An Abstract Application Layer Interface to Transport Services

draft-trammell-taps-interface-01

Brian Trammell

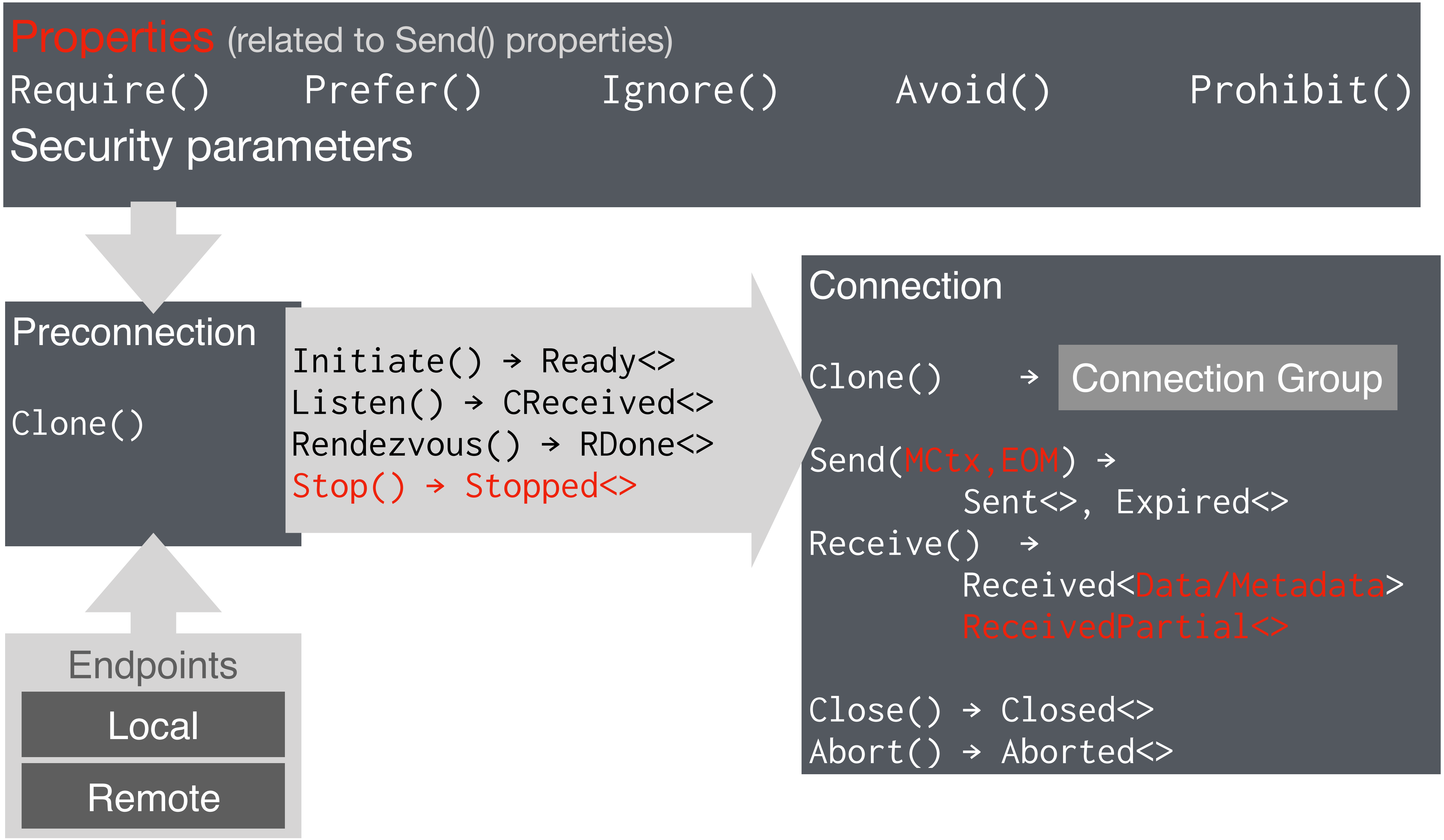
TAPS — IETF 101 — Montreal — Tue 17 July 2018

Interface Design Principles (§3) (a review)

We set out to define a ***single interface*** to a variety of transport protocols to be used in a variety of application design patterns, to enable applications written to a single API to make use of multiple transport protocols in terms of the features they provide, providing:

- explicit support for ***security properties*** as first-order transport features;
- ***asynchronous*** connection, transmission, and reception;
- support for ***multistreaming and multipath*** transport protocols; and
- ***atomic transmission of data***, using application-assisted framing and deframing where necessary.

