

Network Working Group
INTERNET-DRAFT
Category: Experimental
<draft-dekok-radius-ipfix-00.txt>
Expires: April 5, 2013
8 October 2012

A. DeKok
FreeRADIUS
S. Winter
RESTENA

RADIUS accounting via IPFIX
draft-dekok-radius-ipfix

Abstract

The Remote Authentication Dial In User Server (RADIUS) Protocol provides for a number of simple session-based metrics. There is a need to increase the number of metrics, and to add metrics for individual flows within a session. This document leverages IP Flow Information Export (IPFIX) to address both needs.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on April 12, 2011

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info/>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
1.1. Terminology	4
1.2. Requirements Language	4
2. IPFIX in RADIUS	5
2.1. IPFIX-Container Attribute	5
2.2. IPFIX to RADIUS Data Type Mappings	6
2.3. IPFIX ElementID to RADIUS TLV Mapping	7
2.4. Summary of the Mappings	8
3. IPFIX-Container in RADIUS Packets	8
3.1. IPFIX-Container in Access-Accept	8
3.2. IPFIX-Container in Accounting-Request	9
3.3. Administratively Configured Flows	9
4. Caveats	10
5. Examples	10
5.1. Access-Accept	10
5.2. Accounting-Request	11
6. Security Considerations	11
7. IANA Considerations	11
8. References	11
8.1. Normative References	11
8.2. Informative References	12
Appendix A - XML to RADIUS Dictionary Converter	13
Appendix A.1 - Code for XML to RADIUS Dictionary Converter ...	14

1. Introduction

The Remote Authentication Dial In User Server (RADIUS) Protocol defines in [RFC2866] and others metrics for transporting session-based accounting. We wish to extend those capabilities, while having a specification that satisfies the following criteria:

- * follows the RADIUS data model as defined in [RFC6158], and [EXTEN],
- * allows for an increased number of metrics per session,
- * allows for metrics for individual flows within a session,
- * can describe flows in an Access-Accept, for later accounting in an Accounting-Request,
- * requires minimal work from the RADEXT working group to define new metrics

We believe this document satisfies those criteria. We propose to do this by leveraging IP Flow Information Export (IPFIX) entries, defined in [IPFIX]. The benefits of this approach are many. We leverage existing expertise and work, we do not create a new standard to exchange the same information, etc.

1.1. Terminology

This document uses the following terms:

RADIUS client

A device that provides an access service for a user to a network. Also referred to as a Network Access Server, or NAS.

RADIUS server

A device that provides one or more of authentication, authorization, and/or accounting (AAA) services to a NAS.

RADIUS proxy

A RADIUS proxy acts as a RADIUS server to the NAS, and a RADIUS client to the RADIUS server.

1.2. Requirements Language

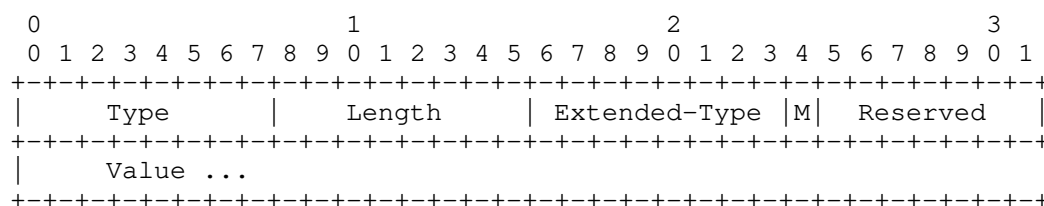
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. IPFIX in RADIUS

This section defines how IPFIX information is carried inside of RADIUS attributes.

2.1. IPFIX-Container Attribute

We define a new attribute, called IPFIX-Container. It leverages the "Long Extended" type defined in [EXTEN] Section X.Y. A summary of the Attribute format is shown below. The fields are transmitted from left to right.



Type

TBD - IANA allocation from the "long extended" type space

Length

The Length of the attribute, as previously defined in [RFC2865] Section 5. Permitted values are between 5 and 255. If a client or server receives an IPFIX-Container attribute with a Length of 2, 3, or 4, then that Attribute MUST be considered to be an "invalid attribute", and be handled as per [EXTEN] Section 2.7.

Extended-Type

TBD - IANA allocation from the "long extended" type space

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 Value field has been fragmented across multiple RADIUS attributes. When the More field is set (1), the attribute SHOULD have a Length field of value 255; it MUST NOT have a length Field of value 4; 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).

When the Length field of an attribute has value less than 255, the More field SHOULD be clear (0).

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 [EXTEN] Section 2.7.

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.

Value

The value is a series of Type-Length-Values (TLVs), which are the IPFIX information elements as defined in [IPFIX]. One flow MUST be encapsulated in one IPFIX-Container attribute. An IPFIX-Container attribute MUST NOT contain zero, or multiple flows.

2.2. IPFIX to RADIUS Data Type Mappings

The IPFIX specification defines a number of data types. Some can be mapped to RADIUS, others cannot. We document those mappings here. Where a data type is not mapped, IPFIX information elements using that data type MUST NOT appear in a RADIUS packet.

We use the RADIUS data types as defined in [RFC6158]. The following table defines the mappings.

Mapping Table

IPFIX	RADIUS
-----	-----
unsigned8	integer
unsigned16	integer
unsigned32	integer
unsigned64'	integer64
ipv4Address	IP Address
ipv6Address	IPv6 Address
string	text

octetArray	string
dateTimeSeconds	date
dateTimeMilliseconds	integer64
dateTimeMicroseconds	integer64
dateTimeNanoseconds	integer64
macAddress	string
boolean	integer

RADIUS does not permit eight (8) or sixteen (16) bit data types, so IPFIX "integer" types of those sizes are mapped to RADIUS 32-bit integers.

We want to avoid creating new RADIUS data types. Therefore, the IPFIX "dateTime" types are mapped to "integer64". The only exception is "dateTimeSeconds", which maps to the RADIUS "date" data type.

RADIUS does not have a "MAC Address" data type. Therefore the IPFIX "macAddress" data type maps to the RADIUS "string" data type.

2.3. IPFIX ElementID to RADIUS TLV Mapping

The IPFIX ElementIDs are fifteen (15) bits long, while the RADIUS attribute IDs are eight (8) bits. We therefore need a mapping from one to the other. The mapping involves allocating two layers of TLVs, which allows for sixteen (16) bits of mapping. The mapping function is as follows:

```
TLV1 = (ElementId >> 8) & 0xFF
TLV2 = ElementId & 0xFF
```

The resulting RADIUS attribute is then TBD.TLV1.TLV2. (IANA NOTE: Replace TBD with the allocated "long extended" attribute, e.g. 245.1).

We do not allocate or name TBD.TLV1 in the RADIUS attribute space. (IANA NOTE: Replace TBD with the allocated "long extended" attribute, e.g. 245.1). It is simple a container which is never referenced by itself.

We do not provide an exhaustive mapping of which IPFIX ElementID is appropriate (or not) for inclusion in which RADIUS packet. We instead provide general guidelines and recommendations.

IPFIX ElementIDs relating to counters and flow identification are useful in RADIUS. Other ElementIDs are less useful, and SHOULD NOT be used. ElementIDs related to security and packet signing MUST NOT be used in RADIUS.

2.4. Summary of the Mappings

The mapping from IPFIX Information Elements to RADIUS attributes is simple and mechanical. New IPFIX Information Elements can be used by RADIUS without requiring new specifications in the RADIUS space.

A short Perl program is available in Appendix A, below. It provides an example of how the mapping can be done automatically. However, it does not provide a standard of any kind.

3. IPFIX-Container in RADIUS Packets

The IPFIX-Container attribute is permitted in Access-Accept packets, and in Accounting-Request packets. It MUST NOT occur in any other kind of RADIUS packet.

3.1. IPFIX-Container in Access-Accept

The IPFIX-Container can be sent in an Access-Accept packet. This indicates a request from the server that the NAS send Accounting-Request packets containing the requested information. The NAS MAY ignore this request, and send different information, or no information.

The contents of the IPFIX-Container are ElementIDs which describe a particular flow. Multiple IPFIX-Container attributes MAY be sent in an Access-Accept.

The ElementIDs which describe a particular flow are not enumerated here. Instead, we suggest ElementIDs relating to IP addresses, protocols, ports, etc. Where ElementIDs relating to flow identification are omitted, the IPFIX-Container SHOULD be interpreted as applying to all traffic.

The only restriction is that each IPFIX-Container attribute MUST contain a flowID. It is RECOMMENDED that the RADIUS server allocate these flowIDs through a robust process that generates globally and temporally unique values. The method for doing so is not described here.

An Access-Accept MUST NOT contain multiple IPFIX-Container attributes with the same flowID.

This restriction comes about as a result of issues seen with Acct-Session-Id. Deployment experience shows that many NAS implementations choose repeating values for Acct-Session-Id. That repetition makes it difficult for RADIUS servers to track individual user sessions. Allowing for flow-based accounting would make that

problem worse.

This document instead requires that flowIDs are allocated by the RADIUS server. NAS implementations MUST NOT create flowIDs.

3.2. IPFIX-Container in Accounting-Request

The IPFIX-Container can be sent in an Accounting-Request packet. This indicates that the NAS is sending accounting information about a particular flow.

The contents of the IPFIX-Container are ElementIDs which describe a the accounting data for a particular flow. Multiple IPFIX-Container attributes MAY be sent in an Accounting-Request. The IPFIX-Container MAY avoid sending ElementIDs which describe the flow. i.e. ElementIDs which are sent in an Access-Accept. The flowID is generally sufficient to uniquely describe a flow.

However, in situations where the flowIDs are administratively configured, those ElementIDs which describe the flow MUST be included in the IPFIX-Container attribute. This inclusion ensures that the combination of flowID and flow description is unique.

The ElementIds which describe accounting for a particular flow are not enumerated here. Instead, we suggest ElementIDs relating to packet counting, octet counting, time, etc.

The only restriction is that each IPFIX-Container attribute MUST contain a flowID. The NAS MUST use flowIDs taken from a corresponding Access-Accept packet, or flowIDs which have been administratively configured. It MUST NOT generate flowIDs by itself.

An Accounting-Request packet MUST NOT contain multiple IPFIX-Container attributes with the same flowID.

3.3. Administratively Configured Flows

A NAS MAY permit an administrator to manually configure flows. In that case, it can send flows in an Accounting-Request packet which were not requested in an Accounting-Accept packet. In that case, ElementIDs which describe the flow MUST be included in the IPFIX-Container attribute. The flowID MUST also be configured by the administrator, and not created by the NAS.

For administratively configured flows, a NAS MAY give the administrator the choice of using incrementing flowIDs for each flow in a session, starting from the value one (1).

4. Caveats

The flows in IPFIX are unidirectional. RADIUS uses different attributes to count data for each direction. As a result, where RADIUS would use two attributes Acct-Input-Octets and Acct-Output-Octets, we would require two IPFIX-Container attributes.

Not all of the IPFIX data types are mapped. We have no suggestions for what to do with the more complex data types.

The IPFIX "basicList" data type appears to be unnecessary in RADIUS. Instead of encapsulating multiple ports in a basicList, we can simply send multiple port descriptions in one IPFIX-Container attribute.

Mapping eight and sixteen-bit IPFIX integers to a 32-bit RADIUS integer type is suboptimal. However, it meets the requirements of [RFC6158]. The major benefit of reusing the RADIUS data model outweighs the minor benefit of a slightly more efficient packing.

IPFIX allows for sending data up to 64K in size. The multiple encapsulations define here allows for only a maximum of 249 octets. This limitation does not matter for the bulk of the ElementIDs which are relevant to RADIUS. For the rest, we offer no suggestion other than to administratively limit strings to a particular length.

Not all implementations will support this specification. Therefore, the existing RADIUS accounting attributes SHOULD be sent in addition to the IPFIX-Container attribute. Sending only IPFIX-Container is NOT RECOMMENDED.

5. Examples

This section contains examples. We use names for the attributes taken from the output of the Perl program in Appendix A. We show only the IPFIX-Container attribute and the TLVs related to IPFIX ElementIDs. We do not show intermediate TLVs.

5.1. Access-Accept

```
IPFIX-Container
  IPFIX-flowId = 1
  IPFIX-flowDirection = 0x00
IPFIX-Container
  IPFIX-flowId = 2
  IPFIX-flowDirection = 0x01
```

5.2. Accounting-Request

```
Acct-Input-Octets = 1024
Acct-Output-Octets = 2048
Acct-Input-Packets = 10
Acct-Output-Packets = 20
IPFIX-Container
  IPFIX-flowId = 1
  IPFIX-flowDirection = 0x00
  IPFIX-octetTotalCount = 1024
  IPFIX-packetTotalCount = 10
IPFIX-Container
  IPFIX-flowId = 2
  IPFIX-flowDirection = 0x01
  IPFIX-octetTotalCount = 2048
  IPFIX-packetTotalCount = 20
```

6. Security Considerations

This document defines new RADIUS attributes, but makes no changes to the RADIUS protocol. As such, there are no security considerations.

