Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 28 November 2026

L. Melegassi
Catellix

27 May 2026

MVPS Performance-Security Coupling Profile: Joint Volume-Independence and Authentication Guarantees for Coherence-BFD with Coherent-Witness Trust (CWT)

draft-melegassi-mvps-perfsec-coupling-00

Abstract

This document specifies the MVPS Performance-Security Coupling Profile, a Profile-of-Profiles binding three previously specified profiles into a single deployable contract:

- MVPS Coherent-Witness Trust (CWT) [I-D.melegassi-santos-ippm-mvps-cwt],
- Coherence-BFD [I-D.melegassi-coherence-bfd], and
- MVPS DDoS Resilience [I-D.melegassi-mvps-ddos-resilience].

Each composed profile proves its own theorems and reports its own measured numbers under its own scale assumption. When deployed together, those scale assumptions diverge by 1-2 orders of magnitude and create five composition holes: a numerical cost-rescaling gap, a double replay-counter rule, an under-specified key-derivation step, an insider verification-DoS gap, and a joint Byzantine-at-limit / collector-split-view gap.

This profile closes all five with three theorems:

T-JCOST-1. Closed-form joint broker CPU cost as a function of (N, T_tick, q, bundle_period); two-path decomposition (CWT path + Coherence-BFD path) avoids double-count; numerical receipt at four scale points.

T-VDOS-1. Per-vantage rate-limit at NIC/XDP fast-path bounds the attacked-broker CPU by a near-constant factor (rate_limit_factor + flood * c_xdp / c_path) instead of the linear blow-up of the unprotected case.

T-RC-1. Acceptance is the AND of the BFD sequence rule and the CWT counter rule; rejection cases are enumerated.

A normative HKDF info-string for cross-profile key separation closes the under-specification of CWT Section 13. The dual-mode aggregation values (D_minimax, D_max) defined in DDoS-Resilience Section 7.2 are bound INTO the cosigned checkpoint of CWT Section 8.2, closing the joint Byzantine + split-view gap.

The profile inherits the full v4.0 theorem catalogue and is conformant to the MVPS Architecture Invariance Theorem [I-D.melegassi-iab-mvps-architecture]. Full proofs are recorded in [PERFSEC-PROOF]; numerical receipts are in evidence/perfsec_joint_cost_receipt.json and evidence/perfsec_verification_dos_receipt.json; the validator scripts/validate_perfsec_coupling.py returns 12/12 PASS.

Status of This Memo

Copyright Notice

Table of Contents

Author's Address

1. Introduction

The MVPS framework has produced three operationally-deep profiles that share the same vantage-broker substrate but were specified in isolation:

- CWT [I-D.melegassi-santos-ippm-mvps-cwt] specifies trust (HMAC-personalized snapshots, Operator Epoch Manifest, witness cosignatures). Section 14.1 reports a 2.10 us per-snapshot HMAC cost and a derived 0.21 % core load at N = 1000 vantages, 1 Hz tick.

- Coherence-BFD [I-D.melegassi-coherence-bfd] specifies sub-second coherence detection. Variant V3 (Echo) operates at T_tick = 50 ms and is the latency-optimal reference. Section 15.3 dimensions the broker NIC at four PPS regimes A-D.

- DDoS Resilience [I-D.melegassi-mvps-ddos-resilience] specifies volume-independent detection at attack rates from 10 Mpps to 5 Tbps equivalent, on top of Coherence-BFD, under three architectural invariants (Section 3 of that document).

Each draft proves its own theorems and reports its own measured numbers under its own scale assumption. The numbers are correct in isolation. When deployed together (which is the realistic deployment), the scale assumptions diverge and produce composition holes that each individual draft cannot see.

1.1. The composition problem (the "sec drops PPS" question)

CWT Section 14.1 cites 0.21 % of one core for HMAC overhead. That number is computed at the abstract reference scale assumption stored in evidence/mvps_cwt_overhead_receipt.json:

```
n_vantages = 1000,   tick_hz = 1,   snaps_per_sec = 1000
```

But Coherence-BFD V3 operates at T_tick = 50 ms (= 20 Hz), which raises broker ingress PPS from 1 000 to 20 000 at the same N. DDoS Resilience Section 6 evaluates N = 10 000, 8 cells, T_tick = 50 ms, which raises ingress PPS to 200 000.

The crypto cost scales linearly with PPS. Worse, in a full-stack deployment the broker pays HMAC TWICE: once on the BFD AuthHMAC path (UDP control packet, per push, Section 4.2 of Coherence-BFD) and once on the CWT path (snapshot in the MVPS bundle, per snapshot, Section 6.4 of CWT). These are distinct packets at the same tick rate; HMAC is paid on each independently.

At Regime C (N = 10 000, T_tick = 50 ms) the joint cost (parser + HMAC_CWT + HMAC_BFD + witness verify) is approximately 174 % of one core, i.e., the broker requires 1.74 dedicated CPU cores BEFORE any coherence algebra.

