RADEXT Working Group
Internet-Draft
Updates: 8559 (if approved)
Intended status: Standards Track
Expires: 18 February 2026

A. DeKok
InkBridge
V. Cargatser
Cisco

17 August 2025

Reverse Change-of-Authorization (CoA) in RADIUS/(D)TLS
draft-ietf-radext-reverse-coa-07

Abstract

This document defines a "reverse Change-of-Authorization (CoA)" path for RADIUS packets. A TLS connection is normally used to forward request packets from a client to a server and to send responses from the server to the client. This specification allows a server to send CoA request packets to the client in "reverse" down that connection, and for the client to send responses to the server. Without this capability, it is in general impossible for a server to send CoA packets to a Network Access Server (NAS) that is located behind a firewall or NAT. This reverse CoA functionality extends the available transport methods for CoA packets, but it does not change anything else about how CoA packets are handled.

This document updates RFC8559.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-radext-reverse-coa/.

Discussion of this document takes place on the RADEXT Working Group mailing list (mailto:radext@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/radext/. Subscribe at https://www.ietf.org/mailman/listinfo/radext/.

Source for this draft and an issue tracker can be found at https://github.com//radext-wg/draft-ietf-radext-reverse-coa.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

This Internet-Draft will expire on 18 February 2026.

Copyright Notice

Table of Contents

1. Introduction

The Remote Authentication Dial-In User Service (RADIUS) protocol [RFC2865] is a client-server protocol where clients send requests to servers, and servers send responses to clients. RADIUS was extended in [RFC5176] to define the ability to change a user's authorization, or disconnect the user via what are generally called "Change-of-Authorization" or "CoA" packets. In this use-case, a server which normally receives authentication requests from a client instead sends CoA requests to that client.

When that inversion of roles takes place, the system sending the CoA requests is acting as a client, and the system receiving those requests is acting as a server. In order to more clearly separate these roles, all connections between RADIUS clients and servers have historically been defined to be one way. A client sends requests to a server, on a port which is dedicated to that role. For RADIUS, there have been separate ports defined for authentication requests, accounting requests, and CoA requests.

The initial transport protocol for all RADIUS messages was the User Datagram Protocol (UDP). [RFC6614] then updated RADIUS to allow packets to be sent over the Transport Layer Security (TLS) protocol. The update also removed the requirement that each type of packet use a dedicated port. Instead, all packets (including CoA) can be be sent over a TLS connection, as discussed in [RFC6614], Section 2.5:

> Due to the use of one single TCP port for all packet types, it is required that a RADIUS/TLS server signal which types of packets are supported on a server to a connecting peer. See also Section 3.4 for a discussion of signaling.

That specification, however, still required that the systems still act as client and server. The client connects to the server, and sends only requests. The server waits for client connections, and only sends responses. This flow of packets is referred to as the "forward" path.

The limitation of this design is that it assumes that a RADIUS client can always contact a RADIUS server. When a RADIUS server wishes to send CoA packets to a RADIUS client, it must initiate a new

connection "reverse" path to that client. Any existing TLS connection from that client is ignored, and is not used. Even worse, the "reverse" path can be blocked by an on-path stateful function (e.g. firewall, NAT, etc.).

The design of RADIUS requires that a client must be able to reach a server. But the reverse path from server to client for CoA is only usable when both client and server share a common and open network. In the past, many organisations which supported roaming did so via dedicated interconnections using IPSec. This design allowed the connected parties to have a route for sending CoA packets, but the direct interconnections could be expensive to set up and maintain. As such, a common practice is now to use RADIUS/(D)TLS.

However, there is often a firewall, NAT, etc. between a client and server which blocks the reverse path for RADIUS/UDP or RADIUS/(D)TLS. This scenario is most evident in a roaming / federated environment such as eduroam ([RFC7593] and [EDUROAM]) or OpenRoaming ([OPENROAMING]). Even though [RFC8559] defines CoA proxying, that specification does not address the issue of NAS reachability. In many roaming environments, there is no direct reverse path from the server to the NAS, as the NAS is not accessible from the Internet. Even if there was a public reverse path, the chain of proxies effectively hides the location of the NAS. Intermediate proxies can (and do) rewrite packet contents to hide NAS identities. It is therefore in many cases impossible for a server to send a request to the NAS.

These limitations can result in business losses and security problems, such as the inability to disconnect a user when their account has been terminated.

1.1. The Solution

This specification solves that problem. The solution is to simply allow CoA packets to go in "reverse" down an existing RADIUS/(D)TLS connection. That is, when a NAS connects to a RADIUS server it normally sends request packets (Access-Request, etc.) and expects to receive response packets (Access-Accept, etc.). This specification extends RADIUS/(D)TLS by permitting a RADIUS server to re-use an existing TLS connection to send CoA packets to the NAS, and permitting the NAS to send CoA response packets to the RADIUS server over that same connection.

While this document specifically mentions RADIUS/(D)TLS, it should be possible to use the same mechanisms on RADIUS/DTLS [RFC7360]. However at the time of writing this specification, no implementations exist for "reverse CoA" over RADIUS/DTLS.

This mechanism does not depend on the underlying transport protocol, or interact with it. It is therefore compatible not only with [RFC6614], and [RFC7360], but also with [I-D.ietf-radext-radiusdtls-bis] which will replace those earlier standards.

1.2. Limitations

This mechanism is not applicable for RADIUS/UDP, as [RFC5176] and [RFC8559] are sufficient for CoA for the cases where the client and server can communicate directly.

When the client and server cannot communicate directly, such as when a they are separated by a firewall or NAT, the nature of UDP makes it impossible to support reverse CoA. Since UDP is connection-less, the server has no way of knowing whether or not the client is still receiving packets on a port. A client may open a port, send a request, and then immediately close the port after receiving a response. Alternatively, there could be a NAT between the client and server. The NAT could time out its UDP state tracking, again with no indication to the server. Therefore, any attempt by a server to use a reverse path for UDP is unlikely to work reliably.

RADIUS/DTLS has similar issues to RADIUS/UDP with respect to NATs. However, there is an underlying TLS session associated with a particular client to server connection. So long as the TLS connection is functional, it can be used to send reverse CoA packets. Where the TLS connection is not functional, no traffic will pass in either direction.

This mechanism is also not suitable for RADIUS/TCP. While it could theoretically be used there, RADIUS/TCP is being deprecated by [I-D.ietf-radext-deprecating-radius]. As such, RADIUS/TCP is unsuitable as a transport mechanism, and no reverse CoA functionality is defined for it.

For the above reasons, therefore, the "reverse CoA" functionality is limited to RADIUS/(D)TLS.

1.3. Chains of Proxies

There are some additional considerations which need to be addressed in order for this specification to work across multiple proxies. While [RFC8559] describes CoA proxying, this specification describes how those systems can implement "reverse CoA" proxying, including processing packets through both an intermediate proxy network, and at any visited network which is not able to directly authenticate the user.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Please see [RFC5176], Section 1.3 for the base terminology that is associated with Change-of-Authorization.

* CoA

  Change-of-Authorization packets. For brevity, when this document refers to "CoA" packets, it means either or both of the CoA-Request [RFC5176], Section 2.2 and Disconnect-Request [RFC5176], Section 2.1 packets.

* ACK

  Change-of-Authorization "positive acknowledgment" packets. For brevity, when this document refers to "ACK" packets, it means either or both of CoA-ACK and Disconnect-ACK packets.

* NAK

  Change-of-Authorization "negative Acknowledgements" packets. For brevity, when this document refers to "NAK" packets, it means either or both of CoA-NAK and Disconnect-NAK packets.

* RADIUS/TLS

  RADIUS over the Transport Layer Security protocol [RFC6614]

* RADIUS/DTLS

  RADIUS over the Datagram Transport Layer Security protocol [RFC7360]

* RADIUS/(D)TLS

  RADIUS over either DTLS or TLS.

* (D)TLS

  Either RADIUS/TLS or RADIUS/DTLS.

* reverse CoA

CoA, ACK, or NAK packets sent over a RADIUS/(D)TLS connection which was made from a RADIUS client to a RADIUS server.

3. Concepts

The reverse CoA functionality is based on two additions to RADIUS. The first addition is a configuration and signaling method, to indicate that a RADIUS client is capable of accepting reverse CoA packets, which is discussed below in Section 4. The second addition is an extension to the "reverse" routing table for CoA packets that was first described in Section 2.1 of [RFC8559]. This addition is discussed below in Section 6.

4. Capability Configuration and Signaling

In order for a RADIUS server to send reverse CoA packets to a client, it must first know that the client is capable of accepting these packets.

Clients and servers implementing reverse CoA MUST have a configuration flag which indicates that the other party supports the reverse CoA functionality. That is, the client has a per-server flag enabling (or not) reverse CoA functionality. The server has a similar per-client flag.

The flag can be used where the parties are known to each other. The flag can also be used in conjunction with dynamic discovery ([RFC7585]), so long as the server associates the flag with the client identity and not with any particular IP address. That is, the flag can be associated with any method of identifying a particular client such as TLS PSK identity, information in a client certificate, etc.

The configuration flag allows administrators to statically enable this functionality, based on out-of-band discussions with other administrators. This process is best used in an environment where all RADIUS proxies are known (or required) to have a particular set of functionality, as with a roaming consortium.

This specification does not define a way for clients and servers to negotiate this functionality on a per-connection basis. The RADIUS protocol has little, if any, provisions for capability negotiations, and this specification is not the place to add that functionality.

Without notification, however, it is possible for clients and servers to have mismatched configurations. Where a client is configured to accept reverse CoA packets and the server is not configured to send them, no packets will be sent. Where a client is configured to not

accept reverse CoA packets and the server is configured to send them, the client will silently discard these packets as per [RFC2865], Section 3. In both of those situations, the reverse CoA functionality will not be available. There may be security issues when it is not possible to disconnect users, or to change their authorization. See Section 9 for some additional comments on this topic.

Any matched configuration for reverse CoA is therefore identical to the situation before reverse CoA was defined. The relevant issues were discussed above in Section 1.

4.1. Proxies

As there is no RADIUS routing protocol, the administrator of a proxy is responsible for manually validating routing of packets across multiple proxies.

5. Connection Management

This specification relies entirely on a client making a (D)TLS connection to a server. Where the client does not make a connection to the server, or where the client quickly closes connections, the reverse CoA functionality will be less useful.

Where a particular client and server combination is determined to support this functionality, the server may need to send reverse CoA packets at any time. A client which does not send watchdog packets may have its connection state discarded by a firewall or NAT. As such, the client SHOULD maintain at least connection open to the server at all times.

The application watchdog mechanism defined in [RFC3539], Section 3.4 SHOULD be used to maintain the connection. The watchdog packet SHOULD be Status-Server, as defined in [RFC5997].

The watchdog timer (Tw) which is defined in [RFC3539], Section 3.4.1 SHOULD be initialized to 15 seconds, instead of the default value of 30 seconds. A value of 30 seconds is likely to be high enough that intermediate nodes may discard connection state. A value of 15 seconds is much less likely to result in the connection state being discarded.

5.1. Dynamic Discovery

When [RFC7585] dynamic discovery is used, systems which need to send CoA packets to a destination can use the "aaa+dynauth" lookup that is defined in [RFC7585], Section 2.1.1.1. That process allows for systems to make (D)TLS connections directly to a destination.

The reverse CoA functionality defined here is therefore useful, but is not strictly necessary when dynamic discovery is used. However, problems arise when system needing to receive CoA packets chooses to not implement dynamic discovery for them, but instead to rely solely on the reverse CoA functionality

Without dynamic discovery, the system necessarily relies instead on the reverse CoA functionality which uses the connections that it makes to servers. If at any time the system drops its connections to a server, the server has no way to send the CoA packets.

As such, it is RECOMMENDED that systems which use [RFC7585] for Access-Request and/or Accounting-Request packets also use the same method for CoA-Request and Disconnect-Request packets.

6. Reverse Routing across Proxies

In normal RADIUS proxying. the forward routing table uses the User-Name attribute (via the Network Access Identifiers (NAIs) [RFC7542]) to map realms to next hop servers. For reverse CoA, [RFC8559], Section 2.1 uses the Operator-Name attribute to map a realm to one or more next hop servers.

This specification extends the [RFC8559], Section 2.1 reverse routing table to allow the next hop to be found via an open (D)TLS connection, rather than a destination hostname or IP address. A server which needs to send reverse CoA packets to clients maintains a list of open (D)TLS connections from clients. It also associates both a reverse CoA capability, and one or more realm with each connection.

A server MUST allow one realm to be associated with multiple connections. A server MUST allow multiple realms to be associated with one connection. That is, the "realm to connection" mapping is not one-to-one, or 1:N, or M:1, it is N:M, i.e. many-to-many.

This process occurs for all on-path RADIUS proxies, except for the final one which sends the CoA packet to the client. That proxy forwards the reverse CoA packet to the client based on the Operator-NAS-Identifier attribute ([RFC8559], Section 3.4) and/or other NAS identification attributes such as NAS-Identifier, NAS-IP-Address, or

NAS-IPv6-Address. The result is that there is a complete forwarding path from the home network which authenticates the user, back to the visited network where the user is currently located.

6.1. Errors and Fail Over

This specification extends [RFC8559], Section 3.2 to handle the situation where the destinaion realm is known, but where there is no connection over which the request can be routed.

When a receives an unexpected reverse CoA packet over a connection, it MUST be silently discarded as per [RFC2865], Section 3. A server SHOULD log a descriptive message about this error. This behavior is unchanged from prior specifications.

When a server supports reverse CoA, and receives a reverse CoA packet which cannot be forwarded, the server MUST return a NAK packet that contains an Error-Cause Attribute having value 502 ("Request Not Routable"). The server SHOULD also log a descriptive message about this error. Logging errors helps an administrator discover and correct any deficiencies in the server configuration, or in its interaction with other systems.

As with normal proxying, a particular packet can sometimes have the choice of more than one connection which can be used to reach a destination. In that case, issues of load-balancing, fail-over, etc. are implementation-defined, and are not discussed here. For the purpose of this specification, when a server needs to send a reverse CoA connection to a NAS, it just chooses a connection which both supports reverse CoA, and which can route packets to the NAS. The server then sends the CoA packet down the chosen connection.

A server can also use RADIUS/UDP to send the reverse CoA packet; there is no requirement that all CoA packets use a "reversed" (D)TLS connection.

After sending a packet, the server then waits for a reply, doing retransmission if necessary. For all issues other than the connection being used, reverse CoA packets are handled as defined in [RFC5176] and in [RFC8559]. This specification permits reverse CoA packets to be sent on what would otherwise be a client to server (D)TLS connection. It does not change the basic functionality of proxying CoA packets.

6.2. Retransmissions

Retransmissions of reverse CoA packets are handled identically to normal CoA packets. That is, the reverse CoA functionality extends the available transport methods for CoA packets, it does not change anything else about how CoA packets are handled.

7. Implementation Status

RFC Editor: This section may be removed before publication.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

7.1. FreeRADIUS

The FreeRADIUS project has implemented this specification in the v3.2.x (https://github.com/FreeRADIUS/freeradius-server/blob/v3.2.x) branch which is available on GitHub. The feature is not enabled by default, and requires a build flag WITH_COA_TUNNEL to be defined before the new functionality is included with the software.

Maturity: The implementation is at a "beta" level, but has been tested to work with other implementations.

Coverage: All of this specification is supported.

Version Compatibility: Earlier versions of this specification are not supported, but the current version is supported.

Licensing: GPLv2

Contact Information: http://freeradius.org/

Date: This information was updated May 2025.

7.2. Cisco

Cisco supports this specification as of Cisco IOS XE Bengaluru 17.6.1 via Vendor-Specific attributes. reference (https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst9300/software/release/17-6/configuration_guide/sec/b_176_sec_9300_cg/configuring_radsec.pdf)

Maturity: The implementation is available in production.

Coverage: All of this specification is supported.

Version Compatibility: Earlier versions of this specification are not supported, but the current version is supported.

Licensing: Proprietary

Contact Information: http://cisco.com/

Date: This information was updated October 2022.

7.3. Aruba

Aruba documentation states that "Instant supports dynamic CoA (RFC 3576) over RadSec and the RADIUS server uses an existing TLS connection opened by the Instant AP to send the request." reference (https://www.arubanetworks.com/techdocs/Instant_83_WebHelp/Content/Instant_UG/Authentication/ConfiguringRadSec.htm)

Maturity: The implementation is available in production.

Coverage: All of this specification is supported.

Version Compatibility: Earlier versions of this specification are not supported, but the current version is supported.

Licensing: Proprietary

Contact Information: http://hp.com/

Date: This information was updated October 2022.

8. Privacy Considerations

This document does not change or add any privacy considerations over previous RADIUS specifications.

9. Security Considerations

It can be necessary to disconnect users, or to change their authorization. It is a security issue when these changes cannot be performed. This specification therefore increases security by making it easier to enforce security policies across a chain of unrelated proxies.

This document also increases network security by removing the requirement for non-standard "reverse" paths for CoA-Request and Disconnect-Request packets.

10. IANA Considerations

This document requests no action from IANA.

11. Acknowledgements

Thanks to Heikki Vatiainen for testing a preliminary implementation in Radiator, and for verifying interoperability with NAS equipment.

12. Changelog

RFC Editor: This section may be removed before publication.

* 00 - taken from draft-dekok-radext-reverse-coa-01
* 01 - Bumped to avoid expiry
* 02 - Bumped to avoid expiry
* 03 - remove dynamic negotiation and cleanups
* 04 - shephards review
* 05 - tweak refs
* 06 - tweak and claify implementation section
* 07 - address IESG review

13. References

13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, DOI 10.17487/RFC2865, June 2000, <https://www.rfc-editor.org/rfc/rfc2865>.

[RFC5176] Chiba, M., Dommety, G., Eklund, M., Mitton, D., and B. Aboba, "Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS)", RFC 5176, DOI 10.17487/RFC5176, January 2008, <https://www.rfc-editor.org/rfc/rfc5176>.

[RFC6614] Winter, S., McCauley, M., Venaas, S., and K. Wierenga, "Transport Layer Security (TLS) Encryption for RADIUS", RFC 6614, DOI 10.17487/RFC6614, May 2012, <https://www.rfc-editor.org/rfc/rfc6614>.

[RFC7360] DeKok, A., "Datagram Transport Layer Security (DTLS) as a Transport Layer for RADIUS", RFC 7360, DOI 10.17487/RFC7360, September 2014, <https://www.rfc-editor.org/rfc/rfc7360>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8559] DeKok, A. and J. Korhonen, "Dynamic Authorization Proxying in the Remote Authentication Dial-In User Service (RADIUS) Protocol", RFC 8559, DOI 10.17487/RFC8559, April 2019, <https://www.rfc-editor.org/rfc/rfc8559>.

13.2. Informative References

[EDUROAM] eduroam, "eduroam", n.d., <https://eduroam.org>.

[I-D.ietf-radext-deprecating-radius] DeKok, A., "Deprecating Insecure Practices in RADIUS", Work in Progress, Internet-Draft, draft-ietf-radext-deprecating-radius-06, 25 May 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-radext-deprecating-radius-06>.

[I-D.ietf-radext-radiusdtls-bis]
Rieckers, J. and S. Winter, "(Datagram) Transport Layer Security ((D)TLS) Encryption for RADIUS", Work in Progress, Internet-Draft, draft-ietf-radext-radiusdtls-bis-08, 20 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-radext-radiusdtls-bis-08>.

[OPENROAMING]
Alliance, W. B., "OpenRoaming: One global Wi-Fi network", n.d., <https://wballiance.com/openroaming/>.

[RFC3539] Aboba, B. and J. Wood, "Authentication, Authorization and Accounting (AAA) Transport Profile", RFC 3539, DOI 10.17487/RFC3539, June 2003, <https://www.rfc-editor.org/rfc/rfc3539>.

[RFC5997] DeKok, A., "Use of Status-Server Packets in the Remote Authentication Dial In User Service (RADIUS) Protocol", RFC 5997, DOI 10.17487/RFC5997, August 2010, <https://www.rfc-editor.org/rfc/rfc5997>.

[RFC7542] DeKok, A., "The Network Access Identifier", RFC 7542, DOI 10.17487/RFC7542, May 2015, <https://www.rfc-editor.org/rfc/rfc7542>.

[RFC7585] Winter, S. and M. McCauley, "Dynamic Peer Discovery for RADIUS/TLS and RADIUS/DTLS Based on the Network Access Identifier (NAI)", RFC 7585, DOI 10.17487/RFC7585, October 2015, <https://www.rfc-editor.org/rfc/rfc7585>.

[RFC7593] Wierenga, K., Winter, S., and T. Wolniewicz, "The eduroam Architecture for Network Roaming", RFC 7593, DOI 10.17487/RFC7593, September 2015, <https://www.rfc-editor.org/rfc/rfc7593>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/rfc/rfc7942>.

Authors' Addresses

Alan DeKok
InkBridge
Email: alan.dekok@inkbridge.io

Vadim Cargatser
Cisco
Email: vcargats@cisco.com