tlog-addenda

for attaching stuff

for conventions: mirror, browse, etc

and for redactability

transparency logs are mostly about proof of existence of data



transparency logs are mostly about proof of existence of data

and ordering it



transparency logs are mostly about proof of existence of data

and ordering it

not about actually conveying it



some logs do support freetext entries

some refuse this (for poisoning resistence)

even where it's supported,
it's not necessarily meant
for *large* content.

So let’s store data
beside the log.

So let's store data *beside* the log.

If data is *beside* the log...

we can content-address for dedup

and we can redact
(for poisoning resistance)

how much size is dedup worth?

case study, static-ct "issuers" dedup:

- **~1B entries, only ~100s issuers.**
  - Content-addressing the issuers makes a Static CT log ~2.8x smaller at rest.
- 1.1 KB cert -> 32-byte hash
- this adds up to ~3TB / 6mo at current CT growth rates!
- and that's bandwidth needed for monitors, too!

What do we need?

What do we need?

... nothing

just do as thou wilt

What do we need?

... nothing

just do as thou wilt

call it business logic and move on

What do we need?

... nothing

just do as thou wilt

call it business logic and move on

... well unless you want an ecosystem of tools

What do we ~~need~~ want?

What do we ~~need~~ want?

- yes, some dedup (see static-ct case study).

- yes, redaction is subtle. let's define it.

What do we ~~need~~ want?

- yes, some dedup (see static-ct case study).

- yes, redaction is subtle. let's define it.

let's standardize these

What do we ~~need~~ want?

- yes, some dedup (see static-ct case study).

- yes, redaction is subtle. let's define it.

... but also let's have nice things:

What do we ~~need~~ want?

- yes, some dedup (see static-ct case study).

- yes, redaction is subtle. let's define it.

- generic log viewer tools!

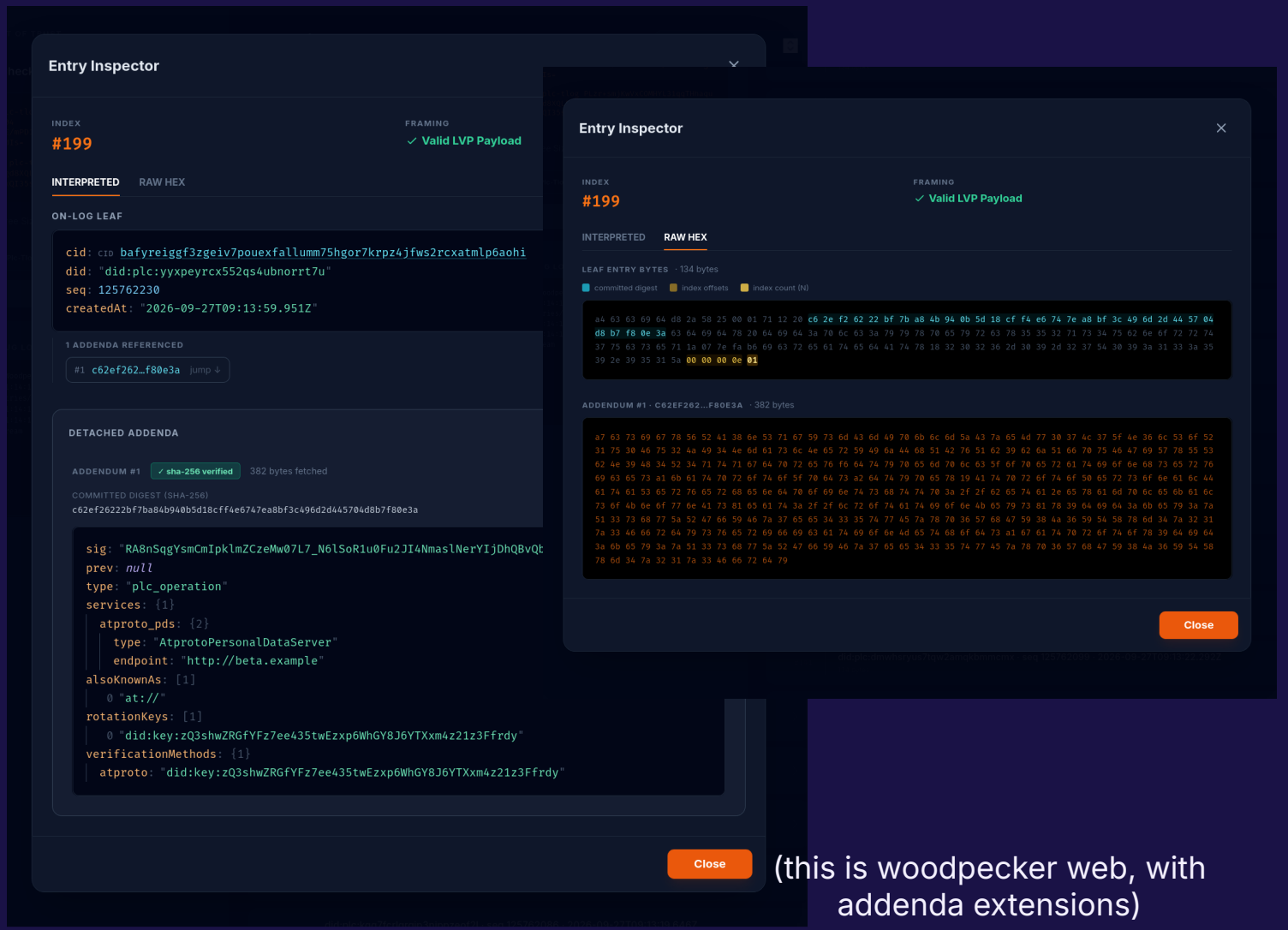
    These should be able to read addenda too!

- generic mirroring tools!

    and helper libs for monitors!

spoiler

we're going roughly here:



(this is woodpecker web, with addenda extensions)

What do we need?

(mechanically)

What do we need?

- way to signal that a log is using this convention

What do we need?

- way to signal that a log is using this convention

- way to parse log entries for addenda refs

What do we need?

- way to signal that a log is using this convention
- way to parse log entries for addenda refs
- path patterns for reading addenda blobs

What do we need?

