

MCP

音楽CLINE

MCP SERVER MIDI COMPOSER



Who are you?

Yu Otsubo (tubone24)

ゆるふわエンジニア@KAG

**妻と犬の二人暮らし。
アスパラが苦手ですが、春になると
飲食店アスパラを使ったメニュー
ばっかりでうんざりしている。**

\$ npx tubone24



What is a MCP?

MCPとは AI（人工知能）モデルと外部
の間のための
橋渡しの略
MCPは B-C
ポートのような役割を果たします。

これは割愛



DEMO

> Task: 8小節の合唱曲を作って。若干マイナー調で、メロディックな音階を作ってほしい

\$0.0000 ×

Checkpoints

Continue [Icons]

API Request...



Task: 8小節の合唱曲を作って。若干マイナー調で、メロディックな音階を作ってほしい

\$0.0104 X

- メロディックな音階を使用

まず、合唱曲を作るためにMCPサーバーの「musicComposer」を使用します。このサーバーには「create_midi」というツールがあり、テキスト形式の音楽データからMIDIファイルを生成できます。

合唱曲なので、複数のボーカルパートを作成する必要があります。一般的な合唱の構成としては、ソプラノ、アルト、テナー、バスの4パートが考えられます。

若干マイナー調ということで、自然的マイナースケール（ナチュラルマイナー）ではなく、メロディックマイナースケールを使用します。メロディックマイナーは上行時に6度と7度を半音上げ、下行時は自然的マイナーに戻るスケールです。これにより、メロディアスな響きが得られます。

Aマイナーのメロディックマイナースケールを使用することにします：

- 上行時: A, B, C, D, E, F#, G#, A
- 下行時: A, G, F, E, D, C, B, A

8小節の簡単な構成を考えます：

- 導入部 (1-2小節)
- メロディの展開 (3-6小節)
- 終結部 (7-8小節)

それでは、MIDIファイルを生成するためのデータを作成しましょう。

MIDIファイルを作成するために、musicComposerのMCPサーバーを使用します。若干マイナー調でメロディックな音階を使った8小節の合唱曲を作成します。

Aマイナーのメロディックマイナースケールを基調とし、ソプラノ、アルト、テナー、バスの4声部で構成します。

Cline wants to use a tool on the `musicComposer` MCP server:

🔒 create_midi

Auto-approve

テキスト形式の音楽データからMIDIファイルを生成します。compositionが長くなる場合はinputSchemaを一度JSONファイルに書き出して読み込んで渡すようにすると安定します。

ARGUMENTS

```
{
  "title": "メロディックマイナーの合唱曲",
  "composition": {
    "bpm": 72,
    "timeSignature": {
      "numerator": 4
```

> Task: 8小節の合唱曲を作って。若干マイナー調で、メロディックな音階を作してほしい

50.0539 x

```
{
  { "pitch": 64, "beat": 13.0, "duration": "2", "velocity": 78 },
  { "pitch": 62, "beat": 15.0, "duration": "2", "velocity": 75 }
},
{
  "name": "テナー",
  "instrument": 53,
  "notes": [
    { "pitch": 60, "beat": 1.0, "duration": "4", "velocity": 75 },
    { "pitch": 59, "beat": 2.0, "duration": "4", "velocity": 72 },
    { "pitch": 60, "beat": 3.0, "duration": "4", "velocity": 75 },
    { "pitch": 62, "beat": 4.0, "duration": "4", "velocity": 78 },
    { "pitch": 64, "beat": 5.0, "duration": "2", "velocity": 80 },
    { "pitch": 65, "beat": 7.0, "duration": "2", "velocity": 78 },
    { "pitch": 64, "beat": 9.0, "duration": "2", "velocity": 75 },
    { "pitch": 62, "beat": 11.0, "duration": "2", "velocity": 72 },
    { "pitch": 60, "beat": 13.0, "duration": "2", "velocity": 75 },
    { "pitch": 57, "beat": 15.0, "duration": "2", "velocity": 72 }
  ]
},
{
  "name": "バス",
  "instrument": 53,
  "notes": [
    { "pitch": 45, "beat": 1.0, "duration": "1", "velocity": 85 },
    { "pitch": 48, "beat": 5.0, "duration": "1", "velocity": 82 },
    { "pitch": 52, "beat": 9.0, "duration": "1", "velocity": 80 },
    { "pitch": 45, "beat": 13.0, "duration": "1", "velocity": 85 }
  ]
}
],
"output_path": "/Users/kagadminmac/Documents/Cline/MCP/test/melodic_minor_chorus.mid"
}
```

