Network Working Group
Internet-Draft
Updates: 6613, 6614 (if approved)
Intended status: Experimental
Expires: April 21, 2016

S. Hartman
Painless Security

October 19, 2015

Larger Packets for RADIUS over TCP
draft-ietf-radext-bigger-packets-04.txt

Abstract

The RADIUS over TLS experiment described in RFC 6614 has opened RADIUS to new use cases where the 4096-octet maximum size limit of RADIUS packet proves problematic. This specification extends the RADIUS over TCP experiment (RFC 6613) to permit larger RADIUS packets. This specification compliments other ongoing work to permit fragmentation of RADIUS authorization information. This document registers a new RADIUS code, an action which requires IESG approval.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Remote Authentication DialIn User Service (RADIUS) over TLS [RFC6614] experiment provides strong confidentiality and integrity for RADIUS [RFC2865]. This enhanced security has opened new opportunities for using RADIUS to convey additional authorization information. As an example, [I-D.ietf-abfab-aaa-saml] describes a mechanism for using RADIUS to carry Security Assertion Markup Language (SAML) messages in RADIUS. Many attributes carried in these SAML messages will require confidentiality or integrity such as that provided by TLS.

These new use cases involve carrying additional information in RADIUS packets. The maximum packet length of 4096 octets is proving insufficient for some SAML messages and for other structures that may be carried in RADIUS.

One approach is to fragment a RADIUS message across multiple packets at the RADIUS layer. RADIUS Fragmentation [RFC7499] provides a mechanism to split authorization information across multiple RADIUS messages. That mechanism is necessary in order to split authorization information across existing unmodified proxies.

However, there are some significant disadvantages to RADIUS fragmentation. First, RADIUS is a lock-step protocol, and only one fragment can be in transit at a time as part of a given request. Also, there is no current mechanism to discover the path Maximum Transmission Unit (MTU) across the entire path that the fragment will travel. As a result, fragmentation is likely both at the RADIUS layer and at the transport layer. When TCP is used, much better transport characteristics can be achieved by fragmentation only at the TCP layer. This specification provides a mechanism to achieve these better transport characteristics when TCP is used. As part of this specification, a new RADIUS code is registered.

This specification is published as an experimental specification because the TCP extensions to RADIUS are currently experimental. The need for this specification arises from operational experience with the TCP extensions. However, this specification introduces no new experimental evaluation criteria beyond those in the base TCP specification; this specification can be evaluated along with that one for advancement on the standards track.

1.1. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Changes to Packet Processing

The maximum length of a RADIUS message is increased from 4096 to 65535. A RADIUS Server implementing this specification MUST be able to receive a packet of maximum length. Servers MAY have a maximum size over which they choose to return an error as discussed in Section 5 rather than processing a received packet; this size MUST be at least 4096 octets.

Clients implementing this specification MUST be able to receive a packet of maximum length; that is clients MUST NOT close a TCP connection simply because a large packet is sent over it. Clients MAY include the Response-Length attribute defined in Section 6 to indicate the maximum size of a packet that they can successfully process. Clients MAY silently discard a packet greater than some configured size; this size MUST be at least 4096 octets. Clients MUST NOT retransmit an unmodified request whose response is larger than the client can process as subsequent responses will likely continue to be too large.

Proxies MUST be able to receive a packet of maximum length without cloing the TCP connection. Proxies SHOULD be able to process and forward packets of maximum length. When a proxy receives a request over a transport with a 4096-octet maximum length and the proxy forwards that request over a transport with a larger maximum length, the proxy MUST include the Response-Length attribute with a value of 4096.

2.1. Status-Server Considerations

This section extends processing of Status-Server messages as described in section 4.1 and 4.2 of [RFC5997].

Clients implementing this specification SHOULD include the Response-Length attribute in Status-Server requests. Servers are already required to ignore unknown attributes received in this message. by including the attribute the client indicates how large of a response it can process to its Status-Server request. It is very unlikely that a response to Status-Server is greater than 4096 octets. However the client also indicates support for this specification which triggers server behavior below.

If a server implementing this specification receives a Response-Length attribute in a Status-Server request, it MUST include a Response-Length attribute indicating the maximum size request it can process in its response to the Status-Server request.

3. Forward and backward Compatibility

An implementation of [RFC6613] will silently discard any packet larger than 4096 octets and will close the TCP connection. This section provides guidelines for interoperability with these implementations. These guidelines are stated at the SHOULD level. In some environments support for large packets will be important enough that roaming or other agreements will mandate their support. In these environments, all implementations might be required to support this specification removing the need for interoperability with RFC 6613. It is likely that these guidelines will be relaxed to the MAY level and support for this specification made a requirement if RADIUS over TLS and TCP are moved to the standards track in the future.

Clients SHOULD provide configuration for the maximum size of a request sent to each server. Servers SHOULD provide configuration for the maximum size of a response sent to each client. If dynamic discovery mechanisms are supported, configuration SHOULD be provided for the maximum size of clients and servers in each dynamic discovery category.

If a client sends a request larger than 4096 octets and the TCP connection is closed without a response, the client SHOULD treat the request as if a request too big error (Section 5) specifying a maximum size of 4096 is received. Clients or proxies sending multiple requests over a single TCP connection without waiting for responses SHOULD implement capability discovery as discussed in Section 3.2.

By default, a server SHOULD not generate a response larger than 4096 octets. The Response-Length attribute MAY be included in a request to indicate that larger responses are acceptable. Other attributes or configuration MAY be used as an indicator that large responses are likely to be acceptable.

A proxy that implements both this specification and RADIUS Fragmentation [RFC7499] SHOULD use RADIUS fragmentation when the following conditions are met:

1. A packet is being forwarded towards an next hop whose configuration does not support a packet that large.

2. RADIUS Fragmentation can be used for the packet in question.

3.1. Rationale

The interoperability challenge appears at first significant. This specification proposes to introduce behavior where new implementations will fail to function with existing implementations.

However, these capabilities are introduced to support new use cases. If an implementation has 10000 octets of attributes to send, it cannot in general trim down the response to something that can be sent. Under this specification a large packet would be generated that will be silently discarded by an existing implementation. Without this specification, no packet is generated because the required attributes cannot be sent.

The biggest risk to interoperability would be if requests and responses are expanded to include additional information that is not strictly necessary. So, avoiding creating situations where large packets are sent to existing implementations is mostly an operational matter. Interoperability is most impacted when the size of packets in existing use cases is significantly increased and least impacted when large packets are used for new use cases where the deployment is likely to require updated RADIUS implementations.

There is a special challenge for proxies or clients with high request volume. When an RFC 6613 implementation receives a packet that is too large, it closes the connection and does not respond to any requests in process. Such a client would lose requests and might find distinguishing request-too-big situations from other failures difficult. In these cases, the discovery mechanism described in Section 3.2 can be used.

Also, RFC 6613 is an experiment. Part of running that experiment is to evaluate whether additional changes are required to RADIUS. A lower bar for interoperability should apply to changes to experimental protocols than standard protocols.

This specification provides good facilities to enable implementations to understand packet size when proxying to/from standards-track UDP RADIUS.

3.2. Discovery

As discussed in Section 2.1, a client MAY send a Status-Server message to discover whether an authentication or accounting server supports this specification. The client includes a Response-Length attribute; this signals the server to include a Response-Length attribute indicating the maximum packet size the server can process. In this one instance, Response-Length indicate the size of a request that can be processed rather than a response.

4. Protocol-Error Code

This document defines a new RADIUS code, TBDCODE (IANA), called Protocol-Error. This packet code may be used in response to any request packet, such as Access-Request, Accounting-Request, CoA-Request, or Disconnect-Request. It is a response packet sent by a server to a client. The packet indicates to the client that the server is unable to process the request for some reason.

A Protocol-Error packet MUST contain a Original-Packet-Code attribute, along with an Error-Cause attribute. Other attributes MAY be included if desired. The Original-Packet-Code contains the code from the request that generated the protocol error so that clients can disambiguate requests with different codes and the same ID. Regardless of the original packet code, the RADIUS server calculates the Message-Authenticator attribute as if the original packet were an Access-Request packet. The identifier is copied from the original request.

Clients processing Protocol-Error MUST ignore unknown or unexpected attributes.

This RADIUS code is hop-by-hop. Proxies MUST NOT forward a Protocol-Error packet they receive.

5. Too Big Response

When a RADIUS server receives a request that is larger than can be processed, it generates a Protocol-Error response as follows:

The code is Protocol-Error.

The Response-Length attribute MUST be included and its value is the maximum size of request that will be processed.

The Error-Cause attribute is included with a value of TOOBIGTBD.

The Original-Packet-Code attribute is copied from the request.

Clients will not typically be able to adjust and resend requests when this error is received. In some cases the client can fall back to RADIUS Fragmentation. In other cases this code will provide for better client error reporting and will avoid retransmitting requests guaranteed to fail.

6. IANA Considerations

A new RADIUS packet type code is registered in the RADIUS packet type codes registry discussed in section 2.1 of RFC 3575 [RFC3575]. The name is "Protocol-Error" and the code is TBDCODE. The IESG is requested to approve this registration along with approving publication of this document.

The following RADIUS attribute type values [RFC3575] are assigned. The assignment rules in section 10.3 of [RFC6929] are used.

Name	Attribute	Description
Response-Length	TBD	2-octet unsigned integer maximum response length
Original-Packet-Code	TBD2	An integer attribute containing the code from a packet resulting in a Protocol-Error response.

The Response-Length attribute MAY be included in any RADIUS request. In this context it indicates the maximum length of a response the client is prepared to receive. Values are between 4096 and 65535. The attribute MAY also be included in a response to a Status-Server message. In this case the attribute indicates the maximum size RADIUS request that is permitted.

A new Error-Cause value is registered in the registry at http://www.iana.org/assignments/radius-types/radius-types.xhtml#radius-types-18 for "Response Too Big" with value TOOBIGTBD.

7. Security Considerations

This specification updates RFC 6613 and will be used with [RFC6614]. When used over plain TCP, this specification creates new opportunities for an on-path attacker to impact availability. These attacks can be entirely mitigated by using TLS. If these attacks ar acceptable, then this specification can be used over TCP.

8. Acknowledgements

Sam Hartman’s time on this draft was funded by JANET as part of Project Moonshot.

Alan DeKok provided valuable review and text for the Protocol-Error packet code.

Alejandro Perez Mendez provided valuable review comments.

9. References

9.1. Normative References

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

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC3575] Aboba, B., "IANA Considerations for RADIUS (Remote Authentication Dial In User Service)", RFC 3575, July 2003.

[RFC5997] DeKok, A., "Use of Status-Server Packets in the Remote Authentication Dial In User Service (RADIUS) Protocol", RFC 5997, August 2010.

[RFC6613] DeKok, A., "RADIUS over TCP", RFC 6613, May 2012.

[RFC6614] Winter, S., McCauley, M., Venaas, S., and K. Wierenga, "Transport Layer Security (TLS) Encryption for RADIUS", RFC 6614, May 2012.

[RFC6929] DeKok, A. and A. Lior, "Remote Authentication Dial In User Service (RADIUS) Protocol Extensions", RFC 6929, April 2013.

9.2. Informative References

[I-D.ietf-abfab-aaa-saml]
Howlett, J. and S. Hartman, "A RADIUS Attribute, Binding, Profiles, Name Identifier Format, and Confirmation Methods for SAML", draft-ietf-abfab-aaa-saml-09 (work in progress), February 2014.

[RFC7499] Perez-Mendez, A., Ed., Marin-Lopez, R., Pereniguez-Garcia, F., Lopez-Millan, G., Lopez, D., and A. DeKok, "Support of Fragmentation of RADIUS Packets", RFC 7499, DOI 10.17487/RFC7499, April 2015, <http://www.rfc-editor.org/info/rfc7499>.

Author’s Address

Sam Hartman
Painless Security

Email: hartmans-ietf@mit.edu

Network Working Group   S. Hartman
Internet-Draft   Painless Security
Updates: 6613 (if approved)   April 12, 2016
Intended status: Experimental
Expires: October 14, 2016

Larger Packets for RADIUS over TCP
draft-ietf-radext-bigger-packets-07.txt

Abstract

The RADIUS over TLS experiment described in RFC 6614 has opened RADIUS to new use cases where the 4096-octet maximum size limit of RADIUS packet proves problematic. This specification extends the RADIUS over TCP experiment (RFC 6613) to permit larger RADIUS packets. This specification compliments other ongoing work to permit fragmentation of RADIUS authorization information. This document registers a new RADIUS code, an action which requires IESG approval.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Remote Authentication Dial In User Service (RADIUS) over TLS [RFC6614] experiment provides strong confidentiality and integrity for RADIUS [RFC2865]. This enhanced security has opened new opportunities for using RADIUS to convey additional authorization information. As an example, [I-D.ietf-abfab-aaa-saml] describes a mechanism for using RADIUS to carry Security Assertion Markup Language (SAML) messages in RADIUS. Many attributes carried in these SAML messages will require confidentiality or integrity such as that provided by TLS.

These new use cases involve carrying additional information in RADIUS packets. The maximum packet length of 4096 octets is proving insufficient for some SAML messages and for other structures that may be carried in RADIUS.

One approach is to fragment a RADIUS message across multiple packets at the RADIUS layer. RADIUS Fragmentation [RFC7499] provides a mechanism to split authorization information across multiple RADIUS messages. That mechanism is necessary in order to split authorization information across existing unmodified proxies.

However, there are some significant disadvantages to RADIUS fragmentation. First, RADIUS is a lock-step protocol, and only one fragment can be in transit at a time as part of a given request. Also, there is no current mechanism to discover the path Maximum Transmission Unit (MTU) across the entire path that the fragment will travel. As a result, fragmentation is likely both at the RADIUS layer and at the transport layer. When TCP is used, much better transport characteristics can be achieved by fragmentation only at the TCP layer. This specification provides a mechanism to achieve these better transport characteristics when TCP is used. As part of this specification, a new RADIUS code is registered.

This specification is published as an experimental specification because the TCP extensions to RADIUS are currently experimental. The need for this specification arises from operational experience with the TCP extensions. However, this specification introduces no new experimental evaluation criteria beyond those in the base TCP specification; this specification can be evaluated along with that one for advancement on the standards track.

1.1. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Changes to Packet Processing

The maximum length of a RADIUS message is increased from 4096 to 65535. A RADIUS Server implementing this specification MUST be able to receive a RADIUS packet of maximum length. Servers MAY have a maximum size over which they choose to return an error as discussed in Section 5 rather than processing a received packet; this size MUST be at least 4096 octets.

Clients implementing this specification MUST be able to receive a RADIUS packet of maximum length; that is clients MUST NOT close a TCP connection simply because a large packet is sent over it. Clients MAY include the Response-Length attribute defined in Section 6 to indicate the maximum size of a packet that they can successfully process. Clients MAY silently discard a packet greater than some configured size; this size MUST be at least 4096 octets. Clients MUST NOT retransmit an unmodified request whose response is larger than the client can process as subsequent responses will likely continue to be too large.

Proxies MUST be able to receive a RADIUS packet of maximum length without closing the TCP connection. Proxies SHOULD be able to process and forward packets of maximum length. When a proxy receives a request over a transport with a 4096-octet maximum length and the proxy forwards that request over a transport with a larger maximum length, the proxy MUST include the Response-Length attribute with a value of 4096.

2.1. Status-Server Considerations

This section extends processing of Status-Server messages as described in section 4.1 and 4.2 of [RFC5997].

Clients implementing this specification SHOULD include the Response-Length attribute in Status-Server requests. Servers are already required to ignore unknown attributes received in this message. by including the attribute, the client indicates how large of a response it can process to its Status-Server request. It is very unlikely that a response to Status-Server is greater than 4096 octets. However the client also indicates support for this specification which triggers server behavior below.

If a server implementing this specification receives a Response-Length attribute in a Status-Server request, it MUST include a Response-Length attribute indicating the maximum size request it can process in its response to the Status-Server request.

3. Forward and backward Compatibility

An implementation of [RFC6613] will silently discard any RADIUS packet larger than 4096 octets and will close the TCP connection. This section provides guidelines for interoperability with these implementations. These guidelines are stated at the SHOULD level. In some environments support for large packets will be important enough that roaming or other agreements will mandate their support. In these environments, all implementations might be required to support this specification removing the need for interoperability with RFC 6613. It is likely that these guidelines will be relaxed to the MAY level and support for this specification made a requirement if RADIUS over TLS and TCP are moved to the standards track in the future.

Clients SHOULD provide configuration for the maximum size of a request sent to each server. Servers SHOULD provide configuration for the maximum size of a response sent to each client. If dynamic discovery mechanisms are supported, configuration SHOULD be provided for the default maximum size of RADIUS packets sent to clients and servers. If an implementation provides more granular configuration for some classes of dynamic resources, then the implementation SHOULD also provide configuration of default maximum packet sizes at the same granularity. As an example, an implementation that provided granular configuration for resources using a particular trust anchor or belonging to a particular roaming consortium SHOULD provide default packet size configuration at the same granularity.

If a client sends a request larger than 4096 octets and the TCP connection is closed without a response, the client SHOULD treat the request as if a request too big error (Section 5) specifying a maximum size of 4096 is received. Clients or proxies sending multiple requests over a single TCP connection without waiting for responses SHOULD implement capability discovery as discussed in Section 3.2.

By default, a server SHOULD NOT generate a response larger than 4096 octets. The Response-Length attribute MAY be included in a request to indicate that larger responses are acceptable. Other attributes or configuration MAY be used as an indicator that large responses are likely to be acceptable.

A proxy that implements both this specification and RADIUS Fragmentation [RFC7499] SHOULD use RADIUS fragmentation when the following conditions are met:

1. A RADIUS packet is being forwarded towards a next hop whose configuration does not support a packet that large.

2. RADIUS Fragmentation can be used for the packet in question.

3.1. Rationale

The interoperability challenge appears at first significant. This specification proposes to introduce behavior where new implementations will fail to function with existing implementations.

However, these capabilities are introduced to support new use cases. If an implementation has 10000 octets of attributes to send, it cannot in general trim down the response to something that can be sent. Under this specification a large packet would be generated that will be silently discarded by an existing implementation. Without this specification, no packet is generated because the required attributes cannot be sent.

The biggest risk to interoperability would be if requests and responses are expanded to include additional information that is not strictly necessary. So, avoiding creating situations where large packets are sent to existing implementations is mostly an operational matter. Interoperability is most impacted when the size of packets in existing use cases is significantly increased and least impacted when large packets are used for new use cases where the deployment is likely to require updated RADIUS implementations.

There is a special challenge for proxies or clients with high request volume. When an RFC 6613 implementation receives a packet that is too large, it closes the connection and does not respond to any requests in process. Such a client would lose requests and might find distinguishing request-too-big situations from other failures difficult. In these cases, the discovery mechanism described in Section 3.2 can be used.

Also, RFC 6613 is an experiment. Part of running that experiment is to evaluate whether additional changes are required to RADIUS. A lower bar for interoperability should apply to changes to experimental protocols than standard protocols.

This specification provides good facilities to enable implementations to understand packet size when proxying to/from standards-track UDP RADIUS.

3.2. Discovery

As discussed in Section 2.1, a client MAY send a Status-Server message to discover whether an authentication or accounting server supports this specification. The client includes a Response-Length attribute; this signals the server to include a Response-Length attribute indicating the maximum packet size the server can process. In this one instance, Response-Length indicate the size of a request that can be processed rather than a response.

4. Protocol-Error Code

This document defines a new RADIUS code, TBDCODE (IANA), called Protocol-Error. This packet code may be used in response to any request packet, such as Access-Request, Accounting-Request, CoA-Request, or Disconnect-Request. It is a response packet sent by a server to a client. The packet indicates to the client that the server is unable to process the request for some reason.

A Protocol-Error packet MUST contain a Original-Packet-Code attribute, along with an Error-Cause attribute. Other attributes MAY be included if desired. The Original-Packet-Code contains the code from the request that generated the protocol error so that clients can disambiguate requests with different codes and the same ID. Regardless of the original packet code, the RADIUS server calculates the Message-Authenticator attribute as if the original packet were an Access-Request packet. The identifier is copied from the original request.

Clients processing Protocol-Error MUST ignore unknown or unexpected attributes.

This RADIUS code is hop-by-hop. Proxies MUST NOT forward a Protocol-Error packet they receive.

5. Too Big Response

When a RADIUS server receives a request that is larger than can be processed, it generates a Protocol-Error response as follows:

The code is Protocol-Error.

The Response-Length attribute MUST be included and its value is the maximum size of request that will be processed.

The Error-Cause attribute is included with a value of TOOBIGTBD.

The Original-Packet-Code attribute is copied from the request.

Clients will not typically be able to adjust and resend requests when this error is received. In some cases the client can fall back to RADIUS Fragmentation. In other cases this code will provide for better client error reporting and will avoid retransmitting requests guaranteed to fail.

6. IANA Considerations

A new RADIUS packet type code is registered in the RADIUS packet type codes registry discussed in section 2.1 of RFC 3575 [RFC3575]. The name is "Protocol-Error" and the code is TBDCODE. The IESG is requested to approve this registration along with approving publication of this document.

The following RADIUS attribute type values [RFC3575] are assigned. The assignment rules in section 10.3 of [RFC6929] are used.

Name	Attribute	Description
Response-Length	TBD	attribute of type "integer" per Section 5 of RFC 2865 containing maximum response length
Original-Packet-Code	TBD2	An integer attribute containing the code from a packet resulting in a Protocol-Error response.

The Response-Length attribute MAY be included in any RADIUS request. In this context it indicates the maximum length of a response the client is prepared to receive. Values are between 4096 and 65535. The attribute MAY also be included in a response to a Status-Server message. In this case the attribute indicates the maximum size RADIUS request that is permitted.

A new Error-Cause value is registered in the registry at http://www.iana.org/assignments/radius-types/radius-types.xhtml#radius-types-18 for "Response Too Big" with value TOOBIGTBD.

7. Security Considerations

This specification updates [RFC6613] and will be used with [RFC6614]. When used over plain TCP, this specification creates new opportunities for an on-path attacker to impact availability. These attacks can be entirely mitigated by using TLS. If these attacks are acceptable, then this specification can be used over TCP without TLS.

8. Acknowledgements

Sam Hartman’s time on this draft was funded by JANET as part of Project Moonshot.

Alan DeKok provided valuable review and text for the Protocol-Error packet code.

Alejandro Perez Mendez provided valuable review comments.

9. References

9.1. Normative References

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

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC3575] Aboba, B., "IANA Considerations for RADIUS (Remote Authentication Dial In User Service)", RFC 3575, July 2003.

[RFC5997] DeKok, A., "Use of Status-Server Packets in the Remote Authentication Dial In User Service (RADIUS) Protocol", RFC 5997, August 2010.

[RFC6613] DeKok, A., "RADIUS over TCP", RFC 6613, May 2012.

[RFC6614] Winter, S., McCauley, M., Venaas, S., and K. Wierenga, "Transport Layer Security (TLS) Encryption for RADIUS", RFC 6614, May 2012.

[RFC6929] DeKok, A. and A. Lior, "Remote Authentication Dial In User Service (RADIUS) Protocol Extensions", RFC 6929, April 2013.

9.2. Informative References

[I-D.ietf-abfab-aaa-saml] Howlett, J. and S. Hartman, "A RADIUS Attribute, Binding, Profiles, Name Identifier Format, and Confirmation Methods for SAML", draft-ietf-abfab-aaa-saml-09 (work in progress), February 2014.

[RFC7499] Perez-Mendez, A., Ed., Marin-Lopez, R., Pereniguez-Garcia, F., Lopez-Millan, G., Lopez, D., and A. DeKok, "Support of Fragmentation of RADIUS Packets", RFC 7499, DOI 10.17487/RFC7499, April 2015, <http://www.rfc-editor.org/info/rfc7499>.

Author's Address

Sam Hartman
Painless Security

Email: hartmans-ietf@mit.edu

Network Working Group   DeKok, Alan
INTERNET-DRAFT   FreeRADIUS
Updates: 2865,3162,6158,6572
Category: Standards Track
<draft-ietf-radext-datatypes-02.txt>
2 November 2015

Data Types in the Remote Authentication Dial-In User Service Protocol (RADIUS)
draft-ietf-radext-datatypes-02.txt

Abstract

RADIUS specifications have used data types for two decades without defining them as managed entities. During this time, RADIUS implementations have named the data types, and have used them in attribute definitions. This document updates the specifications to better follow established practice. We do this by naming the data types defined in RFC 6158, which have been used since at least RFC 2865. We provide an IANA registry for the data types, and update the RADIUS Attribute Type registry to include a "Data Type" field for each attribute. Finally, we recommend that authors of RADIUS specifications use these types in preference to existing practice.

Status of this Memo

Table of Contents

1. Introduction

RADIUS specifications have historically defined attributes in terms of name, type value, and data type. Of these three pieces of information, only the type value is managed by IANA. There is no management of, or restriction on, the attribute name, as discussed in [RFC6929] Section 2.7.1. There is no management of data type name or definition. Experience has shown that there is a need for well defined data types.

This document defines an IANA registry for data types, and updates the RADIUS Attribute Type registry to use those newly defined data types. It recommends how both specifications and implementations should use the data types. It extends the RADIUS Attribute Type registry to have a data type for each assigned attribute.

In this section, we review the use of data types in specifications and implementations. Whe highlight ambiguities and inconsistencies. The rest of this document is devoted to resolving those problems.

1.1. Specification Problems with Data Types

When attributes are defined in the specifications, the terms "Value" and "String" are used to refer to the contents of an attribute. However, these names are used recursively and inconsistently. We suggest that defining a field to recursively contain itself is problematic.

A number of data type names and definitions are given in [RFC2865] Section 5, at the bottom of page 25. These data types are named and clearly defined. However, this practice was not continued in later specifications.

Specifically, [RFC2865] defines attributes of data type "address" to carry IPv4 addresses. Despite this definition, [RFC3162] defines attributes of data type "Address" to carry IPv6 addresses. We suggest that the use of the word "address" to refer to disparate data types is problematic.

Other failures are that [RFC3162] does not give a data type name and definition for the data types IPv6 address, Interface-Id, or IPv6 prefix. [RFC2869] defines Event-Timestamp to carry a time, but does not re-use the "time" data type defined in [RFC2865]. Instead, it just repeats the "time" definition. [RFC6572] defines multiple attributes which carry IPv4 prefixes. However, an "IPv4 prefix" data type is not named, defined as a data type, or called out as an addition to RADIUS. Further, [RFC6572] does not follow the recommendations of [RFC6158], and does not explain why it fails to follow those recommendations.

These ambiguities and inconsistencies need to be resolved.

