Abstract:
TODO

Status:
This document was last revised or approved by the membership of OASIS 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.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the “Send A Comment” button on the Technical Committee's web page at http://www.oasis-open.org/committees/amqp/.

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 Technical Committee web page (http://www.oasis-open.org/committees/amqp/ipr.php).

Citation format:
When referencing this specification the following citation format should be used:
[amqp-addressing-v1.0]
http://docs.oasis-open.org/amqp/addressing/v1.0/wd01/amqp-addressing-v1.0-wd01.pdf.
Contents

1 Introduction 5
  1.1 Terminology ................................................. 5
  1.2 Normative References ...................................... 5
  1.3 Non-normative References .................................. 6
  1.4 Conformance .................................................. 6
  1.5 Acknowledgements .......................................... 7
  1.6 Revision History ............................................ 8

2 Requirements 9

3 Addresses 10
  3.1 Syntax ....................................................... 10
  3.2 Semantics .................................................... 10
    3.2.1 Registered Domain Resolution .......................... 11
    3.2.2 Example SRV Records .................................. 11

4 Routing 12
  4.1 Routable Messages ........................................... 12
  4.2 Address Patterns .......................................... 12
    4.2.1 Address Pattern ....................................... 12
  4.3 Relay Nodes ................................................ 12
  4.4 Anonymous Terminus ....................................... 13
  4.5 Legacy Addresses ......................................... 13
  4.6 Address Equivalence ...................................... 13
  4.7 Dynamic Routes ............................................ 13
  4.8 Request/Response Examples ................................. 14

5 Appendix I: Scenarios 16
  5.1 Federation .................................................. 16
    5.1.1 Configuration ......................................... 16
    5.1.2 Assumptions ............................................ 16
    5.1.3 Summary ............................................... 16
    5.1.4 Examples ............................................... 16
  5.2 Multilevel Federation ..................................... 16
    5.2.1 Configuration ......................................... 16
    5.2.2 Assumptions ............................................ 17
    5.2.3 Summary ............................................... 17
    5.2.4 Examples ............................................... 17
  5.3 DNS ......................................................... 17
    5.3.1 Configuration ......................................... 17
    5.3.2 Assumptions ............................................ 18
    5.3.3 Summary ............................................... 18
    5.3.4 Examples ............................................... 18
  5.4 Single Broker ............................................... 18
    5.4.1 Configuration ......................................... 18
    5.4.2 Assumptions ............................................ 18
    5.4.3 Summary ............................................... 19
    5.4.4 Examples ............................................... 19
  5.5 Request/Response .......................................... 19
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.5.1 Configuration</td>
<td>19</td>
</tr>
<tr>
<td>5.5.2 Assumptions</td>
<td>19</td>
</tr>
<tr>
<td>5.5.3 Summary</td>
<td>19</td>
</tr>
<tr>
<td>5.5.4 Examples</td>
<td>20</td>
</tr>
</tbody>
</table>
1 Introduction

The Advanced Message Queuing Protocol (AMQP) is an open internet protocol for business messaging. The AMQP Addressing specification defines an addressing syntax and global routing semantics that build on the AMQP Core protocol specification.

1.1 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 IETF RFC 2119 [RFC2119].

The authoritative form of the AMQP Addressing 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 in the excerpts of XML presented in the PDF and HTML representations.

1.2 Normative References

[ASCII]

[IANAHTTPPARAMS]
IANA (Internet Assigned Numbers Authority), Hypertext Transfer Protocol (HTTP) Parameters.
http://www.iana.org/assignments/http-parameters/http-parameters.xml

[IANAPEN]
IANA (Internet Assigned Numbers Authority), Private Enterprise Numbers.
http://www.iana.org/assignments/enterprise-numbers

[IANASUBTAG]
IANA (Internet Assigned Numbers Authority), Language Subtag Registry.
http://www.iana.org/assignments/language-subtag-registry

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

[RFC2234]
http://www.ietf.org/rfc/rfc2234.txt

[RFC3986]
http://www.ietf.org/rfc/rfc3986.txt

[RFC2782]
A. Gulbrandsen, P. Vixie, L. Esibov, A DNS RR for specifying the location of services (DNS SRV). IETF RFC 2782,
February 2005.
http://www.ietf.org/rfc/rfc2782.txt

[RFC6335]
http://www.ietf.org/rfc/rfc6335.txt

1.3 Non-normative References

[AMQPCONNCP]
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

[AMQPFILTERS]
AMQP Capabilities Registry: Filters
http://www.amqp.org/specification/1.0/filters

