
European Commission
7 May 2025
SEMIC LDES 2025 A new standardisation trajectory
Working group I
We start at 14:05
interoperable europe

SEMIC Assets

Vocabularies

- CORE PERSON VOCABULARY
- CORE BUSINESS VOCABULARY
- CORE LOCATION VOCABULARY
- CORE CRITERION & EVIDENCE VOCABULARY
- CORE PUBLIC ORGANISATION VOCABULARY
- CORE PUBLIC EVENT VOCABULARY

Technical

- LDES

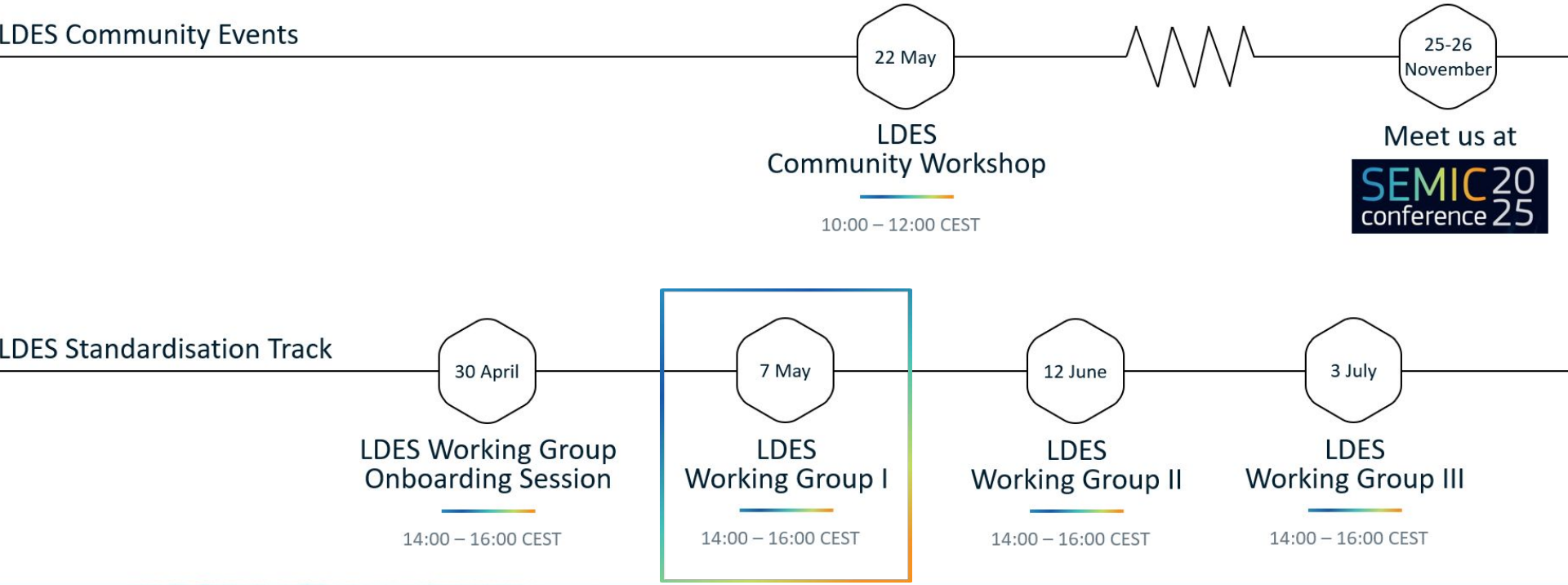
Application Profiles

- CORE PUBLIC SERVICE VOCABULARY
- DCAT-AP
- BReg DCAT-AP
- Geo DCAT-AP
- Stat DCAT-AP
- DCAT-AP for HVD
- ADMS - AP
- ML DCAT - AP

#LDES

European Commission

Upcoming LDES Events



Communication channels

- Online meetings (see previous slide)
- Physical meetings: SEMIC2025
- Core discussions documented in Github issues: https://github.com/SEMICeu/LinkedDataEventStreams/issues/
- Matrix channel: https://matrix.to/#/#ldes:chat.semantic.works
- Want to follow up all the working groups and receive the reports?
  Leave your e-mail address in the chat!

#LDES

European Commission

Agenda

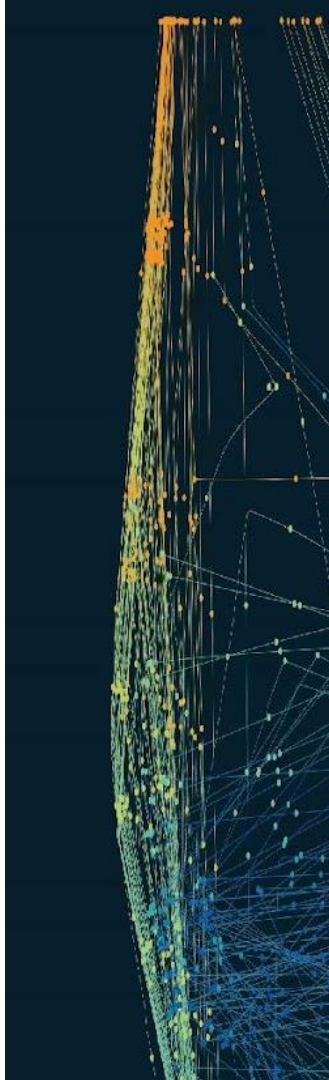
14:05 - 14:15	Welcome & Tour de table
14:15 - 14:25	Reminder of the process and goal
14:25 - 14:50	PR71: the overall rewrite as the basis
14:50 - 15:10	Explanation of the basis of the client algorithm and clarifications for `ldes:timestampPath`, `tree:member`, `ldes:EventSource` done in PR71.
15:10 - 15:50	PR73: Transactions and revisiting retention policies

#LDES

European Commission

Who’s attending?

Tour de table

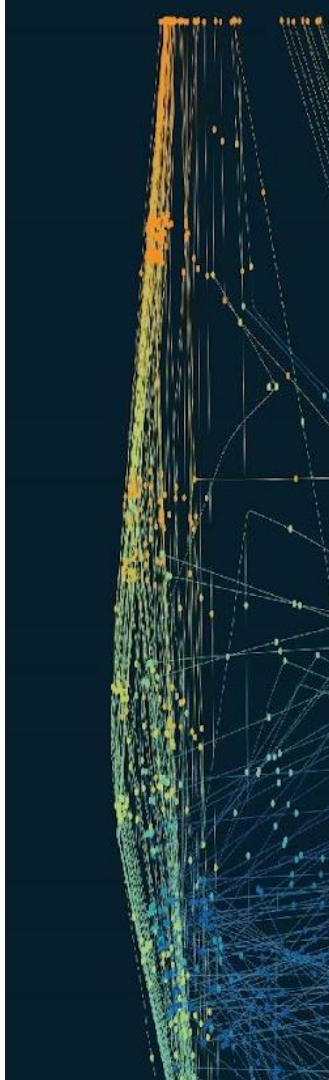


#LDES

European Commission

The 2025 trajectory

Recap



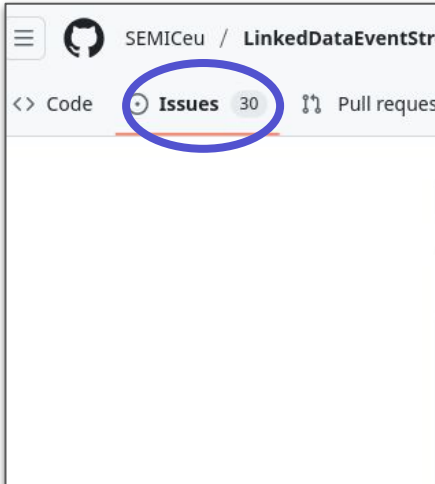
#LDES

European Commission

The spec today

A living document
A vocabulary with terms and usage notes for producers and consumers
So far there has not been a real promise to keep the spec stable

30 issues open on the repository and
a starting note prepared in 2024 by Flanders
Requests for clarity as well as feature requests.



