
A RESOURCE FOR DESIGN SYSTEM AND ENGINEERING TEAMS

Token layer rules.



Semantic tokens in, raw values out.

Private core. Public meaning. Enforced in CI.

25

rules, each with its test

03

layers: core, semantic,
component

13

contrast pairs, checked in light
and dark

Give the codebase one vocabulary, and lock the rest away.

Raw tokens carry no intent. `core.color.blue.600` tells a model the value, not whether it is a link, a brand fill or a focus ring, so the model picks something plausible and the product drifts. Semantic tokens encode the decision: `color.border.focus` means focus ring in every theme. When application code can reach only that layer, a person or an agent has fewer wrong options to choose from.

This guide turns that into 25 rules across three layers: core (private values), semantic (the public API) and component (optional, internal to one component). Each rule names the test that enforces it, and the kit runs every one of them: DTCG 2025.10 token files validated against the official schemas, a rules checker, a WCAG contrast check in light and dark, a Style Dictionary 5 build, and ESLint and stylelint configurations that reject core tokens, component tokens and raw values in application code.

The kit's Button tokens are the same ones the Button contract in Field Guide 02 references and the promptable Button page in Field Guide 04 documents.

Version 1.1 / Sources checked 24 September 2026

Field Guide 03 of the Design × AI series. Tool versions verified 24 September 2026: Style Dictionary 5.5.5, ESLint 10.11, typescript-eslint 8.70, stylelint 17.15.

Practical guidance, not a standard. DTCG 2025.10 is a Final Community Group Report, not a W3C Recommendation. Layer names map to Material 3, Spectrum, Primer and Atlassian vocabulary in the start-here table. Prepared with AI assistance and edited by hand.

START HERE

Three layers, one public API.

The layer names differ between systems; the separation is the same. Map this guide's vocabulary to the one your team already uses before reading the rules.

This guide	Material 3	Spectrum	Primer	Kutschmann (martinfowler.com)
core (<code>core.*</code>)	reference (<code>md.ref</code>)	global	base	option
semantic (no prefix)	system (<code>md.sys</code>)	alias	functional	decision
component (<code>comp.*</code>)	component (<code>md.comp</code>)	component-specific	component and pattern	component

One namespace per restricted layer makes the rules lintable: a single pattern, `--ds-(core|comp)-`, finds everything application code must not touch.

Starting a token system

Sections 01 and 02, then copy the kit's token files and `check-token-rules.mjs`. Decide the semantic grammar before the first alias.

Locking down an existing one

Section 04 first: turn on the ESLint and stylelint gates in warning mode, count the violations, then migrate. Rule R03 comes last.

Opening it to agents

R10, R18 and R21. Agents read `$description`, the generated typed map and the lint errors; the lint, not the prompt, is the control.

Adding a mode or brand

R07 and R08, then add a context to the DTCG resolver. Semantic keys never change between modes; only their aliases do.

Label	Meaning
<code>DTCG SCHEMA</code>	Enforced by the official DTCG 2025.10 JSON Schema (<code>validate-tokens.mjs</code>).
<code>SCRIPT</code>	Enforced by a kit script: <code>check-token-rules.mjs</code> or <code>contrast-check.mjs</code> .
<code>LINT</code>	Enforced by the ESLint or stylelint configuration in application code.
<code>TYPES</code>	Enforced by the TypeScript compiler through the generated token map.
<code>WCAG AA</code>	Traces to a WCAG 2.2 AA success criterion.
<code>PRACTICE</code>	A working method with a review signal rather than a hard gate.

SECTION 01

Core: literal values, kept private.

Core tokens are the palette and the scales. They exist so semantic tokens have something to point at, and for no other reason.

Suggested owners: Design-system lead + token owner

☐ Core tokens hold literal values, never aliases

R01 **SCRIPT**

A core token is the end of every reference chain. If it aliases something, the layering is already broken.

Evidence: `check-token-rules.mjs` fails on any `core.*` token whose `$value` contains a `{reference}`.

Kit script.

☐ Core names describe the value, not the use

R02 **SCRIPT**

`core.color.blue.600`, `core.space.400`. No role words (`text`, `bg`, `border`, `brand`, `danger`) in a core path: meaning belongs one layer up.