Response

Rich Display ☒

「メロディックマイナーの合唱曲」のMIDIファイルを作成しました。ファイルは /Users/kagadminmac/Documents/Cline/MCP/test/melodic_minor_chorus.mid に保存されました。

</

これだけ

(暗すぎる)

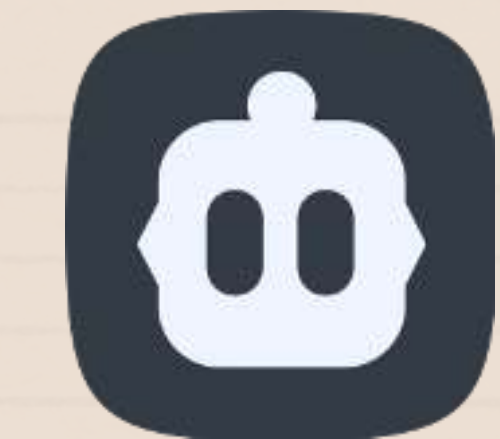
Conclusion

音楽生成はテキスト生成系の
LLMではなく、

素直に音楽生成AI使おう。



どういう仕組み？



Cline
(MCPクライアント)

LLM呼び出し



Amazon Bedrock
(Claude 3.7 sonnet)

ツールとして呼び出し



MCP Server
(studio on npm)



music-writer-js



tubone24/midi-mcp-server



MIDI MCP Server is a Model Context Protocol (MCP) server that enables AI models to generate MIDI files from text-based...



1

Contributor



0

Issues



0

Stars



0

Forks



自作して

わかったこと

注意

今回はMCPのStdIOのみ
Toolsに関してのみ。

promptsとかも面白そうではある

Sequences

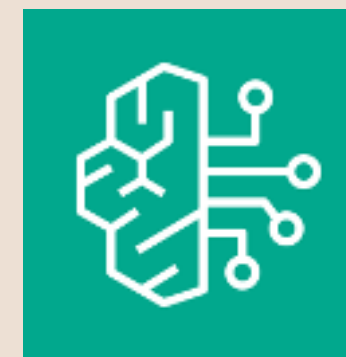
ざっくりシーケンス書くと
こんな感じ
(notificationなど省略)