- way to signal that a log is using this convention
- way to parse log entries for addenda refs
- path patterns for reading addenda blobs
- *minimal else.*

What do we need?

- way to signal that a log is using this convention

- way to parse log entries for addenda refs

- path patterns for reading addenda blobs

- *minimal else.*

What do we need?

- way to signal that a log is using this convention

  → **marker file: ./addenda.v1**

- way to parse log entries for addenda refs

- path patterns for reading addenda blobs

- *minimal else.*

What do we need?

- way to signal that a log is using this convention

  → marker file: ./addenda.v1

- way to parse log entries for addenda refs

  → trailer format: suffix the entry with indexes of addenda refs

- path patterns for reading addenda blobs

- *minimal else.*

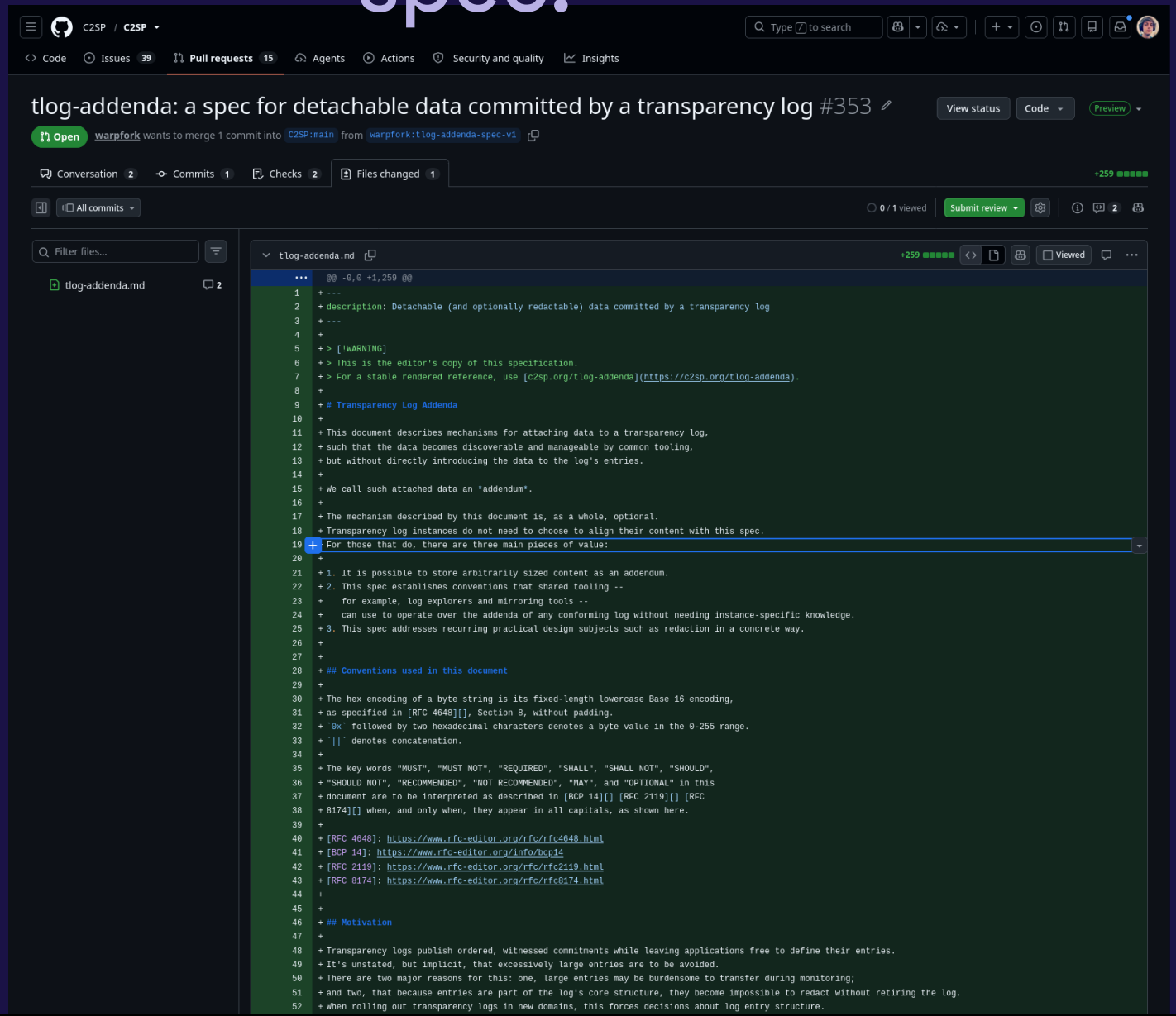
What do we need?

- way to signal that a log is using this convention
  → **marker file: ./addenda.v1**

- way to parse log entries for addenda refs
  → **trailer format: suffix the entry with indexes of addenda refs**
- path patterns for reading addenda blobs
  → **simple content addressing under ./addenda/ab/cd/abcd123f**

- *minimal else.*

so now there's a spec. (draft.)



to see that trailer
format...

let’s zoom back in on our
demo content

INTERPRETED **RAW HEX**

LEAF ENTRY BYTES · 134 bytes

committed digest · index offsets · index count (N)

```
a4 63 63 69 64 d8 2a 58 25 00 01 71 12 20 c6 2e f2 62 22 bf 7b a8 4b 94 0b 5d 18 cf f4 e6 74 7e a8 bf 3c 49 6d 2d 44 57 04
d8 b7 f8 0e 3a 63 64 69 64 78 20 64 69 64 3a 70 6c 63 3a 79 79 78 70 65 79 72 63 78 35 35 32 71 73 34 75 62 6e 6f 72 72 74
37 75 63 73 65 71 1a 07 7e fa b6 69 63 72 65 61 74 65 64 41 74 78 18 32 30 32 36 2d 30 39 2d 32 37 54 30 39 3a 31 33 3a 35
39 2e 39 35 31 5a 00 00 00 0e 01
```

