HTTP Semantics and Content over AMQP Version 1.0

Working Draft 06  
2 July 2019

**Technical Committee:**

[OASIS Advanced Message Queuing Protocol (AMQP) TC](https://www.oasis-open.org/committees/amqp/)

**Chairs:**

Rob Godfrey ([rgodfrey@redhat.com](mailto:rgodfrey@redhat.com)), [Red Hat](http://www.redhat.com/)

Clemens Vasters ([clemensv@microsoft.com](mailto:clemensv@microsoft.com)), [Microsoft](http://www.microsoft.com/)

**Editors:**

Clemens Vasters ([clemensv@microsoft.com](mailto:clemensv@microsoft.com)), [Microsoft](http://www.microsoft.com/)

Alan Conway ([aconway@redhat.com](mailto:aconway@redhat.com)), [Red Hat](http://www.redhat.com/)

**Related work:**

This specification is related to:

* *OASIS Advanced Message Queuing Protocol (AMQP) Version 1.0 Part 0: Overview*. Edited by Rob 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:**

This document defines a mapping of IETF HTTP 1.1 semantics and content as defined in RFC7231 onto AMQP 1.0. This mapping allows for APIs that use HTTP as their application protocol model to leverage AMQP as framing and transfer protocol.

**Status:**

This [Working Draft](https://www.oasis-open.org/policies-guidelines/oasis-defined-terms-2017-05-26#dWorkingDraft) (WD) has been produced by one or more TC Members; it has not yet been voted on by the TC or [approved](https://www.oasis-open.org/policies-guidelines/tc-process#committeeDraft) as a Committee Draft (Committee Specification Draft or a Committee Note Draft). The OASIS document [Approval Process](https://www.oasis-open.org/policies-guidelines/tc-process#standApprovProcess) begins officially with a TC vote to approve a WD as a Committee Draft. A TC may approve a Working Draft, revise it, and re-approve it any number of times as a Committee Draft.

This specification is provided under the [RF on RAND Terms](https://www.oasis-open.org/policies-guidelines/ipr#RF-on-RAND-Mode) Mode of the [OASIS IPR Policy](https://www.oasis-open.org/policies-guidelines/ipr), 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](https://www.oasis-open.org/policies-guidelines/tc-process#wpComponentsCompLang)) 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.

**URI patterns:**

Initial publication URI:  
https://docs.oasis-open.org/amqp/http-over-amqp/v1.0/csd01/http-over-amqp-v1.0-csd01.docx

Permanent "Latest version" URI:  
https://docs.oasis-open.org/amqp/http-over-amqp/v1.0/http-over-amqp-v1.0.docx

Copyright © OASIS Open 2018. All Rights Reserved.

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](https://www.oasis-open.org/policies-guidelines/ipr) 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 PA

**Table of Contents**

[1 Introduction 4](#_Toc13045610)

[1.1 IPR Policy 4](#_Toc13045611)

[1.1 Terminology 4](#_Toc13045612)

[1.2 Normative References 5](#_Toc13045613)

[2 Connection Setup 6](#_Toc13045614)

[1.3 AMQP Start 6](#_Toc13045615)

[1.4 HTTP Start 6](#_Toc13045616)

[3 Addressing 7](#_Toc13045617)

[1.5 URI encoding 7](#_Toc13045618)

[2 HTTP Message Mapping 8](#_Toc13045619)

[2.1 Projected Mode 8](#_Toc13045620)

[2.1.1 Request Line 8](#_Toc13045621)

[2.1.2 Status Line 8](#_Toc13045622)

[2.1.3 Headers 8](#_Toc13045623)

[2.1.4 Entity Body 9](#_Toc13045624)

[2.2 Tunneled Mode 9](#_Toc13045625)

[2.3 Large Messages 10](#_Toc13045626)

[4 Correlation 11](#_Toc13045627)

[2.4 Request/Response message correlation 11](#_Toc13045628)

