

qlog

draft-ietf-quic-qlog-main-schema

draft-ietf-quic-qlog-quic-events

draft-ietf-quic-qlog-h3-events

Robin Marx, Luca Niccolini, Marten Seemann,
Lucas Pardue

IETF 121, Dublin
2024-11-06

Since draft-ietf-quic-qlog-main-schema-09:

- Solved the clocks/timestamp issues
- Spotted one gap in schema extensibility - fixed it
 - Net outcome is an even simpler data model
 - No more categories
 - Allows extending existing events/types AND **add to existing namespaces**

Simpler, see

Name value	Importance	Definition
connectivity:server_listening	Extra	Section 4.1
connectivity:connection_started	Base	Section 4.2
connectivity:connection_closed	Base	Section 4.3
connectivity:connection_id_updated	Base	Section 4.4
connectivity:spin_bit_updated	Base	Section 4.5
connectivity:connection_state_updated	Base	Section 4.6
connectivity:path_assigned	Base	Section 4.7
connectivity:mtu_updated	Extra	Section 4.8
quic:version_information	Core	Section 5.1
quic:alpn_information	Core	Section 5.2
quic:parameters_set	Core	Section 5.3

Name value	Importance	Definition
quic:server_listening	Extra	Section 4.1
quic:connection_started	Base	Section 4.2
quic:connection_closed	Base	Section 4.3
quic:connection_id_updated	Base	Section 4.4
quic:spin_bit_updated	Base	Section 4.5
quic:connection_state_updated	Base	Section 4.6
quic:path_assigned	Base	Section 4.7
quic:mtu_updated	Extra	Section 4.8
quic:version_information	Core	Section 5.1
quic:alpn_information	Core	Section 5.2
quic:parameters_set	Core	Section 5.3

Is the loss_timer_updated event sufficient? [#319](#)

Unclear if current event is enough to really help debug CC loss.

Could use review from users that have used qlog for this purpose.

```
QUICLossTimerUpdated = {  
    ; called "mode" in RFC 9002 A.9.  
    ? timer_type: "ack" /  
        "pto"  
    ? packet_number_space: $PacketNumberSpace  
    event_type: "set" /  
        "expired" /  
        "cancelled"  
  
    ; if event_type == "set": delta time is in ms from  
    ; this event's timestamp until when the timer will trigger  
    ? delta: float32  
  
    * $$quic-losstimerupdated-extension  
}
```

Add events to explicitly indicate send blocking [#132](#) [#444](#)

Proposal:

add new field to **stream_data_moved** and **datagram_data_moved**

? blocked_reason: \$BlockedReason

\$BlockedReason /= "scheduling" /

```
typedef enum QUIC_FLOW_BLOCK_REASON {  
    QUIC_FLOW_BLOCKED_SCHEDULING      = 0x01,  
    QUIC_FLOW_BLOCKED_PACING           = 0x02,  
    QUIC_FLOW_BLOCKED_AMPLIFICATION_PROT = 0x04,  
    QUIC_FLOW_BLOCKED_CONGESTION_CONTROL = 0x08,  
    QUIC_FLOW_BLOCKED_CONN_FLOW_CONTROL = 0x10,  
    QUIC_FLOW_BLOCKED_STREAM_ID_FLOW_CONTROL = 0x20,  
    QUIC_FLOW_BLOCKED_STREAM_FLOW_CONTROL = 0x40,  
    QUIC_FLOW_BLOCKED_APP              = 0x80  
} QUIC_FLOW_BLOCK_REASON;
```

"pacing" /

"amplification_protection" /

"congestion_control" /

"connection_flow_control" /

"stream_flow_control" /

"stream_id_flow_control" /

"application"

JSON-SEQ is painful - [#434](#)



Current Streaming Serialization Format

"A qlog file using the QlogFileSeq schema can be serialized to a **streamable** JSON format called JSON Text Sequences (JSON-SEQ) ([RFC7464])."

