Functionize Quickconnect (ACS Quickconnect)

Non-Public Application Testing · https://docs.functionize.com/hc/en-us/articles/42178397358359-Functionize-Quickconnect-ACS-Quickconnect

QuickConnect is a small, portable desktop application. It creates an end-to-end encrypted connection between Functionize's test-execution engine and the user's computer, so automated tests can reach applications on the user's internal network. It runs on Windows, macOS, and Linux, either from the system tray or in a headless command-line mode.

How the secure connection works

- QuickConnect uses Tailscale to join the user's machine to a private, Functionize-managed network (a "tailnet").
- Tailscale builds its tunnels with WireGuard, a modern, widely audited, open-source VPN protocol. Traffic is encrypted end to end with ChaCha20-Poly1305, and the keys live only on the user's computer and the Functionize test engine. Any relay in between forwards encrypted packets; it cannot read, modify, or decrypt them.
- Handles corporate proxies — auto-detects HTTP/HTTPS/SOCKS5/PAC proxies and prompts for credentials (Basic or NTLM) if the proxy requires authentication.
- The connection is outbound-only and started by the agent. QuickConnect opens no inbound ports on the machine or the network, and Functionize never dials in, so no inbound firewall rules are required.

No installation, no admin rights

- There is no installer. QuickConnect is a single portable executable; the user downloads it and runs it.
- It needs no administrator or root credentials and runs entirely in user space: no kernel driver, no system-wide VPN, and no changes to the machine's network settings. It does not route all of the machine's traffic, only the test traffic for in-scope resources.

The user is always in control

- The connection exists only while QuickConnect is running and connected.
- Clicking Disconnect, or simply closing QuickConnect, immediately tears down every outbound connection. From that moment Functionize can no longer reach any internal resource, and nothing keeps running in the background.
- Each session is stateless: QuickConnect writes no persistent configuration or credentials to disk.

Designed for short, interactive sessions

QuickConnect is meant to run on an engineer's laptop or desktop for the length of a working session. Every run needs a fresh, one-time connection key; keys cannot be reused, and the network node is ephemeral. It is deliberately not a tool for always-on or long-lived access. That is a separate, server-side ACS deployment.

