Event Stream Extensions for AMQP Version 1.0

Committee Specification Draft 01

17 March 2021

This stage:

https://docs.oasis-open.org/amqp/event-streams/v1.0/csd01/event-streams-v1.0-csd01.md (Authoritative)
https://docs.oasis-open.org/amqp/event-streams/v1.0/csd01/event-streams-v1.0-csd01.html
https://docs.oasis-open.org/amqp/event-streams/v1.0/csd01/event-streams-v1.0-csd01.pdf

Previous stage:

N/A

Latest stage of Version 1.0:

https://docs.oasis-open.org/amqp/event-streams/v1.0/event-streams-v1.0.md (Authoritative)
https://docs.oasis-open.org/amqp/event-streams/v1.0/event-streams-v1.0.html
https://docs.oasis-open.org/amqp/event-streams/v1.0/event-streams-v1.0.pdf

Technical Committee:

OASIS Advanced Message Queuing Protocol (AMQP) TC

Chairs:

Rob Godfrey (rgodfrey@redhat.com), Red Hat
Clemens Vasters (clemensv@microsoft.com), Microsoft

Editor:

Clemens Vasters (clemensv@microsoft.com), Microsoft

Related work:

This specification is related to:

- *OASIS Advanced Message Queuing Protocol (AMQP) Version 1.0 Part 0: Overview.* Edited by Robert Godfrey, David Ingham, and Rafael Schloming. Latest version: http://docs.oasis-open.org/amqp/core/v1.0/amqp-core-overview-v1.0.html.

Abstract:

This specification defines a set of AMQP extensions for interaction with event stream engines, including annotations for partition selection and filter definitions for indicating offsets into an extension stream to which a link shall be attached.

Status:

This document was last revised or approved by the OASIS Advanced Message Queuing Protocol (AMQP) TC on the above date. The level of approval is also listed above. Check the "Latest stage" location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=amqp#technical.

TC members should send comments on this specification to the TC's email list. Others should send comments to the TC's public comment list, after subscribing to it by following the instructions at the "Send A Comment" button on the TC's web page at https://www.oasis-open.org/committees/amqp/.

Note that any machine-readable content (Computer Language Definitions) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

Citation format:

When referencing this specification the following citation format should be used:

[Event-Streams-v1.0]

Event Stream Extensions for AMQP Version 1.0. Edited by Clemens Vasters. 17 March 2021. Committee Specification Draft 01. https://docs.oasis-open.org/amqp/event-streams/v1.0/csd01/event-streams-v1.0-csd01.html. Latest stage: https://docs.oasis-open.org/amqp/event-streams/v1.0/event-streams-v1.0.html.

Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

As stated in the OASIS IPR Policy, the following three paragraphs in brackets apply to OASIS Standards Final Deliverable documents (Committee Specification, Candidate OASIS Standard, OASIS Standard, or Approved Errata).

[OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Standards Final Deliverable, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this deliverable.]

[OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this OASIS Standards Final Deliverable by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this OASIS Standards Final Deliverable. OASIS may include such claims on its website, but disclaims any obligation to do so.]

[OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this OASIS Standards Final Deliverable or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Standards Final Deliverable, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.]

The name "OASIS" is a trademark of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see https://www.oasis-open.org/policies-guidelines/trademark for above guidance.

Table of Contents

1 Introduction

This specification defines a set of AMQP extensions for interaction with event stream engines, including annotations for partition selection and filter definitions for indicating offsets into an extension stream to which a link shall be attached.

In the context of this specification, the term "event stream engine" describes a particular archetype of AMQP node that manages sequences of events (messages) in an ordered log structure.

Event logs differ from queue-like structures in that they do not track message delivery state across competing receivers. The message delivery state is instead tracked for each link at its terminus. When attaching a `receive` AMQP link, the receiver MAY indicate an initial offset into the retained event log. The delivery of events then progresses in log order from that initial offset to more recently added events. Omitting the offset results in delivery of events added after attaching the link.

Event stream engines conforming to this specification MAY implement any of the capabilities defined here and they MUST implement the default behaviors defined here when those capabilities are not explicitly used.

