FICO Xpress Solver VS Code Extension (BETA)

This VS Code FICO Xpress Solver extension supports use of FICO Xpress Solver for solving and tuning matrix files.

The Xpress Optimizer employs many diverse techniques when solving a given problem. Which technique to apply and with how much effort to put into it is highly configurable.

This behavior can be modified — or tuned — by setting some control parameters. The full list of these control parameters is available in the Xpress Optimizer Reference Manual.

For details visit the Optimizer Tuning Guide.

This guide assumes some basic familiarity with Xpress Solver.

License

The VS Code FICO® Xpress Solver 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.

Setup

This extension requires a valid installation of the FICO Xpress Solver.

If missing, here's how to install it: Xpress Installation Documentation.

What it does

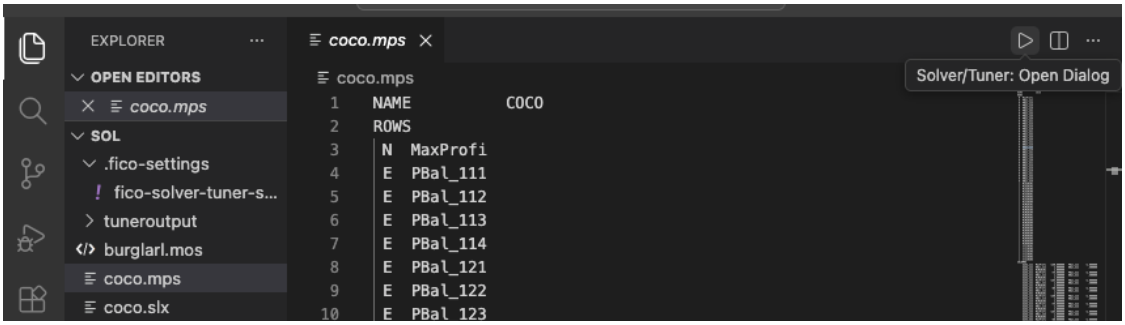
- A dialog drives Solve/Tune execution with options to configure the solver options.
- Solver produces solution files (.slx) and the tuner also puts the tuning results files (.xtr)
- The active tuner **.xtr** can be visualised as a table.

Ways it gets triggered

Editor title buttons

Buttons appear in the editor title (top-right) when conditions match:

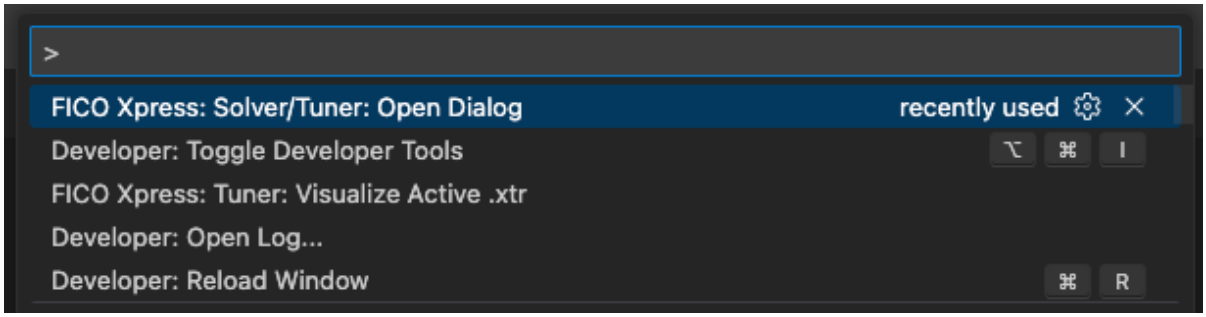
- **Open Dialog button**
  - Appears when you open a matrix file: `.mps` or `.lp`



Matrix play

Command Palette

Both commands show up in the Command Palette:



Command Palette

- **Title:** `Solver/Tuner: Open Dialog`
- **Category:** `FICO Xpress`

Visualise the active tuner .xtr

- **Title:** `Tuner: Visualize Active .xtr`
- **Category:** `FICO Xpress`

Status bar (Play button)

A status bar item is available after the extension loads:

- **`$(play) FICO Xpress Solve/Tune`** → opens the dialog

This is the quickest way to start.

