Cisco Edge Runbooks

Five printable runbooks for IOS XE border and branch routers

Ivan Bogović

2026

Contents

About this pack 1

Runbook 1: Drain a border router before maintenance (BGP graceful shutdown) 2

Runbook 2: Stop a volumetric flood with RTBH 5

Runbook 3: Sub-second failure detection with BFD and OSPF (IOS XE) 8

Runbook 4: Catch a flapping uplink with EEM 10

Runbook 5: Fail over a static default route when the ISP breaks upstream 12

About this pack

Each runbook has the same sections: **When to use, Preconditions, Configuration template, Verification (with expected output), Rollback, Gotchas.** Configs are generated from the Jinja2 templates in the repository, so what you read here is what the templates render with the example variables. Replace addresses, AS numbers and interface names with yours and **test in a lab or window first**.

Text: CC BY 4.0. Templates: MIT. Source: ibogovic-hub.github.io

Runbook 1: Drain a border router before maintenance (BGP graceful shutdown)

Template	`templates/bgp_graceful_shutdown.j2` (modes: `define`, `apply`, `undrain`, `rollback`)
Risk	Low. Sessions are never reset.
Time	5 min to apply, 1-2 min to converge, same to undo
Source	Blog: "Draining a border router without dropping traffic"

When to use

You are about to reload or shut an uplink on a router with eBGP transit and iBGP to the core. `shutdown` or a reload is indistinguishable from a crash: peers tear down routes and traffic is blackholed for the length of convergence. A drain makes this router *less attractive while its sessions are still up*, so traffic has already moved when you pull the cable.

Use this for **planned** work. Failure handling is BFD, timers and graceful restart; drain is policy.

Preconditions

- ☐ The router has at least one alternative path for every prefix you care about (second border router or second transit). Check `show ip bgp 0.0.0.0/0` on the core.
- ☐ You know your local AS and every neighbor to drain (eBGP *and* iBGP-to-core if they carry the transit routes).
- ☐ The drain route-maps do not already exist with different content: `show route-map DRAIN-OUT`.
- ☐ You have a pre-maintenance baseline: `show ip bgp summary` (prefix counts) and the transit-link input rate.
- ☐ Out-of-band access if you are draining the path you are logged in over.

Configuration template

Variables: `local_as`, `neighbors` (list), `prepend_count`, `drain_local_pref`. Example in `examples/bgp_graceful_shutdown.yml`.

Step 1, **define** once and keep permanently, unapplied (`--mode define`):

```
ip bgp-community new-format
ip community-list standard GSHUT permit 65535:0
!
route-map DRAIN-OUT permit 10
 set community 65535:0 additive
 set as-path prepend 65001 65001 65001
!
route-map DRAIN-IN permit 10
 set local-preference 0
!
```

Step 2, **apply** at the start of the window (`--mode apply`):

```
ip bgp-community new-format
ip community-list standard GSHUT permit 65535:0
!
route-map DRAIN-OUT permit 10
 set community 65535:0 additive
 set as-path prepend 65001 65001 65001
!
route-map DRAIN-IN permit 10
 set local-preference 0
!
router bgp 65001
 address-family ipv4 unicast
  neighbor 198.51.100.1 send-community both
  neighbor 198.51.100.1 route-map DRAIN-OUT out
  neighbor 198.51.100.1 route-map DRAIN-IN in
```

```
 exit-address-family
!
! EXEC (not config) - apply the policy without resetting sessions:
! clear ip bgp 198.51.100.1 soft in
! clear ip bgp 198.51.100.1 soft out
```

What it does: outbound tags `65535:0` (RFC 8326 GRACEFUL_SHUTDOWN) and prepends so external peers deprefer you; inbound sets local-preference 0 so *your own* core stops choosing this box as exit. Drain is bidirectional or it is not a drain.

