Prism Central

- Unregister
  - Important Commands
- PCVM & Registration
- Prism Service (Prism Element)
- Registration Port Check

Unregister

Important Commands

Normal (PC reachable)

Run on a PE CVM while Prism Central is reachable

```
ncli multicluster remove-from-multicluster external-ip-address-or-svm-ips=<PC_IP>
```

Force (PC unreachable / already gone)

Run on a PE CVM when Prism Central can't be reached (deleted, network down, etc.)

```
ncli multicluster remove-from-multicluster external-ip-address-or-svm-ips=<PC_IP> force=true
```

Verify

Check registration status before/after

```
ncli multicluster get-cluster-state
```

PCVM & Registration

Genesis on the PCVM

Status of Genesis services on the Prism Central VM

```
genesis status
```

Registration / Multicluster

Registration status of Prism Element clusters against this Prism Central

```
ncli multicluster get-cluster-state
```

Registering/removing Prism Central in current AOS versions mostly happens via the Prism UI (Settings → Prism Central Registration) — the ncli syntax for this varies by version, so it's deliberately not listed here as a command.

Prism Service (Prism Element)

Prism Service (Prism Element)

Determine the current Prism leader

```
cluster status | egrep "CVM|Prism"
curl localhost:2019/prism/leader
```

Stop/start the Prism service cluster-wide

```
allssh '~/cluster/bin/genesis stop prism'
cluster start
```

Stop/restart Prism on a single CVM

```
~/cluster/bin/genesis stop prism
~/cluster/bin/genesis restart
```

Registration Port Check

PE ↔ PC Ports

Source: Nutanix "Port Details List" export (sections Prism Central + AOS), as of 2026-09-28

Port (TCP)	Direction	Level	Purpose
9440	both	required	Prism / API – registration and all communication
9300, 9301	both	required	Mercury Fanout gRPC – preferred PE↔PC channel
ICMP	both	required	PC pings PE VIP / CVM IPs
123 UDP	PC → PE VIP	required	NTP
2106	PC → PE (bi)	ops	LCM
3205, 3260	PC → PE	ops	iSCSI data/control path for PCVM disks
9361, 9362, 9463	PE → PC (bi)	ops	Ikat proxy L4/L7/reverse (pc.2024.3+)
2060	PE → PC (bi)	ops	Atlas – network provisioning
2123 (2120 fallback)	PE → PC	ops	Metropolis – VM recovery points
2100, 22	PE → PC	special	Genesis / SSH – PC deployment only
9441	PC → PE	special	CAC client authentication only
2200	PC → PE (bi)	special	Genesis Gateway – cluster creation from PC

Port Check Script

Checks ping, all ports above for the given direction, Prism API response and clock skew. Plain bash (/dev/tcp) — no nc/nmap needed. Run on both sides.

```
bash pc_port_check.sh pe <PC_IP>                                  # on a PE CVM
bash pc_port_check.sh pc <CVM1> <CVM2> <CVM3> <CLUSTER_VIP>     # on the PCVM
```

required = red if closed, ops = yellow, special = grey. If all ports are open and registration still fails: check DNS/NTP on the PCVM and `ncli multicluster get-cluster-state` on the CVM (still registered elsewhere?).

