

Network Working Group  
INTERNET-DRAFT  
Category: Proposed Standard  
Expires: April 23, 2012  
Updates: 4072

Bernard Aboba  
Microsoft Corporation  
Jouni Malinen  
Devicescape Software  
Paul Congdon  
Hewlett Packard Company  
Joseph Salowey  
Cisco Systems  
22 October 2011

RADIUS Attributes for IEEE 802 Networks  
draft-aboba-radext-wlan-15.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 proposes additional attributes for use within IEEE 802 networks, as well as providing clarifications on the usage of the EAP-Key-Name attribute, updating RFC 4072. The attributes defined in this document are usable both within RADIUS and Diameter.

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 April 23, 2012.

## Copyright Notice

Copyright (c) 2011 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 .....	7
2.3	EAP-Peer-Id .....	8
2.4	EAP-Server-Id .....	9
2.5	Mobility-Domain-Id .....	10
2.6	Preauth-Timeout .....	10
2.7	Network-Id-Name .....	11
2.8	Access-Info .....	12
3.	Table of attributes .....	13
4.	Diameter Considerations .....	14
5.	IANA Considerations .....	15
6.	Security Considerations .....	15
7.	References .....	15
7.1	Normative References .....	15
7.2	Informative References .....	16
	ACKNOWLEDGMENTS .....	17
	AUTHORS' ADDRESSES .....	18

## 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] defined 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. The attributes defined in this document are usable both within RADIUS and Diameter.

### 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 or CoA-Request packet.

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
Type										Length										String...																			

#### Code

TBD1

#### Length

>=3

#### String

The String field is one or more octets, containing the layer 2 endpoint that the user's call is allowed to be terminated on, as specified in the definition of Called-Station-Id in [RFC2865] Section 5.30 and [RFC3580] Section 3.20. In the case of IEEE 802, the Allowed-Called-Station-Id Attribute is used to store the Medium Access Control (MAC) address in ASCII format (upper case only), with octet values separated by a "-". Example: "00-10-A4-23-19-C0". Where restrictions on both the network and authenticator MAC address usage are intended, the network name

MUST be appended to the authenticator MAC address, separated from the MAC address with a ":". Example: "00-10-A4-23-19-C0:AP1". Where no MAC address restriction is intended, the MAC address field MUST be omitted, but the network name field MUST be included. Example: "AP1". Within IEEE 802.11 [IEEE-802.11], the SSID constitutes the network name; within IEEE 802.1X [IEEE-802.1X], the Network-Id Name (NID-Name) constitutes the network name. Since a NID-Name can be up to 253 octets in length, when used with [IEEE-802.1X], there may not be sufficient room within the Allowed-Called-Station-Id Attribute to include a MAC address.

If the user attempts to connect to the NAS from a Called-Station-Id that does not match one of the Allowed-Called-Station-Id attributes, then the user MUST NOT be permitted to access the network.

The Allowed-Called-Station-Id Attribute can be useful in the following situations:

- [1] Where users can connect to a NAS without an Access-Request being sent by the NAS to the RADIUS server (e.g. where key caching is supported within IEEE 802.11 or IEEE 802.1X [IEEE-802.1X]). To avoid elevation of privilege attacks, key cache entries are typically only usable within the network to which the user originally authenticated (e.g. the originally selected network name is implicitly attached to the key cache entry). Also, if it is desired that access to a network name not be available from a particular authenticator MAC address, then the authenticator can be set up not to advertise that particular network name.
- [2] Where pre-authentication may be supported (e.g. IEEE 802.1X pre-authentication). In this situation, the network name typically will not be included in a Called-Station-Id Attribute within the Access-Request, so that the RADIUS server will not know the network that the user is attempting to access. As a result, the RADIUS server may desire to restrict the networks to which the user can subsequently connect.
- [3] Where the network portion of the Called-Station-Id is present within an Access-Request, the RADIUS server can desire to authorize access to a network different from the one that the user selected.



### 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 Accounting-Request packets. More than one EAP-Peer-Id Attribute MUST NOT be included in an Access-Request; one or more EAP-Peer-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-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. 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	2	3	4	5	6	7	8	9
Type										Length										String...																			

#### Code

TBD2

#### Length

>=3

#### 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.

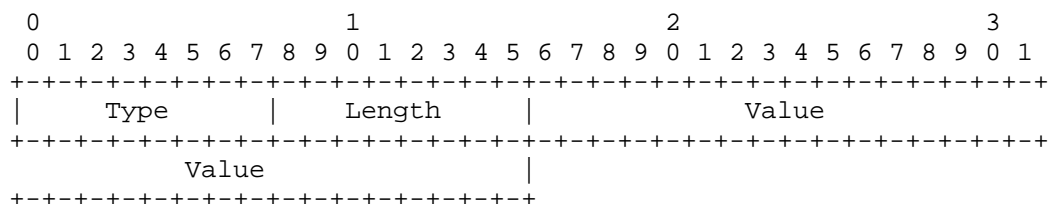




## 2.5. Mobility-Domain-Id

### 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 IEEE 802.11r [IEEE-802.11r]. 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. Since the Mobility Domain Identifier defined in IEEE 802.11r [IEEE-802.11r] is only two octets in length, 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 MDID value.

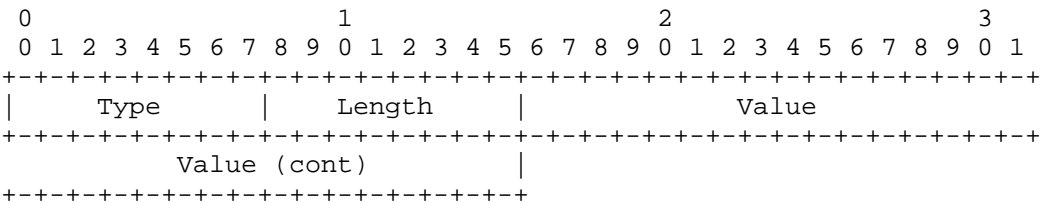
## 2.6. Preauth-Timeout

### 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 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.

This Attribute MAY be sent by the server to the NAS in an Access-Accept. 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 SHOULD 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 a RADIUS Access-Request 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. In order to indicate which network names the Supplicant is permitted to access, the Allowed-Called-Station-Id Attribute is provided within an Access-Accept. 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

TBD7

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. Access-Info

Description

The Access-Info Attribute is utilized by implementations of IEEE-802.1X [IEEE-802.1X] to specify the Access status information field within an Access Information Type Length Value Tuple (TLV) to be sent to the user within MACsec Key Agreement (MKA) or EAPoL-Announcement frames.

A single Access-Info Attribute is permitted within a RADIUS

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

Req	Req	#	Attribute
0+	0	TBD1	Allowed-Called-Station-Id
0-1	0	102	EAP-Key-Name
0	0+	TBD2	EAP-Peer-Id
0	0+	TBD3	EAP-Server-Id
0	0-1	TBD4	Mobility-Domain-Id
0	0	TBD5	Preauth-Timeout
0	0-1	TBD6	Network-Id-Name
0-1	0-1	TBD7	Access-Info

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. Diameter Considerations

The EAP-Key-Name Attribute is already defined as a RADIUS Attribute within Diameter EAP [RFC4072]. When used in Diameter, the other attributes defined in this specification can be used as Diameter AVPs from the Code space 1-255 (RADIUS Attribute compatibility space). No additional Diameter Code values are therefore allocated. The data types and flag rules for the attributes are as follows:

Attribute Name	Value Type	AVP Flag rules				Encr
		MUST	MAY	SHLD NOT	MUST NOT	
Allowed-Called-Station-Id	UTF8String	M	P		V	Y
EAP-Peer-Id	UTF8String	M	P		V	Y
EAP-Server-Id	UTF8String	M	P		V	Y
Mobility-Domain-Id	Unsigned32		P		V	Y
Preauth-Timeout	Unsigned32	M	P		V	Y
Network-Id-Name	UTF8String	M	P		V	Y
Access-Info	Unsigned32	M	P		V	Y

The attributes in this specification have no special translation requirements for Diameter to RADIUS or RADIUS to Diameter gateways; they are copied as is, except for changes relating to headers, alignment, and padding. See also [RFC3588] Section 4.1 and [RFC4005] Section 9.

What this specification says about the applicability of the attributes for RADIUS Access-Request packets applies in Diameter to AA-Request [RFC4005] or Diameter-EAP-Request [RFC4072]. What is said about Access-Challenge applies in Diameter to AA-Answer [RFC4005] or Diameter-EAP-Answer [RFC4072] with Result-Code AVP set to DIAMETER\_MULTI\_ROUND\_AUTH.

What is said about Access-Accept applies in Diameter to AA-Answer or Diameter-EAP-Answer messages that indicate success. Similarly, what is said about RADIUS Access-Reject packets applies in Diameter to AA-Answer or Diameter-EAP-Answer messages that indicate failure.

What is said about COA-Request applies in Diameter to Re-Auth-Request [RFC4005]. What is said about Accounting-Request applies to Diameter Accounting- Request [RFC4005] as well.

## 5. 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
Access-Info	TBD7

## 6. 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].

## 7. References

### 7.1. 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. 802.11-2007, 2007.

## [IEEE-802.11r]

Amendment to Standard for 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 2: Fast BSS Transition, IEEE 802.11r-2008, July 2008.

## [IEEE-802.1X]

IEEE Standard for Local and Metropolitan Area Networks - Port-Based Network Access Control, IEEE 802.1X-2010, February 2010.

[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.

[RFC3588] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., and J. Arkko, "Diameter Base Protocol", RFC 3588, September 2003.

[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.

## 7.2. 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.
- [RFC4005] Calhoun, P., Zorn, G., Spence, D., and D. Mitton, "Diameter Network Access Server Application", RFC 4005, August 2005.
- [RFC5176] Chiba, M., Dommetry, G., Eklund, M., Mitton, D. and B. Aboba, "Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS)", RFC 5176, January 2008.

#### Acknowledgments

The authors would like to acknowledge Mick Seaman, 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](mailto:bernard_aboba@hotmail.com)

Jouni Malinen  
Devicescape Software, Inc.  
900 Cherry Avenue  
San Bruno, CA 94066

EMail: [jkm@devicescape.com](mailto:jkm@devicescape.com)  
Phone: +1 650 829 2600  
Fax: +1 650 829 2601

Paul Congdon  
Hewlett Packard Company  
HP ProCurve Networking  
8000 Foothills Blvd, M/S 5662  
Roseville, CA 95747

Phone: +1 916 785 5753  
Fax: +1 916 785 8478  
EMail: [paul\\_congdon@hp.com](mailto:paul_congdon@hp.com)

Joseph Salowey  
Cisco Systems

EMail: [jsalowey@cisco.com](mailto:jsalowey@cisco.com)

Network Working Group  
INTERNET-DRAFT  
Obsoletes: 4282  
Category: Standards Track  
<draft-dekok-radext-nai-01.txt>  
10 September 2011

DeKok, Alan  
FreeRADIUS

The Network Access Identifier  
draft-dekok-radext-nai-01

Abstract

In order to provide roaming services, it is necessary to have a standardized method for identifying users. This document defines the syntax for the Network Access Identifier (NAI), the user identity submitted by the client during network authentication. "Roaming" may be loosely defined as the ability to use any one of multiple Internet Service Providers (ISPs), while maintaining a formal, customer-vendor relationship with only one. Examples of where roaming capabilities might be required include ISP "confederations" and ISP-provided corporate network access support. This document is a revised version of RFC 4282 [RFC4282], which addresses issues with international character sets, as well as a number of other corrections to the previous document.

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 March 9, 2012.

## Copyright Notice

Copyright (c) 2011 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

Appendix A - Changes from RFC4282 .....	3
1. Introduction .....	4
1.1. Terminology .....	4
1.2. Requirements Language .....	5
1.3. Purpose .....	6
1.4. Motivation .....	6
2. NAI Definition .....	7
2.1. UTF-8 Syntax and Normalization .....	7
2.2. Formal Syntax .....	7
2.3. NAI Length Considerations .....	8
2.4. Support for Username Privacy .....	8
2.5. International Character Sets .....	9
2.6. The Normalization Process .....	10
2.7. Routing inside of AAA Systems .....	10
2.8. Compatibility with Email Usernames .....	11
2.9. Compatibility with DNS .....	11
2.10. Realm Construction .....	12
2.10.1. Historical Practices .....	13
2.11. Examples .....	13
3. Security Considerations .....	14
4. IANA Considerations .....	15
5. References .....	15
5.1. Normative References .....	15
5.2. Informative References .....	16
Appendix A - Changes from RFC4282 .....	18

## 1. Introduction

Considerable interest exists for a set of features that fit within the general category of "roaming capability" for network access, including dialup Internet users, Virtual Private Network (VPN) usage, wireless LAN authentication, and other applications. Interested parties have included the following:

- o Regional Internet Service Providers (ISPs) operating within a particular state or province, looking to combine their efforts with those of other regional providers to offer dialup service over a wider area.
- o National ISPs wishing to combine their operations with those of one or more ISPs in another nation to offer more comprehensive dialup service in a group of countries or on a continent.
- o Wireless LAN hotspots providing service to one or more ISPs.
- o Businesses desiring to offer their employees a comprehensive package of dialup services on a global basis. Those services may include Internet access as well as secure access to corporate intranets via a VPN, enabled by tunneling protocols such as the Point-to-Point Tunneling Protocol (PPTP) [RFC2637], the Layer 2 Forwarding (L2F) protocol [RFC2341], the Layer 2 Tunneling Protocol (L2TP) [RFC2661], and the IPsec tunnel mode [RFC4301].

In order to enhance the interoperability of roaming services, it is necessary to have a standardized method for identifying users. This document defines syntax for the Network Access Identifier (NAI). Examples of implementations that use the NAI, and descriptions of its semantics, can be found in [RFC2194].

This document is a revised version of [RFC4282], which originally defined internationalized NAIs. Differences and enhancements compared to that document are listed in Appendix A.

### 1.1. Terminology

This document frequently uses the following terms:

#### Network Access Identifier

The Network Access Identifier (NAI) is the user identity submitted by the client during network access authentication. In roaming, the purpose of the NAI is to identify the user as well as to assist in the routing of the authentication request. Please note that the NAI may not necessarily be the same as the user's email

address or the user identity submitted in an application layer authentication.

#### Network Access Server

The Network Access Server (NAS) is the device that clients connect to in order to get access to the network. In PPTP terminology, this is referred to as the PPTP Access Concentrator (PAC), and in L2TP terminology, it is referred to as the L2TP Access Concentrator (LAC). In IEEE 802.11, it is referred to as an Access Point.

#### Roaming Capability

Roaming capability can be loosely defined as the ability to use any one of multiple Internet Service Providers (ISPs), while maintaining a formal, customer-vendor relationship with only one. Examples of cases where roaming capability might be required include ISP "confederations" and ISP-provided corporate network access support.

#### Tunneling Service

A tunneling service is any network service enabled by tunneling protocols such as PPTP, L2F, L2TP, and IPsec tunnel mode. One example of a tunneling service is secure access to corporate intranets via a Virtual Private Network (VPN).

### 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].

### 1.3. Purpose

As described in [RFC2194], there are a number of providers offering network access services, and the number of Internet Service Providers involved in roaming consortia is increasing rapidly.

In order to be able to offer roaming capability, one of the requirements is to be able to identify the user's home authentication server. For use in roaming, this function is accomplished via the Network Access Identifier (NAI) submitted by the user to the NAS in the initial network authentication. It is also expected that NASes will use the NAI as part of the process of opening a new tunnel, in order to determine the tunnel endpoint.

### 1.4. Motivation

The changes from [RFC4282] are listed in detail in Appendix A. However, some additional discussion is appropriate to motivate those changes.

The motivation to revise [RFC4282] began with internationalization concerns raised in the context of [EDUROAM]. Section 2.1 of [RFC4282] defines ABNF for realms which limits the realm grammar to English letters, digits, and the hyphen "-" character. The intent appears to have been to encode, compare, and transport realms with the ToASCII operation defined in [RFC5890]. There are a number of problems with this approach:

- o The requirement in Section 2.1 that realms are ASCII conflicts with the Extensible Authentication Protocol (EAP) and RADIUS, which are both 8-bit clean, and which both recommend the use of UTF-8 for identities.
- o Section 2.4 required mappings that are language-specific, and which are nearly impossible for intermediate nodes to perform correctly without information about that language.
- o Section 2.4 requires normalization of user names, which may conflict with local system or administrative requirements.
- o The recommendations in Section 2.4 for treatment of bidirectional characters have proven to be unworkable.
- o The prohibition against use of unassigned code points in Section 2.4 effectively prohibits support for new scripts.
- o No Authentication, Authorization, and Accounting (AAA) client, proxy, or server has implemented any of the requirements



in [RFC4282] Section 2.4, among other sections.

With international roaming growing in popularity, it is important for these issues to be corrected in order to provide robust and inter-operable network services.

## 2. NAI Definition

### 2.1. UTF-8 Syntax and Normalization

UTF-8 characters can be defined in terms of octets using the following ABNF [RFC5234], taken from [RFC3629]:

```
UTF8-xtra-char = UTF8-2 / UTF8-3 / UTF8-4

UTF8-2          = %xC2-DF UTF8-tail

UTF8-3          = %xE0 %xA0-BF UTF8-tail /
                  %xE1-EC 2(UTF8-tail) /
                  %xED %x80-9F UTF8-tail /
                  %xEE-EF 2(UTF8-tail)

UTF8-4          = %xF0 %x90-BF 2( UTF8-tail ) /
                  %xF1-F3 3( UTF8-tail ) /
                  %xF4 %x80-8F 2( UTF8-tail )

UTF8-tail       = %x80-BF
```

These are normatively defined in [RFC3629], but are repeated in this document for reasons of convenience.

See [RFC5198] for a discussion of normalization; implementations of this specification MUST use the Normal Form Composed (NFC) for NAIs.

### 2.2. Formal Syntax

The grammar for the NAI is given below, described in Augmented Backus-Naur Form (ABNF) as documented in [RFC5234].

```
nai              = utf8-username
nai              =/ "@" utf8-realm
nai              =/ utf8-username "@" utf8-realm

utf8-username    = dot-string
dot-string       = string
dot-string       =/ dot-string "." string
string           = utf8-atext
```

```

string          =/ string utf8-atext

utf8-atext      = ALPHA / DIGIT /
                  "!" / "#" /
                  "$" / "%" /
                  "&" / "'" /
                  "*" / "+" /
                  "-" / "/" /
                  "=" / "?" /
                  "^" / "_" /
                  "`" / "{" /
                  "|" / "}" /
                  "~" /
                  UTF8-xtra-char

utf8-realm      = 1*( label "." ) label

label           = utf8-rtext *(ldh-str)
ldh-str         = *( utf8-rtext / "-" ) utf8-rtext
utf8-rtext      = ALPHA / DIGIT / UTF8-xtra-char

```

### 2.3. NAI Length Considerations

Devices handling NAIs MUST support an NAI length of at least 72 octets. Devices SHOULD support an NAI length of 253 octets. However, the following implementation issues should be considered:

- o NAIs are often transported in the User-Name attribute of the Remote Authentication Dial-In User Service (RADIUS) protocol. Unfortunately, RFC 2865 [RFC2865], Section 5.1, states that "the ability to handle at least 63 octets is recommended." As a result, it may not be possible to transfer NAIs beyond 63 octets through all devices. In addition, since only a single User-Name attribute may be included in a RADIUS message and the maximum attribute length is 253 octets; RADIUS is unable to support NAI lengths beyond 253 octets.
- o NAIs can also be transported in the User-Name attribute of Diameter [RFC3588], which supports content lengths up to  $2^{24} - 9$  octets. As a result, NAIs processed only by Diameter nodes can be very long. However, an NAI transported over Diameter may eventually be translated to RADIUS, in which case the above limitations will apply.

### 2.4. Support for Username Privacy

Interpretation of the username part of the NAI depends on the realm in question. Therefore, the utf8-username portion SHOULD be treated

as opaque data when processed by nodes that are not a part of the authoritative domain (in the sense of Section 4) for that realm.

