

SASE ARCHITECTURE & ENDPOINT-TO-VENDOR-CLOUD TROUBLESHOOTING REFERENCE GUIDE

Zscaler vs. Netskope | Architecture | Commands | Evidence | Location Change

Prepared by: Sana Ullah — Senior SASE Engineer

Version 1.0 — September 20, 2026

SCOPE Operational troubleshooting reference for endpoint-to-cloud and private-application access. Validate tenant-specific FQDNs, ports, licenses and policy behavior against current vendor documentation and your deployment. Zscaler content reflects direct hands-on operational experience; Netskope content is based on architecture research and documentation rather than day-to-day operational use, and component naming (e.g. NewEdge Gateway) should be re-verified against current Netskope documentation as vendor terminology evolves.

1. ZSCALER vs. NETSKOPE — QUICK ARCHITECTURE MAP

Concept	Zscaler	Netskope
Endpoint agent	Zscaler Client Connector (ZCC)	Netskope Client
Remote private access	ZPA via Zscaler Service Edge	NPA / UZTNA via NewEdge Gateway
Application-side component	ZPA App Connector	Netskope Publisher
Private app definition	Application Segment	Private App / Private App Segment
Access control	ZPA Access Policy	Private Access / Real-time Protection policy
Device context	Posture Profiles / endpoint signals	Device Classification / Device Intelligence

ZPA REMOTE Endpoint -> ZCC -> ZPA Service Edge -> App Connector -> Private App

NPA / UZTNA NEWEDGE PATH Endpoint -> Netskope Client -> NewEdge Gateway -> Publisher -> Private App

NPA / UZTNA LOCAL PATH Endpoint -> Netskope Client -> Local Broker -> Publisher -> Private App (when deployed/selected)

IMPORTANT Netskope Local Broker and Zscaler Private Service Edge (PSE) both support private/local service-edge designs, but they are not 1:1 equivalents. Treat each as a vendor-specific architecture and selection model.

GOLDEN RULE Successful DNS or TCP/443 testing proves underlay reachability only. It does NOT by itself prove SASE steering, tunnel establishment, authentication, posture, or policy authorization.

2. ENDPOINT → CLOUD / UNDERLAY TESTING COMMANDS

Test	Command / URL	Where	What it proves / caution
Local config	<code>ipconfig /all</code>	CMD / PowerShell	IP, adapter, gateway, DNS and suffix
Routing table	<code>route print</code>	CMD / PowerShell	Default route, VPN routes and metrics
PowerShell routes	<code>Get-NetRoute</code>	PowerShell ¹	Structured Windows route information
DNS	<code>nslookup mobile.zscalertwo.net</code> <code>nslookup addon-xxxxx.eu.goskope.com</code>	CMD / PowerShell	Name resolution only; not service reachability
PowerShell DNS	<code>Resolve-DnsName gateway.gslb.goskope.com</code>	PowerShell ¹	Native DNS lookup and returned records
TCP 443	<code>Test-NetConnection mobile.zscalertwo.net -Port 443</code>	PowerShell ¹	TCP handshake only; does not prove steering
TCP 443 short	<code>tnc gateway.gslb.goskope.com -Port 443</code>	PowerShell ¹	Short form TCP reachability test
TLS / HTTP	<code>curl.exe -v https://ip.zscaler.com</code> <code>curl.exe -v https://vpn.iotsoftsolutions.com</code>	CMD / PowerShell ²	Tests TCP + TLS + HTTP behavior
HTTPS request	<code>Invoke-WebRequest https://ip.zscaler.com</code>	PowerShell ¹	PowerShell application-layer request
Quick probe	<code>ping mobile.zscalertwo.net</code>	CMD / PowerShell	DNS + ICMP clue; failure is not proof of outage
Path clue	<code>tracert gateway.gslb.goskope.com</code>	CMD / PowerShell	Underlay clue; cloud hops may suppress replies
TCP via PortQry	<code>portqry.exe -n mobile.zscalertwo.net -p TCP -e 443</code>	CMD / PowerShell ³	TCP port state; PortQry must be installed
UDP via PortQry	<code>portqry.exe -n addon-xxxxx.eu.goskope.com -p UDP -e 443</code>	CMD / PowerShell ³	Syntax example for a documented UDP/443 service endpoint
TCP via PsPing	<code>psping gateway.gslb.goskope.com:443</code>	CMD / PowerShell ⁴	TCP reachability; Sysinternals PsPing required
Zscaler path check	<code>https://ip.zscaler.com</code>	Browser	Shows whether web traffic appears to traverse Zscaler
Netskope path check	<code>http://notskope.com</code>	Browser	Netskope steering/path diagnostic

PORT NOTE TCP 443 commonly carries HTTPS/TLS. UDP 443 is used by protocols such as QUIC/HTTP/3 and by some vendor-specific UDP/DTLS transport modes. Test UDP only against an endpoint that the product documentation says should accept it.

NPA NOTE For Netskope private-app access, do not treat ping/traceroute failure as proof that the application is down. Validate the supported application protocol and the client/broker/publisher path instead.