1.2. Implementation Problems with Data Types

RADIUS implementations often use "dictionaries" to map attribute names to type values, and to define data types for each attribute. The data types in the dictionaries are defined by each implementation, but correspond to the "ad hoc" data types used in the specifications.

In effect, implementations have seen the need for well-defined data types, and have created them. It is time for RADIUS specifications to follow this practice.

1.3. No Mandated Changes

This document mandates no changes to any RADIUS implementation, past, present, or future. It instead documents existing practice, in order to simplify the process of writing RADIUS specifications, to clarify the interpretation of RADIUS standards, and to improve the communication between specification authors and IANA.

This document suggests that implementations SHOULD use the data types defined here, in preference to any "ad hoc" data types currently in use. This suggestion should have minimal effect on implementations, as most "ad hoc" data types are compatible with the ones defined here. Any difference will typically be limited to the name of the data type.

1.4. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Use of Data Types

The Data Types can be used in two places: specifications, and implementations. This section discusses both uses, and gives guidance on using the data types.

2.1. Specification Use of Data Types

In this section, we give recommendations for how specifications should be written using data types. We first describe how attribute field names can be consistently named. We then describe how attribute definitions should use the data types, and deprecate the use of "Ascii art" for attribute definitions. We suggest a format for new attribute definitions. This format includes recommended fields, and suggestions for how those fields should be described.

Finally, we make recommendations for how new data types should be defined.

2.1.1. Field Names for Attribute Values

Previous specifications used inconsistent and conflicting names for the contents of RADIUS attributes. For example, the term "Value" is used in [RFC2865] Section 5 to define a field which carries the contents of attribute. It is then used in later sections as the sub-field of attribute contents. The result is that the field is defined as recursively containing itself. Similarly, "String" is used both as a data type, and as a sub-field of other data types.

We correct this ambiguity by using context-specific names for various fields of attributes and data types. It then becomes clear that, for example, that a field called "VSA-Data" must contain different data than a field called "EVS-Data". Each new name is defined where it is used.

We also define the following term:

Attr-Data

The "Value" field of an Attribute as defined in [RFC2865] Section 5. The contents of this field MUST be a valid data type as defined in the RADIUS Data Type registry.

We consistently use "Attr-Data" to refer to the contents of an attribute, instead of the more ambiguous name "Value". It is RECOMMENDED that new specifications follow this practice.

In this document, we use the term "Value" to refer to the contents of a data type, where that data type cannot carry other data types. In other cases, we refer to the contents of a data type with a type-specific name, in order to distinguish it from data of other types. For example, the data type "vsa" will contain a data field called "VSA-Data".

These terms are used in preference to the term "String", which was used in multiple incompatible ways. It is RECOMMENDED that future specifications use type-specific names, and the same naming scheme for new types. This use will maintain consistent definitions, and avoid ambiguities.

2.1.2. Attribute Definitions using Data Types

New RADIUS specifications MUST define attributes using data types from the RADIUS Data Type registry. The specification may, of course, define a new data type and use it in the same document. The guidelines given in [RFC6929] MUST be followed when defining a new data type.

Attributes can usually be completely described via the Attribute Type code, name, and data type. The use of "ASCII art" is then limited only to the definition of new data types, and for complex data types.

Use of the new extended attributes [RFC6929] makes ASCII art even more problematic. An attribute can be allocated from the standard space, or from one of the extended spaces. This allocation decision is made after the specification has been accepted for publication. That allocation strongly affects the format of the attribute header, making it nearly impossible to create the correct ASCII art prior to final publication. Allocation from the different spaces also changes the value of the Length field, also making it difficult to define it correctly prior to final publication of the document.

It is therefore RECOMMENDED that "ASCII art" diagrams not be used for new RADIUS attribute specifications.

2.1.3. Format of Attribute Definitions

When defining a new attribute, the following fields SHOULD be given:

Description

A description of the meaning and interpretation of the attribute.

Type

The Attribute Type code, given in the "dotted number" notation from [RFC6929]. Specifications can often leave this as "TBD", and request that IANA fill in the allocated values.

Length

A description of the length of the attribute. For attributes of variable length, a maximum length SHOULD be given. Since the Length may depend on the Type, the definition of Length may be affected by IANA allocations.

Data Type

One of the named data types from the RADIUS Data Type registry.

Value

A description of any attribute-specific limitations on the values carried by the specified data type. If there are no attribute-specific limitations, then the description of this field can be omitted, so long as the Description field is sufficiently explanatory.

Where the values are limited to a subset of the possible range, valid range(s) MUST be defined.

For attributes of data type "enum", a list of enumerated values and names MUST be given, as with [RFC2865] Section 5.6.

Using a consistent format for attribute definitions helps to make the definitions clearer.

2.1.4. Defining a New Data Type

When a specification needs to define a new data type, it should follow the format used by the definitions in Section 3 of this document. The text at the start of the data type definition MUST describe the data type, including the expected use, and why a new data type is required. That text SHOULD include limits on expected values, and why those limits exist. The fields "Name", "Value", "Length", and "Format", MUST be given, along with values.

The "Name" field SHOULD be a single name, all lower-case. Contractions such as "ipv4addr" are RECOMMENDED where they add clarity.

We note that the use of "Value" in the RADIUS Data Type registry can be confusing. That name is also used in attribute definitions, but with a different meaning. We trust that the meaning here is clear from the context.

The "Value" field should be given as to be determined or "TBD" in specifications. That number is assigned by IANA.

The "Format" field SHOULD be defined with "Ascii art" in order to have a precise definition. Machine-readable formats are also RECOMMENDED.

The definition of a new data type should be done only when absolutely necessary. We do not expect a need for a large number of new data types. When defining a new data type, the guideliness of [RFC6929] with respect to data types MUST be followed.

It is RECOMMENDED that vendors not define "vendor specific" data types. As discussed in [RFC6929], those data types are rarely necessary, and can cause interoperability problems.

Any new data type MUST have unique name in the RADIUS Data Type registry. The number of the data type will be assigned by IANA.

2.2. Implementation Use of Data Types

Implementations not supporting a particular data type MUST treat attributes of that data type as being of data type "string", as defined in Section 2.6. It is RECOMMENDED that such attributes be treated as "invalid attributes", as defined in [RFC6929] Section 2.8.

Where the contents of a data type do not match the definition, implementations MUST treat the the enclosing attribute as being an "invalid attribute". This requirement includes, but is not limited to, the following situations:

* Attributes with values outside of the allowed range(s) for the data type, e.g. as given in the data types "integer", "ipv4addr", "ipv6addr", "ipv4prefix", "ipv6prefix", or "enum".

* "text" attributes where the contents do not match the required format,

* Attributes where the length is shorter or longer than the allowed length(s) for the given data type,

The requirements for "reserved" fields are more difficult to quantify. Implementations SHOULD be able to receive and process attributes where "reserved" fields are non-zero. We do not, however, define any "correct" processing of such attributes. Instead, specifications which define new meaning for "reserved" fields SHOULD describe how older implementations process those fields. We expect that such descriptions are derived from practice. Implementations MUST set "reserved" fields to zero when creating attributes.

3. Data Type Definitions

This section defines the new data types. For each data type, it gives a definition, a name, a number, a length, and an encoding format. Where relevant, it describes subfields contained within the data type. These definitions have no impact on existing RADIUS implementations. There is no requirement that implementations use these names.

Where possible, the name of each data type has been taken from previous specifications. In some cases, a different name has been chosen. The change of name is sometimes required to avoid ambiguity (i.e. "address" versus "Address"). Otherwise, the new name has been chosen to be compatible with [RFC2865], or with use in common implementations. In some cases, new names are chosen to clarify the interpretation of the data type.

The numbers assigned herein for the data types have no meaning other than to permit them to be tracked by IANA. As RADIUS does not encode information about data types in a packet, the numbers assigned to a data type will never occur in a packet. It is RECOMMENDED that new implementations use the names defined in this document, in order to avoid confusion. Existing implementations may choose to use the names defined here, but that is not required.

The encoding of each data type is taken from previous specifications. The fields are transmitted from left to right.

Where the data types have inter-dependencies, the simplest data type is given first, and dependent ones are given later.

We do not create specific data types for the "tagged" attributes defines in [RFC2868]. That specification defines the "tagged" attributes as being backwards compatible with pre-existing data types. In addition, [RFC6158] Section 2.1 says that "tagged" attributes should not be used. There is therefore no benefit to defining additional data types for these attributes. We trust that implementors will be aware that tagged attributes must be treated differently from non-tagged attributes of the same data type.

Similarly, we do not create data types for some attributes having complex structure, such as CHAP-Password, ARAP-Features, or Location-Capable. We need to strike a balance between correcting earlier mistakes, and making this document more complex. In some cases, it is better to treat complex attributes as being of type "string", even though they need to be interpreted by RADIUS implementations. The guidelines given in Section 6.3 of [RFC6969] were used to make this determination.

3.1. integer

The "integer" data type encodes a 32-bit unsigned integer in network byte order. Where the range of values for a particular attribute is limited to a sub-set of the values, specifications MUST define the valid range. Attributes with Values outside of the allowed ranges SHOULD be treated as "invalid attributes".

Name

integer

Value

1

Length

Four octets

Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Value                                                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.2. enum

The "enum" data type encodes a 32-bit unsigned integer in network byte order. It differs from the "integer" data type only in that it is used to define enumerated types, such as Service-Type (Section 5.6 of [RFC 2865]). Specifications MUST define a valid set of enumerated values, along with a unique name for each value. Attributes with Values outside of the allowed enumerations SHOULD be treated as "invalid attributes".

Name

enum

Value

2

Length

Four octets

Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Value                                                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.3. ipv4addr

The "ipv4addr" data type encodes an IPv4 address in network byte order. Where the range of address for a particular attribute is limited to a sub-set of possible addresses, specifications MUST define the valid range(s). Attributes with Addresses outside of the allowed range(s) SHOULD be treated as "invalid attributes".

Name

ipv4addr

Value

3

Length

Four octets

Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Address                                                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.4. time

The "time" data type encodes time as a 32-bit unsigned value in network byte order and in seconds since 00:00:00 UTC, January 1, 1970. We note that dates before the year 2015 are likely to be erroneous.

Note that the "time" attribute is defined to be unsigned, which means it is not subject to a signed integer overflow in the year 2038.

Name

time

Value

4

Length

Four octets

Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Time                                                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.5. text

The "text" data type encodes UTF-8 text [RFC3629]. The maximum length of the text is given by the encapsulating attribute. Where the range of lengths for a particular attribute is limited to a sub-set of possible lengths, specifications MUST define the valid range(s). Attributes with length outside of the allowed values SHOULD be treated as "invalid attributes".

Where the text is intended to carry data in a particular format, (e.g. Framed-Route), the format MUST be given. The specification SHOULD describe the format in a machine-readable way, such as via Augmented Backus-Naur Form (ABNF). Attributes with values not matching the defined format SHOULD be treated as "invalid attributes".

Note that the "text" data type does not terminate with a NUL octet (hex 00). The Attribute has a Length field and does not use a terminator. Texts of length zero (0) MUST NOT be sent; omit the entire attribute instead.

Name

text

Value

5

Length

One or more octets.

Format

```
 0
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-
|  Value    ...
+-+-+-+-+-+-+-+-
```

3.6. string

The "string" data type encodes binary data, as a sequence of undistinguished octets. Where the range of lengths for a particular attribute is limited to a sub-set of possible lengths, specifications MUST define the valid range(s). Attributes with length outside of the allowed values SHOULD be treated as "invalid attributes".

Note that the "string" data type does not terminate with a NUL octet (hex 00). The Attribute has a Length field and does not use a terminator. Strings of length zero (0) MUST NOT be sent; omit the entire attribute instead.

Where there is a need to encapsulate complex data structures, and TLVs cannot be used, the "string" data type MUST be used. This requirement include encapsulation of data structures defined outside of RADIUS, which are opaque to the RADIUS infrastucture. It also includes encapsulation of some data structures which are not opaque to RADIUS, such as the contents of CHAP-Password.

There is little reason to define a new RADIUS data type for only one attribute. However, where the complex data type cannot be represented as TLVs, and is expected to be used in many attributes, a new data type SHOULD be defined.

These requirements are stronger than [RFC6158], which makes the above encapsulation a "SHOULD". This document defines data types for use in RADIUS, so there are few reasons to avoid using them.

Name

      string

   Value

      6

   Length

      One or more octets.

   Format

```
       0
       0 1 2 3 4 5 6 7
      +-+-+-+-+-+-+-+-
      |  Octets    ...
      +-+-+-+-+-+-+-+-
```

3.7. concat

The "concat" data type permits the transport of more than 253 octets of data in a "standard space" [RFC6929] attribute. It is otherwise identical to the "string" data type.

If multiple attributes of this data type are contained in a packet, all attributes of the same type code MUST be in order and they MUST be consecutive attributes in the packet.

The amount of data transported in a "concat" data type can be no more than the RADIUS packet size. In practice, the requirement to transport multiple attributes means that the limit may be substantially smaller than one RADIUS packet. As a rough guide, is RECOMMENDED that this data type transport no more than 2048 octets of data.

The "concat" data type MAY be used for "standard space" attributes. It MUST NOT be used for attributes in the "short extended space" or the "long extended space". It MUST NOT be used in any field or subfields of the following data types: "tlv", "vsa", "extended", "long-extended", or "evs".

   Name

      concat

   Value

      7

   Length

      One or more octets.

   Format

```
       0
       0 1 2 3 4 5 6 7
      +-+-+-+-+-+-+-+-
      |  Octets    ...
      +-+-+-+-+-+-+-+-
```

3.8. ifid

   The "ifid" data type encodes an Interface-Id as an 8-octet string in
   network byte order.

   Name

      ifid

   Value

      8

   Length

      Eight octets

   Format

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |     Interface-ID ...
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
           ... Interface-ID                                         |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.9. ipv6addr

   The "ipv6addr" data type encodes an IPv6 address in network byte
   order. Where the range of address for a particular attribute is
   limited to a sub-set of possible addresses, specifications MUST

define the valid range(s). Attributes with Addresses outside of the allowed range(s) SHOULD be treated as "invalid attributes".

Name

ipv6addr

Value

9

Length

Sixteen octets

Format

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Address ...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        ... Address ...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        ... Address ...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        ... Address                                                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.10. ipv6prefix

The "ipv6prefix" data type encodes an IPv6 prefix, using both a prefix length and an IPv6 address in network byte order. Where the range of prefixes for a particular attribute is limited to a sub-set of possible prefixes, specifications MUST define the valid range(s). Attributes with Addresses outside of the allowed range(s) SHOULD be treated as "invalid attributes".

Attributes with a Prefix-Length field having value greater than 128 SHOULD be treated as "invalid attributes".

Name

ipv6prefix

Value

      10

   Length

      At least two, and no more than eighteen octets.

   Format

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |     Reserved    | Prefix-Length |  Prefix ...
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
           ... Prefix ...
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
           ... Prefix ...
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
           ... Prefix                                                |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   Subfields

      Reserved

         This field, which is reserved and MUST be present, is always
         set to zero.

      Prefix-Length

         The length of the prefix, in bits. At least 0 and no larger
         than 128.

      Prefix

         The Prefix field is up to 16 octets in length. Bits outside of
         the Prefix-Length, if included, MUST be zero.

3.11. ipv4prefix

The "ipv4prefix" data type encodes an IPv4 prefix, using both a prefix length and an IPv4 address in network byte order. Where the range of prefixes for a particular attribute is limited to a sub-set of possible prefixes, specifications MUST define the valid range(s). Attributes with Addresses outside of the allowed range(s) SHOULD be treated as "invalid attributes".

Attributes with a Prefix-Length field having value greater than 32

SHOULD be treated as "invalid attributes".

Name

ipv4prefix

Value

11

Length

At least two, and no more than eighteen octets.

Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Reserved      | Prefix-Len|  Prefix ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     ... Prefix                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Subfields

Reserved

This field, which is reserved and MUST be present, is always set to zero.

Prefix-Length

A 6-bit unsigned integer containing the length of the prefix, in bits. The values MUST be no larger than 32.

Prefix

The Prefix field is 4 octets in length. Bits outside of the Prefix-Length MUST be zero. Unlike the "ipv6prefix" data type, this field is fixed length. If the address is all zeros (i.e. "0.0.0.0", then the Prefix-Length MUST be set to 32.

3.12. integer64

The "integer64" data type encodes a 64-bit unsigned integer in network byte order. Where the range of values for a particular attribute is limited to a sub-set of the values, specifications MUST define the valid range(s). Attributes with Values outside of the allowed range(s) SHOULD be treated as "invalid attributes".

Name

integer64

Value

12

Length

Eight octets

Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Value ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      ... Value                                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.13. tlv

The "tlv" data type encodes a type-length-value, as defined in [RFC6929] Section 2.3.

Name

tlv

Value

13

Length

Three or more octets

Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
```

```
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |   TLV-Type    |  TLV-Length   |     TLV-Data ...
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Subfields

TLV-Type

This field is one octet. Up-to-date values of this field are specified according to the policies and rules described in [RFC6929] Section 10. Values of 254-255 are "Reserved" for use by future extensions to RADIUS. The value 26 has no special meaning, and MUST NOT be treated as a Vendor Specific attribute.

The TLV-Type is meaningful only within the context defined by "Type" fields of the encapsulating Attributes, using the dotted-number notation introduced in [RFC6929].

A RADIUS server MAY ignore Attributes with an unknown "TLV-Type".

A RADIUS client MAY ignore Attributes with an unknown "TLV-Type".

A RADIUS proxy SHOULD forward Attributes with an unknown "TLV-Type" verbatim.

TLV-Length

The TLV-Length field is one octet, and indicates the length of this TLV including the TLV-Type, TLV-Length and TLV-Value fields. It MUST have a value between 3 and 255. If a client or server receives a TLV with an invalid TLV-Length, then the attribute which encapsulates that TLV MUST be considered to be an "invalid attribute", and handled as per [RFC6929] Section 2.8.

TLVs having TLV-Length of zero (0) MUST NOT be sent; omit the entire TLV instead.

TLV-Data

The TLV-Data field is one or more octets and contains information specific to the Attribute. The format and length of the TLV-Data field is determined by the TLV-Type and TLV-Length fields.

The TLV-Data field MUST contain only known RADIUS data types. The TLV-Data field MUST NOT contain any of the following data types: "concat", "vsa", "extended", "long-extended", or "evs".

3.14. vsa

The "vsa" data type encodes Vendor-Specific data, as given in [RFC2865] Section 5.26. It is used only in the Attr-Data field of a Vendor-Specific Attribute. It MUST NOT appear in the contents of any other data type.

Where an implementation determines that an attribute of data type "vsa" contains data which does not match the expected format, it SHOULD treat that attribute as being an "invalid attribute".

Name

vsa

Value

14

Length

Five or more octets

Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Vendor-Id                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  VSA-Data ....
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Subfields

Vendor-Id

The 4 octets are the Network Management Private Enterprise Code [PEN] of the Vendor in network byte order.

VSA-Data

The VSA-Data field is one or more octets. The actual format of the information is site or application specific, and a robust implementation SHOULD support the field as undistinguished octets.

The codification of the range of allowed usage of this field is outside the scope of this specification.

The "vsa" data type SHOULD contain as a sequence of "tlv" data types. The interpretation of the TLV-Type and TLV-Data fields are dependent on the vendor's definition of that attribute.

The "vsa" data type MUST be used as contents of the Attr-Data field of the Vendor-Specific attribute. The "vsa" data type MUST NOT appear in the contents of any other data type.

3.15. extended

The "extended" data type encodes the "Extended Type" format, as given in [RFC6929] Section 2.1. It is used only in the Attr-Data field of an Attribute allocated from the "standard space". It MUST NOT appear in the contents of any other data type.

Name

extended

Value

15

Length

Two or more octets

Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Extended-Type | Ext-Data ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Subfields

Extended-Type

The Extended-Type field is one octet. Up-to-date values of this field are specified according to the policies and rules described in [RFC6929] Section 10. Unlike the Type field defined in [RFC2865] Section 5, no values are allocated for experimental or implementation-specific use. Values 241-255 are reserved and MUST NOT be used.

The Extended-Type is meaningful only within a context defined by the Type field. That is, this field may be thought of as defining a new type space of the form "Type.Extended-Type". See [RFC6929] Section 2.5 for additional discussion.

A RADIUS server MAY ignore Attributes with an unknown "Type.Extended-Type".

A RADIUS client MAY ignore Attributes with an unknown "Type.Extended-Type".

Ext-Data

The contents of this field MUST be a valid data type as defined in the RADIUS Data Type registry. The Ext-Data field MUST NOT contain any of the following data types: "concat", "vsa", "extended", "long-extended", or "evs".

The Ext-Data field is one or more octets.

Implementations supporting this specification MUST use the Identifier of "Type.Extended-Type" to determine the interpretation of the Ext-Data field.

3.16. long-extended

The "long-extended" data type encodes the "Long Extended Type" format, as given in [RFC6929] Section 2.2. It is used only in the Attr-Data field of an Attribute. It MUST NOT appear in the contents of any other data type.

Name

long-extended

Value

16

Length

Three or more octets

Format

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Extended-Type |M| Reserved    | Ext-Data ...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Subfields

Extended-Type

This field is identical to the Extended-Type field defined above in Section 2.13.

M (More)

The More field is one (1) bit in length, and indicates whether or not the current attribute contains "more" than 251 octets of data. The More field MUST be clear (0) if the Length field has value less than 255. The More field MAY be set (1) if the Length field has value of 255.

If the More field is set (1), it indicates that the Ext-Data field has been fragmented across multiple RADIUS attributes. When the More field is set (1), the attribute MUST have a Length field of value 255; there MUST be an attribute following this one; and the next attribute MUST have both the same Type and Extended Type. That is, multiple fragments of the same value MUST be in order and MUST be consecutive attributes in the packet, and the last attribute in a packet MUST NOT have the More field set (1).

That is, a packet containing a fragmented attribute needs to contain all fragments of the attribute, and those fragments need to be contiguous in the packet. RADIUS does not support inter-packet fragmentation, which means that fragmenting an attribute across multiple packets is impossible.

If a client or server receives an attribute fragment with the "More" field set (1), but for which no subsequent fragment can be found, then the fragmented attribute is considered to be an "invalid attribute", and handled as per [RFC6929] Section 2.8.

Reserved

This field is 7 bits long, and is reserved for future use. Implementations MUST set it to zero (0) when encoding an attribute for sending in a packet. The contents SHOULD be ignored on reception.

Future specifications may define additional meaning for this field. Implementations therefore MUST NOT treat this field as invalid if it is non-zero.

Ext-Data

The contents of this field MUST be a valid data type as defined in the RADIUS Data Type registry. The Ext-Data field MUST NOT contain any of the following data types: "concat", "vsa", "extended", "long-extended", or "evs".

The Ext-Data field is one or more octets.

Implementations supporting this specification MUST use the Identifier of "Type.Extended-Type" to determine the interpretation of the Ext-Data field.

The length of the data MUST be taken as the sum of the lengths of the fragments (i.e. Ext-Data fields) from which it is constructed. Any interpretation of the resulting data MUST occur after the fragments have been reassembled. If the reassembled data does not match the expected format, each fragment MUST be treated as an "invalid attribute", and the reassembled data MUST be discarded.

We note that the maximum size of a fragmented attribute is limited only by the RADIUS packet length limitation. Implementations MUST be able to handle the case where one fragmented attribute completely fills the packet.

3.17. evs

The "evs" data type encodes an "Extended Vendor-Specific" attribute, as given in [RFC6929] Section 2.4. The "evs" data type is used solely to extend the Vendor Specific space. It MAY appear inside of an "extended" or a "long-extended" data type. It MUST NOT appear in the contents of any other data type.

Where an implementation determines that an attribute of data type "evs" contains data which does not match the expected format, it SHOULD treat that attribute as being an "invalid attribute".

Name

evs

Value

17

Length

Six or more octets

Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Vendor-Id                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Vendor-Type   |  EVS-Data ....
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Subfields

Vendor-Id

The 4 octets are the Network Management Private Enterprise Code [PEN] of the Vendor in network byte order.

Vendor-Type

The Vendor-Type field is one octet. Values are assigned at the sole discretion of the Vendor.

EVS-Data

The EVS-Data field is one or more octets. It SHOULD encapsulate a previously defined RADIUS data type. Non-standard data types SHOULD NOT be used. We note that the EVS-Data field may be of data type "tlv".

The actual format of the information is site or application specific, and a robust implementation SHOULD support the field as undistinguished octets. We recognise that Vendors have complete control over the contents and format of the Ext-Data field, while at the same time recommending that good practices be followed.

Further codification of the range of allowed usage of this field is outside the scope of this specification.

4. Updated Registries

This section defines a new IANA registry for RADIUS data types, and updates the existing RADIUS Attribute Type registry.

4.1. Create a Data Type Registry

This section defines a new RADIUS registry, called "Data Type". Allocation in this registry requires IETF Review. The "Registration Procedures" for this registry are "Standards Action".

The registry contains three columns of data, as follows.

Value

The number of the data type. The value field is an artifact of the registry, and has no on-the-wire meaning.

Description

The name of the data type. The name field is used only for the registry, and has no on-the-wire meaning.

Reference

The specification where the data type was defined.

The initial contents of the registry are as follows.

Value	Description	Reference
1	integer	[RFC2865], TBD
2	enum	[RFC2865], TBD
3	ipv4addr	[RFC2865], TBD
4	time	[RFC2865], TBD
5	text	[RFC2865], TBD
6	string	[RFC2865], TBD
7	concat	TBD
8	ifid	[RFC3162], TBD
9	ipv6addr	[RFC3162], TBD
10	ipv6prefix	[RFC3162], TBD
11	ipv4prefix	[RFC6572], TBD
12	integer64	[RFC6929], TBD
13	tlv	[RFC6929], TBD

14	evs	[RFC6929], TBD
15	extended	[RFC6929], TBD
16	long-extended	[RFC6929], TBD

4.2. Updates to the Attribute Type Registry

This section updates the RADIUS Attribute Type Registry to have a new column, which is inserted in between the existing "Description" and "Reference" columns. The new column is named "Data Type". The contents of that column are the name of a data type, corresponding to the attribute in that row, or blank if the attribute type is unassigned. The name of the data type is taken from the RADIUS Data Type registry, defined above.