It is not in scope for this specification to define how an event stream consumer might handle persistence and failover of initial stream offsets that the consumer wants to track across multiple interactions with the event stream log. Once an AMQP link is established with an initial offset, the event log node's AMQP link terminus will keep track of the consumer offsets and delivery state while the link is active. Each message delivered to a consumer carries a delivery annotation (`event-streams-offset`) that the consumer can hold on to and use as the initial offset for a new link when it wants to resume suspended processing.

1.1 IPR Policy

This specification is provided under the RF on RAND Mode of the OASIS IPR Policy, the mode chosen when the Technical Committee was established. For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC's web page (https://www.oasis-open.org/committees/amqp/ipr.php).

1.2 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] and [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3 Normative References

[AMQP-v1.0]

OASIS Advanced Message Queuing Protocol (AMQP) Version 1.0 Part 0: Overview. Edited by Robert Godfrey, David Ingham, and Rafael Schloming. Latest version: http://docs.oasis-open.org/amqp/core/v1.0/amqp-core-overview-v1.0.html.

[FILTEX-v1.0]

AMQP Filter Expressions Version 1.0. Edited by Clemens Vasters. Latest version: http://docs.oasis-open.org/amqp/filtex/v1.0/filtex-v1.0.docx

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, http://www.rfc-editor.org/info/rfc2119.

[RFC8174]

Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, http://www.rfc-editor.org/info/rfc8174.

1.4 Non-Normative References

[RFC3552]

Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, https://www.rfc-editor.org/info/rfc3552.

2 Definitions

The following terminology is used in this specification to clarify roles. These definitions are narrower than their use in the core AMQP specification.

- `event stream engine` is a software infrastructure managing one or more event logs.
- `event log` maintains a sequence of events.
- `partition` is a subdivision of an `event log`.
- `event log node` refers to the AMQP node where links of senders and receivers attach.
- `producer` is the sender of events into the event log.
- `consumer` is the receiver of events from the event log.
- `consumer group` is a named group of consumers that coordinate consumption of events across multiple partitions.

3 Capabilities and Default Behaviors

3.1 Connection Capability

On connection establishment, the event log node MUST indicate its support of this specification through the exchange of a connection capability (see Section 2.7.1, AMQP). The capability allows for the consumer or producer to detect whether extensions as defined in this specification are understood and may be used.

Capability Name	Definition
AMQP_EVENT_STREAMS_V1_0	If present in the offered-capabilities field of the open frame, the sender of the open supports the use of event streams. If present in the desired-capabilities field of the open frame, the sender of the open MUST use the event streams extensions if the receiver if the open offers this capability, and it MAY shut down the connection if the capability is not offered.

3.2 Default Behaviors

This specification does not modify or override any rules of the core AMQP 1.0 specification.

Even with the capability being negotiated at the connection level, all additional behaviors defined in this specification are OPTIONAL, which means that a producer or consumer MUST be able to successfully interact with a conforming event log implementation just as with any regular AMQP node, for instance:

- Producers and consumers that do not choose a partition on a partitioned event log are either assigned a partition by the event log node, or their links are left to be partition-agnostic.
- Producers are not required to annotate events with partition hints, even on partition-agnostic links. The event log node chooses the partition in absence of such hints.
- Consumers are not required to provide offset filter information. If no filter is provided, the consumer is eligible to receive events accepted by the event log node after the link was attached, equivalent to the 'latest' offset filter.
- Consumers can rely on all existing AMQP mechanisms for delivery-state tracking provided by the AMQP source terminus of the event log node. This means that after attaching a link, the consumer is not required to keep track of offsets for initiating further transfers while the link exists. If the source offers link durability (AMQP 1.0, 3.5.5), the last offset is preserved by the durable terminus on the event log node side in the case of disconnects. The AMQP durable terminus model can indeed completely obviate the need for an external store to keep track of consumer offsets.

4 Partition support

An event log maintains a sequence of events. An event log MAY be split into multiple parallel partitions that are reachable under the same network address, or the responsibility for maintaining an event log might also be shared across multiple AMQP containers/nodes that each have a separate network address and are each responsible for one or more partitions of the event log.

Producers and consumers attach to the event log via AMQP links. Those links MAY be bound to one specific partition (partition-bound), or they MAY be partition-agnostic. On partition-agnostic links, producers MAY provide transfer-level hints for which partition the transfer ought to be routed to.

A partition-agnostic link will generally still be associated with one or multiple partitions on the side of the event log, but that association is dynamic.

