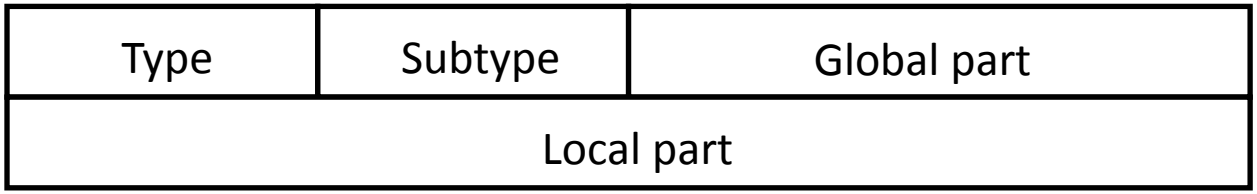
64 Bit Extended Communities

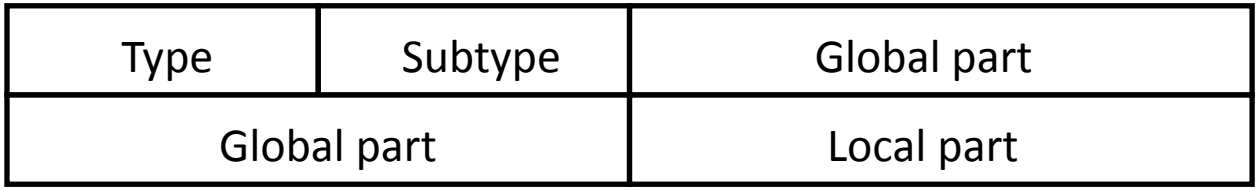
Finding a practical way forward

Problem space

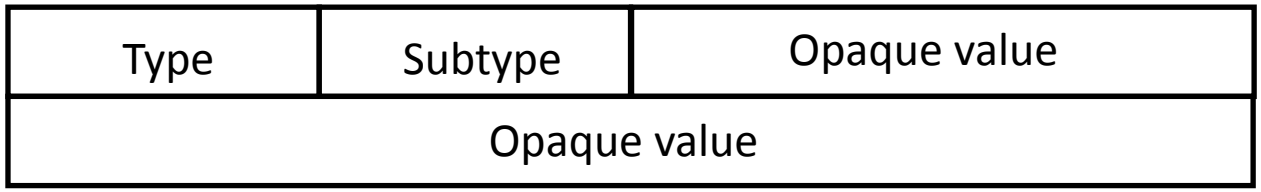
- AS4 is being deployed and starts to show up in AS paths.
- Policy needs to be expressed with AS4 elements.
- No simple mechanism to encode AS4:AS4 constructs.
- Wide communities proposal addresses this specific problem and adds much more functionality too, just at what cost?

Extended communities - RFC4360



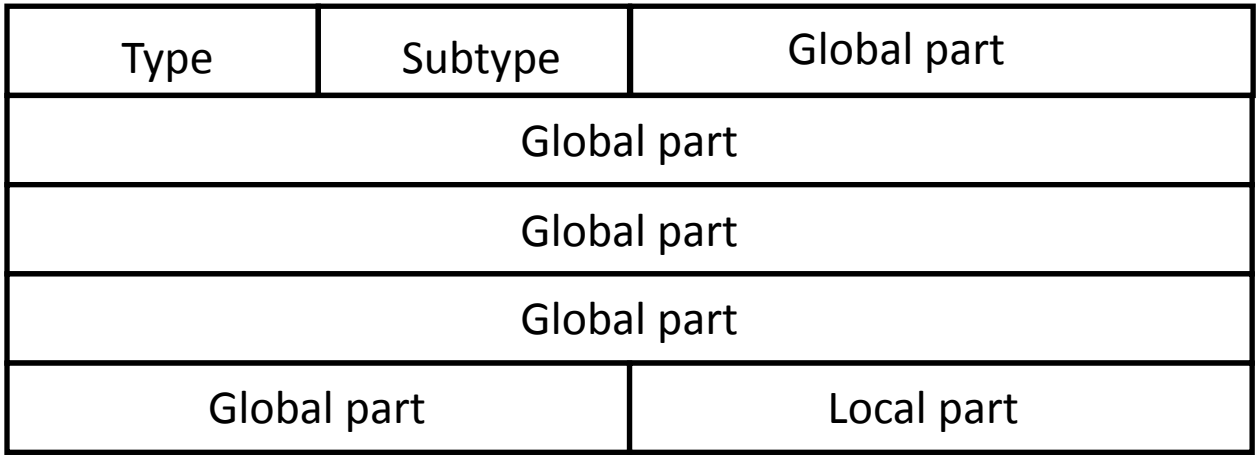


Global part is an IPv4 address.



Opaque value is a formatless 48 bit field.

IPv6 based extcomms - RFC5701



Global part represents an IPv6 address. Technically this part could be reused to carry AS4:AS4

The overall length of RFC5701 extcomm is 20 bytes – which does not align to the rules of BGP error handling specified in RFC7606

This does not seem to be supported by vendors.

