FICO® Xpress Mosel extension for Microsoft VS Code (BETA)

This Microsoft VS Code extension provides a set of features for developers using Xpress Mosel. These features include:

- Mosel language support
- Runtime support for compiling, running, and debugging Mosel code

This documentation assumes that you are familiar with Visual Studio Code, including the installation of extensions.

This is not a general-purpose guide to Mosel development. For more information about Mosel, see the Xpress Mosel documentation.

License

The VS Code FICO® Xpress Mosel extension relies on FICO® Xpress software, which is subject to the Xpress Shrinkwrap License Agreement.

By downloading this extension, you agree to the Community License terms of the Xpress Shrinkwrap License Agreement with respect to the required FICO Xpress software.

See the licensing options overview for additional details and information about obtaining a paid license for FICO Xpress Solver.

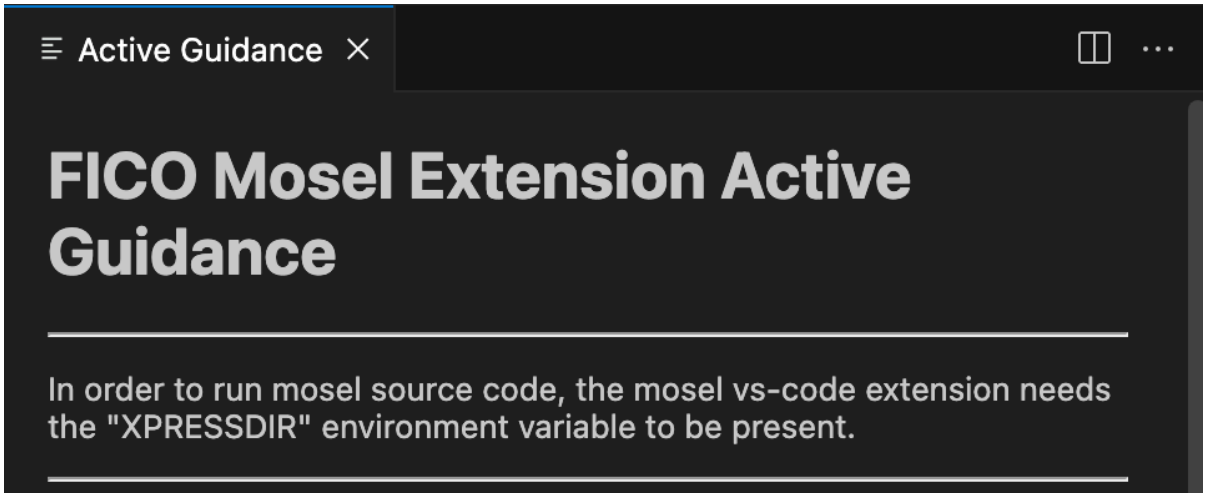
The extension also contains other, separate, distinct software which may be subject to other licenses.

Table of Contents

Setup

The runtime features of this extension (compiling, running, and debugging) require a valid installation of FICO® Xpress. For more information, see the FICO Xpress Installation documentation.

By default, the extension attempts to detect any Xpress installation by checking the value of the `XPRESSDIR` environment variable. If this variable is not available, the runtime features are unavailable. (If you try to use these features, you will see this **Active Guidance** message:



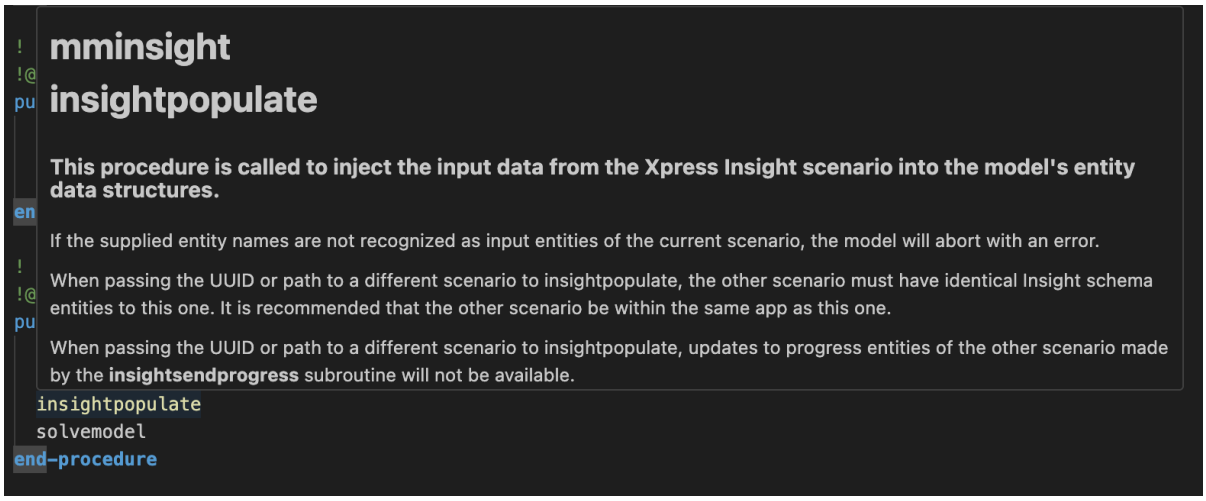
The Active Guidance panel

Features Overview

Language Support Features

Standard Language Support Features

- Syntax highlighting
- Code folding
- Jump to definition
    - Variables, functions, uses, imports, includes
    - Definitions in other files are also supported
- Hover tooltip information
    - Mosel documentation as well as declarations
- Code formatter
- Code completion
- Code diagnostics
    - Includes error feedback from the Mosel parser
- Outline symbols list



The hover tooltip

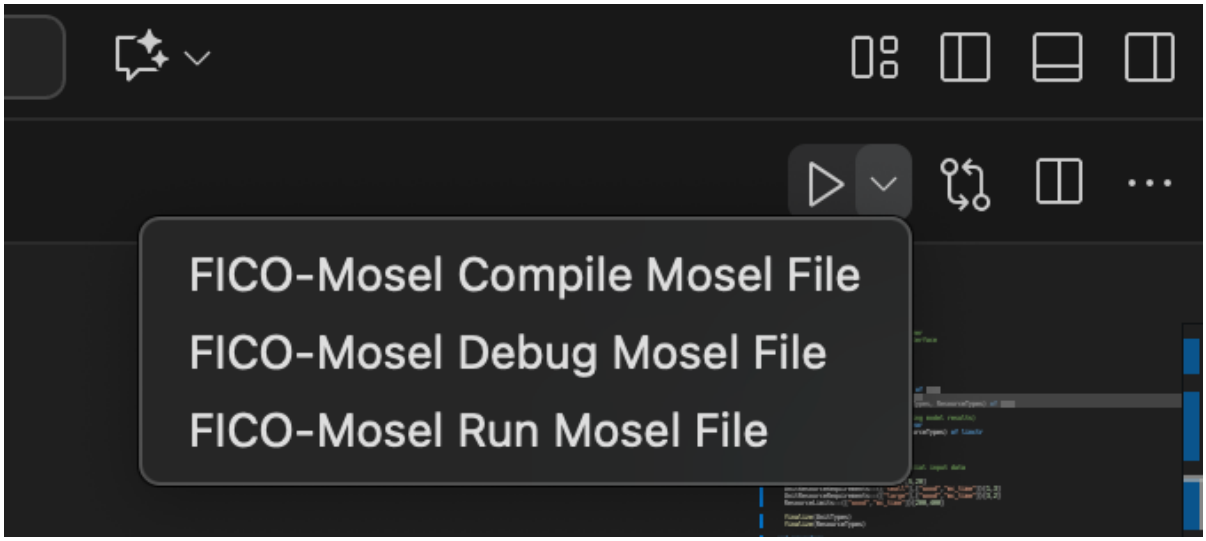
Runtime Features

The Play Button

This button appears on the top-right part of the IDE when a Mosel file (with .mos extension) is open in the currently focused editor.

