VTX Surfaces 1: vector and image surfaces

Nov 15, 2026 · VTX 1 · extension · draft d02 · MIT licence · vtx.foundation/spec/v1/surface-1/

§1 Status and scope

Surfaces 1 adds drawn content to VTX: vector surfaces, a retained 2D scene the client draws and animates itself, and image surfaces, raster images sent as content-addressed resources. It also lets grids show images, so the VT engine can carry Kitty graphics, sixel and iTerm2 inline images. This is draft d02 of an extension of VTX 1, in conformance profile 2, built on the core at draft d03; it is negotiated, so a peer without it never sees it. Draft d02 adds sub-cell positioning of glyphs in grids (`cells.sub`, Section 7). Terms and key words are as in the core specification.

- **No new channels.** Surfaces are scene nodes on channel 1 and images are resources on channel 3, so they get v1's atomic updates, resume, retention and fallback without extra machinery.
- **Later work.** Pixel surfaces come with Media 1, which carries video; window surfaces and WASM drawing are Windows 1 and WASM 1.
- **Definitions.** `ext/surface-1/` in the definitions bundle holds the schemas, the CDDL additions and examples; Section 10 lists them.

§2 Capabilities, limits and resource kinds

A server sends surface nodes and ops only after the client accepted the matching capability.

Capability	Value	Meaning
`surface.vector`	`true`	Vector surfaces (Section 5)
`surface.image`	array of media types	Image surfaces and image items; `image/png` is required in the list
`surface.grid`	`true`	Images placed in grids (Section 7); needs `surface.image`
`cells.sub`	`true`	Sub-cell positioning of glyphs in grids (Section 7)

Limit (HELLO, CAPS)	Default	Exceeded by	Result
`image_px`	16,777,216 (4096 × 4096)	An image whose width × height is larger	Client answers DROP; an op that uses it reports scene ERROR `resource`
`vector_items`	10,000	A `draw` that would put more items in one surface, group members included	Op skipped, scene ERROR `limit`

Resource kinds. `image/png`, `image/jpeg`, `image/webp`, `image/avif` and `image/gif` join the v1 kinds. An image OFFER carries `meta: { w, h, ? frames }`, so the client can refuse an image before downloading it; an image whose decoded size differs from its `meta` is unavailable. With `frames` above 1 the image is animated, and a client with `reduced_motion` shows its first frame.

§3 The surface node

A surface is a scene node of type `surface`, a sibling of `grid`: it may sit at the root, in a region or in any layout component. Its `kind` is fixed when it opens; reopening it with another kind is `invalid_op`. Content never draws outside the surface's box.

Prop	Type	Meaning
`kind`	`vector` or `image`	Fixed at open
`alt`	text, required	Text alternative: the accessible name and the fallback
`size`	[w, h]	Intrinsic size in device-independent pixels; layout scales the box like an image
`fit`	`contain` (default), `cover`, `fill`, `none`	How content maps into the box
`view`	[x, y, w, h]	Vector only: the coordinate box; default `[0, 0, w, h]` from `size`
`res`	resource hash, required for images	Image only: the image to show
`smoothing`	`auto` (default), `pixelated`	Image only: how scaling interpolates
`mouse`	`none` (default), `click`, `drag`, `move`	Which pointer events reach the server, as for grids
`pointer`	`normal` (default), `lock`	Pointer lock, with the `pointer.lock` permission (v1 Section 11)
`zoom`	`{ min, max }`	Lets the user pan and zoom locally between these factors

The common props of v1 apply too: `label`, `description`, `hidden`, `gestures`, `autofocus` and the rest.

§4 Image surfaces

An image surface shows the image resource named by `res`, offered and fetched as in v1 Section 10.

- **Formats.** The server sends only formats the client listed in `surface.image`. PNG is always available; JPEG, WebP, AVIF and GIF are optional.
- **Updates.** A `patch` of `res` swaps the image. The client keeps showing the old image until the new one has arrived and decoded, so the surface never flashes empty.
- **Loading.** Until the first image arrives, the client shows a placeholder of the surface's `size` with its `alt` available to assistive technology.
- **Colour.** Images are sRGB. A client may apply an embedded ICC profile. HDR is out of scope for this draft.
- **Animation.** Animated images loop as the file specifies; with `reduced_motion` the client shows the first frame.