Interface Diagram (as of -01)



Properties (related to Send() properties)

Require() Prefer() Ignore() Avoid() Prohibit()

Security parameters

Preconnection

Clone()

```
Initiate() → Ready<>
Listen() → CReceived<>
Rendezvous() → RDone<>
Stop() → Stopped<>
```

Endpoints

Local

Remote

Connection

```
Clone()     → Connection Group

Send(MCtx,EOM) →
        Sent<>, Expired<>
Receive()  →
        Received<Data/Metadata>
        ReceivedPartial<>

Close() → Closed<>
Abort() → Aborted<>
```

(non-editorial) changes since -00

- #201 Restructure Transport Properties
- #200 Rework Partial Sends and Receives
- #198 Message Receive Metadata
- #195 Ordering of API Events
- #181 Rework Interface Types
- #171 Batching Sends

(non-editorial) changes since -00

- #201 Restructure Transport Properties
- #200 Rework Partial Sends and Receives
- #198 Message Receive Metadata
- #195 Ordering of API Events
- #181 Rework Interface Types
- #171 Batching Sends

#201 Transport Parameters Rework

- All of the various ways to configure stacks (pre-selection, post-selection, and per-send) are related, but were spread throughout the document
- New approach: group all (non-security) parameters into into Properties (new §12), attempt to reclassify them.
  - Note: the authors ***do not think we have this right yet***, but we do think it's less intentionally confusing than it was.
  - Definitely needs reordering (order is kind of random)
  - May need new / renamed axes / classifications.
- Preferences still expressed using `Require()`, `Prefer()`, `Avoid()`, `Prohibit()`; send properties are bound to `MessageContext` passed on `Send()`.

#201: current property "axes"

- Data type

  Boolean / Enumeration / Integer / Preference

- Scope

  Preconnection / Connection / Message

- Classification

Affected Aspects		Path & Protocol Selection	Protocol Operation	Control Flow
Level of Abstraction	Immediate	Selection Property	Protocol Property	Control Property
	↕	Intent		
	Interpreted			

Properties (1/3)

			protocol/ control prop.	
	Type	Select	post-Select	per-Send
12.3.1. Final	bool			✓/?
12.3.2. Reliable Message Transfer	pref	✓		
12.3.3. Configure Reliability Per Message	pref	✓		
12.3.4. Reliable Transfer (Message)	bool			✓
12.3.5. Preservation of Data Ordering	pref	✓		
12.3.6. Ordered	bool			✓
12.3.7. Direction of communication	enum	?	?	
12.3.8. 0-RTT Establishment w/Idem. Send	pref	✓		
12.3.9. Idempotent	bool			✓
12.3.10. Multistream in Group	pref	✓		
12.3.11. Excessive RTX Notification	pref	✓		
12.3.12. Exc. RTX Notification Threshold	int	✓	✓	

Properties (2/3)

			protocol/ control prop.	
	Type	Select	post-Select	per-Send
12.3.13. Soft Error Notification	pref	✓		
12.3.14. Checksum Coverage Control	pref	✓		
12.3.15. Checksum Coverage Length	int			✓
12.3.16. Recv Checksum Requirement	int	✓	✓	
12.3.17. Interface Instance / Type	(enum,pref)	✓		
12.3.18. PvD Instance / Type	(enum,pref)	✓		
12.3.19. Capacity Profile (intent)	enum	✓	✓	✓
12.3.20. Congestion Control	pref	✓		
12.3.21. Niceness	int		✓	✓
12.3.22. Abort Timeout	int	✓		
12.3.23. Connection Group TX Scheduler	enum	✓	✓	

Properties (3/3)

			protocol/ control prop.	
	Type	Select	post-Select	per-Send
12.3.24. Max Idempotent Send Size	int		r/o	
12.3.25. Max No-Frag Send Size	int		r/o	
12.3.26. Max (non-partial?) Send Size	int		r/o	
12.3.27. Max (non-partial?) Recv Size	int		r/o	
12.3.28. PR Send Lifetime	int			✓

Some Observations from the Editor (+discussion)

- Calling these axes is a little misleading: they're not orthogonal
- We have only six distinct kinds of thing:
  - Preference used for selection, scoped to preconnection, read-only after connection.
  - Property used to control how messages are sent, scoped to message (usually boolean, usually linked to selection preference).
  - Property used to control protocol operation, scoped to preconnection + connection (usually integer, controls timeout/buffer size/scheduler), possibly also usable for selection.
  - Property used to inspect protocol operation, scoped to connection, read-only (usually integer, buffer size).
  - Two enumeration/preference tuples for selecting interface/PvD.
  - One intent for path selection and scheduler configuration.

