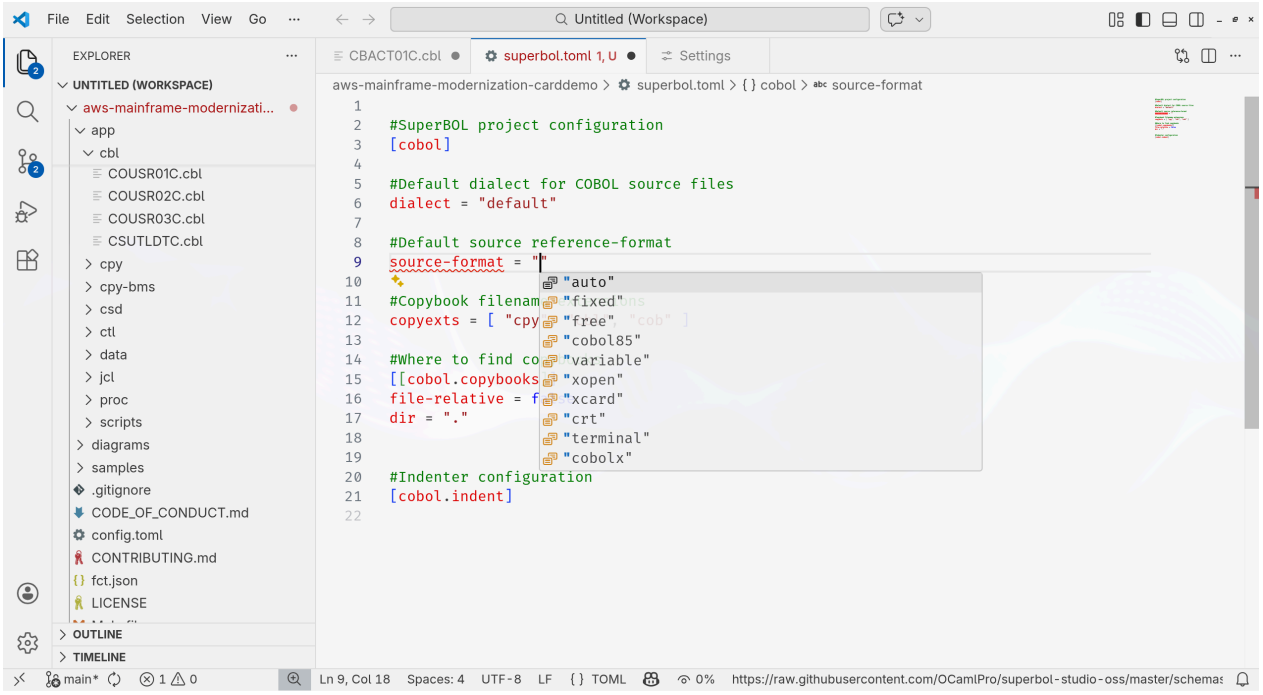


Figure 5. – Edition de `superbol.toml`

5 Fonctionnalités de navigation

5.1 Structure et fil d'Ariane

SuperBOL fournit une vue d'ensemble de votre programme dès que vous l'ouvrez, que vous pouvez utiliser pour naviguer vers des sections ou des symboles spécifiques (données, paragraphes, etc.) Les mêmes informations sont également affichées dans la fenêtre « Fil d'Ariane » (appelé « breadcrumbs » en anglais) qui se trouve généralement au-dessus de la zone d'édition du texte.

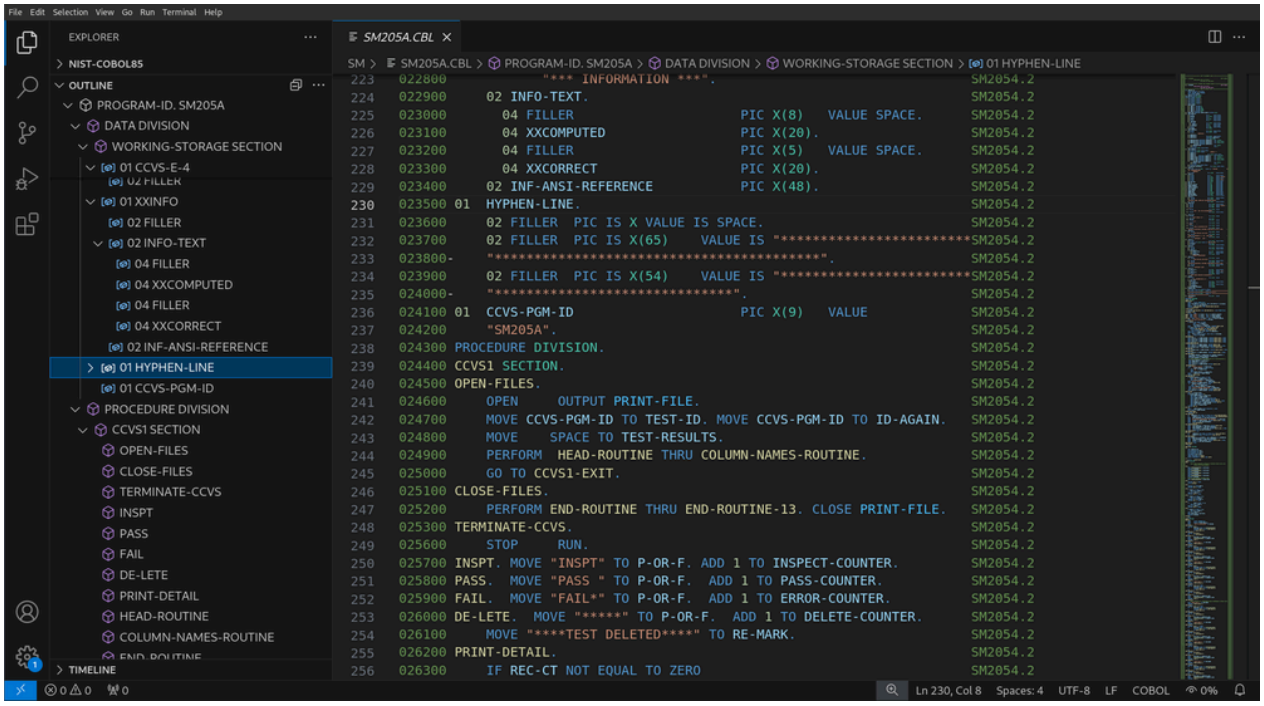


Figure 6. – Structure et fil d'Ariane

5.2 Aller au symbole

Les symboles affichés dans les vues Structure et Fil d'Ariane peuvent également faire l'objet d'une recherche et d'un saut en appuyant sur Ctrl+Maj+O.

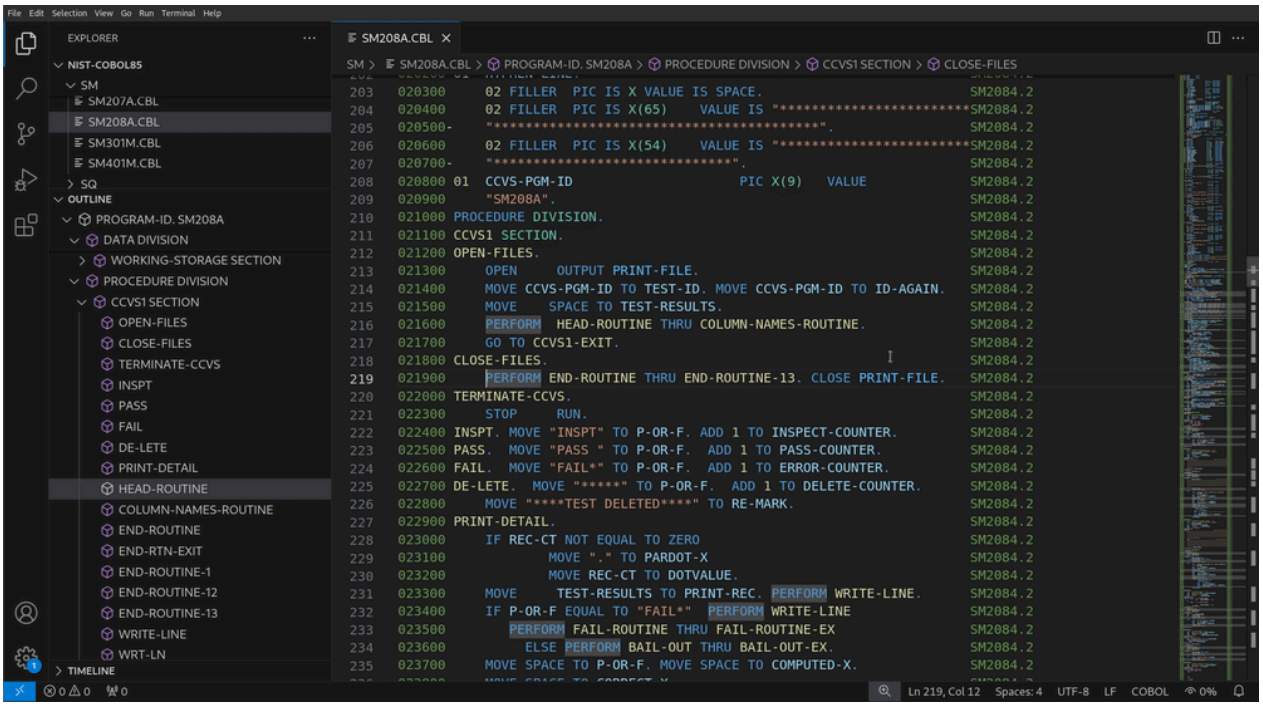


Figure 7. – Aller au symbole

5.3 Aller à la définition

Lorsque vous souhaitez localiser la définition d'un nom d'élément de données dans votre code source, positionnez votre curseur sur ce nom, cliquez avec le bouton droit de la souris et sélectionnez `Atteindre la définition` (ou appuyez sur F12).

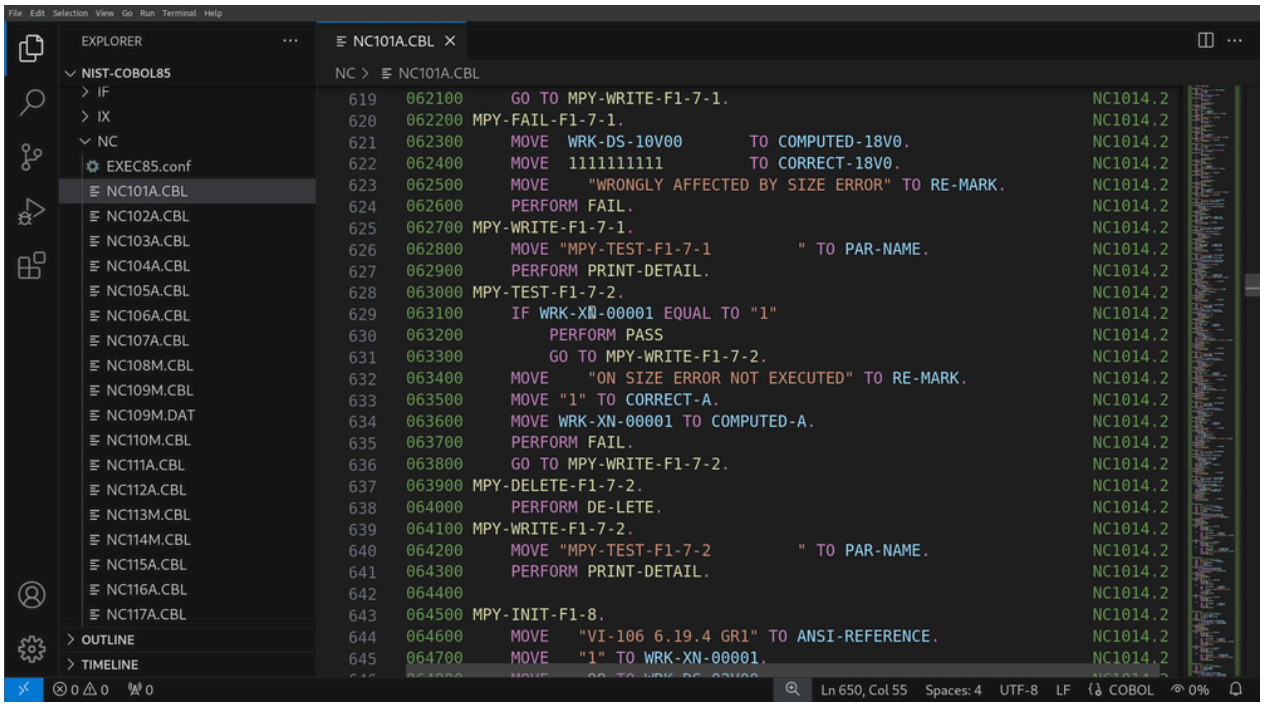


Figure 8. – Atteindre la définition

> **Note**
>
> (Limitation temporaire)
>
> Pour l'instant, les définitions qui appartiennent aux `COMMUNICATION SECTION`, `REPORT SECTION` ou `SCREEN SECTION` de la `DATA DIVISION` sont ignorées par l'extension. En outre, certaines définitions dans les blocs SQL intégrés (`EXEC SQL`) ne sont pas encore prises en compte.

5.4 Aperçu de la définition

Pour avoir un aperçu de la définition d'un tel élément de données, vous pouvez positionner le curseur sur son nom, cliquer avec le bouton droit de la souris et sélectionner `Aperçu > Aperçu de la défintion` (ou appuyer sur la touche Ctrl+Maj+F10). Vous obtiendrez alors une vue de l'emplacement de la définition correspondante, y compris si elle se trouve dans un copybook.

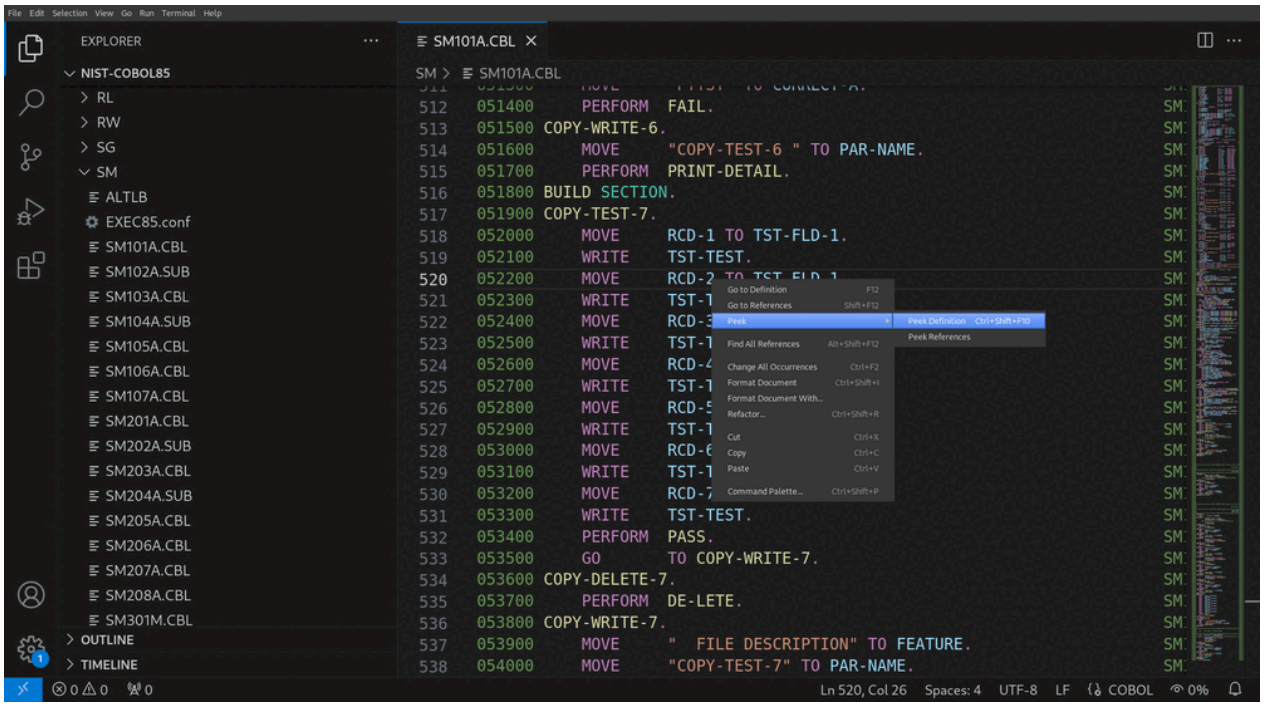


Figure 9. – Aperçu de la définition dans un copybook

5.5 Aller aux références

Si vous souhaitez obtenir une liste de toutes les références à un élément de données nommé, cliquez avec le bouton droit de la souris et sélectionnez `Atteindre les références` (ou appuyez sur Maj+F12). Vous verrez alors l'emplacement de chaque référence à cet élément.

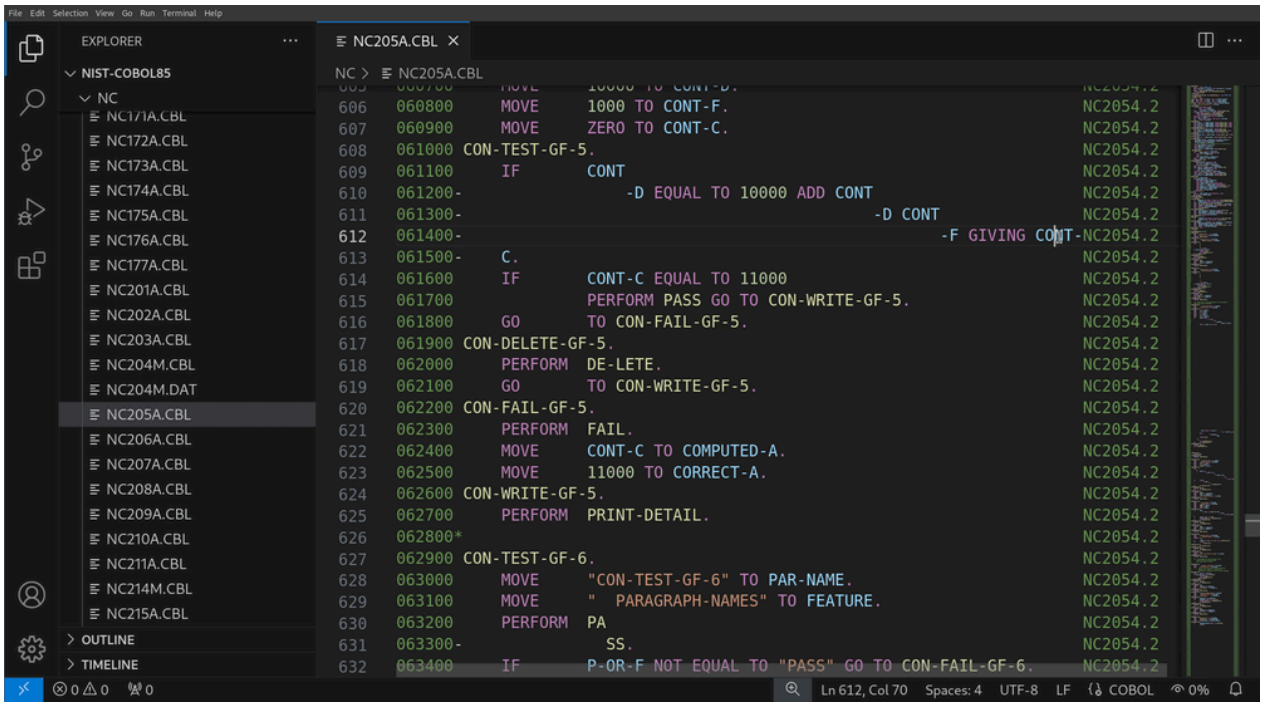


Figure 10. – Aller aux références

> **Note**
>
> (Limitation temporaire)
>
> Les limitations mentionnées dans Aller à la définition s'appliquent également pour les références.

5.6 Information de référence

L'extension présente des informations de référence en ligne au-dessus des définitions des éléments de données et des éléments de la `PROCEDURE DIVISION`.

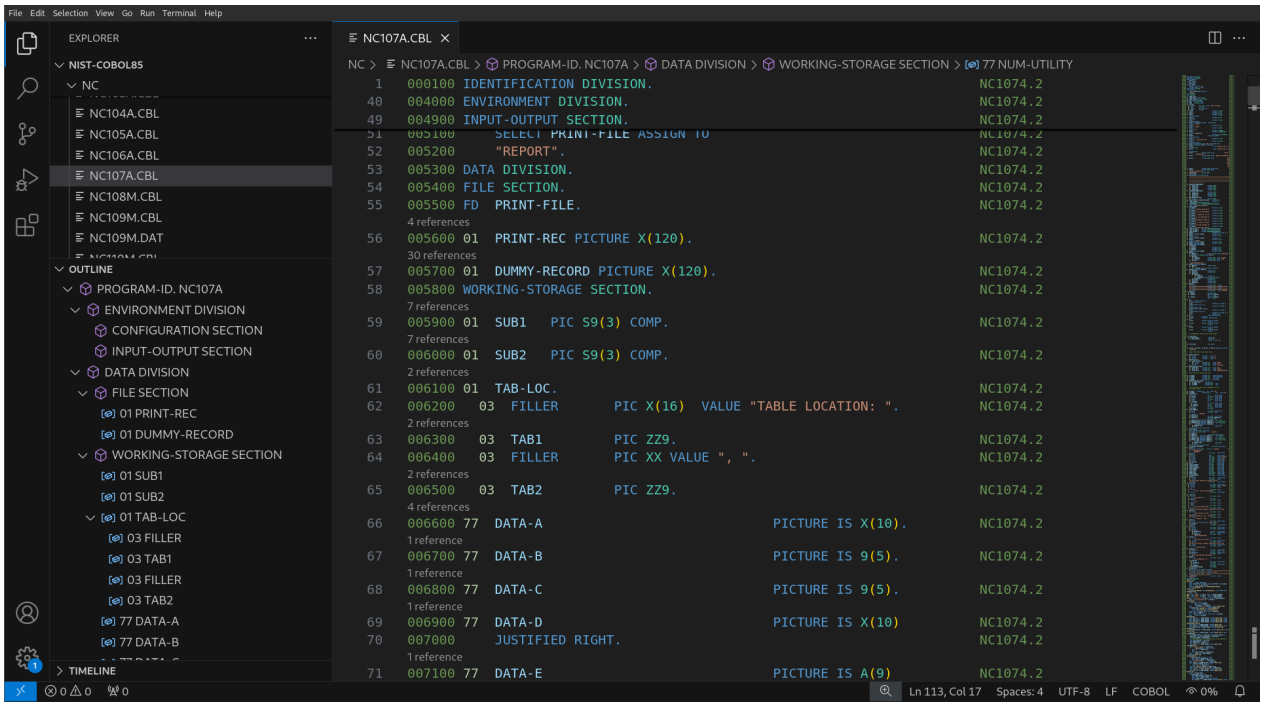


Figure 11. – Informations de référence

> **Note**
>
> Les mêmes limitations que pour Aller à la définition s'appliquent.

