A RESOURCE FOR DESIGN SYSTEM AND ENGINEERING TEAMS

# Token architecture diagram.

Draw it from the token files, or it drifts.

Generated per mode. Checked in CI. Read three ways.

**20**
checks, each with its evidence

**34**
aliases drawn per mode from Field Guide 03's tokens

**09**
errors on the broken fixture; nothing drawn

## A diagram is a claim about the token files. Make the files prove it.

Most token architecture diagrams are drawn once, in a slide, from memory. They show three tidy boxes and a few arrows, and they stop being true the week a semantic token gets a literal value or a component token skips a layer. Nobody notices, because nobody diffs a picture.

This guide treats the diagram as build output. A zero-dependency Node script reads the DTCG 2025.10 resolver and token files, checks that every alias respects the layering, and only then writes the diagram: Mermaid source and a standalone SVG for each mode, a layer view for leadership and the graph as JSON for agents. If the files break the architecture, the script prints the errors and draws nothing. In CI, `--check` fails when the committed diagram no longer matches the files.

The kit ships the diagrams generated from the Field Guide 03 token files (25 core, 23 semantic and 11 component tokens), so every number in this guide is one you can reproduce. Field Guide 23 names the semantic column.

**Version 1.0 / Sources checked 24 September 2026**
Field Guide 22 of the Design × AI series. Pairs with Field Guide 03 (token layer rules) and 23 (semantic token naming). Verified 24 September 2026: Node 22.22, Mermaid 11.15.0 (every generated .mmd file parsed and rendered in Chromium), DTCG 2025.10.

Practical guidance, not a standard. The layer checks are the Field Guide 03 rules restated as graph properties. The generator reads the DTCG resolver's sets and modifiers; it does not implement every resolver feature. Prepared with AI assistance and edited by hand.

START HERE

# One diagram, three readers.

Engineers, leadership and agents need different views of the same graph. The generator writes all three from one source, so they cannot disagree with each other or with the files.

### Engineers

The full diagram per mode, plus `--trace` and `--impact` before any core change. Section 04, item G17.

### Leadership

The layer view: five boxes, counts on the arrows, one sentence about dark mode. Item G18.

### Agents

`tokens.graph.json` and the generator as a verify step. Agents read data, not pictures. Item G19.

### Keeping it true

Sections 01 and 02: generate, check, commit, and fail CI on a stale diagram or a broken layer.

## What the marks mean, in every output.

| Mark | Meaning |
|---|---|
| Arrow A to B | B aliases A. Values flow left to right; the `{reference}` in the file points right to left. |
| Left column | Core: literal values, private, build input only. |
| Middle column | Semantic: decisions, the public API. The only column application code and agents may use. |
| Right column | Component: tokens scoped to one component, each aliasing a semantic token. |
| Dashed box | Deprecated, or a core value that no semantic token uses in any mode. |
| Dashed arrow (poster only) | Dark mode. Solid arrows are light mode. |

| Label | Meaning |
|---|---|
| `SCRIPT` | Done by `generate-diagram.mjs` every time it runs. |
| `CHECK` | The generator exits 1 and draws nothing, or `--check` fails CI. |
| `WCAG A` | Traces to WCAG 2.2 SC 1.1.1 Non-text Content. |
| `AGENT` | Exists so an agent can read the architecture as data. |
| `PRACTICE` | A working method with a review signal rather than a hard gate. |

SECTION 01

# Generate it from the files.

A hand-drawn diagram is a second source of truth that nobody maintains. Draw from the token files, commit the output, and let CI tell you when it is stale.

**Suggested owners:** Design-system lead + token owner

- [ ] **The diagram is build output**

  G01 SCRIPT

  Every file in `diagram/` is written by the generator. The annotated poster is the one hand-made exception, and every number on it comes from a generator run.

  **Evidence:** `diagram/` holds six generated files, each headed "Generated by generate-diagram.mjs. Do not edit."

  Same principle as Field Guide 03 R25: generated files are never hand-edited.

- [ ] **Modes come from the resolver, not from file names**

  G02 SCRIPT

  The generator walks `resolutionOrder` in `tokens.resolver.json`: sets are always included, each modifier context becomes a mode. Two modifiers produce the cross product.

