VTX RDP Bridge 1: Windows hosts through VTX

Nov 15, 2026 · VTX 1 · host binding · MIT licence · vtx.foundation/spec/v1/rdp-bridge-1/

§1 Status and scope

RDP Bridge 1 is a gateway binding of VTX 1, draft d01. Like Wayland Host 1 it is not an extension: it adds no capability, frame or channel. It fixes how a gateway connects VTX clients to hosts that speak RDP, such as Windows session hosts, Windows desktops and RDS farms, so that an organization moving to VTX keeps its Windows applications on every VTX client from the first day.

The bridge is two things at once. Toward the device it is a VTX server, with a full desktop or seamless windows, input, the clipboard and every redirection extension. Toward the host it is an RDP client, with Network Level Authentication, the graphics pipeline, RemoteApp and the virtual channels. Each RDP channel meets its VTX counterpart in the bridge, and nothing of RDP reaches the device.

On the device	Through the bridge	On the Windows host
A full desktop on the displays the user picked	Displays 1 and Media 1 streams	A desktop session with one monitor per display
Applications as the device's own windows	Windows 1	RemoteApp programs
Folders, printers, microphone, camera, smart cards, USB devices	Drives 1, Printers 1, Media Devices 1, Smart Cards 1, USB 1	Redirected drives, printers, audio, cameras, smart cards and USB devices
One sign-in to VTX	A federated certificate, the user's own smart card, or a password typed into the bridge	Network Level Authentication

The parts of the binding that a log can show are also written as an executable reference mapping, and the binding's example logs must match it exactly, on both sides (Section 9).

Where it runs. The bridge runs next to the hosts it serves: in the same network as an RDS farm, or as part of a VTX deployment that also runs Linux session hosts. It can serve the farm's collections as Brokering 1 items, so the user finds Windows and Linux applications in one list.

Out of scope. The other direction, RDP clients reaching VTX hosts, is not this binding's subject. A VNC bridge, which the outline also names, is simpler and can follow the same pattern.

§2 At a glance

> **Diagram: the bridge · seven pairs of VTX and RDP channels.** Drawn in the PDF version, `02-rdp-bridge-1.pdf`.

On each row, the left side is what the device sees and the right side what the Windows host sees; the bridge translates between them. The device only ever speaks VTX, so its trusted zone, permissions and extensions apply to a Windows session exactly as to a Linux one, and the host only ever speaks RDP, so it needs nothing installed.

§3 Connecting and signing in

Which host. The bridge's configuration names the hosts and RemoteApp programs it serves, or reads them from an RDS deployment's RD Web feed. It can publish them as Brokering 1 items: LAUNCH of a desktop or program connects to the host, or to the RD Connection Broker with the collection's load-balancing information, and the bridge follows the broker's redirection to the session host it picks. A user's disconnected Windows sessions appear in RUNNING, so they can be reconnected from any device. Hosts behind an RD Gateway are reached through it.

What the connection carries. The bridge connects when the client's first LAYOUT arrives, so the session starts at the right size on the right displays. It sends the device's keyboard layout as a Windows layout id (the device's `us` as 00000409, `de` as 00000407), the time zone from HELLO's `tz`, and the language from its `locale`, and for RemoteApp the program to start.

The host's identity. The bridge checks the host's TLS certificate against its configuration: an enterprise CA, or keys pinned per host. It never trusts a host on first use, and Network Level Authentication binds the credentials to that checked key.

Signing in to Windows. VTX has no passwords: the user signs in to the bridge with OIDC, WebAuthn, a certificate or a Brokering 1 ticket. Windows, though, needs credentials for an interactive sign-in. The bridge supports three ways, in this order of preference:

Way	How	What the bridge sees
Federated certificate	After the VTX sign-in, an enterprise CA the domain trusts for smart card sign-in issues the user a certificate valid for minutes. The bridge signs in through NLA with it as a smart card sign-in, its key never leaving the bridge's key store. This is how Citrix FAS and AD FS single sign-on for RDS work	No password and no PIN of the user's; a short-lived certificate in its own key store
The user's smart card	NLA with the user's card through Smart Cards 1	The PIN: Windows sends it to the host in both NLA and the sign-in screen, so it passes the bridge. Readers whose PINs are typed on the device only cannot be used for this
A password	The bridge shows its own sign-in form in the scene, as any VTX application can; the trusted zone shows the bridge's verified name, so the user knows who asks. The bridge uses the password for NLA once and forgets it	The password, once

An organization can switch off the second and third ways, and should switch off the third once federated certificates work.