Evidence: Role-word denylist over core paths in `check-token-rules.mjs`.

Kutschmann, option tokens. S08.

☐ Core never ships to application code

R03 **SCRIPT** **LINT**

Core tokens are build inputs only: not in shipped CSS, not in TypeScript exports, not in the package `exports` map. What code cannot load, an agent cannot use.

Evidence: `grep -r -- '--ds-core-' dist/` returns nothing; the package `exports` map has no `./core` entry.

Private option tokens can change without breaking changes. S08.

☐ Every token file validates against DTCG 2025.10

R04 **DTCG SCHEMA**

Colors are objects with a color space and components (an optional `hex` fallback); dimensions are `{ value, unit }` with `px` or `rem`. The schema rejects a hex string color and a `"16px"` string dimension.

Evidence: `validate-tokens.mjs` against the official Format and Resolver schemas.

DTCG 2025.10 Format and Color modules. S01, S02.

Nobody picks the wrong blue from a palette they cannot import.

SECTION 02

Semantic: the only tokens code sees.

The semantic layer is the public API of the design system. Name it like one: a closed grammar, one meaning per name, identical keys in every mode.

Suggested owners: Design-system lead + accessibility lead

☐ Every semantic token aliases a core token

R05 **SCRIPT**

No literals and no references to component tokens. The semantic layer is decisions, and a decision points at a value.

Evidence: `check-token-rules.mjs` fails on a semantic literal or a `comp.*` reference.

Kit script.

☐ Semantic names follow a closed grammar

R06 **SCRIPT**

`category.property.variant: color.text.default, color.bg.brand-hover, color.border.focus, space.inline.md, size.control.sm`. A new name that does not fit the grammar is a design review, not a pull request.

Evidence: Grammar regex over every semantic path in `check-token-rules.mjs`.

Atlassian foundation.property.modifier; Curtis's namespace, object, base, modifier. S05, S07.

☐ The semantic key set is identical in every mode

R07 **SCRIPT**

Light, dark and high-contrast differ only in values. A token that exists in one mode and not another breaks the other mode silently.

Evidence: `check-token-rules.mjs` diffs the key sets of every semantic mode file.

Kit script.

☐ Modes never appear in token names

R08 **SCRIPT**

No `color.text.default-dark`. Modes swap the semantic source through DTCG resolver contexts, Figma variable modes or Tokens Studio themes.

Evidence: No path segment matches `light`, `dark`, `hc` or `high-contrast`.

DTCG 2025.10 Resolver module. S03.

☐ Every foreground pairs with its backgrounds, and passes

R09 **SCRIPT** **WCAG AA**

Text reaches 4.5:1 (3:1 only for large text). Focus rings, control boundaries and meaningful icons reach 3:1 against adjacent colors. Ratios are not rounded: 4.499:1 fails.

Evidence: `contrast-check.mjs` over `contrast-pairs.json`: all 13 pairs pass in both modes, 26 of 26 checks.

WCAG 2.2 SC 1.4.3 and SC 1.4.11, AA. S14, S15.

☐ **Every semantic token says when to use it**R10 **SCRIPT**

A `$description` on every semantic token. It is copied into the generated TypeScript map as JSDoc, which is what an agent reads when it autocompletes `token['color']`.

Evidence: `check-token-rules.mjs` fails on a semantic token without `$description`.

Kit script.

Name the decision once, and every theme inherits it.

SECTION 03

Component tokens: only when they earn it.

Component tokens let one component diverge from the semantic default without forking the theme. They cost names and attention, so each one needs a reason.

Suggested owners: Design-system lead + component owner

☐ Component tokens alias semantic tokens, never core

R11 **SCRIPT**

`comp.button.primary.bg` points at `color.bg.brand`, so dark mode works with no component change.

Evidence: `check-token-rules.mjs` fails on `comp.button.primary.bg -> {core.color.blue.600}`.

Kit script.

☐ Create a component token only when it diverges

R12 **PRACTICE**

Two good reasons: the component differs from the semantic default in at least one theme or brand, or several implementations (frameworks, platforms) must stay in lockstep. Otherwise the component consumes semantic tokens directly.