The IPFIX protocol defines security related ElementIDs such as messageMD5Checksum. These ElementIDs MUST NOT appear in a RADIUS packet. Existing RADIUS security is sufficient to protect the packets.

7. IANA Considerations

This document requests the allocation of a new attribute, "IPFIX-Container", from the "long extended" RADIUS attribute type space.

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.
- [RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.
- [IPFIX] <http://www.iana.org/assignments/ipfix/>
- [EXTEN] DeKok, A, Lior, A, "Remote Authentication Dial In User Service (RADIUS) Protocol Extensions", draft-ietf-radext-radius-

extensions-05.txt, (work in progress)

8.2. Informative References

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

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

Acknowledgments

The initial suggestion to leverage IPFIX was from Benoit Claise.

Appendix A - XML to RADIUS Dictionary Converter

The following page contains a simple Perl script which converts the [IPFIX] XML registry to a RADIUS dictionary file. It is intended as a guide to show that the conversion is mechanical. It does not specify a standard, and implementors are free to use their own method for converting the IPFIX registry to a RADIUS dictionary.

Appendix A.1 - Code for XML to RADIUS Dictionary Converter

```
#!/usr/bin/env perl
use XML::Simple;
use Data::Dumper;

$prefix = "TBD";          # IANA - to be updated

print "ATTRIBUTE IPFIX-Container $prefix long-extended\n";

$map = {
    'unsigned8'    => 'integer',
    'unsigned16'   => 'integer',
    'unsigned32'   => 'integer',
    'unsigned64'   => 'integer64',
    'ipv4Address'  => 'ipaddr',
    'ipv6Address'  => 'ipv6addr',
    'string'       => 'text',
    'octetArray'   => 'string',
    'dateTimeSeconds' => 'date',
    'dateTimeMilliseconds' => 'integer64',
    'dateTimeMicroseconds' => 'integer64',
    'dateTimeNanoseconds' => 'integer64',
    'macAddress'   => 'string',
    'boolean'      => 'integer',
};

$xml = new XML::Simple;
$data = $xml->XMLin("ipfix.xml");
$elements = $data->{registry}->{'ipfix-information-elements'}->{record};

foreach $record (@{$elements}) {
    next if $record->{unassigned};
    next if $record->{reserved};

    $name = $record->{name};
    $name =~ s/\s+//g;

    $supper = 1 + (($record->{elementId} & ~0xff) >> 8);
    $lower = $record->{elementId} & 0xff;
    $oid = "$prefix.$supper.$lower";

    $type = $record->{dataType};
    if (!defined $map->{$type}) {
        print "# ATTRIBUTE IPFix-$name $oid ", $type, "\n";
    } else {
        print "ATTRIBUTE IPFix-$name $oid ", $map->{$type}, "\n";
    }
}
```

}

Authors' Addresses

Alan DeKok
The FreeRADIUS Server Project
<http://freeradius.org/>

Email: aland@freeradius.org

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>

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 11, 2013

A. DeKok
FreeRADIUS
G. Halwasia
S. Danda
M. Kumar
Cisco Systems
October 8, 2012

Capability Negotiation in RADIUS
draft-halwasia-radext-capability-negotiation-01

Abstract

This document describes procedure and mechanism to exchange and negotiate capabilities between RADIUS client and RADIUS server.

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].

RADIUS-specific terminology is borrowed from [RFC2865] and [RFC2866].

Status of this Memo

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

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

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

This Internet-Draft will expire on April 11, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal

Provisions Relating to IETF Documents
(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Solution Description	3
2. RADIUS Packets	3
2.1. Capability-Request RADIUS Packet	3
2.2. Capability-Response RADIUS Packet	4
3. RADIUS Attributes	5
3.1. Capability-Add Attribute	6
3.2. Capability-Withdraw Attribute	6
3.3. Capability-Ack Attribute	7
4. RADIUS Capability-Id	8
5. RADIUS Client Behavior	8
6. RADIUS Server Behavior	9
7. Example	9
8. IANA Considerations	10
8.1. New Registry: Capability-Identifier	10
9. Security Considerations	11
10. Acknowledgements	11
11. References	11
11.1. Normative References	11
11.2. Informative References	12
Authors' Addresses	12

1. Introduction

Remote Authentication Dial In User Service (RADIUS) [RFC2865] and [RFC2866] is widely used protocol for Authentication, Authorization and Accounting. There are quite a lot of extensions which are being done on RADIUS protocol and considering that RADIUS is being deployed quite extensively, it would be nice if RADIUS Client and Server can negotiate the capability to support those extensions. This specification recommends and envision each proposed capability to be as precise and narrowly defined as possible and having said that we envision fairly large number of capabilities rather than few broadly defined ones. For example [I-D.ietf-radext-radius-extensions] proposes extended attributes space along with few other extensions and it would be nice if RADIUS Client and Server can signal and negotiate support for 'extended attributes'. This document describes procedure and mechanism to exchange and negotiate capabilities between RADIUS client and RADIUS server.

1.1. Solution Description

This specification proposes to define two new RADIUS packet types to negotiate capabilities between RADIUS client and RADIUS server as defined in section 2. It also proposes to define 3 new attributes to be carried inside new RADIUS packet types. New RADIUS packets to negotiate capability has been chosen as it has minimal impact on the RADIUS security model and existing implementations. Following sections describes the new RADIUS packet types and attributes and describes their usage in negotiating capabilities. As per this specification Capability-Request RADIUS packets MUST NOT be proxied.

2. RADIUS Packets

This document defines following new RADIUS packet type to enable capability negotiation between RADIUS client and server.

2.1. Capability-Request RADIUS Packet

Capability-Request RADIUS Packets are sent to a RADIUS server to convey the capabilities RADIUS client intends to add and withdraw.

A summary of the Capability-Request packet format is shown below. The fields are transmitted from left to right.



Code

TBA1 - Capability-Request.

Identifier

The Identifier field MUST be changed whenever the content of the Attributes field changes, and whenever a valid reply has been received for a previous request. For retransmissions, the Identifier MUST remain unchanged.

Request Authenticator

The Request Authenticator value MUST be changed each time a new Identifier is used. It is calculated the same way as calculated for Access-Request RADIUS Packet as described in section 3 of RFC2865.

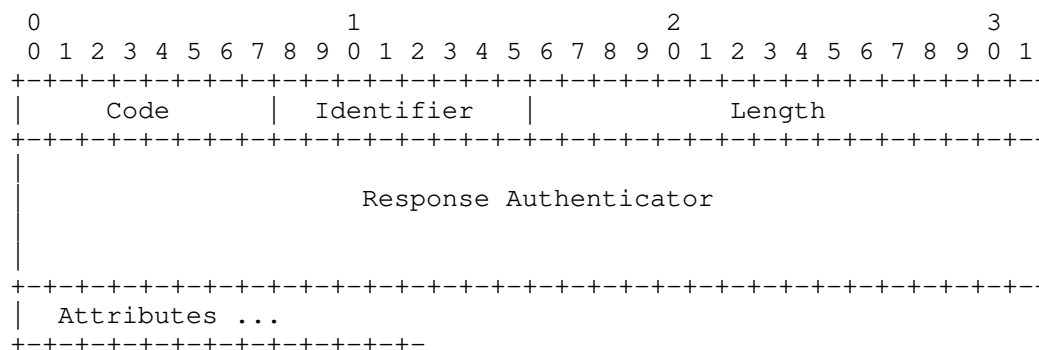
Attributes

The Attribute field is variable in length, and contains the list of Attributes that are required.

2.2. Capability-Response RADIUS Packet

Capability-Withdraw RADIUS Packets are sent to a RADIUS client in response to Capability-Request packet from RADIUS client.

A summary of the Capability-Withdraw packet format is shown below. The fields are transmitted from left to right.



Code

TBA2 - Capability-Response.

Identifier

The Identifier field is a copy of the Identifier field of the Capability-Request which caused this Capability-Response.

Response Authenticator

The Response Authenticator value is calculated from the Capability-Request value. It is calculated the same way as calculated for Access-Accept RADIUS Packet similar to as described in section 3 of RFC2865.

Attributes

The Attribute field is variable in length, and contains the list of Attributes that are required.

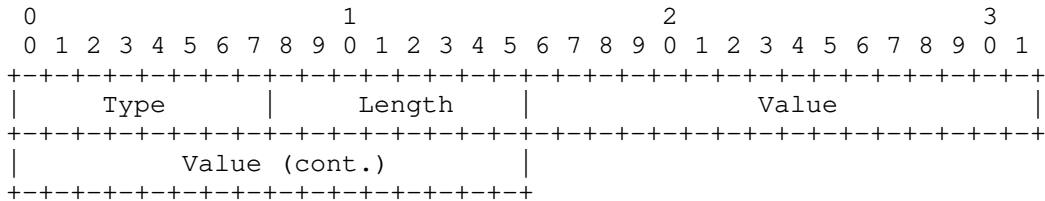
3. RADIUS Attributes

This document defines following new RADIUS attributes to enable capability negotiation between RADIUS client and server.

3.1. Capability-Add Attribute

This attribute indicates the capability which the client wants to add.

The format of the Capability-Add Attribute is:



Type

TBA3 - Capability-Add

Length

6

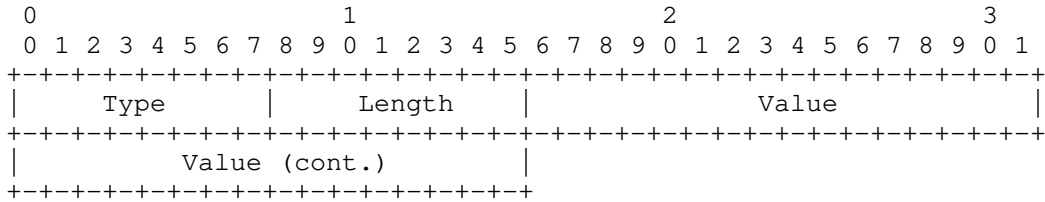
Value

Enumerated Data Type in 4-Octet unsigned integer defined in [RFC6158]. This field contains the capability-id as specified in section 4 of this document.

3.2. Capability-Withdraw Attribute

This attribute indicates the capability which the client wants to withdraw.

The format of the Capability-Withdraw Attribute is:



Type

TBA4 - Capability-Withdraw

Length

6

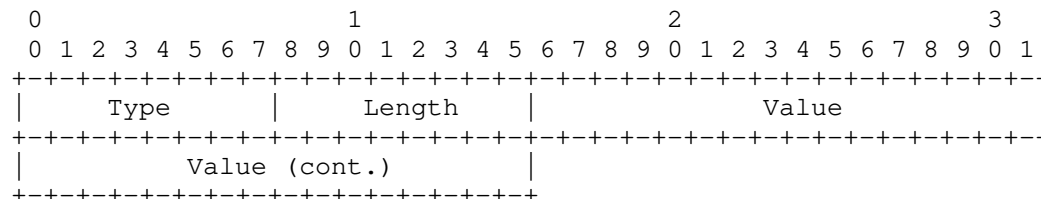
Value

Enumerated Data Type in 4-Octet unsigned integer defined in [RFC6158]. This field contains the capability-id as specified in section 4 of this document.

3.3. Capability-Ack Attribute

This attribute indicates the capability which the server wants to Acknowledge.

The format of the Capability-Ack Attribute is:



Type

TBA5 - Capability-Ack

Length

6

Value

Enumerated Data Type in 4-Octet unsigned integer defined in [RFC6158]. This field contains the capability-id as specified in section 4 of this document.

4. RADIUS Capability-Id

Value field of Capability-Add and Capability-Withdraw attributes defined above contains the Capability-Id of the capability RADIUS client wants to negotiate with RADIUS server. This document does not define any new capability and it's associated Capability-Id. Any specification which wants to use this mechanism of capability negotiation MUST define a new capability. Each new capability MUST be registered with IANA to get a capability-id from capability-id registry.

5. RADIUS Client Behavior

RADIUS Client willing to negotiate capabilities SHOULD send Capability-Request RADIUS Packet (defined in section 2) towards the RADIUS Server. RADIUS Client MUST include Capability-Add Attribute in Capability-Request RADIUS Packet for the capability client wants to add/negotiate. Client can also include Capability-Withdraw Attribute in the RADIUS packet in case it wants to withdraw the

capability it has negotiated earlier. Client MUST NOT add the Capability-Withdraw Attribute in the Capability-Request RADIUS Packet in case it has not negotiated the corresponding attribute earlier. Client can include multiple Capability-Add and/or Capability-Withdraw Attributes in the same Capability-Request RADIUS Packet. RADIUS client MUST add at least one Capability-Add and/or Capability-Withdraw Attribute in Capability-Request RADIUS Packet. Client MUST NOT include the Capability-Add and Capability-Withdraw Attribute for the same capability in the same Capability-Request RADIUS Packet.

Apart from including Capability-Add and/or Capability-Withdraw Attributes in the Capability-Request RADIUS Packet, Client can include NAS-Identifier [RFC2865] or one of the NAS-IP-Address[RFC2865]/NAS-IPv6-Address [RFC3162] for the purpose of RADIUS server to identify client.