It provides quick access to the runtime features of the extension:

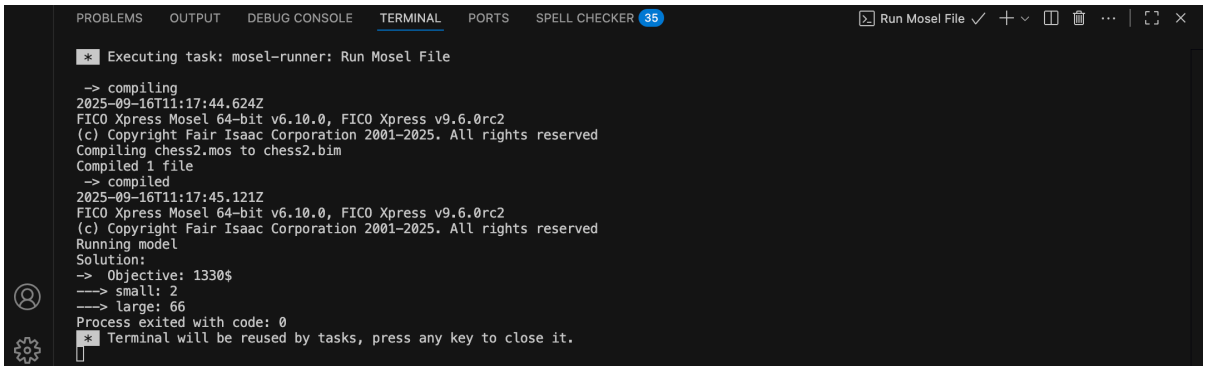
- Compile a Mosel file
  - Generates a `.bim` file.
- Run a Mosel File
  - Compiles the Mosel file and then executes the generated `.bim` file.
- Debug a Mosel File
  - Compiles the Mosel file (as a debug) and then starts a debug session.



The "Play Button" menu, expanded

Run the Currently Selected Mosel File

When you select the option to run a Mosel file, a window opens to show any relevant information and console output. The Mosel file is compiled to a `.bim` file, and then the `.bim` file is executed.



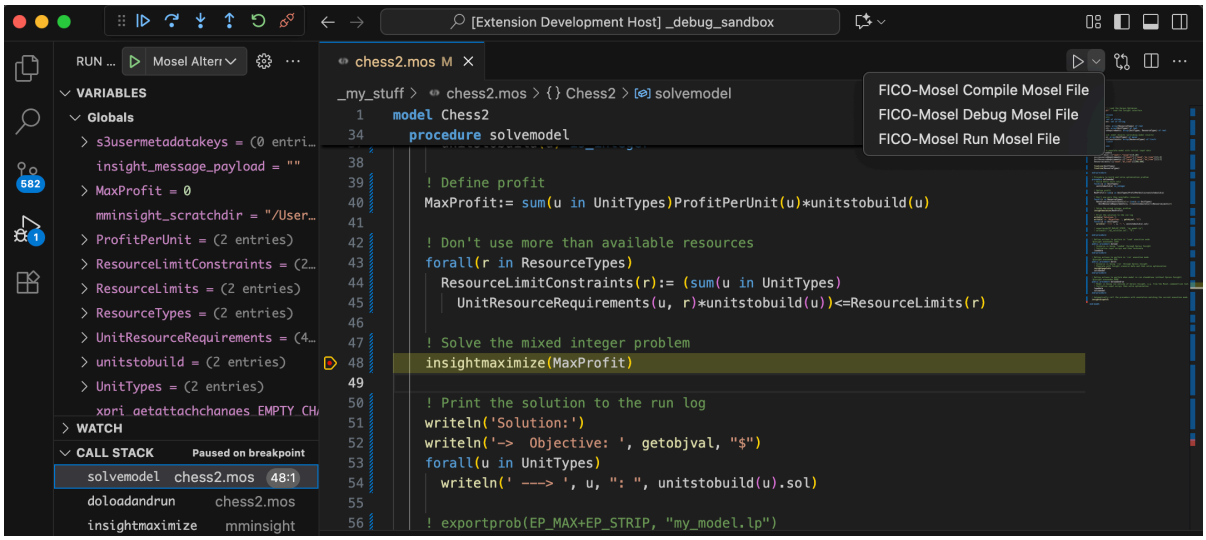
Running a Mosel file

Debug the selected Mosel file

When you select the option to debug a Mosel file, a window opens to show any relevant information and console output. The Mosel file is compiled to a `.bim` file, and then the `.bim` file is debugged.