MCPに規定されるメソッドの
ハンドラーを
MCPサーバーコードに表現
していく

StdIOなら直接JSをnodeから
実行すれば結構面白いことがわかる



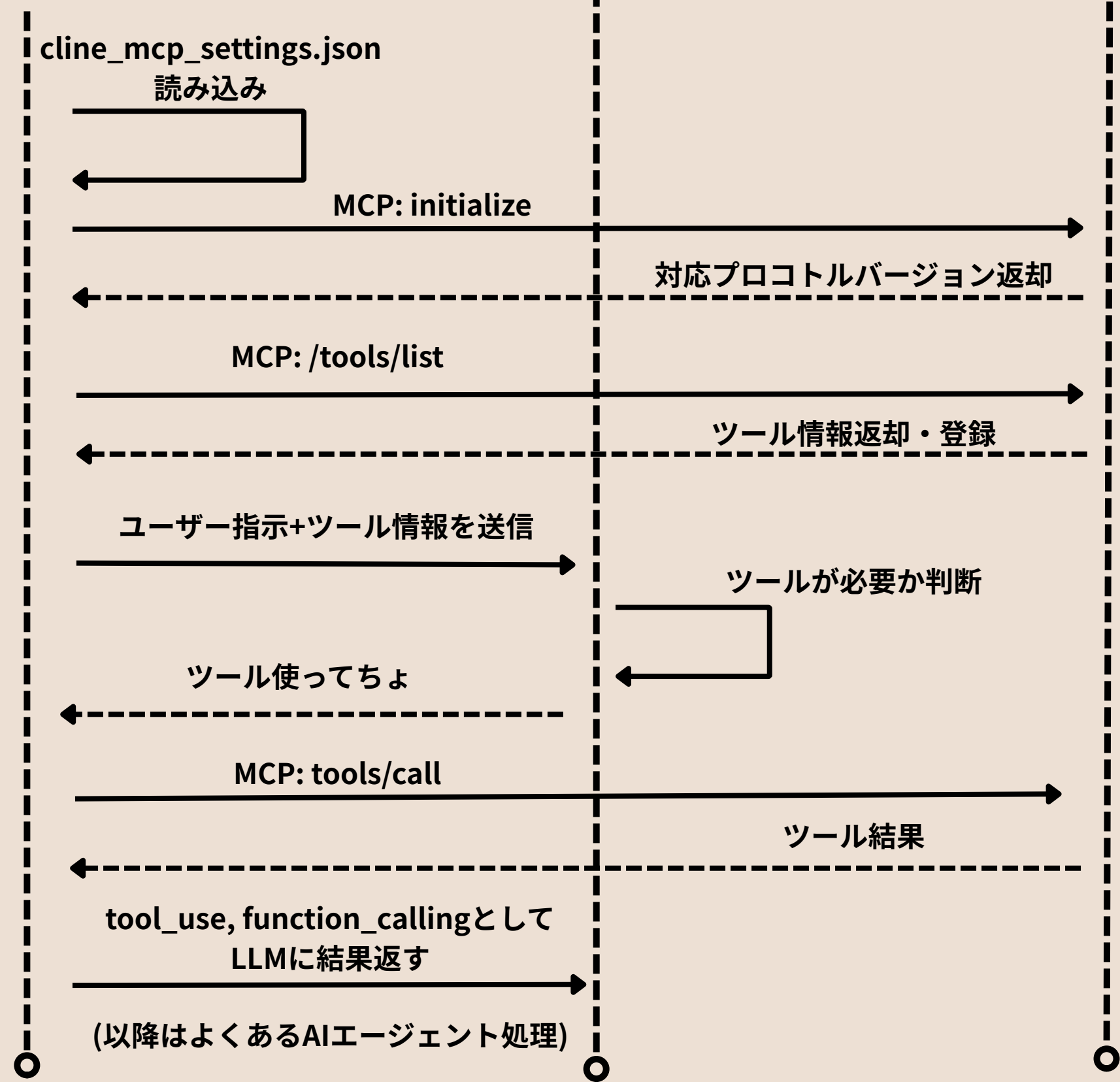
Cline
(MCPクライアント)



Amazon Bedrock
(Claude 3.7 sonnet)



MCP Server
(studio on npm)



MCP Server STUDIO

```
(anaconda3-2023.09-0) kagadminmac%
```

```
(anaconda3-2023.09-0) kagadminmac%
```

```
(anaconda3-2023.09-0) kagadminmac% node build/index.js
```

```
MIDI MCP serve{"jsonrpc":"2.0","id":1,"method":"initialize","params":{"protocolVersion":"2025-03-26","capabilities":{},"clientInfo":{"name":"my-client","version":"1.0.0"}}}  
{"result":{"protocolVersion":"2024-11-05","capabilities":{"tools":{},"batching":{"supported":true}},"serverInfo":{"name":"midi-mcp-server","version":"0.1.0"}}, "jsonrpc":"2.0","id":1}
```

```
█
```

```
[~/Documents/Cline/MCP/midi-mcp-server][main]
```

```
[~/Documents/Cline/MCP/midi-mcp-server][main]
```

```
[~/Documents/Cline/MCP/midi-mcp-server][main]
```