ADDENDUM #1 · C62EF262…F80E3A · 382 bytes

```
a7 63 73 69 67 78 56 52 41 38 6e 53 71 67 59 73 6d 43 6d 49 70 6b 6c 6d 5a 43 7a 65 4d 77 30 37 4c 37 5f 4e 36 6c 53 6f 52
31 75 30 46 75 32 4a 49 34 4e 6d 61 73 6c 4e 65 72 59 49 6a 44 68 51 42 76 51 62 39 62 6a 51 66 70 75 46 47 69 57 78 55 53
62 4e 39 48 34 52 34 71 74 71 67 64 70 72 65 76 f6 64 74 79 70 65 6d 70 6c 63 5f 6f 70 65 72 61 74 69 6f 6e 68 73 65 72 76
69 63 65 73 a1 6b 61 74 70 72 6f 74 6f 5f 70 64 73 a2 64 74 79 70 65 78 19 41 74 70 72 6f 74 6f 50 65 72 73 6f 6e 61 6c 44
61 74 61 53 65 72 76 65 72 68 65 6e 64 70 6f 69 6e 74 73 68 74 74 70 3a 2f 2f 62 65 74 61 2e 65 78 61 6d 70 6c 65 6b 61 6c
73 6f 4b 6e 6f 77 6e 41 73 81 65 61 74 3a 2f 2f 6c 72 6f 74 61 74 69 6f 6e 4b 65 79 73 81 78 39 64 69 64 3a 6b 65 79 3a 7a
51 33 73 68 77 5a 52 47 66 59 46 7a 37 65 65 34 33 35 74 77 45 7a 78 70 36 57 68 47 59 38 4a 36 59 54 58 78 6d 34 7a 32 31
7a 33 46 66 72 64 79 73 76 65 72 69 66 69 63 61 74 69 6f 6e 4d 65 74 68 6f 64 73 a1 67 61 74 70 72 6f 74 6f 78 39 64 69 64
3a 6b 65 79 3a 7a 51 33 73 68 77 5a 52 47 66 59 46 7a 37 65 65 34 33 35 74 77 45 7a 78 70 36 57 68 47 59 38 4a 36 59 54 58
78 6d 34 7a 32 31 7a 33 46 66 72 64 79
```

INTERPRETED **RAW HEX**

LEAF ENTRY BYTES · 134 bytes

committed digest · index offsets · index count (N)

```
a4 63 63 69 64 d8 2a 58 25 00 01 71 12 20 c6 2e f2 62 22 bf 7b a8 4b 94 0b 5d 18 cf f4 e6 74 7e a8 bf 3c 49 6d 2d 44 57 04
d8 b7 f8 0e 3a 63 64 69 64 78 20 64 69 64 3a 70 6c 63 3a 79 79 78 70 65 79 72 63 78 35 35 32 71 73 34 75 62 6e 6f 72 72 74
37 75 63 73 65 71 1a 07 7e fa b6 69 63 72 65 61 74 65 64 41 74 78 18 32 30 32 36 2d 30 39 2d 32 37 54 30 39 3a 31 33 3a 35
39 2e 39 35 31 5a 00 00 00 0e 01
```

Last: N addenda refs

ADDENDUM #1 · C62EF262…F80E3A · 382 bytes

```
a7 63 73 69 67 78 56 52 41 38 6e 53 71 67 59 73 6d 43 6d 49 70 6b 6c 6d 5a 43 7a 65 4d 77 30 37 4c 37 5f 4e 36 6c 53 6f 52
31 75 30 46 75 32 4a 49 34 4e 6d 61 73 6c 4e 65 72 59 49 6a 44 68 51 42 76 51 62 39 62 6a 51 66 70 75 46 47 69 57 78 55 53
62 4e 39 48 34 52 34 71 74 71 67 64 70 72 65 76 f6 64 74 79 70 65 6d 70 6c 63 5f 6f 70 65 72 61 74 69 6f 6e 68 73 65 72 76
69 63 65 73 a1 6b 61 74 70 72 6f 74 6f 5f 70 64 73 a2 64 74 79 70 65 78 19 41 74 70 72 6f 74 6f 50 65 72 73 6f 6e 61 6c 44
61 74 61 53 65 72 76 65 72 68 65 6e 64 70 6f 69 6e 74 73 68 74 74 70 3a 2f 2f 62 65 74 61 2e 65 78 61 6d 70 6c 65 6b 61 6c
73 6f 4b 6e 6f 77 6e 41 73 81 65 61 74 3a 2f 2f 6c 72 6f 74 61 74 69 6f 6e 4b 65 79 73 81 78 39 64 69 64 3a 6b 65 79 3a 7a
51 33 73 68 77 5a 52 47 66 59 46 7a 37 65 65 34 33 35 74 77 45 7a 78 70 36 57 68 47 59 38 4a 36 59 54 58 78 6d 34 7a 32 31
7a 33 46 66 72 64 79 73 76 65 72 69 66 69 63 61 74 69 6f 6e 4d 65 74 68 6f 64 73 a1 67 61 74 70 72 6f 74 6f 78 39 64 69 64
3a 6b 65 79 3a 7a 51 33 73 68 77 5a 52 47 66 59 46 7a 37 65 65 34 33 35 74 77 45 7a 78 70 36 57 68 47 59 38 4a 36 59 54 58
78 6d 34 7a 32 31 7a 33 46 66 72 64 79
```

INTERPRETED **RAW HEX**

LEAF ENTRY BYTES · 134 bytes

committed digest · index offsets · index count (N)

