VTX Brokering 1: app lists, launching and session routing

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

§1 Status and scope

Brokering 1 gives a user one address for everything they may run. The broker lists the published desktops and applications, starts the one the user picks on a session host it chooses, and hands the client over to that host with a one-time ticket. Later, it finds the user's sessions wherever they run, so reconnecting never depends on remembering a host. It is an extension of VTX 1, draft d01, in conformance profile 1.

The broker is an ordinary VTX server that the user signs in to. The session hosts are VTX servers too. Seamless applications open as Windows 1 windows, on any display of Displays 1.

Adds	What it is
`broker`	A capability: the client can list and launch; the server is a broker
`broker` channel	One channel the client opens on its broker connection: the catalog, launching, the user's sessions, reconnecting and signing out
`ticket`	A sign-in method: the client signs in to a session host with the ticket the broker gave it
`launch`	An application event: a session host's session manager is told to start an item

This is the part of a remote desktop farm that RDS calls the connection broker and the RemoteApp and Desktop Connections feed, and that Citrix calls StoreFront and the Delivery Controller.

Out of scope. How a broker and its hosts talk among themselves: registering hosts, reporting load and redeeming tickets go over the farm's own interface; Section 10 says only what a redemption must return. Which host the broker picks. Licensing. Relaying VTX through a gateway at the edge of a network. User profiles that follow the user between hosts.

§2 Capabilities and the broker channel

A client offers `broker` in HELLO; a server that grants it in WELCOME is a broker. After READY, the client opens one channel of type `broker`, and the server answers OPEN_OK with the catalog and the user's sessions.

```
cap-broker = {
  ? "files": bool,            ; client: it can open its own files with published items
}
```

Frame	Direction	What it carries
0x10 CATALOG	Server to client	Every desktop and application the user may run; again after every change

Frame	Direction	What it carries
0x11 LAUNCH	Client to server	Start an item, optionally with files and a display
0x12 ROUTE	Server to client	Where to go: a host, a ticket and the host's key; or "it opened in a session you are attached to"
0x13 REFUSED	Server to client	Why a request failed
0x14 RUNNING	Server to client	The user's sessions across the farm; again after every change
0x15 RECONNECT	Client to server	Get back into a running session
0x16 SIGNOUT	Client to server	End a session

Only the client opens the channel (an odd ID), at most one per connection. LAUNCH, RECONNECT and SIGNOUT carry a request number that grows with each request; at most 8 may wait for an answer at once, and answers may come in any order.

Session hosts. A host does not need the `broker` capability. It offers the `ticket` sign-in method in WELCOME's `auth` list, and accepts only tickets from its own broker. One server can be both broker and host: then ROUTE simply names the same host.

§3 The catalog

The catalog lists what this user may run, and nothing else. The broker sends it whole when the channel opens, and again whenever it changes: an application is published or withdrawn, or the user's groups change.

```
brk-catalog = {                                    ; 0x10 CATALOG, S->C: on open and
after every change
  "version": uint,            ; grows with every change
  "items": [* catalog-item],
}
catalog-item = {
  "id": item-id,
  "name": tstr .size (1..80),
  "kind": "desktop" / "app",
  ? "icon": res-hash,         ; an image resource, fetched on the resource channel
  ? "folder": tstr,           ; up to three levels, e.g. "Office/Mail"
  ? "description": tstr .size (1..200),
  ? "collection": tstr .size (1..64),   ; items of one collection share a session on
one host
  ? "types": [1*32 file-type],          ; the files an application opens
}
file-type = { "mime": tstr, ? "ext": [1*16 tstr] }
```

A small office farm might publish:

id	name	kind	folder	collection	opens
`office-desktop`	Office desktop	desktop			
`evolution`	Mail	app	Office	office	`.eml`, `.ics`
`writer`	Writer	app	Office	office	`.odt`, `.docx`
`calc`	Calc	app	Office	office	`.ods`, `.xlsx`, `.csv`
`gimp`	GIMP	app	Graphics	graphics	`.png`, `.jpg`

Collections. Applications of one collection run in one session on one host, as RemoteApp programs of one collection do. Mail, Writer and Calc above share a session: the second one the user starts opens in the session the first one started, so they can share the clipboard, files and memory. GIMP runs in a session of its own, possibly on a host with a GPU.

