MCP Version Mismatch Handling

How the MCP server handles solutions created with a different FrameworX version, and how to point an MCP client at a specific installed version.

Starting with FrameworX 10.1.5, multiple product versions can be installed side-by-side on the same machine (10.1.4 on .NET 8 and 10.1.5 on .NET 10 run as isolated runtimes by design). When an MCP client attaches to a DesignerMCP server for a given version and then requests a solution that was created with a *different* version, the server must decide whether to open it directly, refuse, or hand control back to the LLM so the user can choose.

This page covers what happens in each case, the structured JSON responses the LLM sees, and how to reconfigure an MCP client to target an older installed version when needed.

Compatibility Baselines

Two facts drive every routing decision:

- **10.1.0 through 10.1.4 are patch releases of the same baseline.** They are interchangeable — any 10.1.x Designer in this range can open any 10.1.0-10.1.4 solution. Crossing inside the baseline does not require a prompt or an explicit upgrade.
- **10.1.5 introduces a new baseline.** Crossing from 10.1.0-10.1.4 into 10.1.5 is an explicit upgrade (schema migration + backup). Crossing the other way is not possible — 10.1.5 solutions cannot be opened by a 10.1.4 Designer.

Legacy 9.2 and earlier (`.tproj`) solutions are out of scope for MCP. Use Solution Center to upgrade them first.

Routing Outcomes

When `open_solution` (DesignerMCP / ConsoleMCP) or `open_workspace` (ConsoleMCP) is called, the MCP server reads the solution's `ProductVersion` and compares it to the running server's version. One of three outcomes applies.

Outcome	When it applies	What the LLM sees
Open directly	Same version, or solution is in the 10.1.0-10.1.4 baseline range and the running server is in the same baseline, or solution is older than the running server and no side-by-side install of the older version exists.	Normal success response. If the solution is crossing a baseline, the Designer silently creates a versioned backup (`_fx-10.1.4.dbsln`) and migrates the schema on open.
Prompt upgrade	Solution is older than the running server AND the matching older version is currently installed side-by-side (the user has a genuine choice).	Structured JSON with `versionMismatch: true` and two `options`: `upgrade` (call `open_solution` again to let the running server migrate with backup) and `use_older_version` (reconfigure the MCP client to point at the older installed version's `DesignerMCP.dll`). See the response example below.
Version newer (cannot open)	Solution was created with a version *newer* than the running server.	Error with code `VERSION_NEWER` and a message identifying both the solution version and the running version. The user must install a newer product version and reconfigure the MCP client.

Prompt-Upgrade Response Format

When the server returns **Prompt upgrade**, the response is success-shape JSON (not an error) so the LLM can present the options in chat rather than fail the turn:

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

```
{
  "success": false,
  "versionMismatch": true,
  "solutionVersion": "fx-10.1.4",
  "runningVersion": "fx-10.1.5",
  "solutionName": "Demo",
  "message": "Solution 'Demo' was created with version fx-10.1.4. This MCP server is running version fx-
10.1.5.",
  "options": [
    {
      "option": "upgrade",
      "description": "Upgrade the solution from fx-10.1.4 to fx-10.1.5. A backup will be created
automatically. To proceed, call open_solution again — the upgrade happens automatically."
    },
    {
      "option": "use_older_version",
      "description": "Keep the solution at version fx-10.1.4. The user must change their MCP server
configuration to point to the older version's DesignerMCP: C:\\Program Files\\Tatsoft\\FrameworX\\fx-10.1.4
\\net8.0\\DesignerMCP.dll"
    }
  ]
}
```

LLM retry semantics

If the user accepts the upgrade, the LLM calls `open_solution` a second time for the same solution. The server remembers that the version mismatch has already been surfaced for that solution name and **bypasses** the check on the retry — the second call proceeds directly into the normal open flow, which creates the versioned backup and migrates the schema transparently.