#LDES

Ambition of this trajectory

1. **At https://w3id.org/ldes/specification**

A consumer-oriented backwards-compatible stable release that this way becomes more testable.

Although we might today already deviate a little bit…

2. **At https://w3id.org/ldes/server-primer**

A server primer with best practices for LDES providers. Principles:

- High performance by default, not after optimization
- Primer remains domain agnostic
- Explaining how to “envelope” your data

#LDES

Not in this trajectory, but something we will be able to pick up as a community

1. At https://github.com/SEMICeu/LDES-implementation-reports/

We'll have implementation reports about:

- DCAT-AP Feeds
- Cultural Heritage Feeds (PR open)
- *Your implementation report? Pull request this today!*

2. LDES Clients compliance report

Linking to existing implementations, and reporting on to what extent the spec has been implemented.

#LDES

In 2025

A public review period for implementations from 3rd of July until SEMIC2025 where we want to publish a stable LDES spec.

#LDES

European Commission

How will we reach **consensus**?

Everyone prepares comments on the PRs before the meeting.

Decision during the call can be:

1. Accept and merge
2. Conditional accept: merge when conditions are validated
3. Full reject

Workshops topics

WS1: Restructuring the spec and extending retention policies

Today's workshop!

WS2: Consumer algorithm: iteration and state management

PRs will be ready 1 week before the workshop and we'll send out an email

WS3: A server primer with best practices

PRs will be ready 1 week before the workshop and we'll send out an email

Open issues we want to solve today

After first a consumer-oriented rewrite of the spec

- Definition of `ldes:timestampPath` and the fact out-of-order is impossible: #10, #35, #49, #61
- Definition of `ldes:EventSource`: #34
- `ldes:immutable` and `ldes:expires` on a `tree:Node`: #53
- named graphs and activity streams-like examples in the spec: #37 , #43
- `tree:member` in LDES points to an IRI, not a blank node #56
- Retention policies:
  - LatestVersionSubset: better description #47
  - Retention policies with a deletion #50
- Distributed transactions: #46

7 May

LDES
Working Group I

14:00 – 16:00 CEST

#LDES

European Commission

Today's meeting: consumer perspective

No MUST/SHOULD/MAY in this meeting for the server.
We will only use MUST/SHOULD/MAY for the client.

From the server perspective, it either works in the client or it doesn't.

In the server primer (workshop 3), those will be derived from this spec:

- MUST for a server is derived from the fact that the client would not work otherwise.
- SHOULD is a *best practice* that makes the standard client algorithm perform well
- MAY provides useful information for the consumer later in the pipeline

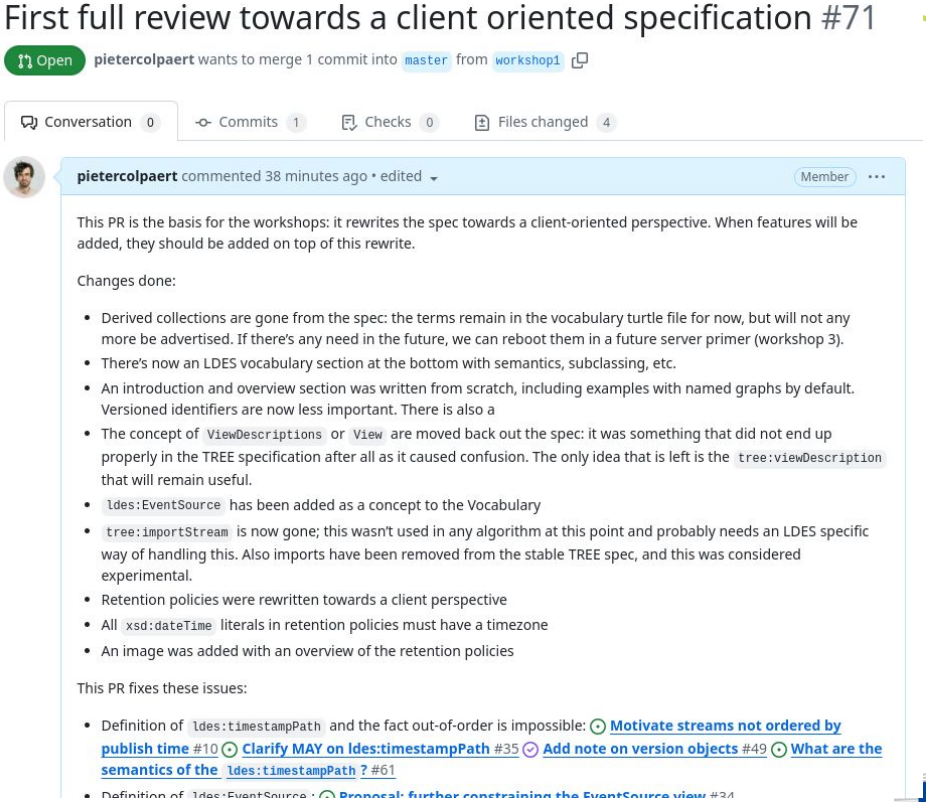
The basis for fixing these issues
PR71

A PR was opened as a *basis* for solving these issues

https://github.com/SEMICeu/LinkedDataEventStreams/pull/71

Live run through the PR, and seeing whether we can agree to merge the PR.

Thanks for the reviews!



First full review towards a client oriented specification #71

Open **pietercolpaert** wants to merge 1 commit into `master` from `workshop1`

Conversation 0 · Commits 1 · Checks 0 · Files changed 4

pietercolpaert commented 38 minutes ago • edited — Member

This PR is the basis for the workshops: it rewrites the spec towards a client-oriented perspective. When features will be added, they should be added on top of this rewrite.

Changes done:

- Derived collections are gone from the spec: the terms remain in the vocabulary turtle file for now, but will not any more be advertised. If there's any need in the future, we can reboot them in a future server primer (workshop 3).
- There's now an LDES vocabulary section at the bottom with semantics, subclassing, etc.
- An introduction and overview section was written from scratch, including examples with named graphs by default. Versioned identifiers are now less important. There is also a
- The concept of `ViewDescriptions` or `View` are moved back out the spec: it was something that did not end up properly in the TREE specification after all as it caused confusion. The only idea that is left is the `tree:viewDescription` that will remain useful.
- `ldes:EventSource` has been added as a concept to the Vocabulary
- `tree:importStream` is now gone; this wasn't used in any algorithm at this point and probably needs an LDES specific way of handling this. Also imports have been removed from the stable TREE spec, and this was considered experimental.
- Retention policies were rewritten towards a client perspective
- All `xsd:dateTime` literals in retention policies must have a timezone
- An image was added with an overview of the retention policies

This PR fixes these issues:

- Definition of `ldes:timestampPath` and the fact out-of-order is impossible: Motivate streams not ordered by publish time #10 Clarify MAY on ldes:timestampPath #35 Add note on version objects #49 What are the semantics of the `ldes:timestampPath` ? #61
- Definition of `ldes:EventSource` : Proposal: further constraining the EventSource view #34

#LDES

European Commission

What’s the decision?