In some situations, NAIs are used together with a separate authentication method that can transfer the username part in a more secure manner to increase privacy. In this case, NAIs MAY be provided in an abbreviated form by omitting the username part. Omitting the username part is RECOMMENDED over using a fixed username part, such as "anonymous", since it provides an unambiguous way to determine whether the username is intended to uniquely identify a single user.

For roaming purposes, it is typically necessary to locate the appropriate backend authentication server for the given NAI before the authentication conversation can proceed. As a result, the realm portion is typically required in order for the authentication exchange to be routed to the appropriate server.

## 2.5. International Character Sets

This specification allows both international usernames and realms. International usernames are based on the use of Unicode characters, encoded as UTF-8. Internationalization of the realm portion of the NAI is based on "Internationalized Email Headers" [RFC5335].

In order to ensure a canonical representation, characters of the username portion in an NAI MUST match the ABNF in this specification as well as the requirements specified in [RFC5891]. In practice, these requirements consist of the following item:

- o Realms MUST be of the form that can be registered as a Fully Qualified Domain Name (FQDN) within the DNS name system.

This list is significantly shorter and simpler than the list in Section 2.4 of [RFC4282]. The form suggested in [RFC4282] depended on intermediate nodes performing canonicalizations based on insufficient information, which meant that the form was not canonical. This document instead suggests (Section 2.10) that the realm owner provide a canonical form of the realm, and that all intermediate nodes use that form without modification.

Specifying the realm requirement as above means that the requirements depend on specifications that are referenced here, rather than copied here. This allows the realm definition to be updated when the referenced documents change, without requiring a revision of this specification.

In general, the above requirement means following the requirements as

specified in [RFC5891]. However, that document is in flux at the time of this writing, and the issues with [RFC4282] mandate a timely update to it.

## 2.6. The Normalization Process

All normalization **MUST** be performed by end systems that take "local" text as input. That is, text that is in an encoding other than UTF-8, or that has locale-specific variations. In a network access setting, such systems are typically the client (e.g. EAP supplicant) and the Authentication, Authorization, and Accounting (AAA) server.

All other AAA systems (proxies, etc.) **MUST NOT** perform normalization. These other systems do not have access to locale and character set information that is available to end systems.

That is, all processing of NAIs from "local" character sets and locales to UTF-8 is performed by edge systems, prior to the NAIs entering the AAA system. Inside of an AAA system, NAIs are sent over the wire in their canonical form, and this canonical form is used for all NAI and/or realm comparisons.

In contrast to the comments in [RFC4282] Section 2.4, we expect AAA systems to perform NAI comparisons, matching, and AAA routing based on the NAI as it is received. This specification provides a canonical representation, ensures that intermediate systems such as AAA proxies do not need to perform translations, and can be expected to work through systems that are unaware of international character sets.

For example, much of the common realm routing can be done on the "utf8-realm" portion of NAI, through simple checks for equality. This routing can be done even if the AAA proxy is unaware of internalized domain names. All that is required is for the AAA proxy to be able to enter, store, and compare 8-bit data.

EAP supplicants **MUST** normalize user names that get placed in the EAP-Response/Identity field. They **MUST NOT** copy localized text into that field. This normalization **SHOULD** be performed once, and then cached for subsequent use.

## 2.7. Routing inside of AAA Systems

Many systems require that the "utf8-realm" portion of the NAI be used to route requests within a AAA proxy network. The semantics of this operation involves a logical AAA routing table, where the "utf8-realm" portion acts as a key, and the values stored in the table are one or more "next hop" AAA servers.

Intermediate nodes MUST use the "utf8-realm" portion of the NAI without modification to perform this lookup. Comparisons between the NAI as given in a AAA packet, and as provisioned in a logical AAA routing table SHOULD be done as a byte-for-byte equality test. The "utf8-realm" provisioned in the logical AAA routing table SHOULD be provisioned prior to the proxy receiving any AAA traffic, and SHOULD be supplied by the "next hop" system that also supplies the other information about the next hop.

This "next hop" information may be IP address, port, RADIUS shared secret, TLS certificates, or a DNS host name.

## 2.8. Compatibility with Email Usernames

As proposed in this document, the Network Access Identifier is of the form user@realm. Please note that while the user portion of the NAI is based on the BNF described in [RFC5198], it has been modified for the purposes of Section 2.2. It does not permit quoted text along with "folding" or "non-folding" whitespace that is commonly used in email addresses. As such, the NAI is not necessarily equivalent to usernames used in e-mail.

However, it is a common practice to use email addresses as user identifiers in AAA systems. The ABNF in Section 2.2 is defined to be close to the "utf8-addr-spec" portion of [RFC5335], while still being compatible with [RFC4282].

In contrast to the comments in [RFC4282] Section 2.5, we state that the internationalization requirements for NAIs and email addresses are substantially similar. The NAI and email identifiers may be the same, and both need to be entered by the user and/or the operator supplying network access to that user. There is therefore good reason for the internationalization requirements to be similar.

## 2.9. Compatibility with DNS

The "realm" portion of the NAI is intended to be compatible with domain names used in DNS systems. However, the "realm" portion within AAA systems is intended to be a UTF-8 string, not an ASCII string as with the DNS protocol. Therefore, AAA systems transporting NAIs in an AAA protocol MUST NOT encode the "utf8-realm" portion using the ToAscii function. That function creates strings that may be transported over DNS, and it is not appropriate for use within an AAA protocol.

When the realm portion of the NAI is used as the basis for name lookups within the DNS system, the ToASCII operation defined in [RFC5890] MAY be used to convert internationalized realm names to

ASCII. This function is normally handled by a DNS resolver library on the local system. When this function is not handled by a DNS resolver library, the AAA system MAY perform the ToAscii conversion itself, before passing the modified realm name to the DNS resolver library.

There is, however, a problem with this approach. A AAA proxy may not have sufficient information in order to perform the ToAscii conversion properly. We therefore RECOMMEND that only the owner of the realm perform the ToAscii conversion. We RECOMMEND that the owner of the realm pre-provision all proxies with the "utf8-realm" portion of the NAI, along with the value returned from passing the "utf8-realm" to the ToAscii function. This key-value pair can then be placed into logical AAA routing table discussed above. Having only one entity run the ToAscii function ensures that the result returned by that function are considered as canonical form by all other participants in a AAA network.

The paragraph above does not negate all of the benefits of using DNS to automatically discover the location of a "next hop" AAA server. Many AAA proxies require a business or legal relationship prior to routing any traffic. This relationship can be leveraged to bootstrap the DNS information located in the logical AAA routing table.

## 2.10. Realm Construction

The home realm usually appears in the realm portion of the NAI, but in some cases a different realm can be used. This may be useful, for instance, when the home realm is reachable only via intermediate proxies.

Such usage may prevent interoperability unless the parties involved have a mutual agreement that the usage is allowed. In particular, NAIs MUST NOT use a different realm than the home realm unless the sender has explicit knowledge that (a) the specified other realm is available and (b) the other realm supports such usage. The sender may determine the fulfillment of these conditions through a database, dynamic discovery, or other means not specified here. Note that the first condition is affected by roaming, as the availability of the other realm may depend on the user's location or the desired application.

The use of the home realm MUST be the default unless otherwise configured.

### 2.10.1. Historical Practices

Some systems have historically used NAI modifications with multiple "prefix" and "suffix" decorations to perform explicit routing through multiple proxies inside of a AAA network. This practice is NOT RECOMMENDED for the following reasons:

- o Using explicit routing paths is fragile, and is unresponsive to changes in the network due to servers going up or down, or to changing business relationships.
- o There is no RADIUS routing protocol, meaning that routing paths have to be communicated "out of band" to all intermediate AAA nodes, and also to all end-user systems (supplicants) expecting to obtain network access.
- o Using explicit routing paths requires thousands, if not millions of end-user systems to be updated with new path information when a AAA routing path changes. This adds huge expense for updates that would be better done at only a few AAA systems in the network.
- o Manual updates to RADIUS paths are expensive, time-consuming, and prone to error.
- o Re-writing of the User-Name in AAA servers means that it may not match the EAP-Response/Identity fields. This mismatch may cause the home AAA server to reject the request as being malformed.
- o Creating compatible formats for the NAI is difficult when locally-defined "prefixes" and "suffixes" conflict with similar practices elsewhere in the network. These conflicts mean that connecting two networks may be impossible in some cases, as there is no way for packets to be routed properly in a way that meets all requirements at all intermediate proxies.
- o Leveraging the DNS name system for realm names establishes a globally unique name space for realms.

In summary, network practices and capabilities have changed significantly since NAIs were first overloaded to define AAA routes through a network. While explicit path routing was once useful, the time has come for better methods to be used.

### 2.11. Examples

Examples of valid Network Access Identifiers include the following:

```
bob
joe@example.com
fred@foo-9.example.com
jack@3rd.depts.example.com
fred.smith@example.com
fred_smith@example.com
fred$@example.com
fred=?#&*+~/^smith@example.com
nancy@eng.example.net
eng.example.net!nancy@example.net
eng%nancy@example.net
@privatecorp.example.net
\ (user\)@example.net
```

Examples of invalid Network Access Identifiers include the following:

```
fred@example
fred@example_9.com
fred@example.net@example.net
fred.@example.net
eng:nancy@example.net
eng;nancy@example.net
(user)@example.net
<nancy>@example.net
```

One example given in [RFC4282] is still permitted by the ABNF, but it is NOT RECOMMENDED because of the use of the ToAscii function to create an ASCII encoding from what is now a valid UTF-8 string.

```
alice@xn--tmonesimerkki-bfbb.example.net
```

### 3. Security Considerations

Since an NAI reveals the home affiliation of a user, it may assist an attacker in further probing the username space. Typically, this problem is of most concern in protocols that transmit the username in clear-text across the Internet, such as in RADIUS, described in [RFC2865] and [RFC2866]. In order to prevent snooping of the username, protocols may use confidentiality services provided by protocols transporting them, such as RADIUS protected by IPsec [RFC3579] or Diameter protected by TLS [RFC3588].

This specification adds the possibility of hiding the username part in the NAI, by omitting it. As discussed in Section 2.4, this is possible only when NAIs are used together with a separate authentication method that can transfer the username in a secure manner. In some cases, application-specific privacy mechanisms have also been used with NAIs. For instance, some EAP methods apply



method-specific pseudonyms in the username part of the NAI [RFC3748]. While neither of these approaches can protect the realm part, their advantage over transport protection is that privacy of the username is protected, even through intermediate nodes such as NASes.

#### 4. IANA Considerations

In order to avoid creating any new administrative procedures, administration of the NAI realm namespace piggybacks on the administration of the DNS namespace.

NAI realm names are required to be unique, and the rights to use a given NAI realm for roaming purposes are obtained coincident with acquiring the rights to use a particular Fully Qualified Domain Name (FQDN). Those wishing to use an NAI realm name should first acquire the rights to use the corresponding FQDN. Using an NAI realm without ownership of the corresponding FQDN creates the possibility of conflict and is therefore discouraged.

Note that the use of an FQDN as the realm name does not require use of the DNS for location of the authentication server. While Diameter [RFC3588] supports the use of DNS for location of authentication servers, existing RADIUS implementations typically use proxy configuration files in order to locate authentication servers within a domain and perform authentication routing. The implementations described in [RFC2194] did not use DNS for location of the authentication server within a domain. Similarly, existing implementations have not found a need for dynamic routing protocols or propagation of global routing information. Note also that there is no requirement that the NAI represent a valid email address.

#### 5. References

##### 5.1. Normative References

[RFC2119]

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

[RFC3629]

Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC5198]

Klensin J., and Padlipsky M., "Unicode Format for Network Interchange", RFC 5198, March 2008

## [RFC5234]

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

## [RFC5890]

Faltstrom, P., Hoffman, P., and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 5890

## 5.2. Informative References

## [RFC2194]

Aboba, B., Lu, J., Alsop, J., Ding, J., and W. Wang, "Review of Roaming Implementations", RFC 2194, September 1997.

## [RFC2341]

Valencia, A., Littlewood, M., and T. Kolar, "Cisco Layer Two Forwarding (Protocol) "L2F"", RFC 2341, May 1998.

## [RFC2637]

Hamzeh, K., Pall, G., Verthein, W., Taarud, J., Little, W., and G. Zorn, "Point-to-Point Tunneling Protocol", RFC 2637, July 1999.

## [RFC2661]

Townsend, W., Valencia, A., Rubens, A., Pall, G., Zorn, G., and B. Palter, "Layer Two Tunneling Protocol "L2TP"", RFC 2661, August 1999.

## [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.

## [RFC3579]

Aboba, B. and P. Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", RFC 3579, September 2003.

## [RFC3588]

Calhoun, P., Loughney, J., Guttman, E., Zorn, G., and J. Arkko, "Diameter Base Protocol", RFC 3588, September 2003.

## [RFC3748]

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

## [RFC4282]

Aboba, B. et al., "The Network Access Identifier", RFC 4282, December 2005.

## [RFC4301]

Kent, S. and S. Keo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

## [RFC5335]

Y. Abel, Ed., "Internationalized Email Headers", RFC 5335, September 2008.

## [EDUROAM]

<http://eduroam.org>, "eduroam (EDUcational ROAMing)"

## [RFC5891]

Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891

## Acknowledgments

The initial text for this document was [RFC4282], which was then heavily edited. The original authors of [RFC4282] were Bernard Aboba, Mark A. Beadles, Jari Arkko, and Pasi Eronen.

The ABNF validator at <http://www.apps.ietf.org/abnf.html> was used to verify the syntactic correctness of the ABNF in Section 2.

## Appendix A - Changes from RFC4282

This document contains the following updates with respect to the previous NAI definition in RFC 4282 [RFC4282]:

- o The formal syntax in Section 2.1 has been updated to forbid non-UTF8 characters. e.g. characters with the "high bit" set.
- o The formal syntax in Section 2.1 has been updated to allow UTF-8 in the "realm" portion of the NAI.
- o The formal syntax in [RFC4282] Section 2.1 applied to the NAI after it was "internationalized" via the ToAscii function. The contents of the NAI before it was "internationalized" were left indeterminate. This document updates the formal syntax to define an internationalized form of the NAI, and forbids the use of the ToAscii function for NAI "internationalization".
- o The grammar for the user and realm portion is based on a combination of the "nai" defined in [RFC4282] Section 2.1, and the "utf8-addr-spec" defined in [RFC5335] Section 4.4.
- o All use of the ToAscii function has been moved to normal requirements on DNS implementations when realms are used as the basis for DNS lookups. This involves no changes to the existing DNS infrastructure.
- o The discussions on internationalized character sets in Section 2.4 have been updated. The suggestion to use the ToAscii function for realm comparisons has been removed. No AAA system implemented the suggestion, so this change should have no operational impact.
- o The section "Routing inside of AAA Systems" section is new in this document. The concept of a "local AAA routing table" is also new, although it accurately describes the functionality of wide-spread implementations.
- o The "Compatibility with EMail Usernames" and "Compatibility with DNS" sections have been revised and updated. We now note that the ToAscii function is required to be used only when a realm name is used for DNS lookups, and even then the function is only used by a DNS resolving library on the local system, and even then we recommend that only the home network perform this conversion.
- o The "Realm Construction" section has been updated to note that editing of the NAI is NOT RECOMMENDED.

- o The "Examples" section has been updated to remove the instance of the IDN being converted to ASCII. This behavior is now forbidden.

#### Authors' Addresses

Alan DeKok  
The FreeRADIUS Server Project  
  
Email: [aland@freeradius.org](mailto:aland@freeradius.org)



Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: August 16, 2013

W. Dec, Ed.  
Cisco Systems, Inc.  
B. Sarikaya  
Huawei USA  
G. Zorn  
Network Zen  
D. Miles  
Google  
B. Lourdelet  
Juniper Networks  
February 12, 2013

RADIUS attributes for IPv6 Access Networks  
draft-ietf-radext-ipv6-access-16

Abstract

This document specifies additional IPv6 RADIUS attributes useful in residential broadband network deployments. The attributes, which are used for authorization and accounting, enable assignment of a host IPv6 address and IPv6 DNS server address via DHCPv6; assignment of an IPv6 route announced via router advertisement; assignment of a named IPv6 delegated prefix pool; and assignment of a named IPv6 pool for host DHCPv6 addressing.

Requirements Language

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

Status of this Memo

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 August 16, 2013.

## 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
2. Deployment Scenarios . . . . .	3
2.1. IPv6 Address Assignment . . . . .	4
2.2. DNS Servers . . . . .	4
2.3. IPv6 Route Information . . . . .	5
2.4. Delegated IPv6 Prefix Pool . . . . .	5
2.5. Stateful IPv6 address pool . . . . .	5
3. Attributes . . . . .	6
3.1. Framed-IPv6-Address . . . . .	6
3.2. DNS-Server-IPv6-Address . . . . .	7
3.3. Route-IPv6-Information . . . . .	8
3.4. Delegated-IPv6-Prefix-Pool . . . . .	9
3.5. Stateful-IPv6-Address-Pool . . . . .	10
3.6. Table of attributes . . . . .	10
4. Diameter Considerations . . . . .	11
5. Security Considerations . . . . .	11
6. IANA Considerations . . . . .	11
7. Acknowledgements . . . . .	12
8. References . . . . .	12
8.1. Normative References . . . . .	12
8.2. Informative References . . . . .	12
Authors' Addresses . . . . .	13



## 1. Introduction

This document specifies additional RADIUS attributes used to support configuration of DHCPv6 and/or ICMPv6 Router Advertisement (RA) parameters on a per-user basis. The attributes, which complement those defined in [RFC3162] and [RFC4818], support the following:

- o Assignment of specific IPv6 addresses to hosts via DHCPv6.
- o Assignment of an IPv6 DNS server address, via DHCPv6 or Router Advertisement [RFC6106].
- o Configuration of more specific routes to be announced to the user via the Route Information Option defined in [RFC4191] Section 2.3.
- o The assignment of a named delegated prefix pool for use with "IPv6 Prefix Options for DHCPv6" [RFC3633].
- o The assignment of a named stateful address pool for use with DHCPv6 stateful address assignment [RFC3315].

## 2. Deployment Scenarios

The extensions in this draft are intended to be applicable across a wide variety of network access scenarios where RADIUS is involved. One such typical network scenario is illustrated in Figure 1. It is composed of a IP Routing Residential Gateway (RG) or host, a Layer 2 Access-Node (AN) e.g. a Digital Subscriber Line Access Multiplexer - DSLAM, an IP Network Access Servers (NASes), and an Authentication Authorization & Accounting (AAA) server.

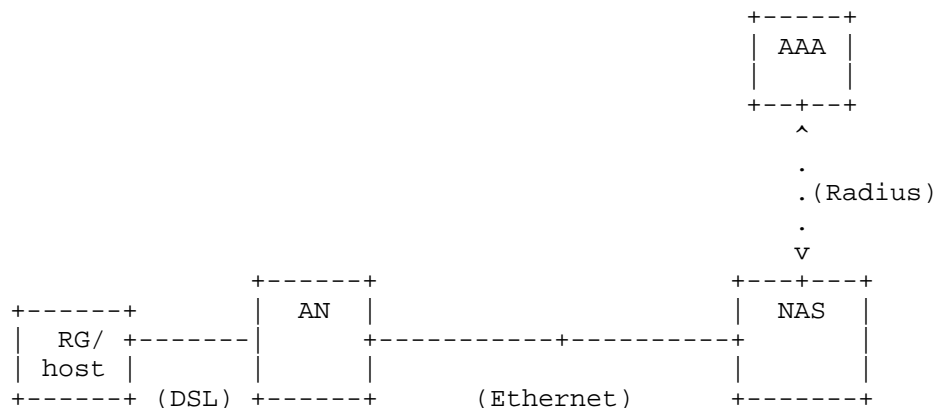


Figure 1

In the depicted scenario the NAS may utilize an IP address configuration protocol (e.g. a DHCPv6 server) to handle address assignment to RGs/hosts. The RADIUS server authenticates each RG/host and returns to the attributes used for authorization and accounting. These attributes can include a host's IPv6 address, a DNS server address and a set of IPv6 routes to be advertised via any suitable protocol, eg ICMPv6 (Neighbour Discovery). The name of a prefix pool to be used for DHCPv6 Prefix Delegation, or the name of an address pool to be used for DHCPv6 address assignment can also be attributes provided to the NAS by the RADIUS AAA server.