> **Astuce**
>
> Cette fonctionnalité peut être activée ou désactivée en réglant le paramètre de configuration `"editor.codeLens"` (vous pouvez taper Ctrl+, et ensuite `codelens` pour changer ce paramètre.)

5.7 Survoler pour afficher les copybooks

Vous êtes-vous déjà demandé ce qui se cachait derrière une directive `COPY` ? Il vous suffit de placer votre curseur sur une telle instruction, et vous verrez apparaître le contenu du copybook.

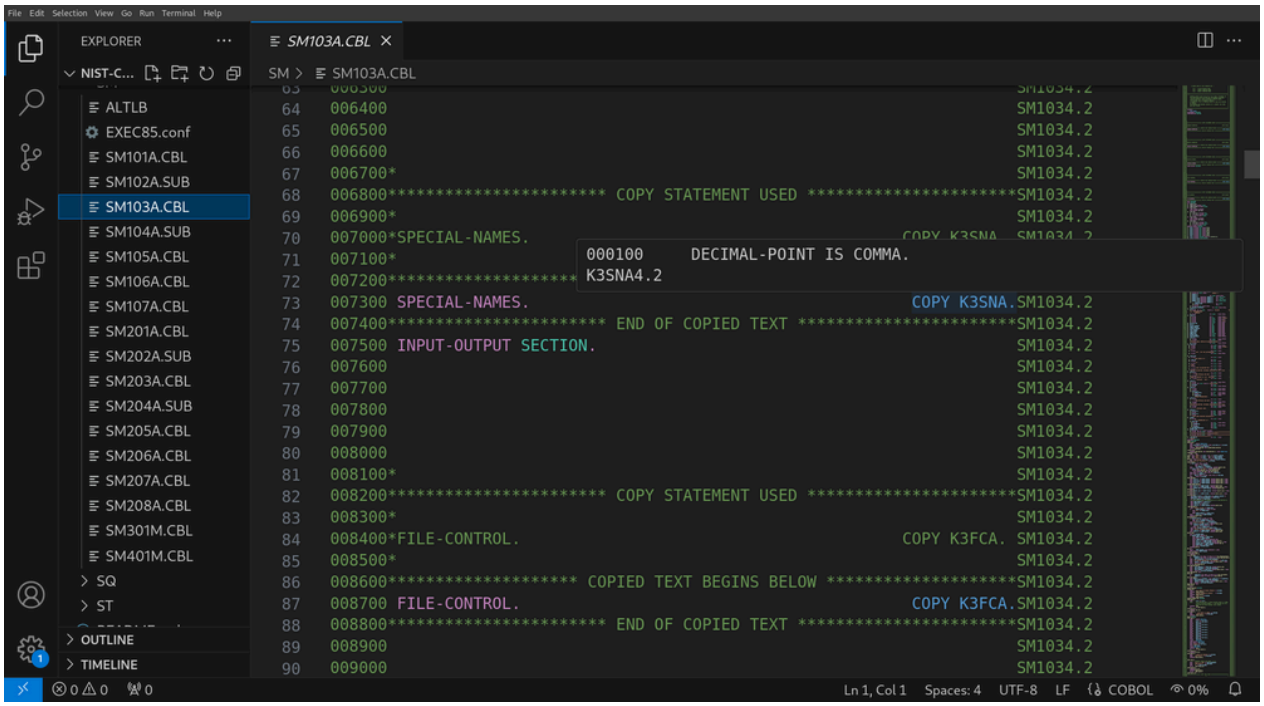


Figure 12. – Survoler un `COPY`

Pour vous assurer que cela fonctionne correctement, vérifiez vos paramètres `"superbol.cobol.copybooks"` et `"superbol.cobol.copyexts"`.

5.8 Passer la souris pour afficher les remplacements du texte source

De plus, vous pouvez voir le texte source qui résulte d'un remplacement par une directive `REPLACE` de la même manière.

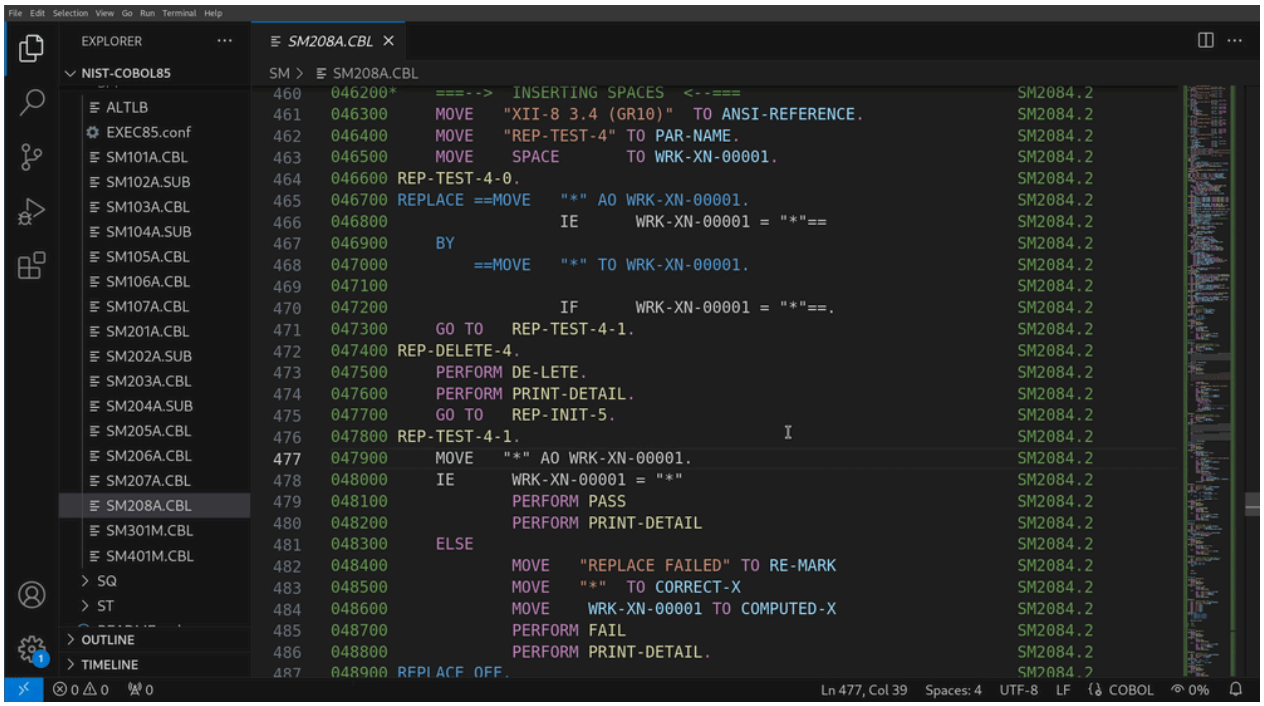


Figure 13. – Survoler un REPLACE

6 Fonctionnalités d'édition

6.1 IntelliSense (auto-complétion)

Lors de l'édition d'un programme, vous pouvez appuyer sur Ctrl+Space pour obtenir des suggestions de mots-clés valides, de mots définis par l'utilisateur (noms d'éléments de données ou de paragraphes) et même des phrases COBOL complètes. Sélectionnez une option à l'aide des flèches de direction et appuyez sur Entrée pour insérer la suggestion sélectionnée.

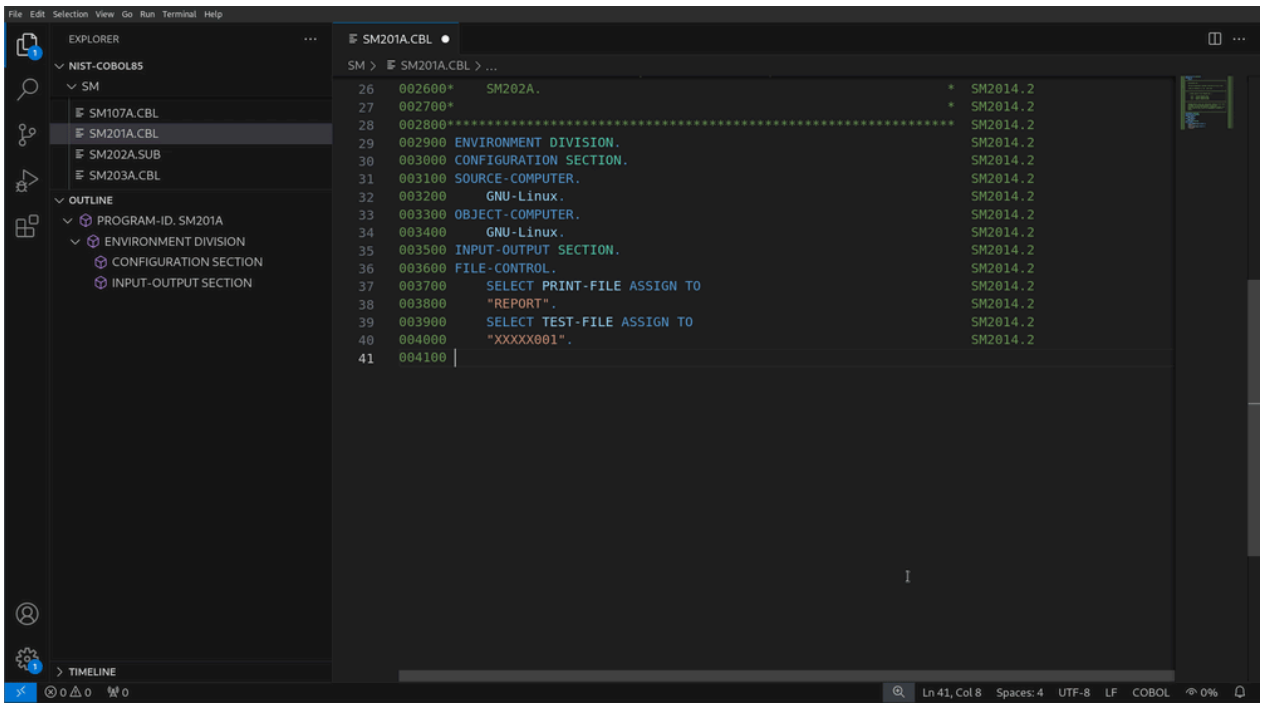


Figure 14. – Phrase IntelliSense

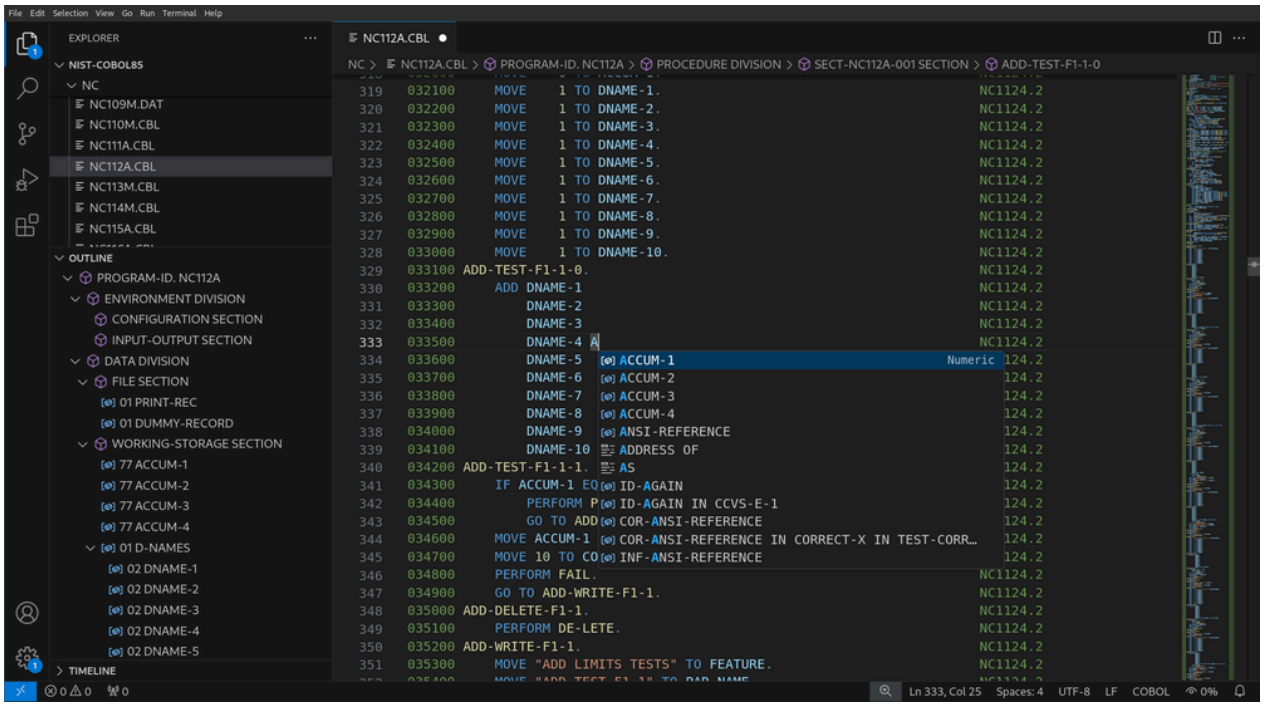


Figure 15. – Élément de données IntelliSense

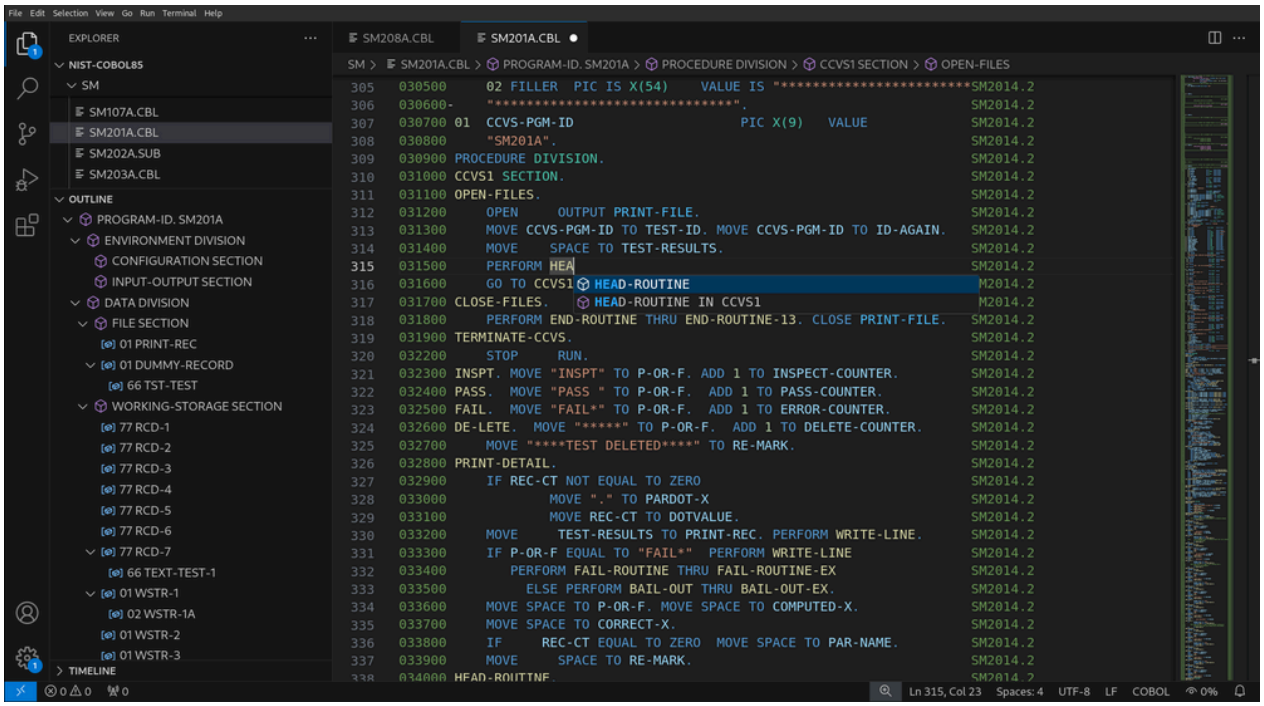


Figure 16. – Paragraphe IntelliSense

(Limitation temporaire)

Les suggestions de mots définis par l'utilisateur ne peuvent pas comprendre des symboles qui sont définis dans les `COMMUNICATION SECTION`, `REPORT SECTION` ou `SCREEN SECTION` de la `DATA DIVISION`. Bien que les mots définis par l'utilisateur qui apparaissent dans les copybooks sont également suggérés, les variables ou les phrases liées au préprocesseur ne le sont pas.

6.2 Renommer les éléments de données, les sections et les paragraphes

Vous pouvez renommer n'importe quelle donnée en appuyant sur F2 lorsque votre curseur est positionné sur l'une de ses références. L'extension vous vous avertira si une référence à l'élément renommé apparaît dans un copybook, dans ce cas le renommage de chaque référence n'est pas effectué).

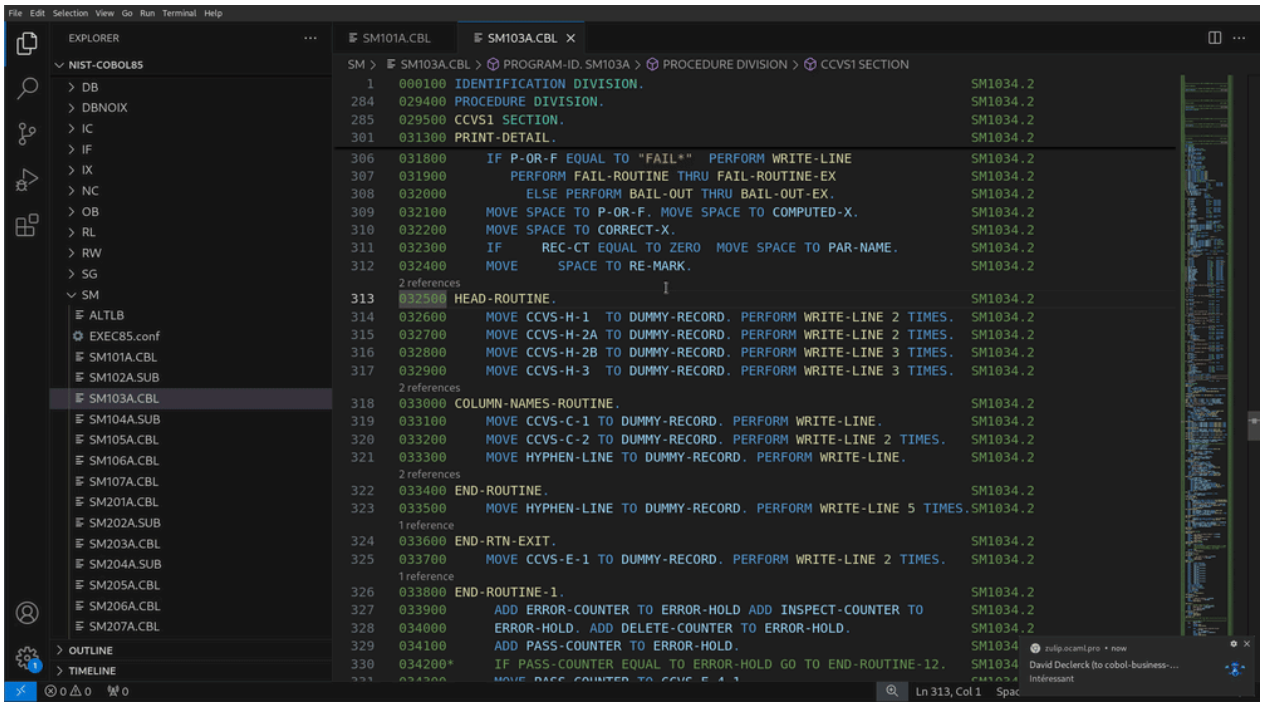


Figure 17. – Renommer le symbole

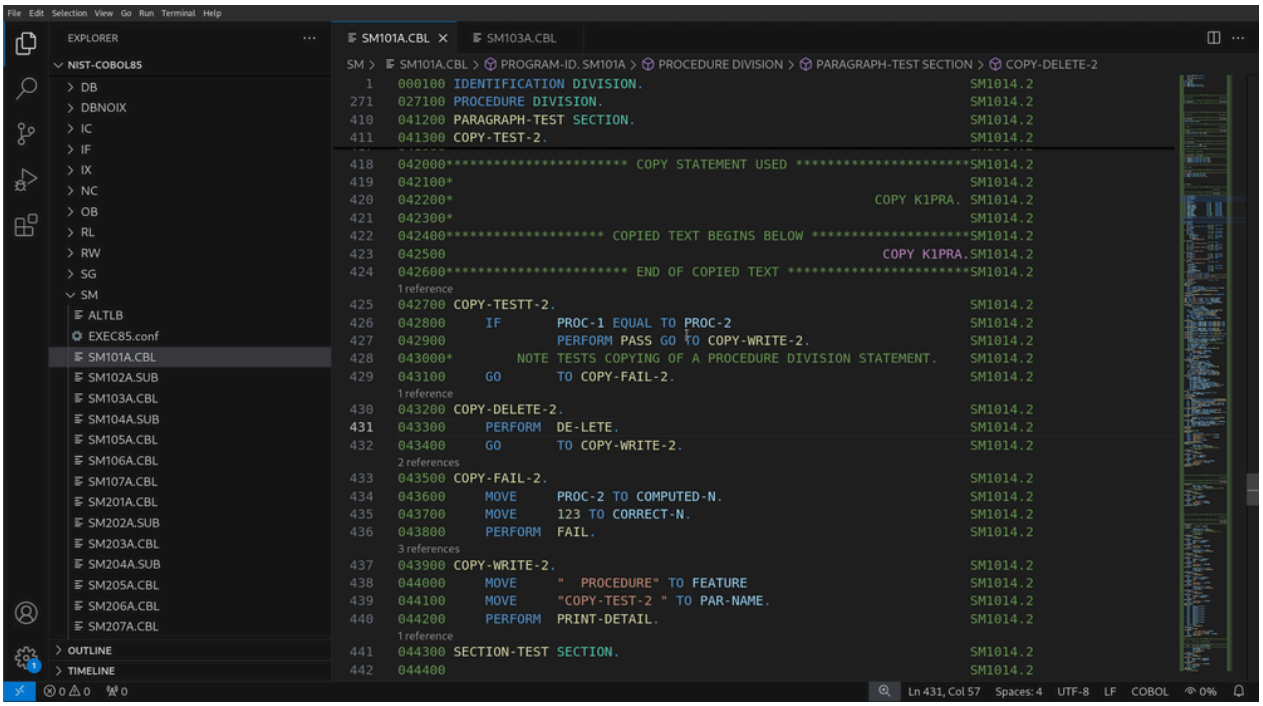


Figure 18. – Renommer le symbole dans un copybook

Les sections et les paragraphes de la `PROCEDURE DIVISION` peuvent également être renommés de la même manière.

Les mêmes limitations que pour la fonction Aller aux références s'appliquent à cette fonction.

