Power BI MCP Server

Complete Tools Reference Guide

Version 6.4.1

29 Powerful Tools

12 Categories | AI-Powered Analysis

Analyze, Modify, and Document Power BI Models
Through an AI-Powered Interface

Generated on November 25, 2025

Welcome to the Power BI MCP Server Tools Guide!

This comprehensive guide explains all **29 tools** available in the MCP-PowerBi-Finvision server. These tools allow you to analyze, modify, and document Power BI models through an AI-powered interface.

What is an MCP Server?

MCP (Model Context Protocol) is a way for AI assistants to access specialized tools. This server provides tools specifically designed for working with Power BI Desktop models, enabling seamless integration between AI capabilities and your data modeling workflow.

How to Read This Guide

- Each tool has a clear description of what it does
- "When to Use" explains the situations where you'd use this tool
- Examples show real-world usage scenarios
- Parameters list what information you need to provide
- Results explain what you get back

Getting Started:
1. Always start by connecting to a Power BI instance (Category 01)
2. Use the analysis tools (Category 05) to understand the model
3. Then use specific tools based on what you need to do

Table of Contents

Quick Reference Guide

Most Commonly Used Tools

01 - Connection (Always First!)

- `detect_powerbi_desktop` - Find open Power BI files
- `connect_to_powerbi` - Connect to one

02 - Schema, Search & Operations

- `search_objects` - Find tables, columns, measures
- `column_operations` - List/modify columns
- `measure_operations` - List/modify measures
- `relationship_operations` - View/modify relationships
- `tmdl_operations` - Export/refactor model
- `batch_operations` - Batch changes (3-5x faster)

03 - DAX Intelligence

- `run_dax` - Execute DAX queries
- `dax_intelligence` - Analyze and optimize DAX

05 - Model Analysis

- `simple_analysis` - Fast 2-5 second overview
- `full_analysis` - Complete health check (120+ rules)

06-07 - Export & Documentation

- `get_live_model_schema` - Export model schema
- `generate_model_documentation_word` - Professional Word doc

10 - Hybrid Analysis (Most Powerful!)

- `export_hybrid_analysis` - Export complete package
- `analyze_hybrid_model` - AI BI Expert analysis

01 - Connection

Detect Power BI Instances

(detect_powerbi_desktop)

What it does: Automatically finds all running Power BI Desktop instances on your computer. This is your starting point for any analysis session.

When to use this tool:

Use this when you want to see which Power BI files are currently open before connecting to one.

Example:

```
Before starting analysis, run this to see: 'Found 2 instances: Sales Report (modified
10 mins ago), Customer Analysis (modified 2 hours ago)'
```

Parameters:

- None required - just run it!

What you get back:

List of all open Power BI Desktop files with their names and when they were last modified.

Connect to Power BI Instance

(connect_to_powerbi)

What it does: Connects to a specific Power BI Desktop file that's currently open on your computer. This establishes the connection needed for all other tools.

When to use this tool:

Always use this first before using any other tools. It's like opening the file to work with it.

Example:

```
After detecting instances, connect to the first one (index 0) for the Sales Report, or
index 1 for Customer Analysis.
```

Parameters:

- model_index (optional, default=0): Which Power BI file to connect to (0 for first, 1 for second, etc.)

What you get back:

Success message confirming connection to the Power BI model.

02 - Schema & Search

Search Objects

(search_objects)

What it does: Searches across all tables, columns, and measures to find objects matching your search term.

When to use this tool:

Use when you're looking for something but don't know which table it's in.

Example:

```
Search for 'Revenue' -> Finds: 'Total Revenue' measure, 'Revenue' column in FactSales,
'RevenueGoal' column.
```

Parameters:

- search_term (required): What to search for
- types (optional): Filter by 'tables', 'columns', 'measures'

What you get back:

All matching tables, columns, and measures with their locations.

Search in Measure Expressions

(search_string)

What it does: Searches inside DAX formulas to find measures that use specific functions or references.

When to use this tool:

Use to find all measures that use a specific calculation method or reference another measure.

