ConsoleMCP Reference (10.1.5 draft)

Draft preview of the 10.1.5 additions to ConsoleMCP — compile feedback on `create_solution_file` and `update_solution_file`, and the new `verify_solution_file` tool.

This is a 10.1.5 draft preview. The tools and parameters described here are not yet released in the currently shipping version. When 10.1.5 ships, this content will be merged into the parent page and this draft retired.

10.1.5 surfaces the compile output that already lives inside a produced `.dbsln`. Both `create_solution_file` and `update_solution_file` now include the shared `build` block in their response, drawn directly from the error columns of the four `*Contents` tables inside the solution file. A new tool, `verify_solution_file`, lets an agent inspect a `.dbsln` after the fact and confirm that every expected object is present and compiled cleanly.

The JSON shape returned by both paths is defined once on MCP SDK Reference (10.1.5 draft) — the "`build` Block" section there is the authoritative shape. This page covers only the ConsoleMCP-specific entry points.

create_solution_file / update_solution_file — `build` block

Both tools already return per-file import attribution and the headless `SolutionCreator.exe` log. In 10.1.5 they additionally carry the shared `build` block, so the agent sees compile status for every Script and Display in the produced solution without a second call.

```
{
  "solution":  "C:\\Solutions\\Foo.dbsln",
  "exitCode":  0,
  "imported":  { "created": 12, "modified": 0, "skipped": 0, "errors": 0 },
  "build": {
    "objects": [
      { "type": "Script",  "name": "Line1_Cycle", "status": "ok",    "diagnostics": [], "elapsedMs": 42 },
      { "type": "Display", "name": "MainPage",    "status": "error",
        "diagnostics": [ { "line": 18, "msg": "Cannot implicitly convert type 'string' to 'int'" } ],
        "elapsedMs": 91 }
    ],
    "summary": { "built": 11, "failed": 1, "skipped": 0, "timestamp": "2026-04-21T16:02:11Z" }
  }
}
```

Source of the data

A `.dbsln` is a SQLite database. Each of the four affected tables — `ScriptsTasks`, `ScriptsClasses`, `ScriptsExpressions`, `DisplaysList` — has a companion `*Contents` table (`ScriptsTasksContents`, `ScriptsClassesContents`, `ScriptsExpressionsContents`, `DisplaysListContents`) whose error columns hold the compile result written at build time by `SolutionCreator`.

ConsoleMCP queries those columns after the build completes and assembles the `build` block from the result. This is NOT stdout parsing of the `SolutionCreator` log — it is a direct read of the compiled result persisted inside the produced `.dbsln`. The same data the Designer displays when it opens the solution is the data ConsoleMCP reports back to the agent.

A non-zero `summary.failed` does not by itself fail the tool call. The `.dbsln` is still produced and the agent can inspect `build.objects[]` to decide whether to correct the workspace and re-run.

verify_solution_file

Reads an existing `.dbsln` and returns two things: a per-table Name inventory of every object in the solution, and the shared `build` block drawn from the `*Contents` tables. Optionally diffs the inventory against a caller-supplied list of expected object names.

This is a verification tool, not a read tool. It returns Names only — no row contents, no field values. Use `get_objects` to read row contents.

```
verify_solution_file(
  solution_name:   string,                              // .dbsln base name or absolute path
  expected_names:  { <TableType>: [<Name>, ...], ... }  // optional diff input
) -> {
  solution:    "<path>",
  inventory:   { <TableType>: [<Name>, ...], ... },
  build:       <shared build block>,
  missing:     { <TableType>: [<Name>, ...], ... },     // present only when expected_names supplied
  unexpected:  { <TableType>: [<Name>, ...], ... }      // present only when expected_names supplied
}
```

Parameter	Required	Notes
solution_name	Yes	The `.dbsln` to open. Either a base name resolved against the standard Solutions folder or an absolute path ending in `.dbsln`.
expected_names	No	Object shaped as `{ table_type: ["Name1", "Name2", ...], ... }`. When supplied, the response includes `missing[]` (expected but not found) and `unexpected[]` (found but not expected) per table type.

What it opens

`verify_solution_file` opens only current-version `.dbsln` files. It performs a direct SQLite read — it does not go through the DbslnExplorer pipeline and does not handle legacy formats, password-protected files, or pre-migration schema versions. A `.dbsln` that does not match the current schema version returns `SOLUTION_VERSION_MISMATCH`; the caller is expected to upgrade the solution through Designer first.

Example

```
verify_solution_file(
  solution_name = "Foo",
  expected_names = {
    "ScriptsTasks":   ["Line1_Cycle", "Line2_Cycle"],
    "DisplaysList":   ["MainPage", "AlarmsPage"],
    "UnsTags":        ["Plant1/Temperature", "Plant1/Pressure"]
  }
)

# Response (abbreviated)
{
  "solution": "C:\\Solutions\\Foo.dbsln",
  "inventory": {
    "ScriptsTasks":   ["Line1_Cycle", "Line2_Cycle"],
    "DisplaysList":   ["MainPage"],
    "UnsTags":        ["Plant1/Temperature", "Plant1/Pressure", "Plant1/Flow"]
  },
  "build": { "objects": [...], "summary": { "built": 2, "failed": 0, "skipped": 0, "timestamp": "..." } },
  "missing": {
    "DisplaysList": ["AlarmsPage"]
  },
  "unexpected": {
    "UnsTags": ["Plant1/Flow"]
  }
}
```

An agent that writes a workspace to a specification typically pairs `create_solution_file` or `update_solution_file` with `verify_solution_file` using the spec's expected-names list. The pair catches both build breaks (via `build.summary.failed`) and inventory drift (via `missing[]` and `unexpected[]`).

Error Codes (new in 10.1.5)

Code	Where	Meaning

SOLUTION_VERSION_MISMATCH	verify_solution_file	The target .dbsln is not a current-version file. Upgrade through Designer first; verify_solution_file does not migrate.
SOLUTION_ENCRYPTED	verify_solution_file	The target .dbsln is password-protected. verify_solution_file does not accept credentials — use the Designer session for protected files.
INVALID_EXPECTED_NAMES	verify_solution_file	expected_names is not shaped as { table_type: [names], ... }. Response includes an examples array with the correct shape.

In this section...