                      Data Types in RADIUS

Abstract

   RADIUS specifications have used data types for two decades without
   defining them as managed entities. During this time, RADIUS
   implementations have named the data types and have used them in
   attribute definitions. This document updates the specifications to
   better follow established practice. We do this by naming the data
   types defined in RFC 6158, which have been used since at least the
   publication of RFC 2865. We provide an IANA registry for the data
   types and update the "RADIUS Attribute Types" registry to include a
   Data Type field for each attribute. Finally, we recommend that
   authors of RADIUS specifications use these types in preference to
   existing practice. This document updates RFCs 2865, 3162, 4072,
   6158, 6572, and 7268.

Status of This Memo

   This is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   Internet Standards is available in Section 2 of RFC 7841.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   http://www.rfc-editor.org/info/rfc8044.

Copyright Notice

Table of Contents

1. Introduction

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

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

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

1.1. Specification Problems with Data Types

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

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

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

   Other failures are that [RFC3162] does not give a data type name and
   definition for the data types IPv6 address, Interface-Id, or IPv6
   prefix. [RFC2869] defines Event-Timestamp to carry a time but does
   not reuse the "time" data type defined in [RFC2865]. Instead, it
   just repeats the "time" definition. [RFC6572] defines multiple
   attributes that carry IPv4 prefixes. However, an "IPv4 prefix" data

type is not named, defined as a data type, or called out as an
addition to RADIUS. Further, [RFC6572] does not follow the
recommendations of [RFC6158] and does not explain why it fails to
follow those recommendations.

These ambiguities and inconsistencies need to be resolved.

1.2. Implementation Problems with Data Types

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

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

1.3. No Mandated Changes

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

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

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

1.4. Requirements Language

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

2. Use of Data Types

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

2.1. Specification Use of Data Types

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

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

2.1.1. Field Names for Attribute Values

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

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

   We also define the following term:

      Attr-Data

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

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

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

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

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

2.1.2. Attribute Definitions Using Data Types

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

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

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

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

2.1.3. Format of Attribute Definitions

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

      Description

         A description of the meaning and interpretation of the
         attribute.

      Type

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

      Length

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

      Data Type

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

      Value

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

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

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

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

2.1.4. Defining a New Data Type

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

   The Name field SHOULD be a single name, all lowercase.

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

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

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

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

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

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

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

2.2. Implementation Use of Data Types

   Implementations not supporting a particular data type MUST treat
   attributes of that data type as being of data type "string", as
   defined in Section 3.5. It is RECOMMENDED that such attributes
   be treated as "invalid attributes", as defined in
   [RFC6929], Section 2.8.

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

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

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

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

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

3. Data Type Definitions

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

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

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

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

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

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

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

3.1. integer

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

   Name

      integer

   Value

      1

   Length

      Four octets

   Format

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

3.2. enum

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

   Name

      enum

   Value

      2

   Length

      Four octets

   Format

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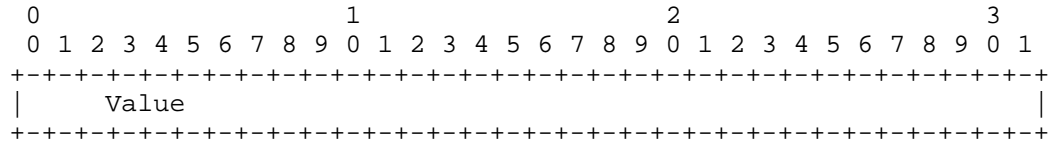
3.3. time

   The "time" data type encodes time as a 32-bit unsigned value in
   network byte order and in seconds since 00:00:00 UTC, January 1,
   1970. We note that dates before the year 2017 are likely to indicate
   configuration errors or lack of access to the correct time.

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

   Name

      time

   Value

      3

   Length

      Four octets

   Format

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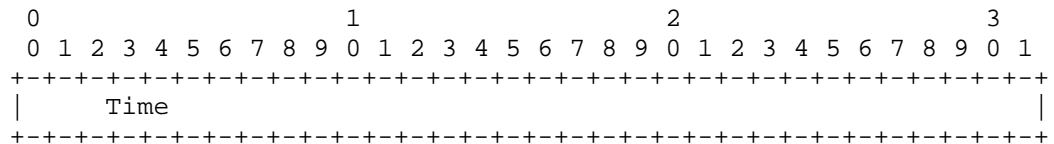
3.4. text