The bypass is per-solution-name and applies only to `PromptUpgrade`. The `VERSION_NEWER` outcome is *not* bypassed on retry: a solution that requires a newer product version cannot be opened no matter how many times the LLM is asked, so the server keeps refusing until the product is updated.

Pointing an MCP Client at a Specific Installed Version

When the user chooses `use_older_version`, the fix is in the MCP client configuration — not in FrameworX. Replace the `DesignerMCP.dll` path with the one that matches the solution's version.

Install-Path Convention

FrameworX 10.1.5 installers place each version in its own folder under `C:\Program Files\Tatsoft\FrameworX\`. The runtime subfolder depends on the version's .NET target.

Solution / running version	DesignerMCP path
10.1.0 – 10.1.4 (net8.0)	`C:\Program Files\Tatsoft\FrameworX\fx-10.1.4\net8.0\DesignerMCP.dll`
10.1.5 and later (net10.0)	`C:\Program Files\Tatsoft\FrameworX\fx-10.1.5\net10.0\DesignerMCP.dll`

The `fx-10.1.X` folders are created by the installer and exist as long as the matching version is installed side-by-side. The default `fx-10` folder used by earlier documentation continues to point at the primary installed version; the versioned folders are the stable path for multi-version setups.

Example: Claude Desktop

To target 10.1.4 from Claude Desktop, edit `%APPDATA%\Claude\claude_desktop_config.json`:

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

```
{
  "mcpServers": {
    "FrameworX-Designer": {
      "command": "dotnet",
      "args": [
        "C:\\Program Files\\Tatsoft\\FrameworX\\fx-10.1.4\\net8.0\\DesignerMCP.dll"
      ],
      "transport": "stdio"
    }
  }
}
```

Save the file, fully quit Claude Desktop, and relaunch. The MCP server that starts up is now the 10.1.4 server — the `versionMismatch` response disappears for 10.1.4 solutions.

Example: VS Code GitHub Copilot

For Copilot, edit `mcp.json` (Command Palette `MCP: Open User Configuration`) and use the same versioned path in `args`:

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

```
{
  "servers": {
    "FrameworX-Designer": {
      "type": "stdio",
      "command": "dotnet",
      "args": [
        "C:\\Program Files\\Tatsoft\\FrameworX\\fx-10.1.4\\net8.0\\DesignerMCP.dll"
      ]
    }
  }
}
```

Restart the MCP server from `MCP: List Servers` after saving.

The `use_older_version` choice is a one-time reconfiguration per solution — it changes which MCP server Claude talks to, not which FrameworX Designer runs. Once pointed at the 10.1.4 server, the user continues to work on the 10.1.4 solution with the 10.1.4 Designer until they decide to upgrade.

Troubleshooting

Symptom	Likely cause	Fix
`VERSION_NEWER` error on `open_solution`	The solution was created with a product version newer than the MCP server. This is expected behavior — the server refuses on purpose to protect the file.	Install the newer product version side-by-side, then edit the MCP client config to point at the newer `DesignerMCP.dll`.
LLM keeps receiving `versionMismatch: true` after the user accepted the upgrade	The second `open_solution` call used a different solution name or path than the first, so the per-solution bypass did not apply.	Call `open_solution` with the same `solution_name` as the first call. The bypass is keyed by that exact name.
No prompt appears when opening a 10.1.3 solution on 10.1.5	10.1.3 is inside the 10.1.0-10.1.4 baseline. The server opens it directly — no prompt is expected in this case.	None. The Designer will silently create a versioned backup if any schema migration is needed.
`use_older_version` path points to a DLL that does not exist	The older product version was uninstalled after the solution was created, or the folder layout was customized at install time.	Reinstall the matching version, or accept `upgrade` instead to migrate the solution onto the currently running version.

Related

- MCP SDK Reference — server surfaces, runtime requirements, and transport model.
- MCP and Claude Setup — end-to-end setup for Claude Desktop and VS Code Copilot.
- Claude Code MCP Setup — Claude Code integration, including ConsoleMCP for file-based engineering.

In this section...