Icons. Icons are image resources, fetched and cached by their hash on v1's resource channel, so a catalog that is sent again costs no icon data.

Rules. Item ids are unique within a catalog. Desktops open no files, so they have no `types`. `version` grows with every CATALOG the broker sends.

§4 At a glance

> **Diagram: launching an application · client, broker and session host.** Drawn in the PDF version, `12-brokering-1.pdf`.

The user signs in once, at the broker. The ticket goes from the broker to the client, from the client to the host, and back to the broker to be redeemed, so the host learns who the user is without a second sign-in. The pin tells the client which host to trust, so it never has to decide on a host's key by itself.

§5 Launching

The client asks for an item; the broker decides where it runs and answers with one ROUTE or one REFUSED. What the broker decides depends on what the user already runs:

The user already has	The broker	ROUTE
Nothing of this item's collection	Picks a host and prepares a new session there	`uri`, `ticket`, `pin`, `expires`
A session of the collection, with this client attached	Asks the host to start the item in that session	`session` and `joined`: the new windows arrive on the connection the client already has
A session of the collection, with nobody attached	Asks the host to start the item in that session	`uri`, `ticket`, `pin`, `expires` and `session`: the client resumes it

In CBOR diagnostic notation, a launch and its hand-over:

```
LAUNCH {"req": 1, "item": "evolution"}
ROUTE  {"req": 1, "uri": "vtx://h12.farm.example.net/office", "ticket": h'5c0e…',
"pin": h'a41f…', "expires": 60}
```

The hand-over. The client connects to the URI as to any VTX server, with one difference: it refuses the host unless the host presents the key whose SHA-256 is `pin`. HELLO carries the URI's target, and `resume` with the session and no token when ROUTE named one. The host offers `ticket` among its sign-in methods, and the client answers with it:

```
AUTH {"method": "ticket", "ticket": h'5c0e…'}
```

The host redeems the ticket with its broker, learns who the user is and what to start, and sends READY. A ticket works once, at that host, for `expires` seconds after ROUTE (at most 300). A ticket that is unknown, used or expired gets AUTH_FAILED with reason `ticket`; the client then asks the broker again.

Refusals.

code	Meaning
`unknown`	No such item in the user's catalog, or no such session
`denied`	The user may not do this, such as opening a farm path outside their own storage
`busy`	No host has room right now; the client may try again later
`unavailable`	The item's hosts are down or being serviced
`limit`	The user runs as many sessions as the farm allows
`file`	The item does not open a file of that type

§6 Sessions

RUNNING lists the user's sessions on every host of the farm, where v1's SESSION_LIST lists them on one server. The broker sends it when the channel opens and after every change: a session starts or ends, an item starts in it, or a connection attaches or detaches.

```
RUNNING {"sessions": [
  {"session": h'3b9d…', "host": "h12.farm.example.net", "items": ["evolution",
"writer"], "attached": true},
  {"session": h'c071…', "host": "g3.farm.example.net", "items": ["gimp"], "attached":
false}]}
```

The user wants to	The client
Leave a session running	Sends v1's DETACH on that host's connection; nothing changes at the broker except `attached`
Come back on the same device	Resumes directly at the host with v1's resume token; the broker is not needed
Come back on another device	Sends RECONNECT; the broker answers with a ROUTE that names the session, and the client resumes it with the ticket. A connection still attached elsewhere gets v1's GOAWAY `taken_over`

The user wants to	The client
End a session	Sends SIGNOUT; the host ends the session as a desktop signs out, and the next RUNNING no longer lists it

A SIGNOUT that fails is answered with REFUSED; one that succeeds has no answer of its own but that RUNNING. A connection attached to the session when it ends gets v1's GOAWAY `session_ended`.

§7 Opening files

A user who double-clicks `invoice.docx` on the device should see it open in the published Writer. The client finds the item whose `types` cover the file and sends LAUNCH with it:

```
LAUNCH {"req": 4, "item": "writer", "files": [{"name": "invoice.docx", "size": 48213,
        "mime": "application/vnd.openxmlformats-
officedocument.wordprocessingml.document"}]}
```

Files from the device. Once the client is connected to the host (or at once, when ROUTE says `joined`), it uploads the files with Transfer 1, several as one set. Opening a file with an application is the user's own choice of that file, so the upload counts as a pick, as Transfer 1 requires. The host's session manager matches the upload with the launch by names and sizes, and opens the files in the item it started.

