

[qlog]

structured event logging

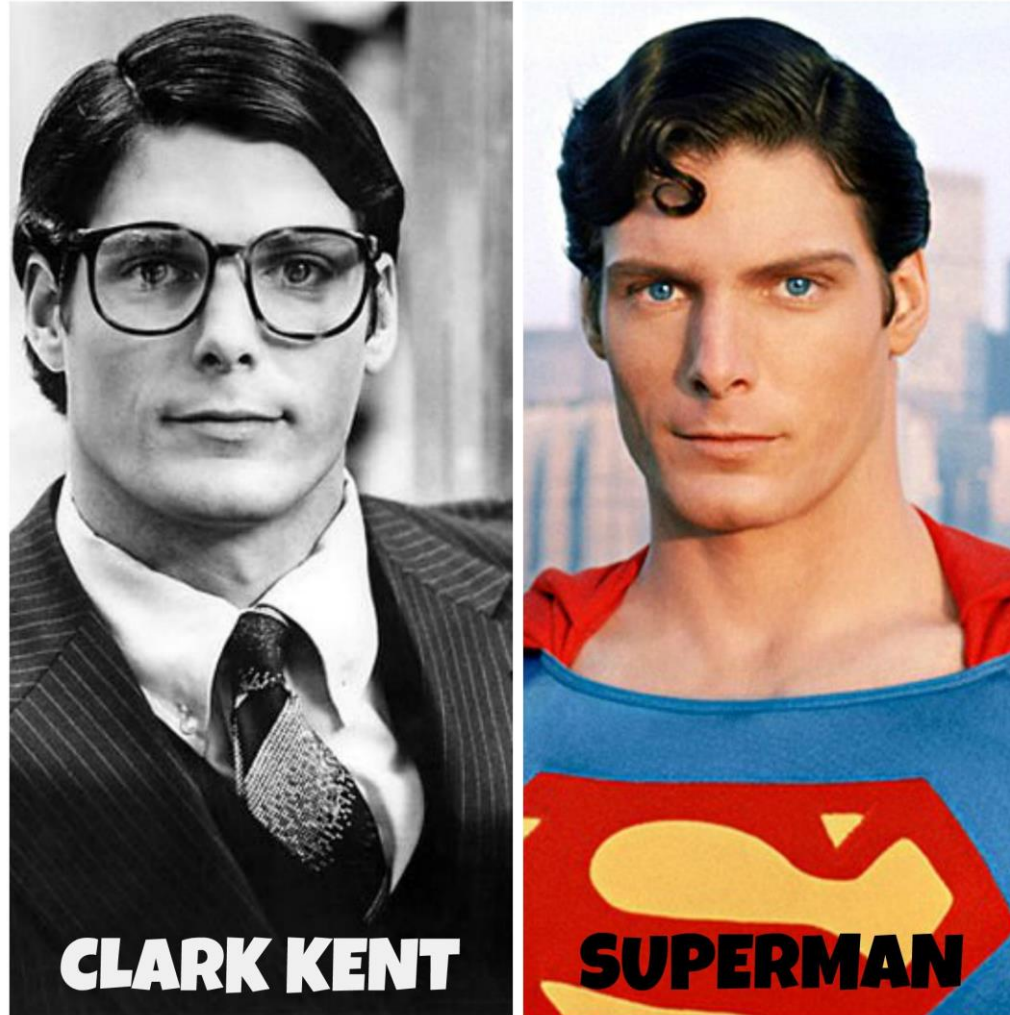
the Superman update



Robin Marx

robin.marx@kuleuven.be

Only editorial changes



<https://www.ietf.org/archive/id/draft-ietf-quic-qlog-main-schema-02.html>

<https://www.ietf.org/archive/id/draft-ietf-quic-qlog-h3-events-01.html>

<https://www.ietf.org/archive/id/draft-ietf-quic-qlog-quic-events-01.html>

Easy to go from TypeScript to CDDL

```
class QlogData {  
  name: string  
  packetCount?: number  
  location: VantagePoint  
  events: Array<Event>  
  raw: bytes  
  
  // any other fields are possible here  
}  
  
enum VantagePoint {  
  server,  
  client  
}
```

TypeScript-inspired

```
QlogData = {  
  name: text  
  ? packetCount: uint32  
  location: VantagePoint  
  events: [* Event]  
  raw: hexstring  
  
  * text ⇒ any  
}  
  
VantagePoint = "client" / "server"
```


Concise Data Definition Language
(**CDDL**, RFC 8610)

CDDL has other nifty features

regexp operator

```
enum CryptoError {  
    crypto_error_{TLS_ALERT}  
}
```

```
; from "crypto_error_0x100" to "crypto_error_0x199"  
CryptoError = text .regexp "crypto_error_0x1[0-9][0-9]"
```





unwrap operator ~

```
Event1 = {  
    ? field1: text  
    field2: uint32  
}
```

```
Event2 = {  
    ? field1: text  
    field2: uint32  
}
```

```
Event1 = {  
    ~SharedFields  
}  
Event2 = {  
    ~SharedFields  
}
```




```
SharedFields = {  
    ? field1: text  
    field2: uint32  
}
```