The headline metric is the apples-to-apples scaling within the joint formula: the same JOINT(N, T_tick) at the CWT abstract reference scale (N = 1000, 1 Hz, full stack) is 0.88 % of one core. Going to Regime C (N = 10 000, T_tick = 50 ms) raises it to 174.25 %. The all-else-equal scaling factor is therefore

```
JOINT_RegimeC / JOINT_CWT_ref  =  174.25 / 0.88  ~=  198x
```

i.e., a Tier-1 deployment requires approximately 200 times the broker CPU budget that the CWT abstract reference suggests. An operator dimensioning the broker by reading ONLY Section 14.1 of [I-D.melegassi-santos-ippm-mvps-cwt] (which reports the HMAC-only single-axis figure of 0.21 % at the same reference scale) and missing the parser, BFD, and witness contributions would under-provision by an additional factor of ~ 4.2x; the worst-case reading error is therefore ~ 830x. Section 3 formalizes the closed-form joint cost as Theorem T-JCOST-1.

1.2. Five composition holes

A full enumeration is documented in the Performance-Security Coupling Audit (docs/MVPS_PERFSEC_COUPLING_AUDIT.txt). The five holes addressed by this profile are:

H1. Cost rescaling. CWT 14.1 numbers are valid at 1 Hz, but Coherence-BFD operates at T_tick = 50 ms (20 Hz) and DDoS Resilience pushes to Regimes C-D. No draft documents the joint scaling; operators sizing from any one draft alone underprovision (Section 3).

H2. Replay double-counter. Coherence-BFD Section 12 and DDoS Resilience Section 2.4 enforce monotonic BFD seq numbers; CWT Section 9 introduces a per-(vantage, epoch) counter that resets at each epoch boundary. No draft specifies the joint acceptance rule (Section 5).

H3. HKDF under-specification. CWT Section 13 mandates a "distinct HKDF label" for the Coherence-BFD HMAC key but does not normalize the info-string format, salt convention, key length, or rotation cadence link (Section 6).

H4. Insider verification-DoS. Architectural invariant I1 [I-D.melegassi-mvps-ddos-resilience] blocks external attackers from the broker NIC. An insider with valid CWT and BFD keys (compromised vantage at root) can flood the broker at any PPS; CWT Section 9 enforces only counter monotonicity, not a maximum increment rate. No draft specifies a per-vantage rate-limit at the broker (Section 4).

H5. Joint Byzantine + split-view. DDoS Resilience Theorem D2 Case 2 (perfect Byzantine hiding) and CWT Theorem T-SPLIT-1 (collector split-view) compose into a joint failure mode where a Byzantine collector chooses which consumers see the Byzantine-included view (alarm) and which see the Byzantine-excluded view (silent). No draft addresses this jointly (Section 7).

1.3. Conventions and definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following symbols are used throughout:

N	Number of admitted vantages.
T_tick	Control-tick period (in milliseconds).

M	Detection multiplier (Coherence-BFD).
q	Witness quorum (CWT, default 2).
k	Number of coherence cells.
PPS_t	Telemetry packets-per-second per path at the broker = N * (1000 / T_tick_ms).
c_json	Median JSON-decode latency, microseconds.
c_bfd_parse	Median BFD binary-parse latency, microseconds.
c_hmac	Median HMAC-SHA256 verify latency, microseconds.
c_ed25519	Median Ed25519 verify latency, microseconds.
c_xdp	Median per-packet XDP/eBPF rate-limit lookup cost, microseconds.

Constant values are pinned by the receipt evidence/mvps_cwt_overhead_receipt.json:

```
c_json       = 4.20 us
c_bfd_parse  = 0.30 us  (binary, ~ 1/14 of JSON-decode cost)
c_hmac       = 2.10 us
c_ed25519    = 78.80 us
c_xdp        = 0.05 us  (eBPF/XDP per-packet hash-map
                         lookup; see Section 4.1 for
                         derivation)
```

2. Joint Threat Model

2.1. Inherited from composed drafts

This profile inherits the threat models of all three composed drafts:

- Adversary capabilities (i)-(j) of CWT Section 4.1 (compromise, impersonation, Sybil, collector compromise, MITM, replay, parser DoS, gauge gap, multi-operator coalition, split-view collector).
- Threat classes 2.1-2.4 of DDoS Resilience (volumetric, distributed multi-region, control-plane targeted, replay and TLV spoofing).
- AuthHMAC and replay assumptions of Coherence-BFD Section 12.

No assumption from the inherited models is relaxed. The two classes below are NEW and arise only at the composition layer.

2.2. T-INSIDER-FLOOD (insider verification-DoS)

Adversary capabilities:

- A vantage host is compromised at root level. The attacker holds the vantage's K_v_epoch (CWT Layer 1) and the vantage's BFD AuthHMAC key.
- The attacker emits VALID-tagged packets at rates far above the vantage's natural tick rate. Example: at T_tick = 50 ms the natural rate is 20 packets/s per path; the attacker pushes 100 000 packets/s per path.

Defense state in the existing drafts:

- CWT Section 9 enforces strict monotonicity of counter. A flood with monotonically increasing counters PASSES.
- Coherence-BFD Section 12 enforces strict monotonicity of

BFD sequence numbers. A flood with monotonic seq numbers PASSES.

- DDoS Resilience Section 8 inherits Coherence-BFD security and explicitly does NOT contemplate insider flood (architectural invariant I1 protects against EXTERNAL attackers only).

Result without this profile: a single compromised vantage at 100 000 packets/s consumes

```
     100 000 * (c_json + c_bfd_parse + 2 * c_hmac)
   = 100 000 *  8.7 us
   = 870 ms/s = 87 % of one core.
```

Ten compromised vantages saturate one core; one hundred saturate ten cores. The broker is approximately denied service before any external attacker reaches it.

