VTX Themes 1: branded themes

Nov 15, 2026 · VTX 1 · extension · draft d01 · MIT licence · vtx.foundation/spec/v1/theme-1/

§1 Status and scope

Themes 1 lets a server brand what the client draws: colours for light and dark mode, the terminal palette, typefaces, base text size, corner radius, density, icons and a logo. A theme is a cached, content-addressed resource that applies to the whole scene or to one region, so one console can host several branded applications side by side. This is draft d01 of an extension of VTX 1, in conformance profile 2, built on the core at draft d03.

- **What a theme never touches.** The trusted zone, the user's accessibility settings, and what content says or whether it is visible: a theme restyles, it cannot hide.
- **Custom components** are a separate extension, Composites 1.

§2 Capability, resource kind, and where a theme applies

A client that accepts `theme: true` takes theme resources and the `theme` prop.

- **Resource kind `theme/vtx`.** One theme document (Section 3) in deterministic CBOR (RFC 8949, Section 4.2.1), so the same theme always has the same hash. The client fetches it once and keeps it across sessions, in the cache partitioned per server (v1 Section 10).
- **The `theme` prop.** On `root`, set with `patch`, and on `region` nodes; its value is the theme's resource hash. It applies to the node and everything inside it, and the nearest themed ancestor wins. The root's theme also brands the client's window: title bar, logo and background.
- **Partial themes.** A field a theme leaves out comes from the next themed ancestor, then from the client's defaults. A region theme can change only the accent colour.
- **Switching.** Until a theme has arrived, the client keeps the styling it had. It then restyles the whole subtree in one frame.
- **Without the capability.** The server sends no theme; a client that meets a `theme` prop anyway ignores it, as it ignores any unknown prop.

§3 Theme content

Every field is optional, so a theme states only what it changes.

Field	Type	Meaning
`name`	text	Display name; informative only
`colors.light`, `colors.dark`	palette (below)	Colours per mode; at least one when `colors` is present
`fonts.ui`, `fonts.mono`, `fonts.heading`	family name or font resource	Interface text; grids and code; headings
`text.base`	10–24	Base text size in px at scale 1

Field	Type	Meaning
`radius`	0–24	Corner radius of components, px
`density`	`compact`, `normal`, `comfortable`	Spacing inside and between components
`icons`	map of icon name to glyph resource	Replaces core icons by name, up to 1,024
`logo`	`{ res, alt }`	An image or glyph resource for the client's title area, with its text alternative

Palette. All colours are `#rrggbb`.

Key	Used for
`fg`, `bg`, `muted`, `accent`, `info`, `success`, `warning`, `critical`	The v1 colour roles, wherever they appear: components, grids, vector surfaces
`panel`	Raised areas: cards, groups, overlays
`border`	Dividers and outlines
`on_accent`	Text and icons on accent-coloured controls
`focus`	Focus rings
`ansi`	16 colours for `p0`–`p15`, the terminal palette; `p16`–`p255` stay fixed

The client uses the variant that matches DISPLAY's `dark`. When the theme lacks that variant, the client uses its own palette for that mode rather than inverting the other one.

```
{"name":"Helper agent","colors":{"light":
{"accent":"#6D28D9","on_accent":"#FFFFFF","focus":"#6D28D9"},"dark":
{"accent":"#C4B5FD","on_accent":"#1E1033","focus":"#C4B5FD"}},"radius":16}
```

§4 Precedence and accessibility

The user's accessibility settings come first, then the theme, then the client's defaults.