1. Accept and merge
2. Conditional accept: merge when conditions are validated.
   Conditions being…
3. Full reject

#LDES

European Commission

Open issues we want to solve today

~~*After first a consumer-oriented rewrite of the spec*~~

- Definition of `ldes:timestampPath` and the fact out-of-order is impossible: #10, #35, #49, #61
- Definition of `ldes:EventSource`: #34
- `ldes:immutable` and `ldes:expires` on a `tree:Node`: #53
- named graphs and activity streams-like examples in the spec: #37 , #43
- `tree:member` in LDES points to an IRI, not a blank node #56
- Retention policies:
  - LatestVersionSubset: better description #47
  - Retention policies with a deletion #50
- Distributed transactions: #46

7 May

LDES
Working Group I

14:00 – 16:00 CEST

#LDES

European Commission

Mainly on context information

- Adding a `tree:shape` and typing your search tree as an `ldes:EventSource` is interesting in the discovery phase
- The Retention Policy, transactions and versioning are interesting for processors in the consumption pipeline

In Workshop 2 we are going to introduce text that will influence the client algorithm.

Today, the next consumer related text is going to be lighter: on the fact that context information

An example of an LDES event source

```
ex:Collection1 a ldes:EventStream;
            tree:view <> ;
            tree:shape <shape.ttl> ;
            ldes:timestampPath dct:created ;
            ldes:versionOfPath dct:isVersionOf ;
            tree:member ex:Subject1v1, ex:Subject2v1 .

<> a ldes:EventSource; # Just one page, but with a retention policy
   ldes:retentionPolicy [
      a ldes:DurationAgoPolicy ;
      tree:value "P1Y"^^xsd:duration
   ] .

ex:Subject1v1 a dct:isVersionOf ex:Subject1 ;
              dct:created "2024-12-21T06:00:00Z"^^xsd:dateTime .
ex:Subject1v1 {
  ex:Subject1 ex:value  "Rosa" .
}

ex:Subject2v1 a dct:isVersionOf ex:Subject2 ;
              dct:created "2024-12-21T13:00:00Z"^^xsd:dateTime  .
ex:Subject2v1 {
  ex:Subject2 ex:value "Nicolas" .
}
```

#LDES

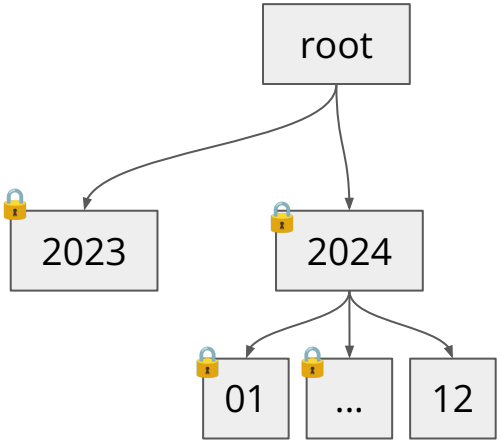
A client: how will it work (fwd: workshop 2)

Based on the caching headers and a client configurable polling interval, it will keep polling this page, each time keeping a state of members that were emitted previously.

State

```
emittedMembers: [{ root: [ ex:Subject1v1, ex:Subject2v1 ] } ]
   → keeps the IDs of the members you saw last time per node
```

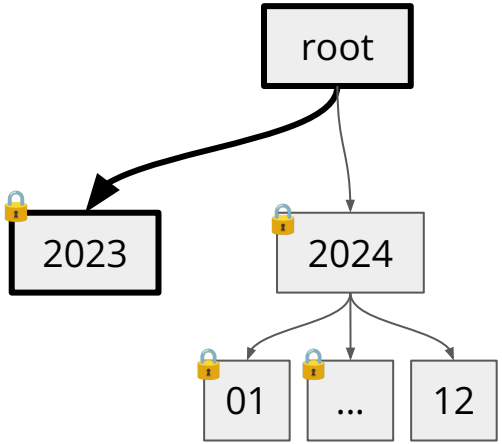
And it also can traverse a search tree



#LDES

European Commission

A client: how will it work (fwd: workshop 2)



Starts **2024-12-21**

- Fetch root node
- follow 2023 link and emit all members in page
- from the root, you can indicate you've seen 2023
- In 2023, you don't need to keep state: the page is flagged as cache immutable
- root node needs to be refetched later

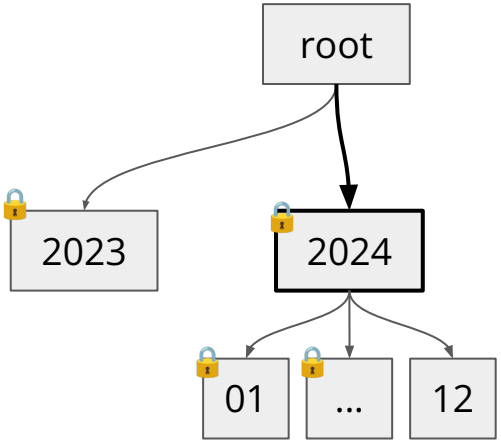
State

```
visitedNodes: [ { <root>: <2023> }]
   → which nodes were already visited last time on non-immutable nodes
emittedMembers: [ ]
   → keeps the IDs of the members you saw last time on non-immutable nodes
refetchNodes: [ {<root>: "2025-01-01T02:00Z" } ]
   → on the non-immutable nodes, keep an expiry timestamp
```

#LDES

European Commission

A client: how will it work (fwd: workshop 2)



Starts **2024-12-21**

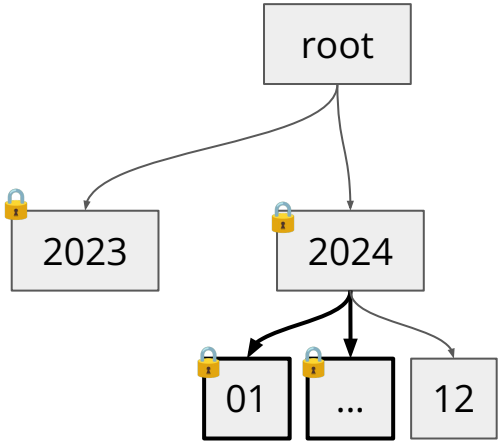
- At the same time of the previous slide, also the 2024 node can be processed
- 2024: the page is immutable, but members will still be added in the leaf node 12
- from the root, you can indicate you've seen 2024

State

```
visitedNodes: [ { <root>: <2023>, <2024> }]
refetchNodes: [ { <root>: "2025-01-01T02:00Z" } ]
emittedMembers: [ ]
```

#LDES

A client: how will it work (fwd: workshop 2)



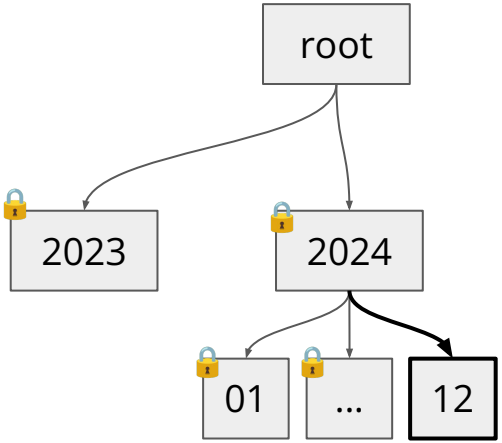
Starts **2024-12-21**

The immutable month pages are processed just like 2023. Once processed, we also don't need to keep them in visited nodes, because 2024 is flagged as immutable already: no relations will be added anymore. As the months are immutable, we don't need to keep the members that were emitted.

State

```
visitedNodes: [ { <root>: <2023>, <2024> }]
refetchNodes: [ { <root>: "2025-01-01T02:00Z" } ]
emittedMembers: [ ]
```

A client: how will it work (fwd: workshop 2)



Starts **2024-12-21**

On this day, the node 2024-12 is still in progress.