Mitigation specified by this profile: per-vantage rate-limit at NIC/XDP fast-path, BEFORE HMAC verification (Section 4).

2.3. T-COMP-SPLIT (Byzantine-at-limit + collector split-view)

Adversary capabilities:

- Adversary holds floor((k - 1) / 2) Byzantine cells (the DDoS Resilience Theorem D2 Case 2 frontier; for k = 8, three Byzantine cells are sufficient).
- Adversary holds the collector key (the CWT T-SPLIT-1 frontier).

Defense state in the existing drafts:

- DDoS Resilience Section 7.2 specifies dual-mode aggregation (D_minimax and D_max reported separately) but BOTH values are computed and published by the collector. A compromised collector can publish either set to anyone.
- CWT Theorem T-SPLIT-1 detects split-view via witness cosignatures over the bundle Merkle root. But the witness sees only the bundle contents the collector submits; the dual-mode values are NOT bound into the cosigned object in the current CWT specification.

Mitigation specified by this profile: dual-mode aggregation values (D_minimax, D_max) are bound into the cosigned checkpoint "extra" field of CWT Section 8.2 (Section 7 of this document), so that any split-view attempt forces a witness mismatch on either dimension, exactly as if the bundle root had differed.

3. Theorem T-JCOST-1 (Joint Cost Bound)

3.1. Two-path architecture

The composed deployment runs TWO independent authenticated paths between each vantage and the broker, at the same tick rate but with different protocols:

```
Path A (CWT, snapshots in MVPS bundle, TCP):
   per-snapshot cost  =  c_json + c_hmac
   ops per second     =  N / T_tick_s

Path B (Coherence-BFD, UDP control packets):
```

```
   per-push cost      =  c_bfd_parse + c_hmac
   ops per second     =  N / T_tick_s

Path C (Witness, Ed25519 cosignatures over bundle Merkle root,
        amortized across many snapshots per bundle):
   per-bundle cost    =  q * c_ed25519
   ops per second     =  cells_k / (bundle_period_ticks *
                                    T_tick_s)
```

Path A and Path B are DISTINCT packets. The broker pays HMAC on each independently; HMAC is therefore counted twice without double-counting any individual operation. Path C is amortized across bundle_period_ticks snapshots and contributes a small fixed cost.

3.2. Closed form

THEOREM T-JCOST-1 (Joint Cost Bound).
Let N, T_tick (ms), q, bundle_period_ticks, cells_k be the deployment parameters. The joint broker CPU cost (microseconds of CPU per second of wall-clock) is:

```
   JOINT(N, T_tick, q)
      =   PPS_t * (c_json + c_hmac)         [Path A]
        + PPS_t * (c_bfd_parse + c_hmac)    [Path B]
        + bundles_per_sec * q * c_ed25519   [Path C]
```

where:

```
   PPS_t           = N * (1000 / T_tick_ms)
   bundles_per_sec = cells_k /
                     (bundle_period_ticks * T_tick_s)
```

The bound is INDEPENDENT of the attack rate R (Theorem D3 of [I-D.melegassi-mvps-ddos-resilience] is inherited under invariants I1, I2, I3).

PROOF (sketch; full proof in [PERFSEC-PROOF] Section 2).
By I1, the broker NIC ingests only telemetry; per D3 the NIC PPS is N * (1000 / T_tick_ms) regardless of R. Path A parser+HMAC and Path B parser+HMAC are summed once per packet each, on disjoint packet streams. Path C cosignature verifications occur once per bundle; the bundle rate is bounded above by cells_k cell-coordinator emissions per bundle_period_ticks * T_tick_s. Sum is the closed form. QED.

3.3. Numerical receipt at four scale points

Receipt: evidence/perfsec_joint_cost_receipt.json
Validator: scripts/validate_perfsec_coupling.py (12/12 PASS)

With the constants pinned in Section 1.3 and parameters (q = 2, bundle_period_ticks = 10, cells_k = 8):

Scenario	N	T_tick	PPS_t	JOINT
CWT abstract ref	1000	1000ms	1 000	0.88 % core
Coh-BFD V3 minimal	1000	50ms	20 000	17.65 % core
DDoS Regime C	10000	50ms	200 000	174.25 % (~2c)
DDoS Regime D	100000	50ms	2 000 000	1740.25 % (~18c)
HFT/orbital	10000	5ms	2 000 000	1742.52 % (~18c)

(1) "% one core" values <= 100 fit within one core.

Headline (DDoS Regime C, two interpretations).

(a) APPLES-TO-APPLES scaling within the joint formula (LEAD METRIC):

```
JOINT_RegimeC      =  174.25 % of one core (~ 1.74 cores)
JOINT_CWT_ref      =    0.88 % of one core (same formula
                                            at N=1k, 1 Hz)
All-else-equal scaling factor = 174.25 / 0.88 ~= 198x
```

A Tier-1 Regime C deployment requires approximately 200 times the broker CPU budget that the CWT abstract reference implies.

(b) WORST-CASE reading error (single-axis HMAC-only figure of Section 14.1 of [I-D.melegassi-santos-ippm-mvps-cwt], 0.21 % at N=1k / 1 Hz):

```
JOINT_RegimeC / 0.21 % ~=  830x
```

This is the worst-case under-provisioning IF the operator reads only the HMAC-only figure and ignores parser, BFD-auth, and witness contributions. This profile's joint formula eliminates the worst-case reading error.