¹ Requires PowerShell 3.0+ with the NetTCPIP/DnsClient modules (Windows 8 / Server 2012 or later). On older PowerShell, these cmdlets do not exist.

² Native curl.exe ships only from Windows 10 version 1803 onward. On earlier builds, typing "curl" invokes the Invoke-WebRequest alias instead, producing different output, unless curl is separately installed.

³ PortQry is a separate Microsoft utility — not built into Windows. Download and install it before use.

⁴ PsPing is a separate Sysinternals utility. First run requires EULA acceptance — add `-accepteula` for non-interactive/scripted use.

3. ZSCALER ZPA — QUICK TROUBLESHOOTING

PRIVATE-APP PATH Endpoint -> ZCC -> Network Detection -> Forwarding Profile -> ZPA Service Edge / PSE -> App Connector -> Private App

S.No.	Checkpoint	Question
1	Client state	Is ZCC authenticated, enabled and healthy?
2	Network detection	Is the device On-Trusted or Off-Trusted? Did the location change?
3	Forwarding Profile	What ZPA forwarding action applies to the detected network state?
4	VPN interaction	Is another VPN connected? Full tunnel or split tunnel?
5	DNS	Does the private-app FQDN resolve using the expected resolver? (e.g. <code>Resolve-DnsName app.iotsoftsolutions.com</code>)
6	Route	Which adapter, default route and route metric win?
7	ZPA reachability	Can the endpoint reach the required ZPA service over the expected protocol/port?
8	App definition	Does the Application Segment contain the correct hostname/IP and port?
9	App Connector	Is it healthy, and can it resolve/reach the private application?
10	Policy / posture	Do identity, posture and ZPA Access Policy authorize the session?
11	Logs	Do ZCC/ZPA logs show the request and where the flow stops?

4. LOCATION CHANGE: WORKS AT HOME, FAILS AT HQ

SCENARIO A user can access a private application from home, arrives at HQ, and access stops. Treat the location change itself as a troubleshooting clue.

Platform	Focused investigation path
Zscaler	Compare Trusted Network detection -> Forwarding Profile -> VPN state -> DNS/routes -> ZPA path -> App Connector/private app -> policy/logs.
Netskope	Compare Client steering/location -> expected NewEdge Gateway vs Local Broker selection -> Publisher -> private app -> identity/device policy -> logs.

PRINCIPLE A location change can alter network classification, DNS, route selection and forwarding behavior. Compare a known-working session with the failing session before assuming the application itself is down.

5. NETSKOPE NPA / UZTNA — QUICK TROUBLESHOOTING

NEWEDGE PATH Endpoint -> Netskope Client -> NewEdge Gateway -> Publisher -> Private App

LOCAL-BROKER PATH Endpoint -> Netskope Client -> Local Broker -> Publisher -> Private App (when Local Broker is deployed and selected)

S.No.	Checkpoint	Question
1	Client	Connected to the correct tenant/user? Is Private Access enabled?
2	Steering	Is the private-app destination selected by the expected Steering Configuration?
3	Identity	Is the correct user/group authenticated?
4	Device posture	Does managed/unmanaged classification and posture satisfy policy?
5	Broker selection	Which gateway path is expected and selected for this location: NewEdge or Local Broker?
6	Publisher	Is the appropriate Publisher healthy and connected to its broker/service?
7	Publisher DNS	Can the Publisher resolve the private application correctly? (e.g. <code>dig app.iotsoftsolutions.com</code>)
8	App reachability	Can the Publisher reach the app IP/FQDN and required TCP/UDP port?
9	Policy	Does the Private App definition and access rule authorize this flow?
10	Logs	Do client/private-access events show where the flow failed?

LOCAL BROKER Broker selection is configuration-dependent. Netskope can evaluate on-prem/off-prem location, configured preference/fallback, reachability and latency. Do not assume every on-prem user automatically uses a Local Broker.

6. NETSKOPE UZTNA — COMMON ACCESS USE CASES

Use case	Mechanism	Key idea
Managed endpoint	Netskope Client	Identity + device posture + private-app access
Third party / BYOD	Netskope One Enterprise Browser	Controlled private-app access without normal client deployment
Data protection	DLP / threat controls	Protect sensitive data, especially on unmanaged devices
Server-initiated flow	ZTNA-brokered flow	Example: remote assistance without broad network exposure
IoT / OT	Device Intelligence + policy	Discover, classify, assess risk and apply segmentation/access controls

POLICY MODEL WHO + DEVICE + ACCESS METHOD -> PRIVATE APP -> ACTION. Exact policy objects, licensing and deployment options vary by tenant.