The updated registry follows in CSV format.

```
Value,Description,Data Type,Reference
1,User-Name,text,[RFC2865]
2,User-Password,string,[RFC2865]
3,CHAP-Password,string,[RFC2865]
4,NAS-IP-Address,ipv4addr,[RFC2865]
5,NAS-Port,integer,[RFC2865]
6,Service-Type,enum,[RFC2865]
7,Framed-Protocol,enum,[RFC2865]
8,Framed-IP-Address,ipv4addr,[RFC2865]
9,Framed-IP-Netmask,ipv4addr,[RFC2865]
10,Framed-Routing,enum,[RFC2865]
11,Filter-Id,text,[RFC2865]
12,Framed-MTU,integer,[RFC2865]
13,Framed-Compression,enum,[RFC2865]
14,Login-IP-Host,ipv4addr,[RFC2865]
15,Login-Service,enum,[RFC2865]
16,Login-TCP-Port,integer,[RFC2865]
17,Unassigned,,
18,Reply-Message,text,[RFC2865]
19,Callback-Number,text,[RFC2865]
20,Callback-Id,text,[RFC2865]
21,Unassigned,,
22,Framed-Route,text,[RFC2865]
23,Framed-IPX-Network,ipv4addr,[RFC2865]
24,State,string,[RFC2865]
25,Class,string,[RFC2865]
26,Vendor-Specific,vsa,[RFC2865]
27,Session-Timeout,integer,[RFC2865]
28,Idle-Timeout,integer,[RFC2865]
29,Termination-Action,enum,[RFC2865]
30,Called-Station-Id,text,[RFC2865]
31,Calling-Station-Id,text,[RFC2865]
```

```
   32,NAS-Identifier,text,[RFC2865]
   33,Proxy-State,string,[RFC2865]
   34,Login-LAT-Service,text,[RFC2865]
   35,Login-LAT-Node,text,[RFC2865]
   36,Login-LAT-Group,string,[RFC2865]
   37,Framed-AppleTalk-Link,integer,[RFC2865]
   38,Framed-AppleTalk-Network,integer,[RFC2865]
   39,Framed-AppleTalk-Zone,text,[RFC2865]
   40,Acct-Status-Type,enum,[RFC2866]
   41,Acct-Delay-Time,integer,[RFC2866]
   42,Acct-Input-Octets,integer,[RFC2866]
   43,Acct-Output-Octets,integer,[RFC2866]
   44,Acct-Session-Id,text,[RFC2866]
   45,Acct-Authentic,enum,[RFC2866]
   46,Acct-Session-Time,integer,[RFC2866]
   47,Acct-Input-Packets,integer,[RFC2866]
   48,Acct-Output-Packets,integer,[RFC2866]
   49,Acct-Terminate-Cause,enum,[RFC2866]
   50,Acct-Multi-Session-Id,text,[RFC2866]
   51,Acct-Link-Count,integer,[RFC2866]
   52,Acct-Input-Gigawords,integer,[RFC2869]
   53,Acct-Output-Gigawords,integer,[RFC2869]
   54,Unassigned,,
   55,Event-Timestamp,time,[RFC2869]
   56,Egress-VLANID,integer,[RFC4675]
   57,Ingress-Filters,enum,[RFC4675]
   58,Egress-VLAN-Name,text,[RFC4675]
   59,User-Priority-Table,string,[RFC4675]
   60,CHAP-Challenge,string,[RFC2865]
   61,NAS-Port-Type,enum,[RFC2865]
   62,Port-Limit,integer,[RFC2865]
   63,Login-LAT-Port,text,[RFC2865]
   64,Tunnel-Type,enum,[RFC2868]
   65,Tunnel-Medium-Type,enum,[RFC2868]
   66,Tunnel-Client-Endpoint,text,[RFC2868]
   67,Tunnel-Server-Endpoint,text,[RFC2868]
   68,Acct-Tunnel-Connection,text,[RFC2867]
   69,Tunnel-Password,string,[RFC2868]
   70,ARAP-Password,string,[RFC2869]
   71,ARAP-Features,string,[RFC2869]
   72,ARAP-Zone-Access,enum,[RFC2869]
   73,ARAP-Security,integer,[RFC2869]
   74,ARAP-Security-Data,text,[RFC2869]
   75,Password-Retry,integer,[RFC2869]
   76,Prompt,enum,[RFC2869]
   77,Connect-Info,text,[RFC2869]
   78,Configuration-Token,text,[RFC2869]
   79,EAP-Message,concat,[RFC2869]
   80,Message-Authenticator,string,[RFC2869]
   81,Tunnel-Private-Group-ID,text,[RFC2868]
   82,Tunnel-Assignment-ID,text,[RFC2868]
   83,Tunnel-Preference,integer,[RFC2868]
   84,ARAP-Challenge-Response,string,[RFC2869]
   85,Acct-Interim-Interval,integer,[RFC2869]
   86,Acct-Tunnel-Packets-Lost,integer,[RFC2867]
   87,NAS-Port-Id,text,[RFC2869]
   88,Framed-Pool,text,[RFC2869]
   89,CUI,string,[RFC4372]
   90,Tunnel-Client-Auth-ID,text,[RFC2868]
   91,Tunnel-Server-Auth-ID,text,[RFC2868]
   92,NAS-Filter-Rule,text,[RFC4849]
   93,Unassigned,,
   94,Originating-Line-Info,string,[RFC7155]
   95,NAS-IPv6-Address,ipv6addr,[RFC3162]
   96,Framed-Interface-Id,ifid,[RFC3162]
   97,Framed-IPv6-Prefix,ipv6prefix,[RFC3162]
   98,Login-IPv6-Host,ipv6addr,[RFC3162]
   99,Framed-IPv6-Route,text,[RFC3162]
   100,Framed-IPv6-Pool,text,[RFC3162]
   101,Error-Cause Attribute,enum,[RFC3576]
   102,EAP-Key-Name,string,[RFC4072][RFC7268]
   103,Digest-Response,text,[RFC5090]
   104,Digest-Realm,text,[RFC5090]
   105,Digest-Nonce,text,[RFC5090]
   106,Digest-Response-Auth,text,[RFC5090]
   107,Digest-Nextnonce,text,[RFC5090]
   108,Digest-Method,text,[RFC5090]
   109,Digest-URI,text,[RFC5090]
   110,Digest-Qop,text,[RFC5090]
   111,Digest-Algorithm,text,[RFC5090]
   112,Digest-Entity-Body-Hash,text,[RFC5090]
   113,Digest-CNonce,text,[RFC5090]
   114,Digest-Nonce-Count,text,[RFC5090]
   115,Digest-Username,text,[RFC5090]
   116,Digest-Opaque,text,[RFC5090]
   117,Digest-Auth-Param,text,[RFC5090]
   118,Digest-AKA-Auts,text,[RFC5090]
   119,Digest-Domain,text,[RFC5090]
   120,Digest-Stale,text,[RFC5090]
   121,Digest-HA1,text,[RFC5090]
   122,SIP-AOR,text,[RFC5090]
   123,Delegated-IPv6-Prefix,ipv6prefix,[RFC4818]
   124,MIP6-Feature-Vector,string,[RFC5447]
   125,MIP6-Home-Link-Prefix,ipv6prefix,[RFC5447]
   126,Operator-Name,text,[RFC5580]
   127,Location-Information,string,[RFC5580]
   128,Location-Data,string,[RFC5580]
   129,Basic-Location-Policy-Rules,string,[RFC5580]
   130,Extended-Location-Policy-Rules,string,[RFC5580]
   131,Location-Capable,enum,[RFC5580]
   132,Requested-Location-Info,enum,[RFC5580]
   133,Framed-Management-Protocol,enum,[RFC5607]
   134,Management-Transport-Protection,enum,[RFC5607]
   135,Management-Policy-Id,text,[RFC5607]
   136,Management-Privilege-Level,integer,[RFC5607]
   137,PKM-SS-Cert,concat,[RFC5904]
   138,PKM-CA-Cert,concat,[RFC5904]
   139,PKM-Config-Settings,string,[RFC5904]
   140,PKM-Cryptosuite-List,string,[RFC5904]
   141,PKM-SAID,text,[RFC5904]
   142,PKM-SA-Descriptor,string,[RFC5904]
   143,PKM-Auth-Key,string,[RFC5904]
   144,DS-Lite-Tunnel-Name,text,[RFC6519]
   145,Mobile-Node-Identifier,string,[RFC6572]
   146,Service-Selection,text,[RFC6572]
   147,PMIP6-Home-LMA-IPv6-Address,ipv6addr,[RFC6572]
   148,PMIP6-Visited-LMA-IPv6-Address,ipv6addr,[RFC6572]
   149,PMIP6-Home-LMA-IPv4-Address,ipv4addr,[RFC6572]
   150,PMIP6-Visited-LMA-IPv4-Address,ipv4addr,[RFC6572]
   151,PMIP6-Home-HN-Prefix,ipv6prefix,[RFC6572]
   152,PMIP6-Visited-HN-Prefix,ipv6prefix,[RFC6572]
   153,PMIP6-Home-Interface-ID,ifid,[RFC6572]
   154,PMIP6-Visited-Interface-ID,ifid,[RFC6572]
   155,PMIP6-Home-IPv4-HoA,ipv4prefix,[RFC6572]
   156,PMIP6-Visited-IPv4-HoA,ipv4prefix,[RFC6572]
   157,PMIP6-Home-DHCP4-Server-Address,ipv4addr,[RFC6572]
   158,PMIP6-Visited-DHCP4-Server-Address,ipv4addr,[RFC6572]
   159,PMIP6-Home-DHCP6-Server-Address,ipv6addr,[RFC6572]
   160,PMIP6-Visited-DHCP6-Server-Address,ipv6addr,[RFC6572]
   161,PMIP6-Home-IPv4-Gateway,ipv4addr,[RFC6572]
   162,PMIP6-Visited-IPv4-Gateway,ipv4addr,[RFC6572]
   163,EAP-Lower-Layer,enum,[RFC6677]
   164,GSS-Acceptor-Service-Name,text,[RFC7055]
   165,GSS-Acceptor-Host-Name,text,[RFC7055]
   166,GSS-Acceptor-Service-Specifics,text,[RFC7055]
   167,GSS-Acceptor-Realm-Name,text,[RFC7055]
   168,Framed-IPv6-Address,ipv6addr,[RFC6911]
   169,DNS-Server-IPv6-Address,ipv6addr,[RFC6911]
   170,Route-IPv6-Information,ipv6prefix,[RFC6911]
   171,Delegated-IPv6-Prefix-Pool,text,[RFC6911]
   172,Stateful-IPv6-Address-Pool,text,[RFC6911]
   173,IPv6-6rd-Configuration,tlv,[RFC6930]
   174,Allowed-Called-Station-Id,text,[RFC7268]
   175,EAP-Peer-Id,string,[RFC7268]
   176,EAP-Server-Id,string,[RFC7268]
   177,Mobility-Domain-Id,integer,[RFC7268]
   178,Preauth-Timeout,integer,[RFC7268]
   179,Network-Id-Name,string,[RFC7268]
   180,EAPoL-Announcement,concat,[RFC7268]
   181,WLAN-HESSID,text,[RFC7268]
   182,WLAN-Venue-Info,integer,[RFC7268]
   183,WLAN-Venue-Language,string,[RFC7268]
   184,WLAN-Venue-Name,text,[RFC7268]
   185,WLAN-Reason-Code,integer,[RFC7268]
   186,WLAN-Pairwise-Cipher,integer,[RFC7268]
   187,WLAN-Group-Cipher,integer,[RFC7268]
   188,WLAN-AKM-Suite,integer,[RFC7268]
   189,WLAN-Group-Mgmt-Cipher,integer,[RFC7268]
   190,WLAN-RF-Band,integer,[RFC7268]
   191,Unassigned,,
   192-223,Experimental Use,,[RFC3575]
   224-240,Implementation Specific,,[RFC3575]
   241,Extended-Attribute-1,extended,[RFC6929]
   241.1,Frag-Status,integer,[RFC7499]
   241.2,Proxy-State-Length,integer,[RFC7499]
   241.{3-25},Unassigned,,
   241.26,Extended-Vendor-Specific-1,evs,[RFC6929]
   241.{27-240},Unassigned,,
   241.{241-255},Reserved,,[RFC6929]
   242,Extended-Attribute-2,extended,[RFC6929]
   242.{1-25},Unassigned,,
   242.26,Extended-Vendor-Specific-2,evs,[RFC6929]
   242.{27-240},Unassigned,,
   242.{241-255},Reserved,,[RFC6929]
   243,Extended-Attribute-3,extended,[RFC6929]
   243.{1-25},Unassigned,,
   243.26,Extended-Vendor-Specific-3,evs,[RFC6929]
   243.{27-240},Unassigned,,
   243.{241-255},Reserved,,[RFC6929]
   244,Extended-Attribute-4,extended,[RFC6929]
   244.{1-25},Unassigned,,
   244.26,Extended-Vendor-Specific-4,evs,[RFC6929]
   244.{27-240},Unassigned,,
   244.{241-255},Reserved,,[RFC6929]
   245,Extended-Attribute-5,long-extended,[RFC6929]
   245.{1-25},Unassigned,,
   245.26,Extended-Vendor-Specific-5,evs,[RFC6929]
   245.{27-240},Unassigned,,
   245.{241-255},Reserved,,[RFC6929]
   246,Extended-Attribute-6,long-extended,[RFC6929]
   246.{1-25},Unassigned,,
   246.26,Extended-Vendor-Specific-6,evs,[RFC6929]
```

```
246.{27-240},Unassigned,,
246.{241-255},Reserved,,[RFC6929]
247-255,Reserved,,[RFC3575]
```

5. Security Considerations

This specification is concerned solely with updates to IANA registries. As such, there are no security considerations with the document itself.

However, the use of inconsistent names and poorly-defined entities in a protocol is problematic. Inconsistencies in specifications can lead to security and interoperability problems in implementations. Further, having one canonical source for the definition of data types means an implementor has fewer specifications to read. The implementation work is therefore simpler, and is more likely to be correct.

The goal of this specification is to reduce ambiguities in the RADIUS protocol, which we believe will lead to more robust and more secure implementations.

6. IANA Considerations

IANA is instructed to create one new registry as described above in Section 3.1. The "TBD" text in that section should be replaced with the RFC number of this document when it is published.

IANA is instructed to update the RADIUS Attribute Type registry, as described above in Section 3.2.

IANA is instructed to require that all allocation requests in the RADIUS Attribute Type Registry contain a "Data Type" field. That field is required to contain one of the "Data Type" names contained in the RADIUS Data Type registry.

IANA is instructed to require that updates to the RADIUS Data Type registry contain the following fields, with the associated instructions:

* Value. IANA is instructed to assign the next unused integer in sequence to new data type definitions.

* Name. IANA is instructed to require that this name be unique in the registry.

* Reference. IANA is instructed to update this field with a

reference
to the document which defines the data type.

7. References

7.1. Normative References

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

[RFC2865]
Rigney, C., Willens, S., Rubens, A. and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC3162]
Aboba, B., Zorn, G., and D. Mitton, "RADIUS and IPv6", RFC 3162, August 2001.

[RFC3629]
Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 3629, November 2003.

[RFC4072]
Eronen, P., et al, "Diameter Extensible Authentication Protocol (EAP) Application", RFC 4072, February 2013.

[RFC6158]
DeKok, A., and Weber, G., "RADIUS Design Guidelines", RFC 6158, March 2011.

[RFC6572]
Xia, F., et al, "RADIUS Support for Proxy Mobile IPv6", RFC 6572, June 2012.

7.2. Informative References

[RFC2868]
Zorn, G., Leifer, D., Rubens, A., Shriver, J., Holdrege, M., and I. Goyret, "RADIUS Attributes for Tunnel Protocol Support", RFC 2868, June 2000.

[RFC2869]
Rigney, C., et al, "RADIUS Extensions", RFC 2869, June 2000.

[RFC6929]
DeKok, A., and Lior, A., "Remote Authentication Dial In User

Service (RADIUS) Protocol Extensions", RFC 6929, April 2013.

[RFC7268]
Aboba, B, et al, "RADIUS Attributes for IEEE 802 Networks", RFC 7268, July 2015.

[RFC7499]
Perez-Mendez A., et al, "Support of Fragmentation of RADIUS Packets", RFC 7499, April 2015.

[PEN]
http://www.iana.org/assignments/enterprise-numbers

Acknowledgments

Stuff

Authors' Addresses

Alan DeKok
The FreeRADIUS Server Project

Email: aland@freeradius.org

Network Working Group   DeKok, Alan
INTERNET-DRAFT   FreeRADIUS
Updates: 2865,3162,6158,6572
Category: Standards Track
<draft-ietf-radext-datatypes-08.txt>
18 October 2016

Data Types in the Remote Authentication Dial-In User Service Protocol (RADIUS)

Abstract

RADIUS specifications have used data types for two decades without defining them as managed entities. During this time, RADIUS implementations have named the data types, and have used them in attribute definitions. This document updates the specifications to better follow established practice. We do this by naming the data types defined in RFC 6158, which have been used since at least the publication of RFC 2865. We provide an IANA registry for the data types, and update the RADIUS Attribute Type registry to include a "Data Type" field for each attribute. Finally, we recommend that authors of RADIUS specifications use these types in preference to existing practice. This document updates RFC 2865, 3162, 6158, and 6572.

Status of this Memo

Table of Contents

1. Introduction

RADIUS specifications have historically defined attributes in terms of name, value, and data type. Of these three pieces of information, the name is recorded by IANA in the RADIUS Attribute Type registry, but not otherwise managed or restricted, as discussed in [RFC6929] Section 2.7.1. The value is managed by IANA, and recorded in that registry. The data type is not managed or recorded in the RADIUS Attribute Type registry. Experience has shown that there is a need to create well known data types, and have them managed by IANA.

This document defines an IANA RADIUS Data Type registry, and updates the RADIUS Attribute Type registry to use those newly defined data types. It recommends how both specifications and implementations should use the data types. It extends the RADIUS Attribute Type registry to have a data type for each assigned attribute.

In this section, we review the use of data types in specifications and implementations. Whe highlight ambiguities and inconsistencies. The rest of this document is devoted to resolving those problems.

1.1. Specification Problems with Data Types

When attributes are defined in the specifications, the terms "Value" and "String" are used to refer to the contents of an attribute. However, these names are used recursively and inconsistently. We suggest that defining a field to recursively contain itself is problematic.

A number of data type names and definitions are given in [RFC2865] Section 5, at the bottom of page 25. These data types are named and clearly defined. However, this practice was not continued in later specifications.

Specifically, [RFC2865] defines attributes of data type "address" to carry IPv4 addresses. Despite this definition, [RFC3162] defines attributes of data type "Address" to carry IPv6 addresses. We suggest that the use of the word "address" to refer to disparate data types is problematic.

Other failures are that [RFC3162] does not give a data type name and definition for the data types IPv6 address, Interface-Id, or IPv6 prefix. [RFC2869] defines Event-Timestamp to carry a time, but does not re-use the "time" data type defined in [RFC2865]. Instead, it just repeats the "time" definition. [RFC6572] defines multiple attributes which carry IPv4 prefixes. However, an "IPv4 prefix" data type is not named, defined as a data type, or called out as an addition to RADIUS. Further, [RFC6572] does not follow the recommendations of [RFC6158], and does not explain why it fails to follow those recommendations.

These ambiguities and inconsistencies need to be resolved.

1.2. Implementation Problems with Data Types

RADIUS implementations often use "dictionaries" to map attribute names to type values, and to define data types for each attribute. The data types in the dictionaries are defined by each implementation, but correspond to the "ad hoc" data types used in the specifications.

In effect, implementations have seen the need for well-defined data types, and have created them. It is time for RADIUS specifications to follow this practice.

1.3. No Mandated Changes

This document mandates no changes to any RADIUS implementation, past, present, or future. It instead documents existing practice, in order to simplify the process of writing RADIUS specifications, to clarify the interpretation of RADIUS standards, and to improve the communication between specification authors and IANA.

This document suggests that implementations SHOULD use the data types defined here, in preference to any "ad hoc" data types currently in use. This suggestion should have minimal effect on implementations, as most "ad hoc" data types are compatible with the ones defined here. Any difference will typically be limited to the name of the data type.

This document updates [RFC6158] to permit the data types defined in the "Data Type registry" as "basic data types", as per Section 2.1 of that document. The recommendations of [RFC6158] are otherwise unchanged.

1.4. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Use of Data Types

The Data Types can be used in two places: specifications, and implementations. This section discusses both uses, and gives guidance on using the data types.

2.1. Specification Use of Data Types

In this section, we give recommendations for how specifications should be written using data types. We first describe how attribute field names can be consistently named. We then describe how attribute definitions should use the data types, and deprecate the use of "ASCII art" for attribute definitions. We suggest a format for new attribute definitions. This format includes recommended fields, and suggestions for how those fields should be described.

Finally, we make recommendations for how new data types should be defined.

2.1.1. Field Names for Attribute Values

Previous specifications used inconsistent and conflicting names for the contents of RADIUS attributes. For example, the term "Value" is used in [RFC2865] Section 5 to define a field which carries the contents of attribute. It is then used in later sections as the sub-field of attribute contents. The result is that the field is defined as recursively containing itself. Similarly, "String" is used both as a data type, and as a sub-field of other data types.

We correct this ambiguity by using context-specific names for various fields of attributes and data types. It then becomes clear that, for example, that a field called "VSA-Data" must contain different data than a field called "EVS-Data". Each new name is defined where it is used.

We also define the following term:

Attr-Data

The "Value" field of an Attribute as defined in [RFC2865] Section 5. The contents of this field MUST be of a valid data type as defined in the RADIUS Data Type registry.

We consistently use "Attr-Data" to refer to the contents of an attribute, instead of the more ambiguous name "Value". It is RECOMMENDED that new specifications follow this practice.

We consistently use "Value" to refer to the contents of a data type, where that data type is simple. For example, an "integer" can have a "Value". In contrast, a Vendor-Specific attribute carries complex information, and thus cannot have a "Value".

For data types which carry complex information, we name the fields based on the data type. For example, a Vendor-Specific attribute is defined to carry a "vsa" data type, and the contents of that data type are described herein as "VSA-Data".

These terms are used in preference to the term "String", which was previously used in ambiguous ways. It is RECOMMENDED that future specifications use type-specific names, and the same naming scheme for new types. This use will maintain consistent definitions, and help to avoid ambiguities.

2.1.2. Attribute Definitions using Data Types

New RADIUS specifications MUST define attributes using data types from the RADIUS Data Type registry. The specification may, of course, define a new data type update the "Data Types" registry, and use the new data type, all in the same document. The guidelines given in [RFC6929] MUST be followed when defining a new data type.

Attributes can usually be completely described via the Attribute Type value, name, and data type. The use of "ASCII art" is then limited only to the definition of new data types, and for complex data types.

Use of the new extended attributes [RFC6929] makes ASCII art even more problematic. An attribute can be allocated from any of the extended spaces, with more than one option for attribute header format. This allocation decision is made after the specification has been accepted for publication. As the allocation affects the format of the attribute header, it is essentially impossible to create the correct ASCII art prior to final publication. Allocation from the different spaces also changes the value of the Length field, also making it difficult to define it correctly prior to final publication of the document.

It is therefore RECOMMENDED that "ASCII art" diagrams not be used for new RADIUS attribute specifications.

2.1.3. Format of Attribute Definitions

When defining a new attribute, the following fields SHOULD be given:

Description

A description of the meaning and interpretation of the attribute.

Type

The Attribute Type value, given in the "dotted number" notation from [RFC6929]. Specifications can often leave this as "TBD", and request that IANA fill in the allocated values.

Length

A description of the length of the attribute. For attributes of variable length, a maximum length SHOULD be given. Since the Length may depend on the Type, the definition of Length may be affected by IANA allocations.

Data Type

One of the named data types from the RADIUS Data Type registry.

Value

A description of any attribute-specific limitations on the values carried by the specified data type. If there are no attribute-specific limitations, then the description of this field can be omitted, so long as the Description field is sufficiently explanatory.

Where the values are limited to a subset of the possible range, valid range(s) MUST be defined.

For attributes of data type "enum", a list of enumerated values and names MUST be given, as with [RFC2865] Section 5.6.

Using a consistent format for attribute definitions helps to make the definitions clearer.

2.1.4. Defining a New Data Type

When a specification needs to define a new data type, it SHOULD follow the format used by the definitions in Section 3 of this document. The text at the start of the data type definition MUST describe the data type, including the expected use, and why a new data type is required. That text SHOULD include limits on expected values, and why those limits exist. The field’s "Name", "Value", "Length", and "Format", MUST be given, along with values.

The "Name" field SHOULD be a single name, all lower-case.

Contractions such as "ipv4addr" are RECOMMENDED where they add clarity.

We note that the use of "Value" in the RADIUS Data Type registry can be confusing. That name is also used in attribute definitions, but with a different meaning. We trust that the meaning here is clear from the context.

The "Value" field SHOULD be given as to be determined or "TBD" in specifications. That number is assigned by IANA.

The "Format" field SHOULD be defined with "ASCII art" in order to have a precise definition. Machine-readable formats are also RECOMMENDED.

The definition of a new data type should be done only when absolutely necessary. We do not expect a need for a large number of new data types. When defining a new data type, the guideliness of [RFC6929] with respect to data types MUST be followed.

It is RECOMMENDED that vendors not define "vendor specific" data types. As discussed in [RFC6929], those data types are rarely necessary, and can cause interoperability problems.

Any new data type MUST have unique name in the RADIUS Data Type registry. The number of the data type will be assigned by IANA.

2.2. Implementation Use of Data Types

Implementations not supporting a particular data type MUST treat attributes of that data type as being of data type "string", as defined in Section 3.6. It is RECOMMENDED that such attributes be treated as "invalid attributes", as defined in [RFC6929] Section 2.8.

Where the contents of a data type do not match the definition, implementations MUST treat the the enclosing attribute as being an "invalid attribute". This requirement includes, but is not limited to, the following situations:

* Attributes with values outside of the allowed range(s) for the data type, e.g. as given in the data types "integer", "ipv4addr", "ipv6addr", "ipv4prefix", "ipv6prefix", or "enum".

* "text" attributes where the contents do not match the required format,

* Attributes where the length is shorter or longer than the allowed length(s) for the given data type,

The requirements for "reserved" fields are more difficult to quantify. Implementations SHOULD be able to receive and process attributes where "reserved" fields are non-zero. We do not, however, define any "correct" processing of such attributes. Instead, specifications which define new meaning for "reserved" fields SHOULD describe how the new meaning is compatible with older implementations. We expect that such descriptions are derived from practice. Implementations MUST set "reserved" fields to zero when creating attributes.

3. Data Type Definitions

This section defines the new data types. For each data type, it gives a definition, a name, a number, a length, and an encoding format. Where relevant, it describes subfields contained within the data type. These definitions have no impact on existing RADIUS implementations. There is no requirement that implementations use these names.

Where possible, the name of each data type has been taken from previous specifications. In some cases, a different name has been chosen. The change of name is sometimes required to avoid ambiguity (i.e. "address" versus "Address"). Otherwise, the new name has been chosen to be compatible with [RFC2865], or with use in common implementations. In some cases, new names are chosen to clarify the interpretation of the data type.

