Développement d'un serveur LSP pour le langage FML

MAMOU Antoine

IMT Atlantique - Équipe P4S

27 juin 2025

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Sommaire

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Objectifs du projet

- Mettre en place un serveur **Language Server Protocol (LSP)** pour le langage FML
- Développer des clients pour différents environnements
- Intégrer le serveur LSP dans l'environnement **OpenFlexo**
- Offrir des services facilitant le développement tels que la **complétion**, les **diagnostics** et le **hover**

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Qu'est-ce que le LSP ?

Le **Language Server Protocol** (LSP) est un protocole standardisé, basé sur **JSON-RPC**, permettant à un éditeur de code (ou IDE) de communiquer avec un serveur qui fournit des services facilitant le développement :

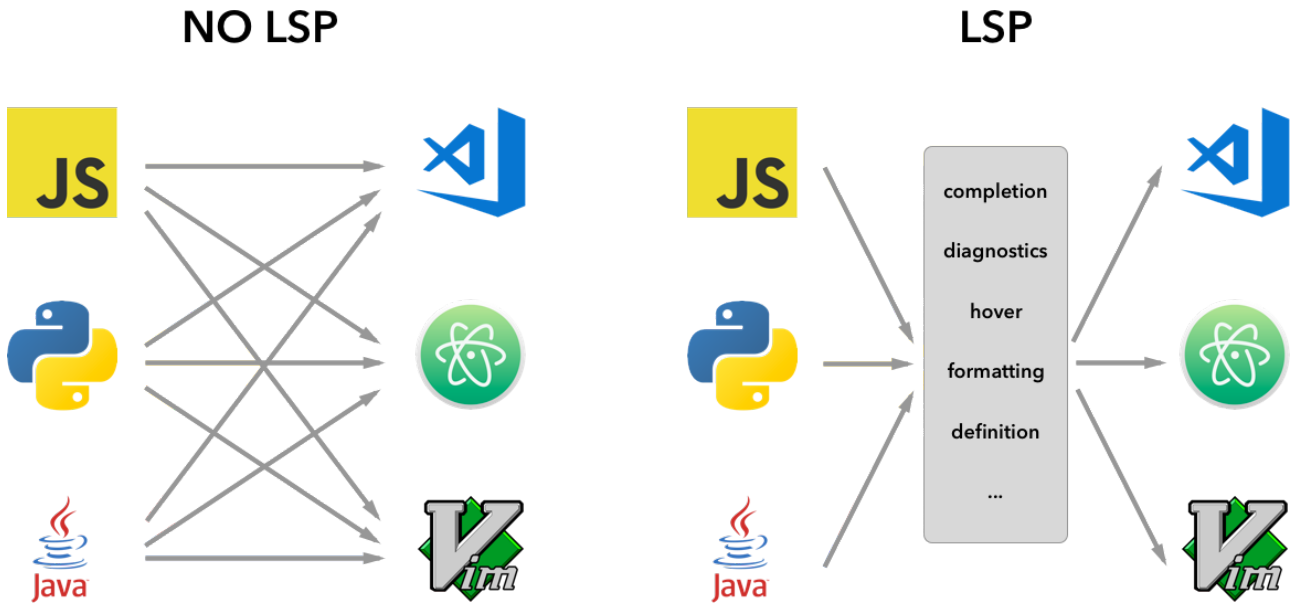
- Autocomplétion
- Navigation dans le code
- Diagnostics d'erreurs
- Informations contextuelles

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Pourquoi utiliser le LSP ?

Le LSP permet de développer un seul serveur pour un langage, utilisable par de nombreux éditeurs compatibles :

- **Un serveur unique** pour plusieurs IDE
- **Réduction des coûts** de développement et de maintenance
- **Interopérabilité** entre outils