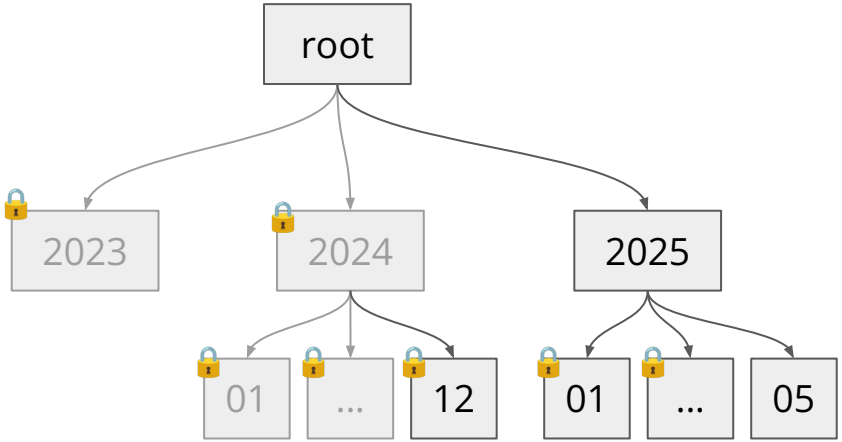
The members are stored in the emittedMembers array, and we'll indicate we'll need to refetch this node

State

```
visitedNodes: [ { <root>: <2023>, <2024> }]
refetchNodes: [ { <root>: "2025-01-01T02:00Z" },
                { <2024-12>: "2024-12-22T02:00Z" } ]
emittedMembers: [ { <2024-12>: [ <Nicolas>, <Rosa>, …
] }]
```

#LDES

A client: how will it work (fwd: workshop 2)



State

```
visitedNodes: [ { <root>: <2023>, <2024> }]
refetchNodes: [ { <root>: 2025-01-01T02:00Z },
                { <2024-12>: 2024-12-22T02:00Z } ]
emittedMembers: [ { <2024-12>: [ <Nicolas>, <Rosa>, …
] }]
```

Resumes today (2025-05-07)

From the state it will see it needs to refetch root and 2024-12

From root, it will see it didn't visit 2025 yet, and will continue processing the pages.

`lde:timestampPath`

The client can replicate and synchronize without it,
but needs it if it wants to understand what property to use for the *order*, which is a prerequisite for consumers that want:

1. the members to be emitted in the intended order
2. to establish the latest version of an entity described in a member
3. to understand retention policies (e.g., keeping the last hour, or latest versions)

A client MAY also use `lde:timestampPath` as a way

- to understand certain pages will be *immutable*, if the t_{now} surpassed the window of the subtree already.
- to prioritize certain relations when it needs to process the LDES in the intended order

FAQ 1: but is it then the logical time, or the physical/committed time?

Neither and both:

It just points at the literal it can use on which order is guaranteed to implement specific LDES consumer functionality.

The semantics of the property it points at is formalized by the domain model we're agnostic of. E.g., `dct:created`, `dct:modified`, `as:published`, `sosa:resultTime`, `prov:generatedAt`, …

Exotic example: it could be that a historical LDES has been published that replays the events of 10 years ago, and uses the historical timestamp as an ldes:timestampPath.

FAQ 2: But then out-of-order is impossible?

It's still possible: you just cannot use `ldes:timestampPath` then,

Use cases for this include:

- a server gets out of order arrivals
- a server makes a part of the stream public at a later stage (e.g., sensor X can now also be published in the LDES with its history)

A simple fix would then be to add a timestamp of when the member did arrive in the LDES.

Limitations

1. Multiple members can happen at the same time, and then order is unspecified within the same timestamp. If we want **strict ordering** we'll need to still look at introducing a different solution at a later time.

2. We cannot have an LDES without `ldes:timestampPath` that:
   a. still documents how the versions supersede each other
   b. use a different property for versions that supersede each other
      vs. a property that indicates an order in additions for e.g., assessing immutability of pages.

Possible future proposal

Instead of setting `ldes:timestampPath`, use 2 other properties:

1. `ldes:sequencePath`: a literal (`xsd:dateTime`, `xsd:string` or `xsd:integer`) that can be used by the consumer to assess immutability of pages
2. `ldes:versionSequencePath`: a literal (`xsd:dateTime`, `xsd:string` or `xsd:integer`) that can only be used by the consumer to assess the latest version, but this can come in out-of-order if this path is not the same as the sequencePath.

Setting all 3 properties differently MUST result however in an illegal LDES exception?

`lde:EventSource`

A chronological search tree that is optimized for the LDES client for an ordered full or latest state replication and synchronization

For discovery to indicate that:

- `lde:timestampPath` has been set
- the path of the relations are *primarily* based on this `lde:timestampPath`

An extra promise of the server

- each member only appears once in this chronological search tree *(not yet in the text)*

#LDES

FFW to workshop 2: establishing when a `tree:Node` is immutable

Proposal: a `tree:Node` can be considered immutable, when at least one of the following is true:

1. it is flagged as immutable in the caching response headers
   `Cache-Control: public, immutable`
2. as an alternative to 1 for when you cannot set HTTP headers:
   it is explicitly flagged as immutable using a newly proposed `ldes:immutable` property
3. the window of the subtree defined by the `ldes:timestampPath` relations on the parent is < t_{now} , given that we clarify `ldes:timestampPath` decides the order of entities as they appear in the LDES.

#LDES

FFW to workshop 2:
When to re-fetch a `tree:Node`?

1. Calculating a fetch time based on caching headers:
   a. If both `max-age` and `Age` are set: $t_{refetch} = t_{now} + (max\text{-}age - age)$
   b. If only `max-age` is set:
      we do not know how long the page already exists, so we need to apply a heuristic
2. Based on a client's configured polling interval
3. New proposal to be made for next group: indicating an expiration at a specific moment in time… Something like:

```
ldes:expiration [
     ldes:lastModified: dateTime at which the Node was last modified
     ldes:timeToLive: duration from the modified date in which the node is expected to
update
]
```

#LDES

European Commission

Open issues we want to solve today

- ~~Definition of `ldes:timestampPath` and the fact out-of-order is impossible: #10, #35, #49, #61~~
- ~~Definition of `ldes:EventSource`: #34~~
- ~~`ldes:immutable` and `ldes:expires` on a `tree:Node`: #53~~
- named graphs and activity streams-like examples in the spec: #37 , #43
- `tree:member` in LDES points to an IRI, not a blank node #56
- Simple transactions: #46
- Retention policies:
  - LatestVersionSubset: better description #47
  - Retention policies with a deletion #50

7 May

LDES
Working Group I

14:00 – 16:00 CEST

#LDES

European Commission

Named graphs are now exemplified

But more context will be needed in the server primer (workshop 3)

EXAMPLE 2 ¶

An example record from a base registry of addresses in the [turtle] format:

```
ex:AddressRecords a ldes:EventStream ;
                  ldes:timestampPath dcterms:created ;
                  ldes:versionOfPath dcterms:isVersionOf ;
                  tree:shape ex:shape2.shacl ;
                  tree:view <> ;
                  tree:member ex:AddressRecord1-activity1 .

ex:AddressRecord1-activity1 dcterms:created "2026-01-01T00:00:00Z"^^xsd:dateTime ;
                            adms:versionNotes "First version of this address" ;
                            dcterms:isVersionOf ex:AddressRecord1 .

 ex:AddressRecord1-activity1 {
    ex:AddressRecord1 dcterms:title "Streetname X, ZIP Municipality, Country" .
}
```