Verification

Expected output below is representative of IOS XE 17.x. Wording and columns vary by release and platform; compare the *fields that matter* (called out under each block), not the exact text.

```
R1# show ip bgp neighbors 198.51.100.1 policy
 Neighbor: 198.51.100.1, Address-Family: IPv4 Unicast
 Locally configured policies:
  route-map DRAIN-IN, in
  route-map DRAIN-OUT, out

R1# show ip bgp 0.0.0.0/0
BGP routing table entry for 0.0.0.0/0, version 412
  65100
    198.51.100.1 from 198.51.100.1 (198.51.100.1)
      Origin IGP, metric 0, localpref 0, valid, external
  65100
    10.0.0.2 from 10.0.0.2 (10.0.0.2)
      Origin IGP, metric 0, localpref 100, valid, internal, best

R1# show interfaces TenGigabitEthernet0/0/0 | include rate
  5 minute input rate 4210000 bits/sec, 612 packets/sec
```

Pass criteria:

1. **policy** lists both route-maps (this is the reliable local check; `advertised-routes` does not print communities).
2. The best path for the default/important prefixes is **not** via the drained neighbor (`localpref 0` on it, `best` elsewhere).
3. Transit input rate falls substantially within 30-60 s. If it does not, the peer ignores both the community and the prepend (usually a static local-pref on their side): call them, or accept the hit and shut the session with a graceful-restart timer.
4. Ask the peer (or use their looking glass) to confirm they see `65535:0` on your routes.

Do the maintenance only after 2 and 3 hold.

Rollback

End of window (**undrain**, `--mode undrain`): detach the route-maps and soft-clear both ways. The route-maps stay defined for the next window:

```
router bgp 65001
 address-family ipv4 unicast
  no neighbor 198.51.100.1 route-map DRAIN-OUT out
  no neighbor 198.51.100.1 route-map DRAIN-IN in
 exit-address-family
!
! EXEC (not config) - re-advertise and re-evaluate:
! clear ip bgp 198.51.100.1 soft in
! clear ip bgp 198.51.100.1 soft out
```

Full **rollback** (`--mode rollback`) additionally deletes everything `define`/`apply` created: `send-community both` on the neighbors, both route-maps, the `GSHUT` community-list and `ip bgp-community new-format` (set `community_new_format: false` if that was already configured before you started). The running-config then equals the pre-define state:

```
router bgp 65001
 address-family ipv4 unicast
  no neighbor 198.51.100.1 route-map DRAIN-OUT out
  no neighbor 198.51.100.1 route-map DRAIN-IN in
  no neighbor 198.51.100.1 send-community both
 exit-address-family
!
no route-map DRAIN-OUT
no route-map DRAIN-IN
no ip community-list standard GSHUT
no ip bgp-community new-format
! EXEC (not config) - re-advertise and re-evaluate:
! clear ip bgp 198.51.100.1 soft in
! clear ip bgp 198.51.100.1 soft out
```

Then watch the input rate climb back and confirm `show ip bgp summary` prefix counts equal your baseline **before** declaring the window closed.

Gotchas

- Do the `clear ip bgp ... soft in/out` lines, they are exec commands, not config. Without them the policy only applies to future updates.
- `send-community both` is required or `65535:0` never leaves the router.
- The AS number in the prepend must be your own, or you create a confusing path.
- iBGP peers need the `DRAIN-IN` half. Forgetting it is the most common reason "drained" routers keep carrying outbound traffic.
- Some providers honour the community, some do not. The only way to know is to ask or test; this template works either way because it also prepends and sets your own local-pref.
- Keep the route-maps in the config permanently, unapplied. Then the window is two commands and a check.

Ivan Bogović - ibogovic-hub.github.io. Text: CC BY 4.0. Config templates: MIT.

Runbook 2: Stop a volumetric flood with RTBH

Templates	`templates/rtbh_trigger.j2` (trigger router), `templates/rtbh_edge.j2` (edge, toward transit)
Risk	Medium. Blackholing **completes the attack for the victim host**.
Time	Pre-wire once (30 min). Trigger in seconds.
Source	Blog: "Stopping a volumetric flood with RTBH before your upstream picks up the phone"

When to use

A single /32 behind your edge absorbs several Gbit/s and saturates the transit link, degrading everything sharing the circuit. Filtering on your own edge is pointless, because the traffic has already crossed the link. The drop has to happen upstream, in seconds.