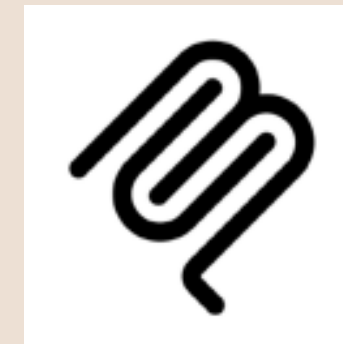




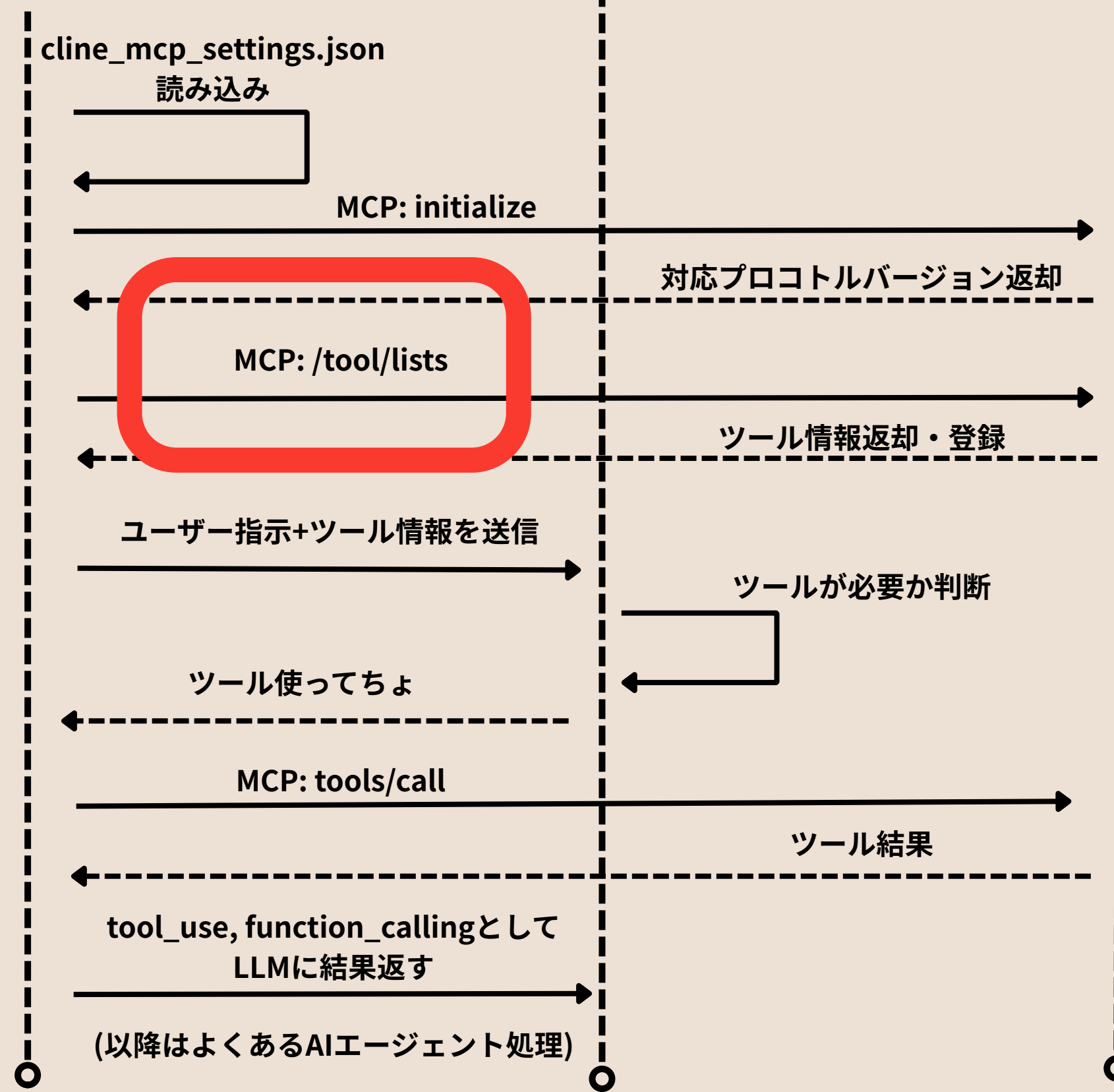
Cline
(MCPクライアント)