[2.5 Zero or Multiple Responses 11](#_Toc13045629)

[2.6 Disposition correlation and settlement 11](#_Toc13045630)

[5 Example use-cases 12](#_Toc13045631)

[2.7 HTTP/AMQP bridging 12](#_Toc13045632)

[2.8 Hybrid HTTP/AMQP 12](#_Toc13045633)

[2.9 Request link addresses 12](#_Toc13045634)

[2.10 Response link addresses 13](#_Toc13045635)

[6 Security Considerations 14](#_Toc13045636)

[7 Conformance 15](#_Toc13045637)

[Appendix A. Acknowledgments 16](#_Toc13045638)

# Introduction

HTTP semantics and HTTP’s content model [RFC7231] are used for a broad variety of networked services. HTTP 1.1 [RFC7230] binds the content model to TCP with a largely text-based protocol encoding, and HTTP/2 [RFC7540] introduced a binary encoding with support for stream multiplexing also on top of TCP.

Like HTTP/2, AMQP [AMQP] provides multiplexed stream communication with per-stream flow control and binary encoding metadata. Different from HTTP, AMQP is a fully bi-directional messaging protocol that allows for transfers and correlated message exchanges in any desired shape and without any directional bias.

For applications of AMQP where the communicating parties wants to use a request/response exchange pattern and one of the parties offers a structured set of resources for the other party to interact with using a uniform interface, the HTTP semantics and content model is a great foundation and there is no need to reinvent it.

In particular, the AMQP Management [AMQP-MAN] specification defines a resource-centric API for managing the topology and configuration of AMQP containers, and that API is modeled on top of the HTTP interface conventions and can indeed be implemented both on top of AMQP over HTTP and HTTP proper.

Furthermore, AMQP provides robust state management for transfers, including transactional coordination of transfers, and recovery from connection failures, far exceeding the capabilities of HTTP over TCP, and is therefore well suited for request/response interactions with HTTP semantics that require a high degree of reliability.

This specification provides a formal mapping of the HTTP semantics and content model on top of AMQP that fully leans on AMQP’s underlying framing model and architectural elements. The goal is for any AMQP 1.0 protocol implementation to be able to use this mapping while concurrently allowing every other message exchange pattern in the same application and even on the same connection.

This specification also clarifies integration with HTTP network endpoints through the existing AMQP WebSockets binding. AMQP containers that offer WebSockets endpoints can help upgrading clients to using HTTP-over-AMQP starting from an initial HTTP 1.1 request similar to how HTTP/2 performs its protocol upgrade.

Lastly, this specification also defines a tunneling model by which an entire HTTP 1.1 message request or response frame can be mapped to an AMQP message. Tunneling is interesting in scenarios where an HTTP network listener to the edge of a system needs to pass raw HTTP requests to a responding party only reachable via an AMQP route.

## IPR Policy

This specification is provided under the [RF on RAND Terms](https://www.oasis-open.org/policies-guidelines/ipr#RF-on-RAND-Mode) Mode of the [OASIS IPR Policy](https://www.oasis-open.org/policies-guidelines/ipr), 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>).

## 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](#3dy6vkm)] and [[RFC8174](#1t3h5sf)] when, and only when, they appear in all capitals, as shown here.

For this specification, the term “client” is derived from HTTP parlance and is used as a synonym for an AMQP container that issues HTTP-over-AMQP requests to another container. To do so, the client might initiate a new connection, but it might also use an existing AMQP connection or some other HTTP route.

## Normative References

**[RFC3986]** T. Berners-Lee, R. Fielding, L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax”, January 2005  
 <<https://www.rfc-editor.org/rfc/rfc3986.txt>>

**[****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>>.

**[RFC7230]** Fielding, R., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

**[RFC7231]** Fielding, R., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

**[RFC7232]** Fielding, R., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC 7232, DOI 10.17487/RFC7232, June 2014, <https://www.rfc-editor.org/info/rfc7232>.