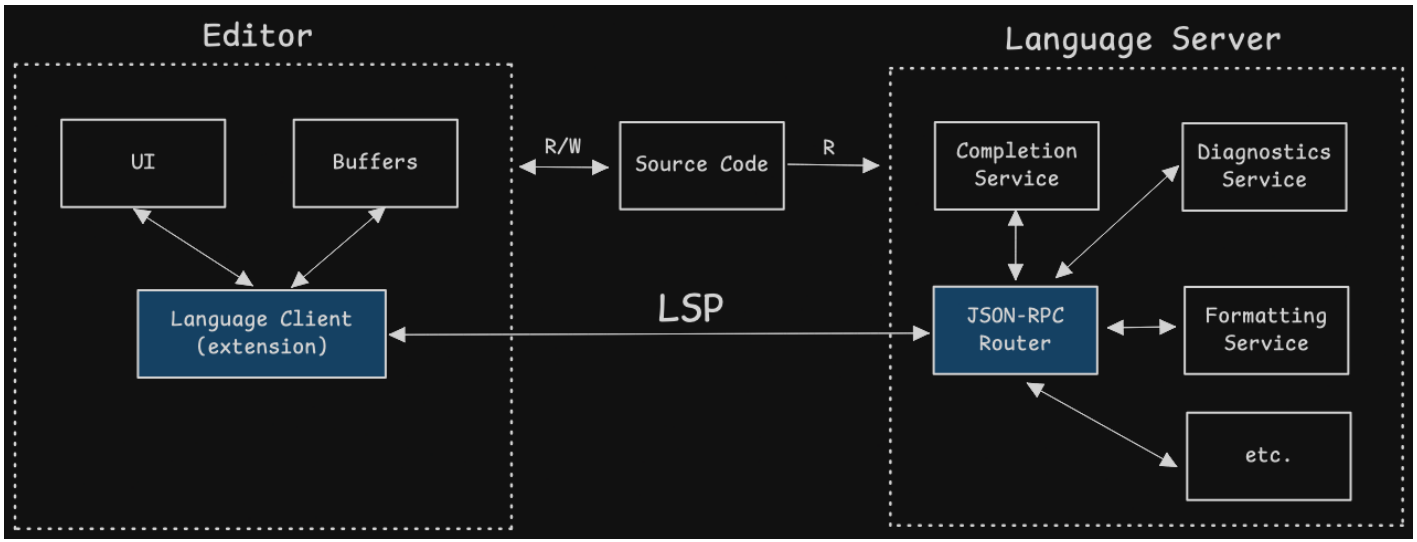


IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Architecture du LSP

Le **Language Server Protocol** repose sur une architecture client-serveur :

- Le client (éditeur/IDE) envoie des requêtes
- Le serveur fournit des réponses selon les services implémentés



IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Format JSON-RPC

Le protocole LSP repose sur **JSON-RPC**, un protocole léger et standardisé pour les échanges client-serveur.

Trois types de messages :

- **Requête** : attend une réponse (ex : completion)
- **Notification** : ne nécessite pas de réponse (ex : ouverture d'un fichier)
- **Réponse** : suite à une requête

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Exemple de requête JSON-RPC

```
{
  "jsonrpc": "2.0",
  "id": 3,
  "method": "textDocument/completion",
  "params": {
    "context": {
      "triggerKind": 1
    },
    "textDocument": {
      "uri": "file:///home/user/Documents/test.simple"
    },
    "position": {
      "line": 0,
      "character": 1
    }
  }
}
```

Figure 1 – Exemple de requête envoyée au format JSON-RPC pour une complétion

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Réponse JSON-RPC

```
{
  "jsonrpc": "2.0",
  "id": 3,
  "result": [
    {
      "label": "var",
      "kind": 14
    }
  ]
}
```