7 Visualisation du flot de contrôle

7.1 Exploration du flot de contrôle

La navigation dans une représentation graphique du flot de contrôle d'un programme COBOL s'avère inestimable lorsqu'il s'agit de déchiffrer sa logique générale. Pour ce faire, ouvrez la palette de commandes (ou tapez Ctrl+Maj+P), et sélectionnez `SuperBOL : Show Control-flow` (vous pouvez également faire un clic droit et sélectionner `Show Control-flow` dans le menu). On vous présente alors une liste de portions de programme à considérer (soit le programme entier, soit des sections individuelles) : sélectionnez un élément pour voir le CFG (*Control-Flow Graph*, soit Graphique de flot de contrôle) correspondant.

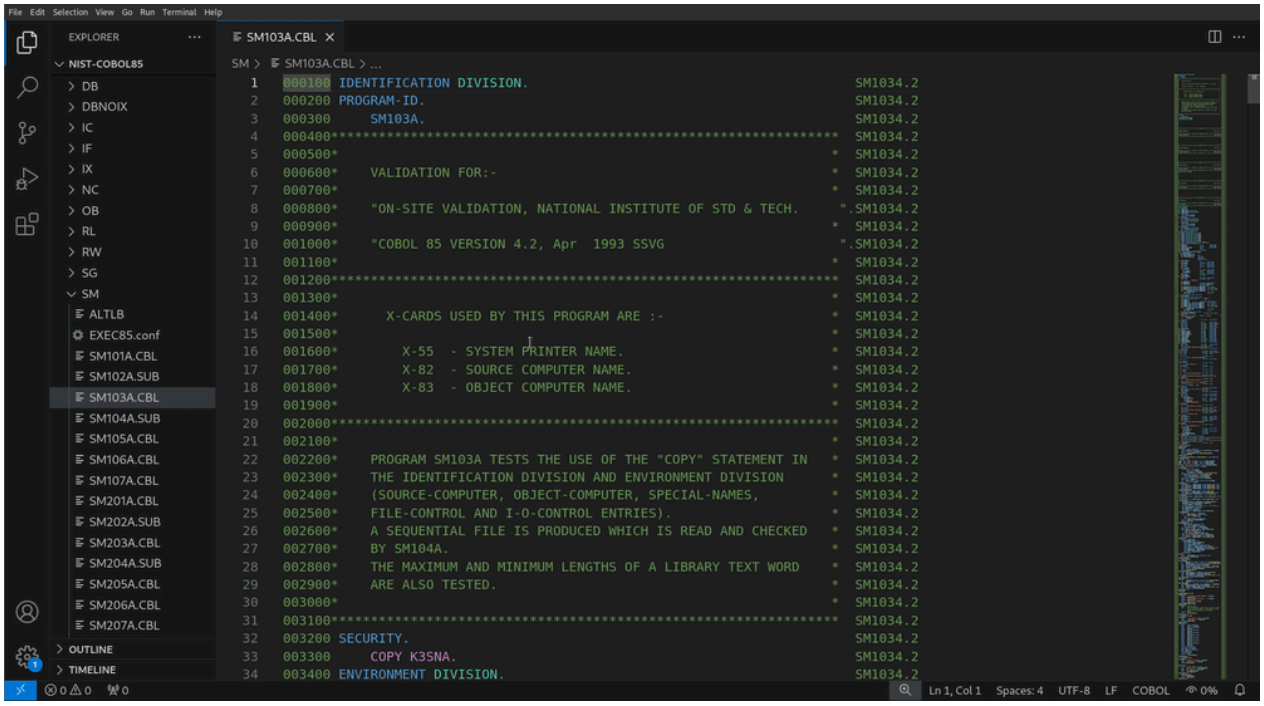


Figure 19. – Affichage du CFG

Différents paramètres permettent d'ajuster le rendu des CFG.

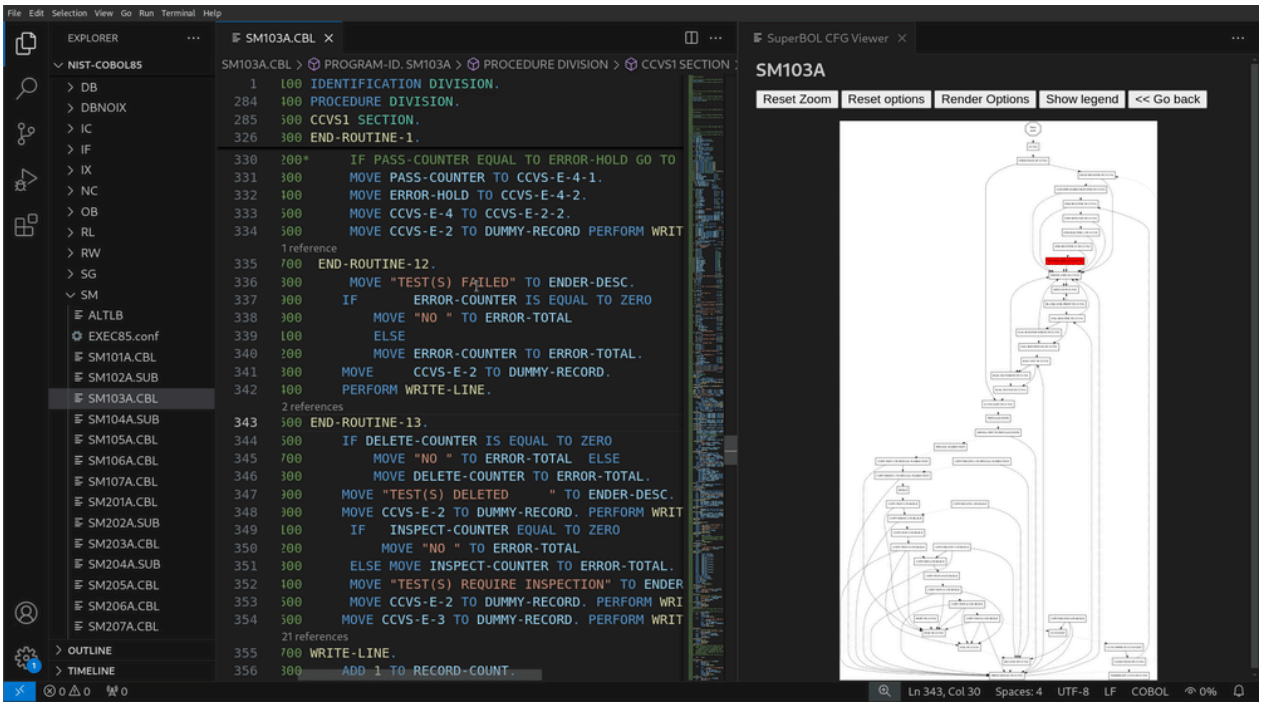


Figure 20. – Simplification du CFG

7.2 CFG en tant que diagramme d'arc

Une représentation des GFC sous forme de diagrammes en arc est également disponible. Dans cette represantion les sections et les paragraphes nommés sont disposés verticalement, et les arcs indiquent la direction du flot de contrôle entre eux.

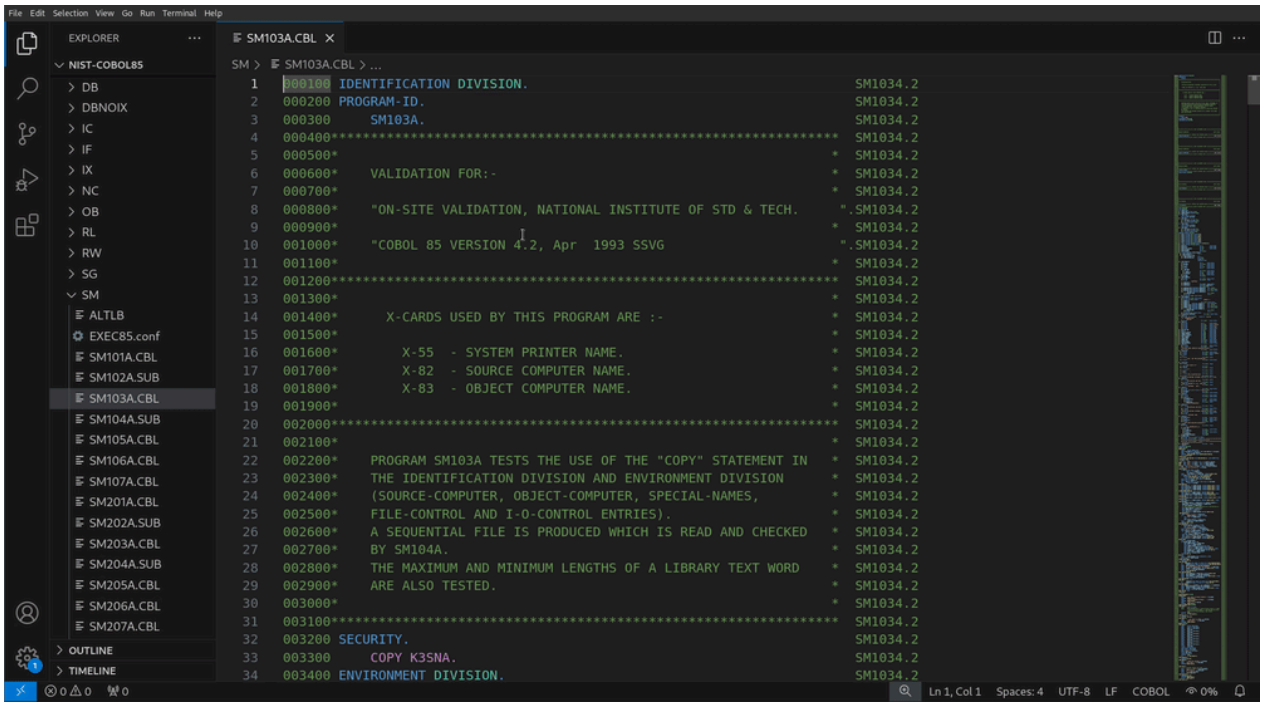


Figure 21. – L’explorateur CFG comme diagramme en arc

8 Compilation des programmes

SuperBOL Studio vous permet de compiler directement vos programmes COBOL avec GnuCOBOL.

> **Note**
>
> Nous recommandons qu'une version de GnuCOBOL qui soit au moins aussi récente que la version 3.2 sur le système exécutant VSCode. Les fonctions de débogage et de couverture supposent respectivement que gdb et gcov sont installés.
>
> Sur les systèmes Windows, les utilisateurs peuvent utiliser des installateurs dédiés qui sont disponibles ici. Les utilisateurs de Linux peuvent se fier à leur gestionnaire de paquets préféré et installer `gnucobol`.

Toutes les actions décrites ci-dessous doivent commencer dans un fichier COBOL *ouvert*, afin que VSCode sache que vos configurations sont faites pour le langage COBOL, avec l'extension SuperBOL.

8.1 Compilation simple

Si vous souhaitez compiler vos programmes COBOL avec les options par défaut de `cobc` :

1. appuyez sur Ctrl+Maj+B, ou cliquez sur l'option de menu `Terminal > Exécuter la tâche de build...`
2. plusieurs options de compilation par défaut vous seront présentées, sélectionnez celle que vous souhaitez exécuter :

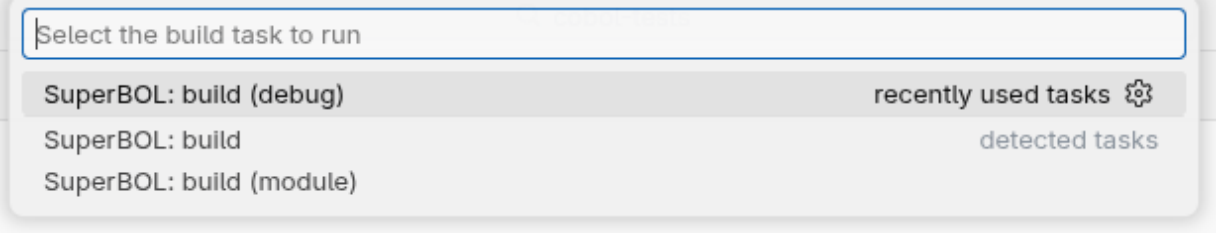


Figure 22. – Options de compilation par défaut

Ces options ont les effets suivants, en supposant que vous ayez ouvert `PROG.cob` :

- `SuperBOL : build` : cette option compile votre programme en tant qu'**exécutable**, **sans** informations de débogage, et exécute la commande `cobc` suivante:

```
cobc -x -fformat=auto -std=default -ext cpy -ext cbl -ext cob PROG.cob -I .
```

- `SuperBOL : build (debug)` : cette option compile votre programme comme **exécutable**, **avec** des informations de débogage, il exécutera la commande `cob` suivante:

```
cobc -x -ftraceall -g -fformat=auto -std=default -ext cpy -ext cbl -ext cob
PROG.cob -I .
```

- `SuperBOL : build (module)` : cette option construira votre programme comme un **module**, **sans** informations de débogage, il exécutera la commande `cobc` suivante:

```
cobc -m -fformat=auto -std=default -ext cpy -ext cbl -ext cob PROG.cob -I .
```

8.2 Paramètres ayant une influence sur les tâches de compilation

Certains paramètres de l'**Utilisateur** ou de l'**Espace de travail** ont une influence sur les commandes exécutées par les tâches de compilation. Vous pouvez consulter la liste des options disponibles pour en savoir plus à leur sujet.

Les options qui ont un effet sur les tâches de compilation sont les suivantes :

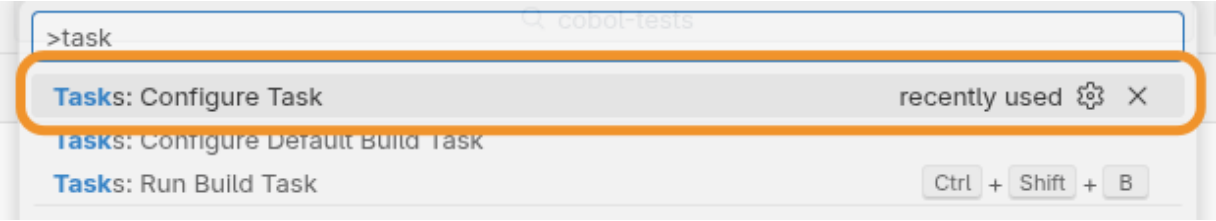
- **superbol.cobol.dialect** : définit l'option `-std` de `cobc`
- **superbol.cobol.sourceFormat** : définit l'option `-fsource-format` de `cobc`.
- **superbol.cobol.copybooks** : ajoute les indicateurs `-I` à `cobc`, un pour chaque valeur du tableau.
- **superbol.cobcPath** : définit le chemin `cobc` qui sera exécuté pour compiler vos programmes.

8.3 Configuration d'une tâche de compilation

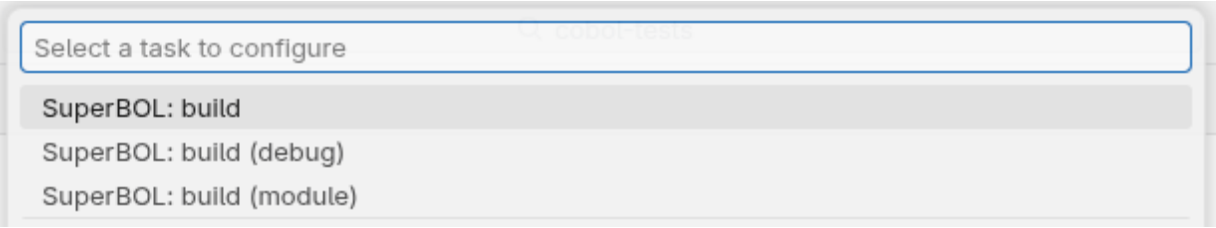
Si vous souhaitez effectuer des tâches de compilation plus complexes, vous pouvez créer votre propre tâche de compilation à partir de n'importe quelle tâche de compilation fournie par SuperBOL.

Pour ce faire, vous pouvez:

1. ouvrir la palette de commandes VSCode avec Ctrl+Maj+P
2. sélectionner l'option `Tâches : Configurer une tâche` :



3. la liste des tâches de compilation par défaut de SuperBOL devrait alors s'afficher :



4. à partir de cette liste, vous pouvez sélectionner n'importe laquelle de ces tâches, mais pour réduire le nombre d'options à éditer, vous pouvez choisir celle qui correspond le mieux aux actions de compilation que vous souhaitez exécuter. Si vous n'êtes pas sûr, vous pouvez sélectionner `SuperBOL : build`.

VSCode créera alors automatiquement un fichier `.vscode/tasks.json`, rempli avec la tâche que vous avez sélectionnée. Si le fichier existe déjà, la tâche que vous avez sélectionnée sera alors ajoutée à la liste des tâches.

En supposant que vous ayez sélectionné la tâche `SuperBOL : build (debug)` plus tôt, vous devriez voir apparaître contenu suivant :

```
{
    "version": "2.0.0",
    "tasks": [
      {
        "type": "superbol",
        "forDebug": true,
        "forCoverage": false,
        "executable": true,
        "extraArgs": [],
        "problemMatcher": [
          "$gnucobol",
          "$gnucobol-warning",
          "$gnucobol-error",
          "$gnucobol-note"
        ],
        "group": "build",
```

```
        "label": "SuperBOL: build (debug)"
      }
    ]
}
```

Les différents champs disponibles dans cette configuration de tâche de compilation sont également disponibles dans les autres configurations de compilation fournies par SuperBOL, mais avec des valeurs différentes. Voici une description de la signification de tous les champs dans cette configuration :

- `"type"` : Ce champ indique à VSCode que cette configuration doit être utilisée avec l'extension SuperBOL.
- `"forDebug"` : si ce champ vaut `true`, le programme sera compilé avec des informations de de débogage, sinon le programme sera compilé comme un programme prêt à la production.
- `"forCoverage"` : si la valeur est `true`, alors le programme sera compilé avec des informations de couverture, sinon le programme sera compilé comme un programme normal.
- `"executable"` : si la valeur est `true`, alors le programme sera exécutable, sinon le programme sera compilé comme un module à exécuter avec un lanceur COBOL (habituellement `cobcrun` pour les modules GnuCOBOL).
- `"extraArgs"` : vous pouvez ajouter des options à ce tableau qui seront passées au compilateur GnuCOBOL pour être utilisées lors de la compilation de votre programme.
- `"problemMatcher"` : ce champ ne doit pas être édité, il permet à VSCode de mettre en évidence les sources d'une erreur de compilation.
- `"group"` : ce champ ne doit pas non plus être édité, il informe VSCode que cette tâche est utilisée pour compiler un programme.
- `"label"` : ce champ contient le nom de la tâche de compilation, il peut être utilisé dans d'autres fichiers de configurations de VSCode, vous pouvez donc lui donner le nom qui vous convient le mieux.
- `"cobcPath"` : ce champ est **caché** par défaut, mais vous pouvez mettre le chemin vers votre compilateur `cobc` si celui-ci n'est pas directement disponible dans votre *PATH* système.

Avec tous ces champs, vous devriez être en mesure de configurer votre tâche de compilation pour qu'elle réponde au mieux à vos besoins.

8.4 Compiler avec une configuration personnalisée

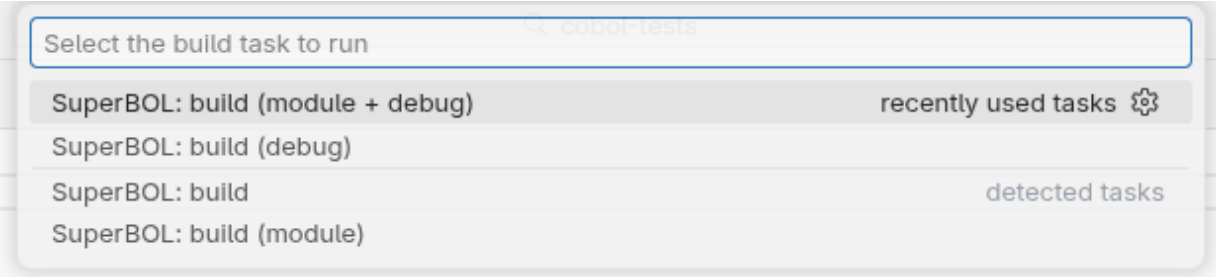
Une fois que vous avez configuré votre tâche de compilation, vous pouvez l'utiliser pour compiler vos programmes COBOL.

Supposons par exemple que vous ayez défini la tâche suivante, qui compile un module GnuCOBOL avec les informations de débogage :

```
{
    "version": "2.0.0",
    "tasks": [
      {
        "type": "superbol",
        "forDebug": true,
        "forCoverage": false,
        "executable": false,
        "extraArgs": [],
        "problemMatcher": [
          "$gnucobol",
          "$gnucobol-warning",
          "$gnucobol-error",
          "$gnucobol-note"
        ],
        "group": "build",
        "label": "SuperBOL: build (module + debug)"
      }
    ]
}
```

Pour utiliser votre tâche de compilation personnalisée :

1. ouvrez le programme COBOL que vous souhaitez compiler
2. appuyez sur Ctrl+Maj+B
3. sélectionnez l'étiquette de votre tâche de compilation personnalisée (`SuperBOL : build (module + debug)` dans notre exemple) :



8.5 Sélection d'une tâche de compilation par défaut

Si vous avez une tâche que vous souhaitez voir exécutée comme tâche de compilation par défaut :

1. appuyez sur Ctrl+Maj+P
2. sélectionnez `Tâches : Configurer la tâche de génération par défaut`
3. sélectionnez la tâche que vous souhaitez exécuter en tant que tâche de compilation par défaut
4. ouvrez les sources de votre programme COBOL
5. appuyez sur Ctrl+Maj+B
6. VSCode exécutera automatiquement la tâche que vous avez sélectionnée par défaut.

8.6 Débogage de vos programmes

Une fois vos programmes créés, vous pouvez consulter Débogage de programmes pour savoir comment vous pouvez déboguer vos programmes COBOL.

8.7 Exemples de configurations personnalisées

Voici quelques exemples de configurations que vous pourriez vouloir mettre en place, dans votre fichier `.vscode/tasks.json`, vous devez les ajouter au tableau `"tasks"`.

8.7.1 Compilation d'un module GnuCOBOL avec les informations de débogage

```
{
    "type": "superbol",
    "forDebug": true, // génère les informations de debug
    "forCoverage": false,
    "executable": false, // compile un module
    "extraArgs": [],
    "problemMatcher": [
      "$gnucobol",
      "$gnucobol-warning",
      "$gnucobol-error",
      "$gnucobol-note"
    ],
    "group": "build",
    "label": "SuperBOL: build (module + debug)"
}
```

8.7.2 Compilation d'un exécutable GnuCOBOL pour vérifier la couverture

```
{
    "type": "superbol",
    "forDebug": false,
```

```
    "forCoverage": true, // génère les informations de couverture
    "executable": true, // compile un exécutable
    "extraArgs": [],
    "problemMatcher": [
      "$gnucobol",
      "$gnucobol-warning",
      "$gnucobol-error",
      "$gnucobol-note"
    ],
    "group": "build",
    "label": "SuperBOL: build (coverage)"
}
```

8.7.3 Envoi d’options supplémentaires à `cobc`

```
{
    "type": "superbol",
    "forDebug": false,
    "forCoverage": false,
    "executable": true,
    "extraArgs": [
      "-fec=all" // pass `-fec=all` to `cobc`
    ],
    "problemMatcher": [
      "$gnucobol",
      "$gnucobol-warning",
      "$gnucobol-error",
      "$gnucobol-note"
    ],
    "group": "build",
    "label": "SuperBOL: build (all exceptions)"
}
```

9 Débogage des programmes

9.1 Débogage

Pour déboguer un programme COBOL, vous devez d'abord exécuter une *tâche de compilation* avec les options de débogage appropriées. Une fois que cela est fait, vous pouvez lancer le programme compilé dans une session de débogage.