**[AMQP]** OASIS Advanced Message Queuing Protocol (AMQP) Version 1.0: Overview. Edited by Rob 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.

**[AMQP-PL]** AMQP Request-Response Messaging with Link Pairing - WD02.https://www.oasis-open.org/committees/document.php?document\_id=61722&wg\_abbrev=amqp

**[AMQP-ADDR]** AMQP Addressing - https://www.oasis-open.org/committees/document.php?document\_id=64359&wg\_abbrev=amqp

**[AMQP-AT]** Using the AMQP Anonymous Terminus for Message Routing Version 1.0 -<http://docs.oasis-open.org/amqp/anonterm/v1.0/anonterm-v1.0.html>

**[AMQP-MAN]** AMQP Management -<https://www.oasis-open.org/committees/document.php?document_id=63931&wg_abbrev=amqp>

# Connection Setup

A client can choose between an AMQP start and an HTTP (WebSockets) start for a connection. Typically, the client will make that decision based on the scheme of a URI. The “amqp” or “amqps” schemes call for an AMQP start, while the “http” and “https” and “ws” and “wss” schemes call for a HTTP start.

## AMQP Start

HTTP over AMQP is a regular application of AMQP 1.0 [AMQP] and uses its default [connection model](http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-transport-v1.0-os.html#section-connections).

Before sending an HTTP request, an initially disconnected client must establish a TCP connection to the remote AMQP container, negotiate TLS, negotiate SASL, establish the AMQP connection, establish an AMQP session, and create a pair of request and response AMQP links.

While that might appear very costly, all of the setup after the TLS handshake and depending on the chosen SASL method, can be performed in a pipelined fashion and without waiting for the other party to acknowledge each step.

A client could indeed send [sasl-init](http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-security-v1.0-os.html#type-sasl-init), [open](http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-transport-v1.0-os.html#type-open), [begin](http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-transport-v1.0-os.html#type-begin), and a pair of [attach](http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-transport-v1.0-os.html#type-attach) frames sequenced inside a single TCP/TLS frame, because none of the information carried in those frames depends on information obtained by the peer. Anticipating an initial flow control grant from the peer after *attach,* this opening TCP/TLS frame could even hold the first [transfer](http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-transport-v1.0-os.html#type-transfer) and a [flow](http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-transport-v1.0-os.html#type-flow) grant for the response. This pipelining model assumes that the required link-credit for the initial transfer and the session-window grant for the frames sent after begin will be issued promptly by the receiving party as it begins the session and accepts the link from its end.

The configuration of the AMQP links is further discussed in section 3 below.

An AMQP session inside a connection can host any number of concurrent links (and therefore link pairs). A connection to a container SHOULD therefore be (re-)used for multiple concurrent requests directed at the same container. While HTTP requests are exchanged, the same connection can concurrently be used for one-way flows like carrying telemetry.

## HTTP Start

When the client starts with an HTTP 1.1 request, it can ask for an upgrade to AMQP over WebSockets as described in the [AMQP WebSocket Binding](http://docs.oasis-open.org/amqp-bindmap/amqp-wsb/v1.0/amqp-wsb-v1.0.html) [AMQP-WSB].

After the initial negotiation of the WebSockets connection, the resulting stream of AMQP frames is identical independent of the initial connection method.

As permitted in [RFC7230, 6.7](https://tools.ietf.org/html/rfc7230#section-6.7), a client MAY ask for an upgrade to other protocols like HTTP/2 and to WebSockets with the AMQP subprotocol in the same request. It is then up to the server to choose which protocol is wants to offer.

A client that supports this specification SHOULD always request an upgrade to AMQP.

# Addressing

The addressing models of AMQP and HTTP are meaningfully different. HTTP has two addressing elements, a network endpoint (*host* and port) and a *request-target*. AMQP has three main elements: A network endpoint, a scope identifier, and a path.

The key difference between the addressing approaches is that AMQP separates the identity of containers (servers or clients) from the identity external IP network endpoints.