The LEAD metric for operator dimensioning is (a). (b) is reported only to make explicit the failure mode this profile prevents.

3.4. Operator dimensioning rule

Implementations conformant to this profile MUST dimension the broker CPU+NIC budget from JOINT(N, T_tick, q) of Section 3.2, not from any single-axis cost cited in any composed draft.

At Regime D (PPS_t >= 1 Mpps) software-only HMAC will not meet the bound. Implementations targeting Regime D MUST use HMAC hardware acceleration (Intel QAT, ARMv8 Crypto Extensions) or kernel-bypass crypto (DPDK + cryptodev), and SHOULD declare the acceleration in the management interface.

4. Theorem T-VDOS-1 (Verification-DoS Bound)

4.1. Rate-limit at NIC/XDP fast-path

To bound the verification-DoS surface (T-INSIDER-FLOOD, Section 2.2), brokers conformant to this profile MUST implement per-vantage rate-limit BEFORE invoking HMAC verification. The rate-limit MUST be enforced in the NIC fast-path (XDP/eBPF on Linux, equivalent on other OSes) so that dropped packets pay only the lookup cost c_xdp (~ 0.05 us), not the verification cost c_hmac (~ 2.10 us).

DERIVATION OF c_xdp (~ 0.05 us). The XDP fast-path executes a single hash-map lookup keyed on the per-vantage rate-limit bucket, plus an atomic decrement. Published measurements of XDP per-packet processing on commodity x86 hardware [XDP-PERF] [CILIUM-XDP] report 30-60 ns for hash-map-and-counter idioms on Intel Xeon E5-class cores at ~ 30 Mpps single-core throughput; AF_XDP and DPDK numbers are tighter still. This document adopts c_xdp = 50 ns as a conservative working value; deployments targeting Regime D MUST measure c_xdp on their own hardware and adjust the T-VDOS-1 bound accordingly. The receipt evidence/perfsec_verification_dos_receipt.json records c_xdp alongside the joint-cost constants for reproducibility.

Rate-limit decisions on incoming packets:

1. Per-vantage token-bucket lookup keyed on the (operator_id, vantage_id) pair extracted from the packet outer header (CWT mvps_cwt wrapper or Coherence-BFD My Discriminator mapping).
2. If token available, decrement and forward to verifier.
3. Else, drop packet, record metric, no verifier call.

4.2. Token-bucket parameters

Defaults (operator MAY tighten for HFT/orbital; MAY relax with explicit risk acceptance):

```
natural_tick_rate(v) = 1000 / T_tick_ms   [packets per second
                                           per path]
rate_limit_factor    = 4                  (REQUIRED >= 2)
burst_factor         = 8                  (REQUIRED >= 2)
rate_limit_pps(v)    = rate_limit_factor * natural_tick_rate(v)
burst_budget(v)      = burst_factor * natural_tick_rate(v)
```

JUSTIFICATION OF rate_limit_factor = 4 (DEFAULT). The factor MUST leave operational headroom for three distinct sources of legitimate short-term burst above the natural tick rate:

- CLOCK JITTER. OS scheduling jitter at sub-millisecond T_tick produces transient bursts of approximately 1.2-1.5x the natural rate when a vantage's previous tick is delayed and the next two ticks fire close together. Observed on commodity Linux at T_tick = 50 ms with PREEMPT_RT off.

- BGP UPDATE BATCHES. At BGP session establishment or after a topology event (peer flap, RIB sync), a vantage observing a multi-prefix announcement burst can legitimately produce 2-3x its natural rate for one to a few seconds, as cell-coordinator buffering empties. Documented in operational RIPE Atlas measurements of full-feed peers.

- RECALIBRATION WINDOW. Coherence-BFD Section 17 specifies a recalibration procedure that collects 600 ticks of BAU samples; during recalibration the vantage MAY emit additional informational packets, contributing up to ~ 1.3x the natural rate.

The factor 4 covers the worst case 1.5 * 3 * 1.3 ~= 5.85x with a conservative rounding down to 4 plus burst budget (Section 4.2). Operators with deterministic real-time platforms (PREEMPT_RT, DPDK pollers, hardware-timestamped probes) MAY set rate_limit_factor = 2 and burst_factor = 2 in steady state at the cost of false-rejecting the larger transient bursts above.

Operators with very long T_tick (>= 1 s) MAY relax the factor further, since absolute jitter shrinks relative to the natural period.

4.3. Closed-form bound and proof

THEOREM T-VDOS-1 (Verification-DoS Bound).
Under the rate-limit policy of Section 4.1 with parameters rate_limit_factor (>= 2) and burst_factor (>= 2), an insider adversary controlling a coalition of admitted vantages and pushing at flood_factor times the natural rate cannot drive the broker CPU above:

```
   JOINT_attacked(flood_factor)
      <= JOINT_baseline *
         ( rate_limit_factor +
           flood_factor * (c_xdp / c_path) )
```

where c_path = c_json + c_bfd_parse + 2 * c_hmac (sum of Path A + Path B per (snapshot, push) pair).

In particular:

(a) At low flood factors (flood_factor <= rate_limit_factor), the broker CPU saturates at rate_limit_factor * JOINT_baseline. The cost ratio does not depend on flood_factor.