Vous pouvez en savoir plus sur les tâches de compilation de SuperBOL dans le chapitre Compilation des programmes.

> **Note**
>
> Nous recommandons qu'une version de GnuCOBOL au moins aussi récente que la version 3.2 soit disponible sur le système exécutant VSCode. Les fonctions de débogage et de couverture supposent respectivement que gdb et gcov sont installés.
>
> Sur les systèmes Windows, les utilisateurs peuvent utiliser des installateurs dédiés qui sont disponibles ici. Les utilisateurs de Linux peuvent se fier à leur gestionnaire de paquets préféré et installer `gnucobol`.

9.2 Lancement du programme compilé pour le débogage

Si nécessaire, vous pouvez placer un point d'arrêt sur les instructions (ou les titres de paragraphes dans la `PROCEDURE DIVISION`) en cliquant sur le point rouge qui apparaît lorsque vous passez le curseur sur la marge de gauche (ou avec F9). Cliquez sur le point rouge ou appuyez à nouveau sur F9 pour supprimer un point d'arrêt. Ensuite, pour lancer le programme en mode débogage, sélectionnez `Exécuter > Démarrer le débogage` (F5). Cela exécutera votre programme jusqu'à ce qu'un premier point d'arrêt soit atteint, ou jusqu'à la fin.

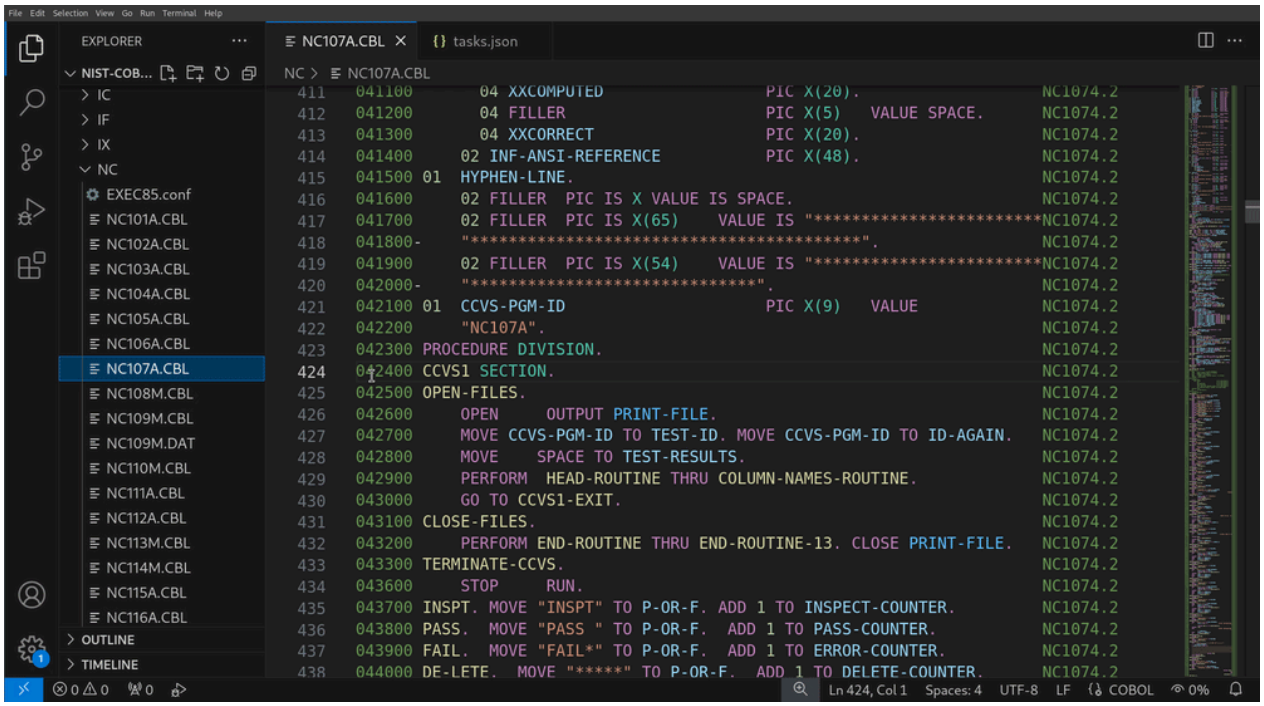


Figure 23. – Démarrer le débogage

Une fois arrêté sur un point d'arrêt, vous pouvez examiner les valeurs des données du programme en utilisant le panneau « VARIABLES » sur la gauche.

Appuyez sur F10 pour passer à l'instruction suivante, ou F5 à nouveau pour continuer jusqu'au prochain point d'arrêt ou jusqu'à la fin de l'instruction.

9.3 Configurations de débogage SuperBOL

SuperBOL peut gérer plusieurs scénarios de débogage, tels que le lancement d'un programme pour le déboguer, l'attachement à un processus en cours d'exécution ou même l'attachement à un processus distant pour le déboguer.

Afin de sélectionner le scénario que SuperBOL doit exécuter, vous devez fournir une liste de configurations stockées dans le fichier `.vscode/launch.json`. Ce fichier peut être généré en suivant les étapes suivantes :

1. Ouvrez le panneau `Run and Debug` (Exécuter et Déboguer) (Ctrl+Maj+D)

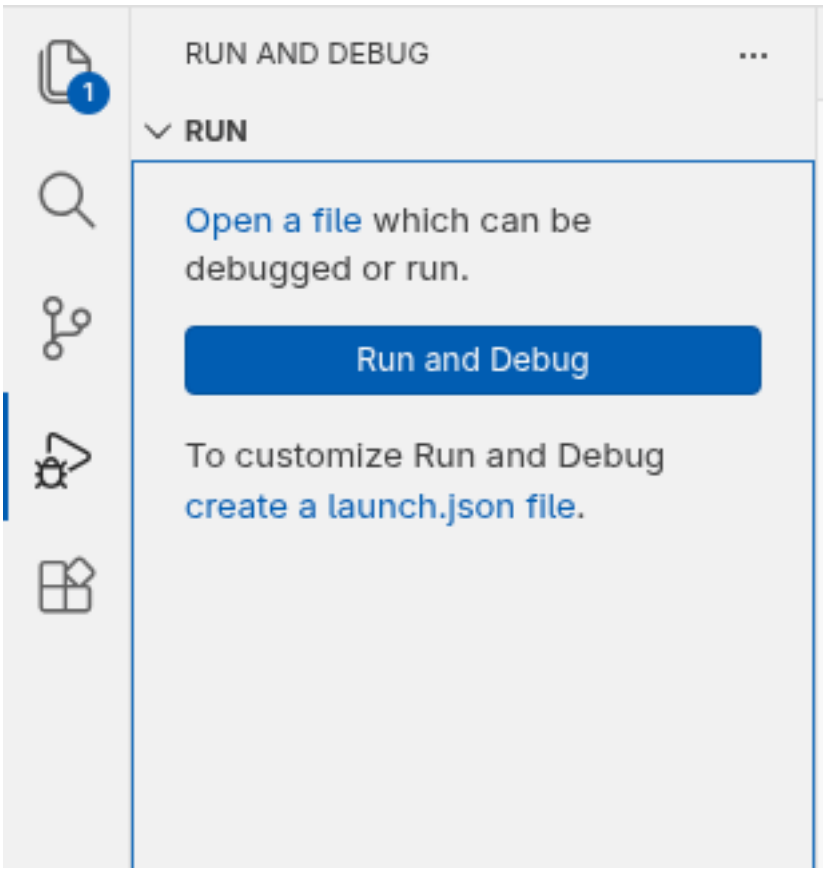
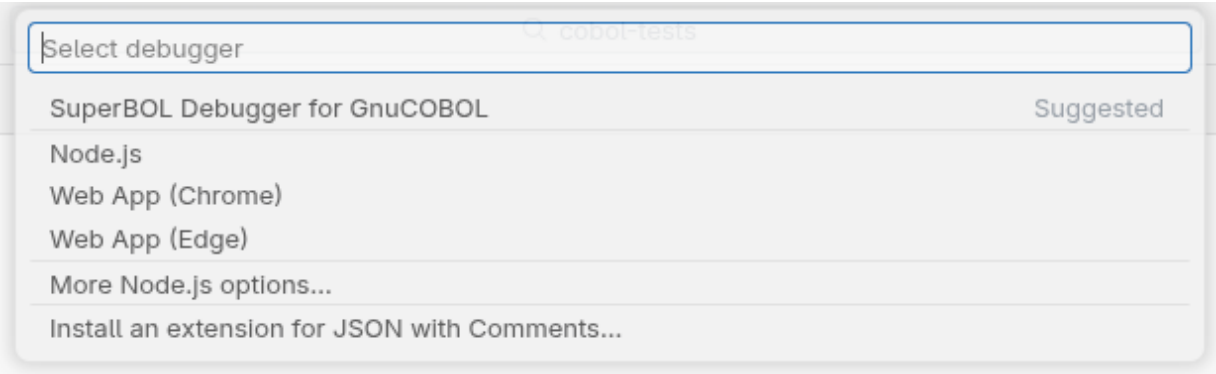


Figure 24. – Exécution et déboguer

2. Cliquez sur le lien `créer un fichier launch.json`. Si vous n'avez pas ce visuel, cela signifie que vous avez déjà un tel fichier, vous devriez être en mesure de l'ouvrir maintenant et de sauter le reste de ces étapes.
3. Une liste de configurations prédéfinies vous sera présentée, vous devriez sélectionner l'option `SuperBOL Debugger for GnuCOBOL`.



4. Le `.vscode/launch.json` sera généré avec des configurations prédéfinies fournies par superbol :

```
{
   // Utilisez IntelliSense pour en savoir plus sur les attributs possibles.
   // Pointez pour afficher la description des attributs existants.
   // Pour plus d'informations, visitez : https://go.microsoft.com/fwlink/?
linkid=830387
   "version" : "0.2.0",
   "configurations": [
       {
           "name": "SuperBOL : debug (launch)",
           "type": "superbol-gdb",
           "request": "launch",
           "preLaunchTask": "SuperBOL : build (debug)",
           "target": "${file}",
           "arguments": ""
       },
       {
           "name": "SuperBOL : debug (attach local)",
           "type": "superbol-gdb",
           "request": "attach",
           "pid": "${input:pid}",
           "target": "${file}"
       },
       {
           "name": "SuperBOL : debug (attach remote)",
           "type": "superbol-gdb",
           "request": "attach",
           "remoteDebugger": "${input:remoteDebugger}",
           "target": "${file}"
       }
   ]
}
```

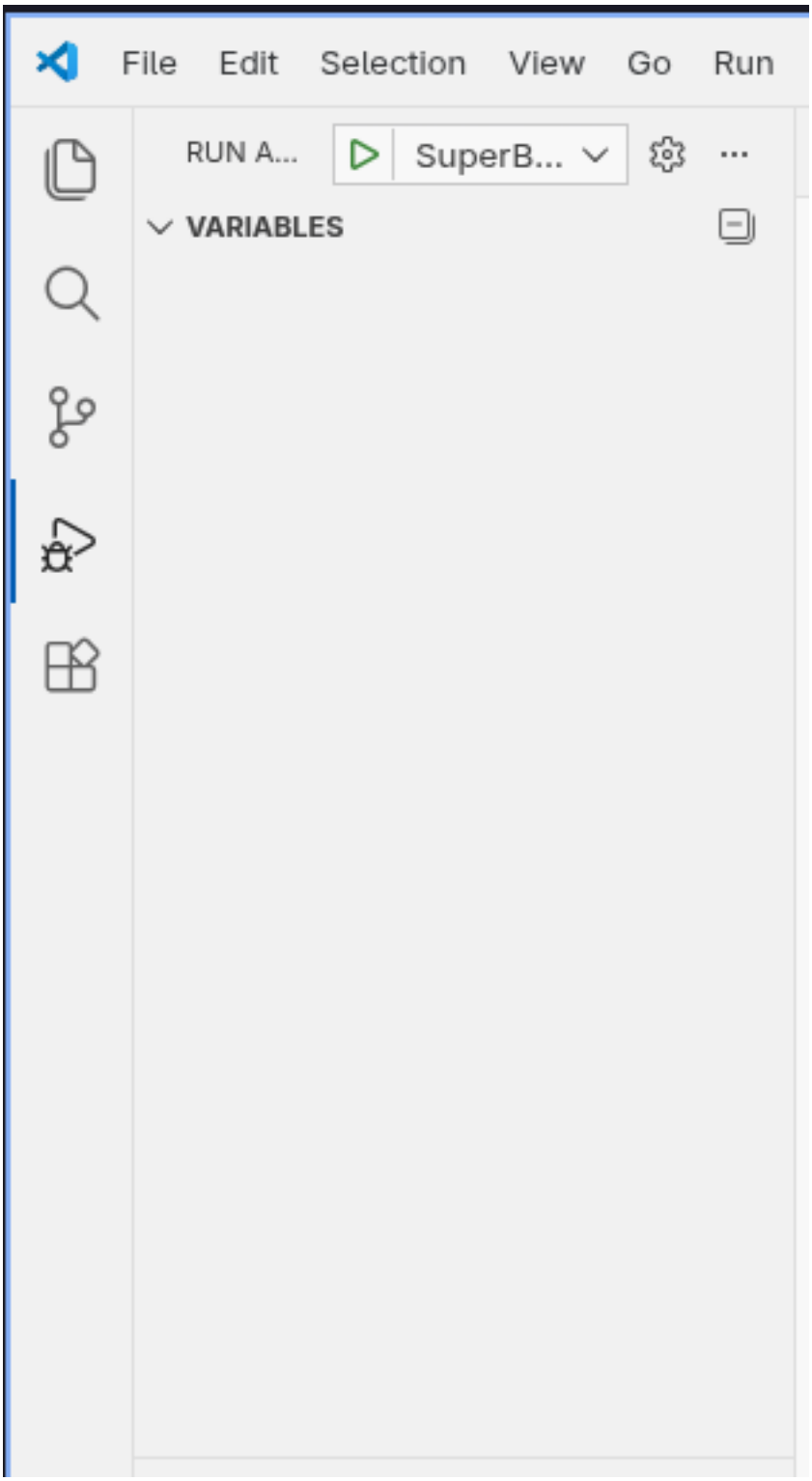
Grâce à ces préréglages, vous pouvez gérer les différents scénarios présentés plus tôt :

- `"SuperBOL : debug (launch)"` essaiera de compiler le programme ouvert et d'exécuter `gdb` sur le programme compilé.
- `SuperBOL : debug (attach local)"` vous demandera un `pid` pour attacher le débogueur `gdb`, et utilisera le fichier COBOL ouvert actuellement comme code source du processus attaché.
- `"SuperBOL : debug (attach remote)"` vous demandera une adresse et un port pour se connecter à un débogueur distant, avec `gdbserver`, et utilisera le fichier COBOL actuellement ouvert comme code source du processus distant.

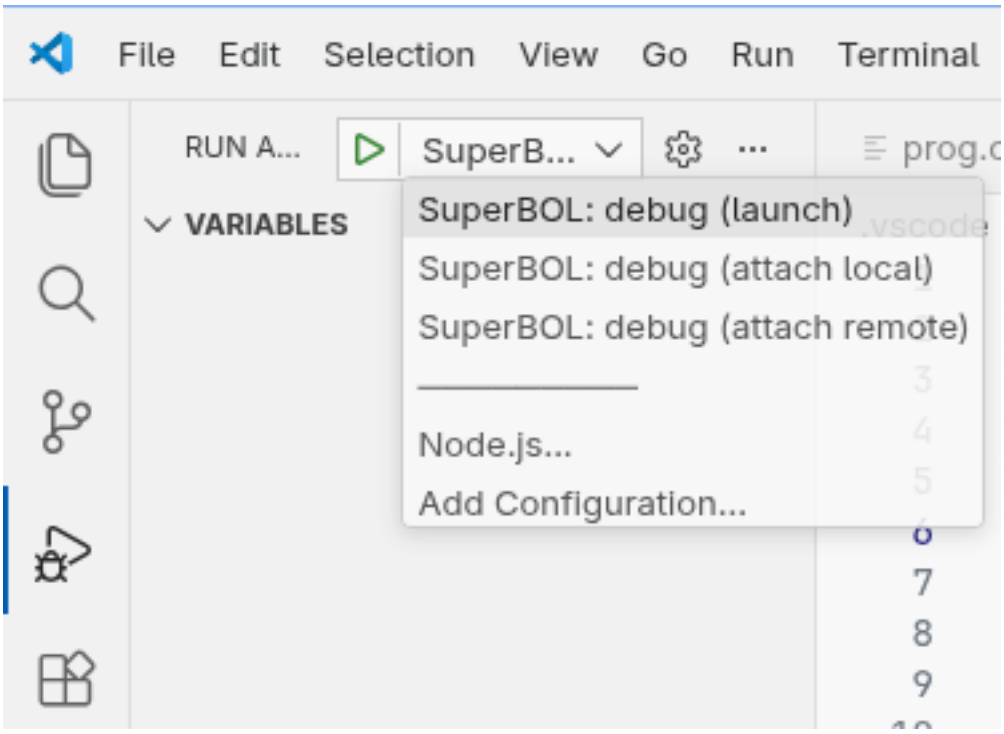
> **Note**
>
> Dans tous les scénarios présentés, vous devez avoir la source COBOL sur votre système local, et avoir le programme que vous souhaitez déboguer ouvert dans votre éditeur.
>
> Pour la configuration `SuperBOL : debug (attach remote)`, le serveur doit avoir `gdbserver` installé et disponible.

Pour choisir la configuration à utiliser pour déboguer votre programme, vous devez :

1. Ouvrez le panneau `Exécuter et déboguer` (Ctrl+Maj+D)



2. Sélectionnez la configuration dans le menu déroulant en haut du panneau.



Pour lancer le débogage, vous pouvez ensuite appuyer sur la touche F5. La configuration que vous avez sélectionnée devrait être exécutée par défaut.

Les deux configurations `remote` ont besoin d'une entrée pour être lancées avec succès, ce qui nécessite une configuration manuelle, qui est expliquée dans la section suivante.

9.4 Gestion des entrées de configuration

Comme vous l'avez peut-être remarqué, certains champs des configurations SuperBOL ont un paramètre `${input:<identifiant>}`. Cela signifie que vous devez fournir une entrée lors du lancement de la configuration.

VSCode ne fournit pas de moyen de configurer automatiquement ces entrées, vous devez donc écrire vous-même certaines configurations. Pour configurer ces entrées, vous devez :

1. Ouvrez le fichier `.vscode/launch.json`.
2. Ajoutez un tableau `inputs` dans le fichier :

```
{
    "version": "2.0.0",
    "configurations": [
        ...
    ],
```

```
    "inputs": []
}
```

3. Dans ce tableau, ajoutez chaque entrée nécessaire aux configurations de débogage, dans notre cas, il s'agit de `"pid"` et `"remoteDebugger"` (vous pouvez n'ajouter qu'un seul des deux si vous n'avez pas l'intention de les utiliser tous les deux). Votre tableau d'entrées devrait ressembler à ceci :

```
{
   "inputs" : [
       {
           "id" : "pid",
           "type" : "promptString",
           "description" : "PID du processus auquel s'attacher".
       },
       {
           "id" : "remoteDebugger",
           "type" : "promptString",
           "description" : "Adresse du débogueur distant".
       }
   ]
}
```

   Dans ce tableau, chaque objet a les mêmes champs, avec des valeurs différentes:

   - `"id"` : L'identifiant de l'entrée, doit être le même que l'identifiant à droite de `${input:<identifiant>}` dans votre configuration.
   - `"type"` : Le type d'entrée à fournir, ici `"promptString"` signifie qu'il vous sera demandé d'entrer une chaîne de caractères avant que la configuration ne soit lancée.
   - `"description"` : Une petite description de l'entrée à fournir.

Maintenant, lorsque vous utilisez la configuration `"attach"`, VSCode vous demandera l'entrée correspondante.

9.5 Personnaliser les configurations de débogage

Les configurations prédéfinies peuvent ne pas être adaptées à votre cas d'utilisation, mais vous pouvez les personnaliser pour répondre à vos besoins.

Pour personnaliser votre configuration de débogage, vous pouvez soit éditer les préconfigurations, soit en ajouter un nouveau dans le fichier `.vscode/launch.json`. Voici les champs que vous pouvez ajouter ou éditer dans les configurations :

- `"name"` (*requis*) : Le nom de la configuration, vous pouvez lui donner le nom que vous souhaitez.
- `"type"` (*requis*) : C'est le seul champ constant, qui doit toujours avoir la valeur `"superbol-gdb"`.
- `"request"` (*requis*) : Cela devrait être soit :
  - `"launch"` : cela signifie que le débogueur s'exécutera sur une cible qu'il lance
  - `"attach"` : cela signifie que le débogueur s'attachera à un processus déjà en cours d'exécution
- `"preLaunchTask"` : Ce champ indique quelle tâche doit être exécutée avant une requête `"launch"`. Il doit correspondre au label de la tâche de compilation qui doit être exécutée avant le lancement. Vous pouvez consulter Configurer une tâche de compilation pour plus d'informations sur ces tâches.
- `"target"` : Chemin vers le programme à déboguer. Vous pouvez le définir avec la valeur `"${file}"` pour déboguer le programme qui est ouvert dans votre éditeur
- `"arguments"` : Un tableau d'arguments à passer au programme débogué. Dans une requête `"attach"`, ces arguments ne seront transmis qu'au redémarrage du programme.
- `"cwd"` : Le répertoire à partir duquel le débogueur doit être lancé.
- `"env"` : Un objet contenant des variables d'environnement, par exemple :

```
{
    "COB_LIBRARY_PATH" : "/usr/lib/cobol/modules"
}
```

- `"useCobcrun"` : Utilise un chargeur de modules COBOL, vous pouvez spécifier le chargeur à utiliser dans `"cobcrunPath"`.
- `"cobcrunPath"` : Chemin d'accès au chargeur de modules cobol, `"cobcrun"` par défaut.
- `"sourceDirs"` : Un tableau de chemins pour spécifier où chercher le code source. Si ce tableau n'est pas fourni, seul le répertoire courant est pris en compte.