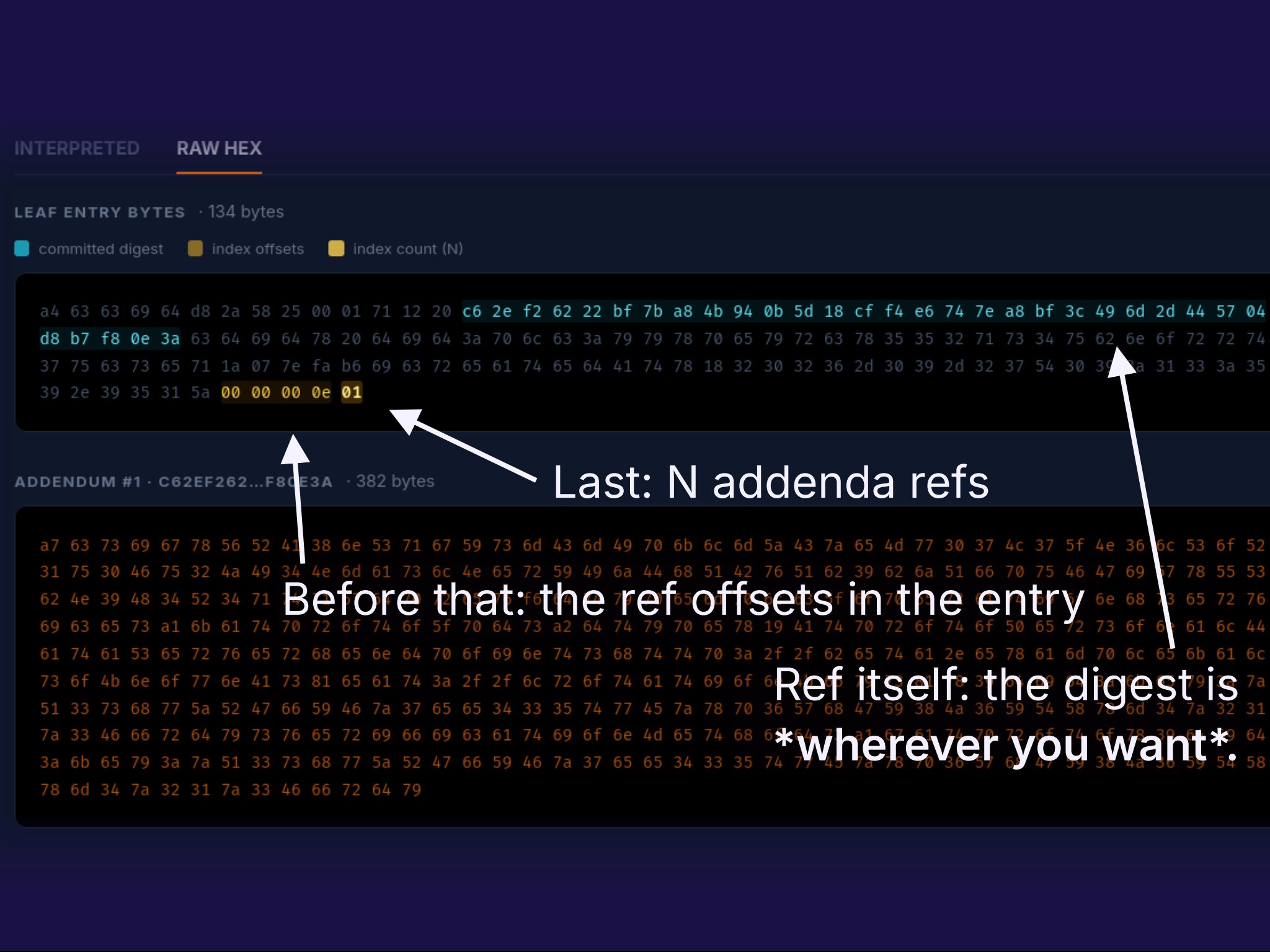
```
a4 63 63 69 64 d8 2a 58 25 00 01 71 12 20 c6 2e f2 62 22 bf 7b a8 4b 94 0b 5d 18 cf f4 e6 74 7e a8 bf 3c 49 6d 2d 44 57 04
d8 b7 f8 0e 3a 63 64 69 64 78 20 64 69 64 3a 70 6c 63 3a 79 79 78 70 65 79 72 63 78 35 35 32 71 73 34 75 62 6e 6f 72 72 74
37 75 63 73 65 71 1a 07 7e fa b6 69 63 72 65 61 74 65 64 41 74 78 18 32 30 32 36 2d 30 39 2d 32 37 54 30 39 3a 31 33 3a 35
39 2e 39 35 31 5a 00 00 00 0e 01
```