Amazon Bedrock
(Claude 3.7 sonnet)



MCP Server
(studio on npm)



Input Schema

ListToolsRequestSchemaで
次のようにスキーマを正しく
指定してあげればツール引数の
構文不正で落ちることはほぼない。

Claudeさまさま。

```
name: "create_midi",
description: "テキスト形式の音楽データからMIDIファイルを生成します。compositionが長くなる場合",
inputSchema: {
  type: "object",
  properties: {
    title: {
      type: "string",
      description: "曲のタイトル",
    },
    composition: {
      type: "object",
      description: "音楽データのオブジェクト",
      properties: {
        bpm: { type: "number" },
        timeSignature: {
          type: "object",
          properties: {
            numerator: { type: "number" },
            denominator: { type: "number" }
          }
        },
      },
    },
    tracks: {
```


Input Schema

ただし、ツール引数が長くなれば
なるほど、LLMのMaxTokenに
ひっつきやすくなる

引っかかった場合は最初から
やり直しになるので注意
(ループを作っちゃう原因に)

Task: 6 4小節のBPM180のチップチューンを作成してください。

```
{
  "pitch": 60, "beat": 6.0, "duration": "16", "velocity": 80},
{"pitch": 64, "beat": 6.25, "duration": "16", "velocity": 80},
{"pitch": 67, "beat": 6.5, "duration": "16", "velocity": 80},
{"pitch": 72, "beat": 6.75, "duration": "16", "velocity": 80},
{"pitch": 60, "beat": 7.0, "duration": "16", "velocity": 80},
{"pitch": 64, "beat": 7.25, "duration": "16", "velocity": 80},
{"pitch": 67, "beat": 7.5, "duration": "16", "velocity": 80},
{"pitch": 72, "beat": 7.75, "duration": "16", "velocity": 80},
{"pitch": 60, "beat": 8.0, "duration": "16", "velocity": 80},
{"pitch": 64, "beat": 8.25, "duration": "16", "velocity": 80},
{"pitch": 67, "beat": 8.5, "duration": "16", "velocity": 80},
{"pitch": 72, "beat": 8.75, "duration": "16", "velocity": 80},

{"pitch": 65, "beat": 9.0, "duration": "16", "velocity": 80},
{"pitch": 69, "beat": 9.25, "duration": "16", "velocity": 80},
{"pitch": 72, "beat": 9.5, "duration": "16", "velocity": 80},
{"pitch": 77, "beat": 9.75, "duration": "16", "velocity": 80},
{"pitch": 65, "beat": 10.0, "duration": "16", "velocity": 80},
{"pitch": 69, "beat": 10.25, "duration": "16", "velocity": 80},
{"pitch": 72, "beat": 10.5, "duration": "16", "velocity": 80},
{"pitch": 77, "beat": 10.75, "duration": "16", "velocity": 80},
{"pitch": 65, "beat": 11.0, "duration": "16", "velocity": 80},
{"pitch": 69, "beat": 11.25, "duration": "16", "velocity": 80},
{"pitch": 72, "beat": 11.5, "duration": "16", "velocity": 80},
{"pitch": 77, "beat": 11.75, "duration": "16", "velocity": 80},
{"pitch": 65, "beat": 12.0, "duration": "16", "velocity": 80},
{"pitch": 69, "beat": 12.25, "duration": "16", "velocity": 80},
{"pitch": 72, "beat": 12.5, "duration": "16", "velocity": 80},
{"pitch": 77, "beat": 12.75, "duration": "16", "velocity": 80},

{"pitch": 65, "beat": 13.0, "duration": "16", "velocity": 80},
{"pitch": 69, "beat": 13.25, "duration": "16", "velocity": 80},
{"pitch": 72, "beat": 13.5, "duration": "16", "velocity": 80},
{"pitch": 77, "beat": 13.75, "duration": "16", "velocity": 80},
{"pitch": 65, "beat": 14.0, "duration": "16", "velocity": 80},
{"pitch": 69, "beat": 14.25, "duration": "16", "velocity": 80},
{"pitch": 72, "beat": 14.5, "duration": "16", "velocity": 80},
{"pitch": 77, "beat": 14.75, "duration": "16", "velocity": 80},
{"pitch": 60, "beat": 15.0, "duration": "16", "velocity": 80},
{"pitch": 64, "beat": 15.25, "duration": "16", "velocity": 80}
```