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

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

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

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

   Name

      text

   Value

      4

   Length

      One or more octets

   Format

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

3.5. string

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

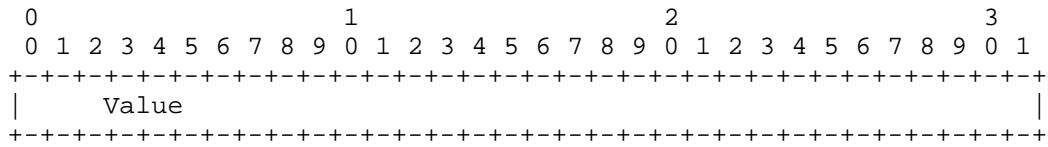   Attributes of type "string" that are allocated in the standard space
   (Section 1.2 of [RFC6929]) are limited to no more than 253 octets of
   data. Attributes of type "string" that are allocated in the extended
   space can be longer. In both cases, these limits are reduced when
   the data is encapsulated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

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

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

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

   Name

      string

   Value

      5

   Length

      One or more octets

   Format

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

3.6. concat

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

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

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

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

   Name

      concat

   Value

      6

   Length

      One or more octets

   Format

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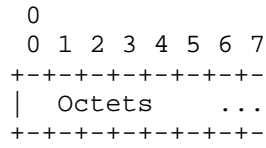
3.7. ifid

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

   Name

      ifid

   Value

      7

   Length

      Eight octets

   Format

       0   1   2   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+
      | Interface-Id ...
      +-+
          ... Interface-Id |
      +-+
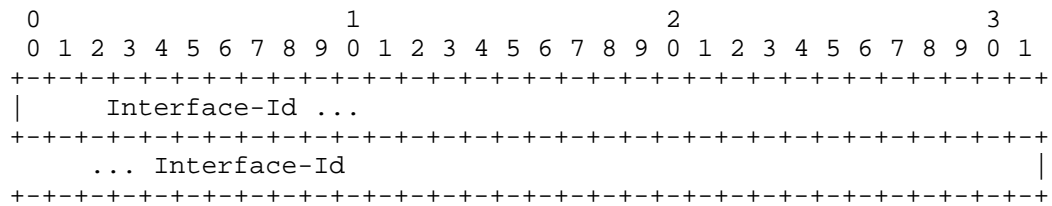
3.8. ipv4addr

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

   Name

      ipv4addr

   Value

      8

   Length

      Four octets

   Format

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

3.9. ipv6addr

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

   Name

      ipv6addr

   Value

      9

   Length

      Sixteen octets

   Format

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

3.10. ipv6prefix

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

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

   Name

      ipv6prefix

   Value

      10

   Length

      At least two, and no more than eighteen, octets

Format

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

Subfields

   Reserved

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

   Prefix-Length

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

   Prefix

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

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

3.11. ipv4prefix

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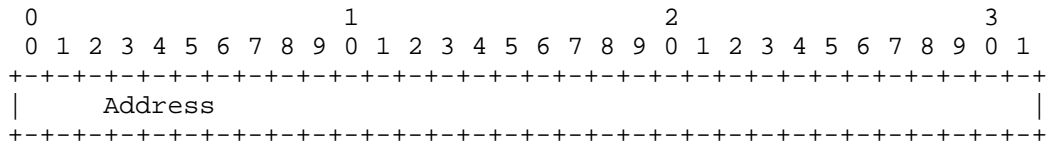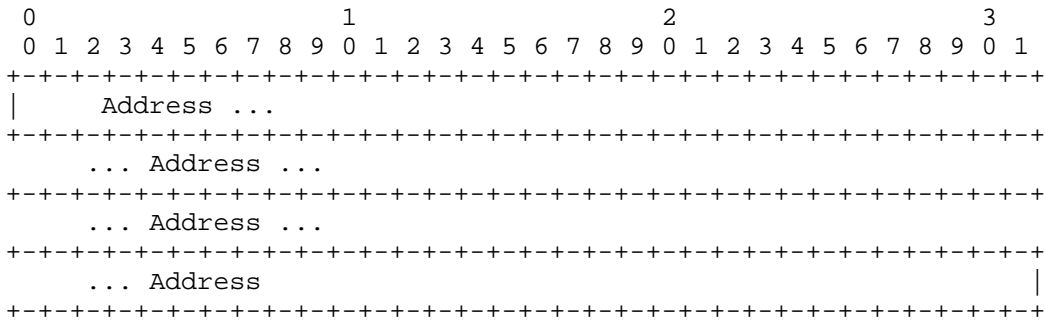   be treated as invalid attributes.