RTBH only works if it is **pre-wired and tested before the incident**. This runbook has two parts: A (pre-wire, do now) and B (trigger, do in the incident).

Preconditions

- ☐ Your provider supports RTBH and you have their *current* community in writing (commonly `65535:666` RFC 7999, or provider-specific such as `3356:9999`). Providers change schemes after mergers.
- ☐ Provider accepts /32s from you (their inbound prefix filter permits more-specifics of your space).
- ☐ A business policy exists: which prefixes may be blackholed unilaterally, who authorises out of hours, maximum duration (ours: 30-minute auto-expiry by a scheduled job removing the static route).
- ☐ Trigger device: ideally a small router or Linux BGP daemon, **not** the production edge, with iBGP to the edge.
- ☐ A harmless test prefix agreed with the provider for the dry run.

Configuration template

A1. Pre-wire: trigger router (`rtbh_trigger.j2`, `victims: []`)

```
ip route 192.0.2.1 255.255.255.255 Null0
!
route-map RTBH-TRIGGER permit 10
 match tag 666
 set ip next-hop 192.0.2.1
 set community 65535:666 no-export additive
 set origin igp
route-map RTBH-TRIGGER deny 20
!
router bgp 64512
 neighbor 10.0.0.1 remote-as 64512
 address-family ipv4 unicast
  redistribute static route-map RTBH-TRIGGER
  neighbor 10.0.0.1 activate
  neighbor 10.0.0.1 send-community both
  neighbor 10.0.0.1 next-hop-self
 exit-address-family
!
```

A2. Pre-wire: edge router toward transit (`rtbh_edge.j2`)

Variables: `aggregates`, `blackhole_ranges` (your space from which /32s may be sent), `provider_community`, `transit_neighbor`.

```
ip community-list standard BLACKHOLE-COMM permit 65535:666
ip route 192.0.2.1 255.255.255.255 Null0
!
ip prefix-list TO-TRANSIT-AGG seq 10 permit 203.0.113.0/24
```

```
ip prefix-list TO-TRANSIT seq 10 permit 203.0.113.0/24
ip prefix-list TO-TRANSIT seq 20 permit 198.51.100.0/24 ge 32 le 32
!
route-map TRANSIT-OUT permit 10
 match ip address prefix-list TO-TRANSIT
 match community BLACKHOLE-COMM
 set community 3356:9999 additive
route-map TRANSIT-OUT permit 20
 match ip address prefix-list TO-TRANSIT-AGG
!
router bgp 64512
 address-family ipv4 unicast
  neighbor 203.0.113.9 send-community both
  neighbor 203.0.113.9 route-map TRANSIT-OUT out
 exit-address-family
```

The edge needs its own discard route (`192.0.2.1` to Null0), otherwise the iBGP-learned /32 has an unresolvable next-hop instead of a discard. The `BLACKHOLE-COMM` community-list is defined here; the normal outbound list (`TO-TRANSIT-AGG`) stays aggregate-only so ordinary /32s never leak.

B. Trigger (in the incident)

Set `victims: [198.51.100.77]` and render, or just type:

```
ip route 198.51.100.77 255.255.255.255 Null0 tag 666
```