Reconnecting. When the VTX client resumes a session, the bridge keeps the RDP connection, or reconnects it with Windows' auto-reconnect cookie if it dropped. A session the user leaves stays connected for as long as VTX keeps it resumable; after that the bridge disconnects it, and Windows keeps it running for the next sign-in.

§4 Displays, graphics and windows

Monitors from displays. Windows lays out monitors in device pixels, VTX in px with a scale per display, so the bridge converts. Each display becomes a monitor of its size times its scale; the primary display is at 0, 0; and every other display keeps touching the display it touches in the VTX layout, its offset along that edge measured in the neighbour's pixels. A laptop at 200% (1512 × 982 px) with a 1440p display at 100% to its right, 229 px higher, becomes a 3024 × 1964 monitor at 0, 0 and a 2560 × 1440 monitor at 3024, −458. Each monitor carries its scale as Windows' desktop scale factor (the display's scale in percent) and the nearest device scale factor (100, 140 or 180). The first LAYOUT goes into the connection; every later one becomes a display control monitor layout, so plugging a display in or out, or changing its scale, takes effect without reconnecting. In a full desktop, only the displays in `use` become monitors.

Pointer positions go the other way: a POINTER at x, y on a surface becomes the host's desktop pixel at the monitor's corner plus x, y times the display's scale, from wherever the window is on the device at that moment.

Graphics. The bridge takes the graphics pipeline (MS-RDPEGFX) with H.264 (AVC420 and AVC444) as its first choice. In a full desktop, the host's surface for each monitor becomes that display's stream; in a RemoteApp session, each window's surface becomes the window's stream. Where one host surface fills one stream in AVC420, the bridge forwards the H.264 frames as they are, without decoding them; otherwise it decodes, composes and encodes again in the client's codec. It acknowledges frames only as fast as FEEDBACK lets it send them, so Windows never draws ahead of the network, and it suppresses output for displays and windows the client reports as not visible.

Pointers. The client draws the pointer: Windows' hidden and default system pointers become the surface's `cursor` none and default, and every other pointer an image resource with its hot spot.

RemoteApp windows. In a RemoteApp session, every window Windows reports through its window orders (MS-RDPERP) becomes a Windows 1 window:

RemoteApp	Windows 1
A new window with a title and position	A window with `title`, and `at`, `size` and `display` from its position in desktop pixels, and its pixel surface
The application id Windows reports	`app`
Windows draws the window's frame itself	`frame` none, so the device does not add a second one
An owned window with a caption; modal	`role` dialog, `owner`; `modal`
A menu or a tooltip	`role` popup or tooltip, `owner`, `offset` from the owner
A new title, position, size or show state	A patch of what changed, after Windows has changed it
A deleted window	`remove`
A drag on Windows' own title bar or border (local move and size)	The `window` op with `begin_move` or `begin_resize`, while the button is held

The other direction is the device's window manager at work: WINDOW `focus` and `blur` become RemoteApp's activate, `move` and `resize` its window move in desktop pixels, and `maximize`, `minimize`, `restore` and `close` its system commands. In every case the bridge tells Windows and patches the scene only once Windows reports the change, so the scene never shows a state the host does not have.

Focus. Windows can bring a window to the foreground on its own. The bridge sends `focus` only when the new foreground window is owned by the window active on the device, such as the dialog the user just opened from it; any other foreground change becomes `attention`. An Outlook reminder cannot take the keyboard while the user types in another program on the device.

§5 Input

VTX input	RDP
KEY `code`	A fast-path keyboard event with the key's scancode (set 1) and extended flag
KEY `repeat`	Another key-down, as every RDP client sends while a key is held
KEY whose `code` has no scancode, such as from an on-screen keyboard	Unicode keyboard events for its `text`
COMPOSE `commit`	Unicode keyboard events, a press and a release per UTF-16 unit
The Caps Lock and Num Lock bits of `mods`	A synchronize event when the session gains focus, so the lock keys agree
POINTER, mouse	Fast-path mouse events at the host's desktop pixel (Section 4); left, middle and right as mouse buttons, back and forward as extended buttons
WHEEL	Wheel and horizontal wheel events, 120 a line; `px` as the matching fraction of a line, so fine scrolling stays smooth
POINTER `dx`, `dy` under pointer lock	Relative mouse events, where the host takes them; otherwise absolute positions
POINTER, touch and pen	Multitouch and pen input (MS-RDPEI)

Keys by position. KEY's `code` names the physical key, and so does a scancode, so the bridge never needs to know what a key types. Windows applies the keyboard layout the connection named, the device's own, so the key labelled Y on a German keyboard (`KeyZ`, scancode 2C) types y, and AltGr with Q types @, exactly as on a local Windows machine with that keyboard.