6. RADIUS Server Behavior

RADIUS Server implementing this specification MUST respond back with Capability-Response RADIUS Packet after receiving a valid Capability-Request RADIUS Packet from the RADIUS Client. As specified in section 5, Client will advertise its capabilities by including Capability-Add and/or Capability-Withdraw Attributes in the same Capability-Request RADIUS Packet. RADIUS Server will find out the common set of agreed upon capabilities based upon the intersection in between capabilities received from client and its own capabilities. RADIUS Server MUST include a Capability-Ack Attribute for each of the agreed upon capabilities in the Capability-Response RADIUS Packet. RADIUS Server MUST NOT include Capability-Ack attributes for all those capabilities which it does not want to support/share with RADIUS Client. If the RADIUS Server does not support any of the capabilities specified in Capability-Request RADIUS Packet, it SHOULD send back an empty Capability-Response RADIUS Packet without including any Capability-Ack attribute.

RADIUS Server implementation which does not support capability negotiation specified in this specification MUST silently discard Capability-Request RADIUS Packet received from RADIUS Client.

7. Example

Following example figure shows the sequence of message exchanges which happens between RADIUS Client and RADIUS Server to negotiate capabilities.

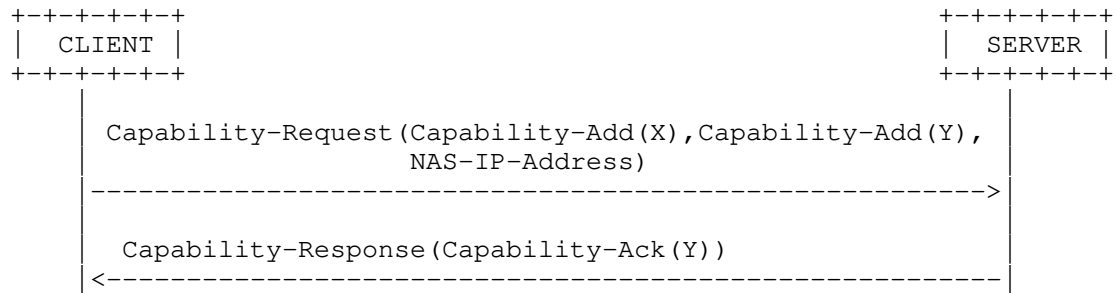


Figure 1: Capability Negotiation between Client and Server

Capability X = 'Understands 64-Bit Integers'

Capability Y = 'Supports Larger than 4K RADIUS packets'

8. IANA Considerations

The authors request that Packet Type, Attribute Types and Attribute Values defined in this document be registered by the Internet Assigned Numbers Authority (IANA) from the RADIUS namespaces as described in the "IANA Considerations" section of RFC 3575 [RFC3575], in accordance with BCP 26 [RFC5226]. For RADIUS packets, attributes and registries created by this document IANA is requested to place them at <http://www.iana.org/assignments/radius-types>.

This document defines the following RADIUS messages:

- Capability-Request
- Capability-Response

This document defines the following attributes:

- Capability-Add
- Capability-Withdraw
- Capability-Ack

Additionally, IANA is requested to create the following new registries listed in the subsections below.

8.1. New Registry: Capability-Identifier

This document also defines an Capability-Identifier registry (used in the value field of Capability-Add, Capability-Withdraw and Capability-Ack Attributes). IANA is requested to just allocate space

for this registry and this document does not request IANA to allocate any value from this registry.

Requests to IANA for a new value for a Capability Identifier will be approved by Expert Review. A designated expert will be appointed by the IESG.

9. Security Considerations

This document defines new RADIUS message types and new Attribute types, but otherwise makes no changes to the security of the RADIUS protocol.

10. Acknowledgements

11. References

11.1. Normative References

- [I-D.ietf-radext-radius-extensions]
DeKok, A. and A. Lior, "Remote Authentication Dial In User Service (RADIUS) Protocol Extensions",
draft-ietf-radext-radius-extensions-06 (work in progress),
June 2012.
- [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.
- [RFC3162] Aboba, B., Zorn, G., and D. Mitton, "RADIUS and IPv6", RFC 3162, August 2001.
- [RFC3575] Aboba, B., "IANA Considerations for RADIUS (Remote Authentication Dial In User Service)", RFC 3575, July 2003.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC6158] DeKok, A. and G. Weber, "RADIUS Design Guidelines", BCP 158, RFC 6158, March 2011.

11.2. Informative References

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

Authors' Addresses

Alan DeKok
FreeRADIUS

Phone:
Email: aland@deployingradius.com

Gaurav Halwasia
Cisco Systems
Cessna Business Park, Sarjapura Marathalli Outer Ring Road
Bangalore, KARNATAKA 560 087
India

Phone: +91 80 4429 2703
Email: ghalwasi@cisco.com

Satyanarayana Danda
Cisco Systems
Cessna Business Park, Sarjapura Marathalli Outer Ring Road
Bangalore, KARNATAKA 560 087
India

Phone: +91 80 4429 2684
Email: sdanda@cisco.com

Manoj Kumar
Cisco Systems
Cessna Business Park, Sarjapura Marathalli Outer Ring Road
Bangalore, KARNATAKA 560 087
India

Phone: +91 80 4429 2635
Email: magoyal@cisco.com

RADEXT Working Group
INTERNET-DRAFT
Category: Proposed Standard
Expires: October 1, 2014
Updates: 3580, 4072

Bernard Aboba
Microsoft Corporation
Jouni Malinen
Devicescape Software
Paul Congdon
Tallac Networks
Joseph Salowey
Cisco Systems
Mark Jones
Azuca Systems
28 March 2014

RADIUS Attributes for IEEE 802 Networks
draft-ietf-radext-ieee802ext-12.txt

Abstract

RFC 3580 provides guidelines for the use of the Remote Authentication Dialin User Service (RADIUS) within IEEE 802 local area networks (LANs). This document defines additional attributes for use within IEEE 802 networks, as well as clarifying the usage of the EAP-Key-Name attribute and the Called-Station-Id attribute. This document updates RFC 3580 as well as RFC 4072.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on October 1, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1.	Introduction	4
1.1	Terminology	4
1.2	Requirements Language	5
2.	RADIUS attributes	5
2.1	Allowed-Called-Station-Id	5
2.2	EAP-Key-Name	6
2.3	EAP-Peer-Id	7
2.4	EAP-Server-Id	8
2.5	Mobility-Domain-Id	9
2.6	Preauth-Timeout	10
2.7	Network-Id-Name	11
2.8	EAPoL-Announcement	12
2.9	WLAN-HESSID	14
2.10	WLAN-Venue-Info	14
2.11	WLAN-Venue-Language	15
2.12	WLAN-Venue-Name	16
2.13	WLAN-Reason-Code	17
2.14	WLAN-Pairwise-Cipher	18
2.15	WLAN-Group-Cipher	19
2.16	WLAN-AKM-Suite	20
2.17	WLAN-Group-Mgmt-Cipher	21
2.18	WLAN-RF-Band	22
3.	Table of attributes	23
4.	IANA Considerations	24
5.	Security Considerations	24
6.	References	25
6.1	Normative References	25
6.2	Informative References	26
	ACKNOWLEDGMENTS	26
	AUTHORS' ADDRESSES	27

1. Introduction

In situations where it is desirable to centrally manage authentication, authorization and accounting (AAA) for IEEE 802 [IEEE-802] networks, deployment of a backend authentication and accounting server is desirable. In such situations, it is expected that IEEE 802 authenticators will function as AAA clients.

"IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage Guidelines" [RFC3580] provides guidelines for the use of the Remote Authentication Dialin User Service (RADIUS) within networks utilizing IEEE 802 local area networks. This document defines additional attributes suitable for usage by IEEE 802 authenticators acting as AAA clients.

1.1. Terminology

This document uses the following terms:

Access Point (AP)

A Station that provides access to the distribution services via the wireless medium for associated Stations.

Association

The service used to establish Access Point/Station mapping and enable Station invocation of the distribution system services.

authenticator

An authenticator is an entity that require authentication from the supplicant. The authenticator may be connected to the supplicant at the other end of a point-to-point LAN segment or wireless link.

authentication server

An authentication server is an entity that provides an authentication service to an authenticator. This service verifies from the credentials provided by the supplicant, the claim of identity made by the supplicant.

Station (STA)

Any device that contains an IEEE 802.11 conformant medium access control (MAC) and physical layer (PHY) interface to the wireless medium (WM).

Supplicant

A supplicant is an entity that is being authenticated by an authenticator. The supplicant may be connected to the authenticator at one end of a point-to-point LAN segment or 802.11 wireless link.

1.2. 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 [RFC2119].

2. RADIUS attributes

2.1. Allowed-Called-Station-Id

Description

The Allowed-Called-Station-Id Attribute allows the RADIUS server to specify the authenticator MAC addresses and/or networks to which the user is allowed to connect. One or more Allowed-Called-Station-Id attributes MAY be included in an Access-Accept, CoA-Request or Accounting-Request packet.

The Allowed-Called-Station-Id Attribute can be useful in situations where pre-authentication is supported (e.g. IEEE 802.11 pre-authentication). In these scenarios, a Called-Station-Id Attribute typically will not be included within the Access-Request so that the RADIUS server will not know the network that the user is attempting to access. The Allowed-Called-Station-Id enables the RADIUS server to restrict the networks and attachment points to which the user can subsequently connect.

A summary of the Allowed-Called-Station-Id Attribute format is shown below. 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	2	3	4	5	6	7	8	9	0	1
Type										Length										String...																					

Code

TBD1

Length

 ≥ 3

String

Code

102 [RFC4072]

Length

>=3

String

The String field is one or more octets, containing the EAP Session-Id, as defined in "Extensible Authentication Protocol (EAP) Key Management Framework" [RFC5247]. Since the NAS operates as a pass-through in EAP, it cannot know the EAP Session-Id before receiving it from the RADIUS server. As a result, an EAP-Key-Name Attribute sent in an Access-Request MUST only contain a single NUL character. A RADIUS server receiving an Access-Request with an EAP-Key-Name Attribute containing anything other than a single NUL character MUST silently discard the Attribute. In addition, the RADIUS server SHOULD include this Attribute in an Access-Accept or CoA-Request only if an EAP-Key-Name Attribute was present in the Access-Request. Since a NAS will typically only include a EAP-Key-Name Attribute in an Access-Request in situations where the Attribute is required to provision service, if an EAP-Key-Name Attribute is included in an Access-Request but is not present in the Access-Accept, the NAS SHOULD treat the Access-Accept as though it were an Access-Reject. If an EAP-Key-Name Attribute was not present in the Access-Request but is included in the Access-Accept, then the NAS SHOULD silently discard the EAP-Key-Name Attribute. As noted in [IEEE-802.1X] Section 6.2.2, the Connectivity Association Key Name (CKN) is derived from the EAP Session-Id, and as described in Section 9.3.3, the CKN is subsequently used in the derivation of the Key Encrypting Key (KEK) and the Integrity Check Value Key (ICK), utilized to protect the secret keys (SAKs) utilized by Media Access Control Security (MACsec). As a result, for the NAS to acquire information needed in the MACsec Key Agreement (MKA) exchange, it needs to include the EAP-Key-Name attribute in the Access-Request and receive it from the RADIUS server in the Access-Accept.

2.3. EAP-Peer-Id

Description

The EAP-Peer-Id Attribute contains a Peer-Id generated by the EAP method. Exactly how this name is used depends on the link layer in question. See [RFC5247] for more discussion. The EAP-Peer-Id Attribute MAY be included in Access-Request, Access-Accept and

It should be noted that not all link layers use this name, and existing EAP method implementations do not generate it. Since the NAS operates as a pass-through in EAP [RFC3748], it cannot know the EAP-Peer-Id before receiving it from the RADIUS server. As a result, an EAP-Peer-Id Attribute sent in an Access-Request MUST only contain a single NUL character. A home RADIUS server receiving an Access-Request an EAP-Peer-Id Attribute containing anything other than a single NUL character MUST silently discard the Attribute. In addition, the home RADIUS server SHOULD include one or more EAP-Peer-Id attributes in an Access-Accept only if an EAP-Peer-Id Attribute was present in the Access-Request. If a NAS receives EAP-Peer-Id Attribute(s) in an Access-Accept without having included one in an Access-Request, the NAS SHOULD silently discard the Attribute(s). A summary of the EAP-Peer-Id Attribute format is shown below. 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										String...																			

String

The String field is one or more octets containing a EAP Peer-Id exported by the EAP method. For details, see [RFC5247] Appendix A. A robust implementation SHOULD support the field as undistinguished octets. Only a single EAP Peer-Id may be included per Attribute.

Description

The EAP-Server-Id Attribute contains a Server-Id generated by the

EAP method. Exactly how this name is used depends on the link layer in question. See [RFC5247] for more discussion. The EAP-Server-Id Attribute is only allowed in Access-Request, Access-Accept, and Accounting-Request packets. More than one EAP-Server-Id Attribute MUST NOT be included in an Access-Request; one or more EAP-Server-Id attributes MAY be included in an Access-Accept.