- RFC 7464 registered media type - **application/json-seq**
- RFC 8091 registered structured syntax suffix - **+json-seq**
- qlog:
 - "When mapping qlog to JSON-SEQ, the Media Type is "application/qlog+json-seq""


```

R_S {"qlog_version":"0.3","qlog_format":"JSON-SEQ","title":"quiche-server qlog","description":"quiche-server
R_S {"time":0.0,"name":"quic:parameters_set","data":{"owner":"local","tls_cipher":"None","original_destinatio
R_S {"time":0.181675,"name":"quic:packet_received","data":{"header":{"packet_type":"initial","packet_number":
R_S {"time":0.181675,"name":"quic:metrics_updated","data":{"smoothed_rtt":333.0,"rtt_variance":166.5,"congest
R_S {"time":4.6397524,"name":"quic:packet_sent","data":{"header":{"packet_type":"initial","packet_number":0,"
R_S {"time":4.6397524,"name":"quic:metrics_updated","data":{"bytes_in_flight":167}}
R_S {"time":4.6397524,"name":"quic:packet_sent","data":{"header":{"packet_type":"handshake","packet_number":0
R_S {"time":4.6397524,"name":"quic:metrics_updated","data":{"bytes_in_flight":1200}}
R_S {"time":4.885065,"name":"quic:packet_sent","data":{"header":{"packet_type":"handshake","packet_number":1,
R_S {"time":4.885065,"name":"quic:metrics_updated","data":{"bytes_in_flight":1671}}
R_S {"time":666.39166,"name":"quic:packet_received","data":{"header":{"packet_type":"initial","packet_number"
R_S {"time":666.39166,"name":"quic:metrics_updated","data":{"min_rtt":661.75195,"smoothed_rtt":661.75195,"lat
R_S {"time":666.59204,"name":"quic:packet_received","data":{"header":{"packet_type":"handshake","packet_numbe
R_S {"time":666.59204,"name":"quic:metrics_updated","data":{"smoothed_rtt":661.777,"latest_rtt":661.95233,"rt
R_S {"time":705.4539,"name":"quic:packet_received","data":{"header":{"packet_type":"handshake","packet_number
R_S {"time":705.4539,"name":"quic:metrics_updated","data":{"smoothed_rtt":666.626,"latest_rtt":700.5689,"rtt

```


JSON-SEQ allows newlines within records

```
{ "qlog_version": "0.3", "qlog_format": "JSON-SEQ", "title": "h3i", "description": "h3i", "trace": { "vantage_point": { "type": "client" }, "records": [ { "time": 0.103657, "name": "http:frame_created", "data": { "stream_id": 0, "frame": { "frame_type": "headers", "headers": [ { "name": ":method", "value": "PUT" }, { "name": ":authority", "value": "lucaspardue.com" }, { "name": ":path", "value": "/" }, { "name": ":scheme", "value": "https" }, { "name": "content-length", "value": "4, \\r\\n 456" } ] } } } ] } }
```

```
{ "time": 0.173069, "name": "http:frame_created", "data": { "stream_id": 0, "frame": { "frame_type": "data" }, "fin_stream": true }
```

JSON-SEQ is not the only show in town

There are other streamable JSON formats in common use:

- NDJSON - <https://github.com/ndjson/ndjson-spec>
- JSON lines - <https://jsonlines.org/>