Composed text. The device's input method composes Chinese, Japanese or Korean, accents by dead keys on phones, and dictation, and only the result reaches Windows, as Unicode. Nothing of the composition is sent, so Windows' own input method stays out of the way.

Shortcuts. The Windows key, Alt+Tab and the like stay with the device unless the user granted `keyboard.lock` for a surface with `keys` all, as for every pixel surface. Ctrl+Alt+Del never leaves the device, which keeps it for itself; Ctrl+Alt+End reaches Windows as keys and opens the same screen, as in every RDP client.

§6 Virtual channels

Every RDP virtual channel the bridge carries meets the VTX extension that does the same job on the device, so the user's protections there hold for Windows hosts too:

RDP channel	VTX	What the bridge does
Clipboard, text and images (MS-RDPECLIP)	v1's clipboard channel	Format lists and data both ways, under `clipboard.read` and `clipboard.write`
Clipboard, files	Transfer 1's FETCH	Windows' file lists and contents as a set of files
File system devices (MS-RDPEFS)	Drives 1	Announces each share as a drive, `\\tsclient\` and its name, and turns Windows' create, read, write, query and directory requests into Drives 1 requests. A read-only share stays read-only
Printers and Easy Print (MS-RDPEPC, MS-RDPEXPS)	Printers 1	Announces the device's printers with the Easy Print driver, and turns each XPS job into PDF
Smart cards (MS-RDPESC)	Smart Cards 1	SCard calls map one to one: connect, transmit, transactions, status. Payment applications stay refused on the device
Audio output (MS-RDPEA)	A Media 1 play stream	Encodes Windows' audio as Opus, with the delay RDP reports
Audio input (MS-RDPEAI)	A Media 1 microphone capture, bound with Media Devices 1	Opens the capture when Windows opens the microphone
Camera (MS-RDPECAM)	Media 1 camera captures, listed and bound with Media Devices 1	Lists the device's cameras to Windows and forwards their H.264 frames
USB (MS-RDPEUSB)	USB 1	URBs one to one; the class rules of USB 1 decide on the device which devices Windows can get

Only what the user shared. The bridge announces to Windows only the folders, printers, readers, cameras and USB devices the user shared through VTX. It never redirects anything on its own, and Windows' plug and play redirection reaches only what VTX already plugged in.

Not carried in this draft. Serial and parallel port redirection (USB serial converters go through USB 1), notification area icons, session shadowing (Viewers 1 shares VTX sessions instead) and calls offloaded to the client, such as Teams' audio and video redirection: calls run in the session with Media Devices 1's devices.

§7 What the bridge refuses, and what it cannot carry

Nothing of the bridge's own. An RDP client can redirect the machine it runs on: its disks, printers, smart card readers, clipboard and sound. The bridge never does. Everything Windows sees of a client comes from the user's device, through VTX, after the user shared it.

No credentials kept. A password is used once for NLA and forgotten; a federated certificate lives minutes; the bridge stores no credential of the user's beyond the session.

What RDP cannot give the device. Some of what VTX offers a Linux host has no counterpart in RDP, so a bridged session lacks it:

VTX offers	In a bridged session
Full screen for a single seamless window	Not available for RemoteApp windows; a full desktop is full screen
Progress on the taskbar entry	Not carried: RemoteApp has no progress
Accessibility of pixel surfaces (Sensors 1)	Not available: RDP does not carry Windows' accessibility tree; a later draft can add it with a helper in the session
A caret position for the input method	Not available, as for Wayland hosts

What the bridge adds. Because the device speaks VTX, a Windows session gains what VTX gives every session: several viewers with roles and sealed recordings (Viewers 1), sign-in with WebAuthn or a VTX device key, approvals in the trusted zone, and one application list with Linux hosts (Brokering 1).