It should be noted that not all link layers use this name, and existing EAP method implementations do not generate it. Since the NAS operates as a pass-through in EAP [RFC3748], it cannot know the EAP-Server-Id before receiving it from the RADIUS server. As a result, an EAP-Server-Id Attribute sent in an Access-Request MUST contain only a single NUL character. A home RADIUS server receiving in an Access-Request an EAP-Server-Id Attribute containing anything other than a single NUL character MUST silently discard the Attribute. In addition, the home RADIUS server SHOULD include this Attribute in an Access-Accept only if an EAP-Server-Id Attribute was present in the Access-Request. A summary of the EAP-Server-Id Attribute format is shown below. The fields are transmitted from left to right.

[illegible]

Code

TBD3

Length

 ≥ 3

String

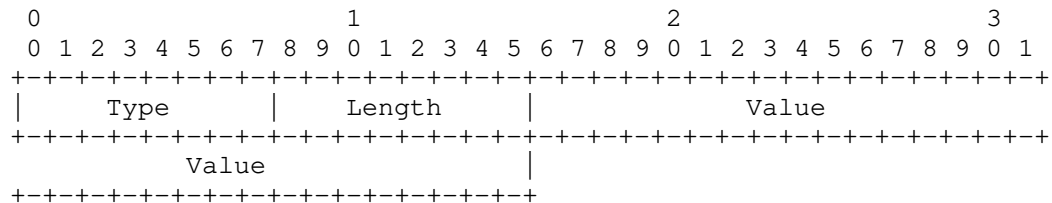
The String field is one or more octets, containing a EAP Server-Id exported by the EAP method. For details, see [RFC5247] Appendix A. A robust implementation SHOULD support the field as undistinguished octets.

2.5. Mobility-Domain-Id

Description

A single Mobility-Domain-Id Attribute MAY be included in an Access-Request or Accounting-Request, in order to enable the NAS

to provide the RADIUS server with the Mobility Domain Identifier (MDID), defined in Section 8.4.2.49 of [IEEE-802.11]. A summary of the Mobility-Domain-Id Attribute format is shown below. The fields are transmitted from left to right.



Code

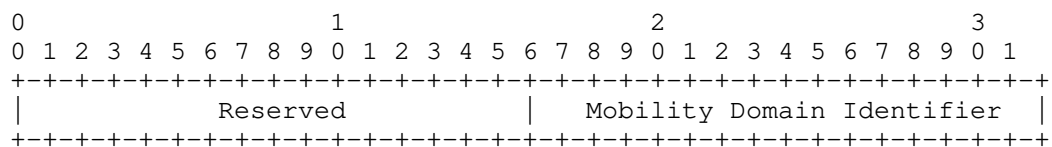
TBD4

Length

6

Value

The Value field is four octets, containing a 32-bit unsigned integer. The two most significant octets MUST be set to zero by the sender, and are ignored by the receiver; the two least significant octets contain the Mobility Domain Identifier (MDID) defined in Section 8.4.2.49 of [IEEE-802.11].



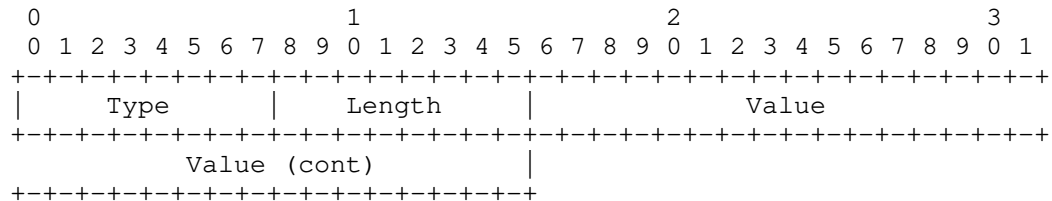
2.6. Preauth-Timeout

Description

This Attribute sets the maximum number of seconds which pre-authentication state is required to be kept by the NAS, without being utilized within a user session. For example, when [IEEE-802.11] pre-authentication is used, if a user has not attempted to utilize the Pairwise Master Key (PMK) derived as a result of pre-authentication within the time specified by the Preauth-Timeout Attribute, the PMK MAY be discarded by the Access Point. However, once the session is underway, the Preauth-Timeout Attribute has no bearing on the maximum session time for the user,

or the maximum time during which key state may be kept prior to re-authentication. This is determined by the Session-Timeout Attribute, if present.

A single Preauth-Timeout Attribute MAY be included within an Access-Accept or CoA-Request packet. A summary of the Preauth-Timeout Attribute format is shown below. The fields are transmitted from left to right.



Code

TBD5

Length

6

Value

The field is 4 octets, containing a 32-bit unsigned integer encoding the maximum time in seconds that pre-authentication state should be retained by the NAS.

2.7. Network-Id-Name

Description

The Network-Id-Name Attribute is utilized by implementations of IEEE-802.1X [IEEE-802.1X] to specify the name of a Network-Id (NID-Name).

Unlike the IEEE 802.11 SSID (which is a maximum of 32 octets in length), the NID-Name may be up to 253 octets in length. Consequently, if the MAC address is included within the Called-Station-Id Attribute, it is possible that there will not be enough remaining space to encode the NID-Name as well. Therefore when used with IEEE 802.1X [IEEE-802.1X], the Called-Station-Id Attribute SHOULD contain only the MAC address, with the Network-Id-Name Attribute used to transmit the NID-Name. The Network-Id-Name Attribute MUST NOT be used to encode the IEEE 802.11 SSID; as

noted in [RFC3580], the Called-Station-Id Attribute is used for this purpose.

Zero or one Network-Id-Name Attribute is permitted within an Access-Request, Access-Challenge, Access-Accept or Accounting-Request packet. When included within an Access-Request packet, the Network-Id-Name Attribute represents a hint of the NID-Name to which the Supplicant should be granted access. When included within an Access-Accept packet, the Network-Id-Name Attribute represents the NID-Name to which the Supplicant is to be granted access. When included within an Accounting-Request packet, the Network-Id-Name Attribute represents the NID-Name to which the Supplicant has been granted access.

A summary of the Network-Id-Name Attribute format is shown below. 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 |      String...      |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Code

TBD6

Length

>=3

String

The String field is one or more octets, containing a NID-Name. For details, see [IEEE-802.1X]. A robust implementation SHOULD support the field as undistinguished octets.

2.8. EAPoL-Announcement

Description

The Extensible Authentication Protocol over Local Area Network (EAPoL)-Announcement Attribute contains EAPoL-Announcement Type Length Value Tuples (TLVs) defined within Table 11-8 of IEEE-802.1X [IEEE-802.1X].

Zero or more EAPoL-Announcement attributes are permitted within an Access-Request, Access-Accept, Access-Challenge, Access-Reject,

Accounting-Request, CoA-Request or Disconnect-Request packet.

When included within an Access-Request packet, EAPoL-Announcement attributes contain EAPoL-Announcement TLVs that the user sent in an EAPoL-Announcement. When included within an Access-Accept, Access-Challenge, Access-Reject, CoA-Request or Disconnect-Request packet, EAPoL-Announcement attributes contain EAPoL-Announcement TLVs that the NAS is to send to the user in a unicast EAPoL-Announcement. When sent within an Accounting-Request packet, EAPoL-Announcement attributes contain EAPoL-Announcement TLVs that the NAS has most recently sent to the user in a unicast EAPoL-Announcement.

A summary of the EAPoL-Announcement Attribute format is shown below. 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      |      String...      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Code

TBD7

Length

>=3

String

The String field is one or more octets, containing EAPoL-Announcement TLVs in the format defined in Figure 11-8 of Section 11.12 of [IEEE-802.1X]. Any EAPoL-Announcement TLV Type MAY be included within an EAPoL-Announcement Attribute, including Organizationally Specific TLVs. If multiple EAPoL-Announcement attributes are present in a packet, their String fields MUST be concatenated before being parsed for EAPoL-Announcement TLVs; this allows EAPoL-Announcement TLVs longer than 253 octets to be transported by RADIUS. Similarly, EAPoL-Announcement TLVs larger than 253 octets MUST be fragmented between multiple EAPoL-Announcement attributes.

2.9. WLAN-HESSID

Description

The WLAN-HESSID attribute contains a MAC address that identifies the Homogenous Extended Service Set. The HESSID is a globally unique identifier that in conjunction with the SSID, encoded within the Called-Station-Id Attribute as described in [RFC3580], may be used to provide network identification for a subscription service provider network (SSPN), as described in Section 8.4.2.94 of [IEEE-802.11]. Zero or one WLAN-HESSID Attribute is permitted within an Access-Request or Accounting-Request packet.

A summary of the WLAN-HESSID Attribute format is shown below. 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      |      String...      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Code

TBD8

Length

19

String

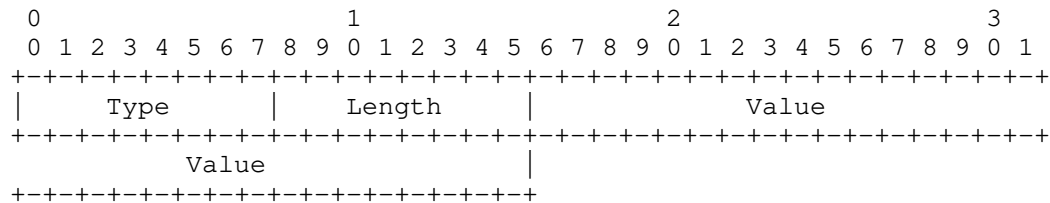
The String field is encoded in upper-case ASCII characters with the octet values separated by dash characters, as described in RFC 3580 [RFC3580]. Example: "00-10-A4-23-19-C0".

2.10. WLAN-Venue-Info

Description

The WLAN-Venue-Info attribute identifies the category of venue hosting the WLAN, as defined in Section 8.4.1.34 of [IEEE-802.11]. Zero or more WLAN-Venue-Info attributes may be included in an Access-Request or Accounting-Request.

A summary of the WLAN-Venue-Info Attribute format is shown below. The fields are transmitted from left to right.



Code

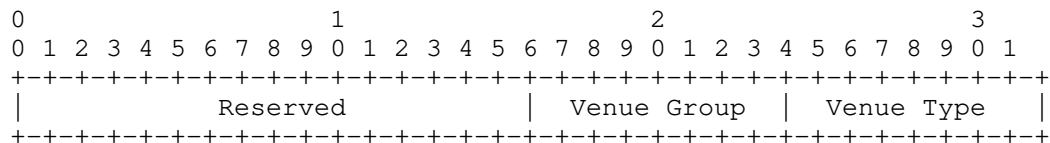
TBD9

Length

6

Value

The Value field is four octets, containing a 32-bit unsigned integer. The two most significant octets **MUST** be set to zero by the sender, and are ignored by the receiver; the two least significant octets contain the Venue Group and Venue Type fields.



Venue Group

The Venue Group field is a single octet and describes the broad category of the venue, e.g. "Assembly". See Section 8.4.1.34 [IEEE-802.11] for Venue Group codes and descriptions.

Venue Type

The Venue Type field is a single octet and describes the venue in a finer granularity within the Venue Group, e.g. "Library". See Section 8.4.1.34 of [IEEE-802.11] for Venue Type codes and descriptions.

2.11. WLAN-Venue-Language

Description

The WLAN-Venue-Language attribute is an ISO-14962-1997 [ISO-14962-1997] encoded string that defines the language used in

the WLAN-Venue-Name attribute. Zero or more WLAN-Venue-Language attributes may be included in an Access-Request or Accounting-Request and each one indicates the language of the WLAN-Venue-Name attribute that follows it.

A summary of the WLAN-Venue-Language Attribute format is shown below. 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      |      String...      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| String (cont)  |
+---+---+---+---+---+

```

Code

TBD10

Length

4-5

String

The String field is a two or three character language code selected from ISO-639 [ISO-639]. A two character language code has a zero ("null" in ISO-14962-1997) appended to make it 3 octets in length.

2.12. WLAN-Venue-Name

Description

The WLAN-Venue-Name attribute provides additional metadata on the BSS. For example, this information may be used to assist a user in selecting the appropriate BSS with which to associate. Zero or more WLAN-Venue-Name attributes may be included in an Access-Request or Accounting-Request in the same or different languages.

A summary of the WLAN-Venue-Name Attribute format is shown below. 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      |      String...      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Code

TBD11

Length

>=3

String

The String field is a UTF-8 formatted field containing the venue's name. The maximum length of this field is 252 octets.

2.13. WLAN-Reason-Code

Description

The WLAN-Reason-Code Attribute contains information on the reason why a station has been refused network access and has been disassociated or de-authenticated. This can occur due to policy or for reasons related to the user's subscription.

A WLAN-Reason-Code Attribute MAY be included within an Access-Reject or Disconnect-Request packet, as well as within an Accounting-Request packet. Upon receipt of an Access-Reject or Disconnect-Request packet containing a WLAN-Reason-Code Attribute, the WLAN-Reason-Code value is copied by the Access Point into the Reason Code field of a Disassociation or Deauthentication frame (see clause 8.3.3.4 and 8.3.3.12 respectively in [IEEE- 802.11]), which is subsequently transmitted to the station.

A summary of the WLAN-Reason-Code Attribute format is shown below. 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      |      Value      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|      Value      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Code

TBD12

Length

6

Value

The Value field is four octets, containing a 32-bit unsigned integer. The two most significant octets MUST be set to zero by the sender, and are ignored by the receiver; the two least significant octets contain the Reason Code values defined in Table 8-36 of Section 8.4.1.7 of [IEEE-802.11].

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										Reason Code																													

2.14. WLAN-Pairwise-Cipher

Description

The WLAN-Pairwise-Cipher Attribute contains information on the pairwise cipher suite used to establish the robust security network association (RSNA) between the AP and mobile device. A WLAN-Pairwise-Cipher Attribute MAY be included within Access-Request and Accounting-Request packets.

A summary of the WLAN-Pairwise-Cipher Attribute format is shown below. 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										Value																			
										Value																													

Code

TBD13

Length

6

Value

The Value field is four octets, containing a 32-bit unsigned integer, in Suite selector format as specified in Figure 8-187 within Section 8.4.2.27.2 of [IEEE-802.11], with values of OUI and Suite type drawn from Table 8-99.