Evidence: Review signal: a component token that aliases the same semantic token in every mode and brand is a deletion candidate.

Nate Baldwin on component-level tokens; Spectrum used them to keep 7+ frameworks consistent. S09.

☐ Component tokens stay inside their component

R13 **LINT**

`--ds-comp-button-*` is allowed only in the Button's own source. Everywhere else, the gate rejects it.

Evidence: stylelint and ESLint overrides that allow `--ds-comp-<name>-` only under `components/<name>/**`.

Primer: component tokens only in component CSS. S06.

A component token is a documented exception, not a second palette.

SECTION 04

Consumption: the gate agents cannot talk past.

These rules apply to application code, whoever writes it. The gate runs the same way for a person, a pull request and an agent's own verify step.

Suggested owners: Engineering lead + design-system lead

☐ Application code imports semantic tokens only

R14 **LINT**

The token package's `exports` map exposes only the semantic entry point and the CSS files, and a restricted-import rule catches deep imports.

Evidence: `@typescript-eslint/no-restricted-imports` with pattern `^@acme/tokens/(core|component)(/|$)`.
Kit ESLint config.

☐ No raw design values in application styles

R15 **LINT**

No hex, no `rgb()`, `hsl()` or `oklch()`, no named colors, no literal spacing or radius. Keywords such as `transparent` and `currentColor` stay allowed.

Evidence: stylelint `color-no-hex`, `color-named`, `function-disallowed-list` and `scale-unlimited/declaration-strict-value`; ESLint `no-restricted-syntax` on string and template literals.

Kit configs; stylelint-declaration-strict-value. S12.

☐ No core or component variables in application code

R16 **LINT**

One pattern covers both restricted layers in CSS values, string literals and template literals.

Evidence: stylelint `declaration-property-value-disallowed-list` with `/var\(\s*--ds-(core|comp)-/`; ESLint `no-restricted-syntax` on the same pattern.

Kit configs.

☐ Property and token role must match

R17 **LINT**

`color` takes `color.text.*`, `background` takes `color.bg.*`, border and outline colors take `color.border.*`. "The colors look the same" misuse breaks in the next theme.

Evidence: stylelint `declaration-property-value-allowed-list` per property family.

Primer's `primer/colors` rule enforces the same idea; Atlassian: choose tokens by meaning, not by matching color. S05, S06.

☐ TypeScript reaches tokens through a generated, closed map

R18 **TYPES**

`token['color.text.default']` with `type SemanticToken = keyof typeof token`. Unknown keys are compile errors. Values are `var(--ds-...)` strings, so inline styles follow the active mode.

Evidence: `tsc --noEmit` reports an unknown key (TS2339 in the kit's violations example).

Kit build: custom Style Dictionary format.

☐ **Utility frameworks expose semantic utilities only**R19 **PRACTICE**

In Tailwind v4, reset the default color namespace and map semantic tokens with `@theme inline`, so `bg-blue-500` does not exist to be guessed.

Evidence: The kit's `tailwind-lockdown.css`. Syntax from the Tailwind v4 theme docs; not executed in the kit's test run.

Tailwind theme namespaces. S13.

☐ **Lint exceptions need a reason, and never cover token rules**R20 **LINT**

An agent should not be able to silence the gate with a comment. Require a description on every disable, forbid disabling the token rules, and run the token job without inline config.

Evidence: `@eslint-community/eslint-comments/require-description` and `no-restricted-disable`; CI runs `eslint --no-inline-config` and `stylelint --ignore-disables`.

eslint-plugin-eslint-comments 4.8.

☐ **The gate runs where agents work**R21 **PRACTICE**

Pre-commit, CI and the agent's own verify step all run the same `lint` and `tokens:*` scripts, and a non-zero exit blocks the merge. Give agents the generated typed map as context; treat the lint, not the prompt, as the control.

Evidence: One `npm run check` used in all three places.

Kit `package.json`.

The lint, not the prompt, is the control.

SECTION 05

Change management: the semantic surface is versioned.

