AMQP Request-Response Messaging with Link Pairing Version 1.0

Committee Specification Draft 01

20 August 2020

This version:
https://docs.oasis-open.org/amqp/linkpair/v1.0/csd01/linkpair-v1.0-csd01.docx (Authoritative)
https://docs.oasis-open.org/amqp/linkpair/v1.0/csd01/linkpair-v1.0-csd01.html
https://docs.oasis-open.org/amqp/linkpair/v1.0/csd01/linkpair-v1.0-csd01.pdf

Previous version:
N/A

Latest version:
https://docs.oasis-open.org/amqp/linkpair/v1.0/linkpair-v1.0.docx (Authoritative)
https://docs.oasis-open.org/amqp/linkpair/v1.0/linkpair-v1.0.html
https://docs.oasis-open.org/amqp/linkpair/v1.0/linkpair-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:
Rob 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:

AMQP defines links as unidirectional transport for messages between a source and a target. A common messaging pattern is that of "request-response", that is, two parties partaking in a bidirectional conversation using messages. This document defines a common pattern for pairing two unidirectional links to create a bidirectional message transport between two endpoints.

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 document 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:

[Link-Pairing-v1.0]

AMQP Request-Response Messaging with Link Pairing Version 1.0. Edited by Rob Godfrey. 20 August 2020. OASIS Committee Specification Draft 01. https://docs.oasis-open.org/amqp/linkpair/v1.0/csd01/linkpair-v1.0-csd01.html. Latest version: https://docs.oasis-open.org/amqp/linkpair/v1.0/linkpair-v1.0.html.

Notices

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 Specifications, OASIS Standards, 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.

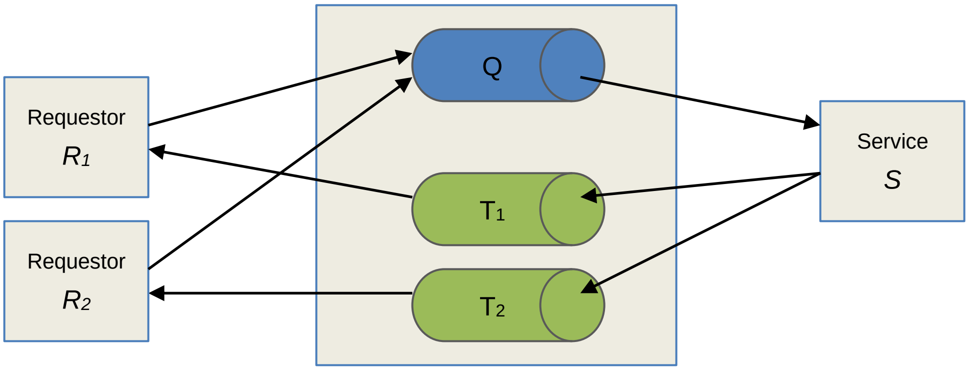
Table of Contents

1 Introduction 5
1.1 Terminology 6
1.2 Normative References 6
2 Link Pairing 7
2.1 Definition 7
2.1.1 Connection Capability 7
2.1.2 Propagation 8
2.2 Establishing A Link Pair 8
2.2.1 Link Property 9
2.2.1.1 Link Pairing and Routing Nodes 10
2.2.2 Pipelining 10
3 Conformance 11
Appendix A. Acknowledgments 12
Appendix B. Revision History 13

1 Introduction

A common messaging pattern is that of "request-response", that is two parties partaking in a bidirectional conversation where “request” messages are sent with the intention of generating a “response” to the requester.

One mechanism for implementing “request-response” messaging between a requestor R_i and a service S is for the service to have a single request queue Q, and for each requestor to create a temporary response queue T_i. Requestors send request messages to Q; the service reads from Q, generates a response and sends the response to the address T_i specified in the request message (this address being the temporary response queue); the requestor.