Verification

> Expected output below is representative of IOS XE 17.x. Wording and columns vary by release and platform; compare the *fields that matter* (called out under each block), not the exact text.

```
R-EDGE# show bgp ipv4 unicast 198.51.100.77/32
BGP routing table entry for 198.51.100.77/32, version 88
  Local
    192.0.2.1 from 10.0.0.9 (10.0.0.9)
      Origin IGP, metric 0, localpref 100, valid, internal, best
      Community: 65535:666 no-export

R-EDGE# show ip cef 198.51.100.77
198.51.100.77/32
  nexthop 192.0.2.1 ... attached to Null0
  (or: receive/drop, depending on release)

R-EDGE# show ip route 192.0.2.1
Routing entry for 192.0.2.1/32
  Known via "static", distance 1, metric 0 (connected)
  Routing Descriptor Blocks:
  * directly connected, via Null0

R-EDGE# show ip bgp neighbors 203.0.113.9 advertised-routes | include 198.51.100.77
 *>  198.51.100.77/32 ...
```

Pass criteria: the /32 is present with the blackhole community; CEF shows a **drop/Null0**, never a real interface; the /32 is in `advertised-routes` to transit; the circuit utilisation falls within a minute. If CEF shows a real interface, the discard route is missing: you have advertised a blackhole that does nothing locally.

Dry-run on a schedule (quarterly) with the test prefix, and save the output above as your known-good reference.

Rollback

Withdraw the blackhole as soon as the attack ends or the agreed maximum duration is reached (`rtbh_trigger.j2 --mode rollback` with `victims` set, or):

```
no ip route 198.51.100.77 255.255.255.255 Null0 tag 666
```

Confirm the /32 is gone from `show bgp ipv4 unicast 198.51.100.77/32` and from the provider's view. To remove the edge pre-wiring entirely (neighbor policy, `TRANSIT-OUT`, both prefix-lists, `BLACKHOLE-COMM`, discard route): `rtbh_edge.j2 --mode rollback`. On the trigger router, `rtbh_trigger.j2 --mode rollback` additionally removes the redistribution, iBGP neighbor, `RTBH-TRIGGER` route-map and discard route (`router bgp` itself stays):

```
router bgp 64512
 address-family ipv4 unicast
  no neighbor 203.0.113.9 route-map TRANSIT-OUT out
  no neighbor 203.0.113.9 send-community both
 exit-address-family
!
no route-map TRANSIT-OUT
no ip prefix-list TO-TRANSIT
no ip prefix-list TO-TRANSIT-AGG
no ip community-list standard BLACKHOLE-COMM
no ip route 192.0.2.1 255.255.255.255 Null0
```

Gotchas

- iBGP must not leak the /32 with `192.0.2.1` to a router lacking the Null0 route, or it becomes unresolvable instead of dropped.
- `no-export` on the trigger keeps the /32 inside your AS toward everything except the explicit transit policy.
- Your normal outbound prefix-list almost certainly caps at the aggregate; the /32 `ge 32 le 32` entry is what lets the blackhole out.
- Make sure `redistribute static` uses the route-map; the discard route itself is untagged and is denied by `RTBH-TRIGGER deny 20`.
- Use 192.0.2.0/24 (TEST-NET-1) only for the discard next-hop; do not reuse it for anything else.
- This is a business decision: you are dropping *all* traffic to the victim to save the circuit.

Ivan Bogović - ibogovic-hub.github.io. Text: CC BY 4.0. Config templates: MIT.

Runbook 3: Sub-second failure detection with BFD and OSPF (IOS XE)

Template	`templates/bfd_ospf.j2` (modes: `apply`, `rollback`)
Risk	Low-medium. A too-aggressive timer causes false flaps.
Time	10 min plus coordination with the far end
Source	Blog: "Fast failure detection with BFD and OSPF on IOS XE"

When to use

WAN circuits that go **grey**: the carrier CPE keeps the Ethernet port up, the router never sees link-down, and OSPF waits for the 40 s dead timer while traffic is blackholed. BFD detects in about 900 ms (300 ms x 3) while OSPF hello timers stay at their conservative defaults. Do **not** instead shrink OSPF hellos: hello processing is control-plane and delays cause adjacency flaps, each triggering SPF in the whole area.