With AMQP, an external client establishes an IP network connection to an AMQP container and becomes a member of an AMQP network. All further operations that involve addressing or routing are performed within the scope of the AMQP network and IP network endpoint information plays no further role. This abstraction permits an AMQP network to span multiple, mutually isolated IP networks. HTTP only operates within the scope of a single IP network and its host information is scoped to that.

To bring the models in alignment for the use with this specification, the following mapping rules MUST be followed. The mapping yields two information items, the HTTP *host* and the HTTP *request-target*. These are mapped to the AMQP message in section 4.

* For URIs with the *http*, *https, ws,* and *wss* schemes (HTTP URI)
  + The “*host*” portion of the URI corresponds to the AMQP network endpoint
  + The URI path and query, the HTTP *request-target*, follows AMQP URI format rules. The request-target corresponds to an AMQP path and therefore to an AMQP node.
* For URIs with the *amqp* and *amqps* schemes (AMQP URI)
  + HTTP *host* corresponds to the AMQP network endpoint if the scope identifier is empty.
  + HTTP *host* corresponds to the AMQP scope identifier if the scope identifier is not empty.
  + The URI path and query follow the AMQP URI format rules as HTTP *request-target*

The effect of these rules is that HTTP URIs that might be given to a client used for an HTTP start and upgrade of the connection are always bound to a particular IP network and can only reference nodes (more correctly: termini) in the container at the given IP network endpoint.

For reaching nodes in containers that don’t correspond to the network endpoint but require routing, the AMQP URI MUST be used, even if the application wants to use an HTTP start for the initial connection. AMQP nodes who produce self-referencing URIs in metadata or content MUST use the AMQP URI.

## URI encoding

AMQP has several fields designated for addresses:

* to and reply-to fields in message properties
* address field in link source and target

AMQP address fields MUST NOT be URI-encoded [RFC3986] even when their contents are derived from a URI-reference. When a URI reference is used to set an AMQP address field it MUST be URI-decoded. When an AMQP address field is used to make a URI-reference for HTTP, it MUST be URI-encoded.

HTTP header values projected into AMQP message-properties MUST be copied verbatim in both directions. Automatic URI encoding and decoding MUST NOT be performed except on the fields listed above.

# HTTP Message Mapping

HTTP Messages are mapped to AMQP in one of two modes: Projected or tunneled. In tunneled mode, which is aimed at the tunneling use-case explained in the introduction, the entire HTTP message, including the leading request or status line, is enclosed in the AMQP body section as a byte array. In projected mode, the HTTP message elements are projected onto their corresponding AMQP message elements as defined here.

## Projected Mode

The projected mapping mode transcribes HTTP headers and content into AMQP elements. An implementation MUST ignore and exclude all RFC7230 headers and RFC7230 information items not explicitly covered below and MUST include all headers and information items from RFC7231 and other HTTP extension specifications.

HTTP requests and responses can be clearly distinguished by the RECOMMENDED "http:request" and "http:response" properties, which carry the version of HTTP as their value. If neither property is present, requests can still be identified by the presence of a "reply-to" property value and responses can be identified by the presence of a "correlation-id" property value:

For HTTP request messages *reply-to* MUST be set, and *message-id* SHOULD be set unless there is only one request being made with this *reply-to* address. For HTTP response messages *correlation-id* MUST be set to the *message-id* of the corresponding request, if present.

### Request Line

The HTTP [request-line](https://tools.ietf.org/html/rfc7230#section-3.1.1) contains the HTTP method, the request-target, and the HTTP version.

* HTTP method MUST be set in the *properties* section, *subject* field.
* HTTP request-target value MUST be URI-decoded and set in the *properties* section, *to* field.
* The version value SHOULD be set in *application-properties* section, as value of the “http:request” string property. The assumed default value is “1.1” if the property is absent.

### Status Line

The HTTP [status-line](https://tools.ietf.org/html/rfc7230#section-3.1.2) contains the HTTP version, the status code, and the reason phrase.

