IPv6 Maintenance
Internet-Draft
Updates: 4862 (if approved)
Intended status: Standards Track
Expires: 16 February 2025

L. Colitti
J. Linkova
X. Ma, Ed.
Google
D. Lamparter
NetDEF, Inc.

15 August 2024

Signalling DHCPv6 Prefix per Client Availability to Hosts
draft-ietf-6man-pio-pflag-09

Abstract

This document defines a "P" flag in the Prefix Information Option (PIO) of IPv6 Router Advertisements (RAs). The flag is used to indicate that the network prefers that clients use the draft-ietf-v6ops-dhcp-pd-per-device deployment model instead of using individual adresses in the on-link prefix assigned using SLAAC or DHCPv6 IA_NA.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 16 February 2025.

Copyright Notice

Table of Contents

1. Introduction

[I-D.ietf-v6ops-dhcp-pd-per-device] documents an IPv6 address assignment model where IPv6 devices obtain dedicated prefixes from the network via DHCPv6 Prefix Delegation [RFC8415].

This model provides devices with large amounts of address space that they can use to individually number VMs or containers or extend the network to downstream devices. It also provides scalability benefits on large networks because network infrastructure devices do not need to maintain state per address: IPv6 neighbor cache, SAVI mappings ([RFC7039]), VXLAN routes, etc. On smaller networks, scaling to support multiple individual IPv6 addresses is less of a concern, because many home routers support hundreds of neighbor cache entries.

Also, many smaller networks currently offer prefix delegation but assume that a limited number of specialized devices and/or applications will require delegated prefixes, and thus do not allocate enough address space to offer prefixes to every device that connects to the network. For example, if hosts enable [I-D.ietf-v6ops-dhcp-pd-per-device] on a home network that only provides a /60, and each host obtains a /64 prefix, then the network will run out of prefixes after 15 devices connect.

Therefore, to safely roll out [I-D.ietf-v6ops-dhcp-pd-per-device] implementations on the client side, it is necessary to have a mechanism for the network to signal to the host which address assignment method is preferred.

2. Requirements Language

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.

3. Rationale

The network administrator might want to indicate to hosts that requesting a prefix via DHCPv6-PD and using that prefix for address assignment (see [I-D.ietf-v6ops-dhcp-pd-per-device]) should be preferred over using individual addresses from the on-link prefix. The information is passed to the host via a P flag in the Prefix Information Option (PIO). The reason for it being a PIO flag is as follows:

* The information must be contained in the Router Advertisement because it must be available to the host before it decides to form IPv6 addresses from the PIO prefix using SLAAC. Otherwise, the host might use SLAAC to form IPv6 addresses from the PIO provided and start using them, even if a unique per-host prefix is available via DHCPv6-PD. Forming addresses via SLAAC is suboptimal because if the host later acquires a prefix using DHCPv6-PD, it can either use both the prefix and SLAAC addresses, reducing the scalability benefits of using DHCPv6-PD, or can remove the SLAAC addresses, which would be disruptive for applications that are using them.

* This information is specific to the particular prefix being announced. For example, a network administrator might want hosts to assign global addresses from delegated prefixes, but use the

PIO prefix to form ULA addresses. Also, in a multihoming situation, one upstream network might choose to assign prefixes via prefix delegation, and another via PIOs.

Note that setting the 'P' flag in a PIO option expresses the operator's preference as to whether hosts should attempt using DHCPv6-PD instead of performing individual address configuration on the prefix. For hosts that honor this preference by requesting prefix delegation, the actual delegated prefix will necessarily be a prefix different from the one from the PIO.

4. P Flag Overview

The P flag (also called DHCPv6-PD preferred flag) is a 1-bit PIO flag, located after the R flag ([RFC6275]). The presence of a PIO with the P flag set indicates that that the network prefers that hosts use Prefix Delegation instead of acquiring individual addresses via SLAAC or DHCPv6 address assignment. This implies that the network has a DHCPv6 server capable of making DHCPv6 Prefix Delegations to every device on the network, as described in [I-D.ietf-v6ops-dhcp-pd-per-device].

Adding the P flag reduces the PIO Reserved1 field ([RFC4861], [RFC8425]) from 5 bits to 4 bits. The resulting format of the Prefix Information Option is as follows:

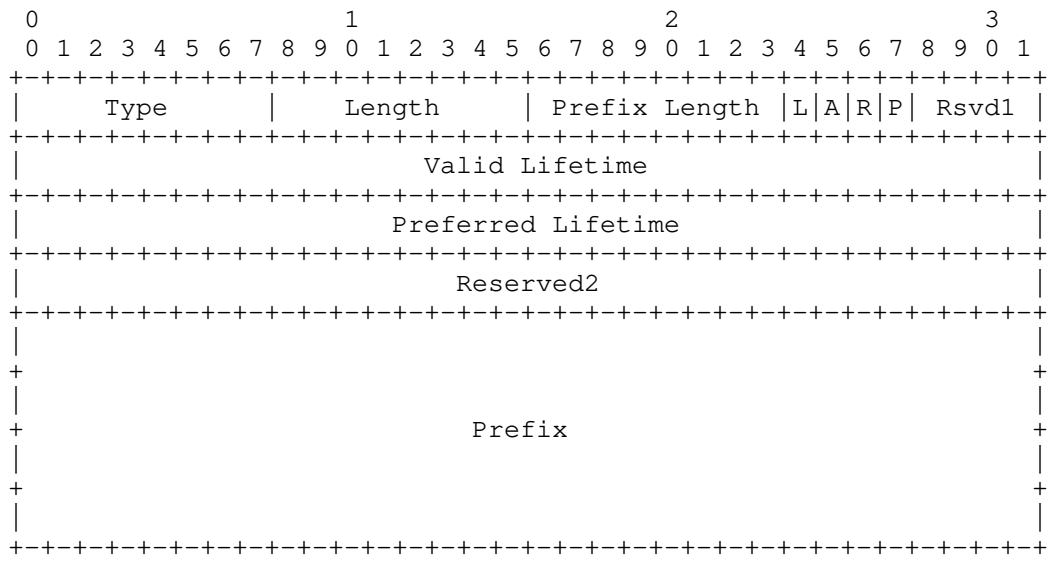


Figure 1

The P flag is independent from the value of the M and O flags in the Router Advertisement. If the network desires to delegate prefixes to devices that support DHCPv6 Prefix Delegation but do not support the P flag, it SHOULD also set the M or O bits in the RA to 1, because some devices, such as [RFC7084] CE routers, might not initiate DHCPv6 Prefix Delegation if both the M and O bits are set to zero.

5. Router Behaviour

Routers SHOULD set the P flag to zero by default, unless explicitly configured by the administrator, and SHOULD allow the operator to set the P flag value for any given prefix.

6. Host Behaviour

This section uses the term host to refer to any node that processes Router Advertisements. This includes both hosts and nodes such as CE Routers [RFC7084] which forward packets but also listen to Router Advertisements.

6.1. Processing the P Flag

This specification only applies to hosts which support acting as DHCPv6 Prefix Delegation clients. Hosts which do not support DHCPv6 prefix delegation MUST ignore the P flag. The P flag is meaningless for link-local prefixes and any Prefix Information Option containing the link-local prefix MUST be ignored as specified in Section 5.5.3 of [RFC4862]. In the following text, all prefixes are assumed not to be link-local.

For each interface, the host MUST keep a list of every prefix that was received from a PIO with the P flag set and currently has a non-zero Preferred Lifetime. The list affects the behaviour of the DHCPv6 client as follows:

* When a prefix's Preferred Lifetime becomes zero, either due to expiration or due to the receipt of a PIO with a Preferred Lifetime of zero, the prefix MUST be removed from the list.

* When the length of the list increases to one, the host SHOULD start requesting prefixes via DHCPv6 prefix delegation unless it is already doing so.

* When the length of the list decreases to zero, the host SHOULD stop requesting or renewing prefixes via DHCPv6 prefix delegation if it has no other reason to do so. The lifetimes of any prefixes already obtained via DHCPv6 are unaffected.

* If the host has already received delegated prefix(es) from one or more servers, then any time a prefix is added to or removed from the list, the host MUST consider this to be a change in configuration information as described in Section 18.2.12 of [RFC8415], and it MUST perform a REBIND, unless it is going to stop the DHCPv6 client because the list became empty. This is in addition to performing a REBIND in the other cases required by that section. Issuing a REBIND allows the host to obtain new prefixes if necessary, e.g., when the network is being renumbered. It also refreshes state related to the delegated prefix(es).

When a host requests a prefix via DHCPv6-PD, it MUST use the prefix length hint Section 18.2.4 of [RFC8415] to request a prefix that is short enough to form addresses via SLAAC.

In order to achieve the scalability benefits of using DHCPv6-PD, the host SHOULD prefer to form addresses from the delegated prefix instead of using individual addresses in the on-link prefix(es). Therefore, when the host requests a prefix using DHCPv6-PD, the host SHOULD NOT use SLAAC to obtain IPv6 addresses from PIOs with the P and A bits set. Similarly, if all PIOs processed by the host have the P bit set, the host SHOULD NOT request individual IPv6 addresses from DHCPv6, i.e., it SHOULD NOT include any IA_NA options in SOLICIT messages. The host MAY continue to use addresses that are already configured.

If the host does not obtain any suitable prefixes via DHCPv6-PD that are suitable for SLAAC, it MAY choose to disable further processing of the P flag on that interface, allowing the host to fall back to other address assignment mechanisms, such as forming addresses via SLAAC (if the PIO has the A flag set to 1) and/or requesting individual addresses via DHCPv6.

6.2. Using Delegated Prefix(es)

If the delegated prefix is too long to be used for SLAAC, the host MUST ignore it. If the prefix is shorter than required for SLAAC, the host SHOULD accept it, allocate one or more longer prefix suitable for SLAAC and use the prefixes as described below.

For every accepted prefix:

* The host MAY form as many IPv6 addresses from the prefix as it chooses.

* The host MAY use the prefix to provide IPv6 addresses to internal components such as virtual machines or containers.

* The host MAY use the prefix to allow devices directly connected to it to obtain IPv6 addresses, e.g., by routing traffic for that prefix to the interface and sending a Router Advertisement containing the prefix on the interface. If the host does so, and it has has formed addresses from the prefix, then it MUST act as though the addresses were assigned to that interface for the purposes of Neighbour Discovery and Duplicate Address Detection.

The host MUST NOT forward packets with destination addresses within a delegated prefix to the interface that it obtained the prefix on, as this will cause a routing loop. This problem will not occur if the host has assigned the prefix to a downstream interface. If the host has not assigned the prefix to a downstream interface, then one way to prevent this problem this is to add to its routing table a high-metric discard route for the delegated prefix. Similarly, the host MUST NOT send packets with destination addresses in the delegated prefix to the interface that it obtained the prefix on.

6.3. Absence of PIOs with P bit set

The P bit is purely a positive indicator, telling nodes that DHCPv6 Prefix Delegation is available and the network prefers that nodes use it, even if they do not have any other reason to run a Prefix Delegation client. The absence of any PIOs with the P bit does not carry any kind of signal to the opposite, and MUST NOT be processed to mean that DHCPv6-PD is absent. In particular, nodes that run DHCPv6-PD due to explicit configuration or by default (e.g., to extend the network) MUST NOT disable DHCPv6-PD on the absence of PIOs with the P bit set. A very common example of this are CE routers as described by [RFC7084].

6.4. On-link Communication

When the network delegates unique prefixes to clients, each client will consider other client’s destination addresses to be off-link, because those addresses are from the delegated prefixes and are not within any on-link prefix. When a client sends traffic to another client, packets will initially be sent to the default router. The router may respond with an ICMPv6 redirect message (Section 4.5 of [RFC4861]). If the client receives and accepts the redirect, then traffic can flow directly from device to device. Therefore, hosts supporting the P flag SHOULD process redirects unless configured otherwise.

6.5. Source Address Selection

For the purpose of source address selection [RFC6724], if the host forms addresses from a delegated prefix, it SHOULD treat those addresses as if they were assigned to the interface on which the prefix was received. This includes placing them in the candidate set, and associating them with the outgoing interface when implementing rule 5.

7. Multihoming

In multi-prefix multihoming, the host generally needs to associate the prefix with the router that advertised it (see for example, [RFC6724] Rule 5.5). If the host supports Rule 5.5, then it SHOULD associate each prefix with the link-local address of the DHCPv6 relay from which it received the REPLY packet. When receiving multiple REPLYs carrying the same prefix from distinct link-local addresses, the host SHOULD associate that prefix with all of these addresses. This can commonly happen in networks with redundant routers and DHCPv6 relays.

8. Modifications to RFC-Mandated Behavior

8.1. Changes to RFC4862

This document makes the following changes to Section 5.5.3 of [RFC4862]:

OLD TEXT

===

For each Prefix-Information option in the Router Advertisement:

a) If the Autonomous flag is not set, silently ignore the Prefix Information option.

===

NEW TEXT: Insert the following text after "For each Prefix-Information option in the Router Advertisement:" but before "If the Autonomous flag is not set, silently ignore the Prefix Information option.":

===

a) If the P flag is set, the node SHOULD treat the Autonomous flag as if it was unset, and use prefix delegation to obtain addresses as described in draft-ietf-6man-pio-pflag.

===

9. Security Considerations

The mechanism described in this document relies on the information provided in the Router Advertisement and therefore shares the same security model as SLAAC. If the network doesn't implement RA Guard [RFC6105], an attacker might send RAs containing the PIO used by the network, set the P flag to 1 and force hosts to ignore the A flag. In the absence of DHCPv6-PD infrastructure, hosts would either obtain no IPv6 addresses or, if they fall back to other IPv6 address assignment mechanisms such as SLAAC and IA_NA, would experience delays in obtaining IPv6 addresses. If the network does not support DHCPv6-Shield [RFC7610], the attacker could also run a rogue DHCPv6 server, providing the host with invalid prefixes or other invalid configuration information.

The attacker might force hosts to oscillate between DHCPv6-PD and PIO-based SLAAC by sending the same set of PIOs with and then w/o P flag set. That would cause the clients to issue REBIND requests, increasing the load on the DHCP infrastructure. However Section 14.1 of [RFC8415] requires that DHCPv6-PD clients rate limit transmitted DHCPv6 messages.

It should be noted that if the network allows rogue RAs to be sent, the attacker would be able to disrupt hosts connectivity anyway, so this document doesn't introduce any fundamentally new security considerations.

Security considerations inherent to the PD-per-device model are documented in Section 15 of [I-D.ietf-v6ops-dhcp-pd-per-device].

10. Privacy Considerations

The privacy implications of implementing the P flag and using DHCPv6-PD to assign prefixes to hosts are similar to privacy implications of using DHCPv6 for assigning individual addresses. If the DHCPv6 infrastructure assigns the same prefix to the same client, then an observer might be able to identify clients based on the highest 64 bits of the client's address. Those implications and recommended countermeasures are discussed in Section 13 of [I-D.ietf-v6ops-dhcp-pd-per-device].

Implementing the P flag support on a host / receiving side enables DHCPv6 on that host. Sending DHCPv6 packets may reveal some minor additional information about the host, most prominently the hostname. This is not a new concern and would apply for any network which uses DHCPv6 and sets 'M' flag in Router Advertisements.

No privacy considerations result from supporting the P flag on the sender side.

11. IANA Considerations

This memo requests that IANA allocate bit 3 from the "IPv6 Neighbor Discovery Prefix Information Option Flags" registry created by [RFC8425] for use as the P flag as described in this document. The following entry should be appended:

PIO Option Bit	Description	Reference
3	P - DHCPv6-PD preferred flag	[THIS DOCUMENT]

Table 1

12. References

12.1. 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/info/rfc2119>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <https://www.rfc-editor.org/info/rfc4861>.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <https://www.rfc-editor.org/info/rfc4862>.

[RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <https://www.rfc-editor.org/info/rfc6724>.

[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/info/rfc8174>.

[RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <https://www.rfc-editor.org/info/rfc8415>.

[RFC8425] Troan, O., "IANA Considerations for IPv6 Neighbor Discovery Prefix Information Option Flags", RFC 8425, DOI 10.17487/RFC8425, July 2018, <https://www.rfc-editor.org/info/rfc8425>.

12.2. Informative References

[RFC6105] Levy-Abegnoli, E., Van de Velde, G., Popoviciu, C., and J. Mohacsi, "IPv6 Router Advertisement Guard", RFC 6105, DOI 10.17487/RFC6105, February 2011, <https://www.rfc-editor.org/info/rfc6105>.

[I-D.ietf-v6ops-dhcp-pd-per-device] Colitti, L., Linkova, J., and X. Ma, "Using DHCPv6-PD to Allocate Unique IPv6 Prefix per Client in Large Broadcast Networks", Work in Progress, Internet-Draft, draft-ietf-v6ops-dhcp-pd-per-device-08, 3 April 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-v6ops-dhcp-pd-per-device-08>.

[RFC6275] Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, DOI 10.17487/RFC6275, July 2011, <https://www.rfc-editor.org/info/rfc6275>.

[RFC7084] Singh, H., Beebee, W., Donley, C., and B. Stark, "Basic Requirements for IPv6 Customer Edge Routers", RFC 7084, DOI 10.17487/RFC7084, November 2013, <https://www.rfc-editor.org/info/rfc7084>.

[RFC7039] Wu, J., Bi, J., Bagnulo, M., Baker, F., and C. Vogt, Ed., "Source Address Validation Improvement (SAVI) Framework", RFC 7039, DOI 10.17487/RFC7039, October 2013, <https://www.rfc-editor.org/info/rfc7039>.

[RFC7610] Gont, F., Liu, W., and G. Van de Velde, "DHCPv6-Shield: Protecting against Rogue DHCPv6 Servers", BCP 199, RFC 7610, DOI 10.17487/RFC7610, August 2015, <https://www.rfc-editor.org/info/rfc7610>.

Acknowledgements

Thanks to Nick Buraglio, Brian Carpenter, Tim Chown, David Farmer, Fernando Gont, Suresh Krishnan, Ted Lemon, Andrew McGregor, Tomek Mrugalski, Michael Richardson, Ole TrÃ¸an, Timothy Winters for the discussions, the input and all contributions.

Authors' Addresses

Lorenzo Colitti
Google
Shibuya 3-21-3,
Japan
Email: lorenzo@google.com

Jen Linkova
Google
1 Darling Island Rd
Pyrmont NSW 2009
Australia
Email: furry13@gmail.com, furry@google.com

Xiao Ma (editor)
Google
Shibuya 3-21-3,
Japan
Email: xiaom@google.com

David 'equinox' Lamparter
NetDEF, Inc.
San Jose,
United States of America
Email: equinox@diac24.net, equinox@opensourcerouting.org

GNAP
Internet-Draft
Intended status: Standards Track
Expires: 27 March 2025

J. Richer, Ed.
Bespoke Engineering
F. Imbault
acert.io

23 September 2024

Grant Negotiation and Authorization Protocol Resource Server Connections
draft-ietf-gnap-resource-servers-09

Abstract

GNAP defines a mechanism for delegating authorization to a piece of software (the client), and conveying the results and artifacts of that delegation to the software. This extension defines methods for resource servers (RS) to connect with authorization servers (AS) in an interoperable fashion.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 27 March 2025.

Table of Contents

1. Introduction

The core GNAP specification ([GNAP]) defines distinct roles for the authorization server (AS) and the resource server (RS). However, the core specification does not define how the RS gets answers to important questions, such as whether a given access token is still valid or what set of access rights the access token is approved for.

While it's possible for the AS and RS to be tightly coupled, such as a single deployed server with a shared storage system, GNAP does not presume or require such a tight coupling. It is increasingly common for the AS and RS to be run and managed separately, particularly in cases where a single AS protects multiple RS's simultaneously.

This specification defines a set of RS-facing APIs that an AS can make available for advanced loosely-coupled deployments. Additionally, this document defines a general-purpose model for access tokens, which can be used in structured, formatted access tokens or in token introspection responses. This specification also defines a method for an RS to derive a downstream token for calling another chained RS.

The means of the authorization server issuing the access token to the client instance and the means of the client instance presenting the access token to the resource server are the subject of the core GNAP specification [GNAP].

1.1. 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.

This document contains non-normative examples of partial and complete HTTP messages, JSON structures, URLs, query components, keys, and other elements. Some examples use a single trailing backslash \ to indicate line wrapping for long values, as per [RFC8792]. The \ character and leading spaces on wrapped lines are not part of the value.

Terminology specific to GNAP is defined in the terminology section of the core specification [GNAP], and provides definitions for the protocol roles: authorization server (AS), client, resource server (RS), resource owner (RO), end user; as well as the protocol elements: attribute, access token, grant, privilege, protected resource, right, subject, subject information. The same definitions are used in this document.

2. Access Tokens

Access tokens are a mechanism for an AS to provide a client instance limited access to an RS. These access tokens are artifacts representing a particular set of access rights granted to the client instance to act on behalf of the RO. While the format of access tokens varies in different systems (see discussion in Section 2.2), the concept of an access token is consistent across all GNAP systems.

2.1. General-purpose Access Token Model

The core GNAP specification [GNAP] focuses on the relationship between the client and the AS. Since the access token is opaque to the client, the core specification does not define a token model. However, the AS will need to create tokens and the RS will need to understand tokens. To facilitate a level of structural interoperability, a common access token model is presented here. Access tokens represent a common set of aspects across different GNAP deployments. This list is not intended to be universal or comprehensive, but rather serves as guidance to implementers in developing data structures and associated systems across a GNAP deployment. These data structures are communicated between the AS and RS either by using a structured token or an API-like mechanism like token introspection (see Section 3.3).

This general-purpose data model does not assume either approach, and in fact both can be used together to convey different pieces of information. Where possible, mappings to the [JWT] standard format are provided for each item in the model.

2.1.1. Value

All access tokens have a _value_, which is the string that is passed on the wire between parties. In order for different access tokens to be differentiated at runtime, the value of a token needs to be unique within a security domain (such as all systems controlled by an AS). Otherwise, two separate tokens would be confused for each other which would lead to security issues. The AS chooses the value, which can be structured as in Section 2.2 or unstructured. When the token is structured, the token value also has a _format_ known to the AS and RS, and the other items in this token model are contained within the token's value in some fashion. When the token is unstructured, the values are usually retrieved by the RS using a service like token introspection described in Section 3.3.