Preconditions

- ☐ The far end will configure BFD too (BFD needs both peers; on a provider-managed circuit, coordinate first).
- ☐ You know whether your platform offloads BFD to hardware. If it runs on the route processor, avoid 50 ms timers on many sessions. 300 x 3 is a sane WAN start.
- ☐ CoPP, if present, has a class that protects UDP 3784 (single-hop), 4784 (multihop), 3785 (echo).
- ☐ Interface is point-to-point (or will be set so) and OSPF process id/router-id are known.

Configuration template

Variables: `interface`, `ospf_pid`, `router_id`, `bfd_interval`, `bfd_min_rx`, `bfd_multiplier`, `bfd_echo`, `per_interface` (true: `ip ospf bfd` on the interface; false: `bfd all-interfaces`), `spf_throttle`, `lsa_throttle`, `copp_acl`.

```
interface GigabitEthernet0/0/1
 description WAN-to-DC2
 ip address 10.255.0.1 255.255.255.252
 ip ospf network point-to-point
 bfd interval 300 min_rx 300 multiplier 3
 no bfd echo
 ip ospf bfd
!
router ospf 10
 router-id 10.0.0.1
 passive-interface default
 no passive-interface GigabitEthernet0/0/1
 timers throttle spf 50 200 5000
 timers throttle lsa 50 200 5000
!
! CoPP: classify BFD into a protected class (attach to your policy-map)
ip access-list extended COPP-BFD
 permit udp any any eq 3784
 permit udp any any eq 4784
 permit udp any any eq 3785
class-map match-any CM-BFD
 match access-group name COPP-BFD
!
```

Echo is off on purpose: echo packets are looped by the peer's forwarding plane and a provider device in the middle may drop or rate-limit packets sourced from your own address. Asynchronous mode is predictable. Attach `CM-BFD` to a protected class in your own CoPP policy-map; that part is site-specific and not templated.

Verification

Expected output below is representative of IOS XE 17.x. Wording and columns vary by release and platform; compare the *fields that matter* (called out under each block), not the exact text.

```
R1# show bfd neighbors details
IPv4 Sessions
NeighAddr         LD/RD    RH/RS     State     Int
10.255.0.2        4097/4098 Up       Up        Gi0/0/1
Session state is UP and not using echo function.
Local Diag: 0, Demand mode: 0, Poll bit: 0
MinTxInt: 300000, MinRxInt: 300000, Multiplier: 3
Rx Count: 1204, Tx Count: 1203
Registered protocols: OSPF

R1# show ip ospf interface GigabitEthernet0/0/1 | include BFD
  BFD enabled

R1# show ip ospf neighbor detail | include BFD
    BFD enabled
```

Pass criteria: state **Up/Up**; `Registered protocols` includes **OSPF**; intervals equal what you designed (BFD negotiates to the *slower* peer, so a mismatched far end silently gives longer detection); OSPF neighbor still FULL. Then prove it: in a window, filter the far end's BFD (or shut the far CPE port) and confirm OSPF drops in about a second.

Rollback

`--mode rollback` removes everything apply set: BFD registration, description, OSPF network type, router-id, SPF/LSA throttles, passive-interface settings and the CoPP class-map/ACL. OSPF keeps running on default timers. The interface IP address is not removed (that would drop the link); if the interface had a different description, network type or router-id before, restore those by hand. IOS answers `no router-id` with an informational `% OSPF: Reload or use "clear ip ospf process"` notice: the configuration line is gone at once, the running OSPF router-id only changes after you clear the process in a window:

```
interface GigabitEthernet0/0/1
 no ip ospf bfd
 no bfd echo
 no bfd interval
 no ip ospf network
 no description
!
router ospf 10
 no timers throttle spf
 no timers throttle lsa
 no passive-interface GigabitEthernet0/0/1
 no passive-interface default
 no router-id 10.0.0.1
!
no class-map match-any CM-BFD
no ip access-list extended COPP-BFD
```

Confirm `show bfd neighbors` is empty for the link and `show ip ospf neighbor` is still FULL.