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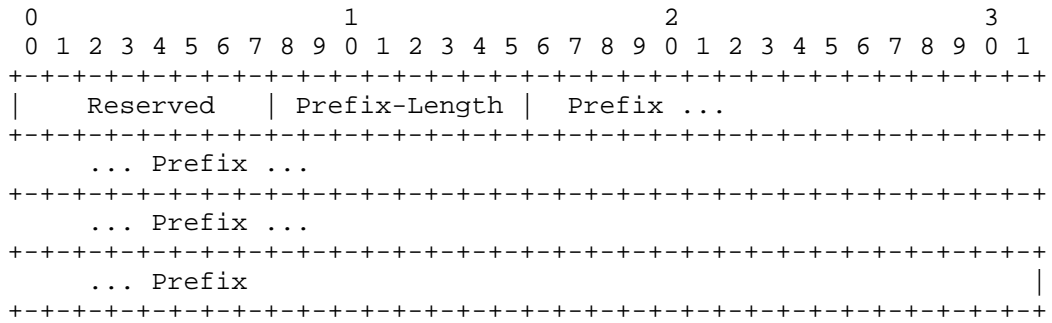   MUST be treated as invalid attributes.

   Name

      ipv4prefix

   Value

      11

   Length

      Six octets

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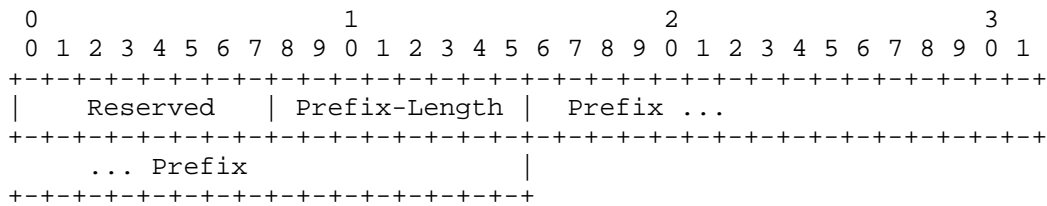
       0   1   2   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+
      | Reserved | Prefix-Length | Prefix ...
      +-+
         ... Prefix |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

   Subfields

      Reserved

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

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

      Prefix-Length

         The length of the prefix, in bits. The values MUST be no
         larger than 32. This field is one octet in length. Note that
         this definition differs from that given in [RFC6572].

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

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

   Prefix

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

3.12. integer64

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

   Name

      integer64

   Value

      12

   Length

      Eight octets

   Format

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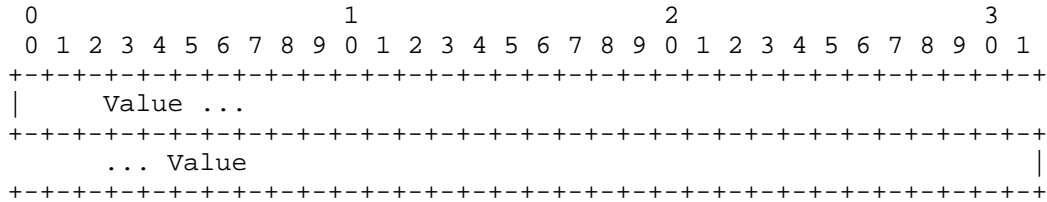   +-+

3.13. tlv

   The "tlv" data type encodes a Type-Length-Value, as defined in
   [RFC6929], Section 2.3.

   Name

      tlv

   Value

      13

   Length

      Three or more octets

   Format

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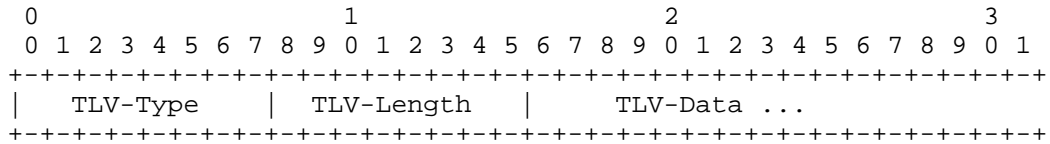
   Subfields