Example:

```
Search for 'CALCULATE' -> Finds all measures using the CALCULATE function in their DAX
code.
```

Parameters:

- search_term (required): Text to search for
- search_in (optional): 'name', 'expression', or 'both'

What you get back:

List of measures where the search term appears, with the relevant DAX code.

Column Operations

(column_operations)

What it does: Unified interface for all column operations: list, get details, statistics, distribution, create, update, delete columns.

When to use this tool:

Use for any column-related task - viewing, analyzing, or modifying columns.

Example:

```
operation='list' table='Sales' -> Lists all columns. operation='statistics'
table='Sales' column='Amount' -> Gets stats.
```

Parameters:

- operation (required): 'list', 'get', 'statistics', 'distribution', 'create', 'update', 'delete'
- table (required for most): Table name
- column (required for single-column ops): Column name

What you get back:

Depends on operation: column list, details, statistics, or operation result.

Measure Operations

(measure_operations)

What it does: Unified interface for all measure operations: list, get details, create, update, delete, rename measures.

When to use this tool:

Use for any measure-related task - viewing DAX code, creating new measures, or modifying existing ones.

Example:

```
operation='list' -> Lists all measures. operation='get' table='Sales' measure='Total
Revenue' -> Gets DAX code.
```

Parameters:

- operation (required): 'list', 'get', 'create', 'update', 'delete', 'rename'
- table (required for most): Table name
- measure (required for single-measure ops): Measure name
- expression (required for create/update): DAX formula

What you get back:

Depends on operation: measure list, DAX details, or operation result.

Relationship Operations

(relationship_operations)

What it does: Unified interface for all relationship operations: list, get, find, create, update, delete relationships between tables.

When to use this tool:

Use to understand or modify how tables are connected in the data model.

Example:

```
operation='list' -> Shows all relationships. operation='find' table='Sales' -> Finds relationships involving Sales table.
```

Parameters:

- operation (required): 'list', 'get', 'find', 'create', 'update', 'delete'
- from_table, from_column, to_table, to_column (for specific relationships)

What you get back:

Relationship details with cardinality, cross-filter direction, and active status.

TMDL Operations

(tmdl_operations)

What it does: Unified TMDL operations: export, find & replace, bulk rename, generate scripts. TMDL is the source code format for Power BI models.

When to use this tool:

Use for version control, batch updates, or refactoring model objects.

Example:

```
operation='export' output_dir='C:\TMDL' -> Exports model. operation='find_replace' pattern='OldName' replacement='NewName'
```

Parameters:

- operation (required): 'export', 'find_replace', 'bulk_rename', 'generate_script'
- output_dir, pattern, replacement, renames (depending on operation)
- dry_run (optional): Preview without applying

What you get back:

Operation results: exported files, matches found, or changes made.

Calculation Group Operations

(calculation_group_operations)

What it does: Manage calculation groups (advanced DAX feature for reusable patterns like time intelligence).

When to use this tool:

Use to create, list, or delete calculation groups.

Example:

```
operation='list' -> Shows all calculation groups with items. operation='create'
name='Time Intel' items=[...]
```

Parameters:

- operation (required): 'list', 'list_items', 'create', 'delete'
- name (for create/delete): Group name
- items (for create): Array of calculation items

What you get back:

Calculation group details or operation result.

Role Operations

(role_operations)

What it does: View Row-Level Security (RLS) and Object-Level Security (OLS) roles.

When to use this tool:

Use to see what security roles exist and their filter expressions.

Example:

```
operation='list' -> Shows: 'Regional Manager' role with DAX filters on Country table
```

Parameters:

- operation (required): 'list'
- role (optional): Specific role name

What you get back:

List of security roles with their table filters and permissions.

Batch Operations

(batch_operations)

What it does: Execute multiple operations in a single call - 3-5x faster than individual operations. Supports transaction rollback.

When to use this tool:

Use when creating/updating many objects at once for better performance.

Example:

```
Create 10 measures in one call, or update multiple columns at once.
```

Parameters:

- operations (required): Array of operation definitions
- use_transaction (optional): Enable rollback on failure

What you get back:

Results for each operation with success/failure status.