It is a result of relying on the TREE CG member extraction algorithm which currently is being revisited at the W3C CG to become more unambiguous. https://github.com/TREEcg/specification/issues/139

Planned at the TREE CG meeting next week, same time. See https://github.com/TREEcg/specification

#LDES

European Commission

`tree:member` points at an IRI in LDES

State

```
visitedNodes: [ { <root>: <2023>, <2024> }]
refetchNodes: [ { <root>: "2025-01-01T02:00Z" },
                {  <2024-12>: "2024-12-22T02:00Z" } ]
emittedMembers: [ { <2024-12>: [ <Nicolas>, <Rosa>, …
] }]
```

Because they’re used by consumers as a key, for example as used in the state by the client

#LDES

European Commission

Open issues we want to solve today

- ~~Definition of `ldes:timestampPath` and the fact out-of-order is impossible: #10, #35, #49, #61~~
- ~~Definition of `ldes:EventSource`: #34~~
- ~~`ldes:immutable` and `ldes:expires` on a `tree:Node`: #53~~
  - ~~⇒ next workshop~~
- ~~named graphs and activity streams-like examples in the spec: #37 , #43~~
- ~~`tree:member` in LDES points to an IRI, not a blank node #56~~
- Simple transactions: #46
- Retention policies:
  - LatestVersionSubset: better description #47
  - Retention policies with a deletion #50
  - Log compaction

7 May

LDES
Working Group I

14:00 – 16:00 CEST

#LDES

European Commission

Introducing transactions

Main discussion happened in:
https://github.com/SEMICeu/LinkedDataEventStreams/issues/46

For a consumer who uses the LDES client

Tells a consumer the event should only be “committed” once the transaction has been finalized.

Three new properties on top of an LDES entity:

```
<LDES> a ldes:EventStream ;
       # Points at a named node or literal that is the string identifying a transaction
       ldes:transactionPath ( … ) ;
       # The path of the finalized indication
       ldes:transactionFinalizedPath ( … ) ;
       # this is the default: looking for a boolean true,
       # but it can also look for e.g., ex:committed.
       ldes:transactionFinalizedObject true .
```

Transaction IDs MUST be unique in an LDES

#LDES

And understanding whether something is possibly a Create, an Update or a Delete?

```
ex:yourLDES ldes:versionDeletePath rdf:type ; #defaults to rdf:type
        ldes:versionDeleteObject as:Delete ;
        ldes:versionCreatePath rdf:type ;
        ldes:versionCreateObject as:Create ;
        ldes:versionUpdatePath rdf:type ;
        ldes:versionUpdateObject as:Update ;
        ldes:versionOfPath as:object .
```

Example:

```
<https://example.org/Dataset1#Event1> a as:Create ;
    as:object <https://example.org/Dataset1> ;
    as:published "2026-10-01T12:00:00Z"^^xsd:dateTime .

<https://example.org/Dataset1#Event1>  {
    <https://example.org/Dataset1> a dcat:Dataset ;
        ...
}
<https://example.org/Dataset1#Event2> a as:Delete ;
    as:object <https://example.org/Dataset1> ;
      as:published "2026-10-01T13:00:00Z"^^xsd:dateTime .
```

#LDES

European Commission

And with transactions

```
ex:yourLDES a ldes:EventStream ;
        ldes:transactionPath ex:transaction ;
        ldes:transactionFinalizedPath ex:transactionEnded ;
        ldes:versionDeleteObject as:Delete ;
        ldes:versionCreateObject as:Create ;
        ldes:versionUpdateObject as:Update ;
        ldes:versionOfPath as:object .
```

Example:

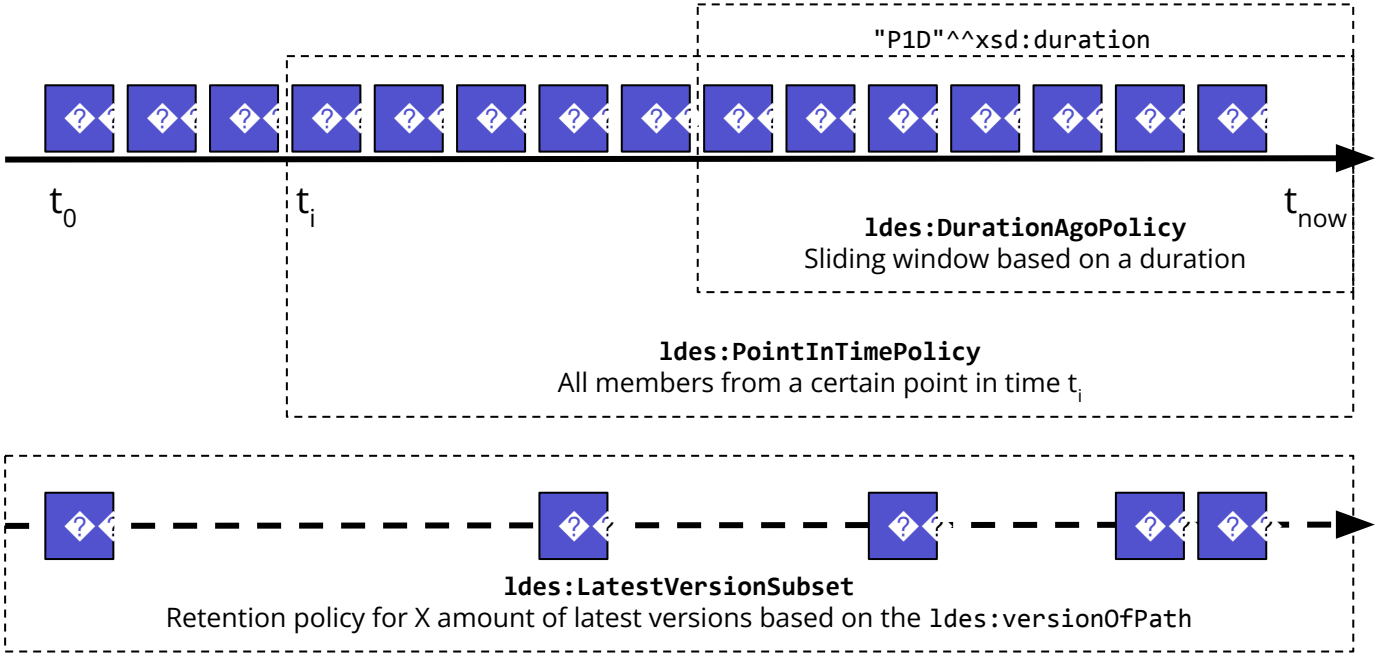
```
<https://example.org/Dataset1#Event1> a as:Create ;
    as:object <https://example.org/Dataset1> ;
    ex:transaction  123456 ;
    as:published "2026-10-01T12:00:00Z"^^xsd:dateTime .

<https://example.org/Dataset1#Event1>  {
    <https://example.org/Dataset1> a dcat:Dataset ;
        ...
}
<https://example.org/Dataset1#Event2> a as:Delete ;
    as:object <https://example.org/Dataset1> ;
    ex:transaction  123456 ;
    ex:transactionEnded true ;
    as:published "2026-10-01T13:00:00Z"^^xsd:dateTime .
```

Open issue:
How does a consumer separate the envelope from the payload?

Revisiting retention policies

And potentially deprecating the way of doing retention policies today.

LDES retention policies today



#LDES

European Commission

What is the functionality a consumer could implement based on a retention policy?

1. Checking whether you are fast enough to get a valid replication

2. Selecting a search tree in the discovery phase
   ⇒ but this algorithm is not specified or not part of the scope of this standardization trajectory. See TREE Discovery nonetheless.