Figure 2 – Réponse à la requête JSON-RPC pour la complétion de code

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Initialisation du LSP

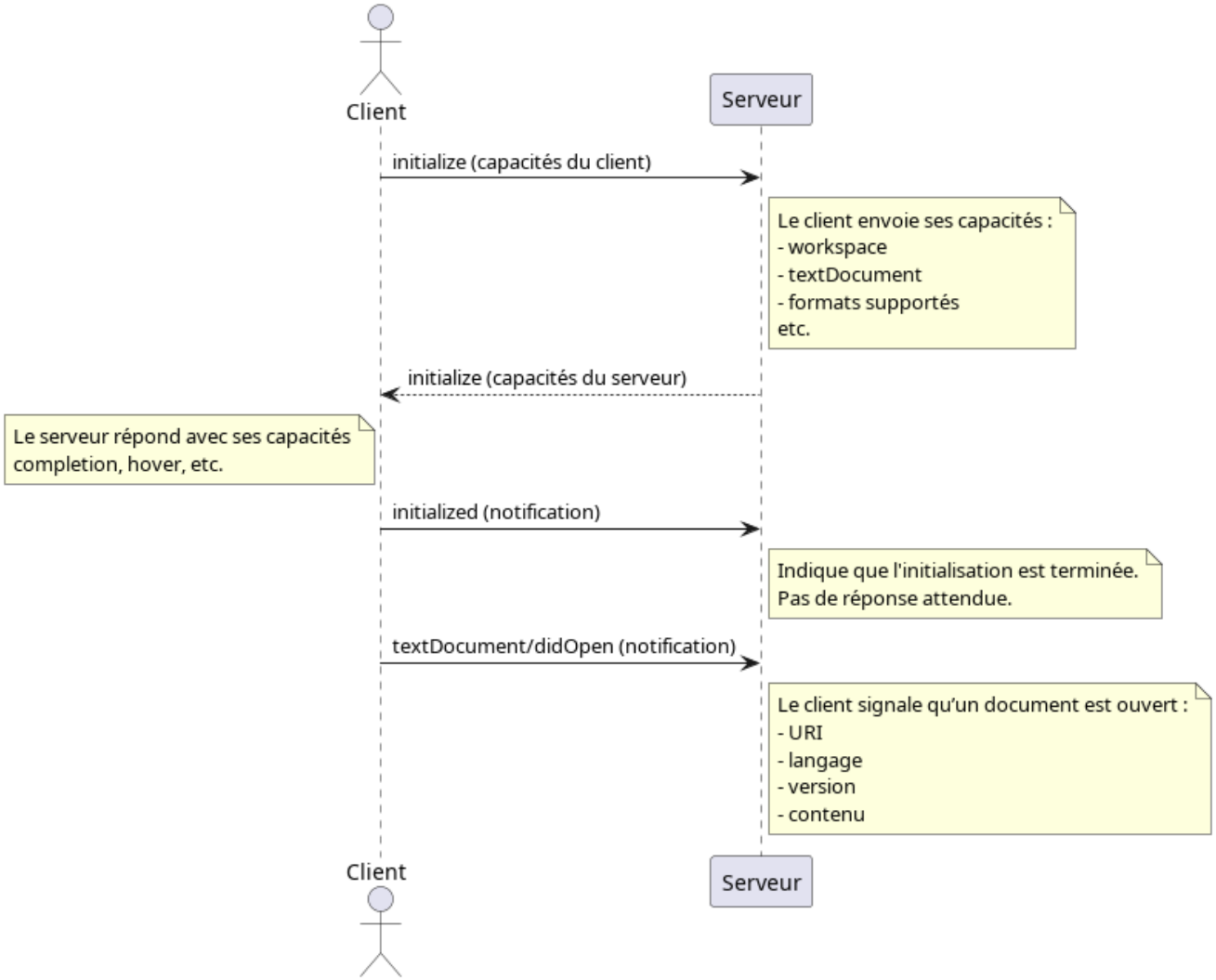


Figure 3 – Diagramme de séquence d'initialisation du LSP

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Exemple d'interaction client-serveur