Using the dialog

The dialog is the main place to configure and run **Solve** and **Tune**.

How it behaves:

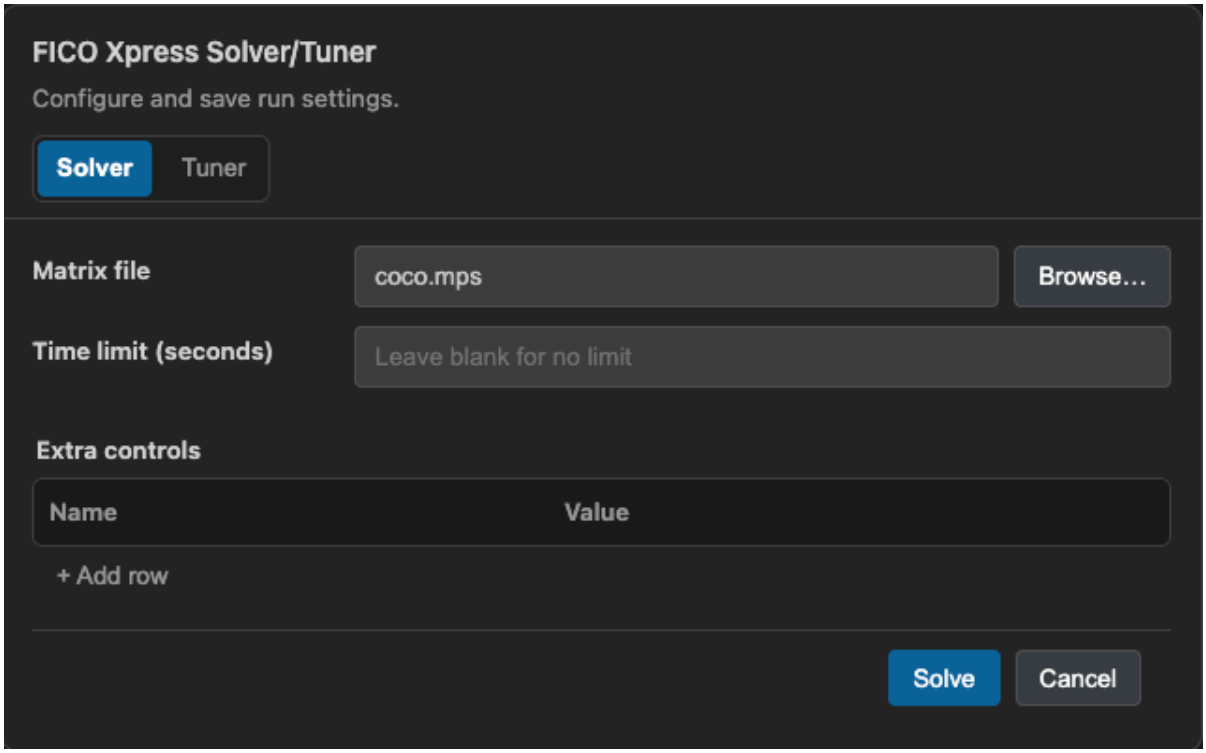
- When you click **Solve** or **Tune**, the dialog saves your settings and starts the run.
- **Browse...** selects files relative to your workspace folder.

Tip:

- If you open the dialog while a `.mps` or `.lp` file is active in the editor, it may pre-fill the matrix path from that file.

Solver mode (Solve)

Use **Solver** when you want to solve a single model and (when feasible) generate a solution file.



Solve dialog

Matrix file

- Required.
- Supported inputs: `.mps` and `.lp`.