The numbers assigned herein for the data types have no meaning other than to permit them to be tracked by IANA. As RADIUS does not encode information about data types in a packet, the numbers assigned to a data type will never occur in a packet. It is RECOMMENDED that new implementations use the names defined in this document, in order to avoid confusion. Existing implementations may choose to use the names defined here, but that is not required.

The encoding of each data type is taken from previous specifications. The fields are transmitted from left to right.

Where the data types have inter-dependencies, the simplest data type is given first, and dependent ones are given later.

We do not create specific data types for the "tagged" attributes defined in [RFC2868]. That specification defines the "tagged" attributes as being backwards compatible with pre-existing data types. In addition, [RFC6158] Section 2.1 says that "tagged" attributes should not be used. There is therefore no benefit to defining additional data types for these attributes. We trust that implementors will be aware that tagged attributes must be treated differently from non-tagged attributes of the same data type.

Similarly, we do not create data types for some attributes having complex structure, such as CHAP-Password, ARAP-Features, or Location-Information. We need to strike a balance between correcting earlier mistakes, and making this document more complex. In some cases, it is better to treat complex attributes as being of type "string", even though they need to be interpreted by RADIUS implementations. The guidelines given in Section 6.3 of [RFC6929] were used to make this determination.

3.1. integer

The "integer" data type encodes a 32-bit unsigned integer in network byte order. Where the range of values for a particular attribute is limited to a sub-set of the values, specifications MUST define the valid range. Attributes with Values outside of the allowed ranges SHOULD be treated as "invalid attributes".

Name

integer

Value

1

Length

Four octets

Format

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Value                                                     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.2. enum

The "enum" data type encodes a 32-bit unsigned integer in network byte order. It differs from the "integer" data type only in that it is used to define enumerated types, such as Service-Type (Section 5.6 of [RFC2865]). Specifications MUST define a valid set of enumerated values, along with a unique name for each value. Attributes with Values outside of the allowed enumerations SHOULD be treated as "invalid attributes".

Name

enum

Value

2

Length

Four octets

Format

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Value                                                     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.3. time

The "time" data type encodes time as a 32-bit unsigned value in network byte order and in seconds since 00:00:00 UTC, January 1, 1970. We note that dates before the year 2016 are likely to indicate configuration errors, or lack of access to the correct time.

Note that the "time" attribute is defined to be unsigned, which means it is not subject to a signed integer overflow in the year 2038.

Name

time

Value

3

Length

Four octets

Format

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Time                                                      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.4. text

The "text" data type encodes UTF-8 text [RFC3629]. The maximum length of the text is given by the encapsulating attribute. Where the range of lengths for a particular attribute is limited to a sub-set of possible lengths, specifications MUST define the valid range(s). Attributes with length outside of the allowed values SHOULD be treated as "invalid attributes".

Attributes of type "text" which are allocated in the standard space (Section 1.2 of [RFC6929]) are limited to no more than 253 octets of data. Attributes of type "text" which are allocated in the extended space can be longer. In both cases, these limits are reduced when the data is encapsulated inside of an another attribute.

Where the text is intended to carry data in a particular format, (e.g. Framed-Route), the format MUST be given. The specification SHOULD describe the format in a machine-readable way, such as via Augmented Backus-Naur Form (ABNF) [RFC5234]. Attributes with values not matching the defined format SHOULD be treated as "invalid attributes".

Note that the "text" data type does not terminate with a NUL octet (hex 00). The Attribute has a Length field and does not use a terminator. Texts of length zero (0) MUST NOT be sent; omit the entire attribute instead.

Name

text

Value

4

Length

One or more octets.

Format

```
 0
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-
|  Value    ...
+-+-+-+-+-+-+-+-
```

3.5. string

The "string" data type encodes binary data, as a sequence of undistinguished octets. Where the range of lengths for a particular attribute is limited to a sub-set of possible lengths, specifications MUST define the valid range(s). Attributes with length outside of the allowed values SHOULD be treated as "invalid attributes".

Attributes of type "string" which are allocated in the standard space (Section 1.2 of [RFC6929]) are limited to no more than 253 octets of data. Attributes of type "string" which are allocated in the extended space can be longer. In both cases, these limits are reduced when the data is encapsulated inside of an another attribute.

Note that the "string" data type does not terminate with a NUL octet (hex 00). The Attribute has a Length field and does not use a terminator. Strings of length zero (0) MUST NOT be sent; omit the entire attribute instead. a Where there is a need to encapsulate complex data structures, and TLVs cannot be used, the "string" data type MUST be used. This requirement includes encapsulation of data structures defined outside of RADIUS, which are opaque to the RADIUS infrastucture. It also includes encapsulation of some data structures which are not opaque to RADIUS, such as the contents of CHAP-Password.

There is little reason to define a new RADIUS data type for only one attribute. However, where the complex data type cannot be represented as TLVs, and is expected to be used in many attributes, a new data type SHOULD be defined.

These requirements are stronger than [RFC6158], which makes the above encapsulation a "SHOULD". This document defines data types for use in RADIUS, so there are few reasons to avoid using them.

Name

   string

Value

   5

Length

   One or more octets.

Format

```
    0
    0 1 2 3 4 5 6 7
   +-+-+-+-+-+-+-+-
   |  Octets    ...
   +-+-+-+-+-+-+-+-
```

3.6. concat

The "concat" data type permits the transport of more than 253 octets of data in a "standard space" [RFC6929] attribute. It is otherwise identical to the "string" data type.

If multiple attributes of this data type are contained in a packet, all attributes of the same type code MUST be in order and they MUST be consecutive attributes in the packet.

The amount of data transported in a "concat" data type can be no more than the RADIUS packet size. In practice, the requirement to transport multiple attributes means that the limit may be substantially smaller than one RADIUS packet. As a rough guide, is RECOMMENDED that this data type transport no more than 2048 octets of data.

The "concat" data type MAY be used for "standard space" attributes. It MUST NOT be used for attributes in the "short extended space" or the "long extended space". It MUST NOT be used in any field or subfields of the following data types: "tlv", "vsa", "extended", "long-extended", or "evs".

Name

   concat

Value

   6

Length

   One or more octets.

Format

```
 0
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-
|  Octets    ...
+-+-+-+-+-+-+-+-
```

3.7. ifid

The "ifid" data type encodes an Interface-Id as an 8 octet IPv6 Interface Identifier network byte order.

Name

ifid

Value

7

Length

Eight octets

Format

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Interface-ID ...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        ... Interface-ID                                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.8. ipv4addr

The "ipv4addr" data type encodes an IPv4 address in network byte order. Where the range of address for a particular attribute is limited to a sub-set of possible addresses, specifications MUST define the valid range(s). Attributes with Addresses outside of the allowed range(s) SHOULD be treated as "invalid attributes".

Name

ipv4addr

Value

8

Length

Four octets

Format

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Address                                                   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.9. ipv6addr

The "ipv6addr" data type encodes an IPv6 address in network byte order. Where the range of address for a particular attribute is limited to a sub-set of possible addresses, specifications MUST define the valid range(s). Attributes with Addresses outside of the allowed range(s) SHOULD be treated as "invalid attributes".

Name

   ipv6addr

Value

   9

Length

   Sixteen octets

Format

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |     Address ...
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
           ... Address ...
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
           ... Address ...
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
           ... Address                                                |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.10. ipv6prefix

The "ipv6prefix" data type encodes an IPv6 prefix, using both a prefix length and an IPv6 address in network byte order. Where the range of prefixes for a particular attribute is limited to a sub-set of possible prefixes, specifications MUST define the valid range(s). Attributes with Addresses outside of the allowed range(s) SHOULD be treated as "invalid attributes".

Attributes with a Prefix-Length field having value greater than 128
MUST be treated as "invalid attributes".

Name

ipv6prefix

Value

10

Length

At least two, and no more than eighteen octets.

Format

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |    Reserved   | Prefix-Length |  Prefix ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         ... Prefix ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         ... Prefix ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         ... Prefix                                                |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Subfields

Reserved

This field, which is reserved and MUST be present, is always
set to zero. This field is one octet in length.

Prefix-Length

The length of the prefix, in bits. At least 0 and no larger
than 128. This field is one octet in length.

Prefix

The Prefix field is up to 16 octets in length. Bits outside of
the Prefix-Length, if included, MUST be zero.

The Prefix field SHOULD NOT contain more octets than necessary
to encode the Prefix.

3.11. ipv4prefix

The "ipv4prefix" data type encodes an IPv4 prefix, using both a prefix length and an IPv4 address in network byte order. Where the range of prefixes for a particular attribute is limited to a sub-set of possible prefixes, specifications MUST define the valid range(s). Attributes with Addresses outside of the allowed range(s) SHOULD be treated as "invalid attributes".

Attributes with a Prefix-Length field having value greater than 32 MUST be treated as "invalid attributes".

Name

ipv4prefix

Value

11

Length

six octets

Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Reserved   | Prefix-Length |  Prefix ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     ... Prefix                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Subfields

Reserved

This field, which is reserved and MUST be present, is always set to zero. This field is one octet in length.

Note that this definition differs from that given in [RFC6572]. See Prefix-Length, below, for an explanation.

Prefix-Length

The length of the prefix, in bits. The values MUST be no larger than 32. This field is one octet in length.

Note that this definition differs from that given in [RFC6572].

As compared to [RFC6572], the Prefix-Length field has increased in size by two bits, both of which must be zero. The Reserved field has decreased in size by two bits. The result is that both fields are aligned on octet boundaries, which removes the need for bit masking of the fields.

Since [RFC6572] required the Reserved field to be zero, the definition here is compatible with the definition in the original specification.

Prefix

The Prefix field is 4 octets in length. Bits outside of the Prefix-Length MUST be zero. Unlike the "ipv6prefix" data type, this field is fixed length. If the address is all zeros (i.e. "0.0.0.0", then the Prefix-Length MUST be set to 32.

3.12. integer64

The "integer64" data type encodes a 64-bit unsigned integer in network byte order. Where the range of values for a particular attribute is limited to a sub-set of the values, specifications MUST define the valid range(s). Attributes with Values outside of the allowed range(s) SHOULD be treated as "invalid attributes".

Name

integer64

Value

12

Length

Eight octets

Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Value ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      ... Value                                                 |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.13. tlv

The "tlv" data type encodes a type-length-value, as defined in [RFC6929] Section 2.3.

Name

tlv

Value

13

Length

Three or more octets

Format

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |   TLV-Type    |  TLV-Length   |     TLV-Data ...
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Subfields

TLV-Type

This field is one octet. Up-to-date values of this field are specified according to the policies and rules described in [RFC6929] Section 10. Values of 254-255 are "Reserved" for use by future extensions to RADIUS. The value 26 has no special meaning, and MUST NOT be treated as a Vendor Specific attribute.

The TLV-Type is meaningful only within the context defined by "Type" fields of the encapsulating Attributes, using the dotted-number notation introduced in [RFC6929].

A RADIUS server MAY ignore Attributes with an unknown "TLV-Type".

A RADIUS client MAY ignore Attributes with an unknown "TLV-Type".

A RADIUS proxy SHOULD forward Attributes with an unknown "TLV-Type" verbatim.

TLV-Length

The TLV-Length field is one octet, and indicates the length of this TLV including the TLV-Type, TLV-Length and TLV-Value fields. It MUST have a value between 3 and 255. If a client or server receives a TLV with an invalid TLV-Length, then the attribute which encapsulates that TLV MUST be considered to be an "invalid attribute", and handled as per [RFC6929] Section 2.8.

TLVs having TLV-Length of two (2) MUST NOT be sent; omit the entire TLV instead.

TLV-Data

The TLV-Data field is one or more octets and contains information specific to the Attribute. The format and length of the TLV-Data field is determined by the TLV-Type and TLV-Length fields.

The TLV-Data field MUST contain only known RADIUS data types. The TLV-Data field MUST NOT contain any of the following data types: "concat", "vsa", "extended", "long-extended", or "evs".

3.14. vsa

The "vsa" data type encodes Vendor-Specific data, as given in [RFC2865] Section 5.26. It is used only in the Attr-Data field of a Vendor-Specific Attribute. It MUST NOT appear in the contents of any other data type.

Where an implementation determines that an attribute of data type "vsa" contains data which does not match the expected format, it SHOULD treat that attribute as being an "invalid attribute".

Name

vsa

Value

14

Length

      Five or more octets

   Format

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                            Vendor-Id                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  VSA-Data ....
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   Subfields

      Vendor-Id

         The 4 octets are the Network Management Private Enterprise Code [PEN] of the Vendor in network byte order.

      VSA-Data

         The VSA-Data field is one or more octets. The actual format of the information is site or application specific, and a robust implementation SHOULD support the field as undistinguished octets.

         The codification of the range of allowed usage of this field is outside the scope of this specification.

         The "vsa" data type SHOULD contain as a sequence of "tlv" data types. The interpretation of the TLV-Type and TLV-Data fields are dependent on the vendor’s definition of that attribute.

         The "vsa" data type MUST be used as contents of the Attr-Data field of the Vendor-Specific attribute. The "vsa" data type MUST NOT appear in the contents of any other data type.

3.15. extended

   The "extended" data type encodes the "Extended Type" format, as given in [RFC6929] Section 2.1. It is used only in the Attr-Data field of an Attribute allocated from the "standard space". It MUST NOT appear in the contents of any other data type.

   Name

      extended

Value

15

Length

Two or more octets

Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Extended-Type | Ext-Data ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Subfields

Extended-Type

The Extended-Type field is one octet. Up-to-date values of this field are specified according to the policies and rules described in [RFC6929] Section 10. Unlike the Type field defined in [RFC2865] Section 5, no values are allocated for experimental or implementation-specific use. Values 241-255 are reserved and MUST NOT be used.

The Extended-Type is meaningful only within a context defined by the Type field. That is, this field may be thought of as defining a new type space of the form "Type.Extended-Type". See [RFC6929] Section 2.5 for additional discussion.

A RADIUS server MAY ignore Attributes with an unknown "Type.Extended-Type".

A RADIUS client MAY ignore Attributes with an unknown "Type.Extended-Type".

Ext-Data

The contents of this field MUST be a valid data type as defined in the RADIUS Data Type registry. The Ext-Data field MUST NOT contain any of the following data types: "concat", "vsa", "extended", "long-extended", or "evs".

The Ext-Data field is one or more octets.

Implementations supporting this specification MUST use the

Identifier of "Type.Extended-Type" to determine the interpretation of the Ext-Data field.

3.16. long-extended

The "long-extended" data type encodes the "Long Extended Type" format, as given in [RFC6929] Section 2.2. It is used only in the Attr-Data field of an Attribute. It MUST NOT appear in the contents of any other data type.

Name

long-extended

Value

16

Length

Three or more octets

Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Extended-Type |M|T| Reserved  | Ext-Data ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Subfields

Extended-Type

This field is identical to the Extended-Type field defined above in Section 3.15.

M (More)

The More field is one (1) bit in length, and indicates whether or not the current attribute contains "more" than 251 octets of data. The More field MUST be clear (0) if the Length field has value less than 255. The More field MAY be set (1) if the Length field has value of 255.

If the More field is set (1), it indicates that the Ext-Data field has been fragmented across multiple RADIUS attributes.

When the More field is set (1), the attribute MUST have a Length field of value 255; there MUST be an attribute following this one; and the next attribute MUST have both the same Type and Extended Type. That is, multiple fragments of the same value MUST be in order and MUST be consecutive attributes in the packet, and the last attribute in a packet MUST NOT have the More field set (1).

That is, a packet containing a fragmented attribute needs to contain all fragments of the attribute, and those fragments need to be contiguous in the packet. RADIUS does not support inter-packet fragmentation, which means that fragmenting an attribute across multiple packets is impossible.

If a client or server receives an attribute fragment with the "More" field set (1), but for which no subsequent fragment can be found, then the fragmented attribute is considered to be an "invalid attribute", and handled as per [RFC6929] Section 2.8.

T (Truncation)

This field is one bit in size and is called "T" for Truncation. It indicates that the attribute is intentionally truncated in this chunk and is to be continued in the next chunk of the sequence. The combination of the M flag and the T flag indicates that the attribute is fragmented (M flag) but that all the fragments are not available in this chunk (T flag). Proxies implementing [RFC6929] will see these attributes as invalid (they will not be able to reconstruct them), but they will still forward them, as Section 5.2 of [RFC6929] indicates that they SHOULD forward unknown attributes anyway.

Please see [RFC7499] for further discussion of the uses of this flag.

Reserved

This field is 6 bits long, and is reserved for future use. Implementations MUST set it to zero

(0) when encoding an attribute for sending in a packet. The contents SHOULD be ignored on reception.

Future specifications may define additional meaning for this field. Implementations therefore MUST NOT treat this field as invalid if it is non-zero.

Ext-Data

The contents of this field MUST be a valid data type as defined in the RADIUS Data Type registry. The Ext-Data field MUST NOT contain any of the following data types: "concat", "vsa", "extended", "long-extended", or "evs".

The Ext-Data field is one or more octets.

Implementations supporting this specification MUST use the Identifier of "Type.Extended-Type" to determine the interpretation of the Ext-Data field.

The length of the data MUST be taken as the sum of the lengths of the fragments (i.e. Ext-Data fields) from which it is constructed. Any interpretation of the resulting data MUST occur after the fragments have been reassembled. If the reassembled data does not match the expected format, each fragment MUST be treated as an "invalid attribute", and the reassembled data MUST be discarded.

We note that the maximum size of a fragmented attribute is limited only by the RADIUS packet length limitation. Implementations MUST be able to handle the case where one fragmented attribute completely fills the packet.

3.17. evs

The "evs" data type encodes an "Extended Vendor-Specific" attribute, as given in [RFC6929] Section 2.4. The "evs" data type is used solely to extend the Vendor Specific space. It MAY appear inside of an "extended" or a "long-extended" data type. It MUST NOT appear in the contents of any other data type.

Where an implementation determines that an attribute of data type "evs" contains data which does not match the expected format, it SHOULD treat that attribute as being an "invalid attribute".

Name

   evs

Value

   17

Length

   Six or more octets

Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Vendor-Id                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Vendor-Type  |  EVS-Data ....
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Subfields

   Vendor-Id

      The 4 octets are the Network Management Private Enterprise Code [PEN] of the Vendor in network byte order.

   Vendor-Type

      The Vendor-Type field is one octet. Values are assigned at the sole discretion of the Vendor.

EVS-Data

The EVS-Data field is one or more octets. It SHOULD encapsulate a previously defined RADIUS data type. Non-standard data types SHOULD NOT be used. We note that the EVS-Data field may be of data type "tlv".

The actual format of the information is site or application specific, and a robust implementation SHOULD support the field as undistinguished octets. We recognise that Vendors have complete control over the contents and format of the Ext-Data field, while at the same time recommending that good practices be followed.

Further codification of the range of allowed usage of this field is outside the scope of this specification.

4. Updated Registries

This section defines a new IANA registry for RADIUS data types, and then updates the existing RADIUS Attribute Type registry to use the data types from the new registry.

4.1. Create a Data Type Registry

This section defines a new registry located under "RADIUS Types", called "Data Type". The "Registration Procedures" for the Data Type registry are "Standards Action".

The Data Type registry contains three columns of data, as follows.

Value

The number of the data type. The value field is an artifact of the registry, and has no on-the-wire meaning.

Name

The name of the data type. The name field is used only for the registry, and has no on-the-wire meaning.

Reference

The specification where the data type was defined.

The initial contents of the registry are as follows.

Value	Description	Reference
1	integer	[RFC2865], TBD
2	enum	[RFC2865], TBD
3	time	[RFC2865], TBD
4	text	[RFC2865], TBD
5	string	[RFC2865], TBD
6	concat	TBD
7	ifid	[RFC3162], TBD
8	ipv4addr	[RFC2865], TBD
9	ipv6addr	[RFC3162], TBD
10	ipv6prefix	[RFC3162], TBD
11	ipv4prefix	[RFC6572], TBD
12	integer64	[RFC6929], TBD
13	tlv	[RFC6929], TBD
14	evs	[RFC6929], TBD
15	extended	[RFC6929], TBD
16	long-extended	[RFC6929], TBD

4.2. Updates to the Attribute Type Registry

This section updates the RADIUS Attribute Type registry to have a new column, which is inserted in between the existing "Description" and "Reference" columns. The new column is named "Data Type". The contents of that column are the name of a data type, corresponding to the attribute in that row, or blank if the attribute type is unassigned. The name of the data type is taken from the RADIUS Data Type registry, as defined above.

The existing registration requirements for the RADIUS Attribute Type registry are otherwise unchanged.

NOTE TO RFC EDITOR: Before the document is published, please remove this note, and the following text in this section.