The following sub-sections discuss how these attributes are used in more detail.

### 2.1. IPv6 Address Assignment

DHCPv6 [RFC3315] provides a mechanism to assign one or more non-temporary IPv6 addresses to hosts. To provide a DHCPv6 server residing on a NAS with one or more IPv6 addresses to be assigned, this document specifies the Framed-IPv6-Address Attribute.

While [RFC3162] permits an IPv6 address to be specified via the combination of the Framed-Interface-Id and Framed-IPv6-Prefix attributes, this separation is more natural for use with PPP's IPv6 Control Protocol than it is for use with DHCPv6, and the use of a single IPv6 address attribute makes for easier processing of accounting records.

Since DHCPv6 can be deployed on the same network as ICMPv6 stateless (SLAAC) [RFC4862], it is possible that the NAS will require both stateful and stateless configuration information. Therefore it is possible for the Framed-IPv6-Address, Framed-IPv6-Prefix and Framed-Interface-Id attributes [RFC3162] to be included within the same packet. To avoid ambiguity in this case, the Framed-IPv6-Address attribute is intended for authorization and accounting of DHCPv6-assigned addresses and the Framed-IPv6-Prefix and Framed-Interface-Id attributes used for authorization and accounting of addresses assigned via SLAAC.

### 2.2. DNS Servers

DHCPv6 provides an option for configuring a host with the IPv6 address of a DNS server. The IPv6 address of a DNS server can also be conveyed to the host using ICMPv6 with Router Advertisements, via the [RFC6106] option. To provide the NAS with the IPv6 address of a DNS server, this document specifies the DNS-Server-IPv6-Address Attribute.

### 2.3. IPv6 Route Information

An IPv6 Route Information option, defined in [RFC4191] is intended to be used to inform a host connected to the NAS that a specific route is reachable via any given NAS.

This document specifies the RADIUS attribute that allows the AAA server to provision the announcement by the NAS of a specific Route Information Option to an accessing host. The NAS may advertise this route using the method defined in [RFC4191], or using other equivalent methods. Any other information, such as preference or life-time values, that is to be present in the actual announcement using a given method is assumed to be determined by the NAS using means not scoped by this document (e.g. Local configuration on the NAS).

While the Framed-IPv6-Prefix attribute defined in [RFC3162] Section 2.3 causes the route to be advertised in an RA, it cannot be used to configure more specific routes. While the Framed-IPv6-Route attribute defined in [RFC3162] Section 2.5 causes the route to be configured on the NAS, and potentially announced via an IP routing protocol, depending on the value of Framed-Routing, it does not result in the route being announced in an RA.

### 2.4. Delegated IPv6 Prefix Pool

DHCPv6 Prefix Delegation (DHCPv6-PD) [RFC3633] involves a delegating router selecting a prefix and delegating it on a temporary basis to a requesting router. The delegating router may implement a number of strategies as to how it chooses what prefix is to be delegated to a requesting router, one of them being the use of a local named prefix pool. The Delegated-IPv6-Prefix-Pool attribute allows the RADIUS server to convey a prefix pool name to a NAS hosting a DHCPv6-PD server and acting as a delegating router.

Since DHCPv6 Prefix Delegation can be used with SLAAC on the same network, it is possible for the Delegated-IPv6-Prefix-Pool and Framed-IPv6-Pool attributes to be included within the same packet. To avoid ambiguity in this scenario, use of the Delegated-IPv6-Prefix-Pool attribute should be restricted to authorization and accounting of prefix pools used in DHCPv6 Prefix Delegation and the Framed-IPv6-Pool attribute should be used for authorization and accounting of prefix pools used in SLAAC.

### 2.5. Stateful IPv6 address pool

DHCPv6 [RFC3315] provides a mechanism to assign one or more non-temporary IPv6 addresses to hosts. Section 3.1 introduces the

Framed-IPv6-Address attribute to be used for providing a DHCPv6 server residing on a NAS with one or more IPv6 addresses to be assigned to the clients. An alternative way to achieve a similar result is for the NAS to select the IPv6 address to be assigned from an address pool configured for this purpose on the NAS. This document specifies the Stateful-IPv6-Address-Pool attribute to allow the RADIUS server to convey a pool name to be used for such stateful DHCPv6 based addressing, and any subsequent accounting.

### 3. Attributes

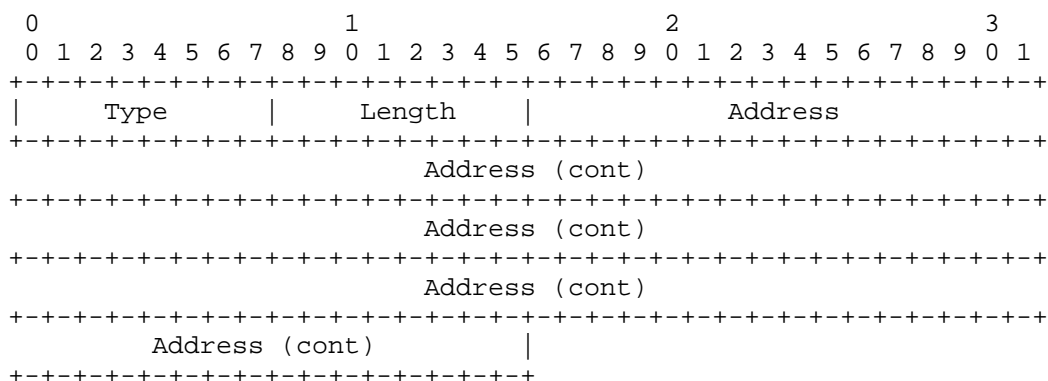
The fields shown in the diagrams below are transmitted from left to right.

#### 3.1. Framed-IPv6-Address

This attribute indicates an IPv6 address that is assigned to the NAS-facing interface of the RG/host. It MAY be used in Access-Accept packets, and MAY appear multiple times. It MAY be used in an Access-Request packet as a hint by the NAS to the RADIUS server that it would prefer these IPv6 address(es), but the RADIUS server is not required to honor the hint. Since it is assumed that the NAS will add a route corresponding to the address, it is not necessary for the RADIUS server to also send a host Framed-IPv6-Route attribute for the same address.

This attribute can be used by a DHCPv6 process on the NAS to assign a unique IPv6 address to the RG/host.

A summary of the Framed-IPv6-Address attribute format is shown below. The format of the address is as per [RFC3162].



## Type

TBA1 for Framed-IPv6-Address

## Length

18

## Address

The IPv6 address field contains a 128-bit IPv6 address.

## 3.2. DNS-Server-IPv6-Address

The DNS-Server-IPv6-Address attribute contains the IPv6 address of a DNS server. This attribute MAY be included multiple times in Access-Accept packets, when the intention is for a NAS to announce more than one DNS server addresses to a RG/host. The same order of the attributes is expected to be followed in the announcements to the RADIUS client. The attribute MAY be used in an Access-Request packet as a hint by the NAS to the RADIUS server regarding the DNS IPv6 address, but the RADIUS server is not required to honor the hint.

The content of this attribute can be inserted in a DHCPv6 option as specified in [RFC3646] or in an IPv6 Router Advertisement as per [RFC6106].

A summary of the DNS-Server-IPv6-Address attribute format is given below. The format of the address is as per [RFC3162].

```

      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      |      Address      |
+-----+-----+-----+-----+-----+-----+-----+
|                                     Address (cont)
+-----+-----+-----+-----+-----+-----+-----+
|                                     Address (cont)
+-----+-----+-----+-----+-----+-----+-----+
|                                     Address (cont)
+-----+-----+-----+-----+-----+-----+-----+
|      Address (cont)      |
+-----+-----+-----+-----+-----+-----+-----+

```

Type

TBA2 for DNS-Server-IPv6-Address

Length

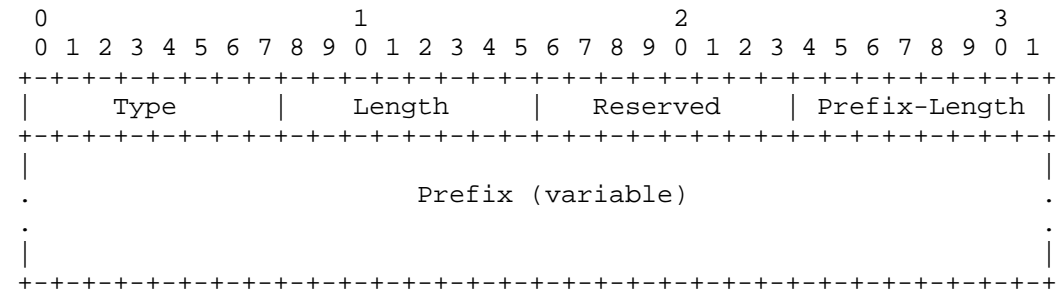
18

Address

The 128-bit IPv6 address of a DNS server.

3.3. Route-IPv6-Information

This attribute specifies a prefix (and corresponding route) for the user on the NAS, which is to be announced using the Route Information Option defined in "Default Router Preferences and More Specific Routes" [RFC4191] Section 2.3. It is used in the Access-Accept packet and can appear multiple times. It MAY be used in an Access-Request packet as a hint by the NAS to the RADIUS server, but the RADIUS server is not required to honor the hint. The Route-IPv6-Information attribute format is depicted below. The format of the prefix is as per [RFC3162].



Type

TBA3 for Route-IPv6-Information

Length

Length in bytes. At least 4 and no larger than 20; typically 12 or less.

## Prefix Length

8-bit unsigned integer. The number of leading bits in the prefix that are valid. The value ranges from 0 to 128. The prefix field is 0, 8 or 16 octets depending on Length.

## Prefix

Variable-length field containing an IP prefix. The prefix length field contains the number of valid leading bits in the prefix. The bits in the prefix after the prefix length (if any) are reserved and MUST be initialized to zero.

## 3.4. Delegated-IPv6-Prefix-Pool

This attribute contains the name of an assigned pool that SHOULD be used to select an IPv6 delegated prefix for the user on the NAS. If a NAS does not support prefix pools, the NAS MUST ignore this attribute. It MAY be used in an Access-Request packet as a hint by the NAS to the RADIUS server regarding the pool, but the RADIUS server is not required to honor the hint.

A summary of the Delegated-IPv6-Prefix-Pool attribute format is shown below.

```

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

```

## Type

TBA4 for Delegated-IPv6-Prefix-Pool

## Length

Length in bytes. At least 3.

## String

The string field contains the name of an assigned IPv6 prefix pool configured on the NAS. The field is not NULL (hexadecimal 00) terminated.

Note: The string data type is as documented in [RFC6158], and carries binary data that is external to the Radius protocol, eg the name of a pool of prefixes configured on the NAS.

### 3.5. Stateful-IPv6-Address-Pool

This attribute contains the name of an assigned pool that SHOULD be used to select an IPv6 address for the user on the NAS. If a NAS does not support address pools, the NAS MUST ignore this attribute. A summary of the Stateful-IPv6-Address-Pool attribute format is shown below. It MAY be used in an Access-Request packet as a hint by the NAS to the RADIUS server regarding the pool, but the RADIUS server is not required to honor the hint.

```

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

```

#### Type

TBA5 for Stateful-IPv6-Address-Pool

#### Length

Length in bytes. At least 3.

#### String

The string field contains the name of an assigned IPv6 stateful address pool configured on the NAS. The field is not NULL (hexadecimal 00) terminated.

Note: The string data type is as documented in [RFC6158], and carries binary data that is external to the Radius protocol, eg the name of a pool of addresses configured on the NAS.

### 3.6. Table of attributes

The following table provides a guide to which attributes may be found in which kinds of packets, and in what quantity. The optional inclusion of the options in Access Request messages is intended to allow for a network access server (NAS) to provide the RADIUS server with a hint of the attributes in advance of user authentication, which may be useful in cases where a user re-connects or has a static address. The server is under no obligation to honor such hints.



Request	Accept	Reject	Challenge	Accounting	#	Attribute
				Request		
0+	0+	0	0	0+	TBA1	Framed-IPv6-Address
0+	0+	0	0	0+	TBA2	DNS-Server-IPv6-Address
0+	0+	0	0	0+	TBA3	Route-IPv6-Information
0+	0+	0	0	0+	TBA4	Delegated-IPv6-Prefix-Pool
0+	0+	0	0	0+	TBA5	Stateful-IPv6-Address-Pool

#### 4. Diameter Considerations

Given that the attributes defined in this document are allocated from the standard RADIUS type space (see Section 6), no special handling is required by Diameter entities.

#### 5. Security Considerations

This document specifies additional IPv6 RADIUS attributes useful in residential broadband network deployments. In such networks, the RADIUS protocol may run either over IPv4 or over IPv6 and known security vulnerabilities of the RADIUS protocol, e.g. [SECI], apply to the attributes defined in this document. A trust relationship between a NAS and RADIUS server is expected to be in place, with communication optionally secured by IPsec or TLS [RFC6614] .

#### 6. IANA Considerations

This document requires the assignment of five new RADIUS attribute types in the "Radius Types" registry (currently located at <http://www.iana.org/assignments/radius-types> for the following attributes:

- o Framed-IPv6-Address
- o DNS-Server-IPv6-Address
- o Route-IPv6-Information
- o Delegated-IPv6-Prefix-Pool
- o Stateful-IPv6-Address-Pool

## 7. Acknowledgements

The authors would like to thank Bernard Aboba, Benoit Claise, Peter Deacon, Alan DeKok, Alfred Hines, Jouni Korhonen, Roberta Maglione, Leaf Yeh, Mark Smith, Pete Resnik, Ralph Droms, Stephen Farrell, Brian Haberman, for their help and comments in reviewing this document.

## 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.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

### 8.2. Informative References

- [RFC3162] Aboba, B., Zorn, G., and D. Mitton, "RADIUS and IPv6", RFC 3162, August 2001.
- [RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.
- [RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, December 2003.
- [RFC3646] Droms, R., "DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3646, December 2003.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.
- [RFC4818] Salowey, J. and R. Droms, "RADIUS Delegated-IPv6-Prefix Attribute", RFC 4818, April 2007.
- [RFC6106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 6106, November 2010.
- [RFC6158] DeKok, A. and G. Weber, "RADIUS Design Guidelines", BCP 158, RFC 6158, March 2011.

- [RFC6614] Winter, S., McCauley, M., Venaas, S., and K. Wierenga,  
"Transport Layer Security (TLS) Encryption for RADIUS",  
RFC 6614, May 2012.
- [SECI] -,  
"http://regul.uni-mb.si/~meolic/ptk-seminarske/  
radius.pdf", November 2001.

## Authors' Addresses

Wojciech Dec (editor)  
Cisco Systems, Inc.  
Haarlerbergweg 13-19  
Amsterdam , NOORD-HOLLAND 1101 CH  
Netherlands

Email: wdec@cisco.com

Behcet Sarikaya  
Huawei USA  
1700 Alma Dr. Suite 500  
Plano, TX  
US

Phone: +1 972-509-5599  
Email: sarikaya@ieee.org

Glen Zorn  
Network Zen  
1310 East Thomas Street  
Seattle, WA  
US

Email: gwz@net-zen.net

David Miles  
Google

Phone:  
Fax:  
Email: david.miles@google.com  
URI:

Benoit Lourdelet  
Juniper Networks  
France

Email: blourdel@juniper.net



Network Working Group  
INTERNET-DRAFT  
Category: Proposed Standard  
Updates: 2865, 2866, 3575, 5176, 6158  
<draft-ietf-radext-radius-extensions-13.txt>  
Expires: August 25, 2013  
25 February 2013

Alan DeKok  
Network RADIUS  
Avi Lior

Remote Authentication Dial In User Service (RADIUS) Protocol  
Extensions  
draft-ietf-radext-radius-extensions-13.txt

Abstract

The Remote Authentication Dial In User Service (RADIUS) protocol is nearing exhaustion of its current 8-bit Attribute Type space. In addition, experience shows a growing need for complex grouping, along with attributes which can carry more than 253 octets of data. This document defines changes to RADIUS which address all of the above problems.

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 June 26, 2013.

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 .....	5
1.1.	Caveats and Limitations .....	6
1.1.1.	Failure to Meet Certain Goals .....	6
1.1.2.	Implementation Recommendations .....	6
1.2.	Terminology .....	7
1.3.	Requirements Language .....	8
2.	Extensions to RADIUS .....	9
2.1.	Extended Type .....	10
2.2.	Long Extended Type .....	11
2.3.	TLV Data Type .....	14
2.3.1.	TLV Nesting .....	16
2.4.	EVS Data Type .....	16
2.5.	Integer64 Data Type .....	18
2.6.	Vendor-ID Field .....	18
2.7.	Attribute Naming and Type Identifiers .....	19
2.7.1.	Attribute and TLV Naming .....	19
2.7.2.	Attribute Type Identifiers .....	19
2.7.3.	TLV Identifiers .....	20
2.7.4.	VSA Identifiers .....	20
2.8.	Invalid Attributes .....	21
3.	Attribute Definitions .....	22
3.1.	Extended-Type-1 .....	23
3.2.	Extended-Type-2 .....	23
3.3.	Extended-Type-3 .....	24
3.4.	Extended-Type-4 .....	25
3.5.	Long-Extended-Type-1 .....	26
3.6.	Long-Extended-Type-2 .....	27
4.	Vendor Specific Attributes .....	28
4.1.	Extended-Vendor-Specific-1 .....	29
4.2.	Extended-Vendor-Specific-2 .....	30
4.3.	Extended-Vendor-Specific-3 .....	31
4.4.	Extended-Vendor-Specific-4 .....	32
4.5.	Extended-Vendor-Specific-5 .....	33
4.6.	Extended-Vendor-Specific-6 .....	35
5.	Compatibility with traditional RADIUS .....	36
5.1.	Attribute Allocation .....	36
5.2.	Proxy Servers .....	37
6.	Guidelines .....	38
6.1.	Updates to RFC 6158 .....	38
6.2.	Guidelines for Simple Data Types .....	38
6.3.	Guidelines for Complex Data Types .....	39
6.4.	Design Guidelines For the New Types .....	40
6.5.	TLV Guidelines .....	41
6.6.	Allocation Request Guidelines .....	41
6.7.	Allocation Requests Guidelines for TLVs .....	42
6.8.	Implementation Guidelines .....	43



6.9. Vendor Guidelines .....	43
7. Rationale for This Design .....	43
7.1. Attribute Audit .....	44
8. Diameter Considerations .....	45
9. Examples .....	45
9.1. Extended Type .....	46
9.2. Long Extended Type .....	47
10. IANA Considerations .....	50
10.1. Attribute Allocations .....	50
10.2. RADIUS Attribute Type Tree .....	50
10.3. Allocation Instructions .....	51
10.3.1. Requested Allocation from the Standard Space ..	52
10.3.2. Requested Allocation from the short extended sp	52
10.3.3. Requested Allocation from the long extended spa	52
10.3.4. Allocation Preferences .....	52
10.3.5. Extending the Type Space via TLV Data Type ....	53
10.3.6. Allocation within a TLV .....	53
10.3.7. Allocation of Other Data Types .....	54
11. Security Considerations .....	54
12. References .....	54
12.1. Normative references .....	54
12.2. Informative references .....	55
Appendix A - Extended Attribute Generator Program .....	56

## 1. Introduction

Under current allocation pressure, we expect that the RADIUS Attribute Type space will be exhausted by 2014 or 2015. We therefore need a way to extend the type space, so that new specifications may continue to be developed. Other issues have also been shown with RADIUS. The attribute grouping method defined in [RFC2868] has been shown to be impractical, and a more powerful mechanism is needed. Multiple attributes have been defined which transport more than the 253 octets of data originally envisioned with the protocol. Each of these attributes is handled as a "special case" inside of RADIUS implementations, instead of as a general method. We therefore also need a standardized method of transporting large quantities of data. Finally, some vendors are close to allocating all of the Attributes within their Vendor-Specific Attribute space. It would be useful to leverage changes to the base protocol for extending the Vendor-Specific Attribute space.