Image surfaces suit photos, screenshots and thumbnails. For charts and diagrams, a vector surface or the `chart` component stays sharp at every scale and follows the theme.

§5 Vector surfaces

A vector surface is a retained 2D scene: the server describes shapes once, and the client draws, scales, hit-tests and animates them without further traffic.

Coordinates. Items use the units of `view`, with the origin top-left and y pointing down. The client maps `view` into the box according to `fit`; strokes and text scale with the content.

Layers and ops. A vector surface holds named layers, drawn in order, the first at the bottom. Item ids are unique within the surface, across all layers and group levels.

Op	Fields	Effect
`draw`	`id`, `layer`, `items`, `before`	Creates or replaces one layer. A new layer goes on top, or under the layer named in `before`. Without `items`, removes the layer
`edit`	`id`, `item`, `set`, `unset`	Sets or removes props of one item, at any depth

Items. Each item is an object whose `t` names its shape.

`t`	Fields
`rect`	`x`, `y`, `w`, `h`, `rx` (corner radius)
`ellipse`	`cx`, `cy`, `rx`, `ry`
`line`	`x1`, `y1`, `x2`, `y2`
`poly`	`points` [x, y, x, y, ...], `closed`
`path`	`d`: the commands `M`, `L`, `Q`, `C` and `Z` of v1 glyph paths, at most 65,536
`text`	`x`, `y`, `text`, `size`, `font`, `weight` (100–900), `italic`, `anchor` (`start`, `middle`, `end`), `baseline` (`top`, `middle`, `alphabetic`, `bottom`), `max_w`
`image`	`x`, `y`, `w`, `h`, `res`, `smoothing`
`group`	`items`, `clip` [x, y, w, h]; groups nest at most 16 deep

Every item also takes `id`, `label`, `hit`, `intent`, `context`, `link`, `hidden`, `opacity` (0–1), `transform` ([a, b, c, d, e, f], as in SVG) and `anim`.

An `image` item needs `surface.image`; a client without it leaves the item empty but still exposes its `label`.

Paint. `fill` and `stroke` take a v1 colour (`#rrggbb`, `p0`–`p255` or a semantic role), a gradient or `none`, so a surface follows the theme and dark mode like every other node. A gradient is `{ linear: [x1, y1, x2, y2] }` or `{ radial: [cx, cy, r] }` with `stops`: [[offset, colour], ...]. Strokes take `stroke_w` (default 1), `dash`, `cap` and `join`; fills take `fill_rule`. Shapes and text fill with `fg` and draw no stroke unless told otherwise; `line` strokes with `fg`.

Text. Text items are real text: assistive technology reads them, search finds them, copying yields them. `font` names a family or a font resource (v1 Section 10), defaulting to the client's interface font. `size` is in view units. With `max_w`, text that does not fit ends in an ellipsis.

Animation. `anim` moves an item's `transform` or `opacity` from its current value to `to` over `dur` milliseconds, with `ease` (`linear`, `in`, `out`, `in-out`), `repeat` (a count or `forever`), `alternate` and `delay`. The client runs it; nothing travels per frame. An `edit` that sets the animated prop, or unsets `anim`, stops it. With `reduced_motion` the client shows the end state at once, or the start state for `forever`.

```
{"op":"open","id":"floor","type":"surface","kind":"vector","parent":"site","alt":"Floor
plan of data hall 2 with the status of each rack","size":[640,400],"view":
[0,0,64,40],"mouse":"click","zoom":{"min":1,"max":8}}
{"op":"draw","id":"floor","layer":"racks","items":[{"t":"group","id":"row-
a","transform":[1,0,0,1,4,8],"label":"Row A","items":
[{"t":"rect","id":"a02","x":4,"y":0,"w":3,"h":6,"rx":0.3,"fill":"critical","hit":"rack","label":"Rack
A02, power alarm","intent":"open-rack","context":{"rack":"A02"},"anim":
{"prop":"opacity","to":0.4,"dur":800,"ease":"in-
out","repeat":"forever","alternate":true}}]}]}
{"op":"edit","id":"floor","item":"a02","set":{"fill":"warning","label":"Rack A02, on
battery"},"unset":["anim"]}
```

§6 Interaction

Surfaces offer three levels of interaction: intents on items for ordinary controls, hit names on pointer events for custom handling, and local zoom that needs no server at all.