Section 3.2.4 of [**AMQP**] defines the reply-to message property as a mechanism to convey the address which response messages should be sent to. Sections 3.5.3 and 3.5.4 of [**AMQP**] define a mechanism for requesting “dynamic” nodes to be created.

The above method works well where the messaging between the requestor R_i and the service S travels via an intermediary *I* which is capable of creating dynamic nodes, however there are a number of drawbacks to its use:

- it requires the container to which the requestor is connected to support the creation of dynamic nodes
- it requires dynamic nodes created on the intermediary to have an address which can be routed to from anywhere in the network
- it requires the creation of the dynamic response node to be completed synchronously before the first request message can be sent (since the requestor must wait to learn the address of the node so that it can be used in the reply-to)

Even without considering the problems of dynamic nodes, the existing mechanisms for request-response have issues when messages are traversing between multiple containers. Consider nodes X (in container C_x) and Z (in container C_z) which can only route to each other via a third node Y (in container C_y). There is no way for X to express in the immutable reply-to that it wants the response to a message to come back to itself in a way that will make sense to Z, especially considering that Y might completely obscure X’s and Z’s existence from each other.

This document defines a mechanism for request-response that does not rely on the creation of dynamic nodes, and allows for a bi-directional conversation to be established even where there exists no way to address nodes in the requester’s domain directly from the service’s domain. This is achieved by explicitly combining two links (one inbound and one outbound) between the same addresses providing a bidirectional communication channel.

1.1 Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

1.2 Normative References

[AMQP] Godfrey, Robert; Ingham, David; Schloming, Rafael, *Advanced Message Queuing Protocol (AMQP) Version 1.0*, 29 October 2012. OASIS Standard. http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-overview-v1.0-os.html.

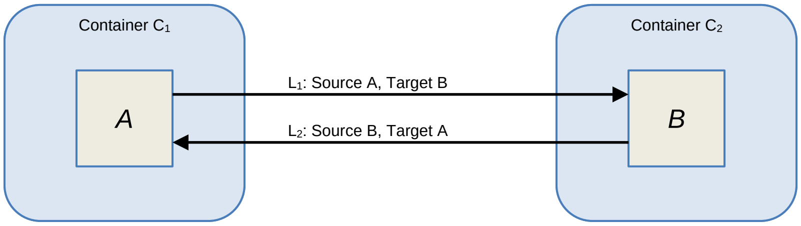
[ANONTERM] *Using the AMQP Anonymous Terminus for Message Routing Version 1.0*. Edited by Robert Godfrey. 17 September 2018. OASIS Committee Specification 01. http://docs.oasis-open.org/amqp/anonterm/v1.0/cs01/anonterm-v1.0-cs01.html.

[RFC2119] Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels”, BCP 14, RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt.

2 Link Pairing

2.1 Definition

A link L_1 provides a unidirectional transport for messages from a source *A* in container *C_1*, to target *B* in container *C_2*. If a second link L_2 is created from source *B* in *C_2* to target *A* in *C_1* then we can logically combine these two links to create a bidirectional message transport between *A* and *B*.



In the standard case for request-response, the service at *B* will send responses to the reply-to address specified on the request message. To inform the service that response messages are to be sent on the dedicated paired link, the request message MUST have the reply-to property set to the literal string value $me. If the reply-to property on a message sent over a paired link is set to a value other than $me, the response from the service at B must be directed to that reply-to address and not sent on the dedicated paired link.

Two links L_1 and L_2 are considered to be paired when the following conditions are met:

- The source container for L_1 is the target container for L_2
- The target container for L_1 is the source container for L_2
- The source address for L_1 is identical to the target address for L_2
- The source address for L_2 is identical to the target address for L_1
- The link name for L_1 is identical to the link name for L_2
- Both L_1 and L_2 have been established with the property paired set to the boolean value true

2.1.1 Connection Capability

On connection establishment a peer MUST indicate whether it supports the creation of pair links and/or whether it may desire to establish a link pair with its partner. This is done through the exchange of connection capabilities (see Section 2.7.1 [**AMQP**]).