We satisfy all of these requirements through the following changes given in this document:

- \* defining an "Extended Type" format, which adds 8 bits of "Extended Type" to the RADIUS Attribute Type space, by using one octet of the "Value" field. This method gives us a general way of extending the Attribute Type Space. (Section 2.1)
- \* allocating 4 attributes as using the format of "Extended Type". This allocation extends the RADIUS Attribute Type Space by approximately 1000 values. (Sections 3.1, 3.2, 3.3, and 3.4)
- \* defining a "Long Extended Type" format, which inserts an additional octet between the "Extended Type" octet, and the "Value" field. This method gives us a general way of adding additional functionality to the protocol. (Section 2.2)
- \* defining a method which uses the additional octet in the "Long Extended Type" to indicate data fragmentation across multiple Attributes. This method provides a standard way for an Attribute to carry more than 253 octets of data. (Section 2.2)
- \* allocating 2 attributes as using the format "Long Extended Type". This allocation extends the RADIUS Attribute Type Space by an additional 500 values. (Sections 3.5 and 3.6)
- \* defining a new "Type Length Value" (TLV) data type. The data type allows an attribute to carry TLVs as "sub-attributes", which can in turn encapsulate other TLVs as "sub-sub-attributes." This change creates a standard way to group a set of Attributes. (Section 2.3)

- \* defining a new "extended Vendor-Specific" (EVS) data type. The data type allows an attribute to carry Vendor-Specific Attributes (VSAs) inside of the new attribute formats. (Section 2.4)
- \* defining a new "integer64" data type. The data type allows counters which track more than  $2^{32}$  octets of data. (Section 2.5)
- \* allocating 6 attributes using the new EVS data type. This allocation extends the Vendor-Specific Attribute Type space by over 1500 values. (Sections 4.1 through 4.6)
- \* Define the "Vendor-ID" for Vendor-Specific attributes to encompass the entire 4 octets of the Vendor field. [RFC2865] Section 5.26 defined it to be 3 octets, with the high octet being zero. (Section 2.5)
- \* Describing compatibility with existing RADIUS systems. (Section 5)
- \* Defining guidelines for the use of these changes for IANA, implementations of this specification, and for future RADIUS specifications. (Section 6)

As with any protocol change, the changes defined here are the result of a series of compromises. We have tried to find a balance between flexibility, space in the RADIUS message, compatibility with existing deployments, and implementation difficulty.

### 1.1. Caveats and Limitations

This section describes some caveats and limitations with the proposal.

#### 1.1.1. Failure to Meet Certain Goals

One goal which was not met by the above modifications is to have an incentive for standards to use the new space. That incentive is being provided by the exhaustion of the standard space.

#### 1.1.2. Implementation Recommendations

It is RECOMMENDED that implementations support this specification. It is RECOMMENDED that new specifications use the formats defined in this specification.

The alternative to the above recommendations is a circular argument of not implementing this specification because no other standards reference it, and also not defining new standards referencing this specification because no implementations exist.

As noted earlier, the "standard space" is almost entirely allocated. Ignoring the looming crisis benefits no one.

## 1.2. Terminology

This document uses the following terms:

### Silently discard

This means the implementation discards the packet without further processing. The implementation MAY provide the capability of logging the error, including the contents of the silently discarded packet, and SHOULD record the event in a statistics counter.

### Invalid attribute

This means that the Length field of an Attribute is valid (as per [RFC2865], Section 5, top of page 25), but the contents of the Attribute do not follow the correct format. For example, an Attribute of type "address" which encapsulates more than four, or less than four, octets of data. See Section 2.8 for a more complete definition.

### Standard space

Codes in the RADIUS Attribute Type Space that are allocated by IANA and that follow the format defined in Section 5 of [RFC2865].

### Extended space

Codes in the RADIUS Attribute Type Space that require the extensions defined in this document, and are an extension of the standard space, but which cannot be represented within the standard space.

### Short extended space

Codes in the extended space which use the "Extended Type" format.

### Long extended space

Codes in the extended space which use the "Long Extended Type" format.

The following terms are used here with the meanings defined in BCP 26 [RFC5226]: "name space", "assigned value", "registration", "Private Use", "Reserved", "Unassigned", "IETF Review", and "Standards Action".

### 1.3. 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. Extensions to RADIUS

This section defines two new attribute formats; "Extended Type"; and "Long Extended Type". It defines a new Type-Length-Value (TLV) data type, an Extended-Vendor-Specific (EVS) data type, and an Integer64 data type. It defines a new method for naming attributes and identifying Attributes using the new attribute formats. It finally defines the new term "invalid attribute", and describes how it affects implementations.

The new attribute formats are designed to be compatible with the attribute format given in [RFC2865] Section 5. The meaning and interpretation of the Type and Length fields is unchanged from that specification. This reuse allows the new formats to be compatible with RADIUS implementations which do not implement this specification. Those implementations can simply ignore the Value field of an attribute, or forward it verbatim.

The changes to the attribute format come about by "stealing" one or more octets from the Value field. This change has the effect that the Value field of [RFC2865] Section 5 contains both the new octets given here, and any attribute-specific Value. The result is that Values in this specification are limited to less than 253 octets in size. This limitation is overcome through the use of the "Long Extended Type" format.

We reiterate that the formats given in this document do not insert new data into an attribute. Instead, we "steal" one octet of Value, so that the definition of the Length field remains unchanged. The new attribute formats are designed to be compatible with the attribute format given in [RFC2865] Section 5. The meaning and interpretation of the Type and Length fields is unchanged from that specification. This reuse allows the new formats to be compatible RADIUS implementations which do not implement this specification. Those implementations can simply ignore the Value field of an attribute, or forward it verbatim.

The changes to the attribute format come about by "stealing" one or more octets from the Value field. This change has the effect that the Value field of [RFC2865] Section 5 contains both the new octets given here, and any attribute-specific Value. The result is that Values in this specification are limited to less than 253 octets in size. This limitation is overcome through the use of the "Long Extended Type" format.

## 2.1. Extended Type

This section defines a new attribute format, called "Extended Type". A summary of the 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      | Extended-Type | Value ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

### Type

This field is identical to the Type field of the Attribute format defined in [RFC2865] Section 5.

### Length

The Length field is one octet, and indicates the length of this Attribute including the Type, Length, Extended-Type, and Value fields. Permitted values are between 4 and 255. If a client or server receives an Extended Attribute with a Length of 2 or 3, then that Attribute MUST be considered to be an "invalid attribute", and handled as per Section 2.8, below.

### Extended-Type

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

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

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

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

### Value

This field is similar to the Value field of the Attribute format

defined in [RFC2865] Section 5. The format of the data MUST be a valid RADIUS data type.

The Value field is one or more octets.

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

The addition of the Extended-Type field decreases the maximum length for attributes of type "text" or "string" from 253 to 252 octets. Where an Attribute needs to carry more than 252 octets of data, the "Long Extended Type" format MUST be used.

Experience has shown that the "experimental" and "implementation specific" attributes defined in [RFC2865] Section 5 have had little practical value. We therefore do not continue that practice here with the Extended-Type field.

## 2.2. Long Extended Type

This section defines a new attribute format, called "Long Extended Type". It leverages the "Extended Type" format in order to permit the transport of attributes encapsulating more than 253 octets of data. A summary of the 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										Extended-Type										M  Reserved									
Value ...																																							

### Type

This field is identical to the Type field of the Attribute format defined in [RFC2865] Section 5.

### Length

The Length field is one octet, and indicates the length of this Attribute including the Type, Length, Extended-Type, and Value fields. Permitted values are between 5 and 255. If a client or server receives a "Long Extended Type" with a Length of 2, 3, or 4, then that Attribute MUST be considered to be an "invalid attribute", and be handled as per Section 2.8, below.



Note that this Length is limited to the length of this fragment. There is no field which gives an explicit value for the total size of the fragmented attribute.

#### Extended-Type

This field is identical to the Extended-Type field defined above in Section 2.1.

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

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

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

#### Reserved

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

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

## Value

This field is similar to the Value field of the Attribute format defined in [RFC2865] Section 5. It may contain a complete set of data (when the Length field has value less than 255), or it may contain a fragment of data.

The Value field is one or more octets.

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

Any interpretation of the resulting data MUST occur after the fragments have been reassembled. The length of the data MUST be taken as the sum of the lengths of the fragments (i.e. Value fields) from which it is constructed. The format of the data SHOULD be a valid RADIUS data type. If the reassembled data does not match the expected format, all fragments MUST be treated as "invalid attributes", and the reassembled data MUST be discarded.

We note that the maximum size of a fragmented attribute is limited only by the RADIUS packet length limitation (i.e. 4096 octets, not counting various headers and overhead). Implementations MUST be able to handle the case where one fragmented attribute completely fills the packet.

This definition increases the RADIUS Attribute Type space as above, but also provides for transport of Attributes which could contain more than 253 octets of data.

Note that [RFC2865] Section 5 says:

If multiple Attributes with the same Type are present, the order of Attributes with the same Type MUST be preserved by any proxies. The order of Attributes of different Types is not required to be preserved. A RADIUS server or client MUST NOT have any dependencies on the order of attributes of different types. A RADIUS server or client MUST NOT require attributes of the same type to be contiguous.

These requirements also apply to the "Long Extended Type" attribute, including fragments. Implementations MUST be able to process non-contiguous fragments -- that is, fragments which are mixed together with other attributes of a different Type. This will allow them to accept packets, so long as the attributes can be correctly decoded.

### 2.3. TLV Data Type

We define a new data type in RADIUS, called "tlv". The "tlv" data type is an encapsulation layer which permits the "Value" field of an Attribute to contain new sub-Attributes. These sub-Attributes can in turn contain "Value"s of data type TLV. This capability both extends the attribute space, and permits "nested" attributes to be used. This nesting can be used to encapsulate or group data into one or more logical containers.

The "tlv" data type re-uses the RADIUS attribute format, as given below:

```

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

```

#### TLV-Type

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

As with Extended-Type above, the TLV-Type is meaningful only within the context defined by "Type" fields of the encapsulating Attributes. That is, the field may be thought of as defining a new type space of the form "Type.Extended-Type.TLV-Type". Where TLVs are nested, the type space is of the form "Type.Extended-Type.TLV-Type.TLV-Type", etc.

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

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

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

#### TLV-Length

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

attribute", and handled as per Section 2.8, below.

#### TLV-Value

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

The TLV-Value field SHOULD encapsulate a standard RADIUS data type. Non-standard data types SHOULD NOT be used within TLV-Value fields. We note that the TLV-Value field MAY also contain one or more attributes of data type "tlv", which allows for simple grouping and multiple layers of nesting.

The TLV-Value field is limited to containing 253 or fewer octets of data. Specifications which require a TLV to contain more than 253 octets of data are incompatible with RADIUS, and need to be redesigned. Specifications which require the transport of empty Values (i.e. Length = 2) are incompatible with RADIUS, and need to be redesigned.

The TLV-Value field MUST NOT contain data using the "Extended Type" formats defined in this document. The base Extended Attributes format allows for sufficient flexibility that nesting them inside of a TLV offers little additional value.

This TLV definition is compatible with the suggested format of the "String" field of the Vendor-Specific attribute, as defined in [RFC2865] Section 5.26, though that specification does not discuss nesting.

Vendors MAY use attributes of type "tlv" in any Vendor Specific Attribute. It is RECOMMENDED to use type "tlv" for VSAs, in preference to any other format.

If multiple TLVs with the same TLV-Type are present, the order of TLVs with the same TLV-Type MUST be preserved by any proxies. The order of TLVs of different TLV-Types is not required to be preserved. A RADIUS server or client MUST NOT have any dependencies on the order of TLVs of different TLV-Types. A RADIUS server or client MUST NOT require TLVs of the same TLV-type to be contiguous.

The interpretation of multiple TLVs of the same TLV-Type MUST be that of a logical "and", unless otherwise specified. That is, multiple TLVs are interpreted as specifying an unordered set of values. Specifications SHOULD NOT define TLVs to be interpreted as a logical "or". Doing so would mean that a RADIUS client or server would make an arbitrary, and non-deterministic choice among the values.

### 2.3.1. TLV Nesting

TLVs may contain other TLVs. When this occurs, the "container" TLV MUST be completely filled by the "contained" TLVs. That is, the "container" TLV-Length field MUST be exactly two (2) more than the sum of the "contained" TLV-Length fields. If the "contained" TLVs over-fill the "container" TLV, the "container" TLV MUST be considered to be an "invalid attribute", and handled as described in Section 2.8, below.

The depth of TLV nesting is limited only by the restrictions on the TLV-Length field. The limit of 253 octets of data results in a limit of 126 levels of nesting. However, nesting depths of more than 4 are NOT RECOMMENDED. They have not been demonstrated to be necessary in practice, and they appear to make implementations more complex. Reception of packets with such deeply nest TLVs may indicate implementation errors or deliberate attacks. Where implementations do not support deep nesting of TLVs, it is RECOMMENDED that the unsupported layers are treated as "invalid attributes".

### 2.4. EVS Data Type

We define a new data type in RADIUS, called "evs", for "Extended Vendor-Specific". The "evs" data type is an encapsulation layer which permits the "Value" field of an Attribute to contain a Vendor-Id, followed by a Vendor-Type, and then vendor-defined data. This data can in turn contain valid RADIUS data types, or any other data as determined by the vendor.

This data type is intended use in attributes which carry Vendor-Specific information, as is done with the Vendor-Specific Attribute (26). It is RECOMMENDED that this data type be used by a vendor only when the Vendor-Specific Attribute Type space has been fully allocated.

Where [RFC2865] Section 5.26 makes a recommendation for the format of the data following the Vendor-Id, we give a strict definition. Experience has shown that many vendors have not followed the [RFC2865] recommendations, leading to interoperability issues. We hope here to give vendors sufficient flexibility as to meet their needs, while minimizing the use of non-standard VSA formats.

The "evs" data type MAY be used in Attributes having the format of "Extended Type" or "Long Extended Type". It MUST NOT be used in any other Attribute definition, including standard RADIUS Attributes, TLVs, and VSAs.

A summary of the "evs" data type 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
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                           Vendor-Id                                           |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Vendor-Type | Vendor-Value .... |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

#### Vendor-Id

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

#### Vendor-Type

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

#### Vendor-Value

The Vendor-Value field is one or more octets. It SHOULD encapsulate a standard RADIUS data type. Using non-standard data types is NOT RECOMMENDED. We note that the Value field may be of data type "tlv". However, it MUST NOT be of data type "evs", as the use cases are unclear for one vendor delegating Attribute Type space to another vendor.

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

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

Note that unlike the format described in [RFC2865] Section 5.26, this data type has no "Vendor length" field. The length of the Vendor-Value field is implicit, and is determined by taking the "Length" of the encapsulating RADIUS Attribute, and subtracting the length of the attribute header (2 octets), the extended type (1 octet), the Vendor-Id (4 octets), and the Vendor-type (1 octet). i.e. For "Extended Type" attributes, the length of the Vendor-Value field is eight (8) less than the value of the Length field. For "Long Extended Type" attributes, the length of the Vendor-Value field is nine (9) less than the value of the Length field.

## 2.5. Integer64 Data Type

We define a new data type in RADIUS, called "integer64", which carries a 64-bit unsigned integer in network byte order.

This data type is intended to be used in any situation where there is a need to have counters which can count past  $2^{32}$ . The expected use of this data type is within Accounting-Request packets, but this data type SHOULD be used in any packet where 32-bit integers are expected to be insufficient.

The "integer64" data type can be used in Attributes of any format, standard space, extended attributes, TLVs, and VSAs.

A summary of the "integer64" data type 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
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Value ...
+-----+-----+-----+-----+-----+-----+-----+-----+
|
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Attributes having data type "integer64" MUST have the relevant Length field set to eight more than the length of the Attribute header. For standard space Attributes and TLVs, this means that the Length field MUST be set to ten (10). For "Extended Type" Attributes, the Length field MUST be set to eleven (11). For "Long Extended Type" Attributes, the Length field MUST be set to twelve (12).

## 2.6. Vendor-ID Field

We define the Vendor-ID field of Vendor-Specific Attributes to encompass the entire 4 octets of the Vendor field. [RFC2865] Section 5.26 defined it to be 3 octets, with the high octet being zero. This change has no immediate impact on RADIUS, as the maximum Private Enterprise Code defined is still within 16 bits.

However, it is best to make advance preparations for changes in the protocol. As such, it is RECOMMENDED that all implementations support four (4) octets for the Vendor-ID field, instead of three (3).

## 2.7. Attribute Naming and Type Identifiers

Attributes have traditionally been identified by a unique name and number. For example, the attribute named "User-Name" has been allocated number one (1). This scheme needs to be extended in order to be able to refer to attributes of Extended Type, and to TLVs. It will also be used by IANA for allocating RADIUS Attribute Type values.

The names and identifiers given here are intended to be used only in specifications. The system presented here may not be useful when referring to the contents of a RADIUS packet. It imposes no requirements on implementations, as implementations are free to reference RADIUS Attributes via any method they choose.

### 2.7.1. Attribute and TLV Naming

RADIUS specifications traditionally use names consisting of one or more words, separated by hyphens, e.g. "User-Name". However, these names are not allocated from a registry, and there is no restriction other than convention on their global uniqueness.

Similarly, vendors have often used their company name as the prefix for VSA names, though this practice is not universal. For example, for a vendor named "Example", the name "Example-Attribute-Name" SHOULD be used instead of "Attribute-Name". The second form can conflict with attributes from other vendors, whereas the first form cannot.

It is therefore RECOMMENDED that specifications give names to Attributes which attempt to be globally unique across all RADIUS Attributes. It is RECOMMENDED that vendors use their name as a unique prefix for attribute names, e.g. Livingston-IP-Pool instead of IP-Pool. It is RECOMMENDED that implementations enforce uniqueness on names, which would otherwise lead to ambiguity and problems.

We recognise that these suggestions may sometimes be difficult to implement in practice.

TLVs SHOULD be named with a unique prefix that is shared among related attributes. For example, a specification that defines a set of TLVs related to time could create attributes named "Time-Zone", "Time-Day", "Time-Hour", "Time-Minute", etc.

### 2.7.2. Attribute Type Identifiers

The RADIUS Attribute Type space defines a context for a particular "Extended-Type" field. The "Extended-Type" field allows for 256



possible type code values, with values 1 through 240 available for allocation. We define here an identification method that uses a "dotted number" notation similar to that used for Object Identifiers (OIDs), formatted as "Type.Extended-Type".

For example, an attribute within the Type space of 241, having Extended-Type of one (1), is uniquely identified as "241.1". Similarly, an attribute within the Type space of 246, having Extended-Type of ten (10), is uniquely identified as "246.10".

### 2.7.3. TLV Identifiers

We can extend the Attribute reference scheme defined above for TLVs. This is done by leveraging the "dotted number" notation. As above, we define an additional TLV type space, within the "Extended Type" space, by appending another "dotted number" in order to identify the TLV. This method can be repeated in sequence for nested TLVs.

For example, let us say that "245.1" identifies RADIUS Attribute Type 245, containing an "Extended Type" of one (1), which is of type "tlv". That attribute will contain 256 possible TLVs, one for each value of the TLV-Type field. The first TLV-Type value of one (1) can then be identified by appending a ".1" to the number of the encapsulating attribute ("241.1"), to yield "241.1.1". Similarly, the sequence "245.2.3.4" identifies RADIUS attribute 245, containing an "Extended Type" of two (2) which is of type "tlv", which in turn contains a TLV with TLV-Type number three (3), which in turn contains another TLV, with TLV-Type number four (4).

### 2.7.4. VSA Identifiers

There has historically been no method for numerically addressing VSAs. The "dotted number" method defined here can also be leveraged to create such an addressing scheme. However, as the VSAs are completely under the control of each individual vendor, this section provides a suggested practice, but does not define a standard of any kind.

The Vendor-Specific Attribute has been assigned the Attribute number 26. It in turn carries a 24-bit Vendor-Id, and possibly additional VSAs. Where the VSAs follow the [RFC2865] Section 5.26 recommended format, a VSA can be identified as "26.Vendor-Id.Vendor-Type".

For example, Livingston has Vendor-Id 307, and has defined an attribute "IP-Pool" as number 6. This VSA can be uniquely identified as 26.307.6, but it cannot be uniquely identified by name, as other vendors may have used the same name.