A debug session starts and is attached to the newly started Mosel process. The debug interface becomes active:

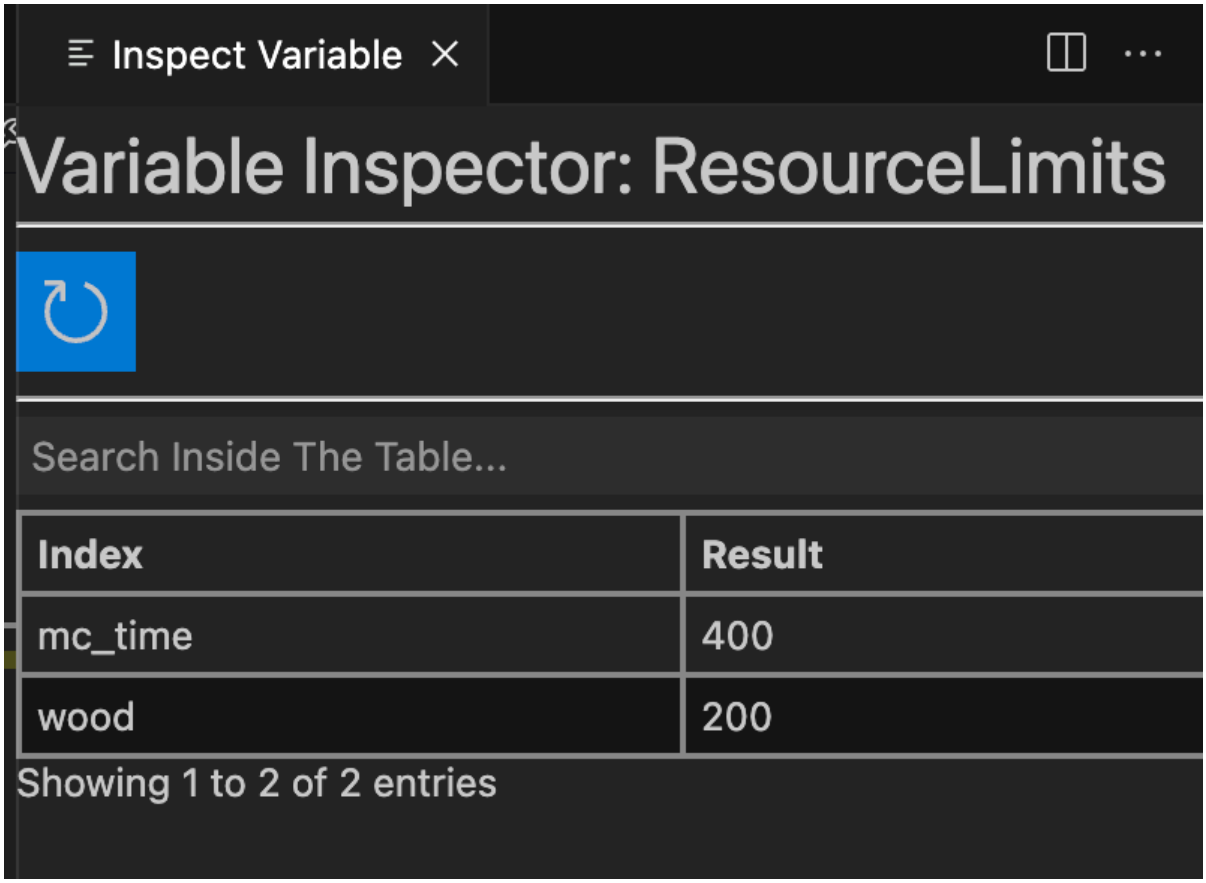
- The sidebar, listing debug information such as variables and the call stack
- The debug console, showing console output
- The debug toolbar, with options such as continue and step over



Debugging a Mosel file

Inspecting Collection Variables During a Debug Session

To inspect a variable value, type `show \[expression\]` in the debug console. The **Inspect Variable** window opens, showing details of the variable value. This window can be manually refreshed when stepping over instructions.



The variable inspector panel while debugging a Mosel file

Advanced Options

The default options are intended to cover the most common use cases, but you can customize the extension for your needs. These customizations are mostly configured using the various JSON files that can be placed in the `.vscode` folder at the root of your project.

Note that these files are parsed by Visual Studio Code using the JSON5 format, which (unlike standard JSON) allows the use of comments.