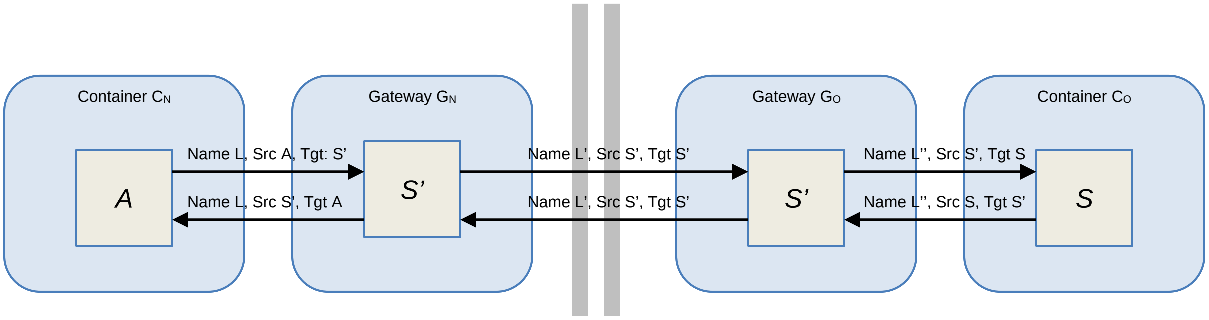
Capability Name	Definition
LINK_PAIR_V1_0	If present in the offered-capabilities field of the open performative, the sender of the open supports the creation of link pairs by its partner. A container which does not support the initiation of link attachment by its partner (for example a client library which will only ever initiate link attachment itself) MUST NOT offer this capability. If present in the desired-capabilities field of the open performative, the sender of the open MAY attempt to initiate link pair creation if the receiver of the open supports this capability.

A container that supports the creation of link pairs MAY only do so for some addresses while not supporting paired links to others. For example, a node providing store-and-forward style semantics cannot support pairing – the store-and-forward node will not itself be generating responses.

2.1.2 Propagation

Where no direct connection is possible between two containers, a bidirectional communication for request response can be created by propagating paired link creation through a network. Such propagation also allows for the ultimate source and target addresses to be hidden.

For example, a container C_O in organization *O* may offer a service at internal address *S*. A gateway G_O is configured to expose this service through a public address *S'*. In organization *N* a client wishes to communicate with this service – the client initiates a paired link with address *S'* on organization N's gateway G_N. The gateway G_N acts on the establishment of this paired link by establishing a paired link to the public address S' on G_O. Finally the gateway G_O establishes a paired link to *S* in container C_O.



Note that the internal addresses *A* and *S* are never visible outside of the network of their own organizations.

2.2 Establishing A Link Pair

A link pair is established by attaching two links with the same name, but in opposite directions with the source of one link being the target of the other (and vice versa) and with both links having the property paired being associated with the `boolean` value `true`.

```
Peer                                  Partner
================================================================
  ATTACH(name=N,            ---------->
       handle=1,                  +-- ATTACH(name=N,
       role=sender,               |          handle=1,
       source=A,                  |          role=receiver,
       target=B,                  |          source=A,
       properties={paired: true}) |          target=B,
                                  |          properties={paired: true})
                          <-------+

ATTACH(name=N,            ---------->
       handle=2,                  +-- ATTACH(name=N,
       role=receiver,             |          handle=2,
       source=B,                  |          role=sender,
       target=A,                  |          source=B,
       properties={paired: true}) |          target=A,
                          <-------+          properties={paired: true})

FLOW(handle=2,            ---+   +--- FLOW(handle=1,
     link-credit=1)           \ /          link-credit=1)
                               X
                              / \
```

2.2.1 Link Property

On creating, reattaching or resuming link which forms one half of a pair, the properties field of the attach performative MUST contain an entry with the literal symbol key paired and boolean value true.