Because core is private, only the semantic layer is a public contract. Version it like one.

Suggested owners: Design-system lead + release owner

☐ Deprecate before removing

R22 **SCRIPT** **TYPES**

Set `$deprecated` to a string naming the replacement and the removal version. The build carries it into `@deprecated` JSDoc, and `@typescript-eslint/no-deprecated` flags every use.

Evidence: The kit's `color.link` token: `"Use color.text.link. Removed in 3.0.0."`, flagged in the violations example.

DTCG `$deprecated`. S01.

☐ Semver follows the semantic surface

R23 **PRACTICE**

Adding a semantic token is a minor release. Changing a semantic value is minor or patch with visual review. Renaming or removing one is major. Core changes are not breaking for consumers, because core is private.

Evidence: CI diffs the exported semantic key set against the last release tag; a removal without a major bump fails.

Recommended practice.

☐ Every rename ships a machine-readable migration map

R24 **PRACTICE**

`{ "color.link": "color.text.link" }`, so a codemod or an agent can apply the rename mechanically.

Evidence: Every key removed since the last release appears in `token-renames.json`.

Kit `token-renames.json`.

☐ Generated files are never hand-edited

R25 **SCRIPT**

CSS and the typed map are build outputs. Editing them hides a change from review and from every other platform.

Evidence: CI regenerates `dist/` and `src/tokens.ts` and fails on a diff.

Kit build.

Public tokens change like an API, because they are one.

APPENDIX A

The three layers in DTCCG 2025.10.

Excerpts from the kit's token files. All five files validate against the official Format and Resolver schemas.

tokens/core/core.tokens.json (excerpt)

```
{
  "core": {
    "color": {
      "$type": "color",
      "blue": {
        "600": { "$value": { "colorSpace": "srgb", "components": [0.0706, 0.3569, 0.8157],
"hex": "#125bd0" } }
      }
    },
    "space": {
      "$type": "dimension",
      "800": { "$value": { "value": 32, "unit": "px" } }
    }
  }
}
```

tokens/semantic/light.tokens.json and dark.tokens.json (excerpts)

```
// light
"border": { "focus": { "$value": "{core.color.blue.600}",
  "$description": "Focus ring. Passes 3:1 against adjacent surfaces (WCAG 1.4.11)." } }
// dark: same key, different alias
"border": { "focus": { "$value": "{core.color.blue.300}",
  "$description": "Focus ring. Passes 3:1 against adjacent surfaces (WCAG 1.4.11)." } }
```

Shown with comments for print; the real files are plain JSON.

tokens/component/button.tokens.json (excerpt)

```
{
  "comp": {
    "button": {
      "primary": {
        "bg": { "$type": "color", "$value": "{color.bg.brand}" },
        "bg-hover": { "$type": "color", "$value": "{color.bg.brand-hover}" },
        "text": { "$type": "color", "$value": "{color.text.on-brand}" }
      }
    }
  }
}
```

tokens/tokens.resolver.json

```
{
  "$schema": "https://www.designtokens.org/schemas/2025.10/resolver.json",
  "version": "2025.10",
  "sets": {
    "core": { "sources": [{ "$ref": "core/core.tokens.json" }] },
    "component": { "sources": [{ "$ref": "component/button.tokens.json" }] }
  },
  "modifiers": {
    "theme": {
      "contexts": {
        "light": [{ "$ref": "semantic/light.tokens.json" }],
        "dark": [{ "$ref": "semantic/dark.tokens.json" }]
      },
      "default": "light"
    }
  },
  "resolutionOrder": [{ "$ref": "#/sets/core" }, { "$ref": "#/modifiers/theme" }, { "$ref": "#/sets/component" }]
}
```

Style Dictionary does not implement the resolver module yet, so the kit's build loops once per mode. Resolver-aware tools such as Terrazzo read this file directly.

APPENDIX B

The gate, as configuration.