Settings Examples

Source Code Indexer Settings

```
// .vscode/settings.json
{
  "fico-xpress-mosel-language-support-settings.indexer-resolved-folders":
[

    // will accept relative paths
    // -> relative to the current workspace
    "./my-third-party-source-code-folder/",

    // will also accept abolute paths
    // "/Users/my-machine-username/dev/my-third-party-source-code-folder/"
  ]
}
```

Source Code Formatter Settings

```
// .vscode/settings.json
{
  "fico-xpress-mosel-language-support-settings.code-format-options": {

    // will override the default per-document value if present
    "insertSpaces": true,

    // will override the default per-document value if present
    "tabSize": 2
  }
}
```

Task Configurations

How to run a custom task

1. Open the command palette:
    - Windows: **Ctrl + Shift + P**
    - Linux: **Ctrl + Shift + P**
    - MacOS: **Command + Shift + P**
2. Select **Tasks : Run Task**. (You can also start typing `run task` and then select from the autocomplete suggestions.)
3. Select your task (for example, `my-mosel-runner-task: Debug Test Mosel File`). If you do not see your task, start typing the task name and then select from the autocomplete suggestions.
4. For any additional prompts, accept the defaults by pressing **Enter**.

Task Example: The `task.json` File

```
// .vscode/tasks.json
{
  "version": "2.0.0",
  "tasks": [

    {
      "label": "some-task-label",
      "type": "some-task-type",
      ...
    },

    {
      "label": "some-other-task-label",
      "type": "some-other-task-type",
      ...
    },

    ...
  ]
}
```

Task Example: Debugging a Mosel File

This example defines a task for debugging a Mosel file with custom options.

```
{
  // [REQUIRED] task public label, must be unique per task
  "label": "fico-xpress-mosel-debug-file: Debug Test Mosel File",
  // [REQUIRED] task type, constant
  "type": "fico-xpress-mosel-debug-file",
  // [REQUIRED] task name, constant
  "task": "debug-file",
  // [REQUIRED] the path to the file to run
  "inputFile": "my-source-folder/test.mos",

  // optional fields:

  // [OPTIONAL] will default to the same folder of the Mosel source file
  "outDir": "my-source-folder",

  // [OPTIONAL] values for a Mosel model's "parameters" block
  "parameters": {
    // [OPTIONAL] example of numerical/boolean/string values
    "NBRofWORKER": 8,
    "ONE_BT_PER_WRKR": false,
    "OUTPUT_SUBFOLDER": "output",
  },

  // [OPTIONAL] will default to false (for the "Debug Mosel File" only)
  "stopOnEntry": true,

  // [OPTIONAL] will default to the XPRESSDIR environment variable value
  "xpressmpPath": "my-xpressmp-folder",
}
```

Task Example: Running a Mosel File

This example defines a task for running a Mosel file with custom options.

```
{
  // [REQUIRED] task public label, must be unique per task
  "label": "fico-xpress-mosel-run-file: Run Test Mosel File",
  // [REQUIRED] task type, constant
  "type": "fico-xpress-mosel-run-file",
  // [REQUIRED] task name, constant
  "task": "run-file",
  // [REQUIRED] the path to the file to run
  "inputFile": "my-source-folder/test.mos",

  // optional fields:

  // [OPTIONAL] will default to the same folder of the Mosel source file
  "outDir": "my-source-folder",

  // [OPTIONAL] values for a Mosel model's "parameters" block
  "parameters": {
    // [OPTIONAL] example of numerical/boolean/string values
    "NBRofWORKER": 8,
    "ONE_BT_PER_WRKR": false,
    "OUTPUT_SUBFOLDER": "output",
  },

  // [OPTIONAL] will default to the XPRESSDIR environment variable value
  "xpressmpPath": "my-xpressmp-folder",
},
```

Task Example: Compiling a Mosel File

This example defines a task for compiling a Mosel file with custom options.

```
// .vscode/tasks.json
{
  // [REQUIRED] task public label, must be unique per task
  "label": "my-mosel-runner-task: Build Test Mosel File",
  // [REQUIRED] task type, constant
  "type": "fico-xpress-mosel-compile-file",
  // [REQUIRED] task name, constant
  "task": "compile-file",
  // [REQUIRED] the path to the file to compile
  "inputFile": "my-source-folder/test.mos",

  // optional fields:

  // [OPTIONAL] will default to the same folder of the Mosel source file
  "outDir": "my-source-folder",

  // [OPTIONAL] will default to the XPRESSDIR environment variable value
  "xpressmpPath": "my-xpressmp-folder",
}
```

Copyright Information