Gotchas

- **Dampening:** a flapping circuit with BFD flaps fast and every flap triggers SPF. Enable `bfd dampening` where your release supports it (syntax varies; not templated) and keep the SPF/LSA throttles above.
- **CoPP:** a templated policy that drops BFD under load causes exactly the outage BFD should shorten.
- **Both ends:** OSPF runs happily without BFD, so you can believe you are protected when you are not. Always check `show bfd neighbors`.
- `ip ospf bfd` (per interface) beats `bfd all-interfaces` when only some links justify it; `ip ospf bfd disable` opts out.
- Floating static routes toward a provider can use the same mechanism (`ip route static bfd`), often a bigger win at small branches.

Ivan Bogović - ibogovic-hub.github.io. Text: CC BY 4.0. Config templates: MIT.

Runbook 4: Catch a flapping uplink with EEM

Template	`templates/eem_flap_guard.j2` (modes: `apply`, `rollback`)
Risk	Low. Observes and logs; it never shuts the interface.
Time	10 min, plus a lab/window test
Source	Blog: "Catching flapping uplinks with EEM before users do"

When to use

Intermittent link flaps shorter than your monitoring poll interval, long enough to drop VoIP and IPsec. By morning the log buffer has wrapped and all you have is "the phones were weird". EEM runs on the box, sees each syslog event, and captures evidence while the problem is happening: counters, controller errors, optic DOM readings, recent log lines. It then emits one tagged syslog line for your SIEM.

It deliberately does **not** shut the interface: on a single-homed branch that turns a degraded link into a dead one.

Preconditions

- ☐ Flash space available (evidence file is overwritten per incident; small ISRs: check `dir flash:`).
- ☐ You decided on `authorization bypass` with your security team (see Gotchas) or have a local EEM user.
- ☐ You can bounce the interface in a lab or on a redundant link in a window to test.
- ☐ The interface name string below is **copied from `show logging` on this device**, not typed from memory.

Configuration template

Variables: `interface` (full name), `occurs`, `period`, `evidence_file`, `syslog_tag`, `aaa_bypass`, `clear_counters`.

```
event manager environment _flap_if GigabitEthernet0/0/0
!
event manager applet UPLINK-FLAP-GUARD authorization bypass
 event syslog pattern "%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/0/0, changed state to down"
     occurs 3 period 300
 action 0.10 cli command "enable"
 action 0.20 cli command "term length 0"
 action 0.30 cli command "show clock | redirect flash:flap-intf.txt"
 action 0.40 cli command "show interface $_flap_if | append flash:flap-intf.txt"
 action 0.50 cli command "show controllers $_flap_if | append flash:flap-intf.txt"
 action 0.60 cli command "show interfaces $_flap_if transceiver detail | append flash:flap-intf.txt"
 action 0.70 cli command "show logging | include $_flap_if | append flash:flap-intf.txt"
 action 0.80 syslog priority errors msg "UPLINK_FLAP: $_flap_if down 3x in 300s, evidence in flash:flap-intf.txt"
 action 0.90 cli command "clear counters $_flap_if" pattern "confirm"
 action 1.00 cli command "y"
!
```

Tune `occurs`/`period` per site: LTE or satellite backup needs a looser threshold than a metro fibre handoff.

Verification

Never deploy an applet you have not seen fire. In a lab or window, bounce the link three times within 5 minutes:

```
conf t
 interface GigabitEthernet0/0/0
  shutdown
  no shutdown
  shutdown
  no shutdown
  shutdown
  no shutdown
end
```

Expected output below is representative of IOS XE 17.x. Wording and columns vary by release and platform; compare the *fields that matter* (called out under each block), not the exact text.

```
R1# show event manager history events
No.  Job Id Proc Status  Time of Event             Event Type   Name
1    1      Actv success Tue Nov10 02:41:07 2026   syslog       applet: UPLINK-FLAP-GUARD

R1# show logging | include UPLINK_FLAP
%HA_EM-3-FMPD_...: UPLINK_FLAP: GigabitEthernet0/0/0 down 3x in 300s, evidence in flash:flap-intf.txt
   (the exact facility prefix varies by release; the UPLINK_FLAP: text is what your SIEM matches)

R1# more flash:flap-intf.txt
(clock, show interface, controllers, transceiver detail, matching log lines)
```

Pass criteria: the applet is in the history with `success`; the syslog line appears; the evidence file contains all sections. If the applet did not fire, the cause is nearly always the syslog pattern (interface name string, subinterface suffix, release wording): copy it from the device's own `show logging`.