The access token value is conveyed the value field of an access_token response from Section 3.2 of [GNAP].

The format and content of the access token value is opaque to the client software. While the client software needs to be able to carry and present the access token value, the client software is never expected nor intended to be able to understand the token value itself.

If structured tokens like [JWT] are used, the value of the token might not be stored by the AS. Instead, a token identifier can be used along with protection by an AS-generated signature to validate and identify an individual token.

2.1.2. Issuer

The access token is issued by the AS as defined by [GNAP]. The AS will need to identify itself in order to allow an RS to recognize tokens that the AS has issued, particularly in cases where tokens from multiple different AS's could be presented to the same RS.

This information is not usually conveyed directly to the client instance, since the client instance should know this information based on where it receives the token from.

In a [JWT] formatted token or a token introspection response, this corresponds to the iss claim.

2.1.3. Audience

The access token is intended for use at one or more RS's. The AS can list a token's intended RS's to allow each RS to ensure that the RS is not receiving a token intended for someone else. The AS and RS have to agree on the nature of any audience identifiers represented by the token, but the URIs of the RS are a common pattern.

In a [JWT] formatted token or token introspection response, this corresponds to the aud claim.

In cases where more complex access is required, the location field of objects in the access array can also convey audience information. In such cases, the client instance might need to know the audience information in order to differentiate between possible RS's to present the token to.

2.1.4. Key Binding

Access tokens in GNAP are bound to the client instance's registered or presented key, except in cases where the access token is a bearer token. For all tokens bound to a key, the AS and RS need to be able to identify which key the token is bound to, otherwise an attacker could substitute their own key during presentation of the token. In the case of an asymmetric algorithm, the AS and RS needs to know only the public key, while the client instance will also need to know the private key in order to present the token. In the case of a symmetric algorithm, all parties will need to either know or be able to derive the shared key.

The source of this key information can vary depending on deployment decisions. For example, an AS could decide that all tokens issued to a client instance are always bound to that client instance's current key. When the key needs to be dereferenced, the AS looks up the client instance to which the token was issued and finds the key information there. The AS could alternatively bind each token to a specific key that is managed separately from client instance information. In such a case, the AS determines the key information directly. This approach allows the client instance to use a different key for each request, or allows the AS to issue a key for the client instance to use with the particular token.

In all cases, the key binding also includes a proofing mechanism, along with any parameters needed for that mechanism such as a signing or digest algorithm. If such information is not included with the proofing key, an attacker could present a token with a seemingly-valid key using an insecure and incorrect proofing mechanism.

This value is conveyed to the client instance in the key field of the access_token response in Section 3.2 of [GNAP]. Since the common case is that the token is bound to the client instance's registered key, this field can be omitted in this case since the client will be aware of its own key.

In a [JWT] formatted token, this corresponds to the cnf (confirmation) claim. In a token introspection response, this corresponds to the key claim.

In the case of a bearer token, all parties need to know that a token has no key bound to it, and will therefore reject any attempts to use the bearer token with a key in an undefined way.

2.1.5. Flags

GNAP access tokens can have multiple data flags associated with them that indicate special processing or considerations for the token. For example, whether the token is a bearer token, or should be expected to be durable across grant updates.

The client can request a set of flags using the flags field of the access_token grant request parameter in Section 2.1.1 of [GNAP].

These flags are conveyed from the AS to the client in the flags field of the access_token section of the grant response in Section 3.2 of [GNAP].

For token introspection, flags are returned in the flags field of the response.

2.1.6. Access Rights

Access tokens are tied to a limited set of access rights. These rights specify in some detail what the token can be used for, how, and where. The internal structure of access rights are detailed in Section 8 of [GNAP].

The access rights associated with an access token are calculated from the rights available to the client instance making the request, the rights available to be approved by the RO, the rights actually approved by the RO, and the rights corresponding to the RS in question. The rights for a specific access token are a subset of the overall rights in a grant request.

These rights are requested by the client instance in the access field of the access_token request in Section 2.1 of [GNAP].

The rights associated with an issued access token are conveyed to the client instance in the access field of the access_token response in Section 3.2 of [GNAP].

In token introspection responses, this corresponds to the access claim.

2.1.7. Time Validity Window

The access token can be limited to a certain time window outside of which it is no longer valid for use at an RS. This window can be explicitly bounded by an expiration time and a not-before time, or it could be calculated based on the issuance time of the token. For example, an RS could decide that it will accept tokens for most calls within an hour of a token's issuance, but only within five minutes of the token's issuance for certain high-value calls.

Since access tokens could be revoked at any time for any reason outside of a client instance's control, the client instance often does not know or concern itself with the validity time window of an access token. However, this information can be made available to it using the expires_in field of an access token response in Section 3.2 of [GNAP].

The issuance time of the token is conveyed in the iat claim of a [JWT] formatted token or a token introspection response.

The expiration time of a token, after which it is to be rejected, is conveyed in the exp claim of a [JWT] formatted token or a token introspection response.

The starting time of a token's validity window, before which it is to be rejected, is conveyed in the nbf claim of a [JWT] formatted token or a token introspection response.

2.1.8. Token Identifier

Individual access tokens often need a unique internal identifier to allow the AS to differentiate between multiple separate tokens. This value of the token can often be used as the identifier, but in some cases a separate identifier is used.

This separate identifier can be conveyed in the jti claim of a [JWT] formatted token or a token introspection response.

This identifier is not usually exposed to the client instance using the token, since the client instance only needs to use the token by value.

2.1.9. Authorizing Resource Owner

Access tokens are approved on behalf of a resource owner (RO). The identity of this RO can be used by the RS to determine exactly which resource to access, or which kinds of access to allow. For example, an access token used to access identity information can hold a user identifier to allow the RS to determine which profile information to return. The nature of this information is subject to agreement by the AS and RS.

This corresponds to the sub claim of a [JWT] formatted token or a token introspection response.

Detailed RO information is not returned to the client instance when an access token is requested alone, and in many cases returning this information to the client instance would be a privacy violation on the part of the AS. Since the access token represents a specific delegated access, the client instance needs only to use the token at its target RS. Following the profile example, the client instance does not need to know the account identifier to get specific attributes about the account represented by the token.

GNAP does allow for the return of subject information separately from the access token, in the form of identifiers and assertions. These values are returned directly to the client separately from any access tokens that are requested, though it's common that they represent the same party.

2.1.10. End User

The end user is the party operating the client software. The client instance can facilitate the end user interacting with the AS in order to determine the end user's identity, gather authorization, and provide the results of that information back to the client instance.

In many instances, the end user is the same party as the resource owner. However, in some cases, the two roles can be fulfilled by different people, where the RO is consulted asynchronously. The token model should be able to reflect these kinds of situations by representing the end user and RO separately. For example, an end user can request a financial payment, but the RO is the holder of the account that the payment would be withdrawn from. The RO would be consulted for approval by the AS outside of the flow of the GNAP request. A token in such circumstances would need to show both the RO and end user as separate entities.

2.1.11. Client Instance

Access tokens are issued to a specific client instance by the AS. The identity of this instance can be used by the RS to allow specific kinds of access, or other attributes about the access token. For example, an AS that binds all access tokens issued to a particular client instance to that client instance's most recent key rotation would need to be able to look up the client instance in order to find the key binding detail.

This corresponds to the client_id claim of a [JWT] formatted token or the instance_id field of a token introspection response.

The client is not normally informed of this information separately, since a client instance can usually correctly assume that it is the client instance to which a token that it receives was issued.

2.1.12. Label

When multiple access tokens are requested or a client instance uses token labels, the parties will need to keep track of which labels were applied to each individual token. Since labels can be re-used across different grant requests, the token label alone is not sufficient to uniquely identify a given access token in a system. However, within the context of a grant request, these labels are required to be unique.

A client instance can request a specific label using the label field of an access_token request in Section 2.1 of [GNAP].

The AS can inform the client instance of a token's label using the label field of an access_token response in Section 3.2 of [GNAP].

This corresponds to the label field of a token introspection response.

2.1.13. Parent Grant Request

All access tokens are issued in the context of a specific grant request from a client instance. The grant request itself represents a unique tuple of:

* The AS processing the grant request
* The client instance making the grant request
* The RO (or set of RO's) approving the grant request (or needing to approve it)
* The access rights granted by the RO
* The current state of the grant request, as defined in Section 1.5 of [GNAP]

The AS can use this information to tie common information to a specific token. For instance, instead of specifying a client instance for every issued access token for a grant, the AS can store the client information in the grant itself and look it up by reference from the access token.

The AS can also use this information when a grant request is updated. For example, if the client instance asks for a new access token from an existing grant, the AS can use this link to revoke older non-durable access tokens that had been previously issued under the grant.

A client instance will have its own model of an ongoing grant request, especially if that grant request can be continued using the API defined in Section 5 of [GNAP] where several pieces of statefulness need to be kept in hand. The client instance might need to keep an association with the grant request that issued the token in case the access token expires or does not have sufficient access rights, so that the client instance can get a new access token without having to restart the grant request process from scratch.

Since the grant itself does not need to be identified in any of the protocol messages, GNAP does not define a specific grant identifier to be conveyed between any parties in the protocol. Only the AS needs to keep an explicit connection between an issued access token and the parent grant that issued it.

2.1.14. AS-Specific Access Tokens

When an access token is used for the grant continuation API defined in Section 5 of [GNAP] (the continuation access token) the token management API defined in Section 6 of [GNAP] (the token management access token), or the RS-facing API defined in Section 3 (the resource server management access token), the AS MUST separate these access tokens from others usable at RS's. The AS can do this through the use of a flag on the access token data structure, by using a special internal access right, or any other means at its disposal. Just like other access tokens in GNAP, the contents of these AS-specific access tokens are opaque to the software presenting the token. Unlike other access tokens, the contents of these AS-specific access tokens are also opaque to the RS.

The client instance is given continuation access tokens only as part of the continue field of the grant response in Section 3.1 of [GNAP]. The client instance is given token management access tokens only as part of the manage field of the grant response in Section 3.1.2 of [GNAP]. The means by which the RS is given resource server management access tokens is out of scope of this specification, but methods could include pre-configuration of the token value with the RS software or granting the access token through a standard GNAP process.

For continuation access tokens and token management access tokens, a client instance MUST take steps to differentiate these special-purpose access tokens from access tokens used at RS's. To facilitate this, a client instance can store AS-specific access tokens separately from other access tokens in order to keep them from being confused with each other and used at the wrong endpoints.

An RS should never see an AS-specific access token presented, so any attempts to process one MUST fail. When introspection is used, the AS MUST NOT return an active value of true for AS-specific access tokens to the RS. If an AS implements its protected endpoints in such a way as it uses token introspection internally, the AS MUST differentiate these AS-specific access tokens from those issued for use at an external RS.

2.2. Access Token Formats

When the AS issues an access token for use at an RS, the RS needs to have some means of understanding what the access token is for in order to determine how to respond to the request. The core GNAP protocol makes neither assumptions nor demands on the format or contents of the access token, and in fact, the token format and contents are opaque to the client instance. However, such token formats can be the topic of agreements between the AS and RS.

Self-contained structured token formats allow for the conveyance of information between the AS and RS without requiring the RS to call the AS at runtime as described in Section 3.3. Structured tokens can also be used in combination with introspection, allowing the token itself to carry one class of information and the introspection response to carry another.

Some token formats, such as Macaroons [MACAROON] and Biscuits [BISCUIT], allow for the RS to derive sub-tokens without having to call the AS as described in Section 4.

The supported token formats can be communicated dynamically at runtime between the AS and RS in several places.

* The AS can declare its supported token formats as part of RS-facing discovery Section 3.1

* The RS can require a specific token format be used to access a registered resource set Section 3.4

* The AS can return the token's format in an introspection response Section 3.3

In all places where the token format is listed explicitly, it MUST be one of the registered values in the GNAP Token Formats Registry Section 5.3.

3. Resource-Server-Facing API

To facilitate runtime and dynamic connections with an RS, the AS can offer an RS-Facing API consisting of one or more of the following optional pieces.

* Discovery

* Introspection

* Token chaining

* Resource reference registration

3.1. RS-facing AS Discovery

A GNAP AS offering RS-facing services can publish its features on a well-known discovery document at the URL with the same schema and authority as the grant request endpoint URL, at the path /.well-known/gnap-as-rs.

The discovery response is a JSON document [RFC8259] consisting of a single JSON object with the following fields:

grant_request_endpoint (string): The location of the AS's grant request endpoint defined by Section 9 of [GNAP]. This URL MUST be the same URL used by client instances in support of GNAP requests. The RS can use this to derive downstream access tokens, if supported by the AS. The location MUST be a URL [RFC3986] with a scheme component that MUST be https, a host component, and optionally, port, path and query components and no fragment components. REQUIRED. See Section 4.

introspection_endpoint (string): The URL of the endpoint offering introspection. The location MUST be a URL [RFC3986] with a scheme component that MUST be https, a host component, and optionally, port, path and query components and no fragment components. REQUIRED if the AS supports introspection. An absent value indicates that the AS does not support introspection. See Section 3.3.

token_formats_supported (array of strings): A list of token formats supported by this AS. The values in this list MUST be registered in the GNAP Token Format Registry in Section 5.3. OPTIONAL.

resource_registration_endpoint (string): The URL of the endpoint offering resource registration. The location MUST be a URL [RFC3986] with a scheme component that MUST be https, a host component, and optionally, port, path and query components and no fragment components. REQUIRED if the AS supports dynamic resource registration. An absent value indicates that the AS does not support this feature. See Section 3.4.

key_proofs_supported (array of strings) A list of the AS's supported key proofing mechanisms. The values of this list correspond to possible values of the proof field of the key section of the request. Values MUST be in the GNAP Key Proofing Methods registry established by [GNAP]. OPTIONAL.

Additional fields are defined in the GNAP RS-Facing Discovery Document Fields registry Section 5.8.

3.2. Protecting RS requests to the AS

Unless otherwise specified, the RS MUST protect its calls to the AS using any of the signature methods defined by Section 7 of [GNAP].

The RS MAY present its keys by reference or by value in a similar fashion to a client instance calling the AS in the core protocol of GNAP, described in Section 7.1 of [GNAP]. In the protocols defined here, this takes the form of the resource server identifying itself using a key field or by passing an instance identifier directly.

```
POST /continue HTTP/1.1
Host: server.example.com
Authorization: GNAP 80UPRY5NM33OMUKMKSKU
Signature-Input: sig1=...
Signature: sig1=...
Content-Type: application/json

"resource_server": {
    "key": {
        "proof": "httpsig",
        "jwk": {
            "kty": "EC",
            "crv": "secp256k1",
            "kid": "2021-07-06T20:22:03Z",
            "x": "-J9OJIZj4nmopZbQN7T8xv3sbeS5-f_vBNSy_EHnBZc",
            "y": "sjrS51pLtu3P4LUTVvyAIxRfDV_be2RYpI5_f-Yjivw"
        }
    }
}
```

or by reference:

```
POST /continue HTTP/1.1
Host: server.example.com
Signature-Input: sig1=...
Signature: sig1=...
Content-Type: application/json

{
    "resource_server": "7C7C4AZ9KHRS6X63AJAO"
}
```

The means by which an RS's keys are made known to the AS are out of scope of this specification. The AS MAY require an RS to pre-register its keys or could allow calls from arbitrary keys in a trust-on-first-use model.

The AS MAY issue access tokens to the RS for protecting the RS-facing API endpoints, called a resource server management access token. If such tokens are issued, the RS MUST present them to the RS-facing API endpoints along with the RS authentication.

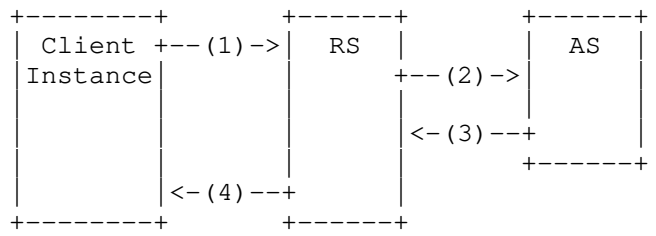
```
POST /continue HTTP/1.1
Host: server.example.com
Authorization: GNAP 80UPRY5NM33OMUKMKSKU
Signature-Input: sig1=...
Signature: sig1=...
Content-Type: application/json

{
    "resource_server": "7C7C4AZ9KHRS6X63AJAO"
}
```

3.3. Token Introspection

The AS issues access tokens representing a set of delegated access rights to be used at one or more RSs. The AS can offer an introspection service to allow an RS to validate that a given access token:

* has been issued by the AS
* is valid at the current time
* has not been revoked
* is appropriate for the RS identified in the call

When the RS receives an access token, it can call the introspection endpoint at the AS to get token information.



1. The client instance calls the RS with its access token.

2. The RS introspects the access token value at the AS. The RS signs the request with its own key (not the client instance's key or the token's key).

3. The AS validates the access token value and the Resource Server's request and returns the introspection response for the token.

4. The RS fulfills the request from the client instance.

The RS signs the request with its own key and sends the value of the access token as the body of the request as a JSON object with the following members:

access_token (string): REQUIRED. The access token value presented to the RS by the client instance.

proof (string): RECOMMENDED. The proofing method used by the client instance to bind the token to the RS request. The value MUST be in the GNAP Key Proofing Methods registry.

resource_server (string or object): REQUIRED. The identification used to authenticate the resource server making this call, either by value or by reference as described in Section 3.2.

access (array of strings/objects): OPTIONAL. The minimum access rights required to fulfill the request. This MUST be in the format described in Section 8 of [GNAP].

Additional fields are defined in the GNAP Token Introspection Request registry Section 5.4.

```
POST /introspect HTTP/1.1
Host: server.example.com
Content-Type: application/json
Signature-Input: sig1=...
Signature: sig1=...
Digest: sha256=...

{
    "access_token": "OS9M2PMHKUR64TB8N6BW7OZB8CDFONP219RP1LT0",
    "proof": "httpsig",
    "resource_server": "7C7C4AZ9KHRS6X63AJAO"
}
```

The AS MUST validate the access token value and determine if the token is active. The parameters of the request provide a context for the AS to evaluate the access token, and the AS MUST take all provided parameters into account when evaluating if the token is active. If the AS is unable to process part of the request, such as not understanding part of the access field presented, the AS MUST NOT indicate the token as active.

An active access token is defined as a token that is all of the following:

* was issued by the processing AS,

* has not been revoked,

* has not expired,

* is bound using the proof method indicated,

* is appropriate for presentation at the identified RS, and

* is appropriate for the access indicated (if present).

The AS responds with a data structure describing the token's current state and any information the RS would need to validate the token's presentation, such as its intended proofing mechanism and key material.

active (boolean): REQUIRED. If true, the access token presented is active, as defined above. If any of the criteria for an active token are not true, or if the AS is unable to make a determination (such as the token is not found), the value is set to false and other fields are omitted.

If the access token is active, additional fields from the single access token response structure defined in Section 3.2.1 of [GNAP] are included. In particular, these include the following:

access (array of strings/objects): REQUIRED. The access rights associated with this access token. This MUST be in the format described in the Section 8 of [GNAP]. This array MAY be filtered or otherwise limited for consumption by the identified RS, including being an empty array, indicating that the token has no explicit access rights that can be disclosed to the RS.

key (object/string): REQUIRED if the token is bound. The key bound

to the access token, to allow the RS to validate the signature of the request from the client instance. If the access token is a bearer token, this MUST NOT be included.

flags (array of strings): OPTIONAL. The set of flags associated with the access token.

exp (integer): OPTIONAL. The timestamp after which this token is no longer valid. Expressed as integer seconds from UNIX Epoch.

iat (integer): OPTIONAL. The timestamp at which this token was issued by the AS. Expressed as integer seconds from UNIX Epoch.

nbf (integer): OPTIONAL. The timestamp before which this token is not valid. Expressed as integer seconds from UNIX Epoch.

aud (string or array of strings): OPTIONAL. Identifiers for the resource servers this token can be accepted at.

sub (string): OPTIONAL. Identifier of the resource owner who authorized this token.

iss (string): REQUIRED. Grant endpoint URL of the AS that issued this token.

instance_id (string): OPTIONAL. The instance identifier of the client instance that the token was issued to.

Additional fields are defined in the GNAP Token Introspection Response registry Section 5.5.

The response MAY include any additional fields defined in an access token response and MUST NOT include the access token value itself.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
    "active": true,
    "access": [
        "dolphin-metadata", "some other thing"
    ],
    "key": {
        "proof": "httpsig",
        "jwk": {
                "kty": "RSA",
                "e": "AQAB",
                "kid": "xyz-1",
                "alg": "RS256",
                "n": "kOB5rR4Jv0GMeL...."
        }
    }
}
```

When processing the results of the introspection response, the RS MUST determine the appropriate course of action. For instance, if the RS determines that the access token's access rights are not sufficient for the request to which the token was attached, the RS can return an error or a public resource, as appropriate for the RS. In all cases, the final determination of the response is at the discretion of the RS.

3.4. Registering a Resource Set

If the RS needs to, it can post a set of resources as described in the Resource Access Rights section of [GNAP] to the AS's resource registration endpoint along with information about what the RS will need to validate the request.

access (array of objects/strings): REQUIRED. The list of access rights associated with the request in the format described in the "Resource Access Rights" section of [GNAP].

resource_server (string or object): REQUIRED. The identification used to authenticate the resource server making this call, either by value or by reference as described in Section 3.2.

token_formats_supported (array of strings): OPTIONAL. The token formats the RS is able to process for accessing the resource. The values in this array MUST be registered in the GNAP Token Formats Registry in Section 5.3. If the field is omitted, the token

format is at the discretion of the AS. If the AS does not support any of the requested token formats, the AS MUST return an error to the RS.

token_introspection_required (boolean): OPTIONAL. If present and set to true, the RS expects to make a token introspection request as described in Section 3.3. If absent or set to false, the RS does not anticipate needing to make an introspection request for tokens relating to this resource set. If the AS does not support token introspection for this RS, the AS MUST return an error to the RS.

Additional fields are defined in the GNAP Resource Set Registration Request registry Section 5.6.

The RS MUST identify itself with its own key and sign the request.

```
POST /resource HTTP/1.1
Host: server.example.com
Content-Type: application/json
Signature-Input: sig1=...
Signature: sig1=...
Digest: ...

{
    "access": [
        {
            "actions": [
                "read",
                "write",
                "dolphin"
            ],
            "locations": [
                "https://server.example.net/",
                "https://resource.local/other"
            ],
            "datatypes": [
                "metadata",
                "images"
            ]
        },
        "dolphin-metadata"
    ],
    "resource_server": "7C7C4AZ9KHRS6X63AJAO"

}
```

The AS responds with a reference appropriate to represent the resources list that the RS presented in its request as well as any additional information the RS might need in future requests.

resource_reference (string): REQUIRED. A single string representing the list of resources registered in the request. The RS MAY make this handle available to a client instance as part of a discovery response as described in Section 9.1 of [GNAP] or as documentation to client software developers.

instance_id (string): OPTIONAL. An instance identifier that the RS can use to refer to itself in future calls to the AS, in lieu of sending its key by value. See Section 3.2.

introspection_endpoint (string): OPTIONAL. The introspection endpoint of this AS, used to allow the RS to perform token introspection. See Section 3.3.

Additional fields are defined in the GNAP Resource Set Registration Response Registry Section 5.7.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
    "resource_reference": "FWWIKYBQ6U56NL1"
}
```

If a resource was previously registered, the AS MAY return the same resource reference value as in previous responses.

If the registration fails, the AS returns an HTTP 400 Bad Request error to the RS indicating that the registration was not successful.

The client instance can then use the resource_reference value as a string-type access reference as defined in Section 8.1 of [GNAP]. This value MAY be combined with any other additional access rights requested by the client instance.

```
{
    "access_token": {
        "access": [
            "FWWIKYBQ6U56NL1",
            {
                "type": "photo-api",
                "actions": [
                    "read",
                    "write",
                    "dolphin"
                ],
                "locations": [
                    "https://server.example.net/",
                    "https://resource.local/other"
                ],
                "datatypes": [
                    "metadata",
                    "images"
                ]
            },
            "dolphin-metadata"
        ]
    },
    "client": "client-12351.bdxqf"
}
```

3.5. Error Responses

In the case of an error from the RS-facing API, the AS responds to the RS with an HTTP 400 (Bad Request) status code and a JSON object consisting of a single error field, which is either an object or a string.

When returned as a string, the error value is the error code:

```
{
    error: "invalid_access"
}
```

When returned as an object, the error object contains the following fields:

code (string): A single ASCII error code defining the error. REQUIRED.

description (string): A human-readable string description of the error intended for the developer of the client. OPTIONAL.

```
{
  "error": {
    "code": "invalid_access",
    "description": "Access to 'foo' is not permitted for this RS."
  }
}
```

This specification defines the following error code values:

"invalid_request": The request is missing a required parameter, includes an invalid parameter value or is otherwise malformed.

"invalid_resource_server": The request was made from an RS that was not recognized or allowed by the AS, or the RS's signature validation failed.

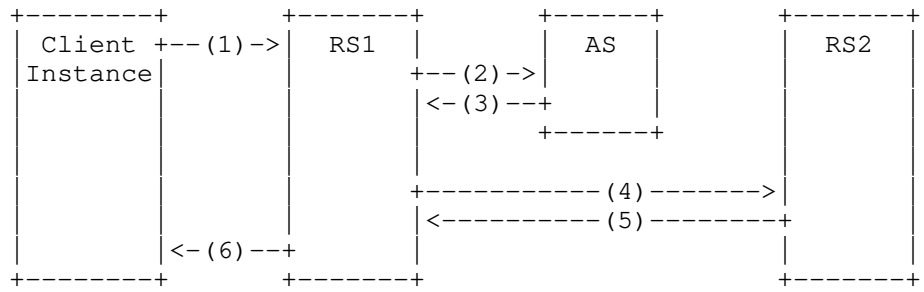
"invalid_access" The RS is not permitted to register or introspect for the requested "access" value.

Additional error codes can be defined in the GNAP RS-Facing Error Codes Registry (Section 5.9).

4. Deriving a downstream token

Some architectures require an RS to act as a client instance and use a derived access token for a secondary RS. Since the RS is not the same entity that made the initial grant request, the RS is not capable of referencing or modifying the existing grant. As such, the RS needs to request or generate a new token access token for its use at the secondary RS. This internal secondary token is issued in the context of the incoming access token.

While it is possible to use a token format (Section 2) that allows for the RS to generate its own secondary token, the AS can allow the RS to request this secondary access token using the same process used by the original client instance to request the primary access token. Since the RS is acting as its own client instance from the perspective of GNAP, this process uses the same grant endpoint, request structure, and response structure as a client instance's request.



1. The client instance calls RS1 with an access token.

2. RS1 presents that token to the AS to get a derived token for use at RS2. RS1 indicates that it has no ability to interact with the RO. Note that RS1 signs its request with its own key, not the token's key or the client instance's key.

3. The AS returns a derived token to RS1 for use at RS2.

4. RS1 calls RS2 with the token from (3).

5. RS2 fulfills the call from RS1.

6. RS1 fulfills the call from the original client instance.

If the RS needs to derive a token from one presented to it, it can request one from the AS by making a token request as described in [GNAP] and presenting the existing access token's value in the "existing_access_token" field.

Since the RS is acting as a client instance, the RS MUST identify itself with its own key in the client field and sign the request just as any client instance would, as described in Section 3.2. The AS MUST determine that the token being presented is appropriate for use at the RS making the token chaining request.

```
POST /tx HTTP/1.1
Host: server.example.com
Content-Type: application/json
Detached-JWS: ejy0...

{
    "access_token": {
        "access": [
            {
                "actions": [
                    "read",
                    "write",
                    "dolphin"
                ],
                "locations": [
                    "https://server.example.net/",
                    "https://resource.local/other"
                ],
                "datatypes": [
                    "metadata",
                    "images"
                ]
            },
            "dolphin-metadata"
        ]
    },
    "client": "7C7C4AZ9KHRS6X63AJAO",
    "existing_access_token": "OS9M2PMHKUR64TB8N6BW7OZB8CDFONP219RP1LT0"
}
```

The AS responds with a token for the downstream RS2 as described in [GNAP]. The downstream RS2 could repeat this process as necessary for calling further RS's.

5. IANA Considerations

IANA is requested to add values to existing registries and to create 5 registries in the Grant Negotiation and Authorization Protocol registry.

5.1. Well-Known URI

The "gnap-as-rs" URI suffix is registered into the Well-Known URIs Registry to support RS-facing discovery of the AS.

URI Suffix: gnap-as-rs

Change Controller: IETF

Specification Document: Section 3.1 of RFC xxxx

Status: Permanent

5.2. GNAP Grant Request Parameters

The following parameter is registered into the GNAP Grant Request Parameters registry:

Name: existing_access_token

Type: string

Reference: Section 4 of RFC xxxx

5.3. GNAP Token Formats

This document defines a GNAP token format, for which IANA is asked to create and maintain a new registry titled "GNAP Token Formats". Initial values for this registry are given in Section 5.3.2. Future assignments and modifications to existing assignment are to be made through the Specification Required registration policy [RFC8126].

The Designated Expert (DE) is expected to ensure that:

* all registrations follow the template presented in Section 5.3.1.

* the format’s definition is sufficiently unique from other formats provided by existing parameters.

* the format’s definition specifies the format of the access token in sufficient detail to allow for the AS and RS to be able to communicate the token information.

5.3.1. Registry Template

Name The name of the format.

Status Whether or not the format is in active use. Possible values are Active and Deprecated.

Description Human-readable description of the access token format.

Reference The specification that defines the token format.

5.3.2. Initial Registry Contents

Name	Status	Description	Reference
jwt-signed	Active	JSON Web Token, signed with JWS	[JWT]
jwt-encrypted	Active	JSON Web Token, encrypted with JWE	[JWT]
macaroon	Active	Macaroon	[MACAROON]
biscuit	Active	Biscuit	[BISCUIT]
zcap	Active	ZCAP	[ZCAPLD]

Table 1: Initial contents of the GNAP Token Formats Registry.

5.4. GNAP Token Introspection Request

This document defines GNAP token introspection, for which IANA is asked to create and maintain a new registry titled "GNAP Token Introspection Request". Initial values for this registry are given in Section 5.4.2. Future assignments and modifications to existing assignment are to be made through the Specification Required registration policy [RFC8126].

The Designated Expert (DE) is expected to ensure that:

* all registrations follow the template presented in Section 5.4.1.

* the claim's definition is sufficiently orthogonal to other claims defined in the registry so as avoid overlapping functionality.

* the claim's definition specifies the syntax and semantics of the claim in sufficient detail to allow for the AS and RS to be able to communicate the token values.

5.4.1. Registry Template

Name The name of the claim.

Type The JSON data type of the claim value.

Reference The specification that defines the claim.

5.4.2. Initial Registry Contents

The table below contains the initial contents of the GNAP Token Introspection Registry.

Name	Type	Reference
access_token	string	Section 3.3 of RFC xxxx
proof	string	Section 3.3 of RFC xxxx
resource_server	object/string	Section 3.3 of RFC xxxx
access	array of strings/objects	Section 3.3 of RFC xxxx

Table 2: Initial contents of the GNAP Token Introspection Request Registry.

5.5. GNAP Token Introspection Response

This document defines GNAP token introspection, for which IANA is asked to create and maintain a new registry titled "GNAP Token Introspection Response". Initial values for this registry are given in Section 5.5.2. Future assignments and modifications to existing assignment are to be made through the Specification Required registration policy [RFC8126].

The Designated Expert (DE) is expected to ensure that:

* all registrations follow the template presented in Section 5.5.1.

* the claim's definition is sufficiently orthogonal to other claims defined in the registry so as avoid overlapping functionality.

* the claim's definition specifies the syntax and semantics of the claim in sufficient detail to allow for the AS and RS to be able to communicate the token values.

5.5.1. Registry Template

Name The name of the claim.

Type The JSON data type of the claim value.

Reference The specification that defines the claim.

5.5.2. Initial Registry Contents

The table below contains the initial contents of the GNAP Token Introspection Registry.

Name	Type	Reference
active	boolean	Section 3.3 of RFC xxxx
access	array of strings/objects	Section 3.3 of RFC xxxx
key	object/string	Section 3.3 of RFC xxxx
flags	array of strings	Section 3.3 of RFC xxxx
exp	integer	Section 3.3 of RFC xxxx
iat	integer	Section 3.3 of RFC xxxx
nbf	integer	Section 3.3 of RFC xxxx
aud	string or array of strings	Section 3.3 of RFC xxxx
sub	string	Section 3.3 of RFC xxxx
iss	string	Section 3.3 of RFC xxxx
instance_id	string	Section 3.3 of RFC xxxx

Table 3: Initial contents of the GNAP Token Introspection Response Registry.

5.6. GNAP Resource Set Registration Request Parameters

This document defines a means to register a resource set for a GNAP AS, for which IANA is asked to create and maintain a new registry titled "GNAP Resource Set Registration Request Parameters". Initial values for this registry are given in Section 5.6.2. Future assignments and modifications to existing assignment are to be made through the Expert Review registration policy [RFC8126].

The Designated Expert (DE) is expected to ensure that:

* all registrations follow the template presented in Section 5.6.1.

* the parameter's definition is sufficiently orthogonal to other parameters defined in the registry so as avoid overlapping functionality.

* the parameter's definition specifies the syntax and semantics of the parameter in sufficient detail to allow for the AS and RS to be able to communicate the resource set.

5.6.1. Registry Template

Name The name of the parameter.

Type The JSON data type of the parameter value.

Reference The specification that defines the token.

5.6.2. Initial Registry Contents

The table below contains the initial contents of the GNAP Resource Set Registration Request Parameters Registry.

Name	Type	Reference
access	array of strings/objects	Section 3.4 of RFC xxxx
resource_server	string or object	Section 3.4 of RFC xxxx
token_formats_supported	string	Section 3.4 of RFC xxxx
token_introspection_required	boolean	Section 3.4 of RFC xxxx

Table 4: Initial contents of the GNAP Resource Set Registration Request Parameters Registry.

5.7. GNAP Resource Set Registration Response Parameters

This document defines a means to register a resource set for a GNAP AS, for which IANA is asked to create and maintain a new registry titled "GNAP Resource Set Registration Response Parameters". Initial values for this registry are given in Section 5.7.2. Future assignments and modifications to existing assignment are to be made through the Expert Review registration policy [RFC8126].

The Designated Expert (DE) is expected to ensure that:

* all registrations follow the template presented in Section 5.7.1.

* the parameter's definition is sufficiently orthogonal to other claims defined in the registry so as avoid overlapping functionality.

* the parameter's definition specifies the syntax and semantics of the claim in sufficient detail to allow for the AS and RS to be able to communicate the resource set.

5.7.1. Registry Template

Name The name of the parameter.

Type The JSON data type of the parameter value.

Reference The specification that defines the parameter.

5.7.2. Initial Registry Contents

The table below contains the initial contents of the GNAP Resource Set Registration Response Parameters Registry.

Name	Type	Reference
resource_reference	string	Section 3.4 of RFC xxxx
instance_id	string	Section 3.4 of RFC xxxx
introspection_endpoint	string	Section 3.4 of RFC xxxx

Table 5: Initial contents of the GNAP Resource Set Registration Response Parameters Registry.

5.8. GNAP RS-Facing Discovery Document Fields

This document defines a means to for a GNAP AS to be discovered by a GNAP RS, for which IANA is asked to create and maintain a new registry titled "GNAP RS-Facing Discovery Document Fields". Initial values for this registry are given in Section 5.8.2. Future assignments and modifications to existing assignment are to be made through the Expert Review registration policy [RFC8126].

The Designated Expert (DE) is expected to ensure that:

* all registrations follow the template presented in Section 5.8.1.

* the field's definition is sufficiently orthogonal to other fields defined in the registry so as avoid overlapping functionality.

* the field's definition specifies the syntax and semantics of the fields in sufficient detail to allow for RS to be able to communicate with the AS.

5.8.1. Registry Template

Name The name of the field.

Type The JSON data type of the field value.

Reference The specification that defines the field.

5.8.2. Initial Registry Contents

The table below contains the initial contents of the GNAP RS-Facing Discovery Registry.

Name	Type	Reference
introspection_endpoint	string	Section 3.1 of RFC xxxx
token_formats_supported	array of strings	Section 3.1 of RFC xxxx
resource_registration_endpoint	string	Section 3.1 of RFC xxxx
grant_request_endpoint	string	Section 3.1 of RFC xxxx
key_proofs_supported	array of strings	Section 3.1 of RFC xxxx

Table 6: Initial contents of the GNAP RS-Facing Discovery Document Fields Registry.

5.9. GNAP RS-Facing Error Codes

This document defines a set of errors that the AS can return to the RS, for which IANA is asked to create and maintain a new registry titled "GNAP RS-Facing Error Codes". Initial values for this registry are given in Section 5.9.2. Future assignments and modifications to existing assignment are to be made through the Specification Required registration policy [RFC8126].

The DE is expected to ensure that:

* all registrations follow the template presented in Section 5.9.1.

* the error response is sufficiently unique from other errors to provide actionable information to the client instance.

* the definition of the error response specifies all conditions in which the error response is returned, and what the client instance's expected action is.

5.9.1. Registration Template

Error:
: A unique string code for the error.

Specification document(s):
: Reference to the document(s) that specify the value, preferably including a URI that can be used to retrieve a copy of the document(s). An indication of the relevant sections may also be included but is not required.

5.9.2. Initial Contents

Error	Specification document(s)
invalid_request	Section 3.5 of RFC xxxx
invalid_resource_server	Section 3.5 of RFC xxxx
invalid_access	Section 3.5 of RFC xxxx

Table 7: Initial contents of the GNAP RS-Facing Error Codes Registry.

6. Security Considerations

In addition to the normative requirements in this document and in [GNAP], implementers are strongly encouraged to consider these additional security considerations in implementations and deployments of GNAP.

6.1. TLS Protection in Transit

All requests in GNAP made over untrusted network connections have to be made over TLS as outlined in [BCP195] to protect the contents of the request and response from manipulation and interception by an attacker. This includes all requests from a client instance to the RS and all requests from the RS to an AS.

6.2. Token Validation

The RS has a responsibility to validate the incoming access token in a manner consistent with its deployment. For self-contained stateless tokens such as those described in Section 2.2, this consists of actions such as validating the token's signature and ensuring the relevant fields and results are appropriate for the request being made. For reference-style tokens or tokens that are otherwise opaque to the RS, the token introspection RS-facing API can be used to provide updated information about the state of the token, as described in Section 3.3.

The RS needs to validate that a token:

* Is intended for this RS (audience restriction)
* Is presented using the appropriate key for the token (see also Section 6.4)
* Identifies an appropriate subject to access the resource (usually this is the resource owner who authorized the token's issuance)
* Is issued by a trusted AS for this resource

Even though key proofing mechanisms have to cover the value of the token, validating the key proofing alone is not sufficient to protect a request to an RS. If an RS validates only the presentation method as described in Section 6.4 without validating the token itself, an attacker could use a compromised key or a confused deputy to make arbitrary calls to the RS beyond what has been authorized by the RO.

6.3. Caching Token Validation Result

Since token validation can be an expensive process, requiring either cryptographic operations or network calls to an introspection service as described in Section 3.3, an RS could cache the results of token validation for a particular token. The trade offs of using a cached validation for a token present an important decision space for implementers: relying on a cached validation result increases performance and lowers processing overhead, but it comes at the expense of the liveness and accuracy of the information in the cache. While a cached value is in use at the RS, an attacker could present a revoked token and have it accepted by the RS.

As with any cache, the consistency of this cache can be managed in a variety of ways. One of the most simple methods is managing the lifetime of the cache in order to balance the performance and security properties. Too long of a cache, and an attacker has a larger window in which to use a revoked token. Too short of a window and the benefits of using the cache are diminished. It is also possible that an AS could send a proactive signal to the RS to invalidate revoked access tokens, though such a mechanism is outside the scope of this specification.

6.4. Key Proof Validation

For key-bound access tokens, the proofing method needs to be validated alongside the value of the token itself as described in Section 6.2. The process of validation is defined by the key proofing method, as described in Section 7.3 of [GNAP].

If the proofing method is not validated, an attacker could use a compromised token without access to the token's bound key.

The RS also needs to ensure that the proofing method is appropriate for the key associated with the token, including any choice of algorithm or identifiers.

The proofing should be validated independently on each request to the RS, particularly as aspects of the call could vary. As such, the RS should never cache the results of a proof validation from one message and apply it to a subsequent message.

6.5. Token Exfiltration

Since the RS sees the token value, it is possible for a compromised RS to leak that value to an attacker. As such, the RS needs to protect token values as sensitive information and protect them from exfiltration.

This is especially problematic with bearer tokens and tokens bound to a shared key, since an RS has access to all information necessary to create a new, valid request using the token in question.

6.6. Token Re-Use by an RS

If the access token is a bearer token, or the RS has access to the key material needed to present the token, the RS could be tricked into re-using an access token presented to it by a client. While it is possible to build a system that makes use of this artifact as a feature, it is safer to exchange the incoming access token for another contextual token for use by the RS, as described in Section 4. This access token can be bound to the RS's own keys and limited to access needed by the RS, instead of the full set of rights associated with the token issued to the client instance.

6.7. Token Format Considerations

With formatted tokens, the format of the token is likely to have its own considerations, and the RS needs to follow any such considerations during the token validation process. The application and scope of these considerations is specific to the format and outside the scope of this specification.

6.8. Over-sharing Token Contents

The contents of the access token model divulge to the RS information about the access token's context and rights. This is true whether the contents are parsed from the token itself or sent in an introspection response.

It's likely that every RS does not need to know all details of the token model, especially in systems where a single access token is usable across multiple RS's. An attacker could use this to gain information about the larger system by compromising only one RS. By limiting the information available to only that which is relevant to a specific RS, such as using a limited introspection reply as defined in Section 3.3, a system can follow the principle of least disclosure to each RS.

6.9. Resource References

Resource references, as returned by the protocol in Section 3.4, are intended to be opaque to both the RS and the client. However, since they are under the control of the AS, the AS can put whatever content it wants into the reference value. This value could unintentionally disclose system structure or other internal details if it processed by an unintended party. Furthermore, such patterns could lead to the

client software and RS depending on certain structures being present in the reference value, which diminishes the separation of concerns of the different roles in a GNAP system.

To mitigate this, the AS should only use fully random or encrypted values for resource references.

6.10. Token Re-Issuance From an Untrusted AS

It is possible for an attacker's client instance to issue its own tokens to another client instance, acting as an AS that the second client instance has chosen to trust. If the token is a bearer token or the re-issuance is bound using an AS-provided key, the target client instance will not be able to tell that the token was originally issued by the valid AS. This process allows an attacker to insert their own session and rights into an unsuspecting client instance, in the guise of a token valid for the attacker that appears to have been issued to the target client instance on behalf of its own RO.

This attack is predicated on a misconfiguration with the targeted client, as it has been configured to get tokens from the attacker's AS and use those tokens with the target RS, which has no association with the attacker's AS. However, since the token is ultimately coming from the trusted AS, and is being presented with a valid key, the RS has no way of telling that the token was passed through an intermediary.

To mitigate this, the RS can publish its association with the trusted AS through either discovery or documentation. Therefore, a client properly following this association would only go directly to the trusted RS directly for access tokens for the RS.

Furthermore, limiting the use of bearer tokens and AS-provided keys to only highly trusted AS's and limited circumstances prevents the attacker from being able to willingly exfiltrate their token to an unsuspecting client instance.

6.11. Introspection of Token Keys

The introspection response defined in Section 3.3 provides a means for the AS to tell the RS the key material needed to validate the key proof of the request. Capture of the introspection response can expose these security keys to an attacker. In the case of asymmetric cryptography, only the public key is exposed, and the token cannot be re-used by the attacker based on this result alone. This could potentially divulge information about the client instance that was unknown otherwise.

If an access token is bound to a symmetric key, the RS will need access to the full key value in order to validate the key proof of the request, as described in Section 6.4. However, divulging the key material to the RS also gives the RS the ability to create a new request with the token. In this circumstance, the RS is under similar risk of token exfiltration and re-use as a bearer token, as described in Section 6.6. Consequently, symmetric keys should only be used in systems where the RS can be fully trusted to not create a new request with tokens presented to it.

6.12. RS Registration and Management

Most functions of the RS-facing API in Section 3 are protected by requiring the RS to present proof of a signing key along with the request, in order to identify the RS making the call, potentially coupled with an AS-specific access token. This practice allows the AS to differentially respond to API calls to different RS's, such as answering introspection calls with only the access rights relevant to a given RS instead of all access rights an access token could be good for.

While the means by which an RS and its keys become known to the AS is out of scope for this specification, it is anticipated that common practice will be to statically register an RS, allowing it to protect specific resources or certain classes of resources. Fundamentally, the RS can only offer the resources that it serves. However, a rogue AS could attempt to register a set of resources that mimics a different RS in order to solicit an access token usable at the target RS. If the access token is a bearer token or is bound to a symmetric key that is known to the RS, then the attacker's RS gains the ability and knowledge needed to use that token elsewhere.

In some ecosystems, dynamic registration of an RS and its associated resources is feasible. In such systems, the identity of the RS could be conveyed by a URI passed in the location field of an access rights request, thereby allowing the AS to limit the view the RS has into the larger system.

7. Privacy Considerations

7.1. Token Contents

The contents of the access token could potentially contain personal information about the end-user, RO, or other parties. This is true whether the contents are parsed from the token itself or sent in an introspection response.

While an RS will sometimes need this information for processing, it's often the case that an RS is exposed to these details only in passing, and not intentionally. For example, consider a client that is issued an access token that is usable for both medical and non-medical APIs. If this access token contains a medical record number to facilitate the RS serving the medical API, then any RS for a non-medical API would also learn the user's medical record number in the process, even though that API has no need to make such a correlation.

To mitigate this, a formatted token could contain separate sections targeted to different RS's to segregate data. Alternatively, token introspection can be used to limit the data returned to each RS, as defined in Section 3.3.

7.2. Token Use Disclosure through Introspection

When introspection is used by an RS, the AS is made aware of a particular token being used at a particular RS. When the RS is a separate system, the AS would not otherwise have insight into this action. This can potentially lead to the AS learning about patterns and actions of particular end users by watching which RS's are accessed and when.

7.3. Mapping a User to an AS

When the client instance receives information about the protecting AS from an RS, this can be used to derive information about the resources being protected without releasing the resources themselves. For example, if a medical record is protected by a personal AS, an untrusted client could call an RS to discover the location of the AS protecting the record. Since the AS is tied strongly to a single RO, the untrusted and unauthorized client software can gain information about the resource being protected without accessing the record itself.

8. Acknowledgements

The editors would like to thank the feedback of the following individuals for their reviews, implementations, and contributions: Aaron Parecki, Adrian Gropper, Andrii Deinega, Annabelle Backman, Dmitry Barinov, Fabien Imbault, Florian Helmschmidt, George Fletcher, Justin Richer, Kathleen Moriarty, Leif Johansson, Mike Varley, Nat Sakimura, Takahiko Kawasaki, Yaron Sheffer.

Finally, the editors want to acknowledge the immense contributions of Aaron Parecki to the content of this document. We thank him for his insight, input, and hard work, without which GNAP would not have grown to what it is.

9. References

9.1. Normative References

[BCP195] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", May 2015, <https://www.rfc-editor.org/info/bcp195>.

[GNAP] Richer, J. and F. Imbault, "Grant Negotiation and Authorization Protocol", Work in Progress, Internet-Draft, draft-ietf-gnap-core-protocol-20, 19 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-gnap-core-protocol-20>.

[JWT] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/rfc/rfc7519>.

[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>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

[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>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

[RFC8792] Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", RFC 8792, DOI 10.17487/RFC8792, June 2020, <https://www.rfc-editor.org/rfc/rfc8792>.

9.2. Informative References

[BISCUIT] "Biscuit Authorization", n.d., <https://www.biscuitsec.org/>.

[MACAROON] "Macaroons: Cookies with Contextual Caveats for Decentralized Authorization in the Cloud", 2014, <https://research.google/pubs/pub41892/>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

[ZCAPLD] "Authorization Capabilities for Linked Data", 2023, <https://w3c-ccg.github.io/zcap-spec/>.

Appendix A. Document History

* -09

  - Updated description of well-known discovery endpoint.

* -08

  - Editorial and IANA updates based on review feedback.

* -07

  - Editorial updates based on review feedback.

* -06

  - Editorial updates based on review feedback.

* -05

  - Added discussion of access tokens used to call the RS-facing AS APIs.

  - Updated IANA sections to align with core (and each other).

  - Added IANA section on introspection requests.

  - Added error responses.

  - Added extended discussion on resource server registration practices.

* -04

  - Editorial cleanup.

  - Updated IANA requirements, including "specification required" registration.

  - Added privacy and security considerations.

  - Clarified and expanded token introspection request and response.

  - Clarified and expanded resource set registration request and response, include example of use of resource reference.

  - Updated discovery.

  - Allow optional tokens on RS-facing API requests.

  - Tighter controls over derived tokens.

* -03

  - Added token model.

  - Added IANA sections.

* -02

  - Editorial and formatting fixes.

* -01

  - Better described RS authentication.

  - Added access token format registry.

  - Filled out introspection protocol.

  - Filled out resource registration protocol.

  - Expanded RS-facing discovery mechanisms.

  - Moved client-facing RS response back to GNAP core document.

* -00

  - Extracted resource server section.

Authors' Addresses

Justin Richer (editor)
Bespoke Engineering
Email: ietf@justin.richer.org
URI: https://bspk.io/

Fabien Imbault
acert.io
Email: fabien.imbault@acert.io
URI: https://acert.io/

GROW
Internet-Draft
Updates: 7854, 8671, 9069 (if approved)
Intended status: Standards Track
Expires: 13 December 2024

J.S. Scudder
Juniper Networks
P. Lucente
NTT

11 June 2024

BMP Peer Up Message Namespace
draft-ietf-grow-bmp-peer-up-04

Abstract

RFC 7854, BMP, uses different message types for different purposes. Most of these are Type, Length, Value (TLV) structured. One message type, the Peer Up message, lacks a set of TLVs defined for its use, instead sharing a namespace with the Initiation message. Subsequent experience has shown that this namespace sharing was a mistake, as it hampers the extension of the protocol.

This document updates RFC 7854 by creating an independent namespace for the Peer Up message. It also updates RFC 8671 and RFC 9069 by moving the defined codepoints in the newly introduced registry. The changes in this document are formal only, compliant implementations of RFC 7854, RFC 8671 and RFC 9069 also comply with this specification.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 13 December 2024.

Table of Contents

1. Introduction

[RFC7854] defines a number of different BMP message types. With the exception of the Route Monitoring message type, these messages are TLV-structured. Most message types have distinct namespaces and IANA registries. However, the namespace of the Peer Up message overlaps that of the Initiation message. As the BMP protocol has been extended, this oversight has become problematic. In this document, we create a distinct namespace for the Peer Up message to eliminate this overlap, and create the corresponding missing registry.

The changes in this document are formal only, compliant implementations of [RFC7854], [RFC8671] and [RFC9069] also comply with this specification.

1.1. Requirements Language

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.

2. String Definition

A string TLV is a free-form sequence of UTF-8 characters whose length in bytes is given by the TLV's Length field. There is no requirement to terminate the string with a null (or any other particular) character -- the Length field gives its termination.

3. Changes to existing RFCs

We update [RFC7854] as follows:

* The "Information TLV" of section 4.4, that was shared between the Initiation and Peer Up message types, is renamed as the "Initiation Information TLV", and is only relevant to the Initiation message type.

* A "Peer Up Information TLV" is defined, and is relevant to the Peer Up message type.

* A "BMP Peer Up Information TLVs" registry is is created, seeded with the Peer Up Information TLV.

Other than as summarized above, and detailed below, there are no other changes.

3.1. Revision to Information TLV, Renamed as Initiation Information TLV

The Information TLV defined in section 4.4 of [RFC7854] is renamed "Initiation Information TLV". It is used only by the Initiation message, not by the Peer Up message.

The definition of Type = 0 is revised to be:

* Type = 0: String. The Information field contains a string (Section 2). The value is administratively assigned. If multiple strings are included, their ordering MUST be preserved when they are reported.

* Type = 1: sysDescr. The Information field contains an ASCII string whose value MUST be set to be equal to the value of the sysDescr MIB-II [RFC1213] object.

* Type = 2: sysName. The Information field contains an ASCII string whose value MUST be set to be equal to the value of the sysName MIB-II [RFC1213] object.

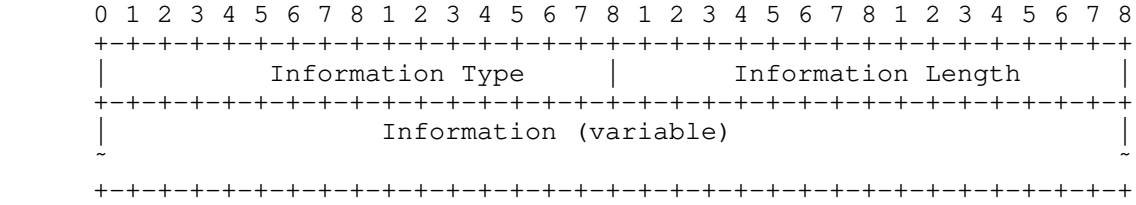
3.2. Revision to Peer Up Notification

The final paragraph of section 4.10 of [RFC7854] references the Information TLV (which is revised above (Section 3.1)). That paragraph is replaced by the following:

* Information: Information about the peer, using the Peer Up Information TLV format defined below (Section 3.3). The String type may be repeated. Inclusion of the Information field is OPTIONAL. Its presence or absence can be inferred by inspection of the Message Length in the common header.

3.3. Definition of Peer Up Information TLV

The Peer Up Information TLV is used by the Peer Up message.



* Information Type (2 bytes): Type of information provided. Defined types are:

  - Type = 0: String. The Information field contains a string (Section 2). The value is administratively assigned. If multiple strings are included, their ordering MUST be preserved when they are reported.

  - Type = 3: VRF/Table Name. The Information field contains a UTF-8 string whose value MUST be equal to the value of the VRF or table name (e.g., RD instance name) being conveyed. The string size MUST be within the range of 1 to 255 bytes.

- Type = 4: Admin Label. The Information field contains a free-form UTF-8 string whose byte length is given by the Information Length field. The value is administratively assigned. There is no requirement to terminate the string with null or any other character.

* Information Length (2 bytes): The length of the following Information field, in bytes.

* Information (variable): Information about the monitored router, according to the type.

4. IANA Considerations

IANA is requested to create a registry within the BMP group, named "BMP Peer Up Message TLVs", reference this document.

Registration procedures for this registry are:

Range	Registration Procedures
0, 3-32767	Standards Action
32768-65530	First Come, First Served
65531-65534	Experimental
1-2, 65535	Reserved

Table 1

Initial values for this registry are:

Type	Description	Reference
0	String	this document
1	Reserved	this document
2	Reserved	this document
3	VRF/Table Name	this document
4	Admin Label	this document
65535	Reserved	this document

Table 2

IANA is also requested to rename the existing "BMP Initiation and Peer Up Information TLVs" registry to "BMP Initiation Information TLVs" and seed it with the following values:

Type	Description	Reference
0	String	this document
1	sysDescr	this document
2	sysName	this document
3	Reserved	this document
4	Reserved	this document
65535	Reserved	this document

Table 3

5. Security Considerations

This rearrangement of deck chairs does not change the underlying security issues inherent in the existing [RFC7854].

6. Implementation status - RFC EDITOR: REMOVE 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. 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.

As of today these vendors have produced an implementation of the BMP Peer Up Namespace:

* FRRouting

* pmacct

7. Acknowledgements

The authors would like to thank Maxence Younsi for his review.

8. Normative References

[RFC1213] McCloghrie, K. and M. Rose, "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", STD 17, RFC 1213, DOI 10.17487/RFC1213, March 1991, <https://www.rfc-editor.org/info/rfc1213>.

[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/info/rfc2119>.

[RFC7854] Scudder, J., Ed., Fernando, R., and S. Stuart, "BGP Monitoring Protocol (BMP)", RFC 7854, DOI 10.17487/RFC7854, June 2016, <https://www.rfc-editor.org/info/rfc7854>.

[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/info/rfc8174>.

[RFC8671] Evens, T., Bayraktar, S., Lucente, P., Mi, P., and S. Zhuang, "Support for Adj-RIB-Out in the BGP Monitoring Protocol (BMP)", RFC 8671, DOI 10.17487/RFC8671, November 2019, <https://www.rfc-editor.org/info/rfc8671>.

[RFC9069] Evens, T., Bayraktar, S., Bhardwaj, M., and P. Lucente, "Support for Local RIB in the BGP Monitoring Protocol (BMP)", RFC 9069, DOI 10.17487/RFC9069, February 2022, <https://www.rfc-editor.org/info/rfc9069>.

Authors' Addresses

John Scudder
Juniper Networks
1194 N. Mathilda Ave
Sunnyvale, CA 94089
United States of America
Email: jgs@juniper.net

Paolo Lucente
NTT
Veemweg 23
3771 Barneveld
Netherlands
Email: paolo@ntt.net

OAuth Working Group
Internet-Draft
Intended status: Standards Track
Expires: 20 March 2025

M.B. Jones
Self-Issued Consulting
P. Hunt
Independent Identity, Inc.
A. Parecki
Okta

16 September 2024

OAuth 2.0 Protected Resource Metadata
draft-ietf-oauth-resource-metadata-10

Abstract

This specification defines a metadata format that an OAuth 2.0 client or authorization server can use to obtain the information needed to interact with an OAuth 2.0 protected resource.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 20 March 2025.

Table of Contents

1. Introduction

This specification defines a metadata format enabling OAuth 2.0 clients and authorization servers to obtain information needed to interact with an OAuth 2.0 protected resource. The structure and content of this specification is intentionally as parallel as possible to that of "OAuth 2.0 Dynamic Client Registration Protocol" [RFC7591], which enables a client to provide metadata about itself to an OAuth 2.0 authorization server and to OAuth 2.0 Authorization Server Metadata [RFC8414], which enables a client to obtain metadata about an OAuth 2.0 authorization server.

The means by which the client obtains the location of the protected resource is out of scope of this document. In some cases, the location may be manually configured into the client; for example, an email client could provide an interface for a user to enter the URL of their JMAP [RFC8620] server. In other cases, it may be dynamically discovered; for example, a user could enter their email address into an email client, the client could perform WebFinger [RFC7033] discovery (in a manner related to the description in Section 2 of "OpenID Connect Discovery 1.0" [OpenID.Discovery]) to find the resource server, then fetch the resource server metadata to find the authorization server to use to obtain authorization to access the user's email.

The metadata for a protected resource is retrieved from a well-known location as a JSON [RFC8259] document, which declares information about its capabilities and optionally, its relationships to other services. This process is described in Section 3.

This metadata can either be communicated in a self-asserted fashion or as a set of signed metadata values represented as claims in a JSON Web Token (JWT) [JWT]. In the JWT case, the issuer is vouching for the validity of the data about the protected resource. This is analogous to the role that the Software Statement plays in OAuth Dynamic Client Registration [RFC7591].

Each protected resource publishing metadata about itself makes its own metadata document available at a well-known location deterministically derived from the protected resource's URL, even when the resource server implements multiple protected resources. This prevents attackers from publishing metadata supposedly describing the protected resource, but that is not actually authoritative for the protected resource, as described in Section 7.3.

Section 2 defines metadata values that a protected resource can publish, which includes things like which scopes are supported, how a client can present an access token, and more. These values may be used by other specifications, such as the jwks_uri used to publish public keys the resource server uses to sign resource responses, for instance, as described in [FAPI.MessageSigning].

1.1. Requirements Notation and Conventions

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.

All uses of JSON Web Signature (JWS) [JWS] and JSON Web Encryption (JWE) [JWE] data structures in this specification utilize the JWS Compact Serialization or the JWE Compact Serialization; the JWS JSON Serialization and the JWE JSON Serialization are not used.

1.2. Terminology

This specification uses the terms "Access Token", "Authorization Code", "Authorization Endpoint", "Authorization Grant", "Authorization Server", "Client", "Client Authentication", "Client Identifier", "Client Secret", "Grant Type", "Protected Resource", "Redirection URI", "Refresh Token", "Resource Owner", "Resource Server", "Response Type", and "Token Endpoint" defined by OAuth 2.0 [RFC6749], the terms "Claim Name", "Claim Value", and "JSON Web Token (JWT)" defined by JSON Web Token (JWT) [JWT].

This specification defines the following term:

Resource Identifier:
: The Protected resource’s resource identifier, which is a URL that uses the https scheme and has no query or fragment components. Protected resource metadata is published at a .well-known location [RFC8615] derived from this resource identifier, as described in Section 3.

2. Protected Resource Metadata

Protected resources can have metadata describing their configuration. The following protected resource metadata values are used by this specification and are registered in the IANA "OAuth Protected Resource Metadata" registry established in Section 8.1:

resource
: REQUIRED. The protected resource's Resource Identifier, as defined in Section 1.2.

authorization_servers
: OPTIONAL. JSON array containing a list of OAuth authorization server issuer identifiers, as defined in [RFC8414], for authorization servers that can be used with this protected resource. Protected resources MAY choose not to advertise some supported authorization servers even when this parameter is used. In some use cases, the set of authorization servers will not be enumerable, in which case this metadata parameter would not be used.

jwks_uri
: OPTIONAL. URL of the protected resource's JSON Web Key (JWK) Set [JWK] document. This contains public keys belonging to the protected resource, such as signing key(s) that the resource server uses to sign resource responses. This URL MUST use the https scheme. When both signing and encryption keys are made available, a use (public key use) parameter value is REQUIRED for all keys in the referenced JWK Set to indicate each key's intended usage.

scopes_supported
: RECOMMENDED. JSON array containing a list of the OAuth 2.0 [RFC6749] scope values that are used in authorization requests to request access to this protected resource. Protected resources MAY choose not to advertise some scope values supported even when this parameter is used.

bearer_methods_supported
: OPTIONAL. JSON array containing a list of the supported methods of sending an OAuth 2.0 Bearer Token [RFC6750] to the protected resource. Defined values are ["header", "body", "query"], corresponding to Sections 2.1, 2.2, and 2.3 of RFC 6750. The empty array [] can be used to indicate that no Bearer methods are supported. If this entry is omitted, no default Bearer methods supported are implied, nor does its absence indicate that they are not supported.

resource_signing_alg_values_supported
: OPTIONAL. JSON array containing a list of the JWS [JWS] signing algorithms (alg values) [JWA] supported by the protected resource for signing resource responses, for instance, as described in [FAPI.MessageSigning]. No default algorithms are implied if this entry is omitted. The value none MUST NOT be used.

resource_name
: Human-readable name of the protected resource intended for display to the end-user. It is RECOMMENDED that protected resource metadata includes this field. The value of this field MAY be internationalized, as described in Section 2.1.

resource_documentation
: OPTIONAL. URL of a page containing human-readable information that developers might want or need to know when using the protected resource. The value of this field MAY be internationalized, as described in Section 2.1.

resource_policy_uri
: OPTIONAL. URL of a page containing human-readable information about the protected resource's requirements on how the client can use the data provided by the protected resource. The value of this field MAY be internationalized, as described in Section 2.1.

resource_tos_uri
: OPTIONAL. URL of a page containing human-readable information about the protected resource's terms of service. The value of this field MAY be internationalized, as described in Section 2.1.

tls_client_certificate_bound_access_tokens
: OPTIONAL. Boolean value indicating protected resource support for mutual-TLS client certificate-bound access tokens [RFC8705]. If omitted, the default value is false.

authorization_details_types_supported
: OPTIONAL. A JSON array containing a list of the authorization details type values supported by the resource server when the authorization_details request parameter [RFC9396] is used.

dpop_signing_alg_values_supported
: OPTIONAL. A JSON array containing a list of the JWS alg values (from the "JSON Web Signature and Encryption Algorithms" registry [IANA.JOSE]) supported by the resource server for validating DPoP proof JWTs [RFC9449].

dpop_bound_access_tokens_required
: OPTIONAL. A boolean value specifying whether the protected resource always requires the use of DPoP-bound access tokens [RFC9449]. If omitted, the default value is false.

Additional protected resource metadata parameters MAY also be used.

2.1. Human-Readable Resource Metadata

Human-readable resource metadata values and resource metadata values that reference human-readable content MAY be represented in multiple languages and scripts. For example, the values of fields such as resource_name, resource_documentation, resource_tos_uri, and resource_policy_uri might have multiple locale-specific metadata values to facilitate use in different locations.

To specify the languages and scripts, BCP 47 [RFC5646] language tags are added to resource metadata parameter names, delimited by a # character. Since JSON [RFC8259] member names are case sensitive, it is RECOMMENDED that language tag values used in Claim Names be spelled using the character case with which they are registered in the "IANA Language Subtag" registry [IANA.Language]. In particular, normally language names are spelled with lowercase characters, region names are spelled with uppercase characters, and languages are spelled with mixed-case characters. However, since BCP 47 language tag values are case-insensitive, implementations SHOULD interpret the language tag values supplied in a case insensitive manner. Per the recommendations in BCP 47, language tag values used in metadata parameter names should only be as specific as necessary. For instance, using fr might be sufficient in many contexts, rather than fr-CA or fr-FR.

For example, a resource could represent its name in English as "resource_name#en": "My Resource" and its name in Italian as "resource_name#it": "La mia bella risorsa" within its metadata. Any or all of these names MAY be displayed to the end-user, choosing which names to display based on system configuration, user preferences, or other factors.

If any human-readable field is sent without a language tag, parties using it MUST NOT make any assumptions about the language, character set, or script of the string value, and the string value MUST be used as is wherever it is presented in a user interface. To facilitate interoperability, it is RECOMMENDED that each kind of human-readable metadata provided includes an instance of its metadata parameter without any language tags in addition to any language-specific parameters, and it is RECOMMENDED that any human-readable fields sent without language tags contain values suitable for display on a wide variety of systems.

2.2. Signed Protected Resource Metadata

In addition to JSON elements, metadata values MAY also be provided as a signed_metadata value, which is a JSON Web Token (JWT) [JWT] that asserts metadata values about the protected resource as a bundle. A set of claims that can be used in signed metadata are defined in Section 2. The signed metadata MUST be digitally signed or MACed using JSON Web Signature (JWS) [JWS] and MUST contain an iss (issuer) claim denoting the party attesting to the claims in the signed metadata. Consumers of the metadata MAY ignore the signed metadata if they do not support this feature. If the consumer of the metadata supports signed metadata, metadata values conveyed in the signed metadata MUST take precedence over the corresponding values conveyed using plain JSON elements.

Signed metadata is included in the protected resource metadata JSON object using this OPTIONAL metadata parameter:

signed_metadata
: A JWT containing metadata values about the protected resource as claims. This is a string value consisting of the entire signed JWT. A signed_metadata metadata value SHOULD NOT appear as a claim in the JWT.

3. Obtaining Protected Resource Metadata

Protected resources supporting metadata MUST make a JSON document containing metadata as specified in Section 2 available at a path formed by inserting a well-known URI string into the protected resource's resource identifier between the host component and the path component, if any. By default, the well-known URI string used is /.well-known/oauth-protected-resource. The syntax and semantics of .well-known are defined in [RFC8615]. The well-known URI path suffix used MUST be registered in the IANA "Well-Known URIs" registry [IANA.well-known]. Examples of this construction can be found in Section 3.1.

The term "application", as used below (and as used in [RFC8414]), encompasses all the components used to accomplish the task for the use case. That can include OAuth clients, authorization servers, protected resources, and non-OAuth components, inclusive of the code running in each of them. Applications are built to solve particular problems and may utilize many components and services.

Different applications utilizing OAuth protected resources in application-specific ways MAY define and register different well-known URI path suffixes for publishing protected resource metadata used by those applications. For instance, if the Example application

uses an OAuth protected resource in an Example-specific way, and there are Example-specific metadata values that it needs to publish, then it might register and use the example-protected-resource URI path suffix and publish the metadata document at the path formed by inserting /.well-known/example-protected-resource between the host and path components of the protected resource's resource identifier. Alternatively, many such applications will use the default well-known URI string /.well-known/oauth-protected-resource, which is the right choice for general-purpose OAuth protected resources, and not register an application-specific one.

An OAuth 2.0 application using this specification MUST specify what well-known URI suffix it will use for this purpose. The same protected resource MAY choose to publish its metadata at multiple well-known locations derived from its resource identifier, for example, publishing metadata at both /.well-known/example-protected-resource and /.well-known/oauth-protected-resource.

3.1. Protected Resource Metadata Request

A protected resource metadata document MUST be queried using an HTTP GET request at the previously specified path.

The consumer of the metadata would make the following request when the resource identifier is https://resource.example.com and the well-known URI path suffix is oauth-protected-resource to obtain the metadata, since the resource identifier contains no path component:

```
GET /.well-known/oauth-protected-resource HTTP/1.1
Host: resource.example.com
```

If the resource identifier value contains a path component, any terminating / MUST be removed before inserting /.well-known/ and the well-known URI path suffix between the host component and the path component. The consumer of the metadata would make the following request when the resource identifier is https://resource.example.com/resource1 and the well-known URI path suffix is oauth-protected-resource to obtain the metadata, since the resource identifier contains a path component:

```
GET /.well-known/oauth-protected-resource/resource1 HTTP/1.1
Host: resource.example.com
```

Using path components enables supporting multiple resources per host. This is required in some multi-tenant hosting configurations. This use of .well-known is for supporting multiple resources per host; unlike its use in [RFC8615], it does not provide general information about the host.

3.2. Protected Resource Metadata Response

The response is a set of claims about the protected resource's configuration. A successful response MUST use the 200 OK HTTP status code and return a JSON object using the application/json content type that contains a set of claims as its members that are a subset of the metadata values defined in Section 2. Other claims MAY also be returned.

Claims that return multiple values are represented as JSON arrays. Claims with zero elements MUST be omitted from the response.

An error response uses the applicable HTTP status code value.

The following is a non-normative example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
 "resource":
   "https://resource.example.com",
 "authorization_servers":
   ["https://as1.example.com",
    "https://as2.example.net"],
 "bearer_methods_supported":
   ["header", "body"],
 "scopes_supported":
   ["profile", "email", "phone"],
 "resource_documentation":
   "https://resource.example.com/resource_documentation.html"
}
```

3.3. Protected Resource Metadata Validation

The resource value returned MUST be identical to the protected resource's resource identifier value into which the well-known URI path suffix was inserted to create the URL used to retrieve the metadata. If these values are not identical, the data contained in the response MUST NOT be used.

If the protected resource metadata was retrieved from a URL returned by the protected resource via the WWW-Authenticate resource_metadata parameter, then the resource value returned MUST be identical to the URL that the client used to make the request to the resource server. If these values are not identical, the data contained in the response MUST NOT be used.

These validation actions can thwart impersonation attacks, as described in Section 7.3.

The recipient MUST validate that any signed metadata was signed by a key belonging to the issuer and that the signature is valid. If the signature does not validate or the issuer is not trusted, the recipient SHOULD treat this as an error condition.

4. Authorization Server Metadata

To support use cases in which the set of legitimate protected resources to use with the authorization server is enumerable, this specification defines the authorization server metadata value protected_resources, which enables the authorization server to explicitly list the protected resources. Note that if the set of legitimate authorization servers to use with a protected resource is also enumerable, lists in the authorization server metadata and protected resource metadata should be cross-checked against one another for consistency when these lists are used by the application profile.

The following authorization server metadata value is defined by this specification and is registered in the IANA "OAuth Authorization Server Metadata" registry established in OAuth 2.0 Authorization Server Metadata [RFC8414].

protected_resources
: OPTIONAL. JSON array containing a list of resource identifiers for OAuth protected resources for protected resources that can be used with this authorization server. Authorization servers MAY choose not to advertise some supported protected resources even when this parameter is used. In some use cases, the set of protected resources will not be enumerable, in which case this metadata parameter will not be present.

5. Use of WWW-Authenticate for Protected Resource Metadata

A protected resource MAY use a WWW-Authenticate response to return a URL to its protected resource metadata to the client. The client can then retrieve protected resource metadata as described in Section 3. The client might then, for instance, determine what authorization server to use for the resource based on protected resource metadata retrieved.

A typical end-to-end flow doing so is as follows. Note that while this example uses the OAuth 2.0 Authorization Code flow, a similar sequence could also be implemented with any other OAuth flow.

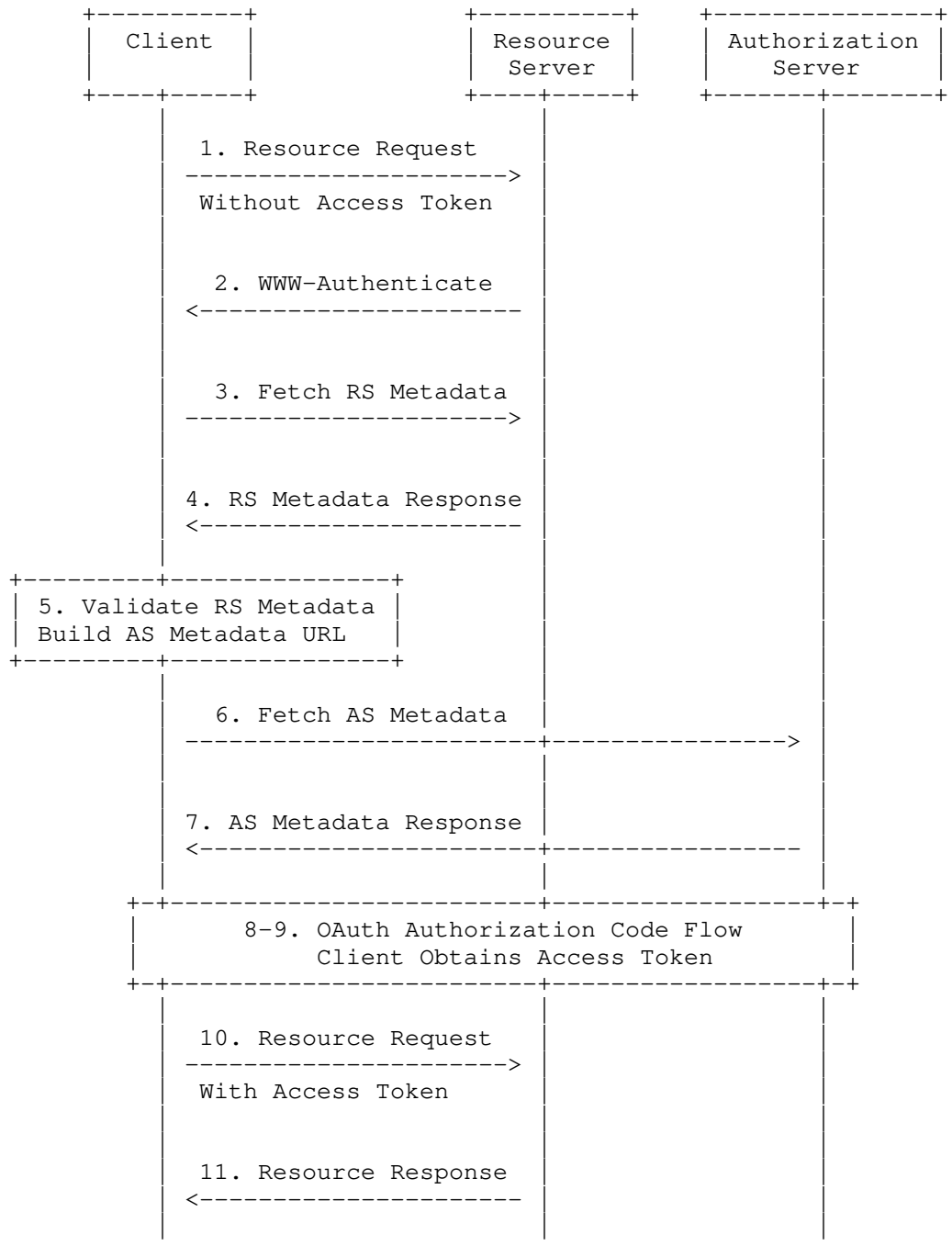


Figure 1: Sequence Diagram

1. The client makes a request to a protected resource without presenting an access token.

2. The resource server responds with a WWW-Authenticate header including the URL of the protected resource metadata.

3. The client fetches the protected resource metadata from this URL.

4. The resource server responds with the protected resource metadata according to Section 3.2.

5. The client validates the protected resource metadata, as described in Section 3.3.

6. The client builds the authorization server metadata URL from an issuer identifier in the resource metadata according to [RFC8414] and makes a request to fetch the authorization server metadata.

7. The authorization server responds with the authorization server metadata document according to [RFC8414].

8. The client directs the user agent to the authorization server to begin the authorization flow.

9. The authorization exchange is completed and the authorization server returns an access token to the client.

10. The client repeats the resource request from step 1, presenting the newly obtained access token.

11. The resource server returns the requested protected resource.

5.1. WWW-Authenticate Response

This specification introduces a new parameter in the WWW-Authenticate response to indicate the protected resource metadata URL:

resource_metadata:
: The URL of the protected resource metadata.

The response below is an example of a WWW-Authenticate header that includes the resource identifier.

```
HTTP/1.1 400 Bad Request
WWW-Authenticate: Bearer error="invalid_request",
  error_description="No access token was provided in this request",
  resource_metadata=
  "https://resource.example.com/.well-known/oauth-protected-resource"
```

The HTTP status code and error string in the response are defined by [RFC6750].

The resource_metadata parameter MAY be combined with other parameters defined in other extensions, such as the max_age parameter defined by [RFC9470].

5.2. Changes to Resource Metadata

At any point, for any reason determined by the protected resource, the protected resource MAY respond with a new WWW-Authenticate challenge that includes a value for the protected resource metadata URL to indicate that its metadata MAY have changed. If the client receives such a WWW-Authenticate response, it SHOULD retrieve the updated protected resource metadata and use the new metadata values obtained, after validating them as described in Section 3.3. Among other things, this enables a resource server to change which authorization servers it uses without any other coordination with clients.

5.3. Client Identifier and Client Authentication

The way in which the client identifier is established at the authorization server is out of scope of this specification.

This specification is intended to be deployed in scenarios where the client has no prior knowledge about the resource server, and the resource server might or might not have prior knowledge about the client.

There are some existing methods by which an unrecognized client can make use of an authorization server, such as using Dynamic Client Registration [RFC7591] to register the client prior to initiating the authorization flow. Future OAuth extensions might define alternatives, such as using URLs to identify clients.

5.4. Compatibility with Other Authentication Methods

Resource servers MAY return other WWW-Authenticate headers indicating various authentication schemes. This allows the resource server to support clients that may or may not implement this specification, and allows clients to choose their preferred authentication scheme.

6. String Operations

Processing some OAuth 2.0 messages requires comparing values in the messages to known values. For example, the member names in the metadata response might be compared to specific member names such as resource. Comparing Unicode [UNICODE] strings, however, has significant security implications.

Therefore, comparisons between JSON strings and other Unicode strings MUST be performed as specified below:

1. Remove any JSON applied escaping to produce an array of Unicode code points.

2. Unicode Normalization [USA15] MUST NOT be applied at any point to either the JSON string or to the string it is to be compared against.

3. Comparisons between the two strings MUST be performed as a Unicode code point to code point equality comparison.

Note that this is the same equality comparison procedure described in Section 8.3 of [RFC8259].

7. Security Considerations

7.1. TLS Requirements

Implementations MUST support TLS. Which version(s) ought to be implemented will vary over time, and depend on the widespread deployment and known security vulnerabilities at the time of implementation. Implementations SHOULD follow the guidance in BCP 195 [RFC8996] [RFC9325], which provides recommendations and requirements for improving the security of deployed services that use TLS.

To protect against information disclosure and tampering, confidentiality protection MUST be applied using TLS with a ciphersuite that provides confidentiality and integrity protection.

7.2. Scopes

The scopes_supported parameter is the list of scopes the resource server is willing to disclose that it supports. It is not meant to indicate that an OAuth client should request all scopes in the list. The client SHOULD still follow OAuth best practices and request tokens with as limited scope as possible for the given operation, as described in Section 2.3 of OAuth 2.0 Security Best Current Practice

[I-D.ietf-oauth-security-topics].

7.3. Impersonation Attacks

TLS certificate checking MUST be performed by the client, as described in Section 7.1, when making a protected resource metadata request. Checking that the server certificate is valid for the resource identifier URL prevents man-in-middle and DNS-based attacks. These attacks could cause a client to be tricked into using an attacker's resource server, which would enable impersonation of the legitimate protected resource. If an attacker can accomplish this, they can access the resources that the affected client has access to using the protected resource that they are impersonating.

An attacker may also attempt to impersonate a protected resource by publishing a metadata document that contains a resource claim using the resource identifier URL of the protected resource being impersonated, but containing information of the attacker's choosing. This would enable it to impersonate that protected resource, if accepted by the client. To prevent this, the client MUST ensure that the resource identifier URL it is using as the prefix for the metadata request exactly matches the value of the resource metadata value in the protected resource metadata document received by the client, as described in Section 3.3.

7.4. Audience-Restricted Access Tokens

If a client expects to interact with multiple resource servers, the client SHOULD request audience-restricted access tokens using [RFC8707], and the authorization server SHOULD support audience-restricted access tokens.

Without audience-restricted access tokens, a malicious resource server (RS1) may be able to use the WWW-Authenticate header to get a client to request an access token with a scope used by a legitimate resource server (RS2), and after the client sends a request to RS1, then RS1 could re-use the access token at RS2.

While this attack is not explicitly enabled by this specification, and is possible in a plain OAuth 2.0 deployment, it is made somewhat more likely by the use of dynamically-configured clients. As such, the use of audience-restricted access tokens and Resource Indicators [RFC8707] is RECOMMENDED when using the features in this specification.

7.5. Publishing Metadata in a Standard Format

Publishing information about the protected resource in a standard format makes it easier for both legitimate clients and attackers to use the protected resource. Whether a protected resource publishes its metadata in an ad-hoc manner or in the standard format defined by this specification, the same defenses against attacks that might be mounted that use this information should be applied.

7.6. Authorization Servers

To support use cases in which the set of legitimate authorization servers to use with the protected resource is enumerable, this specification defines the authorization_servers metadata value, which enables explicitly listing them. Note that if the set of legitimate protected resources to use with an authorization server is also enumerable, lists in the protected resource metadata and authorization server metadata should be cross-checked against one another for consistency when these lists are used by the application profile.

Secure determination of appropriate authorization servers to use with a protected resource for all use cases is out of scope of this specification. This specification assumes that the client has a means of determining appropriate authorization servers to use with a protected resource and that the client is using the correct metadata for each protected resource. Implementers need to be aware that if an inappropriate authorization server is used by the client, that an attacker may be able to act as a man-in-the-middle proxy to a valid authorization server without it being detected by the authorization server or the client.

The ways to determine the appropriate authorization servers to use with a protected resource are in general, application-dependent. For instance, some protected resources are used with a fixed authorization server or set of authorization servers, the locations of which may be well known, or which could be published as metadata values by the protected resource. In other cases, the set of authorization servers that can be used with a protected resource can by dynamically changed by administrative actions or by changes to the set of authorization servers adhering to a trust framework. Many other means of determining appropriate associations between protected resources and authorization servers are also possible.

7.7. Server-Side Request Forgery (SSRF)

The OAuth client is expected to fetch the authorization server metadata based on the value of the issuer in the resource server metadata. Since this specification enables clients to interoperate with RSs and ASs it has no prior knowledge of, this opens a risk for SSRF attacks by malicious users or malicious resource servers. Clients SHOULD take appropriate precautions against SSRF attacks, such as blocking requests to internal IP address ranges. Further recommendations can be found in the OWASP SSRF Prevention Cheat Sheet [OWASP.SSRF].

7.8. Phishing

This specification may be deployed in a scenario where the desired HTTP resource is identified by a user-selected URL. If this resource is malicious or compromised, it could mislead the user into revealing their account credentials or authorizing unwanted access to OAuth-controlled capabilities. This risk is reduced, but not eliminated, by following best practices for OAuth user interfaces, such as providing clear notice to the user, displaying the authorization server's domain name, supporting origin-bound phishing-resistant authenticators, supporting the use of password managers, and applying heuristic checks such as domain reputation.

7.9. Differences between Unsigned and Signed Metadata

Unsigned metadata is integrity protected by use of TLS at the site where it is hosted. This means that its security is dependent upon the Internet Public Key Infrastructure (PKI) [RFC9525]. Signed metadata is additionally integrity protected by the JWS signature applied by the issuer, which is not dependent upon the Internet PKI.

When using unsigned metadata, the party issuing the metadata is the protected resource itself, which is represented by the resource value in the metadata. Whereas, when using signed metadata, the party issuing the metadata is represented by the iss (issuer) claim in the signed metadata. When using signed metadata, applications can make trust decisions based on the issuer that performed the signing -- information that is not available when using unsigned metadata. How these trust decisions are made is out of scope for this specification.

7.10. Metadata Caching

Protected resource metadata is retrieved using an HTTP GET request, as specified in Section 3.1. Normal HTTP caching behaviors apply, meaning that the GET may retrieve a cached copy of the content, rather than the latest copy. Implementations should utlize HTTP caching directives such as Cache-Control with max-age, as defined in [RFC7234], to enable caching of retrieved metadata for appropriate time periods.

8. IANA Considerations

The following registration procedure is used for the registry established by this specification.

Values are registered on a Specification Required [RFC8126] basis after a two-week review period on the oauth-ext-review@ietf.org mailing list, on the advice of one or more Designated Experts. However, to allow for the allocation of values prior to publication, the Designated Experts may approve registration once they are satisfied that such a specification will be published.

Registration requests sent to the mailing list for review should use an appropriate subject (e.g., "Request to register OAuth Protected Resource Metadata: example").

Within the review period, the Designated Experts will either approve or deny the registration request, communicating this decision to the review list and IANA. Denials should include an explanation and, if applicable, suggestions as to how to make the request successful. Registration requests that are undetermined for a period longer than 21 days can be brought to the IESG's attention (using the iesg@ietf.org mailing list) for resolution.

Criteria that should be applied by the Designated Experts includes determining whether the proposed registration duplicates existing functionality, determining whether it is likely to be of general applicability or whether it is useful only for a single application, and whether the registration makes sense.

IANA must only accept registry updates from the Designated Experts and should direct all requests for registration to the review mailing list.

It is suggested that multiple Designated Experts be appointed who are able to represent the perspectives of different applications using this specification, in order to enable broadly-informed review of registration decisions. In cases where a registration decision could

be perceived as creating a conflict of interest for a particular Expert, that Expert should defer to the judgment of the other Experts.

8.1. OAuth Protected Resource Metadata Registry

This specification establishes the IANA "OAuth Protected Resource Metadata" registry for OAuth 2.0 protected resource metadata names. The registry records the protected resource metadata parameter and a reference to the specification that defines it.

8.1.1. Registration Template

Metadata Name:
: The name requested (e.g., "resource"). This name is case-sensitive. Names may not match other registered names in a case-insensitive manner unless the Designated Experts state that there is a compelling reason to allow an exception.

Metadata Description:
: Brief description of the metadata (e.g., "Resource identifier URL").

Change Controller:
: For Standards Track RFCs, list the "IETF". For others, give the name of the responsible party. Other details (e.g., postal address, email address, home page URI) may also be included.

Specification Document(s):
: Reference to the document or documents that specify the parameter, preferably including URIs that can be used to retrieve copies of the documents. An indication of the relevant sections may also be included but is not required.

8.1.2. Initial Registry Contents

* Metadata Name: resource
* Metadata Description: Protected resource’s resource identifier URL
* Change Controller: IETF
* Specification Document(s): Section 2 of [[this specification]]

* Metadata Name: authorization_servers
* Metadata Description: JSON array containing a list of OAuth authorization server issuer identifiers
* Change Controller: IETF
* Specification Document(s): Section 2 of [[this specification]]

* Metadata Name: jwks_uri

   * Metadata Description: URL of the protected resource's JWK Set document
   * Change Controller: IETF
   * Specification Document(s): Section 2 of [[this specification]]

   * Metadata Name: scopes_supported
   * Metadata Description: JSON array containing a list of the OAuth 2.0 scope values that are used in authorization requests to request access to this protected resource
   * Change Controller: IETF
   * Specification Document(s): Section 2 of [[this specification]]

   * Metadata Name: bearer_methods_supported
   * Metadata Description: JSON array containing a list of the OAuth 2.0 Bearer Token presentation methods that this protected resource supports
   * Change Controller: IETF
   * Specification Document(s): Section 2 of [[this specification]]

   * Metadata Name: resource_signing_alg_values_supported
   * Metadata Description: JSON array containing a list of the JWS signing algorithms (alg values) supported by the protected resource for signed content
   * Change Controller: IETF
   * Specification Document(s): Section 2 of [[this specification]]

   * Metadata Name: resource_name
   * Metadata Description: Human-readable name of the protected resource
   * Change Controller: IETF
   * Specification Document(s): Section 2 of [[this specification]]

   * Metadata Name: resource_documentation
   * Metadata Description: URL of a page containing human-readable information that developers might want or need to know when using the protected resource
   * Change Controller: IETF
   * Specification Document(s): Section 2 of [[this specification]]

   * Metadata Name: resource_policy_uri
   * Metadata Description: URL of a page containing human-readable information about the protected resource's requirements on how the client can use the data provided by the protected resource
   * Change Controller: IETF
   * Specification Document(s): Section 2 of [[this specification]]

   * Metadata Name: resource_tos_uri

* Metadata Description: URL of a page containing human-readable information about the protected resource's terms of service
* Change Controller: IETF
* Specification Document(s): Section 2 of [[this specification]]

* Metadata Name: tls_client_certificate_bound_access_tokens
* Metadata Description: Boolean value indicating protected resource support for mutual-TLS client certificate-bound access tokens
* Change Controller: IETF
* Specification Document(s): Section 2 of [[this specification]]

* Metadata Name: authorization_details_types_supported
* Metadata Description: JSON array containing a list of the authorization details type values supported by the resource server when the authorization_details request parameter is used
* Change Controller: IETF
* Specification Document(s): Section 2 of [[this specification]]

* Metadata Name: dpop_signing_alg_values_supported
* Metadata Description: JSON array containing a list of the JWS alg values supported by the resource server for validating DPoP proof JWTs
* Change Controller: IETF
* Specification Document(s): Section 2 of [[this specification]]

* Metadata Name: dpop_bound_access_tokens_required
* Metadata Description: Boolean value specifying whether the protected resource always requires the use of DPoP-bound access tokens
* Change Controller: IETF
* Specification Document(s): Section 2 of [[this specification]]

* Metadata Name: signed_metadata
* Metadata Description: Signed JWT containing metadata values about the protected resource as claims
* Change Controller: IETF
* Specification Document(s): Section 2.2 of [[this specification]]

8.2. OAuth Authorization Server Metadata Registry

The following authorization server metadata value is registered in the IANA "OAuth Authorization Server Metadata" registry established in OAuth 2.0 Authorization Server Metadata [RFC8414].

8.2.1. Registry Contents

* Metadata Name: protected_resources

* Metadata Description: JSON array containing a list of resource identifiers for OAuth protected resources
* Change Controller: IETF
* Specification Document(s): Section 4 of [[this specification]]

8.3. Well-Known URI Registry

This specification registers the well-known URI defined in Section 3 in the IANA "Well-Known URIs" registry [IANA.well-known].

8.3.1. Registry Contents

* URI suffix: oauth-protected-resource
* Change controller: IETF
* Specification document: Section 3 of [[this specification]]
* Related information: (none)

9. References

9.1. Normative References

[IANA.Language] IANA, "Language Subtag Registry", <https://www.iana.org/assignments/language-subtag-registry>.

[JWA] Jones, M.B., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <https://tools.ietf.org/html/rfc7518>.

[JWE] Jones, M.B. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, <https://tools.ietf.org/html/rfc7516>.

[JWK] Jones, M.B., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <https://tools.ietf.org/html/rfc7517>.

[JWS] Jones, M.B., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://tools.ietf.org/html/rfc7515>.

[JWT] Jones, M.B., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://tools.ietf.org/html/rfc7519>.

[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/info/rfc2119>.

[RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <https://www.rfc-editor.org/info/rfc5646>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, DOI 10.17487/RFC6750, October 2012, <https://www.rfc-editor.org/info/rfc6750>.

[RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <https://www.rfc-editor.org/info/rfc7234>.

[RFC7591] Richer, J., Ed., Jones, M., Bradley, J., Machulak, M., and P. Hunt, "OAuth 2.0 Dynamic Client Registration Protocol", RFC 7591, DOI 10.17487/RFC7591, July 2015, <https://www.rfc-editor.org/info/rfc7591>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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/info/rfc8174>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[RFC8414] Jones, M., Sakimura, N., and J. Bradley, "OAuth 2.0 Authorization Server Metadata", RFC 8414, DOI 10.17487/RFC8414, June 2018, <https://www.rfc-editor.org/info/rfc8414>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/info/rfc8615>.

[RFC8705] Campbell, B., Bradley, J., Sakimura, N., and T. Lodderstedt, "OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens", RFC 8705, DOI 10.17487/RFC8705, February 2020, <https://www.rfc-editor.org/info/rfc8705>.

[RFC8707] Campbell, B., Bradley, J., and H. Tschofenig, "Resource Indicators for OAuth 2.0", RFC 8707, DOI 10.17487/RFC8707, February 2020, <https://www.rfc-editor.org/info/rfc8707>.

[RFC8996] Moriarty, K. and S. Farrell, "Deprecating TLS 1.0 and TLS 1.1", BCP 195, RFC 8996, DOI 10.17487/RFC8996, March 2021, <https://www.rfc-editor.org/info/rfc8996>.

[RFC9325] Sheffer, Y., Saint-Andre, P., and T. Fossati, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 9325, DOI 10.17487/RFC9325, November 2022, <https://www.rfc-editor.org/info/rfc9325>.

[RFC9396] Lodderstedt, T., Richer, J., and B. Campbell, "OAuth 2.0 Rich Authorization Requests", RFC 9396, DOI 10.17487/RFC9396, May 2023, <https://www.rfc-editor.org/info/rfc9396>.

[RFC9449] Fett, D., Campbell, B., Bradley, J., Lodderstedt, T., Jones, M., and D. Waite, "OAuth 2.0 Demonstrating Proof of Possession (DPoP)", RFC 9449, DOI 10.17487/RFC9449, September 2023, <https://www.rfc-editor.org/info/rfc9449>.

[UNICODE] The Unicode Consortium, "The Unicode Standard", <https://www.unicode.org/versions/latest/>.

[USA15] Davis, M. and K. Whistler, "Unicode Normalization Forms", Unicode Standard Annex 15, 1 June 2015, <https://www.unicode.org/reports/tr15/>.

9.2. Informative References

[FAPI.MessageSigning] Tonge, D. and D. Fett, "FAPI 2.0 Message Signing", 24 March 2023, <https://openid.net/specs/fapi-2_0-message-signing.html>.

[I-D.ietf-oauth-security-topics]
Lodderstedt, T., Bradley, J., Labunets, A., and D. Fett, "OAuth 2.0 Security Best Current Practice", Work in Progress, Internet-Draft, draft-ietf-oauth-security-topics-29, 3 June 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-security-topics-29>.

[IANA.JOSE]
IANA, "JSON Object Signing and Encryption (JOSE)", <https://www.iana.org/assignments/jose>.

[IANA.well-known]
IANA, "Well-Known URIs", <https://www.iana.org/assignments/well-known-uris>.

[OpenID.Discovery]
Sakimura, N., Bradley, J., Jones, M.B., and E. Jay, "OpenID Connect Discovery 1.0", 15 December 2023, <https://openid.net/specs/openid-connect-discovery-1_0.html>.

[OWASP.SSRF]
OWASP, "OWASP SSRF Prevention Cheat Sheet", <https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html>.

[RFC7033] Jones, P., Salgueiro, G., Jones, M., and J. Smarr, "WebFinger", RFC 7033, DOI 10.17487/RFC7033, September 2013, <https://www.rfc-editor.org/info/rfc7033>.

[RFC8620] Jenkins, N. and C. Newman, "The JSON Meta Application Protocol (JMAP)", RFC 8620, DOI 10.17487/RFC8620, July 2019, <https://www.rfc-editor.org/info/rfc8620>.

[RFC9470] Bertocci, V. and B. Campbell, "OAuth 2.0 Step Up Authentication Challenge Protocol", RFC 9470, DOI 10.17487/RFC9470, September 2023, <https://www.rfc-editor.org/info/rfc9470>.

[RFC9525] Saint-Andre, P. and R. Salz, "Service Identity in TLS", RFC 9525, DOI 10.17487/RFC9525, November 2023, <https://www.rfc-editor.org/info/rfc9525>.

Appendix A. Acknowledgements

The authors of this specification would like to thank the attendees of the IETF 115 OAuth and HTTP API Working Group meetings and the attendees of subsequent OAuth Working Group meetings for their input on this specification. We would would also like to thank Ralph Bragg, Brian Campbell, Deb Cooley, Gabriel Corona, Vladimir Dzhuvinov, George Fletcher, Arnt Gulbrandsen, Pieter Kasselman, David Mandelberg, Tony Nadalin, Rifaat Shekh-Yusef, Filip Skokan, Atul Tulshibagwale, and Bo Wu for their contributions to the specification.

Appendix B. Document History

[[to be removed by the RFC Editor before publication as an RFC]]

-10

* Added metadata parameter declaring RAR types supported.

-09

* Added metadata values declaring support for DPoP and mutual-TLS client certificate-bound access tokens.

* Added missing word caught during IANA review.

* Addressed ART, SecDir, and OpsDir review comments by Arnt Gulbrandsen, David Mandelberg, and Bo Wu, resulting in the following changes.

* Added step numbers to sequence diagram.

* Defined meaning of omitting bearer_methods_supported metadata parameter.

* Added internationalization of human-readable metadata values using the mechanism from [RFC7591].

* Added resource_name metadata parameter, paralleling client_name in [RFC7591].

* Added Security Considerations section on metadata caching.

* Used and referenced Resource Identifier definition.

* Added motivating example of an email client to intro.

-08

* Added Security Considerations about the differences between unsigned and signed metadata, as suggested by Deb Cooley.

* Updated obsolete references.

-07

* Removed extraneous paragraph about downgrade attacks discussing an issue that's already addressed elsewhere in the specification.

-06

* Addressed shepherd review comments by Rifaat Shekh-Yusef.

-05

* Added SVG diagram

-04

* Applied working group last call suggestions by Atul Tulshibagwale.

* Better described the purpose of resource_signing_alg_values_supported and removed resource_encryption_alg_values_supported and resource_encryption_enc_values_supported, per WGLC comments by Vladimir Dzhuvinov and Brian Campbell.

* Applied suggestions by Pieter Kasselman.

-03

* Applied correction by Filip Skokan.

-02

* Switched from concatenating .well-known to the end of the resource identifier to inserting it between the host and path components of it.

* Have WWW-Authenticate return resource_metadata rather than resource.

-01

* Renamed scopes_provided to scopes_supported.

* Added security consideration for scopes_supported.

* Use BCP 195 for TLS recommendations.

* Clarified that resource metadata can be used by clients and authorization servers.

* Updated references.

* Added security consideration recommending audience-restricted access tokens.

* Mention FAPI Message Signing as a use case for publishing signing keys.

-00

* Initial working group version based on draft-jones-oauth-resource-metadata-04.

Authors' Addresses

Michael B. Jones
Self-Issued Consulting
Email: michael_b_jones@hotmail.com
URI: https://self-issued.info/

Phil Hunt
Independent Identity, Inc.
Email: phil.hunt@yahoo.com

Aaron Parecki
Okta
Email: aaron@parecki.com
URI: https://aaronparecki.com/

PIM Working Group
Internet Draft
Intended status: Informational
Expires: 25 March 2025

Y. Liu
China Mobile
M. McBride
Futurewei
Z. Zhang
ZTE
J. Xie
Huawei
C. Lin
New H3C Technologies
24 September 2024

Multicast-only Fast Reroute Based on Topology Independent Loop-free Alternate Fast Reroute
draft-ietf-pim-mofrr-tilfa-06

Status of this Memo

Copyright Notice

Abstract

This document specifies the use of Topology Independent Loop-Free Alternate (TI-LFA) mechanisms with Multicast Only Fast ReRoute (MoFRR) for Protocol Independent Multicast (PIM). TI-LFA provides fast reroute protection for unicast traffic in IP networks by precomputing backup paths that avoid potential failures. By integrating TI-LFA with MoFRR, this document extends the benefits of fast reroute mechanisms to multicast traffic, enabling enhanced resilience and minimized packet loss in multicast networks. The document outlines the necessary protocol extensions and operational considerations to implement TI-LFA in conjunction with MoFRR for PIM, ensuring that multicast traffic is rapidly rerouted in the event of a failure. This document uses the backup path computed by TI-LFA through IGP as a secondary Upstream Multicast Hop (UMH) for PIM. By using the TI-LFA backup path to send PIM secondary join messages hop-by-hop, it achieves the generation of a fast reroute backup path for PIM multicast.

Table of Contents

1. Introduction

The increasing deployment of video services has heightened the importance for network operators to implement solutions that minimize service disruptions caused by faults in the IP networks carrying these services. Multicast-only Fast Reroute (MoFRR), as defined in [RFC7431], offers a mechanism to reduce multicast packet loss in the event of node or link failures by introducing simple enhancements to multicast routing protocols, such as Protocol Independent Multicast (PIM). However, the current MoFRR mechanism, which selects the secondary multicast next hop based solely on the loop-free alternate fast reroute defined in [RFC7431], has limitations in certain multicast deployment scenarios.

This document introduces a new mechanism for Multicast-only Fast Reroute using Topology Independent Loop-Free Alternate (TI-LFA) [I-D.ietf-rtgwg-segment-routing-ti-lfa] fast reroute. Unlike traditional methods, TI-LFA is independent of network topology, enabling broader coverage across diverse network environments. The applicability of this mechanism extends to PIM networks, including scenarios where PIM operates over native IP, as well as public network multicast trees established by PIM. Additionally, this document addresses scenarios involving MDT SAFI for Multicast VPN (MVPN) in [RFC6037] and [RFC6514], covering PIM-SSM Tree, PIM-SM Tree, and BIDIR-PIM Tree deployments.

The TI-LFA mechanism is designed for standard link-state Interior Gateway Protocol (IGP) shortest path and Segment Routing (SR) scenarios. For each destination specified by the IGP in the network, TI-LFA pre-installs a backup forwarding entry for the protected destination, ready to be activated upon the detection of a link failure used to reach that destination. This document leverages the backup path computed by TI-LFA through the IGP as a secondary Upstream Multicast Hop (UMH) for PIM. By using the TI-LFA backup path to send PIM secondary join messages hop by hop, it enables the creation of a fast reroute backup path for PIM multicast.

The protection techniques described in this document are limited to protecting links and nodes within a link-state IGP area. Protecting domain exit routers and/or links attached to other routing domains is beyond the scope of this document. All the Segment Identifiers (SIDs) required are contained within the Link State Database (LSDB) of the IGP.

1.1. Requirements Language

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.

1.2. Terminology

This document utilizes the terminology as defined in [RFC7431] and incorporates the concepts established in [RFC7490]. The definitions of individual terms are not reiterated within this document.

2. Problem Statement

2.1. LFA for MoFRR

Section 3 of [RFC7431] specifies that the secondary UMH in PIM for MoFRR is a Loop-Free Alternate (LFA). However, the traditional LFA mechanism requires that at least one neighbor's next hop to the destination node is an loop-free next hop. Due to existing limitations in network deployments, this mechanism only covers certain network topology environments. In specific network topologies, the corresponding secondary UMH cannot be computed, preventing PIM from establishing a standby multicast tree and thus from implementing MoFRR protection. Consequently, the current MoFRR functionality in PIM is applicable only in network topologies where LFA is feasible.

The limitations of the current MoFRR applicability can be illustrated using the example network depicted in Figure 1. In this example, the metric of the R1-R4 link is 20, the metric of the R5-R6 link is 100, and the metrics of the other links are 10.

For multicast source S1, the primary path of the PIM join packet is R3->R2->R1, and the secondary path is R3->R4->R1, which corresponds to the LFA path of unicast routing. In this scenario, the current MoFRR operates effectively.

For multicast source S2, the primary path of the PIM join packet is R3->R2. However, an LFA does not exist. If R3 sends the packet to R4, R4 will forward it back to R3 because the IGP shortest path from R4 to R1 is R4->R3->R2. In this case, the current MoFRR cannot calculate a secondary UMH. Similarly, for multicast source S3, the current MoFRR mechanism is ineffective.

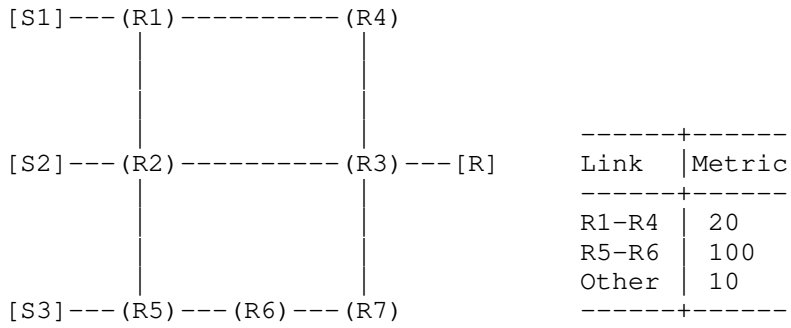


Figure 1: Example Network Topology

2.2. RLFA for MoFRR

The Remote Loop-Free Alternate (RLFA), as defined in [RFC7490], extends the traditional LFA mechanism, allowing it to accommodate a broader range of network deployment scenarios by utilizing a tunnel as an alternate path. The RLFA mechanism requires that there exists at least one node, denoted as node N, in the network where the fault node is neither on the path from the source node to node N nor on the path from node N to the destination node.

[RFC5496] introduces the RPF Vector attribute, which can be included in the PIM Join packet to ensure that the path is selected based on the unicast reachability of the RPF Vector. By combining the RLFA mechanism with the RPF Vector, the secondary multicast tree for MoFRR can be established, thereby supporting a wider range of network topologies compared to the current MoFRR implementation with LFA.

For instance, in the network illustrated in Figure 1, the secondary path for the PIM Join packet towards multicast source S2 cannot be computed by the current MoFRR, as previously described. Utilizing the RLFA mechanism, R3 sends the packet to R4, including an RPF Vector containing the IP address of R1, which serves as the PQ node of R3 with respect to the protected R2-R3 link. Subsequently, R4 forwards the packet to R1 via the R1-R4 link, in accordance with the unicast route associated with the RPF Vector. R1 then continues to forward the packet to R2, thereby establishing the secondary path, R3->R4->R1->R2, using MoFRR with RLFA.

2.3. TI-LFA for MoFRR

While RLFA provides enhanced capabilities over LFA, it remains dependent on the network topology. In the network depicted in Figure 1, the primary path of the PIM Join packet towards multicast source

S3 is R3->R2->R5. However, an RLFA path does not exist because the PQ node of R3 with respect to the protected link R2-R3 is absent. If R3 sends the packet to R7 with an RPF Vector containing the IP address of R5, R7 will forward it back to R3, as the IGP shortest path from R7 to R5 is R7->R3->R2->R5. Similarly, if R3 sends the packet to R7 with an RPF Vector containing the IP address of R6, R7 will forward it to R6, but R6 will then forward it back to R7, as the IGP shortest path from R6 to R5 is R6->R7->R3->R2->R5. In this scenario, MoFRR with RLFA is unable to compute a secondary UMH.

RLFA offers improvements over LFA but still has inherent limitations. [I-D.ietf-rtgwg-segment-routing-ti-lfa] defines a unicast FRR solution based on the TI-LFA mechanism. The TI-LFA mechanism allows the expression of a backup path using an explicit path and imposes no constraints on the network topology, thus providing a more robust FRR mechanism. Unicast traffic can be forwarded according to an explicit path list as an alternate path to protect unicast traffic, achieving full coverage across various network environments.

The alternate path provided by the TI-LFA mechanism is represented as a Segment List, which includes the NodeSID of the P-space node and the Adjacency SIDs of the links between the P-space and Q-space nodes. PIM can look up the corresponding node IP address in the unicast route according to the NodeSID and the IP addresses of the endpoints of the corresponding link in the unicast route according to the Adjacency SIDs. However, multicast protocol packets cannot be directly forwarded along the path of the Segment List.

To establish a standby multicast tree, PIM Join messages need to be transmitted hop-by-hop. However, not all nodes and links on the unicast alternate path are included in the Segment List. If PIM protocol packets are transmitted solely in unicast mode, they effectively traverse the unicast tunnel like unicast traffic and do not pass through the intermediate nodes of the tunnel. Consequently, the intermediate nodes on the alternate path cannot forward multicast traffic because they lack PIM state entries. PIM must create entries on each device hop-by-hop, generating an incoming interface and an outgoing interface list, to form a complete end-to-end multicast tree for forwarding multicast traffic. Therefore, simply sending PIM Join packets using the Segment List, as done with unicast traffic, is insufficient to establish a standby multicast tree.

3. Solution

A secondary Upstream Multicast Hop (UMH) serves as a candidate next-hop that can be used to reach the root of the multicast tree. In

this document, the secondary UMH is derived from unicast routing, utilizing the Segment List computed by TI-LFA.

In essence, the path information from the Segment List is incorporated into the PIM packets to guide hop-by-hop Reverse Path Forwarding (RPF) selection. The IP address corresponding to the Node SID can be used as the segmented root node, while the IP addresses of the interfaces at both endpoints of the link associated with the Adjacency SID can be directly used as the local upstream interface and upstream neighbor.

For the PIM protocol, [RFC5496] defines the PIM RPF Vector attribute, which can carry the node IP address corresponding to the Node SID. Additionally, [RFC7891] defines the explicit RPF Vector, which can carry the peer IP address corresponding to the Adjacency SID.

This document leverages the existing RPF Vector standards, obviating the need for PIM protocol extensions. This approach allows the establishment of a standby multicast tree based on the Segment List calculated by TI-LFA, thereby providing comprehensive MoFRR protection for multicast services across diverse network environments.

Consider a Segment List calculated by TI-LFA as (NodeSID(A), AdjSID(A-B)). Node A belongs to the P space, and node B belongs to the Q space. The IP address corresponding to NodeSID(A) can be retrieved from the local link-state database of the IGP protocol and assumed to be IP-a. Similarly, the IP addresses of the two endpoints of the link corresponding to AdjSID(A-B) can also be retrieved from the local link-state database and assumed to be IP-La and IP-Lb.

Within the PIM process, IP-a is treated as the standard RPF Vector Attribute and added to the PIM Join packet. IP-La is considered the local address of the incoming interface, and IP-Lb is regarded as the address of the upstream neighbor. Consequently, IP-Lb can be included in the PIM Join packet as the explicit RPF Vector Attribute.

The PIM protocol initially selects the RPF incoming interface and upstream neighbor towards IP-a and proceeds hop-by-hop to establish the PIM standby multicast tree until reaching node A. At node A, IP-Lb is treated as the PIM upstream neighbor. Node A identifies the incoming interface in the unicast routing table based on IP-Lb, and IP-Lb is used as the RPF upstream address for the PIM Join packet directed towards node B.

Upon receiving the PIM Join packet at node B, the PIM protocol, finding no additional RPF Vector Attributes, selects the RPF incoming interface and upstream neighbor towards the multicast source directly. The protocol then continues hop-by-hop to establish the PIM standby multicast tree, extending to the router directly connected to the source.

4. Illustration

This section provides an illustration of MoFRR based on TI-LFA. The example topology is depicted in Figure 2. The metric for the R3-R4 link is 100, while the metrics for the other links are 10. All link metrics are bidirectional.

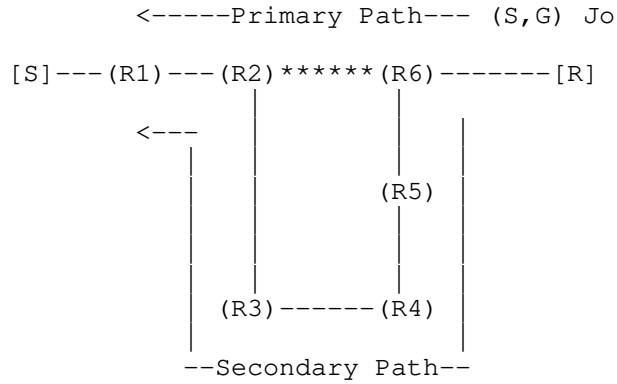


Figure 2: Example Topology

The IP addresses and SIDs involved in the MoFRR calculation are configured as follows:

IPv4 Data Plane (MPLS-SR)

Node	IP Address	Node SID
R4	IP4-R4	Label-R4

Link	IP Address	Adjacency SID
R3->R4	IP4-R3-R4	Label-R3-R4
R4->R3	IP4-R4-R3	Label-R4-R3

IPv6 Data Plane (SRv6)

Node	IP Address	Node SID (End)
R4	IP6-R4	SID-R4

Link	IP Address	Adjacency SID (End.X)
R3->R4	IP6-R3-R4	SID-R3-R4
R4->R3	IP6-R4-R3	SID-R4-R3

The primary path of the PIM Join packet is R6->R2->R1, and the secondary path is R6->R5->R4->R3->R2->R1. However, the traditional LFA does not function properly for the secondary path because the shortest path to R2 from R5 (or from R4) still traverses the R6-R2 link. In this scenario, R6 must calculate the secondary UMH using the proposed MoFRR method based on TI-LFA.

According to the TI-LFA algorithm, the P-space and Q-space are illustrated in Figure 3. The TI-LFA repair path consists of the Node SID of R4 and the Adjacency SID of R4->R3. On the MPLS-SR data plane, the repair segment list is (Label-R4, Label-R4-R3). On the SRv6 data plane, the repair segment list is (SID-R4, SID-R4-R3).

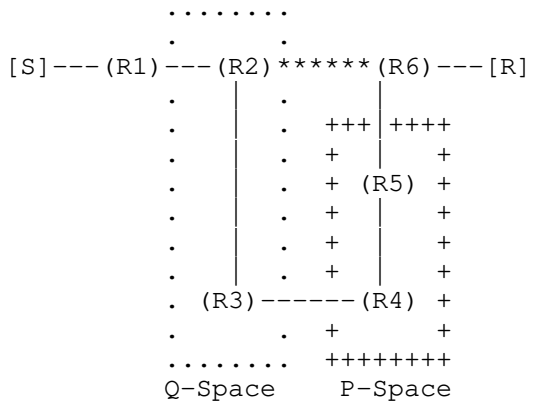


Figure 3: P-Space and Q-Space

In the PIM process, the IP addresses associated with the repair segment list are retrieved from the IGP link-state database.

On the IPv4 data plane, the Node SID Label-R4 corresponds to IP4-R4, which will be carried in the RPF Vector Attribute. The Adjacency SID Label-R4-R3 corresponds to the local address IP4-R4-R3 and the

remote peer address IP4-R3-R4, with IP4-R3-R4 carried in the Explicit RPF Vector Attribute.

On the IPv6 data plane, the End SID SID-R4 corresponds to IP6-R4, which will be carried in the RPF Vector Attribute. The End.X SID SID-R4-R3 corresponds to the local address IP6-R4-R3 and the remote peer address IP6-R3-R4, with IP6-R3-R4 carried in the Explicit RPF Vector Attribute.

Subsequently, R6 installs the secondary UMH using these RPF Vectors.

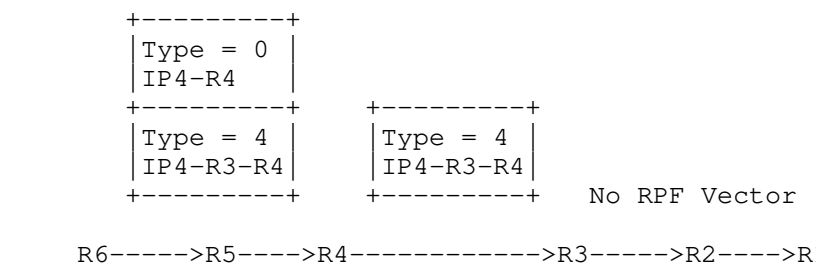


Figure 4: Forwarding PIM Join Packet along Secondary Path (IPv4)

On the IPv4 data plane, the forwarding of the PIM Join packet along the secondary path is shown in Figure 4.

R6 inserts two RPF Vector Attributes in the PIM Join packet: IP4-R4 of Type 0 (RPF Vector Attribute) and IP4-R3-R4 of Type 4 (Explicit RPF Vector Attribute). R6 then forwards the packet along the secondary path.

When R5 receives the packet, it performs a unicast route lookup of the first RPF Vector IP4-R4 and sends the packet to R4.

R4, being the owner of IP4-R4, removes the first RPF Vector from the packet and forwards it according to the next RPF Vector. R4 sends the packet to R3 based on the next RPF Vector IP4-R3-R4, as its PIM neighbor R3 corresponds to IP4-R3-R4.

When R3 receives the packet, as the owner of IP4-R3-R4, it removes the RPF Vector. The packet, now devoid of RPF Vectors, is forwarded to the source through R3->R2->R1 based on unicast routes.

After the PIM Join packet reaches R1, a secondary multicast tree, R1->R2->R3->R4->R5->R6, is established hop-by-hop for (S, G) using MoFRR based on TI-LFA.

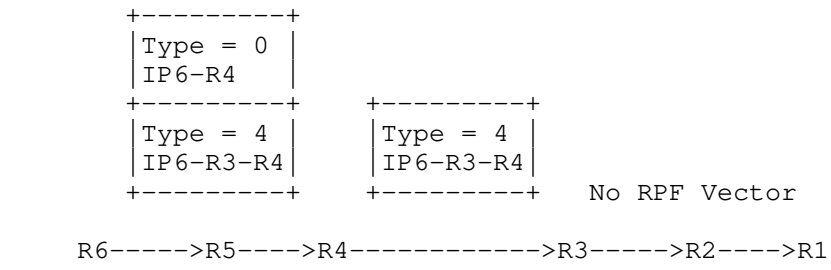


Figure 5: Forwarding PIM Join Packet along Secondary Path (IPv6)

On the IPv6 data plane, the forwarding of the PIM Join packet along the secondary path is illustrated in Figure 5. The procedure is analogous to that of the IPv4 data plane.

5. IANA Considerations

This document has no IANA actions.

6. Security Considerations

This document does not introduce additional security concerns. It does not change the security properties of PIM. For general PIM-SM protocol security considerations, see [RFC7761]. The security considerations of LFA, R-LFA, and MoFRR described in [RFC5286], [RFC7490], and [RFC7431] SHOULD apply to this document.

When deploying TILFA, packets may be sent over nodes and links they were not transported through before, potentially raising the following security issues:

1. Spoofing and False Route Advertisements

   * Dependencies of LFA/R-LFA/TI-LFA on Routing Information

     - LFAs depend on accurate routing information to determine alternate paths. If an attacker can inject false routing information (e.g., by spoofing link-state advertisements), it could cause the network to select suboptimal or malicious paths for LFAs.

     - R-LFA and TI-LFA also depend on accurate routing information, particularly for determining the tunneling paths or explicit paths. False route advertisements could mislead the network into using insecure or compromised paths.

2. Man-in-the-Middle (MitM) Attacks

   * Use of Alternate Paths

     - By rerouting traffic through alternate paths, especially those that traverse multiple hops (as in R-LFA and TI-LFA), the risk of MitM attacks increases if any of the intermediate routers on the alternate path are compromised.

     - TI-LFA, which uses explicit paths, might expose traffic to routers that were not originally part of the primary path, potentially allowing for interception or alteration of the traffic.

3. Confidentiality and Integrity

   * Traffic Encapsulation

     - R-LFA and TI-LFA involve encapsulating traffic, which may expose it to vulnerabilities if the encapsulation mechanisms are not secure. For instance, if IPsec or another secure encapsulation method is not used, an attacker might be able to intercept or alter the traffic in transit.

   * Protection of Explicit Paths

     - TI-LFA relies on explicit paths that are typically defined using segment routing. If these paths are not properly protected, an attacker could manipulate the segment list to reroute traffic through malicious nodes.

4. Increased Attack Surface

   * Extended Topology

     - By introducing LFA, R-LFA, and TI-LFA, the network increases its reliance on additional routers and links, thereby expanding the potential attack surface. Compromise of any router in these alternate paths could expose traffic to unauthorized access or disruption.

To address security issues #1 and #2 mentioned above, control plane protocols need to provide security protection. To mitigate the risks associated with false route advertisements and MitM attacks, it is recommended to use secure routing protocols (e.g., OSPFv3 with IPsec, ISIS HMAC-SHA256, or PIM with IPsec) that provide authentication and integrity protection for routing updates.

To address security issues #3 and #4 mentioned above, these mechanisms need to run within a trusted network. The security of LFA, R-LFA, and TI-LFA mechanisms heavily relies on the trustworthiness of the underlying routing infrastructure. As the solution described in the document is based on Segment Routing technology, readers should be aware of the security considerations related to this technology ([RFC8402]) and its data plane instantiations ([RFC8660], [RFC8754], and [RFC8986]).

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5286] Atlas, A., Ed., and A. Zinin, Ed., "Basic Specification for IP Fast Reroute: Loop-Free Alternates", RFC 5286,DOI 10.17487/RFC5286, September 2008,<http://www.rfc-editor.org/info/rfc5286>.

[RFC5496] Wijnands, IJ., Boers, A., and E. Rosen, "The Reverse Path Forwarding (RPF) Vector TLV", RFC 5496, March 2009.

[RFC7431] Karan, A., Filsfils, C., Wijnands, IJ., Ed., and B. Decraene, "Multicast-Only Fast Reroute", RFC 7431, August 2015.

[RFC7490] Bryant, S., Filsfils, C., Previdi, S., Shand, M., and N. So, "Remote Loop-Free Alternate (LFA) Fast Reroute (FRR)", RFC 7490, April 2015.

[RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I.,Parekh, R., Zhang, Z., and L. Zheng, "Protocol IndependentMulticast - Sparse Mode (PIM-SM): Protocol Specification(Revised)", RFC 7761, March 2016.

[RFC7891] Asghar, J., Wijnands, IJ., Ed., Krishnaswamy, S., Karan, A., and V. Arya, "Explicit Reverse Path Forwarding (RPF) Vector", RFC 7891, June 2016.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017.

[I-D.ietf-rtgwg-segment-routing-ti-lfa] Litkowski, S., Bashandy, A., Filsfils, C., Francois, P., Decraene, B., and D. Voyer, "Topology Independent Fast Reroute using Segment Routing", draft-ietf-rtgwg-segment-routing-ti-lfa-17, work-in-progress, July 2024.

7.2. Informative References

[RFC6037] Rosen, E., Cai, Y., Wijnands, I., "Cisco Systems' Solution for Multicast in BGP/MPLS IP VPNs", RFC6037, October 2010.

[RFC6514] Aggarwal, R., Rosen, E., Morin, T., and Y. Rekhter, "BGP Encodings and Procedures for Multicast in MPLS/BGP IP VPNs", RFC 6514, DOI 10.17487/RFC6514, February 2012, <https://www.rfc-editor.org/info/rfc6514>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402,July 2018, <https://www.rfc-editor.org/info/rfc8402>.

[RFC8660] Bashandy, A., Ed., Filsfils, C., Ed., Previdi, S.,Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with the MPLS Data Plane", RFC 8660,DOI 10.17487/RFC8660, December 2019,<https://www.rfc-editor.org/info/rfc8660>.

[RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J.,Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header(SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <https://www.rfc-editor.org/info/rfc8754>.

[RFC8986] Filsfils, C., Ed., Camarillo, P., Ed., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming", RFC 8986,DOI 10.17487/RFC8986, February 2021,<https://www.rfc-editor.org/info/rfc8986>.

Contributors

Mengxiao Chen
New H3C Technologies
China
Email: chen.mengxiao@h3c.com

Authors' Addresses

Yisong Liu
China Mobile
China
Email: liuyisong@chinamobile.com

Mike McBride
Futurewei Inc.
USA
Email: michael.mcbride@futurewei.com

Zheng(Sandy) Zhang
ZTE Corporation
China
Email: zzhang_ietf@hotmail.com

Jingrong Xie
Huawei Technologies
China
Email: xiejingrong@huawei.com

Changwang Lin
New H3C Technologies
China
Email: linchangwang.04414@h3c.com

RATS Working Group
Internet-Draft
Intended status: Standards Track
Expires: 5 January 2025

H. Birkholz
Fraunhofer SIT
J. O'Donoghue
Qualcomm Technologies Inc.
N. Cam-Winget
Cisco Systems
C. Bormann
UniversitÃ¤t Bremen TZI

4 July 2024

A CBOR Tag for Unprotected CWT Claims Sets
draft-ietf-rats-uccs-10

Abstract

When transported over secure channels, CBOR Web Token (CWT, RFC 8392) Claims Sets may not need the protection afforded by wrapping them into COSE, as is required for a true CWT. This specification defines a CBOR tag for such unprotected CWT Claims Sets (UCCS) and discusses conditions for its proper use.

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-rats-uccs/.

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

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

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

This Internet-Draft will expire on 5 January 2025.

Table of Contents

1. Introduction

A CBOR Web Token (CWT) as specified by [RFC8392] is always wrapped in a CBOR Object Signing and Encryption (COSE, [RFC9052]) envelope. COSE provides -- amongst other things -- end-to-end data origin authentication and integrity protection employed by RFC 8392 as well as optional encryption for CWTs. Under the right circumstances (Section 3), though, a signature providing proof for authenticity and integrity can be provided through the transfer protocol and thus omitted from the information in a CWT without compromising the intended goal of authenticity and integrity. In other words, if communicating parties have a pre-existing security association, they can reuse it to provide authenticity and integrity for their messages, enabling the basic principle of using resources parsimoniously. Specifically, if a mutually secured channel is established between two remote peers, and if that secure channel provides the required properties (as discussed below), it is possible to omit the protection provided by COSE, creating a use case for unprotected CWT Claims Sets. Similarly, if there is one-way authentication, the party that did not authenticate may be in a position to send authentication information through this channel that allows the already authenticated party to authenticate the other party; this effectively turns the channel into a mutually secured channel.

This specification allocates a CBOR tag to mark Unprotected CWT Claims Sets (UCCS) as such and discusses conditions for its proper use in the scope of Remote Attestation Procedures (RATS [RFC9334]) for the conveyance of RATS Conceptual Messages.

This specification does not change [RFC8392]: A true CWT does not make use of the tag allocated here; the UCCS tag is an alternative to using COSE protection and a CWT tag. Consequently, within the well-defined scope of a secure channel, it can be acceptable and economic to use the contents of a CWT without its COSE container and tag it with a UCCS CBOR tag for further processing within that scope -- or to use the contents of a UCCS CBOR tag for building a CWT to be signed by some entity that can vouch for those contents.

1.1. Terminology

The term Claim is used as in [RFC7519].

The terms Claim Key, Claim Value, and CWT Claims Set are used as in [RFC8392].

The terms Attester, Attesting Environment, Evidence, Relying Party and Verifier are used as in [RFC9334].

UCCS: Unprotected CWT Claims Set(s); CBOR map(s) of Claims as defined by the CWT Claims Registry that are composed of pairs of Claim Keys and Claim Values.

Secure Channel: [NIST-SP800-90Ar1] defines a Secure Channel as follows:

> "A path for transferring data between two entities or components that ensures confidentiality, integrity and replay protection, as well as mutual authentication between the entities or components. The secure channel may be provided using approved cryptographic, physical or procedural methods, or a combination thereof"

For the purposes of the present document, we focus on a protected communication channel used for conveyance that can ensure the same qualities as CWT without having the COSE protection available: mutual authentication, integrity protection, confidentiality. (Replay protection can be added by including a nonce claim such as Nonce (claim 10 [IANA.cwt]).) Examples include conveyance via PCIe (Peripheral Component Interconnect Express) IDE (Integrity and Data Encryption), or a TLS tunnel.

All terms referenced or defined in this section are capitalized in the remainder of this document.

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.

2. Deployment and Usage of UCCS

Usage scenarios involving the conveyance of Claims, in particular RATS, require a standardized data definition and encoding format that can be transferred and transported using different communication channels. As these are Claims, the Claims sets defined in [RFC8392] are a suitable format. However, the way these Claims are secured depends on the deployment, the security capabilities of the device, as well as their software stack. For example, a Claim may be securely stored and conveyed using a device's Trusted Execution Environment (TEE, see [RFC9397]) or a Trusted Platform Module (TPM, see [TPM2]). Especially in some resource constrained environments, the same process that provides the secure communication transport is

also the delegate to compose the Claim to be conveyed. Whether it is a transfer or transport, a Secure Channel is presumed to be used for conveying such UCCS. The following sections elaborate on Secure Channel characteristics in general and further describe RATS usage scenarios and corresponding requirements for UCCS deployment.

3. Characteristics of a Secure Channel

A Secure Channel for the conveyance of UCCS needs to provide the security properties that would otherwise be provided by COSE for a CWT. In this regard, UCCS is similar in security considerations to JWTs [RFC8725] using the algorithm "none". RFC 8725 states:

> [...] if a JWT is cryptographically protected end-to-end by a transport layer, such as TLS using cryptographically current algorithms, there may be no need to apply another layer of cryptographic protections to the JWT. In such cases, the use of the "none" algorithm can be perfectly acceptable.

The security considerations discussed, e.g., in Sections 2.1, 3.1, and 3.2 of [RFC8725] apply in an analogous way to the use of UCCS as elaborated on in this document.

Secure Channels are often set up in a handshake protocol that mutually derives a session key, where the handshake protocol establishes the (identity and thus) authenticity of one or both ends of the communication. The session key can then be used to provide confidentiality and integrity of the transfer of information inside the Secure Channel. (Where the handshake did not provide a mutually secure channel, further authentication information can be conveyed by the party not yet authenticated, leading to a mutually secured channel.) A well-known example of a such a Secure Channel setup protocol is the TLS [RFC8446] handshake; the TLS record protocol can then be used for secure conveyance.

As UCCS were initially created for use in RATS Secure Channels, the following section provides a discussion of their use in these channels. Where other environments are intended to be used to convey UCCS, similar considerations need to be documented before UCCS can be used.

4. UCCS in RATS Conceptual Message Conveyance

This section describes a detailed usage scenario for UCCS in the context of RATS in conjunction with its attendant security requirements. The use of UCCS tag CPA601 outside of the RATS context MUST come with additional instruction leaflets and security considerations.

For the purposes of this section, any RATS role can be the sender or the receiver of the UCCS.

Secure Channels can be transient in nature. For the purposes of this specification, the mechanisms used to establish a Secure Channel are out of scope.

In the scope of RATS Claims, the receiver MUST authenticate the sender as part of the establishment of the Secure Channel. Furthermore, the channel MUST provide integrity of the communication between the communicating RATS roles. For data confidentiality [RFC4949], the receiving side MUST be authenticated as well; this is achieved if the sender and receiver mutually authenticate when establishing the Secure Channel. The quality of the receiver's authentication and authorization will influence whether the sender can disclose the UCCS.

The extent to which a Secure Channel can provide assurances that UCCS originate from a trustworthy Attesting Environment depends on the characteristics of both the cryptographic mechanisms used to establish the channel and the characteristics of the Attesting Environment itself. The assurance provided to a Relying Party depends on the authenticity and integrity properties of the Secure Channel used for conveying the UCCS to it.

Ultimately, it is up to the receiver's policy to determine whether to accept a UCCS from the sender and to the type of Secure Channel it must negotiate. While the security considerations of the cryptographic algorithms used are similar to COSE, the considerations of the Secure Channel should also adhere to the policy configured at each of end of the Secure Channel. However, the policy controls and definitions are out of scope for this document.

Where an Attesting Environment serves as an endpoint of a Secure Channel used to convey a UCCS, the security assurance required of that Attesting Environment by a Relying Party generally calls for the Attesting Environment to be implemented using techniques designed to provide enhanced protection from an attacker wishing to tamper with or forge UCCS originating from that Attesting Environment. A possible approach might be to implement the Attesting Environment in a hardened environment such as a TEE [RFC9397] or a TPM [TPM2].

When UCCS emerge from the Secure Channel and into the receiver, the security properties of the secure channel no longer protect the UCCS, which now are subject to the same security properties as any other unprotected data in the Verifier environment. If the receiver subsequently forwards UCCS, they are treated as though they originated within the receiver.

The Secure Channel context does not govern fully formed CWTs in the same way it governs UCCS. As with EATs nested in other EATs (Section 4.2.18.3 (Nested Tokens) of [I-D.ietf-rats-eat]), the Secure Channel does not endorse fully formed CWTs transferred through it. Effectively, the COSE envelope of a CWT (or a nested EAT) shields the CWT Claims Set from the endorsement of the secure channel. (Note that EAT might add a nested UCCS Claim, and this statement does not apply to UCCS nested into UCCS, only to fully formed CWTs.)

5. Considerations for Using UCCS in Other RATS Contexts

This section discusses two additional usage scenarios for UCCS in the context of RATS.

5.1. Delegated Attestation

Another usage scenario is that of a sub-Attester that has no signing keys (for example, to keep the implementation complexity to a minimum) and has a Secure Channel, such as local inter-process communication, to interact with a lead Attester (see Composite Device, Section 3.3 of [RFC9334]). The sub-Attester produces a UCCS with the required CWT Claims Set and sends the UCCS through the Secure Channel to the lead Attester. The lead Attester then computes a cryptographic hash of the UCCS and protects that hash using its signing key for Evidence, for example, using a Detached-Submodule-Digest or Detached EAT Bundle (Section 5 of [I-D.ietf-rats-eat]).

5.2. Privacy Preservation

A Secure Channel which preserves the privacy of the Attester may provide security properties equivalent to COSE, but only inside the life-span of the session established. In general, when a privacy preserving Secure Channel is employed for conveying a conceptual message, the receiver cannot correlate the message with the senders of other received UCCS messages beyond the information the Secure Channel authentication provides.

An Attester must consider whether any UCCS it returns over a privacy preserving Secure Channel compromises the privacy in unacceptable ways. As an example, the use of the EAT UEID Claim Section 4.2.1 of [I-D.ietf-rats-eat] in UCCS over a privacy preserving Secure Channel allows a Verifier to correlate UCCS from a single Attesting Environment across many Secure Channel sessions. This may be acceptable in some use-cases (e.g., if the Attesting Environment is a physical sensor in a factory) and unacceptable in others (e.g., if the Attesting Environment is a user device belonging to a child).

6. IANA Considerations

6.1. CBOR Tag registration

In the CBOR Tags registry [IANA.cbor-tags] as defined in Section 9.2 of [RFC8949], IANA is requested to allocate the tag in Table 1 from the Specification Required space (1+2 size), with the present document as the specification reference.

Tag	Data Item	Semantics
CPA601	map (Claims-Set as per Appendix A of [RFCthis])	Unprotected CWT Claims Set [RFCthis]

Table 1: Values for Tags

// RFC-Editor: This document uses the CPA (code point allocation)
// convention described in [I-D.bormann-cbor-draft-numbers]. For
// each usage of the term "CPA", please remove the prefix "CPA" from
// the indicated value and replace the residue with the value
// assigned by IANA; perform an analogous substitution for all other
// occurrences of the prefix "CPA" in the document. Finally, please
// remove this note.

6.2. Media-Type application/uccs+cbor Registration

IANA is requested to add the following Media-Type to the "Media Types" registry [IANA.media-types].

Name	Template	Reference
uccs+cbor	application/uccs+cbor	Section 6.2 of RFCthis

Table 2: Media Type Registration

Type name: application
Subtype name: uccs+cbor
Required parameters: n/a
Optional parameters: n/a
Encoding considerations: binary (CBOR data item)
Security considerations: Section 7 of RFCthis
Interoperability considerations: none
Published specification: RFCthis

Applications that use this media type: Applications that transfer Unprotected CWT Claims Set(s) (UCCS) over Secure Channels
Fragment identifier considerations: The syntax and semantics of fragment identifiers is as specified for "application/cbor". (At publication of this document, there is no fragment identification syntax defined for "application/cbor".)
Additional information: Deprecated alias names for this type: N/A

Magic number(s): N/A

File extension(s): .uccs

Macintosh file type code(s): N/A
Person and email address to contact for further information: RATS WG mailing list (rats@ietf.org)
Intended usage: COMMON
Restrictions on usage: none
Author/Change controller: IETF

6.3. Content-Format registration

IANA is requested to register a Content-Format number in the "CoAP Content-Formats" subregistry, within the "Constrained RESTful Environments (CoRE) Parameters" registry [IANA.core-parameters], as follows:

Media Type	Encoding	ID	Reference
application/uccs+cbor	-	TBD601	Section 6.3 of RFCthis

Table 3: Content-Format Registration

7. Security Considerations

The security considerations of [RFC8949] apply. The security considerations of [RFC8392] need to be applied analogously, replacing the function of COSE with that of the Secure Channel; in particular "it is not only important to protect the CWT in transit but also to ensure that the recipient can authenticate the party that assembled the claims and created the CWT".

Section 3 discusses security considerations for Secure Channels, in which UCCS might be used. This document provides the CBOR tag definition for UCCS and a discussion on security consideration for the use of UCCS in RATS. Uses of UCCS outside the scope of RATS are

not covered by this document. The UCCS specification -- and the use of the UCCS CBOR tag, correspondingly -- is not intended for use in a scope where a scope-specific security consideration discussion has not been conducted, vetted and approved for that use. In order to be able to use the UCCS CBOR tag in another such scope, the secure channel and/or the application protocol (e.g., TLS and the protocol identified by ALPN) MUST specify the roles of the endpoints in a fashion that the security properties of conveying UCCS via a Secure Channel between the roles are well-defined.

7.1. General Considerations

Implementations of Secure Channels are often separate from the application logic that has security requirements on them. Similar security considerations to those described in [RFC9052] for obtaining the required levels of assurance include:

* Implementations need to provide sufficient protection for private or secret key material used to establish or protect the Secure Channel.

* Using a key for more than one algorithm can leak information about the key and is not recommended.

* An algorithm used to establish or protect the Secure Channel may have limits on the number of times that a key can be used without leaking information about the key.

* Evidence in a UCCS conveyed in a Secure Channel generally cannot be used to support trust in the credentials that were used to establish that secure channel, as this would create a circular dependency.

The Verifier needs to ensure that the management of key material used to establish or protect the Secure Channel is acceptable. This may include factors such as:

* Ensuring that any permissions associated with key ownership are respected in the establishment of the Secure Channel.

* Using cryptographic algorithms appropriately.

* Using key material in accordance with any usage restrictions such as freshness or algorithm restrictions.

* Ensuring that appropriate protections are in place to address potential traffic analysis attacks.

The remaining subsections of this section highlight some aspects of specific cryptography choices that are detailed further in [RFC9053].

7.2. AES-CBC_MAC

* A given key should only be used for messages of fixed or known length.

* Different keys should be used for authentication and encryption operations.

* A mechanism to ensure that IV cannot be modified is required.

Section 3.2.1 of [RFC9053] contains a detailed explanation of these considerations.

7.3. AES-GCM

* The key and nonce pair is unique for every encrypted message.

* The maximum number of messages to be encrypted for a given key is not exceeded.

Section 4.1.1 of [RFC9053] contains a detailed explanation of these considerations.

7.4. AES-CCM

* The key and nonce pair is unique for every encrypted message.

* The maximum number of messages to be encrypted for a given block cipher is not exceeded.

* The number of messages both successfully and unsuccessfully decrypted is used to determine when rekeying is required.

Section 4.2.1 of [RFC9053] contains a detailed explanation of these considerations.

7.5. ChaCha20 and Poly1305

* The nonce is unique for every encrypted message.

* The number of messages both successfully and unsuccessfully decrypted is used to determine when rekeying is required.

Section 4.3.1 of [RFC9053] contains a detailed explanation of these considerations.

8. References

8.1. Normative References

[IANA.cbor-tags]
IANA, "Concise Binary Object Representation (CBOR) Tags", <http://www.iana.org/assignments/cbor-tags>.

[IANA.cwt] IANA, "CBOR Web Token (CWT) Claims", <http://www.iana.org/assignments/cwt>.

[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>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/rfc/rfc7519>.

[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>.

[RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <https://www.rfc-editor.org/rfc/rfc8392>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/rfc/rfc8610>.

[RFC8725] Sheffer, Y., Hardt, D., and M. Jones, "JSON Web Token Best Current Practices", BCP 225, RFC 8725, DOI 10.17487/RFC8725, February 2020, <https://www.rfc-editor.org/rfc/rfc8725>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/rfc/rfc8949>.

[RFC9165] Bormann, C., "Additional Control Operators for the Concise Data Definition Language (CDDL)", RFC 9165, DOI 10.17487/RFC9165, December 2021, <https://www.rfc-editor.org/rfc/rfc9165>.

8.2. Informative References

[I-D.ietf-rats-eat]
Lundblade, L., Mandyam, G., O'Donoghue, J., and C. Wallace, "The Entity Attestation Token (EAT)", Work in Progress, Internet-Draft, draft-ietf-rats-eat-28, 25 June 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-eat-28>.

[IANA.core-parameters]
IANA, "Constrained RESTful Environments (CoRE) Parameters", <http://www.iana.org/assignments/core-parameters>.

[IANA.media-types]
IANA, "Media Types", <http://www.iana.org/assignments/media-types>.

[NIST-SP800-90Ar1]
Barker, E. and J. Kelsey, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators", National Institute of Standards and Technology, DOI 10.6028/nist.sp.800-90ar1, June 2015, <https://doi.org/10.6028/nist.sp.800-90ar1>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://www.rfc-editor.org/rfc/rfc4949>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC8747] Jones, M., Seitz, L., Selander, G., Erdtman, S., and H. Tschofenig, "Proof-of-Possession Key Semantics for CBOR Web Tokens (CWTs)", RFC 8747, DOI 10.17487/RFC8747, March 2020, <https://www.rfc-editor.org/rfc/rfc8747>.

[RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/rfc/rfc9052>.

[RFC9053] Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", RFC 9053, DOI 10.17487/RFC9053, August 2022, <https://www.rfc-editor.org/rfc/rfc9053>.

[RFC9334] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote ATtestation procedureS (RATS) Architecture", RFC 9334, DOI 10.17487/RFC9334, January 2023, <https://www.rfc-editor.org/rfc/rfc9334>.

[RFC9397] Pei, M., Tschofenig, H., Thaler, D., and D. Wheeler, "Trusted Execution Environment Provisioning (TEEP) Architecture", RFC 9397, DOI 10.17487/RFC9397, July 2023, <https://www.rfc-editor.org/rfc/rfc9397>.

[TPM2] "Trusted Platform Module Library Specification, Family â\200\2342.0â\200\235, Level 00, Revision 01.59 ed., Trusted Computing Group", 2019.

Appendix A. CDDL

This appendix is informative.

The Concise Data Definition Language (CDDL), as defined in [RFC8610] and [RFC9165], provides an easy and unambiguous way to express structures for protocol messages and data formats that use CBOR or JSON.

[RFC8392] does not define CDDL for CWT Claims Sets.

```
// RFC-Editor: This document uses the CPA (code point allocation)
// convention described in [I-D.bormann-cbor-draft-numbers].  Please
// replace the number 601 in the code blocks below by the value that
// has been assigned for CPA601 and remove this note.
```

In Figure 1, this specification shows how to use CDDL for defining the CWT Claims Set defined in [RFC8392]. Note that these CDDL rules have been built such that they also can describe [RFC7519] Claims sets by disabling feature "cbor" and enabling feature "json", but this flexibility is not the subject of the present specification.

```
UCCS-Untagged = Claims-Set
UCCS-Tagged = #6.601(UCCS-Untagged)

Claims-Set = {
 * $$Claims-Set-Claims
 * Claim-Label .feature "extended-claims-label" => any
}
Claim-Label = CBOR-ONLY<int> / text
string-or-uri = text

$$Claims-Set-Claims //= ( iss-claim-label => string-or-uri )
$$Claims-Set-Claims //= ( sub-claim-label => string-or-uri )
$$Claims-Set-Claims //= ( aud-claim-label => string-or-uri )
$$Claims-Set-Claims //= ( exp-claim-label => ~time )
$$Claims-Set-Claims //= ( nbf-claim-label => ~time )
$$Claims-Set-Claims //= ( iat-claim-label => ~time )
$$Claims-Set-Claims //= ( cti-claim-label => bytes )

iss-claim-label = JC<"iss", 1>
sub-claim-label = JC<"sub", 2>
aud-claim-label = JC<"aud", 3>
exp-claim-label = JC<"exp", 4>
nbf-claim-label = JC<"nbf", 5>
iat-claim-label = JC<"iat", 6>
cti-claim-label = CBOR-ONLY<7>  ; jti in JWT: different name and text

JSON-ONLY<J> = J .feature "json"
CBOR-ONLY<C> = C .feature "cbor"
JC<J,C> = JSON-ONLY<J> / CBOR-ONLY<C>
```

Figure 1: CDDL definition for Claims-Set

Specifications that define additional Claims should also supply additions to the $$Claims-Set-Claims socket, e.g.:

```
; [RFC8747]
$$Claims-Set-Claims //= ( 8: CWT-cnf ) ; cnf
CWT-cnf = {
  (1: CWT-COSE-Key) //
  (2: CWT-Encrypted_COSE_Key) //
  (3: CWT-kid)
}

CWT-COSE-Key = COSE_Key
CWT-Encrypted_COSE_Key = COSE_Encrypt / COSE_Encrypt0
CWT-kid = bytes

;;; Insert the required CDDL from RFC 9052 to complete these
;;; definitions.  This can be done manually or automated by a
;;; tool that implements an import directive such as:
;# import rfc9052
```

Appendix B. Example

This appendix is informative.

The example CWT Claims Set from Appendix A.1 of [RFC8392] can be turned into a UCCS by enclosing it with a tag number CPA601:

```
 601(
   {
     / iss / 1: "coap://as.example.com",
     / sub / 2: "erikw",
     / aud / 3: "coap://light.example.com",
     / exp / 4: 1444064944,
     / nbf / 5: 1443944944,
     / iat / 6: 1443944944,
     / cti / 7: h'0b71'
   }
 )
```

Appendix C. JSON Support

This appendix is informative.

The above definitions, concepts and security considerations all may be applied to define a JSON-encoded Claims-Set. Such an unsigned Claims-Set can be referred to as a "UJCS", an "Unprotected JWT Claims Set". The CDDL definition in Figure 1 can be used for a "UJCS".

```
UJCS = Claims-Set
```

Appendix D. EAT

This appendix is informative.

The following CDDL adds UCCS-format and UJCS-format tokens to EAT using its predefined extension points (see Section 4.2.18 (submods) of [I-D.ietf-rats-eat]).

```
$EAT-CBOR-Tagged-Token /= UCCS-Tagged
$EAT-CBOR-Untagged-Token /= UCCS-Untagged

$JSON-Selector /= [type: "UJCS", nested-token: UJCS]
```

Acknowledgements

Laurence Lundblade suggested some improvements to the CDDL. Carl Wallace provided a very useful review.

Authors' Addresses

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
64295 Darmstadt
Germany
Email: henk.birkholz@sit.fraunhofer.de

Jeremy O'Donoghue
Qualcomm Technologies Inc.
279 Farnborough Road
Farnborough
GU14 7LS
United Kingdom
Email: jodonogh@qti.qualcomm.com

Nancy Cam-Winget
Cisco Systems
3550 Cisco Way
San Jose, CA 95134
United States of America
Email: ncamwing@cisco.com

Carsten Bormann
UniversitÃ¤t Bremen TZI
Postfach 330440
D-28359 Bremen
Germany
Phone: +49-421-218-63921
Email: cabo@tzi.org

SIDROPS
Internet-Draft
Updates: 8182 (if approved)
Intended status: Standards Track
Expires: 28 January 2025

J. Snijders
Fastly

27 July 2024

Same-Origin Policy for the RPKI Repository Delta Protocol (RRDP) draft-ietf-sidrops-rrdp-same-origin-02

Abstract

This document describes a Same-Origin Policy (SOP) requirement for RPKI Repository Delta Protocol (RRDP) servers and clients. Application of SOP in RRDP client/server communication isolates resources such as Delta and Snapshot files from different Repository Servers, reducing possible attack vectors. This document updates RFC 8182.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 28 January 2025.

Copyright Notice

Table of Contents

1. Introduction

This document specifies a Same-origin policy (SOP) requirement for RPKI Repository Delta Protocol (RRDP) servers and clients. The SOP concept is a security mechanism to restrict how a document loaded from one origin can cause interaction with resources from another origin. See [RFC6454] for an overview of the concept of an "origin". Application of SOP in RRDP client/server communication isolates resources such as Delta and Snapshot files from different Repository Servers, reducing possible attack vectors. This document updates [RFC8182].

1.1. Requirements Language

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.

2. Implications of cross-origin resource requests in RRDP

The first RRDP protocol specification did not explicitly disallow 'cross-origin' URI references from the Update Notification file (Section 3.5.1 of [RFC8182]) towards Delta (Section 3.5.3 of [RFC8182]) and Snapshot (Section 3.5.2 of [RFC8182]) files, and was silent on the topic of HTTP Redirection (Section 15.4 of [RFC9110]).

The implication of cross-origin references in Update Notification files is that one Repository Server can reference RRDP resources on another Repository Server and in doing so inappropriately increase the resource consumption for both RRDP clients and the referenced Repository Server. An adversary could also employ cross-origin HTTP Redirects towards other Repository Servers, causing similar undesirable behavior.

3. Changes to RFC 8182

To overcome the aforementioned issue described in Section 2, RRDP Repository Servers and Clients MUST apply a Same-Origin Policy to both the URIs referenced in an Update Notification File and any HTTP Redirects.

3.1. New Requirements for RRDP Repository Servers

The following checklist items are added to Section 3.5.1.3 of [RFC8182]:

NEW

> * The uri attribute in the snapshot element and optional delta elements MUST be part of the same origin (i.e., represent the same principal), meaning referenced URIs MUST have the same scheme, host, and port as the URI for the Update Notification File specified in the referring RRDP SIA AccessDescription.
>
> * The Repository Server MUST NOT respond with HTTP Redirects towards locations with an origin different from the origin of the Update Notification File specified in the referring RRDP SIA AccessDescription.

3.2. New Requirements for Relying Parties using RRDP

The following adds to Section 3.4.1 of [RFC8182]:

NEW

* The Relying Party MUST verify whether the uri attributes in the Update Notification File are of the same origin as the Update Notification File itself. If this verification fails, the file MUST be rejected and RRDP cannot be used, see Section 3.4.5 of [RFC8182] for considerations.

* The Relying Party MUST NOT follow HTTP Redirection following from attempts to download Update Notification, Delta, and Snapshot files if the target origin is different from the origin of the Update Notification File specified in the referring RRDP SIA AccessDescription. If this verification fails, the RRDP session MUST be rejected and RRDP cannot be used, see Section 3.4.5 of [RFC8182] for considerations.

4. Deployability in the Internet's current RPKI

In the past 2.5 years no RRDP Repository Servers have employed cross-origin URIs in Update Notification Files.

At the moment of writing only one RRDP server (reached following the TALs of the five Regional Internet Registies) employs a same-origin HTTP redirect.

This means that imposing a requirement for the application of a Same-Origin Policy does not cause any existing commonly-used RRDP Repository Server operations to become non-compliant.

5. Security Considerations

This internet-draft patches an oversight in the original RRDP protocol specification: cross-origin requests are detrimental as they allow one repository operator to increase resource consumption for other repository operators and RRDP clients. Another way to avoid undesirable implications (as described in Section 2) would be for a future version of the RRDP protocol to use relative URIs instead of absolute URIs.

6. IANA Considerations

No IANA actions required.

7. References

7.1. 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/info/rfc2119>.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <https://www.rfc-editor.org/info/rfc6454>.

[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/info/rfc8174>.

[RFC8182] Bruijnzeels, T., Muravskiy, O., Weber, B., and R. Austein, "The RPKI Repository Delta Protocol (RRDP)", RFC 8182, DOI 10.17487/RFC8182, July 2017, <https://www.rfc-editor.org/info/rfc8182>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

7.2. Informative References

[FORT-validator]
Leiva, A., "FORT validator", <https://fortproject.net/en/validator>.

[Routinator]
NLNet Labs, "Routinator", <https://github.com/NLnetLabs/routinator/>.

[rpki-client]
Jeker, C., Snijders, J., Dzonsons, K., and T. Buehler, "rpki-client", <https://www.rpki-client.org/>.

[rpki-prover]
Puzanov, M., "rpki-prover", <https://github.com/lolepezy/rpki-prover>.

Appendix A. Acknowledgements

The author wishes to thank Theo Buehler, Claudio Jeker, Alberto Leiva, Tim Bruijnzeels, Ties de Kock, Martin Hoffmann, and Mikhail Puzanov for their helpful feedback, comments, and implementation work. The author wishes to thank Keyur Patel for their review.

Appendix B. Implementation status

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

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 RFC 7942. 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 RFC 7942, "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".

* OpenBSD's [rpki-client]

* Mikhail Puzanov's [rpki-prover]

* FORT project's [FORT-validator]

* NLNet Labs' [Routinator]

Author's Address

Job Snijders
Fastly
Amsterdam
Netherlands
Email: job@fastly.com