Note that there are few restrictions on the size of the numerical values in this notation. The Vendor-Id is a 24-bit number, and the VSA may have been assigned from a 16-bit vendor-specific Attribute Type space. Implementations SHOULD be capable of handling 32-bit numbers at each level of the "dotted number" notation.

For example, the company USR has been allocated Vendor-Id 429, and has defined a "Version-Id" attribute as number 32768. This VSA can be uniquely identified as 26.429.32768, and again cannot be uniquely identified by name.

Where a VSA is a TLV, the "dotted number" notation can be used as above: 26.Vendor-Id.Vendor-Type.TLV1.TLV2.TLV3 where "TLVn" are the numerical values assigned by the vendor to the different nested TLVs.

## 2.8. Invalid Attributes

The term "invalid attribute" is new to this specification. It is defined to mean that the Length field of an Attribute permits the packet to be accepted as not being "malformed". However, the Value field of the attribute does not follow the format required by the data type defined for that Attribute, and therefore the attribute is "malformed". In order to distinguish the two cases, we refer to "malformed" packets, and "invalid attributes".

For example, an implementation receives a packet which is well-formed. That packet contains an Attribute allegedly of data type "address", but which has Length not equal to four. In that situation, the packet is well formed, but the attribute is not. Therefore, it is an "invalid attribute".

A similar analysis can be performed when an attribute carries TLVs. The encapsulating attribute may be well formed, but the TLV may be an "invalid attribute". The existence of an "invalid attribute" in a packet or attribute MUST NOT result in the implementation discarding the entire packet, or treating the packet as a negative acknowledgment. Instead, only the "invalid attribute" is treated specially.

When an implementation receives an "invalid attribute", it SHOULD be silently discarded, except when the implementation is acting as a proxy (see Section 5.2 for discussion of proxy servers). If it is not discarded, it MUST NOT be handled in the same manner as a well-formed attribute. For example, receiving an Attribute of data type "address" containing less than four octets, or more than four octets of data means that the Attribute MUST NOT be treated as being of data type "address". The reason here is that if the attribute does not carry an IPv4 address, the receiver has no idea what format the data

is in, and it is therefore not an IPv4 address.

For Attributes of type "Long Extended Type", an Attribute is considered to be an "invalid attribute" when it does not match the criteria set out in Section 2.2, above.

For Attributes of type "TLV", an Attribute is considered to be an "invalid attribute" when the TLV-Length field allows the encapsulating Attribute to be parsed, but the TLV-Value field does not match the criteria for that TLV. Implementations SHOULD NOT treat the "invalid attribute" property as being transitive. That is, the Attribute encapsulating the "invalid attribute" SHOULD NOT be treated as an "invalid attribute". That encapsulating Attribute might contain multiple TLVs, only one of which is an "invalid attribute".

However, a TLV definition may require particular sub-TLVs to be present, and/or to have specific values. If a sub-TLV is missing, or contains incorrect value(s), or is an "invalid attribute", then the encapsulating TLV SHOULD be treated as an "invalid attribute". This requirement ensures that strongly connected TLVs are handled either as a coherent whole, or are ignored entirely.

It is RECOMMENDED that Attributes with unknown Type, Ext-Type, TLV-Type, or VSA-Type are treated as "invalid attributes". This recommendation is compatible with the suggestion in [RFC2865] Section 5, that implementations "MAY ignore Attributes with an unknown Type".

### 3. Attribute Definitions

We define four (4) attributes of "Extended Type", which are allocated from the "Reserved" Attribute Type codes of 241, 242, 243, and 244. We also define two (2) attributes of "Long Extended Type", which are allocated from the "Reserved" Attribute Type codes of 245 and 246.

Type	Name
----	----
241	Extended-Type-1
242	Extended-Type-2
243	Extended-Type-3
244	Extended-Type-4
245	Long-Extended-Type-1
246	Long-Extended-Type-2

The rest of this section gives a detailed definition for each Attribute based on the above summary.

### 3.1. Extended-Type-1

## Description

This attribute encapsulates attributes of the "Extended Type" format, in the RADIUS Attribute Type Space of 241.{1-255}.

A summary of the Extended-Type-1 Attribute format is shown below. The fields are transmitted from left to right.

										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
Type										Length										Extended-Type										Value ...									

## Type

241 for Extended-Type-1.

## Length

$$\geq 4$$

## Extended-Type

The Extended-Type field is one octet. Up-to-date values of this field are specified in the 241.{1-255} RADIUS Attribute Type Space, according to the policies and rules described in Section 10. Further definition of this field is given in Section 2.1, above.

## Value

The Value field is one or more octets.

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

### 3.2. Extended-Type-2

## Description

This attribute encapsulates attributes of the "Extended Type" format, in the RADIUS Attribute Type Space of 242.{1-255}.

A summary of the Extended-Type-2 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      | Extended-Type | Value ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Type

242 for Extended-Type-2.

Length

>= 4

Extended-Type

The Extended-Type field is one octet. Up-to-date values of this field are specified in the 242.{1-255} RADIUS Attribute Type Space, according to the policies and rules described in Section 10. Further definition of this field is given in Section 2.1, above.

Value

The Value field is one or more octets.

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

### 3.3. Extended-Type-3

Description

This attribute encapsulates attributes of the "Extended Type" format, in the RADIUS Attribute Type Space of 243.{1-255}.

A summary of the Extended-Type-3 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      | Extended-Type | Value ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

```

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

**Type**

243 for Extended-Type-3.

**Length**

>= 4

**Extended-Type**

The Extended-Type field is one octet. Up-to-date values of this field are specified in the 243.{1-255} RADIUS Attribute Type Space, according to the policies and rules described in Section 10. Further definition of this field is given in Section 2.1, above.

**Value**

The Value field is one or more octets.

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

**3.4. Extended-Type-4****Description**

This attribute encapsulates attributes of the "Extended Type" format, in the RADIUS Attribute Type Space of 244.{1-255}.

A summary of the Extended-Type-4 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      | Extended-Type | Value ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

**Type**

244 for Extended-Type-4.

**Length**

>= 4

#### Extended-Type

The Extended-Type field is one octet. Up-to-date values of this field are specified in the 244.{1-255} RADIUS Attribute Type Space, according to the policies and rules described in Section 10. Further definition of this field is given in Section 2.1, above.

#### Value

The Value field is one or more octets.

Implementations supporting this specification MUST use the Identifier of "Type.Extended-Type" to determine the interpretation of the Value Field.

### 3.5. Long-Extended-Type-1

#### Description

This attribute encapsulates attributes of the "Long Extended Type" format, in the RADIUS Attribute Type Space of 245.{1-255}.

A summary of the Long-Extended-Type-1 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										Extended-Type										M										Reserved									
Value ...																																																	

#### Type

245 for Long-Extended-Type-1

#### Length

>= 5

#### Extended-Type

The Extended-Type field is one octet. Up-to-date values of this

field are specified in the 245.{1-255} RADIUS Attribute Type Space, according to the policies and rules described in Section 10. Further definition of this field is given in Section 2.1, above.

#### 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. Further definition of this field is given in Section 2.2, above.

#### 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 field is one or more octets.

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

### 3.6. Long-Extended-Type-2

#### Description

This attribute encapsulates attributes of the "Long Extended Type" format, in the RADIUS Attribute Type Space of 246.{1-255}.

A summary of the Long-Extended-Type-2 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										Extended-Type										M	Reserved									
Value ...																																								

#### Type

246 for Long-Extended-Type-2



Length

>= 5

Extended-Type

The Extended-Type field is one octet. Up-to-date values of this field are specified in the 246.{1-255} RADIUS Attribute Type Space, according to the policies and rules described in Section 10. Further definition of this field is given in Section 2.1, above.

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. Further definition of this field is given in Section 2.2, above.

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 field is one or more octets.

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

#### 4. Vendor Specific Attributes

We define six new attributes which can carry Vendor Specific information. We define four (4) attributes of the "Extended Type" format, with Type codes (241.26, 242.26, 243.26, 244.26), using the "evs" data type. We also define two (2) attributes using "Long Extended Type" format, with Type codes (245.26, 246.26), which are of the "evs" data type.

Type.Extended-Type	Name
-----	----
241.26	Extended-Vendor-Specific-1
242.26	Extended-Vendor-Specific-2
243.26	Extended-Vendor-Specific-3
244.26	Extended-Vendor-Specific-4

245.26                   Extended-Vendor-Specific-5  
 246.26                   Extended-Vendor-Specific-6

The rest of this section gives a detailed definition for each Attribute based on the above summary.

#### 4.1. Extended-Vendor-Specific-1

##### Description

This attribute defines a RADIUS Type Code of 241.26, using the "evs" data type.

A summary of the Extended-Vendor-Specific-1 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      | Extended-Type | Vendor-Id ...
+-----+-----+-----+-----+-----+-----+-----+-----+
| ... Vendor-Id (cont) | Vendor-Type |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Value ....
+-----+-----+-----+-----+-----+-----+-----+-----+

```

##### Type.Extended-Type

241.26 for Extended-Vendor-Specific-1

##### Length

>= 9

##### Vendor-Id

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

##### Vendor-Type

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

##### Value

The Value field is one or more octets. The actual format of the information is site or application specific, and a robust

implementation SHOULD support the field as undistinguished octets.

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

The length of the Value field is eight (8) less than the value of the Length field.

Implementations supporting this specification MUST use the Identifier of "Type.Extended-Type.Vendor-Id.Vendor-Type" to determine the interpretation of the Value field.

#### 4.2. Extended-Vendor-Specific-2

##### Description

This attribute defines a RADIUS Type Code of 242.26, using the "evs" data type.

A summary of the Extended-Vendor-Specific-2 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      | Extended-Type | Vendor-Id ...
+-----+-----+-----+-----+-----+-----+-----+-----+
| ... Vendor-Id (cont) | Vendor-Type |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Value ....
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Type.Extended-Type

242.26 for Extended-Vendor-Specific-2

Length

>= 9

Vendor-Id

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

Vendor-Type

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

sole discretion of the Vendor.

#### Value

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

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

The length of the Value field is eight (8) less than the value of the Length field.

Implementations supporting this specification MUST use the Identifier of "Type.Extended-Type.Vendor-Id.Vendor-Type" to determine the interpretation of the Value field.

### 4.3. Extended-Vendor-Specific-3

#### Description

This attribute defines a RADIUS Type Code of 243.26, using the "evs" data type.

A summary of the Extended-Vendor-Specific-3 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   | Extended-Type | Vendor-Id ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
... Vendor-Id (cont) | Vendor-Type |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Value ....
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

#### Type.Extended-Type

243.26 for Extended-Vendor-Specific-3

#### Length

>= 9

#### Vendor-Id

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

#### Vendor-Type

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

#### Value

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

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

The length of the Value field is eight (8) less than the value of the Length field.

Implementations supporting this specification MUST use the Identifier of "Type.Extended-Type.Vendor-Id.Vendor-Type" to determine the interpretation of the Value field.

### 4.4. Extended-Vendor-Specific-4

#### Description

This attribute defines a RADIUS Type Code of 244.26, using the "evs" data type.

A summary of the Extended-Vendor-Specific-3 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   | Extended-Type | Vendor-Id ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| ... Vendor-Id (cont) | Vendor-Type |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Value ....
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

#### Type.Extended-Type

244.26 for Extended-Vendor-Specific-4

Length

>= 9

Vendor-Id

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

Vendor-Type

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

Value

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

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

The length of the Value field is eight (8) less than the value of the Length field.

Implementations supporting this specification MUST use the Identifier of "Type.Extended-Type.Vendor-Id.Vendor-Type" to determine the interpretation of the Value field.

#### 4.5. Extended-Vendor-Specific-5

Description

This attribute defines a RADIUS Type Code of 245.26, using the "evs" data type.

A summary of the Extended-Vendor-Specific-5 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										Extended-Type										M		Reserved									
																														Vendor-Id											
Vendor-Type										Value ....																															

+-----+

Type.Extended-Type

245.26 for Extended-Vendor-Specific-5

Length

>= 10 (first fragment)  
>= 5 (subsequent fragments)

When a VSA is fragmented across multiple Attributes, only the first Attribute contains the Vendor-Id and Vendor-Type fields. Subsequent Attributes contain fragments of the Value field only.

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. Further definition of this field is given in Section 2.2, above.

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.

Vendor-Id

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

Vendor-Type

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

Value

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

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

Implementations supporting this specification MUST use the

Identifier of "Type.Extended-Type.Vendor-Id.Vendor-Type" to determine the interpretation of the Value field.

#### 4.6. Extended-Vendor-Specific-6

##### Description

This attribute defines a RADIUS Type Code of 246.26, using the "evs" data type.

A summary of the Extended-Vendor-Specific-6 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										Extended-Type										M										Reserved									
										Vendor-Id																																							
Vendor-Type										Value ....																																							

##### Type.Extended-Type

246.26 for Extended-Vendor-Specific-6

##### Length

>= 10 (first fragment)  
>= 5 (subsequent fragments)

When a VSA is fragmented across multiple Attributes, only the first Attribute contains the Vendor-Id and Vendor-Type fields. Subsequent Attributes contain fragments of the Value field only.

##### 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. Further definition of this field is given in Section 2.2, above.

##### 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.



#### Vendor-Id

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

#### Vendor-Type

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

#### Value

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

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

Implementations supporting this specification MUST use the Identifier of "Type.Extended-Type.Vendor-Id.Vendor-Type" to determine the interpretation of the Value field.

### 5. Compatibility with traditional RADIUS

There are a number of potential compatibility issues with traditional RADIUS, as defined in [RFC6158] and earlier. This section describes them.

#### 5.1. Attribute Allocation

Some vendors have used Attribute Type codes from the "Reserved" space, as part of vendor-defined dictionaries. This practice is considered anti-social behavior, as noted in [RFC6158]. These vendor definitions conflict with the attributes in the RADIUS Attribute Type space. The conflicting definitions may make it difficult for implementations to support both those Vendor Attributes, and the new Extended Attribute formats.

We RECOMMEND that RADIUS client and server implementations delete all references to these improperly defined attributes. Failing that, we RECOMMEND that RADIUS server implementations have a per-client configurable flag which indicates which type of attributes are being sent from the client. If the flag is set to "Non-Standard Attributes", the conflicting attributes can be interpreted as being improperly defined Vendor Specific Attributes. If the flag is set the "IETF Attributes", the attributes MUST be interpreted as being of the Extended Attributes format. The default SHOULD be to interpret the

attributes as being of the Extended Attributes format.

Other methods of determining how to decode the attributes into a "correct" form are NOT RECOMMENDED. Those methods are likely to be fragile and prone to error.

We RECOMMEND that RADIUS server implementations re-use the above flag to determine which type of attributes to send in a reply message. If the request is expected to contain the improperly defined attributes, the reply SHOULD NOT contain Extended Attributes. If the request is expected to contain Extended Attributes, the reply MUST NOT contain the improper Attributes.

RADIUS clients will have fewer issues than servers. Clients MUST NOT send improperly defined Attributes in a request. For replies, clients MUST interpret attributes as being of the Extended Attributes format, instead of the improper definitions. These requirements impose no change in the RADIUS specifications, as such usage by vendors has always been in conflict with the standard requirements and the standards process.

Existing clients that send these improperly defined attributes usually have a configuration setting which can disable this behavior. We RECOMMEND that vendors ship products with the default set to "disabled". We RECOMMEND that administrators set this flag to "disabled" on all equipment that they manage.

## 5.2. Proxy Servers

RADIUS Proxy servers will need to forward Attributes having the new format, even if they do not implement support for the encoding and decoding of those attributes. We remind implementers of the following text in [RFC2865] Section 2.3:

The forwarding server MUST NOT change the order of any attributes of the same type, including Proxy-State.

This requirement solves some of the issues related to proxying of the new format, but not all. The reason is that proxy servers are permitted to examine the contents of the packets that they forward. Many proxy implementations not only examine the attributes, but they refuse to forward attributes which they do not understand (i.e. attributes for which they have no local dictionary definitions).

This practice is NOT RECOMMENDED. Proxy servers SHOULD forward attributes, even ones which they do not understand, or which are not in a local dictionary. When forwarded, these attributes SHOULD be sent verbatim, with no modifications or changes. This requirement

includes "invalid attributes", as there may be some other system in the network which understands them.

The only exception to this recommendation is when local site policy dictates that filtering of attributes has to occur. For example, a filter at a visited network may require removal of certain authorization rules which apply to the home network, but not to the visited network. This filtering can sometimes be done even when the contents of the attributes are unknown, such as when all Vendor-Specific Attributes are designated for removal.

As seen in [EDUROAM] many proxies do not follow these practices for unknown Attributes. Some proxies filter out unknown attributes or attributes which have unexpected lengths (24%, 17/70), some truncate the attributes to the "expected" length (11%, 8/70), some discard the request entirely (1%, 1/70), with the rest (63%, 44/70) following the recommended practice of passing the attributes verbatim. It will be difficult to widely use the Extended Attributes format until all non-conformant proxies are fixed. We therefore RECOMMEND that all proxies which do not support the Extended Attributes (241 through 246) define them as being of data type "string", and delete all other local definitions for those attributes.

This last change should enable wider usage of the Extended Attributes format.

## 6. Guidelines

This specification proposes a number of changes to RADIUS, and therefore requires a set of guidelines, as has been done in [RFC6158]. These guidelines include suggestions around design, interaction with IANA, usage, and implementation of attributes using the new formats.

### 6.1. Updates to RFC 6158

This specification updates [RFC6158] by adding the data types "evs", "tlv" and "integer64"; defining them to be "basic" data types; and permitting their use subject to the restrictions outlined below.

The recommendations for the use of the new data types and attribute formats are given below.

### 6.2. Guidelines for Simple Data Types

[RFC6158] Section A.2.1 says in part:

- \* Unsigned integers of size other than 32 bits.

SHOULD be replaced by an unsigned integer of 32 bits. There is insufficient justification to define a new size of integer.

We update that specification to permit unsigned integers of 64 bits, for the reasons defined above in Section 2.5. The updated text is as follows:

- \* Unsigned integers of size other than 32 or 64 bits.  
SHOULD be replaced by an unsigned integer of 32 or 64 bits.  
There is insufficient justification to define a new size of integer.

That section later continues with the following list item:

- \* Nested attribute-value pairs (AVPs).  
Attributes should be defined in a flat typespace.

We update that specification to permit nested TLVs, as defined in this document:

- \* Nested attribute-value pairs (AVPs) using the extended attribute format MAY be used. All other nested AVP or TLV formats MUST NOT be used.

The [RFC6158] recommendations for "basic" data types apply to the three types listed above. All other recommendations given in [RFC6158] for "basic" data types remain unchanged.

### 6.3. Guidelines for Complex Data Types

[RFC6158] Section 2.1 says:

Complex data types MAY be used in situations where they reduce complexity in non-RADIUS systems or where using the basic data types would be awkward (such as where grouping would be required in order to link related attributes).

Since the extended attribute format allows for grouping of complex types via TLVs, the guidelines for complex data types need to be updated as follows:

[RFC6158], Section 3.2.4, describes situations in which complex data types might be appropriate. They SHOULD NOT be used even in those situations, without careful consideration of the described limitations. In all other cases not covered by the complex data type exceptions, complex data types MUST NOT be used. Instead,

complex data types MUST be decomposed into TLVs.

The checklist in Appendix A.2.2 is similarly updated to add a new requirement at the top of that section,

Does the attribute:

- \* define a complex type which can be represented via TLVs?

If so, this data type MUST be represented via TLVs.

Note that this requirement does not over-ride Section A.1, which permits the transport of complex types in certain situations.

All other recommendations given in [RFC6158] for "complex" data types remain unchanged.

#### 6.4. Design Guidelines For the New Types

This section gives design guidelines for specifications defining attributes using the new format. The items listed below are not exhaustive. As experience is gained with the new formats, later specifications may define additional guidelines.

- \* The data type "evs" MUST NOT be used for standard RADIUS Attributes, or for TLVs, or for VSAs.
- \* The data type "tlv" SHOULD NOT be used for standard RADIUS attributes.
- \* [RFC2866] "tagged" attributes MUST NOT be defined in the Extended-Type space. The "tlv" data type should be used instead to group attributes.
- \* The "integer64" data type MAY be used in any RADIUS attribute. The use of 64-bit integers was not recommended in [RFC6158], but their utility is now evident.
- \* Any attribute which is allocated from the "long extended space" of data type "text", "string", or "tlv" can potentially carry more than 251 octets of data. Specifications defining such attributes SHOULD define a maximum length to guide implementations.