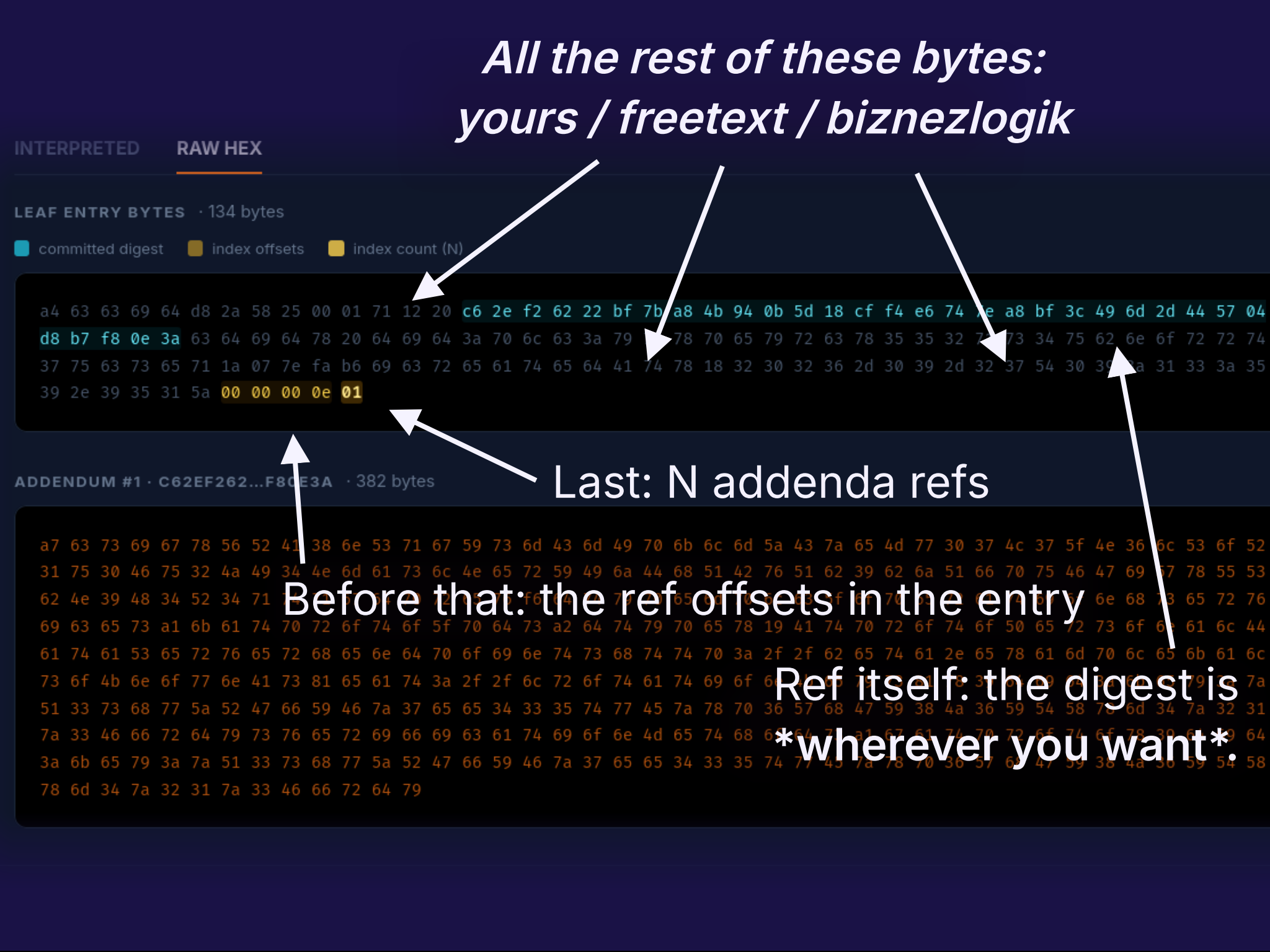
ADDENDUM #1 · **C62EF262…F8CE3A** · 382 bytes

```
a7 63 73 69 67 78 56 52 41 38 6e 53 71 67 59 73 6d 43 6d 49 70 6b 6c 6d 5a 43 7a 65 4d 77 30 37 4c 37 5f 4e 36 6c 53 6f 52
31 75 30 46 75 32 4a 49 34 4e 6d 61 73 6c 4e 65 72 59 49 6a 44 68 51 42 76 51 62 39 62 6a 51 66 70 75 46 47 69 57 78 55 53
62 4e 39 48 34 52 34 71 [illegible] 6e 68 73 65 72 76
69 63 65 73 a1 6b 61 74 70 72 6f 74 6f 5f 70 64 73 a2 64 74 79 70 65 78 19 41 74 70 72 6f 74 6f 50 65 72 73 6f 6e 61 6c 44
61 74 61 53 65 72 76 65 72 68 65 6e 64 70 6f 69 6e 74 73 68 74 74 70 3a 2f 2f 62 65 74 61 2e 65 78 61 6d 70 6c 65 6b 61 6c
73 6f 4b 6e 6f 77 6e 41 73 81 65 61 74 3a 2f 2f 6c 72 6f 74 61 74 69 6f 6e 4b 65 79 73 81 78 39 64 69 64 3a 6b 65 79 3a 7a
51 33 73 68 77 5a 52 47 66 59 46 7a 37 65 65 34 33 35 74 77 45 7a 78 70 36 57 68 47 59 38 4a 36 59 54 58 78 6d 34 7a 32 31
7a 33 46 66 72 64 79 73 76 65 72 69 66 69 63 61 74 69 6f 6e 4d 65 74 68 6f 64 73 a1 67 61 74 70 72 6f 74 6f 78 39 64 69 64
3a 6b 65 79 3a 7a 51 33 73 68 77 5a 52 47 66 59 46 7a 37 65 65 34 33 35 74 77 45 7a 78 70 36 57 68 47 59 38 4a 36 59 54 58
78 6d 34 7a 32 31 7a 33 46 66 72 64 79
```

Last: N addenda refs

Before that: the ref offsets in the entry


INTERPRETED
RAW HEX
LEAF ENTRY BYTES · 134 bytes
committed digest
index offsets
index count (N)
a4 63 63 69 64 d8 2a 58 25 00 01 71 12 20 c6 2e f2 62 22 bf 7b a8 4b 94 0b 5d 18 cf f4 e6 74 7e a8 bf 3c 49 6d 2d 44 57 04
d8 b7 f8 0e 3a 63 64 69 64 78 20 64 69 64 3a 70 6c 63 3a 79 79 78 70 65 79 72 63 78 35 35 32 71 73 34 75 62 6e 6f 72 72 74
37 75 63 73 65 71 1a 07 7e fa b6 69 63 72 65 61 74 65 64 41 74 78 18 32 30 32 36 2d 30 39 2d 32 37 54 30 39 3a 31 33 3a 35
39 2e 39 35 31 5a 00 00 00 0e 01
ADDENDUM #1 · C62EF262...F8CE3A · 382 bytes
Last: N addenda refs
Before that: the ref offsets in the entry
Ref itself: the digest is
wherever you want.


All the rest of these bytes:
yours / freetext / biznezlogik
INTERPRETED
RAW HEX
LEAF ENTRY BYTES · 134 bytes
committed digest
index offsets
index count (N)
a4 63 63 69 64 d8 2a 58 25 00 01 71 12 20 c6 2e f2 62 22 bf 7b a8 4b 94 0b 5d 18 cf f4 e6 74
d8 b7 f8 0e 3a 63 64 69 64 78 20 64 69 64 3a 70 6c 63 3a 79
37 75 63 73 65 71 1a 07 7e fa b6 69 63 72 65 61 74 65 64 41
39 2e 39 35 31 5a 00 00 00 0e 01
ADDENDUM #1 · C62EF262...F8
382 bytes
Last: N addenda refs
Before that: the ref offsets in the entry
Ref itself: the digest is
wherever you want.