(b) At high flood factors, the broker CPU grows SUBLINEARLY in flood_factor with slope c_xdp / c_path. At c_xdp = 0.05 us, c_path = 8.7 us, slope is 0.0057 - i.e., a 1024x flood adds ~ 5.9 to the ratio.

Without the rate-limit, the ratio scales LINEARLY with flood_factor (slope 1, unbounded).

PROOF (full proof in [PERFSEC-PROOF] Section 3).
Let P = N * (1000 / T_tick_s) = baseline PPS per path.
Per-vantage at attack:

```
accepted_pps       <= rate_limit_factor * (1 / T_tick_s)
                      + (burst_factor / duration_s)
                      ~~ rate_limit_factor * (1 / T_tick_s)
dropped_pps         = (flood_factor - rate_limit_factor) *
                      (1 / T_tick_s)
```

Per-vantage cost:

```
cost(v) = accepted_pps * c_path + (accepted_pps + dropped_pps)
          * c_xdp
```

Sum over N vantages:

```
JOINT_attacked = N * cost(v)
```

Substitute and divide by JOINT_baseline = P * c_path:

```
ratio = rate_limit_factor + flood_factor * (c_xdp / c_path)
```

QED.

4.4. Numerical receipt

Receipt: evidence/perfsec_verification_dos_receipt.json
Validator: scripts/validate_perfsec_coupling.py (T-VDOS-1)

At N = 1 000, T_tick = 50 ms, rate_limit_factor = 4, burst_factor = 8:

flood	ratio (no RL)	ratio (with RL)	bound predicted
1	1.0	1.0	4.01
4	4.0	4.0	4.02
16	16.0	4.20	4.09
64	64.0	4.48	4.37
256	256.0	5.57	5.46
1024	1 024.0	10.0	9.89

```
ratio = JOINT_attacked / JOINT_baseline (baseline = 17.5 % core)
```

Without rate-limit, a flood factor of 64 already saturates 11.2 cores; with rate-limit, the same flood saturates only 0.78 core. Without rate-limit, a flood factor of 1024 saturates 179 cores; with rate-limit, it saturates 1.74 cores - the same level as the un-attacked Regime C deployment.

The qualitative claim "rate-limit converts a linear blow-up into a near-constant + sublinear bound" is therefore numerically confirmed.

5. Theorem T-RC-1 (Replay-Counter Coherence)

The composed system carries TWO independent replay counters with different semantics:

- BFD seq: per-(BFD session), monotonic, NEVER resets within M * T_tick (Coherence-BFD Section 12).
- CWT counter: per-(vantage_id, epoch_id), monotonic within an epoch, RESETS to 0 at every epoch_id increment (CWT Section 9.1).

THEOREM T-RC-1 (Replay-Counter Coherence).
A snapshot (or a Coherence-BFD push) is ACCEPTED at the broker if and only if:

```
(a)  bfd_seq    > last_bfd_seq[vantage]               AND
(b)  cwt_counter > last_cwt_counter[vantage, epoch_id] AND
(c)  expires_at > now()
```

Failure of any predicate causes immediate rejection without progress on N, C_1, C_2, C_3 of the underlying detector. Rejection MUST be logged with reason code (REQUIRED for joint auditability):

```
REJECT_REASONS = {
   "bfd-replay"     : (a) failed,
   "cwt-replay"     : (b) failed,
   "double-replay"  : (a) AND (b) failed,
   "stale"          : (c) failed,
   "epoch-mismatch" : OEM does not bind (vantage, epoch_id)
}
```

PROOF (full proof in [PERFSEC-PROOF] Section 4).
Acceptance is the AND of two unforgeable predicates (under HMAC PRF of [RFC2104] for both sides of the AND). Forgery of either predicate requires key compromise, which is excluded by H-TRUST-CWT and the Coherence-BFD Section 12 AuthHMAC hypothesis. Both predicates ACCEPT exactly the legitimate honest stream; their AND therefore ACCEPTS exactly the legitimate honest stream. QED.

EPOCH BOUNDARY (NORMATIVE). At the boundary epoch_id -> epoch_id+1, the broker:

- Initializes last_cwt_counter[vantage, epoch_id+1] := -1 (CWT counter resets per Section 9.1).
- KEEPS last_bfd_seq[vantage] (BFD seq does NOT reset).
- Accepts BOTH old-epoch and new-epoch snapshots during the CWT-defined overlap window (CWT Section 7.3).

Cross-epoch replay (replaying an old-epoch snapshot under the

old epoch_id while the new epoch is current) is rejected by (a) because the BFD seq does not reset. Cross-epoch forge (claiming the new epoch_id with an old BFD seq) is rejected by (a) for the same reason.

The validator scripts/validate_perfsec_coupling.py implements the acceptance predicate (T-RC-1.A) and the epoch-boundary case (T-RC-1.B), both PASS.

6. HKDF Specification for Cross-Profile Keys (closes CWT 13)

CWT Section 13 mandates a "distinct HKDF label" for the Coherence-BFD HMAC key but does not normalize the info-string format, salt convention, key length, or rotation cadence link. This section closes that gap normatively.