   TLV-Type

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

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

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

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

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

TLV-Length

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

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

TLV-Data

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

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

3.14. vsa

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

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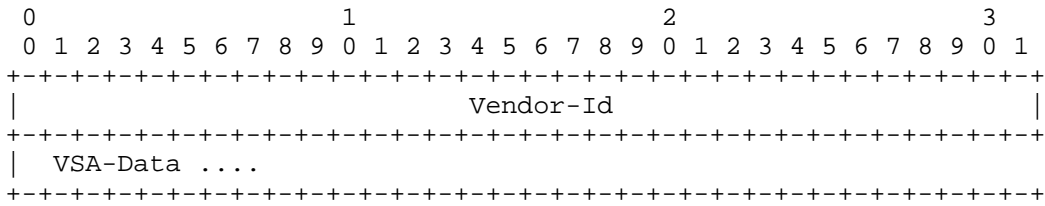
   Name

      vsa

   Value

      14

   Length

      Five or more octets

   Format

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

   Subfields

      Vendor-Id

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

      VSA-Data

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

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

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

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

3.15. extended

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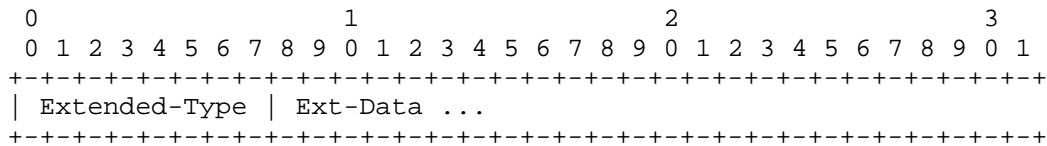
   Name

      extended

   Value

      15

   Length

      Two or more octets

   Format

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

   Subfields

   Extended-Type

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

      The Extended-Type is meaningful only within a context defined
      by the Type field. That is, this field may be thought of as
      defining a new type space of the form "Type.Extended-Type".
      See [RFC6929], Section 2.1 for additional discussion.

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

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

   Ext-Data

      The Ext-Data field is one or more octets.

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

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

3.16. long-extended

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

   Name

      long-extended

   Value

      16

   Length

      Three or more octets

   Format

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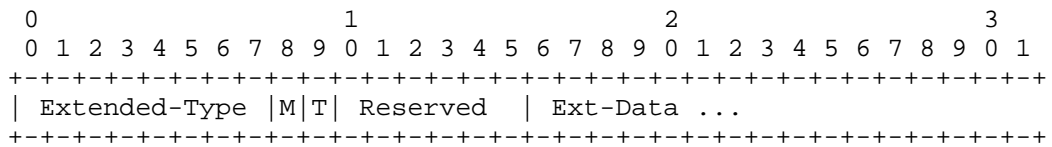
   Subfields

      Extended-Type

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

      M (More)

         The More field (M flag) 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 a value less than 255. The More field MAY be
         set (1) if the Length field has a value of 255.

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

         When the More field is set (1), the Attribute MUST have a
         Length field value of 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 is handled as per [RFC6929], Section 2.8.

      T (Truncation)

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

invalid (they will not be able to reconstruct them), but they
will still forward them, as Section 5.2 of [RFC6929] indicates
that they SHOULD forward unknown attributes anyway.

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

Reserved

This field is six 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 one or more additional
meanings for this field. Implementations therefore MUST NOT
treat this field as invalid if it is non-zero.

Ext-Data

The Ext-Data field is one or more octets.

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

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

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

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

3.17. evs

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

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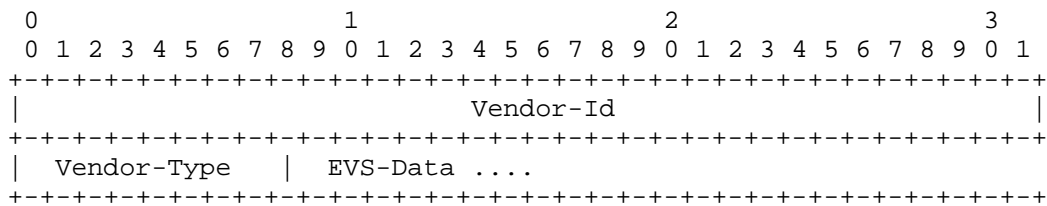
   Name

      evs

   Value

      17

   Length

      Six or more octets

   Format

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

   Subfields

      Vendor-Id

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

      Vendor-Type

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

EVS-Data

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

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

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

4. Updated Registries

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

4.1. New "Data Type" Registry

   This section defines a new registry located under "RADIUS Types",
   called "Data Type". The registration procedures for the "Data Type"
   registry are "Standards Action" [RFC5226].

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

   Value

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

   Description

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

   Reference

      The specification where the data type was defined.