- **Items with intents.** An item with `intent` is a control. Clicking or tapping it, or pressing Enter or Space while it has focus, sends INTENT with `source` set to the surface, `item` to the item's id and the item's `context`. Such an item needs an `id` and a `label`. These items take part in Tab order, in layer order and then item order, and the client draws the focus ring around their bounds.
- **Pointer events.** With `mouse` set, POINTER events over a surface also carry `sx` and `sy`, the position in view units, and `hit`, the hit name of the topmost visible item under the pointer. Gestures (v1 Section 11) carry the same fields.
- **Local zoom.** With `zoom`, the user pans and zooms between `min` and `max` with pinch, wheel or keyboard, and the server is not told. `sx` and `sy` stay in view units whatever the zoom.
- **Links.** An item with `link` opens it through the trusted zone, as `open_url` (v1 Section 12); only `https`, `http` and `mailto` are allowed.
- **Pointer lock.** `pointer: "lock"` works as on grids, for games and remote control.

§7 Grids: images and sub-cell positioning

With `surface.grid`, a grid can show images anchored to its cells, which is what terminal graphics need.

Op	Fields	Effect
`place`	`id`, `pid`, `res`, `row`, `col`, `rows`, `cols`, `z`, `fit`, `alt`	Shows an image over a block of cells; `z` is 1 (above the text, default) or −1 (below it); `fit` defaults to `contain`. Placing an existing `pid` replaces it
`unplace`	`id`, `pid`	Removes a placement

Placements move with their cells: `scroll` moves them with the rows, rows that leave the top take them into scrollback, or drop them when the grid keeps no scrollback, and `clear` removes them all. Writing cells under a placement does not remove it.

The VT engine maps the terminal image protocols onto placements when the client accepted `surface.grid`:

Program output	VTX
Kitty graphics (APC `_G`): transmit (`a=t`, `a=T`)	An image resource: PNG data (`f=100`) as sent; raw RGB and RGBA re-encoded as PNG
Kitty graphics: display (`a=p`, `a=T`) and delete (`a=d`)	`place` and `unplace`; queries (`a=q`) are answered OK
Sixel (DCS q)	Decoded, re-encoded as PNG and placed at the cursor, sized in cells from DISPLAY's `cell_px`; the cursor moves as in xterm
iTerm2 inline images (OSC 1337 `File=…;inline=1`)	An image resource and a `place` at the cursor
DA1 and XTSMGRAPHICS queries	Sixel support reported, with `image_px` as the size limit

Without `surface.grid`, the engine drops image data as in v1 and reports no image support. Kitty's Unicode placeholder mode is not mapped in this draft; placeholder cells show as ordinary text.

Sub-cell positioning. With `cells.sub`, a grid style may carry `sub`, which draws each glyph of a run smaller or shifted inside its cell box, for crisp gauges, sparklines and box drawing. The members count in eighths of a cell:

Member	Range	Effect
`sx`, `sy`	1–8 (default 8)	Width and height of the glyph, anchored at the cell's bottom-left corner
`dx`, `dy`	−7 to 7 (default 0)	Moves the scaled glyph right and up

- The glyph is clipped to its cell box, or to both cells of a wide glyph; it never draws into a neighbour.
- Only the glyph moves. The cell's text, background, underline and strike-through stay put, so copying, search and screen readers see the plain text.
- Custom glyph slots (`g`) follow `sub` like any glyph.
- `sub` lives in a style, so a grid's styles can name each gauge step once: `{"op":"style","id":"term","sid":"bar5","style":{"fg":"accent","sub":{"sy":5}}}`.
- A client without `cells.sub` ignores `sub`, as v1 receivers ignore unknown keys, and draws the plain glyph. For exact vertical gauges on such clients, the eighth blocks U+2581–U+2588 still work. The VT engine never produces `sub`, since no terminal sequence carries it.