Both configurations run in the kit. On the violations examples, ESLint reports 7 errors and stylelint 12; on the clean examples, both report none.

eslint.config.mjs (rules)

```
const HEX = String.raw`/#([0-9a-fA-F]{3,4}|[0-9a-fA-F]{6}|[0-9a-fA-F]{8})\b/`;
const LOWER_TIER_VAR = String.raw`/--ds-(core|comp)-/`;

rules: {
  "@typescript-eslint/no-restricted-imports": ["error", { patterns: [{
    regex: "^@acme/tokens/(core|component)(/|$)",
    message: 'App code consumes semantic tokens only. Import from "@acme/tokens".',
  } ]}],
  "no-restricted-syntax": ["error",
    { selector: `Literal[value=${HEX}]`, message: "Raw hex color. Use a semantic token." },
    { selector: `Literal[value=${LOWER_TIER_VAR}]`, message: "Core/component CSS variable. Use a semantic token." },
    { selector: `TemplateElement[value.raw=${LOWER_TIER_VAR}]`, message: "Core/component CSS variable. Use a semantic token." },
  ],
  "@typescript-eslint/no-deprecated": "error",
}
```

stylelint.config.mjs (rules)

```
plugins: ["stylelint-declaration-strict-value"],
rules: {
  "scale-unlimited/declaration-strict-value": [
    ["/color$/", "fill", "stroke", "background", "box-shadow", "/^padding/", "/^margin/",
    "gap", "/^border-radius/"],
    { ignoreValues: ["transparent", "currentColor", "inherit", "initial", "unset", "none", "0",
    "auto"] },
  ],
  "declaration-property-value-disallowed-list": [{ "/.*"/: ["/var\\(\\s*--ds-(core|comp)-/"]
}],
  "declaration-property-value-allowed-list": [{
    "/^color$/": ["/^var\\(--ds-color-text-/", "inherit", "currentColor"],
    "/^background(-color)?$/": ["/^var\\(--ds-color-bg-/", "transparent", "none"],
  }],
  "color-no-hex": true,
  "color-named": "never",
  "function-disallowed-list": ["rgb", "rgba", "hsl", "hsla", "oklch", "oklab", "lab", "lch",
  "hwb", "color"],
}
```

Exclude generated token CSS from stylelint (it contains hex values by design). Vendor precedents: Atlassian [ensure-design-token-usage](#), Primer [primer/colors](#), Salesforce [no-hardcoded-values-slids2](#).

APPENDIX C

Build output and measured contrast.

Style Dictionary 5.5.5 emits one CSS file per mode and a typed map of `var()` references. The contrast check resolves every declared pair in both modes.

dist/css/semantic.dark.css (excerpt, generated)

```
[data-theme="dark"] {
  --ds-color-text-default: #ffffff; /** Body text and labels on color.bg.surface or
  color.bg.subtle. */
  --ds-color-bg-brand: #80adff; /** Primary actions. One per view region. */
  --ds-color-border-focus: #80adff; /** Focus ring. Passes 3:1 against adjacent surfaces (WCAG
  1.4.11). */
}
```

src/tokens.ts (excerpt, generated)

```
export const token = {
  /** Focus ring. Passes 3:1 against adjacent surfaces (WCAG 1.4.11). */
  'color.border.focus': 'var(--ds-color-border-focus)',
  /** @deprecated Use color.text.link. Removed in 3.0.0. */
  'color.link': 'var(--ds-color-link)',
} as const;
export type SemanticToken = keyof typeof token;
```

Pair	Light	Dark	Minimum	WCAG 2.2
<code>text.default</code> on <code>bg.surface</code>	17.46	19.19	4.5	1.4.3 AA
<code>text.muted</code> on <code>bg.subtle</code>	5.64	6.69	4.5	1.4.3 AA
<code>text.link</code> on <code>bg.surface</code>	6.12	8.54	4.5	1.4.3 AA
<code>text.on-brand</code> on <code>bg.brand</code>	6.12	8.54	4.5	1.4.3 AA
<code>text.on-brand</code> on <code>bg.brand-hover</code>	8.45	11.22	4.5	1.4.3 AA
<code>text.on-danger</code> on <code>bg.danger</code>	5.63	8.41	4.5	1.4.3 AA
<code>border.default</code> on <code>bg.surface</code>	6.27	7.35	3	1.4.11 AA
<code>border.focus</code> on <code>bg.subtle</code>	5.50	7.76	3	1.4.11 AA

8 of the kit's 13 pairs, as printed by `contrast-check.mjs`. WCAG 2.x ratios are defined for sRGB; convert oklch or Display P3 tokens before checking.