Auto-approve: Read, Edit, Commands, Browser, MCP >

Cancel

Input Schema

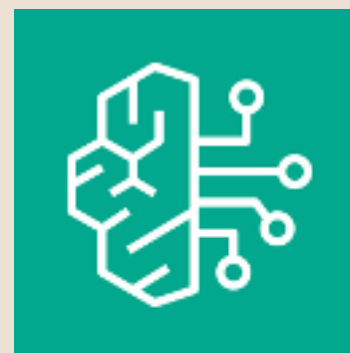
長いInputならClineであれば
一度ファイルに
出力しそちらを参照するように
するとよさそう。

InputSchemaで
「どちらか一方を使ってね」
という指定も可能(oneOf)

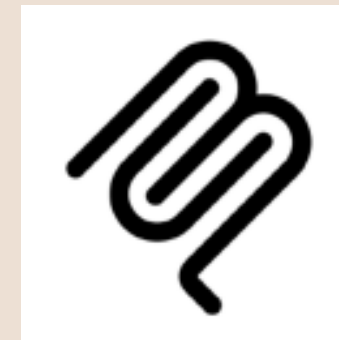
```
    },  
    required: ["bpm", "tracks"]  
  },  
  composition_file: {  
    type: "string",  
    description: "音楽データが含まれるJSONファイルの絶対パス。composition_file",  
  },  
  output_path: {  
    type: "string",  
    description: "出力ファイル絶対パス",  
  },  
  },  
  required: ["title", "output_path"],  
  oneOf: [  
    { required: ["composition"] },  
    { required: ["composition_file"] }  
  ]  
}
```