Rollback

```
no event manager applet UPLINK-FLAP-GUARD
no event manager environment _flap_if
```

(`--mode rollback` renders exactly this.) Optionally `delete flash:flap-intf.txt`. Confirm `show event manager policy registered` no longer lists the applet.

Gotchas

- `occurs 3 period 300` does the real work; one down event is noise.
- `authorization bypass` keeps the applet from failing when AAA command authorization is on and TACACS is unreachable (likely during a flap). Check against your AAA policy; some teams want a dedicated local EEM user instead.
- First action `redirect`, rest `append`: one evidence file per incident, previous overwritten. For history build the filename from a timestamp.
- Clearing counters at the end means the next capture shows deltas, but some NMS pollers handle counter resets badly: set `clear_counters: false` then.
- The transceiver command differs by platform (`show interfaces transceiver detail`, `show hw-module subslot ... transceiver ... status`). Test it by hand first.
- Push the applet with the same config management you already use so it is versioned.

Ivan Bogović - ibogovic-hub.github.io. Text: CC BY 4.0. Config templates: MIT.

Runbook 5: Fail over a static default route when the ISP breaks upstream

Template	`templates/ip_sla_failover.j2` (modes: `apply`, `rollback`)
Risk	Medium. Changes your default route behaviour.
Time	20 min plus a failover test
Source	Blog: "Failing over a static default route when the ISP breaks upstream"

When to use

Branch router, fibre primary plus LTE/DSL backup as floating statics. When the provider fails somewhere *past* the handoff, the interface stays up/up, the CPE still answers ARP, the primary static stays in the table, and the backup never installs. IP SLA probes something beyond the provider edge, a track object turns that into up/down, and the primary route is tied to the track.

Preconditions

- ☐ Primary next-hop and interface known; backup interface and its distance (must be higher than 1).
- ☐ Two independent probe targets **beyond** the provider edge, which your users do not otherwise rely on (they will be pinned to the primary path), or addresses you own. Do not probe the provider next-hop: that tests only the handoff.
- ☐ If the uplink is in a front-door VRF, set `vrf` (probes and pinned routes go into it).
- ☐ You can inject a failure (ACL test below) during a window.
- ☐ LTE data cap understood: probes run on the primary only.

Configuration template

Variables: `primary_interface`, `primary_nexthop`, `backup_interface`, `backup_distance`, `targets`, `sla_base`, `track_list_id` (must not collide with `sla_base + n`), `frequency`, `timeout_ms`, `delay_down`, `delay_up`, `vrf`, `alert_applet`, `alert_message`.

```
ip sla 10
 icmp-echo 9.9.9.9 source-interface GigabitEthernet0/0/0
 frequency 10
 timeout 2000
 threshold 2000
ip sla schedule 10 life forever start-time now
!
ip sla 11
 icmp-echo 1.1.1.1 source-interface GigabitEthernet0/0/0
 frequency 10
 timeout 2000
 threshold 2000
ip sla schedule 11 life forever start-time now
!
track 10 ip sla 10 reachability
 delay down 30 up 60
track 11 ip sla 11 reachability
 delay down 30 up 60
!
track 20 list boolean or
 object 10
 object 11
!
ip route 0.0.0.0 0.0.0.0 GigabitEthernet0/0/0 203.0.113.1 track 20 name PRIMARY
ip route 0.0.0.0 0.0.0.0 Cellular0/1/0 250 name BACKUP
! Pin probe targets to the primary path so probes never follow the backup
ip route 9.9.9.9 255.255.255.255 GigabitEthernet0/0/0 203.0.113.1
ip route 1.1.1.1 255.255.255.255 GigabitEthernet0/0/0 203.0.113.1
!
event manager applet TRACK20-DOWN
 event track 20 state down
 action 1.0 syslog priority errors msg "Primary uplink failed reachability, default route moved to LTE"
```