KEEP WITH THE RELEASE

Leave a token review record.

A record for one token release. It is the trail that lets the next reviewer see what changed on the public surface and what was proven.

Token package version / release date	Semantic keys added / changed / removed
Migration map updated (link)	Schema validation run (link)
Rules check run (link)	Contrast check, all modes (link)
Lint gate on the product repos (link)	Visual review of changed values (link)
Deprecations and removal versions	Owner / next review

A removal without a deprecation is a breaking change.

If a semantic key disappears without having carried `$deprecated` in the previous minor release, the release is major, whatever the changelog says.

SOURCES / MAINTENANCE

Keep the guide current.

Sources checked 24 September 2026. Every configuration quoted in this guide was run against the kit's fixtures on that date with the tool versions listed on the cover.

S01 / DTCG, Format module 2025.10

Token syntax: \$value, \$type, \$description, \$deprecated, aliases, groups and composite types.

<https://www.designtokens.org/TR/2025.10/format/>

S02 / DTCG, Color module 2025.10

Color values as objects: colorSpace, components, optional alpha and hex.

<https://www.designtokens.org/TR/2025.10/color/>

S03 / DTCG, Resolver module 2025.10

Sets, modifiers with contexts, and resolution order for modes and brands.

<https://www.designtokens.org/TR/2025.10/resolver/>

S04 / W3C DTCG, first stable version announcement

28 October 2025. Final Community Group Report, not a W3C Standard.

<https://www.w3.org/community/design-tokens/2025/10/28/design-tokens-specification-reaches-first-stable-version/>

S05 / Atlassian, design tokens

Naming by foundation, property and modifier; choose by meaning.

<https://atlassian.design/foundations/tokens/design-tokens>

S06 / GitHub Primer, token names

Base, functional and component tokens; component tokens stay in component CSS.

<https://primer.style/product/primitives/token-names/>

S07 / Nathan Curtis, Naming Tokens in Design Systems

Namespace, object, base and modifier (EightShapes, 2020).

<https://medium.com/eightshapes-llc/naming-tokens-in-design-systems-9e86c7444676>

S08 / Andreas Kutschmann, Design Token-Based UI Architecture

Option, decision and component tokens; why option tokens stay private (martinfowler.com, 2024).

<https://martinfowler.com/articles/design-token-based-ui-architecture.html>

S09 / Nate Baldwin, Component-level Design Tokens: are they worth it?

Benefits against name bloat; Spectrum's multi-framework case.

<https://medium.com/@NateBaldwin/component-level-design-tokens-are-they-worth-it-d1ae4c6b19d4>

S10 / Style Dictionary, DTCG support

Version 5 builds DTCG color and dimension objects; resolver support tracked in issue 1590.

<https://styledictionary.com/info/dtcg/>

S11 / Material 3, design tokens

Reference, system and component tokens.

<https://m3.material.io/foundations/design-tokens/overview>

S12 / stylelint-declaration-strict-value

Rule id scale-unlimited/declaration-strict-value.

<https://github.com/AndyOG0/stylelint-declaration-strict-value>

S13 / Tailwind CSS v4, theme variables

Resetting a namespace with initial; @theme inline.

<https://tailwindcss.com/docs/theme>

S14 / W3C, Understanding SC 1.4.3 Contrast (Minimum)

4.5:1 text, 3:1 large text; ratios are not rounded.

<https://www.w3.org/WAI/WCAG22/Understanding/contrast-minimum.html>

S15 / W3C, Understanding SC 1.4.11 Non-text**Contrast**

3:1 for component boundaries, states and focus indicators.

<https://www.w3.org/WAI/WCAG22/Understanding/non-text-contrast.html>

Maintenance: recheck Style Dictionary's DTCG and resolver support at each major release, and move the build to the resolver once it lands. Re-run the kit's `npm run check` after any tool upgrade. Update the PDF, HTML, Markdown and JSON together.