  **Evidence:** Against Field Guide 03's resolver the generator finds the modes `light` and `dark` and loads three files for each: core, that mode's semantic file and the Button component file.

  DTCG 2025.10 Resolver module: sets, modifiers, contexts, resolution order. S02.

- [ ] **One diagram per mode, same layout**

  G03 SCRIPT

  Node order is file order in every mode, so light and dark diagrams line up and only arrows move. That makes the mode difference visible at a glance and in a diff.

  **Evidence:** `diff tokens.light.mmd tokens.dark.mmd`: 14 edge lines and the title line change; nothing else.

  Kit output.

- [ ] **Commit the output, and fail CI when it is stale**

  G04 CHECK

  Diagrams live next to the tokens and change in the same pull request. `--check` regenerates in memory and compares byte for byte.

  **Evidence:** `node scripts/generate-diagram.mjs --check` prints `diagram: up to date (6 files, 2 modes, 2 warnings)`; any drift exits 1 and lists the stale files.

  Kit script.

- [ ] **Text first, pictures second**

  G05 SCRIPT AGENT

  Mermaid source renders in GitHub, GitLab and most docs tools, diffs line by line and can be read by a model. The SVG is for slides and print; the JSON is for scripts and agents.

  **Evidence:** All three `.mmd` files parsed and rendered with Mermaid 11.15.0 in Chromium on 24 September 2026.

  Mermaid flowchart syntax; GitHub renders fenced `mermaid` blocks. S04, S05.

**If the diagram can go stale, it already has.**

SECTION 02

# Refuse to draw a lie.

Each check is a Field Guide 03 rule restated as a property of the alias graph.

**Suggested owners:** Token owner + engineering lead

- [ ] **No arrow enters the core column**

  G06 CHECK

  A core token that aliases anything is not a value any more. Code D1.

  **Evidence:** The fixture's `core.color.accent -> {core.color.blue.600}`, reported once for all modes.

  Field Guide 03 R01; option tokens hold values. S03.

- [ ] **Every semantic token has exactly one arrow in, from core**

  G07 CHECK

  No literal (D2), no alias to a semantic or component token (D3). Chains hide which value a theme uses.

  **Evidence:** The fixture reports D2 once and D3 three times: both halves of a cycle and one upward alias.

  Field Guide 03 R05; decision tokens reference option tokens. S03.

- [ ] **Component arrows come from semantic, never from core**

  G08 CHECK

  A component token that skips the semantic layer does not follow the mode. Code D4.

  **Evidence:** `comp.button.primary.bg -> {core.color.blue.600}` fails in every mode.

  Field Guide 03 R11.

- [ ] **Every arrow lands on a token**

  G09 CHECK

  An alias to a missing token or to a group has no value. Code D5.

  **Evidence:** `color.border.focus -> {core.color.blue.650}`, reported with its mode.

  DTCG 2025.10 Format: references target tokens with a `$value`. S01.

- [ ] **No cycles**

  G10 CHECK

  A loop has no value in any tool. Each loop is reported once, with its path. Code D6.

  **Evidence:** `alias cycle color.text.default -> color.text.muted -> color.text.default`.

  DTCG 2025.10: references must not be circular, and tools must report it. S01.

- [ ] **Every mode has the same semantic keys**

  G11 CHECK

  A token that exists only in light breaks dark silently. Code D7.

  **Evidence:** The fixture's `color.bg.subtle` exists in light only and is reported as `light/dark`.

  Field Guide 03 R07.

**Fix the tokens, not the picture.**

SECTION 03

# Say what every mark means.

Token diagrams fail most often on the arrow. Half the diagrams in circulation point from alias to target, the other half the other way, and neither says which.

**Suggested owners:** Design-system lead + docs owner

- [ ] **Unused core values are shown, not hidden**

  G12 SCRIPT

  A core value nothing points at is a gap in the semantic layer or dead weight. It is drawn dashed with a warning, not a failure: a palette may hold values ahead of use.

  **Evidence:** Field Guide 03's tokens have two: `core.color.neutral.200` and `core.space.300`. Warning code W1.

  Kit script.