This model allows for a range of negotiation options for how consumers are affiliated with partitions:

- Producer and consumers choose partitions on their own
- Producer and consumers are assigned partitions by the event log node
- Producer and consumers are partition-agnostic and can send/receive from any partition

The model composes with AMQP link-level redirects as defined in AMQP 1.0, 2.8.18. With link-level redirects, the event log node MAY refuse to attach a link, but rather point to a different AMQP container where the requested or assigned partition is available.

Partition-bound links provide the consumer with the assurance that all transfers belong to the given partition, which is desirable when the partitioning model extends into resources beyond the event log.

The binding of a producer or consumer to a partition is negotiated using the ´event-streams-partition´ link property as the link is being attached.

A consumer or producer MAY request for the link to be bound to a particular partition. If the event log node is not willing grant the request for binding to a specific partition, it MUST reject (attach and immediately detach) the link.

The responding event log node MAY bind any link to a partition if no partition binding has been requested by the producer or consumer. Whether the event log node assigns partition-bound or partition-agnostic links MAY be a configurable implementation choice.

If the event log chose to bind links to partitions for consumers who did not request a partition binding, and the event log wants to renegotiate those assignments, it MAY gracefully close some or all attached links. The respective consumer(s) SHOULD then attempt to establish a new link.

Partition identifiers are opaque to consumers and producers and of type `symbol`. Consumers and producers learn about an event log node's available partitions and their identifiers using a runtime information request.

4.1 Binding producer links

A producer MAY request for the 'send' link to be bound to a specific partition using the `event-streams-partition` link property.

If the requested partition does not exist or if the event log node denies the request, attaching the link MUST be rejected (attached and immediately detached) by the event log node.

The event log node MAY bind the link a partition when none has been requested by the producer.

If the link becomes partition-bound, the event log node MUST set the `event-streams-partition` link property to the bound partition.

4.2 Partition annotations in producer transfers

If a `send` link is partition-agnostic, the sender MAY provide partition-related hints to the `event log node` with the `event-streams-target-partition` delivery annotation or the `event-streams-group-key` message annotation.

If the `event-streams-target-partition` delivery annotation is added to a transfer on a partition-bound link, its value MUST match the `event-streams-partition` link property's value. Otherwise, the transfer MUST be rejected.

4.3 Binding Consumer Links

A consumer MAY request for the 'receive' link to be bound to a specific partition using the `event-streams-partition` link property.

If the requested partition does not exist or if the event log node denies the request, attaching the link MUST be rejected by the event log node.

The event log node MAY bind the link a partition when none has been requested by the consumer.

If the link becomes partition-bound, the event log node MUST set the `event-streams-partition` link property to the bound partition.

4.4 Partition annotations in consumer transfers

The `event-streams-source-partition` delivery annotation MUST be added to all transfers to consumers made over partition-agnostic links.

The annotation is OPTIONAL for transfers over partition-bound links, and if present, its value MUST be equal to the `event-streams-partition` link property value. Otherwise, the transfer MUST be rejected.

4.5 Consumer groups

In scenarios where a group of multiple concurrent event consumers want to receive mutually exclusive subsets of events from a partitioned event log, it is desireable to restrict each partition to one (the "owning") receiver from the group, and for event consumers to shift ownership of the partition dynamically.

Consumers might also want to use an external consensus mechanism to agree on which event processor owns which partition, rather than have the event log engine coordinate the assignments.

This specification does not prescribe any algorithm or strategy for reaching consensus on ownership of partitions amongst a group of consumers, but it provides mechanisms for realizing their outcomes.

A consumer link MAY be bound to a consumer group by setting the `event-streams-consumer-group` link property to the name of the group during attach.

The consumer group MAY be dynamically created during link attach or the event log node MAY require for there to be a such named pre-configured entity, in which case attaching the link MAY fail if such an entity does not exist.

The event log node MUST NOT assign a consumer group on its own. Consumer group membership claims MUST be made by the consumer. Links that are bound to consumer groups MUST also be bound to a partition; they cannot be partition-agnostic.

The selection and binding of a partition to the consumer within a given consumer group follows the negotiation model explained earlier in this section.

If consumer groups are being used, the event log node MUST allow at most one active 'receive' link for each combination of partition and consumer group.