* The version value SHOULD be set in *application-properties* section, as value of the “http:response” string property. The assumed default value is “1.1” if the property is absent.
* HTTP status code MUST be set in the *properties* section, *subject* field.
* HTTP reason phrase is OPTIONAL. The value is set in *application-properties* section, as value of the “http:reason” string property. The property SHOULD be omitted because RFC7230 3.1.2 RECOMMENDS it being ignored by clients.

### Headers

The following HTTP Headers defined in RFC7230 MUST NOT be mapped into AMQP HTTP messages:

* TE
* Trailer
* Transfer-Encoding
* Content-Length
* Via
* Connection
* Upgrade

The HTTP *Host* information MUST follow the addressing rules defined in section 3. While the *Host* header is required in RFC7230, it is OPTIONAL in HTTP AMQP because the container is already uniquely identified through other means. The *Host* value is set in the *application-properties* section, as value of the “http:host” string property. When the property is omitted, the default value is a container-defined scope identifier.

The following RFC7231 headers have special mappings:

* *Content-Type* maps to the *properties* section, *content-type* field.
* *Content-Encoding* maps to the *properties* section, *content-encoding* field.
* *Date* maps to the *properties* section, *creation-time* field.
* *From* maps to the *properties* section, *user-id* field.

The following RFC7234 header has a special mapping:

* The *Expires* value maps to the *properties* section, *absolute-expiry-time* field.

All other HTTP headers, including those defined in RFC7231 and any headers defined by formal HTTP extensions as well as any application specific HTTP headers are added to the *application-properties* section of the message. The header names are not prefixed. Headers with special mappings MUST NOT be added to the *application-properties* section.

Because HTTP header names are case-insensitive but AMQP property names are case-sensitive, all HTTP header names MUST be converted to lower case as they are mapped to AMQP application-property names. The type of all mapped header values is *string*.

### Entity Body

The entity body of the HTTP message is stored as one or more *data* *sections* of the AMQP message. The entity body’s content type and content encoding are described by the *content-type* and *content-encoding* fields of the AMQP *properties* section, following RFC7231 rules. When interpreting the message content, it MUST be considered as equivalent to a single data section obtained by concatenating all the data sections, the data section boundaries MUST be ignored.

## Tunneled Mode

The tunneled mode maps a complete HTTP 1.1 message into an AMQP message body. The HTTP 1.1 message MUST be encoded with the Media Type “message/http” and according to [RFC7230 8.3.1.](https://tools.ietf.org/html/rfc7230#section-8.3.1)

The resulting *content-type* and *content-encoding* metadata values are stored in the respective fields in the AMQP message’s *properties* section.

A tunneled mode message can be distinguished from a projected mode message by having the “message/http” media type in the *content-type* field.

Since tunneled mode might be used for application-specific opaque routing of HTTP content, metadata mappings are OPTIONAL. An implementation MAY use the projected mode’s metadata mapping rules (4.1.1, 4.1.2, 4.1.3) to provide an AMQP infrastructure visibility into the content of the carried message (with the exception of *content-type* and *content-encoding*).

## Large Messages

Both AMQP and HTTP can send large messages as a stream of smaller pieces. HTTP uses RFC7230’s chunked transfer encoding. AMQP allows multiple logical data sections which are further segmented into frames.

A HTTP message is represented as an AMQP message with one or more data sections.

# Correlation

HTTP 1.1 as defined in RFC7230 achieves correlation between requests and responses by mandating that requests must be replied to in the exact request order on the same underlying bidirectional connection. HTTP/2 as defined in RFC7540 achieves the same by confining the pair of request and responses to a single bidirectional stream on top of its multiplexing model.

AMQP’s communication paths, links, are unidirectional and the protocol is completely asynchronous in that all traffic going over links in either direction is independent of each other. Because of that fundamental difference, HTTP messages require explicit correlation on AMQP.