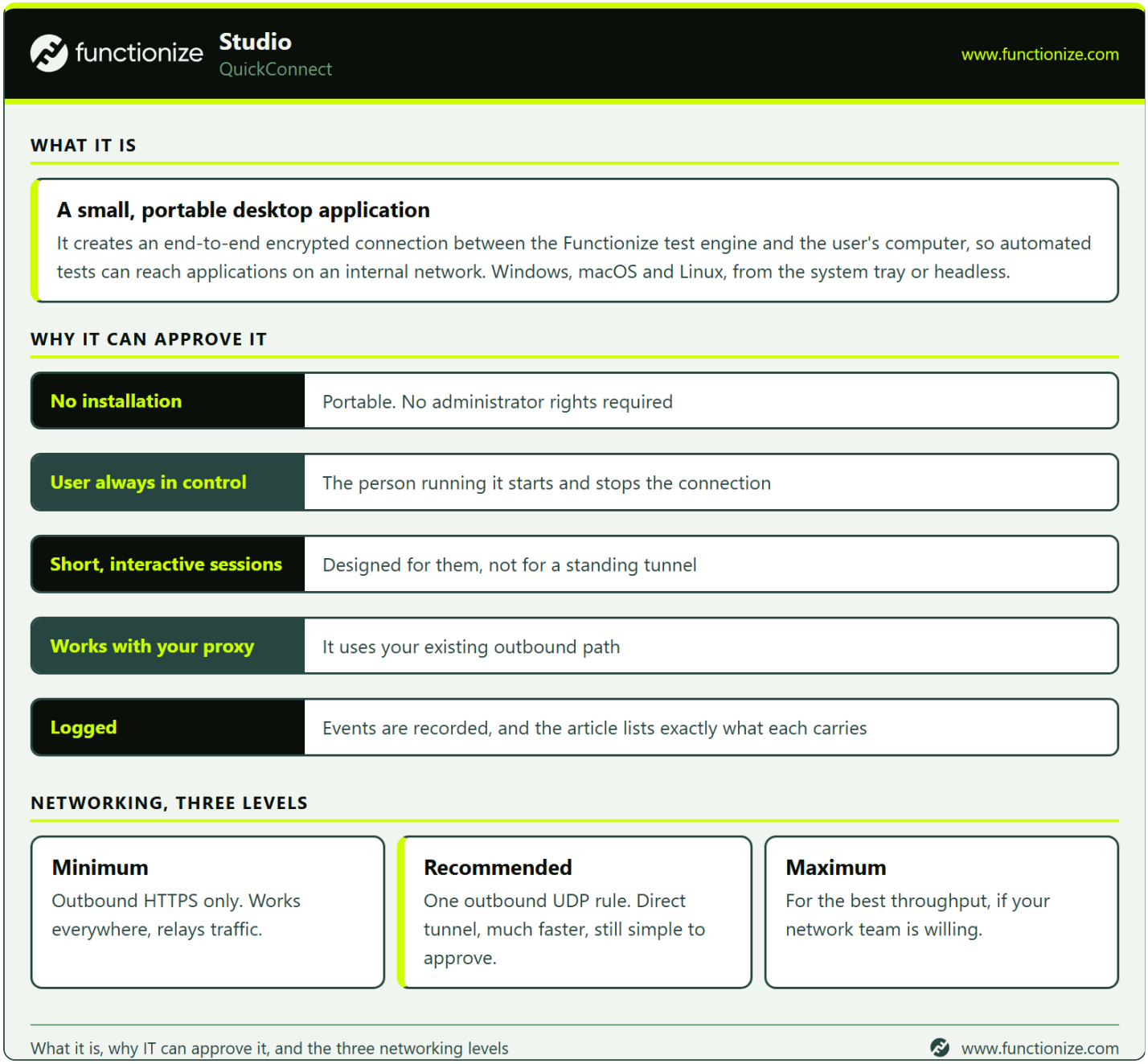
Logging

Every session writes a plain-text log file, so activity is auditable on the user's own machine.

- File name: *Functionize-QuickConnect-YYMMDD-HHMMSS.mmm.log* (for example, *Functionize-QuickConnect-260501-142315.482.log*). Each connection gets its own new file.
- Location: the folder the executable runs from (next to the *.exe* or binary on Windows and Linux, and next to the *.app* on macOS). If that folder is not writable, it falls back to *~/Documents/Functionize QuickConnect/logs/*.
- The log records the connection lifecycle and diagnostics in plain language, and it stays on the user's machine.

Works with your existing proxy

QuickConnect detects and uses the machine's existing proxy configuration automatically, whether that comes from system settings, a PAC file, or environment variables, and it speaks HTTP CONNECT, HTTPS, and SOCKS5. If the proxy needs credentials (Basic, NTLM, or Kerberos), QuickConnect prompts the user for them; otherwise there is nothing to set up.



Works through TLS-inspecting proxies, no bypass required. If a proxy does full TLS inspection or SSL decryption (Zscaler, Netskope, Forcepoint, and the like), QuickConnect detects it and switches to its WSS-Relay fallback. That fallback is ordinary outbound HTTPS on TCP/443, which the proxy can inspect but cannot break, and it rebuilds the same end-to-end-encrypted tunnel from a clean vantage point. The connection still succeeds without anyone disabling or exempting inspection; IT only has to allow outbound access to **.functionize.com* and **.tailscale.com*.

Networking requirements

Minimum

TCP/443 to **.functionize.com and *.tailscale.com.* But please **NOTE** that this give the worst performance.

Best of both worlds (security and speed): one outbound UDP rule for speed

The connection already works with TLS inspection fully enabled, as described above. For the best performance, IT can add one outbound firewall rule that changes no inspection settings and adds no inbound rules:

Allow outbound *UDP/50000* to Functionize's regional peer-relays. The relay IP addresses are published, and kept current, at `https://packages.functionize.com/acs-quickconnect/bootstrap/relays.json`. Allow-list those.