§8 Accessibility and fallback rendering

Every surface must make sense without its pixels.

- **Text alternative.** `alt` is required. An image surface is exposed as an image named by `alt`; a vector surface as a figure named by `alt`, or by `label` when both are set, with `alt` as its description.

- **Items.** Items with a `label` are exposed as children of the figure, in layer order and then item order: items with `intent` as buttons, items with `link` as links, `text` items as text, others as graphics. Items without a `label` are decorative and hidden from assistive technology, apart from `text` items.
- **Preferences.** `reduced_motion` stops animation (Section 5). In high-contrast mode the client may raise stroke widths and map colour roles to its high-contrast palette, which semantic roles make possible.
- **Grid images.** A placement's `alt` is read in place of the image.

Fallback. A server sends surfaces only to clients that accepted the capability; when a CAPS change drops it, the server replaces them as v1 Section 5 describes. Fallback rendering (v1 Section 18) shows a surface as one line, `[Image: alt]` or `[Diagram: alt]`, followed by its items with intents as numbered menu entries, so the actions stay usable on a plain terminal.

§9 Security

Surfaces keep v1's rules: no implicit network access, no code on the client, and nothing that can reach the trusted zone.

Threat	Mitigation
Content fetching from the network	Images arrive only as resources the server offers; the only URL a surface can carry is an item's `link`, opened through the trusted zone
A surface drawing over other content or the trusted zone	Content is clipped to the surface's box; the zone lies outside the scene area
Decompression bombs and decoder exploits	`image_px` is checked against OFFER `meta` and again against the image header before decoding; clients decode in memory-safe or sandboxed code and treat corrupt images as unavailable
Scene exhaustion	`vector_items`, 16 levels of groups, 65,536 commands per path, 16 dash entries
Animation as a resource drain	Animations run at most at the display rate; clients may pause animations of hidden or off-screen surfaces
Hidden escape sequences	Text in items follows v1's control-character rule (v1 Section 7)
Location and camera details in photos	Servers SHOULD strip metadata such as EXIF from images they relay for others

§10 Wire and text-form changes

Everything is additive: new capabilities, limits, resource kinds, a node type, four ops and optional fields on three existing frames.

Where	Change
Capabilities	`surface.vector`, `surface.image`, `surface.grid`, `cells.sub`
HELLO and CAPS limits	`image_px`, `vector_items`
Resource kinds	`image/png`, `image/jpeg`, `image/webp`, `image/avif`, `image/gif`; OFFER `meta { w, h, ? frames }`

Where	Change
Scene node	`surface`, with `kind` `vector` or `image`
Scene ops	`draw`, `edit`, `place`, `unplace`
Grid styles	`sub`: `sx`, `sy`, `dx`, `dy` in eighths of a cell (draft d02)
POINTER (0x12)	`sx`, `sy`, `hit`
GESTURE (0x18)	`sx`, `sy`, `hit`
INTENT (0x16)	`item`
Application interface	`resource` lines accept the image formats
Scene ERROR codes	None new: `resource` for an image that cannot be shown, `limit` for `vector_items` and group depth

Definitions. The v1 CDDL now marks its extension points as standard CDDL sockets (`$$limit-ext`, `$$in-pointer-ext`, `$$in-gesture-ext`, `$$in-intent-ext`, `$res-kind-ext`, `$permission-ext`), and this extension fills them without changing a v1 rule. The bundle's `ext/surface-1/` holds:

File	Defines
`vtx-surface-1.cddl`	The socket additions above; load after `vtx-wire-1.cddl`
`vtx-ops-surface-1.schema.json`	Every v1 op plus surfaces and the four new ops, for producers and constrained decoding
`vtx-app-surface-1.schema.json`	The application interface with the same ops and image resources
`examples/`	A floor plan with live rack status, an image surface, grid placements, 12 lines the schema must reject and 11 that break the scene rules

`tools/validate.py` runs 94 checks for this extension, among them that the v1 schema and the v1 CDDL still reject surface content, and every check in the bundle passes.