Time limit (seconds)

- Optional.
- Leave blank to use the default behaviour.
- Use a positive number of seconds.

Extra controls

- Optional.
- Lets you set Optimizer control parameters as name/value pairs for the run.
- Example use cases:
  - overriding solver strategy controls
  - setting numeric or string controls not exposed as dedicated fields

Notes:

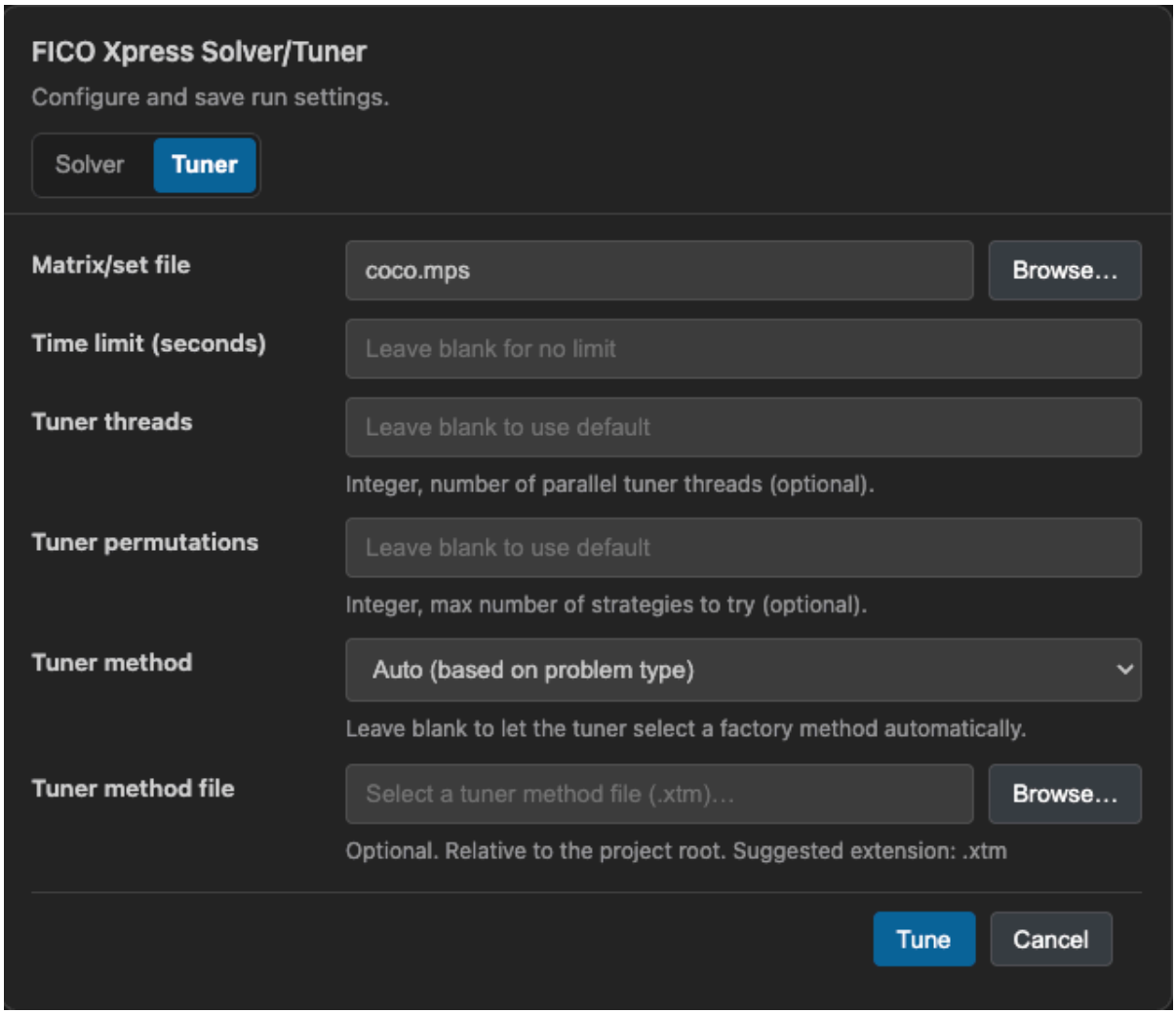
- Each row must have both a name and a value.
- Unknown control names are treated as errors (the run will fail rather than ignoring them).