Files on the farm. A file already in the user's own storage on the farm goes by `path`, and nothing is uploaded. The broker refuses a path outside that storage with `denied`.

Rules. Only a client that offered `files` sends device files. Files go only to an application whose `types` cover each of them, by MIME type or extension; anything else is refused with `file`. At most 16 files per launch.

The host works on a copy of a device file. Saving it back to the device takes a download through the trusted zone (Transfer 1). Drive redirection, which lets a remote application write to the device's own file, is Drives 1.

§8 On the device

A client may place the catalog among the device's own applications, so a published Writer starts from the Start menu, Launchpad or an application launcher like a local one. This is the device's business, not the wire's, but every client should do it the same way:

- **Only when the user agrees,** once per broker, as RDS asks before adding a RemoteApp feed to the Start menu.
- **Marked as remote.** Entries sit in a folder named after the broker, and their names say which broker they come from, such as "Writer (apps.farm.example.net)", so a remote application is never mistaken for a local one.
- **Kept current.** When CATALOG's `version` changes, the client adds, renames and removes entries to match, and fetches new icons by their hash.
- **File associations as "Open with".** The client offers published applications for the file types in their `types`. It never makes one the default handler without the user's say, and the device's own prompt for that stays in place.

Launch links. An entry stores the broker's URI with the item in the fragment: `vtx://apps.farm.example.net/#launch=writer`. The fragment never leaves the client (v1), so the broker sees an ordinary connection; the client connects, opens the broker channel and sends LAUNCH for `writer`. The same link works anywhere a vtx:// link does.

§9 With other extensions, and without the capability

Extension	With brokering
Windows 1	Seamless items open as windows on the host connection. Windows from several hosts sit side by side, each with its own host beside the title
Displays 1	LAUNCH's `display` reaches the host, which names it on the item's first window. A desktop item covers the displays in the layout's `use`
Transfer 1	Device files opened with an item travel as an upload that counts as a pick (Section 7)
Devices 1	A device key signs the client in to the broker without a passkey prompt. Hosts need no device keys: the ticket signs the user in
Viewers 1	A brokered session is shared like any other: viewers join at the host, and need nothing from the broker

Without the capability. A v1 client can still use a farm. The broker shows its catalog as an ordinary VTX application, and a click ends the connection with GOAWAY carrying the host's URI in `reconnect`. The client follows it and signs in at the host again, since v1 sends no token to another host, and trusts the host by WebPKI, DANE or its key on first use. What it misses is the single sign-in, the vouched key, RUNNING and files.

§10 Application interface

On a session host, a brokered target such as `/office` runs a session manager: the application that starts the published programs and lets the compositor show their windows. Each time an item must start in its session, after a redeemed ticket or a join, it gets a `launch` event:

```
{"kind":"launch","item":"evolution"}
{"kind":"launch","item":"writer","files":
[{"name":"invoice.docx","size":48213,"mime":"application/vnd.openxmlformats-
officedocument.wordprocessingml.document"}]}
{"kind":"launch","item":"calc","files":[{"path":"Documents/
budget-2027.ods"}],"display":"d2"}
```

The session manager maps the item to a program through the farm's own configuration and starts it; the compositor then writes the program's windows as Windows 1 ops. Device files arrive as a Transfer 1 upload, which the session manager matches by names and sizes and opens in the program. A farm path is opened where it lies. When `display` is present, the first window names that display.

Redeeming a ticket. Between host and broker, the farm's own interface carries the redemption. This specification fixes only what it must do and return, so that brokers and hosts from different makers can work together:

The host sends	The broker returns
The ticket	The user who signs in
Its own key	The item to start, its files and display
	The session to resume or join, if any

The broker answers only when the ticket is unused, unexpired and was given for the host whose key is sent, and marks it used before it answers. A host that cannot reach its broker refuses the sign-in with reason `ticket`.

§11 Security

A broker signs the user in once and then sends them to machines the user never chose. The rules below keep that hand-over as safe as signing in to each host directly.