The updated registry follows in CSV format.

```
Value,Description,Data Type,Reference
1,User-Name,text,[RFC2865]
2,User-Password,string,[RFC2865]
3,CHAP-Password,string,[RFC2865]
4,NAS-IP-Address,ipv4addr,[RFC2865]
5,NAS-Port,integer,[RFC2865]
6,Service-Type,enum,[RFC2865]
7,Framed-Protocol,enum,[RFC2865]
8,Framed-IP-Address,ipv4addr,[RFC2865]
9,Framed-IP-Netmask,ipv4addr,[RFC2865]
10,Framed-Routing,enum,[RFC2865]
```

```
   11,Filter-Id,text,[RFC2865]
   12,Framed-MTU,integer,[RFC2865]
   13,Framed-Compression,enum,[RFC2865]
   14,Login-IP-Host,ipv4addr,[RFC2865]
   15,Login-Service,enum,[RFC2865]
   16,Login-TCP-Port,integer,[RFC2865]
   17,Unassigned,,
   18,Reply-Message,text,[RFC2865]
   19,Callback-Number,text,[RFC2865]
   20,Callback-Id,text,[RFC2865]
   21,Unassigned,,
   22,Framed-Route,text,[RFC2865]
   23,Framed-IPX-Network,ipv4addr,[RFC2865]
   24,State,string,[RFC2865]
   25,Class,string,[RFC2865]
   26,Vendor-Specific,vsa,[RFC2865]
   27,Session-Timeout,integer,[RFC2865]
   28,Idle-Timeout,integer,[RFC2865]
   29,Termination-Action,enum,[RFC2865]
   30,Called-Station-Id,text,[RFC2865]
   31,Calling-Station-Id,text,[RFC2865]
   32,NAS-Identifier,text,[RFC2865]
   33,Proxy-State,string,[RFC2865]
   34,Login-LAT-Service,text,[RFC2865]
   35,Login-LAT-Node,text,[RFC2865]
   36,Login-LAT-Group,string,[RFC2865]
   37,Framed-AppleTalk-Link,integer,[RFC2865]
   38,Framed-AppleTalk-Network,integer,[RFC2865]
   39,Framed-AppleTalk-Zone,text,[RFC2865]
   40,Acct-Status-Type,enum,[RFC2866]
   41,Acct-Delay-Time,integer,[RFC2866]
   42,Acct-Input-Octets,integer,[RFC2866]
   43,Acct-Output-Octets,integer,[RFC2866]
   44,Acct-Session-Id,text,[RFC2866]
   45,Acct-Authentic,enum,[RFC2866]
   46,Acct-Session-Time,integer,[RFC2866]
   47,Acct-Input-Packets,integer,[RFC2866]
   48,Acct-Output-Packets,integer,[RFC2866]
   49,Acct-Terminate-Cause,enum,[RFC2866]
   50,Acct-Multi-Session-Id,text,[RFC2866]
   51,Acct-Link-Count,integer,[RFC2866]
   52,Acct-Input-Gigawords,integer,[RFC2869]
   53,Acct-Output-Gigawords,integer,[RFC2869]
   54,Unassigned,,
   55,Event-Timestamp,time,[RFC2869]
   56,Egress-VLANID,integer,[RFC4675]
   57,Ingress-Filters,enum,[RFC4675]
   58,Egress-VLAN-Name,text,[RFC4675]
   59,User-Priority-Table,string,[RFC4675]
   60,CHAP-Challenge,string,[RFC2865]
   61,NAS-Port-Type,enum,[RFC2865]
   62,Port-Limit,integer,[RFC2865]
   63,Login-LAT-Port,text,[RFC2865]
   64,Tunnel-Type,enum,[RFC2868]
   65,Tunnel-Medium-Type,enum,[RFC2868]
   66,Tunnel-Client-Endpoint,text,[RFC2868]
   67,Tunnel-Server-Endpoint,text,[RFC2868]
   68,Acct-Tunnel-Connection,text,[RFC2867]
   69,Tunnel-Password,string,[RFC2868]
   70,ARAP-Password,string,[RFC2869]
   71,ARAP-Features,string,[RFC2869]
   72,ARAP-Zone-Access,enum,[RFC2869]
   73,ARAP-Security,integer,[RFC2869]
   74,ARAP-Security-Data,text,[RFC2869]
   75,Password-Retry,integer,[RFC2869]
   76,Prompt,enum,[RFC2869]
   77,Connect-Info,text,[RFC2869]
   78,Configuration-Token,text,[RFC2869]
   79,EAP-Message,concat,[RFC2869]
   80,Message-Authenticator,string,[RFC2869]
   81,Tunnel-Private-Group-ID,text,[RFC2868]
   82,Tunnel-Assignment-ID,text,[RFC2868]
   83,Tunnel-Preference,integer,[RFC2868]
   84,ARAP-Challenge-Response,string,[RFC2869]
   85,Acct-Interim-Interval,integer,[RFC2869]
   86,Acct-Tunnel-Packets-Lost,integer,[RFC2867]
   87,NAS-Port-Id,text,[RFC2869]
   88,Framed-Pool,text,[RFC2869]
   89,CUI,string,[RFC4372]
   90,Tunnel-Client-Auth-ID,text,[RFC2868]
   91,Tunnel-Server-Auth-ID,text,[RFC2868]
   92,NAS-Filter-Rule,text,[RFC4849]
   93,Unassigned,,
   94,Originating-Line-Info,string,[RFC7155]
   95,NAS-IPv6-Address,ipv6addr,[RFC3162]
   96,Framed-Interface-Id,ifid,[RFC3162]
   97,Framed-IPv6-Prefix,ipv6prefix,[RFC3162]
   98,Login-IPv6-Host,ipv6addr,[RFC3162]
   99,Framed-IPv6-Route,text,[RFC3162]
   100,Framed-IPv6-Pool,text,[RFC3162]
   101,Error-Cause Attribute,enum,[RFC3576]
   102,EAP-Key-Name,string,[RFC4072][RFC7268]
   103,Digest-Response,text,[RFC5090]
   104,Digest-Realm,text,[RFC5090]
   105,Digest-Nonce,text,[RFC5090]
   106,Digest-Response-Auth,text,[RFC5090]
   107,Digest-Nextnonce,text,[RFC5090]
   108,Digest-Method,text,[RFC5090]
   109,Digest-URI,text,[RFC5090]
   110,Digest-Qop,text,[RFC5090]
   111,Digest-Algorithm,text,[RFC5090]
   112,Digest-Entity-Body-Hash,text,[RFC5090]
   113,Digest-CNonce,text,[RFC5090]
   114,Digest-Nonce-Count,text,[RFC5090]
   115,Digest-Username,text,[RFC5090]
   116,Digest-Opaque,text,[RFC5090]
   117,Digest-Auth-Param,text,[RFC5090]
   118,Digest-AKA-Auts,text,[RFC5090]
   119,Digest-Domain,text,[RFC5090]
   120,Digest-Stale,text,[RFC5090]
   121,Digest-HA1,text,[RFC5090]
   122,SIP-AOR,text,[RFC5090]
   123,Delegated-IPv6-Prefix,ipv6prefix,[RFC4818]
   124,MIP6-Feature-Vector,string,[RFC5447]
   125,MIP6-Home-Link-Prefix,ipv6prefix,[RFC5447]
   126,Operator-Name,text,[RFC5580]
   127,Location-Information,string,[RFC5580]
   128,Location-Data,string,[RFC5580]
   129,Basic-Location-Policy-Rules,string,[RFC5580]
   130,Extended-Location-Policy-Rules,string,[RFC5580]
   131,Location-Capable,enum,[RFC5580]
   132,Requested-Location-Info,enum,[RFC5580]
   133,Framed-Management-Protocol,enum,[RFC5607]
   134,Management-Transport-Protection,enum,[RFC5607]
   135,Management-Policy-Id,text,[RFC5607]
   136,Management-Privilege-Level,integer,[RFC5607]
   137,PKM-SS-Cert,concat,[RFC5904]
   138,PKM-CA-Cert,concat,[RFC5904]
   139,PKM-Config-Settings,string,[RFC5904]
   140,PKM-Cryptosuite-List,string,[RFC5904]
   141,PKM-SAID,text,[RFC5904]
   142,PKM-SA-Descriptor,string,[RFC5904]
   143,PKM-Auth-Key,string,[RFC5904]
   144,DS-Lite-Tunnel-Name,text,[RFC6519]
   145,Mobile-Node-Identifier,string,[RFC6572]
   146,Service-Selection,text,[RFC6572]
   147,PMIP6-Home-LMA-IPv6-Address,ipv6addr,[RFC6572]
   148,PMIP6-Visited-LMA-IPv6-Address,ipv6addr,[RFC6572]
   149,PMIP6-Home-LMA-IPv4-Address,ipv4addr,[RFC6572]
   150,PMIP6-Visited-LMA-IPv4-Address,ipv4addr,[RFC6572]
   151,PMIP6-Home-HN-Prefix,ipv6prefix,[RFC6572]
   152,PMIP6-Visited-HN-Prefix,ipv6prefix,[RFC6572]
   153,PMIP6-Home-Interface-ID,ifid,[RFC6572]
   154,PMIP6-Visited-Interface-ID,ifid,[RFC6572]
   155,PMIP6-Home-IPv4-HoA,ipv4prefix,[RFC6572]
   156,PMIP6-Visited-IPv4-HoA,ipv4prefix,[RFC6572]
   157,PMIP6-Home-DHCP4-Server-Address,ipv4addr,[RFC6572]
   158,PMIP6-Visited-DHCP4-Server-Address,ipv4addr,[RFC6572]
   159,PMIP6-Home-DHCP6-Server-Address,ipv6addr,[RFC6572]
   160,PMIP6-Visited-DHCP6-Server-Address,ipv6addr,[RFC6572]
   161,PMIP6-Home-IPv4-Gateway,ipv4addr,[RFC6572]
   162,PMIP6-Visited-IPv4-Gateway,ipv4addr,[RFC6572]
   163,EAP-Lower-Layer,enum,[RFC6677]
   164,GSS-Acceptor-Service-Name,text,[RFC7055]
   165,GSS-Acceptor-Host-Name,text,[RFC7055]
   166,GSS-Acceptor-Service-Specifics,text,[RFC7055]
   167,GSS-Acceptor-Realm-Name,text,[RFC7055]
   168,Framed-IPv6-Address,ipv6addr,[RFC6911]
   169,DNS-Server-IPv6-Address,ipv6addr,[RFC6911]
   170,Route-IPv6-Information,ipv6prefix,[RFC6911]
   171,Delegated-IPv6-Prefix-Pool,text,[RFC6911]
   172,Stateful-IPv6-Address-Pool,text,[RFC6911]
   173,IPv6-6rd-Configuration,tlv,[RFC6930]
   174,Allowed-Called-Station-Id,text,[RFC7268]
   175,EAP-Peer-Id,string,[RFC7268]
   176,EAP-Server-Id,string,[RFC7268]
   177,Mobility-Domain-Id,integer,[RFC7268]
   178,Preauth-Timeout,integer,[RFC7268]
   179,Network-Id-Name,string,[RFC7268]
   180,EAPoL-Announcement,concat,[RFC7268]
   181,WLAN-HESSID,text,[RFC7268]
   182,WLAN-Venue-Info,integer,[RFC7268]
   183,WLAN-Venue-Language,string,[RFC7268]
   184,WLAN-Venue-Name,text,[RFC7268]
   185,WLAN-Reason-Code,integer,[RFC7268]
   186,WLAN-Pairwise-Cipher,integer,[RFC7268]
   187,WLAN-Group-Cipher,integer,[RFC7268]
   188,WLAN-AKM-Suite,integer,[RFC7268]
   189,WLAN-Group-Mgmt-Cipher,integer,[RFC7268]
   190,WLAN-RF-Band,integer,[RFC7268]
   191,Unassigned,,
   192-223,Experimental Use,,[RFC3575]
   224-240,Implementation Specific,,[RFC3575]
   241,Extended-Attribute-1,extended,[RFC6929]
   241.1,Frag-Status,integer,[RFC7499]
   241.2,Proxy-State-Length,integer,[RFC7499]
   241.3,Response-Length,integer,[RFC7930]
   241.4,Original-Packet-Code,integer,[RFC7930]
   241.{5-25},Unassigned,,
   241.26,Extended-Vendor-Specific-1,evs,[RFC6929]
   241.{27-240},Unassigned,,
   241.{241-255},Reserved,,[RFC6929]
   242,Extended-Attribute-2,extended,[RFC6929]
   242.{1-25},Unassigned,,
   242.26,Extended-Vendor-Specific-2,evs,[RFC6929]
   242.{27-240},Unassigned,,
   242.{241-255},Reserved,,[RFC6929]
   243,Extended-Attribute-3,extended,[RFC6929]
   243.{1-25},Unassigned,,
   243.26,Extended-Vendor-Specific-3,evs,[RFC6929]
   243.{27-240},Unassigned,,
   243.{241-255},Reserved,,[RFC6929]
   244,Extended-Attribute-4,extended,[RFC6929]
   244.{1-25},Unassigned,,
   244.26,Extended-Vendor-Specific-4,evs,[RFC6929]
   244.{27-240},Unassigned,,
   244.{241-255},Reserved,,[RFC6929]
   245,Extended-Attribute-5,long-extended,[RFC6929]
   245.{1-25},Unassigned,,
   245.26,Extended-Vendor-Specific-5,evs,[RFC6929]
   245.{27-240},Unassigned,,
   245.{241-255},Reserved,,[RFC6929]
   246,Extended-Attribute-6,long-extended,[RFC6929]
   246.{1-25},Unassigned,,
   246.26,Extended-Vendor-Specific-6,evs,[RFC6929]
   246.{27-240},Unassigned,,
   246.{241-255},Reserved,,[RFC6929]
   247-255,Reserved,,[RFC3575]
```

5. Security Considerations

This specification is concerned solely with updates to IANA registries. As such, there are no security considerations with the document itself.

However, the use of inconsistent names and poorly-defined entities in a protocol is problematic. Inconsistencies in specifications can lead to security and interoperability problems in implementations. Further, having one canonical source for the definition of data types means an implementor has fewer specifications to read. The implementation work is therefore simpler, and is more likely to be correct.

The goal of this specification is to reduce ambiguities in the RADIUS protocol, which we believe will lead to more robust and more secure implementations.

6. IANA Considerations

IANA is instructed to create one new registry as described above in Section 4.1. The "TBD" text in that section should be replaced with the RFC number of this document when it is published.

IANA is instructed to update the RADIUS Attribute Type registry, as described above in Section 4.2.

IANA is instructed to require that all allocation requests in the RADIUS Attribute Type registry contain a "Data Type" field. That field is required to contain one of the "Data Type" names contained in the RADIUS Data Type registry.

IANA is instructed to require that updates to the RADIUS Data Type registry contain the following fields, with the associated instructions:

* Value. IANA is instructed to assign the next unused integer in sequence to new data type definitions.

* Name. IANA is instructed to require that this name be unique in the registry.

* Reference. IANA is instructed to update this field with a reference to the document which defines the data type.

7. References

7.1. Normative References

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

[RFC2865]
Rigney, C., Willens, S., Rubens, A. and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC3162]
Aboba, B., Zorn, G., and D. Mitton, "RADIUS and IPv6", RFC 3162, August 2001.

[RFC3629]
Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 3629, November 2003.

[RFC4072]
Eronen, P., et al, "Diameter Extensible Authentication Protocol (EAP) Application", RFC 4072, February 2013.

[RFC6158]
DeKok, A., and Weber, G., "RADIUS Design Guidelines", RFC 6158, March 2011.

[RFC6572]
Xia, F., et al, "RADIUS Support for Proxy Mobile IPv6", RFC 6572, June 2012.

[RFC7499]
Perez-Mendez, A. Ed., et al, "Support of Fragmentation of RADIUS Packets", RFC 7499, April 2015.

7.2. Informative References

[RFC2868]
Zorn, G., Leifer, D., Rubens, A., Shriver, J., Holdrege, M., and I. Goyret, "RADIUS Attributes for Tunnel Protocol Support", RFC 2868, June 2000.

[RFC2869]
Rigney, C., et al, "RADIUS Extensions", RFC 2869, June 2000.

[RFC5234]
Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 5234, January 2008.

[RFC6929]
DeKok, A., and Lior, A., "Remote Authentication Dial In User Service (RADIUS) Protocol Extensions", RFC 6929, April 2013.

[RFC7268]
Aboba, B, et al, "RADIUS Attributes for IEEE 802 Networks", RFC 7268, July 2015.

[RFC7499]
Perez-Mendez A., et al, "Support of Fragmentation of RADIUS Packets", RFC 7499, April 2015.

[PEN]
http://www.iana.org/assignments/enterprise-numbers

Acknowledgments

Thanks to the RADEXT WG for patience and reviews of this document.

Authors’ Addresses

Alan DeKok
The FreeRADIUS Server Project

Email: aland@freeradius.org

RADIUS Extensions Working Group S. Winter
Internet-Draft RESTENA
Intended status: Experimental M. McCauley
Expires: November 1, 2015 AirSpayce
April 30, 2015

NAI-based Dynamic Peer Discovery for RADIUS/TLS and RADIUS/DTLS
draft-ietf-radext-dynamic-discovery-15

Abstract

This document specifies a means to find authoritative RADIUS servers for a given realm. It is used in conjunction with either RADIUS/TLS and RADIUS/DTLS.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

RADIUS in all its current transport variants (RADIUS/UDP, RADIUS/TCP, RADIUS/TLS, RADIUS/DTLS) requires manual configuration of all peers (clients, servers).

Where more than one administrative entity collaborates for RADIUS authentication of their respective customers (a "roaming consortium"), the Network Access Identifier (NAI) [I-D.ietf-radext-nai] is the suggested way of differentiating users between those entities; the part of a username to the right of the @ delimiter in an NAI is called the user's "realm". Where many realms and RADIUS forwarding servers are in use, the number of realms to be forwarded and the corresponding number of servers to configure may be significant. Where new realms with new servers are added or details of existing servers change on a regular basis, maintaining a single monolithic configuration file for all these details may prove too cumbersome to be useful.

Furthermore, in cases where a roaming consortium consists of independently working branches (e.g. departments, national subsidiaries), each with their own forwarding servers, and who add or change their realm lists at their own discretion, there is additional complexity in synchronising the changed data across all branches.

Where realms can be partitioned (e.g. according to their top-level domain ending), forwarding of requests can be realised with a hierarchy of RADIUS servers, all serving their partition of the realm space. Figure 1 show an example of this hierarchical routing.

```
                                    +-------+
                                    |       |
                                    |   .   |
                                    |       |
                                    +---+---+
                                      / | \
                    +----------------/  |  \---------------------+
                    |                   |                        |
                    |                   |                        |
                    |                   |                        |
                 +--+---+            +--+--+                +----+---+
                 |      |            |     |                |        |
                 | .edu |    . . .   | .nl |      . . .     | .ac.uk |
                 |      |            |     |                |        |
                 +--+---+            +--+--+                +----+---+
                  / | \                 | \                      |
                 /  |  \                |  \                     |
                /   |   \               |   \                    |
         +-----+    |    +-----+        |    +------+            |
         |          |          |        |           |            |
         |          |          |        |           |            |
     +---+---+ +----+---+ +----+---+ +--+---+ +-----+----+ +-----+-----+
     |       | |        | |        | |      | |          | |           |
     |utk.edu| |utah.edu| |case.edu| |hva.nl| |surfnet.nl| |soton.ac.uk|
     |       | |        | |        | |      | |          | |           |
     +----+--+ +--------+ +--------+ +------+ +----+-----+ +-----------+
          |                                        |
          |                                        |
       +--+--+                                  +--+--+
       |     |                                  |     |
     +-+-----+-+                                |     |
     |         |                                +-----+
     +---------+
     user: paul@surfnet.nl             surfnet.nl Authentication server
```

Figure 1: RADIUS hierarchy based on Top-Level Domain partitioning

However, such partitioning is not always possible. As an example, in one real-life deployment, the administrative boundaries and RADIUS forwarding servers are are organised along country borders, but generic top-level domains such as .edu do not map to this choice of boundaries (see [I-D.wierenga-ietf-eduroam] for details). These situations can benefit significantly from a distributed mechanism for storing realm and server reachability information. This document describes one such mechanism: storage of realm-to-server mappings in DNS; realm-based request forwarding can then be realised without a static hierarchy such as in the following figure:

```
                                    ---------
                                   /         \
                          ---------           ------------
                         /                                \
                         |    DNS                          -
               ----------|                                  \
              /          \          surfnet.nl NAPTR?       |
        (1)  /            ----       -> radius.surfnet.nl   /
            /                 \                            /
           /                   --------           ---------
          /                            \---------/
         |
         |   ---------------------------------------
         |  /              (2) RADIUS              \
         |  |                                       |
     +---+---+ +----+---+ +----+---+ +--+---+ +-----+----+ +-----+-----+
     |       | |        | |        | |      | |          | |           |
     |utk.edu| |utah.edu| |case.edu| |hva.nl| |surfnet.nl| |soton.ac.uk|
     |       | |        | |        | |      | |          | |           |
     +----+--+ +--------+ +--------+ +------+ +----+-----+ +-----------+
          |                                        |
          |                                        |
       +--+--+                                  +--+--+
       |     |                                  |     |
     +-+-----+-+                                |     |
     |         |                                +-----+
     +---------+
     user: paul@surfnet.nl            surfnet.nl Authentication server
```

Figure 2: RADIUS hierarchy based on Top-Level Domain partitioning

This document also specifies various approaches for verifying that server information which was retrieved from DNS was from an authorised party; e.g. an organisation which is not at all part of a given roaming consortium may alter its own DNS records to yield a result for its own realm.

1.1. Requirements Language

In this document, several words are used to signify the requirements of the specification. The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119. [RFC2119]

1.2. Terminology

RADIUS/TLS Client: a RADIUS/TLS [RFC6614] instance which initiates a new connection.

RADIUS/TLS Server: a RADIUS/TLS [RFC6614] instance which listens on a RADIUS/TLS port and accepts new connections

RADIUS/TLS node: a RADIUS/TLS client or server

[I-D.ietf-radext-nai] defines the terms NAI, realm, consortium.

1.3. Document Status

This document is an Experimental RFC.

The communities expected to use this document are roaming consortia whose authentication services are based on the RADIUS protocol.

The duration of the experiment is undetermined; as soon as enough experience is collected on the choice points mentioned below, it is expected to be obsoleted by a standards-track version of the protocol which trims down the choice points.

If that removal of choice points obsoletes tags or service names as defined in this document and allocated by IANA, these items will be returned to IANA as per the provisions in [RFC6335].

The document provides a discovery mechanism for RADIUS which is very similar to the approach that is taken with the Diameter protocol [RFC6733]. As such, the basic approach (using Naming Authority Pointer (NAPTR) records in DNS domains which match NAI realms) is not of very experimental nature.

However, the document offers a few choice points and extensions which go beyond the provisions for Diameter. The list of major additions/ deviations is

- o provisions for determining the authority of a server to act for users of a realm (declared out of scope for Diameter)
- o much more in-depth guidance on DNS regarding timeouts, failure conditions, alteration of Time-To-Live (TTL) information than the Diameter counterpart
- o a partially correct routing error detection during DNS lookups

2. Definitions

2.1. DNS Resource Record (RR) definition

DNS definitions of RADIUS/TLS servers can be either S-NAPTR records (see [RFC3958]) or Service Record (SRV) records. When both are defined, the resolution algorithm prefers S-NAPTR results (see Section 3.4 below).

2.1.1. S-NAPTR

2.1.1.1. Registration of Application Service and Protocol Tags

This specification defines three S-NAPTR service tags:

Service Tag	Use
aaa+auth	RADIUS Authentication, i.e. traffic as defined in [RFC2865]
aaa+acct	RADIUS Accounting, i.e. traffic as defined in [RFC2866]
aaa+dynauth	RADIUS Dynamic Authorisation, i.e. traffic as defined in [RFC5176]

Figure 3: List of Service Tags

This specification defines two S-NAPTR protocol tags:

Protocol Tag	Use
radius.tls.tcp	RADIUS transported over TLS as defined in [RFC6614]
radius.dtls.udp	RADIUS transported over DTLS as defined in [RFC7360]

Figure 4: List of Protocol Tags

Note well:

The S-NAPTR service and protocols are unrelated to the IANA Service Name and Transport Protocol Number registry.

The delimiter '.' in the protocol tags is only a separator for human reading convenience - not for structure or namespacing; it MUST NOT be parsed in any way by the querying application or resolver.

The use of the separator '.' is common also in other protocols' protocol tags. This is coincidence and does not imply a shared semantics with such protocols.

2.1.1.2. Definition of Conditions for Retry/Failure

RADIUS is a time-critical protocol; RADIUS clients which do not receive an answer after a configurable, but short, amount of time, will consider the request failed. Due to this, there is little leeway for extensive retries.

As a general rule, only error conditions which generate an immediate response from the other end are eligible for a retry of a discovered target. Any error condition involving timeouts, or the absence of a reply for more than one second during the connection setup phase is to be considered a failure; the next target in the set of discovered NAPTR targets is to be tried.

Note that [RFC3958] already defines that a failure to identify the server as being authoritative for the realm is always considered a failure; so even if a discovered target returns a wrong credential instantly, it is not eligible for retry.

Furthermore, the contacted RADIUS/TLS server verifies during connection setup whether or not it finds the connecting RADIUS/TLS client authorized or not. If the connecting RADIUS/TLS client is not found acceptable, the server will close the TLS connection immediately with an appropriate alert. Such TLS handshake failures are permanently fatal and not eligible for retry, unless the connecting client has more X.509 certificates to try; in this case, a retry with the remainder of its set of certificates SHOULD be attempted. Not trying all available client certificates potentially creates a DoS for the end-user whose authentication attempt triggered the discovery; one of the neglected certificates might have led to a successful RADIUS connection and subsequent end-user authentication.

If the TLS session setup to a discovered target does not succeed, that target (as identified by IP address and port number) SHOULD be ignored from the result set of any subsequent executions of the discovery algorithm at least until the target's Effective TTL (see

Section 3.3) has expired or until the entity which executes the algorithm changes its TLS context to either send a new client certificate or expect a different server certificate.

2.1.1.3. Server Identification and Handshake

After the algorithm in this document has been executed, a RADIUS/TLS session as per [RFC6614] is established. Since the dynamic discovery algorithm does not have provisions to establish confidential keying material between the RADIUS/TLS client (i.e. the server which executes the discovery algorithm) and the RADIUS/TLS server which was discovered, TLS-PSK ciphersuites cannot be used in the subsequent TLS handshake. Only TLS ciphersuites using X.509 certificates can be used with this algorithm.

There are numerous ways to define which certificates are acceptable for use in this context. This document defines one mandatory-to-implement mechanism which allows to verify whether the contacted host is authoritative for an NAI realm or not. It also gives one example of another mechanism which is currently in wide-spread deployment, and one possible approach based on DNSSEC which is yet unimplemented.

For the approaches which use trust roots (see the following two sections), a typical deployment will use a dedicated trust store for RADIUS/TLS certificate authorities, particularly a trust store which is independent from default "browser" trust stores. Often, this will be one or few CAs, and they only issue certificates for the specific purpose of establishing RADIUS server-to-server trust. It is important not to trust a large set of CAs which operate outside the control of the roaming consortium, for their issuance of certificates with the properties important for authorisation (such as NAIRealm and policyOID below) is difficult to verify. Therefore, clients SHOULD NOT be pre-configured with a list of known public CAs by the vendor or manufacturer. Instead, the clients SHOULD start off with an empty CA list. The addition of a CA SHOULD be done only when manually configured by an administrator.

2.1.1.3.1. Mandatory-to-implement mechanism: Trust Roots + NAIRealm

Verification of authority to provide AAA services over RADIUS/TLS is a two-step process.

Step 1 is the verification of certificate wellformedness and validity as per [RFC5280] and whether it was issued from a root certificate which is deemed trustworthy by the RADIUS/TLS client.

Step 2 is to compare the value of algorithm’s variable "R" after the execution of step 3 of the discovery algorithm in Section 3.4.3 below (i.e. after a consortium name mangling, but before conversion to a form usable by the name resolution library) to all values of the contacted RADIUS/TLS server's X.509 certificate property "subjectAlternativeName:otherName:NAIRealm" as defined in Section 2.2.

2.1.1.3.2. Other mechanism: Trust Roots + policyOID

Verification of authority to provide AAA services over RADIUS/TLS is a two-step process.

Step 1 is the verification of certificate wellformedness and validity as per [RFC5280] and whether it was issued from a root certificate which is deemed trustworthy by the RADIUS/TLS client.

Step 2 is to compare the values of the contacted RADIUS/TLS server's X.509 certificate's extensions of type "Policy OID" to a list of configured acceptable Policy OIDs for the roaming consortium. If one of the configured OIDs is found in the certificate's Policy OID extensions, then the server is considered authorized; if there is no match, the server is considered unauthorized.

This mechanism is inferior to the mandatory-to-implement mechanism in the previous section because all authorized servers are validated by the same OID value; the mechanism is not fine-grained enough to express authority for one specific realm inside the consortium. If the consortium contains members which are hostile against other members, this weakness can be exploited by one RADIUS/TLS server impersonating another if DNS responses can be spoofed by the hostile member.

The shortcomings in server identification can be partially mitigated by using the RADIUS infrastructure only with authentication payloads which provide mutual authentication and credential protection (i.e. EAP types passing the criteria of [RFC4017]): using mutual authentication prevents the hostile server from mimicking the real EAP server (it can't terminate the EAP authentication unnoticed because it does not have the server certificate from the real EAP server); protection of credentials prevents the impersonating server from learning usernames and passwords of the ongoing EAP conversation (other RADIUS attributes pertaining to the authentication, such as the EAP peer's Calling-Station-ID, can still be learned though).

2.1.1.3.3. Other mechanism: DNSSEC / DANE

Where DNSSEC is used, the results of the algorithm can be trusted; i.e. the entity which executes the algorithm can be certain that the realm that triggered the discovery is actually served by the server that was discovered via DNS. However, this does not guarantee that the server is also authorized (i.e. a recognised member of the roaming consortium). The server still needs to present an X.509 certificate proving its authority to serve a particular realm.

The authorization can be sketched using DNSSEC+DANE as follows: DANE/TLSA records of all authorized servers are put into a DNSSEC zone which contains all known and authorised realms; the zone is rooted in a common, consortium-agreed branch of the DNS tree. The entity executing the algorithm uses the realm information from the authentication attempt, and then attempts to retrieve TLSA Resource Records (TLSA RR) for the DNS label "realm.commonroot". It then verifies that the presented server certificate during the RADIUS/TLS handshake matches the information in the TLSA record.

Example:

Realm = "example.com"

Common Branch = "idp.roaming-consortium.example.

label for TLSA query = "example.com.idp.roaming-consortium.example.

result of discovery algorithm for realm "example.com" = 192.0.2.1:2083

(TLS certificate of 192.0.2.1:2083 matches TLSA RR ? "PASS" : "FAIL")

2.1.1.3.4. Client Authentication and Authorisation

Note that RADIUS/TLS connections always mutually authenticate the RADIUS server and the RADIUS client. This specification provides an algorithm for a RADIUS client to contact and verify authorization of a RADIUS server only. During connection setup, the RADIUS server also needs to verify whether it considers the connecting RADIUS client authorized; this is outside the scope of this specification.

2.1.2. SRV

This specification defines two SRV prefixes (i.e. two values for the "_service._proto" part of an SRV RR as per [RFC2782]):

SRV Label	Use
_radiustls._tcp	RADIUS transported over TLS as defined in [RFC6614]
_radiusdtls._udp	RADIUS transported over DTLS as defined in [RFC7360]

Figure 5: List of SRV Labels

Just like NAPTR records, the lookup and subsequent follow-up of SRV records may yield more than one server to contact in a prioritised list. [RFC2782] does not specify rules regarding "Definition of Conditions for Retry/Failure", nor "Server Identification and Handshake". This specification defines that the rules for these two topics as defined in Section 2.1.1.2 and Section 2.1.1.3 SHALL be used both for targets retrieved via an initial NAPTR RR as well as for targets retrieved via an initial SRV RR (i.e. in the absence of NAPTR RRs).

2.1.3. Optional name mangling

It is expected that in most cases, the SRV and/or NAPTR label used for the records is the DNS A-label representation of the literal realm name for which the server is the authoritative RADIUS server (i.e. the realm name after conversion according to section 5 of [RFC5891]).

However, arbitrary other labels or service tags may be used if, for example, a roaming consortium uses realm names which are not associated to DNS names or special-purpose consortia where a globally valid discovery is not a use case. Such other labels require a consortium-wide agreement about the transformation from realm name to lookup label, and/or which service tag to use.