All other recommendations given in [RFC6158] for attribute design guidelines apply to attributes using the "short extended space" and "long extended space".

## 6.5. TLV Guidelines

The following items give design guidelines for specifications using TLVs.

- \* when multiple attributes are intended to be grouped or managed together, the use of TLVs to group related attributes is RECOMMENDED.
- \* more than 4 layers (depth) of TLV nesting is NOT RECOMMENDED.
- \* Interpretation of an attribute depends only on its type definition (e.g. Type.Extended-Type.TLV-Type), and not on its encoding or location in the RADIUS packet.
- \* Where a group of TLVs is strictly defined, and not expected to change, and totals less than 247 octets of data, they SHOULD request allocation from the "short extended space".
- \* Where a group of TLVs is loosely defined, or is expected to change, they SHOULD request allocation from the "long extended space".

All other recommendations given in [RFC6158] for attribute design guidelines apply to attributes using the TLV format.

## 6.6. Allocation Request Guidelines

The following items give guidelines for allocation requests made in a RADIUS specification.

- \* Discretion is recommended when requesting allocation of attributes. The new space is much larger than the old one, but it is not infinite.
- \* Specifications which allocate many attributes MUST NOT request that allocation be made from the standard space. That space is under allocation pressure, and the extended space is more suitable for large allocations. As a guideline, we suggest that one specification allocating twenty percent (20%) or more of the standard space would meet the above criteria.
- \* Specifications which allocate many related attributes SHOULD define one or more TLVs to contain related attributes.
- \* Specifications SHOULD request allocation from a specific space. The IANA considerations given in Section 9, below, give instruction to IANA, but authors should assist IANA where possible.

- \* Specifications of an attribute which encodes 252 octets or less of data MAY request allocation from the "short extended space".
- \* Specifications of an attribute which always encode less than 253 octets of data MUST NOT request allocation from the long extended space. The standard space, or the short extended space MUST be used instead.
- \* Specifications of an attribute which encodes 253 octets or more of data MUST request allocation from the "long extended space".
- \* When the extended space is nearing exhaustion, a new specification will have to be written which requests allocation of one or more RADIUS Attributes from the "Reserved" portion of the standard space, values 247-255, using an appropriate format ("Short Extended Type", or "Long Extended Type")

An allocation request made in a specification SHOULD use one of the following formats when allocating an attribute type code:

- \* TBDn - request allocation of an attribute from the "standard space". The value "n" should be "1" or more, to track individual attributes which are to be allocated.
- \* SHORT-TBDn - request allocation of an attribute from the "short extended space". The value "n" should be "1" or more, to track individual attributes which are to be allocated.
- \* LONG-TBDn - request allocation of an attribute from the "long extended space". The value "n" should be "1" or more, to track individual attributes which are to be allocated.

These guidelines should help specification authors and IANA communicate effectively and clearly.

#### 6.7. Allocation Requests Guidelines for TLVs

Specifications may allocate a new attribute of type TLV, and at the same time, allocate sub-attributes within that TLV. These specifications SHOULD request allocation of specific values for the sub-TLV. The "dotted number" notation MUST be used.

For example, a specification may request allocation of a TLV as SHORT-TBD1. Within that attribute, it could request allocation of three sub-TLVs, as SHORT-TBD1.1, SHORT-TBD1.2, and SHORT-TBD1.3.

Specifications may request allocation of additional sub-TLVs within an existing attribute of type TLV. Those specifications SHOULD use

the "TBDn" format for every entry in the "dotted number" notation.

For example, a specification may request allocation within an existing TLV, with "dotted number" notation MM.NN. Within that attribute, the specification could request allocation of three sub-TLVs, as MM.NN.TBD1, MM.NN.TBD2, and MM.NN.TBD3.

#### 6.8. Implementation Guidelines

- \* RADIUS client implementations SHOULD support this specification, in order to permit the easy deployment of specifications using the changes defined herein.
- \* RADIUS server implementations SHOULD support this specification, in order to permit the easy deployment of specifications using the changes defined herein.
- \* RADIUS proxy servers MUST follow the specifications in section 5.2

#### 6.9. Vendor Guidelines

- \* Vendors SHOULD use the existing Vendor-Specific Attribute Type space in preference to the new Extended-Vendor-Specific attributes, as this specification may take time to become widely deployed.
- \* Vendors SHOULD implement this specification. The changes to RADIUS are relatively small, and are likely to quickly be used in new specifications.

#### 7. Rationale for This Design

The path to extending the RADIUS protocol has been long and arduous. A number of proposals have been made and discarded by the RADEXT working group. These proposals have been judged to be either too bulky, too complex, too simple, or to be unworkable in practice. We do not otherwise explain here why earlier proposals did not obtain working group consensus.

The changes outlined here have the benefit of being simple, as the "Extended Type" format requires only a one octet change to the Attribute format. The downside is that the "Long Extended Type" format is awkward, and the 7 Reserved bits will likely never be used for anything.



### 7.1. Attribute Audit

An audit of almost five thousand publicly available attributes [ATTR] (2010), shows the statistics summarized below. The attributes include over 100 Vendor dictionaries, along with the IANA assigned attributes:

Count	Data Type
-----	-----
2257	integer
1762	text
273	IPv4 Address
225	string
96	other data types
35	IPv6 Address
18	date
10	integer64
4	Interface Id
3	IPv6 Prefix
4683	Total

The entries in the "Data Type" column are data types recommended by [RFC6158], along with "integer64". The "other data types" row encompasses all other data types, including complex data types and data types transporting opaque data.

We see that over half of the attributes encode less than 16 octets of data. It is therefore important to have an extension mechanism which adds as little as possible to the size of these attributes. Another result is that the overwhelming majority of attributes use simple data types.

Of the attributes defined above, 177 were declared as being inside of a TLV. This is approximately 4% of the total. We did not investigate whether additional attributes were defined in a flat name space, but could have been defined as being inside of a TLV. We expect that the number could be as high as 10% of attributes.

Manual inspection of the dictionaries shows that approximately 20 (or 0.5%) attributes have the ability to transport more than 253 octets of data. These attributes are divided between VSAs, and a small number of standard Attributes such as EAP-Message.

The results of this audit and analysis is reflected in the design of the extended attributes. The extended format has minimal overhead, it permits TLVs, and it has support for "long" attributes.

## 8. Diameter Considerations

The attribute formats defined in this specification need to be transported in Diameter. While Diameter supports attributes longer than 253 octets and grouped attributes, we do not use that functionality here. Instead, we define the simplest possible encapsulation method.

The new formats MUST be treated the same as traditional RADIUS attributes when converting from RADIUS to Diameter, or vice versa. That is, the new attribute space is not converted to any "extended" Diameter attribute space. Fragmented attributes are not converted to a single long Diameter attribute. The new EVS data types are not converted to Diameter attributes with the "V" bit set.

In short, this document mandates no changes for existing RADIUS to Diameter, or Diameter to RADIUS gateways.

## 9. Examples

A few examples are presented here in order to illustrate the encoding of the new attribute formats. These examples are not intended to be exhaustive, as many others are possible. For simplicity, we do not show complete packets, only attributes.

The examples are given using a domain-specific language implemented by the program given in Appendix A. The language is line oriented, and composed of a sequence of lines matching the grammar ([RFC5234]) given below:

```
Identifier = 1*DIGIT *( "." 1*DIGIT )

HEXCHAR = HEXDIG HEXDIG

STRING = DQUOTE 1*CHAR DQUOTE

TLV = "{" SP 1*DIGIT SP DATA SP "}"

DATA = (HEXCHAR *(SP HEXCHAR)) / (TLV *(SP TLV)) / STRING

LINE = Identifier SP DATA
```

The program has additional restrictions on its input that are not reflected in the above grammar. For example, the portions of the Identifier which refer to Type and Extended-Type are limited to values between 1 and 255. We trust that the source code in Appendix A is clear, and that these restrictions do not negatively affect the comprehensibility of the examples.

The program reads the input text, and interprets it as a set of instructions to create RADIUS Attributes. It then prints the hex encoding of those attributes. It implements the minimum set of functionality which achieves that goal. This minimalism means that it does not use attribute dictionaries; it does not implement support for RADIUS data types; it can be used to encode attributes with invalid data fields; and there is no requirement for consistency from one example to the next. For example, it can be used to encode a User-Name attribute which contains non-UTF8 data, or a Framed-IP-Address which contains 253 octets of ASCII data. As a result, it MUST NOT be used to create RADIUS Attributes for transport in a RADIUS message.

However, the program correctly encodes the RADIUS attribute fields of "Type", "Length", "Extended-Type", "More", "Reserved", "Vendor-Id", "Vendor-Type", and "Vendor-Length". It encodes RADIUS attribute data types "evs" and TLV. It can therefore be used to encode example attributes from inputs which are humanly readable.

We do not give examples of "malformed" or "invalid attributes". We also note that the examples show format, rather than consistent meaning. A particular Attribute Type code may be used to demonstrate two different formats. In real specifications, attributes have a static definitions based on their type code.

The examples given below are strictly for demonstration purposes only, and do not provide a standard of any kind.

#### 9.1. Extended Type

The following are a series of examples of the "Extended Type" format.

Attribute encapsulating textual data.

```
241.1 "bob"
-> f1 06 01 62 6f 62
```

Attribute encapsulating a TLV with TLV-Type of one (1).

```
241.2 { 1 23 45 }
-> f1 07 02 01 04 23 45
```

Attribute encapsulating two TLVs, one after the other.

```
241.2 { 1 23 45 } { 2 67 89 }
-> f1 0b 02 01 04 23 45 02 04 67 89
```

Attribute encapsulating two TLVs, where the second TLV is itself

encapsulating a TLV.

```
241.2 { 1 23 45 } { 3 { 1 ab cd } }  
-> f1 0d 02 01 04 23 45 03 06 01 04 ab cd
```

Attribute encapsulating two TLVs, where the second TLV is itself encapsulating two TLVs.

```
241.2 { 1 23 45 } { 3 { 1 ab cd } { 2 "foo" } }  
-> f1 12 02 01 04 23 45 03 0b 01 04 ab cd 02 05 66 6f 6f
```

Attribute encapsulating a TLV, which in turn encapsulates a TLV, to a depth of 5 nestings.

```
241.1 { 1 { 2 { 3 { 4 { 5 cd ef } } } } }  
-> f1 0f 01 01 0c 02 0a 03 08 04 06 05 04 cd ef
```

Attribute encapsulating an extended Vendor Specific attribute, with Vendor-Id of 1, and Vendor-Type of 4, which in turn encapsulates textual data.

```
241.26.1.4 "test"  
-> f1 0c 1a 00 00 00 01 04 74 65 73 74
```

Attribute encapsulating an extended Vendor Specific attribute, with Vendor-Id of 1, and Vendor-Type of 5, which in turn encapsulates a TLV with TLV-Type of 3, which encapsulates textual data.

```
241.26.1.5 { 3 "test" }  
-> f1 0e 1a 00 00 00 01 05 03 06 74 65 73 74
```

## 9.2. Long Extended Type

The following are a series of examples of the "Long Extended Type" format.

Attribute encapsulating textual data.

```
245.1 "bob"  
-> f5 07 01 00 62 6f 62
```

Attribute encapsulating a TLV with TLV-Type of one (1).

```
245.2 { 1 23 45 }  
-> f5 08 02 00 01 04 23 45
```

Attribute encapsulating two TLVs, one after the other.



```

          ccccccccccc"
-> f5 ff 04 80 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa
    aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa
    aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa
    aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa
    aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa
    aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa
    aa aa aa aa aa aa aa aa aa aa ab bb bb bb bb bb bb bb bb bb
    bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb
    bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb
    bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb
    bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb
    bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb
    bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb
    cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc

```

Attribute encapsulating an extended Vendor Specific attribute, with Vendor-Id of 1, and Vendor-Type of 6, which in turn encapsulates more than 251 octets of data.

As the VSA encapsulates more than 251 octets of data, it is split into two RADIUS attributes. The first attribute has the More field set, and carries the Vendor-Id and Vendor-Type. The second attribute has the More field clear, and carries the rest of the data portion of the VSA. Note that the second attribute does not include the Vendor-Id and Vendor-Type fields.

The "Data" portions are indented for readability.

```

245.26.1.6 "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
            aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
            aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
            aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
            aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbb
            bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
            bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
            bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
            bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
            bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbccccc
            ccccccccccccccc"
-> f5 ff 1a 80 00 00 00 01 06 aa aa aa aa aa aa aa aa aa aa aa aa
    aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa
    aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa
    aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa
    aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa
    aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa ab bb bb bb bb
    bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb
    bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb

```

```

bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb
bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb
bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb
bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb f5 18 1a 00 bb
bb bb bb bb cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc

```

## 10. IANA Considerations

This document updates [RFC3575] in that it adds new IANA considerations for RADIUS Attributes. These considerations modify and extend the IANA considerations for RADIUS, rather than replacing them.

The IANA considerations of this document are limited to the "RADIUS Attribute Types" registry. Some Attribute Type values which were previously marked "Reserved" are now allocated, and the registry is extended from a simple 8-bit array to a tree-like structure, up to a maximum depth of 125 nodes. Detailed instructions are given below.

### 10.1. Attribute Allocations

IANA is requested to move the following Attribute Type values from "Reserved", to "Allocated", with the corresponding names:

- \* 241 Extended-Type-1
- \* 242 Extended-Type-2
- \* 243 Extended-Type-3
- \* 244 Extended-Type-4
- \* 245 Long-Extended-Type-1
- \* 246 Long-Extended-Type-2

These values serve as an encapsulation layer for the new RADIUS Attribute Type tree.

### 10.2. RADIUS Attribute Type Tree

Each of the Attribute Type values allocated above extends the "RADIUS Attribute Types" to an N-ary tree, via a "dotted number" notation. Allocation of an Attribute Type value "TYPE" using the new Extended type format results in allocation of 255 new Attribute Type values, of format "TYPE.1" through "TYPE.255". Value twenty-six (26) is assigned as "Extended-Vendor-Specific-\*". Values "TYPE.241" through "TYPE.255" are marked "Reserved". All other values are "Unassigned".

The initial set of Attribute Type values and names assigned by this document is given below.

* 241	Extended-Attribute-1
* 241.{1-25}	Unassigned
* 241.26	Extended-Vendor-Specific-1
* 241.{27-240}	Unassigned
* 241.{241-255}	Reserved
* 242	Extended-Attribute-2
* 242.{1-25}	Unassigned
* 242.26	Extended-Vendor-Specific-2
* 242.{27-240}	Unassigned
* 243	Extended-Attribute-3
* 242.{241-255}	Reserved
* 243.{1-25}	Unassigned
* 243.26	Extended-Vendor-Specific-3
* 243.{27-240}	Unassigned
* 243.{241-255}	Reserved
* 244	Extended-Attribute-4
* 244.{1-25}	Unassigned
* 244.26	Extended-Vendor-Specific-4
* 244.{27-240}	Unassigned
* 244.{241-255}	Reserved
* 245	Extended-Attribute-5
* 245.{1-25}	Unassigned
* 245.26	Extended-Vendor-Specific-5
* 245.{27-240}	Unassigned
* 245.{241-255}	Reserved
* 246	Extended-Attribute-6
* 246.{1-25}	Unassigned
* 245.26	Extended-Vendor-Specific-6
* 246.{27-240}	Unassigned
* 246.{241-255}	Reserved

As per [RFC5226], the values marked "Unassigned" above are available via for assignment by IANA in future RADIUS specifications. The values marked "Reserved" are reserved for future use.

The Extended-Vendor-Specific spaces (TYPE.26) are for Private Use, and allocations are not managed by IANA.

Allocation of Reserved entries in the extended space requires Standards Action.

All other allocations in the extended space require IETF Review.

### 10.3. Allocation Instructions

This section defines what actions IANA needs to take when allocating new attributes. Different actions are required when allocating attributes from the standard space, attributes of Extended Type



format, attributes of the "Long Extended Type" format, preferential allocations, attributes of data type TLV, attributes within a TLV, and attributes of other data types.

#### 10.3.1. Requested Allocation from the Standard Space

Specifications can request allocation of an Attribute from within the standard space (e.g. Attribute Type Codes 1 through 255), subject to the considerations of [RFC3575] and this document.

#### 10.3.2. Requested Allocation from the short extended space

Specifications can request allocation of an Attribute which requires the format Extended Type, by specifying the short extended space. In that case, IANA should assign the lowest Unassigned number from the Attribute Type space with the relevant format.

#### 10.3.3. Requested Allocation from the long extended space

Specifications can request allocation of an Attribute which requires the format "Long Extended Type", by specifying the extended space (long). In that case, IANA should assign the lowest Unassigned number from the Attribute Type space with the relevant format.

#### 10.3.4. Allocation Preferences

Specifications which make no request for allocation from a specific Type Space should have Attributes allocated using the following criteria:

- \* when the standard space has no more Unassigned attributes, all allocations should be performed from the extended space.
- \* specifications which allocate a small number of attributes (i.e. less than ten) should have all allocations made from the standard space.
- \* specifications which would allocate a more than twenty percent of the remaining standard space attributes should have all allocations made from the extended space.
- \* specifications which request allocation of an attribute of data type TLV should have that attribute allocated from the extended space.
- \* specifications which request allocation of an attribute which can transport 253 or more octets of data should have

that attribute allocated from within the long extended space,  
We note that Section 6.5, above requires specifications to  
request this allocation.

There is otherwise no requirement that all attributes within a  
specification be allocated from one type space or another.  
Specifications can simultaneously allocate attributes from both the  
standard space and the extended space.

#### 10.3.5. Extending the Type Space via TLV Data Type

When specifications request allocation of an attribute of data type  
"tlv", that allocation extends the Attribute Type Tree by one more  
level. Allocation of an Attribute Type value "TYPE.TLV", with Data  
Type TLV, results in allocation of 255 new Attribute Type values, of  
format "TYPE.TLV.1" through "TYPE.TLV.255". Values 254-255 are  
marked "Reserved". All other values are "Unassigned". Value 26 has  
no special meaning.

For example, if a new attribute "Example-TLV" of data type "tlv" is  
assigned the identifier "245.1", then the extended tree will be  
allocated as below:

```
* 245.1           Example-TLV
* 245.1.{1-253}   Unassigned
* 245.1.{254-255} Reserved
```

Note that this example does not define an "Example-TLV" attribute.

The Attribute Type Tree can be extended multiple levels in one  
specification when the specification requests allocation of nested  
TLVs, as discussed below.

#### 10.3.6. Allocation within a TLV

Specifications can request allocation of Attribute Type values within  
an Attribute of Data Type TLV. The encapsulating TLV can be  
allocated in the same specification, or it can have been previously  
allocated.

Specifications need to request allocation within a specific Attribute  
Type value (e.g. "TYPE.TLV.\*"). Allocations are performed from the  
smallest Unassigned value, proceeding to the largest Unassigned  
value.

Where the Attribute being allocated is of Data Type TLV, the  
Attribute Type tree is extended by one level, as given in the

previous section. Allocations can then be made within that level.

#### 10.3.7. Allocation of Other Data Types

Attribute Type value allocations are otherwise allocated from the smallest Unassigned value, proceeding to the largest Unassigned value. e.g. Starting from 241.1, proceeding through 241.255, then to 242.1, through 242.255, etc.

### 11. Security Considerations

This document defines new formats for data carried inside of RADIUS, but otherwise makes no changes to the security of the RADIUS protocol.

Attacks on cryptographic hashes are well known, and are getting better with time, as discussed in [RFC4270]. The security of the RADIUS protocol is dependent on MD5 [RFC1311], which has security issues as discussed in [RFC6151]. It is not known if the issues described in [RFC6151] apply to RADIUS. For other issues, we incorporate by reference the security considerations of [RFC6158] Section 5.

As with any protocol change, code changes are required in order to implement the new features. These code changes have the potential to introduce new vulnerabilities in the software. Since the RADIUS server performs network authentication, it is an inviting target for attackers. We RECOMMEND that access to RADIUS servers be kept to a minimum.