Certaines options sont spécifiques à un type de `"request"` :

- dans les requêtes `"launch"` vous pouvez définir :

- `"gdbtty"` : Ce champ vous permet d'utiliser un terminal différent pour les l'entrée et la sortie COBOL. Il peut être mis à `false` pour utiliser le terminal de débogage intégré, `true` pour utiliser un terminal externe, qui peut être choisi parmi `xterm`, `gnome-terminal`, `xfce4-terminal`, ou `konsole`. Vous pouvez également définir le champ avec l'une des valeurs suivantes:
  - `"vscode"` : Utilise le terminal de VSCode.
  - `"xterm"`, `"gnome-terminal"`, `xfce4-terminal`, `"konsole"` : Utilise le terminal spécifié par la chaîne.
  - `"external"` : Utilise l'un des terminaux listés précédemment, le premier qui est disponible pour être exécuté sera utilisé.
- Dans les requêtes `"attach"` vous pouvez définir :
  - `"pid"` : Le PID du programme à attacher. Vous devez utiliser une variable `${input:<identifiant>}` si le PID du programme n'est pas constant.
  - `"remoteDebugger"` : Adresse de `gdbserver`, au format `"hôte:port"`.

9.6 Exemples de configurations de débogage

Ces différents examples se trouvent dans le tableau `"configurations"` du fichier `.vscode/launch.json`.

9.6.1 Déboguer un module GnuCOBOL, avec un chemin `cobcrun` personnalisé

```
{
    // Nom personnalisé pour la configuration
    "name": "SuperBOL: debug (module)",

    "type": "superbol-gdb",

    // Le débogueur exécutera la programme à déboguer
    "request": "launch",

    // Utlisation d'une tâche de compilation personnalisée
    "preLaunchTask": "SuperBOL: build (module + debug)",
    "target": "${file}",

    // Utilsation d'un chargeur de module GnuCOBOL
    "useCobcrun": true,

    // Utilisation d'un `cobcrun` spécifique, donné avec son chemin
    "cobcrunPath": "/opt/gnucobol/bin/cobcrun"
}
```

10 Couverture du code

GnuCOBOL peut instrumenter vos programmes de manière à ce qu'ils génèrent des informations de couverture au moment de l'exécution. Pour activer cette fonctionnalité, vous pouvez mettre le paramètre `forCoverage` à `true` dans la tâche `Superbol : build (debug)` de votre fichier `tasks.json` (voir Compilation des programmes pour plus d'informations sur les tâches.) Ce drapeau indique à l'extension de passer le drapeau `--coverage` au compilateur `cobc`.

10.1 Informations sur la couverture

Les fichiers de couverture générés sont au format `gcov` ; ils sont portables, et vous pouvez les utiliser comme n'importe quel autre fichier de couverture générés pour des programmes écrits avec d'autres langages de programmation.

10.2 Mise en évidence des informations relatives à la couverture

Les données de couverture peuvent être affichées après la fin de l'exécution d'un programme qui a été compilé pour générer ces informations. SuperBOL affichera la couverture ligne par ligne, en mettant en évidence les lignes de votre code source en utilisant des couleurs qui représentent leur état de couverture. Pour activer la mise en évidence de la couverture, vous pouvez ouvrir la palette de commandes (ou tapez Ctrl+Maj+P), et sélectionnez `SuperBOL : Show Coverage`. Vous pouvez également masquer la surbrillance avec avec la commande `SuperBOL : Hide Coverage`, et la mettre à jour après avoir après avoir ré-exécuté votre programme avec `SuperBOL : Update Coverage`.

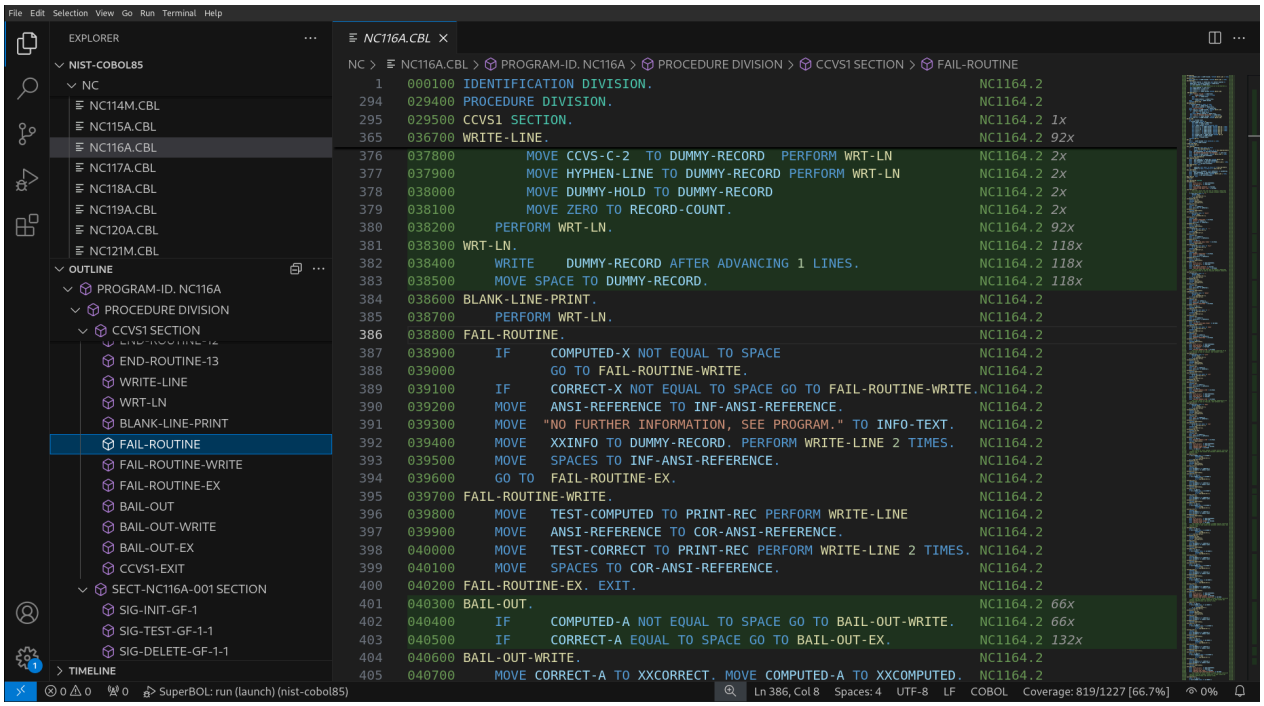


Figure 25. – Couverture du spectacle

11 Utilisation de Vscode en général

11.1 Naviguer à travers les avertissements et les erreurs, alias les problèmes

Les problèmes (erreurs et avertissements) ne sont pas toujours aussi visibles que nous le souhaiterions. Vous pouvez installer une extension pour améliorer leur améliorer leur affichage, typiquement l'extension VSCode « Error Lens » qui affiche les erreurs sous forme de messages en ligne dans le code source.

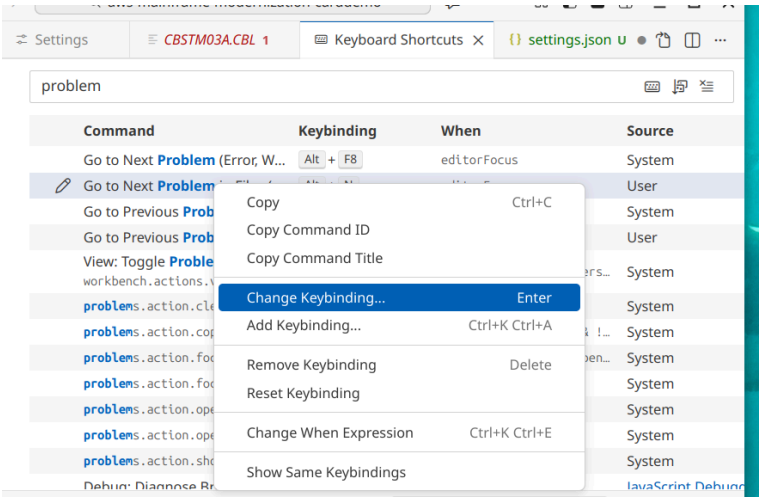
Vous pouvez également utiliser les raccourcis suivants :

- Ctrl+Maj+M : aller au premier problème, et afficher l'onglet « Problèmes », montrant la liste de tous les problèmes trouvés. Vous pouvez utiliser les flèches de votre clavier pour naviguer dans la liste.
- F8 : passer au problème suivant dans le fichier
- Maj+F8 : aller au problème précédent dans le fichier

Certains utilisateurs préfèrent utiliser d'autres raccourcis, par exemple parce que la touche F8 est déjà utilisé pour autre chose.

Pour cela :

- Ouvrez les « Raccourcis clavier » en tapant successivement Ctrl+K Ctrl+S
- Tapez « Problème » pour afficher les raccourcis correspondants.
- Cliquez sur le raccourci que vous souhaitez modifier, et utilisez « Changer de combinaison de touches » pour proposer un nouveau raccourci.
- Par exemple, dans notre cas, nous utiliserons Alt +N pour « Aller au problème suivant dans les fichiers ».



11.2 Passer d'un thème clair à un thème foncé

1. Ouvrez la **Palette de commandes** avec Ctrl+Maj+P
2. Tapez « Thème de couleur ».
3. Sélectionnez **Préférences : Thème de couleur** dans la liste déroulante.
4. Choisissez un thème de couleur, généralement **Moderne sombre** ou **Moderne clair**.

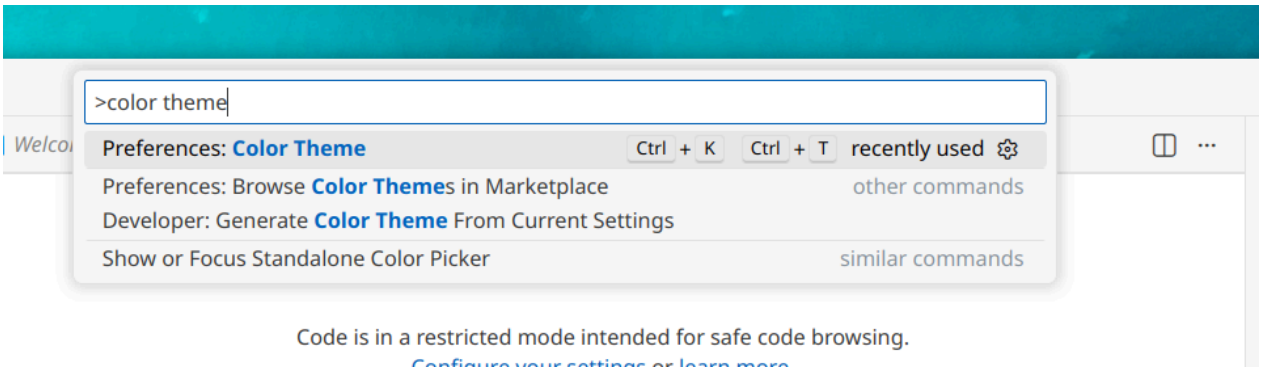


Figure 26. – Selection du Thème de couleur

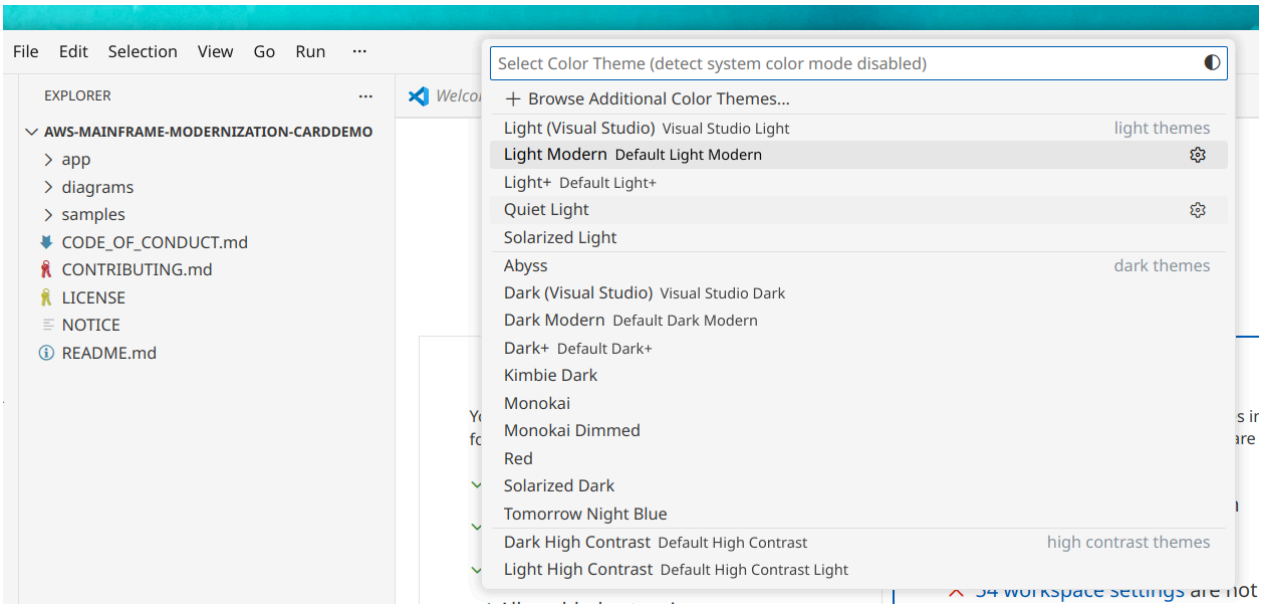


Figure 27. – Selection du Thème de couleur

11.3 La fenêtre contextuelle « Faites-vous confiance aux auteurs des fichiers de ce dossier ? »

Lorsque vous entrez dans un nouveau projet, vous obtenez toujours la fenêtre contextuelle suivante :

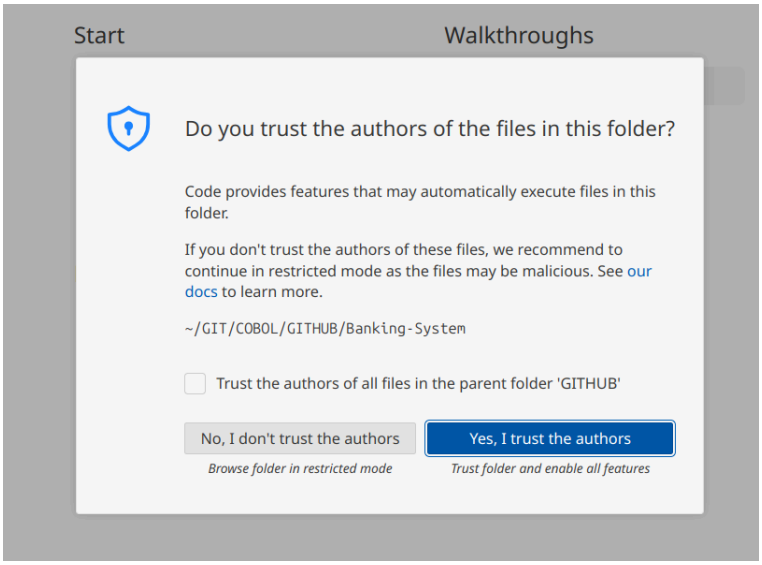


Figure 28. – Popup demande de confiance

Pour utiliser l'extension SuperBOL Studio, vous devez répondre « Oui, je fais confiance aux auteurs ».

Si vous répondez « Non, je ne fais pas confiance aux auteurs », Vscode entre dans un **Mode restreint**, où :

- Les extensions sont désactivées : La plupart des extensions qui font réellement quelque chose (comme les débogueurs, les linters, ou les serveurs de langage pour Python/C++) seront désactivées. Cela empêche un espace de travail malveillant d'utiliser une extension pour exécuter un script en arrière-plan.
- Les tâches sont bloquées : Toute automatisation que vous avez configurée dans `.vscode/tasks.json` (comme votre commande make pour Pandoc) sera désactivée. VSCode ne les exécutera pas car il ne sait pas si la commande est sûre.
- Le débogage est désactivé : Vous ne pouvez pas lancer le débogueur, car cela nécessite l'exécution du code dans l'espace de travail.
- Les paramètres de l'espace de travail sont ignorés : Certains paramètres définis spécifiquement pour ce dossier (dans `.vscode/settings.json`) seront remplacés par vos paramètres utilisateur globaux afin d'éviter que le dossier ne « détourne » le comportement de votre éditeur.

Si vous avez répondu « Non », vous verrez un bouton « Mode restreint » dans le coin inférieur gauche, cliquez dessus et sélectionnez le bouton « Faire confiance » pour quitter le mode restreint.

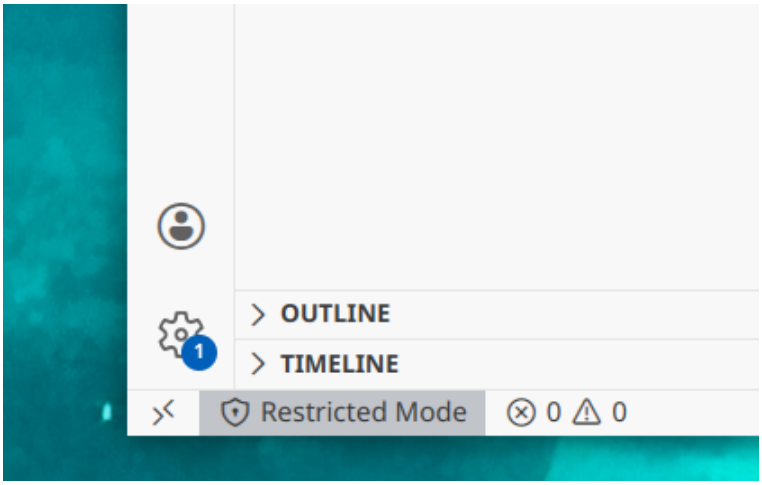


Figure 29. – Mode restreint

12 Ajouter une licence SuperBOL

12.1 Fonctionnalités premium de SuperBOL

Certaines fonctionnalités de SuperBOL sont accessibles aux utilisateurs ayant une licence premium.

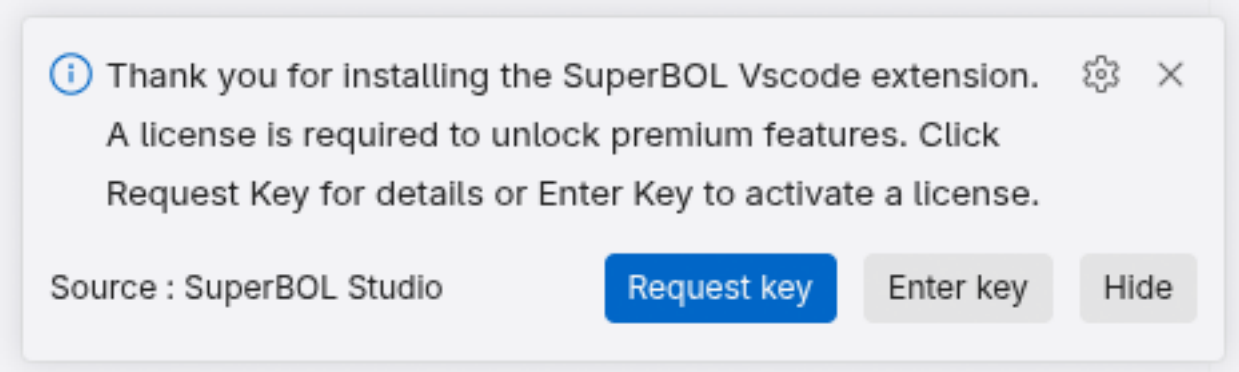
Vous pouvez acheter une de ces licences en nous contactant depuis notre site superbol.eu ou par mail à superbol@ocamlpro.com.

12.2 Enregistrer une licence dans VS Code

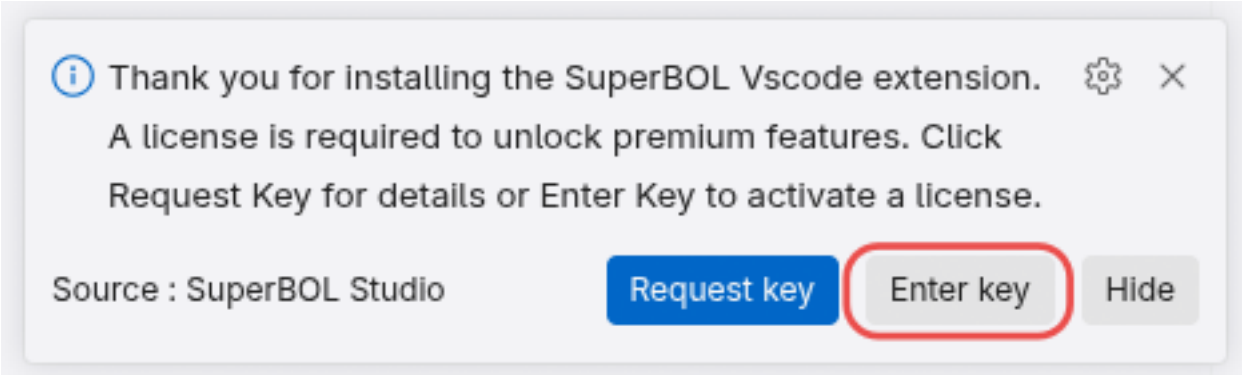
Afin d'enregistrer une licence dans votre application VS Code, vous pouvez soit passer par les `paramètres` de l'extension SuperBOL, soit utiliser le bouton sur la notification qui apparaît au lancement de SuperBOL.

12.2.1 Ajout via la notification de démarrage

Lors du démarrage de SuperBOL, si aucune licence n'est enregistrée, cette notification apparaît en bas à droite de la fenêtre VS Code :

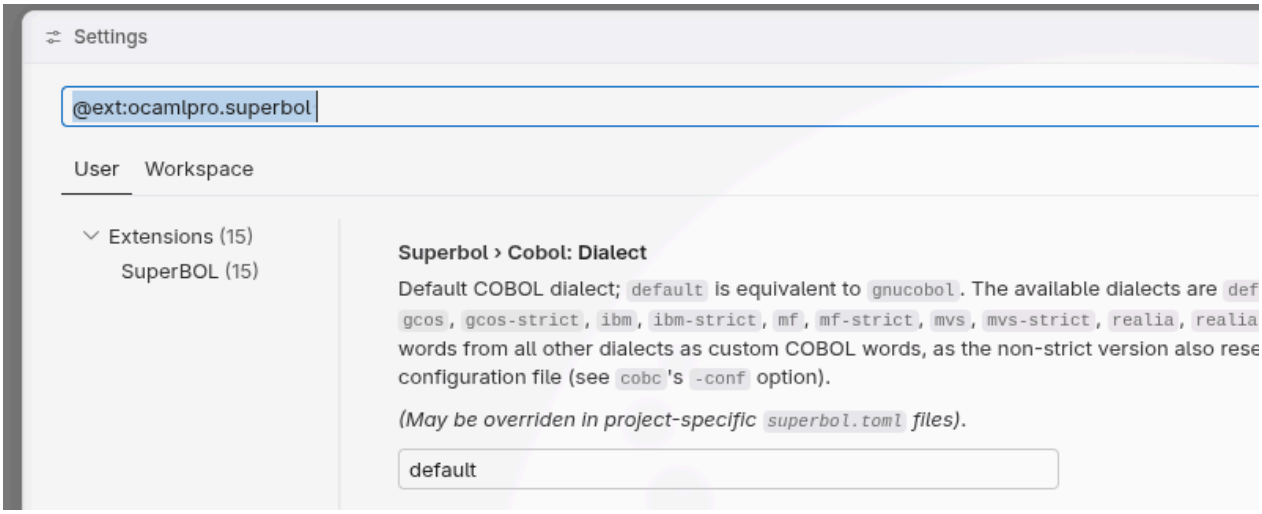


En cliquant sur `Enter key`, la fenêtre des paramètres s'ouvre, vous pouvez alors procéder en suivant les étapes dans la section ajouter une licence.