```

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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               OUI                               | Suite Type |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

2.15. WLAN-Group-Cipher

Description

The WLAN-Group-Cipher Attribute contains information on the group cipher suite used to establish the robust security network association (RSNA) between the AP and mobile device. A WLAN-Group-Cipher Attribute MAY be included within Access-Request and Accounting-Request packets.

A summary of the WLAN-Group-Cipher Attribute format is shown below. 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 |                               Value                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Value                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Code

TBD14

Length

6

Value

The Value field is four octets, containing a 32-bit unsigned integer, in Suite selector format as specified in Figure 8-187

within Section 8.4.2.27.2 of [IEEE-802.11], with values of OUI and Suite type drawn from Table 8-99.

```

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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               OUI                               | Suite Type |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

2.16. WLAN-AKM-Suite

Description

The WLAN-AKM-Suite Attribute contains information on the authentication and key management suite used to establish the robust security network association (RSNA) between the AP and mobile device. A WLAN-AKM-Suite Attribute MAY be included within Access-Request and Accounting-Request packets.

A summary of the WLAN-AKM-Suite Attribute format is shown below. 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 |                               Value                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Value                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Code

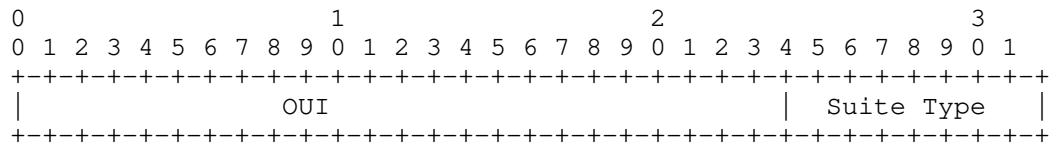
TBD15

Length

6

Value

The Value field is four octets, containing a 32-bit unsigned integer, in Suite selector format as specified in Figure 8-187 within Section 8.4.2.27.2 of [IEEE-802.11], with values of OUI and Suite type drawn from Table 8-101:



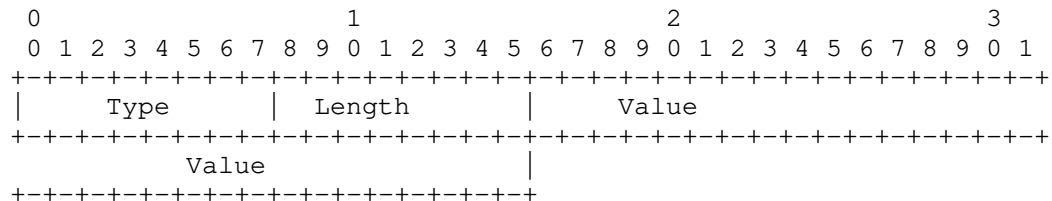
2.17. WLAN-Group-Mgmt-Cipher

Description

The WLAN-Group-Mgmt-Cipher Attribute contains information on group management cipher used to establish the robust security network association (RSNA) between the AP and mobile device.

Zero or one WLAN-Group-Mgmt-Cipher Attribute MAY be included within Access-Request and Accounting-Request packets. Presence of the attribute indicates that the station negotiated to use management frame protection during association.

A summary of the WLAN-Group-Mgmt-Cipher Attribute format is shown below. The fields are transmitted from left to right.



Code

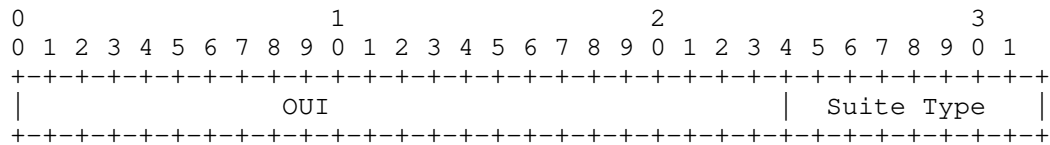
TBD16

Length

6

Value

The Value field is four octets, containing a 32-bit unsigned integer, in Suite selector format as specified in Figure 8-187 within Section 8.4.2.27.2 of [IEEE-802.11], with values of OUI and Suite type drawn from Table 8-99:

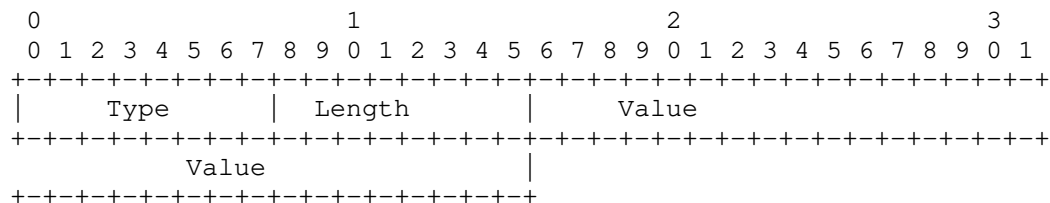


2.18. WLAN-RF-Band

Description

The WLAN-RF-Band Attribute contains information on the RF band used by the Access Point for transmission and reception of information to and from the mobile device. Zero or one WLAN-RF-Band Attribute MAY be included within an Access-Request or Accounting-Request packet.

A summary of the WLAN-RF-Band Attribute format is shown below. The fields are transmitted from left to right.



Code

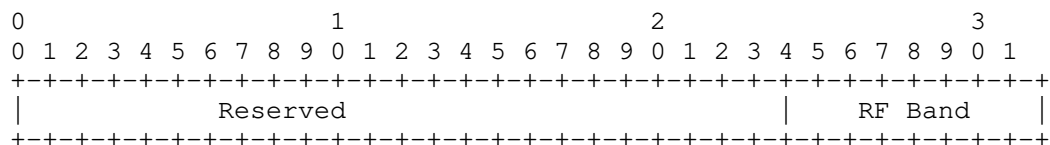
TBD17

Length

6

Value

The Value field is four octets, containing a 32-bit unsigned integer. The three most significant octets MUST be set to zero by the sender, and are ignored by the receiver; the least significant octet contains the RF Band field, whose values are defined in Table 8-53a of [IEEE-802.11ad].



3. Table of attributes

The following table provides a guide to which attributes may be found in which kinds of packets, and in what quantity.

Access-Request	Access-Accept	Access-Reject	Access-Challenge	#	Attribute
0	0+	0	0	TBD1	Allowed-Called-Station-Id
0-1	0-1	0	0	102	EAP-Key-Name
0-1	0+	0	0	TBD2	EAP-Peer-Id
0-1	0+	0	0	TBD3	EAP-Server-Id
0-1	0	0	0	TBD4	Mobility-Domain-Id
0-1	0-1	0	0	TBD5	Preauth-Timeout
0-1	0	0	0	TBD6	Network-Id-Name
0+	0+	0+	0+	TBD7	EAPoL-Announcement
0-1	0	0	0	TBD8	WLAN-HESSID
0-1	0	0	0	TBD9	WLAN-Venue-Info
0+	0	0	0	TBD10	WLAN-Venue-Language
0+	0	0	0	TBD11	WLAN-Venue-Name
0	0	0-1	0	TBD12	WLAN-Reason-Code
0-1	0	0	0	TBD13	WLAN-Pairwise-Cipher
0-1	0	0	0	TBD14	WLAN-Group-Cipher
0-1	0	0	0	TBD15	WLAN-AKM-Suite
0-1	0	0	0	TBD16	WLAN-Group-Mgmt-Cipher
0-1	0	0	0	TBD17	WLAN-RF-Band

CoA-Req	Dis-Req	Acct-Req	#	Attribute
0+	0	0+	TBD1	Allowed-Called-Station-Id
0-1	0	0	102	EAP-Key-Name
0	0	0+	TBD2	EAP-Peer-Id
0	0	0+	TBD3	EAP-Server-Id
0	0	0-1	TBD4	Mobility-Domain-Id
0-1	0	0	TBD5	Preauth-Timeout
0	0	0-1	TBD6	Network-Id-Name
0+	0+	0+	TBD7	EAPoL-Announcement
0	0	0-1	TBD8	WLAN-HESSID
0	0	0-1	TBD9	WLAN-Venue-Info
0	0	0+	TBD10	WLAN-Venue-Language
0	0	0+	TBD11	WLAN-Venue-Name
0	0-1	0-1	TBD12	WLAN-Reason-Code
0	0	0-1	TBD13	WLAN-Pairwise-Cipher
0	0	0-1	TBD14	WLAN-Group-Cipher
0	0	0-1	TBD15	WLAN-AKM-Suite
0	0	0-1	TBD16	WLAN-Group-Mgmt-Cipher
0	0	0-1	TBD17	WLAN-RF-Band

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 the packet.
- 0-1 Zero or one instance of this Attribute MAY be present in the packet.

4. IANA Considerations

This document uses the RADIUS [RFC2865] namespace, see <http://www.iana.org/assignments/radius-types>. This specification requires assignment of a RADIUS attribute types for the following attributes:

Attribute	Type
=====	=====
Allowed-Called-Station-Id	TBD1
EAP-Peer-Id	TBD2
EAP-Server-Id	TBD3
Mobility-Domain-Id	TBD4
Preauth-Timeout	TBD5
Network-Id-Name	TBD6
EAPoL-Announcement	TBD7
WLAN-HESSID	TBD8
WLAN-Venue-Info	TBD9
WLAN-Venue-Language	TBD10
WLAN-Venue-Name	TBD11
WLAN-Reason-Code	TBD12
WLAN-Pairwise-Cipher	TBD13
WLAN-Group-Cipher	TBD14
WLAN-AKM-Suite	TBD15
WLAN-Group-Mgmt-Cipher	TBD16
WLAN-RF-Band	TBD17

Since this specification relies entirely on values assigned by IEEE 802, no registries are established for maintenance by the IANA.

5. Security Considerations

Since this document describes the use of RADIUS for purposes of authentication, authorization, and accounting in IEEE 802 networks, it is vulnerable to all of the threats that are present in other RADIUS applications. For a discussion of these threats, see [RFC2607], [RFC2865], [RFC3162], [RFC3579], [RFC3580] and [RFC5176]. In particular, when RADIUS traffic is sent in the clear, the attributes defined in this document can be obtained by an attacker snooping the exchange between the RADIUS client and server. As a result, RADIUS confidentiality is desirable; for a review of RADIUS security and crypto-agility requirements, see [RFC6421].

While it is possible for a RADIUS server to make decisions on whether to Accept or Reject an Access-Request based on the values of the WLAN-Pairwise-Cipher, WLAN-Group-Cipher, WLAN-AKM-Suite, WLAN-Group-Mgmt-Cipher and WLAN-RF-Band Attributes the value of doing this is limited. In general, an Access-Reject should not be necessary, except where Access Points and Stations are misconfigured so as to enable connections to be made with unacceptable values. Rather than rejecting access on an ongoing basis, users would be better served by fixing the misconfiguration.

Where access does need to be rejected, the user should be provided with an indication of why the problem has occurred, or else they are likely to become frustrated. For example, if the values of the WLAN-Pairwise-Cipher, WLAN-Group-Cipher, WLAN-AKM-Suite or WLAN-Group-Mgmt-Cipher Attributes included in the Access-Request are not acceptable to the RADIUS server, then a WLAN-Reason-Code Attribute with a value of 29 (Requested service rejected because of service provider cipher suite or AKM requirement) SHOULD be returned in the Access-Reject. Similarly, if the value of the WLAN-RF-Band Attribute included in the Access-Request is not acceptable to the RADIUS server, then a WLAN-Reason-Code Attribute with a value of 11 (Disassociated because the information in the Supported Channels element is unacceptable) SHOULD be returned in the Access-Reject.

6. References

6.1. Normative references

[IEEE-802] IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture, ANSI/IEEE Std 802, 1990.

[IEEE-802.11]

Information technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std. 802.11-2012, 2012.

[IEEE-802.11ad]

Information technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, Amendment 3: Enhancements for Very High Throughput in the 60 GHz Band, IEEE Std. 802.11ad-2012, 2012.

[IEEE-802.1X]

IEEE Standard for Local and Metropolitan Area Networks -

Port-Based Network Access Control, IEEE 802.1X-2010, February 2010.

[ISO-639] ISO, "Codes for the Representation of Names of Languages".

[ISO-14962-1997]

ISO, "Space data and information transfer systems - ASCII encoded English", 1997.

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

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

[RFC4072] Eronen, P., Hiller, T. and G. Zorn, "Diameter Extensible Authentication Protocol (EAP) Application", RFC 4072, August 2005.

[RFC5247] Aboba, B., Simon, D. and P. Eronen, "EAP Key Management Framework", RFC 5247, August 2008.

6.2. Informative references

[RFC2607] Aboba, B. and J. Vollbrecht, "Proxy Chaining and Policy Implementation in Roaming", RFC 2607, June 1999.

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

[RFC3579] Aboba, B. and P. Calhoun, "RADIUS Support for Extensible Authentication Protocol (EAP)", RFC 3579, September 2003.

[RFC3580] Congdon, P., Aboba, B., Smith, A., Zorn, G. and J. Roese, "IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage Guidelines", RFC 3580, September 2003.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J. and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[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.

[RFC6421] Nelson, D., "Crypto-Agility Requirements for Remote Authentication Dial-In User Service (RADIUS)", RFC 6421,

November 2011.

Acknowledgments

The authors would like to acknowledge Maximilian Riegel, Dorothy Stanley, Yoshihiro Ohba, and the contributors to the IEEE 802.1 and IEEE 802.11 reviews of this document, for useful discussions.

Authors' Addresses

Bernard Aboba
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

EMail: bernard_aboba@hotmail.com

Jouni Malinen
EMail: j@w1.fi

Paul Congdon
Tallac Networks
6528 Lonetree Blvd.
Rocklin, CA 95765

Phone: +19167576350
EMail: paul.congdon@tallac.com

Joseph Salowey
Cisco Systems
EMail: jsalowey@cisco.com

Mark Jones
Azuca Systems
EMail: mark@azu.ca

RADIUS EXTensions Working Group
Internet-Draft
Intended status: Experimental
Expires: January 3, 2014

A. Perez-Mendez
R. Marin-Lopez
F. Pereniguez-Garcia
G. Lopez-Millan
University of Murcia
D. Lopez
Telefonica I+D
A. DeKok
Network RADIUS
July 2, 2013

Support of fragmentation of RADIUS packets
draft-perez-radext-radius-fragmentation-06

Abstract

The Remote Authentication Dial-In User Service (RADIUS) protocol is limited to a total packet size of 4096 octets. Provisions exist for fragmenting large amounts of authentication data across multiple packets, via Access-Challenge. No similar provisions exist for fragmenting large amounts of authorization data. This document specifies how existing RADIUS mechanisms can be leveraged to provide that functionality. These mechanisms are largely compatible with existing implementations, and are designed to be invisible to proxies, and "fail-safe" to legacy clients and servers.

Status of this Memo

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

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

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

This Internet-Draft will expire on January 3, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Requirements Language	4
2. Scope of this document	4
3. Overview	5
4. Fragmentation of packets	7
4.1. Pre-authorization	8
4.2. Post-authorization	12
5. Chunk size	14
6. Allowed large packet size	15
7. Handling special attributes	16
7.1. Proxy-State attribute	16
7.2. State attribute	17
7.3. Service-Type attribute	17
7.4. Rebuilding the original large packet	17
8. New attribute definition	18
8.1. Frag-Status attribute	18
8.2. Proxy-State-Len attribute	19
8.3. Table of attributes	20
9. Operation with proxies	20
9.1. Legacy proxies	20
9.2. Updated proxies	21
10. Security Considerations	22
11. IANA Considerations	23
12. References	23
12.1. Normative References	23
12.2. Informative References	24
Authors' Addresses	24

1. Introduction