- [ ] **One arrow direction, written in the legend**

  G13 SCRIPT

  This kit draws A to B when B aliases A: values flow toward the consumer, the same direction as the essay's Mermaid diagram. The `{reference}` in the JSON points the opposite way, and the legend says so.

  **Evidence:** The legend line and the second comment line of every `.mmd` file state it.

  C4 notation: a key or legend on every diagram, labelled unidirectional relationships. S07.

- [ ] **Title, scope and counts on every diagram**

  G14 SCRIPT

  The title names the mode; the subtitle says what generated it and how many tokens and aliases it shows. A diagram without counts cannot be checked against the files.

  **Evidence:** SVG subtitle: "25 core, 23 semantic, 11 component tokens, 34 aliases."

  C4 notation: a title describing type and scope. S07.

- [ ] **One meaning per line style**

  G15 PRACTICE

  Generated diagrams use dashed boxes for deprecated or unused tokens. The poster uses dashed arrows for dark mode. Neither reuses the other's meaning, and both say so in their own legend.

  **Evidence:** Review signal: a new style is added to the legend in the same change.

  C4 notation: consistent, explained notation that survives black-and-white printing. S07.

- [ ] **The picture has a text equivalent**

  G16 WCAG A

  Each SVG has a `<title>` and a `<desc>` with the counts and the arrow meaning, referenced by `aria-labelledby`. The long description is the Mermaid source and `tokens.graph.json`, linked next to the image wherever it is published.

  **Evidence:** `role="img"`, `<title>` as the first child, `<desc>` naming `tokens.graph.json`.

  WAI complex images: a short and a long description. MDN on SVG title. S06, S08.

**An arrow nobody can read is decoration.**

SECTION 04

# Present it to the reader in front of you.

The same graph answers different questions. Pick the view by the question, not by the audience's seniority.

**Suggested owners:** Design-system lead

- [ ] **Engineers: trace one token, then check the impact**

  G17 SCRIPT

  Walk one token end to end in both modes, then ask what a core change would touch. Do this before changing any core value, and paste the output into the pull request.

  **Evidence:** `--impact core.color.blue.600` lists five dependents in light and none in dark; `--trace comp.button.primary.bg` ends at #125bd0 in light and #80adff in dark.

  Kit script.

- [ ] **Leadership: the layer view and one sentence**

  G18 PRACTICE

  Show `tokens.summary.mmd`: five boxes and counts on the arrows. The sentence to say: dark mode is 14 alias changes in one file and no component changes. The middle column is the contract; everything else can change behind it.

  **Evidence:** The summary's arrow labels read "23 aliases; 14 change in dark" and "11 aliases, same in every mode".

  Private option tokens allow non-breaking changes. S03.

- [ ] **Agents: the graph file, never the image**

  G19 AGENT

  Give agents `tokens.graph.json` (layers, edges per mode, mode changes, deprecations) and tell them to run the generator as their verify step after touching tokens. An image costs tokens and is read approximately; the JSON is exact.

  **Evidence:** The instruction file names `tokens.graph.json` and `npm run diagram:check`.

  Field Guide 04 covers where agent instructions live.

- [ ] **Review the diagram diff, not the screenshot**

  G20 PRACTICE

  A token pull request that changes an alias changes one edge line in a `.mmd` file. Reviewers read that line; the rendered picture is for the discussion afterwards.

  **Evidence:** Switching `color.border.focus` to another core token changes exactly one edge line in that mode's `.mmd` file.

  Kit output.

**Same graph, three questions, one source.**

APPENDIX A

# The generated diagrams.

Excerpts from the kit's `diagram/` folder, generated from the Field Guide 03 token files. Paste any `.mmd` file into a fenced `mermaid` block to render it; open the `.svg` files directly.

diagram/tokens.summary.mmd (generated, complete)

```
flowchart LR
  core["Core: 25 values<br/>private, build input only<br/>2 unused"]
  semantic["Semantic: 23 decisions<br/>the public API"]
  component["Component: 11 tokens<br/>button only"]
  ui["Component source<br/>button"]
  app["Application code<br/>and agents"]
  core -->|"23 aliases; 14 change in dark"| semantic
  semantic -->|"11 aliases, same in every mode"| component
  component --> ui
  semantic --> ui
  semantic -->|"the only layer it may use"| app
```