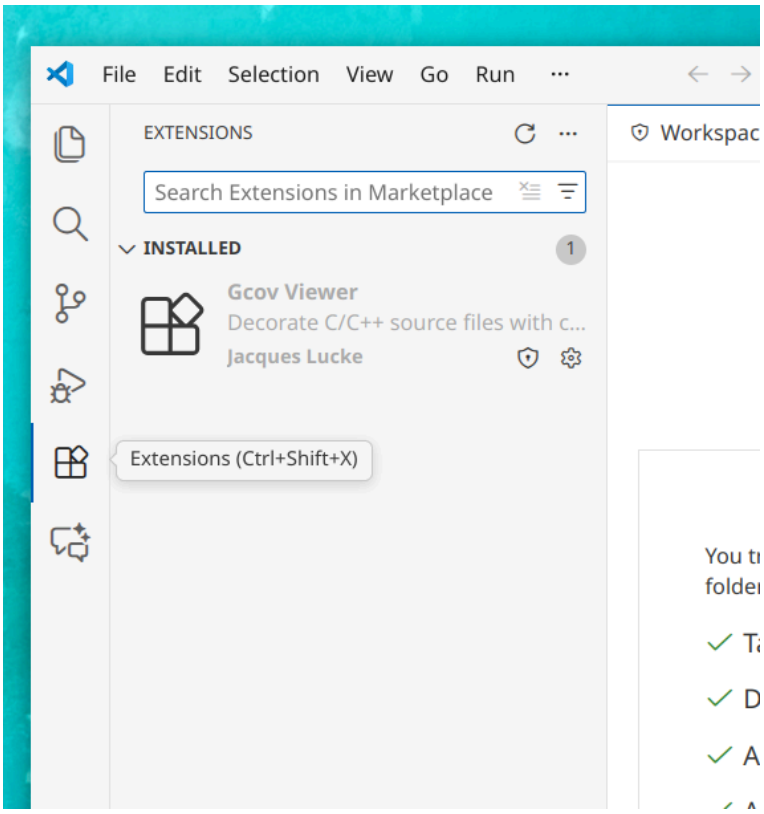
12.2.2 Ajout via les paramètres de l'extension SuperBOL

L'accès aux paramètres peut se faire avec le raccourci clavier Ctrl+,, et ensuite en recherchant `@ext:ocamlpro.superbol`. Vous pouvez ensuite reprendre à partir de l'étape **5** ci-dessous.

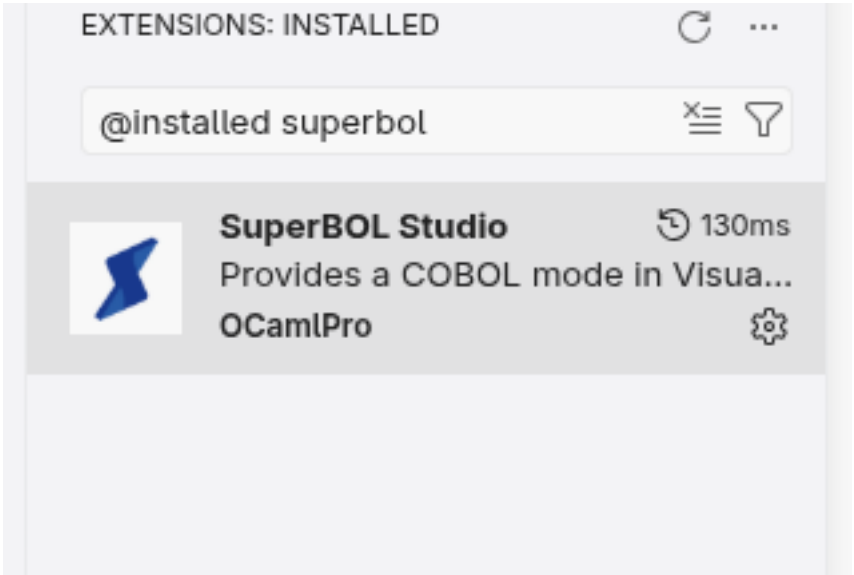


Afin d'accéder à l'ajout de licence via les paramètres de l'extension directement :

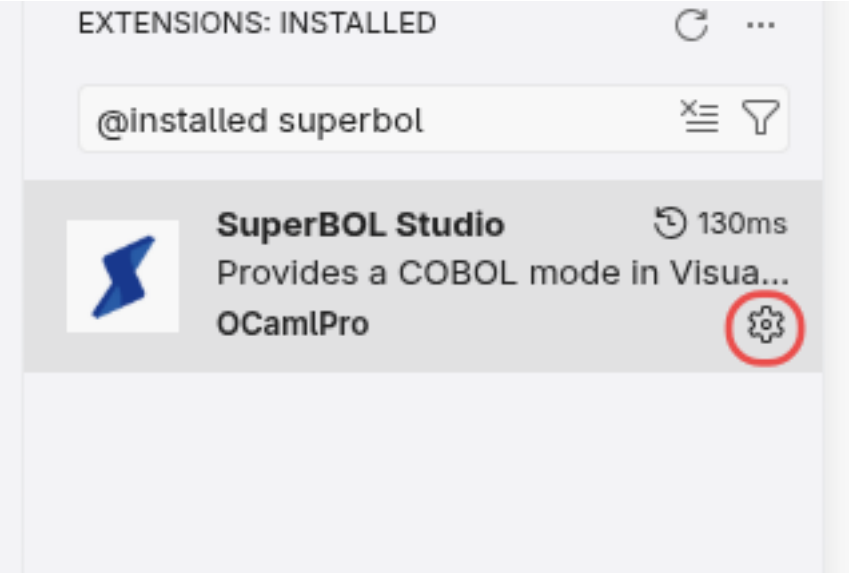
1. Ouvrez le panneau des extensions :



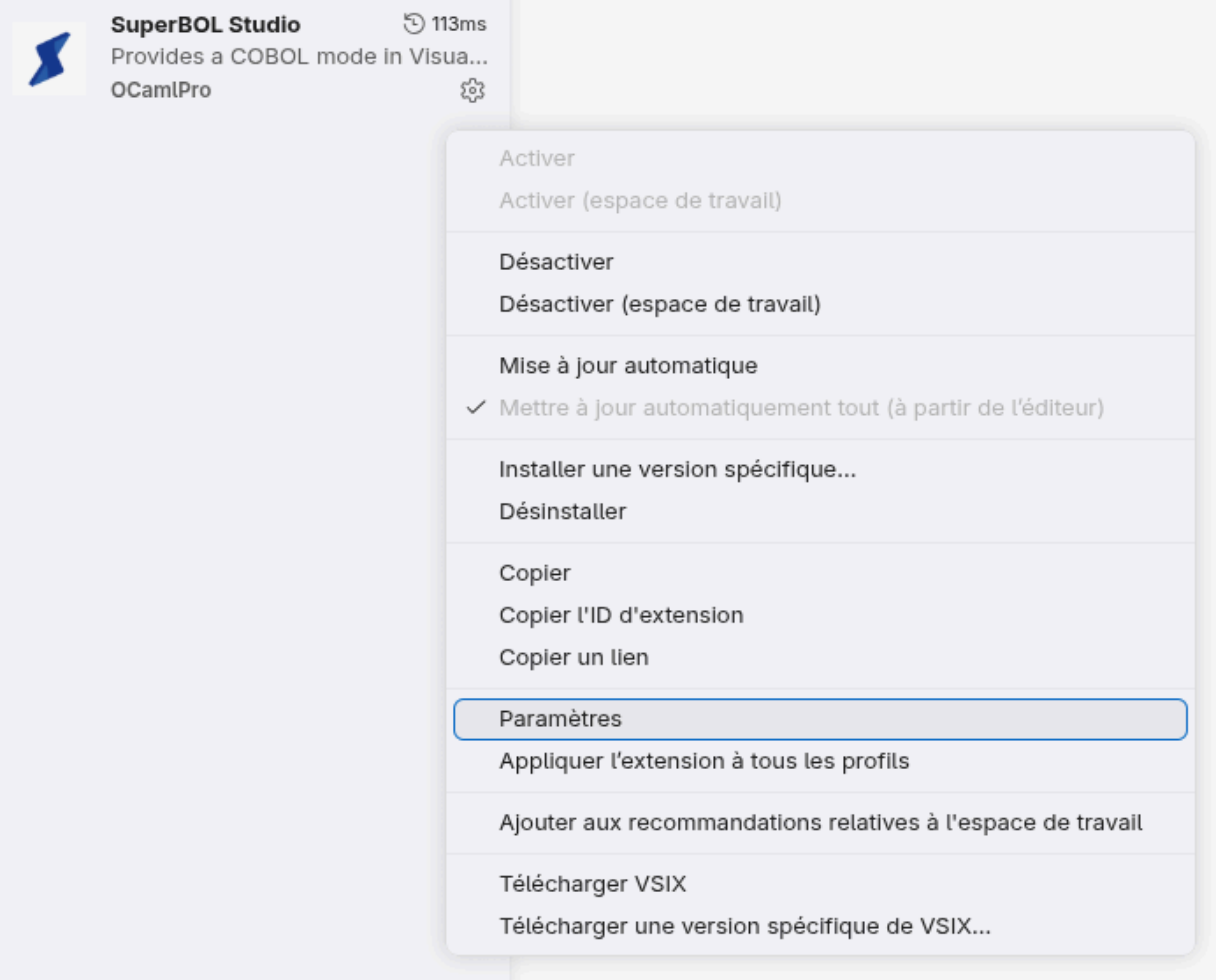
2. Recherchez « SuperBOL » :



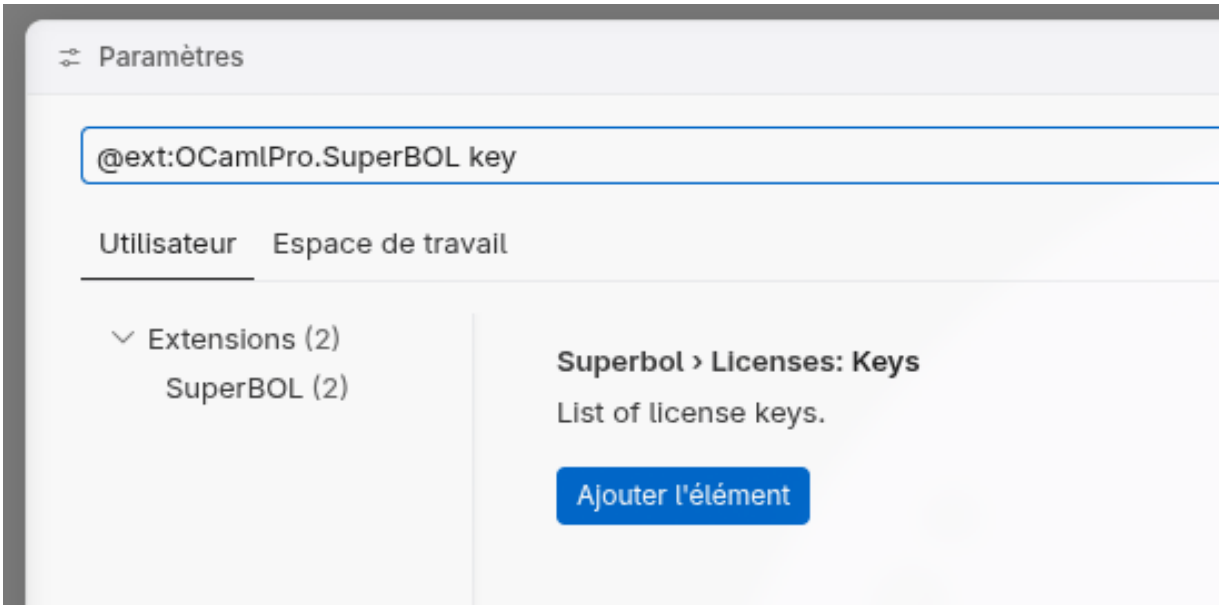
3. Cliquez sur l'icône d'engrenage :



4. Sélectionnez l'option « paramètres » :

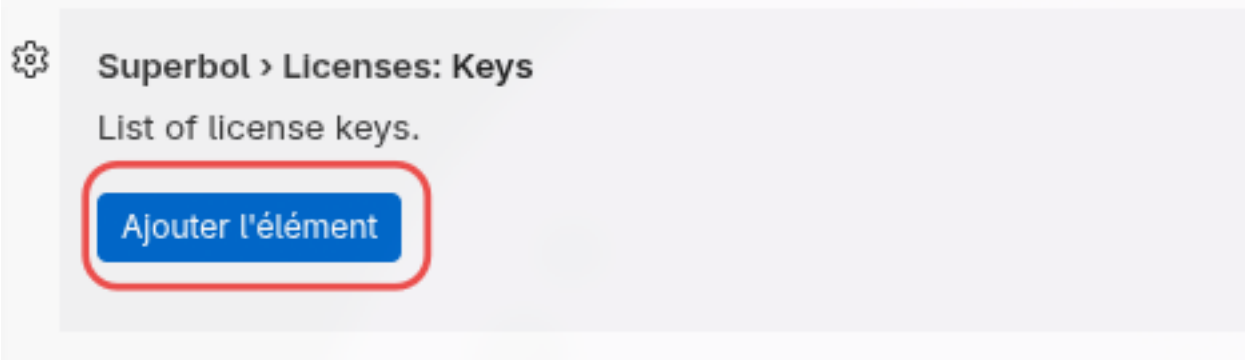


5. Cherchez « key » dans les paramètres :

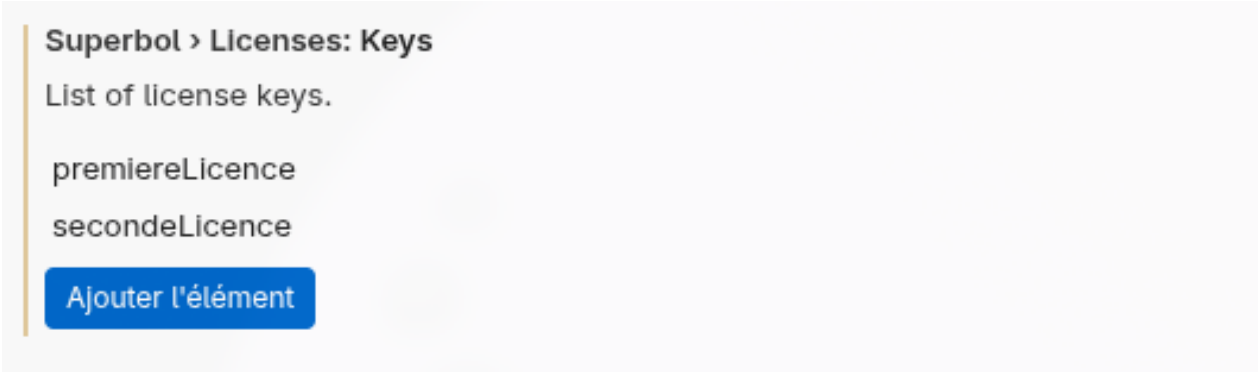


12.2.3 Entrer une nouvelle licence SuperBOL

Une fois que vous avez accès aux paramètres des licences, vous pouvez cliquer sur `Ajouter l'élément`, une zone de texte apparaît alors où vous pouvez copier votre licence.



Une seule licence doit être entrée dans cette zone de texte. Si vous avez plusieurs licence, choisissez `Ajouter l'élément` pour chaque licence différente que vous avez.


Superbol › Licenses: Keys
List of license keys.
premiereLicence
secondeLicence
Ajouter l'élément

13 SuperBOL File Viewer Premium

SuperBOL File Viewer est un outil de visualisation de fichiers de données COBOL. Il prend en charge les types d'organisation suivants :

- `SEQUENTIAL` et `LINE SEQUENTIAL` ;
- `INDEXED` ;
- `RELATIVE`.

Ces fichiers ont typiquement des extensions comme `.dat`, `.seq` ou `.idx`.

> ! Important
>
> SuperBOL File Viewer est une fonctionnalité **Premium**. Une version gratuite permet de visualiser les fichiers comportant au plus 100 enregistrements. Au-delà, une clé de licence est nécessaire (voir Ajouter une licence SuperBOL). Contactez superbol@ocamlpro.com pour les offres commerciales.

13.1 Prérequis

Les fichiers de données COBOL ne peuvent pas être interprétés seuls : leur structure est définie par le programme COBOL qui les utilise. Pour ouvrir un fichier de données, il vous faut donc :

- un **programme COBOL** (fichier `.cob` ou `.cbl`) contenant la déclaration du fichier dans la section `FILE-CONTROL` ainsi que la description des enregistrements dans la `DATA DIVISION` ;
- éventuellement, des **copybooks** (`.cpy`) référencés par ce programme pour décrire la structure des champs.

Il est donc nécessaire d'identifier au préalable quel programme COBOL définit le fichier que vous souhaitez visualiser.

13.2 Ouverture d'un fichier

13.2.1 Étape 1 : Ouvrir le fichier de données

Deux méthodes permettent d'ouvrir un fichier de données dans VSCode :

- **Via le menu Fichier** : allez dans `File > Open File...` et sélectionnez directement votre fichier de données.
- **Via l'explorateur de fichiers** : ouvrez le dossier contenant vos fichiers avec `File > Open Folder...`, puis double-cliquez sur le fichier de données souhaité (par exemple un fichier d'extension `.dat`) dans le panneau explorateur à gauche.

Une fois le fichier ouvert, une fenêtre de sélection de schéma apparaît au centre de l'écran.

13.2.2 Étape 2 : Sélectionner le programme COBOL de définition

L'ouverture du fichier requiert l'association d'un programme COBOL qui en décrit le format.

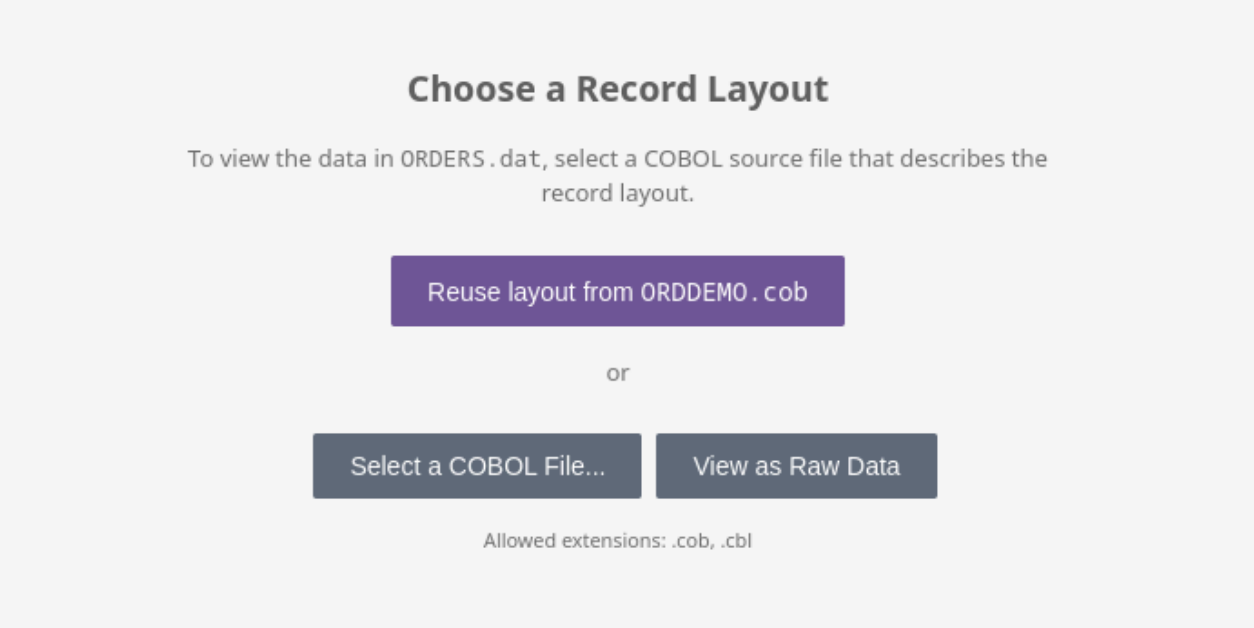


Figure 30. – Fenêtre de sélection du fichier source déclarant le fichier de donnée ouvert.

- **Première ouverture** : cliquez sur `Select a COBOL File...` et sélectionnez le fichier `.cob` ou `.cbl` correspondant sur votre ordinateur. Ce programme peut se trouver dans le projet ouvert ou à n'importe quel autre emplacement sur votre disque. L'association est sauvegardée pour les ouvertures ultérieures de ce fichier de données, dans l'espace de travail courant.
- **Ouvertures suivantes** : si le fichier a déjà été associé à un programme COBOL, un bouton `Reuse layout from <programme>`, rappelant le nom de ce fichier COBOL, apparaît au-dessus et vous permet de réutiliser directement l'association précédente.

Il est également possible de visualiser le fichier brut sans programme COBOL en cliquant sur `View as Raw Data`.

> Astuce
>
> Si vous n'avez pas accès au programme COBOL correspondant, vous pouvez tout de même ouvrir le fichier en cliquant sur `View as Raw Data` pour en inspecter le contenu brut.

13.2.3 Étape 3 : Désambiguïsation (si nécessaire)

> **Note**
>
> Cette étape n'apparaît que si le programme COBOL sélectionné déclare plusieurs fichiers dans sa section `FILE-CONTROL`. Si un seul fichier est déclaré, le fichier s'ouvre directement.

Un même programme COBOL peut déclarer plusieurs fichiers dans sa section `FILE-CONTROL`. Dans ce cas, une fenêtre de désambiguïsation s'affiche pour vous demander de choisir la déclaration correspondant au fichier que vous souhaitez ouvrir.