In the interpreted view:

#199 ✓ Valid LVP Payload

INTERPRETED RAW HEX

ON-LOG LEAF 134 bytes

```
cid: CID bafyreiggf3zgeiv7pouexfallumm75hgor7krpz4jfws2rcxatmlp6aohi
did: "did:plc:yyxpeyrcx552qs4ubnorrt7u"
seq: 125762230
createdAt: "2026-09-27T09:13:59.951Z"
```

1 ADDENDA REFERENCED

#1 c62ef262…f80e3a jump ↓

DETACHED ADDENDA

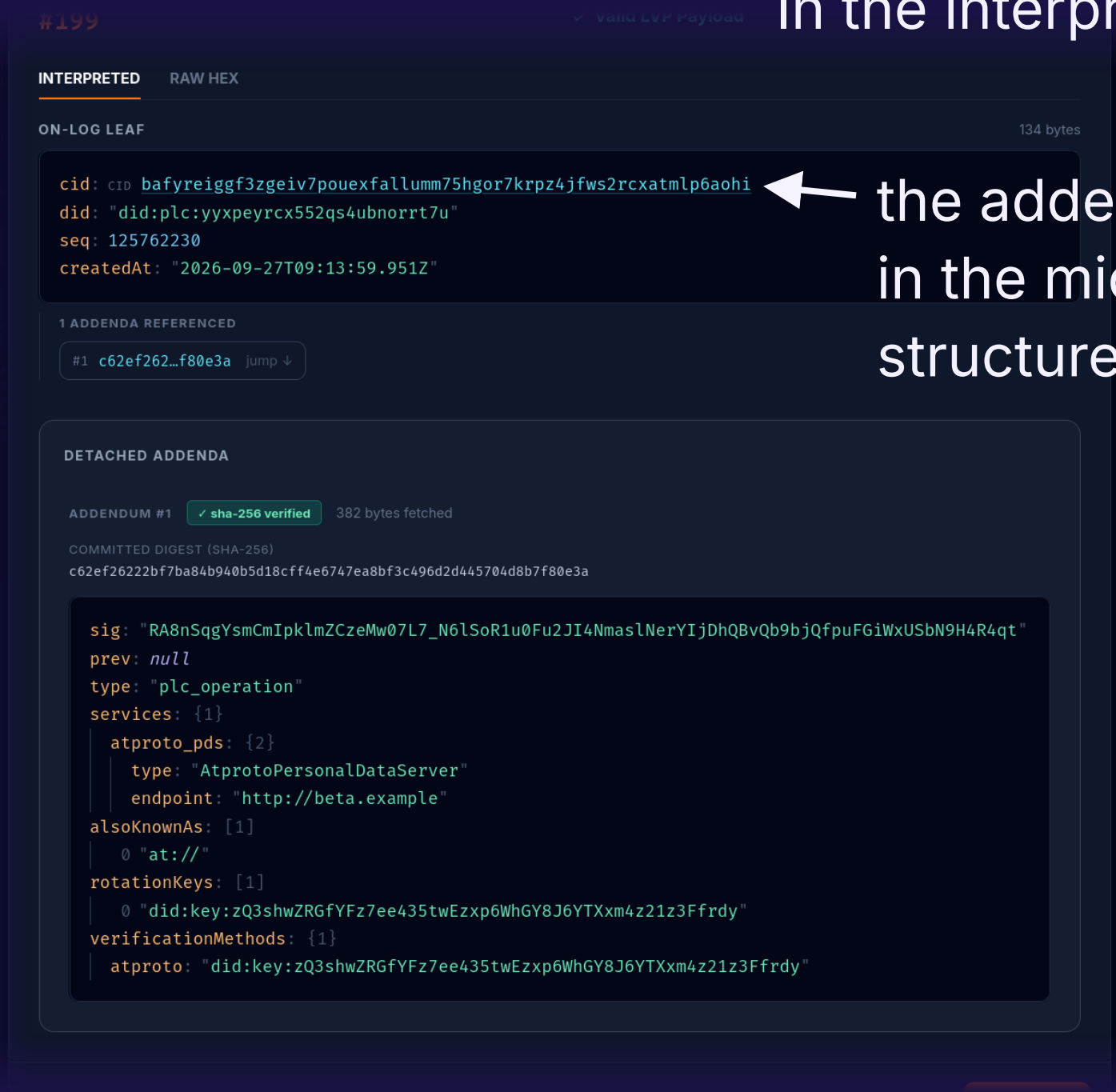
ADDENDUM #1 ✓ sha-256 verified 382 bytes fetched

COMMITTED DIGEST (SHA-256)