The layer view for leadership. Class definitions and the header comments are omitted here.

diagram/tokens.light.mmd (generated, excerpt)

```
%% A --> B means B aliases A. Values flow left to right.
flowchart LR
  subgraph core["Core (private): 25 tokens"]
    core_color_blue_600["core.color.blue.600 = #125bd0"]:::core
    core_color_neutral_200["core.color.neutral.200 = #e2e6ea (unused)"]:::unused
  end
  subgraph semantic["Semantic (public API): 23 tokens"]
    color_bg_brand["color.bg.brand"]:::semantic
    color_link["color.link (deprecated)"]:::deprecated
  end
  subgraph component["Component (scoped): 11 tokens"]
    comp_button_primary_bg["comp.button.primary.bg"]:::component
  end
  core_color_blue_600 --> color_bg_brand
  color_bg_brand --> comp_button_primary_bg
```

| tokens.light.mmd | tokens.dark.mmd |
|---|---|
| `core_color_blue_600 --> color_bg_brand` | `core_color_blue_300 --> color_bg_brand` |
| `core_color_blue_700 --> color_bg_brand_hover` | `core_color_blue_200 --> color_bg_brand_hover` |
| `core_color_neutral_0 --> color_text_on_brand` | `core_color_neutral_950 --> color_text_on_brand` |

Lines 74, 79 and 80: 3 of the 14 edge lines that differ between the light and dark files. The 11 component edges and the 9 dimension aliases are identical.

APPENDIX B

# What the generator prints.

Real terminal output from the kit. Run the same commands after copying the Field Guide 03 token files into `tokens/`, which the ZIP already does.

terminal: the broken fixture

```
$ node scripts/generate-diagram.mjs --tokens examples/violations
warning W1: core.color.accent is used by no semantic token in any mode
error D1 [all modes]: core.color.accent aliases {core.color.blue.600}; core holds literal
values only
error D3 [light]: color.text.default -> {color.text.muted}; semantic tokens alias core tokens
only
error D3 [light]: color.text.muted -> {color.text.default}; semantic tokens alias core tokens
only
error D3 [light]: color.text.link -> {comp.button.primary.bg}; semantic tokens alias core
tokens only
error D2 [light]: color.bg.surface holds a literal; semantic tokens alias core tokens
error D5 [light]: color.border.focus -> {core.color.blue.650}, which is not a token in this
mode
error D4 [all modes]: comp.button.primary.bg -> {core.color.blue.600}; component tokens alias
semantic tokens only
error D6 [light]: alias cycle color.text.default -> color.text.muted -> color.text.default
error D7 [light/dark]: semantic key sets differ: color.bg.subtle

9 errors. No diagram drawn: fix the tokens, not the picture.
```

Every error code D1 to D7 fires at least once. Exit code 1.

terminal: Field Guide 03's tokens

```
$ node scripts/generate-diagram.mjs --check
warning W1: core.color.neutral.200 is used by no semantic token in any mode
warning W1: core.space.300 is used by no semantic token in any mode
diagram: up to date (6 files, 2 modes, 2 warnings)

$ node scripts/generate-diagram.mjs --trace comp.button.primary.bg
light: comp.button.primary.bg -> color.bg.brand -> core.color.blue.600 = #125bd0
dark: comp.button.primary.bg -> color.bg.brand -> core.color.blue.300 = #80adff

$ node scripts/generate-diagram.mjs --impact core.color.blue.600
light: color.text.link, color.bg.brand, comp.button.primary.bg, color.border.focus, color.link
dark: nothing depends on it
```

Warnings are repeated on every run and omitted from the last two commands here.

APPENDIX C

# The SVG files: generated and annotated.

Generated SVGs go in slides and docs. `poster/token-architecture-poster.svg` is an A3 landscape explainer: 14 tokens from the kit, light and dark arrows, both consumers, both blocked paths and six numbered notes.

