Une image contenant personne

Description générée automatiquement

|  |
| --- |
| RÉfÉrentiels  **GoFSH et FHIR**  03/01/2023 |

Sommaire

[Introduction 3](#_Toc123646590)

[Installation 3](#_Toc123646591)

[Installation de Node.js 3](#_Toc123646592)

[Installation de GoFSH 3](#_Toc123646593)

[Mise à jour ou annulation de GoFSH 4](#_Toc123646594)

[API 4](#_Toc123646595)

[Syntaxe 4](#_Toc123646596)

[Paramètres 4](#_Toc123646597)

[Valeur de retour 5](#_Toc123646598)

[Usage 5](#_Toc123646599)

[Exécution de GoFSH 6](#_Toc123646600)

[Exécution 6](#_Toc123646601)

[Entrées GoFSH 8](#_Toc123646602)

[Sorties GoFSH 8](#_Toc123646603)

[Exemple GoFSH avec FHIR 8](#_Toc123646604)

[Étape 1 : Téléchargement des exemples d'artefacts FHIR 8](#_Toc123646605)

[Étape 2 : Exécution de GoFSH 9](#_Toc123646606)

[Étape 3 : Exécution de SUSHI (facultatif) 9](#_Toc123646607)

[Conclusion 9](#_Toc123646608)

# Introduction

Dans ce document nous faisons la présentation de l’outil « GoFSH » qui transforme les artefacts FHIR en définitions FSH.

À l'aide de « GoFSH », on peut transformer automatiquement un guide de mise en œuvre FHIR existant en un projet FSH.

Si on démarre un projet à partir de rien, il faut consulter le document « FHIR\_WITH\_SUSHI » pour l’initialisation d'un projet SUSHI.

# Installation

Dans cette section on définit les instructions pour installer l'outil GoFSH sur notre système.

## Installation de Node.js

GoFSH nécessite Node.js. Pour installer Node.js, rendez-vous sur <https://nodejs.org/> et on sélectionne le téléchargement « LTS ». Si le téléchargement ne convient pas au système d'exploitation, on clique sur le lien "Autres téléchargements" pour obtenir une liste complète des téléchargements. Une fois le programme d'installation téléchargé, on l’exécute en utilisant les options par défaut.

Pour s’assurer que « Node.js » est correctement installé, on ouvre une fenêtre de commande et en tape les deux commandes suivantes. Chaque commande doit renvoyer un numéro de version :

|  |
| --- |
| $ node --version |
| $ npm --version |

## Installation de GoFSH

Pour installer GoFSH, nous ouvrons une invite de commande et nous tapons la commande suivante :

|  |
| --- |
| $ npm install -g gofsh |

Après installation de GoFSH nous lançons la commande ci-dessous pour vérifier son installation :

|  |
| --- |
| $ gofsh –help |

Si la commande génère des instructions sur l'utilisation de l'interface de ligne de commande GoFSH (CLI), on est prêt à exécuter GoFSH.

Il faut utiliser « gofsh -v » pour afficher la version installée de GoFSH et la version de la spécification « FSH » qu'il prend en charge.

GoFSH suit la convention de version sémantique (MAJOR.MINOR.PATCH) :

* **MAJEUR** : Une version majeure a de nouvelles fonctionnalités importantes et, potentiellement, des changements de grammaire ou d'autres changements non rétro compatibles.
* **MINOR** : Contient des fonctionnalités nouvelles ou modifiées, tout en maintenant la rétro compatibilité dans la version majeure.
* **PATCH** : Contient des mises à jour mineures et des corrections de bogues, tout en maintenant la rétro compatibilité dans la version majeure.

## Mise à jour ou annulation de GoFSH

Pour mettre à jour GoFSH vers la dernière version, on exécute à nouveau :

|  |
| --- |
| $ npm install -g gofsh |

Pour revenir à une version précédente de GoFSH, on exécute :

|  |
| --- |
| $ npm install -g gofsh@**{**version**}** |

où la {version} est remplacée par la version « MAJOR.MINOR.PATCH » souhaitée (par exemple,

npm install -g [gofsh@1.5.0](mailto:gofsh@1.5.0)).

# API

Cette section décrit les instructions d'utilisation de l'API de GoFSH dans un projet

« JavaScript (JS)/TypeScript(TS) »

GoFSH expose une fonction « fhirToFsh » qui peut être utilisée pour convertir FHIR en FSH.