This section discusses the relationship between request and response messages. The next section discusses delivery paths and how the request and response delivery paths are correlated.

## Request/Response message correlation

To associate a response with a request, the *correlation-id* value of the response *properties* MUST be set to the *message-id* value of the request *properties*.

Responses MUST use the same mapping mode (see 4) as the corresponding request.

For responding to requests that were split across multiple messages (see 4.1.4.1 and 4.2.1.1), the *correlation-id* is set to the *message-id* of the request’s start message. If the response is split across multiple messages, the correlation-id MUST be set to the same value on all responses.

## Zero or Multiple Responses

AMQP is commonly used in publish/subscribe systems where copies of a single original message are distributed to zero or many subscribers. AMQP permits zero or multiple responses to a message with the reply-to property set, which can be correlated using the correlation-id property. Messages that were mapped from HTTP can be used like any other AMQP message, they are not required to be correlated one-to-one and in order like HTTP 1.1 messages.

## Disposition correlation and settlement

A request transfer MUST be *rejected* when the message is malformed such that producing a response message is impossible. The rejection settlement MAY include error information explaining the reason for the rejection.

If AMQP transactions are supported by all the involved parties, posting of the response transfer and disposition of the request transfer SHOULD be performed under the same AMQP transaction, preventing an inconsistent failure or success state. When the failure cause persists, the request SHOULD ultimately be *rejected* with error information referring to the failed transaction.

# Example use-cases

This section contains non-normative descriptions of how the HTTP-over-AMQP mapping can be used. The examples are:

* HTTP-AMQP bridge: A protocol bridge to connect plain HTTP and AMQP components.
* Hybrid HTTP-AMQP: Dual-protocol components for deeper integration of the protocols.

## HTTP/AMQP bridging

A protocol bridge act as an AMQP or HTTP server on one side, and an HTTP or AMQP client on the other, mapping HTTP requests and responses to AMQP messages as described in this specification. The bridge assumes that connected HTTP clients and servers have no knowledge of AMQP, and that connected AMQP clients have no knowledge beyond the rules for setting message *subject* and *to* addresses. In particular it handles request-response correlation and generation/translation of *reply-to* addresses automatically.

Use cases:

1. Route HTTP over and AMQP network:   
   HTTP client -> ingress bridge -> AMQP routers -> egress bridge -> HTTP server.
2. HTTP client access to AMQP servers:   
   HTTP client -> bridge -> [ AMQP routers -> ] -> AMQP server.
3. AMQP client access to HTTP servers:   
   AMQP client -> [AMQP routers->] -> bridge -> HTTP server

For case 1. the AMQP routing network does not need any knowledge of HTTP. The ingress bridge generates suitable AMQP *reply-to* and *message-id* values to correlate requests and responses. The egress bridge maps HTTP request/response correlation (HTTP1 ordering, HTTP2 stream-identifiers) to AMQP correlation via *reply-to* and *correlation-id*.

Cases 2 and 3 require the AMQP client/server to be aware of the rules for HTTP-over-AMQP messages defined by this specification, but don't require any HTTP protocol support. For the most part, the rules in this specification are compatible with normal AMQP usage - the main exception being the convention for the *subject* field. A bridge MAY adopt more relaxed rules for messages from incoming clients, for example by providing a default *subject* value.

## Hybrid HTTP/AMQP

Components that understand AMQP, HTTP and this specification can provide a deeper integration. For example a hybrid AMQP/HTTP server could accept HTTP 1.1 connections and then allow upgrade to an AMQP connection that can carry HTTP-over-AMQP messages as defined by this specification.