Manage Transactions

(manage_transactions)

What it does: ACID transaction management for model changes - begin, commit, or rollback groups of changes.

When to use this tool:

Use for complex changes where you want all-or-nothing behavior.

Example:

```
Begin transaction -> Make changes -> Commit (or Rollback if something fails)
```

Parameters:

- action (required): 'begin', 'commit', 'rollback', 'status'

What you get back:

Transaction status and ID.

03 - DAX Intelligence

Execute DAX Query

(run_dax)

What it does: Runs a DAX query against the model and returns the results, just like the DAX query window in Power BI.

When to use this tool:

Use to test calculations, preview data, or run analysis queries.

Example:

```
Run: EVALUATE SUMMARIZE(FactSales, DimDate[Year], "Total",
SUM(FactSales[SalesAmount]))
```

Parameters:

- query (required): DAX query (EVALUATE statement)
- top_n (optional): Limit results

What you get back:

Query results as a table with columns and rows.

Comprehensive DAX Intelligence

(dax_intelligence)

What it does: THE MAIN DAX TOOL - Analyzes DAX measures with validation, debugging, optimization, and recommendations. Includes smart measure finder with fuzzy matching!

When to use this tool:

Use whenever you want to understand, debug, or optimize a DAX measure. This is your go-to DAX analysis tool.

Example:

```
Analyze 'Total Revenue' -> Gets: syntax validation, context transitions, performance
metrics, optimization suggestions.
```

Parameters:

- expression (required): Either DAX code OR just measure name (auto-fetches!)
- analysis_mode (optional): 'all' (default), 'analyze', 'debug', or 'report'

What you get back:

Complete DAX analysis with 11 anti-pattern checks, context flow, VertiPaq metrics, and rewritten DAX suggestions.

Analyze Measure Dependencies

(analyze_measure_dependencies)

What it does: Shows the complete dependency tree for a measure - what it depends on and what depends on it.

When to use this tool:

Use to understand measure relationships before making changes, or to trace calculation logic.

Example:

```
Analyze 'Profit Margin' -> Shows: Depends on 'Total Profit' and 'Total Revenue', with full tree.
```

Parameters:

- table (required): Table name
- measure (required): Measure name

What you get back:

Dependency tree showing all referenced measures, columns, and tables in a hierarchical structure.

Get Measure Impact Analysis

(get_measure_impact)

What it does: Shows what would be affected if you change or delete this measure (impact analysis).

When to use this tool:

Use before modifying or deleting a measure to see what else might break.

Example:

```
Check impact of 'Total Sales' -> Shows: Used by 15 other measures. High impact!
```

Parameters:

- table (required): Table name
- measure (required): Measure name

What you get back:

List of all measures, reports, and visuals that reference this measure.

04 - Data Sources

Get Data Sources

(get_data_sources)

What it does: Lists all data sources connected to the model (SQL databases, Excel files, web sources, etc.).

When to use this tool:

Use to understand where the data is coming from.

Example:

```
Returns: 'SQL Server: MyServer\DB, Excel: C:\Data\Budget.xlsx'
```

Parameters:

- None required

What you get back:

List of data sources with connection details.

Get Power Query M Expressions

(get_m_expressions)

What it does: Shows the M (Power Query) code that loads and transforms data for each table.

When to use this tool:

Use to see how data is being loaded and transformed in Power Query.

Example:

```
Get M for 'FactSales' -> Shows: Source = Sql.Database("server", "db"), TransformedData
= ...
```

Parameters:

- table (optional): Filter by specific table

What you get back:

Power Query M code for data loading and transformation.

05 - Model Analysis

Fast Model Analysis (Simple)

(simple_analysis)

What it does: Quick analysis of the model - runs 8 Microsoft MCP operations to get a complete overview in seconds. Perfect for getting started!

When to use this tool:

Use this as your first analysis step to quickly understand the model structure and statistics.