## Syntaxe

|  |
| --- |
| fhirToFsh**(**fhir**[,** options**])** |

## Paramètres

**fhir-** Un tableau de ressources FHIR, représentées sous forme de chaînes ou de JSON.

**options-** Un objet qui peut avoir n'importe quelle combinaison des attributs suivants :

* **dependencies-** Un tableau de chaînes utilisé pour spécifier les dépendances requises pour le traitement du FHIR. Les dépendances doivent utiliser le format : <packageId>@<version>(exemple : [hl7.fhir.us.core@3.0.1](mailto:hl7.fhir.us.core@3.0.1)).
* **logLevel-** Une chaîne qui spécifie le niveau de journalisation à utiliser lors du traitement de FHIR. Les options sont **silent**, **debug**, **info**, **warn** et **error**.
* **style-** Une chaîne représentant le style de la sortie renvoyée. Les options sont :
* **string-** Le FSH généré sera retourné en une seule chaîne. C'est la valeur par défaut.
* **map-** Le FSH généré sera renvoyé sous forme d'objet. Les attributs de l'objet sont :

1. **aliases-** Une chaîne contenant toutes les **Alias** définitions.
2. **profiles-** Une **Map** contenant toutes les **Profile** définitions sous forme de valeurs.
3. **extensions-** Une **Map** contenant toutes les **Extension** définitions sous forme de valeurs.
4. **codeSystems-** Une **Map** contenant toutes les **CodeSystem** définitions sous forme de valeurs.
5. **valueSets-** Une **Map** contenant toutes les **ValueSet** définitions sous forme de valeurs.
6. **instances-** Une **Map** contenant toutes les **Instance** définitions sous forme de valeurs.
7. **invariants-** Une **Map** contenant toutes les **Invariant** définitions sous forme de valeurs.
8. **mappings-** Une **Map** contenant toutes les **Mapping** définitions sous forme de valeurs.

Pour chaque **Map**, les clés sont le nom de la définition FSH. Par exemple, si la définition était :

|  |
| --- |
| Profile: MyPatient  Parent: Patient |

La clé serait MyPatient.

## Valeur de retour

Une **Promise** qui se résout en un objet avec les attributs suivants :

* **fsh**- Le FSH généré, stylisé selon le **style** paramètre.
* **configuration**- Un objet représentant le **sushi-config.yaml** fichier qui serait généré si GoFSH s'exécutait dans une interface de ligne de commande.
* **errors**- Un tableau de chaînes contenant toutes les erreurs détectées lors du traitement.
* **warnings**- Un tableau de chaînes contenant tous les avertissements détectés lors du traitement.

## Usage

Pour utiliser **fhirToFsh**, on doit d'abord installer **gofsh** en tant que dépendance de notre projet :

|  |
| --- |
| $ npm install gofsh |

Une fois **gofsh** installée en tant que dépendance de notre projet, nous pouvons importer et utiliser cette fonction comme indiqué :

|  |
| --- |
| **import** **{** gofshClient **}** from 'gofsh'**;**  *// Example basic usage*  gofshClient  **.**fhirToFsh**([**'{ Your FHIR here }'**])**  **.**then**((**results**)** **=>** **{**  *// handle results*  **})**  **.catch((**err**)** **=>** **{**  *// handle thrown errors*  **});**  *// Example usage with options*  gofshClient  **.**fhirToFsh**([**'{ Your FHIR here }'**],** **{**  dependencies**:** **[**"hl7.fhir.us.mcode@1.0.0"**],**  style**:** "map"**,**  logLevel**:** "silent"**,**  **})**  **.**then**((**results**)** **=>** **{**  *// handle results*  **})**  **.catch((**err**)** **=>** **{**  *// handle thrown errors*  **});** |

# Exécution de GoFSH

## Exécution

« GoFSH » est exécuté à partir de la ligne de commande. La forme générale de la commande d'exécution GoFSH est la suivante :

|  |
| --- |
| $ gofsh **{**input-directory**}** **{**options**}** |

Pendant que « GoFSH » est en cours d'exécution, il imprimera au fur et à mesure les messages d'état lors du traitement des fichiers du projet. Les sections suivantes donnent plus de détails sur l'utilisation de certaines options.

--alias-file :

Nous utilisons cette option pour fournir à « GoFSH » un fichier d'alias existant.