Examples:

a. A general-purpose RADIUS server for realm example.com might have DNS entries as follows:

```
example.com.  IN NAPTR 50 50 "s" "aaa+auth:radius.tls.tcp" ""
_radiustls._tcp.foobar.example.com.

_radiustls._tcp.foobar.example.com.  IN SRV 0 10 2083
radsec.example.com.
```

b. The consortium "foo" provides roaming services for its members only. The realms used are of the form enterprise-name.example. The consortium operates a special purpose DNS server for the (private) TLD "example" which all RADIUS servers use to resolve realm names. "Company, Inc." is part of the consortium. On the consortium’s DNS server, realm company.example might have the following DNS entries:

```
company.example.  IN NAPTR 50 50 "a"
"aaa+auth:radius.dtls.udp" "" roamserv.company.example.
```

c. The eduroam consortium (see [I-D.wierenga-ietf-eduroam] uses realms based on DNS, but provides its services to a closed community only. However, a AAA domain participating in eduroam may also want to expose AAA services to other, general-purpose, applications (on the same or other RADIUS servers). Due to that, the eduroam consortium uses the service tag "x-eduroam" for authentication purposes and eduroam RADIUS servers use this tag to look up other eduroam servers. An eduroam participant example.org which also provides general-purpose AAA on a different server uses the general "aaa+auth" tag:

```
example.org.  IN NAPTR 50 50 "s" "x-eduroam:radius.tls.tcp" ""
_radiustls._tcp.eduroam.example.org.

example.org.  IN NAPTR 50 50 "s" "aaa+auth:radius.tls.tcp" ""
_radiustls._tcp.aaa.example.org.

_radiustls._tcp.eduroam.example.org.  IN SRV 0 10 2083 aaa-
eduroam.example.org.

_radiustls._tcp.aaa.example.org.  IN SRV 0 10 2083 aaa-
default.example.org.
```

2.2. Definition of the X.509 certificate property SubjectAltName:otherName:NAIRealm

This specification retrieves IP addresses and port numbers from the Domain Name System which are subsequently used to authenticate users via the RADIUS/TLS protocol. Regardless whether the results from DNS discovery are trustworthy or not (e.g. DNSSEC in use), it is always important to verify that the server which was contacted is authorized to service requests for the user which triggered the discovery process.

The input to the algorithm is an NAI realm as specified in Section 3.4.1. As a consequence, the X.509 certificate of the server which is ultimately contacted for user authentication needs to be able to express that it is authorized to handle requests for that realm.

Current subjectAltName fields do not semantically allow to express an NAI realm; the field subjectAltName:dNSName is syntactically a good match but would inappropriately conflate DNS names and NAI realm names. Thus, this specification defines a new subjectAltName field to hold either a single NAI realm name or a wildcard name matching a set of NAI realms.

The subjectAltName:otherName:sRVName field certifies that a certificate holder is authorized to provide a service; this can be compared to the target of DNS label's SRV resource record. If the Domain Name System is insecure, it is required that the label of the SRV record itself is known-correct. In this specification, that label is not known-correct; it is potentially derived from a (potentially untrusted) NAPTR resource record of another label. If DNS is not secured with DNSSEC, the NAPTR resource record may have been altered by an attacker with access to the Domain Name System resolution, and thus the label to lookup the SRV record for may already be tainted. This makes subjectAltName:otherName:sRVName not a trusted comparison item.

Further to this, this specification's NAPTR entries may be of type "A" which do not involve resolution of any SRV records, which again makes subjectAltName:otherName:sRVName unsuited for this purpose.

This section defines the NAIRealm name as a form of otherName from the GeneralName structure in SubjectAltName defined in [RFC5280].

```
id-on-naiRealm OBJECT IDENTIFIER ::= { id-on XXX }

ub-naiRealm-length INTEGER ::= 255

NAIRealm ::= UTF8String (SIZE (1..ub-naiRealm-length))
```

The NAIRealm, if present, MUST contain an NAI realm as defined in [I-D.ietf-radext-nai]. It MAY substitute the leftmost dot-separated label of the NAI with the single character "*" to indicate a wildcard match for "all labels in this part". Further features of regular expressions, such as a number of characters followed by a * to indicate a common prefix inside the part, are not permitted.

The comparison of an NAIRealm to the NAI realm as derived from user input with this algorithm is a byte-by-byte comparison, except for the optional leftmost dot-separated part of the value whose content is a single "*" character; such labels match all strings in the same dot-separated part of the NAI realm. If at least one of the sAN:otherName:NAIRealm values matches the NAI realm, the server is considered authorized; if none matches, the server is considered unauthorized.

Since multiple names and multiple name forms may occur in the subjectAltName extension, an arbitrary number of NAIRealms can be specified in a certificate.

Examples:

NAI realm (RADIUS)	NAIRealm (cert)	MATCH?
foo.example	foo.example	YES
foo.example	*.example	YES
bar.foo.example	*.example	NO
bar.foo.example	*ar.foo.example	NO (NAIRealm invalid)
bar.foo.example	bar.*.example	NO (NAIRealm invalid)
bar.foo.example	*.*.example	NO (NAIRealm invalid)
sub.bar.foo.example	*.*.example	NO (NAIRealm invalid)
sub.bar.foo.example	*.bar.foo.example	YES

Figure 6: Examples for NAI realm vs. certificate matching

Appendix A contains the ASN.1 definition of the above objects.

3. DNS-based NAPTR/SRV Peer Discovery

3.1. Applicability

Dynamic server discovery as defined in this document is only applicable for new AAA transactions and per service (i.e. distinct discovery is needed for Authentication, Accounting, and Dynamic Authorization) where a RADIUS entity which acts as a forwarding server for one or more realms receives a request with a realm for which it is not authoritative, and which no explicit next hop is configured. It is only applicable for

a. new user sessions, i.e. for the initial Access-Request. Subsequent messages concerning this session, for example Access-Challenges and Access-Accepts use the previously-established communication channel between client and server.

b. the first accounting ticket for a user session.

c. the first RADIUS DynAuth packet for a user session.

3.2. Configuration Variables

The algorithm contains various variables for timeouts. These variables are named here and reasonable default values are provided. Implementations wishing to deviate from these defaults should make they understand the implications of changes.

DNS_TIMEOUT: maximum amount of time to wait for the complete set of all DNS queries to complete: Default = 3 seconds

MIN_EFF_TTL: minimum DNS TTL of discovered targets: Default = 60 seconds

BACKOFF_TIME: if no conclusive DNS response was retrieved after DNS_TIMEOUT, do not attempt dynamic discovery before BACKOFF_TIME has elapsed. Default = 600 seconds

3.3. Terms

Positive DNS response: a response which contains the RR that was queried for.

Negative DNS response: a response which does not contain the RR that was queried for, but contains an SOA record along with a TTL indicating cache duration for this negative result.

DNS Error: Where the algorithm states "name resolution returns with an error", this shall mean that either the DNS request timed out, or a DNS response which is neither a positive nor a negative response (e.g. SERVFAIL).

Effective TTL: The validity period for discovered RADIUS/TLS target hosts. Calculated as: Effective TTL (set of DNS TTL values) = max { MIN_EFF_TTL, min { DNS TTL values } }

SRV lookup: for the purpose of this specification, SRV lookup procedures are defined as per [RFC2782], but excluding that RFCs "A" fallback as defined in its section "Usage Rules", final "else" clause.

Greedy result evaluation: The NAPTR to SRV/A/AAAA resolution may lead to a tree of results, whose leafs are the IP addresses to contact. The branches of the tree are ordered according to their order/ preference DNS properties. An implementation is executing greedy result evaluation if it uses a depth-first search in the tree along the highest order results, attempts to connect to the corresponding resulting IP addresses, and only backtracks to other branches if the higher ordered results did not end in successful connection attempts.

3.4. Realm to RADIUS server resolution algorithm

3.4.1. Input

For RADIUS Authentication and RADIUS Accounting server discovery, input I to the algorithm is the RADIUS User-Name attribute with content of the form "user@realm"; the literal @ sign being the separator between a local user identifier within a realm and its realm. The use of multiple literal @ signs in a User-Name is strongly discouraged; but if present, the last @ sign is to be considered the separator. All previous instances of the @ sign are to be considered part of the local user identifier.

For RADIUS DynAuth Server discovery, input I to the algorithm is the domain name of the operator of a RADIUS realm as was communicated during user authentication using the Operator-Name attribute ([RFC5580], section 4.1). Only Operator-Name values with the namespace "1" are supported by this algorithm - the input to the algorithm is the actual domain name, preceeded with an "@" (but without the "1" namespace identifier byte of that attribute).

Note well: The attribute User-Name is defined to contain UTF-8 text. In practice, the content may or may not be UTF-8. Even if UTF-8, it may or may not map to a domain name in the realm part. Implementors MUST take possible conversion error paths into consideration when parsing incoming User-Name attributes. This document describes server discovery only for well-formed realms mapping to DNS domain names in UTF-8 encoding. The result of all other possible contents of User-Name is unspecified; this includes, but is not limited to:

Usage of separators other than @.

Encoding of User-Name in local encodings.

UTF-8 realms which fail the conversion rules as per [RFC5891].

UTF-8 realms which end with a . ("dot") character.

For the last bullet point, "trailing dot", special precautions should be taken to avoid problems when resolving servers with the algorithm below: they may resolve to a RADIUS server even if the peer RADIUS server only is configured to handle the realm without the trailing dot. If that RADIUS server again uses NAI discovery to determine the authoritative server, the server will forward the request to localhost, resulting in a tight endless loop.

3.4.2. Output

Output O of the algorithm is a two-tuple consisting of: O-1) a set of tuples {hostname; port; protocol; order/preference; Effective TTL} - the set can be empty; and O-2) an integer: if the set in the first part of the tuple is empty, the integer contains the Effective TTL for backoff timeout, if the set is not empty, the integer is set to 0 (and not used).

3.4.3. Algorithm

The algorithm to determine the RADIUS server to contact is as follows:

1. Determine P = (position of last "@" character) in I.

2. generate R = (substring from P+1 to end of I)

3. modify R according to agreed consortium procedures if applicable

4. convert R to a representation usable by the name resolution library if needed

5. Initialize TIMER = 0; start TIMER. If TIMER reaches DNS_TIMEOUT, continue at step 20.

6. Using the host’s name resolution library, perform a NAPTR query for R (see "Delay considerations" below). If the result is a negative DNS response, O-2 = Effective TTL (TTL value of the SOA record) and continue at step 13. If name resolution returns with error, O-1 = { empty set }, O-2 = BACKOFF_TIME and terminate.

7. Extract NAPTR records with service tag "aaa+auth", "aaa+acct", "aaa+dynauth" as appropriate. Keep note of the protocol tag and remaining TTL of each of the discovered NAPTR records.

8. If no records found, continue at step 13.

9. For the extracted NAPTRs, perform successive resolution as defined in [RFC3958], section 2.2. An implementation MAY use greedy result evaluation according to the NAPTR order/preference fields (i.e. can execute the subsequent steps of this algorithm for the highest-order entry in the set of results, and only lookup the remainder of the set if necessary).

10. If the set of hostnames is empty, O-1 = { empty set }, O-2 = BACKOFF_TIME and terminate.

11. O' = (set of {hostname; port; protocol; order/preference; Effective TTL (all DNS TTLs that led to this hostname) } for all terminal lookup results).

12. Proceed with step 18.

13. Generate R' = (prefix R with "_radiustls._tcp." and/or "_radiustls._udp.")

14. Using the host's name resolution library, perform SRV lookup with R' as label (see "Delay considerations" below).

15. If name resolution returns with error, O-1 = { empty set }, O-2 = BACKOFF_TIME and terminate.

16. If the result is a negative DNS response, O-1 = { empty set }, O-2 = min { O-2, Effective TTL (TTL value of the SOA record) } and terminate.

17. O' = (set of {hostname; port; protocol; order/preference; Effective TTL (all DNS TTLs that led to this result) } for all hostnames).

18. Generate O-1 by resolving hostnames in O' into corresponding A and/or AAAA addresses: O-1 = (set of {IP address; port; protocol; order/preference; Effective TTL (all DNS TTLs that led to this result) } for all hostnames), O-2 = 0.

19. For each element in O-1, test if the original request which triggered dynamic discovery was received on {IP address; port}. If yes, O-1 = { empty set }, O-2 = BACKOFF_TIME, log error, Terminate (see next section for a rationale). If no, O is the result of dynamic discovery. Terminate.

20. O-1 = { empty set }, O-2 = BACKOFF_TIME, log error, Terminate.

3.4.4. Validity of results

The dynamic discovery algorithm is used by servers which do not have sufficient configuration information to process an incoming request on their own. If the discovery algorithm result contains the server's own listening address (IP address and port), then there is a potential for an endless forwarding loop. If the listening address is the DNS result with the highest priorty, the server will enter a tight loop (the server would forward the request to itself, triggering dynamic discovery again in a perpetual loop). If the address has a lower priority in the set of results, there is a potential loop with intermediate hops in between (the server could forward to another host with a higher priority, which might use DNS itself and forward the packet back to the first server). The underlying reason that enables these loops is that the server executing the discovery algorithm is seriously misconfigured in that it does not recognise the request as one that is to be processed by itself. RADIUS has no built-in loop detection, so any such loops would remain undetected. So, if step 18 of the algorithm discovers such a possible-loop situation, the algorithm should be aborted and an error logged. Note that this safeguard does not provide perfect protection against routing loops. One reason which might introduce a loop include the possiblity that a subsequent hop has a statically configured next-hop which leads to an earlier host in the loop. Another reason for occuring loops is if the algorithm was executed with greedy result evaluation, and the own address was in a lower-priority branch of the result set which was not retrieved from DNS at all, and thus can’t be detected.

After executing the above algorithm, the RADIUS server establishes a connection to a home server from the result set. This connection can potentially remain open for an indefinite amount of time. This conflicts with the possibility of changing device and network configurations on the receiving end. Typically, TTL values for records in the name resolution system are used to indicate how long it is safe to rely on the results of the name resolution. If these TTLs are very low, thrashing of connections becomes possible; the Effective TTL mitigates that risk. When a connection is open and the smallest of the Effective TTL value which was learned during discovering the server has not expired, subsequent new user sessions for the realm which corresponds to that open connection SHOULD re-use the existing connection and SHOULD NOT re-execute the dynamic discovery algorithm nor open a new connection. To allow for a change of configuration, a RADIUS server SHOULD re-execute the dynamic discovery algorithm after the Effective TTL that is associated with this connection has expired. The server SHOULD keep the session open during this re-assessment to avoid closure and immediate re-opening of the connection should the result not have changed.

Should the algorithm above terminate with O-1 = empty set, the RADIUS server SHOULD NOT attempt another execution of this algorithm for the same target realm before the timeout O-2 has passed.

3.4.5. Delay considerations

The host’s name resolution library may need to contact outside entities to perform the name resolution (e.g. authoritative name servers for a domain), and since the NAI discovery algorithm is based on uncontrollable user input, the destination of the lookups is out of control of the server that performs NAI discovery. If such outside entities are misconfigured or unreachable, the algorithm above may need an unacceptably long time to terminate. Many RADIUS implementations time out after five seconds of delay between Request and Response. It is not useful to wait until the host name resolution library signals a timeout of its name resolution algorithms. The algorithm therefore controls execution time with TIMER. Execution of the NAI discovery algorithm SHOULD be non-blocking (i.e. allow other requests to be processed in parallel to the execution of the algorithm).

3.4.6. Example

Assume

a user from the Technical University of Munich, Germany, has a RADIUS User-Name of "foobar@tu-m[U+00FC]nchen.example".

The name resolution library on the RADIUS forwarding server does not have the realm tu-m[U+00FC]nchen.example in its forwarding configuration, but uses DNS for name resolution and has configured the use of Dynamic Discovery to discover RADIUS servers.

It is IPv6-enabled and prefers AAAA records over A records.

It is listening for incoming RADIUS/TLS requests on 192.0.2.1, TCP /2083.

May the configuration variables be

DNS_TIMEOUT = 3 seconds

MIN_EFF_TTL = 60 seconds

BACKOFF_TIME = 3600 seconds

If DNS contains the following records:

```
xn--tu-mnchen-t9a.example.  IN NAPTR 50 50 "s"
"aaa+auth:radius.tls.tcp" "" _myradius._tcp.xn--tu-mnchen-
t9a.example.

xn--tu-mnchen-t9a.example.  IN NAPTR 50 50 "s"
"fooservice:bar.dccp" "" _abc123._def.xn--tu-mnchen-t9a.example.

_myradius._tcp.xn--tu-mnchen-t9a.example.  IN SRV 0 10 2083
radsecserver.xn--tu-mnchen-t9a.example.
```

```
      _myradius._tcp.xn--tu-mnchen-t9a.example.  IN SRV 0 20 2083
      backupserver.xn--tu-mnchen-t9a.example.

      radsecserver.xn--tu-mnchen-t9a.example.  IN AAAA
      2001:0DB8::202:44ff:fe0a:f704

      radsecserver.xn--tu-mnchen-t9a.example.  IN A 192.0.2.3

      backupserver.xn--tu-mnchen-t9a.example.  IN A 192.0.2.7
```

Then the algorithm executes as follows, with I = "foobar@tu-m[U+00FC]nchen.example", and no consortium name mangling in use:

1. P = 7

2. R = "tu-m[U+00FC]nchen.example"

3. NOOP

4. name resolution library converts R to xn--tu-mnchen-t9a.example

5. TIMER starts.

6. Result:

```
   (TTL = 47) 50 50 "s" "aaa+auth:radius.tls.tcp" ""
   _myradius._tcp.xn--tu-mnchen-t9a.example.

   (TTL = 522) 50 50 "s" "fooservice:bar.dccp" ""
   _abc123._def.xn--tu-mnchen-t9a.example.
```

7. Result:

```
   (TTL = 47) 50 50 "s" "aaa+auth:radius.tls.tcp" ""
   _myradius._tcp.xn--tu-mnchen-t9a.example.
```

8. NOOP

9. Successive resolution performs SRV query for label _myradius._tcp.xn--tu-mnchen-t9a.example, which results in

```
   (TTL 499) 0 10 2083 radsec.xn--tu-mnchen-t9a.example.

   (TTL 2200) 0 20 2083 backup.xn--tu-mnchen-t9a.example.
```

10. NOOP

11. O’ = {

    (radsec.xn--tu-mnchen-t9a.example.; 2083; RADIUS/TLS; 10; 60),

    (backup.xn--tu-mnchen-t9a.example.; 2083; RADIUS/TLS; 20; 60)

    } // minimum TTL is 47, up’ed to MIN_EFF_TTL

12. Continuing at 18.

13. (not executed)

14. (not executed)

15. (not executed)

16. (not executed)

17. (not executed)

18. O-1 = {

    (2001:0DB8::202:44ff:fe0a:f704; 2083; RADIUS/TLS; 10; 60),

    (192.0.2.7; 2083; RADIUS/TLS; 20; 60)

    }; O-2 = 0

19. No match with own listening address; terminate with tuple (O-1, O-2) from previous step.

The implementation will then attempt to connect to two servers, with preference to [2001:0DB8::202:44ff:fe0a:f704]:2083 using the RADIUS/TLS protocol.

4. Operations and Manageability Considerations

The discovery algorithm as defined in this document contains several options; the major ones being use of NAPTR vs. SRV; how to determine the authorization status of a contacted server for a given realm; which trust anchors to consider trustworthy for the RADIUS conversation setup.

Random parties which do not agree on the same set of options may not be able to interoperate. However, such a global interoperability is not intended by this document.

Discovery as per this document becomes important inside a roaming consortium, which has set up roaming agreements with the other partners. Such roaming agreements require much more than a technical means of server discovery; there are administrative and contractual considerations at play (service contracts, backoffice compensations, procedures, ...).

A roaming consortium’s roaming agreement must include a profile of which choice points of this document to use. So long as the roaming consortium can settle on one deployment profile, they will be able to interoperate based on that choice; this per-consortium interoperability is the intended scope of this document.

5. Security Considerations

When using DNS without DNSSEC security extensions and validation for all of the replies to NAPTR, SRV and A/AAAA requests as described in section Section 3, the result of the discovery process can not be trusted. Even if it can be trusted (i.e. DNSSEC is in use), actual authorization of the discovered server to provide service for the given realm needs to be verified. A mechanism from section Section 2.1.1.3 or equivalent MUST be used to verify authorization.

The algorithm has a configurable completion timeout DNS_TIMEOUT defaulting to three seconds for RADIUS’ operational reasons. The lookup of DNS resource records based on unverified user input is an attack vector for DoS attacks: an attacker might intentionally craft bogus DNS zones which take a very long time to reply (e.g. due to a particularly byzantine tree structure, or artificial delays in responses).

To mitigate this DoS vector, implementations SHOULD consider rate-limiting either their amount of new executions of the dynamic discovery algorithm as a whole, or the amount of intermediate responses to track, or at least the number of pending DNS queries. Implementations MAY choose lower values than the default for DNS_TIMEOUT to limit the impact of DoS attacks via that vector. They MAY also continue their attempt to resolve DNS records even after DNS_TIMEOUT has passed; a subsequent request for the same realm might benefit from retrieving the results anyway. The amount of time to spent waiting for a result will influence the impact of a possible DoS attack; the waiting time value is implementation dependent and outside the scope of this specification.

With Dynamic Discovery being enabled for a RADIUS Server, and depending on the deployment scenario, the server may need to open up its target IP address and port for the entire internet, because arbitrary clients may discover it as a target for their authentication requests. If such clients are not part of the roaming consortium, the RADIUS/TLS connection setup phase will fail (which is intended) but the computational cost for the connection attempt is significant. With the port for a TLS-based service open, the RADIUS server shares all the typical attack vectors for services based on TLS (such as HTTPS, SMTPS, ...). Deployments of RADIUS/TLS with Dynamic Discovery should consider these attack vectors and take appropriate counter-measures (e.g. blacklisting known-bad IPs on a firewall, rate-limiting new connection attempts, etc.).

6. Privacy Considerations

The classic RADIUS operational model (known, pre-configured peers, shared secret security, mostly plaintext communication) and this new RADIUS dynamic discovery model (peer discovery with DNS, PKI security and packet confidentiality) differ significantly in their impact on the privacy of end users trying to authenticate to a RADIUS server.

With classic RADIUS, traffic in large environments gets aggregated by statically configured clearinghouses. The packets sent to those clearinghouses and their responses are mostly unprotected. As a consequence,

- o All intermediate IP hops can inspect most of the packet payload in clear text, including the User-Name and Calling-Station-Id attributes, and can observe which client sent the packet to which clearinghouse. This allows the creation of mobility profiles for any passive observer on the IP path.

- o The existence of a central clearinghouse creates an opportunity for the clearinghouse to trivially create the same mobility profiles. The clearinghouse may or may not be trusted not to do this, e.g. by sufficiently threatening contractual obligations.

- o In addition to that, with the clearinghouse being a RADIUS intermediate in possession of a valid shared secret, the clearinghouse can observe and record even the security-critical RADIUS attributes such as User-Password. This risk may be mitigated by choosing authentication payloads which are cryptographically secured and do not use the attribute User-Password - such as certain EAP types.

- o There is no additional information disclosure to parties outside the IP path between the RADIUS client and server (in particular, no DNS servers learn about realms of current ongoing authentications).

With RADIUS and dynamic discovery,

- This protocol allows for RADIUS clients to identify and directly connect to the RADIUS home server. This can eliminate the use of clearinghouses to do forwarding of requests, and it also eliminates the ability of the clearinghouse to then aggregate the user information that flows through it. However, there exist reasons why clearinghouses might still be used. One reason to keep a clearinghouse is to act as a gateway for multiple backends in a company; another reason may be a requirement to sanitise RADIUS datagrams (filter attributes, tag requests with new attributes, ...).

- Even where intermediate proxies continue to be used for reasons unrelated to dynamic discovery, the number of such intermediates may be reduced by removing those proxies which are only deployed for pure request routing reasons. This reduces the number of entities which can inspect the RADIUS traffic.

- RADIUS clients which make use of dynamic discovery will need to query the Domain Name System, and use a user’s realm name as the query label. A passive observer on the IP path between the RADIUS client and the DNS server(s) being queried can learn that a user of that specific realm was trying to authenticate at that RADIUS client at a certain point in time. This may or may not be sufficient for the passive observer to create a mobility profile. During the recursive DNS resolution, a fair number of DNS servers and the IP hops in between those get to learn that information. Not every single authentication triggers DNS lookups, so there is no one-to-one relation of leaked realm information and the number of authentications for that realm.

- Since dynamic discovery operates on a RADIUS hop-by-hop basis, there is no guarantee that the RADIUS payload is not transmitted between RADIUS systems which do not make use of this algorithm, and possibly using other transports such as RADIUS/UDP. On such hops, the enhanced privacy is jeopardized.

In summary, with classic RADIUS, few intermediate entities learn very detailed data about every ongoing authentications, while with dynamic discovery, many entities learn only very little about recently authenticated realms.

7. IANA Considerations

This document requests IANA registration of the following entries in existing registries:

- S-NAPTR Application Service Tags registry

   * aaa+auth

   * aaa+acct

   * aaa+dynauth

- S-NAPTR Application Protocol Tags registry

   * radius.tls.tcp

   * radius.dtls.udp

This document reserves the use of the "radiustls" and "radiusdtls" service names. Registration information as per [RFC6335] section 8.1.1 is as follows:

Service Name: radiustls; radiusdtls

Transport Protocols: TCP (for radiustls), UDP (for radiusdtls)

Assignee: IESG <iesg@ietf.org>

Contact: IETF Chair <chair@ietf.org>

Description: Authentication, Accounting and Dynamic authorization via the RADIUS protocol. These service names are used to construct the SRV service labels "_radiustls" and "_radiusdtls" for discovery of RADIUS/TLS and RADIUS/DTLS servers, respectively.

Reference: RFC Editor Note: please insert the RFC number of this document. The protocol does not use broadcast, multicast or anycast communication.

This specification makes use of the SRV Protocol identifiers "_tcp" and "_udp" which are mentioned as early as [RFC2782] but do not appear to be assigned in an actual registry. Since they are in wide-spread use in other protocols, this specification refrains from requesting a new registry "RADIUS/TLS SRV Protocol Registry" and continues to make use of these tags implicitly.

This document requires that a number of Object Identifiers be assigned. They are now under the control of IANA following [RFC7299]

IANA is requested to assign the following identifiers:

TBD99 is to be assigned from the "SMI Security for PKIX Module Identifier Registry". The suggested description is id-mod-nai-realm-08.

TBD98 is to be assigned from the "SMI Security for PKIX Other Name Forms Registry." The suggested description is id-on-naiRealm.

RFC Editor Note: please replace the occurences of TBD98 and TBD99 in Appendix A of the document with the actually assigned numbers.

8. References

8.1. Normative References

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

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC2866] Rigney, C., "RADIUS Accounting", RFC 2866, June 2000.