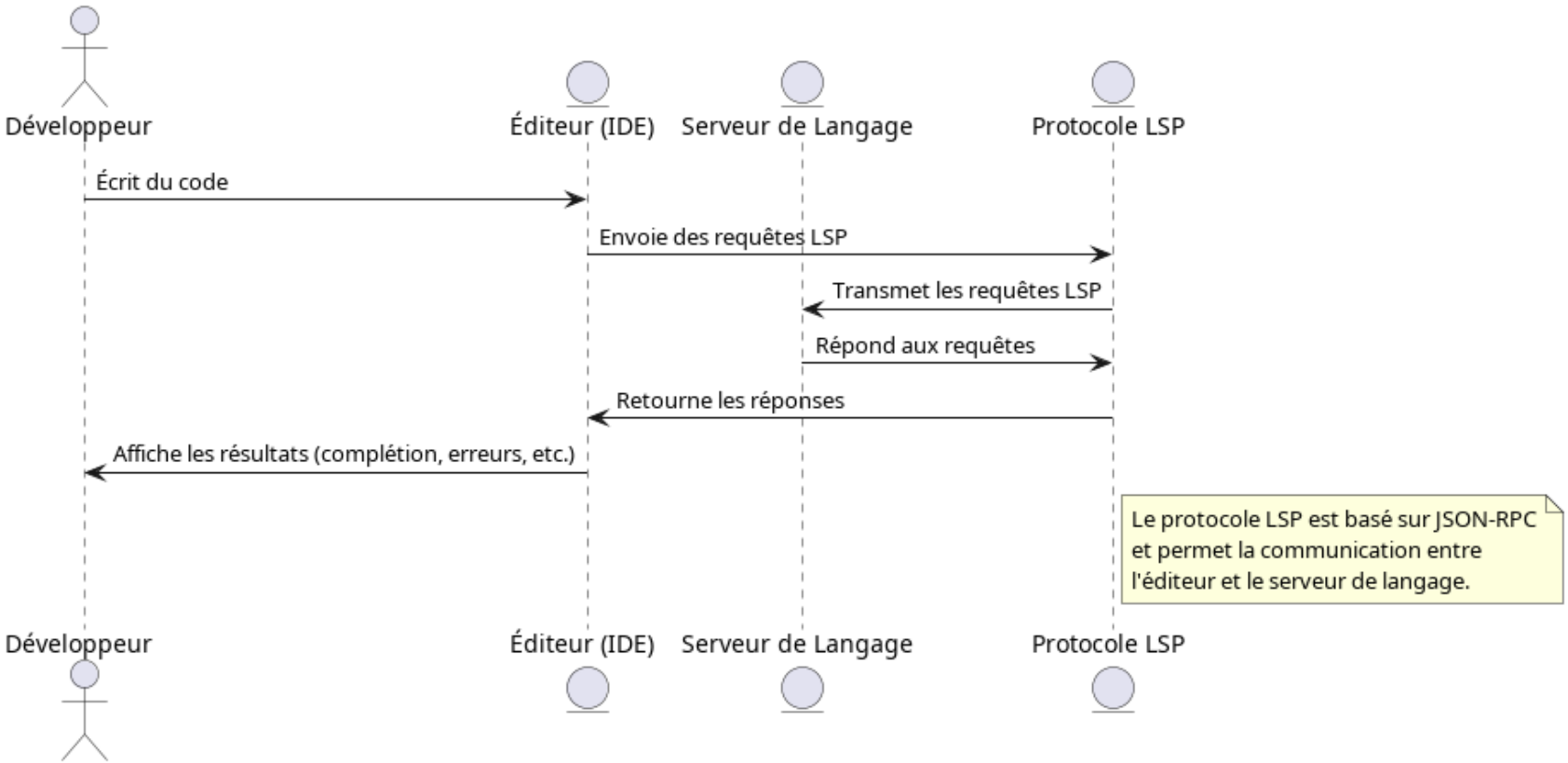


Figure 4 – Diagramme de séquence d'utilisation du protocole LSP

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Types de communication entre le client et le serveur

Le client peut communiquer avec le serveur LSP via différents canaux, selon l'environnement d'utilisation.

Stdin/Stdout :
Le client démarre le serveur en arrière-plan et échange des messages via les flux d'entrée et de sortie standard.

Socket TCP :
Le serveur ouvre un port réseau et attend qu'un client s'y connecte via une connexion TCP. Une fois connectés, le client et le serveur échangent des messages via les flux du socket.

WebSocket :
Dans les environnements web, la communication se fait via WebSocket, permettant une interaction asynchrone et en temps réel entre le client et le serveur.

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Travail exploratoire

Une partie du travail a consisté à comprendre et évaluer :

- Le fonctionnement du LSP
- Les bibliothèques disponibles
- Les différences d'implémentation entre clients (VSCode, Eclipse, Web)
- Les différentes façon de gérer la communication client-serveur

Ce travail a permis de construire une solution adaptée à openflexo, qui pourra facilement être amélioré par la suite.

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Création d'un langage jouet

Pour expérimenter une implémentation simple et isolée du protocole LSP, un **langage jouet** a été conçu spécifiquement.

Ce langage minimaliste a servi de terrain d'essai pour :

- Développer un serveur LSP de base
- Tester les interactions client-serveur
- Valider le fonctionnement des extensions pour différents clients (VS Code, Eclipse, Web)

Cette étape a permis d'identifier les contraintes et les bonnes pratiques avant de passer à l'intégration dans un environnement plus complexe.

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

implémentation du serveur

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Implémentation du serveur LSP

- Serveur développé en **Java**, avec la bibliothèque **LSP4J**
- Fournit une API pour créer un serveur compatible LSP

Trois classes principales :

- `FMLLanguageServer` : point d'entrée du LSP
- `FMLTextDocumentService` : gestion des documents
- `FMLWorkspaceService` : gestion de l'espace de travail

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Architecture modulaire du serveur

La logique est séparée par fonctionnalité via des classes appelées **providers**.