[AMQPLINKCAP]
AMQP Capabilities Registry: Link Capabilities
http://www.amqp.org/specification/1.0/link-capabilities

[AMQPLINKPROP]
AMQP Capabilities Registry: Link Properties
http://www.amqp.org/specification/1.0/link-properties

[AMQPNODEPROP]
AMQP Capabilities Registry: Node Properties
http://www.amqp.org/specification/1.0/node-properties

[AMQPSESSCAP]
AMQP Capabilities Registry: Session Capabilities
http://www.amqp.org/specification/1.0/session-capabilities

[AMQPSESSPROP]
AMQP Capabilities Registry: Session Properties
http://www.amqp.org/specification/1.0/session-properties

[AMQPSOURCECAP]
AMQP Capabilities Registry: Source Capabilities
http://www.amqp.org/specification/1.0/source-capabilities

[AMQPTARGETCAP]
AMQP Capabilities Registry: Target Capabilities
http://www.amqp.org/specification/1.0/target-capabilities

1.4 Conformance

TODO
1.5 Acknowledgements

The following individuals have contributed significantly towards the creation of this specification and are gratefully acknowledged:

- David Ingham (Microsoft)
- Gordon Sim (Red Hat)
- Rafael Schloming (Red Hat)
- Rob Dolin (Microsoft)
- Robert Godfrey (JPMorgan Chase & Co.)
- Steve Huston (Individual)

The following individuals were members of the OASIS AMQP Technical Committee during the creation of this specification and their contributions are gratefully acknowledged:

- Alex Kritikos (Software AG, Inc.)
- Allan Beck (JPMorgan Chase Bank, N.A.)
- Andreas Moravec (Deutsche Boerse AG)
- Angus Telfer (INETCO Systems Ltd.)
- Chet Ensign (OASIS)
- Dale Moberg (Axway Software)
- David Ingham (Microsoft)
- Gordon Sim (Red Hat)
- Jakub Scholz (Deutsche Boerse AG)
- James Kirkland (Red Hat)
- John O’Hara (Individual)
- Jonathan Poulter (Kaazing)
- Laurie Bryson (JPMorgan Chase Bank, N.A.)
- Matthew Arrott (Individual)
- Oleksandr Rudyy (JPMorgan Chase Bank, N.A.)
- Rafael Schloming (Red Hat)
- Ram Jeyaraman (Microsoft)
- Raphael Cohn (Individual)
- Rob Dolin (Microsoft)
- Rob Godfrey (JPMorgan Chase Bank, N.A.)
- Robert Gemmell (JPMorgan Chase Bank, N.A.)
- SURYANARAYANAN NAGARAJAN (Software AG, Inc.)
- Sandeep Puri (Cisco Systems)
- Sanjay Aiyagari (VMware, Inc.)
- Steve Huston (Individual)
• William Henry (Red Hat)
• Wolf Tombe (US Department of Homeland Security)

1.6 Revision History

2013-05-29 : Working Draft 1
• Initial Draft
2 Requirements

- AMQP addresses support global transcription as defined by RFC 3986 [RFC3986] section 1.2.1.
- AMQP addresses may optionally advertise accessibility via a DNS based authority for scenarios where an AMQP endpoint has well known DNS records.
- AMQP addresses may optionally advertise accessibility via a literal IP/port address for scenarios where an AMQP endpoint has no DNS records. This literal syntax must include the same information available via SRV records including IP, port, and the protocol stack in use, e.g. tls over tcp vs dtls over sctp, vs web sockets.
- AMQP addresses may also be used to uniquely identify global endpoints that do not have any associated DNS records nor a stable IP address.
- The AMQP addressing specification defines the relationship between the to field and the source/target fields such that transparent intermediaries can operate at the message level. This means that messages created by implementations conforming to this specification may be multiplexed from N links down to a single link without losing any information about the ultimate destination of the message. Furthermore it should be possible to demux from a single link back to N links bases solely upon the contents of the to field.
- The AMQP addressing specification defines how link targets may be used to establish dynamic routes to/from a remote source. In particular, this enables a dynamic request/response pattern that does not require creating any intermediate temporary nodes.
3 Addresses

Addresses of the form defined by this specification MAY be used in conjunction with the core specification to send messages across a network of cooperating AMQP intermediaries. These addresses have formal meaning when appearing in any of the address fields defined by the AMQP core specification (TODO xref). Specifically the address field inside source and target types, as well as the to field inside message properties type. The addresses defined by this specification MAY be used by applications within the application defined portions of the message, e.g. the application properties or the message body, however such usage has no formally defined meaning.