Example:

```
Run simple analysis -> Gets: Database info, model stats (15 tables, 245 measures), all tables, measures, relationships.
```

Parameters:

- mode (optional): 'all' (default), 'tables', 'stats', 'measures', 'columns', 'relationships', 'roles', 'database', 'calculation_groups'

What you get back:

Comprehensive model overview with expert insights. Execution time: ~2-5 seconds.

Comprehensive Model Analysis (Full)

(full_analysis)

What it does: Deep analysis with Best Practice Analyzer (120+ rules), performance analysis, and data integrity checks.

When to use this tool:

Use for thorough model health check - identifies issues, optimization opportunities, and best practice violations.

Example:

```
Run full analysis -> Finds: 15 best practice violations, 3 performance issues, 2 integrity issues.
```

Parameters:

- scope (optional): 'all', 'measures', 'model', 'performance'
- depth (optional): 'quick', 'balanced', 'thorough'
- max_seconds (optional): Time limit

What you get back:

Detailed analysis with categorized issues and recommendations. Execution time: 10-180 seconds.

06 - Export & Schema

Get Live Model Schema

(get_live_model_schema)

What it does: Exports a lightweight JSON schema of the model structure - optimized for AI analysis with low token usage.

When to use this tool:

Use to get a quick snapshot of the model structure for documentation or analysis.

Example:

```
Export schema -> Returns JSON with: 15 tables, columns, measures, relationships.
```

Parameters:

- None required

What you get back:

Compact JSON schema without DAX expressions, perfect for quick model understanding.

07 - Documentation

Generate Model Documentation (Word)

(generate_model_documentation_word)

What it does: Creates a comprehensive Word document with complete model documentation: tables, columns, measures, relationships, data sources.

When to use this tool:

Use to create professional documentation for sharing with team, stakeholders, or for compliance.

Example:

```
Generate docs -> Creates 45-page Word document with all model details.
```

Parameters:

- output_path (optional): Where to save
- include_sections (optional): Array of sections to include

What you get back:

Word document (.docx) with formatted, professional model documentation.

Update Model Documentation (Word)

(update_model_documentation_word)

What it does: Updates an existing documentation Word file with current model state (faster than regenerating).

When to use this tool:

Use when you already have documentation and just want to refresh it with latest changes.

Example:

```
Update existing docs -> Updates tables section, adds new measures, updates date.
```

Parameters:

- document_path (required): Path to existing Word document
- sections_to_update (optional): Which sections

What you get back:

Updated Word document with current model information.

08 - Comparison

Compare Two Models

(compare_pbi_models)

What it does: Compares two Power BI models side-by-side to find differences (useful for comparing versions or dev vs. prod).

When to use this tool:

Use to see what changed between versions or to compare development and production models.

Example:

```
Compare Model A (dev) vs. Model B (prod) -> Shows: 5 new measures, 2 tables added, 12 measures changed.
```

Parameters:

- model1_index (required): First model index
- model2_index (required): Second model index

What you get back:

Detailed comparison report with all differences categorized (added, removed, modified).

09 - PBIP Analysis

Analyze PBIP Repository (Offline)

(analyze_pbip_repository)

What it does: Analyzes a PBIP project folder (the new Power BI project format) and generates an HTML analysis report - works offline without Power BI running.

When to use this tool:

Use to analyze PBIP projects stored in Git/version control without opening Power BI Desktop.

Example:

```
Analyze C:\Projects\SalesModel.pbip -> Generates interactive HTML report.
```

Parameters:

- pbip_folder_path (required): Path to .pbip project folder
- output_path (optional): Where to save HTML

What you get back:

Interactive HTML report with model analysis, browsable in any web browser.

10 - Hybrid Analysis

Export Hybrid Analysis Package

(export_hybrid_analysis)

What it does: Exports a complete package combining: TMDL files + metadata + sample data from active model. Perfect for AI analysis!

When to use this tool:

Use to create a complete offline analysis package that includes structure, metadata, and actual sample data.

Example:

```
Export Sales model -> Creates folder with: TMDL files, metadata JSON, sample data
parquet files.
```

Parameters:

- pbip_folder_path (required): Path to .SemanticModel folder
- output_dir (optional): Where to save
- sample_rows (optional, default 1000)

What you get back:

Complete package folder with TMDL, JSON metadata, and Parquet sample data files.