### 12. References

#### 12.1. Normative references

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March, 1997.
- [RFC2865] Rigney, C., Willens, S., Rubens, A. and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.
- [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.
- [RFC5176] Chiba, M, et. al., "Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS)", RFC 5176,

January 2008.

[RFC5226] Narten, T. and Alvestrand, H, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, May 2008.

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

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

## 12.2. Informative references

[RFC1321] Rivest, R. "The MD5 Message-Digest Algorithm", RFC 1321, April, 1992

[RFC2868] Zorn, G., et al, " RADIUS Attributes for Tunnel Protocol Support", RFC 2868, June 2000.

[RFC4270] Hoffman, P, and Schneier, B, "Attacks on Cryptographic Hashes in Internet Protocols", RFC 4270, November 2005.

[RFC5234] Crocker, D. (Ed.), and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", RFC 5234, October 2005.

[RFC6151] Turner, S. and Chen, L., "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, March 2011.

[EDUROAM] Internal Eduroam testing page, data retrieved 04 August 2010.

[ATTR] <http://github.com/alandekok/freeradius-server/tree/master/share/>, data retrieved September 2010.

## Acknowledgments

This document is the result of long discussions in the IETF RADEXT working group. The authors would like to thank all of the participants who contributed various ideas over the years. Their feedback has been invaluable, and has helped to make this specification better.

## Appendix A - Extended Attribute Generator Program

This section contains "C" program source which can be used for testing. It reads a line-oriented text file, parses it to create RADIUS formatted attributes, and prints the hex version of those attributes to standard output.

The input accepts a grammar similar to that given in Section 9, with some modifications for usability. For example, blank lines are allowed, lines beginning with a '#' character are interpreted as comments, numbers (RADIUS Types, etc.) are checked for minimum / maximum values, and RADIUS Attribute lengths are enforced.

The program is included here for demonstration purposes only, and does not define a standard of any kind.

```
-----
/*
 * Copyright (c) 2010 IETF Trust and the persons identified as
 * authors of the code. All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * Redistributions of source code must retain the above copyright
 * notice, this list of conditions and the following disclaimer.
 *
 * Redistributions in binary form must reproduce the above copyright
 * notice, this list of conditions and the following disclaimer in
 * the documentation and/or other materials provided with the
 * distribution.
 *
 * Neither the name of Internet Society, IETF or IETF Trust, nor the
 * names of specific contributors, may be used to endorse or promote
 * products derived from this software without specific prior written
 * permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
 * CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
 * MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS
 * BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED
 * TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
 * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
 * ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
```

```
* OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
* OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
*
* Author: Alan DeKok <aland@networkradius.com>
*/
#include <stdlib.h>
#include <stdio.h>
#include <stdint.h>
#include <string.h>
#include <errno.h>
#include <ctype.h>

static int encode_tlv(char *buffer, uint8_t *output, size_t outlen);

static const char *hextab = "0123456789abcdef";

static int encode_data_string(char *buffer,
                              uint8_t *output, size_t outlen)
{
    int length = 0;
    char *p;

    p = buffer + 1;

    while (*p && (outlen > 0)) {
        if (*p == '"') {
            return length;
        }

        if (*p != '\\') {
            *(output++) = *(p++);
            outlen--;
            length++;
            continue;
        }

        switch (p[1]) {
        default:
            *(output++) = p[1];
            break;

        case 'n':
            *(output++) = '\n';
            break;

        case 'r':
            *(output++) = '\r';
```

```
        break;

    case 't':
        *(output++) = '\t';
        break;
    }

    outlen--;
    length++;
}

fprintf(stderr, "String is not terminated\n");
return 0;
}

static int encode_data_tlv(char *buffer, char **endp,
                           uint8_t *output, size_t outlen)
{
    int depth = 0;
    int length;
    char *p;

    for (p = buffer; *p != '\0'; p++) {
        if (*p == '{') depth++;
        if (*p == '}') {
            depth--;
            if (depth == 0) break;
        }
    }

    if (*p != '}') {
        fprintf(stderr, "No trailing '}' in string starting "
            "with \"%s\"\n",
            buffer);
        return 0;
    }

    *endp = p + 1;
    *p = '\0';

    p = buffer + 1;
    while (isspace((int) *p)) p++;

    length = encode_tlv(p, output, outlen);
    if (length == 0) return 0;

    return length;
}
```

```
static int encode_data(char *p, uint8_t *output, size_t outlen)
{
    int length;

    if (!isspace((int) *p)) {
        fprintf(stderr, "Invalid character following attribute "
            "definition\n");
        return 0;
    }

    while (isspace((int) *p)) p++;

    if (*p == '{') {
        int sublen;
        char *q;

        length = 0;

        do {
            while (isspace((int) *p)) p++;
            if (!*p) {
                if (length == 0) {
                    fprintf(stderr, "No data\n");
                    return 0;
                }
                break;
            }

            sublen = encode_data_tlv(p, &q, output, outlen);
            if (sublen == 0) return 0;

            length += sublen;
            output += sublen;
            outlen -= sublen;
            p = q;
        } while (*q);

        return length;
    }

    if (*p == '"') {
        length = encode_data_string(p, output, outlen);
        return length;
    }

    length = 0;
    while (*p) {
```



```
    char *c1, *c2;

    while (isspace((int) *p)) p++;

    if (!*p) break;

    if(!(c1 = memchr(hextab, tolower((int) p[0]), 16)) ||
        !(c2 = memchr(hextab, tolower((int) p[1]), 16))) {
        fprintf(stderr, "Invalid data starting at "
            "\"%s\"\n", p);
        return 0;
    }

    *output = ((c1 - hextab) << 4) + (c2 - hextab);
    output++;
    length++;
    p += 2;

    outlen--;
    if (outlen == 0) {
        fprintf(stderr, "Too much data\n");
        return 0;
    }
}

if (length == 0) {
    fprintf(stderr, "Empty string\n");
    return 0;
}

return length;
}

static int decode_attr(char *buffer, char **endptr)
{
    long attr;

    attr = strtoul(buffer, endptr, 10);
    if (*endptr == buffer) {
        fprintf(stderr, "No valid number found in string "
            "starting with \"%s\"\n", buffer);
        return 0;
    }

    if (**endptr) {
        fprintf(stderr, "Nothing follows attribute number\n");
        return 0;
    }
}
```

```
    if ((attr <= 0) || (attr > 256)) {
        fprintf(stderr, "Attribute number is out of valid "
            "range\n");
        return 0;
    }

    return (int) attr;
}

static int decode_vendor(char *buffer, char **endptr)
{
    long vendor;

    if (*buffer != '.') {
        fprintf(stderr, "Invalid separator before vendor id\n");
        return 0;
    }

    vendor = strtol(buffer + 1, endptr, 10);
    if (*endptr == (buffer + 1)) {
        fprintf(stderr, "No valid vendor number found\n");
        return 0;
    }

    if (!**endptr) {
        fprintf(stderr, "Nothing follows vendor number\n");
        return 0;
    }

    if ((vendor <= 0) || (vendor > (1 << 24))) {
        fprintf(stderr, "Vendor number is out of valid range\n");
        return 0;
    }

    if (**endptr != '.') {
        fprintf(stderr, "Invalid data following vendor number\n");
        return 0;
    }
    (*endptr)++;

    return (int) vendor;
}

static int encode_tlv(char *buffer, uint8_t *output, size_t outlen)
{
    int attr;
    int length;
    char *p;
```

```
    attr = decode_attr(buffer, &p);
    if (attr == 0) return 0;

    output[0] = attr;
    output[1] = 2;

    if (*p == '.') {
        p++;
        length = encode_tlv(p, output + 2, outlen - 2);
    } else {
        length = encode_data(p, output + 2, outlen - 2);
    }

    if (length == 0) return 0;
    if (length > (255 - 2)) {
        fprintf(stderr, "TLV data is too long\n");
        return 0;
    }

    output[1] += length;

    return length + 2;
}

static int encode_vsa(char *buffer, uint8_t *output, size_t outlen)
{
    int vendor;
    int attr;
    int length;
    char *p;

    vendor = decode_vendor(buffer, &p);
    if (vendor == 0) return 0;

    output[0] = 0;
    output[1] = (vendor >> 16) & 0xff;
    output[2] = (vendor >> 8) & 0xff;
    output[3] = vendor & 0xff;

    length = encode_tlv(p, output + 4, outlen - 4);
    if (length == 0) return 0;
    if (length > (255 - 6)) {
        fprintf(stderr, "VSA data is too long\n");
        return 0;
    }
}
```

```
        return length + 4;
    }

static int encode_evs(char *buffer, uint8_t *output, size_t outlen)
{
    int vendor;
    int attr;
    int length;
    char *p;

    vendor = decode_vendor(buffer, &p);
    if (vendor == 0) return 0;

    attr = decode_attr(p, &p);
    if (attr == 0) return 0;

    output[0] = 0;
    output[1] = (vendor >> 16) & 0xff;
    output[2] = (vendor >> 8) & 0xff;
    output[3] = vendor & 0xff;
    output[4] = attr;

    length = encode_data(p, output + 5, outlen - 5);
    if (length == 0) return 0;

    return length + 5;
}

static int encode_extended(char *buffer,
                           uint8_t *output, size_t outlen)
{
    int attr;
    int length;
    char *p;

    attr = decode_attr(buffer, &p);
    if (attr == 0) return 0;

    output[0] = attr;

    if (attr == 26) {
        length = encode_evs(p, output + 1, outlen - 1);
    } else {
        length = encode_data(p, output + 1, outlen - 1);
    }
    if (length == 0) return 0;
    if (length > (255 - 3)) {
        fprintf(stderr, "Extended Attr data is too long\n");
    }
}
```

```
        return 0;
    }

    return length + 1;
}

static int encode_extended_flags(char *buffer,
                                uint8_t *output, size_t outlen)
{
    int attr;
    int length, total;
    char *p;

    attr = decode_attr(buffer, &p);
    if (attr == 0) return 0;

    /* output[0] is the extended attribute */
    output[1] = 4;
    output[2] = attr;
    output[3] = 0;

    if (attr == 26) {
        length = encode_evs(p, output + 4, outlen - 4);
        if (length == 0) return 0;

        output[1] += 5;
        length -= 5;
    } else {
        length = encode_data(p, output + 4, outlen - 4);
    }
    if (length == 0) return 0;

    total = 0;
    while (1) {
        int sublen = 255 - output[1];

        if (length <= sublen) {
            output[1] += length;
            total += output[1];
            break;
        }

        length -= sublen;

        memmove(output + 255 + 4, output + 255, length);
        memcpy(output + 255, output, 4);

        output[1] = 255;
    }
}
```

```
        output[3] |= 0x80;

        output += 255;
        output[1] = 4;
        total += 255;
    }

    return total;
}

static int encode_rfc(char *buffer, uint8_t *output, size_t outlen)
{
    int attr;
    int length, sublen;
    char *p;

    attr = decode_attr(buffer, &p);
    if (attr == 0) return 0;

    length = 2;
    output[0] = attr;
    output[1] = 2;

    if (attr == 26) {
        sublen = encode_vsa(p, output + 2, outlen - 2);
    } else if ((*p == ' ') || ((attr < 241) || (attr > 246))) {
        sublen = encode_data(p, output + 2, outlen - 2);
    } else {
        if (*p != '.') {
            fprintf(stderr, "Invalid data following "
                        "attribute number\n");
            return 0;
        }

        if (attr < 245) {
            sublen = encode_extended(p + 1,
                                     output + 2, outlen - 2);
        } else {
            /*
             *   Not like the others!
             */
            return encode_extended_flags(p + 1, output, outlen);
        }
    }
    if (sublen == 0) return 0;
}
```

```
        if (sublen > (255 - 2)) {
            fprintf(stderr, "RFC Data is too long\n");
            return 0;
        }

        output[1] += sublen;
        return length + sublen;
    }

int main(int argc, char *argv[])
{
    int lineno;
    size_t i, outlen;
    FILE *fp;
    char input[8192], buffer[8192];
    uint8_t output[4096];

    if ((argc < 2) || (strcmp(argv[1], "-") == 0)) {
        fp = stdin;
    } else {
        fp = fopen(argv[1], "r");
        if (!fp) {
            fprintf(stderr, "Error opening %s: %s\n",
                    argv[1], strerror(errno));
            exit(1);
        }
    }

    lineno = 0;
    while (fgets(buffer, sizeof(buffer), fp) != NULL) {
        char *p = strchr(buffer, '\n');

        lineno++;

        if (!p) {
            if (!feof(fp)) {
                fprintf(stderr, "Line %d too long in %s\n",
                        lineno, argv[1]);
                exit(1);
            }
        } else {
            *p = '\0';
        }

        p = strchr(buffer, '#');
        if (p) *p = '\0';

        p = buffer;
```

```
while (isspace((int) *p)) p++;
if (!*p) continue;

strcpy(input, p);
outlen = encode_rfc(input, output, sizeof(output));
if (outlen == 0) {
    fprintf(stderr, "Parse error in line %d of %s\n",
        lineno, input);
    exit(1);
}

printf("%s -> ", buffer);
for (i = 0; i < outlen; i++) {
    printf("%02x ", output[i]);
}
printf("\n");
}

if (fp != stdin) fclose(fp);

return 0;
}
```

-----

## Author's Address

Alan DeKok  
Network RADIUS SARL  
57bis blvd des Alpes  
38240 Meylan  
France

Email: [aland@networkradius.com](mailto:aland@networkradius.com)  
URI: <http://networkradius.com>

Avi Lior  
Email: [avi.ietf@lior.org](mailto:avi.ietf@lior.org)





RADIUS Extensions Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: January 17, 2013

S. Winter  
RESTENA  
July 16, 2012

RADIUS Accounting for traffic classes  
draft-winter-radext-fancyaccounting-02

Abstract

This document specifies new attributes for RADIUS Accounting to enable NAS reporting of subsets of the total traffic in a user session.

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 17, 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. Requirements Language . . . . .	3
2. Definitions . . . . .	3
2.1. Acct-Traffic-Class attribute . . . . .	3
2.1.1. Acct-Traffic-Class-Name attribute . . . . .	4
2.1.2. Acct-Traffic-Class-Input-Octets attribute . . . . .	5
2.1.3. Acct-Traffic-Class-Output-Octets attribute . . . . .	5
2.1.4. Acct-Traffic-Class-Input-Packets attribute . . . . .	5
2.1.5. Acct-Traffic-Class-Output-Packets attribute . . . . .	6
2.2. URN values for attribute Acct-Traffic-Class-Name . . . . .	6
3. Example . . . . .	7
4. Attribute Occurrence Table . . . . .	8
5. Security Considerations . . . . .	9
6. IANA Considerations . . . . .	9
7. Normative References . . . . .	9

## 1. Introduction

RADIUS Accounting [RFC2866] defines counters for octets and packets, both in the incoming and outgoing direction. Usage of these counters enables an operator create volume-based billing models and to execute proper capacity planning on its infrastructure.

The Accounting model is based on the assumption that all traffic in a user session is treated equally; i.e. that there are no differences in the billing model of one class of traffic over another.

Actual deployments suggest that this assumption is no longer valid. In particular, different traffic classes are defined with DSCP; and billing the use of these traffic classes separately is an understandable request.

Plus, the introduction of dual-stack operation on links creates an understandable interest of getting separate statistics about the amount of IPv4 vs. IPv6 usage on a link; be it for billing or statistical reasons.

This document defines Accounting attributes that supplement (but not replace) the accounting counters in RFC2866. It utilizes the new "extended attributes" in RADIUS ([I-D.ietf-radext-radius-extensions]) to a) group accounting reports about traffic classes together and b) enable 64-Bit counts in a single attribute with the Integer64 datatype.

### 1.1. Requirements Language

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

## 2. Definitions

### 2.1. Acct-Traffic-Class attribute

The attribute Acct-Traffic-Class is a TLV container for a group of sub-attributes which specify the class of traffic that is being reported about, and the amount of traffic in a user session that falls into this class.

Attribute: 245.1 Acct-Traffic-Class

Type: TLV

Length: >3 octets

There can be multiple instances of this attribute in a Accounting-Interim-Update or a Accounting-Stop packet. The attribute MUST NOT be present in an Accounting-Start packet.

It is not required that the sum of all traffic in all instances is the total sum of octets and packets in the user's session. I.e. the traffic classes used in the Accounting packet do not need to partition the total traffic in non-overlapping segments.

The total number of octets and packets in a user session continues to be sent in the RFC2866 attributes.

#### 2.1.1.1. Acct-Traffic-Class-Name attribute

The attribute Acct-Traffic-Class-Name, sub-attribute in the group Acct-Traffic-Class, defines the class of traffic for which the other attributes in the instance of Acct-Traffic-Class count octets and packets. Every group instance MUST contain exactly one Acct-Traffic-Class-Name.

Attribute: 245.1.2 Acct-Traffic-Class-Name

Type: STRING

Value: 1-250 octets

There are two options for the value of this attribute.

Option 1: Acct-Traffic-Class-Name string starting with the substring "urn:". Usage of this option implies that the traffic name is in the form of a URN and requires that a public specification of this URN exists. That specification must include the type of traffic being counted with this traffic class, and the exact definition of where in the network packets the byte-count starts and ends. This document defines a set of known, well-defined traffic accounting classes in an IANA-managed registry in Section 2.2. New values for this registry are assigned on expert review basis.

Option 2: Acct-Traffic-Class-Name string not starting with "urn:". This option is for local use of special-purpose accounting as defined by the NAS administrator, where no defined URN matches the meaning of the traffic to be counted. The meaning of the content needs to be communicated out-of-band between the NAS and RADIUS Server operator. Example: Acct-Traffic-Class-Name = "UDP traffic to AS2606".

#### 2.1.2. Acct-Traffic-Class-Input-Octets attribute

The attribute Acct-Traffic-Class-Input-Octets, sub-attribute in the group Acct-Traffic-Class, carries the number of octets that belong to the class of traffic indicated by Acct-Traffic-Class-Name and have been sent to the entity for which the accounting packet was generated. It MUST occur at most once inside every instance of the Acct-Traffic-Class TLV. If a traffic parameter value is transmitted in this attribute in an Accounting-Request "Interim Update", then the final value of that traffic parameter MUST be reported in the corresponding Accounting-Request "Stop".

Attribute: 245.1.3 Acct-Traffic-Class-Input-Octets

Type: Integer64

Value: number of octets sent to entity, matching the class of traffic

#### 2.1.3. Acct-Traffic-Class-Output-Octets attribute

The attribute Acct-Traffic-Class-Output-Octets, sub-attribute in the group Acct-Traffic-Class, carries the number of octets that belong to the class of traffic indicated by Acct-Traffic-Class-Name and have been sent from the entity for which the accounting packet was generated. It MUST occur at most once inside every instance of the Acct-Traffic-Class TLV. If a traffic parameter value is transmitted in this attribute in an Accounting-Request "Interim Update", then the final value of that traffic parameter MUST be reported in the corresponding Accounting-Request "Stop".

Attribute: 245.1.4 Acct-Traffic-Class-Output-Octets

Type: Integer64

Value: number of octets sent from entity, matching the class of traffic

#### 2.1.4. Acct-Traffic-Class-Input-Packets attribute

The attribute Acct-Traffic-Class-Input-Packets, sub-attribute in the group Acct-Traffic-Class, carries the number of packets that belong to the class of traffic indicated by Acct-Traffic-Class-Name and have been sent to the entity for which the accounting packet was generated. It MUST occur at most once inside every instance of the Acct-Traffic-Class TLV. If a traffic parameter value is transmitted in this attribute in an Accounting-Request "Interim Update", then the final value of that traffic parameter MUST be reported in the

corresponding Accounting-Request "Stop".

Attribute: 245.1.5 Acct-Traffic-Class-Input-Packets

Type: Integer64

Value: number of packets sent to entity, matching the class of traffic

#### 2.1.5. Acct-Traffic-Class-Output-Packets attribute

The attribute Acct-Traffic-Class-Output-Packets, sub-attribute in the group Acct-Traffic-Class, carries the number of packets that belong to the class of traffic indicated by Acct-Traffic-Class-Name and have been sent from the entity for which the accounting packet was generated. It MUST occur at most once inside every instance of the Acct-Traffic-Class TLV. If a traffic parameter value is transmitted in this attribute in an Accounting-Request "Interim Update", then the final value of that traffic parameter MUST be reported in the corresponding Accounting-Request "Stop".