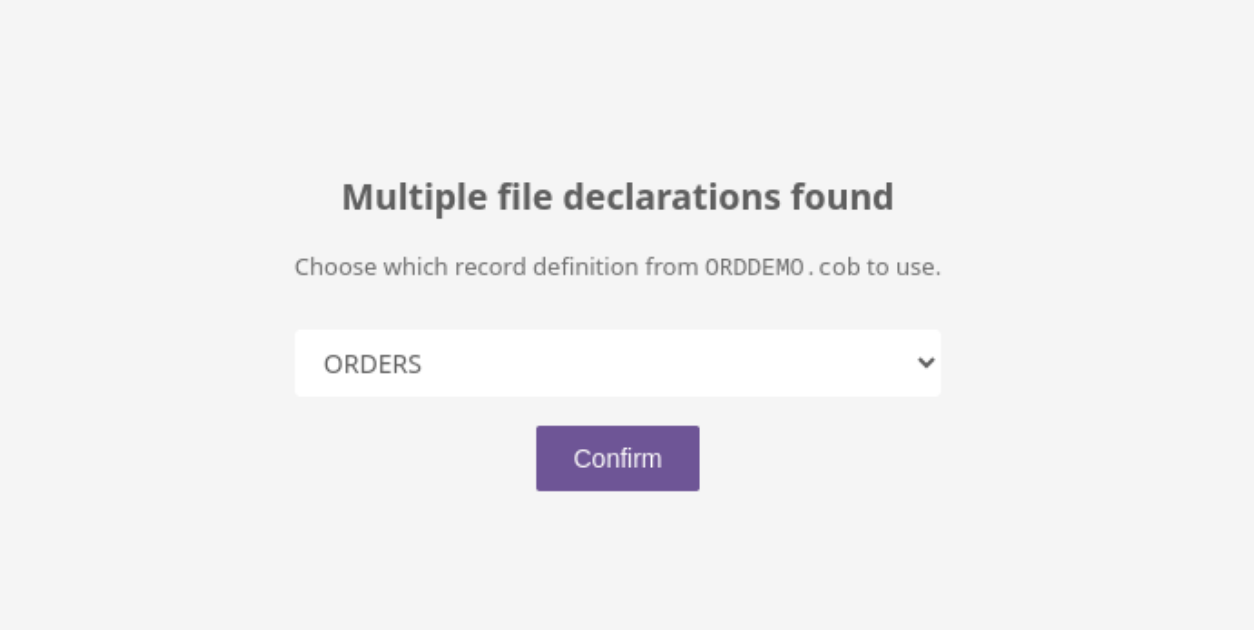


Figure 31. – Fenêtre de désambiguïsation des déclarations de fichier (dans le cas où plusieurs fichiers de données sont déclarés par un même programme COBOL).

Sélectionnez la déclaration appropriée dans la liste déroulante, puis cliquez sur `Confirm` pour valider.

13.3 Interface de la visionneuse

Une fois le fichier ouvert avec le programme COBOL associé, la visionneuse affiche les données structurées sous forme de tableau, avec une ligne par enregistrement et une colonne par champ (les colonnes pouvant être groupées pour représenter les groupes de données COBOL).

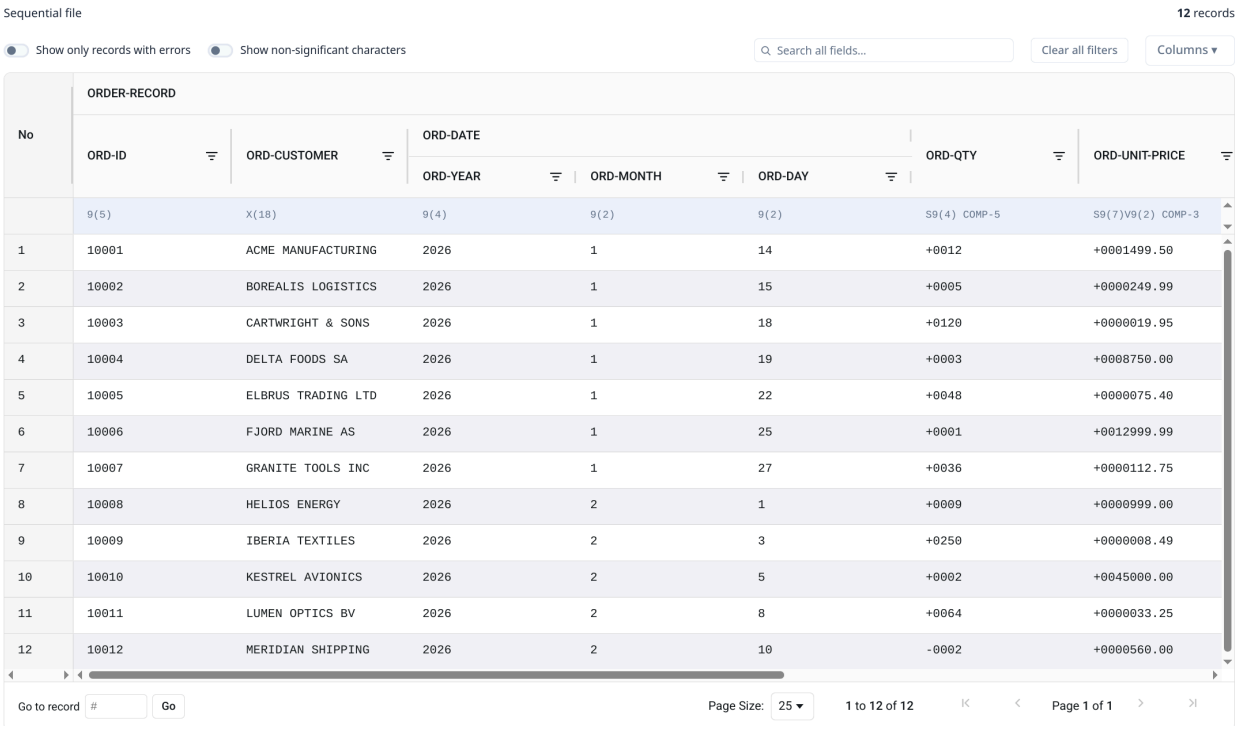


Figure 32. – Interface principale de la visionneuse de fichiers.

De haut en bas, l'interface se divise en trois parties principales dans lesquelles plusieurs interactions sont possibles :

- l'en-tête, indiquant l'organisation du fichier et son nombre total d'enregistrements, ainsi qu'un panneau de contrôle ;
- le tableau de données où les colonnes peuvent être déplacées librement, à l'exception de la colonne `No` des numéros d'enregistrement qui reste fixée à gauche, et où un champ ne peut pas être sorti du groupe auquel il appartient;
- le panneau de navigation.

La première ligne du tableau est fixée sous les noms de colonnes et indique la clause PIC et l'USAGE de chaque champ. Le survol de l'une de ses cellules affiche des informations complémentaires sur le champ, comme le fait qu'il soit signé et la façon dont il est stocké.

13.4 Tableaux (instruction OCCURS)

Un groupe comportant une clause `OCCURS n` est affiché comme un tableau de colonnes appelées « occurrences » indexées par un entier.

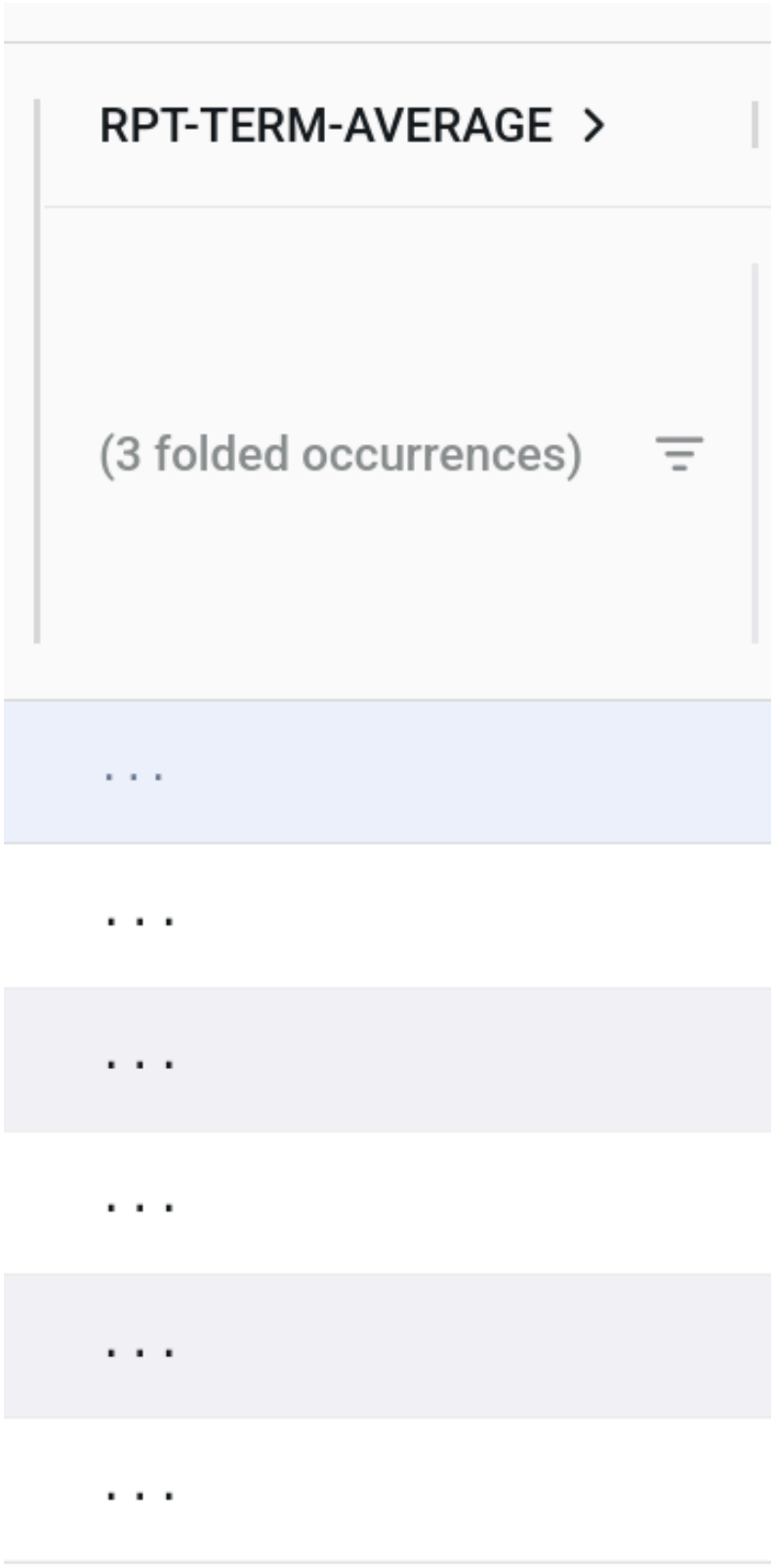
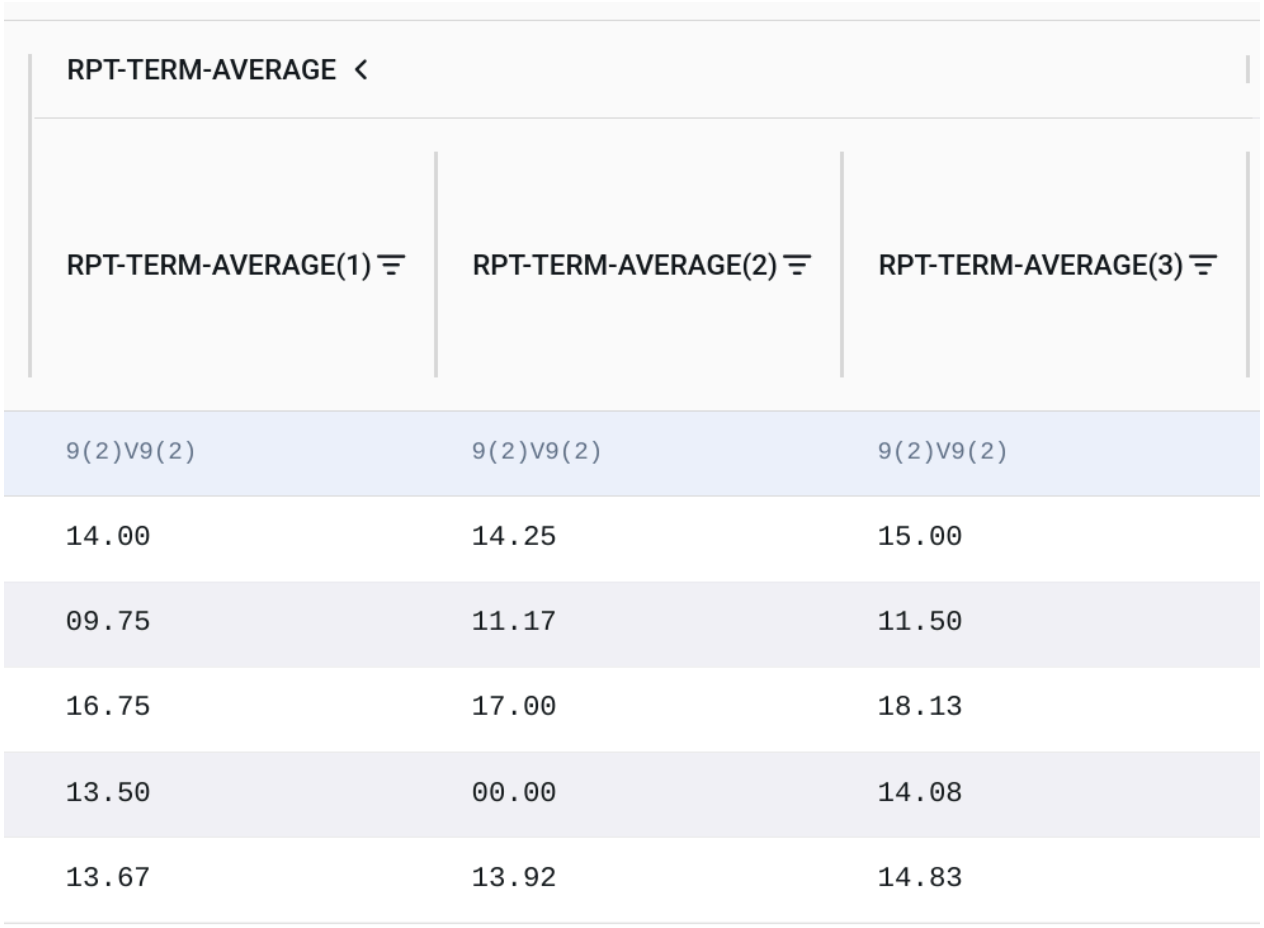


Figure 33. – Groupe `OCCURS` replié : une seule colonne grisée résume les occurrences.

À l'ouverture du fichier, ce tableau est replié et n'affiche qu'une seule colonne grisée, intitulée `(n folded occurrences)` où `n` est le nombre d'occurrences, et dont les cellules contiennent « ... ». Le tableau reste ainsi lisible lorsqu'un enregistrement répète un grand nombre de champs.

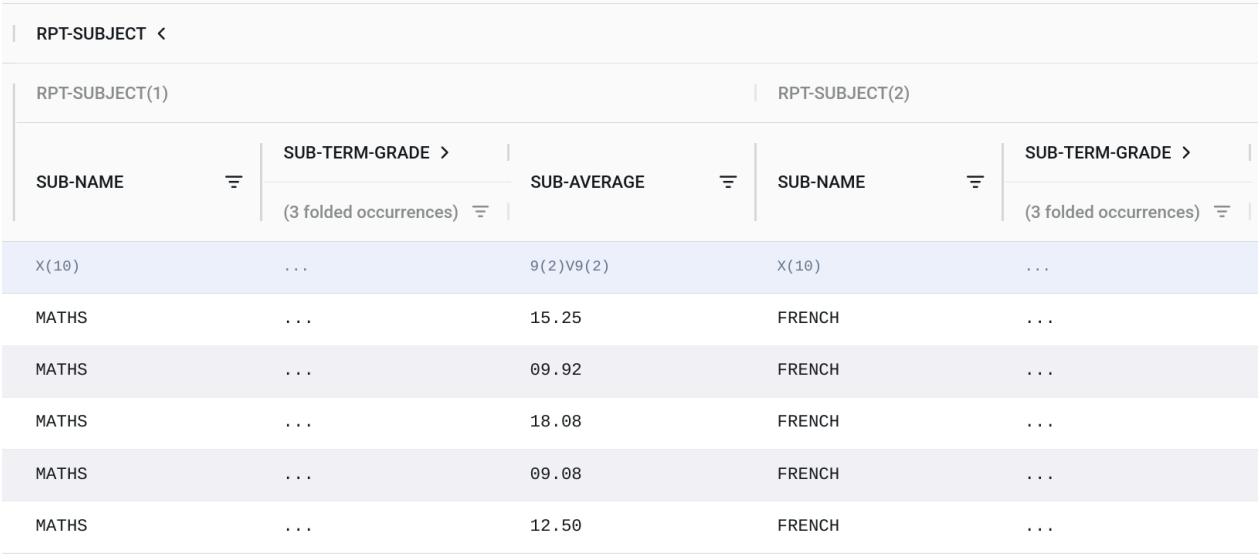


RPT-TERM-AVERAGE		
RPT-TERM-AVERAGE(1)	RPT-TERM-AVERAGE(2)	RPT-TERM-AVERAGE(3)
9(2)V9(2)	9(2)V9(2)	9(2)V9(2)
14.00	14.25	15.00
09.75	11.17	11.50
16.75	17.00	18.13
13.50	00.00	14.08
13.67	13.92	14.83

Figure 34. – Le même tableau `OCCURS` déplié : une colonne par occurrence.

Un clic n'importe où sur l'en-tête du groupe le déplie, et un nouveau clic le replie. Déplié, le groupe remplace cette colonne unique par une colonne par occurrence.

Les en-têtes de ces colonnes reprennent le nom du champ suivi de son indice d'occurrence. Les occurrences restent côte à côte et dans l'ordre de leurs indices lorsque les colonnes sont déplacées.

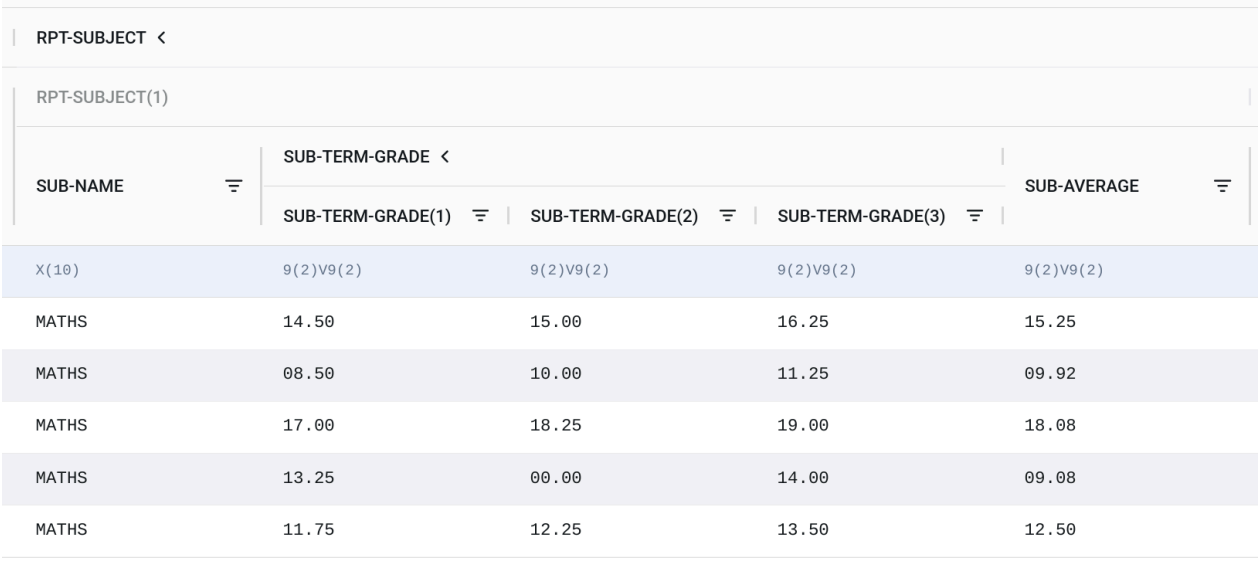


RPT-SUBJECT

RPT-SUBJECT(1)			RPT-SUBJECT(2)	
SUB-NAME	SUB-TERM-GRADE (3 folded occurrences)	SUB-AVERAGE	SUB-NAME	SUB-TERM-GRADE (3 folded occurrences)
X(10)	...	9(2)V9(2)	X(10)	...
MATHS	...	15.25	FRENCH	...
MATHS	...	09.92	FRENCH	...
MATHS	...	18.08	FRENCH	...
MATHS	...	09.08	FRENCH	...
MATHS	...	12.50	FRENCH	...

Figure 35. – Imbrication de tableaux `OCCURS`.

Lorsqu'un tableau `OCCURS` en contient un autre, chaque occurrence du groupe extérieur possède sa propre copie du groupe intérieur, repliée elle aussi à l'ouverture du fichier. Chaque copie se déplie et se replie indépendamment des autres.



RPT-SUBJECT

RPT-SUBJECT(1)

SUB-NAME	SUB-TERM-GRADE			SUB-AVERAGE
	SUB-TERM-GRADE(1)	SUB-TERM-GRADE(2)	SUB-TERM-GRADE(3)	
X(10)	9(2)V9(2)	9(2)V9(2)	9(2)V9(2)	9(2)V9(2)
MATHS	14.50	15.00	16.25	15.25
MATHS	08.50	10.00	11.25	09.92
MATHS	17.00	18.25	19.00	18.08
MATHS	13.25	00.00	14.00	09.08
MATHS	11.75	12.25	13.50	12.50

Figure 36. – Le tableau `OCCURS` intérieur déplié dans la première occurrence du tableau extérieur.

Le filtre d'une colonne d'occurrence ne s'applique qu'à cette occurrence. La recherche globale, qui porte sur l'ensemble des colonnes, porte en revanche sur toutes les occurrences.

Dans la fenêtre de détails d'un enregistrement, les noms de champs conservent leur indice d'occurrence.