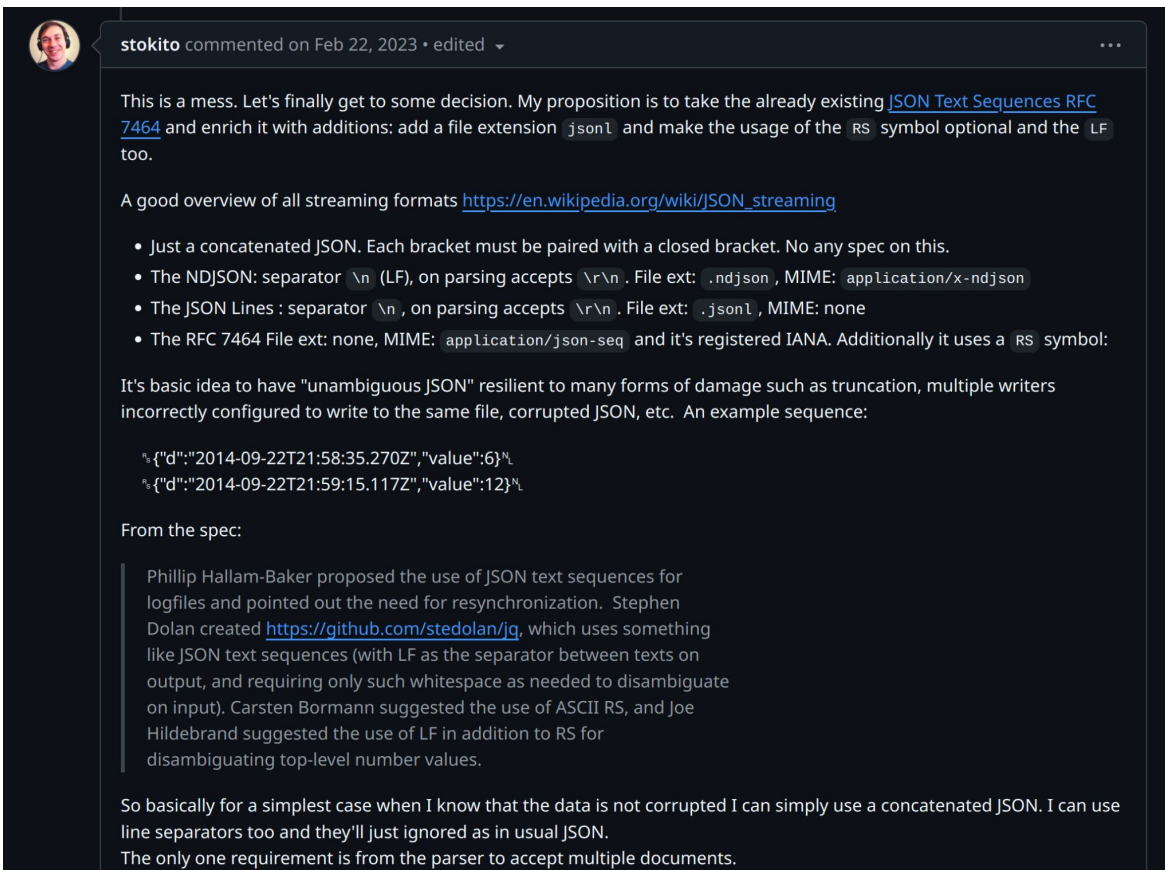
They claim to use media types, but none registered (yet)

- *"The MediaType [RFC6838] for Newline Delimited JSON SHOULD be application/x-ndjson."*
- *"MIME type may be application/jsonl"* - <https://github.com/wardi/jsonlines/issues/19>
- opened 2016 but activity as recent as July 2024

What's in a name

"jsonlines.org and
ndjson.org" -

<https://github.com/wardi/jsonlines/issues/22>



A screenshot of a GitHub comment by user 'stokito' from February 22, 2023. The comment discusses the JSON Text Sequences RFC 7464 and proposes a new file extension '.jsonl' and the use of the 'RS' symbol. It includes a list of four points about different JSON formats and their separators. It also shows an example of a JSON sequence with line separators and a quote from Phillip Hallam-Baker about the need for resynchronization in logfiles.

stokito commented on Feb 22, 2023 • edited

This is a mess. Let's finally get to some decision. My proposition is to take the already existing [JSON Text Sequences RFC 7464](#) and enrich it with additions: add a file extension `.jsonl` and make the usage of the `RS` symbol optional and the `LF` too.

A good overview of all streaming formats https://en.wikipedia.org/wiki/JSON_streaming

- Just a concatenated JSON. Each bracket must be paired with a closed bracket. No any spec on this.
- The NDJSON: separator `\n` (LF), on parsing accepts `\r\n`. File ext: `.ndjson`, MIME: `application/x-ndjson`
- The JSON Lines : separator `\n`, on parsing accepts `\r\n`. File ext: `.jsonl`, MIME: none
- The RFC 7464 File ext: none, MIME: `application/json-seq` and it's registered IANA. Additionally it uses a `RS` symbol:

It's basic idea to have "unambiguous JSON" resilient to many forms of damage such as truncation, multiple writers incorrectly configured to write to the same file, corrupted JSON, etc. An example sequence:

```
%{"d":"2014-09-22T21:58:35.270Z","value":6}%  
%{"d":"2014-09-22T21:59:15.117Z","value":12}%
```


From the spec:

Phillip Hallam-Baker proposed the use of JSON text sequences for logfiles and pointed out the need for resynchronization. Stephen Dolan created <https://github.com/stedolan/jq>, which uses something like JSON text sequences (with LF as the separator between texts on output, and requiring only such whitespace as needed to disambiguate on input). Carsten Bormann suggested the use of ASCII RS, and Joe Hildebrand suggested the use of LF in addition to RS for disambiguating top-level number values.