7. LINUX / CONNECTOR / PUBLISHER COMMANDS

Purpose	Command / Example	What to check
Interface / IP	<code>ip addr ; ip link</code>	Address and interface state
Route / gateway	<code>ip route ; route -n ; netstat -rn</code>	Default gateway and path
Neighbor / ARP	<code>ip neigh ; arp -a</code>	Local L2 neighbor resolution
DNS config	<code>cat /etc/resolv.conf</code>	Configured resolver(s)
DNS query	<code>dig app.iotsoftsolutions.com</code> <code>nslookup gateway.gslb.goskope.com</code>	Resolution and returned records
TCP port	<code>nc -vz app.iotsoftsolutions.com 443</code>	TCP reachability

TLS / HTTP	<code>curl -vk https://app.iotsoftsolutions.com</code>	TLS/HTTP behavior
TLS handshake	<code>openssl s_client -connect mobile.zscalertwo.net:443</code>	Certificate/TLS handshake details
Sockets	<code>ss -ant ; ss -tulnp</code>	Connections/listeners
Path	<code>traceroute <host> ; tracepath <host> ; mtr <host></code>	Underlay path clues
Process	<code>ps -ef ; top</code>	Process/service resource clues
System	<code>free -m ; df -h ; uptime</code>	Memory/disk/load
Logs	<code>journalctl ; tail -f <log> ; grep <pattern> <log></code>	Service/system evidence
Packet capture	<code>tcpdump -ni any host 10.10.10.25 and port 443</code>	Wire-level evidence
Firewall	<code>iptables -L -n ; nft list ruleset</code>	Local firewall rules

8. WORKING vs. FAILING — EVIDENCE CHECKLIST

Evidence	Working	Failing
Compare		
Timestamp / time zone	Correlate endpoint and cloud logs	
Location / network state	Home/HQ; trusted/off-trusted	
Corporate VPN state	Connected? Full/split?	
SASE client state	Connected, error, steering state	
DNS result + resolver	Different resolver/returned IP?	
Default route / metric	Which adapter/path wins?	
Vendor service reachability	Expected TCP/UDP path reachable?	
Identity / posture	Any difference?	
Policy / app mapping	Same user, rule, segment and port?	
Cloud event/log	Did traffic reach the expected service?	

WORKFLOW Endpoint -> IP -> Route/VPN -> DNS -> vendor reachability -> client steering/tunnel -> identity/posture -> policy -> connector/publisher -> application -> logs/packet capture. Compare working vs. failing evidence.

9. ADDITIONAL COMMANDS — ENDPOINT TO INGRESS CLOUD EDGE

Purpose	Where	Command / Example	What it proves / caution
Combined ping+route stats	CMD / PowerShell	<code>pathping mobile.zscalertwo.net</code>	Shows per-hop packet loss over time; more diagnostic than a single tracert for intermittent path issues
Active TCP sessions (native)	PowerShell ¹	<code>Get-NetTCPConnection -RemotePort 443</code>	Native PowerShell alternative to netstat; shows live sessions to the cloud edge on 443
Local DNS cache check	CMD / PowerShell	<code>ipconfig /displaydns</code>	Shows cached DNS entries; a stale cached IP can silently bypass correct SASE steering
Clear local DNS cache	CMD / PowerShell	<code>ipconfig /flushdns</code>	Forces fresh resolution — use after a location/steering change to rule out stale-cache false failures
System proxy check	CMD / PowerShell	<code>netsh winhttp show proxy</code>	Shows the OS-level (WinHTTP) proxy config; important when the agent uses explicit proxy / PAC mode rather than a tunnel
PAC file fetch test	CMD / PowerShell	<code>curl.exe -v https://gateway.gslb.goskope.com/proxy.pac</code>	Confirms the endpoint can actually download the PAC file, not just reach the gateway
Adapter / MTU detail	PowerShell	<code>Get-NetIPInterface Select ifAlias,NlMtu</code>	Shows MTU per adapter; mismatched MTU is a common cause of intermittent tunnel fragmentation issues (GRE/IPsec)
MTU / fragmentation test	CMD / PowerShell	<code>ping -f -l 1472 mobile.zscalertwo.net</code>	Tests for path MTU/fragmentation problems; a failure here but success at a smaller size points to an MTU issue along the tunnel path
Latency to cloud edge	CMD / PowerShell	<code>curl.exe -o NUL -s -w "%{time_connect} % {time_appconnect} %{time_total}\n" https://ip.zscaler.com</code>	Breaks down TCP-connect vs. TLS-handshake vs. total time — useful to isolate whether slowness is network or TLS/app layer
Linux: route for a destination	Linux	<code>ip route get app.iotsoftsolutions.com</code>	Shows exactly which local route/interface will be used to reach that destination right now
Linux: DNS resolver status	Linux	<code>resolvectl status</code>	On systemd-based distros, shows which DNS resolver is actually being used per interface — useful when split-DNS is expected
Linux: MTR summary report	Linux	<code>mtr -rw -c 50 gateway.gslb.goskope.com</code>	Non-interactive MTR report combining ping + traceroute stats over 50 probes; good evidence to attach to a ticket

TIP DNS cache and proxy/PAC checks are easy to skip but catch a real class of "works from home, fails at HQ" tickets that pure port-reachability tests (Section 2) will miss entirely.

¹ Requires PowerShell 3.0+ with the NetTCPIP module (Windows 8 / Server 2012 or later).