Output from Solve

- The run output is printed in the VS Code terminal.
- A solution file may be written as `<matrixName>.slx` alongside the matrix file (depending on solve status and model).

Tuner mode (Tune)

Use **Tuner** when you want the optimizer to try different configurations and produce a tuner results file (`.xtr`) that you can compare in a table view.



Tune dialog

Matrix/set file

- Required.
- You can provide either:
  - a single matrix file (`.mps` / `.lp`) to tune directly, **or**
  - a set file (`.set`) when you want tuning to run over a predefined set.

Tuning against multiple matrices

If you want to tune using more than one matrix, use a `.set` file and list the matrices inside that set in the order you want them processed. This is the standard approach for multi-matrix tuning.

A `.set` file is plain text: **one matrix filename per line** (relative to the workspace/project root), for example:

```
coco.mps
Folio.mps
```

Time limit (seconds)

- Optional.
- Leave blank to use default behaviour.

Tuner threads (optional)

- Integer.
- Limits how many tuner threads are used in parallel.

Tuner permutations (optional)

- Integer.
- Caps how many strategies/configurations the tuner will try.

Tuner method (optional)

- Choose a predefined method, or leave it as **Auto**.

Tuner method file (`.xtm`) (optional)

- Use this when you have a custom tuner method file.

Where to see output (terminal)

Solve/Tune runs are shown in a VS Code terminal (live output).

You will typically see:

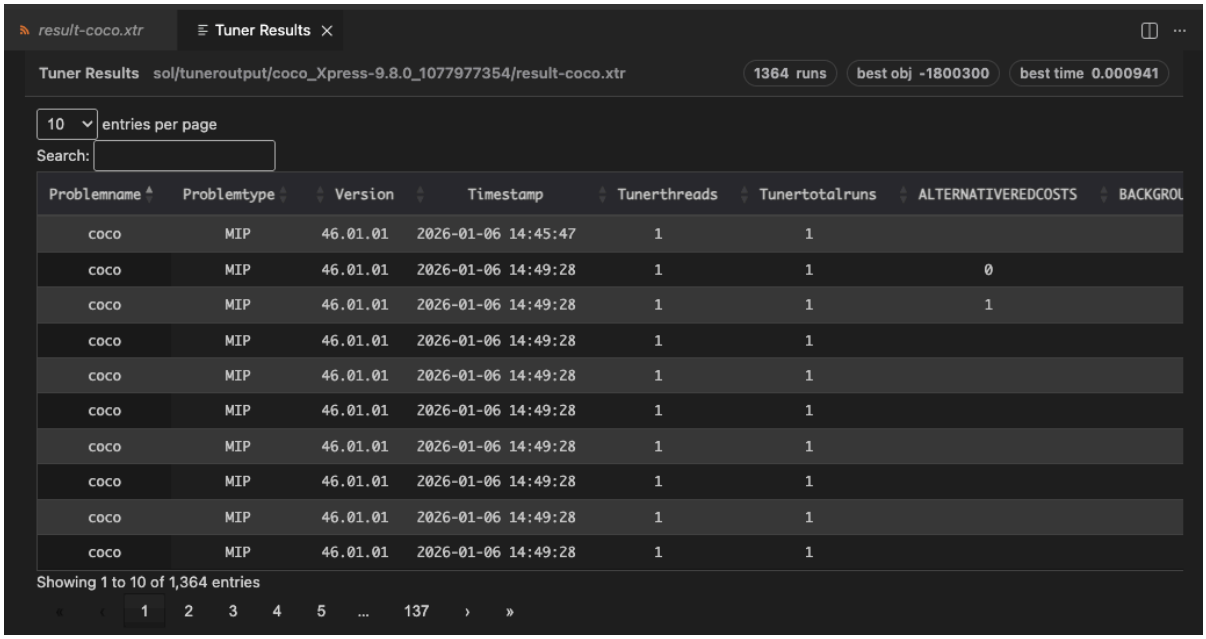
- a “running SOLVE/TUNE …” line at the start
- streaming output from the run
- an exit code at the end