Le <path>peut être relatif ou absolu. En règle générale, « GoFSH » génère automatiquement un fichier **aliases.fsh** en fonction du contenu et des URL rencontrés lors du traitement.

À l'aide de l' option **--alias-file( -a)**, l'utilisateur peut spécifier un fichier FSH (.fsh) existant contenant les alias souhaités définis par l'utilisateur.

Example usage:

|  |
| --- |
| gofsh /path/to/my/ig --alias-file /different/path/to/aliases.fsh |

--style

L' **style** option a quatre valeurs :

* **file-per-definition**: Chaque définition FSH autonome est écrite dans un fichier individuel qui est regroupé dans un dossier en fonction du type de définition FSH dont il s'agit. Seuls les alias sont combinés dans un seul fichier **aliases.fsh**. C'est le choix par défaut.
* **group-by-fsh-type**: Les définitions sont écrites dans des fichiers en fonction du type de définition FSH dont elles sont constituées (alias, profil, extension, etc.).
* **group-by-profile**: Les profils sont écrits chacun dans un fichier individuel. Les instances et les invariants qui se rapportent uniquement à un certain profil sont alors inclus dans le même fichier que ce profil. Les définitions restantes sont regroupées comme dans l' **group-by-fsh-type** option.
* **single-file**: Toutes les définitions sont écrites dans un seul fichier.

--fshing-trip

Il faut utiliser cet indicateur si on souhaite s’assurer que GoFSH ne manque rien, en faisant un aller-retour de FHIR à FSH et vise versa. Lorsque cet indicateur est présent, après l'exécution de GoFSH, SUSHI s'exécutera sur la sortie de GoFSH. La sortie de SUSHI sera ensuite comparée à l'entrée d'origine de GoFSH (FHIR est comparé à FHIR), et une visualisation des différences entre l'entrée d'origine et la sortie SUSHI sera créée dans **<output-folder>/fshing-trip-comparison.html**. Si l'indicateur **--installed-sushi** est défini, ce processus utilisera la version de SUSHI que nous avons globalement installée. Sinon, GoFSH utilisera sa propre version intégrée de SUSHI (qui peut ne pas être la dernière version disponible).

--indent

Lorsque l' option **--indent** est spécifiée, la sortie FSH tirera parti des règles d'indentation, le cas échéant. Cela amènera également les **CodeSystem** définitions à utiliser l'indentation dans les codes hiérarchiques , les **Concept** désignations et **Concept** les propriétés, le cas échéant.

--meta-profile

L'option **--meta-profile** peut être utilisée pour contrôler la manière dont les valeurs de **meta.profile** présentent dans les instances FHIR sont appliquées au mot clé **-InstanceOf** dans FSH. Dans FHIR, la valeur de **meta.profile** représente une affirmation selon laquelle l'instance est conforme au(x) profil(s) donné(s). Une interprétation de **meta.profile** est que l'instance FSH générée est un **InstanceOf** profil, ce qui entraînerait ensuite la vérification de SUSHI pour s'assurer que l'instance suivait les règles de ce profil. Une complication potentielle est que ce **meta.profile** peut avoir plusieurs valeurs, mais dans FSH, une instance peut être **InstanceOf** au plus d’un profil. Une interprétation différente est qu'il **meta.profile** ne s'agit que d'un indice qui n'ajoute aucune

information concrète, et donc l'instance FSH doit ignorer **meta.profile** et doit être **InstanceOf** une ressource FHIR native. Les deux interprétations sont autorisées dans FHIR.

## Entrées GoFSH

GoFSH prend en entrée les définitions de structure FHIR et d'autres définitions de conformité FHIR (par exemple, ValueSets, CodeSystems).

GoFSH exige que ces fichiers soient JSON. Chaque fichier JSON contenu dans le répertoire d'entrée, ou ses sous-répertoires, sera traité par GoFSH en FSH.