When CWT and Coherence-BFD share an operator master key K_op (typical, to simplify rotation), the BFD AuthHMAC key MUST be derived as:

```
K_BFD_auth = HKDF-SHA256(
   salt   = "MVPS-PerfSec-v1|operator:" || operator_id,
   ikm    = K_op,
   info   = "coherence-bfd-auth-v1|session:" || session_id
            || "|epoch:" || ascii(epoch_id),
   length = 32 octets )
```

where:

operator_id	matches the CWT Section 5.4 operator_id.
session_id	is the BFD Discriminator pair canonicalized lexicographically as 16 hex characters (8 octets) "smaller||larger".
epoch_id	matches the CWT Layer 1 epoch_id; BFD does not natively have epochs, so this binds the BFD-auth key lifetime to the CWT epoch lifetime.

NORMATIVE CONSTRAINTS:

- K_BFD_auth MUST be exactly 32 octets.
- K_BFD_auth MUST be derived independently of K_v_epoch (the distinct HKDF salt and info ensure independence under HKDF-SHA256 [RFC5869] PRF security).
- Rotation of K_op (CWT Section 5.5: SHOULD every 90 days) IMPLIES immediate re-derivation of K_BFD_auth. The two rotations are atomic.
- This SUPERSEDES the BFD-auth-only rotation cadence "SHOULD every 30 days" of [I-D.melegassi-coherence-bfd] Section 12 in deployments adopting this profile; the operator MAY rotate K_BFD_auth more frequently than K_op by introducing a counter into the info string, but MUST NOT rotate it less frequently.
- Deployments NOT sharing K_op MAY use the legacy Coherence-BFD AuthHMAC key derivation; the conformance flag of Section 8 then signals a mixed deployment.

The validator scripts/validate_perfsec_coupling.py (check T-HKDF-PS) verifies independence of K_v_epoch and K_BFD_auth under shared K_op, including per-session separation.

7. Joint Composition: Byzantine + Split-View

DDoS Resilience Section 7.2 specifies dual-mode aggregation:

```
   D_minimax^2  : with B_assumed worst cells removed
   D_max^2      : standard max over ALL cells
```

These two values MUST be carried INSIDE the cosigned checkpoint "extra" field of CWT Section 8.2 in the format:

```
   bundle_id=<uuid>;bundle_seq=<uint64>;
   d_minimax=<float>;d_max=<float>
```

The witness, on receiving the checkpoint via POST /add-checkpoint per [C2SP-TLOG-WITNESS], verifies the extra field schema and cosigns the checkpoint bytes including the dual-mode values.

CONSEQUENCE. Theorem T-COMP-SPLIT.

A split-view collector that publishes different (D_minimax, D_max) pairs to different consumers under the same (bundle_id, bundle_seq) forces witnesses to sign different checkpoint bodies. An honest witness, applying the standard "first-cosign-wins per (bundle_id, bundle_seq)" policy of [C2SP-TLOG-WITNESS], refuses the second cosignature request, and any auditor querying any honest witness within one checkpoint window detects the inconsistency.

The split-view detection probability inherits T-SPLIT-1 of CWT (1 - (1 - 1/w)^q for q honest cosignatures of w witnesses) extended to detect mismatch on D_minimax or D_max in addition to root_hash.

PROOF (sketch; full proof in [PERFSEC-PROOF] Section 5).

CWT T-SPLIT-1 already proves split-view detection on the bundle root_hash. By including (D_minimax, D_max) in the extra field and signing the checkpoint bytes that include this field, any divergence in the dual-mode aggregates produces a checkpoint byte mismatch, which the same witness first-cosign-wins policy detects. The reduction is constructive. QED.

IMPLEMENTATION NOTE. Existing CWT implementations that do not include the dual-mode values in the extra field remain conformant to CWT but NOT to this profile. The capability flag of Section 8 distinguishes the two cases.

The validator scripts/validate_perfsec_coupling.py (check T-COMP-SPLIT) verifies that an honest witness refuses the second cosignature on the same (bundle_id, bundle_seq) when the dual-mode body differs, and that the existing signature does NOT cross-validate over the split body.

8. Joint Operational Profile (Capability Quartet)

A deployment is conformant to this profile if and only if it advertises in the bundle capability_flags array all four of:

```
   "cwt-profile-v1"           (CWT)
   "coherence-bfd-v1"         (Coherence-BFD)
   "ddos-resilience-v1"       (DDoS Resilience)
   "perfsec-coupling-v1"      (this document)
```

AND implements:

I-PS-1. Per-vantage rate-limit at NIC/XDP fast-path (Section 4.1) with parameters at or stricter than Section 4.2 defaults.

I-PS-2. HKDF derivation of K_BFD_auth per Section 6 in any deployment sharing K_op across CWT and BFD-auth.

I-PS-3. Dual-mode (D_minimax, D_max) carried inside the cosigned checkpoint extra field per Section 7.

I-PS-4. Joint replay-counter rule of Section 5.

I-PS-5. Broker dimensioning to JOINT(N, T_tick, q) of Section 3.2; receipt schema (Appendix B) reproduced in the operator's deployment-readiness document.

A consumer MUST reject bundles that advertise any of the first three flags but NOT the fourth in any deployment where joint operational hypotheses (insider flood plausible OR Byzantine fraction non-zero) hold.

Deployments operating only the CWT profile (no Coherence-BFD, no DDoS Resilience) MAY ignore this profile. Deployments operating Coherence-BFD without CWT are permitted but UNSAFE under H-TRUST-CWT failure (snapshots unauthenticated); the present profile assumes at least the CWT trust profile.