So basically for a simplest case when I know that the data is not corrupted I can simply use a concatenated JSON. I can use line separators too and they'll just ignored as in usual JSON.

The only one requirement is from the parser to accept multiple documents.

The sharp edges of JSON-SEQ

- Text viewer support varies wildly
 - 
- Doesn't play nice/out-of-the-box with general-purpose tools that are in the common toolkit - grep, vim, etc.
 - “Needs” a conversion step.
- jq, a common JSON processing tool, needs explicit --seq option
 - Fails with cryptic message otherwise
 - Tracking issue - <https://github.com/jqlang/jq/issues/3156>
 - Candidate PR - <https://github.com/jqlang/jq/pull/3191>
- Annoying to deal with, especially for newcomers

Counterpoints

- qlog is a structured logging format
- Most value comes from special-purpose tools
- Common data model, many serialization formats
 - Some defined today, more tomorrow
- Transforms are expected
 - There is no hard mandate for implementations to log in standardized formats
 - E.g. JSON-SEQ to some newline delimited format in one line ...

```
$ jq -c --seq . client-1928a4431b958983545fe79a00fcf7a670f39d8e.sqlog | head
{"qlog_version":"0.3","qlog_format":"JSON-SEQ","title":"quiche-client qlog","description":"quiche-client qlog i
{"time":0,"name":"transport:parameters_set","data":{"owner":"local","tls_cipher":"None","disable_active_migrati
{"time":0.078191,"name":"transport:packet_sent","data":{"header":{"packet_type":"initial","packet_number":0,"ve
{"time":0.078191,"name":"recovery:metrics_updated","data":{"smoothed_rtt":333,"rtt_variance":166.5,"congestion_
{"time":10.191873,"name":"transport:packet_sent","data":{"header":{"packet_type":"initial","packet_number":1,"v
{"time":28.821796,"name":"transport:packet_received","data":{"header":{"packet_type":"initial","packet_number":
{"time":28.821796,"name":"recovery:metrics_updated","data":{"min_rtt":18.629923,"smoothed_rtt":18.629923,"lates
{"time":29.419956,"name":"transport:packet_received","data":{"header":{"packet_type":"handshake","packet_number
{"time":29.551624,"name":"transport:packet_received","data":{"header":{"packet_type":"handshake","packet_number
{"time":29.551624,"name":"transport:parameters_set","data":{"owner":"remote","tls_cipher":"Some(AES128_GCM)","c
```

Do we really need to define a streaming format?

- Archaeology 2021:

- "Decide if streaming serialization needs to be defined" -

<https://github.com/quicwg/qlog/issues/172>

- *"During IETF 111 it was noted that [JSON-SEQ] is an IETF-defined format..."*
 - *"I've been using NDJSON and I like the simplicity, but the inability to have newlines in the JSON output was a point of contention for me" ... "The addition of the \x1E character is gross but solves the problem"*
 - *"My two cents would go to using NDJSON rather than JSON Text Sequences (RFC 7464), because the former is more widespread, easy to use with existing line-oriented tools, and because I do not think we need the properties being provided by RFC 7464 (e.g., process dataset in the middle ..."*
 - *"Thanks all for the additional input. We decided to try the JSON Text Sequences for now and see how it goes (doesn't mean we can't later switch to something else if Text Sequences turn out horrible somehow)."*

Proposed resolutions

1. **Do nothing** - keep JSON-SEQ and live with pain points
2. **Switch** recommended streaming serialization format
 - Line based not JSON-SEQ - but what?
3. **Add** another streaming serialization format
 - Optionally: define it in an ancillary document that is decoupled
4. **Don't normatively recommend** any streaming serialization

Wrapping up

- **Goal: WGLC by end of year**
- 22 issues + 2 PRs left (many editorial about extensibility)
- Implementation status of draft -10:
 - Updates to qvis are planned
 - Cloudflare quiche qlog crate has PR for all latest spec changes - merge to mainline pending on qvis
 - Other implementations have partial support