Property Name	Definition
paired	If present with and having the boolean value true, then sender of the attach intends for this link to form half of a link pair. If not present or the value is anything other than the boolean value true then the sender of the attach does not intend for this link to be paired.

If the value (or existence) of the paired property differs between the attach emitted by the sending and receiving sides of the same link, then the link MUST immediately be detached with the error condition precondition-failed as defined in section 2.8.15 of [**AMQP**].

If an attach is received with the paired property set to true for an address which does not support link pairing, then the attach MUST be emitted with the local terminus set to null (indicating a failure to create the link) and immediately followed by a detach with the closed field having value true and indicating the not-implemented error condition as defined in section 2.8.15 of [**AMQP**].

If an attach is received with the paired property set to true and there exists a link in the opposite direction with the same name, but with the local or remote terminus address differing with that in the address, then the link for which the attach has been received MUST immediately be detached with the error condition precondition-failed as defined in section 2.8.15 of [**AMQP**].

2.2.1.1 Link Pairing and Routing Nodes

A *Routing Node* as defined by **[ANONTERM]** is a node which routes messages based on the `to` field of the `properties` section of a message. If a routing node elects to support link pairing, then it must effectively create a logical link pair between itself and each of the nodes to which it routes messages. If a message is sent to a routing node with a `to` address such that attempting to create a link pair to that address would result in failure, then the routing node MUST treat this as described in 2.2.2 Routing Errors of **[ANONTERM]**.

2.2.2 Pipelining

> **Note:** Pipelining as described in this section SHOULD NOT be attempted unless the initiator of the link pair has *a priori* knowledge that the responding peer is capable of handling pipelined requests. Pipelining relies on the assumption that the recipient of request messages will automatically provide link credit upon attachment of an inbound link, and thus it is safe to transfer a message on the link without waiting for notification that credit is available.

In order to reduce latency and maximize performance, it may be desirable to pipeline the creation and use of the link pair. In order to perform request-response interaction, the initiator requires a sending link (for the request) to be established; a receiving link (for the response) to be established; the receiving link to have sufficient credit to receive the response message; and the request message to be transferred. Note that the credit on the receiving link MUST be issued before the request message is transferred.

```
Peer                                  Partner
================================================================
ATTACH(name=N,            ---------->
       handle=1,
       role=sender,
       source=A,
       target=B,
       properties={paired: true})

ATTACH(name=N,            ---------->
       handle=2,
       role=receiver,
       source=B,
       target=A,
       properties={paired: true})
```

Pipelined establishment may fail if the service is not able to issue credit immediately upon link establishment, in which case the transfer of the request message will cause the link to be detached with the transfer-limit-exceeded error condition.

3 Conformance

An AMQP Container conforms to the requirements of a paired link creator if:

1. It complies with the requirements for the advertising the LINK_PAIR_V1_0 capability in the `offered-capabilities` field of the `open` performative as per 2.1.1.
2. The behavior when receiving an `attach` performative (with or without the paired property being set to `true`) complies with the relevant definitions in 2.2.1.

Appendix A. Acknowledgments

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

Participants:
Alan Conway, Red Hat
Robert Godfrey, Red Hat
Robbie Gemmell, Red Hat
David Ingham, Red Hat
Ted Ross, Red Hat
Clemens Vasters, Microsoft
Keith Wall, Red Hat

Appendix B. Revision History

Revision	Date	Editor	Changes Made
WD01	8-Mar-2017	Robert Godfrey	Initial Working Draft
WD02	6-Oct-2017	Robert Godfrey	Address AMQP-96, AMQP-97, and AMQP-98
WD03	15-May-2019	Robert Godfrey	Refer to Anonymous Terminus specification and clarify behavior when pairing with routing nodes
WD04	14-Jun-2019	Robert Godfrey	Add conformance clauses