## Aug 26, 2026 | Remote write (PRW2) stability

Attendees:

Notes:

- Krajo:
  - Mimir split read path from remote write protobuf: https://github.com/grafana/mimir/pull/16278 merged
  - Mimir implement RW2 RC3 (start timestamp move) part 1: https://github.com/grafana/mimir/pull/16475
  - Prometheus switch to appender V2 and change appender V2:
    - https://github.com/prometheus/prometheus/pull/19455
    - https://github.com/prometheus/prometheus/pull/19457
- Bartek:
  - Rc5
    - https://github.com/prometheus/docs/pull/3085 (exemplar)
    - https://github.com/prometheus/docs/pull/3081 (async receiver)
- Atomic RC
  - https://github.com/prometheus/docs/issues/3087
- RW3
  - If we want to carry more - much more metadata in the future (native metadata?)
  - Grouping series per metadata / metadata separate with pointer from series to it (special symbol number?)
  - Per target scrape cache
  - OTLP export?

## Aug 19, 2026 | Remote write (PRW2) stability

Attendees: Kyle, Paulin, Bartek Płotka gyorgy.krajcsovits@grafana.com

Notes

- Krajo:
  - Delta group progress (split of proto, unsafe casting)
    - Not using StartTimestamp in reads, Mimir proto rc3
    - AppenderV2
      - Exemplar split
- Bartek
  - Async
  - PRW2 Async Support likely consensus is 202 header
  - OTel - best effort collection of exemplars per sample
  - Potential exemplar spec change

        - Allow standalone exemplar series
        - SHOULD be part of the same request as relevant series
- Krjao: Oleg(Grafana) points out that rw2 will double bandwidth if all time series have start timestamps.
    - Should we have done a delta encoding between them to reduce the size?
    - Or advise people to use zero injection instead for cumulative samples?
    - Proposal
        - Default: Cumulative always sends 0, option to add ST per sample / zero injection
        - **4.0 zero injection a default for cumulatives?**
            - Potential for a useful feature for Grafana, not enabled
        - Agent mode? How to reduce the bandwidth and not send the start timestamp all the time for cumulative series?
            - Zero injection is tricky
            - Kyle Eckhart mentioned we can know when a new (for the agent) cumulative series is created. Idea would be to send a zero sample only then to save bandwidth. The receiver side can just reject it as duplicate/out of order (although how would it know when to reject as out of order?)
            - On the other hand we could also try only sending start timestamp when a new series for cumulative is detected - but then the receiver side has to know if it needs to inject zero sample for this start timestamp - which means it needs to know the type.
            - krajo: both cases seems to require some extra information
            - krajo: cumulative series might reset due to wrapping around/reset native histograms, introducing new start timestamp while the series already exists (facepalm)
            - krajo: I wonder if it would work that we simply don't send the start timestamp unless the series is new (and possibly if we detect that start timestamp changed). If the receiver side never does zero injection, then the start timestamp should be attached to the sample right after the start timestamp - and our PromQL code would work I think. This would align with the idea that we only ever do zero injection on scrape, never from remote write/otlp.
- Community member attempting attaching exemplars to samples per spec: https://github.com/prometheus/prometheus/pull/18014
    - B: **Client** vs server overhead, same durability problems
- Metadata PoC?
    - Trying to document what we want to know:
      Design Doc: Prometheus Remote Write v2 - Metadata
- Type and unit label break what you see is what you query in exposition
    - Likely fine
- Type and unit VS metadata
    - Series Record type and unit first citizen

- Hardest things first?
    - Solve help?