- **Tickets.** 32 random bytes, used once, within `expires` seconds (at most 300), and only at the host they were minted for: at redemption the broker checks the key of the host that presents the ticket. Tickets travel only inside the encrypted broker and host connections. A stolen ticket is useless at any other host, and after its first use.
- **Vouched hosts.** The client refuses a host whose key does not match `pin`, even when WebPKI would accept its certificate, so a hijacked DNS name cannot receive the user. For Noise hosts the pin replaces trust on first use. The trusted zone shows the host together with the broker that vouched for it, and Windows 1 still shows each window's host beside its title.
- **Resume tokens stay put.** v1 sends a resume token only to the host that issued it, and Brokering 1 keeps that: a broker never sees one. Getting back in from another device goes through RECONNECT and a fresh ticket.
- **Only the user's own.** The catalog lists only what the user may run, and RUNNING only the user's own sessions. A farm path outside the user's own storage is refused with `denied`.
- **Launch links.** A link with `#launch=` can only start an item from the catalog of the broker it names, after the user signs in there. When such a link comes from outside the client, from a mail or a web page, the client names the broker and the item and asks once before it launches.
- **Limits.** At most 8 requests waiting, 16 files per launch, 64 items per session in RUNNING, and tickets valid for at most 300 seconds.

§12 Definitions and checks

The definitions live in the bundle under `ext/brokering-1/`, next to Displays 1:

File	What it holds
`vtx-brokering-1.cddl`	The capability, the broker channel and its seven frames, and the `ticket` sign-in
`vtx-app-events-brokering-1.schema.json`	Server lines to an application, plus the `launch` event. Brokering 1 adds no scene op, so Displays 1's ops and application schemas apply

File	What it holds
`examples/valid/`	Two catalogs, the launch trace (Mail on one host, Writer with a file joining it, GIMP on a second host, detach and sign-out), the reconnect trace (a second device, an expired ticket, a retry) and a session manager's lines
`examples/invalid/`	Broken catalogs, launch events and traces, each with the check that must catch it

`tools/gen_ext_brokering.py` writes the schema, built on the Displays 1 generator; `tools/make_brokering.py` writes the examples, in the multi-connection trace format of Devices 1. Both are deterministic.

The validator's Section 20 runs 55 checks, and every one passes:

Part	Checks	What they cover
Setup	6	The schema is valid; the CDDL parses alone and with the first nine extensions, Windows 1 and Displays 1; no rule is defined twice; earlier schemas reject `launch`; the core alone rejects the `ticket` sign-in
CDDL and catalogs	21	7 valid samples accepted and 9 malformed ones rejected; 2 catalogs pass the CDDL and the rules of Section 3; 3 broken catalogs pass the CDDL and fail a rule
Traces	22	2 valid traces replayed in full across the broker and the hosts; 20 broken ones, each caught by the right check, among them a reused ticket, a ticket taken to the wrong host, a host that does not match its pin, an expired ticket accepted, a file the item does not open, a join to a session the client is not in, and a signed-out session still running
Application interface	6	4 valid lines, each launch naming a catalog item and only files it opens; 2 rejected

The whole bundle runs 2,543 checks, all passing.

§13 Wire summary

Everything Brokering 1 adds, for the registry and for implementers:

Kind	Addition
Capability	`broker`: the client sends `{}` or `{"files": true}`; a server that grants it is a broker
Channel type	`broker`, opened by the client, at most one per connection: 0x10 CATALOG, 0x11 LAUNCH, 0x12 ROUTE, 0x13 REFUSED, 0x14 RUNNING, 0x15 RECONNECT, 0x16 SIGNOUT
Sign-in method	`ticket`, with AUTH's `ticket` (32 bytes); AUTH_FAILED reason `ticket`
Refusal codes	`unknown`, `denied`, `busy`, `unavailable`, `limit`, `file`
Application event	`{"kind":"launch","item":…,"files":[…],"display":…}`, to a session host's session manager
Launch link	`vtx://broker/#launch=<item>`, read only by the client

No frame of the core, the first nine extensions, Windows 1 or Displays 1 changes meaning. A host still accepts every sign-in method it did before, and a client without Brokering 1 still reaches a farm through GOAWAY's `reconnect` (Section 9).