Analyze Hybrid Model

(analyze_hybrid_model)

What it does: BI EXPERT ANALYSIS - Automatically reads exported package and provides comprehensive AI-powered analysis with recommendations. Supports fuzzy search!

When to use this tool:

Use for the most comprehensive AI-powered model analysis - it reads EVERYTHING internally. Just point it at the analysis folder!

Example:

```
Analyze exported package -> Provides: Model structure analysis, DAX recommendations,
data quality insights, optimization opportunities.
```

Parameters:

- analysis_path (required): Path to exported analysis folder
- operation (required): 'smart_analyze'
- intent (optional): Natural language question for fuzzy search

What you get back:

Comprehensive BI Expert analysis with structure insights, DAX recommendations, data quality analysis, and action items.

11 - Help

Show User Guide

(show_user_guide)

What it does: Displays comprehensive user guide with examples, best practices, and tool usage instructions.

When to use this tool:

Use whenever you need help or want to learn about specific features.

Example:

```
Show user guide -> Returns: Full documentation with categories, descriptions,
examples.
```

Parameters:

- section (optional): Specific section to show

What you get back:

User guide with usage instructions and examples.

12 - Monitoring

Get Token Usage Statistics

(get_token_usage)

What it does: Shows token usage statistics for your session - helps monitor API costs and usage patterns.

When to use this tool:

Use to track how many tokens your analysis has consumed.

Example:

```
Get stats -> Shows: Total tokens used, tokens per tool, session duration.
```

Parameters:

- None required

What you get back:

Token usage breakdown by tool and total session statistics.

Common Workflows

Step-by-step guides for common tasks

Workflow 1: First-Time Model Analysis

Get started with any new Power BI model

1. `detect_powerbi_desktop` - Find your model
2. `connect_to_powerbi` - Connect to it
3. `simple_analysis` - Quick overview (2-5 seconds)
4. `full_analysis` - Deep health check (optional)
5. `generate_model_documentation_word` - Create documentation

Workflow 2: Analyze and Optimize DAX Measure

Debug and improve measure performance

1. `measure_operations (list)` - Find your measure
2. `dax_intelligence` - Analyze it (auto-fetches by name!)
3. `Review suggestions` - Validation, performance, optimizations
4. `measure_operations (update)` - Apply optimizations
5. `analyze_measure_dependencies` - Check what depends on it

Workflow 3: Complete AI-Powered Analysis

The most powerful analysis workflow

1. `export_hybrid_analysis` - Export complete package
2. `analyze_hybrid_model` - AI analyzes EVERYTHING
3. `Review insights` - Structure, DAX, data quality, optimization
4. `Use fuzzy search` - "show me revenue measures"

Workflow 4: Compare Dev vs Production

Find differences between model versions

1. `detect_powerbi_desktop` - Find both models
2. `connect_to_powerbi` - Connect to first (index=0)
3. `compare_pbi_models` - Compare them
4. `Review differences` - Added, removed, modified objects

Workflow 5: Refactor Model Safely

Rename objects with automatic reference updates

1. `tmdl_operations (export)` - Export current model as TMDL
2. `tmdl_operations (bulk_rename)` - Rename objects (updates all references!)
3. `Apply TMDL` - Apply TMDL back to model

Tips for Success

✓ **Always connect first:** Use detect_powerbi_desktop and connect_to_powerbi before any other tools

✓ **Start simple:** Use simple_analysis before full_analysis - it's faster and often sufficient

✓ **Smart measure finder:** The dax_intelligence tool has smart measure finder - just type the measure name!

✓ **Preview first:** Use dry_run=true when testing changes (previews without applying)

✓ **Document regularly:** Export documentation regularly for team sharing

✓ **AI-powered insights:** Use hybrid analysis tools for the most comprehensive AI-powered insights

Getting Help:

- Use tool `show_user_guide` for built-in help
- All tools include detailed error messages to guide you
- Check the GitHub repository for updates and support