structured event logging

the Superman update

Robin Marx  robin.marx@kuleuven.be
Only editorial changes

Easy to go from TypeScript to CDDL

```typescript
class QlogData {
    name: string
    packetCount?: number
    location: VantagePoint
    events: Array<Event>
    raw: bytes

    // any other fields are possible here
}
enum VantagePoint {
    server,
    client
}
```

QlogData = {
    name: text
? packetCount: uint32
location: VantagePoint
events: [* Event]
raw: hexstring

* text ⇒ any
}

VantagePoint = "client" / "server"

Concise Data Definition Language (CDDL, RFC 8610)

https://github.com/quiclog/qlog/tree/master/CDDL/schema
CDDL has other nifty features

**regexp operator**

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

```c
; from "crypto_error_0×100" to "crypto_error_0×199"
CryptoError = text .regexp "crypto_error_0×1[0-9][0-9]"
```

**unwrap operator ~**

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

```c
Event2 = {
    ~SharedFields
    field2: uint32
}
```

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

https://github.com/quiclog/qlog/tree/master/CDDL/schema
CDDL has other nifty features: extension sockets

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

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

```javascript
HTTPEvents = HTTPParametersSet / HTTPFrameCreated / HTTPPushResolved

$ProtocolEventBody ⊔ HTTPEvents
```

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

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.)

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

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