4.5.1 Event log node-assigned partition ownership

If the consumer attaches a link scoped to a consumer group without specifying a partition, the event log node MAY bind a partition to the link as described in 4.3.

If the event log node manages, for instance, 10 partitions and there are 12 candidate consumers trying to establish links, one of two scenarios MAY apply:

1. Two of those consumers MAY see their attach requests immediately rejected with a `detach-forced` error since there are no partitions left to assign.
2. The event log node MAY instead choose not to deny attaching the link outright, but keep further incoming attach requests pending and complete those only once a partition becomes again available for assignment. Attach requests might therefore remain pending for a substantial time span, whose limits MAY be configurable in an implementation specific way.

If the event log node wants to force renegotiation of partition bindings amongst a consumer group, it SHOULD gracefully close all partition-bound links and the respective consumers SHOULD attempt to establish a new link, again with the consumer group, but without specifying a partition.

The event log node MAY also attach consumer group links while leaving them partition-agnostics, allowing for multiple partitions to be associated with a link and for those bindings to change dynamically without having to reestablish the link. In this case, the event log SHOULD use the `event-streams-associated-partitions-changed` delivery annotation on the next transfer to notify the consumer of the new association.

4.5.2 Consumer-negotiated partition ownership

To allow consumers to negotiate partition ownership using an external consensus algorithm that does not establish transparent communication amongst the group of consumers, new assignments of consumers to partitions within the group need to be able to break the links of existing assignments.

For instance, if the consensus algorithm for ownership of a partition were based on a expiring lease on some external object and a competing consumer were to acquire that lease upon expiration, the new owner ought to be able to promptly take ownership of the partition, without asking for the previous owner's explicit cooperation.

To facilitate this, for each combination of partition and consumer group, the event log node maintains a `ulong`-valued `epoch` counter, which is seeded with the value zero and is returned in the `event-streams-epoch` link property to the receiver during a successful attempt to attach.

Any subsequent attempt to attach a link to the same combination of partition and consumer group of an already active link MUST be rejected UNLESS the newly attaching receiver provides an epoch counter greater than the current epoch counter value, expressed using the `event-streams-epoch` link property in attach.

If the epoch counter is provided and greater than the current epoch value, any existing link for the given partition and consumer group is closed by the node, the new link is attached, and the new epoch counter value is recorded.

Whenever the link for a partition within a group is closed by the receiver and no links remain, the respective epoch counter MAY be reset to zero.

4.6. Property definitions

4.6.1 event-streams-partition link property

The `event-streams-partition` link property is a `symbol` value and describes the partition to which the link is bound.

When set by the producer or consumer, the property value is a request for the link to be bound to the given partition. When set by the event log node, the value reflects the effective association.

When not set on the negotiated link, the link is partition-agnostic.

4.6.2 event-streams-consumer-group link property

The `event-streams-consumer-group` link property is a `string` value that identifies a group of consumers and provides a scope for the epoch counter.

An implementation MAY require for the group identifier to correspond to a configured item on the event stream engine and otherwise refuse to attach the link.

A negotiated link carrying this property MUST be partition-bound.

4.6.3 event-streams-epoch link property

The `event-streams-epoch` link property is a `long` integer value and describes the ownership epoch counter for the link relative to the consumer group and while at least one link is attached to the partition within the given consumer group.

If no link is attached to node for the given partition and group, the epoch counter MAY be reset to zero.

4.6.4 event-streams-source-partition delivery annotation

The `event-streams-source-partition` delivery annotation is a `symbol` value and describes the partition from which partition the transfer originates.

If the event log is partitioned, the delivery annotation MUST be added by the event log node before delivery to consumers.

A consumer MUST strip the annotation if it forwards the message onwards.

4.6.5 event-streams-associated-partitions-changed delivery annotation

The `event-streams-assigned-partitions-changed` delivery annotation is an array of `symbol` values and contains the list of partitions associated with the link that the transfer is being carried over.

For all partition-agnostic consumer links, the event log node SHOULD add this annotation to the next transfer to the consumer every time the association of partitions with the link changes.

A consumer MUST strip the annotation if it forwards the message onwards.

4.6.6 event-streams-target-partition delivery annotation

The `event-streams-target-partition` delivery annotation is a `symbol` value and describes the partition to which the transfer MUST be routed.