CDDL has other nifty features: extension sockets

Main
schema

```
class Event {  
  time: number  
  name: string  
  
  data: any  
}
```


```
Event = {  
  time: float64  
  name: text  
  
  data: $ProtocolEventBody  
}
```



HTTP/3
document

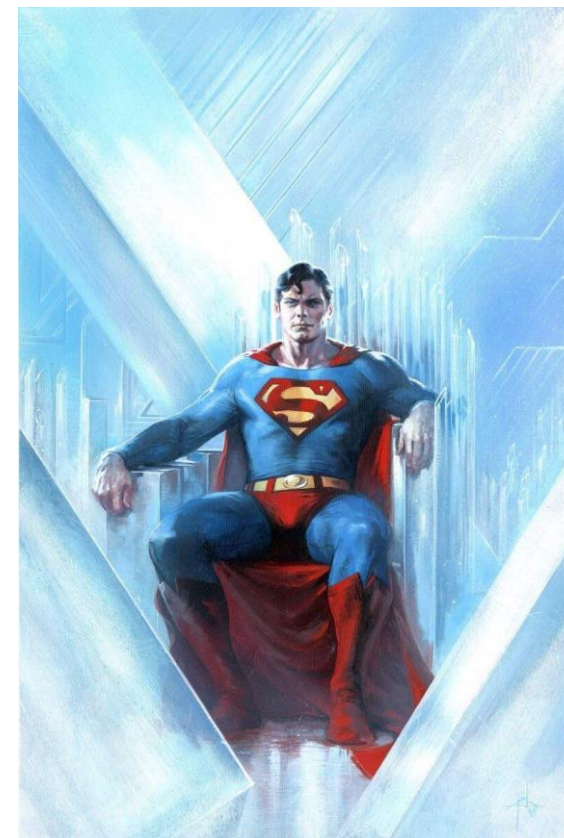
```
HTTPEvents = HTTPParametersSet /  
             HTTPFrameCreated /  
             HTTPPushResolved
```

```
$ProtocolEventBody  $\neq$  HTTPEvents
```



Tip of the iceberg

- Several other useful features
 - `uint32 = uint .size 4`
 - `hexstring = text .regexp "([0-9a-f]{2})*"`
 - TODO: Specify metadata ("importance" of an event: core/base/extra)
 - Using `.feature` control operator of RFC9165
- Big **consistency** updates to the documents overall
 - Fixed all ID-nits problems (mainly too long lines)
 - Proper identifiers/naming for all code blocks and examples
 - Proper names for all events and data structures



But why?

CDDL allows/**will allow** us to automatically:



- Extract definitions from draft documents
- Validate full schema, across documents
 - Using existing CDDL tooling
- Generate dummy qlog JSON files (easy examples!)
- **Generate other schema representations (e.g., JSON-schema, proper TypeScript, Protobuf, ...)**
- **Validate qlog files themselves (e.g., catch typo's, missing required fields, ...)**
 - **With integration in tools such as qvis**

Good for:

- Creating new/updating old QUIC + H3 qlog implementations
- Extend qlog to new features and protocols
 - **WebTransport, MASQUE, MultiPath, ...**
 - Maybe TCP, HTTP/2, TLS, ... down the line

<https://rubygems.org/gems/cddl>

<https://github.com/quiclog/qlog/tree/master/CDDL/schema>

<https://gist.github.com/lnicco/3418e0d7cab2363e3600ff7c6b9988a1>

Direct Next Steps

More editorial stuff

- Align terminology with QUIC, H3 and QPACK docs (#174)
 - Lots of leftovers from older drafts
 - **Especially for QPACK** (HTTP Headers → fields, etc.)
- Writing better prose
- Update descriptions of some events
(e.g., connection_state_updated)

*Editors can propose most of this ourselves, but **we need QPACK support!***

Missing events/fields

- Again mainly **QPACK (#199)** but also **TLS** (several issues)
- Events that show where we were blocked (#132)
- CPU/threading info (#152)
- Other smaller things (e.g., #186, #168, #135, #119, #57, etc.)

<https://github.com/quicwg/qlog/issues/174>

<https://github.com/quicwg/qlog/issues/199>

<https://github.com/quicwg/qlog/issues/132>

<https://github.com/quicwg/qlog/issues/152>

<https://github.com/quicwg/qlog/projects/1#column-15587393>

Road to RFC / Big Open Questions

Design

- Split up main schema? (#197)
 - Move QLOGDIR and .well-known/qlog/ to separate document?
- Versioning and extensibility
 - Versioning of qlog files (#156)
 - Support new Transport Parameters (#176)
 - Support new Protocols (#146)
 - Define where new documents should/could live (#170)
- Security and Privacy considerations (#142)
 - Should we define anything?
 - If yes: what? **High-level guidance vs per-field indicators?**
 - Doesn't seem to be much prior work on this,
feels too big to tackle alone / in qlog



We need **WG**
guidance on these!

<https://github.com/quicwg/qlog/issues/197>
<https://github.com/quicwg/qlog/issues/156>
<https://github.com/quicwg/qlog/issues/176>
<https://github.com/quicwg/qlog/issues/146>
<https://github.com/quicwg/qlog/issues/170>
<https://github.com/quicwg/qlog/issues/142>