The initial contents of the registry are as follows.

```
    Value   Description     Reference
    -----   -----------     -------------------
        1   integer         [RFC2865], RFC 8044
        2   enum            [RFC2865], RFC 8044
        3   time            [RFC2865], RFC 8044
        4   text            [RFC2865], RFC 8044
        5   string          [RFC2865], RFC 8044
        6   concat          RFC 8044
        7   ifid            [RFC3162], RFC 8044
        8   ipv4addr        [RFC2865], RFC 8044
        9   ipv6addr        [RFC3162], RFC 8044
       10   ipv6prefix      [RFC3162], RFC 8044
       11   ipv4prefix      [RFC6572], RFC 8044
       12   integer64       [RFC6929], RFC 8044
       13   tlv             [RFC6929], RFC 8044
       14   vsa             [RFC2865], RFC 8044
       15   extended        [RFC6929], RFC 8044
       16   long-extended   [RFC6929], RFC 8044
       17   evs             [RFC6929], RFC 8044
```

4.2. Updates to the "RADIUS Attribute Types" Registry

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

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

5. Security Considerations

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

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

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

6. IANA Considerations

IANA has created one new registry, as described in Section 4.1.

IANA has updated the "RADIUS Attribute Types" registry, as described
in Section 4.2.

IANA requires that all allocation requests in the "RADIUS Attribute
Types" registry contain a Data Type field, which is required to
contain one of the "Data Type" names contained in the RADIUS "Data
Type" registry.

IANA requires that updates to the RADIUS "Data Type" registry contain
the following fields, with the associated instructions:

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

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

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

7. References

7.1. Normative References

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

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

   [RFC3162] Aboba, B., Zorn, G., and D. Mitton, "RADIUS and IPv6",
              RFC 3162, DOI 10.17487/RFC3162, August 2001,
              <http://www.rfc-editor.org/info/rfc3162>.

   [RFC3629] Yergeau, F., "UTF-8, a transformation format of
              ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629,
              November 2003, <http://www.rfc-editor.org/info/rfc3629>.

   [RFC4072] Eronen, P., Ed., Hiller, T., and G. Zorn, "Diameter
              Extensible Authentication Protocol (EAP) Application",
              RFC 4072, DOI 10.17487/RFC4072, August 2005,
              <http://www.rfc-editor.org/info/rfc4072>.

   [RFC5234] Crocker, D., Ed., and P. Overell, "Augmented BNF for
              Syntax Specifications: ABNF", STD 68, RFC 5234,
              DOI 10.17487/RFC5234, January 2008,
              <http://www.rfc-editor.org/info/rfc5234>.

   [RFC6158] DeKok, A., Ed., and G. Weber, "RADIUS Design Guidelines",
              BCP 158, RFC 6158, DOI 10.17487/RFC6158, March 2011,
              <http://www.rfc-editor.org/info/rfc6158>.

   [RFC6572] Xia, F., Sarikaya, B., Korhonen, J., Ed., Gundavelli, S.,
              and D. Damic, "RADIUS Support for Proxy Mobile IPv6",
              RFC 6572, DOI 10.17487/RFC6572, June 2012,
              <http://www.rfc-editor.org/info/rfc6572>.

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

7.2. Informative References

   [PEN]   IANA, "PRIVATE ENTERPRISE NUMBERS",
              <http://www.iana.org/assignments/enterprise-numbers/>.

   [RFC2868] Zorn, G., Leifer, D., Rubens, A., Shriver, J., Holdrege,
              M., and I. Goyret, "RADIUS Attributes for Tunnel Protocol
              Support", RFC 2868, DOI 10.17487/RFC2868, June 2000,
              <http://www.rfc-editor.org/info/rfc2868>.

   [RFC2869] Rigney, C., Willats, W., and P. Calhoun, "RADIUS
              Extensions", RFC 2869, DOI 10.17487/RFC2869, June 2000,
              <http://www.rfc-editor.org/info/rfc2869>.

   [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an
              IANA Considerations Section in RFCs", BCP 26, RFC 5226,
              DOI 10.17487/RFC5226, May 2008,
              <http://www.rfc-editor.org/info/rfc5226>.

   [RFC6929] DeKok, A. and A. Lior, "Remote Authentication Dial In User
              Service (RADIUS) Protocol Extensions", RFC 6929,
              DOI 10.17487/RFC6929, April 2013,
              <http://www.rfc-editor.org/info/rfc6929>.

   [RFC7268] Aboba, B., Malinen, J., Congdon, P., Salowey, J., and M.
              Jones, "RADIUS Attributes for IEEE 802 Networks",
              RFC 7268, DOI 10.17487/RFC7268, July 2014,
              <http://www.rfc-editor.org/info/rfc7268>.

Acknowledgments

Author's Address

   Alan DeKok
   The FreeRADIUS Server Project

   Email: aland@freeradius.org