Today however, the main purpose is metadata to manually select the right provider you want to use for a particular use case. Client check is not yet implemented.

More complex retention policies needed

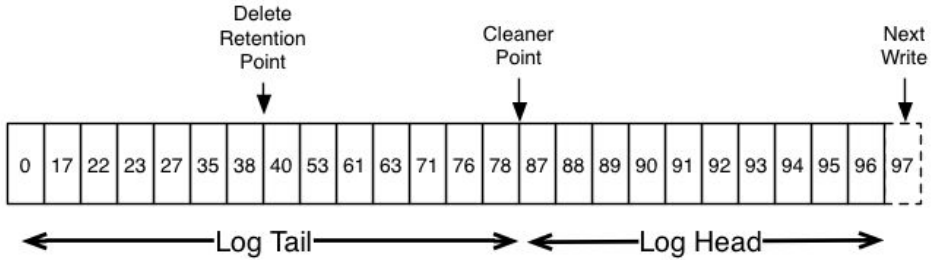
- Different retention policy for deletions
- Composite retention policies taking the intersection of multiple?
  … Although not all intersections make sense.
- Log compaction
- Respecting transaction boundaries

Inspiration: log compaction in Kafka

https://docs.confluent.io/kafka/design/log_compaction.html

Compaction in action

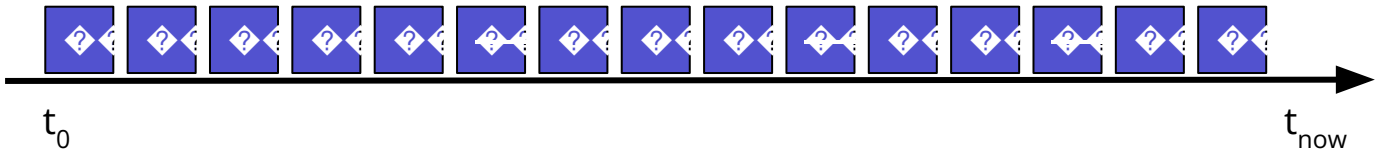
The following image provides the logical structure of a Kafka log, at a high level, with the offset for each message.



The head of the log is identical to a traditional Kafka log. It has dense, sequential offsets and retains all messages. Log compaction adds an option for handling the tail of the log.

#LDES

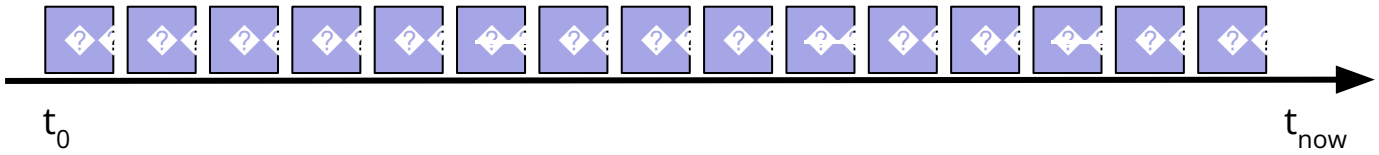
European Commission



Defining no retention policy means a search tree will keep everything.

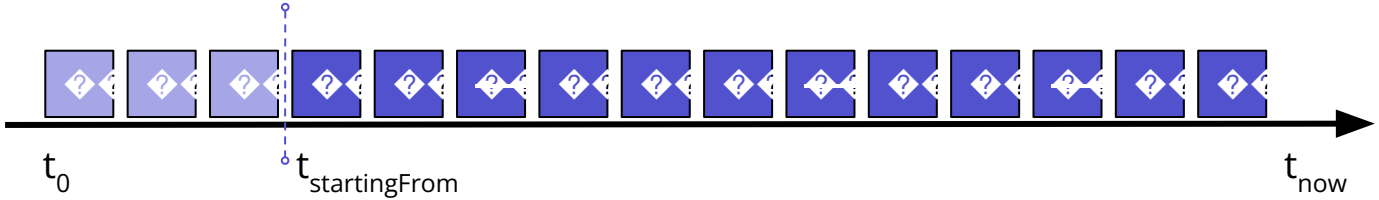
```
ex:LDES a ldes:EventStream ;
        tree:view <> .
```

#LDES



This retention policy keeps nothing in the search tree
although this is of course useless

```
<> ldes:retentionPolicy [] .
```

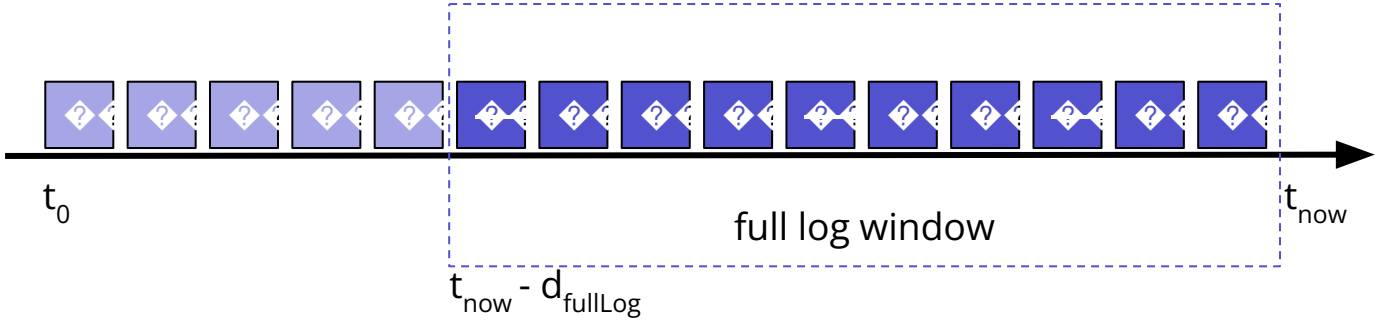


When we add a startingFrom property, all members after or equal to the startingFrom are added.

```
<> ldes:retentionPolicy [
     ldes:startingFrom "tstartingFrom"^^xsd:dateTime ;
] .
```

#LDES

Use case:
I'm an archiver that archives from an LDES event source starting a certain point

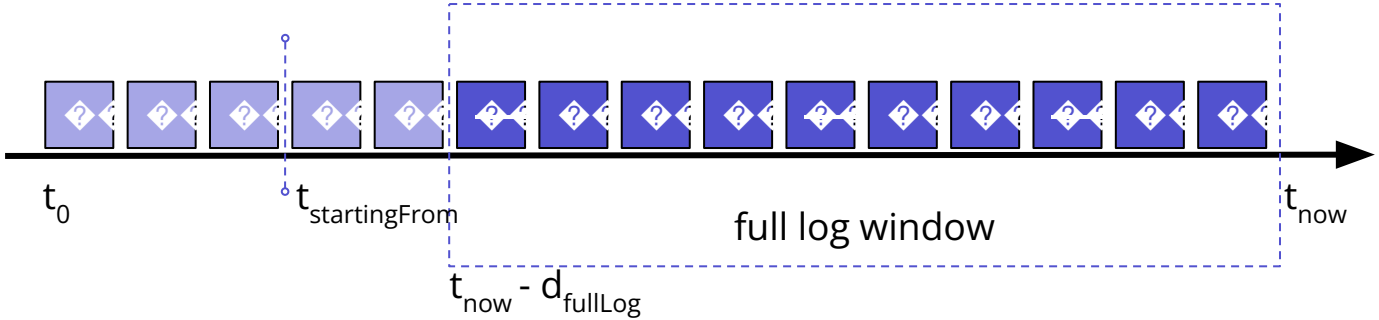


You can also keep a window of members using a duration from the “now”. Overwrites possible other properties, except for the `ldes:startingFrom` property.

```
<> ldes:retentionPolicy [
     ldes:fullLogDuration "d_fullLog"^^xsd:duration
] .
```