- **User settings win.** The user's text size scales `text.base`. `reduced_motion` holds whatever the theme says. In high-contrast or forced-colour mode, the client may set the theme's colours aside entirely.
- **Contrast floor.** `fg`, `muted` and the four status roles need at least 4.5:1 against `bg`, `on_accent` 4.5:1 against `accent`, and `focus` 3:1 against `bg` (WCAG 2.2, criteria 1.4.3 and 1.4.11). The client replaces any colour below its floor with its own default for that role. A server SHOULD check before sending; the definitions include the check.
- **Fonts.** A client may refuse a font that fails sanitising or lacks glyphs it needs, and falls back to its default for that slot.
- **Never themed.** The trusted zone keeps the client's own look. CLAIM's colour stays separate and labelled as claimed (v1 Section 12). The logo appears only in the application's title area, never in or beside the zone.

§5 Terminals and surfaces under a theme

A theme reaches every place a colour role or the terminal palette is used.

- **Grids.** Role colours and `p0`–`p15` come from the nearest theme; `fonts.mono` applies unless the grid sets `font`. A theme change recolours existing cells that use roles or `p0`–`p15`; cells in `#rrggbb` or `p16`–`p255` keep their colour.
- **The VT engine** answers OSC 4, 10 and 11 colour queries from the theme in effect for the grid when there is one, and from DISPLAY otherwise. DISPLAY keeps reporting the client's own, unthemed colours.
- **Vector surfaces** (Surfaces 1) paint with roles, so they follow the theme. Image surfaces are left as they are.

§6 Text form and application interface

Applications declare themes by name and never handle hashes; the server does the encoding.

- **Declaring.** An application writes `{"kind": "theme", "name": …, "theme": {…}}`. Names are lower case, up to 32 characters, and belong to the application's session. The server encodes the theme, hashes it, offers it to the client, and replaces the name with the hash in every later op.
- **Redeclaring** a name points later ops at the new theme. Nodes already themed keep their theme until the application patches them.
- **On the wire**, `theme` is always a resource hash.
- **Deterministic encoding.** Map keys are sorted by their encoded bytes and integers take their shortest form (RFC 8949, Section 4.2.1); themes contain no floats. Two test vectors, Fjord and Helper, are in `ext/theme-1/examples/vectors.json`.

```
{"kind":"theme","name":"helper","theme":{"name":"Helper agent","colors":{"light":
{"accent":"#6D28D9","on_accent":"#FFFFFF","focus":"#6D28D9"}},"radius":16}}
{"op":"patch","id":"root","set":{"theme":"fjord","title":"Fjord operations"}}
{"op":"open","id":"agent","type":"region","title":"Support agent","theme":"helper"}
```

§7 Security and privacy

A theme is styling data: it cannot fetch, run code or reach the trusted zone.

Threat	Mitigation
Network access	Fonts, icons and the logo arrive only as resources the server offers
Hostile font files	Fonts are sanitised as in v1 Section 10
Hiding content with colour or size	The contrast floor (Section 4) and the bounded `text.base`; a theme has no per-node visibility or size
Tracking across servers	Themes are cached per server identity, like every resource
Impersonating another service with a logo	The trusted zone always shows the verified server identity, and the logo never appears beside it

§8 Wire changes and definitions

Themes need no new frames, ops or error codes.

Where	Change
Capabilities	`theme`
Resource kinds	`theme/vtx`
Scene props	`theme` on `root` and `region`
Application interface	The `theme` control line

The bundle's `ext/theme-1/` holds:

File	Defines
`vtx-theme-1.cddl`	`theme-doc` and the new resource kind; load after `vtx-wire-1.cddl`
`vtx-theme-1.schema.json`	One theme, for authors and tools
`vtx-ops-theme-1.schema.json`	Every v1 op, with `theme` on regions
`vtx-app-theme-1.schema.json`	The application interface with `theme` lines
`examples/`	The Fjord and Helper themes, an application stream and a wire stream, 14 lines that must be rejected (12 by the schemas, 2 for naming an undeclared theme), and a theme that passes the schema but fails the contrast floor
`examples/vectors.json`	Each theme's deterministic CBOR and hash

`tools/validate.py` runs 44 checks for this extension, including the contrast floor and the deterministic encoding, and every check in the bundle passes.