[RFC3958] Daigle, L. and A. Newton, "Domain-Based Application Service Location Using SRV RRs and the Dynamic Delegation Discovery Service (DDDS)", RFC 3958, January 2005.

[RFC5176] Chiba, M., Dommety, G., Eklund, M., Mitton, D., and B. Aboba, "Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS)", RFC 5176, January 2008.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[RFC5580] Tschofenig, H., Adrangi, F., Jones, M., Lior, A., and B. Aboba, "Carrying Location Objects in RADIUS and Diameter", RFC 5580, August 2009.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, August 2010.

[RFC6614] Winter, S., McCauley, M., Venaas, S., and K. Wierenga, "Transport Layer Security (TLS) Encryption for RADIUS", RFC 6614, May 2012.

[RFC7360] DeKok, A., "Datagram Transport Layer Security (DTLS) as a Transport Layer for RADIUS", RFC 7360, September 2014.

[I-D.ietf-radext-nai] DeKok, A., "The Network Access Identifier", draft-ietf-radext-nai-15 (work in progress), December 2014.

8.2. Informative References

[RFC4017] Stanley, D., Walker, J., and B. Aboba, "Extensible Authentication Protocol (EAP) Method Requirements for Wireless LANs", RFC 4017, March 2005.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, August 2011.

[RFC6733] Fajardo, V., Arkko, J., Loughney, J., and G. Zorn, "Diameter Base Protocol", RFC 6733, October 2012.

[RFC7299] Housley, R., "Object Identifier Registry for the PKIX Working Group", RFC 7299, July 2014.

[I-D.wierenga-ietf-eduroam] Wierenga, K., Winter, S., and T. Wolniewicz, "The eduroam architecture for network roaming", draft-wierenga-ietf-eduroam-05 (work in progress), March 2015.

Appendix A. Appendix A: ASN.1 Syntax of NAIRealm

```
PKIXNaiRealm08 {iso(1) identified-organization(3) dod(6)
     internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
     id-mod-nai-realm-08 (TBD99) }

 DEFINITIONS EXPLICIT TAGS ::=

 BEGIN

 -- EXPORTS ALL --

 IMPORTS

    id-pkix
    FROM PKIX1Explicit-2009
        {iso(1) identified-organization(3) dod(6) internet(1)
         security(5) mechanisms(5) pkix(7) id-mod(0)
         id-mod-pkix1-explicit-02(51)}
           -- from RFC 5280, RFC 5912

    OTHER-NAME
    FROM PKIX1Implicit-2009
       {iso(1) identified-organization(3) dod(6) internet(1) security(5)
       mechanisms(5) pkix(7) id-mod(0) id-mod-pkix1-implicit-02(59)}
             -- from RFC 5280, RFC 5912
 ;


 -- Service Name Object Identifier

 id-on   OBJECT IDENTIFIER ::= { id-pkix 8 }

 id-on-naiRealm OBJECT IDENTIFIER ::= { id-on TBD98 }

 -- Service Name

 naiRealm OTHER-NAME ::= { NAIRealm IDENTIFIED BY { id-on-naiRealm }}

 ub-naiRealm-length INTEGER ::= 255

 NAIRealm ::= UTF8String (SIZE (1..ub-naiRealm-length))

 END
```

Authors' Addresses

Stefan Winter
Fondation RESTENA
6, rue Richard Coudenhove-Kalergi
Luxembourg 1359
LUXEMBOURG

Phone: +352 424409 1
Fax: +352 422473
EMail: stefan.winter@restena.lu
URI: http://www.restena.lu.

Mike McCauley
AirSpayce Pty Ltd
9 Bulbul Place
Currumbin Waters QLD 4223
AUSTRALIA

Phone: +61 7 5598 7474
EMail: mikem@airspayce.com
URI: http://www.airspayce.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

D. Cheng
Huawei
J. Korhonen
Broadcom Corporation
M. Boucadair
France Telecom
S. Sivakumar
Cisco Systems

October 19, 2015

RADIUS Extensions for IP Port Configuration and Reporting
draft-ietf-radext-ip-port-radius-ext-06

Abstract

This document defines three new RADIUS attributes. For devices that implementing IP port ranges, these attributes are used to communicate with a RADIUS server in order to configure and report TCP/UDP ports and ICMP identifiers, as well as mapping behavior for specific hosts. This mechanism can be used in various deployment scenarios such as CGN (Carrier Grade NAT), NAT64, Provider WLAN Gateway, etc.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

In a broadband network, customer information is usually stored on a RADIUS server [RFC2865] and at the time when a user initiates an IP connection request, the RADIUS server will populate the user's configuration information to the Network Access Server (NAS), which is usually co-located with the Border Network Gateway (BNG), after the connection request is granted. The Carrier Grade NAT (CGN) function may also be implemented on the BNG, and therefore the CGN TCP/UDP port (or ICMP identifier) mapping(s) behavior(s) can be configured on the RADIUS server as part of the user profile, and populated to the NAS in the same manner. In addition, during the operation, the CGN can also convey port/identifier mapping behavior specific to a user to the RADIUS server, as part of the normal RADIUS accounting process.

The CGN device that communicates with a RADIUS server using RADIUS extensions defined in this document may perform NAT44 [RFC3022], NAT64 [RFC6146], or Dual-Stack Lite AFTR [RFC6333] function.

For the CGN case, when IP packets traverse a CGN device, it would perform TCP/UDP source port mapping or ICMP identifier mapping as required. A TCP/ UDP source port or ICMP identifier, along with source IP address, destination IP address, destination port and protocol identifier if applicable, uniquely identify a session. Since the number space of TCP/UDP ports and ICMP identifiers in CGN's external realm is shared among multiple users assigned with the same IPv4 address, the total number of a user's simultaneous IP sessions is likely to be subject to port quota (see Section 5 of [RFC6269]).

The attributes defined in this document may also be used to report the assigned port range in some deployments such as Provider WLAN [I-D.gundavelli-v6ops-community-wifi-svcs]. For example, a visiting host can be managed by a CPE (Customer Premises Equipment) which will need to report the assigned port range to the service platform. This is required for identification purposes (see TR-146 [TR-146] for example).

This document proposes three new attributes as RADIUS protocol's extensions, and they are used for separate purposes as follows:

1. IP-Port-Limit: This attribute may be carried in RADIUS Acces-Accept, Access-Request, Accounting-Request or CoA-Request packet. The purpose of this attribute is to limit the total number of TCP/UDP ports and/or ICMP identifiers that an IP subscriber can use, associated with one or more IPv4 addresses.

2. IP-Port-Range: This attribute may be carried in RADIUS Accounting-Request packet. The purpose of this attribute is to report by an address sharing device (e.g., a CGN) to the RADIUS server the range of TCP/UDP ports and/or ICMP identifiers that have been allocated or deallocated associated with a given IPv4 address for a subscriber.

3. IP-Port-Forwarding-Map: This attribute may be carried in RADIUS Access-Accept, Access-Request, Accounting-Request or CoA-Request packet. The purpose of this attribute is to specify how a TCP/UDP port (or an ICMP identifier) mapping to another TCP/UDP port (or an ICMP identifier), and each is associated with its respective IPv4 address.

This document leverages the protocol defined in [RFC7012] by proposing a mapping between type field of RADIUS TLV and Element ID of IPFIX. It also proposes a few new IPFIX Elements as required by this document (see Section 3).

This document was constructed using the [RFC2629].

2. Terminology

This document makes use if the following terms:

- IP Port: refers to the port numbers of IP transport protocols, including TCP port, UDP port and ICMP identifier.

- IP Port Type: refers to one of the following: (1) TCP/UDP port and ICMP identifier, (2) TCP port and UDP port, (3) TCP port, (4) UDP port, or (5) ICMP identifier.

- IP Port Limit: denotes the maximum number of IP ports for a specific IP port type, that a device supporting port ranges can use when performing port number mapping for a specific user. Note, this limit is usually associated with one or more IPv4 addresses.

- IP Port Range: specifies a set of contiguous IP ports, indicated by the smallest numerical number and the largest numerical number, inclusively.

- Internal IP Address: refers to the IP address that is used as a source IP address in an outbound IP packet sent towards a device supporting port ranges in the internal realm. In the IPv4 case, it is typically a private address [RFC1918].

- External IP Address: refers to the IP address that is used as a source IP address in an outbound IP packet after traversing a device supporting port ranges in the external realm. In the IPv4 case, it is typically a global routable IP address.

- Internal Port: is a UDP or TCP port, or an ICMP identifier, which is allocated by a host or application behind a device supporting port ranges for an outbound IP packet in the internal realm.

- External Port: is a UDP or TCP port, or an ICMP identifier, which is allocated by a device supporting port ranges upon receiving an outbound IP packet in the internal realm, and is used to replace the internal port that is allocated by a user or application.

- External realm: refers to the networking segment where IPv4 public addresses are used in respective of the device supporting port ranges.

- Internal realm: refers to the networking segment that is behind a device supporting port ranges and where IPv4 private addresses are used.

- Mapping: associates with a device supporting port ranges for a relationship between an internal IP address, internal port and the protocol, and an external IP address, external port, and the protocol.

- Port-based device: a device that is capable of providing IP address and IP port mapping services and in particular, with the granularity of one or more subsets within the 16-bit IP port number range. A typical example of this device is a CGN, CPE, Provider WLAN Gateway, etc.

Note the terms "internal IP address", "internal port", "internal realm", "external IP address", "external port", "external realm", and "mapping" and their semantics are the same as in [RFC6887], and [RFC6888].

3. Extensions of RADIUS Attributes and TLVs

These three new attributes are defined in the following sub-sections:

1. IP-Port-Limit Attribute

2. IP-Port-Range Attribute

3. IP-Port-Forwarding-Map Attribute

All these attributes are allocated from the RADIUS "Extended Type" code space per [RFC6929].

3.1. Extended Attributes for IP Ports

3.1.1. IP-Port-Limit Attribute

This attribute is RADIUS Extended-Type, and contains a set of embedded TLVs defined in Section 3.2.1 (IP-Port-Type TLV), Section 3.2.2 (IP-Port-Limit TLV), and Section 3.2.3 (IP-Port-Ext-IPv4-Addr TLV). It specifies the maximum number of IP ports as indicated in IP-Port-Limit TLV, of a specific port type as indicated in IP-Port-Type TLV, and associated with a given IPv4 address as indicated in IP-Port-Ext-IPv4-Addr TLV for an end user.

Note that when IP-Port-Ext-IPv4-Addr TLV is not included as part of the IP-Port-Limit Attribute, the port limit is applied to all the IPv4 addresses managed by the port device, e.g., a CGN or NAT64 device.

The IP-Port-Limit Attribute MAY appear in an Access-Accept packet. It MAY also appear in an Access-Request packet as a hint by the device supporting port ranges, which is co-allocated with the NAS, to the RADIUS server as a preference, although the server is not required to honor such a hint.

The IP-Port-Limit Attribute MAY appear in a CoA-Request packet.

The IP-Port-Limit Attribute MAY appear in an Accounting-Request packet.

The IP-Port-Limit Attribute MUST NOT appear in any other RADIUS packets.

The format of the IP-Port-Limit Attribute is shown in Figure 1. The fields are transmitted from left to right.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     | Extended-Type |    Value ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1

Type:

TBA1.

Length:

This field indicates the total length in bytes of all fields of this attribute, including the Type, Length, Extended-Type, and the entire length of the embedded TLVs.

Extended-Type:

TBA2.

Value:

This field contains a set of TLVs as follows:

IP-Port-Type TLV:

This TLV contains a value that indicates the IP port type. Refer to Section 3.2.1.

IP-Port-Limit TLV:

This TLV contains the maximum number of IP ports of a specific IP port type and associated with a given IPv4 address for an end user. This TLV must be included in the IP-Port-Limit Attribute. Refer to Section 3.2.2.

IP-Port-Ext-IPv4-Addr TLV:

This TLV contains the IPv4 address that is associated with the IP port limit contained in the IP-Port-Limit TLV. This TLV is optionally included as part of the IP-Port-Limit Attribute. Refer to Section 3.2.3.

IP-Port-Limit attribute is associated with the following identifier: Type(TBA1).Extended-Type(TBA2).[IP-Port-Limit TLV (TBA6),IP-Port-Type TLV(TBA5), {IP-Port-Ext-IPv4-Addr TLV(TBA7)}].

3.1.2. IP-Port-Range Attribute

This attribute is RADIUS Extended-Type, and contains a set of embedded TLVs defined in Section 3.2.1(IP-Port-Type TLV), Section 3.2.9(IP-Port-Range-Start TLV), Section 3.2.10 (IP-Port-Range-End TLV), Section 3.2.8 (IP-Port-Alloc TLV), Section 3.2.3 (IP-Port-Ext-IPv4-Addr TLV), and Section 3.2.11 (IP-Port-Local-Id TLV).

This attribute contains a range of contiguous IP ports of a specific port type and associated with an IPv4 address that are either allocated or deallocated by a device for a given subscriber, and the information is intended to send to RADIUS server.

This attribute can be used to convey a single IP port number; in such case IP-Port-Range-Start and IP-Port-Range-End conveys the same value.

Within an IP-Port-Range Attribute, the IP-Port-Alloc TLV is always included. For port allocation, both IP-Port-Range-Start TLV and IP-Port-Range-End TLV must be included; for port deallocation, the inclusion of these two TLVs is optional and if not included, it implies that all ports that are previously allocated are now deallocated. Both IP-Port-Ext-IPv4-Addr TLV and IP-Port-Local-Id TLV are optional and if included, they are used by a port device (e.g., a CGN device) to identify the end user.

The IP-Port-Range Attribute MAY appear in an Accounting-Request packet.

The IP-Port-Range Attribute MUST NOT appear in any other RADIUS packets.

The format of the IP-Port-Range Attribute format is shown in Figure 2. The fields are transmitted from left to right.

```
0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     | Extended-Type |    Value ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2

Type:

TBA1.

Length:

This field indicates the total length in bytes of all fields of this attribute, including the Type, Length, Extended-Type, and the entire length of the embedded TLVs.

Extended-Type:

TBA3.

Value:

This field contains a set of TLVs as follows:

IP-Port-Type TLV:

This TLV contains a value that indicates the IP port type. Refer to Section 3.2.1.

IP-Port-Alloc TLV:

This TLV contains a flag to indicate that the range of the specified IP ports for either allocation or deallocation. This TLV must be included as part of the IP-Port-Range Attribute. Refer to Section 3.2.8.

IP-Port-Range-Start TLV:

This TLV contains the smallest port number of a range of contiguous IP ports. To report the port allocation, this TLV must be included together with IP-Port-Range-End TLV as part of the IP-Port-Range Attribute. Refer to Section 3.2.9.

IP-Port-Range-End TLV:

This TLV contains the largest port number of a range of contiguous IP ports. To report the port allocation, this TLV must be included together with IP-Port-Range-Start TLV as part of the IP-Port-Range Attribute. Refer to Section 3.2.10.

IP-Port-Ext-IPv4-Addr TLV:

This TLV contains the IPv4 address that is associated with the IP port range, as collectively indicated in the IP-Port-Range-Start TLV and the IP-Port-Range-End TLV. This TLV is optionally included as part of the IP-Port-Range Attribute. Refer to Section 3.2.3.

IP-Port-Local-Id TLV:

This TLV contains a local session identifier at the customer premise, such as MAC address, interface ID, VLAN ID, PPP sessions ID, VRF ID, IPv6 address/prefix, etc. This TLV is optionally included as part of the IP-Port-Range Attribute. Refer to Section 3.2.11.

The IP-Port-Range attribute is associated with the following identifier: Type(TBA1).Extended-Type(TBA3).[IP-Port-Alloc TLV (TBA12), IP-Port-Type TLV(TBA5), {IP-Port-Range-Start TLV(TBA13), IP-Port-Range-End TLV(TBA14)}, {IP-Port-Ext-IPv4-Addr TLV (TBA7)}, {IP-Port-Local-Id TLV (TBA15)}].

3.1.3. IP-Port-Forwarding-Map Attribute

This attribute is RADIUS Extended-Type, and contains a set of embedded TLVs defined in Section 3.2.1(IP-Port-Type TLV), Section 3.2.6(IP-Port-Int-Port TLV), Section 3.2.7(IP-Port-Ext-Port TLV), Section 3.2.4(IP-Port-Int-IPv4-Addr TLV) or Section 3.2.5(IP-Port-Int-IPv6-Addr TLV), Section 3.2.11(IP-Port-Local-Id TLV) and Section 3.2.3 (IP-Port-Ext-IP-Addr TLV).

The attribute contains a 2-byte IP internal port number that is associated with an internal IPv4 or IPv6 address, or a locally significant identifier at the customer site, and a 2-byte IP external port number that is associated with an external IPv4 address. The internal IPv4 or IPv6 address, or the local identifier must be included; the external IPv4 address may also be included.

The IP-Port-Forwarding-Map Attribute MAY appear in an Access-Accept packet. It MAY also appear in an Access-Request packet as a hint by the device supporting port mapping, which is co-allocated with the NAS, to the RADIUS server as a preference, although the server is not required to honor such a hint.

The IP-Port-Forwarding-Map Attribute MAY appear in a CoA-Request packet.

The IP-Port-Forwarding-Map Attribute MAY also appear in an Accounting-Request packet.

The attribute MUST NOT appear in any other RADIUS packet.

The format of the IP-Port-Forwarding-Map Attribute is shown in Figure 3. The fields are transmitted from left to right.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     | Extended-Type |    Value ....
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3

Type:

TBA1.

Length:

This field indicates the total length in bytes of all fields of this attribute, including the Type, Length, Extended-Type, and the entire length of the embedded TLVs.

Extended-Type:

TBA4.

Value:

This field contains a set of TLVs as follows:

IP-Port-Type TLV:

This TLV contains a value that indicates the IP port type. Refer to Section 3.2.1.

IP-Port-Int-Port TLV:

This TLV contains an internal IP port number associated with an internal IPv4 or IPv6 address. This TLV must be included together with IP-Port-Ext-Port TLV as part of the IP-Port-Forwarding-Map attribute. Refer to Section 3.2.6.

IP-Port-Ext-Port TLV:

This TLV contains an external IP port number associated with an external IPv4 address. This TLV must be included together with IP-Port-Int-Port TLV as part of the IP-Port-Forwarding-Map attribute. Refer to Section 3.2.7.

IP-Port-Int-IPv4-Addr TLV:

This TLV contains an IPv4 address that is associated with the internal IP port number contained in the IP-Port-Int-Port TLV. For IPv4 network, either this TLV or IP-Port-Local-Id TLV must be included as part of the IP-Port-Forwarding-Map Attribute. Refer to Section 3.2.4.

IP-Port-Int-IPv6-Addr TLV:

This TLV contains an IPv4 address that is associated with the internal IP port number contained in the IP-Port-Int-Port TLV. For IPv6 network, either this TLV or IP-Port-Local-Id TLV must be included as part of the IP-Port-Forwarding-Map Attribute. Refer to Section 3.2.5.

IP-Port-Local-Id TLV:

This TLV contains a local session identifier at the customer premise, such as MAC address, interface ID, VLAN ID, PPP sessions ID, VRF ID, IPv6 address/prefix, etc. Either this TLV or IP-Port-Int-IP-Addr TLV must be included as part of the IP-Port-Forwarding-Map Attribute. Refer to Section 3.2.11.

IP-Port-Ext-IPv4-Addr TLV:

This TLV contains an IPv4 address that is associated with the external IP port number contained in the IP-Port-Ext-Port TLV. This TLV may be included as part of the IP-Port-Forwarding-Map Attribute. Refer to Section 3.2.3.

The IP-Port-Forwarding-Map attribute is associated with the following identifier: Type(TBA1).Extended-Type(TBA4). [IP-Port-Int-Port TLV(TBA10), IP-Port-Ext-Port TLV(TBA11), IP-Port-Type TLV(TBA5), {IP-Port-Int-IPv4-Addr TLV(TBA8) | IP-Port-Int-IPv6-Addr TLV(TBA9)}, {IP-Port-Ext-IPv4-Addr TLV(TBA7)}].

3.2. RADIUS TLVs for IP Ports

3.2.1. IP-Port-Type TLV

This TLV (Figure 4) uses the format defined in [RFC6929]. Its Type field contains a value that uniquely refers to IPFIX Element transportType (TBAx1), and its Value field contains IPFIX Element transportType, which indicates the type of IP transport type as follows:

1:

Refer to TCP port, UDP port, and ICMP identifier as a whole.

2:

Refer to TCP port and UDP port as a whole.

3:

Refer to TCP port only.

4:

Refer to UDP port only.

5:

Refer to ICMP identifier only.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     |         transportType
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
            transportType       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4

Type:

TBA5: This uniquely refers to IPFIX Element ID TBA0.

Length:

6.

transportType:

Integer. This field contains the data (unsigned8) of transportType (TBX1) defined in IPFIX, right justified, and the unused bits in this field must be set to zero.

3.2.2. IP-Port-Limit TLV

This TLV (Figure 5) uses the format defined in [RFC6929]. Its Type field contains a value that uniquely refers to IPFIX Element natTransportLimit (TBAx2), and its Value field contains IPFIX Element natTransportLimit, which indicates the maximum number of ports of a specified IP-Port-Type and associated with a given IPv4 address assigned to a subscriber.

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |      Type     |     Length    |       natTransportLimit
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
             natTransportLimit      |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5

Type:

TBA6: This uniquely refers to IPFIX Element ID Limit TBD.

Length:

6.

natTransportLimit:

Integer. This field contains the data (unsigned16) of natTransportLimit (TBX2) defined in IPFIX, right justified, and the unused bits in this field must be set to zero.

3.2.3. IP-Port-Ext-IPv4-Addr TLV

This TLV (Figure 6) uses the format defined in[RFC6929]. Its Type field contains a value that uniquely refers to IPFIX Element postNATSourceIPv4Address(225), and its Value field contains IPFIX Element postNATSourceIPv4Address, which is the IPv4 source address after NAT operation (refer to [IPFIX]).

IP-Port-Ext-IPv4-Addr TLV can be included as part of the IP-Port-Limit Attribute (refer to Section 3.1.1), IP-Port-Range Attribute (refer to Section 3.1.2), and IP-Port-Forwarding-Map Attribute (refer to Section 3.1.3).

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |      Type     |    Length     |    postNATSourceIPv4Address
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         postNATSourceIPv4Address   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6

Type:

TBA7: The type field uniquely refers to the IPFIX Element ID 225.

Length:

6

postNATSourceIPv4Address:

Integer. This field contains the data (ipv4Address) of postNATSourceIPv4Address (225) defined in IPFIX.

3.2.4. IP-Port-Int-IPv4-Addr TLV

This TLV (Figure 7) uses format defined in [RFC6929]. Its Type field contains a value that uniquely refers to IPFIX Element sourceIPv4Address (8), and its Value field contains IPFIX Element sourceIPv4Address, which is the IPv4 source address before NAT operation (refer to [IPFIX]).

IP-Port-Int-IPv4-Addr TLV can be included as part of the IP-Port-Forwarding-Map Attribute (refer to Section 3.1.3).

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Length    |       sourceIPv4Address
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      sourceIPv4Address         |
+-+--+-+-+-+-+-+-+-++-+-+-+-+-+-+-+
```

Figure 7

Type:

TBA8: The type field uniquely refers to the IPFIX Element ID 8.

Length:

6.

sourceIPv4Address:

Integer. This field contains the data (ipv4Address) of sourceIPv4Address (8) defined in IPFIX.

3.2.5. IP-Port-Int-IPv6-Addr TLV

This TLV (Figure 8) uses format defined in [RFC6929]. Its Type field contains a value that uniquely refers to IPFIX Element sourceIPv6Address(27), and its Value field contains IPFIX Element sourceIPv6Address, which is the IPv6 source address before NAT operation (refer to [IPFIX]).

IP-Port-Int-IPv6-Addr TLV can be included as part of the IP-Port-Forwarding-Map Attribute (refer to Section 3.1.3).

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Length    |        sourceIPv6Address
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                          sourceIPv6Address
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                          sourceIPv6Address
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                          sourceIPv6Address
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        sourceIPv6Address       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8

Type:

TBA9: The type field uniquely refers to the IPFIX Element ID 27.

Length:

18.

sourceIPv6Address:

IPv6 address (128 bits). This field contains the data (ipv6Address) of sourceIPv6Address (27) defined in IPFIX.

3.2.6. IP-Port-Int-Port TLV

This TLV (Figure 9) uses format defined in [RFC6929]. Its Type field contains a value that uniquely refers to IPFIX Element sourceTransportPort (7), and its Value field contains IPFIX Element sourceTransportPort, which is the source transport number associated with an internal IPv4 or IPv6 address (refer to [IPFIX]).

IP-Port-Int-Port TLV is included as part of the IP-Port-Forwarding-Map Attribute (refer to Section 3.1.3).

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Type   |     Length    |      sourceTransportPort
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        sourceTransportPort     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9

Type:

TBA10: This uniquely refers to the IPFIX Element ID 7.

Length:

4.

sourceTransportPort:

Integer. This field contains the data (unsigned16) of sourceTrasnportPort (7) defined in IPFIX, right justified, and unused bits must be set to zero.

3.2.7. IP-Port-Ext-Port TLV

This TLV (Figure 10) uses format defined in [RFC6929]. Its Type field contains a value that uniquely refers to IPFIX Element postNAPTSourceTransportPort (227), and its Value field contains IPFIX Element postNAPTSourceTransportPort, which is the transport number associated with an external IPv4 address(refer to [IPFIX]).

IP-Port-Ext-Port TLV is included as part of the IP-Port-Forwarding-Map Attribute (refer to Section 3.1.3).

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      Type     |     Length    |  postNAPTSourceTransportPort
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      postNAPTSourceTransportPort  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 10

Type:

TBA11: This uniquely refers to the IPFIX Element ID 227 .

Length:

6.

postNAPTSourceTransportPort:

Integer. This field contains the data (unsigned16) of postNAPTSourceTrasnportPort (227) defined in IPFIX, right justified, and unused bits must be set to zero.

3.2.8. IP-Port-Alloc TLV

This TLV (Figure 11) uses format defined in [RFC6929]. Its Type field contains a value that uniquely refers to IPFIX Element natEvent (230), and its Value field contains IPFIX Element "natEvent", which is a flag to indicate an action of NAT operation (refer to [IPFIX]).

When the value of natEvent is "1" (Create event), it means to allocate a range of transport ports; when the value is "2", it means to de-allocate a range of transports ports. For the purpose of this TLV, no other value is used.

IP-Port-Alloc TLV is included as part of the IP-Port-Range Attribute (refer to Section 3.1.2).

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Type      |    Length     |            natEvent
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
               natEvent            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11

Type:

TBA12: This uniquely refers to the IPFIX Element ID 230 .