#200 Partial Send and Receive

- API is organized around atomic write/read of messages
  - (using application-supplied deframing when the underlying transport doesn't do framing, see §8.4)
- But sometimes you have a message (or a real stream) that won't fit into a buffer.
- Solution: partial read/write
  - Introduce optional `EOM` parameter to `Send();` calls with `EOM = false →` still writing to a partial message identified by a given `MessageContext.`
  - `ReceivedPartial<>` event fires when a partial message is received.
- Partial read/write boundaries are not preserved.

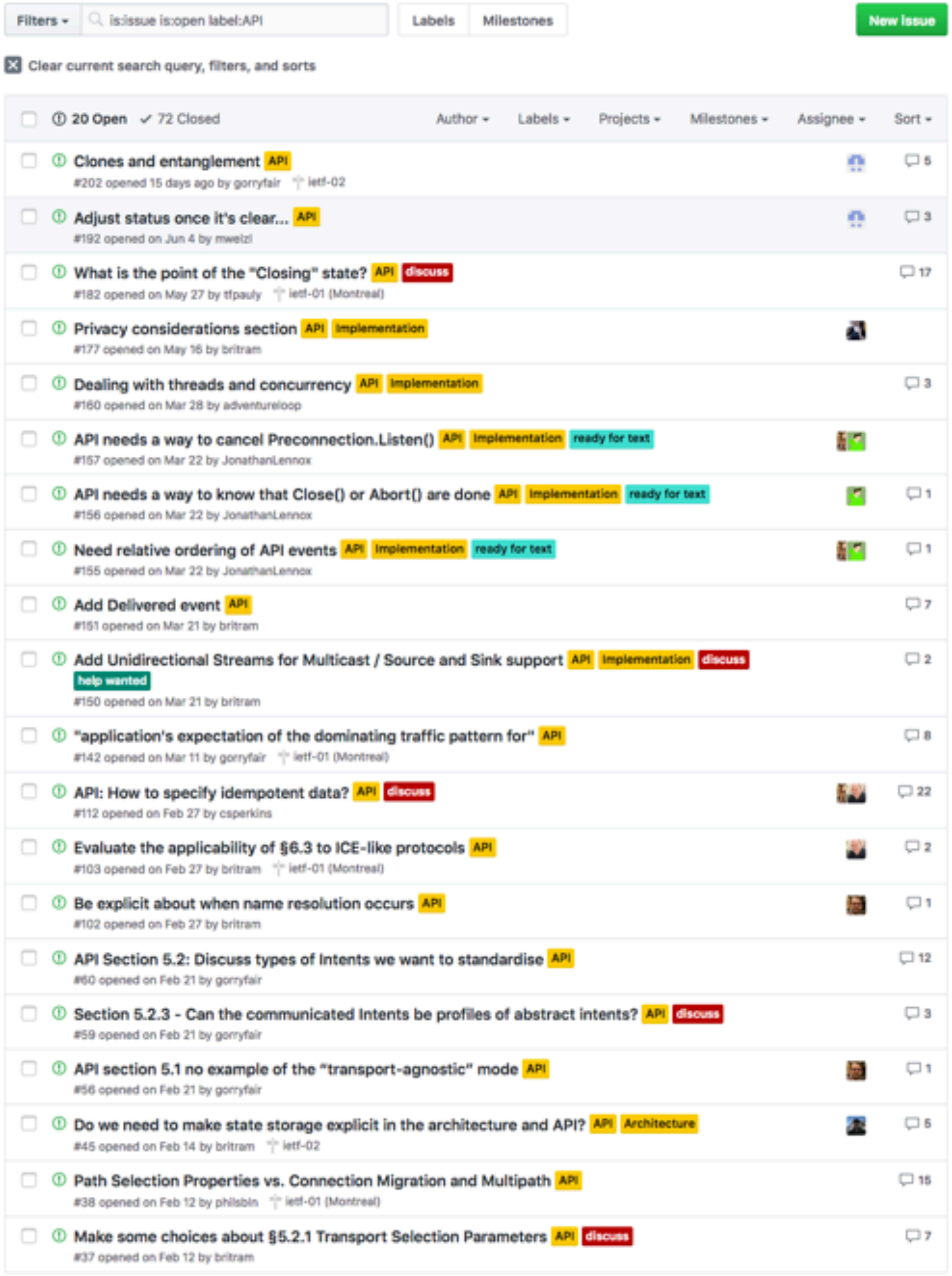
Open issue: API for idempotent Send on establishment (#112 / #124)