- `FMLTextDocumentService` reçoit les requêtes du client
- Elle délègue à des classes spécialisées : `CompletionProvider`, `HoverProvider`, etc.

Cette organisation améliore la lisibilité, la maintenabilité et l'évolutivité du code.

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Lancement du serveur LSP

1. **Démarrage** : Le serveur LSP est lancé (via un script, test, ou en ligne de commande).
2. **Ouverture d'un canal** :
   - **Stdin/Stdout** : communication classique en ligne de commande.
   - **WebSocket** : utilisé pour des environnements web.
   - **Socket TCP** : le serveur écoute sur un port réseau.
3. **Connexion du client** : l'éditeur ou IDE se connecte au serveur.
4. **Initialisation** : le client envoie un message `initialize`, le serveur répond avec ses capacités.
5. **Échange de messages** : le client et le serveur communiquent de manière asynchrone (complétion, hover, etc.).

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Implémentation des clients

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Implémentation côté client

Pour interagir avec le serveur, un client (IDE) doit intégrer une extension ou un plugin LSP.

Contrairement au serveur, le client est souvent spécifique à l'éditeur utilisé :

- API et méthode d'intégration différentes
- Gestion de l'interface utilisateur propre à chaque IDE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Clients développés pour FML

Trois clients ont été mis en place pour interagir avec le serveur LSP de FML :

- Une extension **VS Code**
- Un plugin **Eclipse**
- Un client web basé sur le **Monaco Language Client**

Chaque client utilise des technologies différentes, mais repose sur le même protocole LSP.

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Extension VS Code et plugin Eclipse

Extension VsCode :

- Développée en **TypeScript**, à l'aide de l'API officielle de VS Code.
- Utilise la bibliothèque `vscode-languageclient` pour se connecter au serveur LSP.
- Communication via stdin/stdout ou socket TCP (2 versions).

Plugin Eclipse :

- Développé en **Java**, basé sur `lsp4e` (extension LSP pour Eclipse).
- Intégré directement à l'environnement Eclipse.
- Démarre le serveur LSP comme un sous-processus et communique via **stdin/stdout**.

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Client Web (Monaco + WebSocket)

- Utilise l'éditeur **Monaco** (moteur de VS Code pour le Web).
- Communication avec le serveur LSP via une **WebSocket** grâce à une API exposée par un serveur **Spring Boot**.
- Côté serveur :
  - Le serveur démarre comme une application web classique.
  - Lorsqu'un éditeur web se connecte, une session de communication est ouverte.
  - Le serveur échange ensuite des messages avec le client pour fournir les différents services.
- Côté client :
  - Basé sur `monaco-languageclient` et `vscode-ws-jsonrpc`.
  - Intégré dans une application web.
  - Permet d'avoir l'expérience d'un IDE dans une application web

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Mise en place dans Openflexo

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Intégration du LSP dans Openflexo

L'éditeur d'OpenFlexo est capable de lancer automatiquement le serveur LSP pour FML.

- Le serveur démarre en écoutant un port
- Un éditeur externe (ex : VS Code) peut alors s'y connecter

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Lancement du serveur LSP dans openflexo

Le lancement du serveur consiste à démarrer un processus qui :

- Écoute sur un port réseau en attente de connexion depuis un éditeur.
- Accepte la connexion du client (éditeur ou IDE).
- Établit une communication bidirectionnelle via le protocole LSP.
- Démarre une boucle d'écoute qui attend et traite les messages envoyés par le client
  (par exemple, une demande de `completion` ou de `hover`).

Une fois connecté, le serveur reste actif en arrière-plan tant que le client est ouvert, prêt à répondre aux requêtes.

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Activation du client LSP

L'extension cliente est lancée à l'ouverture d'un fichier `.fml`.

Le client envoie ensuite une **requête d'initialisation** contenant :

- Capacités supportées
- Informations sur l'environnement
- URI du document

Coloration syntaxique :

- Basée sur un fichier `.tmLanguage.json`
- Inspirée de la grammaire Java (FML étant proche de Java)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Conclusion

- Le projet a permis de mieux comprendre le **fonctionnement du LSP** et ses différents modes d'intégration
- Trois clients fonctionnels ont été réalisés pour divers éditeurs
- Le serveur a été intégré dans **OpenFlexo**, ouvrant la voie à une amélioration continue de l'expérience développeur

Ces travaux permettent d'avoir une bonne base pour le développement futur autour du langage FML.

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom