OASIS

Advanced Message Queuing Protocol (AMQP) Enforcing Connection Uniqueness Version 1.0

Committee Specification 01

17 September 2018

Specification URIs

This version:
> http://docs.oasis-open.org/amqp/soleconn/v1.0/cs01/soleconn-v1.0-cs01.xml (Authoritative)
> http://docs.oasis-open.org/amqp/soleconn/v1.0/cs01/soleconn-v1.0-cs01.html
> http://docs.oasis-open.org/amqp/soleconn/v1.0/cs01/soleconn-v1.0-cs01.pdf

Previous version:
> N/A

Latest version:
> http://docs.oasis-open.org/amqp/soleconn/v1.0/soleconn-v1.0.xml (Authoritative)
> http://docs.oasis-open.org/amqp/soleconn/v1.0/soleconn-v1.0.html
> http://docs.oasis-open.org/amqp/soleconn/v1.0/soleconn-v1.0.pdf

Technical Committee:
> OASIS Advanced Message Queuing Protocol (AMQP) TC

Chairs:
> Robert Godfrey (rgodfrey@redhat.com), Red Hat
> Clemens Vasters (clemensv@microsoft.com), Microsoft

Editor:
> Robert Godfrey (rgodfrey@redhat.com), Red Hat

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. 29 October 2012. OASIS Standard. http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-overview-v1.0-os.html.

Abstract:

The Advanced Message Queuing Protocol (AMQP) is an open internet protocol for business messaging. This document defines a mechanism by which two processes communicating using AMQP v1.0 can ensure and enforce that there exists only a single open AMQP connection between the two of them.

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 version" 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/.

This specification is provided under the RF on RAND Terms 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).

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:

[soleconn-v1.0]

Advanced Message Queuing Protocol (AMQP) Enforcing Connection Uniqueness Version 1.0. Edited by Robert Godfrey. 17 September 2018. OASIS Committee Specification 01.
http://docs.oasis-open.org/amqp/soleconn/v1.0/cs01/soleconn-v1.0-cs01.pdf.
Latest version: http://docs.oasis-open.org/amqp/soleconn/v1.0/soleconn-v1.0.pdf.

Notices:

Contents

1   Introduction

1.1   IPR Policy

This specification is provided under the RF on RAND Terms 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", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in [RFC2119] and [RFC8174] when, and only when, they appear in all capitals, as shown here.

The authoritative form of this specification consists of a set of XML source documents. These documents are transformed into PDF and HTML representations for readability. The machine readable version of the AMQP DTD describes the XML used for the authoritative source documents. This DTD includes the definition of the syntax used any excerpts of XML presented in the PDF and HTML representations.

1.3   Normative References

[AMQP]
Godfrey, Robert; Ingham, David; Schloming, Rafael, *Advanced Message Queuing Protocol (AMQP) Version 1.0*. October 2012, OASIS Standard.
<`http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-overview-v1.0-os.html`>

[RFC2119]
Bradner, S., *"Key words for use in RFCs to Indicate Requirement Levels"*, BCP14, RFC2119, DOI 10.17487/RFC2119, March 1997,
<`http://www.rfc-editor.org/info/rfc2119`>

[RFC8174]
Leiba, B., *"Ambiguity of Uppercase vs Lowercase in RFC2119 Key Words"*, BCP14, RFC8174, DOI 10.17487/RFC8174, May 2017,
<`http://www.rfc-editor.org/info/rfc8174`>

1.4   Non Normative References

[AMQPCONNCAP]
AMQP Capabilities Registry: Connection Capabilities
`http://www.amqp.org/specification/1.0/connection-capabilities`

[AMQPCONNPROP]
AMQP Capabilities Registry: Connection Properties
`http://www.amqp.org/specification/1.0/connection-properties`

2   Overview

AMQP connections are established between *containers*. On sending the initial `open` performative, each peer indicates its own container-id in the `container-id` field of `open`. The *container-id* is intended to uniquely identify the container amongst all containers that may establish connections to each other.

It is permitted for multiple connections to be open between two given containers at any given time. While this behavior is generally useful (for instance to allow load balancing across multiple network paths), in certain use cases it is important for the communicating peers to ensure that there is at most one open connection between them.

This document defines how an AMQP container may advertise its ability to detect multiple connections from the same peer, and how its partner may use this ability to request action to remedy the situation such that only one connection remains.

2.1   Dealing With Duplicates

There are essentially two behaviors that a container can implement to deal with unexpected duplicate connections from the same source:

1. refuse the new attempt to connect to the container
2. close the existing connection and admit the new connection

A container complying with this specification MUST be capable of supporting both these behaviors. The behavior the container detecting the duplicates uses MUST be determined by the behavior desired by its partner.

3   Duplicate Connection Detection

Within this section we shall use the term *requesting container* to denote the container which is requesting that its partner detect duplicate connections. We shall use the term *enforcing container* to denote the container which will detect the duplicate connections and perform any necessary enforcement actions.

Note that it is technically possible (though of dubious functional value) for both containers involved in the connection to be acting as both the *requesting container* and *enforcing container* simultaneously. Each container independently indicates its ability to perform the role of enforcer and its desire for its partner to do so.

3.1   Connection Capabilities

Name	Description
sole-connection-for-container	If present in the `desired-capabilities` field of `open` then the sender of `open` desires to act as the *requesting container*. If present in the `offered-capabilities` field of `open` then the sender of `open` performative supports detecting whether multiple connections from the same source container have been opened (i.e. the container is capable of acting as the *enforcing container*). The behavior the *enforcing container* MUST take on detecting a duplicate is determined by the value (if any) associated with the property *sole-connection-enforcement-policy* in the `properties` field of `open` sent by the *requesting container*.

3.2   Connection Properties

Name	Description
sole-connection-enforcement-policy	If present in the `properties` field of `open` sent by the *requesting container* the value determines the action the *enforcing container* MUST take upon detecting that this connection is a duplicate. The value associated with this property MUST be of type `sole-connection-enforcement-policy`. If the property is not present, the *enforcing container* MUST use the `refuse-connection` policy.

sole-connection-detection-policy	If present in the `properties` field of `open` sent by the *enforcing container* the value indicates its behavior when not all connections from the *requesting container* indicate the same requirements with regard to duplicate detection. The value associated with this property MUST be of type `sole-connection-detection-policy`. If the property is not present in the `properties` field of `open` sent by the *enforcing container* then the *requesting container* can assume `strong` detection.

3.2.1 Sole Connection Enforcement Policy

```
<type name="sole-connection-enforcement-policy" class="restricted" source="uint">
    <choice name="refuse-connection" value="0"/>
    <choice name="close-existing" value="1"/>
</type>
```

Defines the action the *enforcing container* MUST take if it detects that there is an already existing connection between the two containers.

Valid Values

0   Upon detecting the existence of an existing connection, the *enforcing container* will refuse the opening of a new connection. If the *enforcing container* has not already sent the `open` for the connection, it MUST add the property *amqp:connection-establishment-failed* to the `properties` field of `open` having boolean value true. The *enforcing container* MUST then immediately send a `close` and the `error` field of `close` MUST have an `error` with the `condition` field of `error` being `invalid-field` and the `info` field of `error` having the `symbol` key *invalid-field* taking the `symbol` value *container-id*.

1   Upon detecting the existence of an existing connection, the *enforcing container* will close the existing connection before opening the new one. The existing connection MUST be closed with the `error` field of `close` having the `condition` field of `error` being `resource-locked`. Further the `info` field of `error` MUST contain the `symbol` key *sole-connection-enforcement* taking the `boolean` value *true*.

3.2.2 Sole Connection Detection Policy

```
<type name="sole-connection-detection-policy" class="restricted" source="uint">
    <choice name="strong" value="0"/>
    <choice name="weak" value="1"/>
</type>
```

Defines the action the *enforcing container* takes if not all connections between the two containers establish the same sole connection enforcement.

Valid Values

0   If the *enforcing container* uses the *strong* detection policy then for every connection which is established, the *enforcing container* will check to see if a connection already exists between the two containers which has a sole connection enforcement requirement (even if the new connection does not state such a requirement). In such cases, the policy of the existing connection will be enforced. If any connections do exist between the two containers, but they do not have a sole connection enforcement requirement, then any sole connection enforcement policy of the new connection will be applied (if present).

1   If the *enforcing container* uses the *weak* detection policy then duplicate detection is only carried out when the new connection requires sole connection enforcement. In this case it is possible that an existing connection which was established with a sole connection detection requirement will no longer be the sole connection between the two containers, as a subsequent connection may not trigger duplicate detection.

4   Conformance

In 3. Duplicate Connection Detection the terms *enforcing container* and *requesting container* are defined for the different roles an implementation might play with respect to this specification. These roles are specific to an individual connection, and the same AMQP container may take on different roles on different connections.

A conforming implementation in the *enforcing container* role MUST adhere to the MUST and REQUIRED level requirements for an *enforcing container* defined in 3.1 Connection Capabilities and 3.2 Connection Properties.

A conforming implementation in the *requesting container* role MUST adhere to the MUST and REQUIRED level requirements for an *requesting container* defined in 3.1 Connection Capabilities and 3.2 Connection Properties.

Appendix A. Acknowlegements

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

- Robert Gemmell (Red Hat)
- Robert Godfrey (Red Hat)
- Ted Ross(Red Hat)

Appendix B. Revision History

Revision	Date	Editor	Changes Made
WD01	23-Mar-2017	Rob Godfrey	Initial Revision
WD02	13-Apr-2017	Rob Godfrey	AMQP-113 : Fix typos, and add clarification
WD03	6-Oct-2017	Rob Godfrey	AMQP-123 : Fix typos AMQP-125 : Add a (minimal) conformance section AMQP-126 : Clarify the strong sole-connection-detection-policy
WD04	20-Apr-2018	Rob Godfrey	AMQP-123 : Fix typos Update conformance section
WD05	15-Jun-2018	Rob Godfrey	Fix typos remove reference to the undefined 'AMQP Network'