

designer_action Reference (10.1.5 draft)

Draft preview of the 10.1.5 additions to `designer_action` and related DesignerMCP tools — new `build` action parameters and compile feedback on write/read responses.

[designer_action Reference](#) designer_action Reference (10.1.5 draft)



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 adds build-visibility signals to the DesignerMCP surface. The `build` action gains two parameters that mirror the `RuntimeBuildAndPublish` dialog, `write_objects` responses carry per-row `compile` feedback for the four tables that hold user-authored .NET, `get_objects(detail='full')` surfaces a `lastCompile` field on those same rows, and the solution context carries a compact `build_state` section.

[designer_action\('build', ...\)](#) — new parameters
[write_objects](#) — per-row compile field
[get_objects\(detail='full'\)](#) — `lastCompile` field
[SolutionContext / RefreshContext](#) — `build_state` section
[Error Codes](#)

The JSON shape returned by the build surface 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 DesignerMCP-specific entry points.

designer_action('build', ...) — new parameters

The `build` action invokes the same pipeline as the Designer "Build and Publish" dialog. In 10.1.5 it accepts two named parameters that mirror the dialog's checkboxes.

```
designer_action(  
  action:          "build",  
  rebuild_all:     bool = false, // false = incremental, true = full rebuild  
  validate_displays: bool = true  // run Display compile pass in addition to Scripts  
) -> { build: <block> }
```

| Parameter | Default | Effect |
|--------------------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| rebuild_all | false | When false, only objects modified since the last build are recompiled; unchanged objects appear in <code>summary</code> . skipped. When true, every object in the four affected tables is recompiled regardless of modification state. Mirrors the <i>Rebuild All</i> checkbox in the Build and Publish dialog. |
| validate_displays | true | When true, <code>DisplaysList</code> code-behind is compiled alongside the three Scripts tables. When false, only Scripts are compiled (useful when iterating on script logic without wanting the Display compile cost). Mirrors the <i>Validate Displays</i> checkbox in the Build and Publish dialog. |

The response is the shared build block. See [MCP SDK Reference \(10.1.5 draft\)](#) for the full shape.

```
# Fast path during iterative development  
designer_action('build')  
# == designer_action('build', rebuild_all=false, validate_displays=true)  
  
# Pre-deploy check - recompile everything  
designer_action('build', rebuild_all=true, validate_displays=true)  
  
# Scripts-only run while debugging script logic  
designer_action('build', validate_displays=false)
```

write_objects — per-row compile field

When a `write_objects` call touches a row in one of the four affected tables (`ScriptsTasks`, `ScriptsClasses`, `ScriptsExpressions`, `DisplaysList`), the response entry for that row carries a `compile` field populated from the save-time incremental compile.

```
{
  "results": [
    {
      "table": "ScriptsTasks",
      "name": "Line1_Cycle",
      "status": "written",
      "compile": {
        "status": "error",
        "diagnostics": [
          { "line": 12, "msg": "The name 'tagx' does not exist in the current context" }
        ]
      }
    },
    {
      "table": "ScriptsClasses",
      "name": "TankUtils",
      "status": "written",
      "compile": { "status": "ok", "diagnostics": [] }
    }
  ]
}
```

Writes to rows outside the four affected tables carry no `compile` field — those objects do not participate in the .NET compile pipeline.

The row-level `compile` reports only what the incremental pipeline saw for that single object. Cross-object breakage — for example, an edit to a `ScriptsClasses` row that breaks a `ScriptsTasks` row referencing it — is surfaced by a subsequent `designer_action('build')` call, not by the `write` response alone.

get_objects(detail='full') — lastCompile field

When `get_objects` is called with `detail='full'` against one of the four affected tables, each returned row carries a `lastCompile` field holding the most recent compile result for that object.

```
{
  "table": "ScriptsTasks",
  "name": "Line1_Cycle",
  "code": "...",
  "lastCompile": {
    "status": "ok",
    "at": "2026-04-21T16:02:11Z",
    "diagnostics": []
  }
}
```

| Field | Meaning |
|----------------------------------------|----------------------------------------------------------------------------|
| <code>lastCompile.status</code> | ok or error, matching the shared build block convention. |
| <code>lastCompile.at</code> | ISO-8601 timestamp of the compile that produced this result. |
| <code>lastCompile.diagnostics[]</code> | Same shape as elsewhere — line and msg per entry, empty when status is ok. |

Calls with `detail='summary'` or the default detail level do not include `lastCompile` — it is only emitted on the full-detail path to keep the summary response compact.

SolutionContext / RefreshContext — build_state section

The context packages returned to the agent on connect and on refresh now include a compact `build_state` section. This gives the agent a one-shot view of compile health without needing to call `designer_action('build')` at session start.

```
{
  "build_state": {
    "built": 42,
    "failed": 1,
    "skipped": 0,
    "timestamp": "2026-04-21T16:02:11Z"
  }
}
```

The fields match the `summary` portion of the shared `build` block. Per-object diagnostics are not included here — callers that need them issue `designer_action('build')` or `get_objects(detail='full')` on the affected table.

Error Codes

| Code | When | Recovery |
|-------------------|------------------------------------------------------------------------------|------------------------------------------------------------------------------|
| BUILD_IN_PROGRESS | Another <code>build</code> call is already running for the active solution. | Wait for the in-flight call to complete. The second call does not queue. |
| INVALID_PARAMETER | <code>rebuild_all</code> or <code>validate_displays</code> is not a boolean. | Response includes an <code>examples</code> array with the correct call form. |

In this section...