3.1 Syntax

The AMQP addressing standard defines a hierarchical path syntax for describing globally routable AMQP addresses. Unresolved elements below are defined by RFC 3986 [RFC3986]:

```
address = "/" domain [ "/" path ]
domain = unregistered-domain
    / registered-domain
registered-domain = "/" reg-name ; reg-name is from RFC3986
    / " srv-literal
srv-literal = [ proto ":" ] host [ ":" port ]
proto = [a-zA-Z_]+
port = *DIGIT
host = IP-literal / IPv4address ; IP-literal and IPv4address are from RFC3986
unregistered-domain = name
path = name [ "/" path ]
name = [^/]*
```

Figure 3.1: BNF

3.2 Semantics

An AMQP global address has two parts: a domain and a path. The path MAY be omitted, however a domain MUST be present to form a valid AMQP address. The path portion of an AMQP address is scoped to its domain. The domain portion of an AMQP address has global scope. This requires that an AMQP domain MUST be unique amongst all the addresses that may appear in any of the set of transitively connected AMQP networks where an address might travel.

An AMQP domain may be either registered or unregistered. An unregistered domain is simply an opaque name. No formally defined authority controls the use of unregistered domains. A registered AMQP domain consists of a DNS name or a literal representation of the service record info that may be obtained from a DNS name. Registered domains make use of DNS infrastructure as a naming authority. Ownership, existence, and collisions of registered domains are managed via standard DNS infrastructure. Unregistered and registered domains are syntactically distinguished by beginning with single vs double slashes respectively, e.g. /unregistered-name/path vs //registered-name/path.
3.2.1 Registered Domain Resolution

A registered domain MAY be resolved via DNS to a protocol, an ip address, and port number. The currently defined protocols are “tcp” and “tls”. (TODO: define what each protocol means) The default protocol is defined to be “tcp”, and the default port number is 5672. (TODO: should we define a protocol and/or port number for either?)

A registered domain consists of either a registered name or an srv-literal. A registered name is resolved by first querying any SRV records for that name in accordance with RFC 2782 [RFC2782] as updated by RFC 6335 [RFC6335]. If no SRV records exist, then any A or AAAA records are used to obtain an ip address, and the default protocol and port are assumed.

If the registered domain is an srv-literal, the protocol, ip address, and port are taken directly from their literal form inside the domain portion of the address.

Note that whilst a non literal registered domain is expected to be registered with DNS, there is no requirement that it have either SRV, A, or AAAA, records, therefore it may not be possible to resolve a given registered domain to a protocol, ip address and port number.

3.2.2 Example SRV Records

- _amqp._tcp.example.net. 86400 IN SRV 5 0 5672 amqp1.example.net.
- _amqp._tcp.example.net. 86400 IN SRV 5 0 5672 amqp2.example.net.
- _amqps._tcp.example.net 86400 IN SRV 4 0 5671 amqp3.example.net.
- _amqps._tcp.example.net 86400 IN SRV 4 0 5671 amqp4.example.net.
- _amqp._sctp.example.net. 86400 IN SRV 5 0 5672 amqp.example.net.
- _amqps._sctp.example.net. 86400 IN SRV 5 0 5671 amqp.example.net.
4 Routing

4.1 Routable Messages

A routable message MUST include the entire valid AMQP address in the to field of the message. If the to address is not specified, it MUST be considered equivalent to the target address specified on the link. If the to address is present and different from the target address, then the behavior is defined by the target node.

4.2 Address Patterns

This specification defines a simple pattern matching syntax for describing classes of addresses.

4.2.1 Address Pattern

```
<type name="address-pattern" class="restricted" source="string">
  <descriptor name="amqp:address-pattern:string" code="0x00000000:0x000000??"/>
</type>
```

Every valid address is also a pattern that matches the address exactly. Additionally a pattern may include the wildcard characters '%' and '*' which are not allowed in valid AMQP addresses. The '%' wildcard matches any valid address character with the exception of '/'. The '*' wildcard matches all valid address characters with no exceptions.

- **example** Matches "example" and no others.
- **example/** Matches any path prefixed with "example/" regardless of length, e.g. "example/", "example/foo", "example/foo/bar", "example/foo/bar/baz". Does NOT match "example", "foo/example", or "foo/bar/example"
- **example/*/foo** Matches any path starting with "example/" and ending with "/foo", e.g. "example/bar/foo". Does NOT match "example/foo" or "example/foo/bar".
- **example/%/bar** Matches any path starting with example, ending with bar and having exactly one intermediate component, e.g. "example/foo/bar", "example/bar", "example/baz/bar". Does NOT match "example/bar" or "example/foo/baz/bar".

TODO: define a real descriptor here

4.3 Relay Nodes

A *relay node* acts as a proxy for links to other nodes. Messages sent over links into a relay node will be relayed to the node referenced in the to field of the message just as if a direct link has been established to that node. Links coming out of a relay node will relay messages from all nodes identified by the address pattern(s) in the source node filter, just as if a direct links have been established from those nodes.
4.4 Anonymous Terminus

A source or target with a null address is referred to as an anonymous terminus. The anonymous terminus is reserved for use as a relay node by containers that support such capabilities. A container MAY advertise anonymous relay capability by use of the ANONYMOUS-RELAY connection capability. (TODO: how do we formally define capabilities?) Containers advertising the ANONYMOUS-RELAY capability MUST resolve the anonymous terminus to a relay node as defined above.

4.5 Legacy Addresses

Address strings appearing in protocol fields with a value that does not start with “/” are considered legacy addresses. These are assumed to be scoped to a single container. There is no way to know in general to which container they are scoped. Implementations MAY assume that these addresses are scoped to the container that produces them.

4.6 Address Equivalence

Containers participating in non local namespaces that also support legacy addresses MUST recognize the following forms as equivalent where <domain> represents the name of the container in the non local namespace:

- /<domain>/<path>
- <path>

Containers MAY participate in multiple non local namespaces and as such recognize even more equivalent forms for a given address.

4.7 Dynamic Routes

Routing information, i.e. the next hop for a message destined to a given address, can be propagated dynamically through a network of AMQP containers as a result of link establishment. This occurs when a container specifies an address for the local terminus.

```
[-----------------------------]
src=ADDR1, tgt=ADDR2
[-----------------------------]
```

Container A establishes an outgoing link to Container B, making Container B aware of the existence of ADDR1.

Figure 4.1: Outgoing Example
Container A establishes an incoming link to Container B, making Container B aware of the existence of ADDR1.

Figure 4.2: Incoming Example

By specifying an address for the local terminus, a container can communicate both the existence of the address and a route for messages addressed there. For this to be useful, the node and/or container associated with the remote terminus needs to be capable of interpreting and acting upon the specified address, which from its perspective will be the address of the remote terminus.

The DYNAMIC-ROUTER capability is defined to identify nodes and containers that can interpret and route based on the address specified in an authorized remote terminus.

A node advertising the DYNAMIC-ROUTER capability MUST route incoming messages to matching and authorized remote terminii.

A container advertising the DYNAMIC-ROUTER capability MUST allow authorized links to/from the addresses present in authorized remote terminii.

4.8 Request/Response Examples

Client Service

```
ATTACH(name=RES, handle=1, role=receiver, source=SOURCE(address="CLIENT123"), target=TARGET(address="SERVICE", capabilities=[DYNAMIC-ROUTER]))
```

```
ATTACH(name=REQ, handle=2, role=sender, source=SOURCE(address="CLIENT123"), target=TARGET(address="SERVICE"))
```

```
TRANSFER(handle=2, ... request ...) -------------------
```

```
<-------------------------- TRANSFER(handle=1, ... response ...)
```

Figure 4.3: Client Initiates Response Link
Figure 4.4: Service Initiates Response Link
Appendix I: Scenarios

5.1 Federation

5.1.1 Configuration

Several brokers from multiple distinct vendors participating in a single federation.

5.1.2 Assumptions

The federation shares a common namespace for addresses. Participating brokers are assigned a distinct domain within the shared namespace.

Each broker has a means to recognize and differentiate between local and non local addresses. Local addresses reference the domain assigned to that broker, and non local addresses reference a non local domain. Furthermore, each broker may have a means to proxy messages and subscription requests to/from non local addresses, thereby functioning as a gateway for some or all of the other brokers participating in the federation. The mechanism for accomplishing and/or configuring this proxying is not specified.

5.1.3 Summary

A standard syntax for differentiating between local and non local portions of an address provides a convenient place to tie in vendor specific configuration mechanisms, thereby encouraging a consistent experience for clients accessing the federated network through gateways provided by distinct vendors.

5.1.4 Examples

- /broker
- /brokerA/queue
- /brokerA/queue/subscription
- /brokerB/topic
- /brokerB/topic/sub-topic
- /brokerB/topic/sub-topic/sub-sub-topic
- /brokerC/service

5.2 Multilevel Federation

5.2.1 Configuration

Messages traveling through a network consisting of multiple federations in different administrative domains.
5.2.2 Assumptions

The multilevel federation shares a common namespace for addresses. Participating federations are assigned a distinct domain within the shared namespace, and each broker in a federation is assigned a distinct name within its federation.

