A proposal for the Extended RADIUS Attribute

Emile van Bergen,
E-Advies

openradius@e-advies.nl

Why an Extended Attribute?

- It is part of the ***Design Guidelines*** work that's given by the Charter:
  - “[The Design Guidelines document] will specifically consider how complex data types may be introduced in a robust manner,
  - maintaining backwards compatibility with existing RADIUS RFCs, across all the classes of attributes: Standard, Vendor-Specific and SDO-Specific.
  - In addition, it will review RADIUS data types and associated backwards compatibility issues.”

Why an Extended Attribute? (cont'd)

- Working on the Design Guidelines, it became clear that to allow complex data consisting of addressable members,
- ... we either need large values, a tree structure to hold the attributes, or both,
- ... while the RADIUS attribute format allows neither.
- The chairs then split the work into a Design Guidelines document (BCP) and an Extended Attribute document (Stds track).

What do the proposals done so far agree upon?

- To create a new RADIUS attribute of type String (in the RADIUS sense, ie. opaque octets), the "***Extended-Attribute***"
- The octet string formed by the concatenation of all instances of that attribute, holds one or more "***Extended Attributes***"
- The encapsulated Extended Attributes have a simple TLV format, but one without RADIUS' space and structure limitations.

Why this form of encapsulation? Why concatenation?

- Without some form of concatenation, RADIUS values are limited to 253 octets – officially not even enough for a DNS name
- Without encapsulating the new format in RADIUS attributes, we break compatibility
- Concatenating at such a low level (below the actual attribute format) makes parsing easy and packing efficient
- (And: it's been done before, with EAP-Message).

What aspects of the header format do we seem to agree upon?

- A wider *Type* (AVP Code) field, to allow more standard attributes to exist – vendors don't tend to adopt other vendors' VSAs
- A wider *Length* field to allow longer values
- A *Vendor* field, to allow vendor specific attributes without another level of attribute encapsulation
- It should persuade vendors to prefer it over vendor specific internal TLV formats, which are inaccessible without specific knowledge

And what do we agree upon with regard to Value data types?

- Other than the Value's length in octets, no value metadata is provided in the header,
- So, as in RADIUS, the data type depends on the individual attribute Type (AVP Code)
- However, to cater for dictionary based implementations and maximum DIAMETER compatibility,
- The DIAMETER base types should be used for Extended Attributes, wherever possible.

Where we disagree: the precise format, and, how to do structuring

- One proposal is to:
- use the full DIAMETER header, ignoring the *Flags* field except for the flag that specifies whether the *Vendor* field is present;
- use the DIAMETER 'grouped' data type to create structured data, which is a type that:
- physically contains complete DIAMETER attributes from the DIAMETER number space, but in quantities and orders (also among attributes of different type, unlike RADIUS) that the parent attribute specifies.

An alternative proposal for the attribute format and structuring method

- Use the following header fields, in this order:
- A 32-bit Std/Vendor field that specifies the number space for the Type field;
- A 32-bit Type (AVP Code) field;
- An 8-bit Tag field;
- An 8-bit Child-Tag field;
- A 16-bit Length field;
- The Value field.
- (next header begins at 4-octet boundary)

Properties of this format

- The header has a fixed size; the 32-bit Std/Vendor field that defines the number space for the Type field is always present;
- The 8-bit Std subfield makes the 24-bit Vendor subfield designate either an SMI Private Entity, an SDO or an IETF standard that requires its own Type number space.
- The tree structure for attributes is created not by physically nesting them, but by re-lational references in the Tag and Child-Tag fields.

Properties of structuring using the Tag and Child-Tag fields

- Complete semantic compatibility with existing use of optional Tags in RADIUS (RFC 2868) to form grouped sets of A/V pairs;
- Any attribute can hold a normal value, while at the same time being able to lightly refer to another set of attributes as its children;
- RADIUS' A/V pair parsing, referencing and database retrieval styles are all preserved;
- The arbitrarily nested structure is created using two fields and is independent from the storage format (packet, memory, database).