The RADIUS [RFC2865] protocol carries authentication, authorization, and accounting information between a Network Access Server (NAS) and an Authentication Server (AS). Information is exchanged between the NAS and the AS through RADIUS packets. Each RADIUS packet is composed of a header, and zero or more attributes, up to a maximum packet size of 4096 octets. The protocol is a request/response protocol, as described in the operational model ([RFC6158], Section 3.1).

The above packet size limitation mean that peers desiring to send large amounts of data must fragment it across multiple packets. For example, RADIUS-EAP [RFC3579] defines how an EAP exchange occurs across multiple Access-Request / Access-Challenge sequences. No such exchange is possible for accounting or authorization data. [RFC6158] Section 3.1 suggests that exchanging large amounts authorization data is unnecessary in RADIUS. Instead, the data should be referenced by name. This requirement allows large policies to be pre-provisioned, and then referenced in an Access-Accept. In some cases, however, the authorization data sent by the server is large and highly dynamic. In other cases, the NAS needs to send large amounts of authorization data to the server. Both of these cases are un-met by the requirements in [RFC6158]. As noted in that document, the practical limit on RADIUS packet sizes is governed by the Path MTU (PMTU), which may be significantly smaller than 4096 octets. The combination of the two limitations means that there is a pressing need for a method to send large amounts of authorization data between NAS and AS, with no accompanying solution.

[RFC6158] recommends three approaches for the transmission of large amount of data within RADIUS. However, they are not applicable to the problem statement of this document for the following reasons:

- o The first approach does not talk about large amounts of data sent from the NAS to a server. Leveraging EAP (request/challenge) to send the data is not feasible, as EAP already fills packet to PMTU, and not all authentications use EAP. Moreover, as noted for NAS-Filter-Rule ([RFC4849]), this approach does entirely solve the problem of sending large amounts of data from a server to a NAS.
- o The second approach is not usable either, as using names rather than values is difficult when the nature of the data to be sent is highly dynamic (e.g. SAML sentences or NAS-Filter-Rule attributes). URLs could be used as a pointer to the location of the actual data, but their use would require them to be (a) dynamically created and modified, (b) securely accessed and (c) accessible from remote systems. Satisfying these constraints

would require the modification of several networking systems (e.g. firewalls and web servers). Furthermore, the set up of an additional trust infrastructure (e.g. PKI) would be required to allow secure retrieving of the information from the web server.

- o PMTU discovery does not solve the problem, as it does not allow to send data larger than the minimum of (PMTU or 4096) octets.

This document provides a mechanism to allow RADIUS peers to exchange large amounts of authorization data exceeding the 4096 octet limit, by fragmenting it across several client/server exchanges. The proposed solution does not impose any additional requirements to the RADIUS system administrators (e.g. need to modify firewall rules, set up web servers, configure routers, or modify any application server). It maintains compatibility with intra-packet fragmentation mechanisms (like those defined in [RFC3579] or in [I-D.ietf-radext-radius-extensions]). It is also transparent to existing RADIUS proxies, which do not implement this specification. The only systems needing to implement the draft are the ones which either generate, or consume the fragmented data being transmitted. Intermediate proxies just pass the packets without changes. Nevertheless, if a proxy supports this specification, it MAY re-assemble the data in order to either examine and/or modify it.

1.1. 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].

2. Scope of this document

This specification describes how a RADIUS client and a RADIUS server can exchange large amounts of data exceeding the 4096 octet limit. Specifically, its scope is limited to the exchange of authorization data, as other exchanges do not require of such a mechanism. In particular, authentication exchanges have already been defined to overcome this limitation (e.g. RADIUS-EAP). Moreover, as they represent the most critical part of a RADIUS conversation, its preferable to not introduce any modification to their operation that may affect existing equipment.

There is no need to fragment accounting packets either. While the accounting process can send large amounts of data, that data is typically composed of many small updates. That is, there is no demonstrated need to send indivisible blocks of more than 4K of data. The need to send large amounts of data per user session often

originates from the need for flow-based accounting. In this use-case, the client may send accounting data for many thousands of flows, where all those flows are tied to one user session. The existing Acct-Multi-Session-Id attribute defined in [RFC2866] Section 5.11 has been proven to work here.

Similarly, there is no need to fragment CoA packets. Instead, the CoA client MUST send a CoA-Request packet containing session identification attributes, along with Service-Type = Additional-Authorization, and a State attribute. Implementations not supporting fragmentation will respond with a CoA-NAK, and an Error-Cause of Unsupported-Service.

Implementations supporting this specification may not be able to change authorization data for a particular session. In that case, they MUST respond with a CoA-NAK, as above. Otherwise, the implementation MUST start fragmentation via Access-Request, using the methods defined here.

The above requirement solves a number of issues. It clearly separates session identification from authorization. Without this separation, it is difficult to both identify a session, and change its authorization using the same attribute. It also ensures that the authorization process is the same for initial authentication, and for CoA.

When a sessions authorization is changed, the CoA server MUST continue the existing service until the new authorization parameters are applied. The change of service SHOULD be done atomically. If the CoA server is unable to apply the new authorization, it MUST terminate the user session.

3. Overview

Authorization exchanges can occur either before or after end user authentication has been completed. An authorization exchange before authentication allows a RADIUS client to provide the RADIUS server with information that MAY modify how the authentication process will be performed (e.g. it MAY affect the selection of the EAP method). An authorization exchange after authentication allows the RADIUS server to provide the RADIUS client with information about the end user, the results of the authentication process and/or obligations to be enforced. In this specification we refer to the "pre-authentication" as the exchange of authorization information before the end user authentication has started, while the term "post-authentication" is used to refer to an authorization exchange happening after this authentication process.

In this specification we refer to the "size limit" as the practical limit on RADIUS packet sizes. This limit is the minimum of 4096 octets, and the current PMTU. We define below a method which uses Access-Request and Access-Accept in order to exchange fragmented data. The NAS and server exchange a series of Access-Request / Access-Accept packets, until such time as all of the fragmented data has been transported. Each packet contains a Frag-Status attribute which lets the other party know if fragmentation is desired, ongoing, or finished. Each packet may also contain the fragmented data, or instead be an "ACK" to a previous fragment from the other party. Each Access-Request contains a User-Name attribute, allowing it to be proxied if necessary. Each Access-Request may also contain a State attribute, which serves to tie it to a previous Access-Accept. Each Access-Accept contains a State attribute, for use by the NAS in a later Access-Request. Each Access-Accept contains a Service-Type indicating that the service being provided is fragmentation, and that the Access-Accept should not be interpreted as providing network access to the end user.

When a RADIUS client or server need to send data that exceeds the size limit, the mechanism proposed in this document is used. Instead of encoding one large RADIUS packet, a series of smaller RADIUS packets of the same type are encoded. Each smaller packet is called a "chunk" in this specification, in order to distinguish it from traditional RADIUS packets. The encoding process is a simple linear walk over the attributes to be encoded. This walk preserves the order of the attributes, as required by [RFC2865]. The number of attributes encoded in a particular chunk depends on the size limit, the size of each attribute, the number of proxies between client and server, and the overhead for fragmentation signalling attributes. Specific details are given in Section 5. A new attribute called Frag-Status (Section 8.1) signals the fragmentation status.

After the first chunk is encoded, it is sent to the other party. The packet is identified as a chunk via the Frag-Status attribute. The other party then requests additional chunks, again using the Frag-Status attribute. This process is repeated until all the attributes have been sent from one party to the other. When all the chunks have been received, the original list of attributes is reconstructed and processed as if it had been received in one packet.

When multiple chunks are sent, a special situation may occur for Extended Type attributes as defined in [I-D.ietf-radext-radius-extensions]. The fragmentation process may split a fragmented attribute across two or more chunks, which is not permitted by that specification. We address this issue by defining a new field in the Reserved field of the "Long Extended Type" attribute format. 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 flags "M" and "T" indicates that the attribute is fragmented (flag M), but that all the fragments are not available in this chunk (flag T).

This last situation is expected to be the most common occurrence in chunks. Typically, packet fragmentation will occur as a consequence of a desire to send one or more large (and therefore fragmented) attributes. The large attribute will likely be split into two or more pieces. Where chunking does not split a fragmented attribute, no special treatment is necessary.

The setting of the "T" flag is the only case where the chunking process affects the content of an attribute. Even then, the "Value" fields of all attributes remain unchanged. Any per-packet security attributes such as Message-Authenticator are calculated for each chunk independently. There are neither integrity nor security checks performed on the "original" packet.

Each RADIUS packet sent or received as part of the chunking process MUST be a valid packet, subject to all format and security requirements. This requirement ensures that a "transparent" proxy not implementing this specification can receive and send compliant packets. That is, a proxy which simply forwards packets without detailed examination or any modification will be able to proxy "chunks".

4. Fragmentation of packets

When the NAS or the AS desires to send a packet that exceeds the size limit, it is split into chunks and sent via multiple client/server exchanges. The exchange is indicated via the Frag-Status attribute, which has value More-Data-Pending for all but the last chunk of the series. The chunks are tied together via the State attribute.

The following sections describe how to perform fragmentation for packets from the NAS to the server, followed by packets from the server to the NAS. We give the packet type, along with a RADIUS Identifier, to indicate that requests and responses are connected. We then give a list of attributes. We do not give values for most attributes, as we wish to concentrate on the fragmentation behaviour, rather than packet contents. Attribute values are given for attributes relevant to the fragmentation process. Where "long extended" attributes are used, we indicate the M (More) and T (Truncation) flags as optional square brackets after the attribute name. As no "long extended" attributes have yet been defined, we use

example attributes, named as "Example-Long-1", etc. The maximum chunk size is established in term of number of attributes (11), for sake of simplicity.

4.1. Pre-authorization

When the client needs to send a large amount of data to the server, the data to be sent is split into chunks and sent to the server via multiple Access-Request / Access-Accept exchanges. The example below shows this exchange.

The following is an Access-Request which the NAS intends to send to a server. However, due to a combination of issues (PMTU, large attributes, etc.), the content does not fit into one Access-Request packet.