Attribute: 245.1.6 Acct-Traffic-Class-Output-Packets

Type: Integer64

Value: number of packets sent from entity, matching the class of traffic

#### 2.2. URN values for attribute Acct-Traffic-Class-Name

The following URN values are defined for RADIUS Accounting Traffic Classes:

Name: "urn:ietf:radius-accounting:ip:4"

Purpose: volume count of IPv4 payloads

Start of byte count: 1st byte of the IP header of the packet

End of byte count: last byte of IP layer of the packet

Name: "urn:ietf:radius-accounting:ip:6"

Purpose: volume count of IPv6 payloads

Start of byte count: 1st byte of the IP header of the packet

End of byte count: last byte of IP layer of the packet

Name: "urn:ietf:radius-accounting:dscp:0"

Purpose: volume count of packet payloads with DSCP = 0

Start of byte count: 1st byte of the IP header of the packet

End of byte count: last byte of IP layer of the packet

Name: "urn:ietf:radius-accounting:tcp"

Purpose: volume count of TCP packets

Start of byte count: 1st byte of the TCP header of the packet

End of byte count: last byte of TCP layer of the packet

Name: "urn:ietf:radius-accounting:udp"

Purpose: volume count of UDP payloads

Start of byte count: 1st byte of the UDP header of the packet

End of byte count: last byte of UDP layer of the packet

(more values to be added...)

### 3. Example

A NAS is configured to create statistics regarding IPv6 usage of CPE for statistical reasons, and of the amount of HTTP traffic sent to the example.com web site for billing reasons.

User john@example.com starts a user session, transfers 1200 Bytes in 10 packets via IPv6 to the internet, and receives 4500 Bytes in 30 packets over IPv6 from the internet.

In the same session, The user visits the IPv4-only example.com web site by sending 6000 bytes in 4 packets to the web site, and receiving 450000 Bytes in 35 packets from the web site.

Then, the user terminates the session and an Accounting-Stop packet is generated.

The NAS sends the recorded octet and packet values to his RADIUS Accounting server. Since there is no URN value for "Traffic on TCP/80 to example.com, all IP versions" for use in the Acct-Traffic-



Class-Name attribute, the NAS has been configured to indicate this class of traffic in a corresponding custom string. The relevant attributes in the Accounting-Stop packet are:

#### Acct-Traffic-Class

Acct-Traffic-Class-Name = "urn:ietf:radius-accounting:ip:6"

Acct-Traffic-Class-Input-Octets = 4500

Acct-Traffic-Class-Output-Octets = 1200

Acct-Traffic-Class-Input-Packets = 30

Acct-Traffic-Class-Output-Packets = 10

#### Acct-Traffic-Class

Acct-Traffic-Class-Name = "Traffic on TCP/80 to example.com, all IP versions"

Acct-Traffic-Class-Input-Octets = 450000

Acct-Traffic-Class-Output-Octets = 6000

Acct-Traffic-Class-Input-Packets = 35

Acct-Traffic-Class-Output-Packets = 4

#### 4. Attribute Occurrence Table

This table lists the allowed occurrences of the previously defined attributes in Accounting packets.

Start	Interim	Stop	Reply	Attribute
----	-----	----	-----	-----
0	0-n	0-s	0	Acct-Traffic-Class
0	0-m	0-t	0	Acct-Traffic-Class-Name
0	0-o	0-u	0	Acct-Traffic-Class-Input-Octets
0	0-p	0-v	0	Acct-Traffic-Class-Output-Octets
0	0-q	0-w	0	Acct-Traffic-Class-Input-Packets
0	0-r	0-x	0	Acct-Traffic-Class-Output-Packets

Figure 1: Attribute Occurrence

Note 1: since all sub-attributes occur at most once inside any given Acct-Traffic-Class TLV, the sub-attributes can not occur more often than the TLV itself. I.e.  $m < n$ ,  $o < n$ ,  $p < n$ ,  $q < n$ , and  $r < n$ .

Note 2: if Acct-Traffic-Class TLVs and their sub-attributes have been sent in Interim-Updates, they MUST also occur in the subsequent Stop packet; while the Stop packet MAY contain additional Acct-Traffic-Class instances. I.e. in the table above: s>=n, t>=m, u>=o, v>=p, w>=q, and x>=r.

## 5. Security Considerations

Reveals user's traffic usage patterns. Shouldn't be sent unencryptedly.

## 6. IANA Considerations

This document has actions for IANA. TBD later.

## 7. Normative References

- |                                     |   |
|-------------------------------------|---|
| [RFC2866]                           | Rigney, C., "RADIUS Accounting", RFC 2866, June 2000.   |
| [RFC2119]                           | Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.  |
| [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. |

## Author's Address

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>



Radext Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: May 3, 2012

L. Yeh, Ed.  
Huawei Technologies  
October 31, 2011

RADIUS Accounting Extensions of Traffic Statistics  
draft-yeh-radext-ext-traffic-statistics-01

Abstract

This document specifies the RADIUS attributes extensions of IPv4 and IPv6 traffic statistics for the differentiated accounting policies and traffic recording on the AAA server.

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 May 3, 2012.

Copyright Notice

Copyright (c) 2011 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
2. Terminology and Language . . . . .	5
3. Deployment Scenarios . . . . .	5
4. Traffic Statistics Attributes . . . . .	5
4.1. Define the Attributes in the Traditional Unsigned Type Space . . . . .	5
4.2. Define the Attributes in the Extended Type Space . . . . .	7
5. Table of Attributes . . . . .	8
6. Security Considerations . . . . .	9
7. IANA Considerations . . . . .	9
8. Acknowledgements . . . . .	9
9. References . . . . .	9
9.1. Normative References . . . . .	9
9.2. Informative References . . . . .	10
Author's Address . . . . .	11

## 1. Introduction

RADIUS has been widely used as the centralized Authentication and Authorization management method for the service provision to the users in Broadband network. [RFC3162], [RFC4818] and [ietf-radext-ipv6-access-05] has specified some attributes to support the service provision of IPv6-only and dual-stack. Radius is also a protocol for carrying accounting information between a Network Access Server and a shared accounting server. In the scenarios of dual-stack or any other IPv6 transition use case, such as DS-Lite, 6rd or the potential 4rd, there is a demand to report the separated IPv4 & IPv6 traffic statistics for the differential accounting and traffic recording.

[BBF TR-187] (Edited by ALU & Cisco), which dedicates for the network architecture models and elements requirements in the PPPoE scenario to support IPv6-only or dual stack for Internet access service, has expressed this demand in its section 9.4. The explicit texts are as follows:

The BNG must also be able to support separate queues for IPv4 and IPv6 traffic, as they may be used to offer IPv4 and IPv6 services with different policies.

Note that BNG of BBF is a kind of NAS of IETF.

R-60 The BNG MUST support forwarding IPv6 and IPv4 traffic in common traffic classes.

R-61 The BNG MUST support forwarding IPv6 and IPv4 traffic in separate traffic classes.

R-64 The BNG MUST support input and output octet counters that are separate for both IPv6 and IPv4 traffic.

R-65 The BNG MUST support input and output packet counters that are separate for both IPv6 and IPv4 traffic.

Per the section 9.4 of BBF TR-187, the NAS is required to support separate queues and counters for IPv4 or IPv6 traffic, and the Radius attributes of Acct-Input-Octets, Acct-Output-Octets, Acct-Input-Packets, Acct-Output-Packets are recommended to use for the combination traffic. That means some new RADIUS attributes is required to report the separated IPv4 or IPv6 traffic statistics.

[draft-maglione-radext-ipv6-acct-extensions-01] (Edited by Telecom Italia, Ericsson & Magyar Telekom) tries to define the following attributes:

IPv6-Acct-Input-Octets  
IPv6-Acct-Output-Octets  
IPv6-Acct-Input-Packets  
IPv6-Acct-Output-Packets  
IPv6-Acct-Input-Gigawords  
IPv6-Acct-Output-Gigawords

for the collecting of IPv6 traffic statistics in RADIUS accounting messages. [draft-hu-v6ops-radius-issues-ipv6-00] (Edited by China Telecom & ZTE) presents the same issue on the accounting for dual-stack traffic statistics, but it sounds like limit to the PPP case. [draft-winter-radext-fancyaccounting-00] also shows the interest to define a group of attributes to report the statistics for various traffic classes, but tries to use the extended type space. And [draft-yeh-radext-dual-stack-access-02] (Edited Huawei) tries to use the traditional format defined in [RFC2865], [RFC2866] and [RFC2869] to extend some new attributes:

Acct-Input-IPv4-Octets  
Acct-Output-IPv4-Octets  
Acct-Input-IPv4-Packets  
Acct-Output-IPv4-Packets  
Acct-Input-IPv4-Gigawords  
Acct-Output-IPv4-Gigawords  
Acct-Input-IPv6-Octets  
Acct-Output-IPv6-Octets  
Acct-Input-IPv6-Packets  
Acct-Output-IPv6-Packets  
Acct-Input-IPv6-Gigawords  
Acct-Output-IPv6-Gigawords

against the dual-stack case for traffic statistics reporting in RADIUS.

[draft-ietf-radext-radius-extensions-02], which is already in the phase of WGLC, has extended the type space of RADIUS attribute and defined the new formats for the extended type attributes with some new data types. That might means the type code in the new extended space will be used to define a new attribute, if it is agreed to move the 'Unassigned' code space (from 144 to 191) to be 'Deprecated'.

This document tries to use both the traditional format defined in [RFC2865] and the new format defined in [draft-ietf-radext-radius-extensions-02] for the extension of IPv4 and IPv6 traffic statistics, and let the WG decides which one is more suitable for the cases mentioned here.

## 2. Terminology and Language

This document describes some new RADIUS attributes and the associated usage on NAS and AAA server. This document should be read in conjunction with the relevant RADIUS specifications, including [RFC2865], [RFC2866], [RFC2869], and [draft-ietf-radext-radius-extensions-02], for a complete mechanism. Definitions for terms and acronyms not specifically defined in this document are defined in RFC2865, RFC2866, RFC2869, and [draft-ietf-radext-radius-extensions-02].

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in BCP 14, [RFC2119].

## 3. Deployment Scenarios

Figure 1 show the typical use case of the traffic statistics reporting for the dual-stack users.

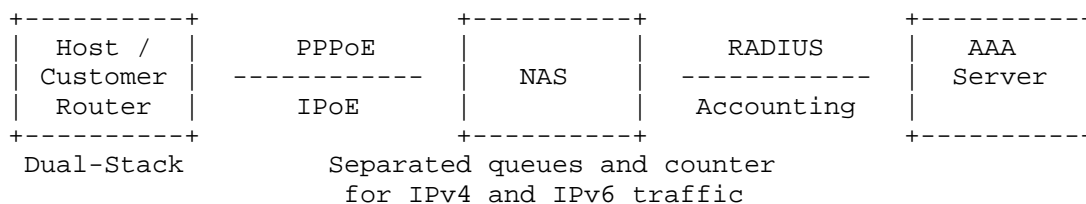


Figure 1: Traffic Statistics of Dual-Stack Users for RADIUS Accounting

Note that traffic statistics reporting is also needed in the IPv6 transition cases, such as DS-Lite, 6rd or the potential 4rd.

## 4. Traffic Statistics Attributes

### 4.1. Define the Attributes in the Traditional Unsigned Type Space

There are 8 new attributes of the traffic statistics, including:



Acct-Input-IPv4-Octets  
Acct-Output-IPv4-Octets  
Acct-Input-IPv4-Packets  
Acct-Output-IPv4-Packets  
Acct-Input-IPv6-Octets  
Acct-Output-IPv6-Octets  
Acct-Input-IPv6-Packets  
Acct-Output-IPv6-Packets

defined in this section per the traditional format defined in [RFC2865].

#### Description

The traffic statistics attributes, including Acct-Input-IPv4-Octets, Acct-Output-IPv4-Octets, Acct-Input-IPv4-Packets, Acct-Output-IPv4-Packets and Acct-Input-IPv6-Octets, Acct-Output-IPv6-Octets, Acct-Input-IPv6-Packets, Acct-Output-IPv6-Packets, indicate how many octets or packets of IPv4 or IPv6 received from the user or sent to the user from the starting of this service provided, and can be present in Accounting-Request records while the Acct-Status-Type is set to Interim-Update or Stop.

For the attribute of Acct-Input-IPv4-Octets, NAS report how many Octets of IPv4 traffic received from the user from the starting of the service authorized.

For the attribute of Acct-Output-IPv4-Octets, NAS report how many Octets of IPv4 traffic sent to the user from the starting of the service authorized.

For the attribute of Acct-Input-IPv4-Packets, NAS report how many packets of IPv4 traffic received from the user from the starting of the service authorized.

For the attribute of Acct-Output-IPv4-Packets, NAS report how many packets of IPv4 traffic sent to the user from the starting of the service authorized.

For the attribute of Acct-Input-IPv6-Octets, NAS report how many Octets of IPv6 traffic received from the user from the starting of the service authorized.

For the attribute of Acct-Output-IPv6-Octets, NAS report how many Octets of IPv6 traffic sent to the user from the starting of the service authorized.

For the attribute of Acct-Input-IPv6-Packets, NAS report how many packets of IPv6 traffic received from the user from the starting of the service authorized.

For the attribute of Acct-Output-IPv6-Packets, NAS report how many packets of IPv6 traffic sent to the user from the starting of the service authorized.

A summary of the Traffic Statistics attributes format is shown as 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 (cont.)																			
																				Value (cont.)																			

Type

TBA<sub>n</sub> (by IANA)

Length

=10

Value

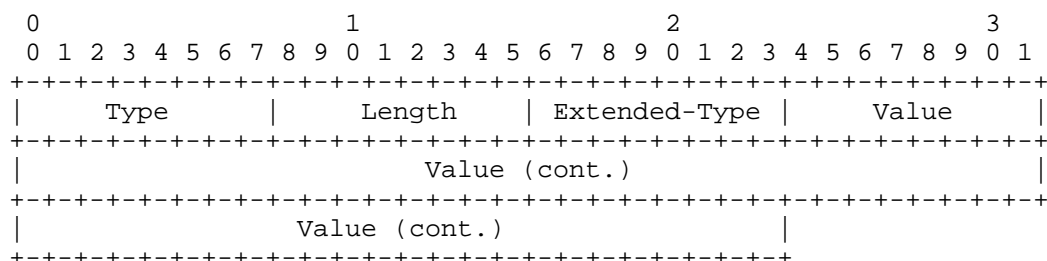
The Value field is 8 octets and uses Integer64 defined in [draft-ietf-radext-radius-extensions-02], for its data type.

4.2. Define the Attributes in the Extended Type Space

Description

The definition and usage of the traffic statistics attributes, including Acct-Input-IPv4-Octets, Acct-Output-IPv4-Octets, Acct-Input-IPv4-Packets, Acct-Output-IPv4-Packets and Acct-Input-IPv6-Octets, Acct-Output-IPv6-Octets, Acct-Input-IPv6-Packets, Acct-Output-IPv6-Packets, are the same as that described in section 4.1

A summary of the Traffic Statistics attributes format is shown as below. The fields are transmitted from left to right.



#### Type

Acct-Input-IPv4-Octets	241.TBA1 (by IANA) or 241.42 (suggested)
Acct-Output-IPv4-Octets	241.TBA2 (by IANA) or 241.43 (suggested)
Acct-Input-IPv4-Packets	241.TBA3 (by IANA) or 241.47 (suggested)
Acct-Output-IPv4-Packets	241.TBA4 (by IANA) or 241.48 (suggested)
Acct-Input-IPv6-Octets	241.TBA5 (by IANA) or 242.42 (suggested)
Acct-Output-IPv6-Octets	241.TBA6 (by IANA) or 242.43 (suggested)
Acct-Input-IPv6-Packets	241.TBA7 (by IANA) or 242.47 (suggested)
Acct-Output-IPv6-Packets	241.TBA8 (by IANA) or 242.48 (suggested)

#### Length

=11

#### Value

The Value field is 8 octets and uses Integer64 defined in [draft-ietf-radext-radius-extensions-02], for its data type.

### 5. Table of Attributes

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

Req- uest	Acc- ept	Rej- ect	Chall -enge	Accounting # Request	Attribute
0	0	0	0	0-1	TBA1 Acct-Input-IPv4-Octets
0	0	0	0	0-1	TBA2 Acct-Output-IPv4-Octets
0	0	0	0	0-1	TBA3 Acct-Input-IPv4-Packets
0	0	0	0	0-1	TBA4 Acct-Output-IPv4-Packets
0	0	0	0	0-1	TBA5 Acct-Input-IPv6-Octets
0	0	0	0	0-1	TBA6 Acct-Output-IPv6-Octets
0	0	0	0	0-1	TBA7 Acct-Input-IPv6-Packets
0	0	0	0	0-1	TBA8 Acct-Output-IPv6-Packets

The meaning of the above table entries is as follows:

- 0 This attribute MUST NOT be present.
- 0+ Zero or more instances of this attribute MAY be present.
- 0-1 Zero or one instance of this attribute MAY be present.
- 1 Exactly one instance of this attribute MUST be present.
- 1+ One or more of these attributes MUST be present.

## 6. Security Considerations

Security issues related RADIUS are described in section 8 of RFC2865 and section 5 of RFC3162.

## 7. IANA Considerations

IANA is requested to assign 8 new attribute types code in the "Radius Types" registry (<http://www.iana.org/assignments/radius-types> for the following attributes:

- Acct-Input-IPv4-Octets
- Acct-Output-IPv4-Octets
- Acct-Input-IPv4-Packets
- Acct-Output-IPv4-Packets
- Acct-Input-IPv6-Octets
- Acct-Output-IPv6-Octets
- Acct-Input-IPv6-Packets
- Acct-Output-IPv6-Packets

IANA should allocate these codes from the standardized type space of the RADIUS attributes using the "IETF Review" policy [RFC5226].

## 8. Acknowledgements

TBD

## 9. References

### 9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

- [RFC2866] Rigney, C., "RADIUS Accounting", RFC 2866, June 2000.
- [RFC2869] Rigney, C., Willats, W., and P. Calhoun, "RADIUS Extensions", RFC 2869, June 2000.
- [RFC3162] Aboba, B., Zorn, G., and D. Mitton, "RADIUS and IPv6", RFC 3162, August 2001.
- [RFC4818] Salowey, J. and R. Droms, "RADIUS Delegated-IPv6-Prefix Attribute", RFC 4818, April 2007.
- [draft-ietf-radext-radius-extensions-02]  
DeKok, A. and A. Lior, "Remote Authentication Dial In User Service (RADIUS) Protocol Extensions", Oct 2011.

## 9.2. Informative References

- [BBF TR-187]  
Broadband Forum, "IPv6 for PPP Broadband Access, Issue 1", May 2010.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [draft-hu-v6ops-radius-issues-ipv6-00]  
Hu, J., Yan, L., Wang, Q., and J. Qin, "RADIUS issues in IPv6 deployments", February 2011.
- [draft-maglione-radext-ipv6-acct-extensions-01]  
Maglione, R., Krishnan, S., Kavanagh, A., Varga, B., and J. Kaippallimalil, "RADIUS Accounting Extensions for IPv6", January 2011.
- [draft-winter-radext-fancyaccounting-00]  
Winter, S., "RADIUS Accounting for traffic classes", March 2011.
- [draft-yeh-radext-dual-stack-access-02]  
Yeh, L. and T. Tsou, "RADIUS Attributes for Dual Stack Access", March 2011.
- [ietf-radext-ipv6-access-05]  
Lourdelet, B., Dec, W., Sarikaya, B., Zorn, G., and D. Miles, "RADIUS attributes for IPv6 Access Networks", July 2011.

Author's Address

Leaf Y. Yeh (editor)  
Huawei Technologies  
F4, Huawei Area, Bantian,  
Longgang District, Shenzhen 518129  
P.R.China

Phone: +86-755-28971871  
Email: leaf.y.yeh@huawei.com