Each broker has a means to recognize and differentiate between broker local, federation local, and non local addresses. Broker local addresses reference both the federation and the name assigned to that broker, federation local addresses reference the federation, but not the local broker, and non local addresses reference an external federation.

Brokers may have configuration that allows proxying messages and/or subscription requests to/from non broker local addresses, thereby functioning as a gateway for some or all of the other brokers participating in the overall federation. The mechanism for accomplishing and/or configuring this proxying is not specified.

5.2.3 Summary

A standard hierarchical syntax allowing the structure of a multilevel federation to be reflected in the address syntax provides a convenient place to tie in vendor specific configuration mechanisms, thereby encouraging a consistent experience for clients accessing the federated network through gateways provided by distinct vendors.

5.2.4 Examples

- /departmentA
- /departmentA/broker
- /departmentA/broker1/queue
- /departmentA/broker1/queue/subscription
- /departmentA/broker2/topic
- /departmentA/broker2/topic/sub-topic
- /departmentA/broker2/topic/sub-topic/sub-sub-topic
- /departmentA/broker3/service
- /departmentB/broker
- /departmentB/broker1/queue
- /departmentB/broker1/queue/subscription
- /departmentB/broker2/topic
- /departmentB/broker2/topic/sub-topic
- /departmentB/broker2/topic/sub-topic/sub-sub-topic
- /departmentB/broker3/service

5.3 DNS

5.3.1 Configuration

A large scale loosely coupled federation of brokers, AMQP services, and AMQP-aware transparent intermediaries.
5.3.2 Assumptions

The number of participating elements in this federation is large enough to make it impractical to manually configure all routes between participants.

Participants in this federation leverage kerberos, SSL certs, claims based auth, or some other authentication scheme such that pairwise configuration of each participant is not necessary in order to achieve mutual authentication.

5.3.3 Summary

For such a deployment it is valuable to be able to leverage existing DNS infrastructure for both provisioning namespaces within the federation, and for establishing routes between the more loosely coupled portions of the federation. The addressing syntax can enable this kind of deployment by providing a clear way to identify when the root name is actually a DNS name that can be used to look up an SRV, A, or AAAA record for use in establishing a route.

5.3.4 Examples

- //example.com
- //example.com/queue
- //example.com/queue/subscription
- //example.com/topic
- //example.com/topic/sub-topic
- //example.com/topic/sub-topic/sub-sub-topic
- //example.com/service
- //amqp.example.com
- //amqp.example.com/queue
- //subdomain.amqp.example.com
- //subdomain.amqp.example.com/queue
- //example.com/departmentA/broker2/queue

5.4 Single Broker

5.4.1 Configuration

A single broker that is only aware of its own namespace.

5.4.2 Assumptions

Such a broker may directly contain entities with simple names such as queues, topics, or other nodes, and may also define their own syntax to expose structure within their own namespace.
5.4.3 Summary

Brokers implemented and deployed before this addressing standard may not recognize the address syntax or semantics defined herein. It is therefore desirable to minimize the chance that non local addresses will conflict with a single broker using addresses to directly reference entities within its own namespace.

5.4.4 Examples

- queue
- queue.subscription
- topic
- topic.sub-topic
- topic.sub-topic.sub-sub-topic
- topic.sub-topic
- topic.sub-topic.sub-sub-topic
- topic.sub-topic.sub-sub-topic
- topic::sub-topic::sub-sub-topic
- topic://name
- queue://name
- service

5.5 Request/Response

5.5.1 Configuration

A request/response style service built on top of the AMQP protocol.

5.5.2 Assumptions

Clients may access the request/response service through a variety of topological configurations, including:

- A direct connection with no intervening intermediaries.
- A single traditional broker.
- A federation of brokers.
- One or more transparent intermediaries

The service itself does not act as a general purpose broker. It is not capable of creating temporary queues.

5.5.3 Summary

A request/response scenario would benefit greatly from a way to route responses that does not depend on the topology used to access the service. It is undesirable to force the use of an intermediary to host a temporary queue, and likewise undesirable to force the choice of a specific location for the temporary queue in a federated...
scenario. As such it is useful for the endpoints themselves to participate directly as addressible entities in the overall network topology.

5.5.4 Examples

- /service
- /broker/service
- //example.com/service
- /client-1234
- /client-4321/request-27