**Action items**

- Exemplar spec change to allow exemplars detached from samples
    - Spec change: At least one element in samples or in histograms MUST be provided. A TimeSeries MUST NOT include both samples and histograms. For series which (rarely) would mix float and histogram samples, a separate TimeSeries message MUST be used.
- Metadata WAL help vs series record

## Aug 12, 2026 | Remote write (PRW2) stability

Attendees: bartek@prometheus.io Bartek Płotka gyorgy.krajcsovits@grafana.com

Notes

- Design Doc: Prometheus Remote Write v2 - Metadata
    - How much do we care about OOO having the correct type and unit?
        - Google has a very strict database - each metric needs a type - but no OOO support
        - Type and Unit as labels somewhat solves this - but it is a breaking change
    - What **type** and **unit** actually break?
    - **WAL metadata optimization?**
        - vs type and unit
            - More records
            - Queue_manager overhead
            - Races possible
    - Leaning into allow type and unit and metadata-wal-records and allow choice?
    - Reserved __type__ problems
- PRW2 Async Support
- Exemplars
    - Works on a small scale, although there is still a > 0% chance that its exemplar will be sent in a totally separate request from its corresponding sample.
    - Because they share the same ref, the exemplar and the sample will always hash to the same shard. However QueueManager.Append and QueueManager.AppendExemplars are called independently by the WAL watcher.
    - When they are queued, they are enqueued as completely separate timeSeries structs in the internal queue: one with sType: tSample and one with sType: tExemplar.
    - Because they are treated as separate items internally, if the queue worker building the batch (buildV2WriteRequest) hits the max_samples_per_send limit or the flush_deadline right between processing the sample and the exemplar, it will cut the batch, sending the sample in one PRW request and the exemplar in the next.
    - We could assume it's fine for now, but then what to do for spec e.g. a light:

        - SHOULD be part of the same request and timeseries

**Action items**

- [ ] Overhead of Metadata WAL
- [ ] Consequences of __type__ and __unit__

---

## Aug 5, 2026 | Remote write (PRW2) stability

Attendees: Kyle, Krajo, Bartek

Notes

- Krajo's OKR have it as goal
- Why doing it:
    - Grafana - reduced bandwidth , customers looking for it
        - However metadata reverts 10% gain
        - People do not want to switch while experimental
        - Metadata: Protocol or do we want solve alway-on Prometheus implementations (known gaps)
- Metadata
    - Shard calculation issue with metadata: https://github.com/prometheus/prometheus/pull/19218
- [meta] PROM-35 Remote Write (PRW2.0) Stability · Issue #16944 · prometheus/prometheus
    - Async Mode
    - Exemplars
        - bug(sending RW2): RW2 sends disconnected exemplars without samples · Issue #17857 · prometheus/prometheus
        - Exemplar per series
            - Currently implemented
            - Violates spec - cannot send request without Samples (or Histograms)
            - Exemplars are per sample on all input interfaces
                - Except RW
                - Implementation per series
            - Options
                a) Remove that rule, it's per series now
                b) Don't change the protocol and implement per sample
                c) Cleaner approach would be to have Sample/Histogram.Exemplars and per sample?
            - Dimensions
                - Assumption: We want to support both modes? Pick one would be amazing

            - Per series (cheaper if that's ok)
            - Per sample for more strict systems
        - How to represent our decision on protocol?
    - Exemplar in a sample?
        - Google: Do we need to repeat it?

- Metadata
    - PRW1 metadata parity ([RW2] metadata.send feature parity with v1 #17191) (in progress [RW2.0] Enable fast metadata look up for Remote Write 2.0 #17436)
    - State
        - metadata-wal-records
        - Type and unit labels
            - Standards metrics does not have it
            - We don't have help
            - Constructing metadata on RW from type and unit, then strips from labels etc
        - Old path is not implemented with RW2 because it's per series (scrape cache)
            - Per series, TSDB
            - Instead of scrape cache
    - **How is it connected to Metadata storage?**
        - Both requires TSDB per series
- Prometheus receives AppenderV2? That implements ST?
    - STs are working with RW receiving (storage/remote: migrate Remote Write 2.x receiver to AppenderV2 #19251) (could be descoped?)

Action items

- [ ] Bartek todo an exemplar proposal
- [ ] Krajo and Kyle to do a metadata proposal