If something fails, the terminal output is the first place to look.

Results (`.xtr`) and visualisation

Auto-open after Tune

After a successful **Tune**, the extension automatically looks for the latest tuner results file (`.xtr`) under `tuneroutput/` and opens it in the results viewer.

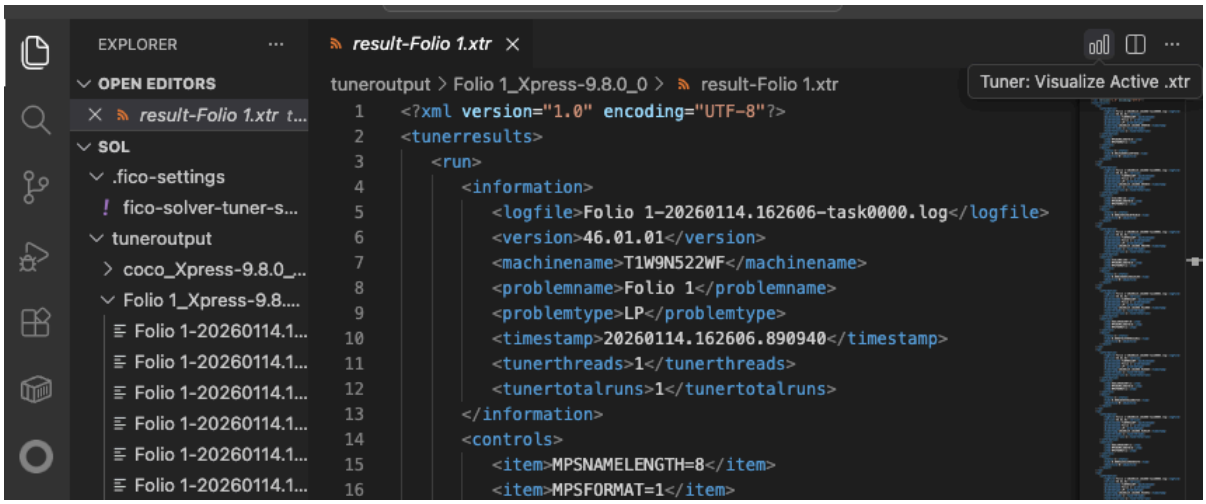


Visualize .xtr

Manual visualisation

You can open the results viewer yourself:

- Use the editor title button (when available)
- Or run `FICO Xpress: Tuner: Visualize Active .xtr`



Visualize .xtr trigger

Important:

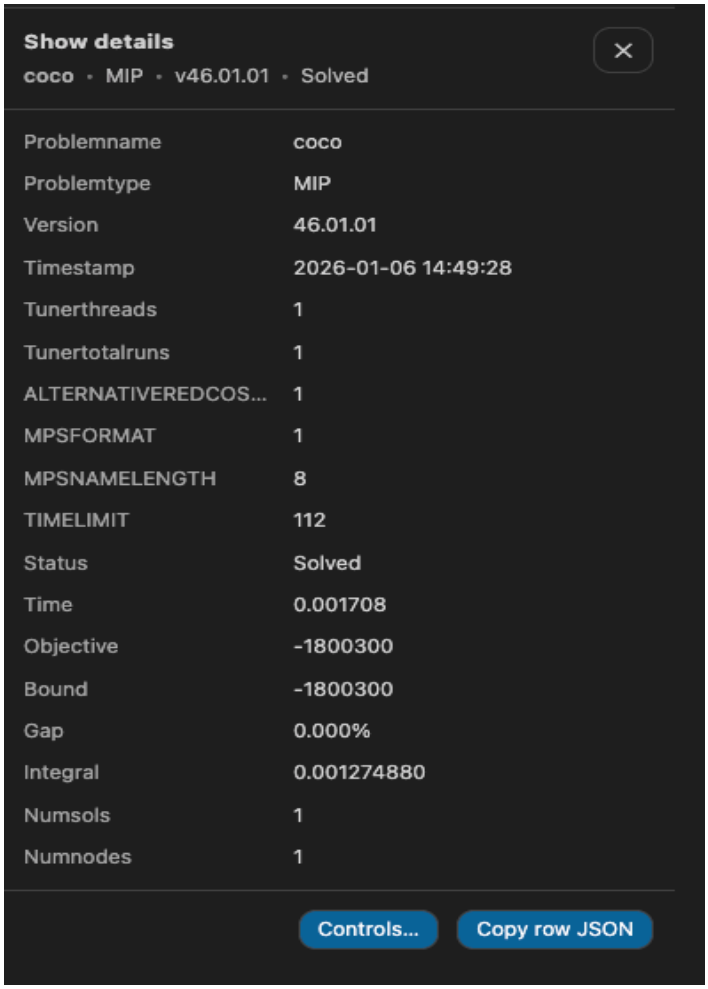
- Manual “Visualize Active .xtr” works when the `.xtr` you want to view is the **currently active editor tab** and is located under `tuneroutput/`.

Using the Results viewer

The results viewer is a table designed for quickly comparing tuning runs.

You can:

- sort and search across runs
- click a row to open **Show details**



Show details

- click **Controls...** to view the controls used for that run and generate snippets in:

Controls code

- Mosel
- Python
- Solver-style `key=value`
- copy:
  - the row JSON
  - the generated controls code

Advanced

Command ids

Most users will only need the command titles shown in the Command Palette. If you need the underlying ids (for keybindings, automation, or internal documentation):

- **Title:** `Solver/Tuner: Open Dialog`
  Command id: `ficoXpressSolverTuner.openDialog`
- **Title:** `Tuner: Visualize Active .xtr`
  Command id: `ficoXpressSolverTuner.visualizeActiveXtr`

Tasks (optional, for repeatable runs)

If you prefer running Solve/Tune through tasks (for repeatable workflows), the extension provides a task type `ficoXpressSolverTuner` with actions:

- `dialog`
- `runSolve`
- `runTune`

Ready-to-use `.vscode/tasks.json`

```
{
  "version": "2.0.0",
  "tasks": [
    {
      "label": "FICO Xpress: Solver/Tuner (Dialog)",
      "type": "ficoXpressSolverTuner",
      "action": "dialog",
      "problemMatcher": []
    },
    {
      "label": "FICO Xpress: Solver (Run Solve)",
      "type": "ficoXpressSolverTuner",
      "action": "runSolve",
      "problemMatcher": []
    },
    {
      "label": "FICO Xpress: Tuner (Run Tune)",
      "type": "ficoXpressSolverTuner",
      "action": "runTune",
      "problemMatcher": []
    }
  ]
}
```

Limitations and expectations

- A working Xpress installation is required (including licensing and environment configuration).
- Very large `.xtr` files can take longer to render inside VS Code.
- Manual visualisation only works when viewing an `.xtr` under `tuneroutput/`.
- Unknown control names in **Extra controls** cause the run to fail (they are not ignored).

Support

Please use this link: Ask-a-question

Copyright Information