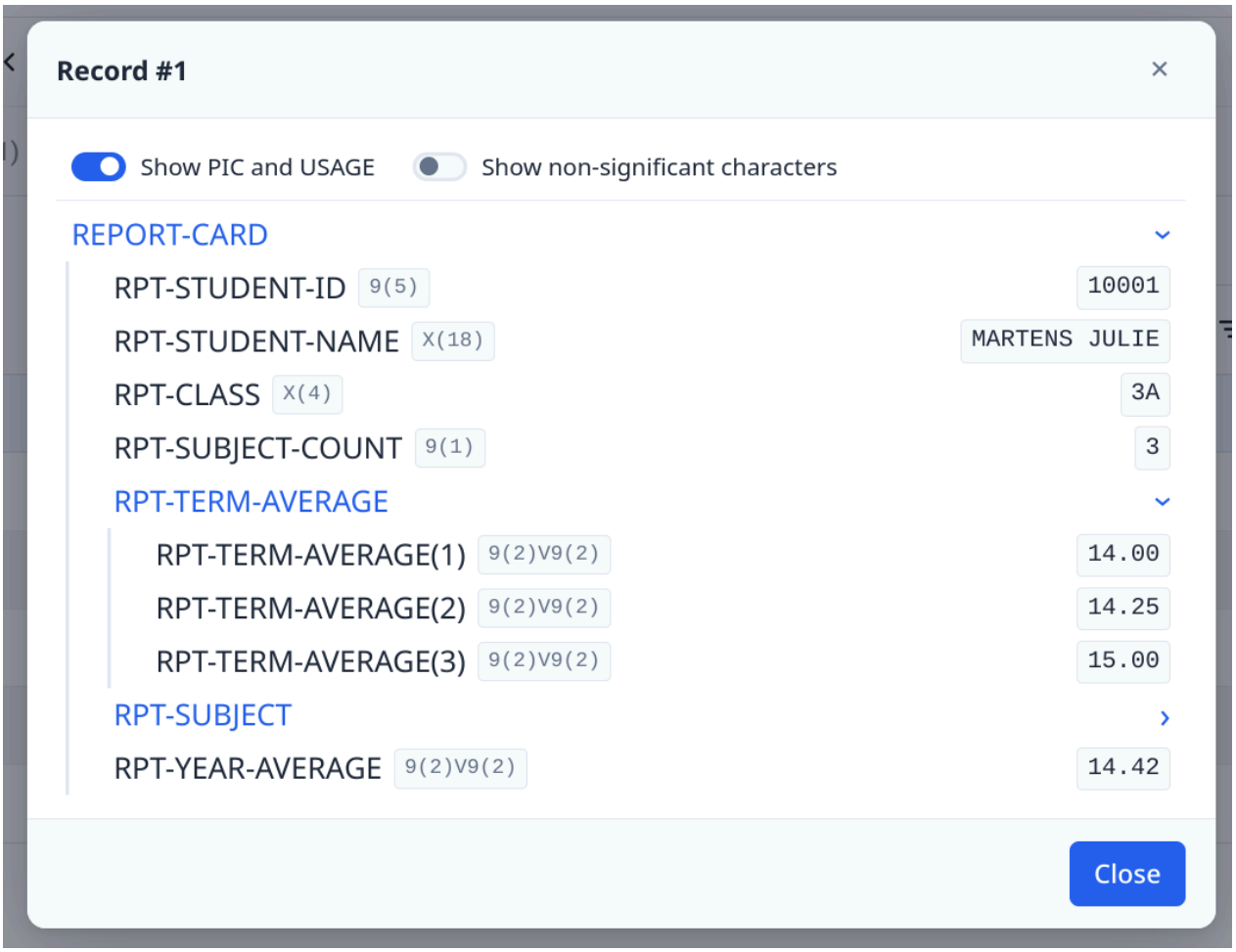


Figure 37. – Fenêtre de détails d'un enregistrement : les occurrences y conservent leur indice.

13.5 Visualisation des données

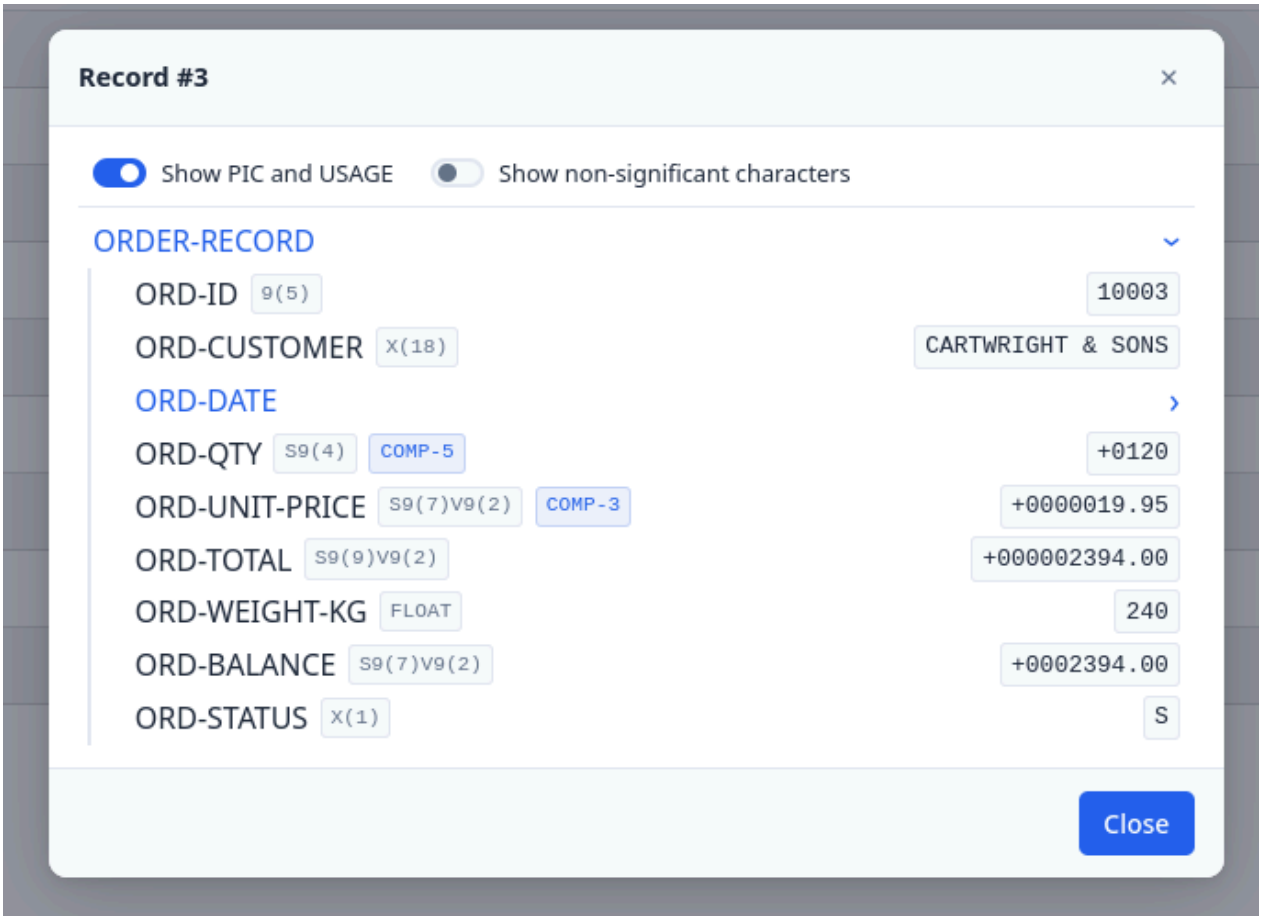


Figure 38. – Détails d'un enregistrement. Chaque champ présente des étiquettes sur le type de donnée PIC, l'encodage et la valeur associée tout à droite.

En double-cliquant sur une ligne du tableau, une fenêtre s'ouvre pour afficher les détails de l'enregistrement associé. La structure de l'enregistrement présente les données de façon arborescente pour distinguer visuellement les différents groupes. Les groupes sont rétractés à l'ouverture de la fenêtre : cliquer sur le nom d'un groupe le déplie, et un nouveau clic le rétracte.

En haut de la fenêtre se trouvent des boutons de formatage :

- un bouton `Show PIC and USAGE` permet d'afficher ou de cacher les étiquettes, ce qui permet notamment d'alléger visuellement l'interface;
- un bouton `Show non-significant characters` permet d'afficher les caractères non-significatifs dans les valeurs à droite : les espaces en tête et en queue des valeurs textuelles sont représentés par des points médians (·) et les zéros en tête des valeurs numériques sont

grisés. Les signes sont également affichés à la position où ils sont stockés, en tête ou en queue de la valeur.

13.6 Formatage

Figure 39. – Boutons de formatage des valeurs des enregistrements.

En haut à gauche de la fenêtre principale se trouvent des boutons permettant de formater les données.

De manière similaire à la fenêtre précédente de détails des données, il est possible d'afficher ou de cacher les caractères non significatifs avec l'interrupteur `Show non-significant characters`. Il s'agit du même réglage dans les deux fenêtres : le modifier dans l'une le modifie dans l'autre. Comme la sélection des colonnes, l'état de cet interrupteur et celui de `Show PIC and USAGE` sont conservés pour les ouvertures ultérieures de la visionneuse.

Avec l'interrupteur `Show only records with errors`, il est aussi possible de ne conserver que les enregistrements comportant des erreurs (notamment de typage, lorsque les valeurs ne correspondent pas au type PIC spécifié). Cela permet de repérer facilement les données à corriger. Les enregistrements comportant des erreurs sont surlignés dans le tableau en permanence, que cet interrupteur soit activé ou non.

13.7 Navigation dans les données

Figure 40. – Interface de navigation dans les enregistrements en bas de la fenêtre principale.

En bas de la fenêtre principale se trouve une interface de navigation. Dans l'ordre :

- un champ `Go to record` permettant de sauter à un numéro spécifique d'enregistrement. Entrer `5` par exemple et cliquer sur `Go` permettra d'aller directement au cinquième enregistrement (qui sera surligné). La touche `Entrée` valide également la saisie, et un numéro d'enregistrement en dehors des bornes du fichier est ignoré;
- une liste déroulante `Page Size` permettant de choisir le nombre d'enregistrements qui s'affichent par page;
- un indicateur des numéros d'enregistrements actuellement affichés par rapport au total (`1 to 10 of 10` sur la capture d'écran);
- un indicateur de la page actuelle par rapport au nombre total de pages (`Page 1 of 1` sur la capture d'écran), entouré de boutons pour passer à la page suivante / précédente ou au début / à la fin du fichier.

13.8 Filtrage

13.8.1 Filtrage de l'ensemble colonnes

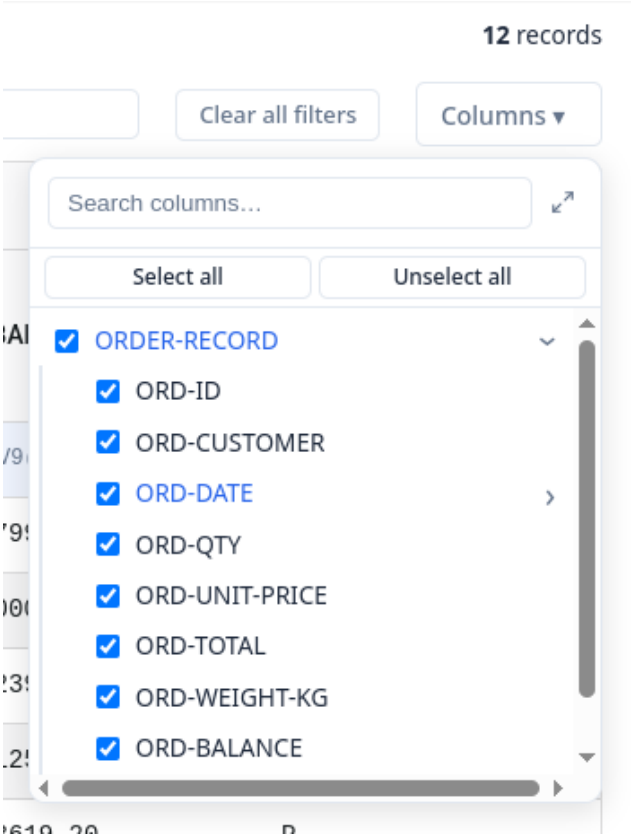


Figure 41. – Panneau de gestion de l'affichage des colonnes.

En haut à droite de la fenêtre principale se trouve un bouton `Columns` permettant d'ouvrir un panneau flottant avec la liste des champs. Cette liste suit la structure de l'enregistrement : les champs d'un groupe apparaissent indentés sous son nom.

Il devient ainsi possible de sélectionner quelles colonnes sont affichées ou cachées en cochant ou décochant les cases correspondantes. Cocher ou décocher un groupe s'applique à l'ensemble de ses champs, et la case d'un groupe partiellement affiché est dans un état intermédiaire. Des boutons `Select all` et `Unselect all` sont aussi présents pour sélectionner et désélectionner l'ensemble des colonnes.

Décocher un groupe `OCCURS` masque également la colonne affichée lorsqu'il est replié, de sorte que le groupe disparaît entièrement du tableau.

La sélection est conservée pour les ouvertures ultérieures de la visionneuse, et le nombre de colonnes cachées est indiqué à côté du bouton `Columns`.

Une barre de recherche en haut du panneau permet de chercher des colonnes spécifiques.

Astuce

Dans le cas où l'on voudrait afficher seulement un petit nombre de colonnes précises, on pourrait donc tout décocher avec `Unselect all` et chercher les colonnes à cocher avec la barre de recherche.

Un bouton « plein écran » en haut à droite de la fenêtre flottante permet aussi d'avoir une vue plus large et confortable dans le cas où il y aurait une grande profondeur des données.

13.8.2 Filtrage des données d'une colonne

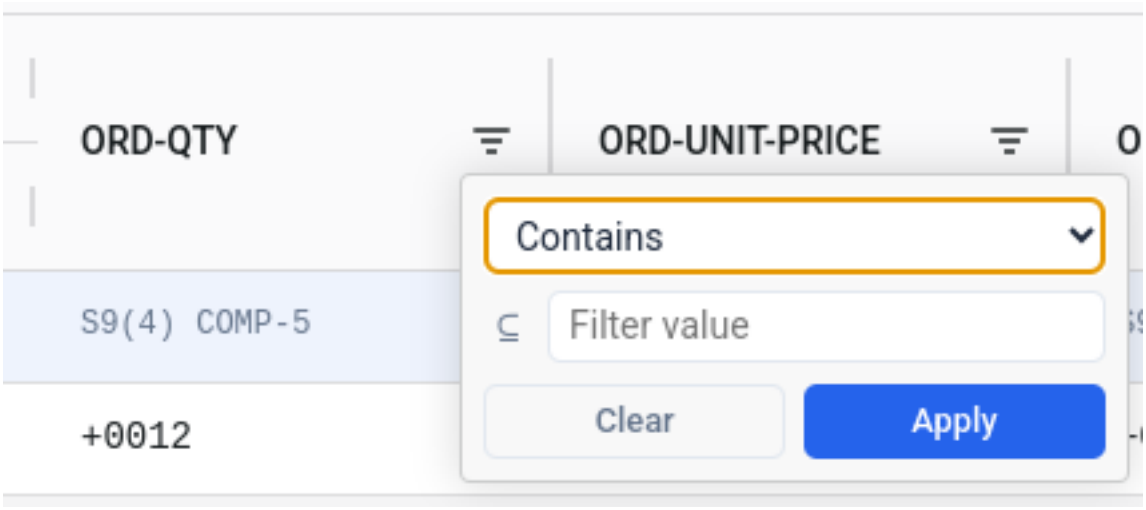


Figure 42. – Fenêtre flottante de filtrage des données d'une colonne.

Chaque colonne possède une icône « entonnoir » à droite de son nom permettant de filtrer les données affichées selon certains critères.

Quel que soit le type du champ, les filtres sont :

- `Contains` pour ne garder que les enregistrements dont la valeur à la colonne sélectionnée contient le texte entré;
- `Starts with` lorsque la valeur commence par l'entrée;
- `Ends with` lorsque la valeur termine par l'entrée;
- `Equals` lorsque la valeur est identique à l'entrée;
- `Blank` lorsque la valeur est vide;
- `Not blank` lorsque la valeur est non vide.

Les comparaisons sont sensibles à la casse. Le filtre est appliqué en cliquant sur `Apply`, et `Clear` le supprime.

13.8.3 Recherche globale

Figure 43. – Barre de recherche globale accompagnée d'un bouton de remise à zéro de tous les filtres.

Une barre de recherche est disponible en haut à droite de la fenêtre principale. Elle permet d'effectuer une recherche de valeur dans l'ensemble des colonnes du fichier de données (de la même manière que l'on rechercherait un mot-clé manuellement dans le filtre associé à chaque colonne) : un enregistrement est conservé dès que l'un de ses champs contient le texte recherché, là aussi de manière sensible à la casse. La validation de la recherche se fait sur la touche `Entrée`.

La barre est suivie d'un bouton `Clear all filters` qui remet à zéro tous les filtres, que ce soit la recherche ou les filtres propres aux colonnes.

14 Références SuperBOL Studio

14.1 Configuration de SuperBOL Studio

Plusieurs fichiers de configuration sont utilisés par VSCode :

- Par défaut, les options sont prises dans la configuration de l'utilisateur dans `$HOME/.config/Code/User/settings.json` (sous Linux)
- Dans un espace de travail, les options proviennent de la configuration de l'espace de travail dans `.vscode/settings.json`
- De plus, le serveur LSP de SuperBOL Studio utilisera également les options de `superbol.toml` à la racine de l'espace de travail, de préférence aux options des fichiers `settings.json`. Ce fichier est également utilisé par d'autres outils de SuperBOL qui ne sont pas liés à VSCode.

La commande **SuperBOL : Write Project Configuration** peut être utilisée pour exporter la configuration du fichier `settings.json` vers le fichier `superbol.toml`.

14.2 Les options SuperBOL dans `settings.json`

- **superbol.cobol.dialect** (chaine de caractères, défaut : `default`) : Dialecte COBOL par défaut pour tous les fichiers

  Les dialectes disponibles sont:

  - **default** et **gnucobol** : un dialecte global incluant tous les autres dialectes
  - **cobol85** : seulement les mots-clés de ANSI COBOL 85
  - **cobol2002** : seulement les mots-clés de COBOL ISO 2002
  - **cobol2014** : uniquement les mots-clés de COBOL ISO 2014
  - **acu** et **acu-strict** : le dialecte ACUCOBOL-GT
  - **bs2000** et **bs2000-strict** : dialecte pour le compilateur COBOL2000 V1.5 de Fujitsu
  - **gcos** et **gcos-strict** : dialecte pour COBOL sur les mainframes BULL GCOS
  - **ibm** et **ibm-strict** : dialecte pour COBOL sur les mainframes IBM z/OS
  - **mf** et **mf-strict** : dialecte pour Visual COBOL de MicroFocus (RocketSoftware maintenant)
  - **mvs** et **mvs-strict** : dialecte pour IBM COBOL pour MVS & VM
  - **realia** et **realia-strict** : dialecte pour CA Realia II
  - **rm** et **rm-strict** : dialecte pour RM-COBOL

- **xopen** : dialecte pour X/Open COBOL

La version `-strict` d'un dialecte permet d'utiliser des mots réservés de tous les autres dialectes comme des mots COBOL personnalisés, car la version non stricte réserve également ces mots.

Vous pouvez également spécifier le chemin absolu vers un fichier de configuration de dialecte GnuCOBOL personnalisé (voir l'option `conf` de `cobc`).

(Peut être surchargé dans les fichiers `superbol.toml` spécifiques au projet)

- **superbol.cobol.sourceFormat** (chaine de caractère, défaut : `auto`) : Format des sources par défaut.

  Les formats disponibles sont:

  - **auto** : SuperBOL essaiera de découvrir le format de manière automatique.
  - **fixed** : l'ancien format, avec une marge de gauche non significative de 6 caractères, un indicateur (`*` pour les commentaires) et une marge droite non significative après la colonne 72.
  - **cobol85** : même chose que `fixed`, mais avec une vérification stricte de la zone A.
  - **variable** : Identique à `fixed`, mais la marge de droite non significative commence après la colonne 252
  - **xcard** : Format ICOBOL xCard. Identique à `fixed`, mais la marge de droite non significative commence après la colonne 255.
  - **free** : le nouveau format libre, sans marges ni limitations. Les commentaires sont introduits par `*>` (sauf dans le dialecte MicroFocus où un `*` doit être placé dans la colonne 1)
  - **xopen** : Format libre X/Open. Le texte commence à la colonne 1, sauf si un indicateur est présent (`*` pour les commentaires, `D` pour le débogage), et les lignes peuvent contenir jusqu'à 80 caractères.
  - **crt** : ICOBOL Format libre. Identique à `xopen`, mais avec des lignes pouvant contenir jusqu'à 320 caractères.
  - **terminal** : Format terminal ACUCOBOL-GT. Identique à `crt`. Principalement compatible avec le format de terminal VAX COBOL.
  - **cobolx** : Indicateur dans la colonne 1, le texte commence à la colonne 2 et se termine à la colonne 255.

  (Peut être surchargé dans les fichiers `superbol.toml` spécifiques au projet)

- **superbol.cobol.copybooks** (tableau, défaut : `[ { "dir : "." }]`) : Chemins vers les copybook. Chaque objet du chemin contient au moins un répertoire dans le champ `dir`, et un champ optionnel `file-relative` (`true` par défaut).

  (Peut être surchargé dans les fichiers `superbol.toml` spécifiques au projet)

- **superbol.cobol.copyexts** (tableau, défaut : `["cpy", "cbl", "cob"]`) : Extensions de fichiers pour la résolution des copybooks.

  (Peut être remplacé dans les fichiers `superbol.toml` spécifiques au projet)

- **superbol.cobcPath** (string, default : « cobc ») : Exécutable du compilateur GnuCOBOL : Chemin d'accès à l'exécutable du compilateur GnuCOBOL

- **superbol.lspPath** (chaîne de caractères, défaut : "") : Exécutable SuperBOL

  Nom de l'exécutable `superbol-free` s'il est disponible dans PATH ; sinon, il peut s'agir d'un chemin absolu. Laisser vide pour utiliser l'exécutable `superbol-free`, s'il est disponible.

- **superbol.forceSyntaxDiagnostics** (booléen, défaut : false) : Force le rapport des diagnostics syntaxiques pour les dialectes autres que `COBOL85`

- **superbol.cacheInGlobalStorage** (booléen, défaut : false) : Utiliser le stockage stockage fourni par Visual Studio Code pour la mise en cache. Lorsque ce paramètre est défini sur *false*, le cache lié au contenu d'un dossier d'espace de travail donné *f* est stocké dans un fichier nommé *f*`/_superbol/lsp-cache`.

  ```
  Note : les fichiers de cache ne sont pas ne sont pas
  supprimés automatiquement, quel que soit leur emplacement.
  ```

- **superbol.debugger.displayVariableAttributes** (booléen, défaut : false) : Afficher les attributs des variables

  Affiche les propriétés de stockage et les attributs des champs (par exemple, la taille des caractères alphanumériques, des chiffres et position de la virgule des numériques).

- **superbol.debugger.libcobPath** (chaine de caractères, défaut : null) : Bibliothèque d'exécution de GnuCOBOL

  Chemin d'accès au fichier de la bibliothèque d'exécution de GnuCOBOL.

- **superbol.debugger.gdbPath** (chaine de caractères, défaut : « gdb ») : Exécutable du débogueur GNU

  Chemin d'accès à l'exécutable du débogueur GNU.

- **superbol.debugger.cobcrunPath** (chaine de caractères, défaut : « cobcrun ») : Programme d'exécution des modules GnuCOBOL

Chemin d'accès à `cobcrun`, le programme d'exécution des modules GnuCOBOL.

14.3 Options SuperBOL dans `superbol.toml`

Toutes les options de `settings.json` ne peuvent pas être définies dans `superbol.toml`. Les options suivantes peuvent être définies dans la section **`[cobol]`** du fichier TOML :

- **`dialect`** : Dialecte par défaut pour les fichiers sources COBOL
- **`source-format`** : Format de référence des sources par défaut
- **`copybooks`** : Où trouver les copybooks
- **`copyexts`** : Extensions des noms de fichiers des copybooks (par exemple `["cpy", "cob"]`)

14.4 Commandes SuperBOL

- **SuperBOL: Restart Language Server**:
- **SuperBOL: Write Project Configuration**: traduction des options du fichier `settings.json` dans le fichier `superbol.toml`.
- **SuperBOL: Show Coverage**:
- **SuperBOL: Hide Coverage**:
- **SuperBOL: Update Coverage**:
- **SuperBOL: Show Control-flow**:
- **SuperBOL: Show Control-flow as an Arc-diagram**:

15 Dépannage