#LDES

Use case:
I’m an event source that only wants to keep the raw sensor values for 1 hour

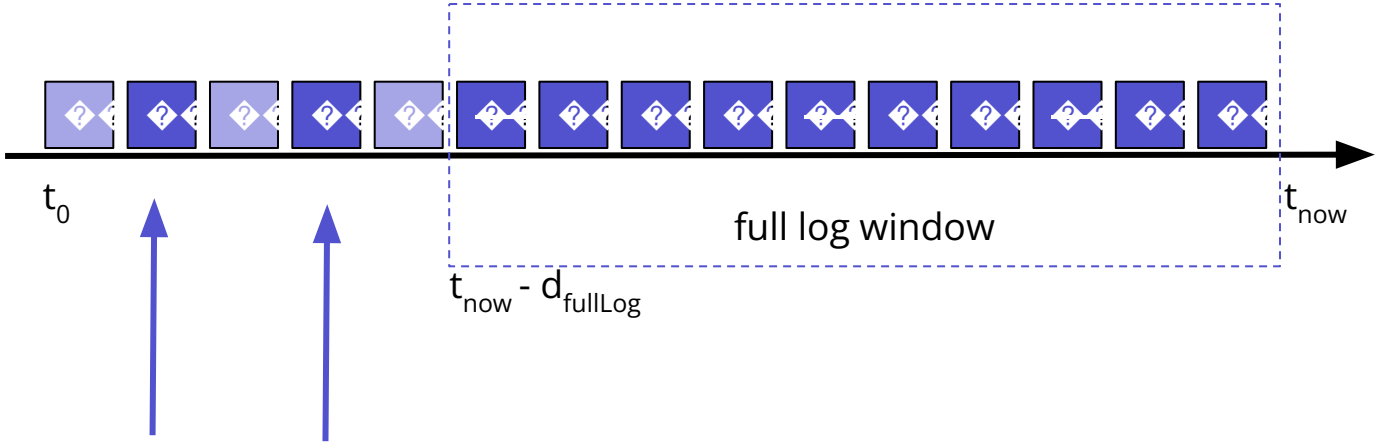


They can be combined both. There is only one possible interpretation that makes sense here.

```
<> ldes:retentionPolicy [
     ldes:startingFrom "tstartingFrom"^^xsd:dateTime ;
     ldes:fullLogDuration "dfullLog"^^xsd:duration ;
] .
```

#LDES

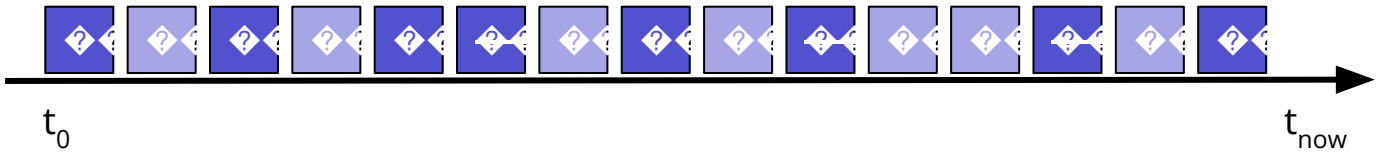
Use case:
I'm an archiver that only wants to keep the raw sensor values for 1 year, but this is still longer than the event source I started syncing with at a specific time.



A full log window MUST respect transaction boundaries if documented!

#LDES

European Commission

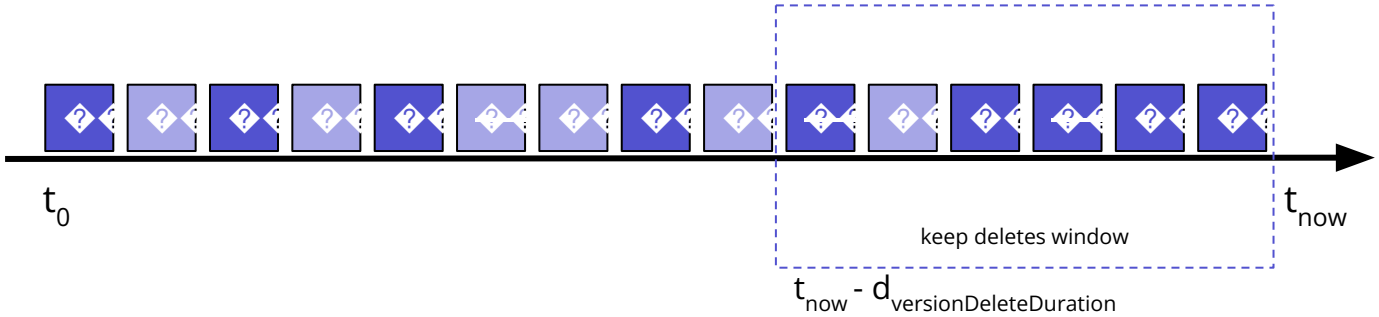


This retention policy only keeps the latest version of entities

```
<> ldes:retentionPolicy [
     ldes:versionAmount 1 ;
] .
```

#LDES

European Commission



This retention policy keeps the last versions, but will not keep tombstones after $d_{versionDeleteDuration}$

```
<> ldes:retentionPolicy [
     ldes:versionAmount 1 ;
     ldes:versionDeleteDuration "d_versionDeleteDuration"^^xsd:duration ;
] .
```

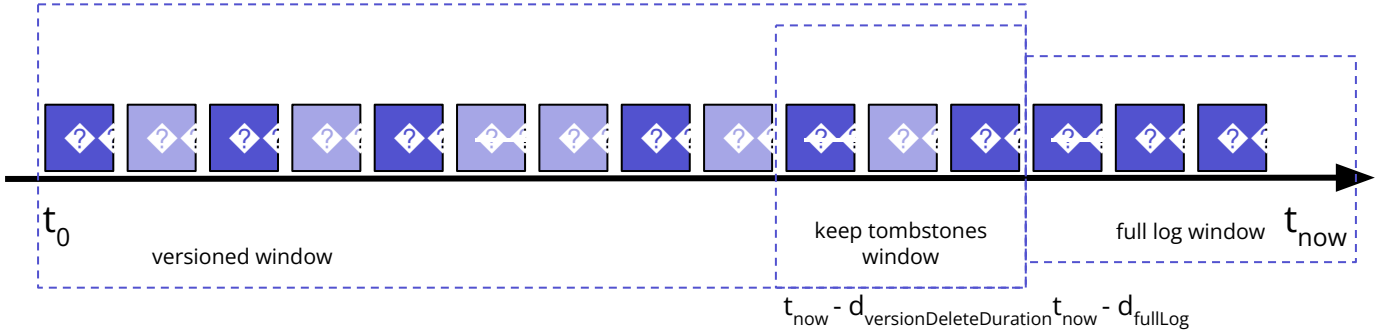
#LDES

European Commission

How do we know what a deletion is?

```
<LDES> ldes:versionDeletePath rdf:type ;
       ldes:versionDeleteObject as:Delete ;
       ldes:versionCreatePath rdf:type ;
       ldes:versionCreateObject as:Create ;
       ldes:versionUpdatePath rdf:type ;
       ldes:versionUpdateObject as:Update .
```