`diagram/tokens.light.svg (generated, structure)`

```
<svg ... viewBox="0 0 1270 874" role="img" aria-labelledby="t d">
<title id="t">Token architecture, light mode</title>
<desc id="d">Three columns: 25 core, 23 semantic and 11 component tokens. 34 arrows; an arrow
from A to B means B aliases A. ...</desc>
<!-- 34 edge paths, 59 node boxes with swatch and resolved value, legend -->
```

Standalone: no scripts, no external fonts required, white background. Opens in any browser and imports into Figma or Keynote.

| Note | What it says | Where the number comes from |
|---|---|---|
| 1 | An arrow is an alias; the value flows right. | Arrow convention, item G13 |
| 2 | Dark mode moves 14 of 23 core-to-semantic arrows, nothing else. | `tokens.graph.json`, `modeChanges.dark` |
| 3 | Core is private; 2 of 25 values feed nothing. | Warning W1, item G12 |
| 4 | Semantic is the contract: 23 keys in every mode. | Check D7; Field Guide 03 R23 |
| 5 | 11 component tokens, all for Button, identical in both modes. | `tokens.summary.mmd` |
| 6 | No arrow skips a layer or points left. | Checks D1 to D7 |

> **Update the poster when the counts change.**
> The kit's test compares the poster's numbers with a fresh generator run, so a token change that alters a count fails until the poster is edited to match.

`.github/workflows/tokens.yml (steps)`

```
- name: Token layer rules (Field Guide 03)
  run: node scripts/check-token-rules.mjs
- name: Semantic names (Field Guide 23)
  run: node scripts/lint-token-names.mjs tokens/semantic/*.tokens.json
- name: Architecture diagram is current (Field Guide 22)
  run: node scripts/generate-diagram.mjs --check
```

Adapt paths to your repo. The three steps need no npm install.

KEEP WITH THE TOKEN RELEASE

# Leave a diagram review record.

A record for one token release. It shows which picture went with which files, and who looked at the changed arrows.

| | |
|---|---|
| **Token package version / release date** | **Generator run: errors (must be 0) / warnings** |
| **Modes found in the resolver** | **Edge lines changed per mode (diff link)** |
| **Impact output for each changed core value** | **Unused core values: kept or removed, and why** |
| **Summary view shared with (who, where)** | **Poster numbers updated (yes / not needed)** |
| **Reviewer / date** | **Next review trigger** |

> **Never fix a diagram by editing it.**
> If a generated file looks wrong, the token files are wrong or the generator is. Change one of those, regenerate and commit both.

SOURCES / MAINTENANCE

# Keep the guide current.

Sources checked 24 September 2026. The generator and the layer checks were run against the Field Guide 03 token files and the kit's broken fixture on that date.


**S01 / DTCG, Format module 2025.10**
Aliases reference whole tokens; references must not be circular and tools must report cycles.
https://www.designtokens.org/TR/2025.10/format/

**S02 / DTCG, Resolver module 2025.10**
Sets, modifiers with contexts and resolution order: where the generator finds modes.
https://www.designtokens.org/TR/2025.10/resolver/

**S03 / Andreas Kutschmann, Design Token-Based UI Architecture**
Option, decision and component tokens; decision tokens reference option tokens; private options allow non-breaking change (martinfowler.com, 2024).
https://martinfowler.com/articles/design-token-based-ui-architecture.html

**S04 / Mermaid, flowchart syntax**
Subgraphs, classDef, link styles, quoting labels; the reserved word end.
https://mermaid.js.org/syntax/flowchart.html

**S05 / GitHub Docs, creating diagrams**
Fenced mermaid blocks render in Markdown on GitHub.
https://docs.github.com/en/get-started/writing-on-github/working-with-advanced-formatting/creating-diagrams

**S06 / W3C WAI, complex images tutorial**
Diagrams need a short description and a long description.
https://www.w3.org/WAI/tutorials/images/complex/

**S07 / C4 model, notation**
Title, key or legend, labelled unidirectional relationships, notation that survives black and white.
https://c4model.com/diagrams/notation

**S08 / MDN, SVG title element**
Accessible short description; first child for SVG 1.1 compatibility; prefer aria-labelledby.
https://developer.mozilla.org/en-US/docs/Web/SVG/Reference/Element/title


Maintenance: regenerate after every token change (`npm run diagram`) and keep `diagram:check` in CI. Recheck the DTCG resolver module at its next release and extend `loadResolver` if new resolution features land. Update the PDF, HTML, Markdown and JSON together.