9. Security Considerations

This profile increases the security posture of the composed system: it does not introduce any new cryptographic primitive, but it removes five composition gaps that an attacker could otherwise exploit.

T-INSIDER-FLOOD (Section 2.2) is bounded by the per-vantage rate-limit (Section 4); the bound is rate_limit_factor + flood_factor * (c_xdp / c_path) instead of the unbounded linear blow-up of the unprotected case. Numerical receipt confirms that at flood_factor = 1024, the broker CPU under rate-limit stays within ~ 10x baseline (i.e., still under control on a small core budget) versus 1024x baseline without.

Cross-profile key separation (Section 6) prevents accidental or hostile reuse of K_v_epoch as a BFD AuthHMAC key (or vice versa); independence reduces to HKDF-SHA256 [RFC5869] PRF security.

Joint Byzantine + split-view (T-COMP-SPLIT, Section 7) is detected at the witness layer because the dual-mode aggregates are cosigned alongside the bundle root. The detection probability inherits T-SPLIT-1 (1 - (1 - 1/w)^q for q honest cosignatures of w witnesses).

Replay-counter coherence (Section 5) prevents the corner case where one of the two counters is silently treated as decisive. Cross-epoch replay is rejected by the BFD seq monotonicity that does NOT reset at epoch boundaries.

Open issues:

- The default rate-limit factors (4 sustained, 8 burst) are heuristic. HFT, orbital, and other low-jitter environments MAY tighten to 2/2. Operators choosing a lower factor MUST disclose the choice in the management interface; downstream consumers MAY rely on the default when not signaled otherwise.

- Hardware HMAC acceleration is RECOMMENDED for Regime C

and REQUIRED for Regime D. Software-only deployments at Regime D will not meet the JOINT bound.

- The witness liveness budget of CWT Section 12.4 (default 5 s) interacts with the bundle cadence of this profile. At bundle_period_ticks = 10 and T_tick = 50 ms, bundles are emitted every 500 ms; witness liveness 5 s permits up to 10 unsigned bundles before degraded mode kicks in. Operators MAY tighten this in latency-sensitive deployments.

10. IANA Considerations

This document requests registration of one MVPS Bundle Capability Flag in the registry defined by [I-D.melegassi-ippm-mvps-bundle] Section 11.3:

Flag name:	perfsec-coupling-v1
Semantics:	Deployment implements the joint contract of Section 8 of this document. Consumers MUST verify joint conformance (I-PS-1 .. I-PS-5) before declaring coherence on bundles that also advertise cwt-profile-v1, coherence-bfd-v1, and ddos-resilience-v1.

No new code points are requested in any other registry.

11. Acknowledgements

The author thanks the operator community of the May 2026 review cycle whose informal questions ("if security drops PPS, where does it drop?") directly motivated the explicit numerical reconciliation of Sections 1 and 3.

The author thanks Joas Antonio dos Santos Barbosa for the security review of the CWT trust profile that made the composition gaps visible at the spec-text level rather than at the deployment level.

The author thanks the IETF BFD, IPPM, and OPSAWG mailing lists for the conventions that this document follows.

12. References

12.1. Normative References

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, May 2010.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017.

[I-D.melegassi-ippm-mvps-bundle] Melegassi, L., "Multi-Vantage Path Snapshot (MVPS): A Canonical Bundle Format for Coordinated Traceroute

Measurements", draft-melegassi-ippm-mvps-bundle-00, May 2026.

[I-D.melegassi-coherence-bfd]
Melegassi, L., "Coherence-BFD: Sub-Second Coherence Detection Using Bidirectional Forwarding Detection Patterns", draft-melegassi-coherence-bfd-00, May 2026.

[I-D.melegassi-mvps-ddos-resilience]
Melegassi, L., "Volume-Independent DDoS Detection via Coherence-BFD: The MVPS DDoS Resilience Profile", draft-melegassi-mvps-ddos-resilience-00, May 2026.

[I-D.melegassi-santos-ippm-mvps-cwt]
Melegassi, L. and J. A. S. Barbosa, "MVPS Trust Profile: Lightweight Authentication via HMAC-SHA256, Operator Epoch Anchors, and Independent Witness Cosignatures for Multi-Vantage Path Snapshots", draft-melegassi-santos-ippm-mvps-cwt-00, May 2026.

[C2SP-TLOG-WITNESS]
Community Cryptography Specification Project, "tlog-witness: A Witness Protocol for Transparency Logs", <https://c2sp.org/tlog-witness>.

12.2. Informative References

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, June 2010.

[I-D.melegassi-mvps-incremental-be]
Melegassi, L., "Incremental Bandwidth-Efficient Multi-Vantage Path Synchrony (BE-MVPS): Cell-Partitioned Coherence with epsilon-Gated Sherman-Morrison Updates", draft-melegassi-mvps-incremental-be-00, May 2026.

[I-D.melegassi-iab-mvps-architecture]
Melegassi, L., "MVPS Architecture: Specification Conformance for the Multi-Vantage Path-Coherence Drafts", draft-melegassi-iab-mvps-architecture-00, May 2026.

[PERFSEC-AUDIT]
Melegassi, L., "MVPS Performance-Security Coupling Audit", docs/MVPS_PERFSEC_COUPLING_AUDIT.txt, Catellix technical note, May 2026.