⇒ From the transactions

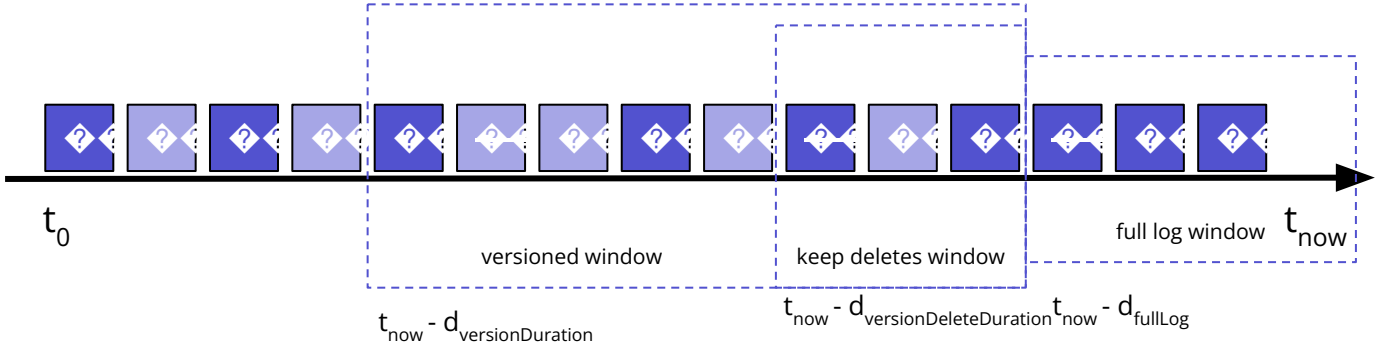


And you can combine it with the keep all to indicate how much of the full event stream you're going to keep regardless of versions. Also here there's only 1 possible interpretation that makes sense.

```
<> ldes:retentionPolicy [
     ldes:fullLogDuration "dfullLog"^^xsd:duration ;
     ldes:versionAmount 1 ;
     ldes:versionDeleteDuration "dversionDeleteDuration"^^xsd:duration ;
] .
```

#LDES

European Commission

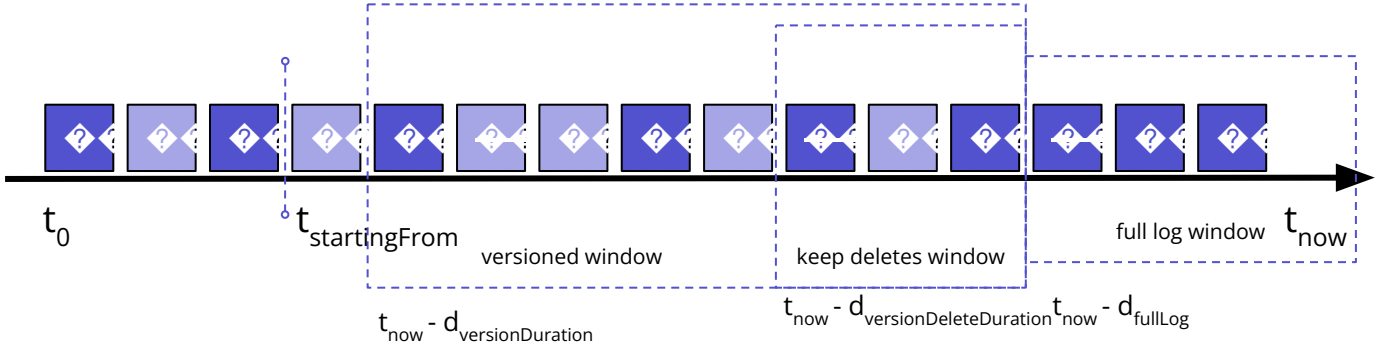


And you can also limit the duration you keep those latest versions

```
<> ldes:retentionPolicy [
     ldes:fullLogDuration "dfullLog"^^xsd:duration ;
     ldes:versionAmount 1 ;
     ldes:versionDuration "dversionDuration"^^xsd:duration ;
     ldes:versionDeleteDuration "dversionDeleteDuration"^^xsd:duration ;
] .
```

#LDES

Use case:
I'm an event source that keeps the latest reading of a sensor for 1 year.



Everything together for the sake of putting everything together

```
<> ldes:retentionPolicy [
     ldes:startingFrom "tstartingFrom"^^xsd:dateTime ;
     ldes:fullLogDuration "dfullLog"^^xsd:duration ;
     ldes:versionAmount 1 ;
     ldes:versionDuration "dversionDuration"^^xsd:duration ;
     ldes:versionDeleteDuration "dversionDeleteDuration"^^xsd:duration ;
] .
```

#LDES

European Commission

Limitations of this new proposal

1. **No type-specific retention policies**
   e.g., in a mixed stream I cannot indicate I want to keep a certain type for a duration of X and another type for a duration of Y
2. **I cannot say I just keep 100 members**
   This used to be possible as an exotic case of the LatestVersionSubset retention policy, but we agreed this would be a bad contract
3. I cannot state I want to set a retention policy using a **different version key** or a **different timestampPath** than what's documented on the LDES
   Should we allow to redefine the ldes:timestampPath here? What property would we propose?

Pull request ready

Transactions and retention policy rework

https://github.com/SEMICeu/LinkedDataEventStreams/pull/73

What’s the decision?

1. Accept and merge
2. Conditional accept: merge when conditions are validated.
   Conditions being…
3. Full reject

#LDES

European Commission

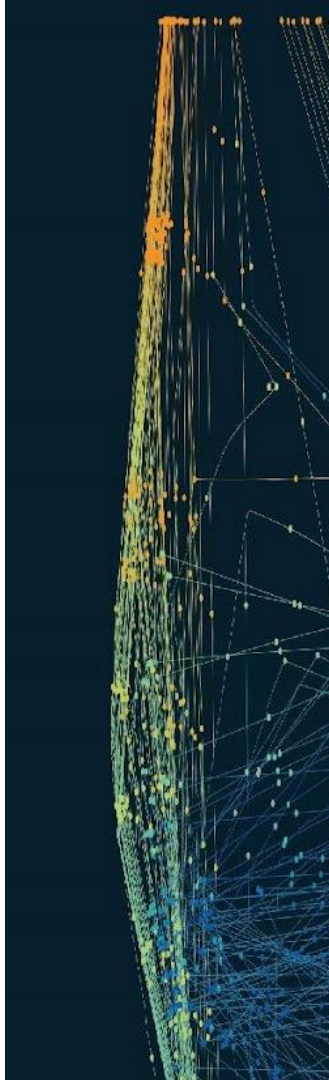
Communication channels

- Online meetings: next one is next week Wednesday same time
- Physical meetings: SEMIC2025
- Core discussions documented in Github issues: https://github.com/SEMICeu/LinkedDataEventStreams/issues/
- Matrix channel: https://matrix.to/#/#ldes:chat.semantic.works
- Want to follow up all the working groups and receive the reports?
  Leave your e-mail address in the chat!

Thanks for joining and see you next month for the second workshop!

#LDES

European Commission



Upcoming LDES Events

LDES Community Events

22 May

LDES Community Workshop

10:00 – 12:00 CEST

25-26 November

Meet us at SEMIC conference 2025

LDES Standardisation Track

30 April

LDES Working Group Onboarding Session

14:00 – 16:00 CEST

7 May

LDES Working Group I

14:00 – 16:00 CEST

12 June

LDES Working Group II

14:00 – 16:00 CEST

3 July

LDES Working Group III

14:00 – 16:00 CEST

Sneak peak of topics

- British Oceanographic Data Centre
- Exchange of industrial emission data
- Chatbot on mobility data
- Base registry of address data
- …

➡ **Register now** via the link in the chat

#LDES

European Commission