- QuickConnect's bulk traffic is WireGuard over UDP, and Functionize runs regional peer-relays on *UDP/50000*. With this rule, the engineer's computer and the test engine carry their data over that fast UDP relay, while only lightweight coordination crosses the inspected TCP/443 path.
- This rule is scoped by IP, not by domain. WireGuard packets go straight to the relay's IP and carry no hostname or TLS SNI for a proxy or firewall to match on, so a **.functionize.com*-style rule will not cover this path. Allow-list the published relay IPs above, or load them as FQDN address objects that your firewall resolves itself.
- This is a firewall **allow**, not an inspection **exemption**: your full TLS inspection on 443 stays exactly as it is. Proxies only carry TCP, so the UDP path is a direct outbound rule that never touches the proxy.
- It is a performance optimization, not a requirement. Without it the connection still works; the data just falls back to the relay over the inspected TCP/443 path, at much higher latency.

Maximum performance

Wildcards are recommended. Every rule is outbound only:

- *TCP/443* to **.tailscale.com and *.functionize.com*: coordination, control plane, and the inspection-tolerant relay with not TLS-inspection.
- *UDP/50000* to the regional peer-relay IPs published at `https://packages.functionize.com/acs-quickconnect/bootstrap/relays.json`: the fast data path, worth allowing even when TCP/443 stays inspected (see above). WireGuard is addressed by IP, so allow-list this path by IP, not by domain.
- *UDP/3478* and *UDP/41641*: direct-connectivity discovery and peer-to-peer WireGuard.

Every rule here is outbound. QuickConnect opens no inbound ports and requires no inbound rules, since all traffic is agent-initiated. You also keep TLS inspection enabled: QuickConnect works through it via the WSS-Relay on TCP/443. The only IP allow-listing is outbound, and only for the optional UDP/50000 fast path (the relay IPs above); baseline connectivity needs none.
For the full protocol, port, and TLS breakdown, see the **connectivity data-flow diagram**.

Telemetry (what QuickConnect reports back)

QuickConnect sends a small stream of connection-health events back to Functionize so we can tell whether sessions are connecting and catch the ones that fail. They go out as outbound HTTPS to *tailgate.functionize.com* (already covered by the **.functionize.com* egress rule above) and use the same proxy or PAC settings as the rest of QuickConnect.

What each event carries

- The event type, from a short fixed list: session started, tunnel established, a proxy or PAC file was detected, TLS inspection was detected, or a connection step failed.
- A coarse reason code when something fails, for example authentication rejected or relay unreachable. No error text, no stack trace.
- A random session ID, generated fresh on each run and not tied to you or your machine.
- A timestamp, the OS family (Windows, macOS, or Linux), and the QuickConnect version.

It never sends the URLs, hostnames, or IP addresses of anything on your network, nor your proxy addresses, your credentials, or any traffic, payloads, or file contents. The OS family and app version are the only identifying details that go with it.

Events are authorized with the same one-time session token used to connect, and that token is used purely as a credential — it never gets written into the telemetry or the log file. QuickConnect holds events in memory, retries a few times, and quietly drops them if they can't be delivered. Nothing touches disk, and your connection never waits on telemetry.

Why IT can approve it

Concern	QuickConnect
Inbound access to our network	None. Outbound-only and agent-initiated, with no listening ports and no inbound rules.
Can Functionize or its relays see our traffic?	No. WireGuard encrypts it end to end; intermediaries forward ciphertext only.
Does it need admin rights or a driver?	No. User space only, with no changes to the OS network stack.
Does it stay installed or persist?	No. Portable binary, stateless per session, nothing left behind.
Persistent or unattended access?	No. One-time key per session; access ends the moment the user disconnects or closes the app
Works behind full TLS inspection?	Yes. It falls back to the WSS-Relay over standard HTTPS, with no "do-not-inspect" exception.
Scope of access	Limited to what the user's machine can already reach; traffic is NAT'd to the machine's own IP.
What data does it send back to Functionize?	Only connection-health events (e.g. "tunnel established", "connection failed"), plus OS family a app version. No URLs, hostnames, IPs, credentials, or traffic contents. See Telemetry above.
What happens to proxy credentials we enter?	Used only to authenticate to your own proxy for the session. Nothing is written to disk — sess are stateless, so credentials never persist.

Related

- ACS Quickconnect Data-flow Diagram — every flow, zone and firewall, in one diagram
- Before You Point Studio at Your Own Application — the access questions to settle first
- Timeouts, Access and Browser Behavior: Getting Studio Into Your Application — getting Studio into your application