Length:

3.

natEvent:

Integer. This field contains the data (unsigned8) of natEvent (230) defined in IPFIX, right justified, and unused bits must be set to zero. It indicates the allocation or deallocation of a range of IP ports as follows:

1:

Allocation

2:

Deallocation

Reserved:

0.

3.2.9. IP-Port-Range-Start TLV

This TLV (Figure 12) uses format defined in [RFC6929]. Its Type field contains a value that uniquely refers to IPFIX Element portRangeStart (361), and its Value field contains IPFIX Element portRangeStart, which is the smallest port number of a range of contiguous transport ports (refer to [IPFIX]).

IP-Port-Range-Start TLV is included as part of the IP-Port-Range Attribute (refer to Section 3.1.2).

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      Type     |     Length    |         portRangeStart
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
             portRangeStart        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 12

Type:

TBA13: This uniquely refers to the IPFIX Element ID 361.

TLV8-Length:

4.

portRangeStart:

Integer. This field contains the data (unsigned16) of (361) defined in IPFIX, right justified, and unused bits must be set to zero.

3.2.10. IP-Port-Range-End TLV

This TLV (Figure 13) uses format defined in [RFC6929]. Its Type field contains a value that uniquely refers to IPFIX Element portRangeEnd (362), and its Value field contains IPFIX Element portRangeEnd, which is the largest port number of a range of contiguous transport ports (refer to [IPFIX]).

IP-Port-Range-End TLV is included as part of the IP-Port-Range Attribute (refer to Section 3.1.2).

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      Type     |     Length    |          portRangeEnd
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
              portRangeEnd         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 13

Type:

TBA14: This uniquely refers to IPFIC Element ID 362.

Length:

4. The Length field for IP-Port-Range-End TLV.

portRangeEnd:

Integer. This field contains the data (unsigned16) of (362) defined in IPFIX, right justified, and unused bits must be set to zero.

3.2.11. IP-Port-Local-Id TLV

This TLV (Figure 14) uses format defined in [RFC6929]. Its Type field contains a value that uniquely refers to IPFIX Element localID (TBAx3), and its Value field contains IPFIX Element localID, which is a local significant identifier as explained below.

In some CGN deployment scenarios such as DS-Extra-Lite [RFC6619] and Lightweight 4over6 [I-D.ietf-softwire-lw4over6], parameters at a customer premise such as MAC address, interface ID, VLAN ID, PPP session ID, IPv6 prefix, VRF ID, etc., may also be required to pass to the RADIUS server as part of the accounting record.

IP-Port-Local-Id TLV can be included as part of the IP-Port-Range Attribute (refer to Section 3.1.2) and IP-Port-Forwarding-Map Attribute (refer to Section 3.1.3).

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Type     |     Length    |        localID ....
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 14

Type:

TBA15: This uniquely refers to IPFIX Element ID TBD.

Length:

Variable number of bytes.

localID:

string. This field contains the data (string) of (TBAX3) defined in IPFIX. This is a local session identifier at the customer premise, such as MAC address, interface ID, VLAN ID, PPP sessions ID, VRF ID, IPv6 address/prefix, etc.

4. Applications, Use Cases and Examples

This section describes some applications and use cases to illustrate the use of the attributes proposed in this document.

4.1. Managing CGN Port Behavior using RADIUS

In a broadband network, customer information is usually stored on a RADIUS server, and the BNG hosts the NAS. The communication between the NAS and the RADIUS server is triggered by a subscriber when the user signs in to the Internet service, where either PPP or DHCP/DHCPv6 is used. When a user signs in, the NAS sends a RADIUS Access-Request message to the RADIUS server. The RADIUS server validates the request, and if the validation succeeds, it in turn sends back a RADIUS Access-Accept message. The Access-Accept message carries configuration information specific to that user, back to the NAS, where some of the information would pass on to the requesting user via PPP or DHCP/DHCPv6.

A CGN function in a broadband network would most likely reside on a BNG. In that case, parameters for CGN port/identifier mapping behavior for users can be configured on the RADIUS server. When a user signs in to the Internet service, the associated parameters can be conveyed to the NAS, and proper configuration is accomplished on the CGN device for that user.

Also, CGN operation status such as CGN port/identifier allocation and de-allocation for a specific user on the BNG can also be transmitted back to the RADIUS server for accounting purpose using the RADIUS protocol.

RADIUS protocol has already been widely deployed in broadband networks to manage BNG, thus the functionality described in this specification introduces little overhead to the existing network operation.

In the following sub-sections, we describe how to manage CGN behavior using RADIUS protocol, with required RADIUS extensions proposed in Section 3.

4.1.1. Configure IP Port Limit for a User

In the face of IPv4 address shortage, there are currently proposals to multiplex multiple subscribers’ connections over a smaller number of shared IPv4 addresses, such as Carrier Grade NAT [RFC6888], Dual-Stack Lite [RFC6333], NAT64 [RFC6146], etc. As a result, a single IPv4 public address may be shared by hundreds or even thousands of subscribers. As indicated in [RFC6269], it is therefore necessary to impose limits on the total number of ports available to an individual subscriber to ensure that the shared resource, i.e., the IPv4 address remains available in some capacity to all the subscribers using it, and port limiting is also documented in [RFC6888] as a requirement.

The IP port limit imposed to a specific subscriber may be on the total number of TCP and UDP ports plus the number of ICMP identifiers, or with other granularities as defined in Section 3.1.1.

The per-subscriber based IP port limit is configured on a RADIUS server, along with other user information such as credentials. The value of these IP port limit is based on service agreement and its specification is out of the scope of this document.

When a subscriber signs in to the Internet service successfully, the IP port limit for the subscriber is passed to the BNG based NAS, where CGN also locates, using a new RADIUS attribute called IP-Port-Limit (defined in Section 3.1.1), along with other configuration parameters. While some parameters are passed to the subscriber, the IP port limit is recorded on the CGN device for imposing the usage of TCP/UDP ports and ICMP identifiers for that subscriber.

Figure 15 illustrates how RADIUS protocol is used to configure the maximum number of TCP/UDP ports for a given subscriber on a NAT44 device.

```
   User                    NAT44/NAS                       AAA
    |                         BNG                         Server
    |                          |                             |
    |                          |                             |
    |----Service Request------>|                             |
    |                          |                             |
    |                          |-----Access-Request -------->|
    |                          |                             |
    |                          |<----Access-Accept-----------|
    |                          |     (IP-Port-Limit)         |
    |                          |     (for TCP/UDP ports)     |
    |<---Service Granted ------|                             |
    |    (other parameters)    |                             |
    |                          |                             |
    |                  (NAT44 external port                  |
    |                   allocation and                       |
    |                   IPv4 address assignment)             |
    |                          |                             |
```

Figure 15: RADIUS Message Flow for Configuring NAT44 Port Limit

The IP port limit created on a CGN device for a specific user using RADIUS extension may be changed using RADIUS CoA message [RFC5176] that carries the same RADIUS attribute. The CoA message may be sent from the RADIUS server directly to the NAS, which once accepts and sends back a RADIUS CoA ACK message, the new IP port limit replaces the previous one.

Figure 16 illustrates how RADIUS protocol is used to increase the TCP/UDP port limit from 1024 to 2048 on a NAT44 device for a specific user.

```
   User                     NAT/NAS                           AAA
    |                         BNG                            Server
    |                          |                               |
    |              TCP/UDP Port Limit (1024)                   |
    |                          |                               |
    |                          |<---------CoA Request----------|
    |                          |       (IP-Port-Limit)         |
    |                          |       (for TCP/UDP ports)     |
    |                          |                               |
    |              TCP/UDP Port Limit (2048)                   |
    |                          |                               |
    |                          |---------CoA Response--------->|
    |                          |                               |
```

Figure 16: RADIUS Message Flow for changing a user’s NAT44 port limit

4.1.2. Report IP Port Allocation/De-allocation

Upon obtaining the IP port limit for a subscriber, the CGN device needs to allocate a TCP/UDP port or an ICMP identifiers for the subscriber when receiving a new IP flow sent from that subscriber.

As one practice, a CGN may allocate a bulk of TCP/UDP ports or ICMP identifiers once at a time for a specific user, instead of one port/ identifier at a time, and within each port bulk, the ports/ identifiers may be randomly distributed or in consecutive fashion. When a CGN device allocates bulk of TCP/UDP ports and ICMP identifiers, the information can be easily conveyed to the RADIUS server by a new RADIUS attribute called the IP-Port-Range (defined in Section 3.1.2). The CGN device may allocate one or more TCP/UDP port ranges or ICMP identifier ranges, or generally called IP port ranges, where each range contains a set of numbers representing TCP/UDP ports or ICMP identifiers, and the total number of ports/identifiers must be less or equal to the associated IP port limit imposed for that subscriber. A CGN device may choose to allocate a small port range, and allocate more at a later time as needed; such practice is good because its randomization in nature.

At the same time, the CGN device also needs to decide the shared IPv4 address for that subscriber. The shared IPv4 address and the pre-allocated IP port range are both passed to the RADIUS server.

When a subscriber initiates an IP flow, the CGN device randomly selects a TCP/UDP port or ICMP identifier from the associated and pre-allocated IP port range for that subscriber to replace the original source TCP/UDP port or ICMP identifier, along with the replacement of the source IP address by the shared IPv4 address.

A CGN device may decide to "free" a previously assigned set of TCP/ UDP ports or ICMP identifiers that have been allocated for a specific subscriber but not currently in use, and with that, the CGN device must send the information of the de-allocated IP port range along with the shared IPv4 address to the RADIUS server.

Figure 17 illustrates how RADIUS protocol is used to report a set of ports allocated and de-allocated, respectively, by a NAT44 device for a specific user to the RADIUS server.

```
   Host                    NAT44/NAS                        AAA
    |                         BNG                          Server
    |                          |                             |
    |                          |                             |
    |----Service Request------>|                             |
    |                          |                             |
    |                          |-----Access-Request -------->|
    |                          |                             |
    |                          |<----Access-Accept-----------|
    |<---Service Granted ------|                             |
    |    (other parameters)    |                             |
   ...                        ...                           ...
    |                          |                             |
    |                          |                             |
    |               (NAT44 decides to allocate               |
    |                a TCP/UDP port range for the user)      |
    |                          |                             |
    |                          |-----Accounting-Request----->|
    |                          |    (IP-Port-Range           |
    |                          |     for allocation)         |
   ...                        ...                           ...
    |                          |                             |
    |               (NAT44 decides to de-allocate            |
    |                a TCP/UDP port range for the user)      |
    |                          |                             |
    |                          |-----Accounting-Request----->|
    |                          |    (IP-Port-Range           |
    |                          |     for de-allocation)      |
    |                          |                             |
```

Figure 17: RADIUS Message Flow for reporting NAT44 allocation/de-allocation of a port set

4.1.3. Configure Forwarding Port Mapping

In most scenarios, the port mapping on a NAT device is dynamically created when the IP packets of an IP connection initiated by a user arrives. For some applications, the port mapping needs to be pre-defined allowing IP packets of applications from outside a CGN device to pass through and "port forwarded" to the correct user located behind the CGN device.

Port Control Protocol [RFC6887], provides a mechanism to create a mapping from an external IP address and port to an internal IP address and port on a CGN device just to achieve the "port forwarding" purpose. PCP is a server-client protocol capable of creating or deleting a mapping along with a rich set of features on a CGN device in dynamic fashion. In some deployment, all users need is a few, typically just one pre-configured port mapping for applications such as web cam at home, and the lifetime of such a port mapping remains valid throughout the duration of the customer’s Internet service connection time. In such an environment, it is possible to statically configure a port mapping on the RADIUS server for a user and let the RADIUS protocol to propagate the information to the associated CGN device.

Figure 18 illustrates how RADIUS protocol is used to configure a forwarding port mapping on a NAT44 device by using RADIUS protocol.

```
Host                         NAT/NAS                          AAA
 |                             BNG                           Server
 |                             |                               |
 |----Service Request------>|                               |
 |                             |                               |
 |                             |---------Access-Request------->|
 |                             |                               |
 |                             |<--------Access-Accept---------|
 |                             |   (IP-Port-Forwarding-Map)    |
 |<---Service Granted ------|                               |
 |    (other parameters)       |                               |
 |                             |                               |
 |                  (Create a port mapping                     |
 |                   for the user, and                         |
 |                   associate it with the                     |
 |                   internal IP address                       |
 |                   and external IP address)                  |
 |                             |                               |
 |                             |                               |
 |                             |------Accounting-Request------>|
 |                             |    (IP-Port-Forwarding-Map)   |
```

Figure 18: RADIUS Message Flow for configuring a forwarding port mapping

A port forwarding mapping that is created on a CGN device using RADIUS extension as described above may also be changed using RADIUS CoA message [RFC5176] that carries the same RADIUS associate. The CoA message may be sent from the RADIUS server directly to the NAS, which once accepts and sends back a RADIUS CoA ACK message, the new port forwarding mapping then replaces the previous one.

Figure 19 illustrates how RADIUS protocol is used to change an existing port mapping from (a:X) to (a:Y), where "a" is an internal port, and "X" and "Y" are external ports, respectively, for a specific user with a specific IP address

```
   Host                     NAT/NAS                         AAA
     |                        BNG                          Server
     |                         |                             |
     |                   Internal IP Address                 |
     |                   Port Map (a:X)                      |
     |                         |                             |
     |                         |<---------CoA Request----------|
     |                         |    (IP-Port-Forwarding-Map)   |
     |                         |                             |
     |                   Internal IP Address                 |
     |                   Port Map (a:Y)                      |
     |                         |                             |
     |                         |---------CoA Response--------->|
     |                         |    (IP-Port-Forwarding-Map)   |
```

Figure 19: RADIUS Message Flow for changing a user's forwarding port mapping

4.1.4. An Example

An Internet Service Provider (ISP) assigns TCP/UDP 500 ports for the subscriber Joe. This number is the limit that can be used for TCP/UDP ports on a NAT44 device for Joe, and is configured on a RADIUS server. Also, Joe asks for a pre-defined port forwarding mapping on the NAT44 device for his web cam applications (external port 5000 maps to internal port 80).

When Joe successfully connects to the Internet service, the RADIUS server conveys the TCP/UDP port limit (1000) and the forwarding port mapping (external port 5000 to internal port 80) to the NAT44 device, using IP-Port-Limit attribute and IP-Port-Forwarding-Map attribute, respectively, carried by an Access-Accept message to the BNG where NAS and CGN co-located.

Upon receiving the first outbound IP packet sent from Joe's laptop, the NAT44 device decides to allocate a small port pool that contains 40 consecutive ports, from 3500 to 3540, inclusively, and also assign a shared IPv4 address 192.0.2.15, for Joe. The NAT44 device also randomly selects one port from the allocated range (say 3519) and use that port to replace the original source port in outbound IP packets.

For accounting purpose, the NAT44 device passes this port range (3500-3540) and the shared IPv4 address 192.0.2.15 together to the RADIUS server using IP-Port-Range attribute carried by an Accounting-Request message.

When Joe works on more applications with more outbound IP sessions and the port pool (3500-3540) is close to exhaust, the NAT44 device allocates a second port pool (8500-8800) in a similar fashion, and also passes the new port range (8500-8800) and IPv4 address 192.0.2.15 together to the RADIUS server using IP-Port-Range attribute carried by an Accounting-Request message. Note when the CGN allocates more ports, it needs to assure that the total number of ports allocated for Joe is within the limit.

Joe decides to upgrade his service agreement with more TCP/UDP ports allowed (up to 1000 ports). The ISP updates the information in Joe’s profile on the RADIUS server, which then sends a CoA-Request message that carries the IP-Port-Limit attribute with 1000 ports to the NAT44 device; the NAT44 device in turn sends back a CoA-ACK message. With that, Joe enjoys more available TCP/UDP ports for his applications.

When Joe travels, most of the IP sessions are closed with their associated TCP/UDP ports released on the NAT44 device, which then sends the relevant information back to the RADIUS server using IP-Port-Range attribute carried by Accounting-Request message.

Throughout Joe’s connection with his ISP Internet service, applications can communicate with his web cam at home from external realm directly traversing the pre-configured mapping on the CGN device.

When Joe disconnects from his Internet service, the CGN device will de-allocate all TCP/UDP ports as well as the port-forwarding mapping, and send the relevant information to the RADIUS server.

4.2. Report Assigned Port Set for a Visiting UE

Figure 20 illustrates an example of the flow exchange which occurs when a visiting UE connects to a CPE offering WLAN service.

For identification purposes (see [RFC6967]), once the CPE assigns a port set, it issues a RADIUS message to report the assigned port set.

```
   UE         CPE             NAS                           AAA
    |                         BNG                         Server
    |                          |                             |
    |                          |                             |
    |----Service Request------>|                             |
    |                          |                             |
    |                          |-----Access-Request -------->|
    |                          |                             |
    |                          |<----Access-Accept-----------|
    |<---Service Granted ------|                             |
    |    (other parameters)    |                             |
   ...          |             ...                           ...
    |<---IP@----|              |                             |
    |           |              |                             |
    |   (CPE assigns a TCP/UDP port                          |
    |   range for this visiting UE)                          |
    |           |                                            |
    |           |--Accounting-Request-...------------------->|
    |           |    (IP-Port-Range                          |
    |           |     for allocation)                        |
   ...          |             ...                           ...
    |           |              |                             |
    |           |              |                             |
    |   (CPE withdraws a TCP/UDP port                        |
    |   range for a visiting UE)                             |
    |           |                                            |
    |           |--Accounting-Request-...------------------->|
    |           |    (IP-Port-Range                          |
    |           |     for de-allocation)                     |
    |           |                                            |
```

Figure 20: RADIUS Message Flow for reporting CPE allocation/de-allocation of a port set to a visiting UE

5. Table of Attributes

This document proposes three new RADIUS attributes and their formats are as follows:

- o IP-Port-Limit: TBA1.TBA2.[TBA6, TBA5, {TBA7}]

- o IP-Port-Range: TBA1.TBA3.[TBA12, TBA5, {TBA13, TBA14}, {TBA7}, {TBA15}].

- o IP-Port-Forwarding-Map: TBA1.TBA4.[TBA10, TBA11, TBA5, {TBA8 | TBA9}, {TBA7}]

The following table provides a guide as what type of RADIUS packets that may contain these attributes, and in what quantity.

Request	Accept	Reject	Challenge	Acct. Request	#	Attribute
0+	0+	0	0	0+	TBA	IP-Port-Limit
0	0	0	0	0+	TBA	IP-Port-Range
0+	0+	0	0	0+	TBA	IP-Port-Forwarding-Map

The following table defines the meaning of the above table entries.

0	This attribute MUST NOT be present in packet.
0+	Zero or more instances of this attribute MAY be present in packet.

6. Security Considerations

This document does not introduce any security issue than what has been identified in [RFC2865].

7. IANA Considerations

This document requires new code point assignments for both IPFIX Elements and RADIUS attributes as explained in the following sections.

7.1. IANA Considerations on New IPFIX Elements

The following are code point assignments for new IPFIX Elements as requested by this document:

o transportType (refer to Section 3.2.1): The identifier of this IPFIX Element is TBAx1. The data type of this IPFIX Element is unsigned8, and the Element’s value indicates TCP/UDP ports and ICMP Identifiers (1), TCP/UDP ports (2), TCP ports (3), UDP ports (4) or ICMP identifiers (5).

o natTransportLimit (refer to Section 3.2.2): The identifier of this IPFIX Element is TBAx2. The data type of this IPFIX Element is unsigned16, and the Element’s value is the max number of IP transport ports to be assigned to an end user associated with one or more IPv4 addresses.

o localID (refer to Section 3.2.11): The identifier of this IPFIX Element is TBAx3. The data type of this IPFIX Element is string, and the Element’s value is an IPv4 or IPv6 address, a MAC address, a VLAN ID, etc.

7.2. IANA Considerations on New RADIUS Attributes

The following are new code point assignment for RADIUS extensions as requested by this document:

- o TBA1: This value is allocated from Radius Extended-Type space. Refer to Section 3.1.1, Section 3.1.2, and Section 3.1.3.

- o TBA2: This is allocated from TBA1, so TBA1.TBA2 identifies a new RADIUS attribute IP-Port-Limit. Refer to Section 3.1.1.

- o TBA3: This is allocated from TBA1, so TBA1.TBA3 indentifies a new RADIUS attribute IP-Port-Range. Refer to Section 3.1.2.

- o TBA4: This is allocated from TBA1, so TBA1.TBA4 indentifies a new RADISU attribute IP-Port-Forwarding-Map. Refer to Section 3.1.3.

- o TBA5 (refer to Section 3.2.1): This is for the Type field of IP-Port-Type TLV. It should be allocated as TLV data type. The Value filed of this TLV contains the data of IPFIX Element transportType (TBAx1).

- o TBA6 (refer to Section 3.2.2): This is for the Type field of IP-Port-Limit TLV. It should be allocated as TLV data type. The Value field of this TLV contains the data of IPFIX Element natTransportLimit(TBAx2).

- o TBA7 (refer to Section 3.2.3): This is for the Type field of IP-Port-Ext-IPv4-Addr TLV. It should be allocated as TLV data type. The Value field of this TLV contains the data of IPFIX Element postNATSourceIPv4Address(225).

- o TBA8 (refer to Section 3.2.4): This is for the Type field of IP-Port-Int-IPv4-Addr TLV. It should be allocated as TLV data type. The Value field of this TLV contains the data of IPFIX Element sourceIPv4Address(8).

- o TBA9 (refer to Section 3.2.5): This is for the Type field of IP-Port-Int-IPv6-Addr TLV. It should be allocated as TLV data type. The Value field of this TLV contains the data of IPFIX Element sourceIPv6Address(27).

- o TBA10 (refer to Section 3.2.6): This is for the Type field of IP-Port-Int-Port TLV. It should be allocated as TLV data type. The Value field of this TLV containss the data of IPFIX Element sourceTransportPort(7).

- o TBA11 (refer to Section 3.2.7): This is for the Type field of IP-Port-Ext-port TLV. It should be allocated as TLV data type. The Value field of this TLV contains the data of IPFIX Element postNAPTSourceTransportPort(227).
- o TBA12 (refer to Section 3.2.8): This is for the Type field of IP-Port-Alloc TLV. It should be allocated as TLV data type. The Value field of this TLV contains the data of IPFIX Element natEvent(230).
- o TBA13 (refer to Section 3.2.9): This is for the Type field of IP-Port-Range-Start TLV. It should be allocated as TLV data type. The Value field of this TLV contains the data of IPFIX Element portRangeStart(361).
- o TBA14 (refer to Section 3.2.10): This is for the Type field of IP-Port-Range-End TLV. It should be allocated as TLV data type. The Value field of this TLV contains the data of IPFIX Element portRangeEnd(362).
- o TBA15 (refer to Section 3.2.11): This is for the Type field of IP-Port-Local-Id TLV. It should be allocated as TLV data type. The Value field of this TLV contains the data of IPFIX Element localID(TBAx3).

8. Acknowledgements

Many thanks to Dan Wing, Roberta Maglione, Daniel Derksen, David Thaler, Alan Dekok, Lionel Morand, and Peter Deacon for their useful comments and suggestions.

9. References

9.1. Normative References

[IPFIX]   IANA, "IP Flow Information Export (IPFIX) Entities", <http://www.iana.org/assignments/ipfix/ipfix.xhtml>.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <http://www.rfc-editor.org/info/rfc1918>.

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

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, DOI 10.17487/RFC2629, June 1999, <http://www.rfc-editor.org/info/rfc2629>.

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

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

[RFC6929] DeKok, A. and A. Lior, "Remote Authentication Dial In User Service (RADIUS) Protocol Extensions", RFC 6929, DOI 10.17487/RFC6929, April 2013, <http://www.rfc-editor.org/info/rfc6929>.

[RFC7012] Claise, B., Ed. and B. Trammell, Ed., "Information Model for IP Flow Information Export (IPFIX)", RFC 7012, DOI 10.17487/RFC7012, September 2013, <http://www.rfc-editor.org/info/rfc7012>.

[TR-146]   Broadband Forum, "TR-146: Subscriber Sessions", <http://www.broadband-forum.org/technical/download/TR-146.pdf>.

9.2. Informative References

[I-D.gundavelli-v6ops-community-wifi-svcs]
Gundavelli, S., Grayson, M., Seite, P., and Y. Lee, "Service Provider Wi-Fi Services Over Residential Architectures", draft-gundavelli-v6ops-community-wifi-svcs-06 (work in progress), April 2013.

[I-D.ietf-softwire-lw4over6]
Cui, Y., Qiong, Q., Boucadair, M., Tsou, T., Lee, Y., and I. Farrer, "Lightweight 4over6: An Extension to the DS-Lite Architecture", draft-ietf-softwire-lw4over6-13 (work in progress), November 2014.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, DOI 10.17487/RFC3022, January 2001, <http://www.rfc-editor.org/info/rfc3022>.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <http://www.rfc-editor.org/info/rfc6146>.

[RFC6269] Ford, M., Ed., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, DOI 10.17487/RFC6269, June 2011, <http://www.rfc-editor.org/info/rfc6269>.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, DOI 10.17487/RFC6333, August 2011, <http://www.rfc-editor.org/info/rfc6333>.

[RFC6619] Arkko, J., Eggert, L., and M. Townsley, "Scalable Operation of Address Translators with Per-Interface Bindings", RFC 6619, DOI 10.17487/RFC6619, June 2012, <http://www.rfc-editor.org/info/rfc6619>.

[RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, DOI 10.17487/RFC6887, April 2013, <http://www.rfc-editor.org/info/rfc6887>.

[RFC6888] Perreault, S., Ed., Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common Requirements for Carrier-Grade NATs (CGNs)", BCP 127, RFC 6888, DOI 10.17487/RFC6888, April 2013, <http://www.rfc-editor.org/info/rfc6888>.

[RFC6967] Boucadair, M., Touch, J., Levis, P., and R. Penno, "Analysis of Potential Solutions for Revealing a Host Identifier (HOST_ID) in Shared Address Deployments", RFC 6967, DOI 10.17487/RFC6967, June 2013, <http://www.rfc-editor.org/info/rfc6967>.

Authors’ Addresses

Dean Cheng
Huawei
2330 Central Expressway
Santa Clara, California 95050
USA

Email: dean.cheng@huawei.com

Jouni Korhonen
Broadcom Corporation
3151 Zanker Road
San Jose 95134
USA

Email: jouni.nospam@gmail.com

Mohamed Boucadair
France Telecom
Rennes
France

Email: mohamed.boucadair@orange.com

Senthil Sivakumar
Cisco Systems
7100-8 Kit Creek Road
Research Triangle Park, North Carolina
USA

Email: ssenthil@cisco.com