## Request link addresses

AMQP defines addresses at two levels: each AMQP *link* has a *source* and *destination* address, and each AMQP message has a *to* and *reply-to* address. Bridging and routing components can use this separation to give additional flexibility. For example:

1. AMQP link per HTTP URL:
   * link *target* == URL
   * message *to* is identical to *target*.
2. AMQP link per HTTP URL hierarchy:
   * link *target* == URL prefix
   * message *to* is URI ref.  
     A HTTP dispatcher might strip the *target* prefix from the *to* address, then treat the result as a URI reference relative to the *target*
3. Anonymous relay:
   * null link *target*
   * message *to* is URL.
4. Named relay:
   * arbitrary link *target*
   * message *to* is URL

The anonymous relay feature is an optional AMQP feature, and when available there is only one anonymous available relay per container. A *named relay* is a normal link used in a similar way ( to route to arbitrary addresses over a single link) but you can use multiple named relays to handle different subsets of available targets, possibly with different security or other needs.

This specification does not mandate a particular use of link addresses, the above are given as examples and do not preclude other uses.

## Response link addresses

This specification does not mandate a particular strategy for chosing *reply-to* addresses or establishing AMQP links for response messages. All of the strategies available for AMQP request-response messaging can be used, for example:

1. Well known reply-to (e.g. a queue or topic on a well-known broker)
2. Paired Links [AMQP-PL]
3. Pre-existing link: established by requester, messages flow from responder

* Dynamic source link: [AMQP 3.5.3], responder creates unique source address.
* Named target link: requester assigns target address.

1. Anonymous or named relay link, must be agreed between responder/requester in advance.

Not all of these options are available in all containers, an implementation can provide some or all of them. This list does not preclude other alternatives.

# Security Considerations

The AMQP security model and HTTP security model both stay intact and an implementation MAY use either or both or neither.

That means that an implementation might choose to allow for the AMQP layer to omit security altogether or use the ANOYMOUS SASL mechanism and only rely on the [“Authorization” HTTP header](https://tools.ietf.org/html/rfc7235#section-4.2) [RFC 7235, 4.3] on a per-request basis.

An implementation might also choose to use a different AMQP SASL mechanism and use the established security context for all HTTP requests performed over the connection.

Since the reply-to path is explicit, some of the response strategies enumerated in 6.4 might require for the responding party to be authorized to deliver the reply.

# Conformance

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

Firstly, that of a Requesting Container– a container which wants to initiate HTTP requests; secondly an Responding Container – a container which responds to such requests.

A Requesting Container is conformant with this specification if:

* It can map HTTP requests to an AMQP message in projected mode (section 4.1) or in tunneled mode (section 4.2) or both, and the message can be sent via an AMQP link to a destination that corresponds to the address indicated in the request.
* It can receive an AMQP message from a source that corresponds to the reply-to destination indicated by the request (see sections 5 and 6) and map received AMQP messages back into HTTP responses in projected mode (section 4.1) or in tunneled mode (section 4.2) or both.
* If the responding and requested containers have not negotiated support for link pairing [AMQP-PL], the reply-to destination MUST NOT use the reserved “$me” reply-to address.

A Responding Container is conformant with this specification if:

* If it can receive AMQP messages holding mapped HTTP requests and can handle them as HTTP requests in projected mode (section 4.1) or in tunneled mode (section 4.2) or both, producing at least one correlated response per request.
* It can map HTTP responses to AMQP messages in projected mode (section 4.1) or in tunneled mode (section 4.2) or both, and can send the reply to a destination indicated by the request’s reply-to address either via the target container’s anonymous terminus [AMQP-AT], if available, or via a link to the indicated target address.

Since HTTP exchanges can be started over a previously existing connection and from either end of an established connection, the AMQP start (2.1) and HTTP start (2.2) connection initiation are both optional.

1. Acknowledgments

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

**Participants:**

Alan Conway, Red Hat

Rob Godfrey, Red Hat

Keith Wall, Red Hat

Robbie Gemmell, Red Hat

Justin Ross, Red Hat

Ted Ross, Red Hat

Oleksandr Rudyy, JP Morgan  
Xin Chen, Microsoft

Clemens Vasters, Microsoft

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Editor** | **Changes Made** |
| [Rev number] | [Rev Date] | [Modified By] | [Summary of Changes] |