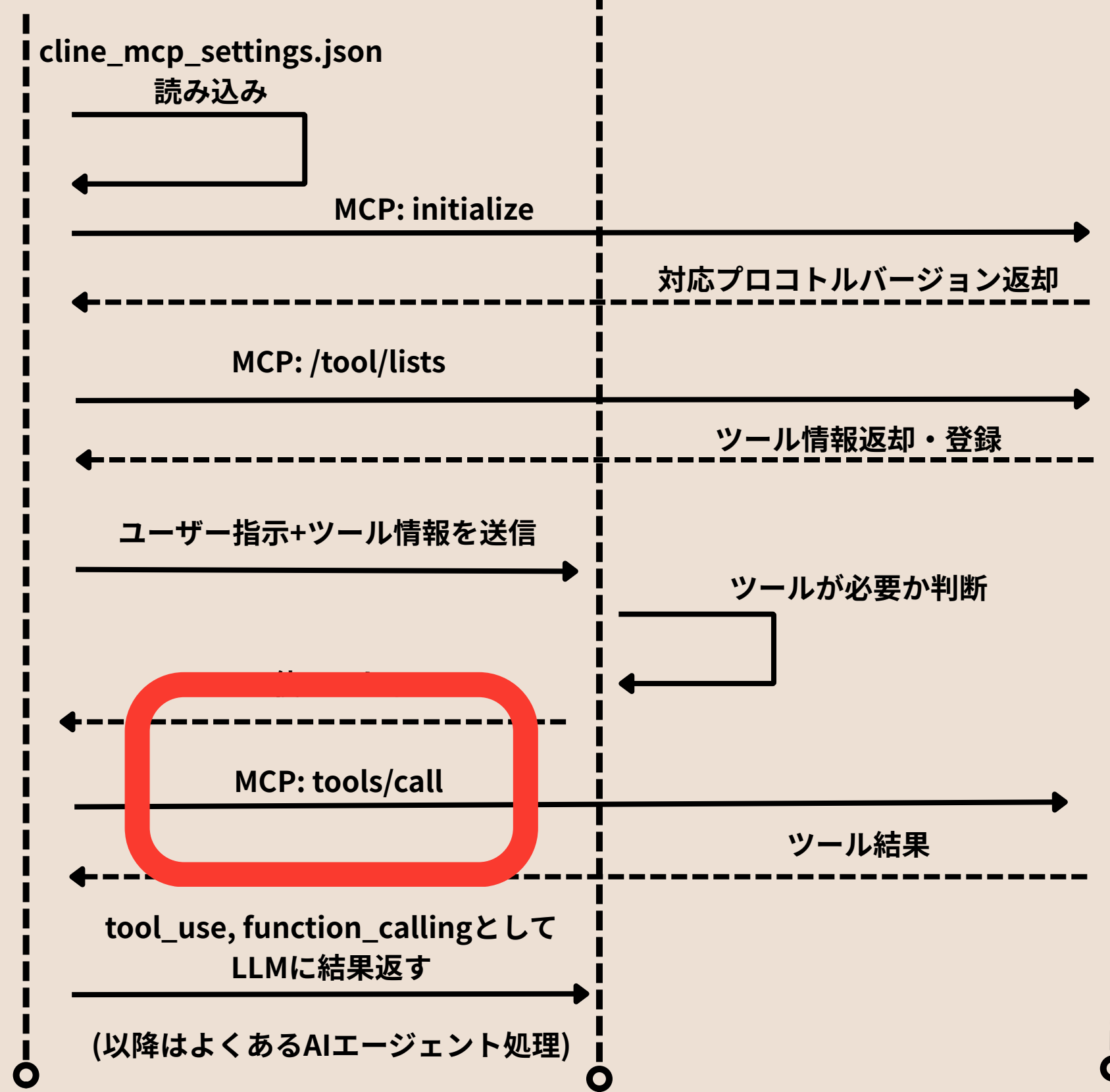
Cline
(MCPクライアント)



Amazon Bedrock
(Claude 3.7 sonnet)



MCP Server
(studio on npm)



Output files

ファイルを出力させるとき、
MCPサーバーには現在動いている
ディレクトリ情報がわからない。
なので、rootディレクトリにファイルを
作ったりしちゃう

一応Feature Requestは上がっている。
<https://github.com/cline/cline/discussions/2635>

```
// 出力パスの処理
let outputFilePath;
if (path.isAbsolute(outputPath)) {
  outputFilePath = outputPath;
} else {
  const safeBaseDir = process.env.HOME || process.env.HOMEPATH;
  const midiDir = path.join(safeBaseDir, 'midi-files');
  if (!fs.existsSync(midiDir)) {
    fs.mkdirSync(midiDir, { recursive: true });
  }
  const fileName = path.basename(outputPath);
  outputFilePath = path.join(midiDir, fileName);
}
```


Output files

なので絶対パスでファイルを
指定させると安定する。

こちらもInputSchemaで指示を
通すことができる。

```
output_path: {  
  type: "string",  
  description: "出力ファイル絶対パス",  
}
```

Conclusion

MCPのLTのために
変なServer作ってしまった。

自作MCP Serverで
おもしろそうなアイデアが
浮かばないので
一緒に考えてほしいです！！



**See you
next LT!**

**THANK YOU
FOR LISTENING!**