[PERFSEC-PROOF]
Melegassi, L., "MVPS Performance-Security Coupling -- Mathematical Existence Proof", docs/MVPS_PERFSEC_COUPLING_PROOF.txt, Catellix technical note, May 2026. Full proofs of T-JCOST-1, T-VDOS-1, T-RC-1, T-COMP-SPLIT, T-NIC-SUFF, T-VOLINV, and T-AUTH-INHERIT, with finite chain back to v4.0 of the MVPS Mathematical Existence Proof.

[PERFSEC-RECEIPTS]
Catellix Research, "MVPS perfsec coupling joint cost and verification-DoS receipts", evidence/perfsec_joint_cost_receipt.json, evidence/perfsec_verification_dos_receipt.json, produced by scripts/validate_perfsec_coupling.py (12/12 PASS).

[XDP-PERF] Hoiland-Jorgensen, T., et al., "The eXpress Data Path: Fast Programmable Packet Processing in the Operating System Kernel", CoNEXT '18, December 2018. Reports 30-60 ns per-packet processing for hash-map-and-counter idioms on commodity x86 cores.

[CILIUM-XDP] Cilium, "BPF and XDP Reference Guide -- Performance", <https://docs.cilium.io/en/stable/bpf/>, accessed May 2026. Documents per-packet hash-map lookup overhead in the 30-100 ns range across recent Linux kernels.

Appendix A. Numerical Example: Tier-1 Operator (Regime C)

Single Tier-1 operator deployment:

```
N         = 10 000 vantages
T_tick    = 50 ms (Coherence-BFD V3 reference)
M         = 3
q         = 2 witnesses
cells_k   = 8
bundle_period_ticks = 10
Regime    = C  (PPS_t = 200 000 per path)
```

Computed (validator: scripts/validate_perfsec_coupling.py):

```
Path A (CWT)      = 200 000 * (4.20 + 2.10) us
                  = 200 000 * 6.30 us
                  = 1 260 ms/s = 126.00 % of one core
Path B (BFD)      = 200 000 * (0.30 + 2.10) us
                  = 200 000 * 2.40 us
                  =   480 ms/s =  48.00 % of one core
Path C (Witness)  = 1.6 bundles/s * 2 * 78.80 us
                  = 252 us/s   =   0.025 % of one core
JOINT             = 174.25 % of one core (~ 1.74 cores)
```

LEAD METRIC (apples-to-apples scaling within the joint formula):

```
Joint-to-joint scaling factor (1 Hz -> 50 ms, N=1k -> N=10k)
   = JOINT_RegimeC / JOINT_CWT_ref
   = 174.25 / 0.88
   = 198x
```

AUXILIARY METRIC (worst-case reading error vs single-axis CWT Section 14.1 HMAC-only figure 0.21 %):

```
Worst-case underprovisioning ratio (CWT-14.1-only sizing)
   = 174.25 / 0.21
   = 830x
```

The 198x figure is the operationally meaningful number; the 830x figure quantifies the additional reading error that an operator incurs by sizing from a single-axis cost rather than this profile's joint formula.

Conclusion: Tier-1 Regime C deployments require AT LEAST 2 dedicated cores for ingestion + verification (NOT counting the D^2 algebra, state updates, or publication). An operator sizing from CWT 14.1 alone, or from any other single-axis number, will under-provision by approximately two orders of magnitude. This is the operational reason to adopt the present profile.

Appendix B. Receipt Schema (Informative)

The validator produces two JSON receipts conforming to the following schemas (see scripts/validate_perfsec_coupling.py for the implementation, evidence/ for the produced files).

B.1. perfsec-joint-cost-receipt-v1

```
{
  "schema":            "perfsec-joint-cost-receipt-v1",
  "issued_at":         "<RFC3339Z>",
  "draft":             "<this I-D handle>",
  "theorem":           "T-JCOST-1",
  "constants_us":      { ... per-op cost constants ... },
  "constants_source":  "evidence/mvps_cwt_overhead_receipt.json",
  "scale_points":      [ { ...one entry per scale point... } ],
  "headline_regime_c_pct_one_core": <float>,
  "speedup_underprovisioning_vs_cwt_abstract": <float>,
  "verdict":           "PASS" | "FAIL"
}
```

B.2. perfsec-verification-dos-receipt-v1

```
{
  "schema":            "perfsec-verification-dos-receipt-v1",
  "issued_at":         "<RFC3339Z>",
  "draft":             "<this I-D handle>",
  "theorem":           "T-VDOS-1",
  "constants_us":      { ... },
  "scenario_n":        <int>,
  "scenario_t_tick_ms":<float>,
  "rate_limit_factor": <float>,
  "burst_factor":      <float>,
  "analytical_slope":  <float, == c_xdp / c_path>,
  "with_rate_limit":   [ ... per-flood-factor row ... ],
  "without_rate_limit":[ ... per-flood-factor row ... ],
  "baseline":          { ... },
  "with_ratio_at_max_flood":    <float>,
  "without_ratio_at_max_flood": <float>,
  "bound_violations":  [],
  "verdict":           "PASS" | "FAIL"
}
```

Author's Address

Leonardo Melegassi
Catellix
Andradina, SP
Brazil

Email: melegassi@catellix.com
URI: https://catellix.com/