```
Access-Request
  User-Name
  User-Password
  Calling-Station-Id
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1
  Example-Long-2 [M]
  Example-Long-2 [M]
  Example-Long-2
```

Figure 1: Desired Access-Request

The NAS therefore must send the attributes listed above in a series of chunks. The first chunk contains eight (8) attributes from the original Access-Request, and a Frag-Status attribute. Since last attribute is "Example-Long-1" with the "M" flag set, the chunking process also sets the "T" flag in that attribute. The Access-Request is sent with a RADIUS Identifier field having value 23. The Frag-Status attribute has value More-Data-Pending, to indicate that the NAS wishes to send more data in a subsequent Access-Request. The NAS also adds a Service-Type attribute, which indicates that it is part of the chunking process. The packet is signed with the Message-Authenticator attribute, completing the maximum number of attributes (11).


```
Access-Request (ID = 23)
  User-Name
  User-Password
  Calling-Station-Id
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [MT]
  Frag-Status = More-Data-Pending
  Service-Type = Additional-Authorization
  Message-Authenticator
```

Figure 2: Access-Request (chunk 1)

Compliant servers receiving this packet will see the Frag-Status attribute, and suspend all authorization and authentication handling until all of the chunks have been received. Non-compliant servers should also see the Service-Type requesting provisioning for an unknown service, and return Access-Reject. Other non-compliant servers may return an Access-Reject, Access-Challenge, or an Access-Accept with a particular Service-Type. Compliant NAS implementations MUST treat these responses as if they had received Access-Reject instead.

Compliant servers who wish to receive all of the chunks will respond with the following packet. The value of the State here is arbitrary, and serves only as a unique token for example purposes. We only note that it MUST be globally and temporally unique.

```
Access-Accept (ID = 23)
  Frag-Status = More-Data-Request
  Service-Type = Additional-Authorization
  State = 0xabc00001
  Message-Authenticator
```

Figure 3: Access-Accept (chunk 1)

The NAS will see this response, and use the RADIUS Identifier field to associate it with an ongoing chunking session. Compliant NASes will then continue the chunking process. Non-compliant NASes will never see a response such as this, as they will never send a Frag-Status attribute. The Service-Type attribute is included in the Access-Accept in order to signal that the response is part of the chunking process. This packet therefore does not provision any network service for the end user.

The NAS continues the process by sending the next chunk, which

includes an additional six (6) attributes from the original packet. It again includes the User-Name attribute, so that non-compliant proxies can process the packet. It sets the Frag-Status attribute to More-Data-Pending, as more data is pending. It includes a Service-Type for reasons described above. It includes the State attribute from the previous Access-accept. It signs the packet with Message-Authenticator, as there are no authentication attributes in the packet. It uses a new RADIUS Identifier field.

```
Access-Request (ID = 181)
  User-Name
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1
  Example-Long-2 [M]
  Example-Long-2 [MT]
  Frag-Status = More-Data-Request
  Service-Type = Additional-Authorization
  State = 0xabc000001
  Message-Authenticator
```

Figure 4: Access-Request (chunk 2)

Compliant servers receiving this packet will see the Frag-Status attribute, and look for a State attribute. Since one exists and it matches a State sent in an Access-Accept, this packet is part of a chunking process. The server will associate the attributes with the previous chunk. Since the Frag-Status attribute has value More-Data-Request, the server will respond with an Access-Accept as before. It MUST include a State attribute, with a value different from the previous Access-Accept. This State MUST again be globally and temporally unique.

```
Access-Accept (ID = 181)
  Frag-Status = More-Data-Request
  Service-Type = Additional-Authorization
  State = 0xdef00002
  Message-Authenticator
```

Figure 5: Access-Accept (chunk 2)

The NAS will see this response, and use the RADIUS Identifier field to associate it with an ongoing chunking session. The NAS continues the chunking process by sending the next chunk, with the final attribute(s) from the original packet, and again includes the original User-Name attribute. The Frag-Status attribute is not included in the next Access-Request, as no more chunks are available

for sending. The NAS includes the State attribute from the previous Access-accept. It signs the packet with Message-Authenticator, as there are no authentication attributes in the packet. It again uses a new RADIUS Identifier field.

```
Access-Request (ID = 241)
  User-Name
  Example-Long-2
  State = 0xdef00002
  Message-Authenticator
```

Figure 6: Access-Request (chunk 3)

On reception of this last chunk, the server matches it with an ongoing session via the State attribute, and sees that there is no Frag-Status attribute present. It then process the received attributes as if they had been sent in one RADIUS packet. See Section 7.4 for further details of this process. It generates the appropriate response, which can be either Access-Accept or Access-Reject. In this example, we show an Access-Accept. The server MUST send a State attribute, which permits link the received data with the authentication process.

```
Access-Accept (ID = 241)
  State = 0x98700003
  Message-Authenticator
```

Figure 7: Access-Accept (chunk 3)

The above example shows in practice how the chunking process works. We re-iterate the implementation and security requirements here.

Each chunk is a valid RADIUS packet, and all RADIUS format and security requirements MUST be followed before any chunking process is applied.

Every chunk except for the last one from a NAS MUST include a Frag-Status attribute, with value More-Data-Pending. The last chunk MUST NOT contain a Frag-Status attribute. Each chunk except for the last from a NAS MUST include a Service-Type attribute, with value Additional-Authorization. Each chunk MUST include a User-Name attribute, which MUST be identical in all chunks. Each chunk except for the first one from a NAS MUST include a State attribute, which MUST be copied from a previous Access-Accept.

Each Access-Accept MUST include a State attribute. The value for this attribute MUST change in every new Access-Accept, and MUST be globally and temporally unique.

4.2. Post-authorization

When the AS wants to send a large amount of authorization data to the NAS after authentication, the operation is very similar to the pre-authorization one. The presence of Service-Type = Additional-Authorization attribute ensures that a NAS not supporting this specification will treat that unrecognized Service-Type as though an Access-Reject had been received instead ([RFC2865] Section 5.6). If the original large Access-Accept packet contained a Service-Type attribute, it will be included with its original value in the last transmitted chunk, to avoid confusion with the one used for fragmentation signalling.

Client supporting this specification MUST include a Frag-Status = Fragmentation-Supported attribute in the first Access-Request sent to the server, in order to indicate they would accept fragmented data from the sever. This is not required if pre-authorization process was carried out, as it is implicit.

The following is an Access-Accept which the AS intends to send to a client. However, due to a combination of issues (PMTU, large attributes, etc.), the content does not fit into one Access-Accept packet.

```
Access-Accept
  User-Name
  EAP-Message
  Service-Type(Login)
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1
  Example-Long-2 [M]
  Example-Long-2 [M]
  Example-Long-2
```

Figure 8: Desired Access-Accept

The AS therefore must send the attributes listed above in a series of chunks. The first chunk contains eight (7) attributes from the original Access-Accept, and a Frag-Status attribute. Since last attribute is "Example-Long-1" with the "M" flag set, the chunking process also sets the "T" flag in that attribute. The Access-Accept

is sent with a RADIUS Identifier field having value 30 corresponding to a previous Access-Request not depicted. The Frag-Status attribute has value More-Data-Pending, to indicate that the AS wishes to send more data in a subsequent Access-Accept. The AS also adds a Service-Type attribute with value Additional-Authorization, which indicates that it is part of the chunking process. Note that the original Service-Type is not included in this chunk. Finally, a State attribute is included to allow matching subsequent requests with this conversation, and the packet is signed with the Message-Authenticator attribute, completing the maximum number of attributes of 11.

```
Access-Accept (ID = 30)
  User-Name
  EAP-Message
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [MT]
  Frag-Status = More-Data-Pending
  Service-Type = Additional-Authorization
  State = 0xcba00004
  Message-Authenticator
```

Figure 9: Access-Accept (chunk 1)

Compliant clients receiving this packet will see the Frag-Status attribute, and suspend all authorization and authentication handling until all of the chunks have been received. Non-compliant clients should also see the Service-Type indicating the provisioning for an unknown service, and will treat it as an Access-Reject.

Clients who wish to receive all of the chunks will respond with the following packet, where the value of the State attribute is taken from the received Access-Accept. They also include the User-Name attribute so that non-compliant proxies can process the packet.

```
Access-Request (ID = 131)
  User-Name
  Frag-Status = More-Data-Request
  Service-Type = Additional-Authorization
  State = 0xcba00004
  Message-Authenticator
```

Figure 10: Access-Request (chunk 1)

The AS receives this request, and uses the State attribute to associate it with an ongoing chunking session. Compliant ASes will

then continue the chunking process. Non-compliant ASes will never see a response such as this, as they will never send a Frag-Status attribute.

The AS continues the chunking process by sending the next chunk, with the final attribute(s) from the original packet. The value of the Identifier field is taken from the received Access-Request. A Frag-Status attribute is not included in the next Access-Accept, as no more chunks are available for sending. The AS includes an State attribute to allow the client to send additional authorization data. The original Service-Type attribute is included in this final chunk.