If the link is not partition-agnostic and if the link is not bound to the exact same partition, the transfer MUST be rejected. If the partition does not exist, the transfer MUST be rejected.

The event log MUST strip the annotation once it has evaluated it.

4.6.7 event-streams-group-key message annotation

The `event-streams-group-key` message annotation is a `string` value and is a grouping key that identifies events that are related and SHOULD therefore be treated as a group for partitioning purposes.

This value differs from the AMQP core message property `group-id` in being an annotation and therefore allowing overrides as messages get routed through multiple partitioned intermediaries where partitioning concerns and therefore the choice of grouping keys might vary.

When set by the producer and used over a partition-agnostic link, the value of the annotation MAY be used by the event log node to bind the transfer to a partition. If the key is used in this way, the event log SHOULD bind transfers with the key to the same partition at all times and until the number of partitions changes.

5 Stream delivery offsets and filters

As stated in the introduction, event logs differ from queue-like structures in that they only track message delivery state at the AMQP link level. While attaching a `receive` AMQP link, the consumer MAY indicate a desired initial offset into the retained log or some other condition based on which events become eligible for delivery. The desired offset is expressed using an AMQP filter expression.

5.1 Delivery annotations

The delivery filter expressions refer to a set of delivery annotations defined in the following.

For a conforming implementation, these delivery annotations MUST be added by the event log node to all messages delivered to consumers.

5.1.1 event-streams-offset

The `event-streams-offset` delivery annotation property is a `symbol` expression whose internal syntax is opaque to the receiver. Receivers MUST NOT interpret or construct offset values. The offset value represents a relative position of the event to the beginning of the log.

Even though the syntax is opaque, the chosen syntax MUST ensure that the lexicographical order of the `event-streams-offset` of any event in the log MUST be greater than the lexicographical order of any other event preceding it in the order of the log.

Receivers learn about offset values either by obtaining the runtime information of a partition, which contains the offset of the most recently added event, or by retaining the value of the last `event-streams-offset` delivery annotation property they read from the partition.

For use with delivery filter expressions, the reserved values MAY be used as constants for comparand values:

- "@earliest" is smaller than all existing offsets in the log.
- "@latest" is greater than all existing offsets in the log.

5.1.2 event-streams-timestamp

The `event-streams-timestamp` delivery annotation property is a timestamp expression and reflects the absolute UTC instant when the event has been added to the event log. This is different from the `creation-time` message property that is set by the sender.

Multiple events added to the log around the same instant MAY have identical timestamps due to the UTC clock resolution concerns. The timestamp only serves as a filter criterion to determine a desired offset into the stream, but MUST NOT be used to determine the order of events.

A distributed event log implementation where the ownership of a log changes between different hosts and where clocks might not be fully synchronized MAY record `event-streams-timestamp` values that reflect the existing clock skew, meaning that some events that appear later in the log than a given event might yet carry earlier UTC timestamps.

5.2 Delivery filters

A delivery filter specifies which events are eligible for delivery from the event log. The delivery filter is provided as a source filter during link attach.

Delivery filters MAY be expressed using any AMQP filter expression that is supported by the source (the event log node). This specification defines two simplified filters that allow for a conforming implementation without having to implement the full AMQP filter expressions specification in a conforming fashion.

The source filter MUST be evaluated against the retained events in the log from earliest to latest. Events matching the source expression are eligible for delivery.

5.2.1 event-streams-delivery-annotations filter

The `event-streams-delivery-annotations` filter type applies to AMQP delivery annotations section (AMQP 3.2.2). The filter evaluates to true if all annotations enclosed in the filter expression match the respective delivery annotation map entries.

Only the `event-streams-offset` and/or `event-streams-timestamp` delivery annotation properties are eligible to be used with this filter.

The property values match the given values in the filter expression, if the value of the property is greater than value given in the filter expression. If the application remembered a particular offset after having consumed the message that carried it, the filter matches messages added after that offset.

```
<type name="event-streams-delivery-annotations-filter" class="restricted"
source="amqp:map" provides="filter">
<descriptor name="amqp:event-streams-delivery-annotations-filter"
code="0x00000000:0x00000200"/>
</type>
```

The filter-type is “amqp:event-streams-delivery-annotations-filter”, with type-code 0x00000000:0x00000200

5.2.2 event-streams-sql filter

The event streams SQL filter expression is a constrained subset of the SQL filter expression defined in AMQP filter expressions [FILTEX, 6].