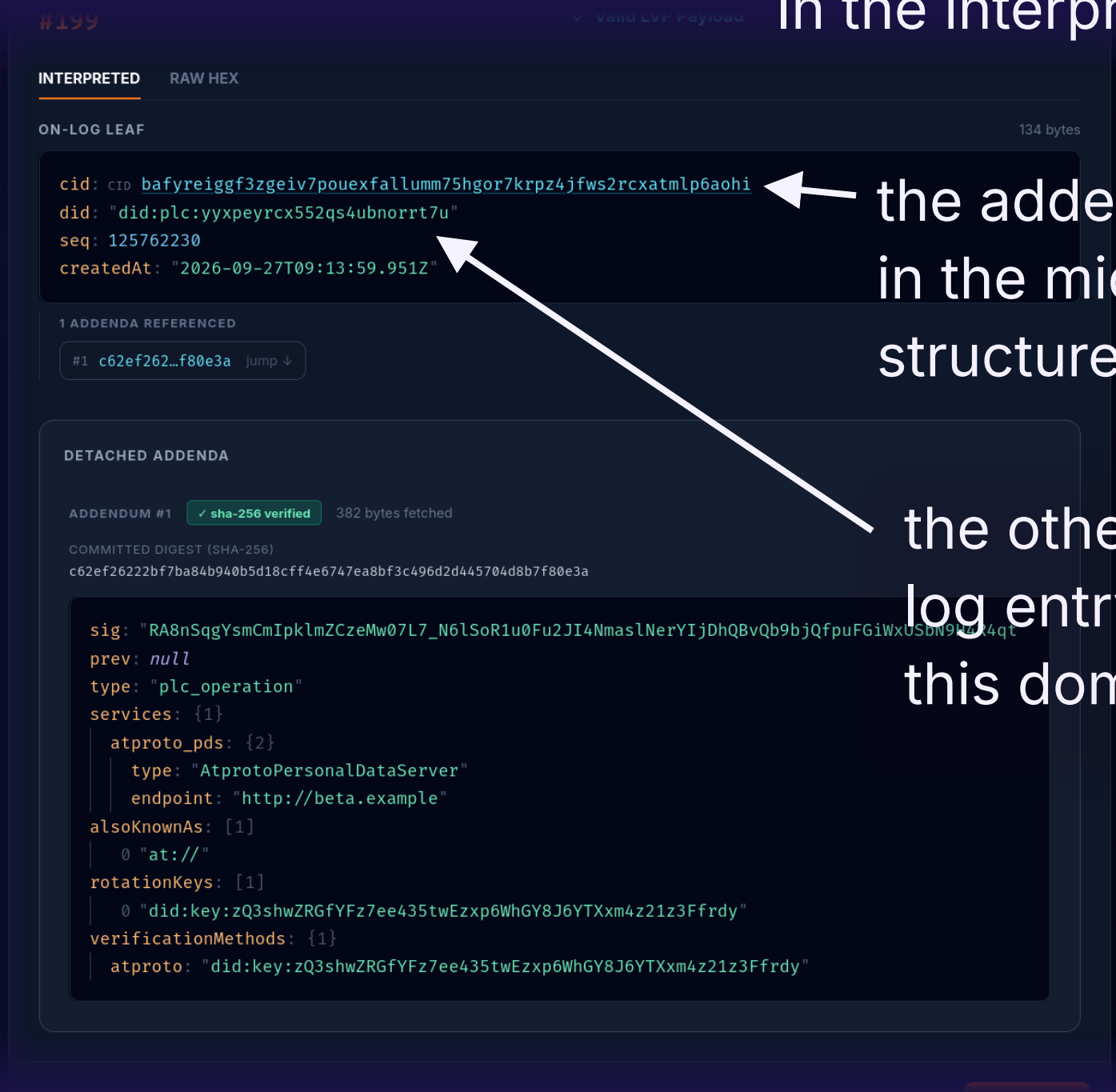
c62ef26222bf7ba84b940b5d18cff4e6747ea8bf3c496d2d445704d8b7f80e3a

```
sig: "RA8nSqgYsmCmIpklmZCzeMw07L7_N6lSoR1u0Fu2JI4NmaslNerYIjDhQBvQb9bjQfpuFGiWxUSbN9H4R4qt"
prev: null
type: "plc_operation"
services: {1}
  atproto_pds: {2}
    type: "AtprotoPersonalDataServer"
    endpoint: "http://beta.example"
alsoKnownAs: [1]
  0 "at://"
rotationKeys: [1]
  0 "did:key:zQ3shwZRGfYFz7ee435twEzxp6WhGY8J6YTXxm4z21z3Ffrdy"
verificationMethods: {1}
  atproto: "did:key:zQ3shwZRGfYFz7ee435twEzxp6WhGY8J6YTXxm4z21z3Ffrdy"
```

In the interpreted view:



the addenda digest is in the middle of this structured data



In the interpreted view:

the addenda digest is in the middle of this structured data

the other things in the log entry are used in this domain for...

now, about redaction

now, about redaction

redaction is pretty simple

you just stop serving the thing

now, about redaction

addenda isn't *in* the log
→ redacting it doesn't *break* the log.

now, about redaction

addenda isn't *in* the log
→ redacting it doesn't *break* the log.

but...

now, about redaction

consider what you still want
to monitor for.

now, about redaction

consider what you still want to monitor for.

If you want "lawful monitors" -- someone who checks some pattern holds true across related log entries...

now, about redaction

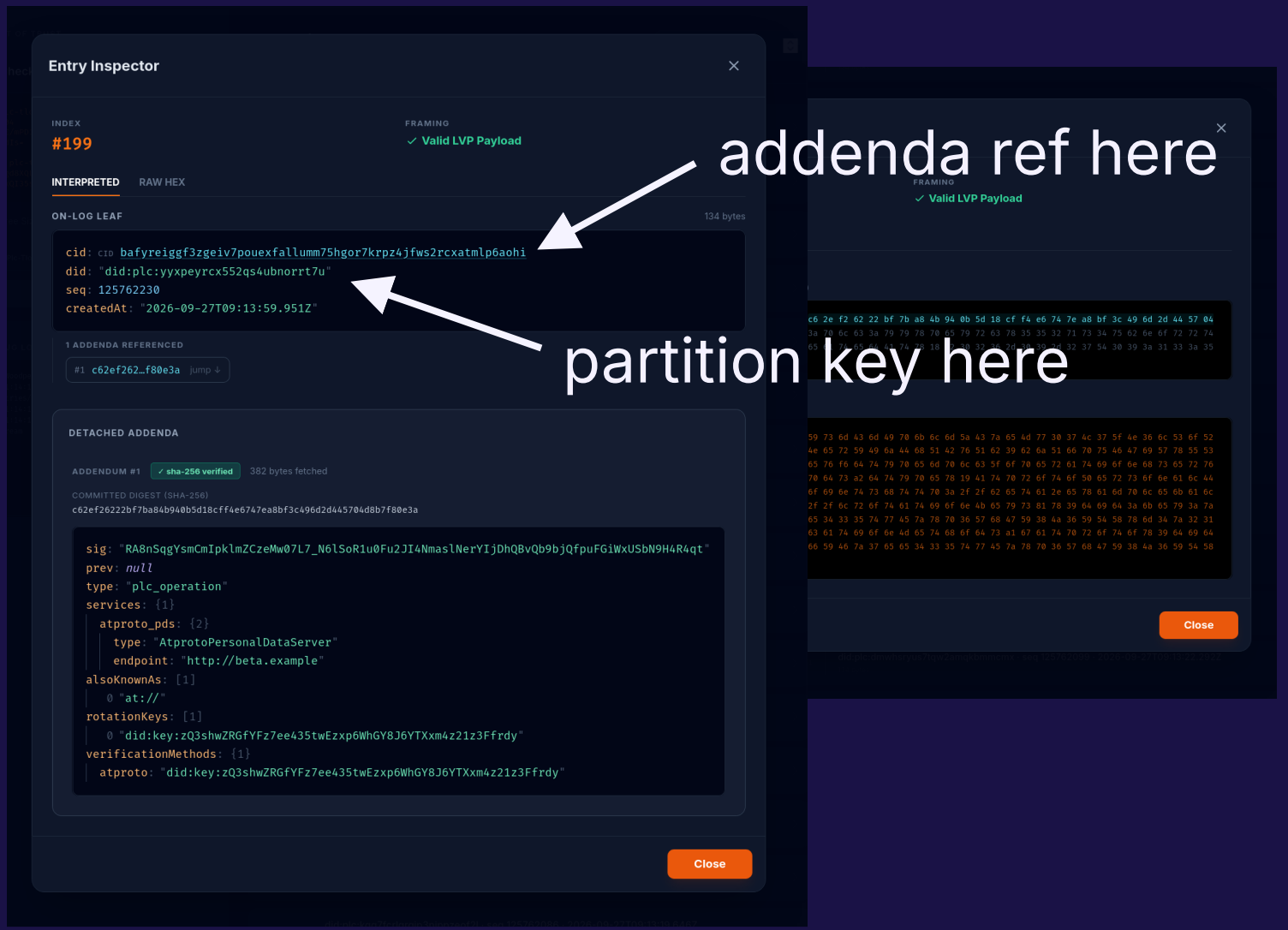
consider what you still want to monitor for.