```
Access-Accept (ID = 131)
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1
  Example-Long-2 [M]
  Example-Long-2 [M]
  Example-Long-2
  Service-Type = Login
  State = 0xfda000005
  Message-Authenticator
```

Figure 11: Access-Accept (chunk 2)

On reception of this last chunk, the client matches it with an ongoing session via the Identifier field, and sees that there is no Frag-Status attribute present. It then processes the received attributes as if they had been sent in one RADIUS packet. See Section 7.4 for further details of this process.

5. Chunk size

In an ideal scenario, each intermediate chunk would be exactly the size limit in length. In this way, the number of round trips required to send a large packet would be optimal. However, this is not possible for several reasons.

1. RADIUS attributes have a variable length, and must be included completely in a chunk. Thus, it is possible that, even if there is some free space in the chunk, it is not enough to include the next attribute. This can generate up to 254 octets of spare space on every chunk.
2. RADIUS fragmentation requires the introduction of some extra attributes for signalling. Specifically, a Frag-Status attribute

(7 octets) is included on every chunk of a packet, except the last one. A RADIUS State attribute (from 3 to 255 octets) is also included in most chunks, to allow the server to bind an Access-Request with a previous Access-Challenge. User-Name attributes (from 3 to 255 octets) are introduced on every chunk the client sends as they are required by the proxies to route the packet to its destination. Together, these attributes can generate from up to 13 to 517 octets of signalling data, reducing the amount of payload information that can be sent on each chunk.

3. RADIUS packets SHOULD be adjusted to avoid exceeding the network MTU. Otherwise, IP fragmentation may occur, having undesirable consequences. Hence, maximum chunk size would be decreased from 4096 to the actual MTU of the network.
4. The inclusion of Proxy-State attributes by intermediary proxies can decrease the availability of usable space into the chunk. This is described with further detail in Section 7.1.

6. Allowed large packet size

There are no provisions for signalling how much data is to be sent via the fragmentation process as a whole. It is difficult to define what is meant by the "length" of any fragmented data. That data can be multiple attributes, which includes RADIUS attribute header fields. Or it can be one or more "large" attributes (more than 256 octets in length). Proxies can also filter these attributes, to modify, add, or delete them and their contents. These proxies act on a "packet by packet" basis, and cannot know what kind of filtering actions they take on future packets. As a result, it is impossible to signal any meaningful value for the total amount of additional data.

Unauthenticated clients are permitted to trigger the exchange of large amounts of fragmented data between the NAS and the AS, having the potential to allow Denial of Service (DoS) attacks. An attacker could initiate a large number of connections, each of which requests the server to store a large amount of data. This data could cause memory exhaustion on the server, and result in authentic users being denied access. It is worth noting that authentication mechanisms are already designed to avoid exceeding the size limit.

Hence, implementations of this specification MUST limit the total amount of data they send and/or receive via this specification. It is RECOMMENDED that the limits be set to a few tens of kilooctets. Any more than this may turn RADIUS into a generic transport protocol, which is undesired. It is RECOMMENDED that this limit be exposed to

administrators, so that it can be changed if necessary.

Implementations of this specification MUST limit the total number of round trips used during the fragmentation process. It is RECOMMENDED that the number of round trips be limited to twenty (20). Any more than this may indicate an implementation error, misconfiguration, or a denial of service (DoS) attack. It is RECOMMENDED that this limit be exposed to administrators, so that it can be changed if necessary.

7. Handling special attributes

7.1. Proxy-State attribute

RADIUS proxies may introduce Proxy-State attributes into any Access-Request packet they forward. Should they cannot add this information to the packet, they may silently discard forwarding it to its destination, leading to DoS situations. Moreover, any Proxy-State attribute received by a RADIUS server in an Access-Request packet MUST be copied into the reply packet to it. For these reasons, Proxy-State attributes require a special treatment within the packet fragmentation mechanism.

When the RADIUS server replies to an Access-Request packet as part of a conversation involving a fragmentation (either a chunk or a request for chunks), it MUST include every Proxy-State attribute received into the reply packet. This means that the server MUST take into account the size of these Proxy-State attributes in order to calculate the size of the next chunk to be sent.

However, while a RADIUS server will always know how many space MUST be left on each reply packet for Proxy-State attributes (as they are directly included by the RADIUS server), a RADIUS client cannot know this information, as Proxy-State attributes are removed from the reply packet by their respective proxies before forwarding them back. Hence, clients need a mechanism to discover the amount of space required by proxies to introduce their Proxy-State attributes. In the following we describe a new mechanism to perform such a discovery:

1. When a RADIUS client does not know how many space will be required by intermediate proxies for including their Proxy-State attributes, it SHOULD start using a conservative value (e.g. 1024 octets) as the chunk size.
2. When the RADIUS server receives a chunk from the client, it can calculate the total size of the Proxy-State attributes that have been introduced by intermediary proxies along the path. This

information MUST be returned to the client in the next reply packet, encoded into a new attribute called Proxy-State-Len.

3. The RADIUS client reacts upon the reception of this attribute by adjusting the maximum size for the next chunk accordingly.

7.2. State attribute

This RADIUS fragmentation mechanism makes use of the State attribute to link all the chunks belonging to the same fragmented packet. However, some considerations are required when the RADIUS server is fragmenting a packet that already contains a State attribute for other purposes not related with the fragmentation. If the procedure described in Section 4 is followed, two different State attributes could be included into a single chunk, incurring into two problems. First, [RFC2865] explicitly forbids that more than one State attribute appears into a single packet.

A straightforward solution consists on making the RADIUS server to send the original State attribute into the last chunk of the sequence (attributes can be re-ordered as specified in [RFC2865]). As the last chunk (when generated by the RADIUS server) does not contain any State attribute due to the fragmentation mechanism, both situations described above are avoided.

Something similar happens when the RADIUS client has to send a fragmented packet that contains a State attribute on it. The client MUST assure that this original State is included into the first chunk sent to the server (as this one never contains any State attribute due to fragmentation).

7.3. Service-Type attribute

This RADIUS fragmentation mechanism makes use of the Service-Type attribute to indicate an Access-Accept packet is not granting access to the service yet, since additional authorization exchange needs to be performed. Similarly to the State attribute, the RADIUS server has to send the original Service-Type attribute into the last Access-Accept of the RADIUS conversation to avoid ambiguity.

7.4. Rebuilding the original large packet

The RADIUS client stores the RADIUS attributes received on each chunk in order to be able to rebuild the original large packet after receiving the last chunk. However, some of these received attributes MUST NOT be stored in this list, as they have been introduced as part of the fragmentation signalling and hence, they are not part of the original packet.

- o State (except the one in the last chunk, if present)
- o Service-Type = Additional-Authorization
- o Frag-Status
- o Proxy-State-Len

Similarly, the RADIUS server MUST NOT store the following attributes as part of the original large packet:

- o State (except the one in the first chunk, if present)
- o Frag-Status
- o Proxy-State (except the ones in the last chunk)
- o User-Name (except the one in the first chunk)

8. New attribute definition

This document proposes the definition of two new extended type attributes, called Frag-Status and Proxy-State-Len. The format of these attributes follows the indications for an Extended Type attribute defined in [I-D.ietf-radext-radius-extensions].

8.1. Frag-Status attribute

This attribute is used for fragmentation signalling, and its meaning depends on the code value transported within it. The following figure represents the format of the Frag-Status attribute.

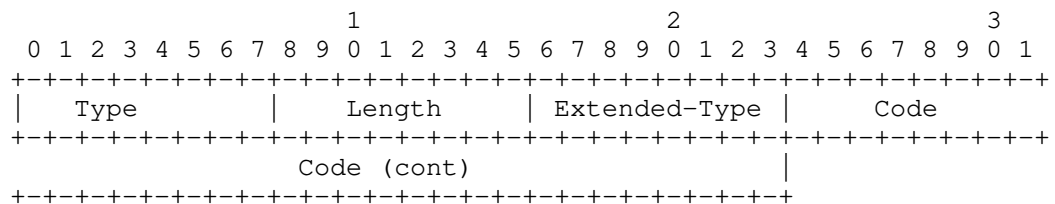


Figure 12: Frag-Status format

Type

To be assigned (TBA)

Length

7

Extended-Type

To be assigned (TBA).

Code

4 byte. Integer indicating the code. The values defined in this specifications are:

- 0 - Reserved
- 1 - Fragmentation-Supported
- 2 - More-Data-Pending
- 3 - More-Data-Request

This attribute MAY be present in Access-Request, Access-Challenge and Access-Accept packets. It MUST not be included in Access-Reject packets.

8.2. Proxy-State-Len attribute

This attribute indicates to the RADIUS client the length of the Proxy-State attributes received by the RADIUS server. This information is useful to adjust the length of the chunks sent by the RADIUS client. The format of this Proxy-State-Len attribute is the following:

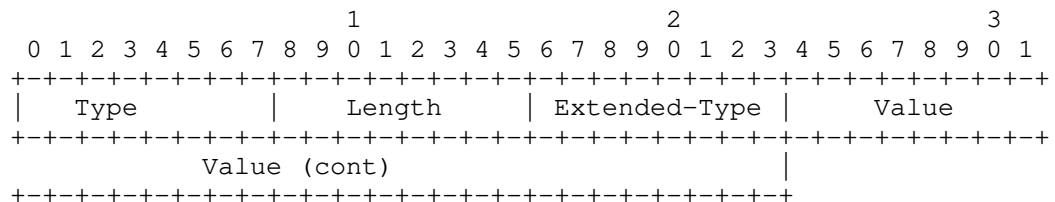


Figure 13: Proxy-State-Len format

Type

To be assigned (TBA)

Length

7

Extended-Type

To be assigned (TBA).

Value

4 octets. Total length (in octets) of received Proxy-State attributes (including headers).

This attribute MAY be present in Access-Challenge and Access-Accept packets. It MUST not be included in Access-Request or Access-Reject packets.

8.3. Table of attributes

The following table shows the different attributes defined in this document related with the kind of RADIUS packets where they can be present.

Attribute Name	Kind of packet			
	Req	Acc	Rej	Cha
Frag-Status	0-1	0-1	0	0-1
Proxy-State-Len	0	0-1	0	0-1

Figure 14

9. Operation with proxies

The fragmentation mechanism defined above is designed to be transparent to legacy proxies, as long as they do not want to modify any fragmented attribute. Nevertheless, updated proxies supporting this specification can even modify fragmented attributes.

9.1. Legacy proxies

As every chunk is indeed a RADIUS packet, legacy proxies treat them as the rest of packets, routing them to their destination. Proxies can introduce Proxy-State attributes to Access-Request packets, even if they are indeed chunks. This will not affect how fragmentation is managed. The server will include all the received Proxy-State attributes into the generated response, as described in [RFC2865].

Hence, proxies do not distinguish between a regular RADIUS packet and a chunk.

9.2. Updated proxies

Updated proxies can interact with clients and servers in order to obtain the complete large packet before start forwarding it. In this way, proxies can manipulate (modify and/or remove) any attribute of the packet, or introduce new attributes, without worrying about crossing the boundaries of the chunk size. Once the manipulated packet is ready, it is sent to the original destination using the fragmentation mechanism (if required). The following example shows how an updated proxy interacts with the NAS to obtain a large Access-Request packet, modify an attribute resulting into a even more large packet, and interacts with the AS to complete the transmission of the modified packet.

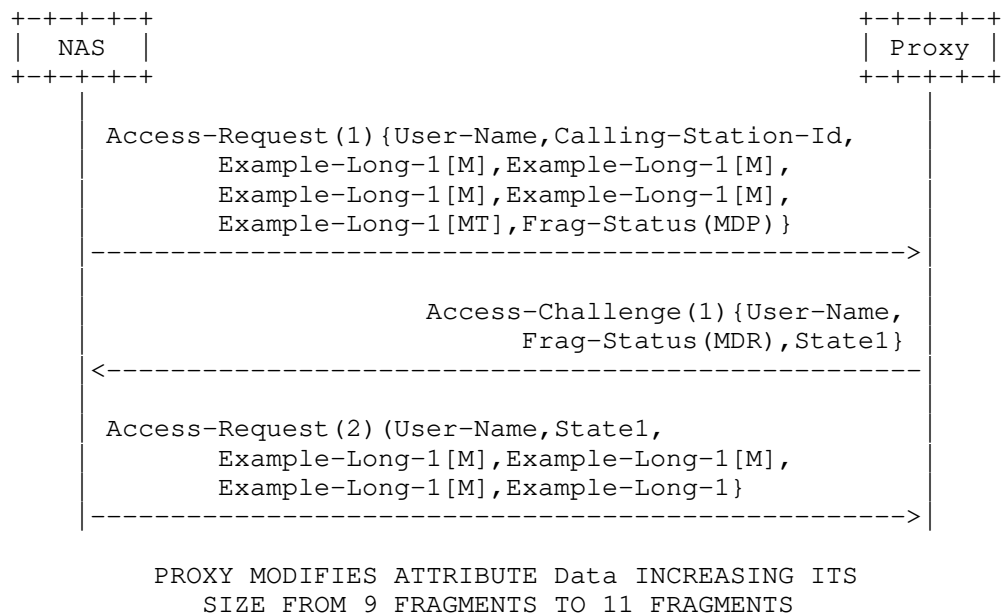


Figure 15: Updated proxy interacts with NAS



Figure 16: Updated proxy interacts with AS

10. Security Considerations

As noted in many earlier specifications ([RFC5080], [RFC6158], etc.) RADIUS security is problematic. This specification changes nothing related to the security of the RADIUS protocol. It requires that all Access-Request packets associated with fragmentation are signed using the existing Message-Authenticator attribute. This signature prevents forging and replay, to the limits of the existing security.

The ability to send bulk data from one party to another creates new security considerations. Clients and servers may have to store large amounts of data per session. The amount of this data can be significant, leading to the potential for resource exhaustion. We therefore suggest that implementations limit the amount of bulk data stored per session. The exact method for this limitation is implementation-specific. Section 6 gives some indications on what could be a reasonable limits.

The bulk data can often be pushed off to storage methods other than the memory of the RADIUS implementation. For example, it can be stored in an external database, or in files. This approach mitigates the resource exhaustion issue, as servers today already store large amounts of accounting data.

11. IANA Considerations

The authors request that Attribute Types and Attribute Values defined in this document be registered by the Internet Assigned Numbers Authority (IANA) from the RADIUS namespaces as described in the "IANA Considerations" section of [RFC3575], in accordance with BCP 26 [RFC5226]. For RADIUS packets, attributes and registries created by this document IANA is requested to place them at <http://www.iana.org/assignments/radius-types>.

This document defines the following RADIUS messages:

- o Frag-Status
- o Proxy-State-Len

Additionally, allocation of a new Service-Type value for "Additional-Authorization" is requested.

12. References

12.1. Normative References

- [I-D.ietf-radext-radius-extensions]
DeKok, A. and A. Lior, "Remote Authentication Dial In User Service (RADIUS) Protocol Extensions",
draft-ietf-radext-radius-extensions-13 (work in progress),
February 2013.
- [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.
- [RFC2866] Rigney, C., "RADIUS Accounting", RFC 2866, June 2000.
- [RFC3575] Aboba, B., "IANA Considerations for RADIUS (Remote Authentication Dial In User Service)", RFC 3575,

July 2003.

- [RFC3579] Aboba, B. and P. Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", RFC 3579, September 2003.
- [RFC5080] Nelson, D. and A. DeKok, "Common Remote Authentication Dial In User Service (RADIUS) Implementation Issues and Suggested Fixes", RFC 5080, December 2007.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC6158] DeKok, A. and G. Weber, "RADIUS Design Guidelines", BCP 158, RFC 6158, March 2011.

12.2. Informative References

- [RFC4849] Congdon, P., Sanchez, M., and B. Aboba, "RADIUS Filter Rule Attribute", RFC 4849, April 2007.

Authors' Addresses

Alejandro Perez-Mendez (Ed.)
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia, 30100
Spain

Phone: +34 868 88 46 44
Email: alex@um.es

Rafa Marin-Lopez
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia, 30100
Spain

Phone: +34 868 88 85 01
Email: rafa@um.es

Fernando Pereniguez-Garcia
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia, 30100
Spain

Phone: +34 868 88 78 82
Email: pereniguez@um.es

Gabriel Lopez-Millan
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia, 30100
Spain

Phone: +34 868 88 85 04
Email: gabilm@um.es

Diego R. Lopez
Telefonica I+D
Don Ramon de la Cruz, 84
Madrid, 28006
Spain

Phone: +34 913 129 041
Email: diego@tid.es

Alan DeKok
Network RADIUS
15 av du Granier
Meylan, 38240
France

Phone: +34 913 129 041
Email: aland@networkradius.com
URI: <http://networkradius.com>