4 octet extcomms - RFC5668

Type	Subtype	Global part
Global part		Local part

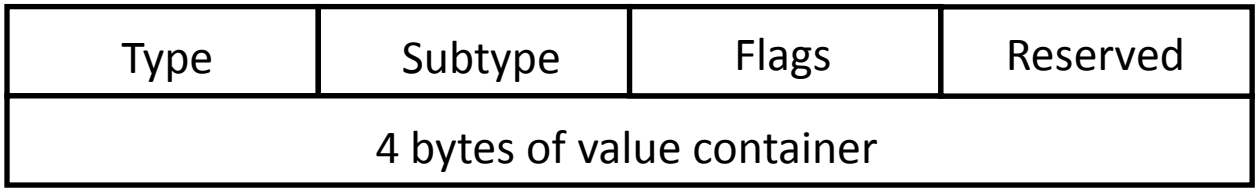
Global part here represents an AS4 number.
Local part is still 16 bits only.

RT, RO, DI, L2VPN ID and others

There are many extcomms used for signalling different aspects of L2VPN and L3VPN operation. All of them reuse RFC4360 definition of AS or IPv4 based extended communities.

These extcomms are used with address families other than IPv4 or IPv6.

EVPN communities - RFC7432



EVPN defines a new format within RFC4360 extcomm - specifically it adds the flags field. Overall extcomm length is still 8 bytes.

Wide extended communities

- It is a new approach, not an extension.
- Adds much more functionality – flexible parameters, propagation scope control, target selection.
- Adds much more complexity.
- AS number is a 32 bit value in wide extcomms.
- Vendors have no firm plans to support it, at least at the moment. This is due to complexity.
- It also adds operational complexity – existing policy constructs likely will need to be changed to accommodate the flexibility of wide extcomms.

Wide extended communities

Type	Flags	Hop Count
Length	Reg/Loc Type = 1	
Reg/Loc Type = 1	Source AS	
Source AS	Context AS	
Context AS	AType = 4	ALen = 4
AVal		

AS4:AS4 extcomm would be encoded in SourceAS:AVal fields.

Possible workarounds

- Use existing 32 bit encoding and split AS4 into two extcomms based on asdot notation.
- Define a mapping scheme to address target AS4 number via a temporary AS number.
- Reuse extcomms defined for other address families or vendor specific extcomms in a creative way.

Asdot and a policy split

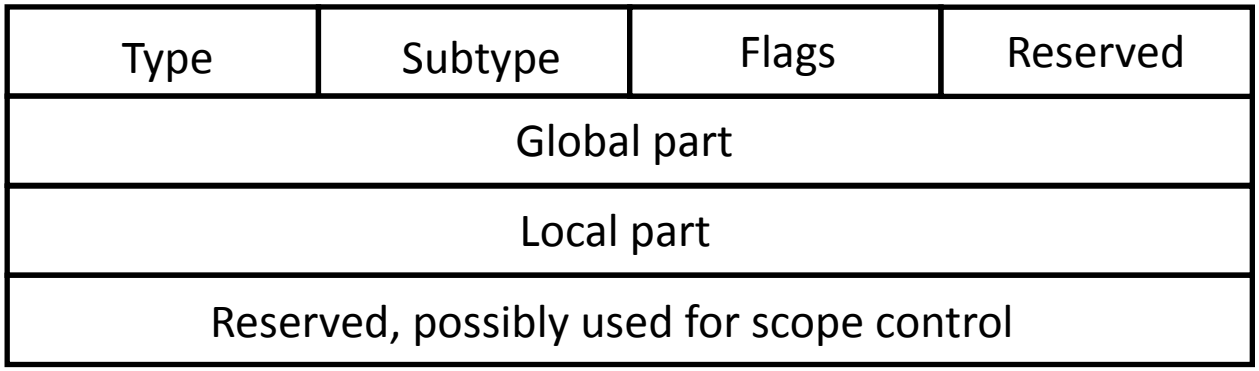
- AS4 number can be split into two 16 bit parts – asdot notation.
- Separate 16 bit parts are carried in two separate extcomms.
- This allows to reuse existing encoding – at a cost of complexity of expressing such policy.
- Possibility of conflicts – might need a third “indicator” extcomm.

Requirements for 64 bit extcomms

- Limit the scope of both the problem and proposed solution.

- Must be able to express AS4:AS4 policy constructs natively on the wire.
- Policy should not be different for AS and AS4.
- Simple and practical approach.

Possible encoding



- Global and local parts carry AS4 values.
- The third 32 bit field is reserved, could be used for RID or AS number of the target system.
- No use for flags field at the moment, possibly leave as reserved.
- 32- (or possibly 64-) bit alignment for software optimizations.
- This is not an attempt to derail wide communities approach. It is a specific solution for a specific problem.

Discussion

- Is this a problem in your environment?
- How painful is it?
- What other requirements would you see?

- Vendors will need to implement this to be practical.