The cost. The bridge is an extra hop and, where it cannot forward H.264 as it is, a second encoding. Running it next to the hosts keeps the hop short; the graphics pipeline's H.264 keeps re-encoding rare.

§8 Trust

What the bridge sees. The bridge ends both connections, so it sees the whole session: the screen, every key, the clipboard, the files that pass, and the password or PIN when the user signs in to Windows that way. It deserves the same care as the session hosts themselves: run it in their network zone, on machines that do nothing else, and keep it patched. It logs connections, not content.

What the device sees. The device connects to the bridge as to any VTX server: its trusted zone shows the bridge's verified name, and the bridge names the Windows host it reached with v1's CLAIM, so the user sees, for example, the host WS-APPS-03 through rdp.contoso.com. The device relies on the bridge to have checked that host's key (Section 3), as it relies on a Linux host for its own applications.

What Windows cannot do. Every permission and rule of the device's extensions holds for a Windows session: Windows gets a folder only once the user shared it, prints only on printers the user shared and never while the device is locked, cannot select a payment application on a card, cannot reflash a USB device, cannot take focus from what the user does on the device, and cannot fill the screen without asking. The bridge is where those rules meet RDP, and the reference mapping checks the ones a log can show.

Recording and sharing. A bridged session can have several viewers and be recorded with Viewers 1, and every viewer is told, as for any VTX session. Windows itself does not know; organizations that must tell Windows users about recording do so in the bridge's sign-in.

§9 Definitions and checks

As a binding, RDP Bridge 1 lives in the bundle under `hosts/rdp-bridge-1/`, beside Wayland Host 1, and is not part of the registry:

File	What it holds
`rdp-bridge-1.json`	The mapping: 52 rows, each a part of RDP, its level (required, optional, bridge, unsupported or refused), the VTX definitions it maps to and why
`examples/valid/log-remoteapp.jsonl`	The RemoteApp log below: 17 things Windows sends, 18 PDUs the bridge sends it, and 32 VTX frames
`examples/valid/log-desktop.jsonl`	A full Windows desktop on two displays with a German keyboard: keys by position and AltGr, the pointer on both displays, a scale change, and the second display unplugged and plugged back
`examples/invalid/logs.jsonl`	The logs broken 14 ways, one at a time

`tools/make_rdp.py` writes them, with everything the bridge sends, on both sides, written out by hand. `tools/rdp_bridge.py` is the reference mapping: given what Windows sends and the client's frames, it produces the VTX frames and the PDUs a bridge sends. A log is a header line, then what Windows sends (`{"rdp": …}`), what the bridge sends Windows (`{"to": …}`) and the VTX frames, in order. PDUs are written by what they do, with the fields that matter, not byte for byte.

The RemoteApp log. Word opens on the laptop at 200%, in device pixels on the host and in px on the device. Marcel types h, which goes as its scancode, and é from his input method, which goes as Unicode. A menu opens against Word and closes; Windows hides the pointer. Save As opens as a modal dialog and takes focus, because Word, the active window, owns it. Marcel drags Word by its own title bar: the device moves it, the bridge tells Windows where in desktop pixels, and the scene follows once Windows confirms. He maximizes it on the display; Windows carries it out and the patch follows. The document gets a new name. An Outlook reminder asks for the foreground and gets attention instead. The display is unplugged, and Windows moves Word to the laptop.

The validator's Section 27 runs 72 checks, and every one passes:

Part	Checks
The mapping's shape: every row with its part of RDP, level, targets and reason	1
Each row's targets exist, in the schemas and in the CDDL of v1 and every extension	52
The reference mapping handles exactly the host events the rows name	1
Each valid log is exactly what the reference mapping sends, on both sides, and keeps every rule of Windows 1 and Displays 1	4
Each broken log fails at the change it was built for	14

The broken logs: monitors sent in px; a key sent by the character it makes instead of its position; pointer positions in px; a menu opened as an ordinary window; a foreground change that took focus on the device; a window move Windows was not told about; a maximize patched before Windows carried it out; a modal dialog without its owner; composed text sent as keys; a title change not forwarded; a window removed before Windows closed it; focus Windows never heard of; an RDP event the binding does not map; and a desktop window left for an unplugged display.

The binding adds nothing to the wire, so no other count changes. The whole bundle runs 2,543 checks, all passing.