Example: Breaking down Connect-Info

Notation for tagged A/V pairs (non-Merit!):

- Attribute[:Tag] = [Child-Tag:][Value]

Example set of Extended Attributes:

- Connect-Info = 1:”52000/31200 V90 LAPM”
- BPS-Down:1 = 52000
- BPS-Up:1 = 31200
- Modulation-Type:1 = V90
- Error-Correction:1 = LAPM

Notes

- The Text value of Connect-Info is not required, but shows that attributes can both have values and children at the same time
- A policy that is depends on whether a connection has LAPM error correction enabled, does not need to pay much attention to the nested structure;
- ... it can just test Error-Correction == LAPM (wildcard tag implied).

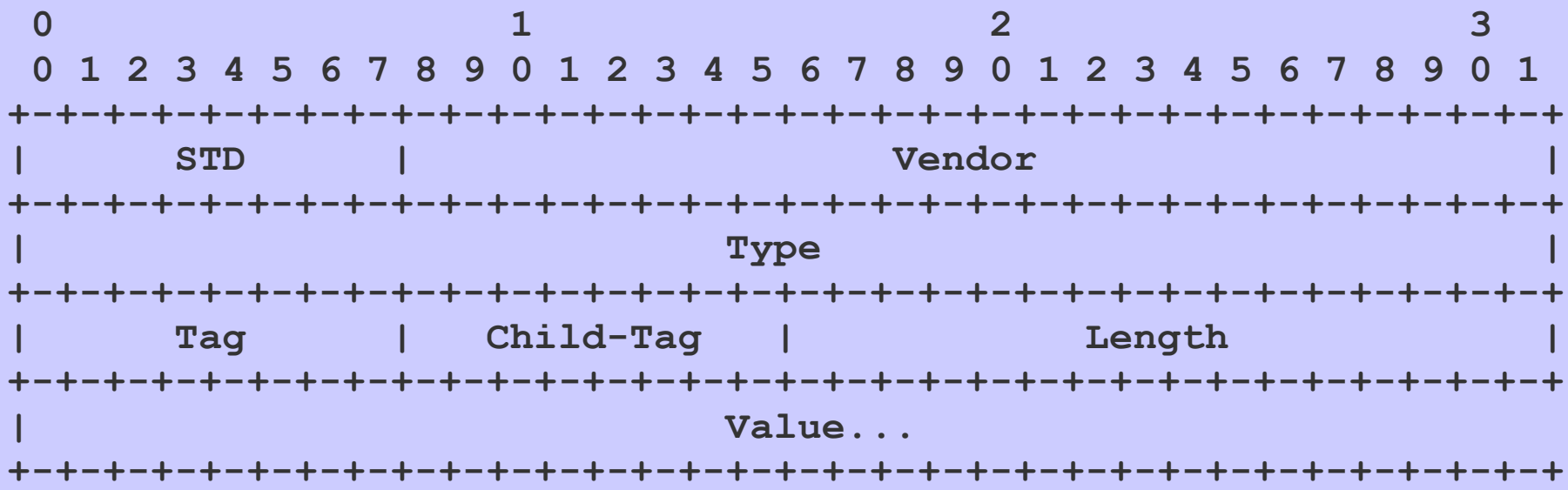
Another Example: WISPr

- WISPr-Location-Name = “KPN,amsh-1234”
- WISPr-Location-Id = “isocc=NL, cc=31,ac=20,network=AmstelHotel”

Expressed using 2 levels of subattributes:

- WISPr-Location = :1 (no value)
  - WISPr-Loc-Operator:1 = :2 (no value)
  - WISPr-Loc-Identification:1 = :3 (no value)
    - WISPr-Operator-Name:2 = “KPN”
    - WISPr-Operator-Loc-Id:2 = “amsh-1234”
    - WISPr-Loc-Id-Iso-Cc:3 = “NL”
    - WISPr-Loc-Id-E164-Cc:3 = 31
    - WISPr-Loc-Id-E164-Ac:3 = 20
    - WISPr-Loc-Id-SSID:3 = “AmstelHotel”

Summary



For more information, see the work in progress:

http://www.e-advies.nl/ietf/
draft-evbergen-radext-extended-attribute-02.txt

Thanks!