If you want "lawful monitors" --
someone who checks some pattern
holds true across related log entries...

then you need to
know what's "related"

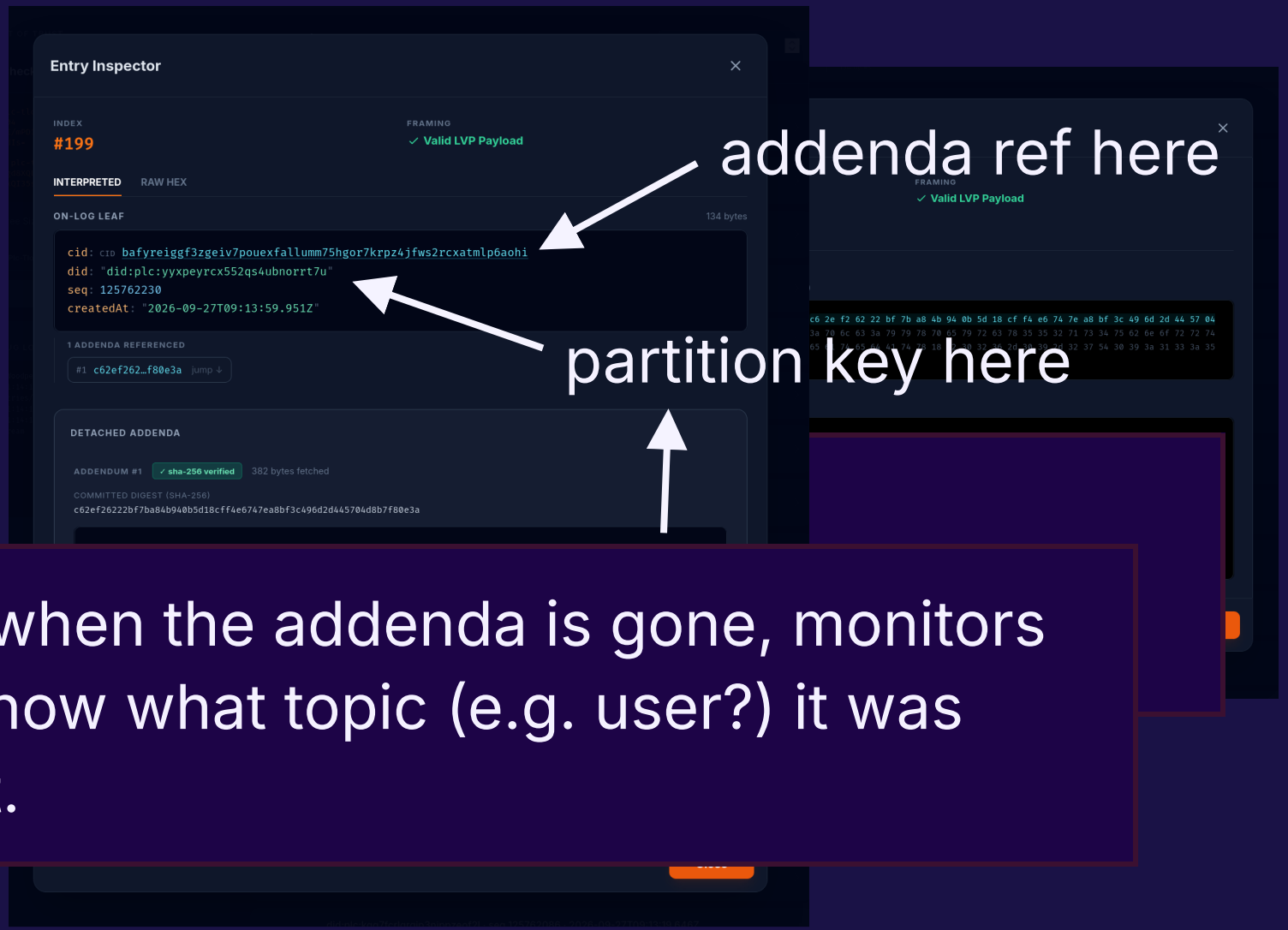
back to our demo...

this is why there's still some data in this entry.



back to our demo...

this is why there's still some data in this entry.



Now when the addenda is gone, monitors still know what topic (e.g. user?) it was about.

(btw geomys is cool)

- spec work
- owls
- we operate CT logs!
- we operate witnesses!



GEOMYS

(btw geomys is cool)

- spec work
- owls
- we operate CT logs!
- we operate witnesses!
- an MTC CA!
- KT auditors
- ... and infrastructure such as that!



GEOMYS

fin

tlog-addenda

@warpfork.bsky.social

warpfork@geomys.org

errata!

- yes you can do this with entryless logs (like sigsum) -- kinda
  - only one addenda at a time; do CAS with H(H(x))...
  - this is not currently described in the v1 of this spec though.