Separate vs same API for Logs and Events

The following table captures arguments in favour of 2 different approaches that Log SIG is discussing for Log and Events APIs.

One API (LoggerProvider, Logger, logBuilder/eventBuidler, etc)	**Separate APIs (LoggerProvider, Logger, EventEmitterProvider, EvenEmitter, etc)**
More consistent with the rest of OpenTelemetry, where each signals has one entry point (Tracer/Meter/Logger)	Less consistent with the rest of OpenTelemetry, Logger and EventEmitter are both entry points into the same signal (from data model perspective)
More consistent with data model ideology. Data emitted through Logger is Log Record.	Less consistent with data model ideology. Data emitted through EventEmitter and Logger is Log Record. May confuse people who don't know that EventEmitter emits Log Records.
Less specific API for events, does not help prevent mistakes. event.domain may easily be forgotten when creating the Logger.	More specialized API for events which enforces and prevents mistakes, e.g. by requiring event.domain value when the EventEmitter is created.
	More intuitive to users of Events API, who may not be very familiar with OpenTelemetry. On the other hand, less intuitive to Log Appender authors, but this is mitigated by the smaller size and closer relation of this group to the project.
Log API is exposed to instrumentations and end user code, which is conflicting with the desire to not compete with existing logging libraries.	Each API is used only in the context for which it is intended - log API for appenders, event API for instrumentations
Events will always be tied to the Log data model	Separate Event API is an abstraction for events and can use different data model behind the scenes, e.g. span events
	"Event" concept in isolation seems to be confused with the Span Event.