- How does the application tell the stack that it wants to send some 0RTT data?
  - Some tradeoffs here, but mainly a bikeshed.

- Option 1: as in #124, hold any data sent until an explicit `Connection.Start()` call.
  - `Send()` before `Start()` is 0RTT if idempotent.
  - `Start()` is always required, even if you don't know what 0RTT is.
- Option 3: 0RTT behavior is implied by 0RTT selection properties.
  - When `Initiate()` is called and selects a 0RTT-capable stack, the actual initiation is delayed slightly to wait for the first `Send()`, which is 0RTT if idempotent.
  - Note this makes racing 0RTT-capable and 0RTT-incapable stacks impossible.
- Option 3.5: as 3, but with a `Preconnection.InitiateNow()` to override the wait-for-`Send()` behavior (e.g. for application protocols where the server sends first)
- Option 5: Add `Preconnection.Send()`, which initiates with 0RTT data.

Next steps

There are still some open issues:
github.com/taps-api/drafts/issues



is:issue is:open label:API

☒ Clear current search query, filters, and sorts

□ ⓘ 20 Open ✓ 72 Closed

- □ ⓘ Clones and entanglement API — 5
  #202 opened 15 days ago by gorryfair · ietf-02
- □ ⓘ Adjust status once it's clear... API — 3
  #192 opened on Jun 4 by mwelzl
- □ ⓘ What is the point of the "Closing" state? API discuss — 17
  #182 opened on May 27 by tfpauly · ietf-01 (Montreal)
- □ ⓘ Privacy considerations section API Implementation
  #177 opened on May 16 by britram
- □ ⓘ Dealing with threads and concurrency API Implementation — 3
  #160 opened on Mar 28 by adventureloop
- □ ⓘ API needs a way to cancel Preconnection.Listen() API Implementation ready for text
  #157 opened on Mar 22 by JonathanLennox
- □ ⓘ API needs a way to know that Close() or Abort() are done API Implementation ready for text — 1
  #156 opened on Mar 22 by JonathanLennox
- □ ⓘ Need relative ordering of API events API Implementation ready for text — 1
  #155 opened on Mar 22 by JonathanLennox
- □ ⓘ Add Delivered event API — 7
  #151 opened on Mar 21 by britram
- □ ⓘ Add Unidirectional Streams for Multicast / Source and Sink support API Implementation discuss help wanted — 2
  #150 opened on Mar 21 by britram
- □ ⓘ "application's expectation of the dominating traffic pattern for" API — 8
  #142 opened on Mar 11 by gorryfair · ietf-01 (Montreal)
- □ ⓘ API: How to specify idempotent data? API discuss — 22
  #112 opened on Feb 27 by csperkins
- □ ⓘ Evaluate the applicability of §6.3 to ICE-like protocols API — 2
  #103 opened on Feb 27 by britram · ietf-01 (Montreal)
- □ ⓘ Be explicit about when name resolution occurs API — 1
  #102 opened on Feb 27 by britram
- □ ⓘ API Section 5.2: Discuss types of Intents we want to standardise API — 12
  #60 opened on Feb 21 by gorryfair
- □ ⓘ Section 5.2.3 - Can the communicated Intents be profiles of abstract intents? API discuss — 3
  #59 opened on Feb 21 by gorryfair
- □ ⓘ API section 5.1 no example of the "transport-agnostic" mode API — 1
  #56 opened on Feb 21 by gorryfair
- □ ⓘ Do we need to make state storage explicit in the architecture and API? API Architecture — 5
  #45 opened on Feb 14 by britram · ietf-02
- □ ⓘ Path Selection Properties vs. Connection Migration and Multipath API — 15
  #38 opened on Feb 12 by philsbln · ietf-01 (Montreal)
- □ ⓘ Make some choices about §5.2.1 Transport Selection Parameters API discuss — 7
  #37 opened on Feb 12 by britram