GoFSH ne nécessite aucune configuration, mais si les artefacts FHIR d'entrée dépendent d'artefacts FHIR non contenus dans FHIR R4, ces dépendances doivent être spécifiées avec l' **-d** indicateur. Par exemple, le profil **mcode-cancer-patient** du **mCODE Implementation** Guide est dérivé du profil

**us-core-patient** du **US Core Implementation Guide** . Si nous souhaitons utiliser GoFSH pour convertir le profil **mcode-cancer-patient** en FSH, nous devons spécifier US Core comme dépendance :

|  |
| --- |
| $ gofsh ./mcode-definitions -d [hl7.fhir.us.core@3.1.0](mailto:hl7.fhir.us.core@3.1.0) |

## Sorties GoFSH

GoFSH remplit un répertoire de sortie, appelé **gofsh** par défaut. Ce répertoire contiendra un répertoire **input/fsh**. Le répertoire **input/fsh** contiendra plusieurs fichiers **.fsh** organisés selon le **file-per-definition** style décrit ci- dessus . De plus, le répertoire **input/fsh** contiendra un fichier **index.txt** qui décrit quel fichier contient chaque définition. Le répertoire **gofsh** contiendra également un fichier **sushi-config.yaml**. Si l'entrée de **GoFSH** inclut une ressource ImplementationGuide, elle est utilisée pour générer la configuration. Sinon, la configuration est générée en déduisant des valeurs à partir de l'entrée, ou en utilisant des valeurs par défaut raisonnables si aucune valeur ne peut être déduite. Ces définitions et configuration sont prêtes à être utilisées avec **SUSHI**, tout ce que nous avons à faire est de lancer SUSHI .

## Exemple GoFSH avec FHIR

Cette section nous guidera à travers un exemple d'utilisation de GoFSH pour transformer des artefacts FHIR en définitions FSH. Cette partie nécessite l’installation de GoFSH.

### Étape 1 : Téléchargement des exemples d'artefacts FHIR

Pour commencer avec quelques exemples d'artefacts FHIR, il faut télécharger le didacticiel « GoFSH » dans le lien suivant <https://fshschool.org/docs/gofsh/tutorial/gofsh-tutorial.zip> et le décompresser dans un répertoire de notre choix.

Une fois le fichier décompressé, nous devrions voir deux artefacts FHIR :

* **StructureDefinition-mcode-genetic-specimen.json**
* **StructureDefinition-mcode-laterality.json**

### Étape 2 : Exécution de GoFSH

Maintenant que GoFSH est installé et que nous échantillonnions des artefacts FHIR, il faut ouvrir une fenêtre de commande et accéder au répertoire contenant les artefacts que nous avons téléchargés. L’exécutions de GoFSH sur ces fichiers se fait par la commande :

|  |
| --- |
| $ gofsh . |

L'exécution de GoFSH va créer un répertoire **gofsh** et le sous-répertoire **input/fsh** , un répertoire contenant plusieurs fichiers qui définissent les fichiers d'entrée d'origine. Dans **input/fsh** , il y aura deux sous-répertoires, **profiles et extensions** . Si nous ouvrons **profiles/GeneticSpecimen.fsh** , nous devrions voir une définition FSH pour un Profile appelé «GeneticSpecimen», et si nous ouvrons **extensions/Laterality.fsh** , nous devrions voir une Extension appelé «**Laterality**».

### Étape 3 : Exécution de SUSHI (facultatif)

Maintenant que nous avons généré des définitions FSH, nous pouvons exécuter SUSHI sur ces définitions pour recréer notre entrée d'origine dans GoFSH.

Tout d'abord, assurons-nous que SUSHI est installé . Ensuite, naviguons on ligne de commande vers le répertoire **gofsh/** et exécutons :

|  |
| --- |
| $ sushi . |

Cette commande va exécuter **SUSHI** sur le contenu de **input/fsh** et va générer la sortie de ce FSH dans un répertoire appelé **fsh-generated/resources**. nous pouvons ensuite comparer la sortie de l'exécution de **GoFSH** puis de **SUSHI** aux fichiers originaux **StructureDefinition-mcode-genetic-specimen.json et StructureDefinition-mcode-laterality.json**.

Alternativement, nous pouvons exécuter automatiquement SUSHI sur la sortie de GoFSH en utilisant l'option **-f**.

La commande est la suivante va exécuter SUSHI sur la sortie de GoFSH et va générer un fichier de comparaison dans **gofsh/fshing-trip-comparison.html** qui affiche les différences entre l'entrée d'origine et la sortie de SUSHI :

|  |
| --- |
| $ gofsh . -f |

# Conclusion

Le projet GoFSH-FHIR est comité sur GitHub à l’adresse suivante : <https://github.com/nizarbs2508/GoFSH-FHIR>. Clonez l’exemple (<https://github.com/nizarbs2508/GoFSH-FHIR.git>), exécutez le et amusez vous à faire des modifications et voir les résultats.