The grammar builds on the grammar defined for those SQL filters, but is constrained as follows:

```
predicate ::=  <boolean_constant> |
               | <expression> <comparison-operator> <expression>
               | <predicate> <binary-logical-operator> <predicate>
expression ::= <constant> | <field_value>
constant   ::= <string_constant> | <integer_constant>
field_value := <field>
field ::= <section> '.'<field_name>
section ::= { 'delivery_annotations' | 'd' }
```

This grammar subset allows for a minimal implementation that is limited to evaluating the delivery annotation properties defined here while being interoperable with the full SQL filter syntax.

Examples:

- Start from event at offset "a4c5" exclusively: `d.event-streams-offset > 'a4c5'`
- Start from event at offset "a4c5" inclusively: `d.event-streams-offset >= 'a4c5'`
- Start from event received after the timestamp: `d.event-streams-timestamp > 1585672841`

```
 <type name="event-streams-sql-filter" class="restricted" source="sql-filter"
provides="filter">
<descriptor name="amqp:event-streams-sql-filter" code="0x00000000:0x00000201"/>
</type>
```

The filter-type is “amqp:event-streams-sql-filter”, with type-code 0x00000000:0x00000201

6 Runtime Information

Each event log node MUST implement a special, subordinate source address named `$info` that allows producers and consumers to obtain information about the configuration and state of the event log, most importantly the current number of partitions and the offset boundaries for each partition.

If the container-relative source address of an event log node is "example", the information source address MUST be "example/$info".

To obtain information from the information source, the producer or consumer opens a receive link and flows at least one unit of link credit. The information source responds with transferring a message that contains the desired information in its `amqp-value` section. If multiple link credits are granted, the information source MAY choose to put reasonable, application-defined temporal separation between transfers.

The `amqp-value` section of the message contains an AMQP `map` at the top level. The map has one reserved entry with the key `partitions`, all other entries MAY be used for application specific extensions.

The `partitions` entry's value is a `list`, with one entry for each partition. The list MUST carry an entry for the "main partition" if the log is unpartitioned.

The entries in the partitions list are `map` objects and MUST contain the following reserved entries for each partition; all other entries MAY be used for application specific extensions.

Key	Type	Description
partition	symbol	The opaque identifier of the partition
earliest-offset	symbol	The earliest retained offset in the partition. MAY be null.
latest-offset	symbol	The latest retained offset in the partition. MAY be null.

7 Safety, Security, and Data Protection Considerations

This specification builds on and extends the AMQP 1.0 specification and does not introduce any further safety, security, or data protection concerns.

8 Conformance

When considering this specification, we can consider three distinct roles an AMQP container may play:

Firstly, that of a producing container, which sends messages to an event log node. Secondly, that of a consuming container, which receives messages from an event log node. Thirdly, that of the event log container, which implements one or more event log nodes.

For producing and consuming containers, there are no conformance constraints, because, as explained in the Default Behaviors section, all behaviors defined herein are optional.

Even though the behaviors are optional, there is a minimal set of features that MUST be implemented by an event log container to claim compliance with this specification:

1. Offer the connection capability defined in 3.1 Connection Capability
2. Implement the runtime information address defined in 6 Runtime Information
3. Implement at least one of the offset filters defined in 5 Delivery Filters

Supporting partitions is not required for conformance.

Appendix A. Acknowledgments

(Note: A Work Product approved by the TC must include a list of people who participated in the development of the Work Product. This is generally done by collecting the list of names in this appendix. This list shall be initially compiled by the Chair, and any Member of the TC may add or remove their names from the list by request.
Remove this note before submitting for publication.)

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

AMQP TC Members:

First Name	Last Name	Company
Paolo	Patierno	Red Hat
Jakub	Scholz	Red Hat
Rob	Godfrey	Red Hat
Clemens	Vasters	Microsoft

Appendix B. Revision History

Revision	Date	Editor	Changes Made
event-streams-v1.0-wd01	2020-06-22	Clemens Vasters	Initial working draft
event-streams-v1.0-wd02	2020-10-01	Clemens Vasters	Addressed feedback and completed missing sections