POWER BI · AUTHORING MCP

A *prompt* library for semantic models

Six prompt patterns that work with the Power BI Authoring MCP Server, eight ready-to-use prompts, and six rules for using it without breaking your model.

Six prompt patterns that work

01 Narrow the scope.
Instead of "fix the model", say "fix the format strings of the measures in the Sales table".

02 Ask for a plan first.
Make "Propose a plan and wait for my approval" a habit.

03 Ask for a before/after table.
Never approve a bulk rename without seeing old and new names.

04 Ask for tests.
After every refactor, have the agent write and run a DAX query that validates the result.

05 Spell out your rules.
Put your naming standard, description length and format string preferences in the prompt.

06 Wrap bulk work in a transaction.
On the local server, use the "Begin a transaction ... commit" pattern.

Prompt library

Replace table, measure and lakehouse names with your own.

Naming standard

```
Analyze my model's naming conventions and suggest renames to ensure consistency.
```

Document the model

```
Add descriptions to all measures, columns, and tables that explain their purpose and the logic behind the DAX in plain business terms.
```

Markdown documentation

```
Generate a Markdown document (.md) that provides complete, professional documentation of this semantic model.
```

Translation

```
Generate a French translation for my model, including tables, columns, and measures.
```

Calculation group

```
Refactor measures 'Sales Amount 12M Avg' and 'Sales Amount 6M Avg' into a calculation group, and add 24M and 3M variants.
```

Power Query parameters

```
Analyze the Power Query code for all tables and create semantic model parameters for hard-coded server and database names.
```

Compare two models

```
Connect to semantic model 'V1' and 'V2' and benchmark the following DAX query against both models.
```

Direct Lake model

```
Create a Direct Lake semantic model against lakehouse 'LH_CorpData' using tables 'Product', 'Sales', 'Store'.
```

Six rules for using it without breaking your model

- **Back up first.** Take a backup of the model before every session.
- **Work in PBIP + Git.** Read every change the agent makes in the TMDL diff, and revert when needed.
- **Try it outside production.** Don't point it at a live model; use a development workspace.
- **Apply least privilege.** If you only need to read, start the local server with --readonly.
- **Keep confirmations on.** The local server asks before writing. Use --skipconfirmation only when you have a backup.
- **Know where your data goes.** Metadata, schemas and query results are sent to your LLM provider.

> The agent works with your permissions. Treat every change it proposes the way you would treat a pull request.