`delay down 30 up 60` is deliberately asymmetric: leave quickly, return only after a full stable minute. The boolean OR means *one* target answering keeps you on the primary, so one anycast resolver having a bad day does not push the branch onto LTE. The pinned host routes make the probes measure **only** the primary path; without them the default moves to LTE, probes succeed over LTE, the track comes back up, and the router flaps every minute.

Verification

Check before the outage, not during.

> Expected output below is representative of IOS XE 17.x. Wording and columns vary by release and platform; compare the *fields that matter* (called out under each block), not the exact text.

```
R1# show ip sla statistics 10
IPSLAs Latest Operation Statistics
IPSLA operation id: 10
        Latest RTT: 11 milliseconds
Latest operation return code: OK
Number of successes: 1840
Number of failures: 0

R1# show track brief
Track  Type          Instance        Parameter   State  Last Change
10     IP SLA        10              reachability Up    2d03h
11     IP SLA        11              reachability Up    2d03h
20     List          boolean or                  Up     2d03h

R1# show ip route static
S*    0.0.0.0/0 [1/0] via 203.0.113.1, GigabitEthernet0/0/0
```

Failover test (no provider call needed): on the primary interface add an inbound ACL that drops ICMP echo-replies from the two probe targets.

1. Block **one** target: tracks 10 or 11 go Down, track 20 stays **Up**, default stays on fibre (proves the OR).
2. Block **both**: tracks 10, 11 go Down, track 20 goes Down after about 30 s, `PRIMARY` leaves `show ip route`, the backup default installs (`S* 0.0.0.0/0 [250/0] via Cellular0/1/0`), and your syslog shows the alert text.
3. Remove the ACL: primary returns about 60 s later.

`debug track` is available but use it sparingly on production.

Rollback

`--mode rollback` removes the tracked route, pinned host routes, track objects, SLA operations and the alert applet. The floating backup route is removed as well, so the running-config returns to the pre-apply state; make sure another default route exists, or set `keep_backup_route: true` to leave it in place:

```
no event manager applet TRACK20-DOWN
no ip route 0.0.0.0 0.0.0.0 GigabitEthernet0/0/0 203.0.113.1 track 20
no ip route 9.9.9.9 255.255.255.255 GigabitEthernet0/0/0 203.0.113.1
no ip route 1.1.1.1 255.255.255.255 GigabitEthernet0/0/0 203.0.113.1
no track 20
no track 10
no ip sla schedule 10
no ip sla 10
no track 11
no ip sla schedule 11
no ip sla 11
no ip route 0.0.0.0 0.0.0.0 Cellular0/1/0 250
```

Re-add a plain `ip route 0.0.0.0 0.0.0.0 <primary-interface> <primary-nexthop>` at the same time if you removed the tracked primary and no other default exists. Verify with `show ip route 0.0.0.0`.

Gotchas

- `source-interface` alone does not stop probes following the default route on every platform/release; the forwarding decision comes from the routing table, hence the pinned host routes. The cost is that those two addresses

are unreachable over backup.
- **NAT:** with PAT per uplink (`ip nat inside source route-map`), existing translations survive failover and keep pointing at the dead path. Clear them on the track event or keep timeouts short.
- Failing outbound over does not move **inbound** services: they arrive on whichever public IP DNS points to.
- **VRF:** set `vrf`; otherwise the probe runs in the global table.
- Do not call the site redundant until you have run the ACL test.

Ivan Bogović - ibogovic-hub.github.io. Text: CC BY 4.0. Config templates: MIT.