Script: pc_port_check.sh

```
#!/usr/bin/env bash
# pc_port_check.sh — checks whether Prism Element (PE) and Prism Central (PC) can reach each other.
# Port list: Nutanix "Port Details List" (sections "Prism Central" and "AOS"), as of 2026-09-28.
#
# Usage:
#   On a PE CVM:  bash pc_port_check.sh pe <PC_IP>
#   On the PCVM:  bash pc_port_check.sh pc <CVM_IP> [<CVM_IP> ... <CLUSTER_VIP>]
#
# Test both directions — many ports must be open both ways.

TIMEOUT=3   # seconds per test

# Format: "port|level|service — purpose"
#   level: required = registration/communication fails without it
#          ops      = features after registration (LCM, backup, networking, ...)
#          special  = deployment/special cases only, failure usually irrelevant
PORTS_PE_TO_PC=(
  "9440|required|Prism/API — registration and all communication"
  "9300|required|Mercury Fanout (gRPC) — preferred channel PE<->PC"
  "9301|required|Mercury Fanout (gRPC) — preferred channel PE<->PC"
  "9361|ops|Ikat-Proxy L4 (pc.2024.3+)"
  "9362|ops|Ikat-Proxy L7 (pc.2024.3+)"
  "9463|ops|Ikat-Proxy Reverse (pc.2024.3+)"
  "2060|ops|Atlas — network provisioning"
  "2123|ops|Metropolis — VM recovery points (mTLS)"
  "2120|special|Metropolis — fallback if 2123 unavailable"
  "2100|special|Genesis — PC deployment only"
  "22|special|SSH — PC deployment only (password reset)"
)
PORTS_PC_TO_PE=(
  "9440|required|Prism/API — registration and all communication"
```

```
  "9300|required|Mercury Fanout (gRPC) — preferred channel PE<->PC"
  "9301|required|Mercury Fanout (gRPC) — preferred channel PE<->PC"
  "2106|ops|LCM — upgrades via Prism Central"
  "3205|ops|iSCSI data path (PCVM disks on cluster)"
  "3260|ops|iSCSI control path (PCVM disks on cluster)"
  "9441|special|CAC client authentication only"
  "2200|special|Genesis Gateway — cluster creation from PC only"
)
# Not testable via TCP: UDP 123 (NTP PC -> PE VIP) — covered by the clock comparison below.

GREEN='\033[32m'; RED='\033[31m'; YELLOW='\033[33m'; GREY='\033[90m'; NC='\033[0m'
FAILED=0; WARN=0

DIRECTION=$1; shift 2>/dev/null
case "$DIRECTION" in
  pe) PORTS=("${PORTS_PE_TO_PC[@]}"); LABEL="PE -> PC" ;;
  pc) PORTS=("${PORTS_PC_TO_PE[@]}"); LABEL="PC -> PE" ;;
  *)  DIRECTION="" ;;
esac
if [ -z "$DIRECTION" ] || [ $# -lt 1 ]; then
  echo "Usage:"
  echo "  on a PE CVM:  bash $0 pe <PC_IP>"
  echo "  on the PCVM:  bash $0 pc <CVM_IP> [<CVM_IP> ... <CLUSTER_VIP>]"
  exit 1
fi

# Tests a TCP port with plain bash — works without nc/nmap
check_port() {
  local host=$1 port=$2 level=$3 info=$4
  if timeout "$TIMEOUT" bash -c "echo > /dev/tcp/$host/$port" 2>/dev/null; then
    printf "  ${GREEN}%-9s${NC} TCP %-5s %-9s %s\n" "OPEN" "$port" "$level" "$info"
  else
    case "$level" in
      required)  printf "  ${RED}%-9s${NC} TCP %-5s %-9s %s  <-- check firewall!\n" "CLOSED"
"$port" "$level" "$info"; FAILED=1 ;;
      ops)  printf "  ${YELLOW}%-9s${NC} TCP %-5s %-9s %s\n" "CLOSED" "$port" "$level"
"$info"; WARN=1 ;;
      *)        printf "  ${GREY}%-9s${NC} TCP %-5s %-9s %s\n" "CLOSED" "$port" "$level"
"$info" ;;
    esac
```

```
  fi
}

for TARGET in "$@"; do
  echo "================================================================================"
  echo " $LABEL   Target: $TARGET   (from $(hostname))"
  echo "================================================================================"

  # 1) Ping — required per Nutanix: PC pings the PE addresses
  if ping -c 2 -W 2 "$TARGET" >/dev/null 2>&1; then
    printf "  ${GREEN}%-9s${NC} ICMP  %-9s %s\n" "OK" "required" "Ping"
  else
    printf "  ${RED}%-9s${NC} ICMP  %-9s %s\n" "NONE" "required" "Ping — PC must be able to
ping PE (VIP/CVMs)"; FAILED=1
  fi

  # 2) Ports
  for entry in "${PORTS[@]}"; do
    IFS='|' read -r port level info <<< "$entry"
    check_port "$TARGET" "$port" "$level" "$info"
  done

  # 3) Is it really Prism on 9440? (HTTP 401 = login required = good)
  CODE=$(curl -sk -o /dev/null -w '%{http_code}' --max-time 5 -X POST -H 'Content-Type:
application/json' -d '{}' \
         "https://$TARGET:9440/api/nutanix/v3/clusters/list")
  case "$CODE" in
    401|200) printf "  ${GREEN}%-9s${NC} %s\n" "OK" "Prism API responds (HTTP $CODE)" ;;
    000)     printf "  ${RED}%-9s${NC} %s\n" "FAILED" "Prism API no response (timeout/TLS)";
FAILED=1 ;;
    *)       printf "  ${YELLOW}%-9s${NC} %s\n" "NOTE" "Prism API responds with HTTP $CODE" ;;
  esac

  # 4) Clock comparison — skew of more than a few minutes makes registration fail
  REMOTE_DATE=$(curl -sk -I --max-time 5 "https://$TARGET:9440/" | grep -i '^date:' | cut -d'
' -f2- | tr -d '\r')
  if [ -n "$REMOTE_DATE" ]; then
    DIFF=$(( $(date -u +%s) - $(date -u -d "$REMOTE_DATE" +%s) )); DIFF=${DIFF#-}
    if [ "$DIFF" -le 60 ]; then
      printf "  ${GREEN}%-9s${NC} %s\n" "OK" "time skew ${DIFF}s"
```

```
    else
      printf "  ${RED}%-9s${NC} %s\n" "FAILED" "time skew ${DIFF}s — check NTP on both
sides!"; FAILED=1
    fi
  else
    printf "  ${YELLOW}%-9s${NC} %s\n" "?" "could not compare clocks"
  fi
  echo
done

echo "Legend: required = red if closed | ops = yellow if closed | special = grey, usually
irrelevant"
if [ $FAILED -eq 1 ]; then
  echo -e "${RED}At least one required check failed${NC} — see red lines above."
elif [ $WARN -eq 1 ]; then
  echo -e "${YELLOW}Registration should work,${NC} but some features (yellow) will not."
else
  echo -e "${GREEN}All required checks passed.${NC} Now test the opposite direction, if not
done yet."
fi
exit $FAILED
```