Advanced Message Queuing Protocol (AMQP) Claims-based Security Version 1.0

Working Draft 01

08 July 2013

Technical Committee:

OASIS Advanced Message Queuing Protocol (AMQP) TC

Chairs:

Ram Jeyaraman (Ram.Jeyaraman@microsoft.com), Microsoft
Robert Godfrey (robert.godfrey@jpmorgan.com), JPMorgan Chase & Co.

Editors:

TODO: update this
Rob Dolin (RobDolin@microsoft.com), Microsoft
Robert Godfrey (robert.godfrey@jpmorgan.com), JPMorgan Chase & Co.
David Ingham (David.Ingham@microsoft.com), Microsoft
Rafael Schloming (rafaels@redhat.com), Red Hat

Additional artifacts:

This prose specification is one component of a Work Product which also includes:

- XML schemas: (list file names or directory name)
- Other parts (list titles and/or file names)

Related work:

This specification is related to:

- *OASIS Advanced Message Queuing Protocol (AMQP) Version 1.0 Part 0: Overview*. 29 October 2012. OASIS Standard. http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-overview-v1.0-os.html.

Abstract:

This specification describes an AMQP authentication scheme based on claims-based security tokens.

Status:

This Working Draft (WD) has been produced by one or more TC Members; it has not yet been voted on by the TC or approved as a Committee Draft (Committee Specification Draft or a Committee Note Draft). The OASIS document Approval Process 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.

Initial URI pattern:

http://docs.oasis-open.org/amqp/amqp-cbs/v1.0/csd01/amqp-cbs-v1.0-csd01.doc

(Managed by OASIS TC Administration; please don't modify.)

Table of Contents

1 Introduction

This specification defines a claims-based security (CBS) extension for AMQP 1.0 **[AMQP]**. The goals for this extension are:

1. To support fine-grained claims-based access control to entities accessible over an AMQP connection.
2. To work with existing AMQP client libraries without change.

To satisfy these goals, a layered protocol is defined to exchange claims tokens over an AMQP connection. This protocol is based on the AMQP Management Specification **[AMQPMAN]** and involves the use of a dedicated link to a special claims-based security node, over which tokens are transferred as standard AMQP messages with a well-defined structure.

A non-goal for this specification is the runtime negotiation and configuration of CBS for a particular connection. It is assumed that applications will be configured out-of-band with the knowledge as to when claims-based security is to be used and what options are supported, e.g., which claim token type is to be used.

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”, October 2012. OASIS Standard. https://www.oasis-open.org/standards#amqpv1.0
[AMQPMAN]	Godfrey, Robert; Ingham, David; Dolin, Rob, “Advanced Message Queuing Protocol (AMQP) Management Version 1.0”, January 9999. OASIS Working Draft.
[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.
[RFC2616]	Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., "Hypertext Transfer Protocol -- HTTP/1.1", RFC2616, June 1999. http://www.w3.org/Protocols/rfc2616/rfc2616.html.
[RFC4422]	Melnikov, A., and Zeilenga, K., “Simple Authentication and Security Layer (SASL)", RFC4422, June 2006. http://tools.ietf.org/html/rfc4422.

1.3 Non-Normative References

[Reference]	[Full reference citation]

// TODO: Add references for:

* SASL PLAIN

* OAuth 2.0

2 Concepts

2.1 Token

A token is opaque data that may be used to authenticate a user and/or authorize the user of a computer system. The tokens described in this specification are bearer tokens; meaning the bearer of the token is treated as the authenticated user.

2.1.1 Token Expiry

Tokens have an expiry time or a time to live (TTL) which limits the usefulness of a token if it is ever compromised.

2.1.2 Token Type

Tokens have a type so the receiving system knows how to handle the token. In this specification, token types are represented as strings that follow the same convention for types defined in the core AMQP specification. Standard token types considered at the time of writing have names prefixed with “amqp:”, e.g., the type of a JSON Web Token is represented as “amqp:jwt”. Proprietary token types should be named using a reverse domain name prefix, e.g., the type of Microsoft Shared Access Signatures could be represented as “com.microsoft:sas”.

2.1.3 Token Value

Tokens have a specific value that contains the information used by the receiving system to authenticate the user and authorize access to the target resource(s). Tokens generally have a native string representation. This specification caters for token type-specific formats.

2.2 Claims-based Security Node

Each AMQP container MUST provide a Claims-Based Security Node (the CBS Node) with the address “`$cbs`”.

3 Overview

TODO: turn this in to prose:

- AMQP connection is established normally.
- SASL scheme is essentially orthogonal but commonly CBS will be used with External.
- Dedicated link is create to CBS Node, along with response link in the manner that should be described in the global addressing spec.
- Links to application endpoints are established normally as required.
- Claim tokens are “put” over the CBS request link before attempting to “access” the associated secured entity.

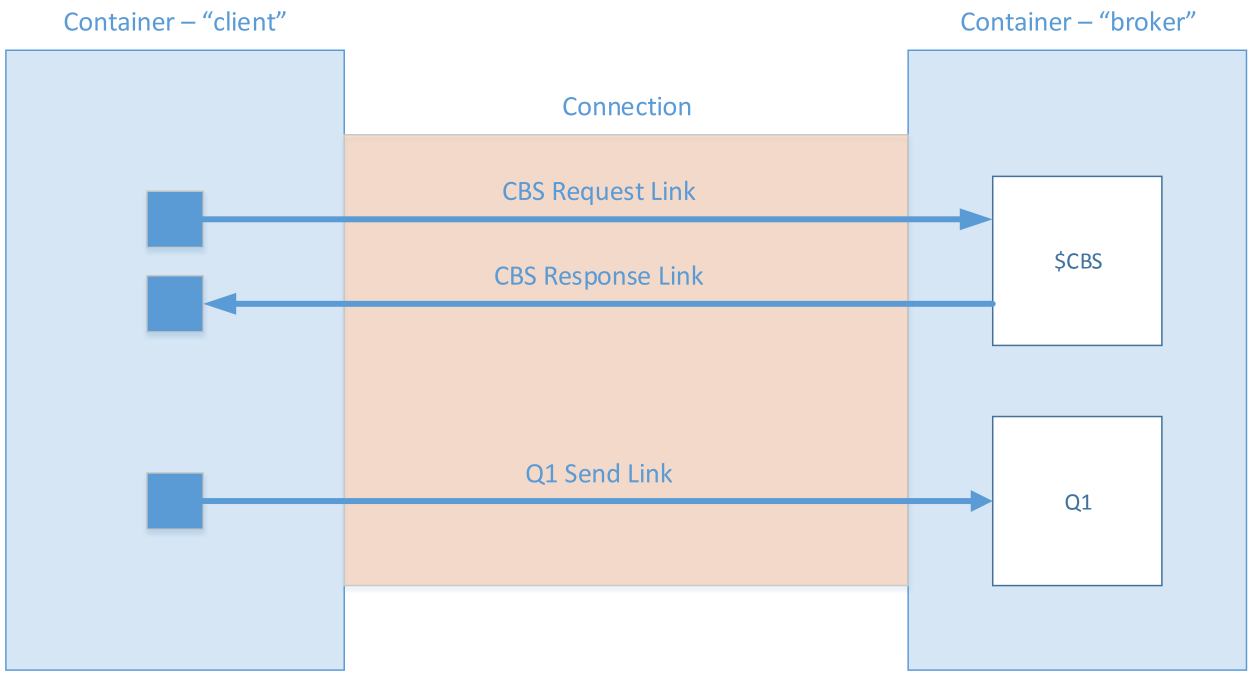

Figure 1: Overview

Note that a condition in which the token was not found should be treated as success.

3.1 Scenarios

3.1.1 Link-based

In this scenario, a single link is being used to exchange messages with a single endpoint and access to this particular endpoint is controlled by claims-based security. In order to be able to exchange messages over a link to that endpoint, an appropriate valid claim is required to be in place.

For example, a message broker hosting a queue with address “q1”, could require a “read” claim in order to receive messages and a “write” claim to send messages. In this example, a client application would need to put a token, containing the appropriate claim, to the CBS Node in advance of establishing the link to “q1”. Periodically, before the token expired, the client would need to send a refreshed token in order to be able to continue to exchange messages.

3.1.2 Message-based

In this scenario, a single link is being used to exchange messages with multiple endpoints. This is sometime referred to as a relayed scenario, in which a client establishes a single link to a relay endpoint over which messages can be exchanged for several endpoints.

For example, consider a message broker hosting queues with addresses “q1” and “q2” and a relay endpoint with address “relay”. To send messages to queues “q1” and “q2”, a client establishes a link with a target address of “relay” and uses the “to” property of messages to specify the desired final address, “q1” or “q2”. This is sometimes referred to as the “anonymous publisher” model. In this example, the broker may require “write” claims for the “relay” as well as for the final destination queues in order to accept a message from the client. Conversely, the broker may be securing just the relay or just the final destination queues. It is assumed that the client is aware of what claims are required through some out-of-band configuration.

In this example, if the relay is being secured then the client application would need to put the token, containing the appropriate claim, to the CBS Node in advance of establishing the link to “relay”. Periodically, before the token expired, the client would need to send a refreshed token in order to be able to continue to exchange messages with the relay. In addition, the client application would need to put appropriate tokens for each target endpoint referenced in the “to” addresses of messages sent via the relay in advance of sending a message.

4 Communicating Tokens

Tokens are communicated between AMQP peers by sending specially-formatted AMQP messages to the Claims-based Security Node. The mechanism follows the scheme defined in the AMQP Management specification **[AMQPMAN]**.

4.1 Putting a Token

A token is sent to the CBS Node by transferring a “put-token” message.

TODO: any more details required regarding the transfer of the put-token message? For example, should we specify whether it’s sent pre-settled?

4.1.1 Request Message

The request message has the following application-properties:

Key	**Optional**	**Value Type**	**Value Contents**
operation	No	string	“`put-token`”
Type	No	string	The type of the token being put, e.g., “amqp:jwt”.
name	No	string	The “audience” to which the token applies.
expiration	Yes	timestamp	The expiry time of the token.

The body of the message MUST contain the token. The type of the body is dependent on the type of token being put. The table below lists the body types for common token types:

Token Type	**Token Description**	**Body Type**
amqp:jwt	JSON Web Token (JWT)	AMQP Value (string)
amqp:swt	Simple Web Token (SWT)	AMQP Value (string)
com.microsoft:sas	Microsoft Service Bus Shared Access Signature (SAS) Token	AMQP Value (string)

4.1.2 Response Message

The response message has the following application-properties:

Key	**Optional**	**Value Type**	**Value Contents**
status-code	No	int	HTTP response code **[RFC2616]**.
status-description	Yes	string	Description of the status.

The body of the message MUST be empty.

If the request was successful then the status-code MUST contain 200.

If the request was unsuccessful due to a processing error then the status-code SHOULD contain 500 and further information MAY be provided in the status-description.

For error conditions related to the content of the request, e.g., unsupported token type, malformed request etc., the status-code SHOULD contain 400 and a detailed description SHOULD NOT be provided in the status-description, in line with general best practice for security-related protocols.

4.2 Deleting a Token

To instruct a peer to delete a token associated with a specific audience, a "delete-token" message can be sent to the CBS Node

4.2.1 Request Message

The request message has the following application-properties:

Key	**Mandatory**	**Value Type**	**Value Contents**
operation	Yes	string	"`delete-token`"
Type	Yes	string	The type of the token being deleted, e.g., "amqp:jwt".
name	Yes	string	The "audience" of the token being deleted.

The body of the message MUST be empty.

4.2.2 Response Message

The response message has the following application-properties:

Key	**Mandatory**	**Value Type**	**Value Contents**
status-code	Yes	int	HTTP response code **[RFC2616]**.
status-description	No	string	Description of the status.

The body of the message MUST be empty.

If the request was successful then the status-code MUST contain 200.

If the request was unsuccessful due to a processing error then the status-code SHOULD contain 500 and further information MAY be provided in the status-description.

For error conditions related to the content of the request, the status-code SHOULD contain 400 and a detailed description SHOULD NOT be provided in the status-description, in line with general best practice for security-related protocols.

5 Error Cases

// TODO: Add error cases

6 Examples

// TODO: Add Examples

7 # Conformance

The last numbered section in the specification must be the Conformance section. Conformance Statements/Clauses go here. [Remove # marker]

Appendix A. Acknowledgments

TODO: update this before we ship.

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

Participants:

Rob Dolin, Microsoft

Robert Godfrey, JP Morgan

David Ingham, Microsoft

The following individuals were members of the OASIS Advanced Message Queueing Protocol (AMQP) Technical Committee during the creation of this specification and their contributions are gratefully acknowledged:

Sanjay Aiyagari, VMware, Inc.

Matthew Arrott, Individual

Allan Beck, JPMorgan Chase Bank, N.A.

Laurie Bryson, JPMorgan Chase Bank, N.A.

Raphael Cohn, Individual

Rob Dolin, Microsoft

Robert Gemmell, JPMorgan Chase Bank, N.A.

Rob Godfrey, JPMorgan Chase Bank, N.A.

William Henry, Red Hat

Steve Huston, Individual

David Ingham, Microsoft

Ram Jeyaraman, Microsoft

James Kirkland, Red Hat

Alex Kritikos, Software AG, Inc.

Dale Moberg, Axway Software

Andreas Moravec, Deutsche Boerse AG

Suryanarayanan Nagarajan, Software AG, Inc.

John O'Hara, Individual

Jonathan Poulter, Kaazing

Sandeep Puri, Cisco Systems

Oleksandr Rudyy, JPMorgan Chase Bank, N.A.

Rafael Schloming, Red Hat

Jakub Scholz, Deutsche Boerse AG

Angus Telfer, INETCO Systems Ltd.

Wolf Tombe, US Department of Homeland Security

Appendix B. Revision History

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